IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 13, No. 3, September 2024, pp. 3273~3282
ISSN: 2252-8938, DOI: 10.11591/ijai.v13.i3.pp3273-3282

Transfer learning scenarios on deep learning for ultrasound-based image segmentation

Didik Bani Unggul, Nur Iriawan, Heri Kuswanto
Department of Statistics, Faculty of Science and Data Analytics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

Article Info

Article history:

Received Jan 3, 2024
Revised Feb 21, 2024
Accepted Mar 6, 2024

Keywords:

Deep learning
Image segmentation
Left ventricular wall
Transfer learning
Ultrasound

ABSTRACT

Deep learning coupled with transfer learning, which involves reusing a pretrained model's network structure and parameter values, offers a rapid and accurate solution for image segmentation. Differing approaches exist in updating transferred parameters during training. In some studies, parameters remain frozen or untrainable (referred to as TL-S1), while in others, they act as trainable initial values updated from the first iteration (TL-S2). We introduce a new state-of-the-art transfer learning scenario (TL-S3), where parameters initially remain unchanged and update only after a specified cutoff time. Our research focuses on comparing the performance of these scenarios, a dimension yet unexplored in the literature. We simulate on three architectures (Dense-UNet-121, Dense-UNet-169, and Dense-UNet-201) using an ultrasound-based dataset with the left ventricular wall as the region of interest. The results reveal that the TL-S3 consistently outperforms the previous state-of-the-art scenarios, i.e., TL-S1 and TL-S2, achieving correct classification ratios (CCR) above 0.99 during training with noticeable performance spikes post-cutoff. Notably, two out of three top-performing models in the validation data also originate from TL-S3. Finally, the best model is the Dense-UNet-121 with TL-S3 and a 20% cutoff. It achieves the highest CCR for training 0.9950, validation 0.9699, and testing data 0.9695, confirming its excellence.

Corresponding Author:

Nur Iriawan
Department of Statistics, Faculty of Science and Data Analytics, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya, Indonesia
Email: nur_i@statistika.its.ac.id

1. INTRODUCTION

Image segmentation is a crucial task in image and video processing. This process involves dividing the image into multiple segments or objects by assigning class labels to each pixel [1]. Its applications are widespread and encompass medical imaging [2]–[4], remote sensing [5]–[7], and the development of autonomous vehicles [8]–[10]. Amid various segmentation methods, deep learning emerges as a promising approach [11]–[13]. They decompose complex mappings into a sequence of simpler ones, each described by different layers [14]. The input is presented in a visible layer, and subsequent hidden layers extract abstract features from it. The refinement of these layers is driven by the results of the training process, rather than manual intervention [15]. With a large number of layers, they can accurately represent input features and effectively perform complex tasks like image segmentation, natural language processing, or stock price prediction [16]. Due to this benefit, deep learning is better than traditional machine learning methods, which still rely on domain expertise for feature extraction.

Deep learning implementation, however, requires a large amount of training data and may require a while to complete [17]. This presents difficulties, particularly in the medical domain where labeled datasets

are scarce [18]. To overcome this issue, transfer learning can be coupled with deep learning approach [17]. Reusing pre-trained network components, such as the structure and parameter values, is part of this process. To be more precise, the network is typically divided into two parts: the part receiving transfer learning and the part not receiving it. The first, leveraging transfer learning, will be structurally identical with parameter values transferred from a pre-trained model. The source model is typically trained on a larger dataset, which may be related or entirely different. The next section is a non-transferred layer, meaning its parameter values are initialized and updated during training.

Furthermore, variations exist in how parameter values are handled in layers affected by transfer learning. These values can be "frozen" (non-trainable) and maintained in that state, or they can be "unfrozen" (trainable) and updated as the training progresses. Some studies treat them as non-trainable parameters [19]–[22]. On the other hand, some researchers utilize transfer learning values for initialization and updating them immediately in the first training iteration [23], [24]. Unfortunately, to the best of our knowledge, no research has evaluated the effectiveness of these two scenarios simultaneously. The majority of the articles only contrasted one scenario of transfer learning with a model that did not employ transfer learning [19], [21]. Furthermore, a lot of applications only construct a transfer learning model without contrasting it with any other models [18]. This leads to a gap in knowledge that requires research. Therefore, this study aims to compare those two parameter update scenarios, as well as introduce a new state-of-the-art transfer learning scenario. This scenario involves updating the newly transferred parameter values only after a specific time point is reached.

Dense-UNet, a deep learning architecture that hybridizes Unet [4] and DenseNet [25], was employed in this investigation. This architecture was implemented to limit the number of model parameters, maximize information flow between network layers, and address vanishing gradient concerns due to its feature reuse and dense connections at each stage [26]. The encoder and the decoder are the two primary components of this architecture in general. The encoder, also known as the contraction path, is responsible for applying transfer learning from a pre-trained model and extracting features. The second component, known as the expanding path or decoder, is amid reconfiguring features and boosting spatial resolution through the use of upsampling operators [4], [27]. These two paths are connected via skip connections, in which the feature maps from the encoder are bypassed and concatenated with the decoder results at specific positions [28].

