Université du Québec
Institut national de la recherche scientifique
Centre Énergie Matériaux Télécommunications

Closing the gap between research and practice in the case of voice biometrics: training improvements and similarity learning for more robust verification

Combler les lacunes entre la recherche et la pratique dans le cas de la biométrie vocale: améliorations de l'entrainement et apprentissage par similarité pour une vérification plus robuste

By

João Monteiro

A thesis submitted in fulfillment of the requirements for the degree of
Doctorate of Sciences, Ph.D
in Telecommunications

Evaluation Committee

Internal evaluator and committee president:	Prof. Douglas O'Shaughnessy INRS-EMT
External evaluator 1:	Prof. Alessandro Koerich École de Technologie Supérieure
External evaluator 2:	Prof. Hemant A. Patil Dhirubhai Ambani Institute of Information and Communication Technology (DA-IICT) Gandhinagar
Research advisor:	Prof. Tiago H. Falk INRS-EMT
Research co-advisor:	Dr. Jahangir Alam Centre de Recherche Informatique de Montréal

Acknowledgements

I would like to acknowledge the support and supervision of Prof. Tiago Falk, and colleagues and collaborators from MuSAE lab: Anderson Ávila, Raymundo Cassani, Abhishek Tiwari, Shruti Kshirsagar, Marília Soares, Olivier Rosanne, Belmir Jesus, Marc-Antoine Moinnereau, Liviu Ivanescu, Stefany Bedoya, João Felipe Santos, and Diana Tobón-Vallejo.

The bulk of the work discussed in this document was developed in collaboration with the *Centre de Recherche Informatique de Montréal* (CRIM), via financial support and technical contribution through my co-supervisor Jahangir Alam and my colleague Gautam Bhattacharya.

While working towards the Ph.D., I was lucky to be able to take on several research internship roles within industry. I would like to thank colleagues from research labs within those companies for their collaboration and guidance: (Huawei) Bharat Venkitesh, Alex Bie, Md Akmal Haidar, and Mehdi Rezagholizadeh. (Google) Xavier Gibert, Jianqiao Feng, Vincent Dumoulin, Dar-Shyang Lee, and Chen Xia. (Borealis AI) Mohamed Osama Ahmed, and Hossein Hajimirsadeghi. I also thank Prof. Ioannis Mitliagkas, with whom I worked in a few projects, for being so supportive and uplifting.

I would like to thank my family for their unconditional support, especially my parents João and Ana, without which this work wouldn't be possible. Lastly, I would like to largely thank my wife (and research partner) Isabela Albuquerque for her unbounded support and care.

Abstract

The field of representation learning, powered by deep neural networks, has shown to be highly effective across several problems and domains, significantly outperforming classical approaches relying on feature engineering. Specifically in the context of voice-biometrics and related tasks, i.e., the main focus of this work, representations learned by neural networks have resulted in highly discriminative utterance-level features. These have shown to perform well on both end-to-end settings as well as to define an embedding on top of which simple classifiers can be trained. Learned features correspond to outputs of a model's inner layers obtained after training on a related auxiliary task. Choosing effective training strategies and classes of models, however, is challenging and resource consuming, as it relies on a trial-and-error approach and is conditional on the particular task of interest. Furthermore, in addition to the trial-and-error limitation of finding the right architecture and training procedure for a given task/data of interest, robustness of the developed models to attacks is another issue, particularly in the context of voice biometrics. For example, relatively simple strategies, such as *replaying* a pass-phrase, the use of text-to-speech *synthesis*, and *voice conversion* systems have shown to be able to fool speaker recognizers. This limitation can allow ill-intended attackers to gain undue access to systems containing private data.

In this thesis, we propose several innovations to address these issues. First, we propose a more efficient multi-task training strategy that combines maximum likelihood estimation with metric learning, and show that the resulting models outperform those trained using either one of the two approaches. Experiments with cross-language speaker verification and spoken language identification are performed to validate the proposed method. Next, we propose an architectural change to the time delay neural network (TDNN) aiming to render it more generally applicable. More specifically, we propose pooling across different levels of the convolutional stack and a new approach to efficiently combine these multiple representations. The updated architecture is shown to not only be more versatile (i.e., can be re-used across different tasks) but the learned representations are also more discriminative. Third, to alleviate the threat of multi-style spoofing to voice biometrics systems, we propose a detection framework based on a model ensemble, in which two models are trained jointly, while a third model learns how to mix their outputs yielding a single decision score. Experimental results with replay and text-to-speech/voice conversion attacks show the proposed ensemble method achieving similar or superior performance when compared to systems specialized on each spoofing strategy separately. Finally, we turn our attention to the definition of more versatile end-to-end training approaches. As such, we propose a set of model components and a training algorithm which can be re-used across a number of different tasks. To do so, we leverage recent metric learning approaches that parameterize semantic similarity measures employing neural networks. We build upon such setting by introducing an extra component, corresponding to a *set of prototypes* representing classes observed at training time. Inference schemes are discussed for cases that require *instance-to-instance* comparisons, such as verification and retrieval, as well as those relying on *instance-to-sample* similarity assessment, such as in the case of prototypical classification.

Keywords: Discriminative verification, learning templates, metric learning, prototypical classification, speaker verification, spoken language identification, spoofing detection.

Résumé

Le domaine de l'apprentissage des représentations s'est avéré très efficace dans plusieurs problèmes et domaines, surpassant considérablement les approches classiques. Plus précisément dans le contexte de la biométrie vocale et des tâches connexes, c'est-à-dire l'objectif principal de ce travail, les représentations apprises par les réseaux de neurones ont abouti à des projections de la parole de faible dimension hautement discriminantes. Ceux-ci ont montré de bonnes performances dans les cas *end-to-end* ainsi que pour définir un encodeur sur lequel des classificateurs simples peuvent être entraînés. Les représentations apprises correspondent aux sorties des couches internes d'un modèle obtenues après apprentissage sur une tâche auxiliaire connexe. Cependant, le choix de stratégies de formation et de classes de modèles efficaces est difficile et demande beaucoup de ressources, car il repose sur une approche par essais et erreurs et est conditionné par la tâche particulière d'intérêt. En plus de la limitation par essais et erreurs de trouver la bonne architecture et la bonne procédure d'entraînement pour une tâche/données d'intérêt données, la robustesse des modèles développés aux attaques est un autre problème, en particulier dans le contexte de la biométrie vocale. Par exemple, des stratégies relativement simples telles que *relecture* d'une phrase de passe, l'utilisation de systèmes de *synthèse* et de *conversion vocale* se sont avérées capables de tromper les locuteurs. Cette limitation peut permettre à des attaquants d'accéder indûment à des systèmes contenant des données privées.

Dans cette thèse, nous proposons plusieurs innovations pour répondre à ces problématiques. Premièrement, nous proposons une stratégie d'entraînement multitâche plus efficace qui combine l'estimation du maximum de vraisemblance avec l'apprentissage métrique, et montrons que les modèles résultants surpassent ceux entraînés en utilisant l'une ou l'autre des deux approches. Des expériences de vérification du locuteur inter-langue et d'identification de la langue parlée sont effectuées pour valider la méthode proposée. Ensuite, nous proposons une modification architecturale du *time delay neural network* (TDNN) visant à le rendre plus généralement applicable. Plus précisément, nous proposons une mutualisation à différents niveaux de la pile convolutive et une nouvelle approche pour combiner efficacement ces multiples représentations. L'architecture mise à jour s'avère non seulement plus polyvalente (c'est-à-dire qu'elle peut être réutilisée dans différentes tâches), mais les représentations apprises sont également plus discriminantes. Troisièmement, pour atténuer la menace d'usurpation multi-style des systèmes de biométrie vocale, nous proposons un cadre de détection basé sur un ensemble de modèles, dans lequel deux modèles sont entraînés conjointement, tandis qu'un troisième modèle apprend à mélanger leurs sorties pour obtenir un seul score de décision. . Les résultats expérimentaux avec les attaques de relecture et de conversion texte-parole/voix montrent que la méthode d'ensemble proposée atteint des performances similaires ou supérieures par rapport aux systèmes spécialisés dans chaque stratégie d'usurpation d'identité séparément. Enfin, nous tournons notre attention vers la définition d'approches de formation de *end-to-end* plus polyvalentes. En tant que tel, nous proposons un ensemble de composants de modèle et un algorithme d'apprentissage qui peuvent être réutilisés dans un certain nombre de tâches différentes. Pour ce faire, nous utilisons des approches récentes d'apprentissage métrique qui paramétrent des mesures de similarité sémantique utilisant des réseaux de neurones. Nous nous appuyons sur un tel cadre en introduisant un composant supplémentaire, correspondant à un ensemble de prototypes représentant les classes observées au moment de l'apprentissage. Les schémas d'inférence sont discutés pour les cas qui nécessitent des comparaisons *instance-to-instance*, telles que la vérification et la récupération, ainsi que ceux qui reposent sur l'évaluation de la similarité *instance-to-sample*.

Mots-clés: Vérification discriminatif, modèles d'apprentissage, apprentissage métrique, classification prototypique, vérification du locuteur, identification de la langue parlée, détection d'usurpation.

Contents

List of Figures

List of Tables

List of Abbreviations

ASV	Automatic Speaker Verification
AUC	Area Under the operation Curve
C_{avg}	Average Cost Performance
CNN	Convolutional Neural Network
CQCC	Constant-Q Cepstral Coefficients
DCF	Detection Cost Function
EER	Equal Error Rate
GMM	Gaussian Mixture Model
LA	Logical Access
LCNN	Light Convolutional Neural Networks
LDA	Linear Discriminant Analysis
LFCC	Linear Frequency Cepstral Coefficients
LID	Language Identification
LSTM	Long Short-Term Memory
MFCC	Mel Frequency Cepstral Coefficients
min-tDCF	Minimum Tandem Detection Cost Function
ML-TDNN	Multi-level Time Delay Neural Network
MLE	Maximum Likelihood Estimation
MMD	Maximum Mean Discrepancy
NIST	National Institute of Standards and Technology
OLR	Oriental Language Recognition
PA	Physical Access
PCA	Principal Component Analysis
PLDA	Probabilistic Linear Discriminant Analysis
ProdSpec	Product Spectra
ResNet	Residual Neural Network
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SRE	Speaker Recognition Evaluation
SVM	Support Vector Machine
TDNN	Time Delay Neural Network
TEMPLE	Template Learners
UBM	Universal Background Model
VGG	Visual Geometry Group

Synopsis

0.1 Introduction

L'apprentissage de représentations utiles à partir de données structurées de grande dimension est l'un des principaux objectifs de l'apprentissage automatique moderne. Les réseaux de neurones profonds se sont révélés capables d'apprendre efficacement de telles représentations sans nécessiter un pré-traitement spécialisé des données analysées, ce qui permet d'améliorer considérablement les performances d'une gamme de tâches (par exemple, [10]). Malgré cette augmentation des performances, les réseaux de neurones profonds présentent également plusieurs lacunes qui peuvent limiter leur utilisation plus répandue dans les domaines appliqués. Ces limitations peuvent inclure: le manque de robustesse contre les changements de distribution et les attaquants générés par l'adversaire, l'inefficacité des données et des procédures d'apprentissage trop spécialisées.

Les travaux de recherche discutés ici constituent une tentative d'aborder ces questions dans une certaine mesure. Nous nous concentrons particulièrement dans le cadre de la biométrie vocale et des tâches connexes. En particulier, nous introduisons des modèles et des algorithmes visant à améliorer à la fois les performances de prédiction, la robustesse et la polyvalence.

0.2 Contexte et travaux connexes

0.2.1 Le problème de vérification

Considérons les instances de données $x \in \mathcal{X}$ telles que chaque x peut être associé à une étiquette de classe $c \in \mathcal{C}$ via une fonction d'étiquetage $f : \mathcal{X} \mapsto \mathcal{C}$. Nous définissons un essai comme une paire d'ensembles d'exemples $\{X_i, X_j\}$, à condition que $f(x_i^k) = f(x_i^l)\ \forall\ k, l \in \{1, 2, ..., |X_i|\}^2$ et $f(x_j^k) = f(x_j^l)\ \forall\ k, l \in \{1, 2, ..., |X_j|\}^2$, afin que nous puissions attribuer des étiquettes de classe à de tels ensembles X_m définissant $f(X_m) = f(x_m)\ \forall\ x_m \in X_m$. Le problème de vérification peut être vu comme, étant donné un essai $\mathcal{T}_{i,j} = \{X_i, X_j\}$, décidant si $f(X_i) = f(X_j)$. Dans ce cas, nous nous référons à $\mathcal{T}$ comme essai cible, ou $f(X_i) \neq f(X_j)$ et le essai sera appelé non-cible. Nous

classons les essais en deux types en fonction des exemples pratiques du problème de vérification. Les essais de type I sont ceux où X_i est appelé échantillon d'inscription, c'est-à-dire un ensemble de points de données représentant une classe donnée, comme une galerie de photos de visage d'un utilisateur donné dans une application de contrôle d'accès, tandis que X_j correspondra à un seul exemple x_{test} à vérifier par rapport à la galerie d'inscription. Pour le cas de type II, à son tour, X_i est simplement une réclamation correspondant à la classe contre laquelle x_{test} sera vérifié. Les classes correspondant aux exemples dans les essais de test peuvent n'avoir jamais été présentées au modèle, et les ensembles X_i et X_j sont généralement petits (< 10).

Dans l'approche Neyman-Pearson [11], la vérification est vue comme un test d'hypothèse, où H_0 et H_1 correspondent à l'hypothèse telle que $\mathcal{T}$ est cible ou non, respectivement [12]. Le test est effectué via le rapport de vraisemblance (LR) suivant:

$$LR = \frac{p(\mathcal{T}\,|H_0)}{p(\mathcal{T}\,|H_1)}, \tag{1}$$

où $p(\mathcal{T}|H_0)$ et $p(\mathcal{T}|H_1)$ correspondent aux modèles de cible, et non-cible essai. La décision est prise en comparant LR avec un seuil δ.

On peut explicitement approximer LR par des approches génératives [13]; les modèles de mélange gaussien étant les plus courants. Dans ce cas, le dénominateur est généralement défini comme un modèle de fond universel (GMM-UBM, [14]), ce qui signifie qu'il est formé sur les données de toutes les classes disponibles, tandis que le numérateur est un modèle affiné sur les données d'inscription. De sorte que, pour essai $\{X_1, X_2\}$, LR sera:

$$LR = \frac{p_{X_1}(X_2)}{p_{UBM}(X_2)} = \frac{p_{X_{Enroll}}(x_{test})}{p_{UBM}(x_{test})}. \tag{2}$$

Alternativement, [15] a montré que les approches discriminative, c'est-à-dire des classificateurs binaires entraînés au-dessus des paires de données pour déterminer s'ils appartiennent à la même classe, ont donné des rapports de vraisemblance utiles pour la vérification. Dans leur cas, une SVM binaire a été formée sur des paires d'i-vecteurs [1] pour la vérification automatique du locuteur. Plus tard, nous nous appuierons sur un tel cadre discriminant, mais avec la différence que nous apprenons un processus d'encodage avec le discriminateur (ici représenté comme un modèle de distance), et montrons qu'il donne les rapports de vraisemblance requis pour la vérification grâce à des résultats d'estimation contrastifs. Ceci est plus général que le résultat de [15], qui montre qu'il existe un classificateur génératif associé à chaque discriminateur dont le rapport de vraisemblance correspond à la sortie du discriminateur, ce qui nécessite que les hypothèses de ce classificateur soient vérifiées.

0.2.1.1 Le problème de vérification pour le cas de la biométrie vocale

La biométrie fait référence aux traits physiologiques ou comportementaux d'une personne qui peuvent être utilisés pour l'identification automatique ou la vérification d'une identité revendiquée. Les systèmes biométriques peuvent utiliser les empreintes digitales, la géométrie de la main, l'iris, la rétine, le visage, la veine de la main, les thermogrammes faciaux, la signature et la voix, ou des combinaisons de ceux-ci pour vérifier l'identité d'un utilisateur [16]. Ces caractéristiques ont été largement utilisées, par exemple, comme contrôle d'accès à des bâtiments ou à des informations sensibles et comme mécanisme d'autorisation dans les transactions financières.

Un système biométrique pratique doit répondre aux exigences de précision de reconnaissance, de vitesse et de ressources spécifiées, être inoffensif pour les utilisateurs, être accepté par la population visée et être suffisamment robuste contre diverses méthodes frauduleuses et attaques contre le système [17]. Comme indiqué dans [18], la vérification du locuteur est une approche d'identification biométrique populaire qui utilise le signal vocal comme signature de l'utilisateur. La principale raison de sa popularité est double: (a) aucun contact direct avec l'individu n'est requis; et (b) des microphones sont disponibles sur la plupart des appareils portables. De plus, des travaux antérieurs ont prouvé que les propriétés de la parole diffèrent de manière pertinente d'un individu à l'autre et, même pour les jumeaux, la similitude est peu probable.

Plus formellement, la vérification du locuteur consiste à accepter ou rejeter une identité revendiquée en comparant deux énoncés, le premier de ces énoncés étant utilisé pour l'enrôlement (produit par le locuteur avec l'identité cible) et le second énoncé est obtenu du locuteur vérifié [19]. Dans le cadre du cas indépendant du texte, la vérification du locuteur est effectuée sur des phrases sans contrainte de contenu phonétique et de longueur arbitraires. La variabilité phonétique ajoutée dans ce scénario représente un facteur défavorable supplémentaire par rapport aux variabilités de la session et du locuteur, présentes dans le cas dépendant du texte [20].

0.2.1.2 Identification de la langue parlée

La tâche d'identification de la langue (LID) consiste à identifier les langues parlées à partir de données vocales de contenu phonétique sans contrainte, ce qui est très utile dans plusieurs applications du traitement de la parole. Par exemple, des outils de reconnaissance de langue peuvent être utilisés pour approximer les vraisemblances conditionnelles. Alternativement, ils peuvent également être utilisés pour la modélisation hiérarchique, où un outil de reconnaissance de langue est placé dans les premières étapes d'un pipeline qui est ensuite suivi par des sous-modules spécifiques à la langue, comme illustré dans la figure 1.2. La reconnaissance vocale ou la vérification du locuteur sont des exemples dans lesquels des informations préalables sur la langue parlée peuvent améliorer les performances. De plus, les applications pratiques directes des dispositifs de reconnaissance de langue peuvent inclure, par exemple, la direction d'appels dans les centres d'appels.

Comme indiqué dans [21], les approches classiques pour l'identification de la langue reposent généralement sur des méthodes introduites à l'origine pour la reconnaissance du locuteur. Les applications des i-vecteurs dans la reconnaissance des langues, par exemple, peuvent être trouvées dans [21] et [22]. Même si les i-vecteurs sont connus pour produire des performances satisfaisantes sur la modélisation du locuteur et de la langue sur plusieurs ensembles de données, en particulier dans des contextes dans lesquels la quantité de données d'entraînement est limitée de manière pertinente, il est également connu que ses performances se dégradent lorsqu'elles sont confrontées à des conditions de test de courte durée. De plus, sa nature non supervisée peut devenir un problème dans certains scénarios, lorsque des données étiquetées disponibles peuvent produire des représentations de faible dimension plus discriminantes. Des approches supervisées ont été introduites en conjonction avec des réseaux de neurones de diverses familles pour l'identification du langage. Par exemple, les approches de [23] et [24] utilisent des architectures neuronales résiduelles et différentes stratégies d'entraînement, telles que la minimisation de la perte de centre [25] et le softmax angulaire [26], tous deux inspirés des applications sur la reconnaissance faciale. Cependant, l'évaluation n'a été effectuée que sur des enregistrements de plus longue durée, avec une dégradation des performances significative observée lors du passage d'enregistrements de 30 secondes à ceux de 3 secondes. De même, d'autres architectures ont été utilisées pour la modélisation de la dépendance temporelle pour l'identification des langues, notamment les réseaux de neurones à retard temporel [27] et les réseaux à mémoire à long terme (LSTM) [28]. Ceux-ci se sont avérés plus performants que les i-vecteurs dans les scénarios de courte durée, tout en n'étant pas aussi performants dans le cas des signaux de longue durée.

0.2.1.3 Détection d'usurpation d'identité

La voix est une biométrie populaire pour les approches d'identification [18, 29, 30]. Néanmoins, la biométrie vocale n'est pas exempte de menaces et des stratégies d'attaque ciblant les systèmes de reconnaissance du locuteur ont été décrites [31]. En tant que tel, la conception de contre-mesures pour produire des applications plus robustes est devenue une direction de recherche populaire. Comme discuté dans [2] et illustré dans la figure 1.4, les attaques peuvent être conçues à différentes étapes d'un pipeline d'authentification basé sur la biométrie. Notamment, les attaques contradictoires récemment introduites ciblant les réseaux de neurones artificiels [32] peuvent agir au niveau du modèle. Dans le cas particulier de la vérification du locuteur et des systèmes de biométrie vocale, d'autres stratégies d'attaque existent également. Celles-ci sont appelées attaques d'usurpation d'identité et représentent une personne ou un programme informatique qui tente de contourner un système d'authentification en falsifiant les données d'un utilisateur légitime.

Dans ce travail, nous nous concentrons sur les attaquants agnostiques au niveau du microphone. Les attaques au niveau du microphone peuvent être réalisées avec différentes méthodes [31, 33, 9]: (i) l'usurpation d'identité d'un autre utilisateur, (ii) la parole synthétique et (iii) l'audio préenregistré (rejoué) d'un utilisateur donné. Ici, nous considérerons (ii) et (iii), qui seront appelés respectivement

attaques d'accès *logique* (LA) et *physique* (PA). Les dernières attaques LA ont tiré parti des avancées récentes en matière de synthèse vocale et de conversion vocale basées sur la modélisation de formes d'onde auto-régressives ou sur des réseaux antagonistes génératifs [34, 35, 36, 37]. Compte tenu des graves conséquences que les attaques d'usurpation d'identité peuvent avoir sur les systèmes de vérification des locuteurs, des recherches récentes se sont concentrées sur le développement de nouveaux algorithmes de détection d'attaques et plusieurs défis ont été organisés (par exemple, [9, 33, 38, 39]).

Des méthodes récentes ont exploré des méthodes de détection d'usurpation *end-to-end* où un seul modèle est formé en une seule étape, et le modèle est capable de générer directement des scores à partir d'exemples de test inédits. Dans [40], un modèle convolutif est entraîné au-dessus de l'audio brut pour la détection des attaquants rejoués. Dans ce cadre, le modèle est capable d'apprendre des représentations qui détecteront les attaques par rejeu. Dans [41], un schéma attentif est introduit de telle sorte qu'une structure U-net est d'abord utilisée pour mapper les caractéristiques d'entrée dans un ensemble de poids d'importance par élément. L'entrée pondérée est ensuite introduite dans une pile de blocs d'anticipation avec des connexions résiduelles produisant des scores finaux. Les réseaux de neurones convolutifs se sont également avérés efficaces pour la détection directe *end-to-end* dans [42].

0.2.2 Architectures de modèles

0.2.2.1 Time delay neural networks (TDNN)

Les approches présentées dans les chapitres suivants utilisent ou améliorent l'architecture TDNN, introduite dans le contexte de l'ASV par le framework x-vecteur [3]. Le modèle est illustré dans la figure 1.7 et plus détaillé dans le tableau 1.1. Plus précisément, une séquence d'entrée de longueur T est notée $x_{[1:T]}$, où chaque $x_i \in \mathbb{R}^d, i \in [T]$, représente une caractéristique vecteur de dimension d à un intervalle de temps donné. Par exemple, chaque x_i pourrait correspondre à des caractéristiques spectrales sur une courte fenêtre temporelle à partir du signal d'entrée. De manière équivalente, nous notons l'ensemble des caractéristiques produites par la pile de couches de convolution par $y_{1:T} \in \mathbb{R}^D, i \in [T]$, où D correspond au nombre de canaux de sortie de la dernière couche convolutive. Nous les appelons des descripteurs locaux de l'audio global étant donné qu'ils correspondent à des caractéristiques d'une fenêtre temporelle relativement courte. D est fixé à 1500 dans le modèle d'origine et à 512 dans notre cas car, comme nous le verrons, notre paramètre nécessite des descripteurs locaux de la dimensionnalité correspondante à travers le modèle. La couche de mise en commun temporelle, également appelée *mise en commun statistique*, concatène les estimations par élément des statistiques de premier et de second ordre de l'ensemble des descripteurs locaux sur l'axe temporel (le symbole "^" dans la figure 1.7 indique l'opération de concaténation). Nous définissons

ainsi le descripteur global V, c'est-à-dire le vecteur de caractéristiques résumant l'ensemble de la séquence d'entrée $x_{[1:T]}$, par ce qui suit:

$$V = \text{cat}[\mu(y_i), \sigma(y_i)], \tag{3}$$

où l'opérateur $v = \text{cat}[v_1, v_2]$ concatène $v_1, v_2 \in \mathbb{R}^D$ tel que $v \in \mathbb{R}^{2D}$, et y_i sont obtenus après la dernière couche de convolution. Le descripteur global V est finalement introduit dans une séquence de couches denses pour produire les sorties correspondant aux log-probabilités sur l'ensemble des classes considérées (par exemple, des locuteurs ou des langues de formation).

0.2.2.2 Architectures résiduelles

Introduits pour la première fois dans [4], les ResNets constituent un ensemble d'architectures composées d'une série de blocs dits résiduels, qui déterminent en quoi une transformation de caractéristiques doit différer du mappage d'identité, plutôt que de savoir comment il devrait différer de zéro [43]. Les transformations de blocs résiduels présentent une forme de base qui, pour une variable d'entrée générique $X \in \mathcal{X}$ et une certaine fonction des données $F : \mathcal{X} \mapsto \mathcal{X}$, sera:

$$X' = F(X) + X. \tag{4}$$

Le terme résiduel vient du fait que l'entrée est directement utilisée pour calculer la sortie de la transformation, qui dans un réseau de neurones représente un chemin direct pour les gradients vers le flux lors de la rétropropagation des gradients pour le calcul des mises à jour SGD, comme illustré sur la figure 1.8. $F(X)$ est généralement un ensemble de couches convolutives, suivies de fonctions d'activation non linéaires et de couches de normalisation. Des variantes de ResNets ont été proposées et utilisées dans différents contextes, tels que la ré-identification de personnes, la détection d'objets et la segmentation, comme c'est le cas de MobileNet [44]. La littérature récente a montré que les blocs résiduels contribuent à produire des pertes qui sont plus faciles à former, dans le sens où les régions chaotiques mal conditionnées deviennent moins fréquentes lorsqu'une telle caractéristique architecturale est utilisée [45]. De plus, les transformations quasi-identitaires ont été étudiées en profondeur et des garanties ont été introduites pour les cas $F(x)$ linéaires et non linéaires dans [46] et [43], respectivement. Dans certains des chapitres qui suivent, nous utiliserons les ResNets comme un réseau sur lequel des procédures d'apprentissage sont conçues de manière à produire des représentations discriminantes au niveau de l'énoncé dépendant du locuteur ou de la langue.

0.2.2.3 Mécanismes d'attention

Attention au niveau du cadre Plusieurs mécanismes d'attention ont été introduits récemment dans des architectures visant à modéliser des propriétés globales de données temporelles, comme dans le cas de la reconnaissance du locuteur ou de la langue. En termes généraux, les blocs d'attention apprennent à peser conditionnellement les pas de temps en fonction des représentations d'entrée sur une couche interne d'un modèle [47]. Considérez $y_{1:T}$ comme un ensemble de vecteurs correspondant aux sorties d'un réseau de neurones donné pour une entrée $x_{1:T}$ – par exemple, une séquence de caractéristiques acoustiques. Une transformation linéaire A est partagée sur tous les pas de temps t, et appliquée à chaque y_t résultant en un ensemble de scalaires $a_{1:T}$, selon:

$$a_t = \tanh\left(Ay_t\right). \tag{5}$$

Un ensemble de poids normalisés totalisant jusqu'à 1 est obtenu via l'opérateur softmax:

$$w_t = \frac{e^{a_t}}{\sum_{t=1}^{T} e^{a_t}}, \tag{6}$$

et la sortie de la couche attention est finalement donnée par:

$$V = \sum_{t=1}^{T} w_t y_t, \tag{7}$$

où V correspondra à une représentation globale au niveau de l'énoncé de la séquence d'entrée entière $x_{[1:t]}$, qui peut être traitée ultérieurement par des couches entièrement connectées pour produire, par exemple, des probabilités conditionnelles sur un ensemble des locuteurs. Une approche alternative au calcul de V serait d'effectuer les opérations de regroupement statistique décrites précédemment mais sur l'ensemble des représentations pondérées $y'_t = w_t y_t$.

Attention au produit scalaire Introduit dans [48] et également appelé *self-attention*, ce composant correspond à une alternative aux modèles récurrents pour la modélisation de données séquentielles et a été introduit avec l'architecture *Transformer*. En considérant le même ensemble de représentations discuté ci-dessus et représenté par $y_{[1:t]}$, il fonctionne comme indiqué par ce qui suit:

$$\text{self-attention}(Q, K, V') = \text{softmax}(\frac{QK^\intercal}{\sqrt{d_k}})V', \tag{8}$$

où Q, K et V', dénommés respectivement *requêtes*, *keys* et *values*, correspondent chacun à une transformation linéaire de $y_{[1ă:t]}$ (supposé être une matrice de dimension $t \times D$), en supposant que

chaque $y_t \in \mathbb{R}^D$, c'est-à-dire:

$$Q = y_{[1:t]}W^Q, \quad K = y_{[1:t]}W^K, \quad V' = y_{[1:t]}W^V. \tag{9}$$

Dans ce cas, chacune des matrices W^Q, W^K et W^V a la dimension $D \times d_k$ et leurs entrées sont traitées comme des paramètres apprenables. Intuitivement, les produits scalaires à l'échelle $\frac{QK^T}{\sqrt{d_k}}$ définissent des poids indiquant l'importance de chaque élément dans la séquence en fonction d'une instance de données spécifique.

0.2.3 Apprentissage des métriques

Être capable d'évaluer efficacement la similitude entre les échantillons à partir des données en cours d'analyse est un problème de longue date dans le cadre de l'apprentissage automatique. Les algorithmes tels que les *K-means*, les classificateurs des plus proches voisins et les méthodes à noyau reposent généralement sur la sélection d'une mesure de similarité ou de distance capable d'encoder les relations sémantiques présentes dans les données de grande dimension en scores réels. Dans cette optique, les approches communément appelées *Distance Metric Learning*, introduites à l'origine dans [49], essaient d'apprendre une distance dite de Mahalanobis, qui, étant donné $x, x' \in \mathbb{R}^D$, aura la forme: $\sqrt{(x-x')^\intercal M(x-x')}$, où $M \in \mathbb{R}^{D\times D}$ est semi-défini positif. Plusieurs extensions ont ensuite été introduites [50, 51, 52].

Dans [53], par exemple, une version en ligne de l'algorithme dans [49] est proposée, tandis qu'une approche basée sur les machines à vecteurs de support (SVM) a été introduite dans [54] pour l'apprentissage de M. Dans [55], une approche théorique de l'information est fournie pour résoudre M en minimisant la divergence entre les distributions gaussiennes associées aux distances apprises et euclidiennes, montrant en outre qu'une telle approche est équivalente à l'apprentissage par noyau de bas rang [56]. Des distances similaires ont également été utilisées dans d'autres contextes, tels que la notation de similarité pour l'apprentissage contrastif [57, 58]. Outre la distance de Mahalanobis, d'autres formes de distance/similarité ont été envisagées dans des travaux récents. Dans [59], par exemple, une matrice de noyau est directement apprise, définissant implicitement une fonction de similarité. Dans [60], des classes de réseaux de neurones sont proposées pour définir des pseudo-distances qui satisfont l'inégalité triangulaire sans être nécessairement symétriques.

Pour le cas particulier de l'apprentissage métrique à distance de Mahalanobis, on peut montrer que $\exists\ W : \sqrt{(x-x')^\intercal M(x-x')} = ||Wx - Wx'||_2$ [53], ce qui signifie qu'il existe une projection linéaire des données après laquelle la distance euclidienne correspondra à la distance de Mahalanobis sur l'espace d'origine. Dans [61], la projection linéaire est remplacée par un encodeur non-linéaire appris $\mathcal{E} : R^D \mapsto R^d$ de sorte que $||\mathcal{E}(x) - \mathcal{E}(x')||_2$ donne une mesure de distance (non Mahalanobis) entre les points de données brutes produisant des propriétés utiles. Les travaux de suivi ont étendu cette idée à plusieurs applications [62, 63, 64, 65]. Une variante supplémentaire de $||Wx - Wx'||_2$,

en plus de l'introduction de $\mathcal{E}$, consiste à changer la distance euclidienne $||\cdot||_2$ avec une alternative mieux adaptée à la tâche d'intérêt. C'est le cas dans [66], où la distance de Hamming est utilisée sur des données codées dans un espace binaire. Dans [67], à son tour, l'encodeur est entraîné de sorte que les distances euclidiennes dans l'espace encodé se rapprochent des divergences de Wasserstein, tandis que dans [68] une distance hyperbolique est utilisée qui est considérée comme adaptée à leur cas d'utilisation particulier.

Sur la base de la littérature couverte, on peut conclure qu'il existe deux directions différentes visant à atteindre un objectif similaire : *apprendre à représenter les données dans un espace métrique où les distances produisent des mécanismes d'inférence efficaces pour diverses tâches.* Alors qu'une direction correspond à l'apprentissage d'une distance ou d'une similitude significative à partir de données brutes, l'autre correspond, étant donné une métrique de distance fixe, à la recherche d'un processus de codage produisant un tel espace métrique souhaitable. Dans ce dernier cadre, des approches telles que les réseaux siamois [5] ont été introduites dans le but d'apprendre explicitement un modèle d'intégration paramétré par un réseau de neurones qui se traduit par un espace de dimension inférieure où les propriétés pertinentes sont présentes, telles que la séparabilité des classes. On suppose souvent que des informations concernant les échantillons de données qui doivent être rapprochés et éloignés les uns des autres dans l'espace d'inclusion sont disponibles.

0.3 Organisation de la thèse

La suite de ce document est organisée comme suit: Le chapitre 2 décrit des propositions en termes de procédures d'apprentissage visant à améliorer les représentations utilisées pour la vérification du locuteur et l'identification de la langue. Les méthodes de détection des attaquants spoofing sont décrites dans le chapitre 3. Des variantes de l'architecture TDNN utilisant des couches attentives à plusieurs niveaux sont présentées avec une évaluation dans le chapitre 4. Dans le chapitre 5, nous introduisons une approche discriminante pour la vérification générative approximative ou, dans une autre perspective, une approche où les (pseudo) espaces métriques sont appris via à la fois un processus d'encodage ainsi qu'une (pseudo) distance. Nous étendons l'approche discutée au chapitre 5 et introduisons l'idée de *template* d'apprentissage au chapitre 6 où nous discutons de la façon de définir des composants de modèle et des procédures d'apprentissage réutilisables entre les tâches et les ensembles de données en attendant un minimum d'implémentations spécifiques aux tâches; dans ce cas, nous élargissons notre champ d'application et en plus d'envisager des évaluations dans des contextes de biométrie vocale, nous testons notre proposition dans des référentiels d'images standard correspondant à des tâches de reconnaissance d'objets et de récupération d'images. Enfin, les conclusions sont résumées dans le chapitre 7 avec une discussion sur les travaux futurs s'appuyant sur nos contributions.

0.4 Chapitre 2: Améliorement de l'entraînement des réseaux neuronaux pour l'apprentissage de la représentation au niveau de la phrase

Dans cette contribution, nous concevons une stratégie d'entraînement particulièrement adaptée pour renforcer davantage la discriminabilité sur les sorties du modèle tout en évitant les problèmes courants apparaissant dans les approches d'apprentissage métrique. Plus précisément, nous évaluons la combinaison des deux cadres décrits précédemment, à savoir: (i) estimation du maximum de vraisemblance via la reconnaissance du locuteur ou de la langue, et (ii) apprentissage métrique. Notre objectif principal est de combiner les avantages offerts par chaque schéma, c'est-à-dire la facilité de l'apprentissage dans le cadre du maximum de vraisemblance ainsi que la discriminabilité fournie par la minimisation de la perte de triplet. L'évaluation de la stratégie de formation proposée est réalisée à l'aide du triplet-network [6] réalisé avec une architecture inspirée du modèle ResNet augmenté d'un composant d'attention *frame-wise*. Pour le cas du LID, des expériences sont effectuées sur les tâches introduites pour le défi AP18-OLR correspondant à des données contenant des enregistrements de paroles téléphoniques de dix langues orientales dans différents contextes, y compris une courte durée de parole et des langues confuses, montrant des améliorations pertinentes en termes des performances de classification sur des bases de référence solides dans toutes les conditions de test étudiées. De plus, une évaluation *end-to-end* est également effectuée, montrant que l'utilisation directe des sorties du modèle en tant que scores, c'est-à-dire l'élimination du PLDA post-formé, surpasse les résultats de i-vectors + PLDA. Pour l'ASV, le cas inter-langues introduit pour le NIST SRE 2016 composé de la parole téléphonique est utilisé pour l'évaluation du schéma proposé. De plus, différentes stratégies de mise en commun, utilisées pour agréger des ensembles de descripteurs locaux dans un espace de représentation à dimension fixe, sont comparées, y compris des statistiques simples de représentations de haut niveau dans la dimension temporelle et des schémas attentifs plus complexes basés sur l'apprentissage.

0.4.1 Mise en place experimentale

Nous proposons l'utilisation d'une architecture convolutive visant à inclure des informations contextuelles à long terme à chaque pas de temps. C'est une caractéristique inhérente aux couches convolutives empilées [69]. Il est important de souligner que, contrairement aux autres approches qui utilisent des convolutions *causal* pour la modélisation de la dépendance temporelle [70], le cas exploré ici suppose l'accès à l'enregistrement complet de la parole pour le calcul de chaque pas de temps de sortie. Cela nous permet de calculer des plongements dépendants de la langue à dimension fixe en s'appuyant sur des enregistrements complets. De plus, une architecture résiduelle est employée. A savoir, un ResNet-50 légèrement modifié [4], c'est-à-dire un ensemble de 50 couches organisé comme un empilement de blocs résiduels, est utilisé tout au long de nos expérimentations. Les entrées sont

des représentations temps-fréquence des données vocales, c'est-à-dire 13 MFCC, qui sont simplement traités comme des images à un canal. Nous introduisons une couche convolutive avant la pile résiduelle de ResNet qui réduit la dimension de fréquence à 1. De plus, un composant d'attention par image est utilisé au-dessus de la dernière couche convolutive en remplacement de la couche de sortie entièrement connectée d'origine. La dimension des canaux correspondant à la dernière couche convolutive dans la pile résiduelle donnera la taille finale des plongements, qui sont ensuite regroupés dans la dimension temporelle. Un schéma illustrant le modèle proposé est présenté dans la Figure 2.1. Deux stratégies d'entraînement différentes sont utilisées ensemble. Tout d'abord, nous entraînons directement le modèle pour la classification en projetant des représentations de données sur une couche de sortie à l'aide d'une couche supplémentaire entièrement connectée, et entraînons le modèle via l'estimation du maximum de vraisemblance, comme cela est généralement fait pour la reconnaissance du locuteur [47, 22, 3]. De plus, dans le but d'imposer la discriminabilité du langage dans y, la minimisation de la perte de triplet au-dessus des représentations est effectuée conjointement avec l'estimation du maximum de vraisemblance, en utilisant une métrique de distance basée sur la similarité en cosinus.

Spécifiquement pour le cas de la vérification du locuteur, le nombre d'enregistrements n'est pas constant entre les locuteurs. Nous concevons une approche pour les exemples d'échantillons afin de présenter des *mini-batches* équilibrés aux modèles tout au long de l'apprentissage. Les *mini-batches* sont construits en sélectionnant séquentiellement des exemples de chaque locuteur. Plus précisément, 5 enregistrements sont échantillonnés avec répétition. Nous illustrons un tel schéma d'échantillonnage dans la figure 2.8. L'indice de locuteur $i \in \{1, 2, ..., N_S\}$ survole l'ensemble des locuteurs et pour chaque valeur de i, un ensemble d'enregistrements choisis au hasard représentés par $x_{i,j}$ est ensuite renvoyé. Une telle approche fournit des *mini-batches* de taille $N_e = S \cdot R$, où R et S correspondent respectivement au nombre de locuteurs par *mini-batch* et d'enregistrements par locuteur. Alors que R est défini sur 5 comme dans l'exemple de la figure 2.8, S est défini sur 24, ce qui donne $N_e = 120$. Une époque d'entraînement est considérée comme terminée lorsque des ensembles de 5 enregistrements sont échantillonnés à partir de chaque locuteur 3 fois, et un budget de 500 époques est utilisé pour chaque cycle d'entraînement.

La littérature précédente a discuté en profondeur de la nécessité de trouver des ensembles de triplets produisant une perte de triplet élevée, c'est-à-dire des triplets *hard* dans le sens où $||y_a - y_-||_2 < ||y_a - y_+||_2$ résultant en un $\mathcal{L}_T$ informatif. Dans [71], les auteurs soutiennent que les modèles de représentation sont capables d'apprendre rapidement à encoder correctement les triplets triviaux, ce qui rend une grande partie des triplets possibles non informatifs. Ceci est illustré dans la figure 2.9 pour les triplets dans une sphère dans $\mathbb{R}^2$ qui représente un triplet *hard* sur la gauche, et ceux-ci apparaissent très fréquemment au début de l'entraînement, et des triplets triviaux à droite, qui représentent la majorité des ensembles d'exemples choisis au hasard après un nombre relativement faible d'itérations d'apprentissage. Étant donné que notre schéma de construction de *mini-batch* garantit que des enregistrements R par locuteur sont disponibles, nous utilisons donc une méthode de sélection de triplet en ligne similaire à celle introduite dans [62]. La version la plus récente du

modèle intègre tous les exemples d'apprentissage N_e, et toutes les paires positives possibles sont prises. Les paires négatives les plus dures sont ensuite sélectionnées de manière à correspondre au nombre de paires positives. Cette procédure devrait donner $N_t = S\frac{R(R-1)}{2}$, ce qui donne au plus $N_t = 240$ étant donné que $S = 24$ et $R = 5$ dans notre cadre, et les triplets tels que $\mathcal{L}_T = 0$ sont rejetés.

0.4.2 Discussion

Nous avons évalué l'utilisation de procédures d'entraînement multitâches où l'estimation du maximum de vraisemblance est combinée à l'apprentissage métrique afin de produire des représentations dépendantes de la langue ou du locuteur. Des stratégies spécifiques sont conçues pour chacune des deux tâches qui nous concernent: LID et ASV. Pour le cas du LID, l'évaluation sur des conditions non triviales, telles que la courte durée de la parole, les langues déroutantes et les enregistrements de tests correspondant à des langues non représentées dans les données d'apprentissage, indique la robustesse de l'approche proposée par rapport à plusieurs des repères connus améliorant de manière pertinente. Plus précisément, des améliorations de 28,51%, 36,83% et 39,88% ont été observées en ce qui concerne les i-vecteurs lorsque la notation est effectuée avec PLDA, pour trois conditions d'évaluation différentes, à savoir les langues de courte durée, déroutantes et l'ensemble ouvert, respectivement.

Pour le cas de l'ASV, la méthode d'entraînement proposée ainsi que les stratégies de régularisation introduites ont été évaluées sur les conditions introduites pour le NIST SRE 2016, dans lesquelles nous avons montré que : (i) Les modèles entraînés dans le cadre de plusieurs tâches surpassent ceux entraînés avec des pertes uniques. (ii) Les stratégies de mise en commun sophistiquées donnent les meilleures performances dans nos expériences, cependant il faut faire face aux difficultés d'entraînement supplémentaires données par de tels systèmes. Nous avons constaté que la pré-entraînement avec une méthode de mise en commun plus simple et enfin l'entraînement du modèle complet aident à cet égard. (iii) Nos meilleurs systèmes offrent des performances de vérification équivalentes ou supérieures par rapport aux méthodes de référence bien connues dans les conditions d'évaluation considérées. La fusion avec les systèmes de base a encore amélioré les performances finales.

0.5 Chapitre 3: Détection des menaces comme moyen d'obtenir une biométrie vocale robuste

Il existe des stratégies d'attaque simples qui peuvent être appliquées à tout type de reconnaissance de locuteur. Un exemple consiste à rejouer la voix de quelqu'un, ce qui est communément appelé **attaque physique** (PA). À titre d'exemple, on pourrait enregistrer la voix de quelqu'un

disant une commande pour obtenir un accès non autorisé à ses appareils portables. Une autre stratégie d'attaque, appelée *attaque logique* (LA), peut être définie à l'aide de la parole synthétisée. En fait, les avancées récentes dans les modèles génératifs conditionnels (par exemple, *Wavenet* [72]) peuvent être utilisées à cette fin pour les cas de conversion texte-parole ou de conversion vocale. Les conséquences potentielles de telles vulnérabilités sont énormes et vont de la perte financière à l'incrimination indue.

Étant donné que ces menaces décrites limitent l'utilisation de systèmes de hautes performances dans des applications réelles, une direction de recherche populaire ces dernières années a été de concevoir des contre-mesures contre de tels attaquants. Pour le cas spécifique de la reconnaissance du locuteur et de l'authentification vocale, par exemple, les récents défis de détection d'usurpation d'identité [38, 39, 9] ont été introduits dans le but de faire progresser l'état de l'art en matière de détection d'attaques. En général, les contre-mesures peuvent être regroupées en deux catégories: les méthodes *defense* et *detection*. Alors que les techniques de défense tentent d'améliorer la robustesse du modèle ou de supprimer les taux de réussite des attaques, les méthodes de détection adoptent une approche différente et tentent de déterminer si l'entrée est authentique ou a été manipulée d'une manière ou d'une autre. Bien que les deux directions soient prometteuses, nous nous concentrons sur l'approche de détection, car elle permet aux fournisseurs de services de savoir quand leurs systèmes sont attaqués. Plus précisément, nous nous intéressons aux approches de détection qui fonctionnent de manière *end-to-end*.

Nous remarquons en outre que, même si les données publiées pour les défis populaires de détection d'usurpation d'identité sont générées en s'assurant que les locuteurs et les types d'attaque varient selon les ensembles de données d'entrainement, de développement et d'évaluation, les systèmes de détection développés reposent sur l'hypothèse forte (et irréaliste) selon laquelle et les données de test sont **distribuées à l'identique** de sorte que la même stratégie d'attaque générale (LA ou PA) apparaîtra à la fois sur les données d'entrainement et de test. Ce faisant, différents modèles, architectures et représentations de l'audio sont utilisés pour les attaques PA ou LA. Cette configuration spécifique à la stratégie, cependant, n'est pas alignée sur des scénarios pratiques et réels où la stratégie d'attaque n'est pas connue *a priori*. Nous cherchons donc également à répondre à cette limitation et à proposer des stratégies de détection agnostiques aux attaques.

0.5.1 Mise en place experimentale

La première étape de notre pipeline de modélisation après l'extraction de caractéristiques correspond à la mise en correspondance d'un enregistrement donné dans un ensemble de descripteurs locaux $V_i \in \mathbb{R}^K$, c'est-à-dire un ensemble de vecteurs représentant des parties d'entrées dans la dimension temporelle. Pour obtenir un tel ensemble de descripteurs, nous considérons les résultats récents de [70] montrant que les réseaux de neurones convolutifs sont bien adaptés à la modélisation de la dépendance temporelle tout en évitant les problèmes d'entraînement courants observés dans

le cas des réseaux de neurones récurrents (par exemple, des gradients de disparition/explosion). À savoir, les réseaux de neurones convolutifs légers [73] sont utilisés étant donné leur faible nombre de paramètres par rapport à d'autres modèles convolutifs bien connus tels que les VGGs [74] ou les ResNets [4].

Les réseaux de neurones convolutifs légers (LCNN) [73], initialement proposés pour la reconnaissance faciale, et les ResNets sont utilisés dans notre cas avec des couches d'attention, visant à permettre au modèle de traiter des enregistrements de longueur variable tout en se concentrant sur des portions spécifiques d'entrées. Les modèles légers sont utilisés étant donné la quantité limitée de données d'entrainement disponibles pour le paramètre d'évaluation considéré, ce qui pourrait dégrader des performances. Nous avons en outre utilisé deux variantes LCNN distinctes et observé que les modèles produisant les performances les plus élevées dépendent du choix des caractéristiques d'entrée. L'évaluation est effectuée dans les conditions de défi ASVspoof 2019, qui fournissent des partitions de données d'entraînement et de développement pour les attaques logiques générées avec des systèmes de conversion texte-parole et voix, ainsi que pour les attaques physiques correspondant à des enregistrements rejoués dans diverses configurations simulées.

Nous appliquons en outre quelques modifications aux modèles en fonction du type de représentation de la parole utilisé. Pour le cas des coefficients cepstraux, nous incluons une couche convolutive d'entrée chargée de réduire la dimension des coefficients à 1, et procédons à un LCNN modifié à 29 couches (LCNN29) qui effectue des convolutions sur la dimension temporelle uniquement. Pour les caractéristiques spectrales, d'autre part, nous saisissons directement des exemples dans un LCNN spatio-temporel standard contenant 9 couches (LCNN9), et une couche convolutive de sortie est ensuite utilisée afin de réduire la dimension de fréquence à 1. ResNet-18 a été utilisé à travers types d'attaques. Dans tous les cas on se retrouve avec un ensemble de vecteurs de dimension K correspondant au nombre de canaux de la dernière couche convolutive.

0.5.2 Discussion

Nous avons abordé le problème de la détection des attaques d'usurpation d'identité contre les systèmes de vérification automatique des locuteurs *end-to-end*, c'est-à-dire que nos modèles mappent les caractéristiques vocales en scores. Tout d'abord, nous avons proposé des variations des réseaux de neurones convolutifs légers compte tenu des limitations de la quantité disponible de données d'entraînement. Les performances du paramètre proposé sont ainsi évaluées sur les données introduites pour le défi ASVspoof 2019, auquel cas nous sommes en mesure d'obtenir des améliorations significatives des performances de détection par rapport aux systèmes de référence bien connus. Nous nous sommes ensuite appuyés sur cette approche et avons introduit différentes stratégies visant à augmenter les données d'entraînement afin de permettre l'utilisation de modèles plus grands pour définir des détecteurs d'attaquants usurpateurs. Ce faisant, nous avons pu multiplier par cinq la taille des corpus disponibles, tout en les rendant plus diversifiés, introduisant ainsi

des effets de régularisation qui ont amélioré la généralisation à de nouvelles conditions. En fait, contrairement aux travaux antérieurs, qui se concentraient sur de simples pipelines de classification ou des réseaux de neurones relativement petits, nous avons pu, via l'augmentation des données, entraîner efficacement des modèles convolutifs couramment utilisés tels que les TDNN.

