Accredited Ranking SINTA 2
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online on: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 7 No. 1 (2023) 153 - 160 ISSN Media Electronic: 2580-0760

Modification of SqueezeNet for Devices with Limited Computational Resources

Rahmadya Trias Handayanto[1], Herlawati[2]
[1]Computer Engineering, Faculty of Engineering, Universitas Islam 45 Bekasi
[2]Informatics, Faculty of Computer Science, Universitas Bhayangkara Jakarta Raya
[1]rahmadya.trias@gmail.com, [2]mrs.herlawati@gmail.com

Abstract

In recent years, the computational approach has shifted from a statistical basis to deep neural network architectures which process the input without explicit knowledge that underlies the model. Many models with high accuracy have been proposed by training the datasets using high performance computing devices. However, only a few studies have examined its use on non-high-performance computers. In fact, most users, who are mostly researchers in certain fields (medical, geography, economics, etc.) sometimes need computers with limited computational resources to process datasets, from notebooks, personal computers, to mobile processor-based devices. This study proposes a basic model with good accuracy and can run lightly on the average computer so that it remains lightweight when used as a basis for advanced deep neural networks models, e.g., U-Net, SegNet, PSPNet, DeepLab, etc. Using several well-known basic methods as a baseline (SqueezeNet, ShuffleNet, GoogleNet, MobileNetV2, and ResNet), a model combining SqueezeNet with ResNet, termed Res-SqueezeNet, was formed. Testing results show that the proposed method has accuracy and inference time of 84.59% and 8.46 second, respectively, which has an accuracy of 2% higher than the SqueezeNet (82.53%) and is close to the accuracy of other baseline methods (from 84.93% to 0.88.01%) while still maintaining the inference speed (below nine second). In addition, residual part of the proposed method can be used to avoid vanishing gradient, hence, it can be implemented to solve more advanced problems which need a lot of layers, e.g., semantic segmentation, time-series prediction, etc.

Keywords: deep learning, squeezenet, resnet, imagenet, convolutional layer

1. Introduction

Deep learning has now been implemented in all fields and various tools, from super computers, laptops to embedded systems. The performance of a deep learning method is no longer only in terms of accuracy, but also computational speed and efficiency because it must be applicable to devices with limited computational resources.

Due to the open contest, many new methods with good performance were proposed. For example, ImageNet Large Scale Visual Recognition Challenge (ILSVRC), has provided 1 million training data with 1000 classes, termed ImageNet [1]. Therefore, researchers in the world can easily propose the best methods using these benchmark datasets.

Pretrained models are widely available with certain programming language implementations, e.g., Python, MATLAB, etc. Figure 1 shows a comparison of pretrained models for various methods that are currently used with respect to accuracy, size, and speed [2].

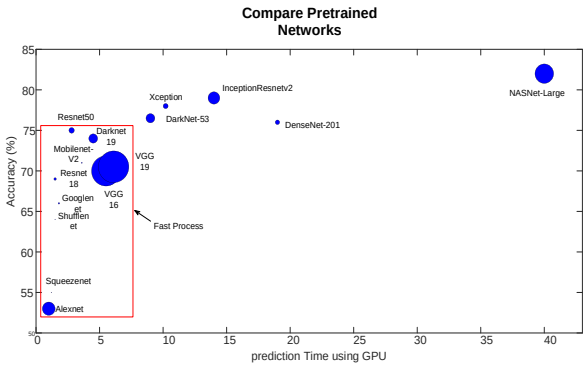

Figure1. Deep Learning Model Performance [2]

The development of these models shows an increase in accuracy but requires large Graphic Processing Unit (GPU) resources. Convolution-based methods, e.g., AlexNet [3] and Visual Geometry Group (VGG) [4], require large resources, so they are not suitable for devices with limited computational resources to process large datasets. This study focuses on the left side of Figure 1, namely AlexNet [3], GoogLeNet [5],

Accepted: 23-08-2022 | Received in revised: 13-01-2023 | Published: 03-02-2023

SqueezeNet [6], [7], ShuffleNet [8], ResNet [9], [10] and MobileNet [11] because they are suitable for devices with limited computational resources. There have been many models for devices with limited computational resources, e.g., MobileNetV2 and modifications of ShuffleNet, namely ShuffleNetV2 [12] and ShuffleNasNet [8], [12] to overcome MobileNetV2 which still requires large GPU resources.