The simulation will be conducted on an ultrasound-based cardiac assessment dataset. Ultrasound, known for its accessibility, affordability, and absence of radiation exposure, addresses key healthcare concerns [29]. However, due to increased noise and decreased contrast, observing certain cardiac features can be challenging, as they are typically difficult to determine and interpret [30]. Therefore, automatic segmentation is urgently required for assistance in identifying the region of interest in ultrasound-based images. Nevertheless, in contrast to other non-invasive imaging modalities like magnetic resonance imaging (MRI) and computed tomography scan (CT-scan), research on automatic segmentation in ultrasound, particularly utilizing deep learning, has been very limited in recent years [31]. To overcome this problem, we employ a publicly available dataset from Hamad Medical Corporation, Qatar University, and Tampere University known as the HMC-QU dataset, accessible at https://www.kaggle.com/datasets/aysendegerli/hmcqu-dataset. This dataset encompasses ultrasound-based assessments featuring diverse patients and viewpoint types. Furthermore, the ground truth is supplied, with the left ventricular wall (LVW) serving as the region of interest (ROI). This is essential to us because LVW movement and structure analysis serves as an early indicator of various heart problems, including myocardial infarction and hypertrophic cardiomyopathy [30], [32]. This dataset has been used in several earlier investigations, either for segmentation or for the identification of structural and movement anomalies [33]–[37]. While deep learning remains the dominant option, none of these studies has explored the use of transfer learning to the extent that we propose. Therefore, our research provides practical benefits for the development of ultrasound-based cardiac image processing in addition to theoretical benefits for deep learning transfer learning scenarios.

2. METHOD

2.1. Dense-UNet architecture

Dense-UNet is a modified U-Net architecture that incorporates dense blocks and transition layers into its structure, drawing inspiration from the DenseNet architecture introduced by [25]. Their layer-to-layer linkages are the main distinction between standard block layers and dense blocks. Each layer in a dense block obtains feature mappings from every layer before it via some concatenation [25]. This feature reuse minimizes the addition of excessive features in each layer, consequently reducing the required parameters. However, it necessitates that the dimensions of feature maps remain unchanged due to concatenation-based merging. This limitation impedes the implementation of a pooling procedure, which is generally resolved by

adding a transition layer. In the original configuration, this transition layer consists of 2×2 average pooling preceded by 1×1 convolution.

Figure 1 illustrates the structure of the nine-stage Dense-UNet. A 7×7 convolution is employed in the first step to process the input dimensions from 224×224 to 112×112. This process continues with the first transition layer, leading us to the first dense block in the second stage. Within a dense block, layer configurations include batch normalization (BN), rectified linear unit (ReLU) activation, 3×3 convolution, another BN, ReLU activation, and 1×1 convolution. This sequence is repeated several times depending on the architectural construction. Subsequently, the second transition layer guides us to the third stage (second dense block). We will continue this process until we reach the fourth dense block in the fifth stage when we have 7×7 feature maps. The next step involves starting 2×2 upsampling and concatenating the result with the final feature maps from the fourth stage. Their results will serve as the input for the fifth dense block, which has the same layer configuration as its mirrored version (third dense block). This provision continues until we reach the ninth stage, concluding with a sigmoid activation layer and a resulting output of 224×224.

Determining how many layers are present in each dense block is another crucial factor. The number of layers in this study, ranging from stage one to stage five, follows the DenseNet-121, DenseNet-169, and DenseNet-201 structure of the original DenseNet versions [25]. The sixth to ninth stages replicate this structure by mirroring the number of layers. Under these conditions, the three Dense-UNet architectures in this study are named Dense-UNet-121, Dense-UNet-169, and Dense-UNet-201.

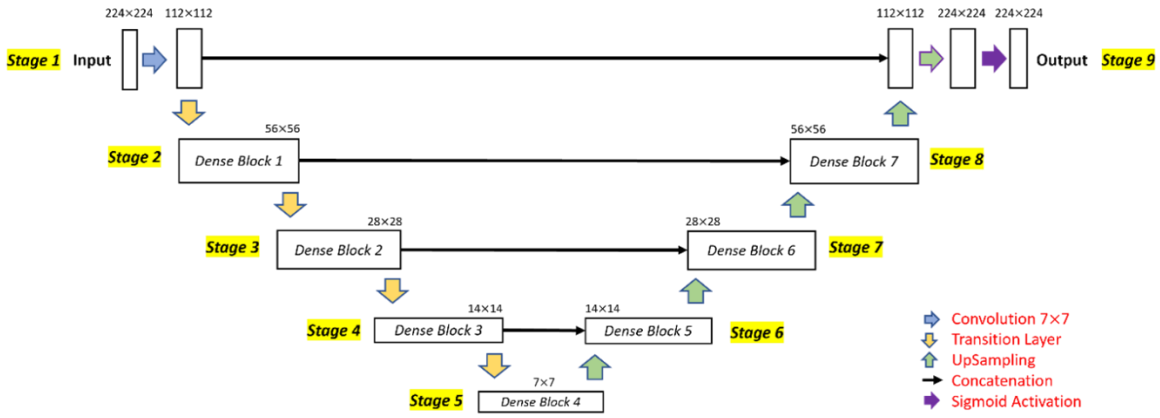


Figure 1. Dense-UNet structure with nine processing stages

2.2. Transfer learning

Transfer learning is a concept in learning that attempts to enhance model performance by employing knowledge acquired from a learning task in one domain (the source domain) to improve performance in a different area (the target domain). Addressing data inadequacy in the target domain is one of the many benefits of this method [17], [38], [39]. It will mitigate this issue by relaxing the assumption that training and testing data must originate from the identical domain. Transfer learning in deep learning refers to pre-training a network on a source domain, frequently a larger dataset like ImageNet [40]. This process leads to a model with optimized parameter values representing previously acquired knowledge. These parameter values are subsequently transferred to another network created particularly for the target dataset. Notably, the two networks are often dissimilar. As a result, the new model incorporates layers that receive the transfer learning results alongside layers that do not. In the Dense-UNet architecture discussed earlier, we can determine that the portion designated for transfer learning is the initial half known as the encoder, encompassing the first to fifth stages. If the architecture has M layers and the encoder consists of K layers ($K < M$), the first K layers will receive the parameter values from the pre-trained model. Furthermore, the other layers will be initialized using either fixed or random numbers [41]. After that, there are different scenarios for how we handle the transferred parameters:

– Scenario 1: freeze scenario (TL-S1). Transferred parameters are regarded as untrainable since their values remain unchanged while they are being trained, essentially freezing them. Layers not undergoing transfer learning will be initialized and updated from the first iteration until the completion of training.
– Scenario 2: unfreeze scenario (TL-S2). In this scenario transfer learning parameter values act as initializations and are changed in real time throughout training. Thus, all parameters are deemed trainable,

regardless of whether they are in layers with or without transfer learning. The initialization procedure is where the differences arise: non-transferred layers begin with glorot uniform initialization, whereas other layers start with values from a pre-trained model.