Comme contribution principale, nous avons introduit une approche basée sur l'ensemble dans le but de permettre aux détecteurs d'être efficaces contre différents types d'attaques d'usurpation d'identité. Nous avons donc proposé un cadre contenant trois composants tels que deux d'entre eux sont connus pour bien performer individuellement dans chacune des deux stratégies d'attaque considérées. Le troisième est ensuite entraîné pour décider comment combiner la décision des autres systèmes en fonction de l'entrée qui lui est présentée. L'évaluation nous a conduits à la conclusion que l'approche proposée surpasse la procédure standard de mise en commun de diverses données d'entraînement et d'entraînement d'un seul modèle, et atteint des performances compétitives par rapport à des modèles spécialisés, entraînés sur des données organisées de manière à correspondre à la condition d'évaluation connue à l'avance. En fait, pour le cas spécifique des attaques PA, nos modèles ont surpassé les systèmes spécialisés privilégiés. Étant donné que la plupart des travaux actuels sur les contre-mesures pour les attaques d'usurpation d'identité se concentrent sur des systèmes spécialisés pour des types particuliers de stratégies d'attaque, nous pensons que l'approche proposée ici est une première étape dans le sens de permettre le déploiement de systèmes de vérification automatique du locuteur sans risque des failles de sécurité, car il fonctionne bien dans les différentes stratégies d'attaque que nous avons prises en compte dans notre évaluation.

0.6 Chapitre 4: TDNN auto-attentif à plusieurs niveaux: une approche générale et efficace pour résumer la parole en représentations discriminantes au niveau de la phrase

Les opérations de regroupement temporel dans les TDNN sont destinées à permettre le calcul de représentations globales au niveau de la phrase de l'entrée audio et à produire un seul vecteur représentant une séquence entière de caractéristiques acoustiques à partir du signal d'entrée. D'un point de vue pratique, les représentations globales peuvent être utiles pour les tâches où la capacité de traiter des séquences de longueurs variables est requise, telles que l'identification du locuteur et/ou de la langue parlée. Cependant, l'inclusion d'une opération de mise en commun au sein d'une architecture *feed-forward* entraîne des défis et des limites, notamment:

1. *Architectures trop spécialisées*: les représentations apprises de niveau inférieur (c'est-à-dire plus proche des entrées) ou de haut niveau (c'est-à-dire plus proche des sorties) peuvent être plus ou moins efficaces selon la tâche/les données sous-jacentes d'intérêt. Par exemple, il est peu probable que le même type de représentation soit utile pour des tâches telles que la vérification du locuteur et l'identification de la langue, car les indices dépendant du locuteur

sont généralement indépendants de la phonétique sous-jacente dans un signal, tandis que la détermination des langues nécessite que les informations phonétiques soient saillantes. Compte tenu de ces variations entre les tâches, nous soutenons que la détermination du bon niveau d'abstraction des représentations apprises, principalement en décidant où effectuer les opérations de regroupement global, dépend de la tâche et, en tant que telle, nécessite la conception d'une architecture spécifique pour chaque tâche différente.

2. *Ignorer les informations complémentaires*: Les TDNN effectuent des opérations de regroupement uniquement après la sortie de la couche convolutive finale, ainsi les caractéristiques globales des autres niveaux du modèle ne sont pas explicitement prises en compte. Une telle opération pourrait éliminer des informations potentiellement discriminatoires pour les tâches en aval. Nous soutenons qu'une solution qui tient compte simultanément des caractéristiques regroupées dans différentes parties du modèle a le potentiel de produire des représentations apprises plus discriminantes, résultant en des modèles plus généralisables.

Afin de remédier à ces limitations, nous proposons de modifier l'architecture TDNN et de calculer l'opération de regroupement indépendamment sur les cinq couches de convolution du modèle. De plus, nous proposons l'utilisation d'une couche auto-attentive [48] pour donner au modèle la possibilité de sélectionner, au moment de l'apprentissage, le meilleur schéma de combinaison entre les caractéristiques obtenues à différents niveaux et de calculer une nouvelle séquence de vecteurs globaux comme fonction de l'ensemble des représentations. Enfin, une dernière opération de regroupement est appliquée sur la séquence résultante pour produire une représentation au niveau de l'énoncé. Le schéma proposé offre les avantages suivants par rapport aux TDNN conventionnels:

1. *Polyvalence*: par opposition à la conception d'une nouvelle architecture pour chaque nouvelle tâche, l'architecture proposée introduit une approche orientée aux données qui permet au modèle d'apprendre quelles couches fournissent des informations plus discriminantes pour la mise en commun globale. En tant que telle, une même architecture peut être réutilisée pour différentes tâches.
2. *Généralité*: les facteurs globaux dans chaque couche sont explicitement pris en compte, par opposition à la dernière couche convolutive uniquement. Ainsi, des informations complémentaires obtenues à partir de différentes couches peuvent être exploitées. Nous soulignons en outre qu'un tel schéma définit des classes de modèles qui contiennent des mécanismes d'agrégation simples comme des cas particuliers; c'est-à-dire que des schémas simples tels que la moyenne des représentations regroupées à différents niveaux ou la sélection d'une couche spécifique peuvent tous être récupérés par le modèle proposé si ce sont les meilleures solutions pour la tâche/les données à accomplir.
3. *Apprentissage*: L'opération de mise en commun entre les couches agit comme des connexions de saut/résiduel, produisant ainsi des pertes qui sont plus faciles à entraîner contre [4, 45].

4. *Transférabilité*: Nous remarquons en outre que la stratégie de mise en commun proposée peut être utilisée dans n'importe quelle architecture qui maintient la dimension des données fixe dans tout l'encodeur (par exemple, ResNets [4]).

Nous mesurons ainsi la polyvalence de l'architecture proposée, appelée TDNN auto-attentif multi-niveaux (ML-TDNN), sur deux tâches *end-to-end*: l'identification de la langue parlée et la détection d'attaque par usurpation. De plus, nous évaluons davantage le modèle proposé lorsqu'il est utilisé comme encodeur pour la vérification du locuteur. Dans tous ces cas, nous trouvons des preuves soutenant l'affirmation selon laquelle les informations globales des couches de bas niveau contiennent des informations complémentaires qui peuvent être exploitées à des étapes ultérieures du modèle pour améliorer les performances.

0.6.1 Mise en place experimentale

Le modèle proposé s'appuie sur l'architecture TDNN originale discutée au chapitre 1, illustrée à la figure 1.7, et plus détaillée dans le tableau 1.1. Dans ce cas, une séquence d'entrée de longueur T est notée $x_{[1:T]}$ représentant des vecteurs de caractéristiques acoustiques ordonnées de dimension d (par exemple, MFCCs), où chaque $x_i \in \mathbb{R}^d, i \in [T]$,. L'ensemble d'entités généré par une pile de couches est noté $y_{1:T} \in \mathbb{R}^D, i \in [T]$. Pour le TDNN standard, l'opération de mise en commun temporelle effectuée à la fin de la pile convolutive convertit la séquence $y_{1:T}$ en une représentation au niveau de l'énoncé V, étant le seul composant du modèle capable de résumer le contenu du séquence dans une représentation globale. Nous soutenons que c'est : i) *trop restrictif*, car il ignore les informations précieuses disponibles dans les couches précédentes du modèle, et ii) *trop spécifique*, car différentes tâches devraient obliger les concepteurs à rechercher la bonne couche où appliquer l'opération de mutualisation. En tant que tel, nous proposons le schéma de pooling multi-niveaux illustré à la figure 4.1.

Ensuite, nous employons une couche d'attention [48] afin que chaque V_k puisse être pris en compte en fonction de sa pertinence afin d'aboutir à des représentations discriminantes. En d'autres termes, nous tirons parti de l'ensemble en profondeur des résumés globaux de la séquence d'entrée en incluant un composant de modélisation de séquence dans l'architecture. Dans notre cas, le nombre de têtes d'attention est traité comme un hyperparamètre. Il est important de souligner que même si le coût de calcul des couches d'auto-attention est proportionnel au carré de la longueur d'entrée, le modèle proposé est favorisé par le fait que la longueur de séquence est fixe et modérée, car elle correspond à la profondeur de la pile convolutive (c'est-à-dire 5). De plus, une telle couche définit une classe de modèles qui incluent des schémas aussi simples que la moyenne/sélection à des relations non linéaires plus compliquées entre les éléments. Cependant, n'importe quelle couche de modélisation de séquence alternative peut être utilisée dans ce cas, produisant des variations du modèle proposé.

Les évaluations empiriques réalisées ici visent à mettre en évidence la polyvalence de l'architecture proposée; en tant que tel, nous montrons que sa réutilisation à travers les tâches ne nécessite pas d'adaptations spécifiques aux données. Ainsi, nous évaluons délibérément le modèle proposé à travers différentes tâches et ensembles de données. Les ensembles de données et les tâches utilisés pour l'évaluation sont les suivants: détection des attaques d'usurpation d'identité, identification de la langue parlée et vérification du locuteur.

0.6.2 Discussion

Nous avons introduit une variante de l'architecture TDNN dans laquelle les opérations de regroupement temporel sont effectuées sur toutes les couches de la pile convolutive plutôt que seulement à son extrémité, comme dans l'architecture standard. Nous appelons l'architecture proposée ML-TDNN. En particulier, nous proposons un composant de modèle visant à combiner des représentations globales de différentes couches en traitant les statistiques globales calculées dans différentes parties du modèle comme des séquences, et en traitant lesdites séquences à l'aide d'une couche attentive multi-têtes. L'évaluation est effectuée sur deux tâches *end-to-end* (détection d'attaque d'usurpation d'identité et identification de la langue) et une tâche d'encodage (vérification du locuteur). Les résultats expérimentaux ont montré que la méthode proposée surclassait systématiquement diverses références utilisant des caractéristiques globales obtenues à partir d'une seule couche, soulignant ainsi que des informations complémentaires pouvaient être efficacement exploitées à partir de couches de bas niveau proches de l'entrée. De plus, les lignes de base de chaque tâche considérée consistaient en des modèles spécialisés pour ladite tâche. La méthode proposée a obtenu de meilleurs (ou au pair) résultats que ces modèles spécialisés, montrant ainsi leurs capacités de généralisation à travers les tâches. D'autres améliorations ont également pu être observées une fois, au moment du test, l'auto-fusion appliquée à partir de représentations obtenues à partir de différentes couches du même modèle.

La principale limitation de la méthode proposée est le fait que la polyvalence obtenue a un coût de calcul supplémentaire, car la mise en commun des opérations sur plusieurs couches et la combinaison des représentations résultantes entraînent des calculs supplémentaires. En tant que tels, les cas à ressources limitées, tels que les applications sur l'appareil (de périphérie) peuvent ne pas bénéficier directement de ce type de modèle. Néanmoins, des mécanismes pour la compression du modèle ou la distillation des connaissances [75] pourraient être utilisés pour atténuer ce problème.

0.7 Chapitre 5: Apprentissage des (pseudo) espaces métriques pour la vérification discriminatif

Ici, nous nous intéressons au cas d'apprentissage métrique, et plus important encore, nous tournons notre attention vers son application au problème de vérification, c'est-à-dire celui de comparer des paires de données et de déterminer si elles appartiennent à la même classe. Le problème de vérification se pose dans les applications où des comparaisons de deux petits échantillons sont nécessaires, telles que la vérification faciale/empreinte digitale/vocale [76], la récupération d'images [77, 78], et ainsi de suite. Au moment du test, l'inférence est souvent effectuée pour répondre à deux types de questions :

1. Deux exemples donnés appartiennent-ils à la même classe?
2. Un exemple de test appartient-il à une classe revendiquée spécifique?

Dans les deux cas, les exemples de test peuvent appartenir à des classes qui n'ont jamais été présentées au modèle lors de l'apprentissage. Les approches de vérification actuelles sont généralement composées de plusieurs composants formés de manière gourmande [79, 3], et une approche *end-to-end* fait toujours défaut.

Les espaces euclidiens ne conviendront pas, en général, pour représenter tout type de structure souhaité exprimé dans les données (par exemple, asymétrie [60] ou hiérarchie [80]). Pour éviter d'avoir à sélectionner une distance adéquate compte tenu de chaque nouveau problème auquel nous sommes confrontés, ainsi que pour faire face aux difficultés d'apprentissage mentionnées précédemment, nous proposons d'augmenter le cadre d'apprentissage métrique et d'entraîner conjointement un encodeur (qui intègre des données brutes dans un espace de dimension inférieure) et un (pseudo) modèle de distance adapté au problème d'intérêt. Une approche de vérification de bout en bout est ensuite définie en utilisant une telle pseudo-distance pour calculer les scores de similarité. Les deux modèles ensemble, paramétrés par des réseaux de neurones, définissent un (pseudo) espace métrique dans lequel l'inférence peut être effectuée efficacement puisque désormais les propriétés sémantiques des données (par exemple, les écarts entre les classes) sont codées par des scores. Ce faisant, nous avons constaté que plusieurs interprétations apparaissent à partir d'une telle pseudo-distance apprise, et cela peut être interprété comme un rapport de vraisemblance dans un test d'hypothèse de Neyman-Pearson, ainsi qu'une mesure approximative de divergence entre les distributions conjointes des paires d'exemples positifs (mêmes classes) et des paires d'exemples négatives (classes différentes). De plus, même si nous n'appliquons pas de modèles pour satisfaire les propriétés d'une métrique réelle, nous observons empiriquement l'apparition de telles propriétés.

0.7.1 Mise en place experimentale

Nous considérons le cadre où à la fois un mécanisme de codage, ainsi qu'un certain type de similitude ou de distance entre les points de données doivent être appris. Supposons que $\mathcal{E} : \mathbb{R}^D \to \mathbb{R}^d$ et $\mathcal{D} : \mathbb{R}^d \times \mathbb{R}^d \to (0, 1)$ soient des mappages déterministes auxquels on se référera comme codeur et modèle de distance, respectivement, et seront tous deux paramétrés par des réseaux de neurones. De telles entités ressemblent à un espace métrique, nous l'appellerons donc *pseudo espace métrique*. Nous avons observé empiriquement que l'introduction de propriétés de distance dans $\mathcal{D}$, c'est-à-dire en le contraignant à être symétrique et en l'obligeant à satisfaire l'inégalité triangulaire, n'a pas entraîné d'amélioration des performances, mais a rendu l'entraînement instable. Cependant, étant donné que les modèles entraînés se comportent approximativement comme une distance réelle, nous utilisons l'analogie, mais fournissons en outre des interprétations alternatives des sorties de $\mathcal{D}$.

Tant $\mathcal{E}$ que $\mathcal{D}$ sont implémentés en tant que réseaux de neurones. Dans nos expériences, $\mathcal{E}$ sera convolutif (2-d pour les images et 1-d pour l'audio) tandis que $\mathcal{D}$ est un empilement de couches entièrement connectées qui prennent en entrée des representations concaténés de paires d'exemples. Nous effectuons des étapes de mise à jour simultanées pour $\mathcal{E}$ et $\mathcal{D}$ car nous avons observé que cela était plus rapide que les mises à jour alternatives, tout en offrant les mêmes performances. Des stratégies de régularisation standard telles que la décroissance du poids et le lissage des étiquettes [81] sont également utilisées. Nous avons constaté empiriquement que l'utilisation d'une perte de classification auxiliaire multi-classes accélère considérablement l'entrainement. Étant donné que notre approche nécessite des étiquettes pour déterminer quelles paires d'exemples sont positives ou négatives, nous utilisons davantage les étiquettes pour calculer une telle perte auxiliaire.

Pour évaluer le cadre décrit, nous nous appuyons sur trois ensembles d'expériences : (i) une analyse de preuve de concept, (ii) un benchmark ASV à grande échelle, et (iii) des expériences supplémentaires où nous analysons le comportement de la méthode proposée en termes de présence de propriétés de distance, ses performances sous des décalages de distribution, et sa sensibilité au choix d'architecture pour le modèle de distance. Dans la première partie de notre évaluation, nous menons des expériences de validation de principe qui tirent parti d'ensembles de données d'images et de modèles bien établis pour simuler les cas de vérification et valider notre proposition par rapport à l'apprentissage métrique sous des distances standard. Pour cela, nous rapportons les résultats de tous les essais créés pour les ensembles de tests de CIFAR-10 et *Mini*ImageNet. Dans le premier, les mêmes 10 classes d'exemples apparaissent pour les partitions d'entrainement et de test, dans ce que nous appelons la vérification en *closed-set*. Pour le cas de *Mini*ImageNet, étant donné que cet ensemble de données a été conçu pour des applications d'apprentissage à *few-shot*, nous avons une évaluation *open-set* pour la vérification puisqu'il existe 64, 16 et 20 classes disjointes d'entraînement, de validation et de test.

Nous passons ensuite à une *évaluation réaliste à grande échelle*. Pour cela, nous utilisons le corpus VoxCeleb [82, 7], correspondant à des enregistrements audio d'entretiens extraits de vidéos

youtube, ce qui signifie qu'il n'y a aucun contrôle sur les conditions acoustiques présentes dans les données. De plus, alors que la majeure partie du corpus correspond à la parole en anglais, d'autres langues sont également présentes, de sorte que les enregistrements de test proviennent de locuteurs différents par rapport aux données d'entrainement, et potentiellement aussi de langues et d'environnements acoustiques différents. Nous utilisons spécifiquement la deuxième version du corpus afin que les données d'entraînement soient composées d'enregistrements de 5994 locuteurs tandis que trois ensembles de tests sont disponibles : (i) **VoxCeleb1 Test set**, qui est composé d'énoncés de 40 locuteurs, (ii) **VoxCeleb1-E**, c'est-à-dire la première version complète des données contenant 1251 locuteurs, et (iii) **VoxCeleb1-H**, correspondant à un sous-ensemble des essais dans **VoxCeleb1-E** afin que les non-cible essai soient conçus pour être difficiles à discriminer en utilisant les métadonnées pour faire correspondre des facteurs tels que la nationalité et le sexe des locuteurs. Nous rapportons ensuite les expériences réalisées pour observer si les sorties de $\mathcal{D}$ présentent des propriétés de distances réelles, et enfin vérifier l'influence de l'architecture de $\mathcal{D}$ sur les performances finales.

0.7.2 Discussion

Nous avons introduit un cadre dans lequel des tests à 2 échantillons sur petits échantillons peuvent être effectués efficacement. Cela convient et compare les paires de données pour déterminer si elles appartiennent à la même classe. Plusieurs interprétations d'un tel cadre sont fournies, y compris l'apprentissage conjoint d'un encodeur et d'un métrique, ainsi qu'une estimation contrastive sur des paires de données. Nous avons utilisé des résultats d'estimation contrastifs pour montrer que les solutions du problème posé produisent des règles de décision optimales dans des cas de vérification, résultant en des décisions correctes pour tout choix de seuil. En termes de contributions pratiques, la méthode proposée simplifie à la fois l'entrainement dans le cadre d'apprentissage métrique, car elle ne nécessite aucun schéma pour sélectionner des paires négatives d'exemples, et simplifie également les pipelines de vérification, qui sont généralement constitués de plusieurs composants individuels, chacun un contribuant à des défis spécifiques lors des phases de formation et de test. Nos modèles peuvent être utilisés *end-to-end* en utilisant les sorties de $\mathcal{D}$ pour noter les essais de test produisant de solides performances même dans des conditions ouvertes à grande échelle et réalistes où les classes de test sont différentes de celles observées au moment d'entrainement.

0.8 Chapitre 6: Apprentissage des partitions pour définir des modèles d'apprentissage polyvalents

Dans cette contribution, nous nous appuyons sur le cadre introduit au chapitre 5 pour définir l'idée de *TEMPlate LEarners* (TEMPLE): un ensemble de composants de modèle et une procédure d'apprentissage qui peuvent être réutilisés à travers différents scénarios pour atteindre des perfor-

mances similaires à celles des modèles spécifiques à une tâche, produisant un cadre de modélisation *polyvalent à usage général*. Pour ce faire, nous utilisons à nouveau des méthodes d'apprentissage métrique, mais considérons également les approches géométriques introduites à l'origine pour aborder la classification à *few-shot*; *réseaux prototypes* en particulier [83]. Plus précisément, nous nous concentrons sur les cas d'apprentissage métrique où à la fois un encodeur et un modèle de similarité ou de distance sont entraînés conjointement [84, 85, 86], mais nous introduisent un ensemble de prototypes de classe utilisés afin d'attribuer des points aux classes. Les modèles TEMPLE comprennent trois composants principaux:

1. Un encodeur qui mappe des données dans un espace de dimension inférieure;
2. Un modèle de similarité qui mappe une paire de représentations concaténées en un score de similarité;
3. Un ensemble de prototypes de classe où chacun résume une classe entière en un vecteur.

Sur la base de modèles définis par lesdits composants, nous pouvons alors concevoir différents mécanismes d'inférence en fonction de la tâche d'intérêt. Pour le cas de la classification multiclasse, par exemple, on peut prédire la classe d'une instance de test particulière en mesurant sa similarité par rapport à chaque prototype et en l'attribuant à la classe dont le prototype est le plus similaire. De même, des tâches reposant sur des comparaisons par paires peuvent être effectuées, telles que la vérification (comparer deux instances de données et déterminer si elles appartiennent à la même classe) ou la récupération (comparer une instance de test à une galerie et déterminer les éléments k dans la galerie le l'instance de test considérée est la plus similaire). De plus, à chaque fois que de nouvelles classes apparaissent, l'adaptation du modèle consiste simplement à mettre à jour la liste des prototypes, tout en gardant l'encodeur et la similarité inchangés, permettant ainsi une adaptation rapide et évitant des problèmes tels que l'oubli des classes passées ou le surapprentissage des nouvelles.

Nous soulignons que, étant donné la large applicabilité de TEMPLE, contrairement aux chapitres précédents, nous élargissons ici notre champ d'application et évaluons nos propositions sur des tâches au-delà des applications liées à la biométrie vocale. En particulier, nous utilisons à la fois des expériences ASV et des références standard sur les images pour fournir des preuves empiriques et soutenir l'affirmation selon laquelle les modèles définis par TEMPLE fonctionnent à égalité avec les modèles spécialisés spécifiques à une tâche dans différents scénarios et types de données. En plus de cela, nous avons observé que les classificateurs définis dans ce cadre améliorent la robustesse contre les attaques et les changements entre les distributions de données d'entraînement et de test. De plus, l'approche proposée prend en charge l'inclusion de nouvelles classes apparaissant une fois la formation terminée, ce qui nécessite simplement d'inclure de nouveaux prototypes (ou de repartitionner l'espace) obtenus à partir de petits échantillons. Cela donne un mécanisme simple mais compétitif pour la classification à *few-shot*.

0.8.1 Mise en place experimentale

L'apprentissage du modèle est effectué pour appliquer les propriétés suivantes:

1. La similarité mesurée entre un exemple particulier et le prototype correspondant à ses étiquettes de classe doit être élevée par rapport aux similarités mesurées entre des prototypes représentant des classes différentes.
2. Les similitudes mesurées entre les exemples de la même classe doivent être élevées, tandis que les exemples de classes différentes doivent donner un faible score de similitude.

Nous concevons des objectifs d'entraînement visant à faire respecter ces propriétés. Pour la première propriété, nous considérons un échantillon d'apprentissage de taille m et utilisons le critère d'entropie croisée standard, mais **utilisons la similarité mesurée entre une instance d'apprentissage et chaque prototype comme ensemble de logits**, comme par opposition aux couches de sortie définies par une transformation affine, couramment utilisées dans les classificateurs standard. Pour que la similarité apprise soit **significative** pour les comparaisons par paires, nous utilisons un objectif de classification binaire également utilisé par [87] et [86]. Cette classification vise à discriminer des paires d'exemples de la même classe et de classes différentes

Dans l'évaluation que nous effectuons, nous cherchons à tester l'approche proposée à travers une variété de tâches et de modalités de données puisque notre objectif principal est de montrer que les instances TEMPLE fonctionnent au moins à égalité avec les méthodes spécialisées considérant différents cas. À savoir, nous commençons par des tâches qui reposent sur des comparaisons par paires d'instances de test et exécutons des évaluations sous le paramètre de vérification en nous concentrant sur une tâche ASV sur le paramètre de vérification à grande échelle défini par le corpus VoxCeleb [82, 7]. Nous procédons ensuite à des benchmarks d'images auquel cas, en plus de vérifier les performances de prédiction, nous évaluons différentes notions de robustesse des classifieurs induites par TEMPLE. De telles références d'images couvrent les tâches suivantes: classification multi-classes, auquel cas nous évaluons les classificateurs basés sur TEMPLE à l'aide de MNIST [88] et CIFAR-10 [89], et *observe une précision robuste améliorée.* Nous effectuons en outre une évaluation sur des tâches de reconnaissance d'objets en considérant des images à plus grande résolution et sous un décalage de domaine. Pour cela, nous utilisons le benchmark PACS [90] où nous constatons que les stratégies de classification proposées introduites ici **surclassent les alternatives récemment introduites**, et plus important encore, c'est le cas dans les conditions les plus difficiles (par exemple, s'entraîner sur des images naturelles et évaluer sur des croquis). Nous évaluons ensuite notre approche proposée sur les tâches de récupération d'images en utilisant des références populaires telles que CARS196 [91] et CUB200-2011 [92]. Enfin, nous discutons des stratégies pour repartitionner facilement l'espace afin que de nouvelles classes puissent être évaluées au moment du test, auquel cas nous rapportons des expériences utilisant *Mini*ImageNet [93]. Les ablations sont également rapportées en utilisant ImageNet [94] pour montrer l'importance de l'utilisation de la perte de classification auxiliaire.

Nous remarquons que la procédure d'entraînement présentée dans l'algorithme 2 est utilisée pour les **modèles d'entraînement utilisés pour toutes les tâches discutées ci-dessus**, et qu'aucune spécialisation pour aucune tâche d'intérêt n'est effectuée puisque nous cherchons des preuves concernant l'efficacité de la l'approche proposée consiste à produire un ensemble suffisamment général de composants (c'est-à-dire $\mathcal{E}$, $\mathcal{S}$ et $\mathcal{C}$) et un algorithme d'entraînement qui fonctionnent au même niveau ou mieux que les alternatives. En tant que tel, à travers différents ensembles de données et tâches, l'encodeur $\mathcal{E}$ est le seul composant spécifique à chaque évaluation dans le sens où son architecture nécessite une spécification pour chaque source de données spécifique.

0.8.2 Discussion

Nous avons introduit TEMPLE: un ensemble de composants de modèle et une procédure de formation qui simplifie la réutilisation des procédures d'apprentissage à travers différents types de tâches et de données. Les composants du modèle sont donnés par les éléments suivants: un encodeur responsable de mapper des données vers un espace de dimension inférieure, un modèle de similarité qui génère un score de similarité lorsqu'une paire d'intégrations est donnée, et un ensemble de prototypes de classe où chacun représente une classe observée pendant la formation. Au moment du test, différents schémas d'inférence peuvent être définis au-dessus desdits composants afin de permettre son utilisation dans différents cas. Nous avons présenté des preuves empiriques montrant que les classificateurs définis sous TEMPLE produisent des améliorations, notamment : (1) une meilleure robustesse à l'adversaire, puisque de petites perturbations de la norme ont été observées pour avoir un effet moindre sur l'inférence basée sur la distance par rapport aux classificateurs standard. (2) une robustesse améliorée contre les changements de distribution, ce qui indique que la stratégie de formation proposée est plus efficace pour éviter les modèles qui reposent sur des corrélations entre des facteurs et des étiquettes spécifiques au domaine. Dans ce cas, les informations de domaine ne sont pas aussi utiles qu'elles peuvent l'être pour le cas de l'estimation du maximum de vraisemblance avec des classificateurs standard. De plus, les performances d'un ensemble de tâches telles que la vérification et la récupération d'images ont en outre montré que les modèles TEMPLE étaient compétitifs ou meilleurs que les alternatives conçues pour ces applications particulières.

0.9 Conclusions

Nous avons proposé plusieurs méthodes pour améliorer les performances des modèles basés sur les réseaux de neurones ciblant spécifiquement la biométrie vocale et d'autres tâches connexes, telles que l'identification de la langue parlée. De plus, nous considérons en outre la tâche de détecter les attaquants usurpateurs de tels systèmes. De plus, des propositions plus largement applicables sont discutées dans les derniers chapitres, et celles-ci ont des applications potentielles en dehors du domaine de la parole, donc des applications d'images sont également explorées, y compris

la reconnaissance d'objets à partir d'images, la récupération d'images et la robustesse contre les perturbations. A ce titre, nous présentons les conclusions de la thèse sous deux parties: *contributions appliquées* comprenant les chapitres 2-4 se concentrant sur les cas de vérification automatique du locuteur (ASV), identification de la langue (LID) et les contre-mesures d'usurpation d'identité. La deuxième partie correspond à davantage de *contributions fondamentales* dans le sens où elles sont applicables à tous les domaines et comprennent les chapitres 5 et 6.

Contributions appliquées: Nous avons apporté des contributions en termes à la fois de conception de formation et d'architectures de modèles, ce qui a permis d'améliorer les représentations des niveaux d'énoncé dépendant de la langue et du locuteur. En particulier, nous avons introduit un schéma multi-tâches où une combinaison d'apprentissage métrique et d'estimation de vraisemblance maximale est utilisée comme signal d'apprentissage pour les réseaux de neurones. Des stratégies de formation pratique supplémentaires sont également introduites, notamment des schémas d'augmentation des données et des approches d'échantillonnage résultant en des *mini-batches* contenant des paires d'exemples informatifs.

Pour la détection de l'usurpation d'identité, nous avons introduit des architectures de modèle, ainsi que des procédures d'entraînement résultant en des prédicteurs qui fonctionnent *end-to-end*, c'est-à-dire que nos modèles mappent directement les caractéristiques vocales dans les scores. Nous avons proposé des variantes à l'architecture de réseau de neurones convolutifs légers en ajoutant une couche d'attention pour détecter les attaques de durée arbitraire. Les performances ont été validées sur les données du défi ASVspoof 2019, où nous avons montré des améliorations significatives des performances de détection par rapport aux systèmes de référence bien connus. Nous avons également observé empiriquement que la meilleure représentation de la parole pour la détection d'usurpation d'identité dépendait de la stratégie d'usurpation d'identité utilisée. En tant que tel, nous avons proposé une approche basée sur l'ensemble dans le but de permettre aux détecteurs d'être efficaces contre différents types d'attaques d'usurpation d'identité contre les systèmes de vérification des locuteurs. L'ensemble reposait sur trois composants, dont deux étaient optimisés par attaque et le troisième pour décider de la meilleure façon de combiner leurs décisions en fonction du signal vocal d'entrée. L'évaluation a montré que la méthode proposée (i) surpassait les méthodes qui combinent les données des deux types d'attaques (entraînement multi-conditions) et entraînent un seul modèle, ainsi que (ii) atteint des performances compétitives par rapport aux modèles spécialisés formés sur des données organisées pour correspondre à la condition d'évaluation connue à l'avance. En fait, pour le cas spécifique des attaques PA, notre modèle proposé a surpassé les systèmes spécialisés privilégiés.

Enfin, nous avons introduit une variante de l'architecture TDNN - omniprésente dans le contexte de la biométrie vocale - où les opérations de regroupement temporel sont effectuées sur toutes les couches de la pile convolutive, plutôt que seulement à son extrémité. Nous appelons l'architecture proposée ML-TDNN (multi-level self-attention TDNN). En particulier, nous proposons un composant de modèle visant à combiner des représentations globales de différentes couches en traitant

les statistiques globales calculées dans différentes parties du modèle comme des séquences. Nous utilisons ensuite une couche attentive pour traiter ces séquences et finalement obtenir une représentation au niveau de l'énoncé à partir de la séquence traitée des statistiques globales.

Contributions fondamentaux: Nous avons proposé un cadre d'apprentissage de métrique augmentée où un encodeur et une (pseudo) distance sont entraînés conjointement. Cette paire de composants définit un (pseudo) espace métrique où l'inférence peut être effectuée efficacement pour la vérification. Nous utilisons le terme *pseudo* pour indiquer qu'un tel composant doit seulement ressembler à une distance, alors que l'approche proposée ne l'oblige pas à satisfaire les propriétés des distances réelles. Nous posons ensuite un problème d'estimation qui résulte sur des (pseudo) espaces métriques où les relations sémantiques entre les points de données, telles que définies via des étiquettes, peuvent être évaluées via des mesures de distance. En particulier, nous exploitons les résultats d'estimation contrastive pour montrer que les solutions optimales du problème posé sont telles que: (i) le modèle de distance optimal pour tout encodeur fixe donne le rapport de vraisemblance pour un test de rapport de vraisemblance de Neyman-Pearson, et (ii) l'encodeur optimal induit une forte divergence Jensen-Shannon entre les distributions conjointes des paires d'exemples positives et négatives. Ces résultats impliquent que, pour des modèles d'encodeur et de distance optimaux, l'utilisation de mesures de distances induites par les modèles appris aboutit à des règles de décision optimales pour les paramètres de vérification, entraînant des décisions correctes pour tout choix de seuils positifs. Pour une mise en oeuvre pratique de notre méthode proposée, nous avons paramétré les deux composants en tant que réseaux de neurones. De plus, un algorithme d'apprentissage simple a été proposé, qui ne nécessite pas d'étapes lourdes, telles que le minage dur-négatif, souvent nécessaire dans les cas d'apprentissage métrique standard. Les évaluations sur les tâches de vérification à grande échelle fournissent des preuves empiriques de l'efficacité de l'utilisation directe des sorties de la distance apprise pour l'inférence, surpassant les classificateurs en aval couramment utilisés.

Enfin, nous avons tiré parti de l'approche décrite ci-dessus et l'avons étendue afin que d'autres types de tâches puissent être pris en charge, en plus de la vérification. En particulier, nous avons introduit un troisième composant correspondant à un ensemble de prototypes de classes: un ensemble de vecteurs, chacun représentant une classe particulière. Avec les trois composants, nous avons introduit *TEMPlate LEarners* (TEMPLE), un ensemble de composants de modèle accompagné d'une procédure d'apprentissage qui peut être réutilisé pour différentes tâches et types de données. Sous TEMPLE, la définition d'un modèle sur une source de données particulière nécessite simplement la mise en oeuvre d'une procédure d'encodage pour ce cas particulier. Des schémas d'inférence ont ensuite été introduits au-dessus des modèles TEMPLE, permettant leur utilisation pour les cas qui nécessitent des comparaisons d'instance à instance, telles que la vérification et la récupération, ainsi que ceux qui reposent sur l'évaluation de la similarité d'instance à échantillon, comme dans le cas de la classification prototypique.

Chapter 1

Introduction

Learning useful representations from high-dimensional structured data is one of the main goals of modern machine learning. So-called deep neural networks have shown to be able to efficiently learn such representations without requiring specialized pre-processing of the data under analysis, yielding substantial performance improvements across a range of tasks (e.g., [10]). Despite this increase in performance, deep neural networks also have several shortcomings that can limit their more widespread usage in applied domains. These limitations can include:

Lack of robustness against distribution shifts Common variations across training and testing data represent a direct violation of the i.i.d. assumption (i.e., that training and testing data are independently observed from a fixed distribution). This assumption is behind most of the supervised learning generalization guarantees built within the empirical risk minimization framework. Consider as an example the case of an object recognizer trained on natural images. In this case, it will likely observe a performance degradation if testing data consists of drawings from the same classes, for instance. Similarly, a speaker recognizer trained on audio predominantly in English will likely have its performance affected if trials containing speech in some other language are presented to the model at testing time.

Recent literature in domain adaptation/generalization has introduced more general settings, thus relaxing the i.i.d. assumption to some extent to help cope with practical situations [95, 96]. However, there is still much room for improvement, as most approaches require data from a par-

ticular target data distribution or some assumption over the test data relating it to the training domains. This requirement is unpractical given that a large number of possible unseen conditions might appear for a deployed model. In the context of this work, domain shifts will mainly occur through variations across training and testing conditions for the case of speaker verification, including different languages and/or channel conditions.

Lack of robustness against adversarially-generated attackers Current state-of-the-art deep learning-based predictors have also been shown to be vulnerable to *adversarial examples* [97, 98]. In fact, it is a known property of neural networks that it is possible to impose large variations in their outputs by only slightly changing their inputs. This is because neural networks often define non-smooth mappings and their outputs might vary sharply by moving in a small neighborhood [1] around a given point. Attackers might exploit this property to fool deployed models into making certain (erroneous) decisions. Several methods have been proposed in the recent literature and shown to fool highly performing classifiers with slight changes to the inputs that are imperceptible to humans.

Such threat can be critical in context of voice biometrics, as private data could be accessed without the necessary credentials. In fact, it has been observed that simple attack strategies, such as replaying someone's voice or synthesizing audio with the voice of a target speaker, suffices to fool an otherwise well-performing speaker recognizer [99, 100]. However, both replay and synthetic attacks can introduce artifacts in the signal that, despite being imperceptible to the human ear, can be detected by a machine. As such, approaches to detect replayed or synthesized audio have emerged as a research thread. One issue with existing spoofing detection approaches is that they are commonly optimized for one particular attack type. In practice, however, it is not known what attack type will be used, hence a attack-agnostic approach is needed. In this thesis, we address this issue via conditional modeling based on an ensemble of specialized predictors.

Data inefficiency A popular recipe for training neural networks is the combination of large architectures in terms of number of parameters and massive datasets. In fact, performance scaling laws (e.g., relating prediction accuracy with model/data size) were recently studied for the case of language modeling [101, 102], as well as more generally [103, 104]. However, large scale models

[1] In the sense of some L_p-norm.

and datasets are not practical in several real-world scenarios. As a matter of fact, collecting and, more importantly for the case of supervised settings, *labeling* data is quite often prohibitively costly or even infeasible. As such, devising data-efficient learning strategies is a stepping-stone towards practical machine learning.

For the case of speaker verification, data-efficiency is a built-in requirement given the open-endedness feature of the problem itself, i.e., trials from speakers not presented to the model during training appear at testing time. As such, devising efficient approaches able to 'zero-shot' verify unseen speakers is at the core of the verification problem. We tackle this issue by devising new architectures and training procedures allowing predictors to better generalize across scenarios.

Overly specialized learning procedures A less technical, yet relevant issue preventing the broad application of machine learning in practice is the fact that state-of-the-art learning procedures are too specialized to particular downstream applications. Engineering teams need to be able to use and maintain very different models for different tasks, even in cases where tasks are related. As an example, consider the cases of object recognition and image retrieval: while the two tasks seem somewhat related in terms of inference procedures used to solve them, state-of-the-art approaches designed to tackle those tasks vary significantly. We argue that more versatile learning procedures that perform close to state-of-the-art across a range of tasks would reduce the engineering burden related to using machine learning in the *real-world*. In order to alleviate this issue, we introduce the idea of learning templates, i.e., standardized training procedures and model components that require minimal steps in order for them to be used in a new task/dataset. As will be shown, models resulting from instances of such templates perform on par or better than state-of-the-art predictors across a number of tasks.

Given the described limitations of current machine learning technology, the research work discussed herein constitutes an attempt towards addressing these issues to some extent. We focus particularly in the context of voice biometrics and related tasks. In the remainder of this Chapter, we will introduce the specific objectives of this work and present a discussion on background topics and past work related to the approaches that will be discussed in later chapters. In particular, we formally define our problems of interest, model architectures and attention mechanisms used in the discussed approaches, the metric learning framework which will be used across different cases, and

datasets considered in the empirical evaluations. We then summarize our contributions, present the publications related to this thesis, and finally discuss the organization of this document.

1.1 Objectives

The research work described in this thesis has as its main objective closing the gap between machine learning research and practice. In particular, we focus on applications to the domain of voice biometrics and related tasks. In further detail, specific objectives are summarized in the following:

1. Design training strategies yielding discriminative utterance-level representations of audio, useful for voice biometrics and spoken language identification applications.
2. Design versatile architectures that can be re-used across different tasks within the context of voice biometrics.
3. Introduce effective approaches to detect spoofing attackers to speaker recognizers. Those should should be agnostic to the attack strategy and able to perform well under replays or synthesized attacks.
4. Define more efficient approaches to verification based on the metric learning framework. Those should rely less on costly approaches such as hard-negative triplet mining.
5. Define versatile learning procedures. That is, training strategies which are re-usable across cases.

1.2 Background and related work

In what follows, we present background content required for the following chapters. Those include notation and content that we directly use and build upon. Pointers to additional information covering topics that are mentioned in the thesis are presented in the Appendix A.

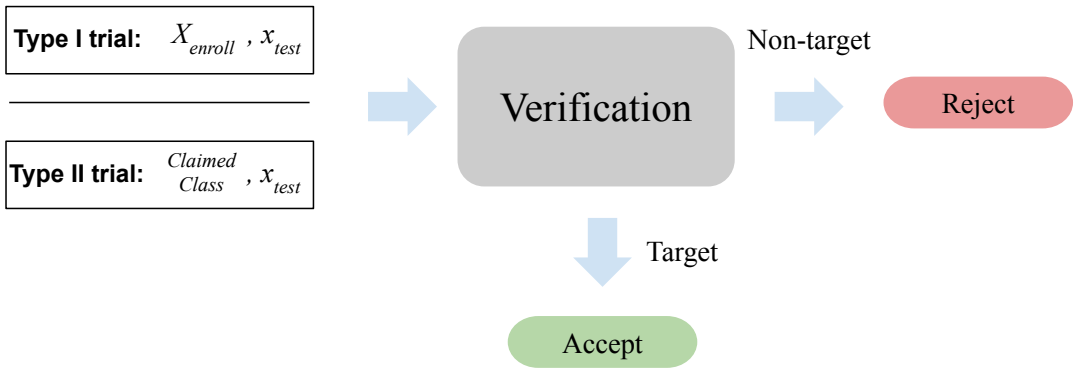


Figure 1.1 – The verification problem.

1.2.1 Problem definition

1.2.1.1 The verification problem

Given data instances $x \in \mathcal{X}$ such that each x can be associated to a class label $c \in \mathcal{C}$ through a labeling function $f : \mathcal{X} \mapsto \mathcal{C}$, we define a *trial* as a pair of sets of examples $\{X_i, X_j\}$, provided that $f(x_i^k) = f(x_i^l)\ \forall\ k, l \in \{1, 2, ..., |X_i|\}^2$ and $f(x_j^k) = f(x_j^l)\ \forall\ k, l \in \{1, 2, ..., |X_j|\}^2$, so that we can assign class labels to such sets X_m defining $f(X_m) = f(x_m)\ \forall\ x_m \in X_m$. The verification problem can be thus viewed as, given a trial $\mathcal{T}_{i,j} = \{X_i, X_j\}$, deciding whether $f(X_i) = f(X_j)$, in which case we refer to $\mathcal{T}$ as *target trial*, or $f(X_i) \neq f(X_j)$ and the trial will be called *non-target*.

The verification problem is illustrated in Figure 1.1. We categorize trials into two types in accordance to practical instances of the verification problem. Type I trials are those such that X_i is referred to as enrollment sample, i.e., a set of data points representing a given class, such as a gallery of face pictures from a given user in an access control application, while X_j will correspond to a single example x_{test} to be verified against the enrollment gallery. For the type II case, in turn, X_i is simply a *claim* corresponding to the class against which x_{test} will be verified. Classes corresponding to examples within test trials might have never been presented to the model, and sets X_i and X_j are typically small (< 10).

Under the Neyman-Pearson approach [11], verification is seen as a hypothesis test, where H_0 and H_1 correspond to the hypothesis such that $\mathcal{T}$ is target or otherwise, respectively [12]. The test is performed through the following likelihood ratio (LR):

$$LR = \frac{p(\mathcal{T}\,|H_0)}{p(\mathcal{T}\,|H_1)}, \tag{1.1}$$

where $p(\mathcal{T}|H_0)$ and $p(\mathcal{T}|H_1)$ correspond to models of target, and non-target (or impostor) trials. The decision is made by comparing LR with a threshold δ.

One can explicitly approximate LR through generative approaches [13]; Gaussian mixture models (GMM) being the most common. In this case, the denominator is usually defined as a universal background model (GMM-UBM, [14]), meaning that it is trained on data from all available classes, while the numerator is a fine-tuned model on enrollment data so that, for trial $\{X_1, X_2\}$, LR will be:

$$LR = \frac{p_{X_1}(X_2)}{p_{UBM}(X_2)} = \frac{p_{X_{Enroll}}(x_{test})}{p_{UBM}(x_{test})}. \tag{1.2}$$

Alternatively, authors in [15] showed that discriminative settings, i.e., binary classifiers trained on top of data pairs to determine whether they belong to the same class, yielded likelihood ratios useful for verification. In their case, a binary SVM was trained on pairs of i-vectors [1] for automatic speaker verification. Later on, we will build upon such discriminative setting, but with the difference that we learn an encoding process along with the discriminator (here represented as a distance model), and show it to yield likelihood ratios required for verification through contrastive estimation results. This is more general than the result in [15], which shows that there exists a generative classifier associated to each discriminator which likelihood ratio matches the discriminator's output, requiring such classifier's assumptions to hold.

We remark that current verification approaches are composed of complex pipelines containing several components [1, 79, 3], including a pretrained data encoder, followed by a downstream classifier, such as probabilistic linear discriminant analysis (PLDA) [105, 106], and score normalization [87], each contributing practical issues (e.g., cohort selection) to the overall system. This renders both training and testing of such systems difficult. Some of the approaches proposed herein are

a step towards end-to-end verification, i.e., from data to scores via a single forward pass, thus simplifying inference.

1.2.1.2 The verification problem for the case of voice biometrics

Biometrics refer to physiological or behavioral traits of a person which can be employed for automatic identification or verification of a claimed identity. Biometric systems can make use of fingerprints, hand geometry, iris, retina, face, hand vein, facial thermograms, signature, and voice, or combinations thereof to verify a user's identity [16]. Such traits have been widely used, for example, as access control to buildings or sensitive information and as an authorization mechanism in financial transactions.

In [17], a list is given of required features that must be satisfied in order for a physical/behavioral characteristic to be used as a biometric. The list includes:

- *Universality* - all individuals have the characteristic;
- *Distinctiveness* - individuals differ with respect to this characteristic;
- *Persistence* - acceptable variability over time;
- *Collectability* - measurable;
- *Performance* - of the systems relying on it;
- *Acceptability* - willingness of individuals to have this characteristic measured;
- *Circumvention* - how easily the system can be fooled using fraudulent methods.

In summary, a practical biometric system should meet the specified recognition accuracy, speed, and resource requirements, be harmless to the users, be accepted by the intended population, and be sufficiently robust to various fraudulent methods and attacks to the system [17]. As pointed out in [18], speaker verification is a popular biometric identification approach which uses the speech signal as a user's signature. The main reason for its popularity is twofold: (a) no direct contact with the individual is required; and (b) microphones are available on most portable devices. Moreover, authors argue that previous work has proven that speech properties differ relevantly across individuals and, even for twins, similarity is unlikely [29, 30].

More formally, speaker verification consists of accepting or rejecting a claimed identity by comparing two utterances, the first of these utterances being used for enrollment (produced by the

speaker with the target identity) and the second utterance is obtained from the verified speaker [19]. Under the text-independent setting, speaker verification is performed on top of unconstrained phrases of arbitrary phonetic content and length. The added phonetic variability in this scenario represents an extra adverse factor when compared to the session and speaker variabilities, present in the text-dependent case [20].

Classical approaches for automatic speaker verification (ASV) divide the problem into two distinct phases: (i) compute low-dimensional speaker representations; and (ii) perform binary classification on top of pre-computed representations of enrollment and test utterances. So-called i-vectors [1] are known to be the state-of-the-art representations for ASV. Generally, i-vectors are obtained by firstly training what is usually referred to as universal background model, consisting of a Gaussian mixture model (GMM-UBM) (typically in the form of a Gaussian mixture model, GMM-UBM) [14] trained with the expectation-maximization algorithm. After doing so, statistics of latent variables are concatenated to form supervectors, which, when subjected to factor analysis reduce in dimensionality and result in i-vectors. Factor analysis aims at obtaining a low-dimensional representation which embeds both channel- and speaker-dependent information. The described algorithm is illustrated in Figure 1.3. Inferences can then be made via either discrimninative approaches using, for example, support vector machines and logistic regression, or generative approaches, such as PLDA [106]. One of the shortcomings of the i-vector+PLDA approach is its lack of robustness to short-duration recordings [47, 107]. Recent approaches [3] substituted i-vectors by the output of some inner layer of a 1-dimensional convolutional model commonly referred to as time-delay neural network (TDNN) [108], trained under the multi-class classification setting to recognize a training set of speakers. Representations obtained from TDNNs training as speaker recognizers are often referred to as x-vectors.

Several subsequent proposals re-used the idea of performing speaker recognition so as to enforce speaker dependency in inner layers, varying the types of architectures and training settings. This is the case of [109], for instance, which compared different architectures based on the Visual Geometry Group (VGG) model [74], first introduced for objects recognition in images, as well as combinations with intra-loss [110], which minimizes the variance of the embedding of each speaker within a training minibatch. Moreover, in [111], the authors evaluate a very deep neural network under the described speaker recognition training setup, and combined the cross-entropy with the center-loss [25]. This approach accumulates first order statistics of the average embeddings of each speaker within the

training data throughout training, aiming at enforcing a low variability in the representations of a given speaker.

Another direction was adopted in proposals based on the metric learning framework. Such approaches aim at explicitly learning representations for which class-discriminability is ensured. The metric learning approach was first explored in large scale for face verification applications [62]. However, training difficulties involved in this setting are commonly reported [71] and several stabilizing strategies have been proposed, such as hard triplets mining, pre-training, and combination of triplets loss with center- or intra-losses. A large-scale speaker verification setting in which triplet loss is employed along with mining strategies can be found in [112]. Finally, variations of the maximum likelihood estimation training strategy, where some geometric structure is imposed over learned representations, was proposed. A notable example of such a case is the additive margin softmax approach [113].

1.2.1.3 Language identification

The language identification (LID) task consists of identifying spoken languages from speech data of unconstrained phonetic content, which is highly useful across several applications within speech processing. For instance, language recognizers can be employed to approximate conditional likelihoods. Alternatively, they can also be used for hierarchical modelling, where a language recognizer is placed in early stages of a pipeline which is then followed by language-specific sub-modules, as illustrated in Figure 1.2. Speech recognition or speaker verification are examples in which prior information about the spoken language can boost performance. Moreover, direct practical applications of language recognizers can include directing calls in call-centers, as an example.

As pointed out in [21], classical approaches for language identification commonly rely on methods originally introduced for speaker recognition. Applications of i-vectors in language recognition, for instance, can be found in [21] and [22].

Even though i-vectors are known to yield satisfactory performance on both speaker and language modeling across several datasets, especially in contexts within which the amount of training data is relevantly constrained, it is also known that its performance degrades when faced with short-duration test conditions. Moreover, its unsupervised nature might become an issue in cer-

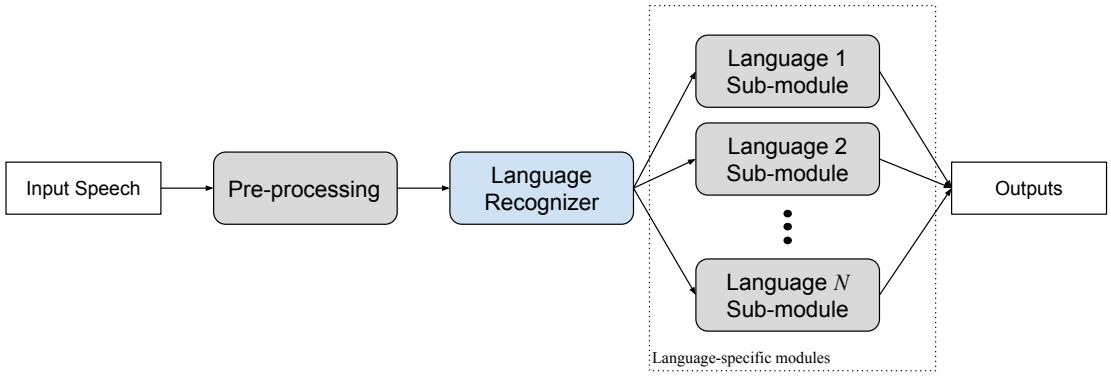


Figure 1.2 – Hierarchical pipeline with a language recognizer streaming inputs into language-specific sub-modules.

