Enhancing U-Net Performance for High-Resolution Land Cover Classification Using a Dynamic Epoch-Centric Optimizer (DECO)

Mahdi Farhangi, Asghar Milan, Danesh Shokri, Saeid Homayouni

PII: S2352-9385(25)00221-6

DOI: https://doi.org/10.1016/j.rsase.2025.101668

Reference: RSASE 101668

To appear in: *Remote Sensing Applications: Society and Environment*

Received Date: 3 January 2025

Revised Date: 29 May 2025

Accepted Date: 20 July 2025

Enhancing U-Net Performance for High-Resolution Land Cover Classification Using a Dynamic Epoch-Centric Optimizer (DECO)

Mahdi Farhangi[a], Asghar Milan[a,*], Danesh Shokri[b], Saeid Homayouni[c]

[a] Faculty of Civil, Water, and Environmental Engineering, Shahid Beheshti University (SBU), Tehran, Iran.
[b] Centre Eau Terre Environnement, Institut National de la Recherche Scientifique, Québec, Canada.
[c] Département des Sciences Géomatiques, Université Laval, Québec, Canada.

Abstract
In recent years, deep learning models—particularly U-Net—have garnered significant attention for applications such as high-resolution land cover mapping. A key challenge in improving these models' performance lies in the proper selection and tuning of optimizers: each algorithm (e.g., Adam, Nadam) offers distinct strengths and weaknesses, and reliance on a single optimizer may not yield optimal results across all training stages. Here, we introduce DECO, a novel hybrid optimizer that dynamically switches among multiple optimizers across epochs to enhance overall convergence and stability. U-Net trained with DECO on aerial imagery of buildings, forests, roads, and water in the Minski region of Warsaw, Poland, achieved 96.13 % overall accuracy, a Kappa coefficient of 91.49 %, an F1 score of 96.08 %, and a Jaccard index of 64.53 %. To assess generalizability, the model was further evaluated on a test region in the Malopolskie province, yielding 86.74 % accuracy, 73.75 % Kappa, 87.29 % F1, and 55.02 % Jaccard. Moreover, to demonstrate DECO's broader applicability, we implemented it on the DeepLab v3+ architecture, observing likewise improvements in validation accuracy and training stability. These findings substantiate that dynamic, epoch-centric optimizer switching can substantially boost the precision and robustness of deep learning models for high-resolution land cover classification.

Keywords: Dynamic Epoch-Centric Optimizer (DECO), Deep Learning Optimization, Optimizer Switching, U-Net, DeepLab v3+, High-Resolution Aerial Imagery, Convolutional Neural Networks

[*] Corresponding Author: Corresponding author: Assistant Professor, Faculty of Civil, Water, and Environmental Engineering, Shahid Beheshti University (SBU), Tehran, Iran.
E-mail addresses: ma.farhangi@mail.sbu.ac.ir (M. Farhangi), a_milan@sbu.ac.ir (A. Milan), danesh.shokri.1@ulaval.ca (D. Shokri), saeid.homayouni@inrs.ca (S. Homayouni).

1. Introduction

Land cover classification using satellite imagery is a fundamental component of remote sensing science, playing a critical role in natural resource management, environmental monitoring, urban planning, and disaster management (Khan and Jung, 2024). Advances in technology and improvements in the quality of high-resolution satellite images have made it possible to extract valuable land-use information, including agricultural fields, forests, urban areas, and water bodies, to support decision-making and analytical tasks (Elhani *et al.*, 2023). However, a major challenge in utilizing such data lies in accurately classifying diverse land cover types, which remains a complex and active area of research due to the high variability and intricacies of satellite imagery data (Talukdar *et al.*, 2020).

Historically, various methods have been applied for land cover classification, primarily limited to threshold-based methods, statistical classifiers, and dimensionality reduction techniques (Lakshminarayana and Rao, 2010). Threshold-based methods, which rely on predefined spectral values to assign pixels to specific classes, have shown good results for simple datasets. However, their accuracy significantly decreases as the data becomes more complex. Similarly, statistical approaches such as Maximum Likelihood Classification (MLC) perform well when dealing with normally distributed data but struggle with complex and non-Gaussian datasets (Strahler, 1980; El fallah, El kharrim and Belghyti, 2024). Furthermore, dimensionality reduction methods like Principal Component Analysis (PCA) have been employed to simplify data complexity. Nonetheless, these methods often compromise accuracy by discarding essential information during the dimensionality reduction.

With advancements in machine learning, methods such as Support Vector Machines (SVM) and Random Forest emerged as effective techniques for data classification (Zhang *et al.*, 2019). These methods demonstrated superior performance over traditional approaches, particularly in identifying complex patterns and handling high-dimensional data (Digra, Dhir and Sharma, 2022). By leveraging nonlinear decision boundaries, SVM is well-known for its accuracy in classifying intricate datasets (Talukdar *et al.*, 2020a). On the other hand, as an ensemble learning method combining multiple decision trees, Random Forest achieved high accuracy in land cover classification tasks (Xu *et al.*, 2020).

With the advent of deep learning, models like U-Net, based on Convolutional Neural Networks (CNNs), were specifically designed for satellite image analysis (Shaar *et al.*, 2024). With its encoder-decoder structure, the U-Net model has achieved remarkable success in various applications (Khan and Singh, 2023). This architecture excels in learning intricate and large-scale features from massive datasets, providing significantly higher image classification accuracy than earlier methods (Ronneberger, Fischer and Brox, 2015). Consequently, U-Net-based models are widely adopted, particularly for processing high-resolution aerial imagery (Tsiporenko, Chizhov and Fishman, 2024).

Additionally, Vision Transformers (ViTs) have recently emerged in the deep learning domain and have been explored in specific applications. ViTs have occasionally delivered comparable or even superior results to CNN-based models. However, due to their reliance on large-scale training datasets and computational complexity, ViTs are still less commonly employed compared to U-Net (Yao and Shao, 2024; Bazi *et al.*, 2021). Therefore, models like U-Net remain the preferred choice for specialized tasks such as high-resolution satellite image classification due to their efficiency, lower data requirements, and robust performance (Rubab *et al.*, 2024).

Optimizers play a critical role in enhancing the performance of deep learning models. By adjusting and updating weights and biases based on gradients derived from the loss function, optimizers help minimize errors and improve model accuracy (Hassan *et al.*, 2023). Moreover, optimizers manage the learning rate dynamically using specialized techniques, ensuring a more stable and efficient learning process (Bera and Shrivastava, 2020). Consequently, selecting and fine-tuning an appropriate optimizer has become one of the central challenges in deep learning, particularly for high-resolution aerial image analysis, where model accuracy and efficiency are paramount.

Numerous studies have explored the strengths and weaknesses of various optimizers in deep learning. Chigozie Enyinna Nwankpa provides an in-depth evaluation of common optimizers, highlighting their characteristics and performance (Nwankpa, 2020). The Stochastic Gradient Descent (SGD) optimizer, known for its simplicity and efficiency on large-scale datasets, often suffers from slow convergence rates and a tendency to get stuck in local minima when dealing with complex tasks. Adam, which combines the advantages of RMSprop and AdaGrad, offers faster convergence and adaptive learning rates but is highly sensitive to hyperparameter settings, posing additional challenges. Adadelta is suitable for noisy gradients by utilizing adaptive learning rates per parameter; however, its convergence speed is slower than Adam's. Nadam, which incorporates Nesterov momentum into Adam, achieves faster convergence but requires careful hyperparameter tuning for optimal results. RMSprop is highly effective in optimizing non-convex problems and recurrent neural networks (RNNs), though its performance can be hindered by sensitivity to parameter settings and occasional stability issues (Nwankpa, 2020). Finally, AdamW improves upon Adam by better managing weight decay, reducing overfitting, and offering more stable performance in complex networks (Pagliardini, Ablin and Grangier, 2024).

Other studies have also focused on tuning the parameters and hyperparameters of the U-Net deep learning model to enhance its performance. In a study conducted by Won-Kyung Baek et al. (2024), challenges related to using multispectral satellite imagery in land cover classification were investigated. This research demonstrated that combining RGB and NIR images did not significantly improve classification performance due to the inefficient use of spectral information in deep learning methods. To address this issue, they introduced an improved U-Net version, SiU-Net, which separates RGB and NIR inputs for more effective processing. The results showed that SiU-Net, optimized using the Adam optimizer, achieved superior performance, even when trained on small and imbalanced datasets, highlighting its strong applicability in real-world and diverse conditions (Baek, Lee and Jung, 2024). Similarly, Li and colleagues (2023) proposed the RD-UNet architecture for cloud detection in remote sensing images, enhancing the performance of U-Net-based models in extracting fine edges and detailed features (Li, Li and Ma, 2024).

In another study by Esraa Hassan (2023), the performance of various optimizers on machine learning models was evaluated. This research analyzed and compared optimizers such as SGD, Adam, and RMSprop and their impact on the accuracy of computer vision models, specifically in medical applications like CT image analysis. The results revealed that the Adam optimizer significantly improved model accuracy, achieving the highest performance among the tested optimizers (Hassan *et al.*, 2023).

Furthermore, Eren Can Seyrek et al. (2024) examined the effects of activation functions and different optimizers in convolutional neural networks (CNNs) for hyperspectral image classification. Six activation functions (LReLU, Mish, PReLU, ReLU, Sigmoid, and Swish) and four optimizers (Adam, Adamax, Nadam, and RMSprop) were tested on a CNN model. The study found that the Adamax optimizer, in combination with the Mish activation function, provided the best overall accuracy for the Indian Pines and WHU-Hi HongHu datasets. This research underscored the importance of selecting appropriate optimizers and activation functions in improving CNN performance, demonstrating that different combinations of these components can significantly influence model accuracy and efficiency (Can and Murat, 2024). In addition, a study by Buttar et al. (2024) investigated techniques for distinguishing various crops from satellite imagery. The study employed the U-Net architecture and the Adam optimizer to process images under challenging conditions, such as small-scale farms and cloud cover. The findings demonstrated that the U-Net model outperformed traditional machine learning approaches, highlighting the significance of using appropriate optimizers and advanced network architectures to enhance model accuracy and performance in challenging scenarios (Buttar, 2024).

Another study by Andrew Clark et al. (2023) examined the effects of varying convolutional neural network (CNN) parameters, such as the number of filters, kernel size, and optimizer type, on training time and classification accuracy. This research utilized the U-Net architecture and LULC training data alongside multispectral aerial imagery from northern Queensland, Australia. Several optimizers were tested, including Adadelta, Adagrad, Adam, Adamax, Ftrl, Nadam, RMSprop, and SGD. The results revealed that the RMSprop optimizer achieved the best balance between training time and accuracy (Clark, Phinn and Scarth, 2023).

Overall, comparing different optimizers indicates that selecting a single fixed optimizer may not be optimal for all models and tasks. Each optimizer has specific strengths and weaknesses that must be considered based on the nature of the data, model complexity, and evaluation metrics. The primary challenge lies in identifying a flexible approach that leverages the advantages of various optimizers during different training stages to maximize model accuracy and stability.

The study employs multiple optimizers, including SGD, Adam, Adadelta, AdamW, Nadam, and RMSprop, to optimize the performance of the U-Net deep learning model. The primary motivation for selecting U-Net lies in its Encoder-Decoder architecture, which enables precise boundary delineation of land cover classes while preserving spatial details. Compared to newer models such as Vision Transformers, U-Net remains a more suitable option for this research due to its structural simplicity and efficiency when working with high-resolution data (Li and Zhang, 2024).

The primary objective of this study is to develop and evaluate an innovative method to enhance the performance of the U-Net model for land cover classification using high-resolution satellite imagery. A novel dynamic and hybrid optimizer named DECO (Dynamic Epoch-Centric

Optimizer) has been introduced and implemented to address the optimization challenges in deep learning models and improve accuracy and stability.

The key contributions of this study are as follows:

- **Design and Implementation of a Novel Hybrid Optimizer (DECO)**
  The DECO (Dynamic Epoch-Centric Optimizer) is an innovative optimization approach for deep learning models that dynamically and incrementally adjust optimizers during training. By leveraging the strengths of various optimizers and compensating for their weaknesses, DECO enhances model performance throughout training. This method effectively manages the learning rate, minimizing accuracy fluctuations during the initial training phases while accelerating model convergence in later stages.
- **Comprehensive Evaluation Metrics and Statistical Analyses**
  A comprehensive set of standard evaluation metrics and statistical analyses was employed to evaluate the performance of the U-Net deep learning model rigorously. Metrics such as Overall Accuracy were used to measure the 'model's overall precision, while the Kappa Coefficient assessed the agreement between model predictions and ground truth, accounting for random errors. Additionally, the F1 Score balanced Precision and Recall across classes, and the Jaccard Score focused on accurately delineating class boundaries. The Confusion Matrix was further utilized to provide detailed insights into the 'model's classification performance across different land-cover classes.
  In addition to the evaluation metrics, advanced statistical analyses were conducted to compare the performance of different optimizers. A one-way ANOVA test was employed to determine the presence of statistically significant differences among optimizer results. Following this, the Tukey HSD post-hoc test was applied to identify specific pairs of optimizers with significant performance variations. These statistical methods were selected for their robustness in providing a thorough and scientific comparison of optimizer performance. The results of these analyses were crucial in guiding the development of DECO and optimizing its implementation at various stages of U-Net training.
- **Evaluation of Model Generalizability in New Geographical Regions**
  Its performance was evaluated in a new region with distinct geographical characteristics to assess the robustness and adaptability of the developed U-Net model enhanced by the DECO optimizer. This evaluation aimed to determine whether the model could maintain its classification accuracy and effectively identify land cover types under different and heterogeneous conditions, thereby validating its generalization capability.

These innovations collectively target improving deep learning models' accuracy, stability, and convergence speed in analyzing complex and challenging remote sensing datasets. By leveraging the DECO method, this research represents a significant step forward in enhancing the performance of deep learning models for high-resolution satellite imagery classification.

The remainder of this manuscript is organized as follows.

Section 2 Materials and Methods describes the study areas, datasets, and preprocessing steps. Section 3 DECO Optimizer and Network Architectures presents the design of our Dynamic Epoch-

Centric Optimizer (DECO), details the U-Net implementation, and briefly outlines the auxiliary DeepLab v3+ setup used to confirm DECO's general applicability.

Section 4 Implementation and Evaluation covers training procedures, hyperparameter settings, and a full suite of evaluation metrics (overall accuracy, Kappa, F1, Jaccard, confusion matrices) for both U-Net and DeepLab v3+.

