INTERNATIONAL JOURNAL
ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Classification of Vegetation Land Cover Area Using Convolutional Neural Network

Galan Ramadan Harya Galib [a], Irwan Budi Santoso [a], Cahyo Crysdian [a,*]

[a] *Computer Science Department, Faculty of Science and Technology, UIN Maulana Malik Ibrahim, Malang, Indonesia*
Corresponding author: [*]cahyo@ti.uin-malang.ac.id

***Abstract*—The decrease and reduction of vegetation land or forest area over time has become a serious and significant problem to be considered. Increasing the Earth's temperature is a consequence of deforestation, which can contribute to climate change. The other issues that researchers face concern diversity and various objects in satellite imagery that may be difficult for computers to identify using traditional methods. This research aims to develop a model that can classify vegetation land cover areas on high-resolution images. The data used is sourced from the ISPRS (International Society for Photogrammetry and Remote Sensing) Vaihingen. The model used is a Convolutional Neural Network (CNN) with a VGG16-Net Encoder architecture. Tests were conducted on eight scenarios with training and test data ratios of 80:20% and 70:30%. The classifier method that we employed in this research is argmax and threshold. We also compared the performance of Neural Networks with two hidden layers and three hidden layers to investigate the impact of adding another layer on the Neural Network's performance in classifying vegetation land cover areas. The results show that using the threshold classifier method can save training time compared to the argmax method. By increasing the number of hidden layers in the neural network, model performance improves, as shown by increases in recall, accuracy, and F1-score metrics. However, there is a slight decrease in the precision metric. The model achieved its best performance with a precision (Pre) of 99.5%, accuracy (Acc) of 83.3%, and F1-score (Fs) of 70.3%, requiring a training time (T-time) of 16 minutes and 41 seconds and an inference time (I-time) of 0.1535 seconds.**

***Keywords*—Classification; convolutional neural network; vegetation land cover.**

Manuscript received 30 Aug. 2024; revised 8 Dec. 2024; accepted 23 Jan. 2025. Date of publication 31 Jul. 2025.

I. Introduction

The decrease in vegetation land cover has an impact on the increase in the Earth's temperature. This can lead to climate change [1], [2]. On 26 July 2023, BPS (Badan Pusat Statistik) published data on Indonesia's net deforestation rate within and outside forest areas for the period from 2013 to 2022 [3]. The data showed that the annual total reforestation rate never exceeds the total deforestation rate in Indonesia. As a result, the deforestation rate is always higher than the reforestation rate. The data on deforestation in Indonesia is shown in Fig. 1.

The reduction of forest area and vegetation land over time has become a serious and pressing problem to be addressed by both the government and Indonesian citizens. Deforestation can lead to reduced water catchment areas, resulting in flooding [4]. In addition, according to Abbass et al. [5] deforestation is also a cause of climate change that has a negative impact on the agricultural sector, human health levels, forestry sector, tourism sector, economic growth, increased greenhouse gas emissions, and loss of habitat and biodiversity.

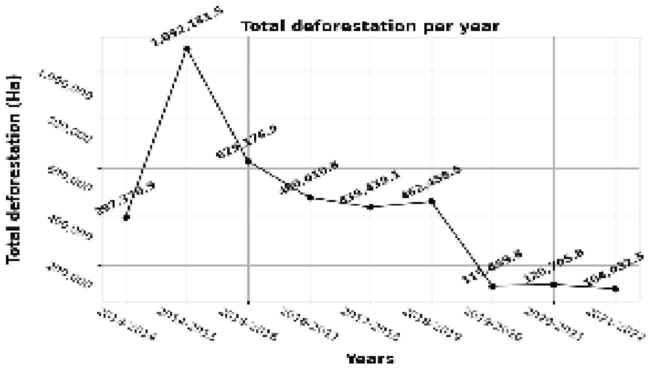


Fig. 1 Total deforestation per year

The government is taking the issue of deforestation in Indonesia seriously. According to the Minister of Environment and Forestry of the Republic of Indonesia's

Regulation Number 1 of 2022, several government strategies are outlined to reduce the rate of deforestation through the Ministry of Environment and Forestry. Some of the efforts made by the Ministry of Environment and Forestry include rehabilitating forest areas, controlling forest and land fires, mitigating climate change, increasing the use of new renewable energy sources, and expanding the area of vegetation land cover. Furthermore, to monitor and anticipate deforestation, a system is needed that can effectively and efficiently classify vegetation areas using satellite images or high-resolution images.

Convolutional Neural Network (CNN) is one of the methods in deep learning for computer vision tasks that uses images as input and optimizes and sets weight parameters through the learning process on the model, so that the model can distinguish between one object and another [6], [7], [8], [9], [10]. CNN is designed to be able to extract and detect significant features from an image without any human supervision [11]. The key to CNN is the convolution operation in which there is a kernel or filter applied to each image location.

CNN is one of the best methods to understand the content of an image, extract the features from the image, and also performs well for tasks like segmentation, classification, object detection, and image retrieval [12]. In the context of processing and analyzing remote sensing or high-resolution images, CNN performance can surpass that of other machine learning methods in detecting objects or performing pixel-wise classification of land cover classes. Then for remote sensing, CNN can be used for mapping areas, crop disease detection, agriculture, land cover change analysis, and forestry sectors [13], [14], [15].

While many image processing methods, such as edge detection, can aid in examining satellite images, they may not always be sufficient for accurately classifying vegetation. To enhance edge detection in image analysis, Crysdian [16], [17] investigated ways to assess edge detector performance without ground truth. However, depending solely on spatial features like edges or shapes can be restrictive for complex tasks, such as classifying vegetation in satellite imagery. Color and other spectral information found in satellite photos are crucial in differentiating between different types of land cover. CNN models are effective for vegetation classification tasks in satellite imagery because, in contrast to edge-based methods, they can use both spatial and spectral information. By leveraging the rich spectral data available in multi-band satellite imagery, CNN can capture subtle differences in vegetation types that may not be visible from spatial features alone.

Furthermore, satellite imagery is a picture of the shape of the Earth's surface obtained through one or more satellites operating in space to observe the Earth. According to Sussman et al. [18] the definition of satellite imagery is an image that comes from a satellite in Earth orbit. Researchers widely use satellite images to monitor various objects on Earth. For example, Yang et al. [19] used satellite imagery in agriculture to monitor the development of plants in rice fields. Then, in the field of geology, Eldosouky et al. [20] utilize ASTER satellite imagery to locate mining minerals. Next, in research conducted by Si et al. [21] satellite imagery is used to predict cloud movement. In addition, Mujetahid et al. [22] utilized satellite imagery from Sentinel 1 and 2 through Google Earth Engine to monitor the impact of illegal logging, which resulted in deforestation in the forest area of South Sulawesi Province. Next is research by Timilsina et al. [23] which utilizes the results of Quickbird satellite imagery and Google Earth to map changes in tree cover areas in the urban area of Hobart, Tasmania, Australia, using the CNN method. On the other research that has been done by Aldiansyah and Saputra [24] detecting land cover change in area of Southeast Sulawesi using Google Earth Engine to get the satellite imagery. Then the research by Yuh et al. [25] monitoring the land cover change in the northern portion of the Mayo Rey Department, which is located in the North Province of Cameroon, using data from Landsat 7 and Landsat 8.