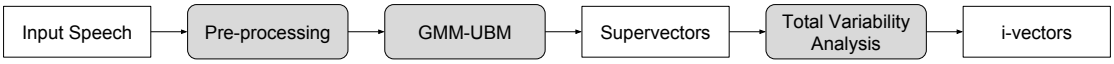


Figure 1.3 – i-vectors front-end description [1].

tain scenarios, when available labeled information may yield more discriminable low-dimensional representations.

Supervised approaches have been introduced in conjunction with neural networks of various families for language identification. For instance, the approaches in [23] and [24] employ residual neural architectures and different strategies for training, such as center-loss minimization [25] and angular softmax [26], both inspired from applications on face recognition. However, evaluation has been performed only on longer duration recordings, with relevant performance degradation observed when moving from 30-second recordings to 3-second ones. Similarly, other architectures were employed for time-dependency modelling for language identification, including time delay neural networks [27] and long short-term memory (LSTM) networks [28]. These have shown to outperform i-vectors in short-duration scenarios, while not performing as well in the case of long-duration signals.

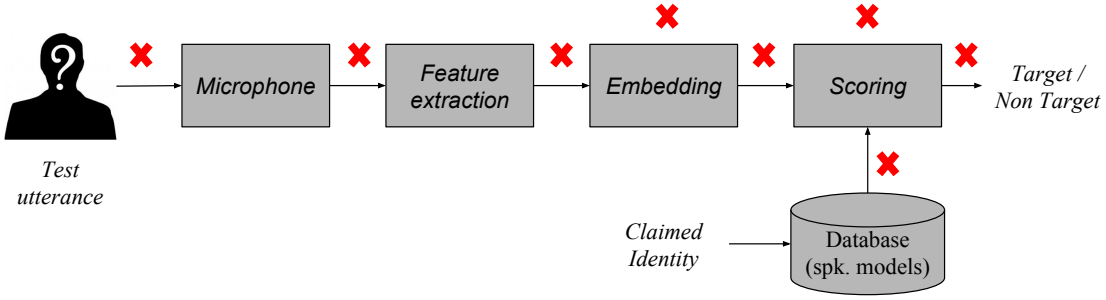


Figure 1.4 – Illustration of a generic Automatic Speaker Verification (ASV) system. Red crosses indicate parts of the system which can be potentially exploited by an attacker to bypass or fool the verification process. We focus on microphone-level attacks, i.e., the attack is performed on the input signal prior to being captured by the system. Figure is inspired by [2]-Fig. 1.

1.2.1.4 Spoofing detection

As discussed above, voice is a popular biometric for identification approaches [18, 29, 30]. Nonetheless, voice biometrics are not free from threats and attack strategies targeting speaker recognition systems have been described [31]. As such, the design of counter-measures to yield more robust applications has become a popular research direction.

As discussed in [2] and illustrated in Figure 1.4, attacks can be designed at various stages of a biometrics-based authentication pipeline. Notably, recently-introduced adversarial attacks targeting artificial neural networks [32] can act at the model-level. In the specific case of speaker verification and voice biometric systems, other attack strategies also exist. These are termed "spoofing attacks" and represent a person or a computer program that tries to overcome an authentication system by forging the data of a legitimate user.

In this work, we focus on microphone-level model agnostic attackers. Microphone-level attacks can be realized with different methods [31, 33, 9]: (i) impersonation of another user, (ii) synthetic speech, and (iii) pre-recorded (replayed) audio from a given user. Herein, we will consider (ii) and (iii), which will be referred to as *logical access* (LA) and *physical access* (PA) attacks, respectively. The latest LA attacks have taken advantage of recent advances in speech synthesis and voice conversion based on auto-regressive waveform modeling or generative adversarial networks [34, 35, 36, 37].

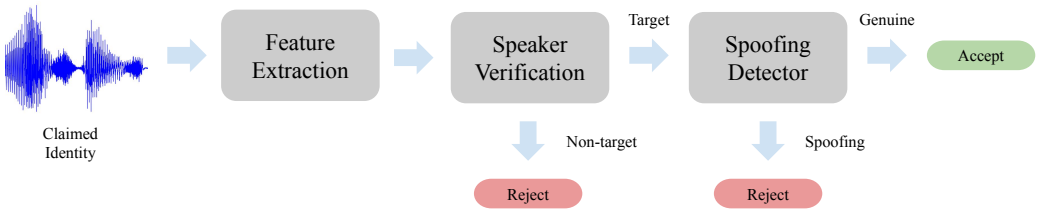


Figure 1.5 – Scheme 1: The spoofing detector is invoked once inputs are classified as target with respect to claimed identities.

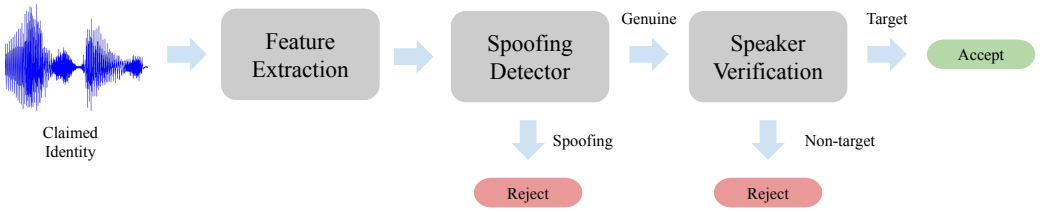


Figure 1.6 – Scheme 2: The spoofing detector is used first and only samples classified as genuine are passed through to the speaker verification block.

Given the serious consequences that spoofing attacks can have on speaker verification systems, recent research has focused on the development of new attack detection algorithms and several challenges have been organized (e.g., [9, 33, 38, 39]). Figures 1.5 and 1.6 present block diagrams of two possible settings, where spoofing detectors are used in tandem with speaker verification systems. In both the cases, the input corresponds to an audio signal along with a claimed identity. The spoofing detector can be used if the claimed identity is verified as target, i.e., true and claimed identities match, or the opposite can be done so that only samples classified as genuine by the spoofing detector will be verified against the claimed identity.

In terms of countermeasures aimed at detecting spoofing attackers, a generative classifier was introduced in [99] following a similar approach to that of linear discriminant analysis (LDA), such that one generative model is trained per class. However, in this case, GMMs are employed for modelling the class-conditional features rather than simple Gaussians with shared covariances as in the LDA case. Given a sequence of feature vectors denoted by O corresponding to a speech signal, the genuine *vs.* spoofed speech decision is thus made based on the following likelihood ratio $l(O)$:

$$l(O) = \log p(O|\theta_g) - \log p(O|\theta_s), \tag{1.3}$$

where θ_g and θ_s correspond to the parameters of independently trained frame-level GMMs on only genuine or spoofing data, respectively, while $\log p(O|\theta_g)$ and $\log p(O|\theta_s)$ are the average log-likelihood across all frames in O for genuine and spoofing data, respectively.

A common spoofing attack detection method has been to find speech representations or features that emphasize the artifacts introduced by spoofing methods, and employ such representations along with GMM-based classifiers to detect them. Various low-level speech representations have been investigated for this purpose, including: (i) spectral amplitude and phase [114, 115, 116, 117, 118, 119], and (ii) combined amplitude-phase [115, 116, 117]. More specifically, the authors in [114] employ a GMM classifier on top of cepstral coefficients computed after a constant Q transform, called constant Q cepstral coefficients (CQCC). In [115], features tailored for anti-spoofing based on the infinite impulse response constant Q-transform spectrum (IIR-CQT) were introduced, such that cepstral coefficients were extracted by filtering the speech signal spectrum with an infinite impulse response and decorrelating using either a discrete cosine transform or principal component analysis. Cochlear filter cepstral coefficients and excitation source-based features are evaluated in [118] and [119], respectively. GMM-based detectors are evaluated in [120] for a setting where LA and PA training data are pooled together while the detector is evaluated on top of each data partition. However, as will be further discussed, doing so is not enough to recover the performance of specialized models in each type of attack.

Another strategy employed in recent work is usually referred to as tandem representations. In such case, a frame-level neural network is trained for spoofing detection, and finally a GMM classifier is trained on top of inner layers outputs, predicted posteriors, and speech features [115, 116]. A similar approach to that of tandem features is proposed in [121]. In that case, a neural network is trained in a supervised manner at the frame-level. Statistics of intermediate representations are used as global descriptors of full recordings, on top of which scoring can be performed. The supervised training is carried out so as to determine whether individual frames correspond to an attack or not. Moreover, independently trained fully-connected neural networks on top of different speech features have been explored in [122] and [123]. Models receive concatenated frames as inputs and classify a full recording as genuine or spoofing. A score-level fusion of the set of detectors is then employed for final decision.

Finally, more recent methods have explored end-to-end spoofing detection methods where a single model is trained in one step, and the model is able to directly output scores given unseen test examples. In [40], a convolutional model is trained on top of raw audio for the detection of replayed attackers. In this setting, the model is able to learn representations which will detect replay attacks. In [41], an attentive end-to-end scheme is introduced such that a U-net structure is first employed to map the input features into a set of element-wise importance weights. The weighted input is then fed into a stack of feed-forward blocks with residual connections yielding final scores. Convolutional neural networks were also shown to be effective for direct end-to-end detection in [42].

1.2.2 Model architectures

1.2.2.1 Time delay neural networks (TDNN)

Approaches presented in the following chapters either use or improve upon the TDNN architecture, introduced in the context of automatic speaker verification by the x-vector framework [3]. The model is illustrated in Figure 1.7 and further detailed in Table 1.1. More specifically, an input sequence of length T is denoted as $x_{[1:T]}$, where each $x_i \in \mathbb{R}^d, i \in [T]$, represents a feature vector of dimension d (e.g., mel frequency cepstral coefficients) at a given time frame. Equivalently, we denote the set of features output by the stack of convolution layers by $y_{1:T} \in \mathbb{R}^D, i \in [T]$, where D corresponds to the number of output channels of the last convolutional layer. We refer to those as local descriptors of the overall audio given that they correspond to features of a relatively short time window. D is set to 1500 in the original model and to 512 in some of our cases since, as will be discussed, our setting might require local descriptors of matching dimensionality across the model. The temporal pooling layer, also referred to as *statistical pooling*, concatenates element-wise estimates of first- and second-order statistics of the set of local descriptors across the temporal axis (the symbol " ˆ " in Fig. 1.7 indicates the concatenation operation). We thus define the global descriptor V, i.e., the feature vector summarizing the entire input sequence $x_{[1:T]}$, by the following:

$$V = \text{cat}[\mu(y_i), \sigma(y_i)], \tag{1.4}$$

where the operator $v = \text{cat}[v_1, v_2]$ concatenates $v_1, v_2 \in \mathbb{R}^D$ such that $v \in \mathbb{R}^{2D}$, and y_i are obtained after the last convolution layer. The global descriptor V is finally fed into a sequence of dense layers

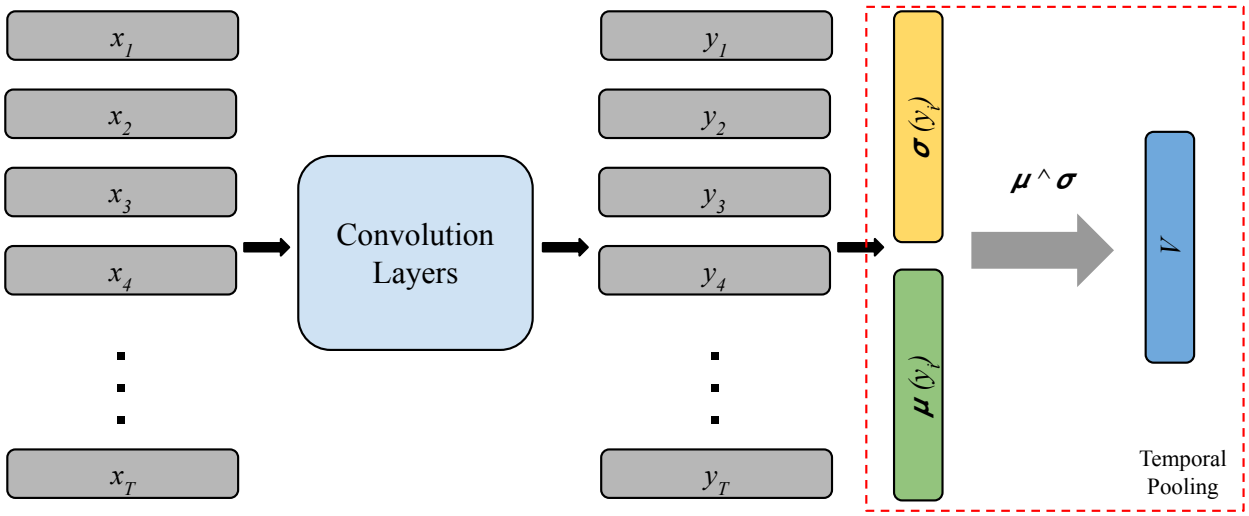


Figure 1.7 – Conventional TDNN overview [3].

Table 1.1 – Standard TDNN architecture. T indicates the duration of features in number of frames and d the feature vector dimensionality. Batch normalization is further employed after each layer except temporal pooling.

Layer	***Input Dimension***	***Output dimension***
Conv1d+ReLU	d × T	512 × T
Conv1d+ReLU	512 × T	512 × T
Conv1d+ReLU	512 × T	512 × T
Conv1d+ReLU	512 × T	512 × T
Conv1d+ReLU	512 × T	1500 × T
Temporal Pooling	1500 × T	3000
Linear+ReLU	3000	512
Linear+ReLU	512	512
Linear+ReLU	512	# classes

to yield the outputs corresponding to log-probabilities over the set of classes under consideration (e.g., training speakers or languages).

Past work has considered several modifications of the original TDNN architecture in order to improve performance in tasks such as speaker verification. In [124], for instance, several practical considerations were evaluated, such as the differences in performance given by applying batch normalization before or after activation functions were applied. Model variations focusing specifically in the pooling strategy were proposed in [125], where a learnable gating mechanism was employed in order to assign more or less importance to specific frames prior to pooling. Similarly, a linear layer was used in [126] to learn how important individual frames are. These approaches, however,

are limited in that each frame is evaluated by itself, and a better informed pooling strategy should leverage the sequence structure in order to decide which frames matter more or less.

Alternatively, approaches such as the model discussed in [127] consider higher order statistics during pooling, but similarly to the base case, only representations from a given layer are used to compute the overall utterance-level representation used by top layers of the model. Closer to our work is the approach discussed in [128] where global statistics of different layers are concatenated prior to the TDNN dense layers close to the outputs. While such direction is an improvement compared to other approaches, given that bottom layers are considered as well, we hypothesize that simply concatenating low-level representations is sub-optimal in that the sequential nature of such set of feature vectors is not leveraged. We argue that some type of sequence processing mechanism can enrich the final pooled vector and yield lower-dimensional pooled features, thus saving on parameters and computations in higher layers. While any sequence model could be used (e.g., RNNs), we focus on self-attention given its efficiency and ease of training.

1.2.2.2 Residual architectures

Firstly introduced in [4], ResNets constitute a set of architectures made up of a series of so-called residual blocks, which determine how a feature transformation should differ from the identity mapping[2], rather than how it should differ from zero [43]. Residual block transformations present a basic form that, for a generic input variable $X \in \mathcal{X}$, is given by:

$$X' = F(X) + X. \tag{1.5}$$

The residual term comes from the fact that the input is directly used to compute the transformation's output, which in a neural network represents a direct path for gradients to "flow" during backpropagation of gradients for computation of SGD updates, as illustrated in Figure 1.8. $F(X)$ is generally a set of convolutional layers, followed by nonlinear activation functions and normalization layers. Variations of ResNets have been proposed and used in different contexts, such as person re-identification, object detection, and segmentation, as is the case of MobileNet [44].

[2]The identity mapping outputs its inputs.

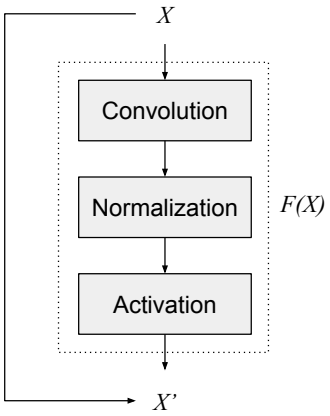


Figure 1.8 – Residual block [4].

Recent literature has shown that residual blocks contribute to yielding loss landscapes which are easier to train, in the sense that ill-conditioned chaotic landscape regions become less frequent when such an architectural feature is employed [45]. Moreover, near identity transformations were studied in depth and guarantees were introduced for the linear and nonlinear $F(x)$ cases in [46] and [43], respectively. In some of the chapters that follow, we will use ResNets as a backbone network on top of which training procedures are devised so as to yield discriminative speaker- or language dependent utterance-level representations.

1.2.2.3 Attention mechanisms

Frame-wise attention Several attention mechanisms have been introduced recently in architectures aimed at modeling global properties of temporal data, as in the cases of speaker or language recognition. In general terms, frame-wise attention blocks learn to conditionally weigh time-steps given input representations on some inner layer of a model [47]. Consider $y_{1:T}$ as a set of vectors corresponding to the outputs of a given neural network for some input $x_{1:T}$ – e.g., a sequence of acoustic features, such as mel filterbanks. A linear transformation A is shared across all time-steps t, and applied to each y_t resulting in a set of scalars $a_{1:T}$, according to:

$$a_t = \tanh\left(Ay_t\right). \tag{1.6}$$

A set of normalized weights summing up to 1 is obtained through the softmax operator:

$$w_t = \frac{e^{a_t}}{\sum_{t=1}^{T} e^{a_t}}, \tag{1.7}$$

and the attention layer output is finally given by:

$$V = \sum_{t=1}^{T} w_t y_t, \tag{1.8}$$

where V will correspond to a global utterance-level representation of the entire input sequence $x_{[1:t]}$, which can be further processed by fully-connected layers to yield, for instance, conditional probabilities over a set of training speakers. An alternative approach to computing V would be to perform statistical pooling operations described previously but over the set of weighted representations $y'_t = w_t y_t$.

Scaled dot product attention Introduced in [48] and also referred to as *self-attention*, such component corresponds to an alternative to recurrent models for modeling sequential data and was introduced along with the Transformer architecture. Considering the same set of representations discussed above and represented by $y_{[1:t]}$, it operates as indicated by the following:

$$\text{self-attention}(Q, K, V') = \text{softmax}(\frac{QK^{\intercal}}{\sqrt{d_k}})V', \tag{1.9}$$

where Q, K, and V', denominated *queries*, *keys*, and *values*, respectively, each correspond to a linear transformation of $y_{[1:t]}$ (assumed to be a matrix of dimension $t \times D$), assuming each $y_t \in \mathbb{R}^D$ i.e.,:

$$Q = y_{[1:t]}W^Q, \quad K = y_{[1:t]}W^K, \quad V' = y_{[1:t]}W^V. \tag{1.10}$$

In this case, each of the matrices W^Q, W^K, and W^V, have dimension $D \times d_k$ and their entries are treated as learnable parameters. Intuitively, the scaled dot-products $\frac{QK^T}{\sqrt{d_k}}$ define weights indicating the importance of each element in the sequence given a specific data instance.

Additionally, a multi-head setting is often employed such that the self-attention operation described above is applied multiple times using independent sets of parameters. The final sequence

is finally given by a linear projection of the concatenation of the outputs of each head to the space of dimension d_k. The number of self-attention heads as well as d_k are usually treated as a hyperparameters.

1.2.3 Metric Learning

Being able to efficiently assess similarity across samples from data under analysis is a long standing problem within machine learning. Algorithms such as K-means, nearest-neighbors classifiers, and kernel methods generally rely on the selection of some similarity or distance measure able to encode semantic relationships present in high-dimensional data into real scores. Under this view, approaches commonly referred to as *Distance Metric Learning*, introduced originally in [49], try to learn a so-called Mahalanobis distance, which, given $x, x' \in \mathbb{R}^D$, will have the form: $\sqrt{(x-x')^\intercal M(x-x')}$, where $M \in \mathbb{R}^{D\times D}$ is positive semidefinite. Several extensions of that setting were then introduced [50, 51, 52].

In [53], for instance, an online version of the algorithm in [49] is proposed, while an approach based on support vector machines (SVM) was introduced in [54] for learning M. In [55] an information-theoretic approach is provided to solve for M by minimizing the divergence between Gaussian distributions associated with the learned and the Euclidean distances, further showing such an approach to be equivalent to low-rank kernel learning [56]. Similar distances have also been used in other settings, such as similarity scoring for contrastive learning [57, 58]. Besides the Mahalanobis distance, other forms of distance/similarity have been considered in recent work. In [59], for example, a kernel matrix is directly learned, implicitly defining a similarity function. In [60], classes of neural networks are proposed to define pseudo-distances which satisfy the triangle inequality while not being necessarily symmetric.

For the particular case of Mahalanobis distance metric learning, one can show that $\exists\, W$: $\sqrt{(x-x')^\intercal M(x-x')} = ||Wx - Wx'||_2$ [53], which means that there exists a linear projection of the data after which the Euclidean distance will correspond to the Mahalanobis distance on the original space. In [61], the linear projection is substituted by a learned non-linear encoder $\mathcal{E} : R^D \mapsto R^d$ so that $||\mathcal{E}(x) - \mathcal{E}(x')||_2$ yields a (non-Mahalanobis) distance measure between raw data points yielding useful properties. Follow-up work has extended such an idea to several applications [62, 63, 64, 65]. One extra variation of $||Wx - Wx'||_2$, besides the introduction of $\mathcal{E}$, is to switch the Euclidean

distance $||\cdot||_2$ with an alternative better suited for the task of interest. That is the case in [66], where the Hamming distance is used over data encoded to a binary space. In [67], in turn, the encoder is trained so that Euclidean distances in the encoded space approximate Wasserstein divergences, while in [68] a hyperbolic distance is employed which is argued to be suitable for their particular use case.

Based on the covered literature, one can conclude that there are two different directions aimed at achieving a similar goal: *learn to represent the data in a metric space where distances yield efficient inference mechanisms for various tasks.* While one direction corresponds to learning a meaningful distance or similarity from raw data, the other corresponds to, given a fixed distance metric, finding an encoding process yielding such desirable metric space. Under the latter setting, approaches such as Siamese Networks [5] have been introduced with the aim at explicitly learning an embedding model parameterized by a neural network which results in a lower-dimensional space, where relevant properties hold, such as class separability. It is often assumed that information regarding which data samples should be close together and far apart in the embedding space is available.

Training is performed such that examples are presented in pairs in which case x_a is referred to as anchor and its pair can be either positive or negative, represented as x_+ or x_-, indicating whether the pair belongs to the same or different classes, respectively. Such setting is illustrated in Figure 1.9 for a model with parameters W.

The loss $\mathcal{L}_\mathcal{S}$ will be thus given by the following, for a set of n pairs of examples, and a transformation $y = \mathcal{E}(x)\mathbb{R}^D \mapsto \mathbb{R}^d$ defined by the neural network from data to embedding space:

$$\mathcal{L}_\mathcal{S} = \sum_{i=1}^{n} Z_i ||\mathcal{E}(x_1^i) - \mathcal{E}(x_2^i)||_p, \tag{1.11}$$

where Z_i will be 1 if x_1^i and x_2^i belong to the same class and -1 otherwise, and the operator $||\cdot||_p$ defines a metric on the embedding space.

Follow-up work has exploited and extended the scheme introduced with siamese networks for several applications. Triplet networks represent an alternative to C in which, during training, triplets of examples are drawn from the dataset such that an anchor sample x_a is selected along with corresponding positive and negative examples x_+ and x_-, respectively. The embedding model

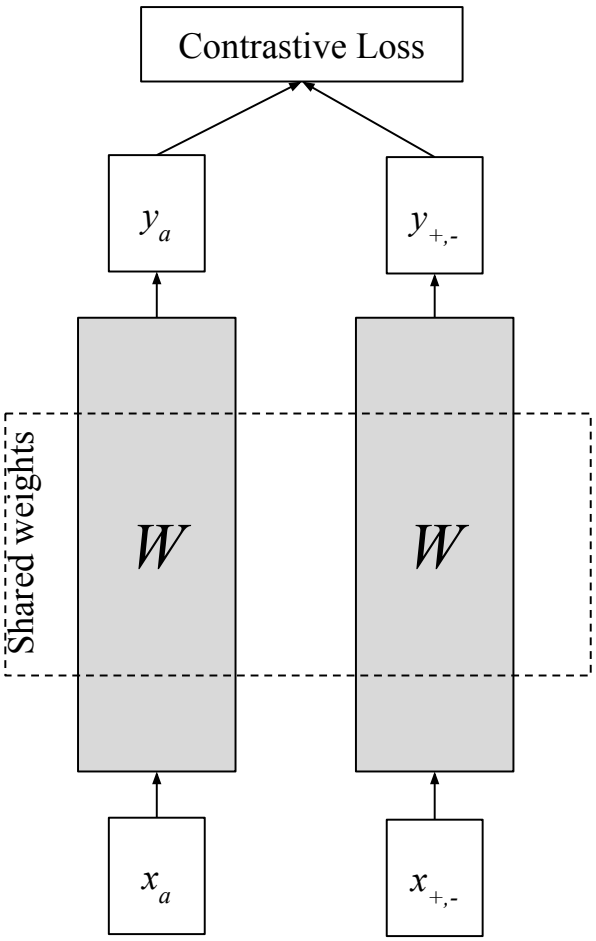


Figure 1.9 – Illustration of a siamese network setting. Model weights are shared across the two branches, and pairs can be either positive or negative [5].

is thus trained to minimize or maximize some L_p norm of the difference between samples in the embedding space depending on whether they should be close or distant, given the available prior information. Classifiers trained on embeddings obtained from trained triplet networks were shown to achieve high performance on object recognition tasks [6]. An illustration of triplet-networks is shown in Figure 1.10.

The triplet loss $\mathcal{L}_\mathcal{T}$ for a given set of triplets of size n is given by:

$$\mathcal{L}_\mathcal{T} = \sum_{i=1}^{n} \max(||\mathcal{E}(x_a^i) - F(x_+^i)||_p - ||\mathcal{E}(x_a^i) - F(x_-^i)||_p + \alpha, 0), \tag{1.12}$$

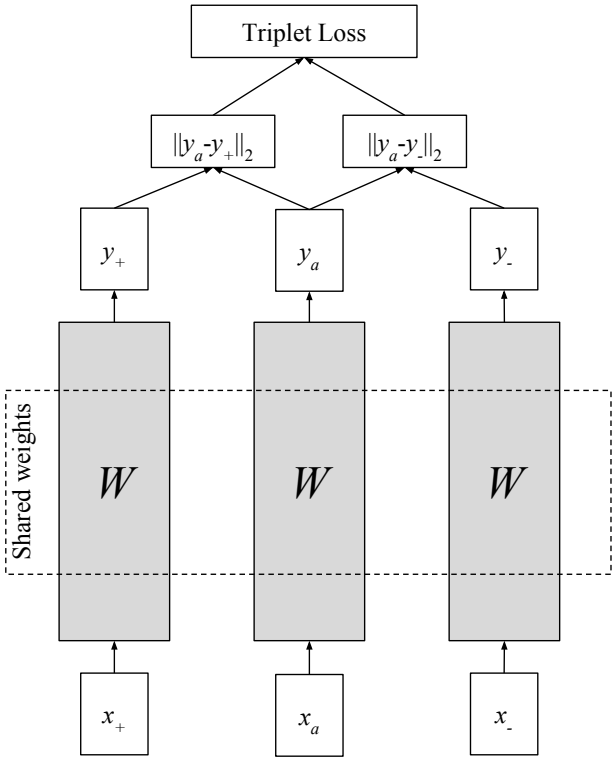


Figure 1.10 – Illustration of a triplet network setting. Model weights are shared across the three branches [6].

where α is a user-defined hyperparameter, and the operator $\max(\cdot, 0)$ is intended to avoid solutions which achieve low loss by simply maximizing the term $||\mathcal{E}(x_a^i) - \mathcal{E}(x_-^i)||_p$, avoiding minimizing the term corresponding to the positive pair.

Several applications of triplet networks were proposed in recent years. The problem of finding correspondences between images via local descriptors was tackled in [129], where authors trained an embedding model using a variation of triplet loss in which anchor and positive pairs are swapped in case the distance between anchor and negative samples in the embedded space is greater than the distance between positive and negative samples. Furthermore, such representation learning approaches have been employed in biometrics applications, such as Google's FaceNet face recognizer [62] and Baidu's Deep Speaker recognizer [112]. In both FaceNet and Deep Speaker, triplets selection, and in particular, finding close anchor and negative pairs, showed to have a major impact on training. As such, having strategies to mine such pairs is crucial to achieve low test error rates.

In this thesis, we will introduce an alternative where both the encoder and distance are learned jointly. Close to such an approach is the method discussed in [85] where, similarly to our setting,

Table 1.2 – Numbers of speakers and utterances for each partition of the VoxCeleb corpus [7]. Our models are trained on the training partition of VoxCeleb2 and evaluated on the three test partitions described below.

Partition	Data Description	# Speakers	# Utt.	# Trials (target)
Train	VoxCeleb2	5994	1092009	–
Test	VoxCeleb1 (test)	40	4715	37720 (18860)
	VoxCeleb1 (ext)	1251	145375	581480 (290743)
	VoxCeleb1 (hard)	1190	138137	552536 (276270)

both encoder and distance are trained, with the main differences lying in the facts that our method is fully end-to-end[3] while in their case training happens separately. Moreover, training of the distance model in that case is done by imitation learning of cosine similarities.

1.2.4 Datasets

1.2.4.1 Speaker verification

The VoxCeleb corpus, more specifically its second release [7], is employed for evaluation under the speaker verification setting. The corpus is comprised of audio collected from YouTube videos corresponding to interviews under unconstrained acoustic environments. Evaluation is carried out under the open-set verification condition where test data correspond to speakers unseen during training. Moreover, while the bulk of the data is represented by speech in English, speakers of different nationalities are present in the data and, as such, varying languages appear within the recordings. Three evaluations are made available, namely: the test set of the first release of the corpus, the full training set of the first release (unseen during training) referred to as *extended*, and a *hard* set of trials where available meta-data was used to create trials likely to be difficult to discriminate. While the training data is comprised of audio from 5994 speakers, each of the described test partitions are represented by 40, 1251, and 1190 speakers, respectively, all disjoint from the set of speakers present in the training data. The *hard* partition contains trials from a subset of the same 1251 speakers as in the *extended* test data. A summary of data statistics is reported in Table 1.2.

[3] What authors refer to as end-to-end requires pretraining an encoder in the metric learning setting with a standard distance.

In addition to VoxCeleb, we further consider the evaluation case introduced for the 2016 and 2018 editions of the NIST Speaker Recognition Evaluation (NIST-SRE), corresponding to a cross-language setting. The NIST SRE 2016 evaluation focuses on telephone speech recorded over different types of handsets and in different languages. For example, training data was available in English, whereas test and enrollment partitions correspond to languages such as Mandarin, Cantonese, Cebuano, and Tagalog. Moreover, duration variability in the test data is also introduced to make the task more challenging, as is the addition of unlabeled in-domain training data, and imbalanced single- and multi-enrollment trials [130, 131]. With the 2018 edition, on the other hand, training data is spoken predominantly in English while the evaluation data (both enrollment and test data) is in Tunisian Arabic. Besides the language mismatch, variations due to different codecs are further observed (PSTN versus PSTN and VOIP). This evaluation further introduces an additional task corresponding to performing speaker verification on audio from video, under a multi-speaker test scenario [132].

1.2.4.2 Language identification

For the language identification task, we consider the data and evaluation conditions introduced for the AP18-OLR Challenge [8]. The data corresponds to audio from ten languages, and the following evaluation conditions were defined:

1. *Short-duration*: Considers only test recordings with duration lower than 1 second.
2. *Confusing languages*: Test trials correspond to pairs of languages known to be to difficult to distinguish, i.e., Cantonese, Korean, and Mandarin.
3. *Unseen languages*: Test recordings correspond to languages not observed within the training sample.

A total of 214560, 22071, and 404160 test trials (i.e., a pair *claimed language/test recording*) were made available for each of the evaluation conditions discussed above. Class-wise statistics of train and evaluation data are reported in Table 1.3.

Table 1.3 – Language identification dataset statistics [8].

Language	Train		Evaluation	
	# Speakers	Utt./Speaker	# Speakers	Utt./Speaker
Cantonese	24	320	6	300
Mandarim	24	300	6	300
Indonesian	24	320	6	300
Japanese	24	320	6	300
Russian	24	300	6	300
Korean	24	300	6	300
Vietnamese	24	300	6	300
Kazakh	86	50	86	20
Tibetan	34	330	34	50
Uyghur	353	20	353	5

Table 1.4 – Number of genuine and spoofing recordings in training, development, and evaluation partitions for logical and physical access attacks [9].

	# Speakers	# Recordings			
		Logical Access		Physical Access	
		Bona fide	Spoof	Bona fide	Spoof
Train	20	2580	22800	5400	48600
Development	20	2548	22296	5400	24300
Evaluation	67	7355	63882	18090	116640

1.2.4.3 Detecting spoofing attacks

For the case of spoofing detection, the evaluation setting introduced for the ASVspoof 2019 challenge [9] is considered. In particular, two independent sub-tasks relative to the detection of two different classes of attacks are used for evaluation:

1. *Logical access attacks*: Consisting of synthetic speech created using both voice conversion and text-to-speech systems.
2. *Physical access attacks*: Consisting of simulated replays of genuine audio clips exhaustively considering varied acoustic conditions, such as three different room sizes, three distances to the microphone, and three levels of reverberation.

The interested reader is referred to [9] for more details about the dataset and attacks creation. In turn, statistics regarding the amount of data in each class/condition can be found in Table 1.4. We further highlight that a disjoint set of speakers is used to generate different data partitions and the algorithms make no use of speaker identity information.

1.2.4.4 Standard image benchmarks

In Chapters 5 and 6, in addition to evaluations on voice biometrics tasks, we further employ standard image-based benchmarks since, in those cases, our proposals have a broader scope and thus require an evaluation on tasks other than those that compose the focus of this thesis. Such benchmarks are listed below:

1. MNIST [88]
2. CIFAR-10 [89]
3. ImageNet [94]
4. *Mini*ImageNet [93]
5. CARS196 [91]
6. CUB200-2011 [92]
7. PACS [90]

1.3 Summary of contributions

The contributions of the research work reported in this thesis can be summarized as follows:

1. A multi-task training scheme is proposed with the goal of enforcing speaker- or language-dependency into learned utterance-level representations. This is mainly achieved by a combination of speaker/language recognition training under a maximum likelihood setting combined with triplet loss minimization. Additional techniques are further introduced in order to enable effective training, such as a particular scheme to sample training examples and construct mini-batches, and a maximum entropy regularizer, used to enforce embeddings not to concentrate.
2. As a countermeasure to attacks targeting voice biometrics systems, we propose an end-to-end setting for independent detection of either logical or physical presentation attackers. By end-to-end, we mean that the proposed models are able to output a score indicating how likely it is that its input is an attack when inputs correspond to general-purpose time-frequency representations of speech data. Modified light convolutional neural networks (LCNN) [73], originally proposed for face recognition, are employed in our case along with an attention

layer, aimed at enabling the model to process varying length recordings while focusing on specific portions of inputs. We further discuss data augmentation strategies able to render the training data more diverse while conserving artifacts that give away attackers, which allows moving to larger architectures such as TDNNs.

3. We extend end-to-end spoofing detection approaches previously introduced and propose attack agnostic detection systems effective in detecting both LA and PA spoofing attacks. We do so via ensembling specialists so as to detect both replays and synthetic attackers using a single system. We additionally carry out empirical evaluations to determine which speech representation is more suitable for performing detection under each type of attack. We further evaluate which speech representation is more suitable for detecting which type of attack was presented to the model.
4. A TDNN variation able to account for global representations at various levels of the convolutional stack is introduced, referred to as ML-TDNN. Such architecture uses self-attention as a mechanism for combining global statistics obtained from different layers. The proposed approach allows for the aggregation component to be incorporated by any model that keeps a fixed dimensionality throughout depth, and also allows for simpler aggregation mechanisms, such as direct averaging or selecting a specific layer, to be reachable by the training algorithm. Moreover, such layers are *efficient* since short sequences are formed by combining representations across model depth. In doing so, we end up with a more versatile model in the sense that a fixed architecture can now be used across tasks, which further results in more discriminative features, indicating that complementary information can be efficiently extracted from different layers of a model. We report evidence showing that model architectures that directly rely on global statistics extracted from early layers perform better than those that only account for features of a given top layer. We found this to be the case across a number of tasks.
5. We propose an augmented metric learning framework where an encoder and a (pseudo) distance are trained jointly and define a (pseudo) metric space where inference can be done efficiently for verification. We show that the optimal distance model for any fixed encoder yields the likelihood-ratio for a Neyman-Pearson hypothesis test, and it further induces a high Jensen-Shannon divergence between the joint distributions of positive and negative pairs. The introduced setting is trained in an end-to-end manner, and inference can be performed with a single forward pass, greatly simplifying current verification pipelines which

involve several sub-components. Evaluation on large scale verification tasks provides empirical evidence of the effectiveness in directly using outputs of the learned pseudo-distance for inference, outperforming commonly used downstream classifiers.

6. As a final contribution, we build upon our proposals to introduce a set of model components accompanied by a training procedure that can be re-used for different tasks and types of data. Given that tasks other than verification are now supported, we enlarge our scope to tasks such as multi-class classification and retrieval, and evaluate models on standard image benchmarks. Under such setting, defining a model on a particular data source requires simply implementing an encoding procedure for that particular case. Empirical evidence is provided to support the claim that models defined by the proposed framework perform on par with task-specific specialized models. In addition to that, we observed object recognizers defined under this setting to result in improved robustness against adversarial attackers and covariate shift between training and testing data distributions. The proposed approach further supports the inclusion of new classes appearing posterior to training, which we do by simply including new prototypes obtained from small samples. Doing so yields a simple, yet competitive mechanism for few-shot classification.

1.4 Publications

Publications included in the thesis

Articles published in refereed journals

J. Monteiro, J. Alam, T. Falk, "Generalized End-to-End Detection of Spoofing Attacks to Automatic Speaker Recognizers", Computer Speech and Language.

J. Monteiro, J. Alam, T. Falk, "Residual convolutional neural network with attentive feature pooling for end-to-end language identification from short-duration speech.", Computer Speech and Language.

Under review

J. Monteiro, J. Alam, T. Falk, "Multi-level Self-attentive TDNN: A General and Efficient Approach to Summarize Speech Into Discriminative Utterance-level Representations", Speech Communication.

J. Monteiro, I. Albuquerque, J. Alam, T. Falk, "TEMPLE: defining versatile TEMPlate LEarners via prototypical classifiers with learned similarities", IEEE Transactions on Pattern Analysis and Machine Intelligence.

Conference workshops

J. Monteiro, J. Alam, T. Falk, "A versatile and efficient approach to summarize speech into utterance-level representations", Efficient Natural Language and Speech Processing Workshop at NeurIPS 2021. ***Oral presentation***

Conference proceedings

J. Monteiro, I. Albuquerque, J. Alam, R. D. Hjelm, T. Falk "An end-to-end approach for the verification problem: learning the right distance", 37th International Conference on Machine Learning (ICML), 2020. arXiv:2002.09469

J. Monteiro, J. Alam, T. Falk, "A Multi-condition Training Strategy for Countermeasures Against Spoofing Attacks to Speaker Recognizers", The Speaker and Language Recognition Workshop (Odyssey), 2020.

J. Monteiro, J. Alam, T. Falk, "On The Performance of Time-Pooling Strategies for End-to-End Spoken Language Identification", 12th Language Resources and Evaluation Conference (LREC), 2020.

J. Monteiro, J. Alam, T. Falk, "An Ensemble Based Approach for Generalized Detection of Spoofing Attacks to Automatic Speaker Recognizers", 45th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2020.

J. Monteiro, J. Alam, T. Falk, "End-to-end Detection of Attacks to Automatic Speaker Recognizers with Time-attentive Light Convolutional Neural Networks", IEEE MLSP, 2019.

J. Monteiro, J. Alam, T. Falk, "Combining Speaker Recognition and Metric Learning for Speaker-Dependent Representation Learning", Interspeech, 2019.

J. Monteiro, J. Alam, G. Bhattacharya, T. Falk. "End-to-end language identification using a residual convolutional neural network with attentive temporal pooling", 27th European Signal Processing Conference (EUSIPCO), 2019. ***Oral presentation***

Other publications

Articles published in refereed journals

X. Liu*, **J. Monteiro***, I. Albuquerque, Y. Lai, C. Jiang, S. Zhang, T. Falk, J. Liang, "Single-shot real-time compressed ultrahigh-speed imaging enabled by a snapshot-to-video autoencoder", Photonics Research. ****Equal contribution***

Under review

I. Albuquerque, **J. Monteiro**, M. Darvishi, T. Falk, I. Mitliagkas "Generalizing to unseen domains via distribution matching", IEEE Transactions on Systems, Man, and Cybernetics: Systems. arXiv:1911.00804

I. Albuquerque, **J. Monteiro**, O. Rosanne, T. Falk "Estimating Distribution Shifts for Predicting Cross-Subject Generalization in Electroencephalography based Mental Workload Assessment", IEEE Transactions on Human-Machine Systems.

Conference workshops

J. Monteiro, M. O. Ahmed, H. Hajimirsadeghi, G. Mori, "Not too close and not too far: enforcing monotonicity requires penalizing the right points", XAI 4 Debugging Workshop at NeurIPS 2021. ***Oral presentation***

I. Albuquerque, **J. Monteiro**, T. Falk, "Randomly projecting out distribution shifts for improved robustness", Workshop on Distribution Shifts: Connecting Methods and Applications at NeurIPS 2021.

A. Bie, B. Venkitesh, **J. Monteiro**, M. A. Haidar, M. Rezagholizadeh "Fully quantizing transformer-based ASR for edge deployment", Workshop on Hardware Aware Efficient Training at ICLR 2021.

Conference proceedings

J. Monteiro, X. Gibert, J. Feng, V. Dumoulin, D.S. Lee "Domain Conditional Predictors for Domain Adaptation", Pre-registration workshop at NeurIPS, 2020.

M. Ravanelli, J. Zhong, S. Pascual, P. Swietojanski, **J. Monteiro**, J. Trmal, Y. Bengio, "Multi-task self-supervised learning for Robust Speech Recognition", 45th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2020. arXiv:2001.09239

I. Albuquerque*, **J. Monteiro***, T. Doang, B. Considine, T. Falk, I. Mitliagkas, "Multi-objective training of Generative Adversarial Networks with multiple discriminators", 36th International Conference on Machine Learning (ICML), 2019. arxiv:1901.08680 ****Equal contribution***

T. Doan, **J. Monteiro**, I. Albuquerque, B. Mazoure, A. Durand, J. Pineau, R. D. Hjelm. "Online Adaptative Curriculum Learning for GANs", The 33rd AAAI Conference on Artificial Intelligence, 2019. arXiv:1808.00020

J. Monteiro, J. Alam, "Development of Voice Spoofing Detection Systems for 2019 Edition of Automatic Speaker Verification and Countermeasures Challenge", IEEE ASRU, 2019.

J. Monteiro, I. Albuquerque, Z. Akhtar and T. Falk. "Generalizable Adversarial Examples Detection Based on Bi-model Decision Mismatch", IEEE SMC, 2019.

G. Bhattacharya, **J. Monteiro**, J. Alam, and P. Kenny. "Generative Adversarial Speaker Embedding Networks for Domain-Robust End-to-End Speaker Verification", 44th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2019.

1.5 Open-source code

Here, we list and link to open-source code produced within research projects reported in this document.

1.5.1 Open-source code implementing experiments included in the thesis

1. `https://github.com/joaomonteirof/multitask_asv`
2. `https://github.com/joaomonteirof/e2e_LID`
3. `https://github.com/joaomonteirof/e2e_antispoofing`
4. `https://github.com/joaomonteirof/e2e_verification`
5. `https://github.com/joaomonteirof/sim_learn`

1.5.2 Other open-source code

1. `https://github.com/joaomonteirof/hGAN`
2. `https://github.com/joaomonteirof/SMART_COUSP_Reconstruction`
3. `https://github.com/BorealisAI/monotonicity-mixup`

1.6 Thesis organization

The remainder of this document is organized as follows: Chapter 2 describes proposals in terms of training procedures aimed at improving representations used for speaker verification and language identification. Methods for detection of spoofing attackers are described in Chapter 3. Variations of the TDNN architecture using multi-level attentive layers are presented along with evaluation in Chapter 4. In Chapter 5, we introduce a discriminative approach to approximate generative verification or, in other view, an approach where (pseudo) metric spaces are learned via both and encoding process as well as a (pseudo) distance. We extend the approach discussed in Chapter 5 so that it can perform tasks other than verification, and introduce the idea of learning templates in Chapter 6. In this case, we discuss how to define model components and learning procedures that are re-usable across tasks and datasets, pending minimal task-specific implementations; in this case, we enlarge our scope and besides considering evaluations in voice biometrics contexts, we

further test our proposal in standard image benchmarks corresponding to object recognition and image retrieval tasks. Finally, conclusions are summarized in Chapter 7 along with discussion on future work building upon our contributions. Pointers to background material are provided in the Appendix A.

Chapter 2

Improving neural network training for utterance-level representation learning

2.1 Preamble

This chapter is compiled from material extracted from the two following publications: Section 2.3 covers the application of our proposals to language identification tasks as reported in [133] and published in *Computer Speech & Language*. In Section 2.4, we then discuss variations of the proposed setting in applications to speaker verification cases, as discussed in [134], published in the *Proceedings of INTERSPEECH (2019)*.

2.2 Introduction

In the case of ASV or LID tasks, training of approaches such as x-vectors [3] or any other neural network-based system has generally been performed under the multi-class classification setting, i.e., the model is used as a classifier aiming to identify the speaker/language from a given input utterance. The outputs of a final softmax layer thus parameterize a data-conditional categorical distribution over the set of training speakers or languages, and parameters are learned via maximum likelihood estimation (MLE) through minimization of the cross entropy loss. At testing time, outputs of intermediate layers are used as low-dimensional representations on top of which a binary classifier

can be trained for verification purposes for open set conditions, while the outputs are directly used as scores within closed set testing conditions.

Another relevant body of literature on representation learning for ASV or LID is based on metric learning methods [5]. Under this setting, a representation model maps sequences of speech features into a low-dimensional space and training is carried out so as to minimize the distance between embeddings from the same class while maximizing the distance between those from different classes. The metric learning approach was explored in large scale cases for face verification [62], but training difficulties have been commonly reported [71]. As such, stabilizing strategies have been proposed, such as triplets mining, supervised pre-training, and combination of triplets loss with center- [25] or intra-losses [110]. Even though the metric learning framework seems to be a good fit to learn speaker-dependent low-dimensional representations, the training difficulties observed when training is performed under this setting (e.g., finding informative sets of negative examples at training time) have to be dealt with.

In this contribution, we design a training strategy particularly tailored to further enforce discriminability on model outputs while avoiding common issues appearing in metric learning settings. Specifically, we evaluate the combination of the two previously-described frameworks, namely: (i)-MLE via speaker or language recognition, and (ii)-metric learning. Our main goal is to combine the advantages offered by each scheme, i.e., the relative easiness of training under the maximum likelihood setting along with the discriminability provided by triplet loss minimization. Evaluation of the proposed training strategy is performed using the triplet-network [6] realized with an architecture inspired in the ResNet model augmented with a frame-wise attention component. For the case of LID, experiments are performed on the tasks introduced for the AP18-OLR challenge corresponding to data containing recordings of telephone speech from ten oriental languages under different settings, including short-duration of speech and confusing languages, showing relevant improvements in terms of classification performance over strong baselines in all studied test conditions. Moreover, end-to-end evaluation is also carried out showing that directly utilizing model outputs as scores, i.e., discarding post-trained PLDA, outperforms i-vectors+PLDA's results. For ASV, the cross-language setting introduced for NIST SRE 2016 composed of telephone speech is employed for evaluation of the proposed scheme. Additionally, different pooling strategies, used to aggregate sets of local descriptors into a fixed-dimensional representation space, are compared including simple statistics of high-level representations across the time dimension and more complex learning-based

attentive schemes. In what follows, we split the presentation for the case of LID and ASV so that training details required for either application are specified more clearly.

2.3 Application to language identification

2.3.1 Model used for evaluation of the proposed training scheme

As mentioned previously, we propose the use of a convolutional architecture aiming to include long-term contextual information at each time-step. This is an inherent feature of stacked convolutional layers [69]. It is important to highlight that, differently from other approaches that employ *causal* convolutions for temporal dependency modelling [70], the setting explored herein assumes access to the full speech recording for computation of each output time-step. This allows us to compute fixed dimensional language-dependent embeddings relying on full recordings. Moreover, a residual architecture is employed. Namely, a slightly modified ResNet-50[1] [4], i.e., a set of 50 layers organized as a stack of residual blocks, is used throughout our experiments. Inputs are time-frequency representations of speech data, i.e., 13 mel frequency cepstral coefficients (MFCCs), which are simply treated as one-channel images. We introduce a convolutional layer prior to ResNet's residual stack which shrinks the frequency dimension to 1. Moreover, a frame-wise attention component is employed on top of the last convolutional layer substituting the original fully connected output layer. The channel dimension corresponding to the last convolutional layer within the residual stack will yield the final embedding sizes, which are then pooled across the time dimension using the frame-wise mechanism described in Chapter 1. A diagram illustrating the proposed model is presented in Figure 2.1.

2.3.2 Training details

As discussed above, two different training strategies are used together to enforce language dependency on embeddings y. First, we directly train the model for classification by projecting y onto an output layer using a fully connected additional layer, and train the model via MLE, i.e., with multi-class cross entropy minimization, as commonly done for speaker recognition [47, 22, 3].