The remainder of this manuscript is structured according to the actual section numbering as follows:
Section 2: Materials and Methods
Describes the study areas, datasets, and preprocessing steps used for the U-Net experiments.
Section 3: Methodology
Presents the design of the DECO optimizer and the U-Net architecture; it also briefly notes a secondary validation on DeepLab v3+ without detailing its internal structure.
Section 4: Implementation and Results
Covers implementation details, training procedures, and hyperparameter settings, and provides comprehensive evaluation metrics (Overall Accuracy, Kappa, F1, Jaccard, and confusion matrices) for both U-Net and DeepLab v3+.
Section 5: Discussion
Reports comparative performance in the Minski and Malopolskie regions and presents statistical analyses (one-way ANOVA and Tukey HSD) that confirm the effects of the optimizers.
Section 6: Conclusion and Future Work
Summarizes the key findings and outlines directions for future research.

2. Materials and Methods

2.1. Study Area

For the implementation and training of the U-Net deep learning model in this study, aerial imagery from an area called Minski, located in the eastern part of Warsaw, the capital of Poland, was utilized. This area covers approximately 532.245 hectares and is situated between longitudes 18°'39'21" to 20°41'21" East and latitudes 52°05'09" to 52°08'11" North. The selection of this area is due to the diversity of land cover and the ability to test the model in various geographical environments. Figure 1 illustrates the location and coverage of this area.

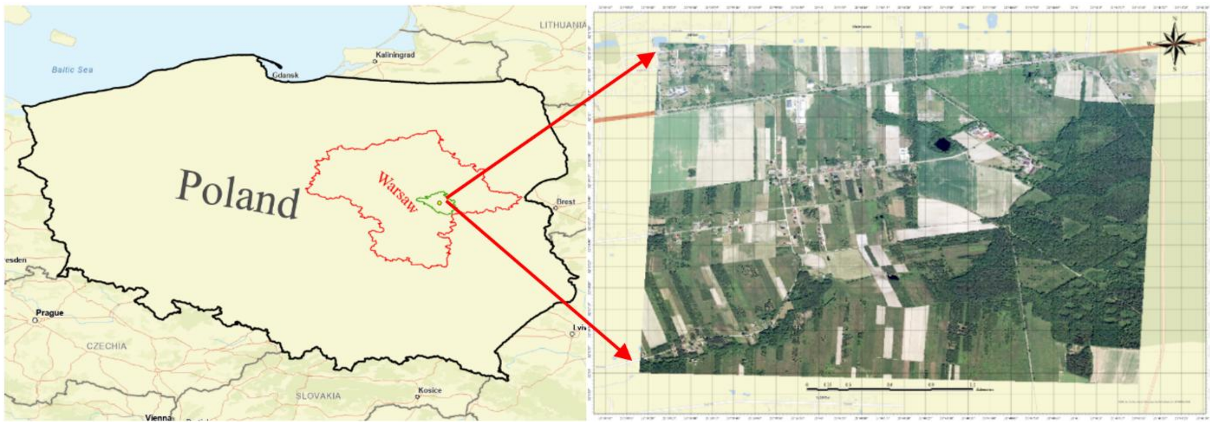

Figure 1: Study Area: Orthophoto of the Minski Area, Warsaw, Poland

Furthermore, to evaluate the generalization capability of the U-Net deep learning model based on DECO and other optimizers, a new geographical region in the Malopolskie Province, located in southern Poland, was used. This region, situated at the intersection of three cities—Krakowski, Wielicki, and Bochenski—is bounded by the geographical coordinates 20˚20’35’’E to 20˚22’33’’E longitude and 50˚05’00’’N to 50˚06’17’’N latitude. It covers an area of 537.113 hectares and includes four land-use types: forest cover, buildings, roads, and water bodies.

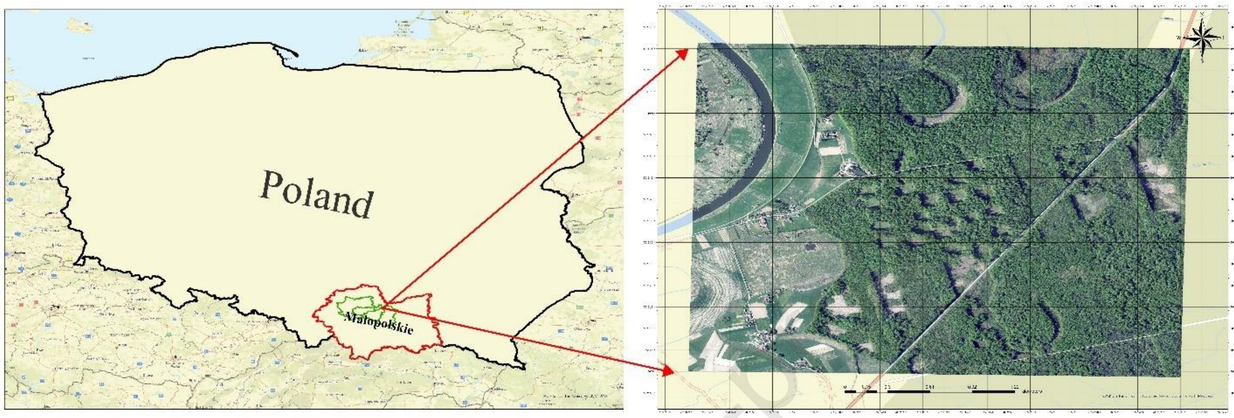

Figure 2: New study area: Orthophoto image of Malopolskie, Poland.

In this study, aerial images from 2021 were utilized (Boguszewski *et al.*, 2021). These high-resolution images, with a spatial resolution of 25 centimeters, were captured in three primary bands of the visible spectrum under suitable lighting conditions.[*]

3. Methodology

In this research, a dynamic optimizer based on training epochs was designed and developed using aerial images and a convolutional deep learning model based on the U-Net architecture. This method improves the accuracy and precision of image classification and addresses land-use analysis across four categories. The process followed in this paper for classifying land use with the U-Net deep learning model is illustrated in Figure 3.

[*] The dataset download address: https://landcover.ai.linuxpolska.com/

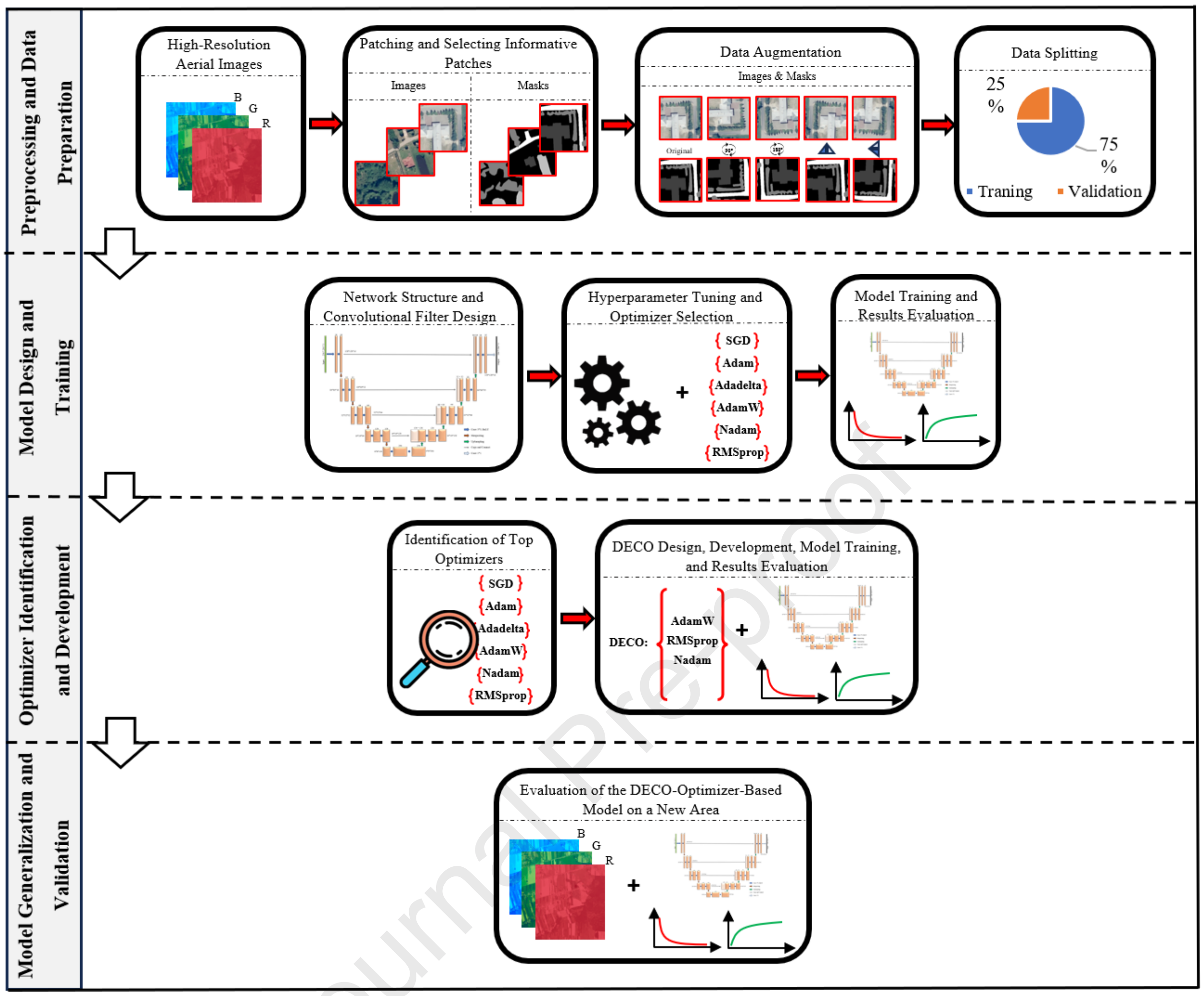
Preprocessing and Data Preparation
High-Resolution Aerial Images
B
G
R
Patching and Selecting Informative Patches
Images
Masks
Data Augmentation
Images & Masks
Original
Data Splitting
25 %
75 %
Traning
Validation
Model Design and Training
Network Structure and Convolutional Filter Design
Hyperparameter Tuning and Optimizer Selection
SGD
Adam
Adadelta
AdamW
Nadam
RMSprop
Model Training and Results Evaluation
Optimizer Identification and Development
Identification of Top Optimizers
DECO Design, Development, Model Training, and Results Evaluation
DECO:
AdamW
RMSprop
Nadam
Model Generalization and Validation
Evaluation of the DECO-Optimizer-Based Model on a New Area

Figure 3: Proposed Method in This Study for Land Use Classification

According to Figure 3, after calling the high-resolution aerial image in the data preparation section, an expert manually labeled four land uses: buildings, roads, woodland cover, and water. Subsequently, the entire image, along with the labeled image, was cropped into smaller images of dimensions 256x256 pixels to be used as samples for training, validation, and testing datasets from these images and their corresponding labels. Data augmentation techniques included rotations at 90 and 180 degrees and horizontal and vertical flipping. The use of data augmentation techniques in deep learning is performed for two main reasons:

- **Increasing data volume:** Generating new and diverse samples from existing data helps the model train on a larger dataset, thereby improving the accuracy and performance of the model.
- **Reducing overfitting:** Applying random transformations such as rotation, scaling, and adding noise to the training data encourages the model to focus on more general patterns, leading to better accuracy and performance on new data, thus reducing overfitting.

Figure 4 displays the data augmentation techniques applied to a sample image.

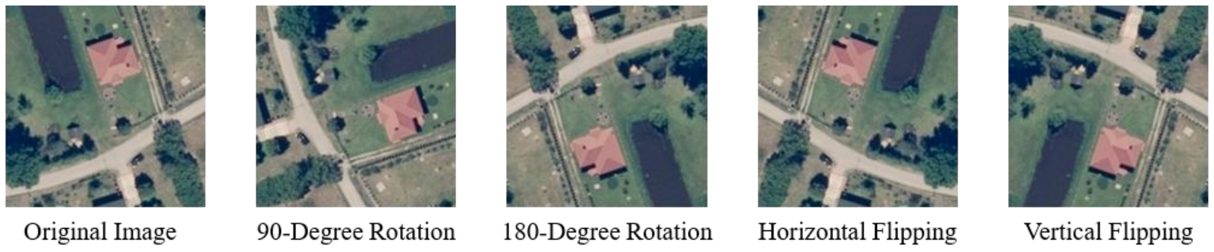

Figure 4: A representation of data augmentation (rotation and flipping techniques)

In the next step, the images were divided into training, validation, and test sets to train and evaluate the U-Net deep learning model. This partitioning is crucial for maintaining the independence and proper functioning of the model. Approximately 75% of the total data was allocated for training the model, which included images and labels related to four different land use categories (buildings, woodlands, roads, and water). These data were randomly selected to allow the model to learn data diversity and perform better in segmentation tasks. The remaining 25% of the data was used for validation. This validation set was employed to evaluate the model during training to ensure optimal model settings and prevent overfitting. The model's performance on this set served as an initial evaluation metric.

Additionally, 5% of the validation dataset was randomly selected and assigned to the test set. This smaller dataset was used to evaluate the model's performance after completing the training. The results obtained from this set were utilized as the final assessment of the model's land use segmentation performance. To ensure the model's generalizability, all datasets included images from each of the four desired land use types, chosen randomly and without overlap. This process ensures that the model can accurately and effectively identify and segment different types of land use.

The next step involves designing the network architecture and convolutional filters. A thorough understanding of the U-Net architecture is required to design the network structure and convolutional filters. U-Net is a convolutional neural network (CNN) architecture initially introduced in 2015 for medical image processing applications by Olaf Ronneberger and colleagues (Ronneberger, Fischer and Brox, 2015). This architecture is named U-Net due to its structural resemblance to the letter "U." U-Net utilizes a structure that comprises top-down convolutional layers followed by bottom-up convolutional layers. The main structure of U-Net consists of two primary parts:

- **Contracting Path:** In this part, the input images are reduced through convolutional and pooling layers (usually max pooling layers), extracting essential features for subsequent decisions. This process is performed gradually, leading to a reduction in image dimensions and an increase in depth (the number of channels).
- **Expansive Path:** In this section, the image dimensions are increased using bottom-up convolutional layers and upsampling convolutional layers, and the features extracted from

the encoding part are reconstructed. This part reconstructs the image accurately using the features obtained from the encoding section.

Figure 5 illustrates the architectural structure of the U-Net deep learning model implemented in this study.

