Analyzing the Impact of Optimization Techniques on U-Net-Based Building Detection Performance

Suheyla Piltan Altaş, Fevzi Karsli

Karadeniz Technical University,Turkiye

Abstract

As a result of the growth of cities and the increase in migration from rural to urban areas, the urban population has grown remarkably. Building extraction is important in many practical and strategic areas. Automatic detection of buildings from images is important in terms of both speed and preventing interpretation errors arising from the differences in experience of experts. One of the main difficulties encountered in studies on automatic building detection is the generalization problem originating from the difference in the characteristics of roofs in complex environments. Recently, software and hardware systems have gained great importance due to the development of new technologies. As a result of these innovations, studies on deep learning architecture have increased. The training of deep learning architectures aims to minimize the loss function during learning. There are many optimization algorithms based on various mathematical principles; however, an optimization algorithm that can be generalized to all problems and is optimal for all conditions is still not fully defined. Therefore, the studies in literature continue to be experimental. In this study, the effects of optimization techniques on the automatic detection of buildings with different roof types from aerial images using the U-Net architecture are analyzed. In this study, Adam, Nadam, and RMSprop optimization techniques were used. The effects of optimization techniques on classification performance were investigated by examining computational costs and performance metrics.

Keywords: Building Detection, Adam, Nadam, RMSProp, U-Net

1. Introduction

Migration from villages to cities has caused population density in urban areas, and the number of buildings has increased. Building extraction from remotely sensed images is important in many fields, such as population monitoring, urban planning, and geographic information systems (GIS) (Erener, 2013).

In recent years, the use of remotely sensed images has become widespread with rapidly developing sensors, hardware and software technologies. These images provide temporal, spatial, and spectral information about the earth. With the diversity of sensor types and the development of high-resolution cameras, it is possible to obtain detailed images from aerial images.

Building types have different structures and geometric shapes according to the characteristics of the geography where they are located. Therefore, building extraction brings various difficulties (Ji et al., 2018). At the same time, when the building roofs and the ground have similar reflectance values, it sometimes leads to background complexity. Identifying the boundaries of densely grouped, partially contiguous small buildings is a more challenging problem than sparse, scattered, and large buildings (Wen et al., 2019).

Many algorithms have been developed and used for the extraction of building details from remotely sensed images. Although the traditionally preferred algorithms are simple, easy to implement, and require little data, they have some disadvantages due to their low success on variable data, their lack of generalizability and the need for partial human intervention.

Human beings have been in search of developing systems that think, analyze, learn, and decide like themselves since the beginning of their existence. This aim, which was identified with the concept of ‘artificial intelligence’ in the past, has led to the development of the concepts of ‘machine learning’ and ‘deep learning’ with the developing technology. With deep learning algorithms, which are widely used, especially in the field of image processing, it has become widespread to use models that are fast, automatic, generalizable, highly accurate, and able to easily process large data and require minimum human intervention.

The aim of the deep learning network training is to ensure that the prediction converges to ground truth as much as possible. The difference between the predicted and ground truth is expressed as a loss function, and this difference is minimized in an ideal network. Model performance improves with the learning performed throughout training, and the learning process increases until the model is the best representation of the ground truth (Chollet, 2021).

The optimization technique directly affects the training time and model performance. An optimization technique goes through several iterations to improve the accuracy of the model and approximate the true presentation of the data (Zaheer and Shaziya, 2019). To date, there is no theoretical background that provides a clear framework on how to choose and implement an optimization technique suitable for the model (Choi et al., 2019). Therefore, the studies in relevant research and literature are based on empirical research, assumptions, and comparisons.

Deep learning networks are optimized in several ways, such as structural optimization of the network model in the training phase, determining the parameters of the defined network structure, applying pre-processing steps to the datasets, and selecting the best optimization technique (Reyad et al., 2023).

Jawahar et al. (2023) detected the damage of Hurricane Harvey from satellite images in their study. The researchers used Adam, RMSprop, and SGD optimizers to train the model and emphasized that the Adam optimizer outperformed the other optimizers with an accuracy score of 98.57%. For the same model, the RMSprop algorithm gave 98.48% and SGD gave 98.32% accuracy scores.

