International Journal of Electrical and Computer Engineering (IJECE)
Vol. 15, No. 4, August 2025, pp. 3876~3887
ISSN: 2088-8708, DOI: 10.11591/ijece.v15i4.pp3876-3887

Optimizing convolutional neural network hyperparameters to enhance liver segmentation accuracy in medical imaging

Iwan Purnama[1], Agus Perdana Windarto[2], Solikhun[3]
[1]Information Technology Study Program, Faculty of Science and Technology, Universitas Labuhanbatu, Rantauprapat, Indonesia
[2]Master's Program, Informatics Study Program, STIKOM Tunas Bangsa, Pematangsiantar, Indonesia
[3]Informatics Engineering Program, STIKOM Tunas Bangsa, Pematangsiantar, Indonesia

Article Info

Article history:

Received Jun 16, 2024
Revised Mar 21, 2025
Accepted Mar 23, 2025

Keywords:

Convolutional neural network
DeepLabV3Plus
Hyperparameter optimization
Liver segmentation
ResNet

ABSTRACT

Liver segmentation in medical imaging is a crucial step in various clinical applications, such as disease diagnosis, surgical planning, and evaluation of response to therapy, which require a high degree of precision for accurate results. This research focuses on increasing the accuracy of liver segmentation by optimizing hyperparameters in the convolutional neural network (CNN) model using the developed ResNet architecture. The uniqueness of this research lies in the application of hyperparameter optimization methods such as random search and Bayesian optimization, which allow broader and more efficient exploration than conventional approaches. The results show that the DeepLabV3Plus model (the proposed model) significantly outperforms the standard ResNet in the image segmentation task. DeepLabV3Plus shows excellent performance with an MIoU score of 0.965, a PA Score of 0.929, and a meager loss value of 0.011. These results show that DeepLabV3Plus is able to recognize and predict segmentation areas very accurately and consistently and minimize prediction errors effectively. In conclusion, the results of this study show a significant improvement in segmentation accuracy, with the optimized model providing better performance in the evaluation.

Corresponding Author:

Iwan Purnama
Information Technology Study Program, Faculty of Science and Technology, Universitas Labuhanbatu
Street Sisingamangaraja No.126 A KM 3.5 Aek Tapa, Bakaran Batu, Subdistrict Rantau Sel., Regency Labuhanbatu, North Sumatra 21418, Indonesia
Email: iwanpurnama2014@gmail.com

1. INTRODUCTION

Liver segmentation in medical imaging plays a vital role in various clinical applications, including diagnosis of liver diseases, surgical planning, and monitoring response to therapy [1]–[3]. Accurate segmentation of this organ is critical because errors in liver identification and delineation can result in incorrect diagnosis or inappropriate surgical planning, which in turn can harm the patient. Therefore, reliable and accurate segmentation methods are needed to support appropriate clinical decisions [4], [5].

In recent years, convolutional neural networks (CNN) have emerged as a potent tool in medical image segmentation tasks [6]. CNNs are capable of capturing complex features from medical images and have demonstrated excellent performance in various segmentation tasks. One of the famous and influential CNN architectures for medical image segmentation is ResNet, which is specifically designed to address challenges in image segmentation with high resolution and complex details [7]–[10].

Previous literature studies conducted by [11] in this study showed that the three CNNs achieved an average Dice score above 90% in liver segmentation with contrast computed tomography (CT) images of ordinary quality. However, the results also showed that the three CNNs had decreased performance in liver segmentation in CT images with low doses and no contrast. This suggests that liver segmentation on low-dose, no-contrast CT images is still a challenging task and requires further improvement before it can be applied in clinical practice. Meanwhile, in the following study conducted by [12], the results of research on liver tumor segmentation were based on 3D CNN with multiple scales. This study aims to overcome the difficulties of liver tumor segmentation from CT images, which include low contrast between liver tumors and healthy tissue, as well as the complex shape, size, and location of liver tumor images. The method developed is a CNN-based liver and liver tumor segmentation algorithm, which is equipped with a three-dimensional dual pathway (TDP-CNN) to maintain a balance of segmentation performance and computing resource requirements. Experimental results show that the proposed algorithm performs well in liver and liver tumor segmentation. However, some improvements can be made, especially in the segmentation of small liver tumors. In addition, the use of fully connected conditional random fields (FC-CRF) helps improve liver tumor segmentation results. This method showed better results than several other liver and liver tumor segmentation algorithms in the 2017 MICCAI competition.

From the previous research that has been described, the performance of the CNN model is very dependent on choosing the correct hyperparameters, such as learning rate, batch size, number of filters, and kernel size [13], [14]. Hyperparameters that are not optimal can cause the model to experience overfitting or underfitting, thereby reducing segmentation accuracy [15], [16]. Therefore, hyperparameter optimization is a crucial step in developing more accurate and reliable CNN models. Hyperparameter optimization methods such as grid search [17]–[19], and Bayesian optimization have been proposed to find optimal hyperparameter combinations [20]–[23]. Grid search explores the entire hyperparameter space systematically but often requires a very long time and significant computational resources. In contrast, random search and Bayesian optimization offer a more efficient approach by randomly exploring the hyperparameter space or using probabilistic models to predict the most promising hyperparameter combinations [24].

Based on the literature studies that have been described, this research focuses on the application of hyperparameter optimization methods to increase the accuracy of liver segmentation using a CNN model with Resnet architecture [25]–[28]. By using random search and Bayesian optimization methods, this research aims to find the optimal combination of hyperparameters that can significantly improve model performance. The results of this research are expected to make a significant contribution to the field of medical image segmentation and improve clinical applications in diagnosis and patient treatment planning [12], [29], [30]. Effective hyperparameter optimization improves segmentation accuracy and the model's ability to generalize to new data, thereby ensuring its use in a variety of clinical conditions [31], [32]. Thus, this research focuses not only on improving the model's performance but also on its practical applicability in real medical environments [33].

2. METHOD

This research was conducted to improve the accuracy of liver segmentation in medical images by analyzing and optimizing CNN hyperparameters. The segmentation results obtained provide valuable insights into the effectiveness of various CNN model optimization techniques. These findings can improve our understanding of how to effectively tune CNN hyperparameters to improve segmentation accuracy in the context of medical images.

2.1. Data collection method

In an effort to obtain relevant data and information, the method used in data collection is open-source data, which is available on the *Kaggle.com* website. The availability of these resources allows researchers to explore relevant data and build research series that suit their needs. The data source from *Kaggle.com* is one of the key aspects in obtaining a quality dataset for this research.

2.2. Flowchart of CNN model optimization

In this research, CNN were trained using different optimizers and different batch sizes for comparison. The authors adopted the approach of using a typical deep learning framework such as TensorFlow. By exploiting the advantages of this framework, they can manage and train CNN models efficiently. To provide a clear visual representation of the optimization flow of the CNN model used in this research, we present a detailed flow diagram in Figure 1.