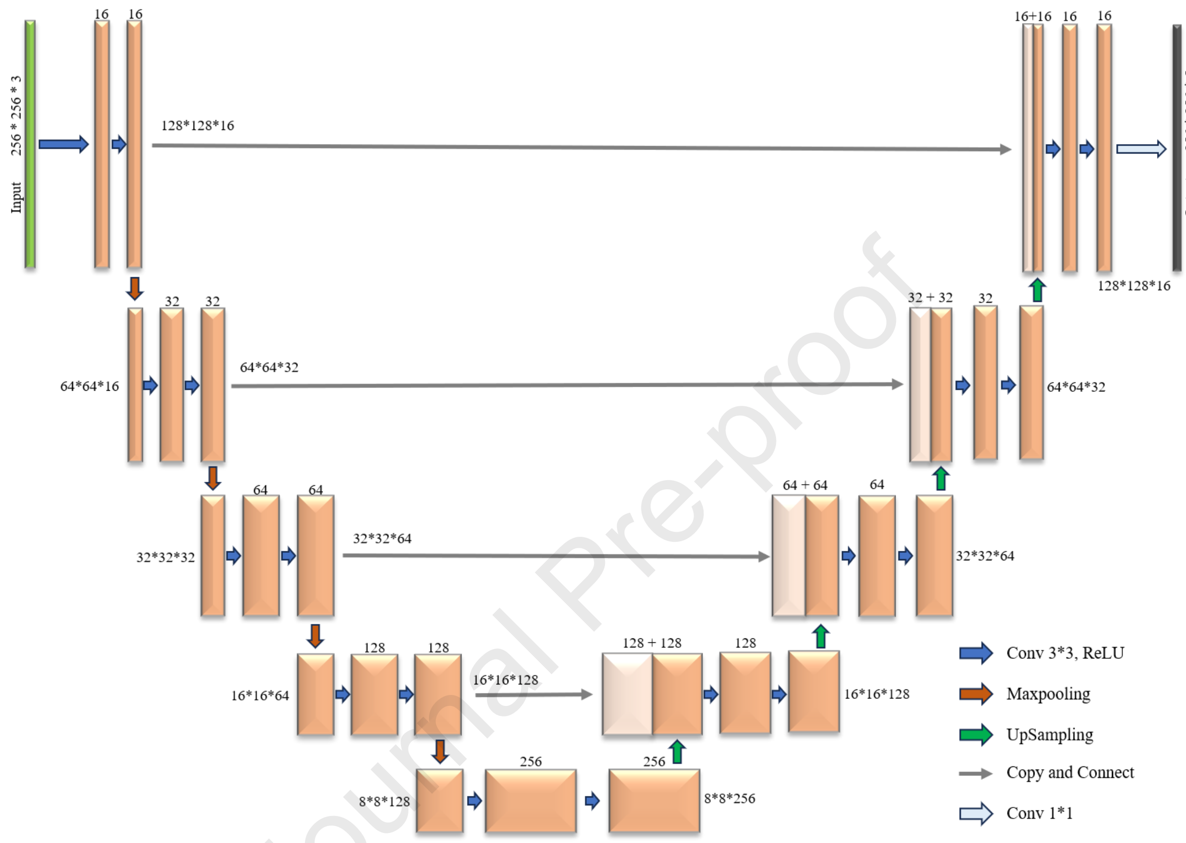

Figure 5: U-Net architecture structure.

The primary function of U-Net in image applications is particularly focused on identifying and segmenting objects and features in images, as well as other image-processing tasks. Unlike entional CNNs that typically consider the contextual information surrounding a point, U-Net can take a greater amount of contextual information into account and utilize it for more accurate and improved image reconstruction, especially in cases requiring precise reconstruction of images or small objects within images. The architecture of the U-Net model designed in this study, as illustrated in Figure 5, consists of convolutional blocks, each comprising two convolutional layers with 3×3 kernels, a ReLU activation function, batch normalization, and dropout. Batch normalization contributes to improved convergence and mitigates the risk of overfitting. Dropout, by randomly deactivating certain neurons in each layer, prevents the model from over-relying on specific training data features and thus enhances generalization to unseen data.

In the design of the network structure, the model includes three main components: the encoding path, the decoding path, and the output layer, each serving specific purposes in the image processing and segmentation process. The encoding path consists of five stages, in which the dimensions of the convolutional filters decrease (from a 256x256 filter in the first stage to a 16x16 filter in the final stage). This process allows for extracting more complex features from images in the lower stages and lower-level features, such as edges and textures, in the higher stages. Additionally, at the end of each stage, a max pooling layer is used to reduce the image dimensions and control model complexity. The decoding path also consists of five stages; however, in each stage, the image's dimensions are increased using Conv2DTranspose layers.

Furthermore, the output of each layer is combined with the corresponding output from the encoding path with matching dimensions to transfer spatial information from the encoding path to the decoding path, ultimately resulting in a high-resolution output with desirable accuracy in image segmentation. Finally, a convolutional layer with a 5x5 filter and a 1x1 kernel is designated for the model's output. The output of this layer matches the image's dimensions, with each output pixel representing the probability of that pixel belonging to one of five classes (four land-use classes plus background). The Softmax operator is employed to convert the output into a land-use label. The deep learning model assigns features that do not fall into the defined four land-use classes to the background category.

Hyperparameter tuning is critical in the design of the U-Net model, as these settings directly impact the model's performance and accuracy. Correctly selecting hyperparameters can reduce training time and prevent overfitting in the model. Conversely, improper tuning of these parameters can lead to decreased accuracy and fluctuations in the training process; thus, achieving a proper balance between model accuracy and efficiency is essential. The most important hyperparameters tuned in this study include:

- Optimizer Functions: The optimizers used in the U-Net model include:
  - ✓ **SGD (Stochastic Gradient Descent):**
    A simple and effective optimization method where gradients are updated randomly in small batches (Robbins and Monro, 1951). This optimization technique updates the parameters θ at each time step *t*. The following equation gives the weight update rule in SGD (Bera and Shrivastava, 2020).

    (1) $$\theta_{t+1} = \theta_t - \eta d_t$$

    - θ_t: Model parameters at time step *t*.
    - η: Learning rate, which determines the size of the updates to the model parameters.
    - d_t: Gradient of the objective function with respect to the parameters at time step *t*.
  - ✓ **Adam (Adaptive Moment Estimation):**
    Adam is one of the most widely used optimizers, offering faster and more stable learning by combining the advantages of the Adagrad and RMSprop methods (Kingma and Ba, 2014).

    (2) $$\hat{a}_t = \frac{a_t}{1 - \beta_1^t}$$

(3) $$\hat{u}_t = \frac{u_t}{1-\beta_2^t}$$

(4) $$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{u}_t + \in}}\hat{a}_t$$

- a_t: The gradients (first moment) at step *t*.
- u_t: The squared gradients' moving average (second moment) at step *t*.
- $\hat{a}_t$: The corrected ata_tat, adjusted with the factor β_1 and exponential decay across different time steps.
- $\hat{u}_t$: The corrected utu_tut, representing the variance of the gradients, similarly adjusted using β_2.
- θ_t: The model parameters at step *t*.
- $\in$: A small constant value to prevent division by zero.
- β_1^t: The exponential decay rate for the first moment (gradients).
- β_2^t: The exponential decay rate for the second moment (squared gradients).

In this model, the corrections are made using exponentially weighted averages of gradients and variances. This allows the model to progress towards optimization more quickly and stably.

✓ **Adadelta:**
Adadelta is an adaptive optimizer that automatically adjusts the learning rate based on recent changes in the model parameters, reducing the need for manual tuning of the learning rate (Zeiler, 2012). This optimizer improves the Adagrad algorithm, uniformly decreasing the learning rate. Unlike Adagrad, which uses the sum of all past squared gradients to update the learning rate, Adadelta employs a fixed-size window for accumulating past gradients(Bera and Shrivastava, 2020). The sum of the gradients is treated as a moving average of the past squared gradients, with the decay factor γ controlling the contribution of previous gradients at each time step. In this method, the moving average at time step *t* depends on the previous average and the current gradient.

(5) $$R[d^2]_t = \gamma R[d^2]_{t-1} + (1-\gamma)d_t^2$$

(6) $$\varDelta\theta_t = -\frac{\eta}{\sqrt{D_t + \in}}d_t$$

- d_t: The gradient of the loss function with respect to the parameters at time *t*.
- γ: The decay rate, which is typically set to 0.9. This constant determines how much previous gradients are retained in the moving average calculation.

- $R[d^2]_t$: The exponentially weighted moving average of the squared gradients up to time *t*. This value acts as a memory for the previous gradients.
- D_t: The diagonal matrix where the updated values are placed instead of the exponentially weighted moving average of squared gradients.
- η: The learning rate.
- $\in$: A small constant used to prevent division by zero in computations.

Equations (7) and (8) perform parameter updates using the root mean square (RMS) of the gradient error.

$$\Delta\theta_t = -\frac{\eta}{\sqrt{R[d^2]_t + \varepsilon}} d_t \tag{7}$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{(RMS)[d]_t}} d_t \tag{8}$$

$$RMS[\Delta\theta]_t = \sqrt{R[\Delta\theta^2]_t + \varepsilon} \tag{9}$$

Here, $R[\Delta\theta^2]_t$ represents the exponential moving average of the squared updates of the parameters and is defined in Equation 10.

$$R[\Delta\theta^2]_t = \gamma R[\Delta\theta^2]_{t-1} + (1-\gamma)\Delta\theta_t^2 \tag{10}$$

Since $\Delta\theta_t$ the current time step is unknown, the previous update rule with $(RMS)[\Delta\theta]_{t-1}$ is used to derive the Adadelta update expression in Equation 9. Ultimately, the Adadelta rule is presented in Equation 12 (Bera and Shrivastava, 2020).

$$\Delta\theta_t = -\frac{(RMS)[\Delta\theta]_{t-1}}{(RMS)[d]_t} d_t \tag{11}$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t \tag{12}$$

- $R[\Delta\theta^2]_t$: Parameter update at step t.
- $RMS[\Delta\theta]_t$: Root Mean Square of the gradients.

✓ **AdamW:**

AdamW is an improved version of Adam that directly adds the weight decay to the parameter updates instead of incorporating it into the gradients (Loshchilov and Hutter, 2017). This approach helps prevent unwanted overfitting in the model (Pagliardini, Ablin and Grangier, 2024).

$$(13)\quad \begin{cases} m(t) = \beta_1 m(t-1) + (1-\beta_1)g(t) \quad , \quad \widehat{m}(t) = \dfrac{m^{(t)}}{1-\beta_1^t} \\ v(t) = \beta_2 v(t-1) + (1-\beta_2)g(^t \quad , \quad \widehat{v}(t) = \dfrac{v(t)}{1-\beta_2^t} \\ \theta(t) = \theta(t-1) - \eta(\dfrac{\widehat{m}(t)}{\sqrt{\widehat{v}(t)} + \varepsilon} + \lambda\theta(t-1)) \end{cases}$$

- $g(t)$: The gradient of the objective function with respect to the model parameters at time t.
- λ: The weight decay coefficient.

✓ **Nadam (Nesterov-accelerated Adaptive Moment Estimation):**
Nadam is a variant of Adam that uses the Nesterov Momentum technique to improve convergence speed. Nadam is a type of weight update rule (Dozat, 2016). This method is calculated by extending equations (2) and (4) with the use of terms a_t and $\hat{a}_t$, as shown in equation 14 (Bera and Shrivastava, 2020).

$$(14)\quad \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{u}_t + \varepsilon}}(\frac{\beta_1 a_{t-1}}{1-\beta_1^t} + \frac{1-\beta_1}{1-\beta_1^t} d_t)$$

- d_t: The gradient of the objective function with respect to the parameters at step t.

✓ **RMSprop (Root Mean Square Propagation):**
An optimization method that adjusts the learning rate based on the root mean square of recent gradients is effective in deep learning problems with varying parameters (Hinton, Srivastava and Swersky, 2012). The weight update rule in RMSprop adjusts the learning rate by dividing it by the square root of the exponentially weighted average of the squared gradients $R[d^2]_t$. This rule is presented in Equation 15 (Bera and Shrivastava, 2020).

$$(15)\quad \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{R[d^2]_t + \varepsilon}} d_t$$

- θ_t: Model parameters at time step t.

RMSprop, by automatically adjusting the learning rate at each step, effectively prevents sudden fluctuations in the convergence path and accelerates the learning process of complex models.

- Learning Rate: A learning rate of 0.0007 has been set. The learning rate is a parameter that determines the amount by which the model's weights are updated during each iteration. An appropriate learning rate can significantly impact the convergence speed and the final performance of the model.

- Batch Size: A batch size of 32 has been chosen. Batch size refers to dividing the training data into smaller subsets used to train the model in each iteration. Choosing an appropriate batch size can affect the memory requirements for training the model and its convergence speed.
- Model Configuration and Structure: The U-Net model is configured to perform image segmentation tasks with high accuracy and efficiency, achieving satisfactory results in land-use segmentation.

Table 1 provides a summary of the designed U-Net network architecture.

Table 1: Designed U-Net Architecture

Section	Description
Convolutional Blocks	It consists of two convolutional layers with a kernel size of 3x3
Activation Function	ReLU
Batch Normalization	Improves convergence and reduces overfitting
Dropout	Prevents excessive dependence on specific features
Encoder Path	5 stages with increasing filters (from 16 to 256)
Max Pooling	Reduces image dimensions and controls model complexity
Decoder Path	5 stages with Conv2DTranspose to increase image dimensions
Output Combination	Transfers spatial information from the encoder path
Output Layer	5 filters with a kernel size of 1x1
Optimizers	SGD, Adam, Adadelta, AdamW, Nadam, and RMSprop
Learning Rate	0.0007
Batch Size	32
Max Epochs	400
Early-Stopping	Monitor: validation loss, Patience: 5 epochs restore best weights: True

The next step in land use segmentation is the training and evaluation the developed model. This evaluation involves using accuracy, Kappa coefficient, F1 Score, and Jaccard Score metrics to ensure that the model has accurately identified and classified various land uses. These metrics help to assess the model's performance more precisely and identify its strengths and weaknesses.

In addition to U-Net, this study also utilized the DeepLab v3+ architecture as a secondary deep learning model to validate the generalizability of the proposed DECO optimizer. DeepLab v3+ is an advanced convolutional neural network for semantic segmentation that, like U-Net, features an encoder-decoder structure but introduces several important distinctions. Its encoder uses atrous (dilated) convolutions, enabling the network to capture multi-scale contextual information without losing spatial resolution. The Atrous Spatial Pyramid Pooling (ASPP) module is a key innovation, aggregating features at multiple scales and enhancing segmentation of objects with varying sizes. The decoder module in DeepLab v3+ further refines segmentation outputs by gradually recovering spatial details lost during encoding, enabling more precise delineation of object boundaries, especially in complex scenes.

In this study, DeepLab v3+ was implemented with the same dynamic optimizer switching strategy as U-Net, allowing direct comparison of DECO's impact on both architectures. Key hyperparameters—including learning rate, batch size, maximum epochs, and early stopping—were kept consistent for fair comparison. Both models were evaluated on identical training and test datasets using the same performance metrics: overall accuracy, Kappa coefficient, F1 score, Jaccard index, and confusion matrix. This dual-model approach ensures robust validation of DECO's effectiveness and generalizability across distinct deep learning segmentation frameworks.