Research related to predicting pixel classes in high-resolution digital images has been conducted in numerous previous studies using various methods. First research by Abdalla et al. [26], they modified the VGG16 Net model by incorporating an encoder for RGB image segmentation. They ran the model on a NVIDIA GeForce GTX 1080 Ti, equipped with 3458 CUDA cores and 16 GB of GPU memory. The strategy involved a learning rate of 0.001, 100 iterations, and the use of cross-entropy as the loss function. They utilized a dataset containing 400 images with a dimension of 400x500, which was divided into 60% for training, 20% for validation, and the remaining 20% for testing.

In a study conducted by Yi et al. [27], they introduced the DeepResUnet method to perform urban building segmentation at pixel scale from VHR (Very High Resolution) remote sensing imagery. They conduct all steps of the experiment using Keras and TensorFlow with an NVIDIA GeForce GTX 1080 Ti GPU (11 GB RAM). They perform patching or split the image into patches with a size of 256x256 dimensions. After that, they split the data into a training set and a testing set with a ratio of 80:20 %. They employed Glorot to initialize the network parameters and used cross-entropy loss during training. For the optimizer, they adopted the Adam optimizer to optimize the training loss with a learning rate of 0.001. Because of the limitation of the GPU, a batch size of 6 was chosen in the experiment.

Xiong et al. [28] proposed TCU-Net by combining CNN and Transformer to extract shorelines from multispectral remote sensing images. All of the experiments are performed with NVIDIA GeForce RTX 3090 and using the PyTorch framework. They chose the Chinese coastline on the Yellow Sea as the designated study area and acquired the images from Gaofen-6 (GF-6) in February 2023. Due to the large size of the image, they cropped it and then divided it into ocean and land categories. Ultimately, they select only images that contain both ocean and land, resulting in 2100 images and 2100 labels with a dimension of 512x512. The models were trained with a learning rate of 0.001 and the AdamW optimizer, using momentum of 0.9 and a weight decay of 0.01. To speed up the training process, they set the batch size to 16 and the epoch number to 100.

In the study by Yang et al. [29], the AD-HRNet model was developed, integrating HRNet with attention mechanisms and dilated convolution. Implemented in PyTorch on two NVIDIA GeForce RTX 2080Ti GPUs, it used the ISPRS Potsdam and Vaihingen datasets. The Potsdam dataset had 24 training and 14 testing images, subdivided into 512x512 patches, resulting in 5400 training and 3150 testing images.

The Vaihingen dataset had 16 training and 17 testing images, resulting in 479 training and 555 testing patches. They used the SGD optimizer with a 0.01 learning rate and weighted cross-entropy loss. The Potsdam dataset ran for 484 epochs, while Vaihingen ran for 100 epochs.

Another study by Yu et al. [30] introduces a hybrid architecture called ICTANet, which incorporates convolutional and Transformer architectures to improve the segmentation performance on remote sensing urban imagery. All experiments were implemented using PyTorch on a single NVIDIA GeForce RTX 2080 Ti GPU with 11 GB of memory. They employed the AdamW optimizer with a learning rate of 0.001 and momentum of 0.01 to train and optimize the model. They used the ISPRS Postdam and Vaihingen dataset. They patched or cropped the image into 512x512 using a sliding window approach. To improve the performance of models, image augmentation such as horizontal and vertical flipping, rotation, and scaling is employed. For ISPRS Vaihingen dataset, they split it into data training (45.5%), validation (3.0%), and testing (51.5%). Then, for the ISPRS Potsdam dataset, they split it into data training (57.9%), validation (2.6%), and testing (39.5%).

Finally, Su et al. [31] proposed an improved U-Net architecture called AtrousDenseUDeconvNet, which combined DenseNet, dilated convolution, and DeconvNet. To train the model, they use an Intel i9-7900x processor with 32GB RAM and GTX 1070, 1080, 3090 GPUs. All of the program is coded using the package TensorFlow 2.0 and Keras 2.3.1. They used the ISPRS Potsdam dataset for this research. They are augmenting the image data with random cropping, rotation, mirroring, exposure adjustment, contrast adjustment, chroma adjustment, and adding noise. The data was divided into training, validation, and test sets in a 4:1:1 ratio. Cross-entropy loss function with Adam optimizer and a learning rate of 0.001 was used in the training process.

According to the literature, numerous researchers have investigated high-resolution image segmentation using sophisticated models. However, this research focuses on simpler models and optimizing computational resources to train them by minimizing the number of model parameters, without significantly compromising performance. The implementation and research of image segmentation using complex architectures is the primary focus of earlier research by Abdalla et al. [26] and Dechesne et al. [32]. However, these studies did not consider the advantages of using classifiers at the end of the neural network (threshold and argmax), or the simplicity of alternative models. Furthermore, according to the literature review, no studies have examined how improving hidden layers in basic neural networks can aid in classifying vegetation and land cover. This study aims to fill the gaps by examining efficient training techniques and the benefits of simpler architectures.

II. Materials and Method

A. Research Stage

The steps used in the research to classify vegetation land cover areas using CNN are shown in Fig. 2.

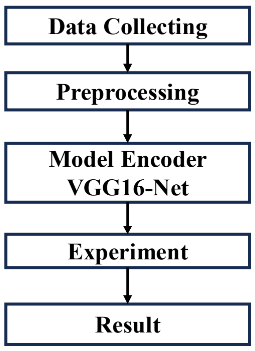


Fig. 2 Research Stage

In Fig. 2, there are several stages of the research process as follows:

1) Collecting high-resolution image data from the International Society for Photogrammetry and Remote Sensing (ISPRS) Vaihingen.

2) Data preprocessing is a stage that prepares images to be included in the model. This stage provides normalization to scale the pixel value and patching to divide high-resolution images into the same dimension.

3) Implementation of modeling using Encoder VGG16-Net to classify the pixel image (pixel-wise classification).

4) In the experiment stages, we define testing scenarios, such as splitting data for training and testing, varying the number of hidden layers in the neural network to perform pixel-wise classification, and the classifier method that determines the pixel class.

5) Obtain the results from pixel-wise classification of vegetation land cover.

B. Work Environment

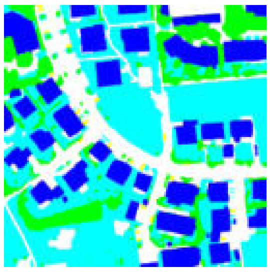
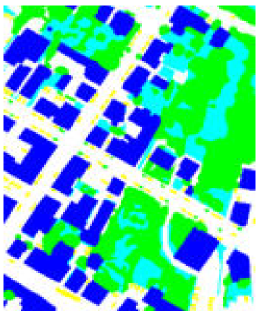
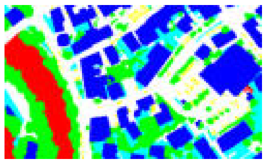
The work environment in this study is Python 3.9.16 with a Notebook on Visual Studio Code. The training and testing are conducted on a laptop with an AMD Ryzen 5 5500U processor with Radeon Graphics, running at 2.1GHz (12 CPUs) and 16GB of RAM.