– Scenario 3: freeze-unfreeze scenario (TL-S3). Parameters in layers affected by transfer learning will remain unchanged for an initial portion of the training process. In other words, only the parameters in layers not affected by transfer learning will be updated, while those influenced by transfer learning will be frozen. After reaching a pre-defined epoch threshold, the transfer learning layer is unfrozen, and training continues across all layers. The transfer learning cutoff will be explored at various stages, including 20%, 40%, 60%, and 80% of the total training epochs. This exploration will clarify how the timing of the transition impacts the final outcome.

In this study, we will simulate the three scenarios that are depicted in Figure 2.

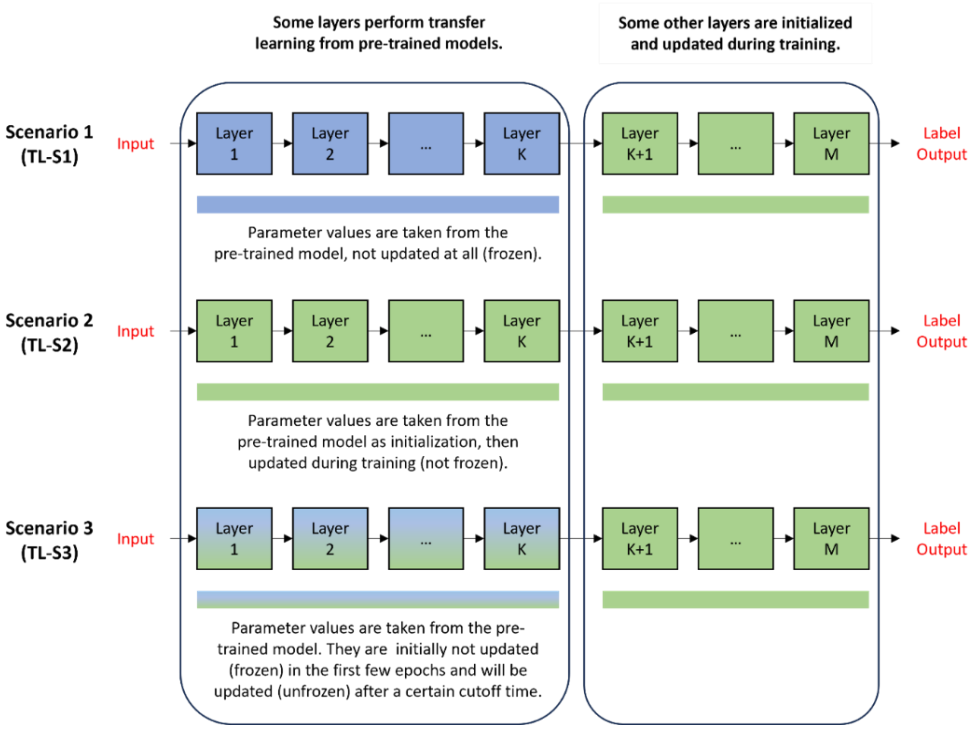


Figure 2. Three parameter updating scenarios in transfer learning

2.3. Optimization technique

In architectures like Dense-UNet, "trainable parameters" encompass weights and biases in convolution layers, as well as scale and shift parameters in BN layers. During training, these parameters are optimized, commencing with initial values generated by glorot-uniform initialization [42]. This technique uses a uniform distribution with an interval limit of $[-a, a]$, where a is calculated employing (1). The values of n_{in} and n_{out} represent the number of input and output units of the layer, respectively.

$$a = \frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}} \tag{1}$$

Next, the adaptive moment (Adam) technique proposed in [43] will be utilized for updating the initial value iteratively. This method updates parameter values using bias-corrected values of gradients' first and second moments estimations. Algorithm 1 illustrates the procedure. The first component that must be calculated is the gradient of the loss function with respect to the model parameters, denoted by g_t where t is the index of iteration performed. The binary cross-entropy loss function as in (2) was selected to suit the binary classification task.

$$L_i = -\left[c_i \, log\left(p(c_i)\right) + (1 - c_i) \, log\left(1 - p(c_i)\right)\right]\text{q} \tag{2}$$

Algorithm 1. Adam Optimization
Require: β_1, β_2, η, ε: Hyperparameter
Require: $f_t(\theta_{t-1})$: Stochastic objective function as in (2) with trainable parameter θ_{t-1} at time-step $t-1$
Require: θ_0: Initial parameter vector generated from glorot uniform
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize time-step)
While θ_t not converged do
$t \leftarrow t+1$
Get gradients w.r.t. stochastic objective at timestep t using (3):

$$g_t = \nabla_\theta f_t(\theta_{t-1}) \quad (3)$$

Update biased 1st moment estimate using (4):

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad (4)$$

Update biased 2nd moment estimate using (5):

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \quad (5)$$

Compute bias-corrected 1st moment estimate using (6):

$$\widehat{m}_t = \frac{1}{1-(\beta_1)^t} m_t \quad (6)$$

Compute bias-corrected 2nd moment estimate using (7):

$$\hat{v}_t = \frac{1}{1-(\beta_2)^t} v_t \quad (7)$$

Update parameters using (8):