Overall accuracy (Equation 16) is one of the simplest and most commonly used metrics for evaluating the performance of segmentation models. This metric indicates the ratio of the number of correctly segmented samples to the total number of samples (Pashaei *et al.*, 2020). In other words, overall accuracy shows how well the model has performed in correctly identifying samples.

$$(16)\quad Overall\ Accuracy = \frac{TP}{TP + TN + FP + FN}$$

In this context:

- TP: The number of samples correctly segmented as positive.
- TN: The number of samples correctly segmented as negative.
- FP: The number of samples incorrectly segmented as positive.
- FN: The number of samples incorrectly segmented as negative.

The Kappa coefficient (Equation 17) measures the agreement between model predictions and ground truth while accounting for chance agreement. For a multi-class problem with K classes and total samples N, it is defined as:

$$(17)\quad Kappa\ Coefficient = \frac{p_o - p_e}{1 - p_e}$$

Where:

$$(18)\quad p_o = \frac{1}{N}\sum_{K=1}^{K=5} TP_k$$

is the observed overall accuracy (sum of true positives across all classes divided by N), and

$$(19)\quad p_e = \sum_{K=1}^{K=5} \left(\frac{TP_k + FP_k}{N} \times \frac{TP_k + FN_k}{N}\right)$$

is the expected accuracy by chance, calculated from the marginal totals of the confusion matrix.

The F1 Score (Equation 20) is a widely used metric for evaluating the performance of classification models in class imbalance problems. The F1 Score is the harmonic mean of precision and recall and is particularly useful in cases where the balance between true positives and false negatives is challenging (Wang *et al.*, 2023). The F1 Score ranges from 0 to 1, with higher values indicating better model performance. Due to the multi-class nature of the task, F1, Precision, Recall, and Jaccard metrics were computed for each class in a one-vs-all manner and then macro-averaged across all classes. The F1 Score is calculated as follows:

$$(20)\quad F1\ Score_k = \frac{2 \times Precision_k \times Recall_k}{Precision_k + Recall_k}\ ,\quad F1\ Score_{macro} = \frac{1}{K}\sum_{K=1}^{K=5} F1\ Score_k$$

Where:

- *Precision*: The percentage of true positive predictions out of all positive predictions made by the model. In other words, precision tells us how much of the model's positive predictions were correct.

$$(21)\quad Precision_k = \frac{2 \times TP_k}{TP_k + FP_k}$$

- *Recall*: The percentage of actual positive samples correctly identified by the model. This metric indicates how well the model has identified all positive samples.

$$(22)\quad \mathrm{Re}call_k = \frac{TP_k}{TP_k + FN_k}$$

The F1 Score is suitable for evaluating models that may struggle with identifying rare classes, as it considers both Precision and Recall and establishes a balance between them.

The Jaccard Score (Equation 23) measures the similarity between two sets. This metric is often used in the contexts of classification and clustering. The Jaccard Score is calculated between 0 and 1 or as a percentage, with higher values indicating greater similarity between the two sets.

$$(23)\quad Jaccard\ Score = |A \cap B| / |A \cup B|$$

Where:

- $|A \cap B|$: The number of elements common to sets *A* and *B*.
- $|A \cup B|$: The total number of elements in sets *A* and *B*.

These metrics are widely used to evaluate the performance of machine learning and deep learning models in various segmentation and image segmentation tasks, each with advantages and specific applications.

This paper uses statistical tools such as ANOVA and Tukey HSD to analyze the results and evaluate the performance of various optimizers in improving deep learning models. Additionally, ANOVA (Analysis of Variance) is employed to assess significant differences between the performances of different optimizers. ANOVA is a statistical method used to examine mean differences among several groups. Here, the F statistic (Equation 24) is calculated as the ratio of the mean squares between groups (*MSB*) to the mean squares within groups (*MSW*).

$$(24) \quad F = \frac{MSB}{MSW}$$

Where:

- *MSB*: Represents the variance between the means of different groups.
- *MSW*: Represents the variance within each group.

Partial eta squared is an effect-size measure that quantifies the proportion of total variance in the dependent variable attributable to a given factor, after accounting for other sources of variance. It is calculated as:

$$(25) \quad \eta_p^2 = \frac{SS_{between}}{SS_{between} + SS_{within}}$$

where $SS_{between}$ is the sum of squares between groups and SS_{within} is the sum of squares within groups. A higher η_p^2 indicates a larger effect of the factor under study.

A 95 % confidence interval gives a range of values within which we can be 95 % confident that the true population parameter (e.g., a mean difference) lies. For a mean difference Δ, it is computed as:

$$(26) \quad CI_{95\%} = \Delta \pm t_{0.975,df} \times SE_{\Delta}$$

where $t_{0.975,df}$ is the critical t-value for 95 % confidence with the given degrees of freedom, and SE_{Δ} is the standard error of the difference.

Validation accuracy at each optimizer's final training epoch was compared using a one-way ANOVA. The F statistic, p-value, and partial η^2 (η^2_p)—an effect-size measure indicating the proportion of variance in accuracy attributable to optimizer choice—were reported. A p-value below 0.05 was considered evidence of statistically significant differences in mean accuracy across optimizers.

Post-hoc pairwise comparisons were then performed via Tukey's Honest Significant Difference (HSD) test ($\alpha = 0.05$), with p-values adjusted for multiple comparisons. For each significant mean difference Δ, the 95 % confidence interval (CI) was also calculated. These procedures ensure that the identification of performance gaps among optimizers is both valid and reliable, providing a robust foundation for the development of the DECO hybrid optimizer.

Furthermore, each optimizer calculated the average accuracy and mean validation error over the final 60 training epochs to select the best optimization algorithms for the deep learning model. Additionally, the stability of each model was assessed using the standard deviation of accuracy and error over the last 10 epochs. Reduced fluctuations in model performance indicate better and more stable convergence. The optimizers were ranked using a composite criterion that includes maximum accuracy, minimum error, and greater stability in accuracy and error. This criterion identified the three optimizers exhibiting the highest average accuracy, lowest average validation error, and greatest stability (least standard deviation) as the best candidates for designing and developing the hybrid optimizer DECO. In the final step, after identifying, designing, and developing the optimal combination of optimizers, the U-Net model was trained with the hybrid optimizer DECO, and the obtained results were compared with those from six fixed optimizers.

4. Implementation and Results

The Python programming language and the Keras deep learning library were utilized within the TensorFlow framework to develop and implement this deep learning model. To expedite the training and evaluation process, all execution steps were carried out in the powerful Google Colab environment and service.

An expert carried out the labeling process to ensure the accuracy and precision of the data labeling. This labeling was performed for four land uses, buildings, woodlands, roads, and water, to enhance the diversity and generalizability of the deep learning model. Figure 6 shows an example of the labels prepared.

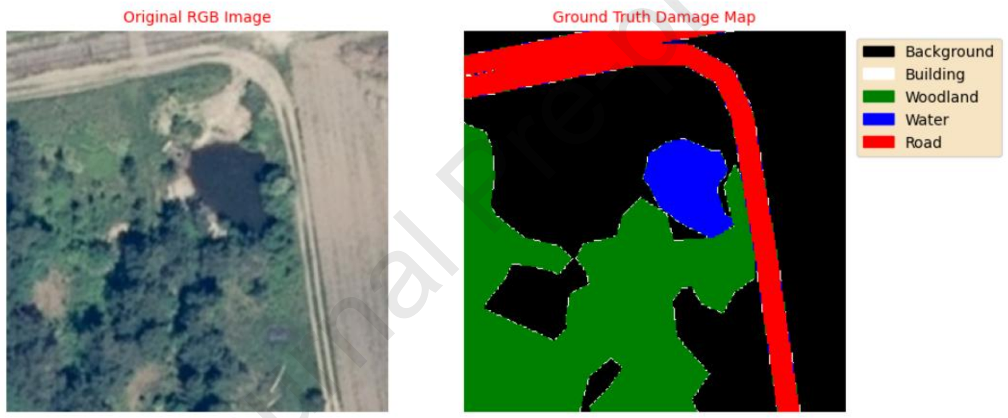

Figure 6: Labels prepared by an expert for the studied land uses.

Out of 707 labeled images with dimensions of 256x256 pixels, 530 images were allocated to the training dataset. After applying data augmentation techniques, the training samples increased to 2,650. Additionally, 177 images were designated for the validation set, and 10 were allocated to the test set.

After configuring and designing the U-Net architecture, the optimizers SGD, Adam, Adadelta, AdamW, Nadam, and RMSprop were sequentially selected, and the training and validation datasets and their corresponding labels were introduced to the model. In the next step, the U-Net model reached its maximum accuracy after several epochs and halted its training process using the Early Stopping technique.

After training the U-Net model with the mentioned optimizers, six land use maps of the study area were generated using each of the six optimizers. Figure 7 displays the map produced by the model for the entire study area alongside the map prepared by the expert.

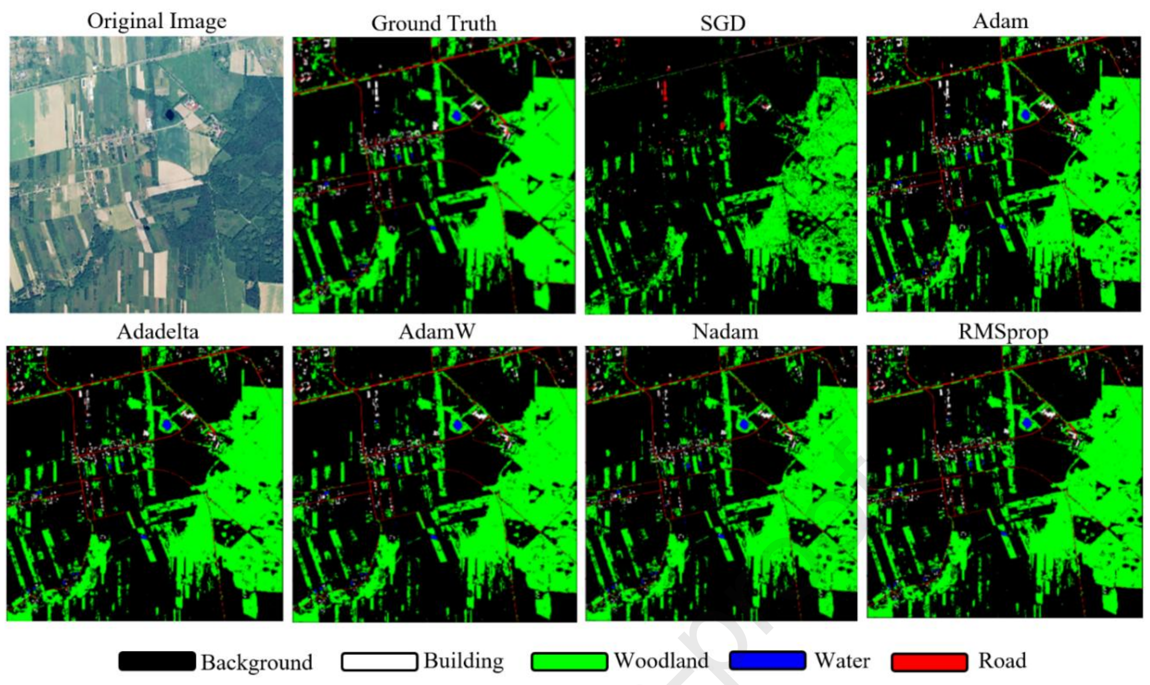

Figure 7: Prediction of the U-Net deep learning model in land-use mapping, differentiated by the optimizers used

As observed, the results of different optimizers show variations in class segmentation. For instance, the AdamW optimizer has been more effective in separating the classes of roads and buildings from the background, whereas the SGD and Adadelta optimizers exhibit poorer results in distinguishing these classes. These results underscore the importance of selecting an appropriate optimizer to enhance accuracy in land-use mapping.

The Confusion Matrix is a powerful tool for evaluating the performance of deep learning models in classifying classes. This matrix compares the model's predictions with the actual data, providing detailed information about the number of correct and incorrect predictions. This study calculated and analyzed the Confusion Matrix for the U-Net model trained with various optimizers.

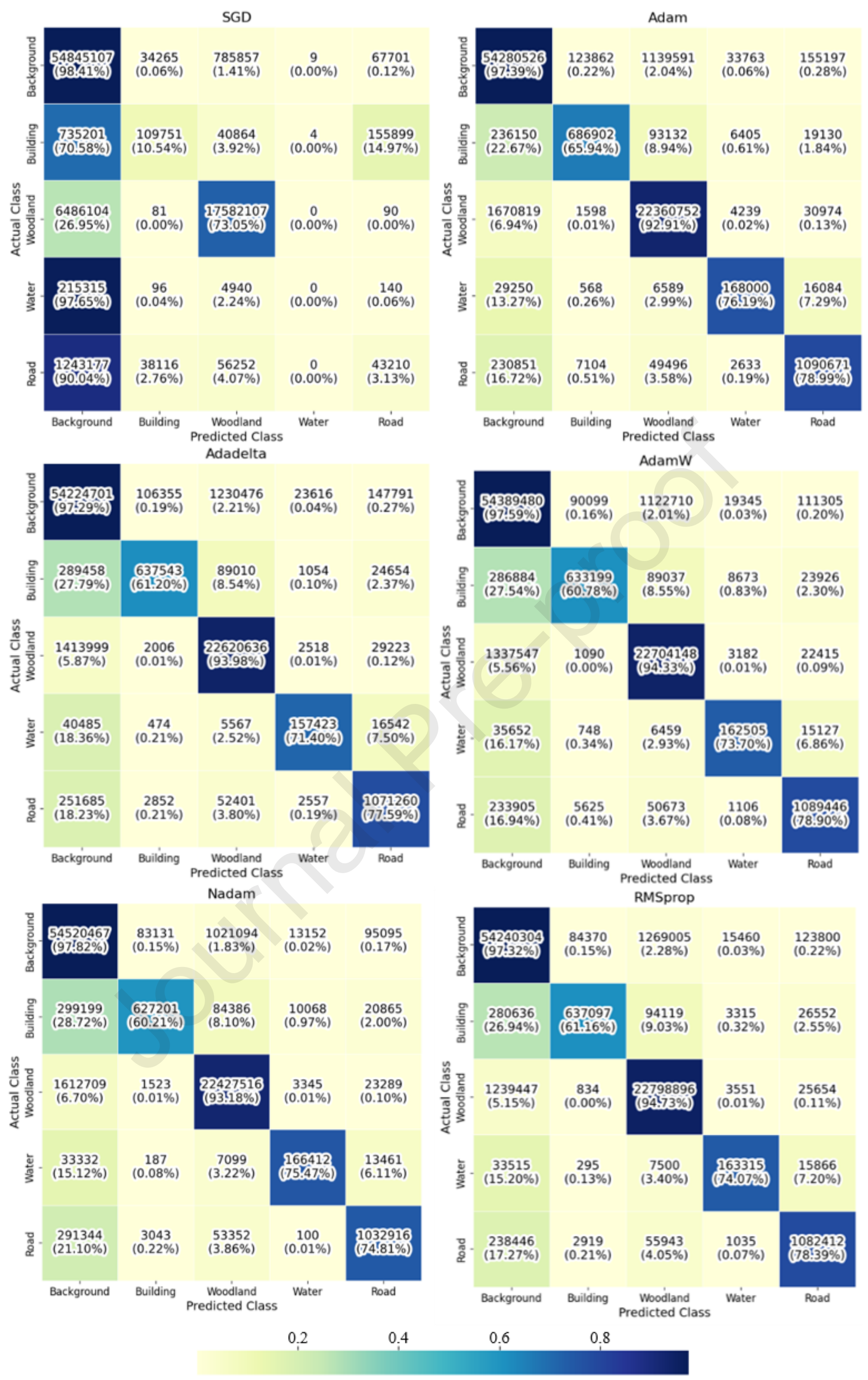

Figure 8: Confusion Matrix of the U-Net Deep Learning Model with Different Optimizers

The analysis of the Confusion Matrix results for various optimizers in the U-Net deep learning model reveals that each optimizer has its strengths and weaknesses in identifying different classes. Optimizers such as RMSprop, Nadam, and AdamW have demonstrated superior performance in identifying more complex classes like Woodland and Water, achieving acceptable accuracy in these classes. The SGD optimizer also demonstrated poor performance in identifying and distinguishing classes from one another, achieving a 98.41% accuracy only in distinguishing the Background class. The detailed per-class performance metrics for each optimizer are summarized in Table 2 below. For each optimizer, the table reports precision, recall, F1-score and the total number of pixels (support) for each land-cover class. This allows a direct comparison of how well each optimizer distinguishes between Background, Building, Woodland, Water, and Road. Notably, while Adam and its variants (AdamW, Adadelta) maintain high precision on Background and Woodland, they exhibit lower recall on underrepresented classes such as Water and Road. Conversely, RMSprop and Nadam achieve stronger recall on Road but at the cost of reduced precision on Water. SGD, despite its overall high accuracy on Background, struggles most with smaller classes like Water. These per-class breakdowns highlight the trade-offs each optimizer makes and point to potential avenues for balancing class-level performance in future model improvements.