Bouanane et al. (2024) used U-Net and Dense U-Net architectures and examined the effects of Momentum GD, NAG, AdaGrad, RMSprop, and Adam optimizers on the change analysis. Emphasizing that model performance in a deep learning network is directly correlated with task, dataset, and architecture, the authors stated that optimization algorithms also significantly affect model performance. They reported that Adam and RMSprop optimization algorithms gave good results in terms of performance metrics, but the Adam optimizer was more stable at low learning rates.

Shahrabadi et al. (2023) analyzed deep learning architectures used for bridge defect detection with different optimizers and learning rates. They analyzed Adam, Nadam, RMSprop, and SGD algorithms and reported the Adam algorithm provides a stable learning period, while the Nadam and RMSprop optimizers stand out with high performance.

Anagün et al. (2021) compared the effectiveness of SGD, RMSprop, Adam, Adagrad, Adamax, and Nadam optimizers for updating CNN model weights. They found that Adam and Nadam optimizers are more stable than other optimizers, while Adamax is faster in updating the model weight in the backpropagation phase.

Taffese et al. (2025) used YOLO v8 architecture to detect cracks in concrete structures in their study. They compared SGD, AdamW, Adam, NAdam, RAdam and RMSprop optimizers and found that the YOLO V8 model with SGD optimizer detects cracks with outstanding accuracy and speed.

González et al. (2025) performed a YOLO v8-based analysis on images from the Maxar GeoEye-1 satellite of the Hurricane Maria disaster. The researchers investigated momentum SGD, RMSprop, Adam, Adamax, NAdam, and AdamW optimizers. As a result, they concluded that the reliability of the momentum SGD is better than the Adam optimizer in several factors (training robustness, fast approximation, and consistent prediction generation).

In this study, the impact of training the U-Net architecture, which is widely used in building extraction from images, with different optimization techniques on model performance is analyzed. In this context, the U-Net architecture was trained on an open dataset, the "Massachusetts Buildings Dataset" (Mnih, 2013), with Adam, Nadam, and RMSprop optimization techniques based on gradient descent, which are widely used in the literature, and the effects of the techniques on classification were examined with performance metrics.

2. Methods

2.1 U-Net Architecture

U-Net is a convolutional neural network-based algorithm. Announced in 2015 in "U-Net: Convolutional Networks for Biomedical Image Segmentation" (Ronneberger et al., 2015), the model is designed for segmenting biomedical images requiring intensive analysis and pixel-level segmentation.

Convolutional neural networks are composed of three basic layers: input layer, convolution layer, pooling layer and fully connected layer. These networks are built on a special mathematical process called convolution (Goodfellow et al., 2016).

Each step in the network learns a pattern. Once learned, the pattern can be recognized again. This allows the network to continue the process by building on what was learned in the previous layer in each convolutional layer without the need for relearning. Thus, the learning process in the training phase will be simplified from specific to general induction (Chollet, 2021).

In a convolutional neural network, the dimensions of the feature maps are reduced in the pooling layer. Then the feature map is extracted. In the U-Net model, the dimension reduction is followed by a dimension increase. Thus, the feature maps are restored to their original resolution. In this respect, it is similar to an encoder-decoder network. The name of the algorithm comes from the shape of the architecture (Figure 1).

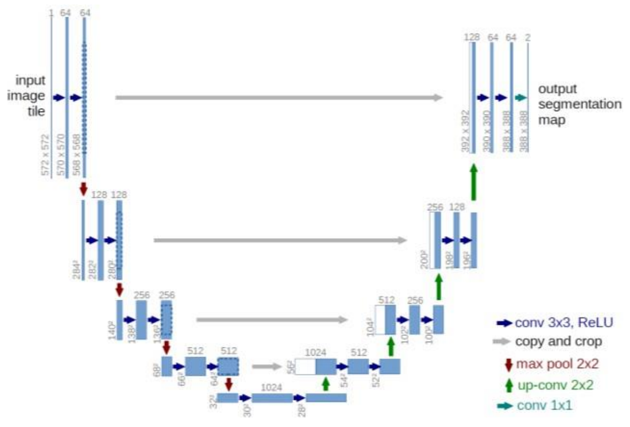

Figure 1. U-Net architecture (Ronneberger vd., 2015)

2.2 Optimization Algorithms: Adam, Nadam and RMSprop

The impact of an optimization algorithm on the model depends on the problem, data quality, data characteristics, and model architecture. In some cases, the techniques used can imitate each other. The lack of a theoretical description of the impact of the techniques on model performance has led the studies on these techniques to rely on empirical bases and comparative analysis. In addition, hyperparameters are crucial for model performance. (Choi et al., 2019).