C. Data Collecting

This study uses high-resolution images and ground truth data from the ISPRS website (https://www.isprs.org) to classify vegetation land coverage areas. The primary data is from Vaihingen, Germany, taken using UAVs. The study utilizes 18 images, comprising nine high-resolution images and nine corresponding ground truths, categorized into six classes: waterproof surfaces, backgrounds, trees, buildings, low vegetation, and cars. The vegetation class consists of trees and low vegetation, while the non-vegetational class includes waterproof surface, background, building, and car classes. The study presents examples of high-resolution and ground truth images in Table 1.

The ground truth part of images is represented by different colors, with white representing the waterproof surface class, red representing the background, green representing trees, old blue representing buildings, cyan representing low vegetation, and yellow representing automobiles. The study aims to provide a comprehensive understanding of vegetation land coverage areas using digital images.

TABLE I
EXAMPLE HIGH-RESOLUTION IMAGE AND GROUND TRUTH

Number	High-Resolution Image	Ground Truth
1		
2		
3		

D. Data Preprocessing

There are two stages in image preprocessing on this study, namely normalization and patching. The first process that occurs in the image preprocessing phase is normalization. The purpose of doing the normalization of the image is to ease the computational [33]. Normalization is done by dividing each pixel value of the image by 255. The normalization equation is as follows:

$$pixel\ norm = \frac{pixel\ original}{255} \quad (1)$$

where pixel norm is normalized pixel values in the range between 0 and 1, pixel original is the original pixel value in the image.

The following process involves splitting or patching high-resolution images with ground truth into small pieces, i.e., patching them into 128x128 pixels. This is because the Vaihingen image dataset has extensive and varied dimensions (see Table 1, which displays the sample data); therefore, it is necessary to break down the dimensions into smaller pieces to facilitate model learning. After patching, the number of images is 2526 for both high-resolution and ground truth images, resulting in a total of 5052. Examples of patching results are shown in Fig. 3 and Fig. 4.

E. Pixel-wise Classification

Pixel-wise classification is the last layer in the VGG16-Net Encoder architecture before the output. The pixel-wise classification process runs in several stages. The first step is to train the neural network using the input data, which are feature maps from the last convolutional layer. These feature maps are the result of a feature extraction from an image that has gone through a series of convolutions, max pooling, and unpooling within the VGG-16 Net encoder architecture. The VGG-16 Net encoder architecture is described in Fig. 6.

For the entire convolution process, we used the Gaussian kernel, based on the research carried out by S. Bhairannawar et al. [34] that are shown in Fig. 5.

$$\begin{bmatrix} 21 & 31 & 21 \\ 31 & 48 & 31 \\ 21 & 31 & 21 \end{bmatrix} * \frac{1}{256}$$

Fig. 5. Gaussian Kernel

Each image will generate one feature map. The dimension of the feature maps is the same as that of the image before the feature extraction was performed, and the dimension of this feature map is also the dimension of the ground truth of the extracted image.

The neural network blocks mentioned in Fig. 7 analyze each pixel in the feature maps, which the neural networks would then evaluate to determine the most significant prediction value for the vegetation or non-vegetation class. The prediction results of the class would then be compared to the ground truth on the same pixel index.

Two classifier methods are used in this study. The first method is with argmax. The way argmax works is by obtaining the index with the greatest value between the neurons on the output layer, where the neuron O1 represents a non-vegetational class, and the neuron O2 represents the vegetational class. If the O1 neuron ($Z_{output}[0]$) has a higher predictive value compared to the O2 neuron ($Z_{output}[1]$), then the predicted class is class 0 or non-vegetation. So is the opposite. The argmax equation is as follows:

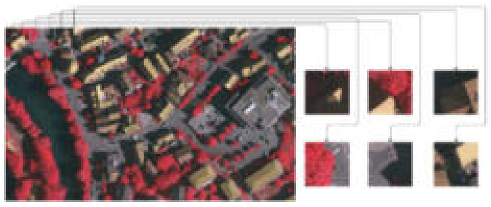


Fig. 3 Patching a high-resolution image into 128x128 dimensions

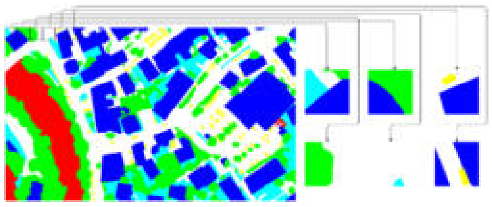


Fig. 4 Patching ground truth into 128x128 dimensions

$$\hat{y} = \begin{cases} 0\ if\ \ Z_{output}[0] > Z_{output}[1] \\ 1\ if\ Z_{output}[1] > Z_{output}[0] \end{cases} \quad (2)$$

where $\hat{y}$ is index result with largest prediction value in Z_{output}, $Z_{output}[0]$ is the value on the first index (neuron O1) representing the class of non-vegetation, and $Z_{output}[1]$ is is the value on the first index (neuron O1) representing the class of vegetation.

The second method is thresholding. The working method of thresholding is to set threshold values as class definers. In this study the threshold value used is 0.5. If the O1 neuron ($Z_{output}[0]$) has a predictive value higher than the threshold value, then the predicted class is class 0 or non-vegetation. And the other way around. The argmax equation is as follows:

$$\hat{z} = \begin{cases} 0 & \text{if } Z_{output}[0] > 0.5 \\ 1 & \text{if } Z_{output}[1] > 0.5 \end{cases} \tag{3}$$

where $\hat{z}$ is index result with largest prediction value in Z_{output}, $Z_{output}[0]$ is the value on the first index (neuron O1) representing the class of non-vegetation, and $Z_{output}[1]$ is the value on the first index (neuron O1) representing the class of vegetation.