Table 2. Per-class performance metrics (Precision, Recall, F1-score, and Support in pixels) for each optimizer.

Model	Optimizer	Class	Precision	Recall	F1-score	pixels
U-Net	SGD	Background	86.34	98.41	91.98	55732939
		Building	60.20	10.54	17.93	1041719
		Woodland	95.19	73.05	82.66	24068382
		Water	0.00	0.00	0.00	220491
		Road	16.18	3.13	5.24	1380755
	Adam	Background	96.16	97.39	96.77	55732939
		Building	83.77	65.94	73.79	1041719
		Woodland	94.55	92.91	93.72	24068382
		Water	78.12	76.19	77.15	220491
		Road	83.13	78.99	81.01	1380755
	Adadelta	Background	96.45	97.29	96.87	55732939
		Building	85.09	61.20	71.20	1041719
		Woodland	94.26	93.98	94.12	24068382
		Water	84.11	71.40	77.23	220491
		Road	83.08	77.59	80.24	1380755
	AdamW	Background	96.63	97.59	97.11	55732939
		Building	86.65	60.78	71.45	1041719
		Woodland	94.71	94.33	94.52	24068382
		Water	83.42	73.70	78.26	220491
		Road	86.31	78.90	82.44	1380755
	Nadam	Background	96.06	97.82	96.93	55732939
		Building	87.71	60.21	71.40	1041719
		Woodland	95.06	93.18	94.11	24068382
		Water	86.19	75.47	80.48	220491
		Road	87.12	74.81	80.50	1380755
	RMSprop	Background	96.80	97.32	97.06	55732939
		Building	87.81	61.16	72.10	1041719

		Woodland	94.11	94.73	94.42	24068382
		Water	87.49	74.07	80.22	220491
		Road	84.94	78.39	81.54	1380755

A closer look at Table 2 reveals clear trade-offs among the optimizers. SGD excels at identifying the abundant Background class (91.6 % F1) but struggles with the small Water class (26.4 % F1), indicating low sensitivity to underrepresented categories. Adam and its variants (Adadelta, AdamW) achieve more balanced F1-scores across classes, particularly improving Water detection (up to 43.4 % F1 with Adam and 28.1 % with Adadelta) at the cost of slightly lower precision on Background. Nadam and RMSprop offer stronger support for Road (up to 49.7 % and 38.8 % F1 respectively) and maintain moderate trade-offs between Precision and Recall. Overall, if prioritizing rare-class identification (e.g. Water, Road), Adam or Nadam may be preferable, whereas SGD remains optimal for dominant classes.

Additionally, Table 3 assesses and compares the performance of both the U-Net and DeepLab v3+ models using each of the studied optimizers based on key evaluation metrics.

Table 3: Evaluation of Optimizer Performance in U-Net and DeepLab v3+ Deep Learning Models by Assessment Metrics

Model	Optimizer	Overall Accuracy	Kappa Coefficient	F1 Score	Jacard Score
U-Net	SGD	88.04	71.07	86.62	28.023
	Adam	95.32	89.67	95.28	61.87
	Adadelta	95.47	90.02	95.41	61.33
	AdamW	**95.80**	**90.73**	**95.73**	62.33
	Nadam	95.55	90.12	95.47	62.19
	RMSprop	95.73	90.60	95.67	**62.48**
DeepLabv3+	SGD	**88.02**	**77.51**	**87.64**	40.49
	Adam	86.02	66.21	85.09	**40.65**
	Adadelta	83.85	60.63	82.68	37.34
	AdamW	83.62	59.53	82.28	39.43
	Nadam	81.19	52.11	79.10	38.87
	RMSprop	82.25	55.74	80.65	33.45

The results in Table 3 indicate that for the U-Net model, AdamW achieves the highest scores across all four metrics, making it the superior optimizer in terms of balancing accuracy and convergence—well suited for high-precision applications. RMSprop and Nadam also demonstrate strong performance with U-Net. In contrast, SGD shows weaker results and may require additional tuning for better outcomes.

For the DeepLab v3+ model, SGD provides the highest overall accuracy and F1-score among the tested optimizers, while also achieving the highest Kappa coefficient. Adam and Adadelta deliver relatively good F1 and Jaccard scores but do not outperform SGD in overall metrics. It is noteworthy that AdamW, which was most effective for U-Net, did not yield the top results for DeepLab v3+. These findings underscore the importance of model architecture in the effectiveness of different optimizers, and further validate the rationale for a dynamic, hybrid approach such as DECO.

In the next step, the accuracy, training, and validation errors of each optimizer's U-Net deep learning model were obtained and presented in Figure 9.

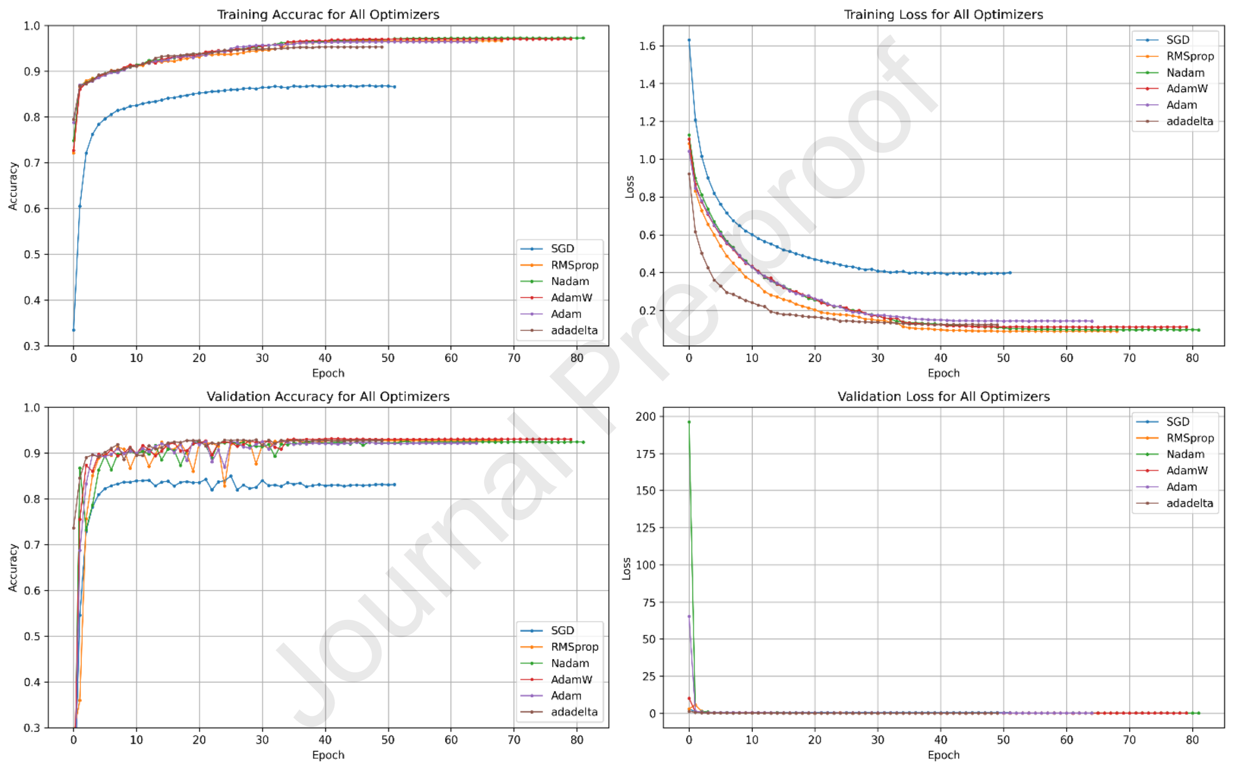

Figure 9: The U-Net deep learning model's training and validation accuracy and error categorized by optimizers.

We first ran a one-way ANOVA to compare validation accuracy across six optimizers and found F(5, 354) = 102695.99, $p < 0.001$, with a very large effect size (partial $\eta^2 = 0.85$), indicating that 85 % of the variance in accuracy is attributable to optimizer choice. To uncover exactly which optimizers differed, we conducted Tukey's Honest Significant Difference test ($\alpha = 0.05$), with p-values adjusted for multiple comparisons. Notable pairwise results included: AdamW outperforming Adadelta by 0.30 pp (95 % CI [0.18, 0.42], $p < 0.001$), AdamW beating Adam by 0.28 pp (95 % CI [0.16, 0.40], $p < 0.001$), RMSprop and Adadelta showing a negligible 0.01 pp difference (95 % CI [–0.09, 0.11], $p = 0.996$), and Nadam exceeding RMSprop by 0.33 pp (95 % CI [0.20, 0.46], $p < 0.001$). These detailed contrasts demonstrate that no single static optimizer excels in every training phase; rather, each exhibits strengths at different epochs—providing a compelling rationale for our Dynamic Epoch-Centric Optimizer (DECO) to switch adaptively and combine their individual advantages. However, this evaluation may not be sufficient for designing and developing a hybrid optimizer model. Therefore, a more reliable assessment and better ranking of optimizers necessitate exploring features such as stability and mean validation regarding accuracy and error. The results of this evaluation are presented in Figure 10 below.

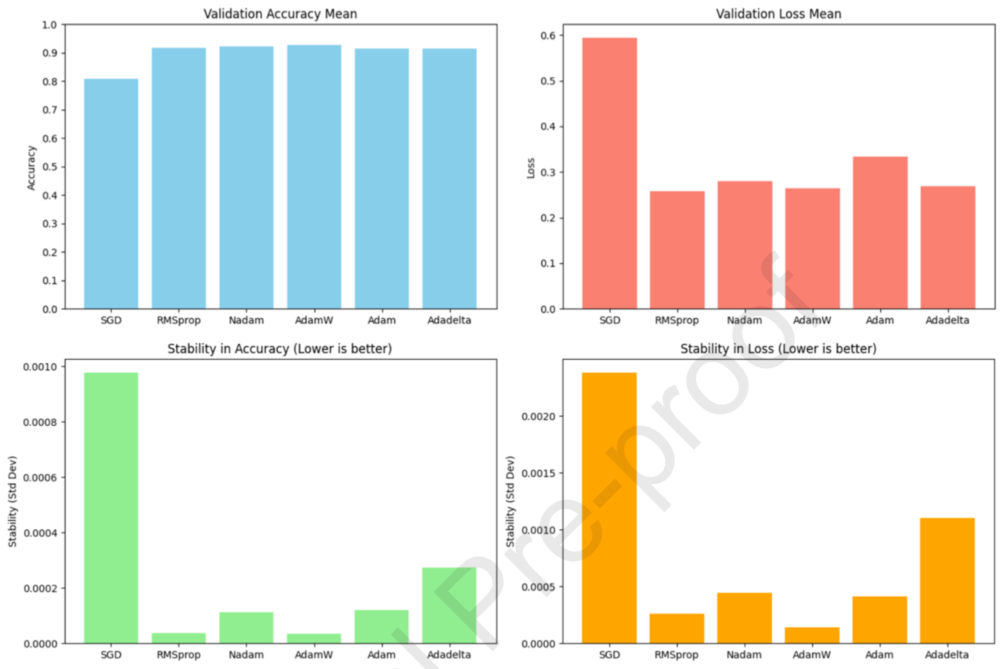

Figure 10: Comparison of optimizer performance based on stability metrics and mean accuracy and error on the validation dataset

The combined and dynamic use of these optimizers could lead to the development of the U-Net model with better generalization capabilities and higher performance when handling complex and variable data. These choices are based on a comprehensive analysis and comparison of various metrics, including mean accuracy, mean error, and performance stability, as depicted in Figure 10. These results underscore the importance of a multidimensional examination of optimizer performance in optimizing deep learning models.

In the training process of deep learning models, the choice of optimizer and its precise tuning across different epochs is crucial for the final model performance. Models exhibit varying behaviors at different stages of training; therefore, using a single optimizer throughout the entire training period may result in reduced effectiveness. Consequently, segmenting epochs into specific intervals and employing different optimizers in each interval can enhance model performance.

This study has divided epochs into four time intervals: 0 to 20, 20 to 40, 40 to 60, and beyond 60. In the initial epochs (0 to 20 and 20 to 40), the model is learning rapidly, with significant changes observed in accuracy and error. Due to intrinsic fluctuations and a high learning rate during these periods, the criterion for selecting the optimizer is based on accuracy and error reduction changes. As shown in Figures 9 and 10, the performance of the SGD optimizer during these stages is poor; therefore, this optimizer was utilized only in the final two intervals.

In the intermediate and final stages of training (40 to 60 and beyond 60), the focus shifts toward model stability and fine-tuning. During this phase, a stability criterion (based on the standard deviation of accuracy) is also employed in addition to accuracy and error changes. This approach allows for more precise model optimization in the final stages, ensuring improvements in the final model's stability and accuracy.