In this study, the effects of Adam (Kingma and Ba, 2014), Nadam (Dozat, 2016), and RMSprop (Tieleman and Hinton, 2012) optimization techniques on the model performance of the U-Net architecture are analyzed.

The RMSprop optimization technique uses adaptive learning rates. It can use a variable learning rate according to the parameters in the optimization process. During the step sizes are iteratively changed according to the gradient magnitudes during the update. Thus, the learning rate for each parameter can be adaptively determined, and this technique is often preferred, especially in memory-limited situations.

Another well-known optimization technique, Adam uses momentum and the gradient square to update the model weights. It is an efficient technique with low memory requirements. The directional changes of the weights can be adjusted by determining both the direction and the size of the steps.

The Nadam optimization technique emerged by integrating Nesterov momentum into the Adam technique. Thus, the convergence speed is increased, and high performance is achieved (Ruder, 2016).

2.3 Dataset

The dataset used in the study is the Massachusetts Buildings Dataset, which consists of 151 aerial images of the Boston area (Mnih, 2013). The publicly available dataset consists of 137 training, 10 test, and 4 validation images. Each image is 1500×1500 pixels for an area of 2.25 km² (Figure 2).

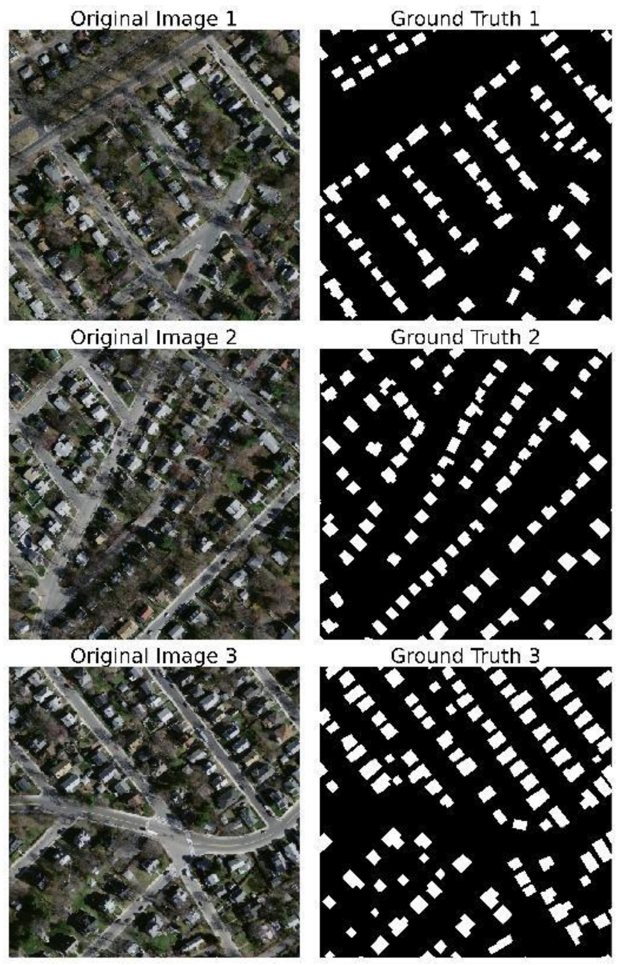

Figure 2. Massachusetts buildings dataset examples (Mnih, 2013)

2.4 Performance Metrics

Performance metrics are used to evaluate the success of deep learning models. In this study, the success of the U-Net model trained with Adam, Nadam, and RMSprop optimization techniques is analyzed by considering the accuracy (1), precision (2), recall (3), F1-score (4) and IoU (5) values commonly used in literature. In addition, the training time of the models are analyzed. In this context, true positive (TP), true negative (TN), false positive (FP), and false negative (FN) values are used.