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t}+\varepsilon} \widehat{m}_t \quad (8)$$

End while
Return θ_t (Resulting parameters)

The loss for the i^{th} pixel, denoted as L_i, is defined for $i = 1, \ldots, N$ with N representing the total pixels in the output image. The actual classification class of i^{th} pixel is notated by $c_i \in \{0,1\}$, in which $c_i = 0$ is for the background and $c_i = 1$ is for the ROI. Lastly,$p(c_i)$ is the predicted probability of belonging to class c_i calculated by the model. After finding the g_t, we are able to calculate the exponentially weighted moving average of the gradient (m_t) and squared gradient (v_t). This step demands us to configure the hyperparameter values $\beta_1, \beta_2 \in [0,1)$ as the exponential decay rates for the moment estimates. We then utilize the bias-corrected versions of m_t and v_t along with η and ε to update parameter values from θ_{t-1} to θ_t. We also set the hyperparameter values at $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\eta = 10^{-6}$, and $\varepsilon = 10^{-8}$.

2.4. Evaluation metric

This study utilizes the correct classification ratio (CCR) to evaluate the model performances. This metric is demonstrated in (9). GT_j represents the ground truth area for class j, while Seg_j depicts the model's corresponding segmentation area. Class $j = 0$ is designated for the background (non-ROI) and $j = 1$ is for the LVW area (ROI).$|GT_j \cap Seg_j|$ denotes the number of pixels from class j which are accurately classified by the model. $|GT|$ can be measured by counting the number of pixels from the union of the GT_0 and GT_1 areas. The CCR values vary between 0 and 1. All pixels are appropriately categorized and our segmentation precisely matches the ground truth if the CCR value is one. A decreasing CCR indicates deteriorating segmentation results.

$$\text{CCR} = \sum_{j=0}^{1} \frac{|GT_j \cap Seg_j|}{|GT|} \quad (9)$$

2.5. Dataset and experimental setup

An online echocardiogram (ECG) dataset made available by HMC-QU was employed in this study. Specifically, we concentrated on a subset of the dataset that included 109 ECG video recordings that had 224×224 pixels ground truth available. These videos presented the apical 4-chamber view at a resolution of 636×422 pixels. They had a frame rate of 25 frames per second and ranged in duration from 1 to 3 seconds.

The videos were randomly divided into training, validation, and testing sets (80%:10%:10%), resulting in 87 training videos, 11 for validation, and 11 for testing. The images were then extracted and fed into preprocessing. They were center-cropped to 422×422 pixels and resized to 224×224 pixels. The red, green, and blue channels' color intensities were then extracted into three matrices. The matrix elements, initially ranging from 0 to 255, were normalized to a 0-to-1 range, serving as input for the deep learning architecture.

The training utilized a batch size of 10, with 10 images selected at random for each iteration. Each epoch concluded after processing all images, and this procedure was repeated for 100 epochs. The experiment was conducted on Google Colab using an NVIDIA V100 GPU, with Python 3 and the Keras framework chosen for their effectiveness and executability.

3. RESULTS AND DISCUSSION

Figure 3 displays an example of many ultrasound pictures processed during this study. The original photographs that were selected at random are presented in the first row, and the ground truth for those images is displayed in the second row. The combined result, which displays the location of the LVW area determined by the ground truth, can be examined in the third row. The models use ground truth as a reference during training to effectively learn and recognize LVW characteristics.

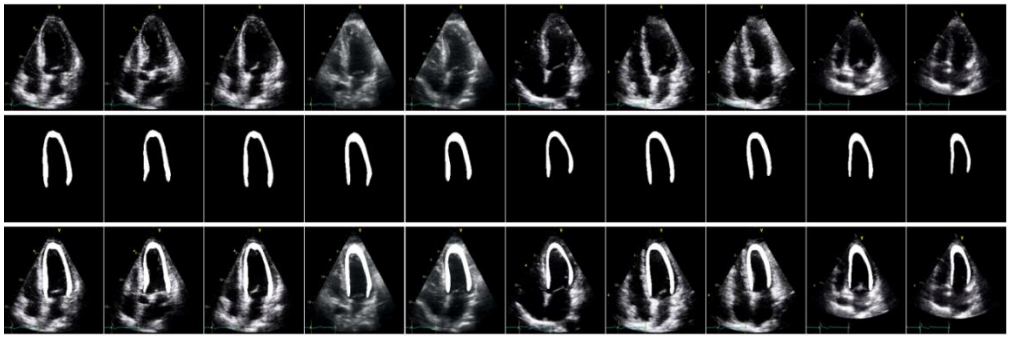


Figure 3. Some examples of ultrasound-based images and their ground truth (mask)

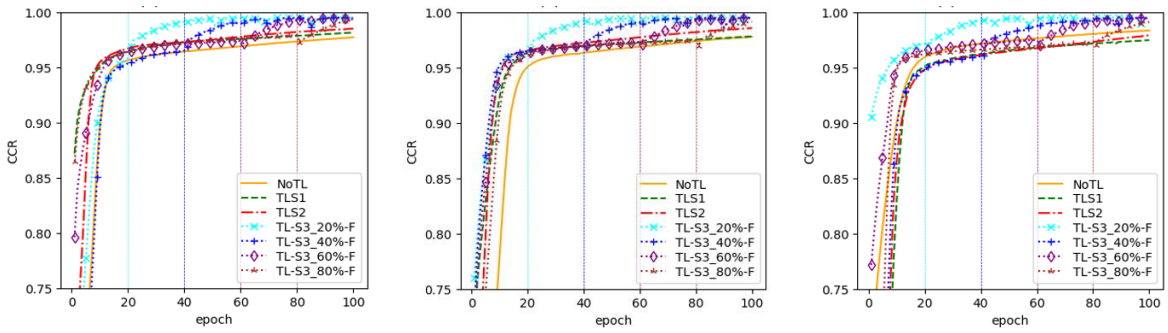
Table 1 summarizes the training durations (in seconds), loss values, and CCR for the three Dense-UNet architectures with various transfer learning scenarios. Notably, across all architectures, the suggested third scenario (TL-S3) consistently outperforms models without transfer learning (NoTL), TL-S1, and TL-S2. The models under TL-S3 achieve a remarkable CCR exceeding 0.99, a level not attained by models from other scenarios. Furthermore, TL-S3 models demonstrate far lower losses than the others, with reductions ranging from 82% to 97%.