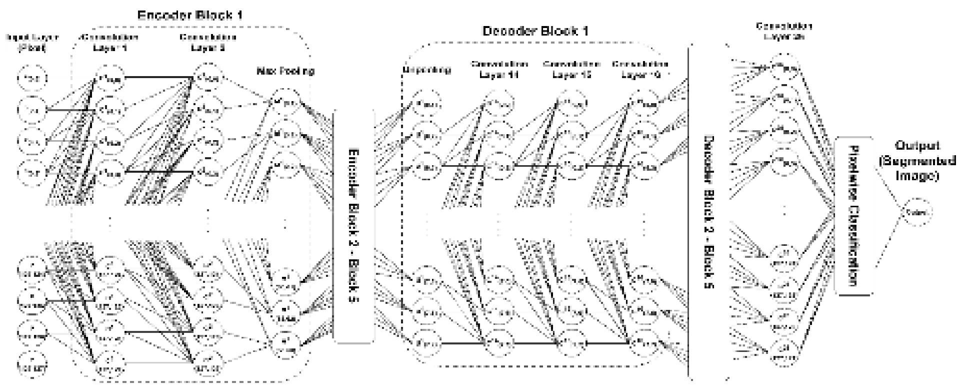


Fig. 5 VGG16-Net Encoder architecture

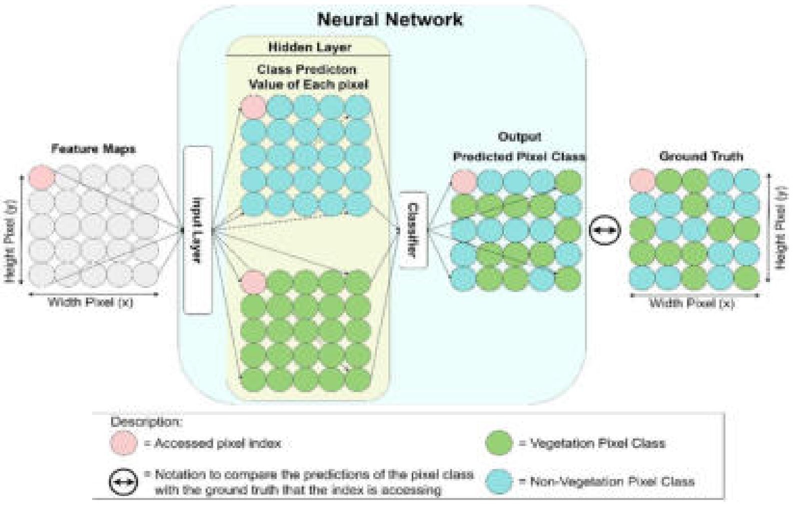


Fig. 6 Pixel-wise classification process

F. Proposed Neural Network Architecture

This study uses two different neural network architectures. The first neural network architecture had two hidden layers with details 1 input layer, 2 Hidden Layers, and 1 output. Neural network architecture with 2 hidden layers shown in Fig. 7.

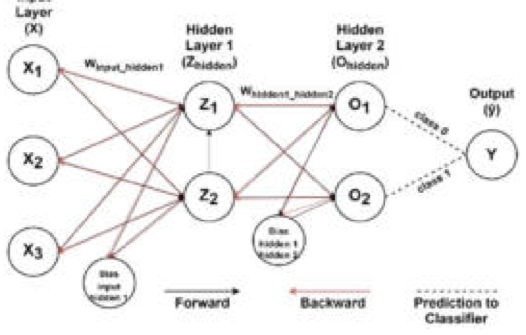


Fig. 7 Neural network 2 hidden layers

The first layer of the three hidden layer neural network architecture is an input layer with 3 neurons (X_1, X_2, X_3) each receiving the RGB (Red Green Blue) value of the pixel of the image accessed. The second and third hidden layers have the same number of neurons. The number of neurons in each of these hidden layers is two, meaning that the class of pixels to be classified is vegetative and non-vegetative. Then on the third layer there's one output neuron. This output neuron serves to accommodate the calculations of the classifier method that determines the class of the accessed pixel index.

The second neural network architecture has three hidden layers with details 1 input layer, 3 hidden Layers, and 1 output. Neural network architecture with 3 hidden layers shown in Fig. 8.The first layer of the two hidden layer neural network architecture is an input layer with 3 neurons (X_1, X_2, X_3) each receiving the RGB (Red Green Blue) value of the pixel of the image accessed. Then on the second hidden layer has three neurons, which means the same as the input layer. That is, three neurons represent the number of channels of images that enter the neural network. The third and fourth hidden layers have the same number of neurons. The number of neurons in each of these hidden layers is two, meaning that the class of pixels to be classified is vegetative and non-vegetative.

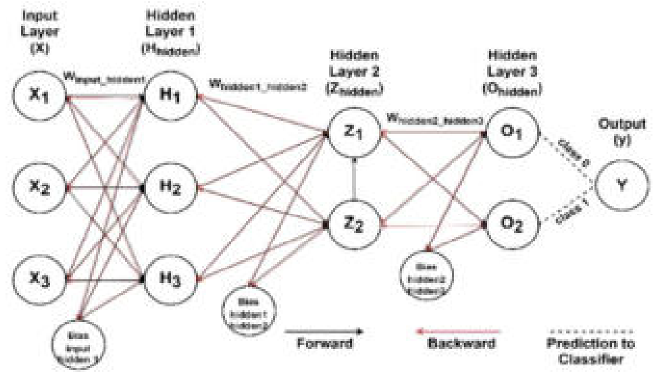


Fig. 8 Neural network with three hidden layers

Then, on the fifth layer, there is one output neuron. This output neuron serves to accommodate the calculations of the classifier method that determines the class of the accessed pixel index.

G. Loss Function

From the comparison of the ground truth with the pixel class prediction, the total error is obtained. The error calculation is done using the binary cross-entropy function. The binary cross-entropy (BCE) equation is as follows:

$$BCE = -\frac{1}{N}\sum_{i=1}^{H}\sum\nolimits_{j=1}^{W}(G_{ij}\,.\log(P_{ij}) + (1-G_{ij}).\log(1-P_{ij})) \quad (4)$$

where BCE is error value from the calculation, H is height of the image, W is width of the image, N is the total number of pixels $N = H \times W$), G_{ij} is the ground truth value on pixels i, j, where G_{ij} can be 0 (non-vegetation) or 1 (vegetation), and P_{ij} is the probability of a pixel class prediction at pixel location i, j.

III. Results and Discussion

To find the best model performance for a pixel-wise classification task. In this study, we conduct several model scenario configurations on the splitting data ratio for training and testing, the classifier on the layer-wise pixel of the neural network, and the size of the hidden layer in the neural network. This study implements VGG16-Net Encoder with a custom neural network architecture on the pixel-wise classification layer. There are eight experimental scenarios in this study, namely scenarios 1-A-J, 1-A-K, 1-B-J, 1-B-K, 2-A-J, 2-A-K, 2-B-J, and 2-B-K. The proposed model scenarios are shown in Table 2.

All scenarios use epoch 20, batch size 128, learning rate 0.001, and error tolerate 0.1. These model scenario settings are the approach to achieving optimal performance and minimizing the potential for overfitting or under fitting issues

TABLE II
Proposed Model Scenarios

Scenario	Ratio	Classifier	Hidden Layer
1-A-J	80:20	Argmax	2
1-A-K	80:20	Argmax	3
1-B-J	80:20	Threshold	2
1-B-K	80:20	Threshold	3
2-A-J	70:30	Argmax	2
2-A-K	70:30	Argmax	3
2-B-J	70:30	Threshold	2
2-B-K	70:30	Threshold	3

A. Scenario 1-A-J

This scenario uses 80% training data split (2021 images) and 20% testing data. (505 images). For the classifier method used is argmax. Then the architecture is NN with two hidden layers. Fig. 9 shows the loss or error graph of the training results.