Accuracy is the proportion of correct predictions out of all predictions made by the architecture during the testing phase (1). Precision is a metric that shows the number of true positives within positive predictions (2). Recall measures the proportion of true positives to correct predictions (3). The harmonic mean of precision and recall values gives the F1 score (4).

IoU (Intersection over Union) is a metric that expresses how much the predicted sample overlaps with the ground truth. It is frequently used, especially in classification problems. It is an indicator of how well the model distinguishes between building and background. It is represented by values ranging between 0 and 1. A value of 1 represents the best overlap, and 0 represents the worst overlap (5).

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{3}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{5}$$

where TP: True Positive - Correct detection of predictions that are actually building
TN True Negative - Correct detection of forecasts that are not actually buildings
FP: False Positive - A building that is not actually a building but is identified as a building as a result of estimation
FN: False Negative - A building that is actually a building but is not identified as a building as a result of estimation
Area of Overlap: The intersection of ground truth and prediction
Area of Union: Combination set of ground truth and prediction

Model performances are visually analyzed with the ROC curve graph. ROC curve contains false positive rate values in the row and true positive rate values in the column. This curve is frequently used in the evaluation of models. The closer the curve converges to the upper left corner, the more successful the model is. AUC represents the area under the curve and has a maximum value of 1.

The training time of the 3 trainings, the epoch value of the moment when the training was stopped by early stopping, and the epoch value of the moment when the minimum validation loss value was reached were recorded and analyzed about the training process.

Once training is complete, predictions are generated using the new test images received. This metric, called Inference Time, is in seconds. FPS refers to the data processed by the model in one second. By recording these metrics, the adaptability of the model to real-time applications is discussed.

2.5. Parameters

In this study, the parameter values of these algorithms, which are frequently used in literature, are used to evaluate the performance of the optimization algorithms accurately and are given in Table 1. However, the learning rates were tested experimentally, and the training was completed by determining the most appropriate learning rate for the model.

Parameters	Values		
	Adam	Nadam	RMSprop
Learning Rate (α)	0.001	0.001	0.0001
β_1	0.9	0.975	-
β_2	0.999	0.999	-
rho	-	-	0.9
ε	1e-8	1e-8	1e-8

Table 1. Optimizer parameters

In the training step, ReLU is used as the activation function on the U-Net model, and sigmoid functions are used in the output layer; the binary cross-entropy function is used as the loss function. Dropout rate is set to 0.2, and batch size is set to 2. The limitations of the graphics card are considered in choosing these values. To prevent overfitting, the training started with 100 epochs, and the early stopping technique is used to automatically stop the training if the validation loss of the model does not improve in 5 epochs.

3. Results

The dataset is preprocessed, and each image is divided into 256×256 pixel areas. The dataset consists of 1000 training, 100 validation, and 150 test images, totaling 1250 images.

The U-Net model is trained three times using Adam, Nadam, and RMSprop optimizers on 1000 images using TensorFlow and Keras libraries on GTX 1060 in a Python environment.

The epoch count is initially defined as 100 for all training. Loss Over Epoch (Figure 3) plots are generated for each training process, and model training is evaluated by monitoring the changes in training loss and validation loss. The graphs show the epoch at which the early stopping method used to prevent overfitting stops training and the epoch at which the model receives its last updated weights.

The trained model is analyzed with performance metrics using test images (Table 2). The performance metrics used are as follows: accuracy, precision, recall, F1 score, and IoU.

Performance Metrics	U-Net with Optimization Algorithms		
	Adam	Nadam	RMSprop
Accuracy	0.9132	**0.9189**	0.9158
Precision	**0.8436**	0.7940	0.7906
Recall	0.6437	**0.7499**	0.7326
F1 Score	0.7302	**0.7713**	0.7605
IoU	0.5751	**0.6278**	0.6136

Table 2. Performance results

In addition, information on training time, inference time, FPS metrics, and the number of epochs is given in Table 3. Epoch (a) in the table is the number of manual epochs entered by the user at the beginning of the training. Epoch (b) is the epoch when the early stopping method stops training. Epoch (c) is the number of epochs of the weights that were restored after the training was stopped.