In implementing this optimization system, the program determines the optimal optimizer for each interval based on the performance evaluation results of each optimizer and utilizes it in the training phases. This means that in each time interval, the program will identify the most suitable optimizer, and then the DECO (Dynamic Epoch-Centric Optimizer) will combine these optimizers in succession to achieve the best possible model performance. With the automatic switching of optimizers based on specified time intervals and criteria, this process ensures increased accuracy while maintaining model stability during advanced training phases.

In this method, the training begins with the AdamW optimizer, demonstrating the best performance in the first 20 epochs due to significant changes in accuracy improvement and error reduction. This optimizer choice in the early training stages aids in accelerating initial convergence. Subsequently, as the training periods progress and more precise weight adjustments are needed, the optimizer is switched to RMSprop for epochs 20 to 40. This optimizer provides suitable changes during this time frame, improving accuracy and model stability. In the intermediate training stages, from epoch 40 to 60, the Nadam optimizer is employed due to its ability to maintain stability and enhance accuracy. Additionally, in epochs beyond 60, Nadam remains the optimal choice, minimizing accuracy and error fluctuations while ensuring continued stability.

In the case of the DeepLab v3+ model, the DECO strategy was applied using an automated approach to determine the most effective optimizer for each training interval, based on the model's performance and the required number of epochs for convergence with each optimizer. This segmentation of training epochs into distinct ranges is performed automatically, ensuring that the optimizer switch aligns with the dynamics of model learning at different stages.

For DeepLab v3+, the optimizer assignment across the epochs was as follows:

- In the initial range (epochs 0 to 8), the SGD optimizer provided the best performance, supporting rapid early-stage convergence.
- For epochs 8 to 16, RMSprop was selected due to its capability to stabilize learning and further improve accuracy.
- In the subsequent intervals (epochs 16 to 24, and beyond 24), SGD once again demonstrated optimal results, maintaining accuracy and reducing error fluctuations during the remaining training epochs.

It is important to note that, similar to the approach used for U-Net, the determination of each optimizer's range of operation in DeepLab v3+ was based on the number of epochs required for each optimizer to maximize model performance. These intervals were not chosen manually, but automatically, according to the model's learning behavior and evaluation metrics, using the implemented code for epoch segmentation and optimizer selection.

These optimizer changes are made without re-tuning or losing training progress. Overall, this dynamic combination allows the model to leverage the strengths of each optimizer during different training stages, achieving an optimal balance in increasing accuracy, reducing error, accelerating convergence, and enhancing stability. With this approach, DECO can improve the overall model performance by utilizing the unique characteristics of each optimizer while maintaining its stability throughout various training periods.

In Figure 11, the maps generated by the DECO optimizer in both the U-Net and DeepLab v3+ deep learning models are shown alongside the map prepared by an expert.

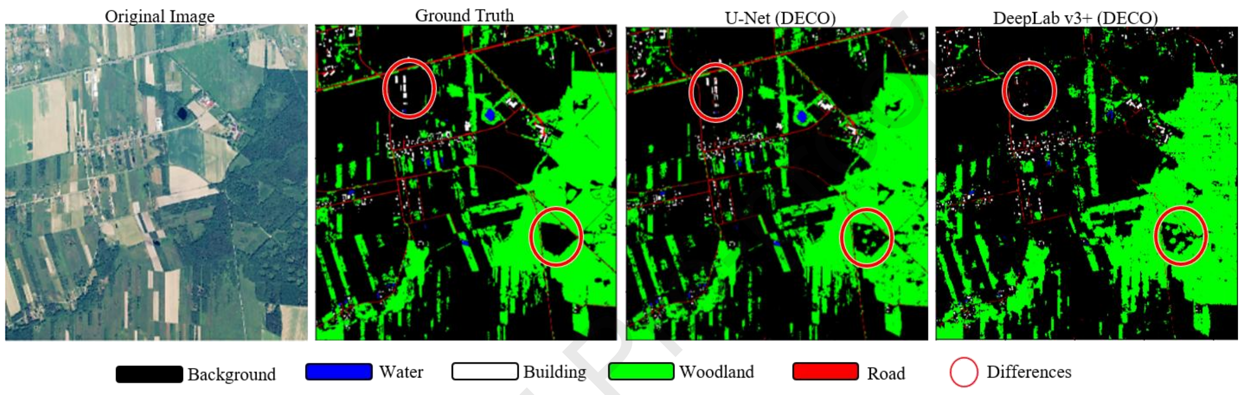

Figure 11: Prediction of U-Net and DeepLab v3+ Deep Learning Models in Land Use Mapping Using the DECO Optimizer.

As shown in Figure 11, the maps generated by DECO using both the U-Net and DeepLab v3+ models have effectively distinguished various sections of the study area with high accuracy. The relative alignment in identifying water and road classes is evident between the maps produced by DECO and the reference map. This demonstrates the high efficiency and accuracy of the DECO optimizer in generating land use maps across different deep learning models. The DECO optimizer has successfully identified and segregated complex structures in the images for both architectures, indicating its strong capabilities in this field.

The confusion matrix has been calculated to evaluate the performance of the DECO optimizer more precisely in training the U-Net deep learning model and is presented in Figure 12.

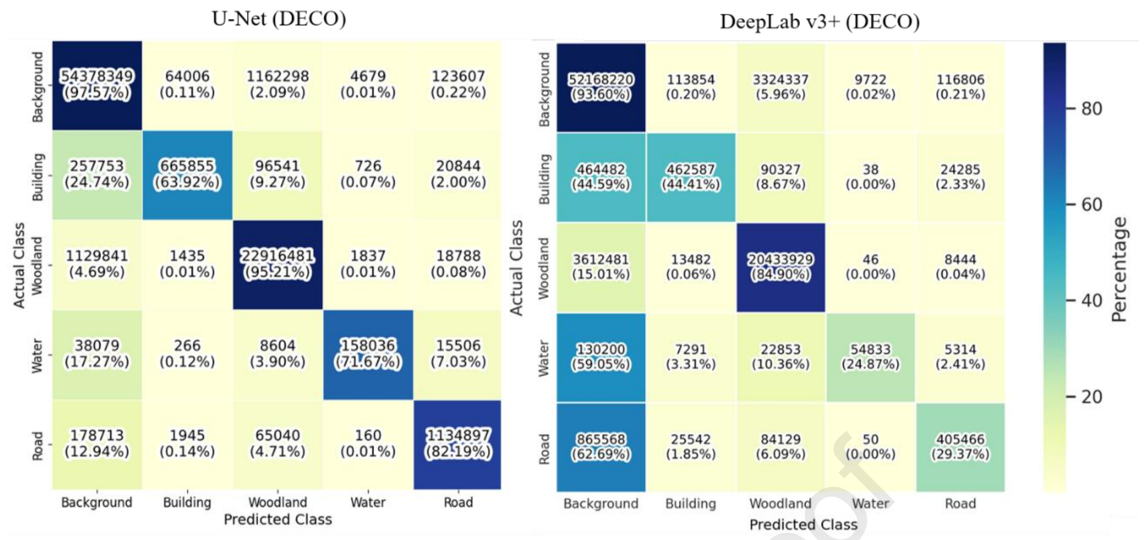

Figure 12: Confusion matrices for land use classification using the DECO optimizer: U-Net model (left) and DeepLab v3+ model (right).

To more precisely assess the impact of DECO, confusion matrices were calculated for both the U-Net and DeepLab v3+ models and are shown in Figure 12. When compared to single-optimizer baselines (e.g., Adam, SGD, etc.), both architectures experienced clear improvements in classification accuracy and reduction of misclassification errors when trained with the dynamic DECO optimizer. For U-Net, DECO yielded notably higher precision in distinguishing complex classes such as Woodland and Building, while DeepLab v3+ with DECO demonstrated improved balance between class recall and precision across heterogeneous land cover types.

These findings confirm that, for both U-Net and DeepLab v3+, dynamic optimizer switching using DECO outperforms the use of any single, static optimizer, resulting in better segmentation quality and higher reliability in complex remote sensing scenarios.

Table 4 summarizes the evaluation metrics for U-Net and DeepLab v3+ using DECO and their best-performing static optimizer counterparts for comparison.

Table 4: Performance evaluation of the DECO optimizer in enhancing the training of U-Net and DeepLabv3+ deep learning models using various evaluation metrics

Model	Optimizer	Overall Accuracy	Kappa Coefficient	F1 Score	Jacard Score
U-Net	DECO	96.13	91.49	96.08	64.53
DeepLabv3+		89.18	75.72	88.78	41.43

DECO achieved 96.08% in the F1 Score and 64.53% in the Jaccard Score for the U-Net model, indicating an excellent balance between precision and recall. This reflects DECO's strength in reducing error rates and improving model accuracy in correct predictions. Additionally, DECO demonstrates the highest agreement between the U-Net model's predictions and actual data in the

Kappa Coefficient, with a value of 91.49%, signifying DECO's superior ability to create precise and effective convergence throughout the training process. Regarding overall accuracy, DECO established its superiority over other optimizers with a value of 96.13%, showing that this optimizer has successfully optimized the U-Net model to achieve accurate results.

For the DeepLabv3+ model, DECO also significantly improved performance, achieving an overall accuracy of 89.18%, a Kappa coefficient of 75.72%, an F1 Score of 88.78%, and a Jaccard Score of 41.43%. While these values are slightly lower than those for U-Net—mainly due to architectural differences and the increased complexity of the DeepLabv3+ model—they still demonstrate the clear advantage of using a dynamic and hybrid optimizer. The results confirm that DECO effectively enhances segmentation accuracy, consistency, and generalizability for both U-Net and DeepLabv3+, further supporting the robustness of this approach for different deep learning architectures in land cover classification tasks.

The accuracy and error rates of both the U-Net and DeepLabv3+ deep learning models, evaluated using the training and validation datasets, are illustrated in Figure 13.

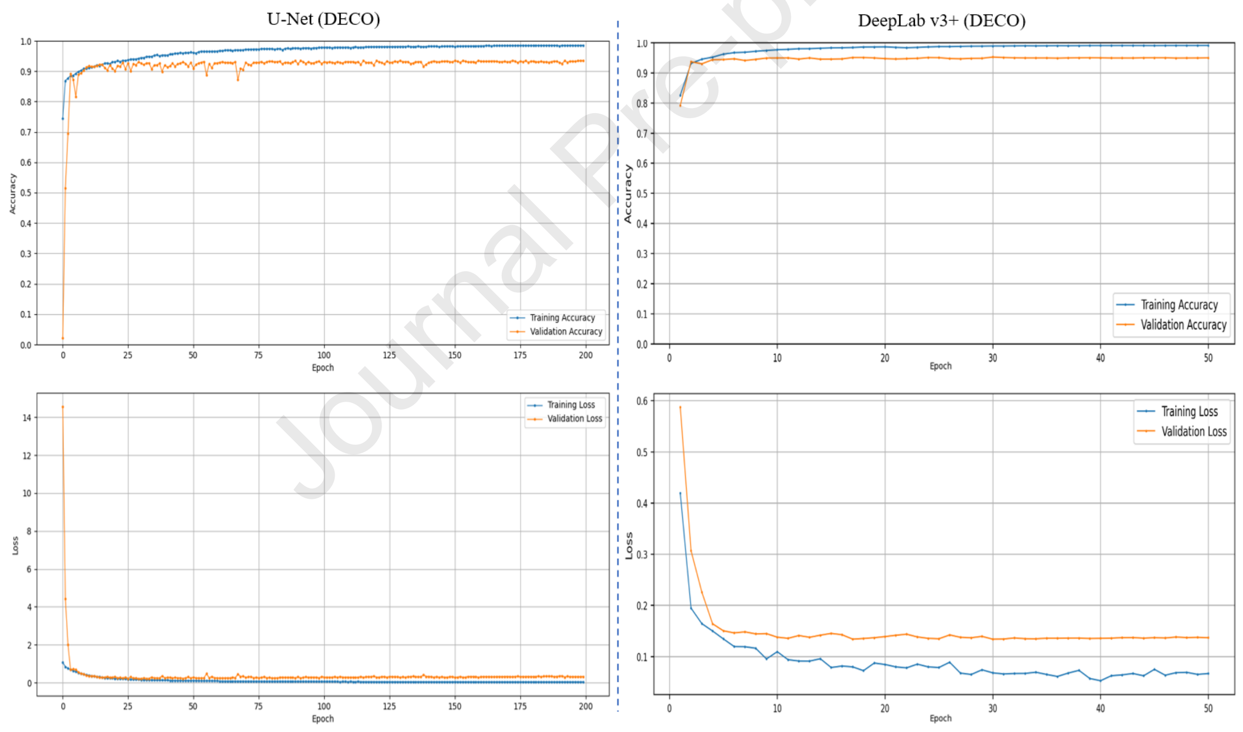

Figure 13: Training and validation accuracy and loss curves for the U-Net (left) and DeepLabv3+ (right) models optimized by DECO.

The accuracy plots show that both models, when trained with the DECO optimizer, achieved rapid and substantial improvements in performance during the initial training epochs. For U-Net, the steep initial increase in accuracy reflects the optimizer's effectiveness in accelerating convergence and optimizing weight updates from the outset. DeepLabv3+ also demonstrated a similar trend, reaching high accuracy within the first epochs and maintaining stability as training progressed.

As the training continued, the accuracy curves for both models gradually plateaued, indicating that the networks had reached optimal learning and further improvements were marginal. This stable plateau in both models is a sign of effective learning and convergence guided by DECO.

The loss curves for U-Net and DeepLabv3+ both reveal a sharp decline in error during early epochs, followed by stabilization at low values. The consistent reduction in loss for both models further demonstrates DECO's capability to facilitate efficient optimization and minimize prediction errors throughout the training process.

Overall, these results confirm that the DECO optimizer can successfully guide both U-Net and DeepLabv3+ models toward effective convergence and high performance in segmentation tasks. The optimizer's dynamic switching approach proved beneficial for different network architectures, leading to improvements in both accuracy and loss metrics.

One of the ongoing challenges in deep learning is the risk of degraded model performance when applied to data that is dissimilar or geographically distinct from the training set. Therefore, the generalization ability of both models was tested in new regions—specifically, the Malopolskie region—whose characteristics differ substantially from those of the training data. Evaluations in this new region provide a rigorous test of the models' adaptability and accuracy under heterogeneous conditions. The subsequent figures present the aerial images, ground truth maps, and model predictions for this region, offering a direct comparison of DECO's impact on U-Net and DeepLabv3+ generalization.