Table 1. The performance results of Dense-UNet models

Architecture	Transfer Learning Scenario	Training		Validation		Duration (in second)
		Loss	CCR	Loss	CCR	
Dense-UNet-121	No TL	0.2338	0.9772	0.2461	0.9685	2,983
	TL-S1	0.0918	0.9815	0.1166	0.9685	1,955
	TL-S2	0.1509	0.9849	0.1793	0.9679	2,966
	TL-S3 20%-F	0.0095	0.9950	0.2063	0.9699	2,857
	TL-S3 40%-F	0.0106	0.9948	0.1765	0.9694	2,666
	TL-S3 60%-F	0.0169	0.9925	0.1369	0.9677	2,455
	TL-S3 80%-F	0.0169	0.9925	0.1282	0.9692	2,250
Dense-UNet-169	No TL	0.2491	0.9777	0.2610	0.9679	3,849
	TL-S1	0.1639	0.9780	0.1763	0.9694	2,383
	TL-S2	0.1555	0.9854	0.1886	0.9674	3,838
	TL-S3 20%-F	0.0095	0.9950	0.2074	0.9678	2,752
	TL-S3 40%-F	0.0101	0.9949	0.2038	0.9678	2,552
	TL-S3 60%-F	0.0110	0.9946	0.1823	0.9689	2,373
	TL-S3 80%-F	0.0198	0.9913	0.1321	0.9667	2,170
Dense-UNet-201	No TL	0.1391	0.9833	0.1648	0.9694	4,874
	TL-S1	0.2200	0.9746	0.2282	0.9683	3,112
	TL-S2	0.2872	0.9791	0.3048	0.9669	4,893
	TL-S3 20%-F	0.0100	0.9949	0.1905	0.9684	4,668
	TL-S3 40%-F	0.0108	0.9947	0.1691	0.9680	4,327
	TL-S3 60%-F	0.0115	0.9945	0.1804	0.9687	3,989
	TL-S3 80%-F	0.0204	0.9910	0.1085	0.9679	3,707

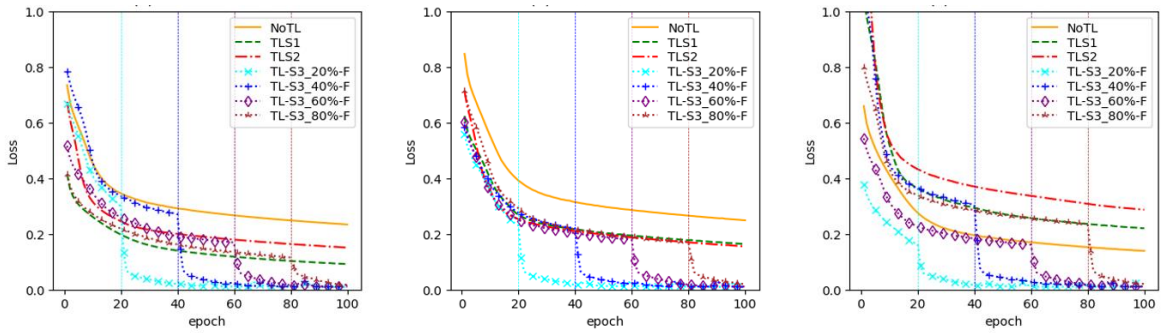
No TL: Without transfer learning; TL-S1: Transfer learning scenario 1 (freeze scenario); TL-S2: Transfer learning scenario 2 (unfreeze scenario); TL-S3 20%, 40%, 60%, 80%-F: Transfer learning scenario 3 (freeze-unfreeze scenario with non-freezing start cutoffs being 20%, 40%, 60%, and 80% of the total epoch, respectively).

Investigation reveals that when transfer learning parameters are unfrozen after the cutoff, the TL-S3 models perform noticeably better. TL-S3 20%-F models, for example, exhibit better performance spikes after 20 epochs, whereas TL-S3 40%-F models show better performance surges after 40 epochs. The TL-S3 60%-F and TL-S3 80%-F models also exhibit this pattern. The learning curve provides a visual representation of this circumstance, with Figures 4(a) to 4(c) representing CCR and Figures 5(a) to 5(c) representing loss. It validates the hypothesis that temporarily freezing transfer learning layers enables the model to adapt to the current case's characteristics without disrupting the robust feature extraction of pre-trained layers. After the non-transfer learning layer stabilizes, unfreezing the transfer learning layer boosts performance by iteratively updating all parameters. This performance jump occurs shortly after the cutoff.