Other Metrics	U-Net with Optimization Algorithms		
	Adam	Nadam	RMSprop
Training Time (min)	36.38	54.27	47.16
Inference Tıme (s)	4.56	4.50	4.44
FPS	32.9	33.36	33.77
Epoch (a)	100	100	100
Epoch (b)	22	29	27
Epoch (c)	17	24	22

Table 3. Other metrics

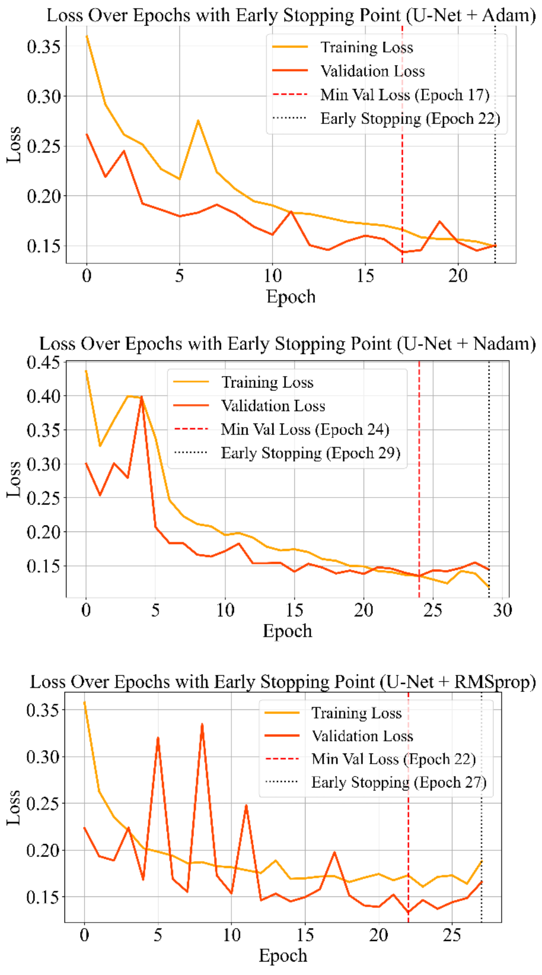

Figure 3. Loss over epochs with early stopping point

ROC curves for each training are obtained (Figure 4). An ROC curve contains false positive rate values in the row and true positive rate values in the column. This curve is often used to test the classification success of models.