[1] Architecture details of ResNet-50: `http://ethereon.github.io/netscope/#/gist/db945b393d40bfa26006`

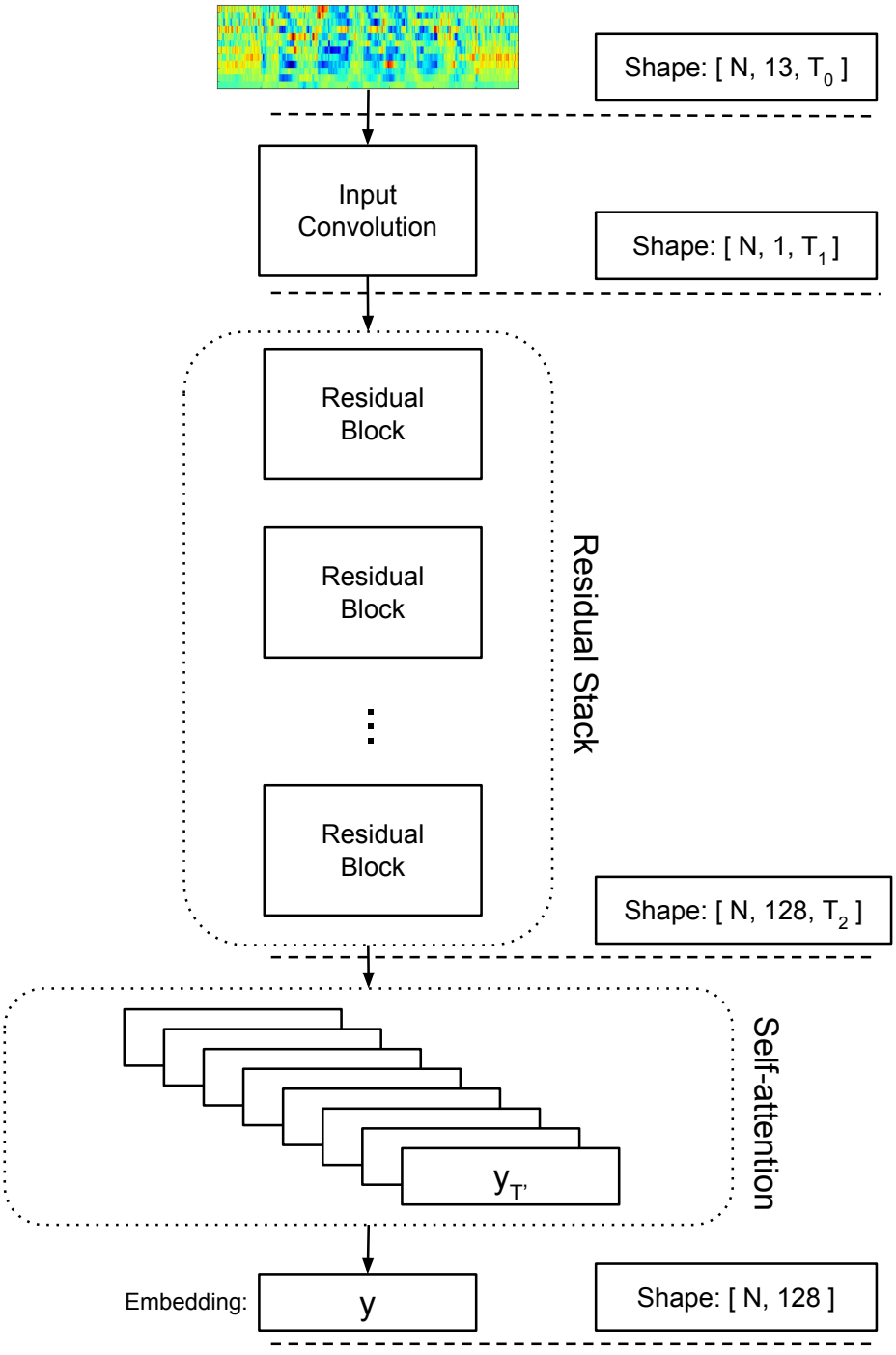


Figure 2.1 – Proposed residual convolutional neural network model with frame-wise attention employed for language recognition. The shapes indicate the dimensionality of the processed data within different parts of the model. N indicates the number of examples contained in a batch of data, while T_0, T_1, and T_2 stand for the dimensionality across the time dimension in different points.

Moreover, aiming to enforce language discriminability in y, triplet-loss minimization on top of y is jointly performed along with MLE, employing a distance metric based on the cosine similarity.

The most common definition of triplet loss is given by:

$$\mathcal{L}_T = \max(d_+ - d_- + \alpha, 0), \tag{2.1}$$

where d_+ and d_- correspond to a distance measure between pairs of embeddings obtained from recordings of the same language, and from different languages, respectively. Parameter α is a hyperparameter commonly referred to as margin.

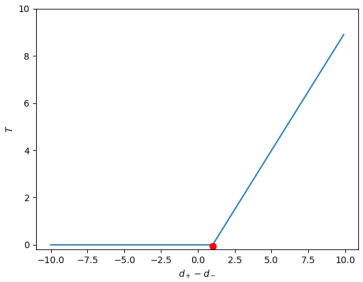
The operator $\max(x, 0),\ x \in \mathbb{R}$, is used so that triplets, i.e., pairs corresponding to the same and different languages, respectively, that already have low d_+ and high d_- stop influencing training. Here, we follow the approach in [71] and enforce concentration of same language embeddings by using a soft-margin variation of the triplet loss given by:

$$\mathcal{L}_T = \text{softplus}(d_+ - d_-), \tag{2.2}$$

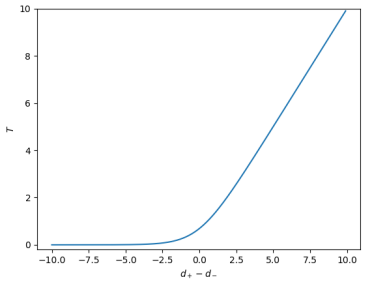
where the softplus operator is defined as:

$$\text{softplus}(x) = \log(1 + e^x). \tag{2.3}$$

Figures 2.2a and 2.2b illustrate how T changes as a function of $(d_+ - d_-)$ for the hard- and soft-margin cases, respectively. One can notice that for high $(d_+ - d_-)$, i.e., in a triplet in which the negative pair is closer than the positive one, both loss definitions will behave similarly, and T will depend linearly on $(d_+ - d_-)$. However, in low loss scenarios, the hard-margin definition will simply set $T = 0$ once $(d_+ - d_-) < \alpha$, which is convenient when mining schemes are employed at training time to ensure only triplets with high loss are used. In our training pipeline, on the other hand, randomly selected triplets are employed in order to avoid the excessive cost involved in mining methods and thus, the soft-margin definition of T makes it possible to obtain a training signal even from examples in which $(d_+ - d_-) > \alpha$.



(a) Hard-margin triplet loss as a function of $(d_+ - d_-)$. The red dot corresponds to α.



(b) Soft-margin triplet loss as a function of $(d_+ - d_-)$. Notice $\mathcal{L}_T > 0$ even if $(d_+ - d_-) < \alpha$

Figure 2.2 – Illustration of triplet losses under hard or soft margins.

Several works have proposed variations of triplet loss in which different distances d are employed. Here, we define d as:

$$d(y_1, y_2) = 1 - \frac{y_1 \cdot y_2}{||y_1||_2||y_2||_2}, \tag{2.4}$$

where the second term is the cosine of the smallest angle between y_1 and y_2. The definition of $d(y_1, y_2)$ making use of the cosine similarity as a distance metric is intended for it not to depend on the norms of embeddings, as would be the case if some L_p norm were used instead, i.e., $d(y_1, y_2) = ||y_1 - y_2||_p$. Employing embeddings with unconstrained norm will let the model benefit from the cross entropy loss term, otherwise the two different objectives would be counter-acting each other, since both losses would simultaneously rely on their ability to control embeddings norms.

Training is carried out using gradient-based minimization of the sum of cross entropy and soft-margin triplet losses. RMSProp [135], a variation of SGD in which gradients are scaled by a running average of their recent magnitudes, is employed for optimization with its smoothing constant set to 0.99. The global learning rate starts at 0.001 and is halved once the classification error rate, measured on a validation set held out of training, plateaus for 30 epochs. A single Titan X NVIDIA GPU is used for training.

Data sampling and triplet selection: Mini-batches of size 64 are assembled such that two random recordings of each language are sampled sequentially to form same language pairs (positive), and a random recording from a different language is selected to compose the different language pair (negative). One epoch is considered finished when each language is selected 1000 times to compose positive pairs. A budget of 500 epochs is used for each training run, which takes approximately 48 hours under the described setting.

Moreover, recordings are processed on the fly during training in such a way that, for recordings longer than 6 seconds, a random continuous segment of 6 seconds duration is selected. Short-duration training recordings are elongated with initial frames so as to make them reach a minimum of 6 seconds duration. To further increase the diversity on training samples, each mini-batch has its length randomly selected to lie between 3-6 seconds before feeding it to the neural network. At testing time, on the other hand, full recordings are used for scoring in the end-to-end setting or to obtain low-dimension representations, without any processing prior to forward-pass test recordings into the trained model.

2.3.3 Evaluation details, results, and discussion

We evaluate our proposed framework using the dataset introduced for the AP18-OLR Challenge [8] described in Chapter 1, which consists of recordings with unconstrained phonetic content of telephone speech in 10 different oriental languages. Information about speaker identity, gender, or age were not utilized, nor was phonetic information. We introduce multi-condition training data by augmenting the original train partition with supplementary noisy speech, created by corrupting original samples adding reverberation (reverberation time varies from 0.25s - 0.75s), adding noise at signal-to-noise ratio (SNR) ranging from 0 to 15 dB, as well as by adding background noise, such as music (SNR within 5-15 dB), and babble (SNR varies from 10 to 20 dB). Babble, music, and other noise signals were selected from the MUSAN corpus [136] and the room impulse responses (RIRs) used to simulate reverberant effects were taken from *openslr*[2]. Multi-style training compensates for environmental mismatch between training and test data. Due to the supervised training nature commonly employed in deep neural networks, use of data augmentation is ubiquitous and allows for better generalization of the network by creating diversity in the training data. Moreover, voice activity detectors are usually used in the pre-processing phase for speaker recognition applications. Here, we empirically observed better performance when silence segments were kept in both training and test recordings, thus VADs are not required. We hypothesize that the way short pauses occur within speech might be characteristic in each language, and our model is thus learning such pause patterns.

[2] `http://www.openslr.org/`

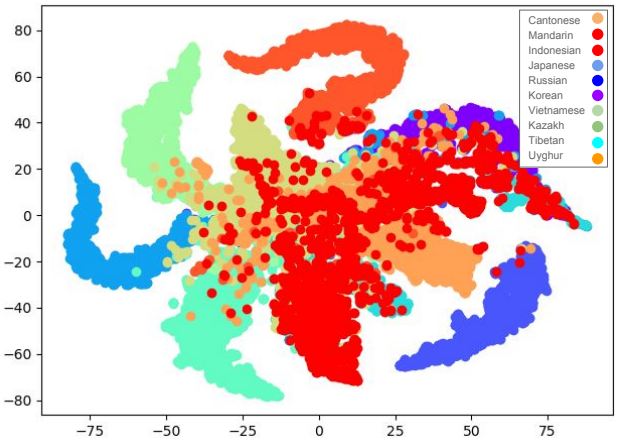


Figure 2.3 – 2-dimensional t-SNE embeddings of test recordings obtained after the attention layer. Each color stands for one different language. Better viewed in color.

The first step to evaluate the proposed model and training scheme is to visually inspect embeddings generated from data which was not used during training. Embeddings dimensionality were set to 128, which means the number of channels of the last convolutional layer is set to 128. The 2-dimensional *t*-SNE [137] embeddings of test data representations learned by our proposed model are shown in Figure 2.3. Each color represents a different language. As desired, each different color is clustered together in different regions of the embedding space, thus assuring class separability, which is useful for posterior scoring strategies.

We proceed to evaluate the proposed system under the closed set verification setting. We do so by using the provided development set, which corresponds to test data released for the AP17-OLR challenge [138], in which a set of short-duration recordings, i.e., speech segments containing less than 1 second total duration, is provided. Trials lists are included along with test data for both short-duration condition and full-length test recordings, which can include short recordings as well. Three strategies are used for scoring trials:

1. PLDA trained on the embeddings of the full set of training data.
2. The cosine similarity between enrollment language models obtained by averaging embeddings of all training recordings from a given target language, and the embedding of the test recording.
3. End-to-end: the output of the softmax layer corresponding to the claimed language is used as score.

PLDA and cosine similarity are employed as backends across all implemented benchmark systems, and linear discriminant analysis (LDA) is further employed to reduce the dimensionality of embeddings in the case of PLDA and cosine similarity backends. The final embeddings in such cases correspond to 64 dimensional representations, which we observed to work well across considered conditions.

Several benchmark systems are evaluated for comparison, including results reported in [8], obtained with i-vectors [1] using both LDA and PLDA backends, along with two neural network based systems: a TDNN [27] and a LSTM [28], in both cases using end-to-end scoring. Moreover, in-house implementations of other benchmark systems include:

1. i-vectors. For this system, a 2048-Gaussians full covariance universal background model (UBM) along with a 400-dimensional i-vector extractor are trained using MFCC features on the training data. For scoring, cosine similarity and PLDA backends are used. Before scoring, LDA is applied for reducing the i-vectors dimensionality.
2. Tandem: i.e., low-dimensional embeddings obtained from principal component analysis (PCA) of statistics from a GMM-UBM trained on top of tandem features, similar to the setting described in [139]. However, here tandem features correspond to output posterior distribution of the phonetic neural networks, rather than a bottleneck layer output. The phonetic neural networks were trained on the THCHS30 corpus, an open Chinese speech database [140].
3. cLSTM: a convolutional-recurrent model consisting of 6 convolutional layers followed by a 2-layered bi-directional LSTM, trained with the same setting as the proposed model. In this case, evaluation is performed with the same end-to-end strategy employed by our proposed model, as well as using PLDA and cosine similarity as backends, in which case previous dimensionality reduction is performed using LDA.

Results in terms of equal error rate (EER) and average cost performance (C_{avg}) are reported in Table 2.1. Both EER and C_{avg} are better when closer to 0. While EER consists of the value of the false acceptance rate at the threshold in which it matches the false rejection rate, C_{avg}, on the other hand, averages the missing and false alarm probabilities for each target/non-target pair of languages. More details about both metrics can be found in [8].

As expected, the performance of the i-vector+PLDA approach drops substantially from the full-length to short-duration settings, in both the in-house implementation and that reported in [8].

Table 2.1 – Performance comparison of proposed system (last three rows) and benchmarks based on equal error rate (%) and average cost performance (C_{avg}). A total of 220510 trials were processed in both short-duration and full-length cases.

		Short-duration		*Full-length*	
		EER (%)	C_{avg}	EER (%)	C_{avg}
Benchmarks [8]	i-Vector+LDA	18.04	0.1784	6.12	0.0598
	i-Vector+PLDA	17.51	0.1746	5.86	0.0596
	TDNN	14.04	**0.1282**	11.31	0.1034
	LSTM	15.92	0.1452	12.76	0.1154
Benchmarks (our implementation)	i-Vector+Cosine	18.37	0.1786	5.48	0.0514
	i-Vector+PLDA	17.95	0.1756	5.42	0.0513
	Tandem+Cosine	15.13	0.1457	5.27	0.0508
	Tandem+PLDA	14.57	0.1431	4.80	0.0454
	LSTM+Cosine	21.05	0.2046	5.91	0.0564
	LSTM+PLDA	20.31	0.1982	5.17	0.0482
	LSTM	24.17	0.2330	4.74	0.0450
Proposed	LDA+Cosine	15.05	0.1465	4.47	0.0435
	LDA+PLDA	14.14	0.1361	3.59	0.0343
	End-to-end	**13.26**	0.1291	**2.76**	**0.0257**

TDNN and LSTM, in turn, perform better than i-vectors in the short-duration case, but not in the full-length evaluation, thus suggesting that such models are not effective in handling longer-term dependencies. This could be due to the limited context in the TDNN case, or the known training difficulties in the long sequences regime faced by RNNs [141], including LSTMs.

As can be seen, our proposed approaches (last three rows in Table 2.1) outperform all considered baselines in the full-length evaluation, indicating the added context together with the employed attentive pooling effectively improve modelling of long-term dependencies. More importantly, for the end-to-end scoring, i.e., without the use of any extra training step after training the convolutional model, EER in both short-duration and full-length conditions are lower when compared to all evaluated benchmarks in both testing conditions. We further highlight that PLDA and cosine similarity backends, which would be valid scoring strategies in a complete open set evaluation scenario, are also able to outperform benchmarks in the full-length case, while on par in short-duration. We further observe tandem features are able to outperform i-vectors, TDNN, and LSTM, which suggests such phonetics-dependent representations can be a promising space to train our system, which we intend to evaluate in future work.

We further evaluate the proposed approach using the test conditions introduced for the AP18-OLR challenge, consisting of: (a) short-duration recordings with up to 1 second, (b) confusing

Table 2.2 – Performance comparison of proposed system (last three rows) and benchmarks based on equal error rate (%) and average cost performance (C_{avg}). Confusing languages correspond to Cantonese, Korean, and Mandarin. A total of 214560, 22071, and 404160 trials were processed, respectively, for each evaluation condition: short-duration, confusing-languages, and unseen languages.

		Short-duration		*Confusing languages*		*Unseen non-target languages*	
		EER (%)	C_{avg}	EER (%)	C_{avg}	EER (%)	C_{avg}
Benchmarks	i-Vector+Cosine	18.02	0.1780	10.71	0.1069	7.77	0.0577
	i-Vector+PLDA	17.50	0.1743	10.66	0.1059	7.51	0.0524
	Tandem+Cosine	15.73	0.1502	13.81	0.1387	8.98	0.0683
	Tandem+PLDA	15.30	0.1461	13.33	0.1324	8.37	0.0596
	LSTM+Cosine	20.10	0.1978	9.11	0.0840	7.78	0.0537
	LSTM+PLDA	19.14	0.1896	8.78	0.0819	7.49	0.0490
	LSTM	24.00	0.2321	7.54	0.0738	7.57	0.0491
Proposed	LDA+Cosine	14.63	0.1432	9.81	0.0967	6.44	0.0463
	LDA+PLDA	13.48	0.1328	8.28	0.0810	5.97	0.0369
	End-to-end	**12.62**	**0.1246**	**6.80**	**0.0669**	**5.65**	**0.0315**

languages, in which only trials corresponding to 3 languages known to be difficult to distinguish are included (Cantonese, Korean, and Mandarin), and (c) an open set condition, in which non-target test recordings in languages not present in the training data are included. Results are once again reported in terms of EER and C_{avg} in Table 2.2.

The results show that end-to-end scoring of the proposed method outperforms all compared benchmarks in all evaluation conditions. This supports the claim that convolutional layers along with attention mechanisms form an effective approach to model language dependencies in different time-scales. Particularly, even though a fully-supervised setting is employed at trainining time, in the sense that language information is required for each train recording, the introduced test recordings from unseen languages at training time do not significantly affect the performance of our proposed approach, regardless of the employed scoring strategy. This is due to the effect imposed by triplet loss minimization, which enforces both class-separability and concentration of embeddings belonging to the same class. The minimization of triplet loss acts as a regularization term and avoids domain-dependency, e.g., speaker- or channel-dependency, on learned embeddings, which would incur in generalization issues, as commonly observed in applications with small training datasets. The model trained using the proposed strategy was able to yield improvements in terms of C_{avg} of 28.51%, 36.83%, and 39.88% for short-duration, confusing languages, and open set evaluation conditions, respectively, when compared to an i-vector system with PLDA scoring.

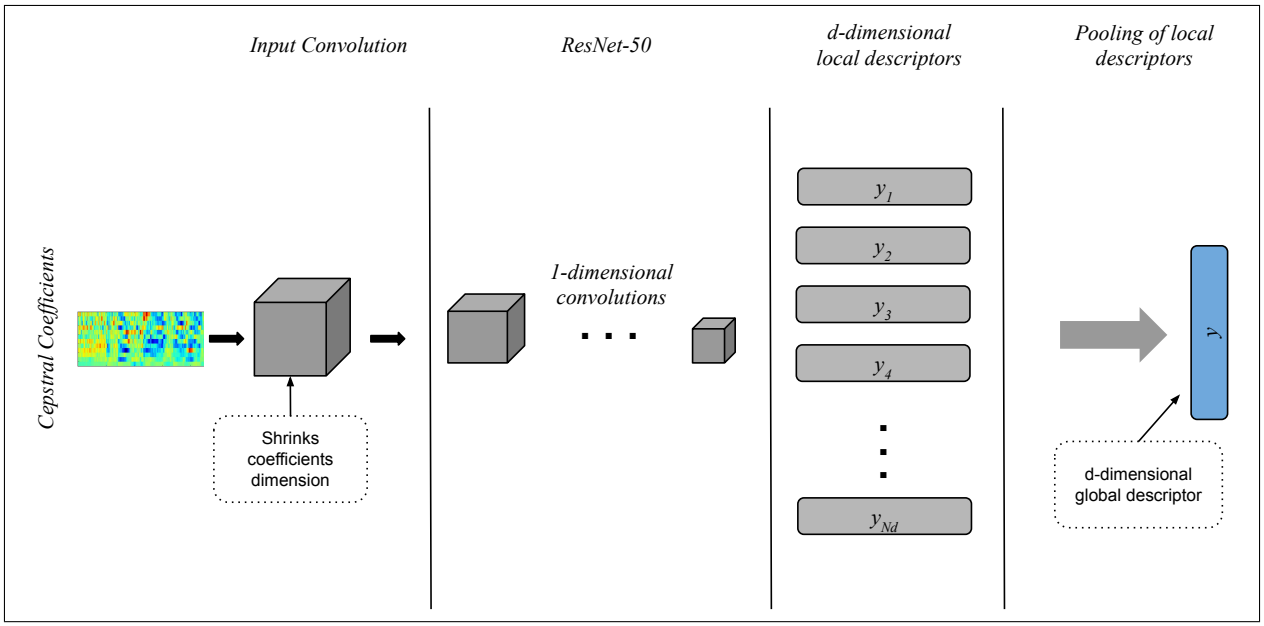


Figure 2.4 – Diagram representing our proposed system. Features are mapped into local descriptors, which aggregated to yield final representations.

2.4 Application to speaker verification

In the case of ASV, a modified ResNet-50 [4] is used throughout our experiments. However, unlike the generic triplet-network represented in Figure 1.10 in which we assumed data samples have a fixed dimension D, text-independent ASV systems usually have to deal with recordings of varying lengths. We thus split the mapping F from data to embeddings into two separate stages: (i) map speech features into a set of vectors representing parts of the input across the time dimension, and (ii) aggregate such local descriptors into a single vector representing the complete input recording. A block diagram describing those two stages is shown in Figure 2.4.

Mel frequency cepstral coefficients are treated as 1-channel images and fed into an input convolutional layer containing 32 filters of dimension $[N_{CC}, 3]$, where N_{CC} is the number of MFCCs in the input, resulting in a 32-channel temporal representation. This representation then serves as input for the Resnet, which operates over the temporal dimension only. The output of the convolutional layers is a set of N_d local descriptors $y_i \in \mathbb{R}^d$ representing parts of the input across time, where N_d is a function of the input length L, and d will be given by number of filters in the last convolutional layers, set to 512. The following step consists in aggregating the set of local descriptors y_i into what we refer to as a global descriptor, i.e., the vector $y \in \mathbb{R}^d$ representing an utterance of arbitrary

length. We consider three temporal pooling strategies as detailed below and depicted in Figures 2.5, 2.6, and 2.7:

Statistics pooling: The global descriptor y will be given by a linear projection P of concatenated element-wise estimates of first- and second-order statistics of the set of local descriptors $y_i \in \mathbb{R}^d, i \in \{1, 2, ..., N_d\}$:

$$y = P \cdot \text{cat}[\mu(y_i), \sigma(y_i)], \tag{2.5}$$

where $P_{[2d,d]}$ entries are learned jointly with convolutional layers parameters. The operator $v = \text{cat}[v_1, v_2]$ concatenates $v_1, v_2 \in \mathbb{R}^d$ such that $v \in \mathbb{R}^{2d}$, and P's dimension is such that the dimension of y is 256.

Attentive pooling: We augment the previous pooling scheme with a weighing method often referred to as soft-attention. A linear transformation A, whose entries are learned along with the complete model, is first applied to each local descriptor y_i, resulting in the set of scalars $a_{1:N_d}$:

$$a_i = \tanh\left(A \cdot y_i\right). \tag{2.6}$$

A set of normalized weights summing up to 1 is then obtained through the softmax operator:

$$w_i = \frac{e^{a_i}}{\sum_{i=1}^{N_d} e^{a_i}}, \tag{2.7}$$

and the global descriptor y is finally given by the projection of concatenated statistics of weighted local descriptors, i.e.:

$$y = P \cdot \text{cat}[\mu(w_i \cdot y_i), \sigma(w_i \cdot y_i)]. \tag{2.8}$$

Recurrent attentive pooling: Lastly, a recurrent model is employed along with the soft-attention scheme described above so that its hidden layer can be further used as a summary of the set of local descriptors. The recurrent model is implemented as a two-layered bi-directional LSTM [28] with its hidden layers set to a size of 256. The LSTM will first map the set y_i into a new sequence y'_i and a hidden state h hence, y becomes:

$$y = P \cdot \text{cat}[\mu(w_i \cdot y'_i), \sigma(w_i \cdot y'_i), h], \tag{2.9}$$

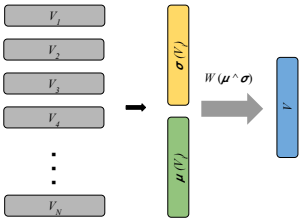


Figure 2.5 – Statistics pooling.

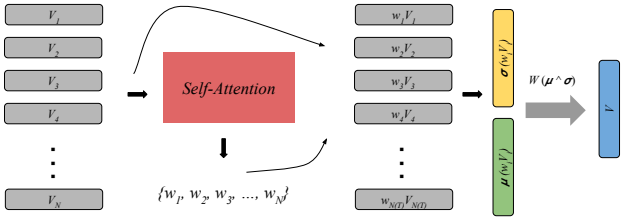


Figure 2.6 – Attentive pooling.

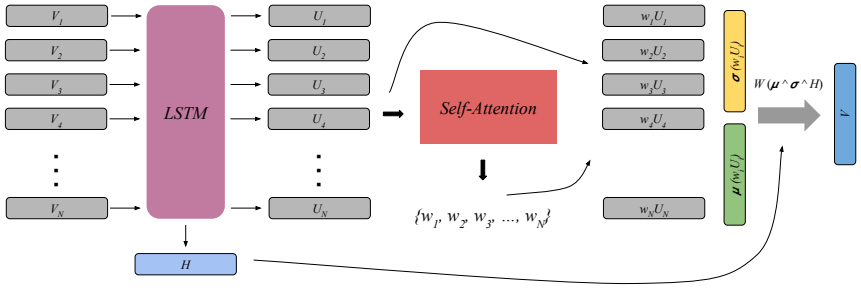


Figure 2.7 – Recurrent attentive pooling.

where weights w_i are obtained with soft-attention on y'_i rather than y_i as in the previous pooling scheme.

2.4.1 Training loss

Speaker recognition and triplet loss minimization are performed jointly during training. Such two-loss components can be computed once the embedding y is obtained through one of the pooling approaches described. Triplet loss will be computed on top of embedding projected on the unit sphere: $y^p = \frac{y}{||y||_2}$, and for the speaker recognition term, a softmax output layer is employed so that the multi-class cross entropy can be computed using speaker identities as class labels. Training is performed so as to minimize the sum of the two losses.

The triplet loss $\mathcal{L}_T$ can be computed as described in Equation 1.12. We employ the L_2 norm in this case and compute the loss on top of the projected embeddings on the hypersphere y^p, which thus yields:

$$\mathcal{L}_T = \frac{1}{N_t} \sum_{i=1}^{N_t} \max(||y^p_{i,a} - y^p_{i,+}||_2 - ||y^p_{i,a} - y^p_{i,-}||_2 + \alpha, 0). \tag{2.10}$$

Moreover, in order to perform speaker recognition, our models contain an additional dense layer which, followed by the application of the softmax operator, defines a transformation over embeddings y. A vector of probabilities p over the set of training speakers is then obtained through F': $R^d \rightarrow \Delta^{N_S-1}$, where N_S is the number of speakers and Δ^{N_S-1} defines a simplex in R^{N_S}, i.e., the sum of the components of any vector in Δ^{N_S-1} is equal to 1. We do so to ensure the vector $p = F'(y)$ corresponds to the parameters of a categorical distribution over the set of training speakers.

Performing maximum likelihood estimation on the conditional categorical distribution discussed above is equivalent to minimize the cross entropy loss $\mathcal{L}_{CE}$, which for a set of N_e embeddings y_i will be given by:

$$\mathcal{L}_{CE} = \frac{-1}{N_e} \sum_{i}^{N_e} \sum_{j}^{d} \log[F'(y_i)_j] \cdot t_{i,j}, \tag{2.11}$$

where the inner summation is performed over the elements of the vector $\log p_i$, as defined by the term $\log[F'(y_i)_j]$, and $t_i \in R^{N_S}$ is a *one-hot* encoded vector representing the identity of a given speaker, i.e., the entry of t_i given by $t_{i,j}$ will be 1 if the identity of the speaker corresponding to example i is j, and 0 otherwise.

The combined training loss $\mathcal{L} = \mathcal{L}_T + \mathcal{L}_{CE}$ will be finally, given a set of examples of size N_e, along with the *one-hot* encoded speaker identities t:

$$\mathcal{L} = \frac{-1}{N_e} \sum_{i}^{N_e} \sum_{j}^{d} \log[F'(y_i)_j] \cdot t_{i,j} + \frac{1}{N_t} \sum_{i=1}^{N_t} \max(||y^p_{i,a} - y^p_{i,+}||_2 - ||y^p_{i,a} - y^p_{i,-}||_2 + \alpha, 0). \tag{2.12}$$

Note that Equation 2.12 does not clarify how one can obtain a set of N_t triplets out of a set of N_e training examples. We describe the triplet selection procedure as well as the method we employ to sample examples from the pool of training recordings in the following Section.

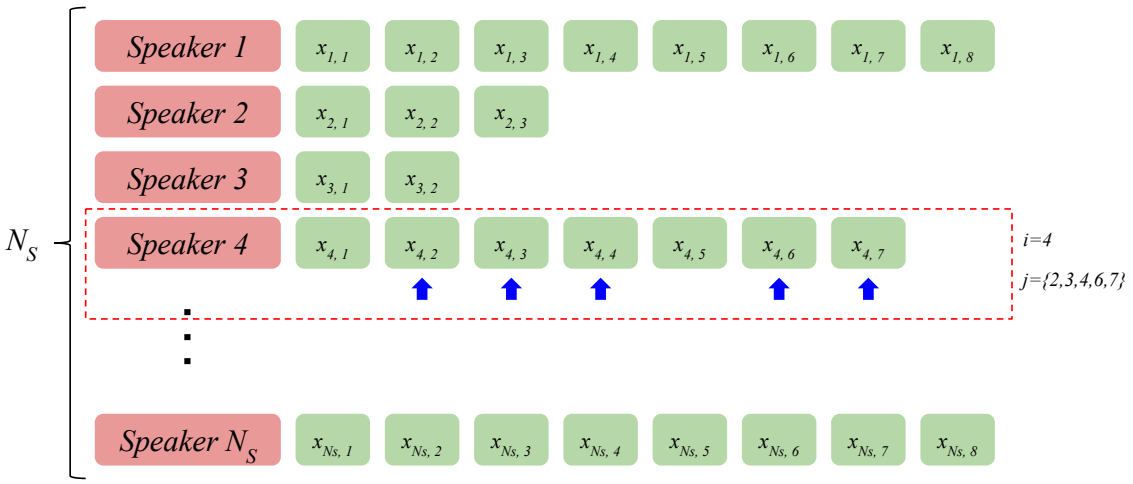


Figure 2.8 – Sampling training examples from the dataset. Speakers are selected sequentially and five recordings are randomly selected for each such speaker so as to compose a training mini-batch.

2.4.2 Mini-batch construction and triplets selection

Given that the number of training recordings is not constant across speakers, we devise a particular approach to sample examples so as to present balanced mini-batches to models throughout training. Mini-batches are constructed through sequentially picking examples from each speaker. More specifically, 5 recordings are sampled with repetition. We exemplify such a sampling scheme in Figure 2.8. The speaker index $i \in \{1, 2, ..., N_S\}$ cycles over the set of speakers and for each value of i, a set of randomly selected recordings represented by $x_{i,j}$ is then returned. Such an approach provides mini-batches of size $N_e = S \cdot R$, where R and S correspond to the number of speakers per mini-batch and the number of recordings per speaker, respectively. While R is set to 5 as per the example in Figure 2.8, S is set to 24, which gives $N_e = 120$. One training epoch is considered finished when sets of 5 recordings are sampled from each speaker 3 times, and a budget of 500 epochs is used for each training run.

We further use the described sampling scheme to increase the diversity of the training data at hand. Once each recording $x_{i,j}$ is selected, it is further processed such that a randomly positioned window of 10 seconds will be selected for those longer than 10 seconds, and shorter-duration training examples are elongated by repeating initial frames so as to make them reach a minimum of 10 seconds duration. Additionally, each mini-batch has its length randomly selected to lie between 3-10 seconds before being fed into the model during training iterations. At testing time, however, recordings are fed with their original length.

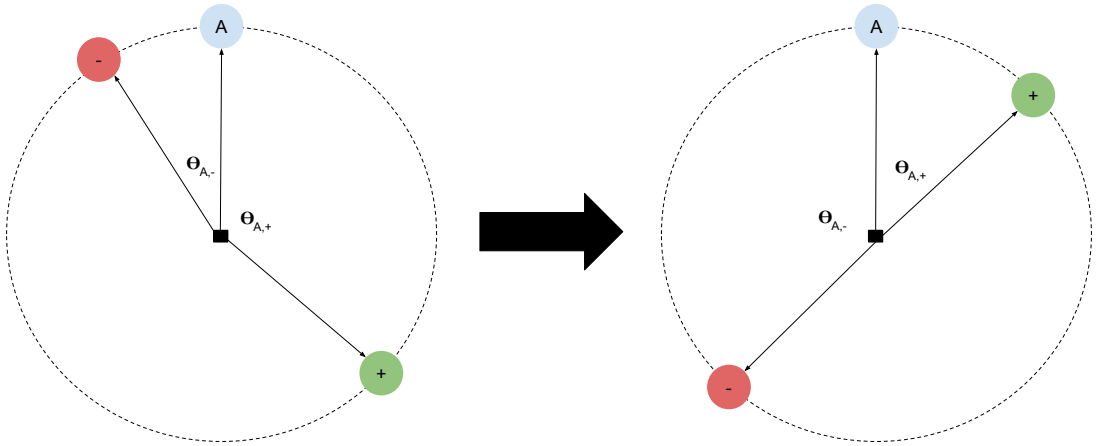


Figure 2.9 – Hard and easy triplets. Triplet loss minimizes the distance between anchor and positive examples while maximizing that between anchor and negative cases.

Previous literature has discussed in depth the need of finding sets of triplets yielding a high triplet loss, i.e., *hard* triplets in the sense that $||y_a - y_-||_2 < ||y_a - y_+||_2$ resulting in an informative $\mathcal{L}_T$. In [71], authors argue representation models are able to quickly learn to map trivial triplets correctly, which renders a large fraction of possible triplets uninformative. This is exemplified in Figure 2.9 for triplets in a sphere in $\mathbb{R}^2$ which represents a "hard" triplet on the left, and those appear very frequently early on during training, and "easy" triplets on the right, which represent the majority of randomly selected sets of examples after a relatively small number of training iterations.

Given that our mini-batch construction scheme ensures R recordings per speaker are available, we thus employ an online triplet selection method similar to that introduced in [62]. The most recent version of the model embeds all the N_e training examples, and all possible positive pairs are taken. The hardest negative pairs are then selected so as to match the number of positive pairs. This procedure should yield $N_t = S\frac{R(R-1)}{2}$, which gives at most $N_t = 240$ given that $S = 24$ and $R = 5$ in our setting, and triplets such that $\mathcal{L}_T = 0$ are discarded.

2.4.3 Maximum Entropy Regularization

It is well known that two random vectors on a high-dimensional hypersphere will likely be orthogonal. This can be problematic in the case speaker-dependent embeddings or global descriptors y lie in the unit sphere in $\mathbb{R}^d$, since dot products in that case wouldn't discriminate between positive

and negative pairs[3]. We employ an entropy regularizer so as to enforce speaker representations to spread across the sphere, which favors discriminability. A regularization penalty was introduced in [142] specifically to this end based on the Kozachenko-Leononenko estimator of the differential entropy [143]. For a finite sample of size n, we use a slightly modified maximum entropy regularizer for embeddings $y \in \mathbb{R}^d$:

$$H_n = -\sum_{j=1}^{n} \rho_{n,j}, \tag{2.13}$$

where $\rho_{n,j} = \min_{j \neq k} ||y_j - y_k||_2$. A coefficient λ is further added in order to control the influence of the regularizer. We set λ to 0.01 in all our experiments.

A rather practical observation we would like to provide is that such regularization penalty alleviates numerical instabilities observed when training PLDA on top of embeddings obtained using the described systems. Global descriptors extracted with earlier versions of the setting described here would very often yield design matrices of incomplete rank, which is not the case when embeddings are encouraged to spread over the sphere.

2.4.4 Other training details

A pseudocode describing the training procedure is presented in Algorithm 1 in order to summarize the above described training steps. The employed parameter's update rule is SGD. Weight decay is further employed as a regularization penalty to avoid overfitting to training data. A schedule is defined such that the learning rate is halved if a validation loss does not improve for at least 15 epochs. In order to do so, we built a validation set by selecting all the recordings of a group of 50 randomly selected speakers, taken out of training data. The generalization performance is assessed by the EER obtained by using cosine similarity to score a set of trials created at training time out of recordings from the validation set.

The initial learning rate and the weight decay coefficient are set to 1e−2 and 5e−5, respectively, given those yielded the best results on a grid search across considered settings. Momentum coefficient is set to the default value of 0.9. Model training is implemented in Pytorch [144] and takes approximately 8 days to complete in a single NVIDIA Titan X GPU. All results are reported for the model that achieved the lowest validation EER during training.

[3]The dot product $a' \cdot b'$ between two vectors $a' = \frac{a}{||a||_2}$ and $b' = \frac{b}{||b||_2}$ will be proportional to the Euclidean distance $||a' - b'||_2$.

Algorithm 1 Training Speaker Embedding Model.

$F, F' \leftarrow InitializeModel()$
F_{best}, $EER_{best} \leftarrow F, \inf$
repeat
  $x, l \leftarrow SampleMiniBatch()$
  $y \leftarrow F(x)$
  $y^p \leftarrow \frac{y}{||y||_2}$
  $p \leftarrow Softmax(F'(y))$
  $y^p_a, y^p_+, y^p_- \leftarrow SelectTriplets(y^p)$
  $H_n \leftarrow SumMinimumDistancePairs(y^p)$
  $\mathcal{L} \leftarrow \mathcal{L}_{CE}(p, l) + \mathcal{L}_T(y^p_a, y^p_+, y^p_-)$
  $\nabla_{\mathcal{L}} \leftarrow BackPropagate(F, F', \mathcal{L}, H_n)$
  $F, F' \leftarrow SGD(F, F', \nabla_{\mathcal{L}})$
  $EER \leftarrow Validate(F, x_{val}, l_{val})$
  if $EER < EER_{best}$ **then**:
    $EER_{best} \leftarrow EER$
    $F_{best} \leftarrow F$
  endif
until Maximum number of iterations reached
return F_{best}

2.4.5 Evaluation and Discussion

We evaluate the proposed setting (model and training scheme) under the conditions introduced for the NIST SRE 2016. In order to train speaker embedding models, a first training dataset is built by combining the data from *NIST SRE*s from 2004 to 2010, *Mixer* 6, as well as *Switchboard*-2, phases 1, 2, and 3, summing up to approximately 7000 speakers, out of which we remove all the recordings from 50 speakers to be used as validation set. Speech features correspond to 23 MFCCs obtained with a short-time Fourier transform using a 25 ms Hamming window with 60% overlap. An energy-based voice activity detector is employed to filter out non-speech frames. Multi-condition training data is further introduced by augmenting the original train partition with supplementary noisy speech in order to enforce the model's robustness across varying conditions. We thus created additional versions of training recordings as similarly done in [3], i.e., by corrupting original samples adding reverberation (reverberation time varies from 0.25 s - 0.75 s), as well as by adding background noise, such as music (signal-to-noise ratio, SNR, within 5-15 dB), and babble (SNR varies from 10 to 20 dB). Noise signals were selected from the MUSAN corpus [136] and the room impulse responses to simulate reverberation from *openslr*. A second larger training set is constructed by further adding the recordings from VoxCeleb [7] after downsampling them to 8kHz, which adds up to approximately

Table 2.3 – EER (lower is better) obtained for the same system trained with different losses. The combination of triplet loss and cross entropy yields speaker-dependent representations.

Back-end	***Train loss***	***All***	***Cantonese***	***Tagalog***
PLDA	Triplet loss	29.51%	27.25%	31.94%
	cross entropy	20.81%	17.01%	24.72%
	Combined	14.10%	9.23%	19.09%
Adapted PLDA	Triplet loss	26.97%	24.47%	29.48%
	cross entropy	18.81%	14.52%	22.57%
	Combined	**10.51%**	**6.44%**	**14.67%**

14000 speakers, and, in that case, all recordings from 100 speakers are taken out of training data so as to serve as a validation set throughout training. The smaller train dataset is used by default, and models trained with the addition of VoxCeleb data will be indicated.

PLDA was employed for scoring trials after dimensionality reduction of embeddings using LDA. PLDA is trained only on embeddings from the *SRE* partition of the training data. The model adaptation scheme introduced in [145] is also utilized for PLDA to help overcome the domain shift observed across train and evaluation data due to different spoken languages and noise conditions. To do so, embeddings of provided unlabelled data in target languages are clustered, and clusters are used as speaker identities, which are then employed for training a second PLDA model. The final back-end is obtained by simply averaging the covariance matrices of the two PLDA models. We further highlight that reported results are obtained from models that achieved the minimal validation loss throughout training (F_{best}), and the evaluation data is only made available to the models to generate the reported metrics, not being used at the development phase.

The first experiment we perform consists of an ablation study to compare the performance of the proposed multi-task setting with cases such that only one of either speaker recognition or metric learning is employed as training task for the embedding model, F. One of our proposed models is trained with the combined loss L while equivalent systems are trained with cross entropy or triplet loss only. The attentive pooling strategy was used for the three models. Results on the SRE-16 evaluation data are shown in Table 2.3 in terms of EER, where one can notice that verification performance on the evaluation set is significantly improved when the multi-task training is employed when compared to systems trained with only one of the loss components. That is the case for Cantonese, Tagalog, and the pooled set of languages, as well as with or without PLDA adaptation, which leads us to conclude that both losses are complementary and should be used jointly, as is the case of the proposed multi-task training.

Table 2.4 – EER (lower is better) obtained using different pooling strategies to aggregate local descriptors into embeddings.

Back-end	***Pooling***	***All***	***Cantonese***	***Tagalog***
PLDA	Statistics	14.07%	8.95%	19.14%
	Attention	14.10%	9.23%	19.09%
	LSTM	15.98%	11.92%	19.46%
	Pretrain+LSTM	14.36%	8.95%	19.70%
Adapted PLDA	Statistics	10.87%	6.79%	15.00%
	Attention	10.51%	6.44%	14.67%
	LSTM	10.58%	6.59%	14.71%
	Pretrain+LSTM	**9.99%**	**5.90%**	**13.53%**

In Table 2.4, we compare the verification performance of the three considered pooling strategies on evaluation data in terms of EER. We further include a fourth system obtained by first training a model with attention only, and then fine-tuning the model after including the LSTM block in the pooling stage. One interesting finding is that the pooling methods benefit differently from PLDA adaptation, and the rank of best performers, not considering the pre-trained model, changes considerably once adaptation is used. This indicates the extra capacity in terms of number of parameters at the pooling level is used to learn domain-dependent patterns, and thus the performance boost achieved with adaptation is higher in such cases. Regarding pre-training, its benefit is observed both with and without adaptation, yielding the lowest EER obtained from single systems within our evaluation, and its effect in performance is further highlighted by the fact that a model with the same architecture but trained from scratch (LSTM) is not able to outperform the simpler attention-based pooling. We hence conclude the training difficulties introduced with the LSTM block outweigh the benefit of the extra capacity, and pre-training helps alleviate such training difficulties by providing the LSTM with features that are already meaningful from the start.

We proceed and further perform a comparison among our proposed method and well-known systems. For comparison purposes, i-vectors [1] are obtained as described in [146], i.e., by computing a 2048-Gaussians full covariance UBM using the unlabelled partition of NIST SRE 2016. Total variability analysis is performed on top of UBM's Baum-Welch statistics obtained for recordings from the *SRE* partition of the same training data as used for our models, resulting in a 600-dimensional extractor; i-vectors are finally reduced to a dimension of 200 with LDA. An x-vector system is also obtained using its Kaldi recipe [3] thus yielding embeddings of dimension 512 later reduced to 150 with LDA. The same training data as our systems was used for training x-vectors, and scoring of both i- and x-vectors is performed with PLDA employing the same model adaptation

Table 2.5 – Comparison of proposed systems with well-known baseline methods. Results correspond to verification EER (lower is better). " * " indicates that models trained with the larger training set are included.

	System	***All***	***Cantonese***	***Tagalog***
Baseline	x-vector [148]	11.90%	6.50%	16.30%
	x-vector+Attention [150]	10.21%	4.61%	14.15%
	x-vector	10.02%	5.82%	14.31%
	i-vector	12.68%	8.17%	17.25%
Proposed	Statistics	10.87%	6.79%	15.00%
	Attention	10.51%	6.44%	14.67%
	LSTM	10.58%	6.59%	14.71%
	Pretrain+LSTM	9.99%	5.90%	13.53%
	Pretrain+LSTM*	**9.53%**	**5.41%**	**13.22%**
Fusion	Proposed	8.41%	4.95%	11.93%
	Proposed*	8.04%	4.62%	11.52%
	Proposed + x-vectors	7.63%	4.23%	11.01%
	Proposed + x-vectors + i-vectors	**7.46%**	**4.14%**	**10.91%**

approach as described above. Results in Tables 2.5 and 2.6 correspond to the EER and $DCF10$ [147] for:

- A set of baseline systems on the first rows given by EERs obtained by above described i- and x-vectors. For the case of EER only, further results reported in the x-vector original paper [148] as well as in a more recent work in which x-vectors were augmented with structured soft-attention [149] are also included.
- Our proposed systems in the middle rows correspond to varying time pooling strategies. We further included our best system (Pretrain+LSTM) trained on the larger training set, which we indicate by the symbol " * ".
- Sum fusion performed at the score-level. We report results obtained by fusing scores of proposed systems only, including and not including the system trained on the larger training dataset, as well as proposed models combined with x-vectors, and also i-vectors.

As expected, the pre-trained system attains the lowest EER among single systems for the case of the pooled set of trials, while attaining a similar level of $DCF10$. The effect of the increase in the amount of training data is also observed given that performance in all metrics and for all evaluation conditions is improved once the amount of training data is higher. By fusing the considered systems we observe further improvement, thus reaching the lowest EER across all evaluation conditions.

Table 2.6 – Comparison of proposed systems with well-known baseline methods on SRE-16 evaluation set. Results reported in terms of $DCF10^{-2}$ and $DCF10^{-3}$ (lower is better). " * " indicates that models trained with the larger training set are included.

		DCF10^{-2}			**DCF10^{-3}**		
	System	***All***	***Cantonese***	***Tagalog***	**All**	**Cantonese**	**Tagalog**
Baseline	x-vector	0.6429	0.4769	0.7930	0.8464	0.6875	0.9556
	i-vector	0.7199	0.5636	0.8551	0.8845	0.7492	0.9689
Proposed	Statistics	0.7039	0.5540	0.8222	**0.8985**	0.7305	**0.9599**
	Attention	0.7078	0.5442	0.8349	0.9186	0.7148	0.9899
	LSTM	0.7175	0.5506	0.8491	0.9268	0.7445	0.9860
	Pretrain+LSTM	0.7118	0.4983	0.8429	0.9478	0.6868	0.9868
	Pretrain+LSTM*	**0.6859**	**0.4799**	**0.8170**	0.9214	**0.6747**	0.9766
Fusion	Proposed	0.6345	0.4522	0.7830	0.8956	0.6099	0.9756
	Proposed*	0.6256	0.4366	0.7756	0.8922	0.5936	0.9728
	Proposed* + x-vectors	0.5949	0.4154	0.7472	0.8587	0.5830	0.9632
	Proposed* + x-vec + i-vec	**0.5814**	**0.3997**	**0.7371**	**0.8484**	**0.5636**	**0.9547**

2.5 Conclusion

We evaluated the effectiveness of employing multi-task training procedures where maximum likelihood estimation is combined with metric learning so as to yield language- or speaker-dependent representations. Specific strategies are designed for each of the two tasks we are concerned with: LID and ASV. For the case of LID, evaluation on non-trivial conditions, such as short-duration of speech, confusing languages, and non-target test recordings corresponding to languages not represented within training data, indicates the robustness of the proposed approach relative to several well-known benchmarks relevantly improving both EER and C_{avg} in all considered evaluation settings. Specifically, improvements of 28.51%, 36.83%, and 39.88% were observed with respect to i-vectors when scoring is performed with PLDA, for three different evaluation conditions, namely, short-duration, confusing languages, and open set, respectively.

For the case of ASV, the proposed training method along with introduced sampling and regularization strategies were evaluated on the conditions introduced for NIST SRE 2016 in which we showed that: (i) models trained under the multi-task setting outperform those trained with single losses. (ii) sophisticated pooling strategies yield the best performance in our experiments, however, one has to deal with the additional training difficulties given by such systems. We found that pre-training with a simpler pooling method and finally training the complete model helps in that regard. (iii) our best systems yield equivalent or superior verification performance relative to well-known benchmark methods in the considered evaluation conditions. Fusion with baseline systems further improved final performance.

Chapter 3

Detecting threats as a means for robust voice biometrics

3.1 Preamble

This chapter is compiled from material extracted from the following three publications: Section 3.3 corresponds to [42], published in the *Proceedings of the IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. Section 3.4 covers [151], published in the *Proceedings of the Odyssey Speaker and Language Recognition Workshop (2020)*. Finally, in Section 3.5 we discuss [152] which was published in *Computer Speech & Language*.

3.2 Introduction

Artificial neural networks were observed to present properties that might be exploited by attackers. Specifically, the outputs of such models may vary greatly given subtle and often imperceptible variations in the inputs [97, 98]. So-called adversarial attacks [153] showed that one can leverage such inherent properties in order to fool trained models in such a way that attack instances are not detectable by human observers. The described vulnerability constitutes one of the major factors limiting the vast deployment of neural network-based technologies into safety-critical applications. While most of the recent work on the development of adversarial attacks and defenses have tar-

geted computer vision applications, it was recently shown that the same threat appears in the case of models tailored to speech processing, such as speech and speaker recognition [154, 155].

Besides the above described threat of adversarial attacks, which affect applications of neural networks in general, for the specific case of voice biometrics, even simpler attack strategies exist which can be applied to any type of model, neural network or otherwise. One example is replaying someone's voice, which is commonly referred to as a *physical attack* (PA). As a simple and realistic example, one could record someone's voice saying a command to gain unauthorized access to their portable devices. Another attack strategy, referred to as *logical attack* (LA), can be defined using synthesized speech. In fact, recent advances in conditional generative models (e.g., Wavenet [72]) may be used to this end for both text-to-speech or voice conversion settings. The potential consequences of such vulnerabilities are tremendous and range from financial loss to undue incrimination.

Given that these described threats limit the use of high performance systems in real applications, a popular research direction in recent years has been to design countermeasures against such attackers. For the specific case of speaker recognition and voice authentication, for instance, recent spoofing detection challenges [38, 39, 9] were introduced with the goal of pushing forward the state-of-the-art in attack detection. In general, countermeasures can be grouped into two categories: *defense* and *detection* methods. While defense techniques try to either improve the model robustness or suppress the success rates of attacks, detection methods take a different approach and attempt to determine if the input is genuine or was somehow manipulated. While both directions are promising, we focus on the detection approach, as it allows service providers to know when their systems are under attack.