(a) (b) (c)

Figure 4. Learning curve for CCR values: (a) Dense-UNet-121, (b) Dense-UNet-169, and (c) Dense-UNet-201



(a) (b) (c)

Figure 5. Learning curve for loss values: (a) Dense-UNet-121, (b) Dense-UNet-169, and (c) Dense-UNet-201

The average CCR increase for TL-S3 models during the next twenty epochs was 0.0216, compared to 0.0048 for other scenarios. This approximately five-fold difference highlights how much preferable the TL-S3 scenario is. We discover that the TL-S3 20%-F scenario corresponds to the best-performing model during training in each Dense-UNet architecture. Dense-UNet-121, 169, and 201 with this scenario had CCR values of 0.9950, 0.9950, and 0.9949, respectively, placing them among the top three in terms of both CCR and loss. With a CCR of 0.9699, the Dense-UNet-121 model with TL-S3 20%-F also leads in validation. Dense-UNet-121 with TL-S3 40%-F and Dense-UNet-169 with TL-S1 are the second and third-best models, respectively, with CCR values of 0.9694. TL-S3 scenario models were able to maintain two of the top three positions in this instance. Then, a different testing dataset was employed to further evaluate these three models, which were determined to be the best options. Once more, the model with the greatest CCR of 0.9695 was Dense-UNet-121 with TL-S3 20%-F. It performed better than Dense-UNet-121 with TL-S3 40%-F and Dense-UNet-169 with TL-S1, which had CCR values of 0.9685 and 0.9681, respectively. The results demonstrate the strong segmentation capabilities of Dense-UNet-121, confirming its superior performance with TL-S3 20%-F. It continuously achieves the greatest CCR (0.9950, 0.9699, and 0.9695, respectively) across training, validation, and testing datasets.

When comparing models with and without transfer learning, models with transfer learning generally demonstrate faster training times. Dense-UNet-201 TL-S2 is an exception, taking 19 seconds longer than

Dense-UNet-201 without transfer learning. In other circumstances, transfer learning steadily quickens the training process. Second, we anticipated that TL-S1 would demonstrate the most rapid training time. The rationale behind this approach stemmed from the observation that TL-S1 requires fewer learned parameters than TL-S2 and TL-S3. Nevertheless, our research indicates that this hypothesis is valid exclusively when comparing TL-S1 and TL-S2. Interestingly, certain of the models in the TL-S3 scenario required less training time than those in TL-S1. This result provides an interesting novel perspective to our investigation, indicating that the special parameter update approach employed by TL-S3 may help enhance the effectiveness of training. We additionally discover that among TL-S3 models, the training period varies depending on the cutoff position selection. The earlier the transition from non-trainable (freeze) to trainable (unfreeze) status occurs, the longer the training duration. This condition is attributed to the increasing proportion of epochs with a full-scale trainable parameter set. In terms of processing time, our best model, the Dense-UNet-121 with TL-S3 20%-F, also performed well. With 2,857 seconds of duration, it is faster than 52% of other models.

Lastly, Figure 6 provides a visualization of data segmentation testing with our best model. The original photos are displayed in the top row, and a comparison of the ROI contour generated by the model (red line) and the ground truth (blue line) is presented in the bottom row. This figure illustrates how the model can segment data from a new dataset that was not utilized during training.

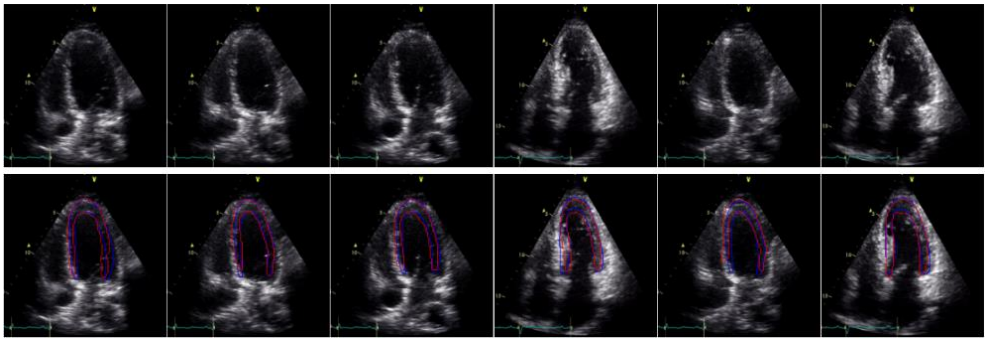


Figure 6. Segmentation results produced by the best model

4. CONCLUSION

This study provides several important conclusions. Firstly, during training, the TL-S3 scenario consistently outperforms other scenarios, achieving CCRs over 0.99 and losses under 0.0205. This superiority is explained by TL-S3's learning curve exhibiting a performance increase after surpassing the freezing cutoff. Five times higher than the rest, the average CCR increase in the 20 epochs post-cutoff is 0.0216. Furthermore, the excellence of TL-S3 extends to validation process, securing top positions in terms of the highest CCR. In summary, the Dense-UNet-121 model with TL-S3 20%-F is deemed the best, achieving a training duration of 2,857 seconds and attaining the highest CCR values for training, validation, and testing data (0.9950, 0.9699, and 0.9695, respectively). This study establishes opportunities for further research on the TL-S3 scenario by raising two crucial issues: first, determining the optimal transition point from 'untrainable' to 'trainable' status, and second, exploring how distinct training parameter adjustments can be made for each layer impacted by transfer learning. These investigations are expected to enhance the robustness and performance of the deep learning model with transfer learning.

ACKNOWLEDGEMENTS

The research presented in this paper was supported by Department of Statistics, Institut Teknologi Sepuluh Nopember and Indonesia Endowment Fund for Education Agency under scholarship no. KET-438/LPDP.4/2022.

REFERENCES

[1] R. Szeliski, *Computer vision: algorithms and applications*. Cham: Springer, 2022.
[2] R. Ranjbarzadeh, A. Caputo, E. B. Tirkolaee, S. J. Ghoushchi, and M. Bendechache, "Brain tumor segmentation of MRI images: a comprehensive review on the application of artificial intelligence tools," *Computers in Biology and Medicine*, vol. 152, 2023, doi: 10.1016/j.compbiomed.2022.106405.