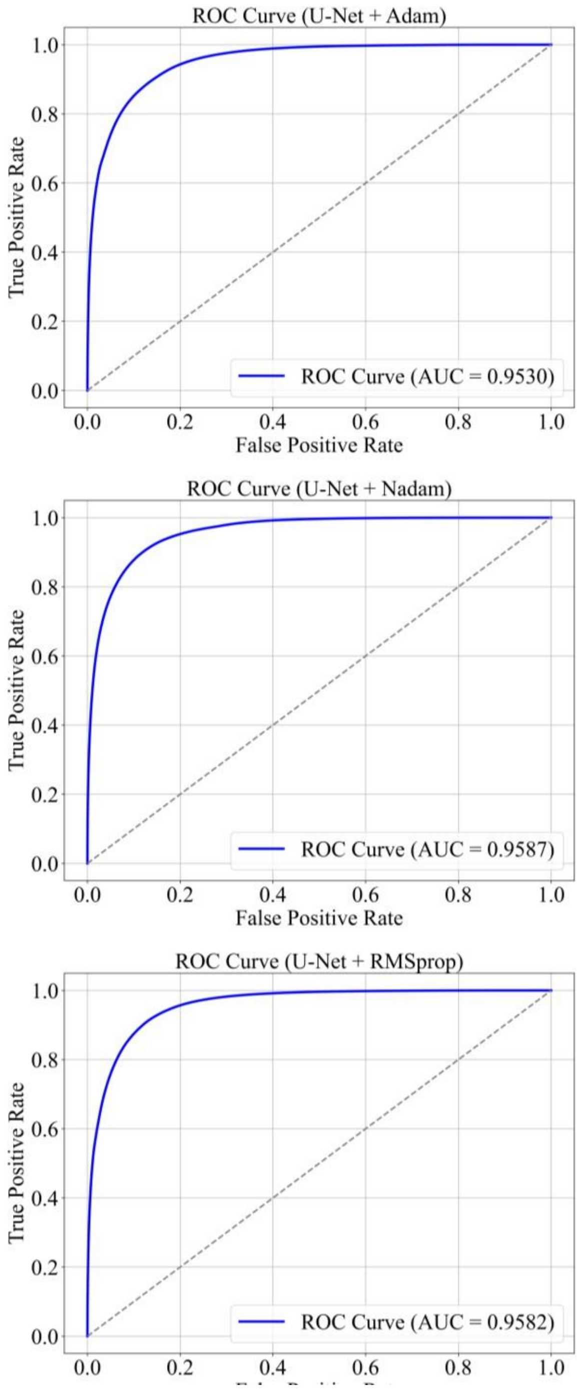

Figure 4. ROC curves

The performance of the trained models was analyzed with test images. Figure 5 is a representation of a sample image. The first column of the figure is composed of test images from the Massachusetts Buildings Dataset and their masks. In the second column, the model prediction for the same test image is placed. To discuss the findings of the building detection, the notable cases are marked. Comments on these cases will be discussed in the Discussion & Conclusion section.

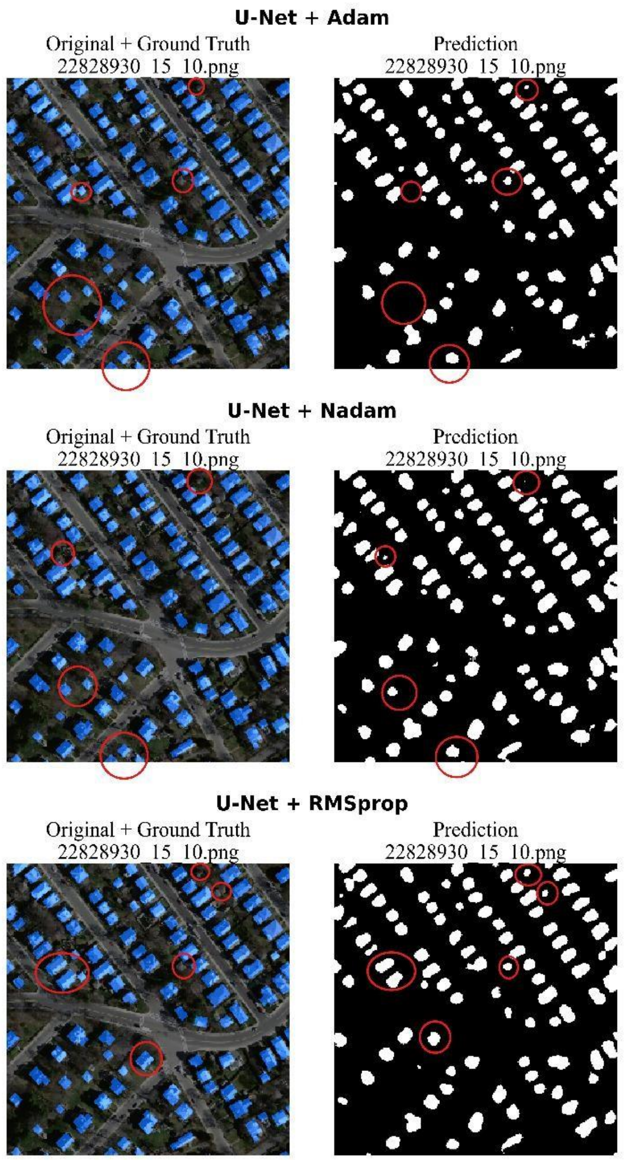

Figure 5. Ground truth and prediction on a sample test image

4. Discussion&Conclusion

The data set used in the study is a medium-scale data set. Model training and testing phases were performed on a GTX 1060 GPU.

Loss Over Epoch graphs were analyzed, and validation loss and training loss values were monitored to make inferences about the training performance of the model (Figure 3).

For the model trained with the Adam optimizer, the initial epoch value was defined by the user as 100 for all training. Early stopping stopped the training at epoch 22 to avoid overfitting. Throughout 22 epochs, the validation loss got the smallest value at epoch 17. Therefore, the weights at epoch 17 were loaded back, and the model training was completed.

Since the initial model weights were chosen randomly, the training loss was initially as high as 0.40 and decreased to 0.15 in the last epoch.