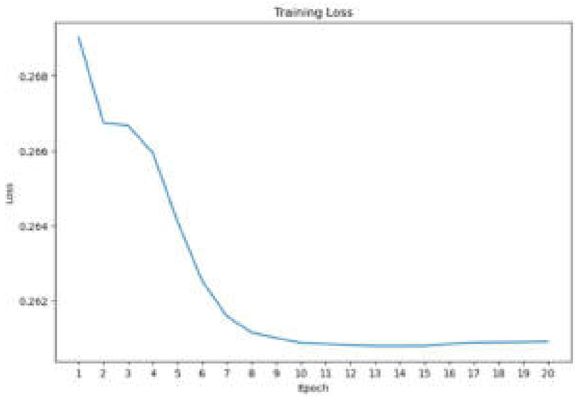


Fig. 9 Loss graph of scenario 1-A-J

The training time duration to reach 20th epoch is 14 minutes and 51 seconds. From this training process the loss value obtained is 0.260902. Optimal weight and bias values used in testing process. Tests conducted in 1-A-J scenarios yielded a precision of 99.7%, F1-Score of 63.4% accuracy of 75.6% with inference time 0.1873 seconds.

B. Scenario 1-B-J

This scenario uses an 80% training data split (2021 images) and a 20% testing data. (505 images). The classifier method used is a threshold. Then the architecture is NN with two hidden layers. Fig. 10 shows the loss or error graph of the training results.

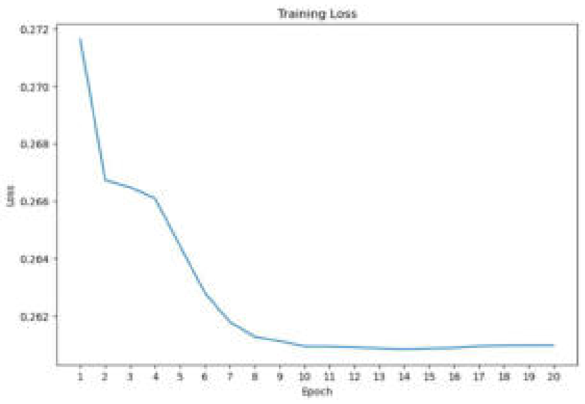


Fig. 10 Loss graph of scenario 1-B-J

The training time duration to reach the 20th epoch is 14 minutes and 31 seconds. From this training process, the loss value obtained is 0.260971. Optimal weight and bias values are used in the testing process. Tests conducted in 1-B-J scenarios yielded a precision of 99.7%, an F1-score of 63.2%, and an accuracy of 75.4% with an inference time of 0.1816 seconds.

C. Scenario 2-A-J

This scenario uses a 70% training data split (1769 images) and a 30% testing data. (757 images). The classifier method used is argmax. Then the architecture is NN with two hidden layers. Fig. 11 shows the loss or error graph of the training results.

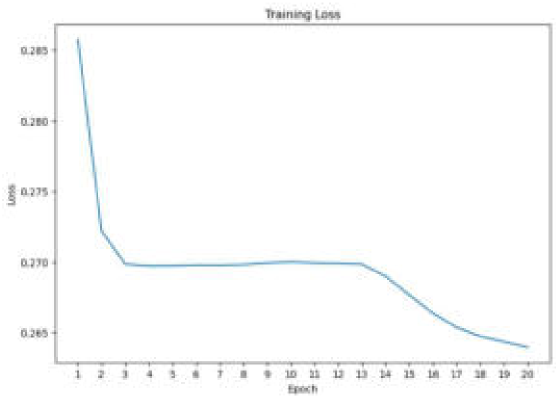


Fig. 11 Loss graph of scenario 2-A-J

The training time duration to reach the 20th epoch is 13 minutes and 12 seconds. From this training process, the loss value obtained is 0.263995. The optimal weight and bias values of the training results are stored and then used in the testing process. Tests conducted in 2-A-J scenarios yielded a precision of 99.7%, an F1-score of 59.9%, and an accuracy of 79.2% with an inference time of 0.1910 seconds.

D. Scenario 2-B-J

This scenario uses a 70% training data split (1769 images) and a 20% testing data. (757 images). The classifier method used is a threshold. Then the architecture is an NN with two hidden layers. Fig. 12 shows the loss/error graph of the training results.

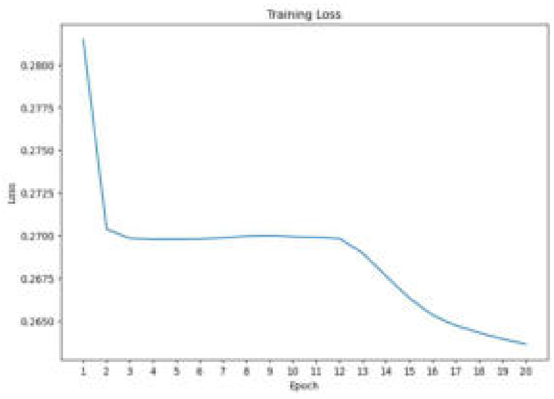


Fig. 12 Loss graph of scenario 2-B-J

The training time duration to reach the 20th epoch is 12 minutes and 36 seconds. From this training process, the loss value obtained is 0.263652. The optimal weight and bias values of the training results are stored and then used in the testing process. Tests conducted in 2-B-J scenarios yielded a precision of 99.7%, an F1-score of 60.6%, and an accuracy of 79.5% with an inference time of 0.2134.

E. Scenario 1-A-K

This scenario uses an 80% training data split (2021 images) and a 20% testing data. (505 images). The classifier method used is a threshold. Then the architecture is NN with three hidden layers. Fig. 13 shows the loss or error graph of the training results.

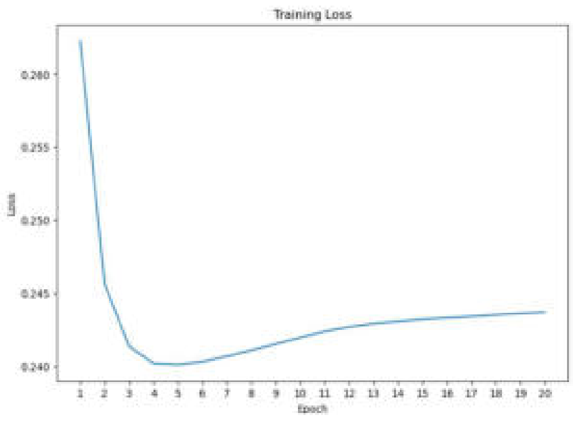


Fig. 13 Loss graph of scenario 1-A-K

The training time duration to reach the 20th epoch is 20 minutes and 41 seconds. From this training process, the loss value obtained is 0.243704. The optimal weight and bias values of the training results are stored and then used in the testing process. Tests conducted in 2-B-J scenarios yielded a precision of 99.5%, an F1-score of 70.1%, and an accuracy of 79% with an inference time of 0.1993 seconds.

F. Scenario 1-B-K

This scenario uses an 80% training data split (2021 images) and a 20% testing data. (505 images). The classifier method used is a threshold. Then the architecture is an NN with three hidden layers. Fig. 14 shows the loss or error graph of the training results.