Figure 1 displays the steps involved in the CNN model optimization process, from the data collection stage to the evaluation of the results. This diagram helps readers easily understand the steps involved in training and optimizing a CNN model and how changing the optimizer and batch size affects the model performance. Thus, it is a useful visual guide to clarify the experimental process and findings in this research.

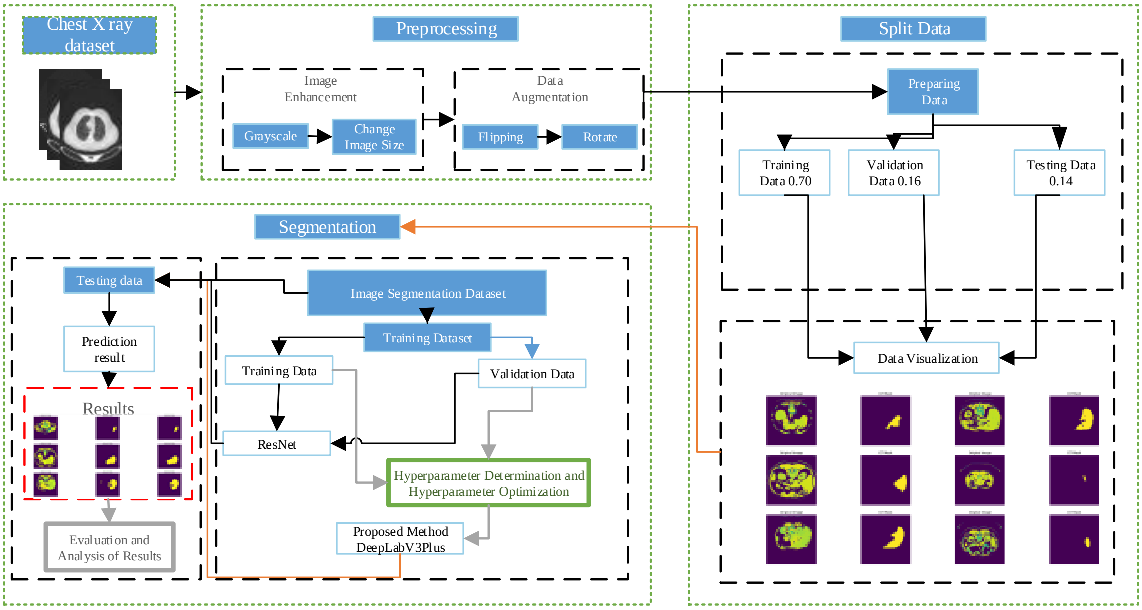


Figure 1. Model optimization flowchart

Figure 1 is the flowchart above depicting the workflow in optimizing the CNN model. The following is a detailed explanation of each step in the flowchart, starting with the data collection process that will be used for model training. The next stage is data preprocessing, including normalization, scaling, and removal of irrelevant or missing data. After that, data augmentation is carried out to increase and enrich the dataset with variations such as rotation, shift, and mirroring. The data was then transformed and divided into three parts: 70% for model training, 16% for model validation during training to avoid overfitting, and 14% for follow-up training. Next, the CNN network architecture that will be used is designed, followed by determining and optimizing the model hyperparameters, such as number of layers, batch size, and learning rate. The training dataset is used to train the CNN model; then, the model results are evaluated and analyzed using validation data to measure model performance. Optionally, external validation can be performed using a different or additional dataset to ensure the generalization of the model. The final stage is the discussion and conclusion of the results, where this process ensures that the resulting model has optimal and reliable performance. This flowchart shows a systematic approach to building and optimizing a CNN model, from data collection to evaluating and validating the final results. This comprehensive approach ensures that the resulting model has optimal and reliable performance.

2.3. Optimized CNN architecture

The CNN architecture in this research uses the DeepLabV3Plus model. DeepLabV3Plus is a compelling and complex model used for image segmentation. It combines advanced features from several existing architectures, such as ResNet, and is equipped with several unique techniques to improve accuracy and performance. Figure 2 shows the optimization model.

Figure 2 In general, the DeepLabV3Plus model has the following architecture for its convolutional part (CNN):