More specifically, we are interested in detection approaches that operate in an end-to-end manner. Here, end-to-end is referred to systems comprised of a single component able to receive audio as input (or general purpose audio representations) and output scores indicating how likely it is that the input has been tampered. The main advantage of end-to-end settings over conventional pipeline-based methods that rely on several internal blocks, is simplicity. End-to-end systems allow for inference schemes that require a single forward pass, while pipeline methods usually have to deal with several blocks, each one with their specific challenges and limitations.

We further remark that, even though the data released for the challenges mentioned above are generated making sure that speakers and attack types vary across train, development, and evaluation

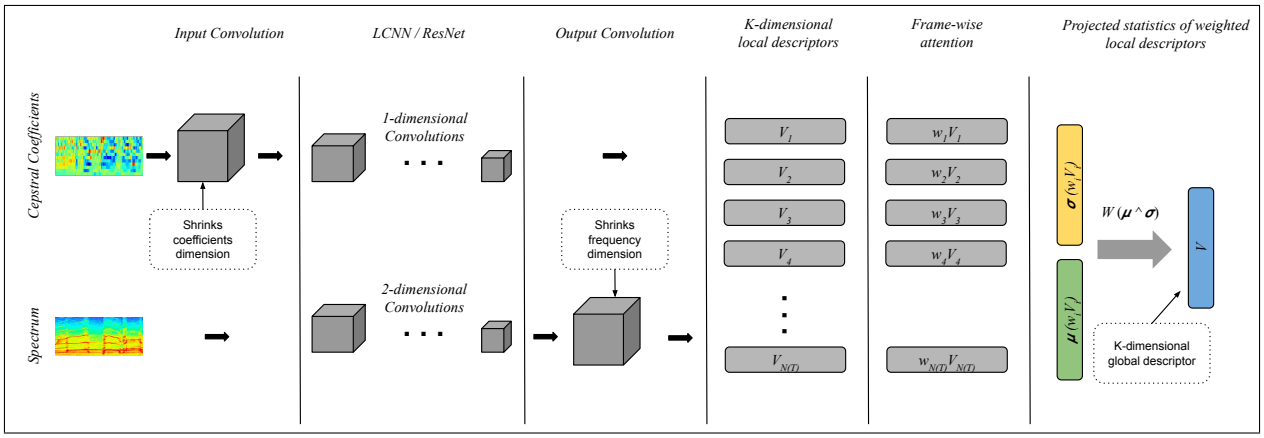


Figure 3.1 – Cepstral Coefficients are first shrunk through a convolutional layer and then fed into a LCNN29 modified with 1-dimensional time convolutions. Spectral features on the other hand are directly fed into a standard time-spatial LCNN9, and have the frequency dimension shrunk later on. The number of channels K of the last convolutional layers yields the dimensionality of the final representation V, which is given by a linear projection of concatenated statistics of weighted local descriptors.

datasets, the developed detection systems rely on the strong (and unrealistic) assumption that training and testing data are *identically distributed* so that the same general attack strategy (LA or PA) will appear on both training and testing data. By doing so, varying models, architectures, and input features are used for either PA or LA attacks. This strategy-specific configuration, however, is not aligned with practical, real-life scenarios where the attack strategy is not known *a priori*. We thus also seek to address such limitation and propose attack-agnostic detection strategies.

The remainder of this chapter will be organized as follows: Section 3.3 describes our first proposal of an end-to-end scheme where light-weight convolutional models are designed along with a training strategy resulting in strong performance on both LA and PA cases. In Section 3.4, we scale the approach introduced in Section 3.3 to larger models by employing data augmentation strategies that conserve the artifacts that enable detection of attackers. We finally introduce an attack-agnostic detection approach in Section 3.5, where a conditional modeling approach is proposed so that a single model (or set of models) can be used to detect both physical and logical attacks.

3.3 An end-to-end setting for spoofing detection

3.3.1 Model and training

As a countermeasure to attacks targeting voice biometrics systems, we propose an end-to-end setting for detection of either logical or physical attacks. We once more remark that by end-to-end we mean that the proposed models are able to output a score indicating how likely it is that its input is an attack when inputs correspond to general purpose time-frequency representations of speech data. This is unlike common settings in which features and the classifier/scorer are obtained separately. Modified light convolutional neural networks (LCNN) [73], originally proposed for face recognition, are employed in our case along with an attention layer, aimed at enabling the model to process varying length recordings while focusing on specific portions of inputs. The light models are used given the limited amount of train data made available for the considered evaluation setting, which might result in overfiting and degradation of generalization performance. We further used two distinct LCNN variations with increasing depths, and observed that models yielding the higher performance depend on the choice of input features. Evaluation is carried out under the ASVspoof 2019 challenge conditions, which provides training and development data partitions for both logical attacks generated with text-to-speech and voice conversion systems, as well as physical attacks corresponding to replayed recordings under various simulated configurations.

3.3.1.1 Speech representation

We employ two sets of general purpose time-frequency representations of speech. Both spectral representations and cepstral coefficients are considered, and different modeling strategies are used in each case, as illustrated in Figure 3.1. Specifically, we report results for models trained on top of the power spectrum (Spec), the product spectrum (ProdSpec) introduced in [156] and later used in [157] as a countermeasure for replay spoofing detection, linear frequency cesptral coefficients [100] (LFCC), and the constant Q cesptral coefficients (CQCC) [158]. In all the cases, features are obtained with a short-time Fourier transform with length 512 using a 20ms Hamming window with 50% overlap. The end result corresponds to 257 frequency bins for spectral representations and 30 coefficients stacked with *delta* and *delta-delta* coefficients for the LFCC and CQCC cases, thus resulting in a dimensionality of 90.

3.3.1.2 Extraction of local descriptors

The first stage within our modelling pipeline following feature extraction corresponds to mapping a given recording into a set of local descriptors $V_i \in \mathbb{R}^K$, i.e., a set of vectors representing parts of inputs across the time dimension. To obtain such set of descriptors, we consider recent findings in [70] showing that convolutional neural networks are well-suited for time dependency modelling while avoiding common training issues observed in the case of recurrent neural networks (e.g., vanishing/exploding gradients). Namely, light convolutional neural networks [73] are employed given their low parameter count with respect to other well known convolutional models, such as VGGs [74] or ResNets [4].

LCNNs employ the so-called max-feature-map (MFM) activation function, which corresponds to an alternative to the commonly used maxout operator, i.e., when the neuron's output is given by its greatest input. When using MFM activations on the particular case of convolutional layers, the *max* operator is taken element-wise on splits of same size taken from feature maps. Two schemes are proposed in [31] so that either feature maps are split into two or three groups, resulting in 1/2 or 2/3 of the number of input feature maps after MFM is applied, referred to as MFM-2/1 and MFM-3/2, respectively. An illustration of the latter is depicted in Figure 3.2.

We further apply some modifications to LCNN depending on which type of speech representation is used. For the case of cepstral coefficients, we include an input convolutional layer responsible to shrink the coefficients dimension to 1, and proceed with a modified 29-layered LCNN (LCNN29) which performs convolutions on the time dimension only. For spectral features on the other hand, we directly input examples into a standard spatio-temporal LCNN containing 9 layers (LCNN9), and an output convolutional layer is then employed so as to shrink the frequency dimension to 1. In both the cases, we then end up with a set of vectors of dimension K matching the number of channels of the last convolutional layer. K was considered a hyperparameter and we found the value of 128 to yield good performance across all cases considered herein.

The same frame-wise attention scheme introduced in Chapter 1 is considered here with the goal of combining local descriptors V_i into a single global descriptor V. We briefly re-introduce the operations corresponding to this component with a matching notation within the chapter for clarity. Moreover, we slightly modify the pooling operation described originally since here we include second-order statistics of representations besides the average. More specifically, given an utterance

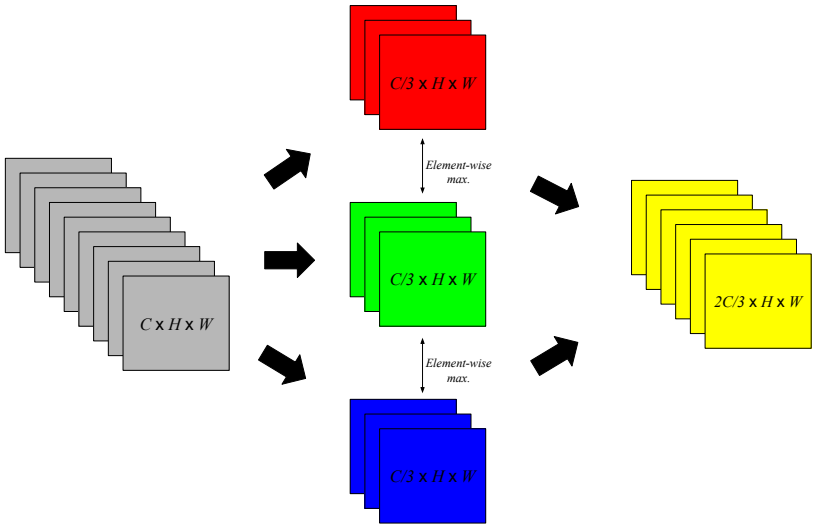


Figure 3.2 – Application of a 3/2 MFM activation on C channels of dimension $H \times W$. Input set of channels is split in 3 groups, the minimal element-wise element in each triplet is removed.

of arbitrary length, it will be mapped to a fixed-dimensional vector. Consider $V_{1:N(T)}$ as a set of vectors corresponding to the outputs of a given neural network for some input, where T is the duration of the input and the number of local descriptors is represented as a function N(T). The parameters of the attention layer are the components of a linear transformation A, shared across all time-steps i, and applied to each V_i resulting in a set of scalars $a_{1:N(T)}$, according to:

$$a_i = \tanh\left(AV_i\right). \tag{3.1}$$

A set of normalized weights summing up to 1 is thus obtained through the softmax operator:

$$w_i = \frac{e^{a_i}}{\sum_{i=1}^{N(T)} e^{a_i}}, \tag{3.2}$$

and component-wise first- and second-order statistics of weighted local descriptors $w_i V_i$ are computed to finally yield the global descriptor V:

$$V = \mathrm{cat}[\mu(w_i V_i), \sigma(w_i V_i)]. \tag{3.3}$$

The described model is represented in Figure 3.1 for each of the considered input representations. Global descriptors are thus employed for final binary classification, which is performed in our case with a single fully-connected layer.

3.3.1.3 Training

Training is carried out with SGD using mini-batches of size 16 and 32 for the cases of spectral and cepstral coefficients, respectively. For each model and features combination, a grid search is performed for the best values of both learning rate and weight decay coefficient, and we found the values of 0.001 and 0.00005 to yield the best results across all considered cases on a validation set held out of training, containing randomly selected 100 and 1000 recordings corresponding to clean and attack examples, respectively. Momentum is also employed with its coefficient set to the default value of 0.9.

Each training example is pre-processed such that if it is shorter than 10 seconds, it is repeated up to that length. In the case it is longer, we select a random 10 seconds segment. Moreover, since all the examples within a mini-batch are exactly 10 seconds long, we randomly trim all the examples to be within 3-10 seconds. This is done so as to generate diversity in the samples presented to the model and artificially augment the size of the train dataset, since this will make training examples different each time they are sampled. Additionally, given the imbalance in the number of examples corresponding to clean and attack recordings, we oversample clean examples such that each mini-batch is balanced, i.e., we sample pairs of training examples by sequentially iterating over the set of recordings corresponding to attacks. To sample clean recordings, we pick the recording with index $k = j \bmod N_{clean}$, where $j \in \{0, 1, ..., N_{attack} - 1\}$ is the index of attack recordings, and N_{clean} and N_{attack} are the number of clean and spoofing training recordings. The sequence of indices j is further randomized to provided diverse mini-batches at each epoch. Training proceeds up to convergence of the loss measured in the validation set held out of training, which takes approximately 12 hours in a single NVIDIA Titan X GPU[1].

3.3.2 Evaluation

The proposed methods are evaluated using the data introduced for the ASVspoof 2019 challenge. Two types of attacks are considered: logical and physical access attacks, corresponding to synthetic speech and replayed recordings, respectively. Logical access attacks were created using both voice conversion and text-to-speech systems, while replay attacks are simulated from clean recordings

[1]Code is available at: `https://github.com/joaomonteirof/e2e_antispoofing`

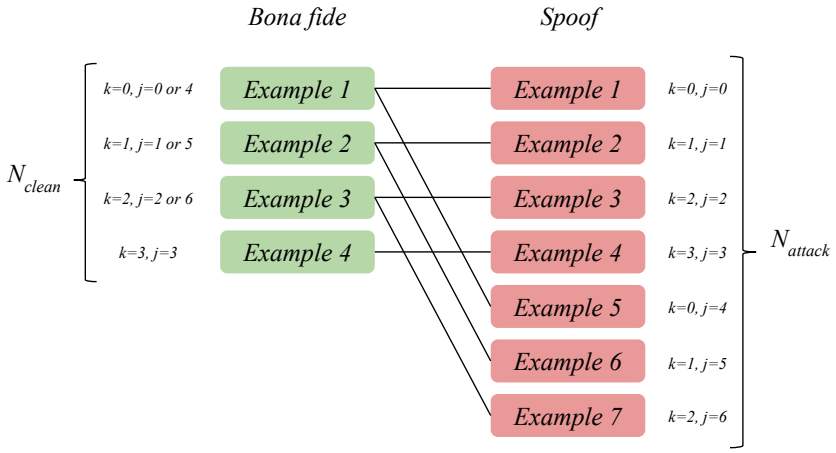


Figure 3.3 – Sampling strategy for constructing mini-batches. Clean examples are sampled several times per epoch so as to ensure mini-batches are balanced.

considering exhaustive combinations of 3 room sizes, 3 distances to the microphone, and 3 levels of reverberation. Further details on the dataset are discussed in Chapter 1.

In this set of evaluations, benchmark systems are those provided by the ASVspoof 2019 organizers. These consist of Gaussian Mixture Models (GMM) classifiers trained on frame-level features, i.e., two independent GMMs are trained on each of considered classes (genuine or spoofing). Scores are given by the difference of log-likelihoods provided by the two generative models for a given test instance. Reference performance is provided for GMMs trained on both LFCC (LFCC-GMM) and CQCC (CQCC-GMM). Additionally, we included our own implementation of CQCC-GMM, and further evaluated an i-vector system [1], in which case, a 512-Gaussians full covariance universal background model (UBM) along with a 400-dimensional i-vector extractor are trained on the training partition using CQCC features. I-vectors further have their dimensionality reduced to 150 using linear discriminant analysis. Scoring is finally performed with a probabilistic linear discriminant analysis (PLDA) classifier [106].

Evaluation metrics verified on the development partition of ASVspoof 2019 are reported in Tables 3.1 and 3.2 for the case of logical and physical attacks, respectively. Results correspond to the equal error rate (EER) and the normalized minimum tandem detection cost function (min-tDCF). EER consists of the value of the *miss rate*, given by the fraction of miss-classified clean test recordings with respect to the number of trials corresponding to genuine samples, at the threshold in which it matches the *false alarm rate*, i.e., the ratio between the number of miss-classified spoofing

trials and the number of spoofing test recordings. The min-tDCF, in turn, was recently introduced in [159], and was designed especially for the evaluation of spoofing detection countermeasures when used alongside a speaker recognizer. The interested reader can refer to [9] for a detailed description regarding both evaluation metrics. We further highlight that the development set is not used during training in any way. A validation set is rather constructed by taking recordings out from train data, which is then used for deciding when to stop training as well as for fine tuning of hyperparameters. Only models completely trained have access to development data.

Results for the challenge benchmark systems as well as our own evaluated baselines are presented along with only the instances of our systems which were able to outperform some of the considered baselines. For the logical access case, systems based on LFCC and CQCC achieved relevant performance. Specifically for the case of LFCC, a relative improvement of approximately 50% in terms of both EER and min-tDCF is observed with respect to the best compared system (CQCC-GMM). In the case of physical presentation attackers, three of our proposed settings were able to perform substantially better than the considered benchmarks. Most notably, LCNN9 trained on top of product spectral features attained detection metrics one order of magnitude lower than those of the considered benchmarks.

Additional results are reported in Table 3.3 for the evaluation partition for the case of physical access attacks. Even though a mismatch is observed between training and evaluation data in terms of attack generation conditions, i.e., different channel and reverberation settings were employed to generate attacks in each case, our proposed model's detection performance is substantially better than all the tested benchmarks. We attribute the performance boost observed by the adoption of LCNNs+attention (compared to GMM classifiers and i-vectors+PLDA) to the better handling of temporal dependencies. It is well-known that convolutional layers over the time dimension yield long-term temporal modelling through depth, i.e., outputs depend on far apart portions of the input when convolutional layers are stacked. Moreover, the attention layer represents an effective means for the model to assign importance to portions of the input in a data-driven fashion. GMMs and the UBM for the i-vectors extraction, however, completely disregard temporal modelling by assuming frame-level features are independent. Additionally, the low parameter count offered by the adoption of LCNNs along with the efficient handling of time pooling using a simplistic attention layer, as opposed to employing recurrent models to this end, help on circumventing the rather limited amount of training data.

Table 3.1 – EER and min-tDCF for logical access attacks on development partition. Both scores are better when closer to 0.

	Feature-Model	EER(%)	min-tDCF
ASVspoof benchmarks [9]	LFCC-GMM	2.71	0.0663
	CQCC-GMM	0.43	0.0123
Internal baselines	CQCC-GMM	0.39	0.0110
	i-vector-PLDA	0.70	0.0210
Proposed	CQCC-LCNN29	1.07	0.0321
	LFCC-LCNN29	**0.20**	**0.0048**

Table 3.2 – EER and min-tDCF for physical access attacks on development partition. Both scores are better when closer to 0.

	Feature-Model	EER(%)	min-tDCF
ASVspoof benchmarks [9]	LFCC-GMM	11.96	0.2554
	CQCC-GMM	9.87	0.1953
Internal baselines	CQCC-GMM	9.70	0.1842
	i-vector-PLDA	9.17	0.2310
Proposed	CQCC-LCNN29	2.93	0.0752
	Spec-LCNN9	2.00	0.0488
	ProdSpec-LCNN9	**0.87**	**0.0232**

Table 3.3 – EER and min-tDCF for physical access attacks on evaluation partition. Both scores are better when closer to 0.

	Feature-Model	EER(%)	min-tDCF
ASVspoof benchmarks	LFCC-GMM	13.54	0.3017
	CQCC-GMM	11.04	0.2454
Proposed	**ProdSpec-LCNN9**	**1.66**	**0.0445**

More importantly, our experimental results further show the best choice of data representation and modelling architectures to be dependent on the type of attack, which might represent a threat from the perspective of a real-world application of such countermeasures. If we consider that attackers evolve over time, a detector can have its performance degraded if attackers are generated using a novel strategy, thus being able to generalize across diverse spoofing strategies should be considered, which we tackle later on within this Chapter in Section 3.5.

3.4 Scaling end-to-end detection to larger models via artifact-preserving data augmentations

3.4.1 Augmentation approach

Data augmentation can be defined as the process of increasing the amount and diversity of existing training data. It is performed with two goals: (i) as a form of regularization strategy to improve the performance measured on test data by directly increasing the number of training examples, and (ii) to compensate possible mismatches between the training and testing conditions, which can be seen as a way of inducing robustness across varying noise conditions. For the case of speech processing applications, augmentation is often performed by supplementing a dataset with similar data, artificially created by introducing additive and convolutive noises, for instance. Augmentation schemes became common practice in speech-based applications, such as speaker and speech recognition.

In this section, we perform evaluation using the data introduced for the ASVspoof 2019 challenge [9]. Specifically, we are particularly interested in being able to introduce perturbations in the signals in such a way that the artifacts related to spoofing attacks are preserved, while at the same time the artifacts we introduce with the augmentation process are different from those indicative of attacks. For instance, we empirically found that, depending on the type of attack strategy, this can be achieved through speed perturbations, as well as bandpass filtering, and those offer the extra advantage of not requiring any external corpora to be used, complying with evaluation guidelines of popular challenges such as ASVSpoof. We further observed simple online strategies, such as feeding models with random continuous chunks of signals (as opposed to presenting the complete audio recording) to be effective as an augmentation strategy. The approach we evaluate to increase the amount of training data, both bona fide as well as spoofing examples, is to apply speed perturbations, in which case we use perturbation factors of 0.9 and 1.1, to low pass filter with a cut-off frequency of 3.8 kHz, and to high pass filter with the same cut-off frequency set to 3.8 kHz. The described procedure is summarized in Figure 3.4. By doing so, we are able to increase the size of our corpus by five times with respect to the original training data. Along with that, we perform additional online transformations which will be further described once we detail how mini-batches are constructed at training time.

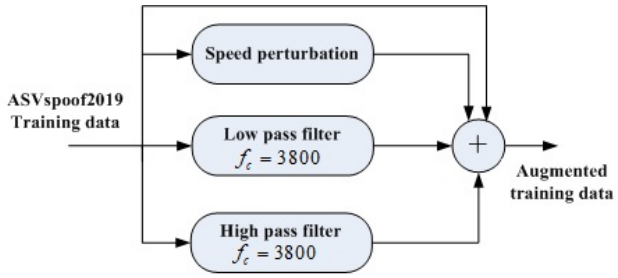


Figure 3.4 – Data augmentation via speed perturbation, low pass, and high pass filtering of ASVspoof 2019 training data.

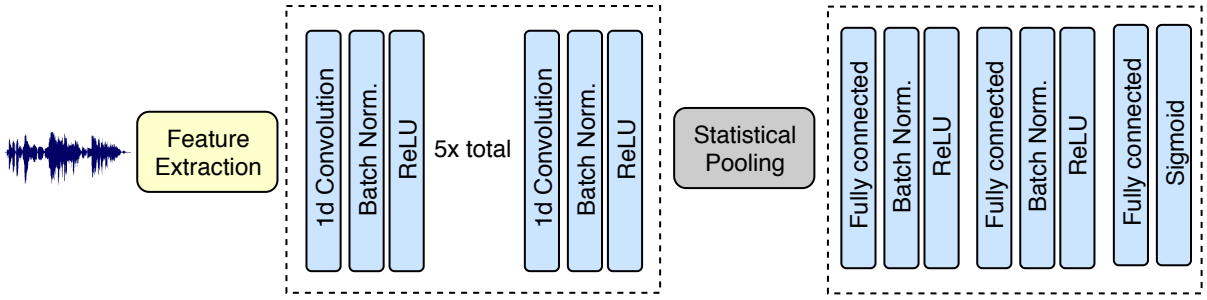


Figure 3.5 – End-to-end detection of spoofing attacks. All convolutions maintain the time resolution. Statistical pooling corresponds to concatenated mean and variance, obtained across the time dimension. N represents the dimension of feature vectors, which correspond to 90 and 257 for LFCC and ProdSpec, respectively.

3.4.2 Model description

Given the supplemental data we introduced, we now implement our model using larger architectures instead of the LCNN architecture used previously in Section 3.3. Namely, the TDNN architecture commonly used for speaker verification is employed. Moreover, we modified the standard TDNN architecture so that pre-activation batch normalization is performed at each convolution and fully-connected layer. This procedure was introduced in [160] and shown in [124] that classification accuracy is improved when batch normalization is applied prior to activation. A summary of the employed architecture is shown in Table 1.1 in Chapter 1 while the complete end-to-end score computation used specifically for the spoofing detection case is depicted in Figure 3.5.

3.4.3 Training details

Training follows the setup introduced in [42] and described in Section 3.3; i.e., with SGD performed to minimize the binary cross entropy loss in a standard binary classification setting. Mini-batches of effective size of 16 (balanced) are used for both of the cases of ProdSpec and LFCCs. The learning rate and weight decay coefficients were set to 0.001 and 0.00005, and performance is monitored throughout training with a validation set we created by removing 100 and 1000 randomly selected recordings corresponding to clean and attack examples, respectively. Momentum is also employed with its coefficient set to the default value of 0.9.

We further perform extra data augmentation strategies in an online manner, which conveniently also helps to deal with the varying duration across recordings. Those correspond to signal manupulations discussed in the previous Section. That is, each training example is pre-processed such that if it is shorter than 10 seconds, it is repeated up to that length. In the case it is longer, we select a random 10 second segment and this process is repeated whenever a given example is sampled. Since we employ that procedure and ensure all examples within a mini-batch are exactly 10 seconds long, we randomly trim all examples to be within 3-10 seconds, where the exact duration is sampled uniformly from that range for every mini-match. This is done so as to generate diversity in the samples in terms of duration since trained models are expected to be able to perform detection given samples of arbitrary duration at testing time. We also employ the sampling strategy described previously and illustrated in Figure 3.3 such that each mini-batch is balanced. Training proceeds up to a fixed budget of epochs or convergence of the loss measured in the validation set held out of training. This took approximately 12 hours in a single NVIDIA Titan X GPU.

3.4.4 Evaluation

Evaluation on both development and test partitions of ASVSpoof 2019 is performed in terms of EER and min-tDCF. Since we created our validation sets by removing data from the training partition, the development data is not used in any way during training nor for early stopping or hyperparameter selection. All results are reported for the best performing model we could observe across training in each data partition during a fixed budget of 100 training epochs. In addition to our proposed systems, we further report the performance of baselines for comparison, which include: (i) those provided by the ASVspoof 2019 organizers, which consist of GMM-based classifiers trained

Table 3.4 – The min-tDCF and EER(%) results for PA task on the development set. Lower values are better.

	System	***EER(%)***	***min-tDCF***
ASVSpoof baselines [9]	CQCC-GMM	9.87	0.1953
	LFCC-GMM	11.96	0.2554
Our baselines	CQCC-GMM	9.70	0.1840
	i-vector/PLDA	9.17	0.2310
Ours	TDNN	2.30	0.0690
	TDNN+SP	0.87	0.0299
	TDNN+Filt.	**0.74**	**0.0244**
	TDNN+SP+Filt.	1.34	0.0439

on frame-level features, reference performance is provided for GMMs trained on both LFCC as well as CQT-based cepstral coefficients (CQCC) [114]. And (ii) our own baselines obtained using the Kaldi toolkit [161] corresponding to a GMM classifier on top of CQCCs, as well as an i-vector system [1] scored with a probabilistic linear discriminant analysis (PLDA) classifier [106]. Both the GMM- and i-vector-based systems use 512-Gaussian components for training bonafide/spoof models and the universal background model (UBM), respectively. I-vectors were finally obtained with total-variability analysis performed on top of UBM's supervectors, yielding 400-dimensional representations of audio clips, using CQCC features. Prior to training PLDA, i-vectors further have their dimensionality reduced to 150 using linear discriminant analysis.

Tables 3.4 and 3.5 present the EER and min-tDCF scores obtained by the baselines, as well as our proposed systems for the PA and LA tasks, respectively, considering the development data. Proposed models are indicated by *TDNN*, *TDNN+SP*, *TDNN+Filt*, and *TDNN+SP+Filt*, which corresponds to: models trained with the original training corpus and the previously described online augementation strategy performed while assembling mini-batches, offline augmented train data with speed perturbation (SP) only, bandpass filtering (Filt) only, and augmented data with both speed perturbation and bandpass filtering.

In both PA and LA cases, end-to-end approaches are able to outperform more standard GMM-classifers, as well as our i-vector/PLDA system in terms of both EER and min-tDCF. Moreover, specifically for replay attacks as presented in Table 3.4, we observe that adding speed perturbations yielded an improvement in the detection performance. However, once the complete set of augmentations is employed, we actually observe a degradation in performance. Nevertheless, as will be further discussed, the opposite is observed in the case of evaluation data (c.f. Table 3.6),

Table 3.5 – The min-tDCF and EER(%) results for the LA task on the development set. Lower values are better.

	System	***EER(%)***	***min-tDCF***
ASVSpoof baselines [9]	CQCC-GMM	0.43	0.0123
	LFCC-GMM	2.71	0.0663
Our baselines	CQCC-GMM	0.39	0.0104
	i-vector/PLDA	0.70	0.0211
Ours	TDNN	0.07	0.0015
	TDNN+SP	**0.04**	0.0012
	TDNN+Filt.	0.08	**0.0011**
	TDNN+SP+Filt.	0.08	0.0018

Table 3.6 – The min-tDCF and EER(%) results for PA task on the evaluation test set. Lower values are better.

	System	***EER(%)***	***min-tDCF***
ASVSpoof baselines [9]	CQCC-GMM	11.04	0.2454
	LFCC-GMM	13.54	0.3017
Our baselines	CQCC-GMM	11.16	0.2478
	ivector/PLDA	10.18	0.2687
Ours	TDNN	4.46	0.1337
	TDNN+SP	2.18	0.0777
	TDNN+Filt.	1.84	0.0611
	TDNN+SP+Filt.	**1.77**	**0.0597**

which suggests that the mismatch across training and development data is much smaller than that across training and evaluation data. The observed performance degradation is thus an indication of reduced overfitting to train data, which is beneficial given the improved generalization observed when evaluation data is used to assess performance.

Performance on the evaluation data is reported in Tables 3.6 and 3.7 for PA and LA attacks, respectively. In this case, we once more observe the proposed end-to-end approaches outperforming the considered baselines. However, for PA attacks specifically, we now observe that the use of more data augmentation consistently implies improved detection performance, which confirms our hypothesis in that the more diverse train data introduces some sort of regularization and avoids overfitting to the types of attack strategies utilized to create train data.

We further stress the observation made above considering the mismatch in results observed between development and evaluation data for all considered systems, baseline or otherwise, for the specific case of LA attacks. In fact, for that particular evaluation case, most systems reach a strong detection performance on development data, while observe a more severe degradation when we

Table 3.7 – The min-tDCF and EER(%) results for LA task on the evaluation test set. The lower the values of min-tDCF and EER the better is the performance.

	System	***EER(%)***	***min-tDCF***
ASVSpoof baselines [9]	CQCC-GMM	9.57	0.2366
	LFCC-GMM	8.09	0.2116
Our baselines	CQCC-GMM	8.91	0.2157
	ivector/PLDA	16.55	0.4201
Ours	TDNN	**7.00**	**0.1653**
	TDNN+SP	8.89	0.1769
	TDNN+Filt.	8.22	0.1769
	TDNN+SP+Filt.	7.12	0.1674

move to evaluation data. This is due to the different approaches used to create attacks so as to compose both data partitions; i.e., generative approaches used to create the development partition are similar to those used to create train data. The online augmentation helps in this regard, working as a regularization strategy and enabling better generalization to the new conditions introduced with the evaluation data. However, the artifacts introduced with speed perturbation as well as bandpass filtering appear to overlap with those introduced by the speech synthesis approaches utilized in order to create the attacks, and thus yield a slight degradation in performance, but not due to overfitting in this case. In fact, we hypothesize the opposite happens for LA, since the augmented genuine samples appear alike to synthetic attackers, an effect similar to that of label noise is introduced, yielding a too strong regularization strategy for this particular evaluation. The end-to-end models we trained are nevertheless able to outperform considered baselines by a large amount.

3.5 Attack-agnostic strategy to detect both logical and replay attacks

3.5.1 Proposed Model

Here, we describe an approach aimed at being able to detect both LA and PA attacks simultaneously, without any prior knowledge as to which type of strategy was employed in order to create attack signals. Figure 3.6 illustrates the strategy we propose in order to be able to detect attackers generated using different strategies. In that case, inputs x_{LA}, x_{MIX}, and x_{PA} correspond to features obtained from a given audio sample. We discuss the selection of the feature space in each case in

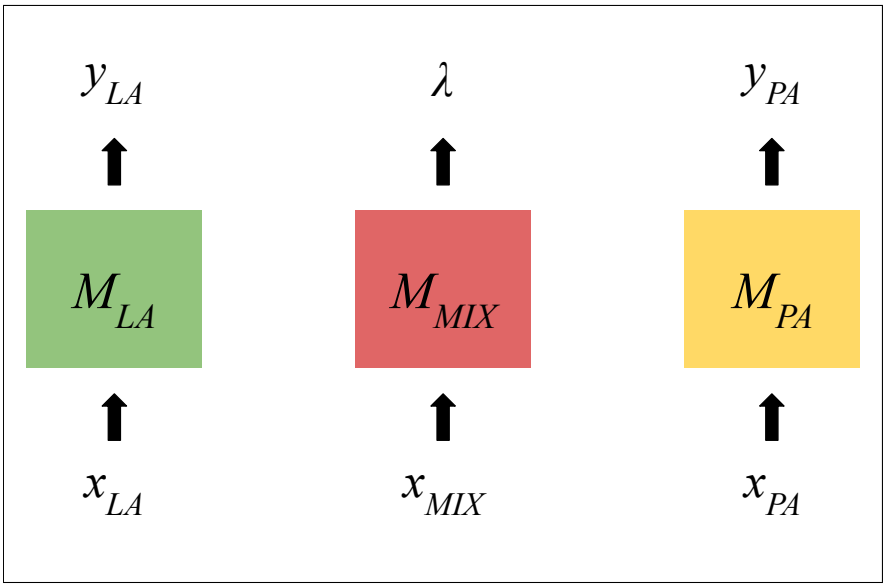


Figure 3.6 – General scheme illustrating the proposed ensemble strategy.

Section 3.5.5. Each of M_{LA}, M_{MIX}, and M_{PA} are such that $M : X \rightarrow [0, 1]$, i.e., speech features are mapped into a score in $[0, 1]$. The output $\lambda = M_{MIX}(x_{MIX})$ is then used to compute a convex combination of scores y_{LA} and y_{PA}:

$$y = \lambda y_{LA} + (1 - \lambda) y_{PA}, \tag{3.4}$$

and score y can be thus employed to jointly train the whole ensemble as a binary classifier of genuine *vs.* attack samples in a single-shot, considering both LA and PA examples are represented within train data.

The underlying assumption when doing so is that there are combinations of type of features and models which are better tailored for each of the considered LA or PA attack strategies. Further, M_{MIX} would be able to decide which of M_{LA} or M_{PA} should be given more importance for each input audio example.

Each ensemble component, as described in Figure 3.6, is required to map features extracted from an audio sample of arbitrary length into a score $\in [0, 1]$. We do so by using the same approach described in Section 3.3, i.e., by splitting the mapping into three stages: (i) from audio to local descriptors, (ii) temporal pooling, i.e., from a set of local descriptors to a global descriptor, and (iii) a mapping from a global descriptor to a score, as illustrated in Figure 3.7.

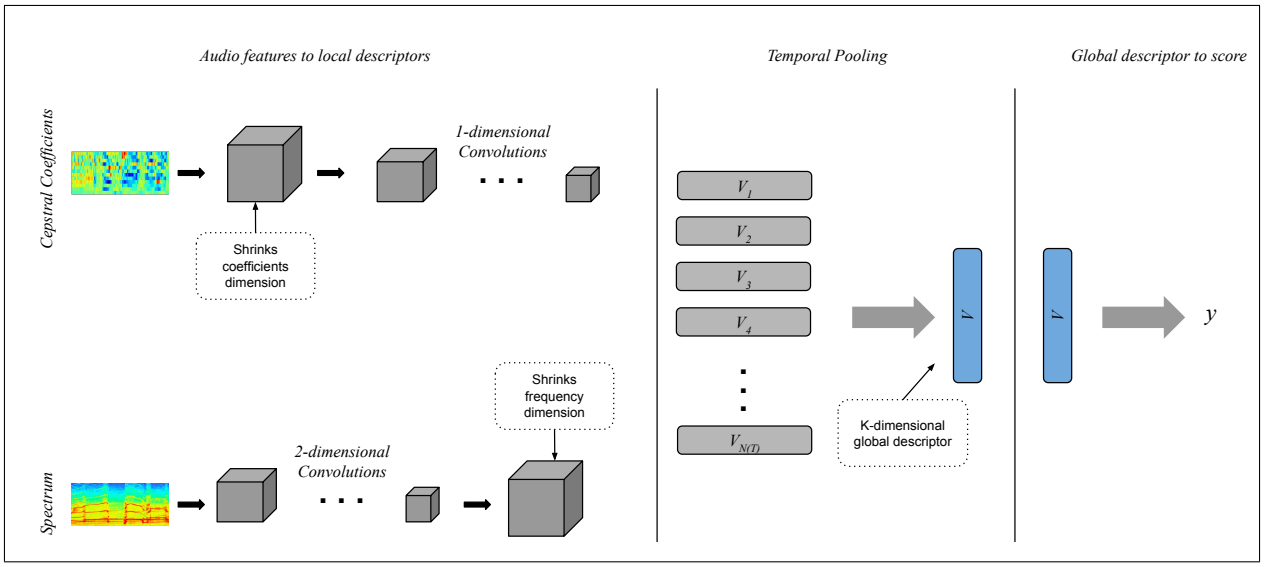


Figure 3.7 – Cepstral coefficients are first shrunk through a convolutional layer and then fed into a stack of 1-dimensional time convolutions. Spectral features on the other hand are directly fed into a set of time-spatial convolutions, and have the frequency dimension shrunk later on. The number of channels K of the last convolutional layers yields the dimensionality of the final representation V, finally projected into an output score $y \in [0, 1]$. The number of local feature vectors N is a function of the number of input frames T.

3.5.2 Training

We jointly train M_{LA}, M_{MIX}, and M_{PA}, and in order to do so, we consider the standard maximum likelihood setting through the minimization of the binary cross entropy loss (BCE) for the predicted scores y. Moreover, we empirically found that further including individual losses so as to ensure each model performs well on its own significantly accelerates convergence, and further consider such losses measured for y_{LA} and y_{PA}. The binary cross entropy loss for a generic score $s \in [0, 1]$ and the true label $l \in \{0, 1\}$, is given by:

$$BCE(s, l) = -l \log(s) - (1 - l) \log(1 - s). \tag{3.5}$$

The binary cross entropy loss discussed above is employed to define the training losses of M_{LA}, M_{PA}, and the ensemble-level loss. If we assume the true label to be such that $l = 0$ for genuine audio examples and $l = 1$ in the case of attacks, the binary cross entropy is given by $BCE(y_{LA}, l)$, $BCE(y_{PA}, l)$, and $BCE(y, l)$, respectively. Moreover, we want to enforce a particular behavior in M_{MIX} such that a higher importance is assigned to M_{LA} if the input corresponds to an LA attack. In this case, λ would be closer to 1. Alternately, y_{PA} is to be assigned higher importance for PA

attacks, thus λ should be close to 0. Hence, we define the total training loss L as:

$$L = BCE(y, l) + BCE(y_{LA}, l) + BCE(y_{PA}, l) + BCE(\lambda, l^*), \tag{3.6}$$

where l^* will be 1 in case the input corresponds to an LA attack, 0.5 for the case it is a genuine example, and 0 for the case where a PA attack is presented.

Two versions of LCNNs are employed, such that a deep LCNN-29 is used in the 1-dimensional case while a LCNN-9 is employed for spectral representations. For the ResNet case, ResNet-18 was the variation chosen given that we observed deeper versions to strongly overfit to train data. Training and development data are composed of pooled examples corresponding to LA, PA, and genuine utterances. Adam [162] is the chosen update rule using mini-batches of size 16 and 32 for the cases of spectral and cepstral coefficients, respectively, when LCNNs are used, and a size of 8 is employed for ResNets.

Mini-batches are sampled using the same approach described in Section 3.3. In fact, as discussed previously, a significant imbalance on the number of examples corresponding to genuine and attack recordings is observed in the considered train data. As such, in our initial experiments, in which case naive random data samplers were employed, models quickly converged to the condition where they simply predict every input is an attack. While several sampling strategies were proposed to deal with imbalanced data (c.f. [163]), we found oversampling genuine examples and ensuring each mini-batch is balanced to effectively and efficiently fix that issue. Moreover, each sampled training example is pre-processed such that if it is shorter than 8 seconds, it is repeated up to that length. In the case it is longer, we select a random 8 seconds continuous segment. Since all examples within a mini-batch are exactly 8 seconds long, we randomly trim every mini-batch to be within 3-8 seconds before feeding them to the models. This artificially augments the size of the train dataset since this will make training examples different each time they are sampled.

A linear learning rate warm-up is employed in the first 1000 training iterations, and the same exponential decay as in [48] is employed after that. Label smoothing [81] is further employed to avoid overfitting. A grid search is performed for the best values of both learning rate and weight decay coefficient. We found the values of 0.001 and 0.00005 to yield the best results across all considered cases on development data held out of training. Adam's β_1 and β_2 are set to 0.98 and

0.90, respectively. Training proceeds up to convergence of the loss measured in the validation set. Training is performed on a single NVIDIA Titan X GPU.

3.5.3 Experimental Setup and Evaluation

Once more, the approach discussed herein is evaluated under the conditions introduced for the ASVspoof 2019 challenge. The development set is used for hyperparameter tuning and the performances reported in all tables correspond to the models that achieved the best performance in development data. Evaluation data is used at final evaluation only. Different modelling strategies are explored depending on the type of speech features used to represent the data, as illustrated in Figure 3.7. Temporal convolutions are performed for the case of cepstral coefficients while spectral representations are processed by 2-dimensional frequency-time convolutions. We thus follow the findings reported in [42] and use both linear frequency cesptral coefficients [100] (LFCC), showed to be well suited for detecting LA attacks, and the product spectrum (ProdSpec) introduced in [156] and later used in [157] as a countermeasure for replay spoofing detection. The modified group delay [99, 164] cepstral coefficients were further evaluated as an attempt to check whether features computed on top of the phase spectrum would be effective on discriminating attackers from genuine audio samples.

3.5.4 Evaluating single models trained on pooled data

Given that our main interest is to be able to use the same model across LA and PA attackers, we start our evaluation by analyzing the performance of simply pooling training data from both considered strategies. Results in terms of EER are reported in Table 3.8 for each of the considered types of features using ResNets. In the last column in the table, we report reference performances obtained by the best specialized model, i.e., using only training data corresponding to the particular type of attack used for evaluation. Such models will be used as a target of performance and will be generally referred to as *Privileged*, and by that we mean that the countermeasure has prior knowledge as to which type of attackers appear at testing time, and its training data is made so as to match the development/evaluation data in terms of type of attack.

Table 3.8 – EER obtained by pooling LA and PA training data. Models are trained on top of different speech features. Column *Privileged* refers to the best performance we could obtain with a privileged model, training on data corresponding to the same type of attack it would face at evaluation time. Privileged systems correspond to 1-dimensional ResNet along with LFCCs for LA attacks and 2-dimensional ResNets on top of ProdSpec for the case of PA attacks.

Data	Attack Type	EER			Privileged
		LFCC	ProdSpec	MGDCC	
Dev.	*LA*	0.05%	0.01%	0.30%	0.04%
	PA	2.38%	0.85%	3.88%	0.52%
Eval.	*LA*	14.39%	12.77%	13.13%	6.38%
	PA	2.96%	4.30%	5.99%	1.98%

In most cases, the combination of 2-dimensional convolutions and ProdSpec features yielded the best detection performance. However, regardless of the type of model and features, we observe a severe degradation in performance of the models trained on the pooled data when compared to the specialized case of reference, which suggests that naively creating bigger datasets with different types of attacks is not enough to recover the performance one can obtain when the attack strategy is known in advance.

3.5.5 Selecting the best approach to model the mixture coefficient

One important step in our design consists in determining which kinds of speech representations will be used in x_{LA}, x_{PA}, and x_{MIX}. For the first two cases, we leverage previous findings [42] and set *LFCC* and *ProdSPec* for x_{LA}, and x_{PA}, respectively. For the case of x_{MIX}, we empirically determined which type of feature would be the most effective in detecting types of attackers, and did so through training M_{MIX} by itself, utilizing only the term $BCE(\lambda, l^*)$ of the loss introduced in Section 3.5.1. Moreover, regarding the type of models required to properly model λ, we assume the problem of discriminating LA and PA attackers is as difficult as that of discriminating genuine *vs.* spoofing. As such, we employ the same types of models we use as detectors. However, in order to verify such assumption, we run experiments after drastically reducing the model capacity and observe their resulting performance. We do so by training a linear model on top of concatenated global first- and second-order statistics of input features.

The EER in the development and evaluation data for both cases are reported in Table 3.9 for ResNets while reference performance of linear probe models are presented in parenthesis. Since the labels l^* used to train M_{MIX} are such that $l^* = 0$, $l^* = 1$, and $l^* = 0.5$ for LA, PA, and genuine

Table 3.9 – Analysis on the performance in terms of detection EER of M_{MIX} models trained on top of different features to discriminate LA and PA attacks. Reference performance for linear models trained on top of the same features with the same computational budget are presented in parenthesis for each evaluation case.

Data	Attack Type	EER		
		LFCC	**ProdSpec**	**MGDCC**
Dev.	*LA*	0.12% (12.47%)	0.08% (11.57%)	0.06% (13.90%)
	PA	1.30% (24.63%)	0.61% (17.85%)	3.17% (15.25%)
Eval.	*LA*	14.05% (18.34%)	11.72% (34.17%)	18.19% (23.53%)
	PA	2.08% (34.36%)	2.80% (21.78%)	4.00% (19.80%)

examples, respectively, we expect values of λ close to the extreme values of 0 or 1 to be indicative of attacks. The distance from λ to 0.5 is then used to compute the detection score given by: $|\lambda - 0.5|$. In almost all the evaluation cases, the combination of 2-dimensional convolutions and ProdSpec features yielded the best detection performance. We thus set x_{MIX} to ProdSpec. As expected, the low capacity linear models yield a severe degradation in discrimination performance in all the cases leading us to conclude that large high capacity models similar to the privileged detectors are required to model λ.

3.5.6 Evaluation of the proposed approach

We now proceed to an evaluation in terms of EER as well as min-tDCF. In this case, our systems are compared with the baselines provided by ASVspoof 2019 organizers, which consist of GMM classifiers trained on frame-level features. Reference performance is provided for GMMs trained on both LFCC (LFCC-GMM) and Constant Q Cepstral Coefficients (CQCC-GMM). Extra baselines correspond to our own implementation of CQCC-GMM systems, well known i-vectors [1] along with PLDA, and end-to-end ResNets trained and tested on the same attack strategy. Such baselines are referred to in the tables as *privileged*, which once more indicates that the countermeasure has prior knowledge as to which type of attackers appear at testing time. We further included a mixture of two privileged models (indicated by LA/PA), where the final score is given by the maximum over the scores of the best privileged models trained independently on LA and PA data. The performance of models trained on pooled training data considering both attack strategies and discussed in Table 3.8 are further included for a complete comparison along with an ablation experiment. In this case, the mixture model has a more limited capacity corresponding to an ensemble where a linear model

such as that evaluated in Table 3.9 is used to model λ. Results for the LA and PA cases are reported in Tables 3.10 and 3.11, respectively.

Considering that our systems are comprised of different components, we can make use of the different outputs to score test recordings. We thus report results obtained by directly using y_{LA} and y_{PA} as scores, which we refer to in the tables by the name of their corresponding input features, i.e., LFCC and ProdSpec, respectively, and further include the results given by the final output y. One additional scoring strategy which is also evaluated corresponds to $|\lambda - 0.5|$ as discussed in the results reported in Table 3.9, in which case we expect extreme values of λ to be indicative of attackers. We refer to that score as *Lambda* in the tables.

We first highlight the proposed strategy outperforms models trained on pooled data for both cases of LA and PA attacks. This once more indicates that simply creating bigger datasets with different types of attacks is not enough to recover the performance one can get when the attack strategy is known in advance. Moreover, we remark that both ResNets and LCNNs trained and scored under the proposed ensemble setting are competitive when compared to privileged specialized systems, and even better in the case of LA attacks on development data and PA attacks both for development and evaluation data in terms of EER. Also, all systems, including the privileged baselines and the challenge benchmarks, presented a significant drop in performance from development to evaluation data for the specific case of LA attacks (see Table 3.10). This is due to the significant mismatch between the speech synthesis and voice conversion algorithms used to generate each data partition. Nonetheless, we observed the mixture score y degraded much less when compared to individual sub-systems (*LFCC* and *ProdSpec*) and models trained with the pooled dataset, thus further highlighting the effectiveness of the introduced setting in generalizing to novel conditions.

Additional conclusions we can draw from results presented in Tables 3.10 and 3.11 regard the fusion of privileged systems (indicated by LA/PA in both tables) as well as the ablation cases in which we significantly reduced the size of the model yielding mixture coefficients λ by simply using a linear model instead of ResNets/LCNNs. For the latter, we observed some degradation in performance once we compared fused privileged systems with their individual components (more significant in the LA case), being outperformed by our proposed setting in all considered evaluations. Moreover, as expected, reducing the representation capacity used to model λ is hurtful in that doing so degrades the performance of the ensemble, but most notably, the performance obtained by using

its output to score test recordings (as indicated by *Lambda* in the tables) is significantly worse compared to the same scoring strategy when ResNets or LCNNs are used.

We finally remark that the mixture score y is consistently more effective in detecting attackers in all evaluation cases when compared to the individual sub-component systems, thus supporting our hypothesis that there is a particular combination of feature space and modeling strategy better suited for each type of attack. In particular, the proposed approach is shown to be effective in combining sub-systems by deciding which should be *trusted* more given the input signal.

On the other hand, it can be seen in each case that the best performers do not follow the insights previously-obtained in [42] with specialized systems showing that specialized models are such that LFCCs and 1-dimensional convolutions were the best setting for LA attacks while ProdSpec and 2-dimensional convolutions would yield a better performance for the case of PA attacks. We conjecture that this is due to the pooling of PA and LA examples to compose training data. Since PA attacks are close to genuine examples in the space of LFCC, and similarly, LA attacks look genuine in the ProdSpec space[2], pooling both types of attacks is akin to corrupting training labels, which induces a strong regularization effect, leading to underfitting and greatly increasing the training sample complexity for individual systems. Nevertheless, the mixture model is able to compensate for the introduced suboptimality, yielding significant improvements over the single systems trained on the same pooled data.

3.6 Conclusion

We tackled the problem of detecting spoofing attacks to ASV systems in an end-to-end manner, i.e., our models map speech features into scores. First, we proposed variations of the light convolutional neural networks given limitations in available amount of training data. Performance of the proposed setting is thus evaluated on data introduced for the ASVspoof 2019 challenge, in which case we are able to achieve significant detection performance improvements with respect to well-known reference baseline systems. We further built on top of such an approach and introduced different strategies aimed at augmenting the training data in order to allow for larger models to be used to define detectors of spoofing attackers. By doing so, we were able to increase the size of

[2]As supported by the fact that LFCC and ProdSpec are shown to be better suited for detecting LA and PA attacks, respectively, for the case of individual systems.

Table 3.10 – The t-DCF and EER results for the LA task on the development and evaluation sets. The lower the values of min-tDCF and EER the better is the performance.. Training for ensemble systems is performed on top of combined LA and PA data.

System Description		Dev.		Eval.	
		EER	**t-DCF**	**EER**	**t-DCF**
Priviledged [9]	CQCC-GMM	0.43%	0.0123	9.57%	0.2366
	LFCC-GMM	2.71%	0.0663	8.09%	0.2116
Priviledged	CQCC-GMM	0.39%	0.0104	8.91%	0.2157
	i-vector/PLDA	0.70%	0.0211	16.05%	0.4201
	LFCC-ResNet	0.04%	0.0004	**6.38%**	**0.1423**
	LA/PA	0.08%	0.0023	10.82%	0.2322
Pooled data	LFCC	0.08%	0.0023	14.38%	0.3231
	ProdSpec	0.01%	0.0002	12.77%	0.2448
	MGDCC	0.27%	0.0066	13.13%	0.2953
Ablation (Linear mixture model)	LFCC	0.16%	0.0048	15.08%	0.3303
	ProdSpec	0.17%	0.0023	23.25%	0.3348
	Lambda	11.70%	0.2900	32.36%	0.7159
	Mixture	0.17%	0.0031	14.41%	0.3355
Proposed - ResNet	LFCC	0.08%	0.0021	15.84%	0.3476
	ProdSpec	0.03%	0.0002	15.73%	0.2725
	Lambda	0.04%	0.0004	13.12%	0.2962
	Mixture	**0.01%**	**0.0002**	9.87%	0.1890
Proposed - LCNN	LFCC	4.56%	0.1531	16.79%	0.4825
	ProdSpec	0.09%	0.0030	11.36%	0.2160
	Lambda	0.34%	0.0115	21.64%	0.3340
	Mixture	0.03%	0.0003	8.32%	0.2073

available corpora by a factor of five, while making it more diverse, thus introducing regularization effects that improved generalization to novel conditions. In fact, unlike past work, which focused on simple classification pipelines or relatively small neural networks, via data augmentation we were able to effectively train commonly used convolutional models, such as TDNNs.