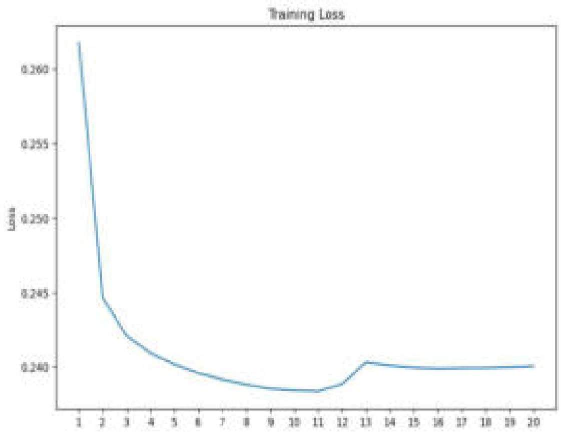


Fig. 14 Loss graph of scenario 1-B-K

The training duration to reach the 20th epoch is 19 minutes and 5 seconds. From this training process, the loss value obtained is 0.240046. The optimal weight and bias values of the training results are stored and then used in the testing process. Tests conducted in 2-B-J scenarios yielded a precision of 99.5%, an F1-score of 70.1%, and an accuracy of 79% with an inference time of 0.2223 seconds.

G. Scenario 2-A-K

This scenario uses a 70% training data split (1769 images) and a 30% testing data. (757 images). The classifier method used is argmax. Then the architecture is an NN with three hidden layers. Fig. 15 shows the loss or error graph of the training results.

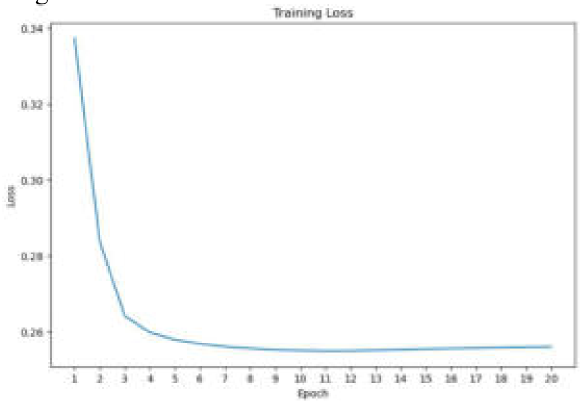


Fig. 15 Loss graph of scenario 2-A-K

The training duration to reach the 20th epoch is 20 minutes and 41 seconds. From this training process, the loss value obtained is 0.243704. The optimal weight and bias values of the training results are stored and then used in the testing process. Tests conducted in 2-B-J scenarios yielded a precision of 99.5%, an F1-score of 70.3%, and an accuracy of 83.3% with an inference time of 0.1611 seconds.

H. Scenario 2-B-K

This scenario uses a 70% training data split (1769 images) and a 20% testing data. (505 images). The classifier method used is a threshold. Then the architecture is an NN with three hidden layers. Fig. 16 shows the loss or error graph of the training results.

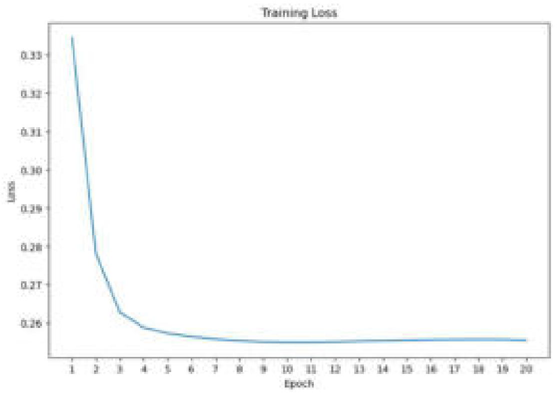


Fig. 16 Loss graph of scenario 2-B-K

The training time duration to reach 20th epoch is 20 minutes and 16 seconds. From this training process the loss value obtained is 0.255468. The optimal weight and bias values of the training results are stored and then used in the testing process. Tests conducted in 2-B-K scenarios yielded a precision of 99.5%, F1-Score of 70.2%, and accuracy of 83.3% with inference time 0.1536 seconds.

The results of testing each scenario are collected into a single table to make it easier to compare the performance of each scenario. The results are presented in Table 3. Comparison of the fastest training time in this study was obtained by the 2-B-J scenario of 12 minutes and 36 seconds, whereas for the longest training duration in the study the 1-A-J script of 20 minutes and 41 seconds. By comparison, the difference between the fastest and the longest training time is likely to be large, with a difference of 8 minutes and 5 seconds, or 64.15%. The fastest inference time in this study was obtained by scenario 2-B-K, at 0.1536 seconds. A summary of the duration of the training time of each scenario is visually presented in Fig. 17, and the inference time of each scenario is visually presented in Fig. 18.

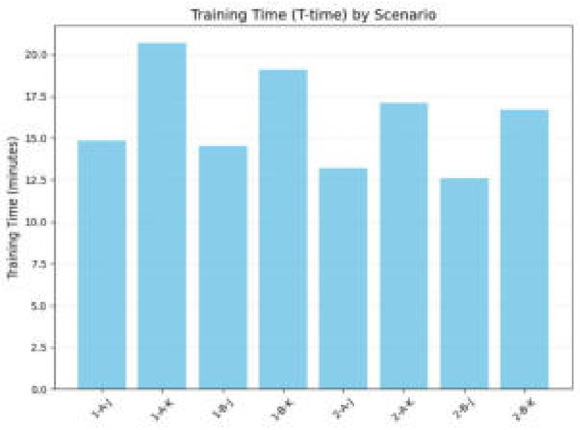


Fig. 17 Visualize the training time of each scenario

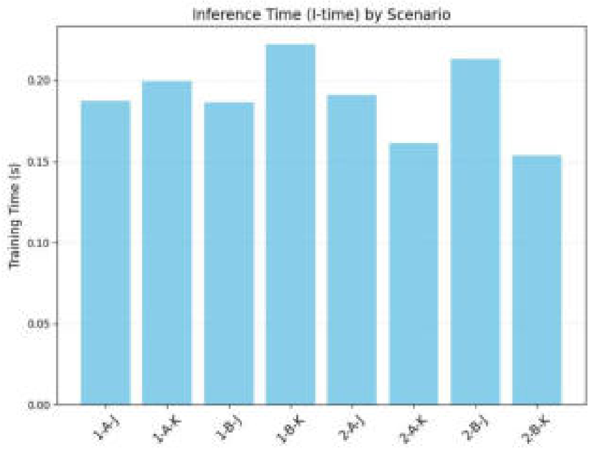


Fig. 18 Visualize the training time of each scenario

TABLE III
TESTING RESULTS OF EACH SCENARIO

Scenario	Pre	Acc	Fs	T-time	I-time	Parameters
1-A-J	99.7%	75.6%	63.4%	14m 51s	0.1873s	98381
1-A-K	99.5%	79%	70.1%	20m 41s	0.1993s	
1-B-J	99.7%	75.5%	63.2%	14m 31s	0.1861s	
1-B-K	99.5%	79%	70.1%	19m 5s	0.2223s	

Scenario	Pre	Acc	Fs	T-time	I-time	Parameters
2-A-J	99.7%	79.2%	59.9%	13m 12s	0.1910s	98393
2-A-K	99.5%	83.3%	70.3%	17m 6s	0.1611s	
2-B-J	99.7%	79.5%	60.6%	12m 36s	0.2134s	
2-B-K	99.5%	83.3%	70.3%	16m 41s	0.1536s	

For the best scenario, we conclude that scenario 2-B-K is the best scenario. Based on the Precision, F1-Score, Accuracy, and Inference Time, scenario 2-B-K achieves the best result compared to other scenarios. The visualization of Precision, F1-Score, and Accuracy of each scenario is shown in Fig. 19.