Many researchers have modified ShuffleNet to get a better performance model but still light and fast. Modifications to other methods, i.e., SqueezeNet, are still rarely studied. Therefore, this study tries to modify SqueezeNet to obtain an accuracy that is not much different from other baseline models. As shown in Figure 1, this method has the lowest accuracy rate, even though it does not require too many GPU resources.

This research contributes to proposing a fast and accurate model from a simple basic model to be suitable for devices with limited computational resources, e.g., notebook, personal computer, etc. in a specific application from classification to object detection without the need of high-performance computing (supercomputers and computer clusters). There is a compromise between accuracy and speed.

The paper is organized as follows. After discussing the basic methods that are most widely applied in the industry, i.e., transfer learning and data augmentation, the methods that become the baseline for this research are discussed, including the data that will be used for testing.

The proposed model, Res-Squeeze Net, after being assembled, will be followed by a training process to produce a model that is ready to be tested. After the results and discussion section, the paper ends with a conclusion and future study.

2. Research Methods

Quantitative research is used in this study with several matrices to measure the performance of the proposed model. To see the performance, five baseline models were analyzed before creating a proposed model, including SqueezeNet, ShuffleNet, GoogLeNet, ResNet, and MobileNet. Each model has its strengths and weaknesses. These models are models developed by the researchers after the success of the early Deep Learning model, AlexNet.

After the analysis stage, the process of making a proposed model is carried out by taking the advantages of one model and anticipating its weaknesses with methods from other models so that a better model is obtained.

These baseline models have been trained using ImageNet datasets. Training results are available in many languages e.g., Python, MATLAB, etc.

2.1 Transfer Learning

ImageNet is a benchmark dataset consisting of one million images with 1000 labels/classes. The five baseline models have been trained using ImageNet on high performance computers (NVIDIA® Tesla® P100 and a mini-batch size of 128), so users around the world can take advantage of the pretrained models.

To maintain the previous training using ImageNet, when implemented in a new domain with different classes, the training results with ImageNet datasets can be preserved through a transfer learning mechanism [13]–[19]. We only trained with new datasets.

Figure 2 shows the transfer learning framework. First, certain models, such as GoogLeNet, are trained with one million images that have 1000 classes. After the training process the model is known as the pretrained model. In practice, not all classes are needed to solve certain problems. Transfer learning works by replacing the final layer, usually the fully connected layer, with a new layer according to the number of new data classes, for this research there are nine classes. To avoid the training results with the previous data (ImageNet datasets) still being stored, it is necessary to freeze the initial layer of the model. The training then runs using new data with a new number of classes. This method can be done for other Deep Learning models.

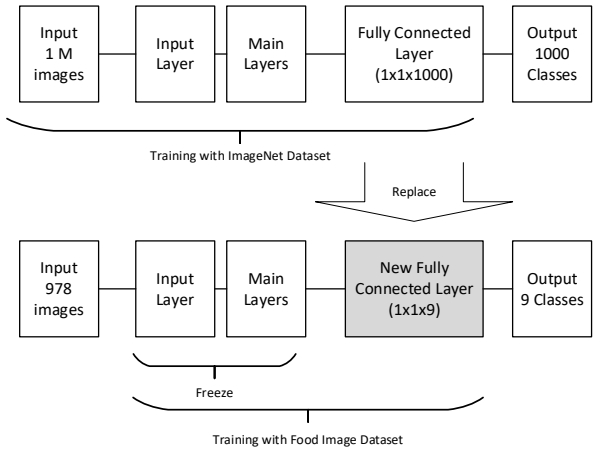

Figure 2. Transfer Learning

2.2 Data Augmentation

Some datasets, e.g., medical data, have a limited amount of data. Unfortunately, deep learning requires large training data to produce good performance. Therefore, it is necessary to add training data by making slight changes, e.g., rotation, translation, brightness enhancement, and many more, to increase the amount of training data properly.