As our main contribution, we introduced an ensemble-based approach with the goal of enabling detectors to be effective across varying types of spoofing attacks. We thus proposed a setting containing three components such that two of those are known to perform well individually in each of the two considered attack strategies. The third is then trained so as to decide how to combine the decision of the other systems depending on the input it is presented with. Evaluation led us to the conclusion that the proposed approach outperforms the standard procedure of pooling diverse training data and training a single model, and achieved competitive performance when compared to specialized models, trained on data curated so as to match the evaluation condition known in advance. In fact, for the specific case of PA attacks, our models outperformed the privileged

Table 3.11 – The t-DCF and EER results for the PA task on the development and evaluation sets. The lower the values of min-tDCF and EER the better is the performance. Training for ensemble systems is performed on top of combined LA and PA data.

System Description		Dev.		Eval.	
		EER	**t-DCF**	**EER**	**t-DCF**
Priviledged [9]	CQCC-GMM	9.87%	0.1953	11.04	0.2454
	LFCC-GMM	11.96%	0.2554	13.54	0.3017
Priviledged	CQCC-GMM	9.70%	0.1840	11.16	0.2478
	i-vector/PLDA	9.17%	0.2310	10.18	0.2687
	ProdSpec-ResNet	0.87%	**0.0232**	1.98%	**0.0579**
	LA/PA	1.28%	0.0453	2.06%	0.0728
Pooled data	LFCC	2.39%	0.0835	2.96%	0.1017
	ProdSpec	0.85%	0.0251	4.31%	0.1538
	MGDCC	3.89%	0.1174	5.99%	0.1858
Ablation (Linear mixture model)	LFCC	2.12%	0.0752	5.66%	0.1847
	ProdSpec	2.74%	0.0962	5.78%	0.2058
	Lambda	15.48%	0.3522	19.48%	0.4771
	Mixture	1.17%	0.0322	2.40%	0.0850
Proposed - ResNet	LFCC	1.87%	0.0656	3.99%	0.1408
	ProdSpec	3.80%	0.1111	4.94%	0.1479
	Lambda	1.32%	0.0317	2.29%	0.0641
	Mixture	**0.78%**	0.0275	**1.75%**	0.0606
Proposed - LCNN	LFCC	17.46%	0.4357	19.62%	0.4844
	ProdSpec	2.46%	0.0730	6.16%	0.1799
	Lambda	1.22%	0.0414	2.33%	0.0830
	Mixture	**0.78%**	0.0244	2.28%	0.0803

specialized systems. Given that most of the current work on countermeasures for spoofing attacks focuses on specialized systems for particular types of attack strategies, we believe the approach proposed herein is a first step in the direction of enabling ASV systems to be deployed free of the risk of security breaches, since it performs well across the different attack strategies we considered in our evaluation.

Chapter 4

Multi-level self-attentive TDNN: a general and efficient approach to summarize speech into discriminative utterance-level representations

4.1 Preamble

This chapter is compiled from material extracted from [165], under review in *Speech Communication*.

4.2 Introduction

While in Chapter 2 our focus was directed at developing effective training strategies yielding discriminative representation, we now turn our attention to model designs resulting in discriminative features. In particular, time delay neural networks (TDNN), discussed in Chapter 1, have been widely employed in speech processing applications, most notably within the space of voice biometrics. The architecture consists of a sequence of dilated 1-dimensional convolution layers which

operate across the temporal dimension. The convolutional stack is followed by a *temporal pooling layer*, which concatenates component-wise first- and second-order statistics over the time axis. The outputs of the pooling layer are finally passed through two fully-connected layers to yield outputs corresponding to conditional log-probabilities over the set of training speakers or languages.

The *temporal pooling* operation summarized above – or even more complex aggregation schemes, such as those discussed in Chapter 2 for the ASV evaluation – is intended to enable the computation of global utterance-level representations of the audio input and yield a single vector representing an entire sequence of acoustic features from input signal. From a practical perspective, global representations can be useful for tasks where the ability to process sequences of varying lengths is required, such as speaker and/or spoken language identification. However, including a pooling operation within a feed-forward architecture results in challenges and limitations, including:

1. *Overly specialized architectures*: lower-level (i.e., closer to inputs) or high-level (i.e., closer to outputs) learned representations might be more or less effective depending on the underlying task/data of interest. For instance, it's unlikely that the same type of representation would be useful for tasks such as speaker verification and language identification, since speaker-dependent cues are generally independent of the underlying phonetics within a signal, while determining languages does require phonetic information to be salient in learned representations. Given those variations across tasks, we argue determining the right *level of abstraction* of learned representations, mostly via deciding where to perform global pooling operations, is task-dependent and, as such, requires the design of a specific architecture for each different task.
2. *Ignoring complementary information*: TDNNs perform pooling operations only after the output of the final convolutional layer, thus global features from other levels of the model are not explicitly accounted for. Such operation could discard potentially discriminatory information for downstream tasks. We argue that a solution that simultaneously accounts for pooled features across different parts of the model has the potential to yield more discriminative learned representations, resulting in more generalizable models.

In order to address these limitations, we propose to modify the TDNN architecture and compute the pooling operation independently across the five convolution layers of the model. Moreover, we propose the use of a self-attentive layer [48] to give the model the ability to select, at training

time, the best combination scheme between features obtained at different levels and to compute a novel sequence of global vectors as a function of the entire set of representations. Finally, a last pooling operation is applied on the resulting sequence to yield an utterance-level representation. The proposed scheme offers the following advantages over conventional TDNNs:

1. *Versatility*: as opposed to designing a new architecture for each new task, the proposed architecture introduces a data-oriented approach that allows for the model to learn which layers provide more discriminative information for global pooling. As such, a single architecture can be re-used across different tasks, as will be observed in the evaluation section.
2. *Generality*: global factors in each layer are explicitly considered, as opposed to just the last convolutional layer. As such, complementary information obtained from different layers can be leveraged. We further highlight that such a scheme defines classes of models that contain simple aggregation mechanisms as particular cases; i.e., simple schemes such as averaging pooled representations from different levels, or selecting a specific layer can all be recovered by the proposed model if those are the best solutions for the task/data at hand.
3. *Learnability*: The pooling operation across layers acts as skip/residual connections, thus yielding loss landscapes that are easier to train against [4, 45].
4. *Transferability*: We further remark that the proposed pooling strategy can be used in any architecture that keeps the data dimension fixed throughout the encoder (e.g., ResNets [4]).

We thus gauge the versatility of the proposed architecture, referred to as multi-level self-attentive TDNN (ML-TDNN), on two end-to-end tasks: spoken language identification and spoofing attack detection. Moreover, we further evaluate the proposed model when it is used as an embedding encoder for speaker verification. Across all such cases, we find evidence supporting the claim that global information from low-level layers contains complementary information that can be leveraged at later stages of the model to improve performance.

4.3 Proposed Model

The proposed model builds upon the original TDNN architecture discussed in Chapter 1, illustrated in Figure 1.7, and further detailed in Table 1.1. In that case, an input sequence of length T is denoted as $x_{[1:T]}$ representing ordered acoustic feature vectors of dimension d (e.g., MFCCs), where

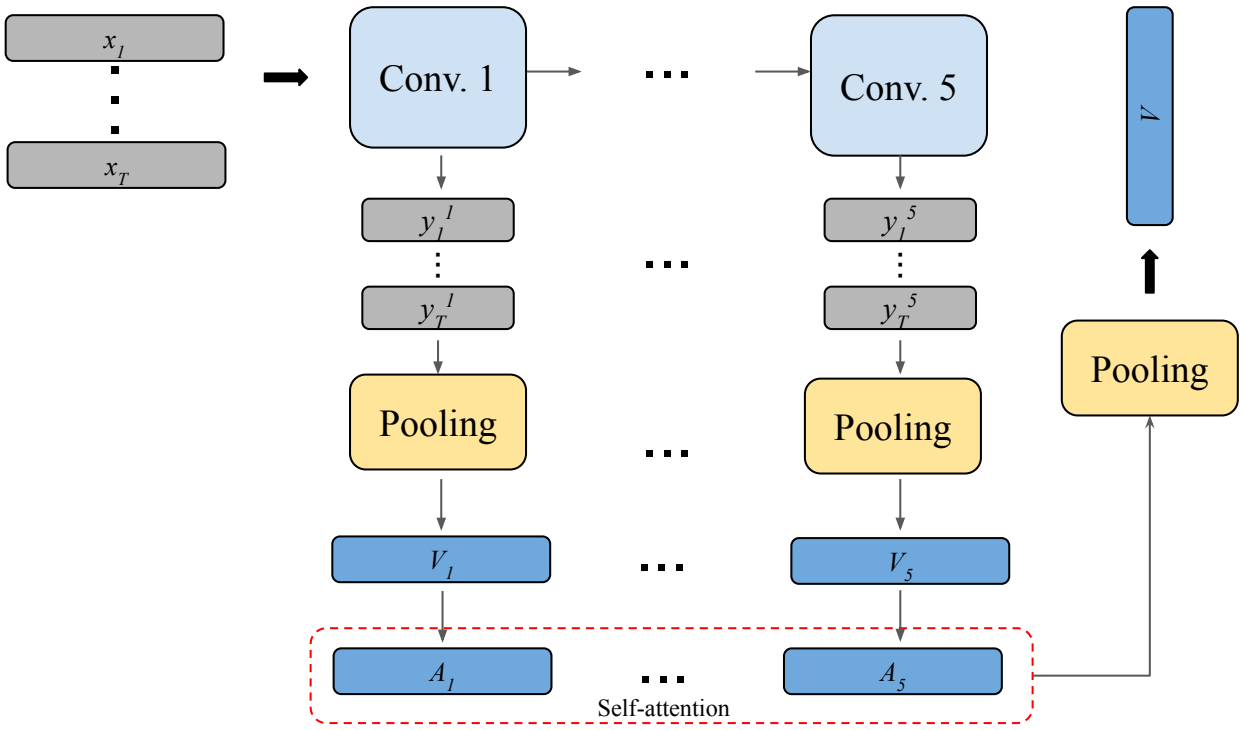


Figure 4.1 – Proposed TDNN with multi-level self-attentive temporal pooling.

each $x_i \in \mathbb{R}^d, i \in [T]$,. The set of features output by a stack of layers is denoted by $y_{1:T} \in \mathbb{R}^D, i \in [T]$. For the standard TDNN, the time pooling operation performed at the end of the convolutional stack converts the sequence $y_{1:T}$ into an utterance-level representation V, being the only model component able to summarize the content of the sequence into a global representation. We argue that this is: i) *too restrictive*, as it ignores valuable information available in earlier layers of the model, and ii) *too specific*, since different tasks should require designers to search for the right layer where to apply the pooling operation. As such, we propose the multi-level pooling scheme illustrated in Figure 4.1. In this case, we extract a sequence of local descriptors $y^k_{[1:T]}$ for each convolution layer, i.e., $k \in [1, 2, ..., 5]$. The same temporal pooling operation described above is now performed across every layer k, thus yielding a sequence of global descriptors denoted $V_{[1:5]}$.

Next, we employ a self-attention layer [48] (c.f. Chapter 1 for a discussion on attentive layers) so that each V_k can be taken into account depending on how relevant they are in order to result in discriminative representations. In other words, we take advantage of the depth-wise set of global summaries of the input sequence by including a sequence modeling component into the architecture. In our case, the number of self-attention heads is treated as a hyperparameter, and we empirically found the value of 16 to yield good results for the evaluation tasks considered herein. It is important

to emphasize that even though the computation cost of self-attention layers scales with the square of the input length, the proposed model is favoured by the fact that the sequence length is fixed and moderate, as it corresponds to the depth of the convolutional stack (i.e., 5). Moreover, such a layer defines a class of models that include schemes as simple as averaging/selection to more complicated non-linear relationships between elements. However, any alternative sequence modeling layer can be used in this case yielding variations of the proposed model.

We refer to the sequence output by the self-attentive transformation as $A_{[1:5]}$. Lastly, in order to combine the set of global descriptors $A_{[1:5]}$ into a single representation, we make use of a final statistical pooling layer that operates across the depth of the network, rather than across time. This yields the global descriptor:

$$V = \mu(A_k). \tag{4.1}$$

We then proceed as usual and feed V into a sequence of fully-connected components leading to the output layer. Training is carried out so that the parameters of all the components described above are learned jointly after being randomly initialized.

4.4 Experimental Setup

The empirical evaluations carried out herein are aimed at highlighting the versatility of the proposed architecture; as such, we show that re-using it across tasks does not require data-specific adaptations. Thus, we deliberately evaluate the proposed model across different tasks and datasets. Datasets and tasks used for evaluation are as follows (c.f. Chapter 1 for further details on all such datasets):

Detecting Spoofing Attacks For the case of spoofing detection, the evaluation setting introduced for the ASVspoof 2019 challenge [9] is considered with its two independent sub-tasks: *logical access attacks* (LA) and *physical access attacks* (PA) representing generated and simulated replay attacks, respectively.

Language Identification For the language identification task, we consider the data and evaluation conditions introduced for the AP18-OLR Challenge [8]. The data corresponds to audio

from ten languages, and the following evaluation conditions were defined: *short-duration*, *confusing languages*, and *unseen languages*

Speaker Verification The second release of the VoxCeleb corpus [7] is employed for evaluating the ML-TDNN when it is used as an embedding encoder. The dataset is given by audio collected from YouTube videos corresponding to interviews under unconstrained acoustic environments. Three evaluations are made available, namely: the test set of the first release of the corpus, the full training set of the first release (unseen during training) referred to as *extended*, and a *hard* set of trials, where available meta-data was used to create trials likely to be difficult to discriminate.

4.5 Experimental Results and Discussion

In the experiments below, we aim to find evidence in support of the following claims:

1. A single architecture can be re-used across a set of distinct tasks while achieving high prediction performance, as well as yielding effective representations for open-set tasks.
2. Global features collected from low-level layers contain complementary information that can be leveraged to improve accuracy.

We further remark that, since self-attentive aggregation includes as particular cases simple schemes, such as averaging representations obtained from all the layers as well as selecting specific layers to be used, we focus our computation budget on baseline systems that do not explicitly use any global summary of low-level representations. To further gauge the effectiveness of the proposed methods, results are compared with state-of-the-art approaches recently reported in the literature.

4.5.1 Detecting spoofing attacks

For end-to-end detection of spoofing attacks, binary classifiers are trained following the strategies discussed in [42]. More specifically, audio features are extracted such that 30 linear frequency cepstral coefficients [100] (LFCC) stacked with delta and double-delta coefficients are used for logical access attack detection. For physical access attacks, in turn, product spectra (ProdSpec) [117] with 257 frequency bins are used. Train data is augmented offline following the approach in [151] and

Table 4.1 – Detection performance on the evaluation set of the logical access task of the ASVspoof 2019.

	System	EER (%)	min-tDCF
Challenge baselines [9]	CQCC-GMM	9.57	0.2366
	LFCC-GMM	8.09	0.2116
Our baselines	CQCC-GMM	8.91	0.2157
	ivector/PLDA	16.55	0.4201
	ResNet [152]	6.38	0.1423
	TDNN [151]	7.00	0.1653
Proposed	ML-TDNN	**6.07**	**0.1327**

Table 4.2 – Detection performance on the evaluation set of the physical access task of the ASVspoof 2019.

	System	EER (%)	min-tDCF
Challenge baselines [9]	CQCC-GMM	11.04	0.2454
	LFCC-GMM	13.54	0.3017
Our baselines	CQCC-GMM	11.16	0.2478
	ivector/PLDA	10.18	0.2687
	ResNet [152]	1.98	0.0579
	TDNN [151]	1.77	0.0597
Proposed	ML-TDNN	**1.32**	**0.0470**

the sampling approach proposed in [42] to deal with unbalanced classes is further employed. The development data is used for cross-validation during model selection and hyperparameter tuning. We report performance in terms of the equal error rate (EER) and the normalized minimum tandem detection cost function (min-tDCF) [159] (see [9] for a detailed description and motivation for these evaluation metrics). Lower values of the metrics suggest improved performance. Experimental results are reported in Tables 4.1 and 4.2 for the logical and physical access attacks, respectively. In addition to the conventional TDNN approach [151], several other baselines are considered, including the two used in the ASVspoof 2019 challenge [9]. In particular, methods based on ResNets [4] trained on top of the same features, GMM-based classifiers trained with LFCC or constant-Q cepstral coefficients (CQCC), and a system based on the i-vector/PLDA combination [1, 106] are explored. As can be seen from both tables, the proposed ML-TDNN reduced EER and min-tDCF across both attack methods, thus better separating attacks and genuine samples. This can be due to an induced more well-conditioned loss landscape and/or access to global information in early layers close to the original data.

4.5.2 Spoken language identification

For spoken language identification, we train models using the multi-task method described in [134] and Chapter 2, where a metric-learning approach is used along with a standard maximum likelihood training strategy. The training loss is minimized using SGD and the same learning rate schedule discussed in [48] is employed. Evaluation is performed under the end-to-end setting so that predictions are made by directly forwarding data through the model, without using any external classifier. We do so by following the approach discussed in [133], where the output unit corresponding to the claimed class in a trial is used as a verification score. Regarding data preparation, pre-processing steps follow those discussed in [133].

Results are reported in Table 4.3 in terms of EER and the average cost performance (C_{avg}). Details about both metrics can be found in [8]. We highlight that the reported results were computed using the official scripts released for the AP18-OLR challenge. We further remark that results are reported for two variants of the proposed model, indicated by ML-TDNN (A) and (B) in Table 4.3, where differences are based on the validation datasets used to tune their hyperparamaters. More specifically, for model (A), only the short-duration partition of the development data was used, whereas for model (B) the complete development set was used. As can be seen from the obtained results, an apparent trade-off is found regarding the performance on the short-duration condition when using model (A). In particular, while using the short-duration development data improved accuracy under this condition, it degraded the performance on the other tasks. For practical implementations, we recommend using both models (A) and (B) in parallel and mixing their scores. In Table 4.3, we report the performance given by the classifier resulting from the sum of scores provided by models A and B (last row, termed A+B). Lastly, the proposed ML-TDNN method is shown to outperform conventional TDNNs across all the three evaluation conditions. In fact, with the exception of the short-duration condition, the proposed ML-TDNN outperformed all the benchmarks, including those relying on ResNets with recurrent pooling [166], which considers a complex training strategy involving pre-training steps of convolutional layers, shown to be important for short-duration conditions.

Table 4.3 – Spoken language identification performance for the three evaluation conditions considered on the AP18-OLR challenge.

	Short-duration		**Confusing**		**Unseen**	
	EER (%)	C_{avg}	EER (%)	C_{avg}	EER (%)	C_{avg}
i-vector+Cosine [166]	18.02	0.178	10.71	0.107	7.77	0.058
i-vector+PLDA [166]	17.50	0.174	10.66	0.106	7.51	0.052
ResNet (Stats.) [166]	**10.85**	0.112	3.63	0.036	4.23	0.020
ResNet (Attention) [166]	10.97	0.111	4.34	0.043	4.58	0.023
ResNet (LSTM) [166]	11.76	0.115	3.34	0.032	4.00	0.021
TDNN	13.16	0.126	4.30	0.058	4.80	0.036
ML-TDNN (A)	11.17	**0.109**	3.23	0.033	4.20	0.023
ML-TDNN (B)	14.68	0.138	**2.80**	**0.029**	**3.53**	**0.016**
ML-TDNN (A+B)	10.23	0.101	2.42	0.025	3.06	0.015

4.5.3 Speaker Verification

In the case of embedding encoders aimed at speaker verification tasks, both audio pre-processing and model training closely follow the recipe in [134], i.e., a multi-task training procedure uses both speaker recognition with the additive margin softmax approach [113], and supervised contrastive learning in order to train the embedding encoders. Hyperparameter selection is performed via random search with cross-validation performed using the test data of the first release of VoxCeleb. The best model is then used to evaluate the other partitions. We remark that we chose *not to employ test-time augmentation* given the computation overhead it incurs. Scoring of test trials is efficiently performed via simply computing the cosine similarity between embedded audio of both enrollment and test recordings considering *the full audio in a single forward pass* without the need for multiple embedding runs for each test trial, as employed by the other approaches (e.g., [7]).

For scoring, given that the architecture we evaluate is significantly changed with respect to the standard TDNN, we compare different embedding spaces as induced by outputs of different layers of the model (as indicated by ML-TDNN "I" or "O" indicating inner or outer dense layers, respectively). An illustration of the model along with the layers where representations are collected at testing phase are shown in Fig. 4.2. We then analyze a similar scheme as that reported in [128], where we perform prediction by combining pieces of evidence from different parts of the model. In our case, however, we fuse the scores directly, as opposed to the features, since we observed that to yield better results. Performance resulting from this score combination scheme is indicated by "(I+O)" in Table 4.4, where the achieved EER values obtained are reported. As can be seen, the proposed approach outperforms a number of benchmarks. Moreover, fusing scores from features

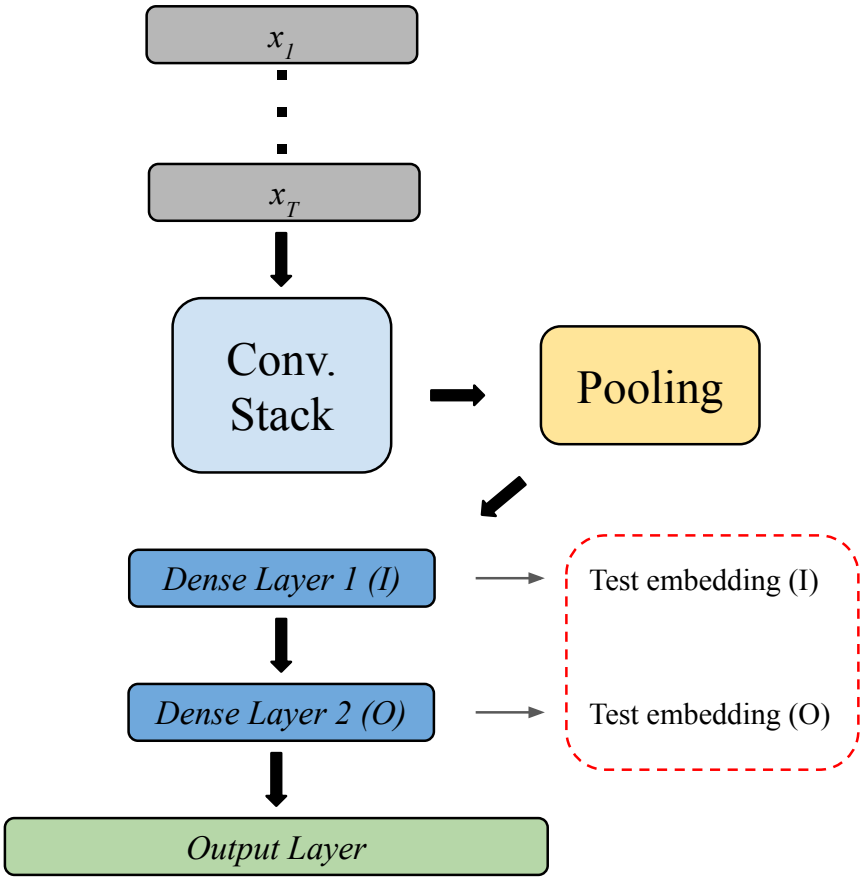


Figure 4.2 – Illustration of the use of the proposed model as an embedding encoder. Two embedding layers are considered as indicated by "I" or "O" as a reference to the inner or outer dense layers. Scoring trials independently and averaging the scores (I+O) was observed to yield improvements in most cases.

obtained in different layers shows to be a simple yet effective strategy, suggesting that, also at testing time, combining available information collected in different parts of the model can be beneficial and result in more effective decision scores.

4.6 Conclusion

We introduced a variation of the TDNN architecture in which temporal pooling operations are performed across all the layers of the convolutional stack rather than only at its end, as in the standard x-vector architecture. We term the proposed architecture ML-TDNN (multi-level self-attention TDNN). In particular, we propose a model component aimed at combining global representations from different layers by treating global statistics computed in different parts of

Table 4.4 – Verification performance on the VoxCeleb test partitions reported in terms of EER (%). Performance of our models is reported for representations obtained in different layers: (I)–Inner or penultimate dense layer, (O)–Outer or last dense layer, (I+O)–their average.

	Scoring	**EER** (%)
		VoxCeleb1 test set
Chung et al. [7]	Cosine	3.95
Xie et al. [167]	Cosine	3.22
Hajavi et al. [168]	Cosine	4.26
Xiang et al. [169]	Cosine	2.69
Kaldi recipe [86]	PLDA	2.51
Monteiro et al. [86]	Learned sim.	2.51
Wang et al. [127]	Cosine	2.41
ML-TDNN (I) (*Ours*)	Cosine	2.68
ML-TDNN (O) (*Ours*)	Cosine	2.24
ML-TDNN (I+O) (*Ours*)	Cosine	**2.14**
		Extended
Chung et al. [7]	Cosine	4.42
Xie et al. [167]	Cosine	3.13
Xiang et al. [169]	Cosine	2.76
Kaldi recipe [86]	PLDA	2.60
Monteiro et al. [86]	Learned sim.	2.57
Wang et al. [127]	Cosine	2.59
ML-TDNN (I) (*Ours*)	Cosine	2.71
ML-TDNN (O) (*Ours*)	Cosine	2.33
ML-TDNN (I+O) (*Ours*)	Cosine	**2.20**
		Hard
Chung et al. [7]	Cosine	7.33
Xie et al. [167]	Cosine	5.06
Xiang et al. [169]	Cosine	4.73
Kaldi recipe [86]	PLDA	4.62
Monteiro et al. [86]	Learned sim.	4.73
Wang et al. [127]	Cosine	4.33
ML-TDNN (I) (*Ours*)	Cosine	5.77
ML-TDNN (O) (*Ours*)	Cosine	**4.06**
ML-TDNN (I+O) (*Ours*)	Cosine	4.34

the model as sequences, and processing said sequences using a transformer-style multi-head self-attentive layer. Evaluation is performed on two end-to-end tasks (spoofing attack detection and language identification) and an embedding task (speaker verification). Experimental results showed that the proposed method consistently outperformed various benchmarks that use global features obtained from a single layer, thus highlighting that complementary information could be efficiently leveraged from low-level layers close to the input. Moreover, baselines in each considered task consisted of models specialized to said task. The proposed method achieved better (or at par) results

than such specialized models, thus showing their generalization capabilities across tasks. Further improvements could also be seen once, at testing time, self-fusion was applied from representations obtained from *different layers of the same model*.

The main limitation of the proposed method is the fact that the obtained versatility comes at an additional computational cost, as pooling operations across several layers and combining the resulting representations incurs additional computations. As such, resource-constrained settings, such as on-device (edge) applications might not directly benefit from this type of model. Notwithstanding, bootstrapping mechanisms for model compression or knowledge distillation [75] could be used to alleviate this issue.

Chapter 5

Learning (pseudo) metric spaces for discriminative verification

5.1 Preamble

This chapter is compiled from material extracted from [86], published in the *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

5.2 Introduction

In several settings, learning useful representations is generally a side effect of the solution of a pre-defined task. For example, while learning the decision surface in a classification problem, inner layers of artificial neural networks are shown to make salient cues of input data which are discriminable. Moreover, in unsupervised settings, bottleneck layers of autoencoders as well as approximate posteriors from variational autoencoders have all been shown to embed relevant properties of input data which can be leveraged in downstream tasks. Rather than employing a neural network to solve some task and hope learned features are useful, approaches such as *siamese networks* [170], which can be included in a set of approaches commonly referred to as *Metric Learning*, have been introduced with the goal of explicitly inducing features holding desirable properties. In this setting, an encoder is trained so as to minimize or maximize a *distance* measured across pairs of encoded

examples, depending on whether the examples within each pair belong to the same class or not. Follow-up work leveraged this idea for several applications [5, 6], which include, for instance, the verification problem in biometrics, as is the case of FaceNet [62] and Deep-Speaker [64], which are used for face and speaker recognition, respectively. However, as pointed out in recent work [62, 63, 78, 64, 65], careful selection of training pairs is crucial to ensure a reasonable sample complexity during training given that most triplets of examples quickly reach the condition such that distances measured between pairs from the same class are smaller than those of the pairs from different classes.

Here, we are concerned with the metric learning setting, and more importantly, we turn our attention to its application to the verification problem, i.e., that of comparing data pairs and determining whether they belong to the same class. The verification problem arises in applications where comparisons of two small samples are required such as face/finger-print/voice verification [76], image retrieval [77, 78], and so on. At testing time, inference is often performed to answer two types of questions:

1. Do two given examples belong to the same class?
2. Does a test example belong to a specific claimed class?

In both the cases, test examples might belong to classes never presented to the model during training. Current verification approaches are usually comprised of several components trained in a greedy manner [79, 3], and an end-to-end approach is still lacking.

Euclidean spaces will not, in general, be suitable for representing any desired type of structure expressed in the data (e.g., asymmetry [60] or hierarchy [80]). To avoid the need to select an adequate distance given every new problem we are faced with, as well as to deal with the training difficulties mentioned previously, we propose to augment the metric learning framework and jointly train an encoder (which embeds raw data into a lower-dimensional space) and a (pseudo) distance model tailored to the problem of interest. An end-to-end approach for verification is then defined by employing such pseudo-distance to compute similarity scores. Both models together, parameterized by neural networks, define a (pseudo) metric space in which inference can be performed efficiently since now semantic properties of the data (e.g., discrepancies across classes) are encoded by scores. While doing so, we found several interpretations appear from such learned pseudo-distance, and it can be further interpreted as a likelihood ratio in a Neyman-Pearson hypothesis test, as well as

an approximate divergence measure between the joint distributions of positive (same classes) and negative (different classes) pairs of examples. Moreover, even though we do not enforce models to satisfy properties of an actual metric[1], we empirically observe such properties to appear.

5.3 The verification problem

We consider the definition and notation introduced in Chapter 1 for the verification problem, which we intentionally partially re-introduce here for clarity since it involves the definition of the notation used throughout the chapter. In summary, given data elements $x \in \mathcal{X}$, each x can be associated to a class label $y \in \mathcal{Y}$ through a labeling function $f : \mathcal{X} \to \mathcal{Y}$. We define a *trial* $\mathcal{T}$ as a pair of sets of examples $\{X_i, X_j\}$, provided that $f(x_i^k) = f(x_i^l)\ \forall\ k, l \in \{1, 2, ..., |X_i|\}^2$ and $f(x_j^k) = f(x_j^l)\ \forall\ k, l \in \{1, 2, ..., |X_j|\}^2$, so that we can assign class labels to such sets X_m defining $f(X_m) = f(x_m)\ \forall\ x_m \in X_m$. The verification problem can be thus viewed as, given a trial $\mathcal{T}_{i,j} = \{X_i, X_j\}$, decide whether $f(X_i) = f(X_j)$, in which case we refer to $\mathcal{T}$ as *target trial*, or $f(X_i) \neq f(X_j)$ and the trial will be called *non-target*.

Moreover, under the Neyman-Pearson approach [11], verification is seen as a hypothesis test, where H_0 and H_1 correspond to the hypothesis such that $\mathcal{T}$ is target or otherwise [12], and the test is performed through the following likelihood ratio (LR):

$$LR = \frac{p(\mathcal{T}\,|H_0)}{p(\mathcal{T}\,|H_1)}, \tag{5.1}$$

where $p(\mathcal{T}\,|H_0)$ and $p(\mathcal{T}\,|H_1)$ correspond to models of target, and non-target trials. The decision is made by comparing LR with a threshold δ. A common approach to approximate LR through generative approaches [13] employs a universal background model (GMM-UBM, [14]) trained on data from all available classes, while the numerator is a fine-tuned model on enrollment data so that, for trial $\{X_1, X_2\}$, LR will be:

$$LR = \frac{p_{X_1}(X_2)}{p_{UBM}(X_2)} = \frac{p_{X_{Enroll}}(x_{test})}{p_{UBM}(x_{test})}. \tag{5.2}$$

[1] Symmetry, identity of indiscernibles, and triangle inequality.

5.4 Learning pseudo metric spaces

We consider the setting where both an encoding mechanism, as well as some type of similarity or distance across data points are to be learned. Assume $\mathcal{E} : \mathbb{R}^D \to \mathbb{R}^d$ and $\mathcal{D} : \mathbb{R}^d \times \mathbb{R}^d \to (0, 1)$ are deterministic mappings which will be referred to as encoder and distance model, respectively, and will be both parameterized by neural networks. Such entities resemble a metric-space, thus we will refer to it as *pseudo metric space*. We empirically observed that introducing distance properties in $\mathcal{D}$, i.e., by constraining it to be symmetric and enforcing it to satisfy the triangle inequality, did not result in improved performance, yet rendered training unstable. However, since trained models are found to approximately behave as an actual distance, we make use of the analogy, but further provide alternative interpretations of $\mathcal{D}$'s outputs.

Data samples are such that $x \in \mathcal{X} \subset \mathbb{R}^D$, and $z = \mathcal{E}(x)$ represents embedded data in $\mathbb{R}^d$. It will be usually the case that $D \gg d$. Once more, each data example can be further assigned to one of L class labels through a labeling function $f : \mathcal{X} \to \{1, ..., L\}$. Moreover, we define positive and negative pairs of examples denoted by $+$ or $-$ superscripts such that $x^+ = \{x_i, x_j\} \implies f(x_i) = f(x_j)$, as well as $x^- = \{x_i, x_j\} \implies f(x_i) \neq f(x_j)$. The same notation is employed in the embedding space so that $z^+ \coloneqq \mathcal{E}(x^+) \coloneqq \{\mathcal{E}(x_i), \mathcal{E}(x_j)\} \implies f(x_i) = f(x_j)$, and $z^- \coloneqq \mathcal{E}(x^-) \coloneqq \{\mathcal{E}(x_i), \mathcal{E}(x_j)\} \implies f(x_i) \neq f(x_j)$. We will denote the sets of all possible positive and negative pairs by $\mathcal{X}^+$ and $\mathcal{X}^-$, respectively, and further define a probability distribution p over $\mathcal{X}$ which, along with f, will yield p^+ and p^- over $\mathcal{X}^+$ and $\mathcal{X}^-$. Similarly to the setting in [171], which introduces a discriminator over pairs of samples, we are interested in $\mathcal{E}^*$ and $\mathcal{D}^*$ such that:

$$\mathcal{E}^*, \mathcal{D}^* \in \arg\min_{\mathcal{E},\mathcal{D}} \; - \mathbb{E}_{x^+ \sim p^+} \log(\mathcal{D} \circ \mathcal{E}(x^+)) - \mathbb{E}_{x^- \sim p^-} \log(1 - \mathcal{D} \circ \mathcal{E}(x^-)), \tag{5.3}$$

and $\circ$ indicates composition so that $\mathcal{D} \circ \mathcal{E}(x^+) \coloneqq \mathcal{D}(\mathcal{E}(x^+))$. Such problem is separable in the parameters of $\mathcal{E}$ and $\mathcal{D}$ and iterative solution strategies might include either alternate or simultaneous updates. We found the latter to converge faster in terms of wall-clock time and both approaches reach similar performance. We thus perform simultaneous updates while training.

The problem stated in (5.3) corresponds to finding $\mathcal{E}$ and $\mathcal{D}$ which will ensure that semantically close or distant samples, as defined through f, will preserve such properties in terms of distance in the new space, while doing so in lower dimension. We stress the fact that class labels define which

samples should be close together or far apart, which means that the same underlying data can yield different pseudo metric spaces if different semantic properties are used to define class labels. For example, if one considers that, for a given set of speech recordings, class labels are equivalent to speaker identities, recordings from the same speaker are expected to be clustered together in the embedding space, while different results can be achieved if class labels are assigned corresponding to spoken language, acoustic conditions, and so on.

5.4.1 Different interpretations for the distance model

Besides the view of $\mathcal{D}$ as a distance-like object defining a metric-like space $\{\mathcal{E}(\mathcal{X}), \mathcal{D}\}$, here we discuss some other possible interpretations of its outputs. We start by justifying the choice of the training objective defined in (5.3) by showing it to yield the likelihood ratio of particular trials of type I corresponding to a single enrollment example against a single test example, i.e., $\mathcal{T} = \{x_{enroll}, x_{test}\}$. In both of the next two propositions, proofs directly reuse results from the contrastive estimation and generative adversarial networks literature [172, 173] to show $\mathcal{D}$ can be used for verification.

Proposition 1. *The optimal $\mathcal{D}$ for any fixed $\mathcal{E}$ yields a simple transformation of the likelihood ratio stated in Eq. 1.1 for trials of the type $\mathcal{T} = \{x_{enroll}, x_{test}\}$.*

Proof. We first define p_z^+ and p_z^-, which correspond to the counterparts of p^+ and p^- induced by $\mathcal{E}$ in the embedding space. Now consider the loss $\mathcal{L}$ defined in Eq. 5.3:

$$\mathcal{L} = -\mathbb{E}_{z^+ \sim p_z^+} \log(\mathcal{D}(z^+)) - \mathbb{E}_{z^- \sim p_z^-} \log(1 - \mathcal{D}(z^-)),$$

$$= -\int_{\mathcal{Z}^+} p_z^+(z^+) \log(\mathcal{D}(z^+)) - \int_{\mathcal{Z}^-} p_z^-(z^-) \log(1 - \mathcal{D}(z^-)),$$

$$= -\int_{\mathcal{Z}'} p_z^+(z') \log(\mathcal{D}(z')) + p_z^-(z') \log(1 - \mathcal{D}(z')), \tag{5.4}$$

where $\mathcal{Z}'$ corresponds to $\mathcal{Z}^+ \cup \mathcal{Z}^-$ or equivalently $\mathcal{E}(\mathcal{X}^+) \cup \mathcal{E}(\mathcal{X}^-)$. Since $\mathcal{D}(z') \in (0,1)\ \forall\ z' \in \mathcal{Z}'$, the above integrand $p_z^+(z')\log(\mathcal{D}(z')) + p_z^-(z')\log(1-\mathcal{D}(z'))$ has its maximum at:

$$\mathcal{D}^*(z') = \frac{p_z^+(z')}{p_z^+(z') + p_z^-(z')} = \frac{1}{1 + \left(\frac{p_z^+(z')}{p_z^-(z')}\right)^{-1}}. \tag{5.5}$$

The rightmost equality on the above is of course only valid for $z' \in supp(p_z^+)$. Nevertheless, $\mathcal{D}^*(z')$ is in any case meaningful for verification. In fact, as will be discussed in Proposition 2, the optimal encoder is the one that induces $supp(p_z^+) \cap supp(p_z^-) = \varnothing$. Considering trial $\mathcal{T} = \{x_{enroll}, x_{test}\}$, we can write the ratio $\frac{p_z^+(z')}{p_z^-(z')}$ as:

$$\frac{p_z^+(z')}{p_z^-(z')} = \frac{p_z^+(\mathcal{E}(x_{enroll}), \mathcal{E}(x_{test}))}{p_z^-(\mathcal{E}(x_{enroll}), \mathcal{E}(x_{test}))} := \frac{p(T|H_0)}{p(T|H_1)}. \square \tag{5.6}$$

Proposition 1 indicates that the discussed setting can be used in an end-to-end manner to yield verification decision rules against a threshold δ for trials of a specific type.

The following lemma will be necessary for the next result:

Lemma 1. *If $supp(p_z^+) \cap supp(p_z^-) = \varnothing$, any positive threshold $0 < \delta < \infty$ yields optimal decision rules for trials $\mathcal{T} = \{x_{enroll}, x_{test}\}$.*

Proof. We prove the lemma by inspecting the decision rule under the considered assumptions in the two possible test cases: if $\mathcal{T}$ is non-target $\implies \frac{p_z^+(\mathcal{E}(x_{enroll}),\mathcal{E}(x_{test}))}{p_z^-(\mathcal{E}(x_{enroll}),\mathcal{E}(x_{test}))} = 0 < \delta$. If $\mathcal{T}$ is target $\implies \frac{p_z^+(\mathcal{E}(x_{enroll}),\mathcal{E}(x_{test}))}{p_z^-(\mathcal{E}(x_{enroll}),\mathcal{E}(x_{test}))} \to \infty > \delta$, completing the proof. $\square$

We now proceed and use the optimal discriminator into $\mathcal{L}$, which yields the following result for the optimal encoder:

Proposition 2. *Minimizing $\mathcal{L}$ yields optimal decision rules for any positive threshold.*

Proof. We plug $\mathcal{D}^*$ into $\mathcal{L}$ so that for any z' we obtain:

$$\begin{aligned}\mathcal{L} = &- \mathbb{E}_{z' \sim p_z^+} \log \Big(\frac{p_z^+(z')}{p_z^+(z') + p_z^-(z')} \Big) \\ &- \mathbb{E}_{z' \sim p_z^-} \log \Big(\frac{p_z^-(z')}{p_z^+(z') + p_z^-(z')} \Big) \\ = &- KL\Big(p_z^+ || p_z^+ + p_z^-\Big) - KL\Big(p_z^- || p_z^+ + p_z^-\Big) \\ = &\log 4 - 2JSD(p_z^+ || p_z^-).\end{aligned} \tag{5.7}$$

$\mathcal{L}$ is therefore minimized ($\mathcal{L}^* = 0$) iff $\mathcal{E}$ yields $supp(p_z^+) \cap supp(p_z^-) = \varnothing$, which results in optimal decision rules for any positive threshold, invoking lemma 1. □

We thus showed the proposed training scheme to be convenient for 2-sample tests under small sample regimes, such as in the case of verification, given that: (i) the distance model is also a discriminator which approximates the likelihood ratio of the joint distributions over positive and negative pairs[2], and the encoder will be such that it induces a high divergence across such distributions, rendering their ratio amenable to decision making even in cases where verified samples are as small as single enrollment and test examples.

On a speculative note, we provide yet another view of $\mathcal{D}$ by defining the kernel function $\mathcal{K} = \mathcal{D}$. If we assume $\mathcal{K}$ to satisfy Mercer's condition (which won't likely be the case within our setting since $\mathcal{K}$ will not be symmetric nor positive semidefinite), we can invoke Mercer's theorem and state that there is a feature map to a Hilbert space where verification can be performed through inner products. Training in the described setting could be viewed such that minimizing $\mathcal{L}$ becomes equivalent to building such a Hilbert space where classes can be distinguished by directly scoring data points one against the other. We hypothesize that constraining $\mathcal{K}$ to sets where Mercer's condition does hold might yield an effective approach for the problems we consider herein, which we intend to investigate in future work.

5.4.2 Training

We now describe the procedure we adopt to minimize $\mathcal{L}$ as well as some practical design decisions made based on empirical results. Both $\mathcal{E}$ and $\mathcal{D}$ are implemented as neural networks. In our

[2]The joint distribution over negative pairs is simply the product of marginals: $p^-(x_i, x_j) = p(x_i)p(x_j)$.

experiments, $\mathcal{E}$ will be convolutional (2-d for images and 1-d for audio) while $\mathcal{D}$ is a stack of fully-connected layers which take as input concatenated embeddings of pairs of examples. Training is carried out with standard mini-batch SGD with momentum. We perform simultaneous update steps for $\mathcal{E}$ and $\mathcal{D}$ since we observed that to be faster than alternate updates, while yielding the same performance. Standard regularization strategies such as weight decay and label smoothing [81] are also employed. We empirically found that employing an auxiliary multi-class classification loss significantly accelerates training. Since our approach requires labels to determine which pairs of examples are positive or negative, we make further use of the labels to compute such auxiliary loss, which will be indicated by $\mathcal{L}_{CE}$. To allow for computation of $\mathcal{L}_{CE}$, we project $z = \mathcal{E}(x)$ onto the simplex Δ^{L-1} using a fully-connected layer. Minimization is then performed on the sum of the two losses, i.e., we solve $\mathcal{E}, \mathcal{D} \in \arg\min \mathcal{L}' = \mathcal{L} + \mathcal{L}_{CE}$, where the CE subscript in $\mathcal{L}_{CE}$ indicates the multi-class cross-entropy loss.

All hyperparameters are selected with a random search over a pre-defined grid. For the particular case of the auxiliary loss $\mathcal{L}_{CE}$, besides the standard cross-entropy, we also ran experiments considering one of its so-called *large margin* variations. We particularly evaluated models trained with the additive margin softmax approach [113]. The choice between the two types of auxiliary losses (standard or large margin) was a further hyperparameter and the decision was based on the random search over the two options. The grid used for hyperparameters selection along with the values chosen for each evaluation are presented in Section 5.6. A pseudocode describing our training procedure is presented in Algorithm 2.

Algorithm 2 Training procedure.

$\mathcal{E}, \mathcal{D} = InitializeModels()$
repeat
  $x, y = SampleMiniBatch()$
  $z = \mathcal{E}(x)$
  $z^+ = GetAllPositivePairs(z, y)$
  $z^- = GetAllNegativePairs(z, y)$
  $y' = ProjectOntoSimplex(z)$ `# i.e., a learned linear layer followed by softmax.`
  $\mathcal{L}' = \mathcal{L}(z^+, z^-, \mathcal{D}) + \mathcal{L}_{CE}(y', y)$
  $\mathcal{E}, \mathcal{D} = UpdateRule(\mathcal{E}, \mathcal{D}, \mathcal{L}')$ `# i.e., run optimizer step.`
until Maximum number of iterations reached
return $\mathcal{E}, \mathcal{D}$

5.5 Evaluation

To evaluate the described framework, we rely on three sets of experiments: (i) a proof-of-concept analysis, (ii) a large-scale ASV benchmark, and (iii) extra experiments, where we analyze the behavior of the proposed method in terms of the presence of distance properties, its performance under distribution shifts, and its sensitivity to the choice of architecture for the distance model. In the first part of our evaluation, we run proof-of-concept experiments that take advantage of well-established image datasets and models to simulate verification settings and validate our proposal when compared against metric learning under standard distances. For that, we report results on all trials created for the test sets of CIFAR-10 and *Mini*ImageNet. In the former, the same 10 classes of examples appear for both train and test partitions, in what we refer to as closed set verification. For the case of *Mini*ImageNet, since that dataset was designed for few-shot learning applications, we have an open set evaluation for verification since there are 64, 16, and 20 disjoint classes of training, validation, and test examples.

We then move on to a *large scale realistic evaluation*. To this end, we make use of the VoxCeleb corpus [82, 7], corresponding to audio recordings of interviews taken from Youtube videos, which means there's no control over the acoustic conditions present in the data. Moreover, while most of the corpus corresponds to speech in English, other languages are also present, so that test recordings are from different speakers relative to the train data, and potentially also from different languages and acoustic environments. We specifically employ the second release of the corpus so that training data is composed of recordings from 5994 speakers while three test sets are available: (i) **VoxCeleb1 Test set**, which is made up of utterances from 40 speakers, (ii) **VoxCeleb1-E**, i.e., the complete first release of the data containing 1251 speakers, and (iii) **VoxCeleb1-H**, corresponding to a subset of the trials in **VoxCeleb1-E** so that non-target trials are designed to be hard to discriminate by using the meta-data to match factors, such as nationality and gender of the speakers. We then report experiments performed to observe whether $\mathcal{D}$'s outputs present properties of actual distances, and finally check the influence of $\mathcal{D}$'s architecture on final performance.

In our final set of experiments, we study properties of the proposed pair of models by analyzing to which extent properties of actual distances hold after the proposed training procedure is performed. To evaluate the behavior of the method when training and testing data distributions shift apart, we consider the evaluation case introduced within the NIST SRE 2018, where the languages spoken

on test recordings are not the same as those corresponding to training data. Finally, we evaluate the sensitivity of the approach to choices in the architecture of $\mathcal{D}$. In particular, we check for the effect of the depth of $\mathcal{D}$ in terms of its number of layers in the resulting performance.

Our main baselines for proof-of-concept experiments correspond to the same encoders as in the evaluation of our proposed approach, while $\mathcal{D}$ is dropped and replaced by the Euclidean distance. In those cases, however, in order to get the most challenging baselines, we perform online selection of hard negatives. Our baselines closely follow the setting described in [134]. All such baselines are referred to as *triplet* in the tables with results as a reference to the training loss in those cases. Unless specified, all models, baseline or otherwise, are trained from scratch, and the same computation budget is used for training and hyperparameter search for all models we trained.

Performance is assessed in terms of the difference to 1 of the area under the operating curve, indicated by 1-AUC in the tables, and also in terms of equal error rate (EER). EER indicates the operating point (i.e., threshold selection) at which the miss and false alarm rates are equal. Both metrics are better when closer to 0. We consider different strategies to score test trials. Both cosine similarity and PLDA are considered in some cases, and when the output of $\mathcal{D}$ is directly used as a score we then indicate it by E2E in reference to *end-to-end*[3]. We further remark that cosine similarity can also be used to score trials in our proposed setting, and we observed some performance gains when applying simple sum fusion of the two available scores. Additional implementation details are included in Section 5.6.

5.5.1 Proof-of-concept evaluation on CIFAR-10 and *Mini*ImageNet

The encoder for evaluation on both CIFAR-10 and *Mini*ImageNet was implemented as a ResNet-18 [4]. Results are reported in Table 5.1. Results indicate the proposed scheme indeed yields effective inference strategies under the verification setting compared to traditional metric learning approaches, while using a more simplified training scheme since: (i) no sort of approach for harvesting hard negative pairs (e.g., [62, 78]) is needed in our case, and those are usually expensive, (ii) the method does not require large batch sizes, and (iii) we employ a simple loss with no hyperparameters that have to be tuned, as opposed to margin-based triplet or contrastive losses. We further highlight that the encoders trained with the proposed approach have the possibility for trials to be

[3]Scoring trials with cosine similarity can be also seen as end-to-end.

Table 5.1 – Evaluation of models trained under the proposed approach on image data.

		Scoring	*EER*	*1-AUC*
CIFAR-10	Triplet	Cosine	3.80%	0.98%
	Proposed	E2E	3.43%	0.60%
		Cosine	3.56%	1.03%
		Cosine + E2E	3.42%	0.80%
Mini*ImageNet* *(Validation)*	Triplet	Cosine	28.91%	21.58%
	Proposed	E2E	28.64%	21.01%
		Cosine	30.66%	23.70%
		Cosine + E2E	28.49%	20.90%
Mini*ImageNet* *(Test)*	Triplet	Cosine	29.68%	22.56%
	Proposed	E2E	29.26%	22.04%
		Cosine	32.97%	27.34%
		Cosine + E2E	29.32%	22.24%

further scored with cosine similarities, which yields a performance improvement in some cases when combined with $\mathcal{D}$'s output.