[3] N. Salpea, P. Tzouveli, and D. Kollias, "Medical image segmentation: a review of modern architectures," in *Computer Vision – ECCV 2022 Workshops*, 2023, pp. 691–708, doi: 10.1007/978-3-031-25082-8_47.
[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Cham: Springer, 2015, pp. 234–241, doi: 10.1007/978-3-319-24574-4_28.
[5] S. M. Azimi, C. Henry, L. Sommer, A. Schumann, and E. Vig, "SkyScapes - fine-grained semantic understanding of aerial scenes," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 7392–7402, doi: 10.1109/ICCV.2019.00749.
[6] P. Bhadoria, S. Agrawal, and R. Pandey, "Image segmentation techniques for remote sensing satellite images," *IOP Conference Series: Materials Science and Engineering*, vol. 993, no. 1, pp. 1–17, 2020, doi: 10.1088/1757-899X/993/1/012050.
[7] B. E. -Zahouani *et al.*, "Remote sensing imagery segmentation in object-based analysis: a review of methods, optimization, and quality evaluation over the past 20 years," *Remote Sensing Applications: Society and Environment*, vol. 32, 2023, doi: 10.1016/j.rsase.2023.101031.
[8] D. Feng *et al.*, "Deep multi-modal object detection and semantic segmentation for autonomous driving: datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2021, doi: 10.1109/TITS.2020.2972974.
[9] D. -V. Giurgi, T. J. -Laurain, M. Devanne, and J. -P. Lauffenburger, "Real-time road detection implementation of UNet architecture for autonomous driving," in *2022 IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, 2022, pp. 1–5, doi: 10.1109/IVMSP54334.2022.9816237.
[10] L. Lizhou and Z. Yong, "A closer look at U-net for road detection," in *Tenth International Conference on Digital Image Processing (ICDIP 2018)*, 2018, doi: 10.1117/12.2503282.
[11] M. Aljabri and M. AlGhamdi, "A review on the use of deep learning for medical images segmentation," *Neurocomputing*, vol. 506, pp. 311–335, 2022, doi: 10.1016/j.neucom.2022.07.070.
[12] B. Sistaninejhad, H. Rasi, and P. Nayeri, "A review paper about deep learning for medical image analysis," *Computational and Mathematical Methods in Medicine*, vol. 2023, pp. 1–10, 2023, doi: 10.1155/2023/7091301.
[13] S. M. Khaniabadi, H. Ibrahim, I. A. Huqqani, F. M. Khaniabadi, H. A. M. Sakim, and S. S. Teoh, "Comparative review on traditional and deep learning methods for medical image segmentation," in *2023 IEEE 14th Control and System Graduate Research Colloquium (ICSGRC)*, 2023, pp. 45–50, doi: 10.1109/ICSGRC57744.2023.10215402.
[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts: MIT Press, 2016.
[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.
[16] E. Alaros, M. Marjani, D. A. Shafiq, and D. Asirvatham, "Predicting consumption intention of consumer relationship management users using deep learning techniques: a review," *Indonesian Journal of Science and Technology*, vol. 8, no. 2, pp. 307–328, 2023, doi: 10.17509/ijost.v8i2.55814.
[17] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning – ICANN 2018*, Cham: Springer, 2018, pp. 270–279, doi: 10.1007/978-3-030-01424-7_27.
[18] P. Kora *et al.*, "Transfer learning techniques for medical image analysis: a review," *Biocybernetics and Biomedical Engineering*, vol. 42, no. 1, pp. 79–107, 2022, doi: 10.1016/j.bbe.2021.11.004.
[19] A. A. Pravitasari *et al.*, "UNet-VGG16 with transfer learning for MRI-based brain tumor segmentation," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 18, no. 3, pp. 1310–1318, 2020, doi: 10.12928/TELKOMNIKA.v18i3.14753.
[20] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1717–1724, doi: 10.1109/CVPR.2014.222.
[21] D. A. Rasyid, G. H. Huang, and N. Iriawan, "Segmentation of low-grade gliomas using U-Net VGG16 with transfer learning," in *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2021, pp. 393–398, doi: 10.1109/Confluence51648.2021.9377093.
[22] O. T. Bişkin, İ. Kırbaş, and A. Çelik, "A fast and time-efficient glitch classification method: a deep learning-based visual feature extractor for machine learning algorithms," *Astronomy and Computing*, vol. 42, 2023, doi: 10.1016/j.ascom.2022.100683.
[23] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in Neural Information Processing Systems*, vol. 4, pp. 3320–3328, 2014.
[24] Z. Yang, J. Yue, Z. Li, and L. Zhu, "Vegetable image retrieval with fine-tuning VGG model and image hash," *IFAC-PapersOnLine*, vol. 51, no. 17, pp. 280–285, 2018, doi: 10.1016/j.ifacol.2018.08.175.
[25] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.
[26] Y. Cao, S. Liu, Y. Peng, and J. Li, "DenseUNet: densely connected UNet for electron microscopy image segmentation," *IET Image Processing*, vol. 14, no. 12, pp. 2682–2689, 2020, doi: 10.1049/iet-ipr.2019.1527.
[27] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, vol. 39, no. 4, pp. 3431–3440, doi: 10.1109/CVPR.2015.7298965.
[28] S. Cai, Y. Tian, H. Lui, H. Zeng, Y. Wu, and G. Chen, "Dense-UNet: a novel multiphoton in vivo cellular image segmentation model based on a convolutional neural network," *Quantitative Imaging in Medicine and Surgery*, vol. 10, no. 6, pp. 1275–1285, 2020, doi: 10.21037/QIMS-19-1090.
[29] J. E. -Taraboulsi, C. P. Cabrera, C. Roney, and N. Aung, "Deep neural network architectures for cardiac image segmentation," *Artificial Intelligence in the Life Sciences*, vol. 4, pp. 1–19, 2023, doi: 10.1016/j.ailsci.2023.100083.
[30] A. Degerli *et al.*, "Early detection of myocardial infarction in low-quality echocardiography," *IEEE Access*, vol. 9, pp. 34442–34453, 2021, doi: 10.1109/ACCESS.2021.3059595.
[31] C. Chen *et al.*, "Deep learning for cardiac image segmentation: a review," *Frontiers in Cardiovascular Medicine*, vol. 7, pp. 1–33, 2020, doi: 10.3389/fcvm.2020.00025.
[32] J. A. U. -Moral *et al.*, "Contrast-enhanced echocardiographic measurement of left ventricular wall thickness in hypertrophic cardiomyopathy: comparison with standard echocardiography and cardiac magnetic resonance," *Journal of the American Society of Echocardiography*, vol. 33, no. 9, pp. 1106–1115, 2020, doi: 10.1016/j.echo.2020.04.009.
[33] O. Hamila *et al.*, "Fully automated 2D and 3D convolutional neural networks pipeline for video segmentation and myocardial infarction detection in echocardiography," *Multimedia Tools and Applications*, vol. 81, no. 26, pp. 37417–37439, 2022, doi: 10.1007/s11042-021-11579-4.
[34] G. Sanjeevi, U. Gopalakrishnan, R. K. Pathinarupothi, and T. Madathil, "Automatic diagnostic tool for detection of regional wall motion abnormality from echocardiogram," *Journal of Medical Systems*, vol. 47, no. 1, 2023, doi: 10.1007/s10916-023-01911-w.