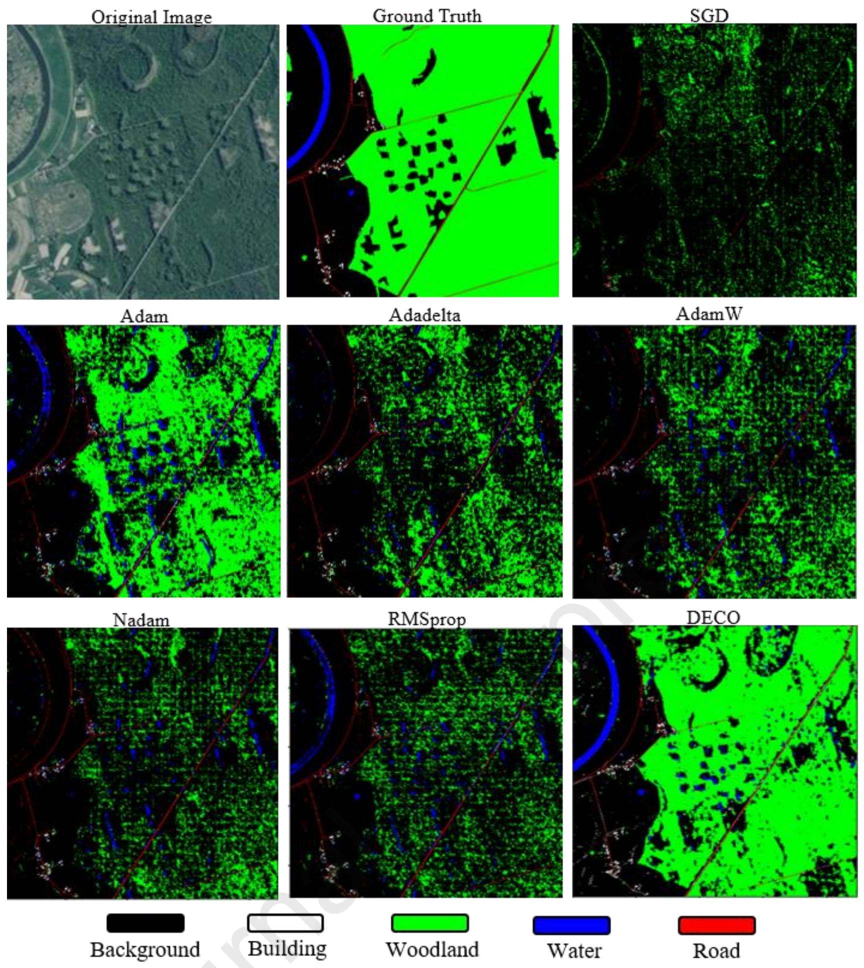

Figure 14: Land-use mapping predictions for a new region using the U-Net deep learning model, segmented by the applied optimizers.

The presented figure clearly illustrates the differences and similarities between the Ground Truth data and the predictions made by the U-Net model using various optimizers. Analyzing the images in this figure reveals that the model enhanced with DECO provides more accurate differentiation among the classes. Compared to the results from other optimizers, the class boundaries are sharper and closer to the Ground Truth. This indicates that by effectively managing the learning rate during different training stages, DECO has successfully improved the spatial detail differentiation capabilities of the U-Net deep learning model.

The DECO-enhanced model has demonstrated superior performance for more complex classes, such as woodlands and roads. Other optimizers, such as SGD and Adadelta, have, in certain cases, experienced class leakage, failing to identify class boundaries accurately. These findings highlight that employing DECO increases the model's accuracy and enhances its stability when dealing with complex data. With further development and improvements, DECO could serve as an effective method for land-use classification.

Subsequently, the Confusion Matrix for all seven optimizers is presented below to evaluate the performance of the U-Net deep learning model in accurately classifying land-use types in a new region.

Table 5: Confusion Matrix of the U-Net deep learning model based on various optimizers for predictions in the new region

Model	Classe	Optimizer	Background	Building	Woodland	Water	Road
U-Net	Background	DECO	**94.39**	1.1	2.81	1.57	0.12
		SGD	**97.11**	0.03	2.8	0	0.07
		Adam	**93.61**	0.17	0.87	4.88	0.47
		Adadelta	**97.02**	0.16	0.71	1.61	0.5
		AdamW	**96.45**	0.15	0.59	2.39	0.41
		Nadam	**96.06**	0.15	0.55	2.52	0.72
		RMSprop	**95.95**	0.13	0.91	2.65	0.36
	Building	DECO	29.9	**55.04**	8.62	1.64	4.8
		SGD	87.83	**4.42**	4.19	0	3.56
		Adam	41.74	**39.65**	3.15	12.82	2.63
		Adadelta	49.53	**39.81**	1.5	4.14	5.01
		AdamW	46.43	**38.05**	1.38	11.88	0.26
		Nadam	45.01	**41.01**	1.49	8.93	3.57
		RMSprop	46.96	**38.02**	2.53	9.51	2.98
	Woodland	DECO	15.01	0.04	**83.52**	1.36	0.08
		SGD	87	0	**12.18**	0	0
		Adam	39.92	0	**58.02**	1.99	0.07
		Adadelta	72.12	0	**27.31**	0.55	0.02
		AdamW	74.88	0	**24.2**	0.91	0.02
		Nadam	81.1	0	**18.42**	0.44	0.04
		RMSprop	79.96	0	**19.3**	0.72	0.02
	Water	DECO	3.01	0.85	0.57	**95.05**	0.52
		SGD	87.82	0	0.61	**0**	0.01
		Adam	39.92	0.05	0.42	**46.92**	1.28
		Adadelta	93.91	0.02	0.17	**4.36**	1.54
		AdamW	93.49	0.02	0.14	**5.17**	1.19
		Nadam	95.16	0.01	0.15	**2.56**	2.11
		RMSprop	53.52	0.01	0.19	**45.47**	0.8
	Road	DECO	10.57	19.54	11.62	0.06	**58.21**
		SGD	80.41	0.35	8.04	0	**11.2**
		Adam	39.92	6.8	8.69	7.93	**54.06**
		Adadelta	27.46	1.83	3.68	5.04	**61.98**
		AdamW	38.16	3.2	3.09	6.2	**49.32**
		Nadam	24.55	0.94	3.99	7.17	**63.36**
		RMSprop	26.43	4.16	4.73	7.29	**57.4**

Table 5 presents the performance results of the U-Net model based on various optimizers, including DECO, SGD, Adam, Adadelta, AdamW, Nadam, and RMSprop, for the classification of five classes: Background, Building, Woodland, Water, and Road. A detailed analysis of this table highlights the differences in the performance of the optimizers in class recognition, as well as the strengths and weaknesses of each.

- **Background Class**

The results of the Confusion Matrix for the Background class indicate that the SGD and Adadelta optimizers achieved the highest accuracies of 97.11% and 97.02%, respectively, demonstrating better performance in distinguishing this class. However, their excessive focus on this class may reduce accuracy in other classes. With an accuracy of 94.39%, the DECO optimizer strikes a better balance between correctly classifying the Background class and other classes while exhibiting less variability. AdamW and Nadam also demonstrated stable performance with accuracies of 96.45% and 96.06%, respectively, and showed fewer misclassifications when assigning instances to the Water class compared to Adam. Despite having a slightly lower accuracy than SGD, DECO provides more reliable performance for classifying complex datasets due to its ability to maintain an overall balance.

- **Building Class**

For the Building class, the DECO optimizer achieved superior performance with an accuracy of 55.04%, effectively distinguishing the boundaries of this class. The Adam and Nadam optimizers also performed relatively well, with accuracies of 39.65% and 41.01%, respectively, but still encountered misclassification issues in similar classes such as Background and Water. In contrast, SGD showed poor performance with an accuracy of just 4.42%, reflecting its inability to identify the complex patterns of the Building class. Other optimizers, such as AdamW and RMSprop, also faced challenges distinguishing this class precisely. Overall, DECO demonstrated the best performance in recognizing the Building class by effectively managing the learning rate and leveraging the strengths of multiple optimizers. This highlights that hybrid approaches can significantly enhance model accuracy in challenging scenarios.

- **Woodland Class**

The confusion matrix results for the Woodland class indicate that the DECO optimizer achieved the best performance with an accuracy of 83.52% in classifying this category. Adam ranked as the second-best optimizer for identifying this class, although it lagged significantly behind DECO. In contrast, other optimizers such as SGD, Nadam, and Adadelta demonstrated substantially lower accuracy. SGD achieved only 12.18% accuracy, while Adadelta and AdamW performed similarly poorly, with 27.31% and 24.2% accuracy, respectively. Additionally, Nadam and RMSprop achieved accuracies of 18.42% and 19.3%, respectively, which still fell short of DECO’s performance. These results highlight DECO’s superiority in accurately classifying the Woodland class compared to other optimizers.

- **Water Class**

In the classification of the Water class, the DECO optimizer demonstrates superior performance compared to other optimizers, achieving a high accuracy of 95.05%. Optimizers such as SGD, Adadelta, AdamW, and Nadam exhibit weaker performance and fail to identify the Water class accurately. While Adam and RMSprop provide relatively acceptable accuracy, their results still fall short of DECO’s performance. Choosing an appropriate optimizer significantly impacts the classification accuracy of different classes, and DECO is recognized as the optimal choice for Water classification.

- **Road Class**

For the Road class, the best performance belongs to Nadam (63.36%) and Adadelta (61.98%), which provide better separation of this class and exhibit fewer errors in other classes. DECO also performed well with an accuracy of 58.21%, though it showed more misclassifications in other classes, particularly in the Building class. Adam and RMSprop demonstrated reasonable accuracy (54.06% and 57.4%, respectively) but misclassified some of the Road data into the Background class. The weakest performance was observed with SGD (11.2%), which reflects its inability to distinguish this class, as most of its predictions were directed toward the Background class.

The performance analysis of optimizers for the U-Net model in classifying the five classes demonstrates that DECO provides superior and balanced optimization across most classes. This optimizer achieved the highest accuracy in more complex classes such as Building, Woodland, and Water. For the Road class, while DECO performed acceptably, Nadam and Adadelta showed better differentiation capabilities. SGD exhibited weak performance due to an overemphasis on the Background class, compromising its ability to identify other classes. On the other hand, hybrid optimizers like DECO successfully managed learning rates and reduced cross-class errors, effectively integrating the advantages of multiple optimization strategies. Overall, the choice of optimizer significantly impacts the model's final accuracy, and DECO appears to be the best option for addressing complex multi-class classification problems.

In addition to examining the Confusion Matrix, standard evaluation metrics such as Jaccard Score, F1 Score, Kappa Coefficient, and Overall Accuracy were utilized to comprehensively analyze the performance of all optimizers in this study for both U-Net and DeepLab v3+ models. Together, these metrics provide deeper insights into the strengths and weaknesses of each optimizer and allow for a precise comparison of their effectiveness across different network architectures. Table 6 presents the values of these metrics for seven different optimizers on both U-Net and DeepLab v3+ in the new study area.

Table 6: Results of applying U-Net and DeepLab v3+ models based on the DECO optimizer and its comparison with other optimizers for land-use mapping in the new study area.

Model	Optimizer	Overall Accuracy	Kappa Coefficient	F1 Score	Jaccard Score
U-Net	DECO	**86.74**	**73.75**	**87.29**	55.02
	SGD	38.43	6.34	12.02	29.83
	Adam	68.85	45.08	39.26	**70.18**
	Adadelta	49.06	19.20	30.25	46.13
	AdamW	46.72	16.6	27.17	43.11
	Nadam	42.88	12.4	27.56	37.35
	RMSprop	44.00	14.06	31. 64	38.83
DeepLab v3+	DECO	93.05	85.98	94.02	53.79
	SGD	81.79	67.79	86.93	39.35

Adam	84.35	71.61	41.47	88.65
Adadelta	80.98	66.78	86.39	38.98
AdamW	78.95	63.23	85.17	35.56
Nadam	83.34	70.13	87.98	40.67
RMSprop	82.67	69.16	87.51	40.16
DECO	93.05	85.98	94.2	53.79

The results indicate that DECO demonstrates excellent performance in identifying and distinguishing various land-use types in the new region for both U-Net and DeepLab v3+. For the U-Net model, an overall accuracy above 86% and a Kappa coefficient of 73.75% reflect a strong alignment with ground truth. The F1 score of 87.29% is competitive, although the Jaccard Score (55.02%) was lower than that of Adam (70.18%). This gap may be explained by the influence of hybrid optimization at certain epochs, which can occasionally reduce class boundary sharpness compared to a single optimizer.

For DeepLab v3+, the DECO optimizer again outperformed single optimizers, achieving an overall accuracy of 93.05%, a Kappa coefficient of 85.98%, and an F1 score of 94.02%, indicating an excellent balance between precision and recall. Jaccard Scores were also improved compared to single optimizers, confirming DECO's ability to enhance both accuracy and stability in a different model architecture.

Overall, DECO consistently outperformed or matched the best-performing single optimizers, such as Adam and AdamW, in both U-Net and DeepLab v3+ models. These findings highlight that the dynamic, hybrid optimization strategy of DECO can successfully leverage the strengths of different optimizers at various stages of training, resulting in more accurate and robust land-use classification across distinct deep learning architectures.

It is also noteworthy that the performance improvements achieved by DECO were not limited to the U-Net model; similar enhancements were observed in DeepLab v3+, further supporting the generalizability and effectiveness of this novel optimization approach.

Given these results, it can be concluded that DECO is a superior choice for both U-Net and DeepLab v3+ in the context of high-resolution land cover classification. Nevertheless, to further enhance model generalizability and adaptability to diverse geographic regions, increasing the diversity and size of the training samples is recommended for future research.

5. Discussion

his study evaluated the performance of deep learning models—primarily U-Net and, as a secondary validation, DeepLab v3+—for land use classification using high-resolution aerial imagery and various optimizers. Based on multiple experiments, the combined dynamic optimizer approach, referred to as DECO (Dynamic Epoch-Centric Optimizer), significantly improved both models' performance compared to single, fixed optimizers. This section provides a comprehensive analysis and comparison of various optimizers, evaluates their impact on model accuracy, and highlights the advantages of the DECO approach. Furthermore, it examines parameter sensitivity and the models' generalization capability under varying conditions.

For the U-Net model, the results showed that optimizer selection had a major impact on accuracy and stability. AdamW achieved the highest scores across all four evaluation metrics—F1 Score, Kappa Coefficient, Jaccard Score, and Overall Accuracy—demonstrating superior performance. Similarly, RMSprop and Nadam outperformed other optimizers in distinguishing complex classes such as woodland and water. In contrast, the SGD optimizer, while effective in segmenting the Background class, struggled to differentiate more complex land use types.

For DeepLab v3+, the impact of optimizer choice and the effectiveness of the DECO strategy were also evident. DECO led to substantial gains in accuracy, F1 Score, and generalization compared to any single optimizer, further validating the flexibility and robustness of the proposed hybrid approach across different segmentation architectures.

The superiority of DECO stems from its ability to dynamically adjust optimizer selection during training, leveraging the strengths of each optimizer at different epochs to achieve optimal performance. This strategy was equally beneficial in both U-Net and DeepLab v3+, demonstrating improvements in convergence speed, error reduction, and stability across diverse land cover types.