The validation loss tends to decrease over the 22 epochs but shows a fluctuating curve. The curve formed many local minima. From epoch 17 onwards, validation loss increased again.

Training with the Nadam optimizer is not consistent in the first epochs compared to training with the Adam optimizer. After 3 - 4 epochs, both training loss and validation loss decreased stably.

The lowest validation loss value was obtained at epoch 24. After this epoch, both curves followed a fluctuating trend. Since the validation loss value did not decrease during 5 epochs, the training was stopped at epoch 29, and the model weights at epoch 24 were used to construct the current weights.

The graph where the validation loss curve shows the most deviation belongs to the training with RMSprop. At epoch 22, the minimum degree was reached. The training loss curve is more stable than the validation loss curve and decreases steadily. Early stopping stopped training at epoch 27.

Comparing the 3 plots, it is observed that the training loss curve fluctuates during the training phase with the Nadam optimizer. Adam and Nadam showed similar performance, but Adam is more stable in both validation loss and training loss curves.

When the ROC curves (Figure 4) were analyzed, it can be said that all models are ideal in class distinction. It was found that the model with the best class separation belongs to the Nadam optimizer. In this respect, the model is a more successful classifier than the others.

When the results of the performance metrics are analyzed (Table 2), the highest accuracy score belongs to the Nadam optimizer, with 0.9189. In the test phase, both classes (buildings and background) were correctly predicted by this model with a high score.

The highest precision score belongs to the Adam optimizer, with 0.8436. The majority of the pixels predicted to belong to the building class were correctly predicted by this model.

The highest recall score belongs to the Nadam optimizer, with 0.7499. This metric indicates the proportion of the total pixels that should have been identified as buildings.

The Nadam optimizer is a successful optimizer with an F1 score of 0.7713. The F1 score is especially important for analyzing imbalanced data sets. This metric balances between precision and recall.

Intersection Over Union (IoU) calculates the overlap between the model's prediction and the ground truth. The Nadam optimizer also performed quite successfully in this metric, achieving the highest score of 0.6278.

Table 3 shows that the model trained with the Adam optimizer is the fastest-trained model, with a time of 36.38 min. It reached the lowest validation loss in its graph in the shortest time. The use of the Nadam optimizer instead of Adam in the training resulted in a significant increase in the training time.

When the inference time (s), which is a measure of the prediction time of the trained model on new images, and the FPS metrics, which express the amount of data the model processes in one second, are examined, it is found that the model trained with the RMSprop optimizer is faster than the others.
Figure 5 shows that the Adam optimizer found buildings overall but had difficulty in detecting small and closely located buildings. However, the model produced fewer false labels compared to the other two.

The Nadam optimizer is more successful than Adam in detecting small and close buildings. It also managed to detect many buildings. However, it produced false building labels, especially in areas with similar spectral characteristics as the building roof.

The RMSprop optimizer is the training optimizer that preserves the geometric shapes of buildings the least. Building corners that are sharp in the ground truth are detected as soft by the model. False building generation in the background is noticeable.

In this study, building segmentation was performed on a medium-sized dataset. The Massachusetts Buildings Dataset (Mnih, 2013) is a dataset used in many studies. After preprocessing the dataset, training and testing phases were completed using a total of 1250 images with 256×256 resolution.

The study was carried out with a video card with 6 GB of memory. This requires experimental tuning of the selected parameters.

Studies on the dataset show that the U-Net model trained with the Nadam optimizer has a stronger overall performance than the other models. The model trained with this optimizer performs best in the majority of the success metrics. Although there were serious fluctuations in the first epochs during the model training, a stable learning process was realized in the rest of the training process. However, training with the Nadam optimizer took longer compared to the other two.

The RMSprop optimizer contributed to speeding up the model. The model trained with this optimizer is the closest to real-time applications. However, it was inferior to the other optimizers in terms of fluctuations in the training process, performance metrics results, and smoothing sharp building boundaries.

The model trained with the Adam optimizer is generally successful but has difficulty in detecting small buildings. It is less suitable for real-time applications than other optimizers.

The findings show that the selected optimization algorithm significantly affects the model's success and its compatibility with real-time applications.

Existing optimization techniques are directly affected by data-dependent properties such as data set size, class distribution, and the capacity of the system used in the training phase. The behavior of optimization algorithms sometimes makes it difficult to generalize. For this reason, existing studies in literature continue experimentally.