a. ResNet Backbone: This part is the essential part that provides input image features. DeepLabV3Plus uses ResNet as a backbone, which has a deep structure with Residual blocks that enables better learning even for intense networks.
b. Atrous spatial pyramid collection (ASPP): This is an essential part of the DeepLab model. ASPP allows the network to expand its visual range without increasing the number of parameters too much. This allows the network to obtain broader contextual information, which is helpful in image segmentation tasks.
c. Decoder (DeepLabV3Plus): Once important features are obtained from ASPP, the Decoder functions to refine and recover data to produce more precise segmentation. In DeepLabV3Plus, there are several mechanisms, such as skip connections, that help strengthen the representation. Output layer: Finally, the model produces output in the form of the expected pixel segmentation, which corresponds to the number of classes specified in the class. Overall, DeepLabV3Plus is an intense and complex model that leverages advanced technologies in CNNs for image segmentation tasks.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 112, 112]           3,136
       BatchNorm2d-2         [-1, 64, 112, 112]             128
              ReLU-3         [-1, 64, 112, 112]               0
         MaxPool2d-4           [-1, 64, 56, 56]               0
            Conv2d-5           [-1, 64, 56, 56]          36,864
      BatchNorm2d-20           [-1, 64, 56, 56]             128
             ReLU-21           [-1, 64, 56, 56]               0
           Conv2d-22           [-1, 64, 56, 56]          36,864
      BatchNorm2d-23           [-1, 64, 56, 56]             128
             ReLU-24           [-1, 64, 56, 56]               0
       BasicBlock-99          [-1, 256, 14, 14]               0
          Conv2d-116          [-1, 512, 14, 14]       2,359,296
     BatchNorm2d-117          [-1, 512, 14, 14]           1,024
            ReLU-118          [-1, 512, 14, 14]               0
          Conv2d-119          [-1, 512, 14, 14]       2,359,296
     BatchNorm2d-120          [-1, 512, 14, 14]           1,024
            ReLU-121          [-1, 512, 14, 14]               0
      BasicBlock-122          [-1, 512, 14, 14]               0
  ResNetEncoder-123  [[-1, 1, 224, 224], [-1, 64, 112, 112], [-1, 64, 56, 56], [-1, 128, 28, 28], [-1, 256, 14, 14], [-1, 512, 14, 14]]
          Conv2d-124          [-1, 256, 14, 14]         131,072
     BatchNorm2d-125          [-1, 256, 14, 14]             512
            ReLU-126          [-1, 256, 14, 14]               0
          Conv2d-127          [-1, 512, 14, 14]           4,608
          Conv2d-128          [-1, 256, 14, 14]         131,072
     BatchNorm2d-129          [-1, 256, 14, 14]             512
            ReLU-130          [-1, 256, 14, 14]               0
          Conv2d-131          [-1, 512, 14, 14]           4,608
          Conv2d-132          [-1, 256, 14, 14]         131,072
     BatchNorm2d-133          [-1, 256, 14, 14]             512
            ReLU-134          [-1, 256, 14, 14]               0
          Conv2d-135          [-1, 512, 14, 14]           4,608
          Conv2d-136          [-1, 256, 14, 14]         131,072
     BatchNorm2d-137          [-1, 256, 14, 14]             512
            ReLU-138          [-1, 256, 14, 14]               0
AdaptiveAvgPool2d-139            [-1, 512, 1, 1]               0
          Conv2d-140            [-1, 256, 1, 1]         131,072
     BatchNorm2d-141            [-1, 256, 1, 1]             512
            ReLU-142            [-1, 256, 1, 1]               0
          Conv2d-143          [-1, 256, 14, 14]         327,680
     BatchNorm2d-144          [-1, 256, 14, 14]             512
            ReLU-145          [-1, 256, 14, 14]               0
         Dropout-146          [-1, 256, 14, 14]               0
            ASPP-147          [-1, 256, 14, 14]               0
          Conv2d-148          [-1, 256, 14, 14]           2,304
          Conv2d-149          [-1, 256, 14, 14]          65,536
     BatchNorm2d-150          [-1, 256, 14, 14]             512
            ReLU-151          [-1, 256, 14, 14]               0
UpsamplingBilinear2d-152          [-1, 256, 56, 56]               0
          Conv2d-153           [-1, 48, 56, 56]           3,072
     BatchNorm2d-154           [-1, 48, 56, 56]              96
            ReLU-155           [-1, 48, 56, 56]               0
          Conv2d-156          [-1, 304, 56, 56]           2,736
          Conv2d-157          [-1, 256, 56, 56]          77,824
     BatchNorm2d-158          [-1, 256, 56, 56]             512
            ReLU-159          [-1, 256, 56, 56]               0
DeepLabV3PlusDecoder-160          [-1, 256, 56, 56]               0
          Conv2d-161            [-1, 2, 56, 56]             514
UpsamplingBilinear2d-162          [-1, 2, 224, 224]               0
        Identity-163          [-1, 2, 224, 224]               0
      Activation-164          [-1, 2, 224, 224]               0
================================================================
Total params: 22,431,442
Trainable params: 22,431,442
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 163.52
Params size (MB): 85.57
Estimated Total Size (MB): 249.28
```

Figure 2. Proposed model optimization (DeepLabV3Plus)

3. RESULTS AND DISCUSSION

Based on the training results, validation results, and testing results, analysis was carried out using various hyperparameters and different architectures to see the image segmentation results in liver image segmentation. This analysis aims to identify the best configuration that produces the most accurate and consistent segmentation. By comparing the performance of each model through relevant evaluation metrics, such as accuracy, precision, and recall, it can be determined which method provides optimal results in the task of liver image segmentation. The results of these experiments will provide valuable insights into the effectiveness of various approaches in processing and analyzing medical images and lead to improvements in the performance of segmentation models in the future.

3.1. Result