5.5.2 Large-scale verification with VoxCeleb

We now proceed and evaluate the proposed scheme in a more challenging scenario corresponding to realistic audio data for speaker verification. To do so, we implement $\mathcal{E}$ as the well-known time-delay architecture [108] employed within the x-vector setting, showed to be effective in summarizing speech into speaker- and spoken language-dependent representations [3, 174]. The model consists of a sequence of dilated 1-dimensional convolutions across the temporal dimension, followed by a time pooling layer, which simply concatenates element-wise first- and second-order statistics over time. Statistics are finally projected into an output vector through fully-connected layers. Speech is represented as 30 MFCCss obtained with a short-time Fourier transform using a 25ms Hamming window with 60% overlap. All the data is downsampled to 16kHz beforehand. An energy-based voice activity detector is employed to filter out non-speech frames. We augment the data by creating noisy versions of training recordings using exactly the same approach as in [3]. Further practical details are discussed in Section 5.6.

We compared our models with a set of published results as well as the results provided by the popular Kaldi recipe[4], considering scoring using cosine similarity or PLDA. For the Kaldi baseline, we found the same model as ours to yield relatively weak performance. As such, we decided to search

[4]Kaldi recipe: `https://github.com/kaldi-asr/kaldi/tree/master/egs/voxceleb`

over possible architectures in order to make it a stronger baseline. We thus report the best model we could find which has the same structure as ours, i.e., it is made up of convolutions over time followed by temporal pooling and fully-connected layers, while the convolutional stack is deeper, which makes the comparison unfair in their favor.

We further evaluated our models using PLDA by running just the part of the same Kaldi recipe corresponding to the training of that downstream classifier on top of representations obtained from our encoder. Results are reported in Table 5.2 and support our claim that the proposed framework can be directly used in an end-to-end manner. It is further observed that it outperformed standard downstream classifiers, such as PLDA, by a significant difference while not requiring any complex training procedure, as common metric learning approaches usually do. We employ simple random selection of training pairs. Ablation results are also reported, in which case we dropped the auxiliary loss $\mathcal{L}_{\mathcal{CE}}$ and trained the same $\mathcal{E}$ and $\mathcal{D}$ using the same budget in terms of number of iterations, showing that having the auxiliary loss improves performance in the considered evaluation.

5.5.3 Extra experiments

5.5.3.1 Speaker verification under domain shift

In this experiment, we evaluate the performance of the proposed setting when test data significantly differs from the training examples. To do so, we employ the data introduced for one of the tasks of the 2018 edition of the NIST Speaker Recognition Evaluation (SRE)[5]. We specifically consider the CTS task so that test data corresponds to spontaneous conversational telephone speech spoken in Tunisian Arabic, while the bulk of the train data is spoken in English. Besides the language mismatch, variations due to different codecs are further observed (PSTN *vs.* PSTN and VOIP).

The main training dataset (English) is built by combining the data from *NIST SRE*s from 2004 to 2010, *Mixer* 6, as well as *Switchboard*-2, phases 1, 2, and 3, and the first release of *VoxCeleb*, yielding a total of approximately 14000 speakers. Audio representations correspond to 23 MFCCs obtained using a short-time Fourier transform with a 25ms Hamming window and 60% overlap. The audio data is downsampled to 8kHz. Further pre-processing steps are the same as those performed

[5] `https://www.nist.gov/system/files/documents/2018/08/17/sre18_eval_plan_2018-05-31_v6.pdf`

Table 5.2 – Evaluation of models trained under the proposed approach on VoxCeleb.

	Scoring	*Training set*	*EER*
VoxCeleb1 Test set			
Nagrani et al. (2017) [82]	PLDA	VoxCeleb1	8.80%
Cai et al. (2018) [175]	Cosine	VoxCeleb1	4.40%
Okabe et al. (2018) [126]	Cosine	VoxCeleb1	3.85%
Hajibabaei & Dai (2018) [176]	Cosine	VoxCeleb1	4.30%
Ravanelli & Bengio (2019) [177]	Cosine	VoxCeleb1	5.80%
Chung et al. (2018) [7]	Cosine	VoxCeleb2	3.95%
Xie et al. (2019) [167]	Cosine	VoxCeleb2	3.22%
Hajavi & Etemad (2019) [168]	Cosine	VoxCeleb2	4.26%
Xiang et al. (2019) [169]	Cosine	VoxCeleb2	2.69%
Kaldi recipe	PLDA	VoxCeleb2	2.51%
Proposed	Cosine	VoxCeleb2	4.97%
Proposed	E2E	VoxCeleb2	2.51%
Proposed	Cosine + E2E	VoxCeleb2	2.51%
Proposed	PLDA	VoxCeleb2	3.75%
Ablation ($-\mathcal{L}_{CE}$)	E2E	VoxCeleb2	3.44%
VoxCeleb1-E			
Chung et al. (2018) [7]	Cosine	VoxCeleb2	4.42%
Xie et al. (2019) [167]	Cosine	VoxCeleb2	3.13%
Xiang et al. (2019) [169]	Cosine	VoxCeleb2	2.76%
Kaldi	PLDA	VoxCeleb2	2.60%
Proposed	Cosine	VoxCeleb2	4.77%
Proposed	E2E	VoxCeleb2	2.57%
Proposed	Cosine + E2E	VoxCeleb2	2.53%
Proposed	PLDA	VoxCeleb2	3.61%
Ablation ($-\mathcal{L}_{CE}$)	E2E	VoxCeleb2	3.70%
VoxCeleb1-H			
Chung et al. (2018) [7]	Cosine	VoxCeleb2	7.33%
Xie et al. (2019) [167]	Cosine	VoxCeleb2	5.06%
Xiang et al. (2019) [169]	Cosine	VoxCeleb2	4.73%
Kaldi recipe	PLDA	VoxCeleb2	4.62%
Proposed	Cosine	VoxCeleb2	8.61%
Proposed	E2E	VoxCeleb2	4.73%
Proposed	Cosine + E2E	VoxCeleb2	4.69%
Proposed	PLDA	VoxCeleb2	5.98%
Ablation ($-\mathcal{L}_{CE}$)	E2E	VoxCeleb2	7.76%

for experiments with VoxCeleb, i.e., an energy-based voice activity detector is followed by data augmentation performed via distorting original samples adding reverberation and background noise.

Baseline: For performance reference, we trained the well-known x-vector setting [3] using its Kaldi recipe[6]. In that case, PLDA is employed for scoring test trials. The same training data used

[6] `https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v1/local/nnet3/xvector`

Table 5.3 – Evaluation of models under domain shift. Target data corresponds to speech spoken in Arabic. Fine-tuning on datasets including target data yields an improvement in verification performance.

	Training domain	*Scoring*	*EER*
Snyder et al. (2018) [3]	English	PLDA	11.30%
	English+Arabic	Adapted PLDA	9.44%
Proposed	English	E2E	13.61%
	Multi-language	E2E	8.43%

to train our systems is employed in this case as well. The recipe performs the following steps: (i) training of a TDNN (same architecture as in our case) as a multi-class classifier over the set of training speakers using the same training data utilized to train our proposed model; (i) preparation of PLDA's training data, in which case the *SRE* partition of the training set is encoded using the second to last layer of the TDNN, embeddings are length-normalized and mean-centered using the average of an unlabelled sample from the target domain and finally have their dimensionality reduced using Linear Discriminant Analysis; (iii) training of PLDA; (iv) scoring of test trials. Additionally, in order to cope with the described domain shift, the model adaptation scheme introduced in [145] is also utilized for PLDA so that a second PLDA model is trained on top of target data. The final downstream classifier is then obtained by averaging the parameters of the original and target domain models. Both results obtained with and without the described scheme are reported in Table 5.3.

For the case of the proposed approach, training is carried out using the training data described above corresponding to speech spoken in English. We re-used the setting found to work well on the experiments reported with the VoxCeleb corpus in Section 5.5.2 including all hyperparameters, architecture, data sampling and mini-batch construction strategies, and computational budget. We additionally build a multi-language training set including data corresponding to the target domain so that we can fine-tune our model. The complementary training data corresponds to the data introduced for the 2012 (English) and 2016 (Cantonese+Tagalog) editions of NIST SRE as well as the development partition of NIST SRE 2018 which corresponds to the target domain of evaluation data (Arabic). This is done so as to increase the amount of data within the complementary partition and avoid overfitting to the small amount of target data. The combination of such data sources yields approximately 800 speakers. We train our models on the large out-of-domain dataset and fine-tune the resulting model in the multi-language complementary data.

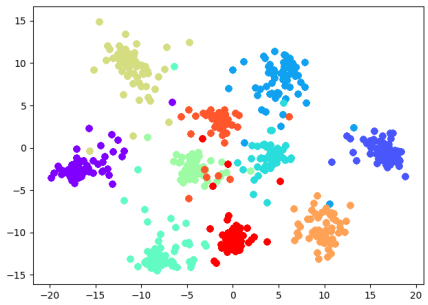
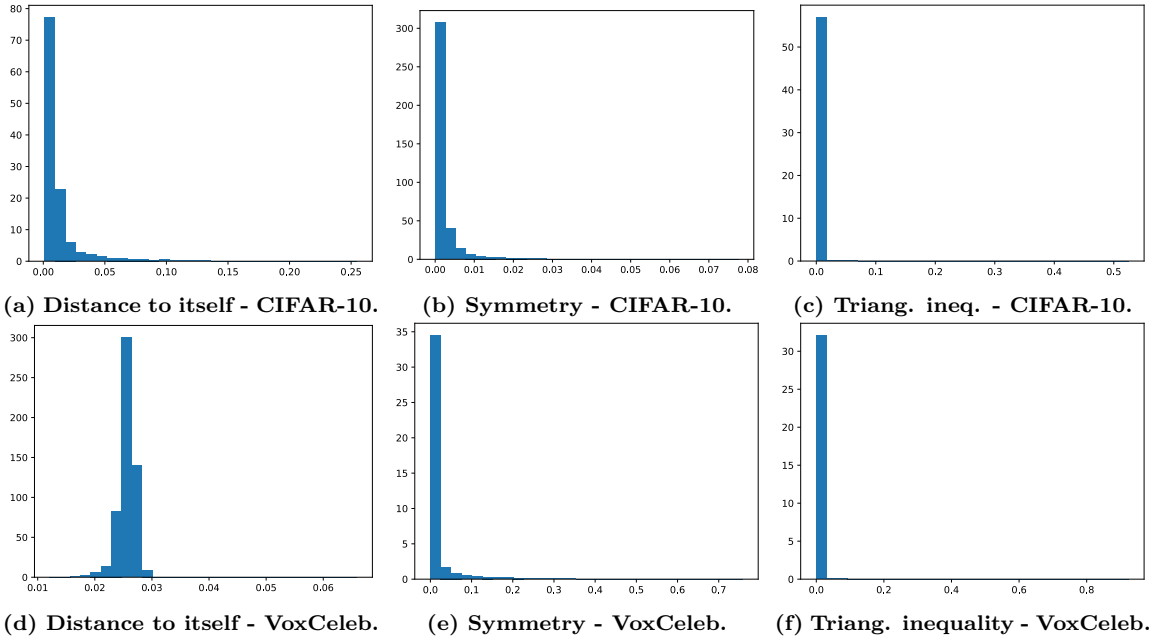


Figure 5.1 – MNIST embeddings on a 2-dimensional space. Each color represents test examples corresponding to a digit from 0 to 9.

Results in terms of equal error rate are presented in Table 5.3. While our model appears to be more domain dependent when compared to PLDA, as indicated by results where only out-of-domain data is employed, it significantly improves once a relatively small amount of target domain data is provided. We stress the fact that the proposed setting dramatically simplifies verification pipelines and completely removes practical issues such as those related to processing steps prior to training of the downstream classifier.

5.5.3.2 Checking for distance properties in trained models

We now empirically evaluate how $\mathcal{D}$ behaves in terms of properties of distances or metrics, such as symmetry, for instance. We start by plotting embeddings from $\mathcal{E}$ and do so by training an encoder on MNIST [88] under the proposed setting (without the auxiliary loss $\mathcal{L}_{CE}$ in this case) so that its outputs are given by $z \in \mathbb{R}^2$. We then plot the embeddings of the complete MNIST's test set on Fig. 5.1, where the raw embeddings in $\mathbb{R}^2$ are directly displayed in the plot. Interestingly, classes are reasonably clustered in the Euclidean space even if such behavior was never enforced during training. We proceed and directly check for distance properties in $\mathcal{D}' = 1 - \mathcal{D}$. For the test set of CIFAR-10 as well as for **VoxCeleb1 Test set**, we plot histograms of (i) the distance to itself for all the test examples, (ii) a symmetry measure given by the absolute difference of the outputs of $\mathcal{D}'$ measured in the two directions for all possible test pairs, and (iii) a measure of how much $\mathcal{D}'$ satisfies the triangle inequality, which we do by measuring $\max[\mathcal{D}'(b,c) - (\mathcal{D}'(a,b) + \mathcal{D}'(a,c)), 0]$ for a random sample taken from all possible triplets of examples $\{a, b, c\}$. Proper metrics should have all such quantities equal 0. In Figure 5.2, it can be seen that once more, even if any particular behavior is enforced over $\mathcal{D}$ at its training phase, resulting models approximately behave as proper metrics. We thus hypothesize the relatively easier training observed in our setting, in the sense



(a) Distance to itself - CIFAR-10. (b) Symmetry - CIFAR-10. (c) Triang. ineq. - CIFAR-10.

(d) Distance to itself - VoxCeleb. (e) Symmetry - VoxCeleb. (f) Triang. inequality - VoxCeleb.

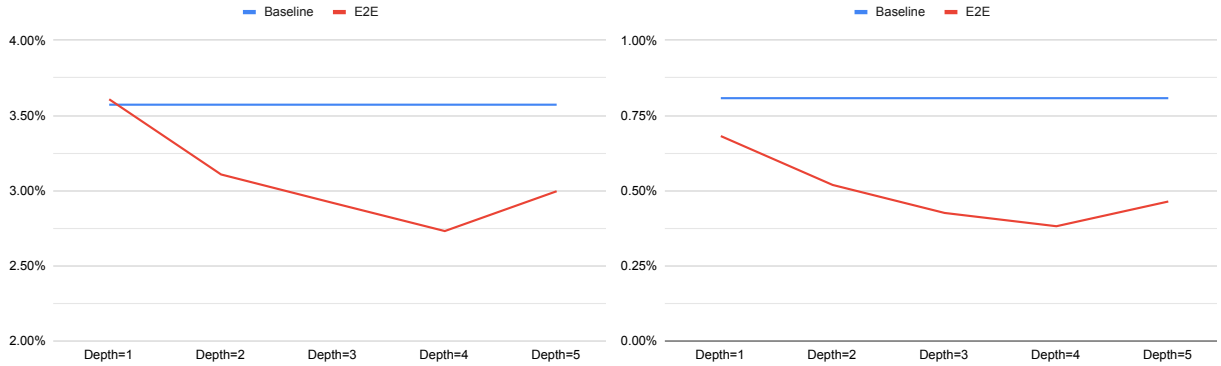
Figure 5.2 – Evaluation of distance properties of trained models.

that it works well without complicated schemes for selection of negative pairs, is due to the not so constrained distances induced by $\mathcal{D}$.

5.5.3.3 Varying the depth of the distance model for verification on ImageNet

We performed closed set verification on the full ImageNet [94] with distance models of increasing depths (1 to 5) to verify whether our setting is stable with respect to some of the introduced hyperparameters. With this experiment, we specifically intend to assess how difficult it would be in practice to find a good architecture for the distance model. Our models are compared against encoders with the same architecture, but trained using a standard metric learning approach, i.e the same training scheme as that employed for baselines reported in Table 5.1.

For this case, the encoder $\mathcal{E}$ is implemented as the convolutional stack of a ResNet-50 followed by a fully-connected layer used to project the output representations to the desired dimensionality, and we employ an embedding dimension of 128 across all reported models. $\mathcal{D}$ is once more implemented as a stack of fully-connected layers in which case we set the sizes of all hidden layers to 256. Training is performed such that the parameters of the convolutional portion of $\mathcal{E}$ are initialized from a pre-trained model for multi-class classification on ImageNet, and this approach is used for both our models as well as the baseline. We then perform SGD on the combined loss discussed above using



(a) Verification in terms of EER on all trials created by pairing all test examples of ImageNet. Results indicate that defining the architecture of the distance model is not difficult in practice given that models of varying depths yield a relatively small performance range.

(b) Verification in terms of AUC on all trials created by pairing all test examples of ImageNet. Results indicate that defining the architecture of the distance model is not difficult in practice given that models of varying depths yield a relatively small performance range.

Figure 5.3 – Effects of increasing depth of the distance model.

the standard multi-class cross entropy as auxiliary loss. Moreover, given the large number of classes in ImageNet compared to commonly used batch sizes, in order to be able to always find positive pairs throughout training, mini-batches are constructed using the same strategy as that employed for experiments with VoxCeleb, i.e., we ensure at least 5 examples per class appear in each mini-batch. The learning rate is set to 0.001 and is reduced by a factor of 0.1 every 10 epochs. Training is carried out for 50 epochs. Evaluation is performed over trials obtained from building all possible pairs of examples from the test partition of ImageNet. Results are reported in Figures 5.3a and 5.3b in terms of EER and the area over the operating curve (1-AUC), respectively. Scoring for the case of baseline encoders is performed with cosine similarity between encoded examples from test trials. While standard metric learning encoders make for strong baselines, all evaluated distance models are able to perform on pair (depth=1) or better than (depth>1) such models.

The results discussed herein provide empirical evidence for the claim that tuning the hyper-parameters we introduced in comparison to previous settings, i.e., the architecture of the distance model, is not so challenging in that we achieve reasonably stable performance for verification on ImageNet when varying the depth of the distance model. Yet another empirical finding supporting that claim consists of the fact that similar architectures of the distance model were found to work well across all the datasets/domains we evaluated on. We specifically found that distance models with 3 or 4 hidden layers with 256 units each work well across datasets, which we believe might be a reasonable starting point for extending the approach we discussed to other datasets.

5.6 Implementation details

5.6.1 Architecture of the distance model

$\mathcal{D}$ is implemented as a stack of fully-connected layers with LeakyReLU activations. Dropout is further used in between the last hidden and the output layer. The number and size of hidden layers as well as the dropout probability were tuned for each experiment.

5.6.2 CIFAR-10 and *Mini*ImageNet

5.6.2.1 Hyperparameters

The grid used on the hyperparameter search for each hyperparameter is presented next. A budget of 100 runs was considered and each model was trained for 600 epochs. Hyperparameters yielding the best EER on the validation data for our proposed approach and the triplet baseline are represented by * and †, respectively. In all experiments, the mini-batch size was set to 64 and 128 for CIFAR-10 and *Mini*ImageNet, respectively. A reduce-on-plateau schedule for the learning rate was employed, while its patience was a further hyperparameter included in the search.

CIFAR-10:

- Learning rate: $\{0.5, 0.1, 0.01^{*,\dagger}, 0.001\}$
- Weight decay: $\{0.01, 0.001^{*}, 0.0001^{\dagger}, 0.00001\}$
- Momentum: $\{0.1, 0.5, 0.9^{*,\dagger}\}$
- Label smoothing: $\{0.01, 0.1, 0.2^{*,\dagger}\}$
- Patience: $\{1, 5, 10^{*,\dagger}, 20\}$
- Number of $\mathcal{D}$ hidden layers: $\{2, 3^{*}, 4, 5\}$
- Size of $\mathcal{D}$ hidden layers: $\{128, 256, 350^{*}, 512\}$
- $\mathcal{D}$ dropout probability: $\{0.01, 0.1, 0.2^{*}\}$
- Type of auxiliary loss: {Standard cross-entropy, Additive margin*,†}

***Mini*ImageNet:**

- Learning rate: $\{0.5, 0.1^{\dagger}, 0.01^{*}, 0.001\}$
- Weight decay: $\{0.01, 0.001^{*}, 0.0001^{\dagger}, 0.00001\}$
- Momentum: $\{0.1, 0.5, 0.9^{*,\dagger}\}$
- Label smoothing: $\{0.01, 0.1^{*}, 0.2^{\dagger}\}$
- Patience: $\{1, 5, 10^{*,\dagger}, 20\}$
- Number of $\mathcal{D}$ hidden layers: $\{2, 3^{*}, 4, 5\}$
- Size of $\mathcal{D}$ hidden layers: $\{128, 256, 350^{*}, 512\}$
- $\mathcal{D}$ dropout probability: $\{0.01, 0.1^{*}, 0.2\}$
- Type of auxiliary loss: {Standard cross-entropy, Additive margin*,†}

5.6.3 Voxceleb

5.6.3.1 Encoder architecture

We implement $\mathcal{E}$ as the well-known TDNN architecture employed within the x-vector setting [3], which consists of a sequence of dilated 1-dimensional convolutions across the temporal dimension, followed by a time pooling layer, which simply concatenates element-wise first- and second-order statistics over time. Concatenated statistics are finally projected into an output vector through two fully-connected layers. Pre-activation batch normalization is performed after each convolution and fully-connected layer. A summary of the employed architecture is shown in Table 1.1.

5.6.3.2 Data augmentation and feature extraction

We augment the training data by simulating diverse acoustic conditions using supplementary noisy speech, as done in [3]. More specifically, we corrupt the original samples by adding reverberation (reverberation time varies from 0.25 s - 0.75 s) and background noise, such as music (signal-to-noise ratio, SNR, within 5-15 dB), and babble (SNR varies from 10 dB to 20 dB). Noise signals were selected from the MUSAN corpus [136] and the room impulse responses samples from [178] were used to simulate reverberation. All the audio pre-processing steps including feature extraction, degradation with noise as well as silence frames removal was performed with the Kaldi toolkit [161] and are openly available as the first step of the recipe in

https://github.com/kaldi-asr/kaldi/tree/master/egs/voxceleb. The corpora used for augmentation are also openly available at https://www.openslr.org/.

In order to deal with recordings of varying duration within a mini-batch, we pad all recordings to a maximum duration set in advance. We do so by repeating the signal up until it reaches the maximum duration or taking a random continuous chunk with the maximum duration for the case of long utterances.

5.6.3.3 Mini-batch construction

Given the large number of classes in the VoxCeleb case (corresponding to the number of speakers, i.e., 5994), we need to ensure several examples belonging to the same speaker exist in a mini-batch to allow for positive pairs to exist. We thus create a list of sets of five recordings belonging to the same speaker, and such sets are randomly selected at training time. Mini-batches are constructed through sequentially picking examples from the list, and the list is recreated once all elements are sampled. Such approach provides mini-batches of size $N_e = S \cdot R$, where R and S correspond to the number of speakers per mini-batch and recordings per speaker, respectively. While R is set to 5, S is set to 24, which gives an effective mini-batch size of $N_e = 120$.

5.6.3.4 Hyperparameters

Training was carried out with a linear learning rate warm-up, employed during the first iterations, and the same exponential decay as in [48] is employed after that. A budget of 40 runs was considered and each model was trained for a budget of 600k iterations. The best set of hyperparameters, as assessed in terms of EER measured over a random set of trials created from **VoxCeleb1-E**, was then used to train a model from scratch for a total of 2M iterations. We report the results obtained by the best model within the 2M iterations in terms of the same metric used during the hyperparameter search. Selected values are indicated by *.

The grid used for the hyperparameter search is presented next. In all experiments, the mini-batch size was set to 24, which, given the sampling strategy employed in this case, yields an effective batch size of 120. We further employed gradient clipping and searched over possible clipping thresholds.

- Base learning rate: $\{2.0, 1.5^*, 1.0, 0.5, 0.1\}$
- Weight decay: $\{0.001^*, 0.0001, 0.00001\}$
- Momentum: $\{0.7, 0.85, 0.95^*\}$
- Label smoothing: $\{0.0, 0.1^*, 0.2]\}$
- Embedding size d: $\{128, 256^*, 512\}$
- Maximum duration (in number of frames): $\{300, 500, 800^*\}$
- Gradient clipping threshold: $\{10^*, 20, 50\}$
- Number of $\mathcal{D}$ hidden layers: $\{1, 2, 3, 4^*\}$
- Size of $\mathcal{D}$ hidden layers: $\{128, 256^*, 350, 512\}$
- $\mathcal{D}$ dropout probability: $\{0.01, 0.1^*, 0.2\}$
- Type of auxiliary loss: {Standard cross-entropy, Additive margin*}

5.7 Conclusion

We introduced a setting where small sample 2-sample tests can be performed efficiently. This is suitable and compare data pairs to determine whether they belong to the same class. Several interpretations of such framework are provided, including joint encoder and distance metric learning, as well as contrastive estimation over data pairs. We used contrastive estimation results to show the solutions of the posed problem yield optimal decision rules under verification settings, resulting in correct decisions for any choice of threshold. In terms of practical contributions, the proposed method simplifies both the training under the metric learning framework, as it does not require any scheme to select negative pairs of examples, and also simplifies verification pipelines, which are usually made up of several individual components, each one contributing specific challenges at training and testing phases. Our models can be used in an end-to-end manner by using $\mathcal{D}$'s outputs to score test trials yielding strong performance even in large scale and realistic open set conditions where test classes are different from those seen at training time[7].

[7] Code for reproducing our experiments can be found at: `https://github.com/joaomonteirof/e2e_verification`

Chapter 6

Learning partitions to define versatile learning templates

6.1 Preamble

This chapter is compiled from material extracted from [179], under review in the *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

6.2 Introduction

In this contribution, we leverage the framework introduced in Chapter 5 and build upon it so as to define the idea of TEMPlate LEarners (TEMPLE): a set of model components and a training procedure that can be re-used across different scenarios to reach similar performance to that of task-specific models yielding a *versatile/general-purpose* modeling framework. To do so, we again make use of *metric learning* methods, but also consider the geometric approaches originally introduced to tackle few-shot-classification; *prototypical networks* in particular [83]. More specifically, we focus on metric learning settings where both an encoder and a similarity or distance models are trained jointly [84, 85, 86], but augment such setting with a set of class prototypes used in order to assign points to classes. TEMPLE models comprise three main components:

1. An encoder that embeds data into a lower-dimensional space;
2. A similarity model which maps a pair of concatenated representations into a similarity score;
3. A set of class prototypes where each one summarizes a whole class into a vector in the embedding space.

Based on models defined by said components, we can then devise different inference mechanisms depending on the task of interest. For the case of multi-class classification, for example, one can predict the class of a particular test instance through measuring its similarity against each prototype and assigning it to the class whose prototype it is most similar to. Similarly, tasks relying on pairwise comparisons can be performed, such as verification (comparing two data instances and determining whether they belong to the same class) or retrieval (comparing a test instance against a gallery and determining the k elements in the gallery the considered test instance is most similar to). Moreover, each time new classes appear, adapting the model simply consists of updating the list of prototypes, while keeping the encoder and similarity unchanged, thus enabling fast adaptation and avoiding issues, such as forgetting past classes or overfitting to the new ones.

We highlight that, given the broad applicability of TEMPLE, unlike in previous chapters, here we broaden our scope and evaluate our proposals on tasks beyond applications related to voice biometrics. In particular, we use both ASV experiments and also standard benchmarks on images to provide empirical evidence and support the claim that models defined by TEMPLE perform on par with task-specific specialized models across different scenarios and types of data. In addition to that, we observed that classifiers defined under this setting result in improved robustness against adversarial attacks and covariate shifts between training and testing data distributions. Moreover, the proposed approach supports the inclusion of new classes appearing after training is complete, which simply requires including new prototypes (or repartitioning the space) obtained from small samples. Doing so yields a simple yet competitive mechanism for few-shot classification.

6.3 Background

A relatively recent research direction within the space of metric learning settings consists of jointly training an encoder along with a distance/similarity model. This is the case discussed in [84], where a symmetric model was used to map the absolute difference of a pair of representations

into a similarity score. In [85], training of a distance/similarity model is done by imitation learning of cosine similarities measured between representations, which the authors claim simplifies training compared to the direct use of cosine scores. The approach in [60], in turn, focuses on distance models supporting asymmetric properties of the data, while still satisfying the triangle inequality. Learned Bregman divergences were evaluated in [180]. Completely unconstrained similarity models, in the sense that none of the properties, such as symmetry, are imposed in the learned distance, were proposed in [86] and discussed in Chapter 5 specifically for verification tasks. Learnable similarities parameterized by neural networks were further employed by [181, 182] for the implementation of learned kernels, and used to perform 2-sample tests based in the Maximum Mean Discrepancy (MMD) [183].

Prototypical networks [83], introduced as a geometric approach to tackle the case of few-shot classification, follow a similar idea to that of metric learning in the sense that training consists of building a metric space, where distances are indicative of properties of the data. However, performing inference using such models requires partitioning spaces using class prototypes, i.e., a set of vectors representing each class, thus enabling its use to classification tasks since one can assign a test instance to the class corresponding to its closest prototype. Employing this approach under few-shot classification settings requires a new partitioning of the space to be computed once small samples representing new classes are presented to the model. To define TEMPLE, we propose a strategy to extend the setting proposed in [86] and include a partitioning with prototypes in the learned *pseudo* metric space so that the final model can be used to perform tasks, such as multi-class and few-shot classification, while still supporting tasks involving pairwise comparisons.

6.4 Defining learning templates via trainable similarity measures

We use the term *template learner* – TEMPLE for short – to refer to a set of model components combined with a standardized learning procedure. In this case, *template* is meant to indicate that TEMPLE can be re-used across different types of data and tasks, and it can be applied pending implementation of the model components tailored to the task at hand, i.e., a task-specific encoder needs to be implemented. We argue that defining such a type of general purpose learning scheme can facilitate the practical application of learning techniques, given that training models on new data sources and for new tasks simply requires defining a specific architecture for a component

prior to running a standardized training procedure. This does not require designers to concern themselves with aspects, such as determining suitable loss functions or data sampling strategies, for instance. AutoML [184] approaches can also benefit from such type of standardized building blocks given the reduced design search space it yields. Moreover, combining TEMPLE with cross-modality standardized architectures, such as the recently introduced Perceivers [185, 186], can further simplify re-using templates across very different types of data and tasks. In what follows, we introduce in further detail the model components and the training procedure that define TEMPLE, and discuss inference approaches that can be implemented on top of such models to solve diverse tasks.

6.4.1 Model components

Let pairs (x, y) represent instances from $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^D$ indicates the input space while $\mathcal{Y}$ corresponds to the space of labels, which will often be defined by discrete sets in the cases considered herein. The following components are to be considered:

1. An *encoder* $\mathcal{E} : \mathbb{R}^D \mapsto \mathbb{R}^d$ responsible for mapping data to lower-dimensional representations.
2. A *similarity* model $\mathcal{S} : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ which maps a pair of representations to similarity scores.
3. A set of *prototypes* $\mathcal{C} \in \mathbb{R}^{|\mathcal{Y}| \times d}$ each one corresponding to a vector representing an element in $\mathcal{Y}$.

As will be further discussed, these three components can be used to perform different types of inference regarding properties of underlying data, and thus solve different tasks.

6.4.2 Training

Training is carried out to enforce the following properties:

1. The similarity as measured between a particular example and the prototype corresponding to its class labels should be high relative to similarities measured between prototypes representing other classes.
2. The similarities measured between examples from the same class should be high, while examples from different classes should yield a low similarity score.

We design training objectives aimed at enforcing such properties. For the first property, we consider a training sample of size m and employ the standard multi-class cross-entropy criterion, but *use the similarity measured between a training instance and each prototype as the set of logits*, as opposed to output layers defined by an affine transformation, commonly employed in standard classifiers. More specifically, we perform maximum likelihood estimation on the multinoulli conditional distribution defined by:

$$P(\mathcal{Y}|x') = \text{softmax}(\mathcal{S}(\mathcal{E}(x'), \mathcal{C}_{1:|\mathcal{Y}|})), \tag{6.1}$$

where $\mathcal{C}_i$, $i \in [|\mathcal{Y}|]$, indicates the prototype corresponding to class i. The corresponding training loss, denoted $\mathcal{L}_{class}$, will be then given by:

$$\mathcal{L}_{class} = -\frac{1}{m}\sum_{i=1}^{m} \log \frac{e^{\mathcal{S}(\mathcal{E}(x_i), \mathcal{C}_{y_i})}}{\sum_{j=1}^{|\mathcal{Y}|} e^{\mathcal{S}(\mathcal{E}(x_i), \mathcal{C}_j)}}, \tag{6.2}$$

where x_i and y_i indicate the i-th training example. In order for the learned similarity to be *meaningful* for pairwise comparisons, we make use of a binary classification objective also used by [87] and [86]. This classification is aimed at discriminating pairs of examples from the same and from different classes as per:

$$\mathcal{L}_{pair} = -\frac{1}{|\mathcal{T}^+|}\sum_{x^+ \in \mathcal{T}^+} \log(\sigma(\mathcal{S}(\mathcal{E}(x^+)))) - \frac{1}{|\mathcal{T}^-|}\sum_{x^- \in \mathcal{T}^-} \log(1 - \sigma(\mathcal{S}(\mathcal{E}(x^-)))), \tag{6.3}$$

where σ stands for the logistic function, and x^+ and x^- indicate pairs of examples denominated *trials* and denoted by $\mathcal{T}$, i.e., $\mathcal{T} = \{x', x''\}$. The sums are taken over the set of positive or target trials $\mathcal{T}^+$ obtained from the training sample, i.e., those for which x' and x'' belong to the same class, and the set of negative or non-target trials $\mathcal{T}^-$. Note that we further define the application of the encoder over a trial, which is denoted as $\mathcal{E}(\mathcal{T}) = \{\mathcal{E}(x'), \mathcal{E}(x'')\}$.

Initializing and updating the list of prototypes: $\mathcal{C}$ is initialized randomly such that its entries are i.i.d. sampled from a standard Gaussian distribution. We update $\mathcal{C}$ every iteration through a moving average which is given by the following at iteration t: $\mathcal{C}_t = \lambda\mathcal{C}_{t-1} + (1-\lambda)\mathcal{C}^*_{t-1}$, where $\lambda \in [0, 1]$ is a hyperparameter, and $\mathcal{C}^*_{t-1}$ is a copy of $\mathcal{C}_{t-1}$, where the rows corresponding to classes observed in the current mini-batch are substituted by the average representations of each such classes.

Practical details: Both $\mathcal{E}$ and $\mathcal{S}$ are implemented as neural networks, and $\mathcal{C}$ is a matrix where each row represents a prototype for a particular class. Moreover, while $\mathcal{E}$ is task-specific, $\mathcal{S}$ corresponds to a fully-connected network. As usual, training is carried out with SGD with gradients estimated over mini-batches of training data. In order to compute $\mathcal{L}_{pair}$, each mini-batch needs to contain multiple examples from the same class otherwise T^+ will be empty. We sample mini-batches ensuring that is the case (c.f. Section 6.5.3 for further implementation details). Finally, we empirically observed that including a standard classification loss accelerates convergence across all evaluations performed. We include a dense output layer to allow for computation of such a loss, which we denote by $\mathcal{L}_{aux}$. Training is carried out to minimize the total loss $\mathcal{L} = \mathcal{L}_{class} + \mathcal{L}_{pair} + \mathcal{L}_{aux}$. A high-level training procedure is depicted in Algorithm 3, and model components are illustrated in Figure 6.1.

Algorithm 3 Training procedure.

$\mathcal{E}, \mathcal{S} = InitializeModels()$
$\mathcal{C} = InitializePrototypes()$
repeat
  $x, y = SampleMinibatch()$
  $z = \mathcal{E}(x)$
  $\mathcal{C} = UpdatePrototypes(z, y, \mathcal{C})$
  $z^+ = GetPositivePairs(z, y)$
  $z^- = GetNegativePairs(z, y)$
  $y' = DenseLayer(z)$ `# Used for computing auxiliary loss.`
  $\mathcal{L} = \mathcal{L}_{pair} + \mathcal{L}_{class} + \mathcal{L}_{aux}$
  $\mathcal{E}, \mathcal{S} = UpdateRule(\mathcal{E}, \mathcal{S}, \mathcal{L})$ `# i.e., run optimizer step.`
until Maximum number of iterations reached
return $\mathcal{E}, \mathcal{S}, \mathcal{C}$

6.4.3 Testing

We now define the set of tasks one can tackle using trained $\mathcal{E}$, $\mathcal{S}$, and $\mathcal{C}$ along with the inference mechanisms employed for each such task.

Multi-class classification: For the case where one is given a test instance x' and desires to determine its class label y', this can be achieved by the following classifier:

$$\arg\max_{i \in [|\mathcal{Y}|]} \mathcal{S}(\mathcal{E}(x'), \mathcal{C}_i). \tag{6.4}$$

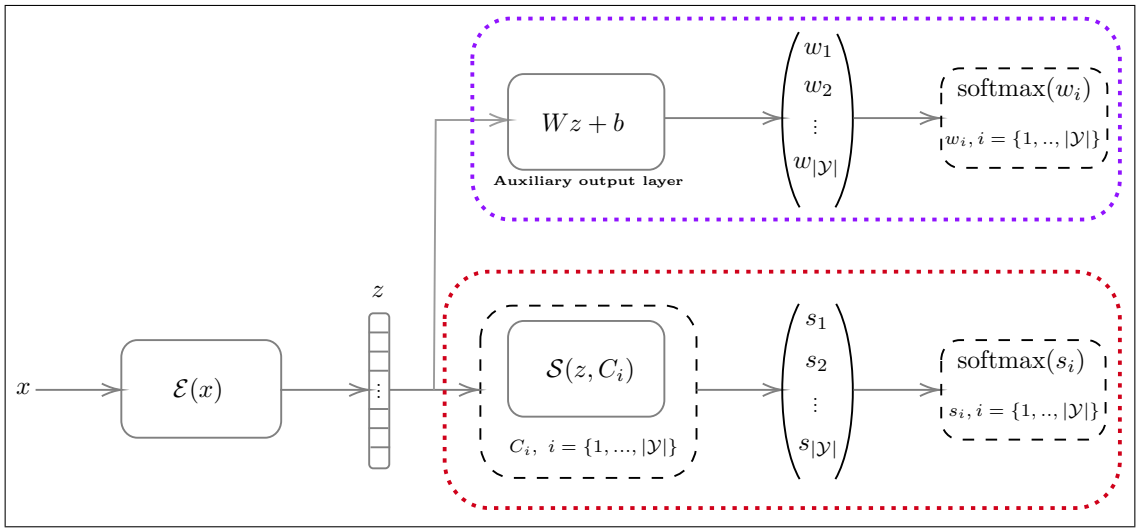


Figure 6.1 – Components defining TEMPLE models. Implementing a model for a particular task simply requires the definition of $\mathcal{E}$.

Few-shot classification: If new classes are considered after training, repartitioning can be performed with few data points by creating a new set of prototypes $\mathcal{C}'$ defined such that each entry corresponds to the average representation of each new class. Inference is thus performed following the schemed defined for multi-class classification.

Verification: Now assume one is given a trial $\{x', x''\}$ and desires to determine whether their respective labels are such that $y' = y''$. One can then do the following:

$$\mathbb{1}_{\mathcal{S}(\mathcal{E}(x'),\mathcal{E}(x''))>\delta}, \tag{6.5}$$

where δ is an user-defined decision threshold.

Retrieval: Given a test instance x' and a gallery of instances denoted by $X = \{x_1, x_2, ..., x_n\}$: $x_i \in \mathcal{X}\ \forall\ i \in [n]$, determine k elements in X such that at least one of their labels matches the underlying label y' of x'. The result will be:

$$\underset{x'' \in X}{\text{k-arg max}}\, \mathcal{S}(\mathcal{E}(x'), \mathcal{E}(x'')), \tag{6.6}$$

where the operator k-arg max denotes repeating the arg max operator k times, removing the current result each time prior to the next arg max operation.

6.5 Evaluation

In the evaluation we carry out, we seek to test the proposed approach across a variety of tasks and modalities of data since our main goal is to show evidence that TEMPLE instances perform at least on par with specialized methods considering different cases. Namely, we start with tasks that rely on pairwise comparisons of test instances and run evaluations under the verification setting focusing on an ASV task on the large scale verification setting defined by the VoxCeleb corpus [82, 7], also studied in Chapter 5. We then proceed to image benchmarks in which case, in addition to checking for prediction performance, we evaluate different notions of robustness of classifiers induced by TEMPLE. Such image benchmarks cover the following tasks: multi-class classification, in which case we evaluate TEMPLE-based classifiers on MNIST [88] and CIFAR-10 [89], and *observe improved robust accuracy* against popular norm-constrained adversaries. We further perform evaluation on object recognition tasks considering larger resolution images and under domain shift. For that, we employ the standard PACS benchmark [90], where we find that the proposed classification strategies introduced herein *outperform recently introduced alternatives*, and more importantly, that is the case in the most challenging conditions, where a notable domain mismatch is observed (e.g., train on natural images and evaluate on sketches). We then evaluate our proposed approach on image retrieval tasks employing popular benchmarks, such as CARS196 [91] and CUB200-2011 [92]. Finally, we discuss strategies to easily repartition the space so that new classes can be evaluated at testing time, in which case we report experiments using *Mini*ImageNet [93]. Ablations are further reported using the full ImageNet [94] to show the importance of the use of the auxiliary classification loss.

We remark that the training procedure presented in Algorithm 2 is employed for *training models used for all tasks discussed above*, and no specialization to any task of interest is performed since we seek evidence regarding how effective the proposed approach is in yielding a general enough set of components (i.e., $\mathcal{E}$, $\mathcal{S}$, and $\mathcal{C}$) and training algorithm which perform on par or better than alternatives. As such, across different datasets and tasks, the encoder $\mathcal{E}$ is the only component that is specific to each evaluation in the sense that its architecture requires specification for each specific data source.

6.5.1 ASV experiments on VoxCeleb

In order to perform evaluations that rely on pairwise comparisons of test instances, we consider the verification setting where trials corresponding to pairs of test instances are presented to the model. Its task is then to decide whether the examples in the trial belong to the same class. We make use of the VoxCeleb corpus [82, 7] which consists of a large scale set of audio clips collected from videos of interviews available online.

We compared models trained on the second release of the corpus, which is composed of audio recordings from 5994 different speakers, against a set of published results on the three test partitions made available along with that release. These partitions include: (**i**)-**VoxCeleb1 Test set**, which correspond to data obtained from 40 speakers, (**ii**)-**VoxCeleb1-Extended**, which is given by the complete first release of the corpus and contains 1251 speakers, and (**iii**)-**VoxCeleb1-Hard**, which is made up of a subset of the data from *VoxCeleb1-Extended* yielding trials known to be hard to distinguish. We highlight that *the set of speakers represented in all test partitions is disjoint to the set of speakers appearing in training data.* The encoder $\mathcal{E}$ in this case corresponds to the 1-dimensional convolutional model introduced by [3]. Details on the audio pre-processing and feature extraction are included in Section 6.5.3.

Results are reported in terms of EER in Table 6.1, where we compare the verification performance of different TEMPLE models using scoring strategies against a number of alternative methods. Interestingly, scores obtained via cosine similarities measured between outputs of $\mathcal{E}$ are observed to be discriminative, and in some cases even perform slightly better than learned similarities, which is not explicitly enforced by the described training algorithm. This is likely due to the type of cross-entropy criteria used to implement the auxiliary loss. In this case, the additive margin softmax [113] was used to implement classification criteria corresponding to two terms composing the training objective, and it was empirically observed to result in discriminative embeddings via inner product scores. We thus take advantage of the fact that the cosine similarity can be further used to score test trials. Specifically, we observed that combining it with learned scores given by $\mathcal{S}$ by simply summing both similarities yields a further boost in performance. In fact, such efficient combination schemes between cosine similarities and learned scores yield the best performance amongst compared methods, which include a setting where PLDA is used for scoring. We finally remark that the only difference between TEMPLE instances and the approach discussed

Table 6.1 – Verification performance on the VoxCeleb test partitions reported in terms of EER (%).

	Scoring	**EER** (%)
		VoxCeleb1 test set
Chung et al. (2018) [7]	Cosine	3.95
Xie et al. (2019) [167]	Cosine	3.22
Hajavi & Etemad (2019) [168]	Cosine	4.26
Xiang et al. (2019) [169]	Cosine	2.69
Monteiro et al. (2020) [86]	PLDA	2.51
Monteiro et al. (2020) [86]	*SIM*	2.51
Ours	Cosine	2.62
Ours	*SIM*	2.55
Ours	Cosine + *SIM*	**2.45**
		Extended
Chung et al. (2018) [7]	Cosine	4.42
Xie et al. (2019) [167]	Cosine	3.13
Xiang et al. (2019) [169]	Cosine	2.76
Monteiro et al. (2020) [86]	PLDA	2.60
Monteiro et al. (2020) [86]	*SIM*	2.57
Ours	Cosine	2.69
Ours	*SIM*	2.75
Ours	Cosine + *SIM*	**2.55**
		Hard
Chung et al. (2018) [7]	Cosine	7.33
Xie et al. (2019) [167]	Cosine	5.06
Xiang et al. (2019) [169]	Cosine	4.73
Monteiro et al. (2020) [86]	PLDA	4.62
Monteiro et al. (2020) [86]	*SIM*	4.73
Ours	Cosine	4.48
Ours	*SIM*	4.76
Ours	Cosine + *SIM*	**4.39**

in [86] and Chapter 5 is the prototypical classification loss. Comparing both approaches, however, shows such modification slightly degrades prediction performance using scores from $\mathcal{S}$, which can be seen as a side affect of the added flexibility since the model can now perform other types of tasks. Nonetheless, the simple score combination scheme recovers back the performance and yields the best verification performance we observed.

6.5.2 Experiments on image benchmarks

6.5.2.1 Robustness against adversaries

We report in Table 6.2 the accuracy obtained using convolutional classifiers trained on MNIST and CIFAR-10 considering both clean data and FGSM [153] and PGD [187] attacks[1] under L_∞ budgets. We consider the white-box access model in which the attacker has full access to the target model. Models trained using Algorithm 2 are compared against previously proposed defense strategies. Specifically, adversarial training (AT) [187], adversarial logit pairing (ALP) [188], triplet loss adversarial training (TLA) [189], and TRADES [190] are considered for comparison. The results given by an undefended model, as reported by [189], are also included for reference.

A standard LeNet and the wide residual architecture introduced by [187] were employed for the cases of MNIST and CIFAR-10, respectively, and an attack budget of 0.3 and $\frac{8}{255}$ was considered when each such dataset was evaluated. We evaluated our models both with and without adversarial training, and report the results obtained when inference is performed using the scheme represented in expression 6.4. This scheme is denoted as *SIM* to indicate that the similarity model $\mathcal{S}$ is used for inference. Moreover, for cases where the auxiliary output layer used to compute $\mathcal{L}_{aux}$ is used to predict labels of test instances is indicated by *DOL* in a reference to *dense output layer*.

In order to have a full white-box access model, each output layer is exposed to the attacker in each evaluation so that attacks are created accounting for the specific inference procedure that will be used for testing. Based on the reported results, we verify that similarity/distance based inference is inherently less affected by small norm adversarial perturbations. This could be observed in both MNIST and CIFAR-10, where our undefended models showed higher robust accuracy than standards undefendend classifiers across considered attacks, with the *SIM* case outperforming *DOL* and the undefended standard classifier. With adversarial training, both inference mechanisms yielded higher performance than the considered alternatives against considered attackers; importantly, this was achieved *without affecting the clean accuracy to the same extent as in previous methods.*

The described results lead us to the conclusion that distance-based inference is inherently less affected by small norm perturbations in inputs when compared to standard classifiers. More specifically, it is easier from the attacker's perspective to increase the value of the activation of the output

[1] Attacks were implemented using *FoolBox*: `https://foolbox.readthedocs.io/en/stable/index.html`

Table 6.2 – Adversarial robustness evaluation in term of accuracy (%) considering PGD and FGSM attackers under L_∞ budgets of 0.3 and $\frac{8}{255}$ for the cases of MNIST and CIFAR-10, respectively. The number of steps employed for each attack is represented within parenthesis. We consider evaluations obtained with the similarity classifier as indicated by *SIM* as well as utilizing the auxiliary output layer which we indicate by *DOL*.

MNIST				
	Clean	PGD (40)	PGD (100)	FGSM (1)
Undefended	99.20	0.00	0.00	34.48
AT	99.24	97.31	96.58	94.82
ALP	98.91	97.34	96.62	95.06
TLA	99.52	**98.17**	**97.72**	96.96
TRADES ($1/\lambda = 6$)	99.48	96.07	-	95.6
Ours - *DOL*	99.31	0.02	0.01	17.18
Ours - *SIM*	**99.36**	23.85	13.61	68.51
Ours - *DOL* + AT	98.71	95.04	93.78	97.62
Ours - *SIM* + AT	98.79	95.35	93.98	**97.84**
CIFAR-10				
	Clean	PGD (7)	PGD (20)	FGSM (1)
Undefended	95.01	0.00	0.00	13.35
AT	87.14	55.63	49.79	45.72
ALP	89.79	60.29	51.89	48.5
TLA	86.21	58.88	53.87	51.59
TRADES ($1/\lambda = 1$)	88.64	-	49.14	48.9
TRADES ($1/\lambda = 6$)	84.92	-	**56.61**	56.43
Ours - *DOL*	**96.20**	47.39	9.27	57.42
Ours - *SIM*	96.14	65.46	27.87	**65.46**
Ours - *DOL* + AT	93.29	**80.73**	54.29	46.85
Ours - *SIM* + AT	92.55	80.04	56.13	52.98

unit of a wrong class than it is for it to move an embedding closer to the prototype of a wrong class. Including adversarial perturbations at training time, as described in the results corresponding to adversarial training, makes it even more difficult for the attacker to be effective since larger perturbations in the input are necessary in order to move an embedding towards the prototype of a different class. It is likely that adversarial training induces a (pseudo) metric space where class centers have larger pairwise distances when compared to the standard cases, where no adversarial perturbations are presented to the model during training. We hypothesize that regularization penalties that aim to force prototypes to lie further apart are likely to increase robustness. Even more so for input perturbations bounded in norm.

Table 6.3 – Evaluation on the PACS benchmark in terms of accuracy (%) for the cases where each of the available domains are left out of training.

	P	A	C	S	Average
Dou et al. (2019) [191]	95.01	82.89	80.49	72.29	82.67
Gulrajani & Lopes-Paz (2020) [193]	**97.80**	**88.10**	78.00	79.10	85.75
Chattopadhyay et al. (2020) [192]	94.49	82.57	78.11	78.32	83.37
Ours - *SIM*	97.07	86.38	**83.66**	**84.63**	**87.93**

6.5.2.2 Robustness under domain shift

We now assess the performance of the proposed classification strategy once domain shifts across train and test data occur. We do so by making use of the PACS domain-generalization benchmark [90] consisting of 224x224 RGB images distributed into 7 classes and originated from four different *domains*: Photo (P), Art painting (A), Cartoon (C), and Sketch (S). We follow the *leave-one-domain-out* evaluation protocol such that data from three out of the four available domains are used for training while evaluation is carried out on the data from the left out domain. A comparison is carried out with recent methods specifically designed to enable out-of-distribution generalization introduced by [191] and [192], as well as with the results reported by [193], where standard classifiers were evaluated against domain generalization approaches. As per the common practice for evaluation on PACS, experiments were carried out using a ResNet-50 [4] pretrained on ImageNet.