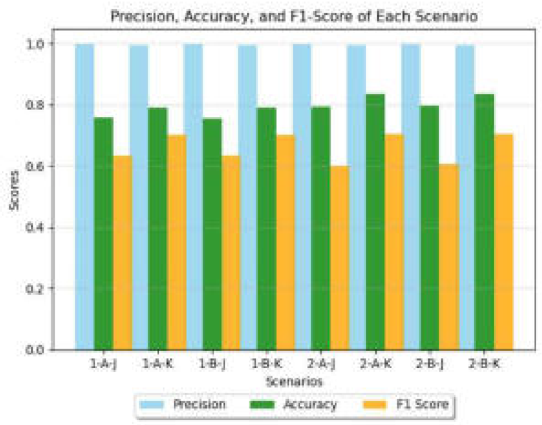


Fig. 19 Comparison of the metrics of all scenarios

Then we compared the performance based on the duration of training time between scenario 2-A-K and 2-B-K. The fastest training time between the two comparisons is on scenario 2-B-K with a training time duration of 16 minutes and 41 seconds. Scenario 2-B-K has a different 25 seconds compared to 2-A-K. In detail, we also compare the performance results of the best model scenario (scenario 2-B-K) with those of previous research. A comparison with the existing method is presented in Table 4 and Fig. 20.

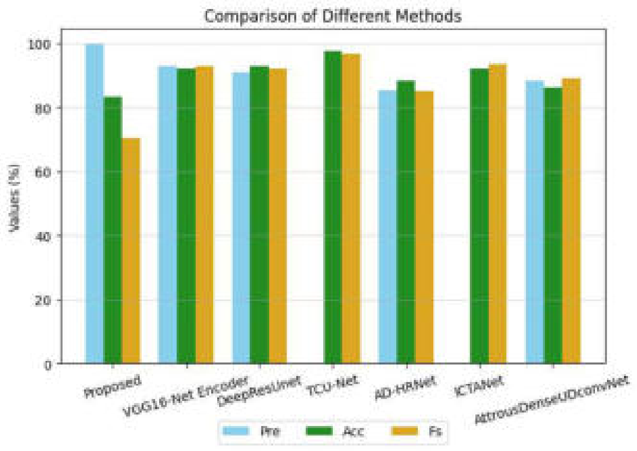


Fig. 20 Comparison with existing method

The proposed method's overall performance has a good result, achieving a precision of 99.5%, an accuracy of 83.3%, and an F1-score of 70.3%. The performance results of the proposed method are comparable to those of previous models. Furthermore, it excels in training efficiency, with a training time of 50.05 seconds per epoch and a parameter count of 98.393, making it more manageable than other models, such as DeepResUnet, which achieves an accuracy of 92.8% but incurs a significantly longer training time of 2520 seconds per epoch. The efficiency of the proposed method in terms of training time, inference time, and computing resources makes it a compelling choice for applications that require rapid training and effective performance.

TABLE IV
COMPARISON WITH EXISTING METHOD

Method	Pre (%)	Acc (%)	Fs (%)	T-time(s) /epoch	I-time (s)	Parameters
Proposed	99.5	83.3	70.3	50.05	0.153	98393
VGG16-Net Encoder	93	92	93	46.8	0.23	14.7 M
DeepResUnet	91	92.8	91.9	2520	0.069	2.79 M
TCU-Net	-	97.5	96.6	88.8	14.63	1.72 M
AD-HRNet	85.3	88.18	85	-	0.052	8.73 M
ICTANet	-	92	93.4	138	7.24	14.02 M
AttrousDenseUDconvNet	88.3	86.2	89	-	1.574	18.35 M

IV. CONCLUSION

By analyzing different architectures and approaches to classifying vegetation land cover in high-resolution imagery using neural networks, this study explores the challenges associated with it. With a precision of 99.7%, an accuracy of 83.3%, and an F1-score of 70.3%, the proposed models demonstrate excellent performance, as shown by the study, through various testing scenarios. These results demonstrate the benefits of deeper networks and the use of threshold classifiers for effectively classifying the pixels of high-resolution images, while maintaining a competitive training duration of 12 to 20 minutes. Finally, this study presents insightful information to improve classification methods for environmental management, highlighting the need to conduct classifier selection and to upgrade model architecture for performance improvement.

ACKNOWLEDGMENT

This research would not be possible without the support of the Intelligent System Laboratory and the Master Program of Computer Science, UIN Maulana Malik Ibrahim Malang, to provide data access and other necessary resources and computing facilities.

REFERENCES

[1] D. Lawrence et al., "The unseen effects of deforestation: Biophysical effects on climate," Front. For. Glob. Change, vol. 5, pp. 1-13, Mar. 2022, doi: 10.3389/ffgc.2022.756115.

[2] M. Leon et al., "Effect of deforestation on climate change: A co-integration and causality approach with time series," Sustainability, vol. 14, no. 18, Sep. 2022, doi: 10.3390/su141811303.

[3] Badan Pusat Statistik (BPS), "Deforestation rate (netto) in Indonesia, inside and outside forest area 2013-2022 (Ha/year)," 2022. [Online]. Available: https://www.bps.go.id/en/statistics-table/1/MjA4MSMx/deforestation-rate--netto--in-indonesia--inside-and-outside-forest-area-2013-2022--ha-year-.html.

[4] C. Ramadhan, R. Dina, and E. Nurjani, "Spatial and temporal based deforestation proclivity analysis on flood events with applying watershed scale (case study: Lasolo watershed in Southeast Sulawesi, Central Sulawesi, and South Sulawesi, Indonesia)," Int. J. Disaster Risk Reduct., vol. 93, Jul. 2023, doi: 10.1016/j.ijdrr.2023.103745.

[5] K. Abbass et al., "A review of the global climate change impacts, adaptation, and sustainable mitigation measures," Environ. Sci. Pollut. Res., vol. 29, pp. 42539-42559, Jun. 2022, doi: 10.1007/s11356-022-19718-6.

[6] D. Bhatt et al., "CNN variants for computer vision: History, architecture, application, challenges and future scope," Electronics, vol. 10, no. 20, Oct. 2021, doi:10.3390/electronics10202470.