The model training results in the article include some important information explaining the performance of the optimized CNN model for liver segmentation in 3D medical images using the DeepLabV3Plus model as the proposed model and ResNet. In Figure 3 (see in appendix), the following is an explanation of the results for each epoch: You can see the training process presented in Figure 3. In the training process shown in Figure 3, it can be seen that after the 8th epoch, training is stopped. This is caused by the loss value not decreasing for five consecutive epochs, which indicates that the model has reached convergence or there is a possibility that the model is overfitting. The total training time for DeepLabV3Plus is 280,596 minutes, while for the other models, it only stops at epoch 5 with a time of 90 minutes. This difference in training time is quite significant and shows the existence of time inefficiencies in DeepLabV3Plus training, which takes much longer. However, it has excellent prediction accuracy results compared to the standard ResNet model. Time efficiency is an essential aspect of the model training process, especially in contexts that require significant computing resources and a long time. Excessive, time-consuming training not only increases operational costs but can also slow down the model development iteration cycle. Therefore, finding a balance between model quality and training time efficiency is crucial.

However, in contrast to DeepLabV3Plus, the ResNet model only requires five epochs before training is stopped. However, the training results from ResNet could be more satisfactory, indicating that this model has not been able to produce the expected performance in a shorter training time. This suggests that although ResNet is more efficient in training time, the quality of the results obtained is different from DeepLabV3Plus. In order to increase the efficiency and effectiveness of the training process, consider the use of techniques such as more adaptive early stopping, the use of regularization to reduce overfitting, or even the exploration of more efficient but still high-performance model architectures. Through this approach, an optimal balance can be achieved between training time and the quality of the results obtained. In Figure 4, we can see the mIOU, PA score, and loss curves on Figure 4(a) DeepLabV3Plus and Figure 4(b) ResNet.

Well, here is a comprehensive description of the training results of two models, DeepLabV3Plus and ResNet, based on the curves shown in the figure.

a. DeepLabV3Plus (a)

– Mean intersection over union (IoU) learning curve:

The training curve shows a significant increase in the initial epoch and stabilizes after the 3rd epoch. The validation curve follows the training curve with little difference, indicating that the model does not suffer from significant overfitting. After the 8th epoch, training was stopped because the loss value did not decrease for five consecutive epochs, indicating that the model had reached convergence.

– Pixel accuracy (PA) learning curve:

The training curve shows a significant increase in the first two epochs, then stabilizes until the 8^{th} epoch. The validation curve shows a substantial and stable initial increase after the 2^{nd} epoch. Close training and validation curves indicate the model has consistent performance between training and validation data.

– Loss learning curve:

The training loss curve shows a drastic decrease in the first two epochs and starts to stabilize until the 8^{th} epoch. The validation loss curve also shows a significant reduction in the first two epochs and stabilizes after that. The stability of the validation loss curve shows that the model does not experience substantial overfitting.

b. ResNet (b)

– Mean intersection over union (IoU) learning curve:

The training curve shows stable and high IoU values throughout five epochs. The validation curve shows a drastic decline after the first epoch and remains low, indicating the model cannot generalize the validation data well. The low validation curve suggests this model may be overfitting the training data.

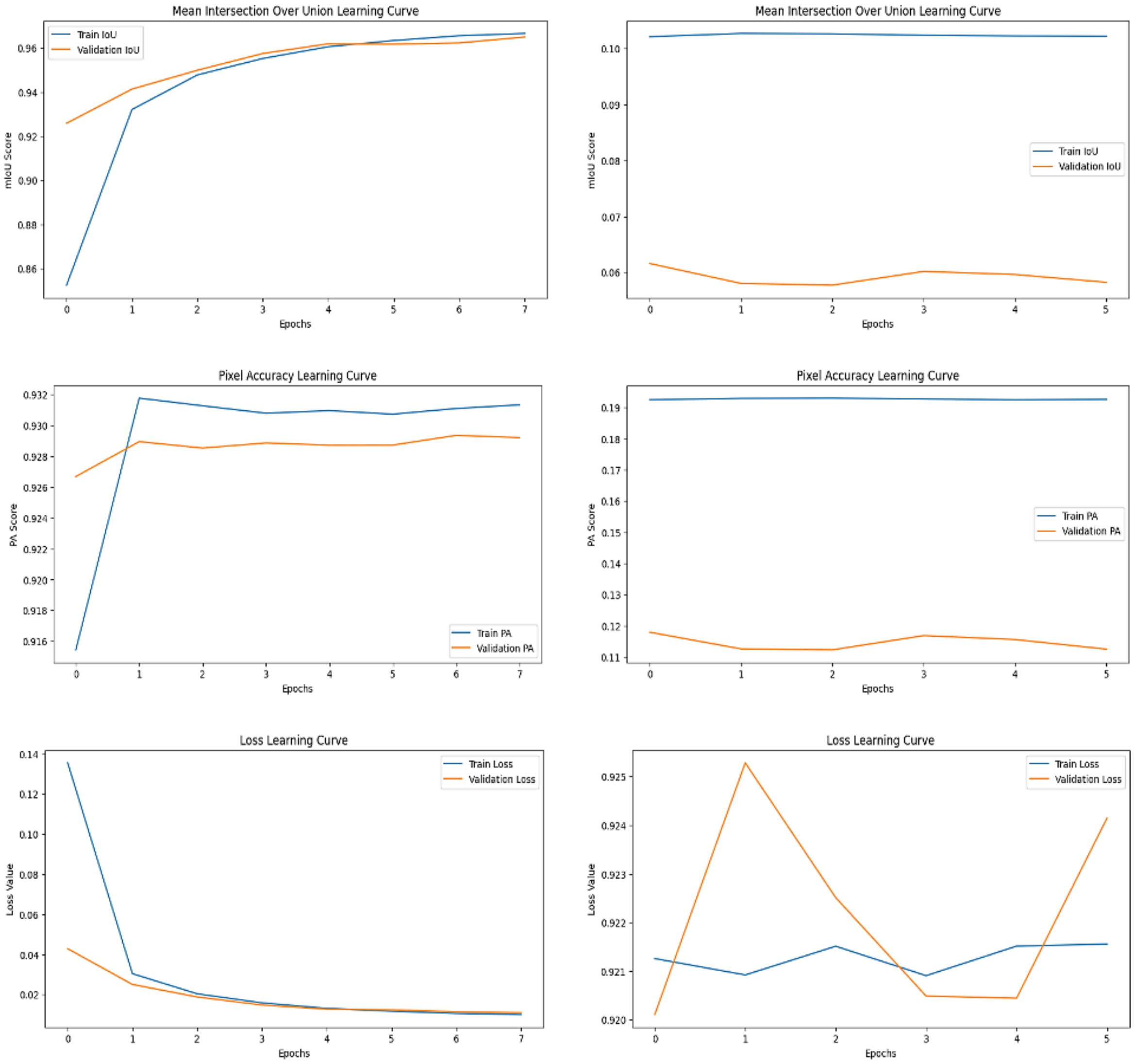
– Pixel accuracy (PA) learning curve:

The training curve shows high and stable PA values throughout the five epochs. The validation curve significantly decreases after the first epoch and remains low. Like the IoU curve, the model is experiencing overfitting and cannot generalize well.

– Loss learning curve:

The training loss curve shows stable and relatively low values throughout the five epochs. The validation loss curve shows large fluctuations, with higher values than the training curve. Fluctuations and high validation loss values indicate the model is unstable and unable to generalize on validation data.

DeepLabV3Plus shows better results compared to ResNet. This model shows good stability and generalization ability, as seen from the training and validation curves, which are almost parallel and stable. Even though it requires longer training time (280,596 minutes), the results are more consistent and reliable. ResNet, however, shows poor performance on validation data, with strong indications of overfitting. Despite the much shorter training time (90 minutes), the results obtained could have been better and more reliable for predictions on new data.



(a) (b)

Figure 4. Graph of mean intersection over union learning curve, pixel accuracy learning curve, and loss learning curve CXR image segmentation training process using architecture: (a) DeepLabV3Plus and (b) ResNet

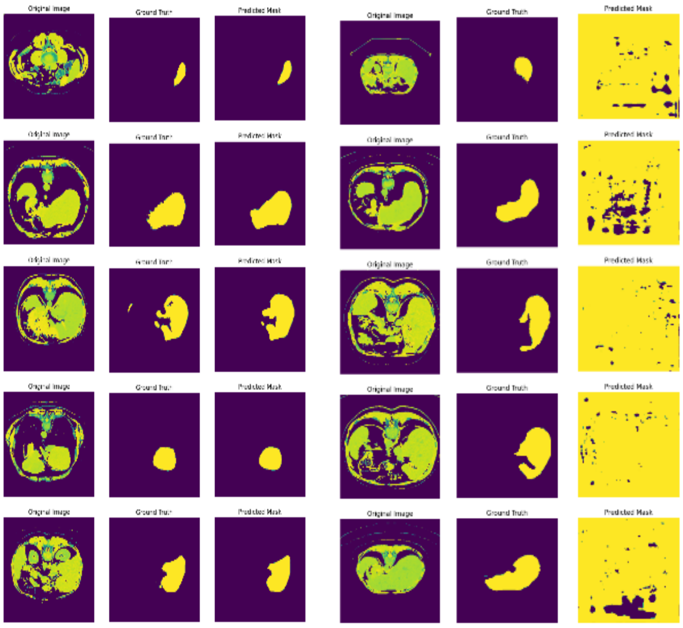
3.2. Discussion

In this section, we will analyze and discuss the research results obtained from training two models, DeepLabV3Plus and ResNet. This analysis includes evaluating the performance of both models based on the resulting learning curves, including mean intersection over union (IoU), pixel accuracy (PA), and loss. This discussion will also include a comparison between the efficiency of training time and the quality of results obtained from both models. By understanding the differences and advantages of each model, we can draw more precise conclusions about which model is more effective to use in a specific application context. Next, we will also discuss the implications of these results for future use of computational resources and model development strategies. In Figure 5, the prediction results of the following two models are compared on Figure 5(a) DeepLabV3Plus and Figure 5(b) ResNet.

In Figure 5, DeepLabV3Plus: This model shows superior performance in image segmentation. The prediction results are very close to the ground truth mask, demonstrating the ability of this model to recognize and delineate objects in images with high precision. DeepLabV3Plus succeeded in capturing the details and structure of objects very well, producing accurate segmentation. ResNet: This model shows abysmal performance in image segmentation tasks. The prediction results are far from the ground truth mask and show many errors. ResNet appears to need to improve in recognizing and delineating objects in images, resulting in inaccurate and noisy segmentation. Comparison: The performance difference between