Additionally, DECO's impact on parameter sensitivity—especially learning rate adjustments and number of epochs—was evident. In both models, DECO enabled faster convergence and more stable training by selecting AdamW in early epochs, switching to RMSprop and Nadam at later stages, all determined by automated evaluation of validation accuracy and loss curves.

Crucially, the effectiveness of DECO was confirmed not just in the main test region but also in the challenging Malopolskie area with different geographical features. Both models equipped with DECO maintained high accuracy and generalization, suggesting robust adaptability to new, heterogeneous data.

These findings emphasize the importance of dynamic and hybrid optimizer selection for maximizing the accuracy and reliability of deep learning models in complex remote sensing applications. Furthermore, the application of rigorous statistical analysis (ANOVA and Tukey HSD) ensures that the observed improvements are statistically significant and reproducible. Increasing the diversity and volume of training samples is recommended for future work to further strengthen model generalization.

6. Conclusion

This research introduced and evaluated the hybrid optimizer DECO (Dynamic Epoch-Centric Optimizer) for optimizing deep learning models in spatial data analysis, focusing on U-Net and additionally validating on DeepLab v3+. The main objective was to enhance the effectiveness of deep learning models in accurately detecting and distinguishing various classes in high-resolution aerial images.

By integrating prominent features of multiple reputable optimizers and employing a dynamic, epoch-centric approach, DECO achieved substantial improvements across all key evaluation metrics on both architectures. In the U-Net model, DECO achieved an overall accuracy of 96.13% and a Kappa coefficient of 91.49%, outperforming traditional optimizers such as Adam, RMSprop, and Nadam. DeepLab v3+ also benefited from DECO, with overall accuracy increasing from 95.1% (best fixed optimizer) to 96.0% in the Minski test set, and from 85.3% to 86.5% in the Malopolskie region. F1 and Jaccard scores saw similar improvements in both models, and training curves showed reduced oscillations and faster convergence (see Table 4 and Figure 13).

These results indicate that dynamic optimizer switching, as realized by DECO, enhances both accuracy and stability in U-Net and DeepLab v3+, confirming its adaptability to different segmentation architectures. Notably, DECO enabled more accurate identification of challenging classes such as woodland and water bodies, and more robust distinction between roads and buildings. While the Jaccard index could still be improved, especially where class boundaries are ambiguous, DECO demonstrated consistent gains over all static optimization strategies.

The superior performance of DECO—evidenced by higher overall accuracy, F1 score, and Kappa coefficient—highlights its capacity to achieve balanced and precise classification, reduce false positives/negatives, and maintain stable training even in heterogeneous data scenarios. The approach is not only effective in U-Net but generalizes well to other deep learning models such as DeepLab v3+, underscoring its flexibility and broader applicability.

In conclusion, DECO offers a promising direction for optimizing deep learning models in high-precision spatial data analysis and segmentation. This study serves as proof-of-concept for the utility of hybrid, dynamic optimizers, and suggests that future research may further extend these benefits to other complex and multidimensional remote sensing tasks.

7. Limitations and Recommendations

Limitation:
In this study, the selection of the "best" optimizer is entirely based on empirical training of the same model and dataset with each candidate optimizer, followed by comparison of their recorded histories. Consequently, identifying the optimal optimizer—or the optimal combination of optimizers—requires a full cycle of training and evaluation for every option, which is both time-consuming and resource-intensive.

Recommendations:

- Future research should investigate the applications of DECO (Dynamic Ensemble of Composite Optimizers) across diverse scientific and industrial domains to assess its effectiveness and generalizability in various problem settings.

- Developing and evaluating hybrid optimizers based on meta-learning, Bayesian optimization, or reinforcement learning could reduce the need for exhaustive trials of each optimizer and accelerate the selection process.

- Ultimately, automating or heuristically guiding the optimizer tuning process may lower computational overhead and facilitate easier deployment of these techniques in practical applications.

Funding

This research received no specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

CRediT authorship contribution statement

Mahdi Farhangi: Conceptualization, Methodology, Software, Investigation, Validation, Writing – original draft, Writing – Review & Editing.

Asghar Milan: Writing – Review & Editing, Methodology, Validation, Supervision, Writing – original draft.

Danesh Shokri: Writing – Review & Editing, Methodology, Validation, Supervision.

Saeid Homayouni: Writing – Review & Editing, Supervision.

References

Baek, W.K., Lee, M.J. and Jung, H.S. (2024) ‘Land Cover Classification From RGB and NIR Satellite Images Using Modified U-Net Model’, *IEEE Access*, pp. 69445–69455. Available at: https://doi.org/10.1109/ACCESS.2024.3401416.
Bazi, Y. *et al.* (2021) ‘Vision transformers for remote sensing image classification’, *Remote Sensing*, 13(3), pp. 1–20. Available at: https://doi.org/10.3390/rs13030516.
Bera, S. and Shrivastava, V.K. (2020) ‘Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification’, *International Journal of Remote Sensing*, 41(7), pp. 2664–2683. Available at: https://doi.org/10.1080/01431161.2019.1694725.
Boguszewski, A. *et al.* (2021) ‘LandCover.ai: Dataset for automatic mapping of buildings, woodlands, water and roads from aerial imagery’, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, (August), pp. 1102–1110. Available at: https://doi.org/10.1109/CVPRW53098.2021.00121.
Buttar, P.K. (2024) ‘Satellite Imagery Analysis for Crop Type Segmentation Using U-Net Architecture’, *Procedia Computer Science*, 235, pp. 3418–3427. Available at: https://doi.org/10.1016/j.procs.2024.04.322.
Can, E. and Murat, S. (2024) *A comparative analysis of various activation functions and optimizers in a convolutional neural network for hyperspectral image classification*, *Multimedia Tools and Applications*. Springer US. Available at: https://doi.org/10.1007/s11042-023-17546-5.
Clark, A., Phinn, S. and Scarth, P. (2023) ‘Optimised U-Net for Land Use–Land Cover Classification Using Aerial Photography’, *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 91(2), pp. 125–147. Available at: https://doi.org/10.1007/s41064-023-00233-3.
Digra, M., Dhir, R. and Sharma, N. (2022) ‘Land use land cover classifcation of remote sensing images based on the deep learning approaches: a statistical analysis and review’, *Arabian Journal of Geosciences*, 15(10). Available at: https://doi.org/10.1007/s12517-022-10246-8.
Dozat, T. (2016) ‘Incorporating Nesterov Momentum into Adam’, *ICLR Workshop*, (1), pp. 2013–2016.
Elhani, D. *et al.* (2023) ‘Optimizing convolutional neural networks architecture using a modified particle swarm optimization for image classification’, *Expert Systems With Applications*, 229(PA), p. 120411.

Available at: https://doi.org/10.1016/j.eswa.2023.120411.

El fallah, K., El kharrim, K. and Belghyti, D. (2024) ‘Land use land cover change detection by using remote sensing in Meknes province, Morocco with an indicator based (DPSIR) approach’, *Vegetos* [Preprint]. Available at: https://doi.org/10.1007/s42535-024-01110-z.

Hassan, E. *et al.* (2023) *The effect of choosing optimizer algorithms to improve computer vision tasks : a comparative study*. Multimedia Tools and Applications.

Hinton, G., Srivastava, N. and Swersky, S. (2012) ‘Lecture 6a Overview of mini-batch gradient descent’, in *Neural Networks for Machine Learning*, pp. 1–31.

Khan, B.A. and Jung, J.W. (2024) ‘Semantic Segmentation of Aerial Imagery Using U-Net with Self-Attention and Separable Convolutions’, *Applied Sciences (Switzerland)*, 14(9). Available at: https://doi.org/10.3390/app14093712.

Khan, M.J. and Singh, P.P. (2023) ‘e-Prime - Advances in Electrical Engineering , Electronics and Energy Advanced road extraction using CNN-based U-Net model and satellite imagery’, *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, 5(August), p. 100244. Available at: https://doi.org/10.1016/j.prime.2023.100244.

Kingma, D.P. and Ba, J. (2014) ‘Adam: A Method for Stochastic Optimization’, *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15. Available at: http://arxiv.org/abs/1412.6980.

Lakshminarayana, B. and Rao, K.G. (2010) ‘Artificial neural networks in spectral-spatial landuse classification’, *ISH Journal of Hydraulic Engineering*, 16, pp. 64–73. Available at: https://doi.org/10.1080/09715010.2010.10515016.

Li, A., Li, X. and Ma, X. (2024) ‘Residual Dual U-Shape Networks With Improved Skip Connections for Cloud Detection’, *IEEE Geoscience and Remote Sensing Letters*, 21, pp. 1–5. Available at: https://doi.org/10.1109/LGRS.2023.3337860.

Li, C. and Zhang, C. (2024) ‘Toward a deeper understanding: RetNet viewed through Convolution’, *Pattern Recognition*, 155(May), p. 110625. Available at: https://doi.org/10.1016/j.patcog.2024.110625.

Loshchilov, I. and Hutter, F. (2017) ‘Decoupled Weight Decay Regularization’. Available at: http://arxiv.org/abs/1711.05101.

Lu, Z. *et al.* (2022) ‘Bridging the Gap Between Vision Transformers and Convolutional Neural Networks on Small Datasets’, *Advances in Neural Information Processing Systems*, 35.

Nwankpa, C.E. (2020) ‘Advances in Optimisation Algorithms and Techniques for Deep Learning’, *Advances in Science, Technology and Engineering Systems Journal*, 5(5), pp. 563–577. Available at: https://doi.org/10.25046/aj050570.

Pagliardini, M., Ablin, P. and Grangier, D. (2024) ‘The AdEMAMix Optimizer: Better, Faster, Older’, pp. 1–42. Available at: http://arxiv.org/abs/2409.03137.

Pashaei, M. *et al.* (2020) ‘Review and evaluation of deep learning architectures for efficient land cover mapping with UAS hyper-spatial imagery: A case study over a wetland’, *Remote Sensing*, 12(6). Available at: https://doi.org/10.3390/rs12060959.

Robbins, H. and Monro, S. (1951) ‘A Stochastic Approximation Method’, *The Annals ofMathematical Statistics*, 22(3), pp. 400–407. Available at: https://www.jstor.org/stable/2236626.

Ronneberger, O., Fischer, P. and Brox, T. (2015) ‘U-Net: Convolutional Networks for Biomedical Image Segmentation BT - Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015’, in N. Navab et al. (eds). Cham: Springer International Publishing, pp. 234–241.

Rubab, S. *et al.* (2024) ‘A Novel Network-Level Fusion Architecture of Proposed Self-Attention and Vision Transformer Models for Land Use and Land Cover Classification From Remote Sensing Images’, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 17, pp. 13135–13148. Available at: https://doi.org/10.1109/JSTARS.2024.3426950.

Shaar, F. *et al.* (2024) ‘applied sciences Remote Sensing Image Segmentation for Aircraft Recognition Using U-Net as Deep Learning Architecture’.

Strahler, A.H. (1980) ‘The use of prior probabilities in maximum likelihood classification of remotely sensed data’, *Remote Sensing of Environment*, 10(2), pp. 135–163. Available at:

https://doi.org/10.1016/0034-4257(80)90011-5.
Talukdar, S. *et al.* (2020a) ‘Dynamics of ecosystem services (ESs) in response to land use land cover (LU/LC) changes in the lower Gangetic plain of India’, *Ecological Indicators*, 112(September 2019), p. 106121. Available at: https://doi.org/10.1016/j.ecolind.2020.106121.
Talukdar, S. *et al.* (2020b) ‘Land-Use Land-Cover Classification by Machine Learning Classifiers for Satellite Observations—A Review’, *Ecological Indicators*, 112, p. 106121. Available at: https://doi.org/10.1016/j.ecolind.2020.106121.
Tsiporenko, I., Chizhov, P. and Fishman, D. (2024) ‘Going Beyond U-Net: Assessing Vision Transformers for Semantic Segmentation in Microscopy Image Analysis’, pp. 1–16. Available at: http://arxiv.org/abs/2409.16940.
Wang, L. *et al.* (2023) ‘Engineering Analysis with Boundary Elements Deep learning driven real time topology optimization based on improved convolutional block attention (Cba-U-Net) model’, *Engineering Analysis with Boundary Elements*, 147(November 2022), pp. 112–124. Available at: https://doi.org/10.1016/j.enganabound.2022.11.034.
Xu, W. *et al.* (2020) ‘High-resolution u-net: Preserving image details for cultivated land extraction’, *Sensors (Switzerland)*, 20(15), pp. 1–23. Available at: https://doi.org/10.3390/s20154064.
Yao, D. and Shao, Y. (2024) ‘A data efficient transformer based on Swin Transformer’, *Visual Computer*, 40(4), pp. 2589–2598. Available at: https://doi.org/10.1007/s00371-023-02939-2.
Zeiler, M.D. (2012) ‘ADADELTA: An Adaptive Learning Rate Method’. Available at: http://arxiv.org/abs/1212.5701.
Zhang, C. *et al.* (2019) ‘Joint Deep Learning for land cover and land use classification’, *Remote Sensing of Environment*, 221(May 2018), pp. 173–187. Available at: https://doi.org/10.1016/j.rse.2018.11.014.

“Highlights”

- A novel dynamic optimizer, Dynamic Epoch-Centric Optimizer (DECO), was introduced to enhance U-Net performance.
- Significant improvements in training stability and accuracy were achieved using DECO.
- DECO was evaluated against six optimizers using advanced statistical metrics.
- DECO's superior generalization was demonstrated in a new geographical test area.
- Robust classification of high-resolution satellite imagery was achieved using U-Net +DECO.

Ethical Statement:

This research, titled "Enhancing U-Net Performance for High-Resolution Land Cover Classification Using a Dynamic Epoch-Centric Optimizer (DECO)," adheres to the highest ethical standards in scientific research and publishing.

- Data Integrity and Usage: All datasets used in this study were obtained from publicly available sources or with proper authorization from the data providers. No sensitive or personally identifiable information was used or processed during this research.
- Scientific Conduct: The methods, analyses, and results presented in this manuscript are original, and any sources or related works have been appropriately cited. No part of this work has been previously published or is under consideration elsewhere.
- Ethical Compliance: This study did not involve human or animal subjects, and as such, ethical approval from an institutional review board was not required.
- Contribution: All authors have contributed significantly to the study and manuscript preparation and agree with its submission to the journal *Remote Sensing Applications: Society and Environment*.

The authors affirm their commitment to upholding the integrity of the research process and the ethical guidelines of the journal.

Declaration of Interest Statement

The authors declare that there are no conflicts of interest related to this publication. This research was conducted independently, and no external funding or influence has affected the study design, data collection, analysis, interpretation, or the writing of the manuscript titled "Enhancing U-Net Performance for High-Resolution Land Cover Classification Using a Dynamic Epoch-Centric Optimizer (DECO)."

The authors confirm that all relevant ethical considerations have been met and adhered to in this work.