[35] I. Adalioglu, M. Ahishali, A. Degerli, S. Kiranyaz, and M. Gabbouj, "SAF-Net: self-attention fusion network for myocardial infarction detection using multi-view echocardiography," in *Computing in Cardiology*, 2023, pp. 1–4, doi: 10.22489/CinC.2023.240.
[36] Y. Li, W. Lu, P. Monkam, Z. Zhu, W. Wu, and M. Liu, "LVSnake: accurate and robust left ventricle contour localization for myocardial infarction detection," *Biomedical Signal Processing and Control*, vol. 85, 2023, doi: 10.1016/j.bspc.2023.105076.
[37] A. Degerli, S. Kiranyaz, T. Hamid, R. Mazhar, and M. Gabbouj, "Early myocardial infarction detection over multi-view echocardiography," *Biomedical Signal Processing and Control*, vol. 87, pp. 1–12, 2024, doi: 10.1016/j.bspc.2023.105448.
[38] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, "Transfer learning: a friendly introduction," *Journal of Big Data*, vol. 9, no. 1, pp. 1–19, 2022, doi: 10.1186/s40537-022-00652-w.
[39] A. H. Zim *et al.*, "Smart manufacturing with transfer learning under limited data: towards data-driven intelligences," *Materials Today Communications*, vol. 37, 2023, doi: 10.1016/j.mtcomm.2023.107357.
[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017, doi: 10.1145/3065386.
[41] H. Li, M. Krček, and G. Perin, "A comparison of weight initializers in deep learning-based side-channel analysis," in *Applied Cryptography and Network Security Workshops*, Cham: Springer, 2020, pp. 126–143, doi: 10.1007/978-3-030-61638-0_8.
[42] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research*, vol. 9, pp. 249–256, 2010.
[43] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," *Arxiv-Computer Science*, vol. 1, pp. 1–15, 2015.

BIOGRAPHIES OF AUTHORS

Didik Bani Unggul earned his Bachelor of Science in Statistics from Universitas Indonesia, graduating in 2020. Currently, he is pursuing a master's degree in statistics at Institut Teknologi Sepuluh Nopember in Surabaya, Indonesia. Actively involved in projects at the Laboratory of Computational Statistics and Data Science, his research areas of interest include deep learning, biomedical image processing, and computational statistics. He can be contacted at email: 6003212005@student.its.ac.id or didikbaniunggul@gmail.com.

Nur Iriawan received a bachelor's degree in statistics from the Institut Teknologi Sepuluh Nopember (ITS) Surabaya, a master's degree in computer science from the University of Maryland, USA, and a Ph.D. in statistics from Curtin University of Technology, Australia. He is a professor at the Department of Statistics, Faculty of Science and Data Analytics, ITS, Surabaya. He also serves as the head the Laboratory of Computational Statistics and Data Science. He has supervised and co-supervised over 20 master and 10 Ph.D. students. He has authored or co-authored more than 60 Scopus articles, with 12 H-indexes and over 1,000 citations. His research interests encompass stochastic processes, statistical computations, and Bayesian models. He can be contacted at email: nur_i@statistika.its.ac.id.

Heri Kuswanto holds a Statistics B.Sc. (2003) and M.Sc. (2005) from Institut Teknologi Sepuluh Nopember, Indonesia, and a Dr.rer.pol. in statistics (econometrics) from Leibniz Hannover University, Germany (2009). He further pursued a postdoctoral degree at Laval University, Canada, focusing on the calibration of ensemble weather forecasts in 2010. Currently a professor in statistics at ITS, he also serves as the Director of Graduate Program and Academic Development. His academic career includes appointments as the Head of Climate Change Research Group. His research spans weather forecast, solar radiation management, computational statistics, time series forecasting, econometrics, machine learning, and advanced data analysis. He also received awards such as the Harvard Residency Program on Solar Geoengineering and DAAD Scholarship for Doctoral research in Germany. He can be contacted at email: heri_k@statistika.its.ac.id.