DeepLabV3Plus and ResNet is very significant. DeepLabV3Plus is more effective in segmenting images compared to ResNet. Previous learning curve analysis shows that DeepLabV3Plus has better generalization capabilities and does not experience overfitting like ResNet. These results emphasize the importance of choosing a suitable model for image segmentation tasks, where DeepLabV3Plus is proven more reliable and effective than ResNet.

To provide a clearer picture of the performance of the two models that have been tested in this research, below we present a table of evaluation results which includes three main metrics: mean intersection over union (MIoU) score, pixel accuracy (PA) score, and loss value. These metrics provide a comprehensive view of how well each model performs image segmentation. MIoU Score is a metric used to measure how well a model predicts segmentation by comparing the overlap area between predictions and ground truth. PA Score measures pixel accuracy in segmentation, namely the percentage of correctly classified pixels. Loss Value shows how well the model minimizes prediction errors during training. Table 1 shows the performance comparison results between the proposed model (DeepLabV3Plus) and the ResNet model. The data in this table shows significant differences in segmentation capabilities between the two models, which will be discussed further in the analysis and discussion section.



(a) (b)

Figure 5. Inference and performance analysis of AI models with GradCAM (a) DeepLabV3Plus and (b) ResNet

Table 1. Comparison of DeepLabV3Plus and ResNet analysis

Method	MIoU score	PA score	Loss value
Proposed model (DeepLabV3Plus)	0.965	0.929	0.011
ResNet	0.060	0.117	0.921

Table 1 research results show that the DeepLabV3Plus model is significantly superior to ResNet in the image segmentation task. DeepLabV3Plus achieved an MIoU score of 0.965 and a PA score of 0.929, much higher than ResNet, which only achieved an MIoU score of 0.060 and a PA Score of 0.117. In addition, the loss value of DeepLabV3Plus is very low (0.011) compared to ResNet (0.921), indicating that DeepLabV3Plus is more effective in minimizing prediction errors. In conclusion, DeepLabV3Plus is more accurate and efficient for image segmentation, while ResNet shows abysmal performance in this context.

Although this study successfully improved the liver segmentation accuracy using the DeepLabV3Plus model with hyperparameter optimization, there are still some unanswered questions and opportunities for future research. One aspect that needs to be further explored is the generalization of the model to various variations of medical images, such as low-quality or high-noise images. In addition, adapting this model to segment other organs or in multi-organ scenarios could be a promising research direction. The use of more sophisticated data augmentation techniques and few-shot or self-supervised learning-based approaches can also be investigated to further improve the model performance without requiring large amounts of training data.

4. CONCLUSION

Based on the research results, the DeepLabV3Plus model is significantly superior to ResNet in image segmentation tasks. DeepLabV3Plus achieved an MIoU score of 0.965, a PA score of 0.929, and a meager loss value of 0.011. This shows its ability to recognize and predict segmentation areas very accurately and consistently, as well as minimize prediction errors effectively. In contrast, ResNet shows abysmal performance with an MIoU score of 0.060, a PA score of 0.117, and a high loss value of 0.921, indicating its incompetence in image segmentation tasks. This significant difference confirms that DeepLabV3Plus is a more effective and efficient choice for applications that require precise image segmentation.

FUNDING INFORMATION

The authors declare that no funding was received for this study.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Iwan Purnama	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Agus Perdana Windarto	✓	✓				✓		✓	✓	✓	✓	✓		
Solikhun	✓		✓	✓			✓			✓	✓		✓	

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available from the author, Kaggle, upon reasonable request.

REFERENCES

[1] S. Arora, S. K. Tripathy, R. Gupta, and R. Srivastava, “Exploiting multimodal CNN architecture for automated teeth segmentation on dental panoramic X-ray images,” in *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, Mar. 2023, vol. 237, no. 3, pp. 395–405. doi: 10.1177/09544119231157137.

[2] X. Meng, Z. Xing, and S. Lu, “A deep learning-based lung nodule density classification and segmentation method and its effectiveness under different CT reconstruction algorithms,” *National Medical Journal of China*, vol. 101, no. 7, pp. 476–480, 2021, doi: 10.3760/cma.j.cn112137-20201123-03171.

[3] S. Nagoor and S. V. Jinny, “A dual fuzzy with hybrid deep learning architecture based on CNN with hybrid metaheuristic algorithm for effective segmentation and classification,” *International Journal of Information Technology*, vol. 15, no. 1, pp. 531–543, Jan. 2023, doi: 10.1007/s41870-022-01106-5.

[4] J. Amose, P. Manimegalai, and R. Pon Selchiya, “Classification of adventitious lung sounds: Wheeze, crackle using machine learning techniques,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, no. 3, pp. 1143–1152, 2023.

[5] D. W. Kim, M. S. Kim, J. Lee, and P. Park, “Adaptive learning-rate backpropagation neural network algorithm based on the minimization of mean-square deviation for impulsive noises,” *IEEE Access*, vol. 8, pp. 98018–98026, 2020, doi: 10.1109/ACCESS.2020.2997010.

[6] H. Listiyono, Z. Budiarso, and A. P. Windarto, “An optimized CNN architecture for accurate 3D liver segmentation in medical images,” *Journal of Image and Graphics*, vol. 13, no. 1, pp. 15–23, 2025, doi: 10.18178/joig.13.1.15-23.

[7] P. Alkhairi, W. Wanayumini, and B. H. Hayadi, “Analysis of the adaptive learning rate and momentum effects on prediction problems in increasing the training time of the backpropagation algorithm,” in *AIP Conference Proceedings*, 2024, p. 20049. doi: 10.1063/5.0203374.

[8] K. H. Niralgikar and M. A. Bulsara, “Frequent pattern data mining based clustering and classification using fuzzy back propagation resnet convolutional networks,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 10, no. 2s, pp. 201–204, 2022.