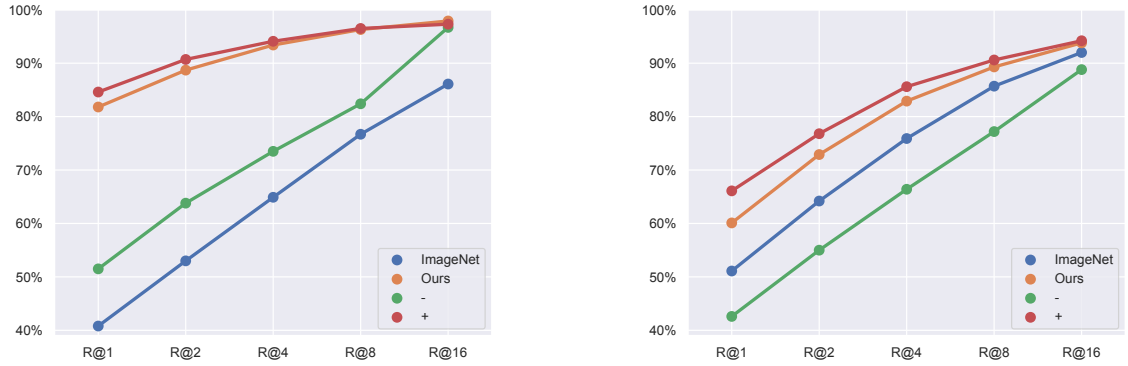
Both per-domain and averaged results are reported in Table 6.3. Considering the average of performances obtained after each domain is left out, we observe improved robustness when similarity-based classification is employed when compared to both standard classifiers and domain generalization approaches. This suggests that distance-based TEMPLE classifiers rely less on domain-specific factors that might correlate with labels on training domains. We hypothesize that such property comes from the metric learning framework used to train our models, i.e., domain-specific information is less helpful when trying to minimize the combination of $\mathcal{L}_{class}$ and $\mathcal{L}_{pair}$, which renders resulting classifiers less dependent on the underlying domains used at training time. A gap in performance in our favor can be particularly observed for the evaluation cases where domains corresponding to cartoons and sketches are left out, given that such domains present a large discrepancy compared to the natural images that compose the bulk of training data. On the other hand, for the photos category in which the underlying data correspond to natural images, the standard classifier discussed in [193] outperforms our model.

6.5.2.3 Image retrieval

We further verified the performance of the proposed approach on another set of tasks which require comparisons of pairs of examples. In this case, we considered the retrieval setting where a test example is compared against a gallery and k "similar" examples need to be selected from that gallery. We thus make use of the CARS196 [91] and CUB200-2011 [92] datasets and closely follow the evaluation protocol discussed by [78].

Results are reported in terms of *Recall@k* [194] (the higher the better), or *R@k* for short, and summarized in Figures 6.2a and 6.2b, while the complete set of results is reported in Tables 6.4 and 6.5, respectively. Given a data point, *Recall@k* measures the probability of its set of top-k most similar test instances to contain at least one exemplar of its underlying class. Compared approaches consist of several metric learning methods specifically designed for the retrieval problem. We use the indicators + and - to refer to the highest and lowest performances amongst the considered baselines. Results were obtained considering a ResNet-50 pretrained on ImageNet and fine tuned on each of the considered datasets. As a reference, we further report the results obtained by the pretrained model prior to fine tuning. As can be seen, the proposed approach is competitive in that its performance lies close to the + line (which corresponds to a strong baseline using an ensemble of metrics approach [195]) and outperforms most of the compared methods. In fact, this is achieved while relying on a simpler and general training procedure that, unlike specialized approaches, do not require any special mining strategy of hard triplets, and use moderate batch sizes, thus enabling practical training in single GPU hardware.

In summary, based on the discussed set of results we highlight that, both in the cases of tasks that require instance-to-instance comparisons (verification and retrieval) and those that require instance-to-sample similarity assessments (classification), TEMPLE models were observed to be competitive in terms of prediction performance relative to task-specific approaches, while being simple and general. Moreover, distance-based prediction was observed to be more robust to both distribution shifts and norm-bounded adversarial perturbations. In what follows, we discuss a final application of TEMPLE where new classes appear at testing time. We discuss a simple approach to update $\mathcal{C}$ and enable prediction from exemplars of those new classes.



(a) *R@K* evaluation of TEMPLE models on the CARS196 dataset.

(b) *R@K* evaluation of TEMPLE models on the CUB200-2011 dataset.

Figure 6.2 – Evaluation on retrieval tasks in terms of *R@K*.

Table 6.4 – *R@K* (%) evaluation of proposed methods on the CARS196 dataset.

	R@1	R@2	R@4	R@8	R@16
Schroff et al. (2015) [62]	51.5	63.8	73.5	82.4	–
Oh Song et al. (2019) [194]	53.0	65.7	76.0	84.3	–
Song et al. (2016) [196]	58.1	70.6	80.3	87.8	–
Sohn (2016) [197]	71.1	79.7	86.5	91.6	–
Yuan et al. (2017) [198]	73.7	83.2	89.5	93.8	96.7
Wu et al. (2017) [78]	79.6	86.5	91.9	95.1	97.3
Roth et al. (2019) [199]	82.6	89.1	93.2	–	–
ImageNet	40.8	53.0	64.9	76.7	86.1
SIM	81.8	88.7	93.4	96.3	97.9
Sanakoyeu et al. (2019) [195] (Ensemble of metrics)	84.6	90.7	94.1	96.5	–

6.5.2.4 Few-shot classification

We further evaluate the proposed framework under the few-shot classification setting in which case new classes are presented to the model after training, and small samples from each novel class are made available. We run evaluations considering the *Mini*ImageNet [93] dataset which consists of a subset of 100 classes from ImageNet containing 600 images for each class. We follow the setting introduced by [201], which splits the data so that 64, 16, and 20 disjoint sets of classes are included in the training, validation, and testing partitions, respectively.

Results are reported in terms of average top-1 accuracy along with boundaries of its 95% confidence interval considering a sample of 1000 randomly selected tasks. Each task is randomly created by giving to the model a set containing *N-ways* classes and *K-shots* examples per class, while 15 test

Table 6.5 – *R@K* (%) evaluation of proposed methods on the CUB200-2011 dataset.

	R@1	R@2	R@4	R@8	R@16
Ustinova & Lempitsky (2016) [200]	52.8	64.4	74.7	83.9	90.4
Ustinova & Lempitsky (2016) [200]	50.3	61.9	72.6	82.4	88.8
Schroff et al. (2015) [62]	42.6	55.0	66.4	77.2	–
Oh Song et al. (2019) [194]	43.6	56.6	68.6	79.6	–
Song et al. (2016) [196]	48.2	61.4	71.8	81.9	–
Sohn (2016) [197]	51.0	63.3	74.3	83.2	–
Yuan et al. (2017) [198]	53.6	65.7	77.0	85.6	91.5
Wu et al. (2017) [78]	63.6	74.4	83.1	90.0	94.2
Roth et al. (2019) [199]	66.1	76.8	85.6	–	–
ImageNet	51.1	64.6	75.9	85.7	92.0
SIM	60.1	72.9	82.9	89.3	93.8
Sanakoyeu et al. (2019) [195] (Ensemble of metrics)	65.9	76.6	84.4	90.6	–

examples per class are evaluated in each task. The encoder $\mathcal{E}$ is implemented as the convolutional stack of the ResNet-12 architecture, which is also used in all the compared approaches.

A comparison is carried out with a set of approaches carefully designed for the few-shot classification setting, sometimes including sophisticated adaptation schemes for the novel classes and having the evaluation process simulated at training time with the so-called *episodic training*. For the case of our model, on the other hand, we intend to verify how simply training it using Algorithm 2 fares against such specialized methods. As such, we train our models on the training partition of *Mini*ImageNet, and once data from novel classes is given at testing time, we make use of it to build a novel set of class prototypes $\mathcal{C}'$ which is then used to define the classifier expressed in (6.4), i.e., no fine tuning of $\mathcal{E}$ and $\mathcal{S}$ is performed.

Results reported in Table 6.6 show that such a simple approach can yield performance inline with recent sophisticated approaches. We additionally computed the accuracy yielded by substituting $\mathcal{S}$ by the cosine similarity and observed that summing both scores resulted in an accuracy gain for the 1-shot case. Observed results suggest that prototypes defined by simple statistics of embeddings from new classes are positioned far from the prototypes defined during training, which makes it possible to perform classification of examples from those new classes without assigning them to classes observed during training. We remark that more complex schemes were tested where the prototypes matrix $\mathcal{C}'$ of classes appearing at testing time were learned. More specifically, we froze the parameters of $\mathcal{E}$ and $\mathcal{S}$ and learned $\mathcal{C}'$ to minimize a cross-entropy criterion. We evaluated randomly initializing $\mathcal{C}'$ as well as initializing it from statistics of the test data. However, doing

Table 6.6 – 5-way few-shot classification on *mini*ImageNet. Results consist of average top-1 accuracy along with confidence intervals considering 1000 randomly selected tasks. All evaluations consider a ResNet-12 architecture.

	1-shot	**5-shots**
MatchNet [93]	**63.08±0.80**	75.99±0.60
SNAIL [202]	55.71±0.99	68.88±0.92
AdaResNet [203]	56.88±0.62	71.94±0.57
TADAM [204]	58.50±0.30	76.70±0.30
MetaOptNet [205]	62.64±0.61	**78.63±0.46**
Prototypical Networks [205]	59.25±0.64	75.60±0.48
Ours - Cosine	59.00±0.65	77.52±0.49
Ours - *SIM*	60.28±0.65	75.26±0.52
Ours - Cosine + *SIM*	61.27±0.65	76.83±0.50

Table 6.7 – Classification performance in terms of accuracy (%).

	Top-1	**Top-5**
DOL	**73.15**	**91.26**
SIM	71.33	90.33
Ablation	70.37	89.85

so resulted in severe overfitting and performance degradation. Similarly to what is discussed for results adversarial robustness in Section 6.5.2.1, i.e., constraining $\mathcal{C}$ so that prototypes are pairwise distant might be beneficial as a regularization strategy for fine-tuning $\mathcal{C}'$ while avoiding overfitting.

6.5.2.5 Ablation study

We performed ablations using the full ImageNet in order to assess the importance of the use of the auxiliary loss $\mathcal{L}_{aux}$. We trained models with and without $\mathcal{L}_{aux}$ in the training partition of ImageNet using Algorithm 2 and performed evaluations in terms of multi-class classification performance and verification. Results for each case are reported in Tables 6.7 and 6.8. Here, $\mathcal{E}$ was implemented as the convolutional stack of a ResNet-50. For the classification case, performance is assessed in terms of top-1 and top-5 accuracy, while EER and the area above the operation curve (i.e., 1-AUC) are reported for the case of verification. Verification trials were defined by creating all possible pairs of examples out of the test data. In both the cases, removing the auxiliary output layer negatively affected performance, more notably so in the case of verification using cosine scores. Since such an auxiliary layer does not add any significant cost and boosts performance across considered evaluations, its use is justified.

Table 6.8 – Verification performance in terms of EER (%) and 1-AUC (%).

	Scoring	EER	1-AUC
Complete model	*SIM*	**3.54**	**0.63**
Complete model	Cosine	5.33	1.60
Ablation	*SIM*	4.02	0.75
Ablation	Cosine	10.06	3.93

6.5.3 Implementation details

Training: Training was carried out with SGD with momentum for most of the experiments, except for the case of MNIST, where Adam with default hyperparameters was employed, and retrieval where Adam was employed with an initial learning rate of $1e-5$ reduced by a factor of 0.1 on epochs 20 and 50, and hyperparameters, such as β_1 and β_2 were set to 0.5 and 0.999, respectively. Standard schedules for the learning rate were employed for the case of CIFAR-10, consisting of a reduction by a factor of 0.1 every 30 epochs, while a decay every 10 epochs was used for ImageNet experiments, in which case pretrained encoders were employed to speed up convergence. For the specific case of VoxCeleb, we found the additive margin softmax [113] yielded higher performance when used as an auxiliary loss, and in that case the learning rate schedule introduced in [48] was employed. We set $\lambda = 0.9$ across all experiments. Regularization strategies, such as weight decay and label smoothing [81] were further employed.

Mini-batch construction for cases corresponding to large label sets: In cases where the size of the label set $|\mathcal{Y}|$ was larger than the batch sizes being employed, which was the case for experiments performed with ImageNet, VoxCeleb, and the retrieval datasets, we needed to define a sampling strategy which would ensure that multiple examples from observed classes would appear in each mini-batch so as to enable the definition of target trials T^+ in order to compute $\mathcal{L}_{pair}$, as per its definition in Equation 6.3. We thus build mini-batches such that 5 examples per class are included for each class observed, and we make sure to update such groups of 5 examples every epoch to allow for diverse minibatches throughout training.

Data preparation for verification experiments on VoxCeleb: The audio data from VoxCeleb is augmented following the procedure discussed in [3]. We thus add reverberation, using reverberation times within 0.25s - 0.75s, and further add background noise consisting of music

samples (SNR within 5-15dB), and babble samples (SNR within 10-20dB). Noise samples were picked from the MUSAN corpus [136] while room impulse responses used to simulate reverberation were picked from [178]. The data used for distortions of the original audio are available at `https://www.openslr.org/`. Features of audio are extracted such that 30 mel frequency cepstral coefficients are obtained with a short-time Fourier transform using a 25ms Hamming window with 60% overlap. Audio is downsampled to 16kHz and a simple energy voice activity detector filters out silent frames.

6.6 Conclusion

We introduced TEMPLE: a set of model components and a training procedure which simplifies re-using of learning procedures across different types of tasks and data. Model components are given by the following: an encoder responsible for embedding data into a lower-dimensional space, a similarity model which outputs a similarity score when given a pair of embeddings, and a set of class prototypes where each one represents a class observed during training. At testing time, different inference schemes can be defined on top of said components so as to enable its use across different settings. We presented empirical evidence showing classifiers defined under TEMPLE to yield improvements including: (1) improved adversarial robustness, since small perturbations in norm were observed to have a lesser effect on distance-based inference compared to standard classifiers. (2) improved robustness against distribution shifts, which indicates the proposed training strategy is more effective in avoiding models that rely on correlations between domain-specific factors and labels. In this case, domain information is not as helpful as it can be for the case of maximum likelihood estimation with standard classifiers. Moreover, performance across a set of tasks, such as verification and image retrieval further showed TEMPLE models to perform competitively with or better than alternatives designed targeting those particular applications.

Chapter 7

Conclusions and Future Research Directions

7.1 Conclusions

In this dissertation, we proposed several methods to improve the performance of models based on neural networks specifically targeting voice biometrics and other related tasks, such as spoken language identification. Moreover, we further consider the task of detecting spoofing attackers to such systems. In addition, more broadly applicable proposals are discussed in the final chapters, and those have potential applications outside the speech domain, and hence, image applications are also explored, including object recognition from images, image retrieval, and robustness against adversarial perturbations. As such, we present the thesis conclusions under two parts: *applied contributions* comprising chapters 2- 4 focusing on the cases of automatic speaker verification (ASV), language identification (LID), and spoofing countermeasures. The second part corresponds to more *fundamental contributions* in the sense that they are applicable across domains and include Chapters 5 and 6.

7.1.1 Applied contributions

We provided contributions in terms of both training design and model architectures resulting in improved language- and speaker-dependent utterance-level representations. In particular, we introduced a multi-task scheme where a combination of metric learning and maximum likelihood estimation is used as a training signal for neural networks. Additional practical training strategies are further introduced including data augmentation schemes and sampling approaches resulting in mini-batches containing informative pairs of examples. Entropy regularization is also discussed and observed to improve performance. Results in terms of both geometric properties of the embedding space and prediction performance support the claim that the proposed framework is effective in learning discriminative representations. Specifically, evaluation is performed on non-trivial conditions, such as short-duration speech segments and confusing languages (LID), as well as the setting containing language mismatches for ASV. Obtained results further indicate the robustness and improved accuracy of the proposed approach relative to several well-known benchmarks. An ablation study further showed that the proposed combination of maximum likelihood and metric learning outperformed models trained individually under each criterion. These findings suggest that the proposed method combines the relative easiness of training under the maximum likelihood setting, as well as the discriminability imposed on representations obtained with metric learning.

For spoofing detection, in turn, we introduced model architectures, as well as training procedures resulting in predictors that operate in an end-to-end fashion, i.e., our models directly map speech features into scores. We proposed variations to the light convolutional neural network architecture by adding an attention layer to detect attacks of arbitrary duration. Performance was validated on the ASVspoof 2019 challenge data, where we showed significant detection performance improvements with respect to well-known reference baseline systems.

In addition, we evaluated different training-data augmentation strategies in order to allow for more complex models to be found. A challenge with data augmentation for spoofing is that signal transformations should not mask or remove artifacts introduced by the attack strategies, given that those are needed for spoofing detection. We found that speed perturbations and bandpass filtering satisfied these requirements for replay attacks, yielding a simple and efficient approach to increase the amount of data five-fold and improve the diversity of the available train data. In turn, simpler trimming methods across time showed to be helpful for the case of logical access attacks.

It is important to emphasize that while our experiments considered an offline data augmentation approach (i.e., corrupted copies of the data were generated prior to actually training models), the proposed transformations are simple enough to be generated on-the-fly while models are being updated. This can potentially further increase the diversity of train data and make its size virtually unbounded, as data instances appear in a different version every time they are sampled, i.e., each time a particular recording is selected during training, it appears different to the model due to the random nature of the transformations we apply.

We also empirically observed that the best speech representation for spoofing detection depended on the spoofing strategy used. For example, product spectra was the best choice for replay attacks, whereas LFCCs were more helpful for the case of logical attacks. As such, we proposed an ensemble-based approach with the goal of enabling detectors to be effective across varying types of spoofing attacks to speaker verification systems. The ensemble relied on three components, where two were optimized per attack and the third to decide how to best combine their decisions based on the input speech signal. Evaluation on the ASVspoof 2019 challenge data showed that the proposed method (i) outperformed methods that combine data from both the attack types (multi-condition training) and train a single model, as well as (ii) achieved competitive performance when compared to specialized models trained on data curated to match the evaluation condition known in advance. In fact, for the specific case of PA attacks, our proposed model outperformed the privileged specialized systems. Given that most of the current work on countermeasures for spoofing attacks focuses on specialized systems for particular types of attack strategies, we believe the approach proposed herein is a first step in the direction of enabling automatic speaker verification systems to be deployed free of the risk of security breaches, since it performs well across the different attack strategies.

Finally, we introduced a variation of the TDNN architecture – ubiquitous within the context of voice biometrics – where temporal pooling operations are performed across all layers of the convolutional stack, rather than only at its end. We term the proposed architecture ML-TDNN (multi-level self-attention TDNN). In particular, we propose a model component aimed at combining global representations from different layers by treating global statistics computed in different parts of the model as sequences. We then use a self-attentive layer to process these sequences and finally obtain an utterance-level representation from the processed sequence of global statistics. Experiments are performed on three tasks: ASV, LID, and spoofing detection. Results showed that the proposed method consistently outperformed various benchmarks that use global features obtained

from a single layer. We conclude that complementary information can be efficiently leveraged from low-level layers, i.e., those close to the inputs. Moreover, the proposed architecture was observed to perform at least as well as specialized models, designed to target a particular task. This suggests that the proposed ML-TDNNs are versatile enough to be re-used across settings without specific adaptations. In terms of limitations, we highlight the fact that the obtained versatility comes at an additional computational cost, as pooling operations across several layers and their combinations incur additional computations. As such, resource-constrained settings, such as on-device applications might not directly benefit from this type of model. Notwithstanding, bootstrapping mechanisms for model compression or knowledge distillation [75] could be used to alleviate this issue.

7.1.2 Fundamental contributions

We proposed an augmented metric learning framework where an encoder and a (pseudo) distance are trained jointly. This pair of components defines a (pseudo) metric space, where inference can be performed efficiently for verification. We use the term *pseudo* to indicate that such component only needs to resemble a distance, while the proposed approach does not require it to satisfy properties of actual distances. We then pose an estimation problem that results on (pseudo) metric spaces where semantic relationships between data points, as defined via labels, can be assessed via distance measures. In particular, we leverage contrastive estimation results to show that optimal solutions of the posed problem are such that: (i) the optimal distance model for any fixed encoder yields the likelihood-ratio for a Neyman-Pearson likelihood ratio test, and (ii) the optimal encoder induces a high Jensen-Shannon divergence between the joint distributions of positive and negative pairs of examples. These results imply that, for optimal encoder and distance models, using measures of distances induced by the learned models result in optimal decision rules for verification settings, resulting in correct decisions for any choice of positive thresholds[1]. For a practical implementation of our proposed method, we parameterized both components as neural networks. Moreover, a simple training algorithm was proposed, which does not require cumbersome steps, such as hard-negative mining, often needed in standard metric learning cases. Evaluations on large scale verification tasks provide empirical evidence of the effectiveness of directly using outputs of the learned distance for inference, outperforming commonly used downstream classifiers.

[1] Scores are calibrated in this case.

Finally, we leveraged the approach discussed above and extended it so that other types of tasks can be supported, in addition to verification. In particular, we introduced a third component corresponding to a set of class prototypes: a set of vectors, each one representing a particular class. With the three components, we introduced TEMPlate LEarners (TEMPLE), a set of model components accompanied by a training procedure that can be re-used for different tasks and types of data. Under TEMPLE, defining a model on a particular data source requires simply implementing an encoding procedure for that particular case. Inference schemes were then introduced on top of TEMPLE models, enabling their use for cases that require *instance-to-instance* comparisons, such as verification and retrieval, as well as those relying on *instance-to-sample* similarity assessment, such as in the case of prototypical classification.

Empirical evaluation provides evidence that the proposed setting is able to match the performance of specialized approaches across a diverse set of benchmarks. Moreover, classifiers defined under this setting resulted in improvements in: (i) adversarial robustness, since small perturbations in norm were observed to have a smaller effect on distance-based inference compared to standard classification strategies, and (ii) robustness against distribution shifts, which indicates the proposed training strategy is more effective in avoiding models that rely on correlations between training domain factors and labels. In this case, domain information is not as helpful as it can be for the case of maximum likelihood estimation with standard classifiers. The proposed approach also supports the inclusion of new classes appearing posterior to training, which we do by simply including new prototypes (or repartitioning the space) obtained from small samples. Doing so yields an efficient and competitive mechanism for few-shot classification.

7.2 Future work

In the following, we discuss research directions that build upon the proposals introduced herein. We specifically discuss ideas that extend the ML-TDNN architecture presented in Chapter 4. In this case, we conjecture that a training scheme where classification is performed at various levels can further boost the discriminability of representations learned with models that perform global pooling across different layers. Additionally, we discuss further extensions of the approach proposed in Chapters 5 and 6, so as to support applications of learned distances to more tasks and define

robust classifiers. In this case, we discuss defining data-dependent kernel functions using such learned distances and imposing properties on induced (pseudo) metric spaces.

Multi-level classification to improve multi-level pooling In order to improve on top of the ML-TDNN architecture introduced in Chapter 4, future work should explore the use of auxiliary output layers placed throughout the model, similarly to the inception architecture [81], so that different embedding encoders can be defined using a single model. These can all be used to produce a combined score at testing time. In order for such strategy to be effective, however, we hypothesize that extra regularization strategies might be required so that predictions across layers are diverse in the sense that errors across layers differ, hence motivating the use of model ensembles. The framework of model distillation, originally introduced in [75], can be utilized to that end. That is, auxiliary output layers defined in parts of the model close to inputs are trained using noisy labels defined by predictions of the final output layer. Lastly, learning how to combine the scores obtained throughout the model in a data-dependent fashion could also be explored, i.e., a model component would be trained to decide which subset of the available scores are more relevant given the observed data, which could lead to further versatility.

Data-dependent kernels for learning versatile predictors In [86], we found learned similarities to perform extremely well in verification settings where questions such as the following need to be answered at inference time: *do two given instances belong to the same class?* - even if examples from novel classes are being tested. We then propose three sets of problems where one can leverage learned similarities when treating them as *kernel functions tailored to the data of interest*: non-parametric 2-sample tests based on MMD scores [183]; outlier/novelty detection, in which case we envision the use of approaches, such as one-class SVM [206] using the learned kernel. An issue that needs to be accounted for in this case is the fact that there is no guarantee that learned similarities will yield Mercer's kernels. We hypothesize that having a kernel which is *approximately* symmetric positive semidefinite suffices in order to efficiently solve the considered tasks, in which case such properties can be enforced during training. If that's not the case, an alternative would be to control the learned kernel's range through the choice of its output layer activation function, as well as inducing symmetry through a symmetrized kernel defined by $\mathcal{K}_S = f(\mathcal{K}(x', x''), \mathcal{K}(x'', x'))$, where $f : \mathbb{R}^2 \to \mathbb{R}$ is symmetric, e.g., $f(\cdot, \cdot) = \max(\cdot, \cdot)$, and $\mathcal{K}$ is the learned similarity function.

Adversarial robustness via low-density learned metric spaces As discussed in Section 6.5.2.1, distance-based inferences were observed to be inherently more robust relative to standard multi-class classification. In addition, distance-based classifiers were also observed to benefit more from adversarial training. This is likely due to the fact that larger norm perturbations are required to move embeddings closer to the prototype of a wrong class than it is to modify an output layer so that the highest logit corresponds to the wrong class. Such observation then indicates that one can induce more robust distance-based classifiers by placing class centroids at a larger distance from one another. Doing so would likely require a larger perturbation in the inputs, rendering attackers detectable.

We thus envision augmenting the training objective described in Algorithm 3 by including a regularization term that assigns a higher importance during training to encoders that result in larger distances between prototypes of different classes. Specifically, we consider defining the similarity matrix S_{ij}, where $S(i,j) = \mathcal{S}(C(i), C(j))$. Some property of S can then be used to define a regularizer, such as its Frobenius or spectral norm. We remark that computing S would be efficient since it only scales with the number of classes for fixed $\mathcal{E}$ and $\mathcal{S}$.

Bibliography

[1] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[2] M. Faundez-Zanuy, "On the vulnerability of biometric security systems," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 6, pp. 3–8, 2004.

[3] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proceeedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2006, pp. 1735–1742.

[6] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92.

[7] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," *Proc. Interspeech 2018*, pp. 1086–1090, 2018.

[8] Z. Tang, D. Wang, and Q. Chen, "Ap18-olr challenge: Three tasks and their baselines," *arXiv preprint arXiv:1806.00616*, 2018.

[9] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K. A. Lee, "Asvspoof 2019: Future horizons in spoofed and fake audio detection," *arXiv preprint arXiv:1904.05441*, 2019.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] J. Neyman and E. S. Pearson, "Ix. on the problem of the most efficient tests of statistical hypotheses," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 231, no. 694-706, pp. 289–337, 1933.

[12] H. Jiang and L. Deng, "A bayesian approach to the verification problem: Applications to speaker verification," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 8, pp. 874–884, 2001.

[13] L. Deng and D. O'Shaughnessy, *Speech processing: a dynamic and optimization-oriented approach*. CRC Press, 2018.

[14] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[15] S. Cumani, N. Brümmer, L. Burget, P. Laface, O. Plchot, and V. Vasilakakis, "Pairwise discriminative speaker verification in the i-vector space," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 6, pp. 1217–1227, 2013.

[16] A. K. Jain, L. Hong, and Y. Kulkarni, "A multimodal biometric system using fingerprint, face and speech," in *Proceedings of 2nd Int'l Conference on Audio-and Video-based Biometric Person Authentication, Washington DC*, 1999, pp. 182–187.

[17] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Transactions on circuits and systems for video technology*, vol. 14, no. 1, pp. 4–20, 2004.

[18] A. Fazel and S. Chakrabartty, "An overview of statistical pattern recognition techniques for speaker verification," *IEEE Circuits and Systems Magazine*, vol. 11, no. 2, pp. 62–81, 2011.

[19] W. Li, T. Fu, H. You, J. Zhu, and N. Chen, "Feature sparsity analysis for i-vector based speaker verification," *Speech Communication*, vol. 80, pp. 60–70, 2016.

[20] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech communication*, vol. 52, no. 1, pp. 12–40, 2010.

[21] N. Dehak, P. A. Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction," in *Twelfth annual conference of the international speech communication association*, 2011.

[22] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *INTERSPEECH*, 2018.

[23] W. Cai, Z. Cai, W. Liu, X. Wang, and M. Li, "Insights into end-to-end learning scheme for language identification," *arXiv preprint arXiv:1804.00381*, 2018.

[24] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," *arXiv preprint arXiv:1804.05160*, 2018.

[25] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 499–515.

[26] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2017, p. 1.

[27] K. J. Lang, A. H. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural networks*, vol. 3, no. 1, pp. 23–43, 1990.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] M. M. Homayounpour and G. Chollet, "Discrimination of voices of twins and siblings for speaker verification," in *Fourth European Conference on Speech Communication and Technology*, 1995.

[30] A. Ariyaeeinia, C. Morrison, A. Malegaonkar, and S. Black, "A test of the effectiveness of speaker verification for differentiating between identical twins," *Science & Justice*, vol. 48, no. 4, pp. 182–186, 2008.

[31] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, and H. Li, "Spoofing and countermeasures for speaker verification: A survey," *speech communication*, vol. 66, pp. 130–153, 2015.

[32] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[33] P. Korshunov, S. Marcel, H. Muckenhirn, A. R. Gonçalves, A. S. Mello, R. V. Violato, F. O. Simoes, M. U. Neto, M. de Assis Angeloni, J. A. Stuchi *et al.*, "Overview of btas 2016 speaker anti-spoofing competition," in *2016 IEEE 8th international conference on biometrics theory, applications and systems (BTAS)*, 2016, pp. 1–6.

[34] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *9th ISCA Speech Synthesis Workshop*, 2016, pp. 125–125.

[35] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[36] A. Tamamori, T. Hayashi, K. Kobayashi, K. Takeda, and T. Toda, "Speaker-dependent wavenet vocoder." in *Eighteenth Annual Conference of the International Speech Communication Association*, 2017.

[37] T. Kaneko, H. Kameoka, N. Hojo, Y. Ijima, K. Hiramatsu, and K. Kashino, "Generative adversarial network-based postfilter for statistical parametric speech synthesis," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4910–4914.

[38] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, "Asvspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[39] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The asvspoof 2017 challenge: Assessing the limits of replay spoofing attack detection," in *Eighteenth Annual Conference of the International Speech Communication Association*, 2017.

[40] H. Muckenhirn, M. Magimai-Doss, and S. Marcel, "End-to-end convolutional neural network-based voice presentation attack detection," in *2017 IEEE international joint conference on biometrics (IJCB)*, 2017, pp. 335–341.

[41] C.-I. Lai, A. Abad, K. Richmond, J. Yamagishi, N. Dehak, and S. King, "Attentive filtering networks for audio replay attack detection," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6316–6320.

[42] J. Monteiro, J. Alam, and T. H. Falk, "End-to-end detection of attacks to automatic speaker recognizers with time-attentive light convolutional neural networks," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2019, pp. 1–6.

[43] P. L. Bartlett, D. P. Helmbold, and P. M. Long, "Gradient descent with identity initialization efficiently learns positive definite linear transformations by deep residual networks," *arXiv preprint arXiv:1802.06093*, 2018.

[44] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[45] H. Li, Z. Xu, G. Taylor, and T. Goldstein, "Visualizing the loss landscape of neural nets," *arXiv preprint arXiv:1712.09913*, 2017.

[46] M. Hardt and T. Ma, "Identity matters in deep learning," *arXiv preprint arXiv:1611.04231*, 2016.

[47] G. Bhattacharya, J. Alam, and P. Kenny, "Deep speaker embeddings for short-duration speaker verification," in *Proc. Interspeech 2017*, 2017, pp. 1517–1521.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[49] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Advances in neural information processing systems*, 2003, pp. 521–528.

[50] A. Globerson and S. T. Roweis, "Metric learning by collapsing classes," in *Advances in neural information processing systems*, 2006, pp. 451–458.

[51] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[52] Y. Ying and P. Li, "Distance metric learning with eigenvalue optimization," *Journal of machine Learning research*, vol. 13, no. Jan, pp. 1–26, 2012.

[53] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng, "Online and batch learning of pseudo-metrics," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 94.

[54] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *Advances in neural information processing systems*, 2004, pp. 41–48.

[55] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 209–216.

[56] B. Kulis, M. Sustik, and I. Dhillon, "Learning low-rank kernel matrices," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 505–512.

[57] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[58] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," *arXiv preprint arXiv:1906.05849*, 2019.

[59] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine learning research*, vol. 5, no. Jan, pp. 27–72, 2004.

[60] S. Pitis, H. Chan, K. Jamali, and J. Ba, "An inductive bias for distances: Neural nets that respect the triangle inequality," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HJeiDpVFPr

[61] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 539–546.

[62] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[63] H. Shi, Y. Yang, X. Zhu, S. Liao, Z. Lei, W. Zheng, and S. Z. Li, "Embedding deep metric for person re-identification: A study against large variations," in *European conference on computer vision*. Springer, 2016, pp. 732–748.

[64] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[65] C. Zhang, K. Koishida, and J. H. Hansen, "Text-independent speaker verification based on triplet convolutional neural network embeddings," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 9, pp. 1633–1644, 2018.

[66] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, "Hamming distance metric learning," in *Advances in neural information processing systems*, 2012, pp. 1061–1069.

[67] N. Courty, R. Flamary, and M. Ducoffe, "Learning wasserstein embeddings," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SJyEH91A-

[68] M. Nickel and D. Kiela, "Learning continuous hierarchies in the Lorentz model of hyperbolic geometry," in *International Conference on Machine Learning*, 2018, pp. 3776–3785.

[69] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[70] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[71] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[72] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[73] X. Wu, R. He, Z. Sun, and T. Tan, "A light cnn for deep face representation with noisy labels," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2884–2896, 2018.

[74] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[75] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[76] D. A. Reynolds, "An overview of automatic speaker recognition technology," in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4. IEEE, 2002, pp. IV–4072.

[77] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[78] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2840–2848.

[79] P. Kenny, T. Stafylakis, P. Ouellet, M. J. Alam, and P. Dumouchel, "PLDA for speaker verification with utterances of arbitrary duration," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 7649–7653.

[80] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Advances in neural information processing systems*, 2017, pp. 6338–6347.

[81] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[82] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," *Proc. Interspeech 2017*, pp. 2616–2620, 2017.

[83] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in neural information processing systems*, 2017, pp. 4077–4087.

[84] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

[85] N. Garcia and G. Vogiatzis, "Learning non-metric visual similarity for image retrieval," *Image and Vision Computing*, vol. 82, pp. 18–25, 2019.

[86] J. Monteiro, I. Albuquerque, J. Alam, R. D. Hjelm, and T. Falk, "An end-to-end approach for the verification problem: learning the right distance," in *International Conference on Machine Learning*, 2020.

[87] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 42–54, 2000.

[88] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[89] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[90] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Deeper, broader and artier domain generalization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5542–5550.

[91] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *Proceedings of the IEEE international conference on computer vision workshops*, 2013, pp. 554–561.

[92] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[93] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *Advances in neural information processing systems*, 2016, pp. 3630–3638.

[94] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[95] I. Albuquerque, J. Monteiro, M. Darvishi, T. H. Falk, and I. Mitliagkas, "Generalizing to unseen domains via distribution matching," *arXiv preprint arXiv:1911.00804*, 2019.

[96] J. Monteiro, X. Gibert, J. Feng, V. Dumoulin, and D.-S. Lee, "Domain conditional predictors for domain adaptation," *Pre-registration workshop, NeurIPS 2020*, 2020.

[97] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[98] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[99] Z. Wu, T. Kinnunen, E. S. Chng, H. Li, and E. Ambikairajah, "A study on spoofing attack in state-of-the-art speaker verification: the telephone speech case," in *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*. IEEE, 2012, pp. 1–5.

[100] M. Sahidullah, T. Kinnunen, and C. Hanilçi, "A comparison of features for synthetic speech detection," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[101] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[102] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[103] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," *arXiv preprint arXiv:1712.00409*, 2017.

[104] Y. Bahri, E. Dyer, J. Kaplan, J. Lee, and U. Sharma, "Explaining neural scaling laws," *arXiv preprint arXiv:2102.06701*, 2021.

[105] S. Ioffe, "Probabilistic linear discriminant analysis," in *European Conference on Computer Vision*. Springer, 2006, pp. 531–542.

[106] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *IEEE 11th International Conference on Computer Vision (ICCV)*, 2007, pp. 1–8.

[107] L. Li, Y. Chen, Y. Shi, Z. Tang, and D. Wang, "Deep Speaker Feature Learning for Text-independent Speaker Verification," *ArXiv e-prints*, May 2017.

[108] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[109] S. Yadav and A. Rai, "Learning discriminative features for speaker identification and verification," *Proc. Interspeech 2018*, pp. 2237–2241, 2018.

[110] N. Le and J.-M. Odobez, "Robust and discriminative speaker embedding via intra-class distance variance regularization," *Proc. Interspeech 2018*, pp. 2257–2261, 2018.

[111] N. Li, D. Tuo, D. Su, Z. Li, D. Yu, and A. Tencent, "Deep discriminative embeddings for duration robust speaker verification," *Proc. Interspeech 2018*, pp. 2262–2266, 2018.

[112] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep Speaker: an End-to-End Neural Speaker Embedding System," *ArXiv e-prints*, May 2017.

[113] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[114] M. Todisco, H. Delgado, and N. Evans, "A new feature for automatic speaker verification anti-spoofing: Constant q cepstral coefficients," in *Speaker Odyssey Workshop, Bilbao, Spain*, vol. 25, 2016, pp. 249–252.

[115] J. Alam and P. Kenny, "Spoofing detection employing infinite impulse response—constant q transform-based feature representations," in *2017 25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 101–105.

[116] M. J. Alam, P. Kenny, V. Gupta, and T. Stafylakis, "Spoofing detection on the asvspoof2015 challenge corpus employing deep neural networks," in *Proc. Odyssey*, 2016, pp. 270–276.

[117] M. J. Alam, P. Kenny, G. Bhattacharya, and T. Stafylakis, "Development of crim system for the automatic speaker verification spoofing and countermeasures challenge 2015," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[118] T. B. Patel and H. A. Patil, "Combining evidences from mel cepstral, cochlear filter cepstral and instantaneous frequency features for detection of natural vs. spoofed speech," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[119] ——, "Effectiveness of fundamental frequency (f 0) and strength of excitation (soe) for spoofed speech detection," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5105–5109.

[120] G. Suthokumar, K. Sriskandaraja, V. Sethu, C. Wijenayake, and E. Ambikairajah, "An investigation about the scalability of the spoofing detection system," in *2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*, 2018, pp. 1–5.

[121] N. Chen, Y. Qian, H. Dinkel, B. Chen, and K. Yu, "Robust deep feature for spoofing detection—the sjtu system for asvspoof 2015 challenge," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[122] X. Xiao, X. Tian, S. Du, H. Xu, E. S. Chng, and H. Li, "Spoofing speech detection using high dimensional magnitude and phase features: The ntu approach for asvspoof 2015 challenge," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[123] X. Tian, Z. Wu, X. Xiao, E. S. Chng, and H. Li, "Spoofing detection from a feature representation perspective," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2119–2123.

[124] H. Zeinali, L. Burget, J. Rohdin, T. Stafylakis, and J. H. Cernocky, "How to improve your speaker embeddings extractor in generic toolkits," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6141–6145.

[125] L. You, W. Guo, L.-R. Dai, and J. Du, "Deep neural network embeddings with gating mechanisms for text-independent speaker verification," *Proc. Interspeech 2019*, pp. 1168–1172, 2019.

[126] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," *Proc. Interspeech 2018*, pp. 2252–2256, 2018.

[127] S. Wang, Y. Yang, Y. Qian, and K. Yu, "Revisiting the statistics pooling layer in deep speaker embedding learning," in *2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, 2021, pp. 1–5.

[128] Y. Tang, G. Ding, J. Huang, X. He, and B. Zhou, "Deep speaker embedding learning with multi-level pooling for text-independent speaker verification," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6116–6120.

[129] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks." in *BMVC*, vol. 1, no. 2, 2016, p. 3.

[130] S. O. Sadjadi, T. Kheyrkhah, A. Tong, C. S. Greenberg, D. A. Reynolds, E. Singer, L. P. Mason, and J. Hernandez-Cordero, "The 2016 nist speaker recognition evaluation." in *Interspeech*, 2017, pp. 1353–1357.

[131] N. Brummer, A. Swart, J. Jorrın-Prieto, P. Garcıa, L. Buera, P. Matejka, O. Plchot, M. Diez, A. Silnova, X. Jiang *et al.*, "Abc nist sre 2016 system description," in *Proc. of the NIST SRE 2016 workshop*, 2016.

[132] J. Alam, N. Brummer, L. Burget, M. Diez, O. Glembek, P. Kenny, M. Klco, F. Landini, A. Lozano-Diez, P. Matejka *et al.*, "Abc nist sre 2018 system description," in *Proc. of the NIST SRE 2018 workshop*, 2018.

[133] J. Monteiro, J. Alam, and T. H. Falk, "Residual convolutional neural network with attentive feature pooling for end-to-end language identification from short-duration speech," *Computer Speech & Language*, vol. 58, pp. 364–376, 2019.

[134] ——, "Combining speaker recognition and metric learning for speaker-dependent representation learning," *Proc. Interspeech 2019*, pp. 4015–4019, 2019.

[135] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[136] D. Snyder, G. Chen, and D. Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.

[137] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[138] Z. Tang, D. Wang, Y. Chen, and Q. Chen, "Ap17-olr challenge: Data, plan, and baseline," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2017*. IEEE, 2017, pp. 749–753.

[139] M. J. Alam, P. Kenny, and V. Gupta, "Tandem features for text-dependent speaker verification on the reddots corpus." in *INTERSPEECH*, 2016, pp. 420–424.

[140] Z. Z. Dong Wang, Xuewei Zhang, "Thchs-30 : A free chinese speech corpus," 2015. [Online]. Available: http://arxiv.org/abs/1512.01882

[141] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks." *ICML (3)*, vol. 28, pp. 1310–1318, 2013.

[142] A. Sablayrolles, M. Douze, C. Schmid, and H. Jégou, "Spreading vectors for similarity search," *arXiv preprint arXiv:1806.03198*, 2018.

[143] J. Beirlant, E. J. Dudewicz, L. Györfi, and E. C. Van der Meulen, "Nonparametric entropy estimation: An overview," *International Journal of Mathematical and Statistical Sciences*, vol. 6, no. 1, pp. 17–39, 1997.

[144] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[145] D. Garcia-Romero, A. McCree, S. Shum, N. Brummer, and C. Vaquero, "Unsupervised domain adaptation for i-vector speaker recognition," in *Proceedings of Odyssey: The Speaker and Language Recognition Workshop*, 2014.

[146] J. Alam, G. Bhattacharya, and P. Kenny, "Speaker verification in mismatched conditions with frustratingly easy domain adaptation," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 176–180.

[147] A. F. Martin and C. S. Greenberg, "The nist 2010 speaker recognition evaluation," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[148] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification." in *Interspeech*, 2017, pp. 999–1003.

[149] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *arXiv preprint arXiv:1703.03130*, 2017.

[150] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification," in *Proc. Interspeech*, 2018, pp. 3573–3577.

[151] J. Monteiro, J. Alam, and T. H. Falk, "A multi-condition training strategy for countermeasures against spoofing attacks to speaker recognizers," in *Proceedings of the Odyssey Speaker and Language Recognition Workshop, Tokyo, Japan*, 2020, pp. 1–5.

[152] ——, "Generalized end-to-end detection of spoofing attacks to automatic speaker recognizers," *Computer Speech & Language*, vol. 63, p. 101096, 2020.

[153] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *ArXiv e-prints*, Dec. 2014.

[154] N. Carlini and D. Wagner, "Audio Adversarial Examples: Targeted Attacks on Speech-to-Text," *ArXiv e-prints*, Jan. 2018.

[155] F. Kreuk, Y. Adi, M. Cisse, and J. Keshet, "Fooling End-to-end Speaker Verification by Adversarial Examples," *ArXiv e-prints*, Jan. 2018.

[156] D. Zhu and K. K. Paliwal, "Product of power spectrum and group delay function for speech recognition," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1. IEEE, 2004, pp. I–125.

[157] M. J. Alam, G. Bhattacharya, and P. Kenny, "Boosting the performance of spoofing detection systems on replay attacks using q-logarithm domain feature normalization," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 393–398.

[158] M. Todisco, H. Delgado, and N. Evans, "Constant q cepstral coefficients: A spoofing countermeasure for automatic speaker verification," *Computer Speech & Language*, vol. 45, pp. 516–535, 2017.

[159] T. Kinnunen et al., "t-dcf: a detection cost function for the tandem assessment of spoofing countermeasures and automatic speaker verification," *arXiv preprint arXiv:1804.09618*, 2018.

[160] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[161] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.

[162] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[163] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[164] Z. Wu, E. S. Chng, and H. Li, "Detecting converted speech and natural speech for anti-spoofing attack in speaker recognition," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[165] J. Monteiro, J. Alam, and T. H. Falk, "Multi-level self-attentive tdnn: A general and efficient approach to summarize speech into discriminative utterance-level representations," *Speech Communication (Under review)*, vol. 1, pp. 1–1, 2021.

[166] J. Monteiro, M. J. Alam, and T. Falk, "On the performance of time-pooling strategies for end-to-end spoken language identification," in *Proceedings of the 12th Language Resources and Evaluation Conference*, 2020, pp. 3566–3572.

[167] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5791–5795.

[168] A. Hajavi and A. Etemad, "A deep neural network for short-segment speaker recognition," *Proc. Interspeech 2019*, pp. 2878–2882, 2019.

[169] X. Xiang, S. Wang, H. Huang, Y. Qian, and K. Yu, "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition," *arXiv preprint arXiv:1906.07317*, 2019.

[170] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a siamese time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737–744.

[171] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," *arXiv preprint arXiv:1808.06670*, 2018.

[172] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 297–304.

[173] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[174] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur, "Spoken language recognition using x-vectors." in *Odyssey*, 2018, pp. 105–111.

[175] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 74–81. [Online]. Available: http://dx.doi.org/10.21437/Odyssey.2018-11

[176] M. Hajibabaei and D. Dai, "Unified hypersphere embedding for speaker recognition," *arXiv preprint arXiv:1807.08312*, 2018.

[177] M. Ravanelli and Y. Bengio, "Learning speaker representations with mutual information," *Proc. Interspeech 2019*, pp. 1153–1157, 2019.

[178] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5220–5224.

[179] J. Monteiro, I. Albuquerque, J. Alam, and T. H. Falk, "Temple: defining versatile template learners via prototypical classifiers with learned similarities," *Journal of machine Learning Research (Under review)*, vol. 1, pp. 1–1, 2021.

[180] K. Cilingir, R. Manzelli, and B. Kulis, "Deep divergence learning," in *International Conference on Machine Learning*, 2020.

[181] L. Wenliang, D. Sutherland, H. Strathmann, and A. Gretton, "Learning deep kernels for exponential family densities," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 6737–6746. [Online]. Available: http://proceedings.mlr.press/v97/wenliang19a.html

[182] F. Liu, W. Xu, J. Lu, G. Zhang, A. Gretton, and D. J. Sutherland, "Learning deep kernels for non-parametric two-sample tests," in *International Conference on Machine Learning*, 2020.

[183] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[184] L. Zimmer, M. Lindauer, and F. Hutter, "Auto-pytorch tabular: Multi-fidelity metalearning for efficient and robust autodl," *arXiv preprint arXiv:2006.13799*, 2020.

[185] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver: General perception with iterative attention," *arXiv preprint arXiv:2103.03206*, 2021.

[186] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer *et al.*, "Perceiver io: A general architecture for structured inputs & outputs," *arXiv preprint arXiv:2107.14795*, 2021.

[187] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[188] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *arXiv preprint arXiv:1803.06373*, 2018.

[189] C. Mao, Z. Zhong, J. Yang, C. Vondrick, and B. Ray, "Metric learning for adversarial robustness," in *Advances in Neural Information Processing Systems*, 2019, pp. 480–491.

[190] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *International Conference on Machine Learning*, 2019, pp. 7472–7482.

[191] Q. Dou, D. C. de Castro, K. Kamnitsas, and B. Glocker, "Domain generalization via model-agnostic learning of semantic features," in *Advances in Neural Information Processing Systems*, 2019, pp. 6450–6461.

[192] P. Chattopadhyay, Y. Balaji, and J. Hoffman, "Learning to balance specificity and invariance for in and out of domain generalization," *arXiv preprint arXiv:2008.12839*, 2020.

[193] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," *arXiv preprint arXiv:2007.01434*, 2020.

[194] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4004–4012.

[195] A. Sanakoyeu, V. Tschernezki, U. Buchler, and B. Ommer, "Divide and conquer the embedding space for metric learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 471–480.

[196] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy, "Learnable structured clustering framework for deep metric learning," *arXiv preprint arXiv:1612.01213*, vol. 1, no. 2, p. 8, 2016.

[197] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in neural information processing systems*, 2016, pp. 1857–1865.

[198] Y. Yuan, K. Yang, and C. Zhang, "Hard-aware deeply cascaded embedding," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 814–823.

[199] K. Roth, B. Brattoli, and B. Ommer, "Mic: Mining interclass characteristics for improved metric learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8000–8009.

[200] E. Ustinova and V. Lempitsky, "Learning deep embeddings with histogram loss," in *Advances in Neural Information Processing Systems*, 2016, pp. 4170–4178.

[201] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," 2016.

[202] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *International Conference on Learning Representations*, 2018.

[203] T. Munkhdalai, X. Yuan, S. Mehri, and A. Trischler, "Rapid adaptation with conditionally shifted neurons," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3664–3673.

[204] B. Oreshkin, P. R. López, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 721–731.

[205] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665.

[206] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, 2000, pp. 582–588.

[207] C. M. Bishop, "Pattern recognition," *Machine learning*, vol. 128, no. 9, 2006.

[208] E. Conrad, S. Misenar, and J. Feldman, "Chapter 2 - domain 1: Access control," in *CISSP Study Guide (Second Edition)*, second edition ed., E. Conrad, S. Misenar, and J. Feldman, Eds. Boston: Syngress, 2012, pp. 9–62. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781597499613000029

[209] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.

[210] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1, pp. 151–175, 2010.

[211] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Appendix A

Background content

In what follows, we list, for each chapter, references to topics that are assumed to be known by the reader. References to both content reported in our background section in Chapter 1 as well as external references are included.

A.1 Chapter 2

1. Gaussian mixture models: [207]
2. Universal background models: [14]
3. i-vector: [1, 21]
4. x-vector: [3]
5. ResNets: [4], Sec. 1.2.2.2
6. Metric Learning: [49], [61], Sec. 1.2.3
7. Equal error rate: [208]
8. Linear discriminant analysis: [207]
9. Probabilistic linear discriminant analysis: [105, 106]

A.2 Chapter 3

1. Product spectrum: [156]
2. Gaussian mixture models: [207]
3. LCNNs: [73]
4. Equal error rate: [208]
5. Detection cost function: [9]

A.3 Chapter 4

1. Self-attention: [48], `https://jalammar.github.io/illustrated-transformer/`
2. TDNN: [3]
3. Equal error rate: [208]

A.4 Chapter 5

1. Metric Learning: [49], [61], Sec. 1.2.3
2. (Noise) Contrastive estimation: [172]
3. Generative adversarial networks: [173]
4. Reproducing kernel Hilbert spaces: [209]

A.5 Chapter 6

1. Prototypical classifiers: [83]
2. Retrieval: [78]
3. Adversarial perturbations: [98], `https://github.com/bethgelab/foolbox`
4. Out-of-distribution generalization: [210], [211]