[7] V. S. Venkat and M. Lokanath, "Single image super resolution using deep convolutional neural networks," in *Proc. 2018 3rd IEEE Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT)*, May 2018, pp. 1250-1258, doi: 10.1109/RTEICT42901.2018.9012445.

[8] X. Zhao et al., "A review of convolutional neural networks in computer vision," Artif. Intell. Rev., vol. 57, no. 4, Apr. 2024, doi:10.1007/s10462-024-10721-6.

[9] M. Xin and Y. Wang, "Research on image classification model based on deep convolution neural network," EURASIP J. Image Video Process., vol. 2019, no. 1, Dec. 2019, doi: 10.1186/s13640-019-0417-8.

[10] S. Minaee et al., "Image segmentation using deep learning: A survey," IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 7, pp. 3523-3542, Jul. 2022, doi: 10.1109/TPAMI.2021.3059968.

[11] L. Alzubaidi et al., "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," J. Big Data, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[12] A. Khan et al., "A survey of the recent architectures of deep convolutional neural networks," Artif. Intell. Rev., vol. 53, no. 8, pp. 5455-5516, Dec. 2020, doi: 10.1007/s10462-020-09825-6.

[13] T. Kattenborn et al., "Review on convolutional neural networks (CNN) in vegetation remote sensing," ISPRS J. Photogramm. Remote Sens., vol. 173, pp. 24-49, Mar. 2021, doi: 10.1016/j.isprsjprs.2020.12.010.

[14] D. J. Lary et al., "Machine learning in geosciences and remote sensing," Geosci. Front., vol. 7, no. 1, pp. 3-10, Jan. 2016, doi:10.1016/j.gsf.2015.07.003.

[15] J. Jagannathan and C. Divya, "Deep learning for the prediction and classification of land use and land cover changes using deep convolutional neural network," Ecol. Inform., vol. 65, Nov. 2021, doi:10.1016/j.ecoinf.2021.101412.

[16] C. Crysdian, "Performance measurement without ground truth to achieve optimal edge," Int. J. Image Data Fusion, vol. 9, no. 2, pp. 170-193, Apr. 2018, doi: 10.1080/19479832.2017.1384764.

[17] C. Crysdian, "The evaluation of entropy-based algorithm towards the production of closed-loop edge," *JOIV Int. J. Inf. Vis.*, vol. 7, no. 4, pp. 481-488, Dec. 2023, doi: 10.62527/joiv.7.4.1727.

[18] R. W. Sussman, G. M. Green, and L. K. Sussman, "Satellite imagery, human ecology, anthropology, and deforestation in Madagascar," *Hum. Ecol.*, vol. 22, no. 3, pp. 333-354, 1994, doi:10.1007/BF02168856.

[19] C. Yang et al., "Using high-resolution airborne and satellite imagery to assess crop growth and yield variability for precision agriculture," *Proc. IEEE*, vol. 101, no. 3, pp. 582-592, Mar. 2013, doi:10.1109/JPROC.2012.2196249.

[20] A. M. Eldosouky et al., "Integration of ASTER satellite imagery and 3D inversion of aeromagnetic data for deep mineral exploration," *Adv. Space Res.*, vol. 68, no. 9, pp. 3641-3662, Nov. 2021, doi:10.1016/j.asr.2021.07.016.

[21] Z. Si et al., "Photovoltaic power forecast based on satellite images considering effects of solar position," *Appl. Energy*, vol. 302, Nov. 2021, doi: 10.1016/j.apenergy.2021.117514.

[22] A. Mujetahid, M. Nursaputra, and A. S. Soma, "Monitoring illegal logging using Google Earth Engine in Sulawesi Selatan tropical forest, Indonesia," *Forests*, vol. 14, no. 3, Mar. 2023, doi:10.3390/f14030652.

[23] S. Timilsina, J. Aryal, and J. B. Kirkpatrick, "Mapping urban tree cover changes using object-based convolution neural network (OB-CNN)," *Remote Sens.*, vol. 12, no. 18, Sep. 2020, doi:10.3390/rs12183017.

[24] S. Aldiansyah and R. A. Saputra, "Comparison of machine learning algorithms for land use and land cover analysis using Google Earth Engine (case study: Wanggu watershed)," *Int. J. Remote Sens. Earth Sci.*, vol. 19, no. 2, pp. 197-210, Dec. 2022, doi:10.30536/j.ijreses.2022.v19.a3803.

[25] Y. G. Yuh et al., "Application of machine learning approaches for land cover monitoring in northern Cameroon," *Ecol. Inform.*, vol. 74, May 2023, doi: 10.1016/j.ecoinf.2022.101955.

[26] A. Abdalla et al., "Fine-tuning convolutional neural network with transfer learning for semantic segmentation of ground-level oilseed rape images in a field with high weed pressure," *Comput. Electron. Agric.*, vol. 167, Dec. 2019, doi:10.1016/j.compag.2019.105091.

[27] Y. Yi et al., "Semantic segmentation of urban buildings from VHR remote sensing imagery using a deep convolutional neural network," *Remote Sens.*, vol. 11, no. 15, Aug. 2019, doi:10.3390/rs11151774.

[28] X. Xiong et al., "TCUNet: A lightweight dual-branch parallel network for sea-land segmentation in remote sensing images," *Remote Sens.*, vol. 15, no. 18, Sep. 2023, doi: 10.3390/rs15184413.

[29] X. Yang et al., "Semantic segmentation for remote sensing images based on an AD-HRNet model," *Int. J. Digit. Earth*, vol. 15, no. 1, pp. 2376-2399, 2022, doi: 10.1080/17538947.2022.2159080.

[30] X. Yu, S. Li, and Y. Zhang, "Incorporating convolutional and transformer architectures to enhance semantic segmentation of fine-resolution urban images," *Eur. J. Remote Sens.*, vol. 57, no. 1, 2024, doi: 10.1080/22797254.2024.2361768.

[31] Z. Su et al., "An improved U-Net method for the semantic segmentation of remote sensing images," *Appl. Intell.*, vol. 52, no. 3, pp. 3276-3288, Feb. 2022, doi: 10.1007/s10489-021-02542-9.

[32] C. Dechesne, P. Lassalle, and S. Lefèvre, "Bayesian U-Net: Estimating uncertainty in semantic segmentation of earth observation images," *Remote Sens.*, vol. 13, no. 19, Oct. 2021, doi:10.3390/rs13193836.

[33] X. Pei et al., "Robustness of machine learning to color, size change, normalization, and image enhancement on micrograph datasets with large sample differences," *Mater. Des.*, vol. 232, Aug. 2023, doi:10.1016/j.matdes.2023.112086.

[34] S. S. Bhairannawar et al., "FPGA based efficient multiplier for image processing applications using recursive error free Mitchell log multiplier and KOM architecture," *Int. J. VLSI Des. Commun. Syst.*, vol. 5, no. 3, pp. 93-114, Jun. 2014, doi: 10.5121/vlsic.2014.5309.