[9] L. I. Kesuma, “Elrei: Ensemble learning of ResNet, efficientnet, and Inception-V3 for lung disease classification based on chest X-Ray image,” *International Journal of Intelligent Engineering and Systems*, vol. 16, no. 5, pp. 149–161, Oct. 2023, doi: 10.22266/ijies2023.1031.14.

[10] S. Shivadekar, B. Kataria, S. Hundekari, K. Wanjale, V. P. Balpande, and R. Suryawanshi, “Deep learning based image classification of lungs radiography for detecting COVID-19 using a deep CNN and ResNet 50,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, no. 1s, pp. 241–250, 2023.

[11] H. S. Hoang, C. Phuong Pham, D. Franklin, T. Van Walsum, and M. Ha Luu, “An evaluation of CNN-based liver segmentation methods using multi-types of CT abdominal images from multiple medical centers,” in *Proceedings - 2019 19th International Symposium on Communications and Information Technologies, ISCIT 2019*, 2019, pp. 20–25. doi: 10.1109/ISCIT.2019.8905166.

[12] L. Meng, Y. Tian, and S. Bu, “Liver tumor segmentation based on 3D convolutional neural network with dual scale,” *Journal of Applied Clinical Medical Physics*, vol. 21, no. 1, pp. 144–157, Jan. 2020, doi: 10.1002/acm2.12784.

[13] D. Ruan, F. Zhang, L. Zhang, and J. Yan, “Optimal modifications in CNN for bearing fault classification and adaptation across different working conditions,” *Journal of Vibration Engineering & Technologies*, vol. 12, no. 3, pp. 4075–4095, Mar. 2024, doi: 10.1007/s42417-023-01106-0.

[14] K. M. Sunnetci, “Face Mask Detection Using GoogLeNet CNN-Based SVM Classifiers,” *Gazi University Journal of Science*, vol. 36, no. 2, pp. 645–658, 2023, doi: 10.35378/gujs.1009359.

[15] A. Khalili Fakhrabadi, M. J. Shahbazzadeh, N. Jalali, and M. Eslami, “A hybrid inception-dilated-ResNet architecture for deep learning-based prediction of COVID-19 severity,” *Scientific Reports*, vol. 15, no. 1, p. 6490, Feb. 2025, doi: 10.1038/s41598-025-91322-3.

[16] A. Kaur and D. Gupta, “A Hybrid Classification Model for Prediction of Academic Performance of Students: An EDM Application,” *Lecture Notes in Electrical Engineering*, vol. 1040 LNEE, pp. 59 – 71, 2023, doi: 10.1007/978-981-99-2271-0_6.

[17] K. Chong and N. Shah, “Comparison of naive Bayes and SVM classification in grid-search hyperparameter tuned and non-hyperparameter tuned healthcare stock market sentiment analysis,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 12, 2022, doi: 10.14569/IJACSA.2022.0131213.

[18] Y. Li, J. Zhang, T. Li, H. Liu, J. Li, and Y. Wang, “Geographical traceability of wild Boletus edulis based on data fusion of FT-MIR and ICP-AES coupled with data mining methods (SVM),” *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, vol. 177, pp. 20–27, Apr. 2017, doi: 10.1016/j.saa.2017.01.029.

[19] A. H. Shahid and M. P. Singh, “Sciencedirect a novel approach for coronary artery disease diagnosis using hybrid particle swarm optimization based emotional neural network,” *Biocybernetics and Biomedical Engineering*, vol. 40, no. 4, pp. 1568–1585, Oct. 2020, doi: 10.1016/j.bbe.2020.09.005.

[20] X. Zhang, N. Han, and J. Zhang, “Comparative analysis of VGG, ResNet, and GoogLeNet architectures evaluating performance, computational efficiency, and convergence rates,” *Applied and Computational Engineering*, vol. 44, no. 1, pp. 172–181, 2024, doi: 10.54254/2755-2721/44/20230676.

[21] I. A. Alharbi, A. J. Almalki, and C. C. Zou, “Hyperparameter optimization and comparison of student performance prediction algorithms,” in *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2021, pp. 889–894. doi: 10.1109/CSCI54926.2021.00207.

[22] Y. Ali, E. Awwad, M. Al-Razgan, and A. Maarouf, “Hyperparameter search for machine learning algorithms for optimizing the computational complexity,” *Processes*, vol. 11, no. 2, p. 349, Jan. 2023, doi: 10.3390/pr11020349.

[23] B. Drăgulescu and M. Bucos, “Hyperparameter tuning using automated methods to improve models for predicting student success,” *Communications in Computer and Information Science*, vol. 1283 CCIS, pp. 309 – 320, 2020, doi: 10.1007/978-3-030-59506-7_25.

[24] A. P. Windarto, P. Alkhairi, and J. Muslim, “Enhancing premier league match outcome prediction using support vector machine with ensemble techniques: a comparative study on bagging and boosting,” *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 9, no. 1, pp. 94–103, Feb. 2025, doi: 10.29207/resti.v9i1.6173.

[25] R. S. Gaikwad and S. C. Gandage, “MCNN: Visual sentiment analysis using various deep learning framework with deep CNN,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, no. 2s, pp. 265–278, 2023.

[26] A. Jamali, S. K. Roy, and P. Ghamisi, “WetMapFormer: A unified deep CNN and vision transformer for complex wetland mapping,” *International Journal of Applied Earth Observation and Geoinformation*, vol. 120, p. 103333, Jun. 2023, doi: 10.1016/j.jag.2023.103333.

[27] O. Habibi, M. Chemmakha, and M. Lazaar, “Performance evaluation of CNN and pre-trained models for Malware classification,” *Arabian Journal for Science and Engineering*, vol. 48, no. 8, pp. 10355–10369, Aug. 2023, doi: 10.1007/s13369-023-07608-z.

[28] I. A. Showmik, T. F. Sanam, and H. Imtiaz, “Human activity recognition from Wi-Fi CSI data using principal component-based wavelet CNN,” *Digital Signal Processing*, vol. 138, p. 104056, Jun. 2023, doi: 10.1016/j.dsp.2023.104056.

[29] Q. Dou, H. Chen, Y. Jin, L. Yu, J. Qin, and P. A. Heng, “3D deeply supervised network for automatic liver segmentation from CT volumes,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9901 LNCS, pp. 149–157, 2016, doi: 10.1007/978-3-319-46723-8_18.

[30] B. C. Anil and P. Dayananda, “Automatic liver tumor segmentation based on multi-level deep convolutional networks and fractal residual network,” *IETE Journal of Research*, vol. 69, no. 4, pp. 1925–1933, May 2023, doi: 10.1080/03772063.2021.1878066.

[31] N. N. Prakash, V. Rajesh, D. L. Namakhwa, S. Dwarkanath Pande, and S. H. Ahammad, “A DenseNet CNN-based liver lesion prediction and classification for future medical diagnosis,” *Scientific African*, vol. 20, p. e01629, Jul. 2023, doi: 10.1016/j.sciaf.2023.e01629.

[32] M. Ahmad *et al.*, “A lightweight convolutional neural network model for liver segmentation in medical diagnosis,” *Computational Intelligence and Neuroscience*, vol. 2022, 2022, doi: 10.1155/2022/7954333.

[33] P. Alkhairi and Z. Situmorang, “Application of data mining to analyze employee satisfaction with HR services with the C4. 5 algorithm,” [in Bahasa], *Jurasik (Jurnal Riset Sistem Informasi dan Teknik Informatika)*, vol. 7, no. 1, p. 40, 2022.

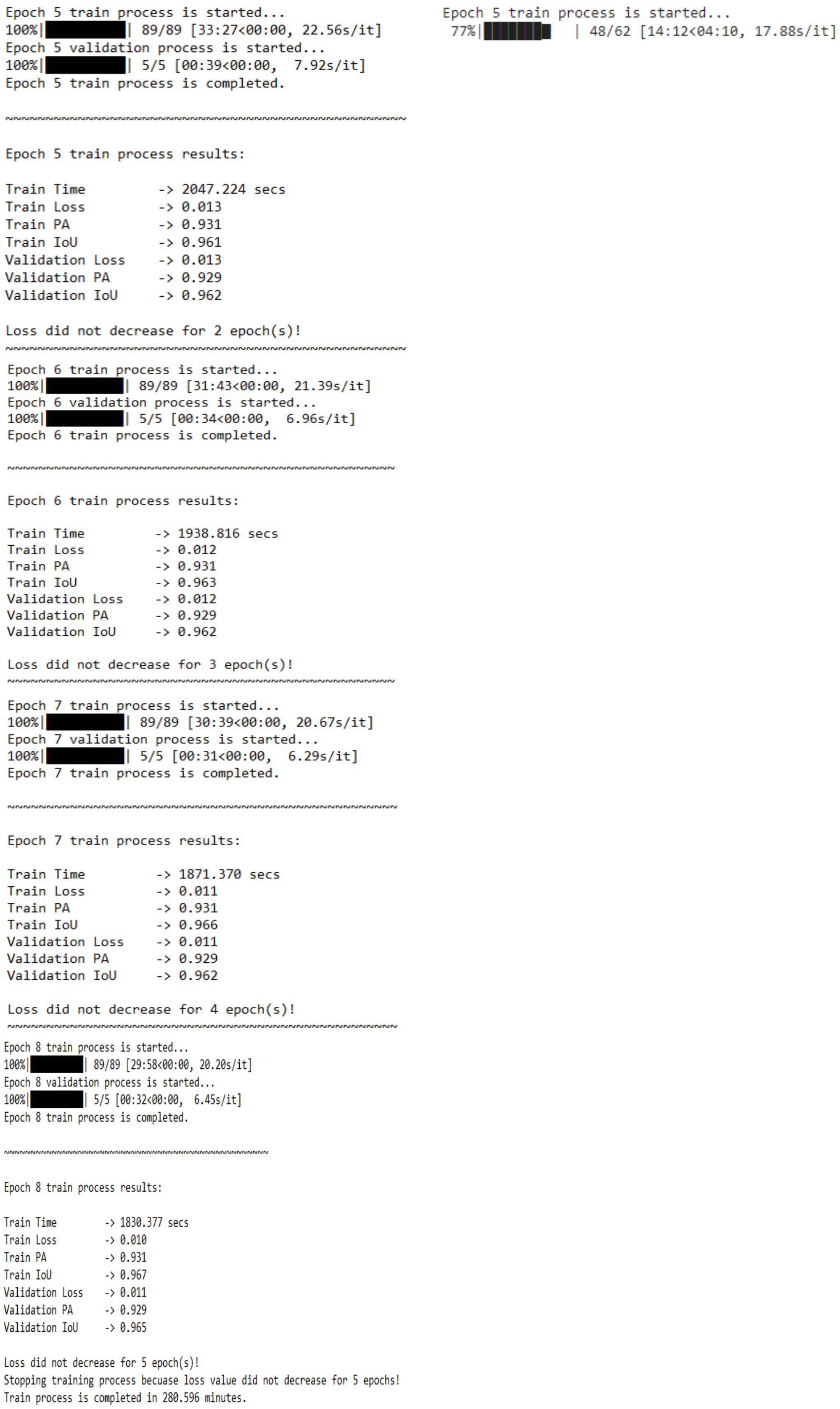
APPENDIX

Proposed model (DeepLabV3Plus)