Consequently, this study investigates the effects of optimization algorithms on the performance of the U-Net architecture for the building segmentation task on a medium-sized dataset. In addition, the effects of the selected optimizer on the suitability of the model for real-time applications are examined.

Future work will examine the effects of more optimizers on various large datasets and high-end environments to provide a general perspective on optimizers. In addition, the effects of optimizers on data sets with unbalanced class distributions will be investigated.

References

Anagün, Y., Işık, Ş., 2021: Contribution Analysis of Optimization Methods on Super-Resolution. *Afyon Kocatepe Univ. J. Sci. Eng.*, 21(6), 1343–1352.

Bouanane, K., Dokkar, B., Allaoui, M., Meddour, B., Kherfi, M. L., Hedjam, R., 2024: Behaviors of first-order optimizers in the context of sparse data and sparse models: a comparative study. *Digit. Signal Process.*, 153, 104637.

Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., Dahl, G. E., 2019: On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446*.

Chollet, F., 2021: *Deep Learning with Python*. 2nd ed., Manning Publications, Shelter Island, New York, USA.

Dozat, T., 2016: Incorporating Nesterov Momentum into Adam. In: Proc. 4th Int. Conf. Learning Representations (ICLR), Workshop Track, San Juan, Puerto Rico, 2–4 May 2016, 1–4.

Erener, A., 2013: Classification method, spectral diversity, band combination and accuracy assessment evaluation for urban feature detection. *Int. J. Appl. Earth Obs. Geoinf.*, 21, 397–408.

González, C. L. M., Montoya, G. A., Garzón, C. L., 2025: Toward Reliable Post-Disaster Assessment: Advancing Building Damage Detection Using You Only Look Once Convolutional Neural Network and Satellite Imagery. *Mathematics*, 13(7), 1–29.

Goodfellow, I., Bengio, Y., Courville, A., 2016: *Deep Learning*. MIT Press.

Jawahar, M., Jani Anbarasi, L., Jasmine S, G., Daya JL, F., Ravi, V., Chakrabarti, P., 2023: TRS-Net: Tropical revolving storm disasters analysis and classification based on multispectral images using 2-D deep convolutional neural network. *Multimed. Tools Appl.*, 82(30), 46651–46671.

Ji, S., Wei, S., Lu, M., 2018: Fully convolutional networks for multisource building extraction from an open aerial and satellite imagery data set. *IEEE Trans. Geosci. Remote Sens.*, 57(1), 574–586.

Kingma, D. P., Ba, J., 2014: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mnih, V., 2013: *Machine Learning for Aerial Image Labeling*. Doctoral Dissertation, University of Toronto, Canada.

Reyad, M., Sarhan, A. M., Arafa, M., 2023: A modified Adam algorithm for deep neural network optimization. *Neural Comput. Appl.*, 35(23), 17095–17112.

Ronneberger, O., Fischer, P., Brox, T., 2015: U-Net: Convolutional networks for biomedical image segmentation. In: *Med. Image Comput. Comput.-Assist. Interv. – MICCAI 2015*, 18(3), 234–241. Springer.

Ruder, S., 2016: An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Shahrabadi, S., Gonzalez, D., Sousa, N., Adão, T., Peres, E., Magalhães, L., 2023: Benchmarking deep learning models and hyperparameters for bridge defects classification. *Procedia Comput. Sci.*, 219, 345–353.

Taffese, W.Z., Sharma, R., Afsharmovahed, M.H., Manogaran, G., Chen, G., 2025: Benchmarking YOLOv8 for Optimal Crack Detection in Civil Infrastructure. *arXiv preprint* arXiv:2501.06922.

Tieleman, T., Hinton G., 2012: Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 26.

Wen, Q., Jiang, K., Wang, W., Liu, Q., Guo, Q., Li, L., Wang, P., 2019: Automatic building extraction from Google Earth images under complex backgrounds based on deep instance segmentation network. *Sensors*, 19(2), 333.

Zaheer, R., Shaziya, H., 2019: A study of the optimization algorithms in deep learning. In: *Proc. 3rd Int. Conf. Inventive Systems and Control (ICISC)*, 536–539.