```
Start training process...
Epoch 1 train process is started...
  0%|          | 0/89 [00:00<?, ?it/s]
100%|██████████| 89/89 [32:59<00:00, 22.24s/it]
Epoch 1 validation process is started...
100%|██████████| 5/5 [00:38<00:00,  7.60s/it]
Epoch 1 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 1 train process results:

Train Time        -> 2017.495 secs
Train Loss        -> 0.135
Train PA          -> 0.915
Train IoU         -> 0.852
Validation Loss   -> 0.043
Validation PA     -> 0.927
Validation IoU    -> 0.926

Loss decreased from inf to 0.043!
Saving the model with the best loss value...
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 2 train process is started...
100%|██████████| 89/89 [32:36<00:00, 21.99s/it]
Epoch 2 validation process is started...
100%|██████████| 5/5 [00:35<00:00,  7.16s/it]
Epoch 2 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 2 train process results:

Train Time        -> 1992.583 secs
Train Loss        -> 0.030
Train PA          -> 0.932
Train IoU         -> 0.932
Validation Loss   -> 0.025
Validation PA     -> 0.929
Validation IoU    -> 0.941

Loss decreased from 0.043 to 0.025!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 3 train process is started...
100%|██████████| 89/89 [48:02<00:00, 32.38s/it]
Epoch 3 validation process is started...
100%|██████████| 5/5 [00:50<00:00, 10.08s/it]
Epoch 3 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 3 train process results:

Train Time        -> 2932.599 secs
Train Loss        -> 0.020
Train PA          -> 0.931
Train IoU         -> 0.948
Validation Loss   -> 0.019
Validation PA     -> 0.929
Validation IoU    -> 0.950

Loss decreased from 0.025 to 0.019!
Saving the model with the best loss value...
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 4 train process is started...
100%|██████████| 89/89 [36:08<00:00, 24.36s/it]
Epoch 4 validation process is started...
100%|██████████| 5/5 [00:33<00:00,  6.71s/it]
Epoch 4 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 4 train process results:

Train Time        -> 2201.720 secs
Train Loss        -> 0.016
Train PA          -> 0.931
Train IoU         -> 0.955
Validation Loss   -> 0.015
Validation PA     -> 0.929
Validation IoU    -> 0.957

Loss did not decrease for 1 epoch(s)!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

ResNet

```
Start training process...
Epoch 1 train process is started...
100%|██████████| 62/62 [18:27<00:00, 17.86s/it]
Epoch 1 validation process is started...
100%|██████████| 4/4 [00:18<00:00,  4.57s/it]
Epoch 1 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 1 train process results:

Train Time        -> 1125.339 secs
Train Loss        -> 0.921
Train PA          -> 0.193
Train IoU         -> 0.102
Validation Loss   -> 0.920
Validation PA     -> 0.118
Validation IoU    -> 0.062

Loss decreased from inf to 0.920!
Saving the model with the best loss value...
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 2 train process is started...
100%|██████████| 62/62 [18:26<00:00, 17.85s/it]
Epoch 2 validation process is started...
100%|██████████| 4/4 [00:19<00:00,  4.77s/it]
Epoch 2 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 2 train process results:

Train Time        -> 1125.577 secs
Train Loss        -> 0.921
Train PA          -> 0.193
Train IoU         -> 0.103
Validation Loss   -> 0.925
Validation PA     -> 0.113
Validation IoU    -> 0.058

Loss did not decrease for 1 epoch(s)!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 3 train process is started...
100%|██████████| 62/62 [18:17<00:00, 17.71s/it]
Epoch 3 validation process is started...
100%|██████████| 4/4 [00:20<00:00,  5.24s/it]
Epoch 3 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 3 train process results:

Train Time        -> 1118.901 secs
Train Loss        -> 0.922
Train PA          -> 0.193
Train IoU         -> 0.103
Validation Loss   -> 0.923
Validation PA     -> 0.112
Validation IoU    -> 0.058

Loss did not decrease for 2 epoch(s)!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 4 train process is started...
100%|██████████| 62/62 [18:17<00:00, 17.70s/it]
Epoch 4 validation process is started...
100%|██████████| 4/4 [00:19<00:00,  4.80s/it]
Epoch 4 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 4 train process results:

Train Time        -> 1116.890 secs
Train Loss        -> 0.921
Train PA          -> 0.193
Train IoU         -> 0.102
Validation Loss   -> 0.920
Validation PA     -> 0.117
Validation IoU    -> 0.060

Loss did not decrease for 3 epoch(s)!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```



```
Epoch 5 train process is started...
100%|██████████| 89/89 [33:27<00:00, 22.56s/it]
Epoch 5 validation process is started...
100%|██████████| 5/5 [00:39<00:00,  7.92s/it]
Epoch 5 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 5 train process results:

Train Time        -> 2047.224 secs
Train Loss        -> 0.013
Train PA          -> 0.931
Train IoU         -> 0.961
Validation Loss   -> 0.013
Validation PA     -> 0.929
Validation IoU    -> 0.962

Loss did not decrease for 2 epoch(s)!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Epoch 6 train process is started...
100%|██████████| 89/89 [31:43<00:00, 21.39s/it]
Epoch 6 validation process is started...
100%|██████████| 5/5 [00:34<00:00,  6.96s/it]
Epoch 6 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 6 train process results:

Train Time        -> 1938.816 secs
Train Loss        -> 0.012
Train PA          -> 0.931
Train IoU         -> 0.963
Validation Loss   -> 0.012
Validation PA     -> 0.929
Validation IoU    -> 0.962

Loss did not decrease for 3 epoch(s)!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Epoch 7 train process is started...
100%|██████████| 89/89 [30:39<00:00, 20.67s/it]
Epoch 7 validation process is started...
100%|██████████| 5/5 [00:31<00:00,  6.29s/it]
Epoch 7 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 7 train process results:

Train Time        -> 1871.370 secs
Train Loss        -> 0.011
Train PA          -> 0.931
Train IoU         -> 0.966
Validation Loss   -> 0.011
Validation PA     -> 0.929
Validation IoU    -> 0.962

Loss did not decrease for 4 epoch(s)!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Epoch 8 train process is started...
100%|██████████| 89/89 [29:58<00:00, 20.20s/it]
Epoch 8 validation process is started...
100%|██████████| 5/5 [00:32<00:00,  6.45s/it]
Epoch 8 train process is completed.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Epoch 8 train process results:

Train Time        -> 1830.377 secs
Train Loss        -> 0.010
Train PA          -> 0.931
Train IoU         -> 0.967
Validation Loss   -> 0.011
Validation PA     -> 0.929
Validation IoU    -> 0.965

Loss did not decrease for 5 epoch(s)!
Stopping training process becuase loss value did not decrease for 5 epochs!
Train process is completed in 280.596 minutes.
```

```
Epoch 5 train process is started...
 77%|███████▋  | 48/62 [14:12<04:10, 17.88s/it]
```

Figure 3. Comparison of training process between proposed model DeepLabV3 plus and ResNet

BIOGRAPHIES OF AUTHORS

Iwan Purnama studied at SDN 117844 Ajamu, MTS Al-Ikhlas Ajamu, MAS Al-Ikhlas Ajamu, AMIK STIEKOM Sumatera Utara (D3), STMIK Triguna Dharma Medan (S1), Universitas Putra Indonesia "YPTK" Padang (S2) and Universitas Negeri Padang (S3). Currently Lecturer and Dean of the Faculty of Science and Technology, Labuhanbatu University. Founder of the IBAY KOMPUTER Course and Training Institute (LKP), Founder of WebBagusmurah.com. Active in organizational activities APTIKOM, FKPT, Himpunan Dosen Gemilang Indonesia (hdgi.org), Labuhanbatu Berbagi Gemilang Foundation, Chair of the DPW SUMUT Forum Komunikasi Dosen and others. His profile can be found on Youtube @IbayKomputer and https://s.id/iwanpurnama. SINTA ID: 6163127, SSRN ID: 4733732. He also can be contacted at email: iwanpurnama2014@gmail.com.

Agus Perdana Windarto the authors were born in Pematangsiantar on August 30 1986. They completed their Masters in Computer Science and Doctoral (Ph.D.) degrees at Putra Indonesia University 'YPTK' Padang in 2014 and 2024. The author has been an active lecturer at STIKOM Tunas Bangsa since 2012, teaching in the Information Systems program. Their research focuses on artificial intelligence (decision support systems, expert systems, data mining, neural networks, fuzzy logic, deep learning, and genetic algorithms). Additionally, the author has served as a reviewer for various nationally accredited journals (SINTA 2 - SINTA 6) and manages a community called “Pemburu Jurnal” at STIKOM Tunas Bangsa. They have won multiple research grant proposals from DIKTI (twice in 2018-2019), DIKTI Community Service Grant (once in 2019), PKM-P Grant (as a student advisor in 2018), and PKM-AI Grant (as a student advisor in 2019). The author is also part of the Relawan Jurnal Indonesia (RJI) community in North Sumatra, the Data Science Indonesia Researchers Association (PDSI), the Forum of Higher Education Communities (FKPT), and is a co-founder of the Yayasan Adwitiya Basurata Inovasi (Yayasan Abivasi) foundation with fellow professors. He can be contacted at email: agus.perdana@amiktunasbangsa.ac.id or agus.perdana@abivasi.id.

Solikhun is an associate professor in the STIKOM Tunas Bangsa Informatics Engineering Undergraduate Study Program. In 2022 he will also become Chair of the STIKOM Tunas Bangsa Informatics Engineering Undergraduate Study Program. He is a writer who completed his Doctoral education at the Faculty of Computer Science and Information Technology, University of North Sumatra. where he is the author/co-author of more than 2013 research publications. Can be contacted via email: solikhun@amiktunasbangsa.ac.id. SINTA ID: 5972880.