For example, data with one rotation and translation for every image will doble the size of the training data. More additional modification i.e., brightness enhancement, flipping, and other methods also give

DOI: https://doi.org/10.29207/resti.v7i1.4446

more size of the training data which is very suitable to be applied to data that is difficult to obtain, such as x-rays, microscopic data, and others. However, for data that is easily obtained, it is better to use it directly because the results are certainly much better because too much augmentation invites over fitting [10].

Even though Deep Learning has a layer that have a function as filter, it's better to still pre-process the data. Of course, if the data being trained has very different characteristics from ImageNet, training from the beginning needs to be done, for example satellite images, photos of bacteria, cancer, and other medical data.

2.3 Learning

After replacing the fully connected layer with a layer that matches the training data as well as freezing the initial layers of the model, the training process is carried out. The hyper-parameters, i.e., training method, mini-batch size, maximum epoch, and initial learning rate are stochastic gradient descent with momentum (sgdm), 10, 6, and 0.0003, respectively. These parameters are set to all method for performance comparison.

The training process sets *w* to produce the smallest possible error with a loss function, can be seen in equation 1.

$$E = \sum_{X \in \Omega} \omega(x) log_{pl(x)} x \quad (1)$$

Where $pl(x)$ is the loss function of SoftMax, $l: \Omega \longrightarrow \{1, \ldots, k\}$ label value for each image. The SoftMax S is calculated by the equation 2.

$$S_j = \frac{\exp(y_j)}{\sum_{j=1}^{N} \exp(y_j)} \quad (2)$$

Where the final output of convolution Y = $(y_1,y_2,\ldots,Y_n)^T$, and output SoftMax layer S = $(s_1,s_2,\ldots,s_N)^T$.

All models will be trained using the same ordinary device, i.e., intel i5 CPU and GeForce MX130 GPU using MATLAB language. The accuracy metrics are overall accuracy and mean intersection over union (mIoU) calculated from confusion matrix using equation 3.

$$mIoU = \frac{1}{N_{class}} \sum_{i=1}^{N_{class}} \frac{TP(i)}{TP(i) + FP(i) + FN(i)} \quad (3)$$

Where TP, FP, FN, and N represent true positive, false positive, false negative, and number of classes, respectively. The overall accuracy is calculated by the equation 4.

$$Overall\ Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

The mIoU metric is more strict than overall accuracy since this metric takes out true negative.

The calculation of accuracy is done after the confusion matrix is created as a result of model validation. This matrix contains the number of true positives, true negatives, false positives, and false negatives.

Another important concept that underlies the emergence of Deep Learning is the ReLu function which is a linear function which is defined as equation 5 [20]:

$$ReLU = \begin{cases} x\ x \geq 0 \\ 0\ x < 0 \end{cases} \quad (5)$$

With the ReLu function, a Neural Network can have many layers, which was previously difficult for a sigmoid function or tangent sigmoid. So, Neural Networks can be run with a greater number of layers and give rise to the term Deep Learning or Deep Neural Networks.

Hardware developments have also triggered the rapid development of Deep Learning, namely the Central Processing Unit (CPU), Graphics Processing Unit (GPU), and memory. Google also introduced a new processing technique known as the Tensor Processing Unit (TPU) and is available in the Integrated Development Environment Google Collaboratory (https://colab.research.google.com/).

2.4 GoogLeNet

The model known as Inception version 1 is the winner of the 2014 ILSVRC contest. Convolution 1x1 is placed in the middle with the inception module. At the end, global average pooling is installed, instead of fully connected layer. Figure 3 shows the inception module of GoogLeNet. This model has 22 layers, more than its predecessors, CNN and VGG, which have 16 and 18 layers, but not as long as ResNet. An important aspect of GoogLeNet is inception which can be applied to other models such as DeepLab for semantic segmentation on the encoder section. GoogLeNet's competitors include AlexNet, ZFNet, SPPNet, and VGGNet.

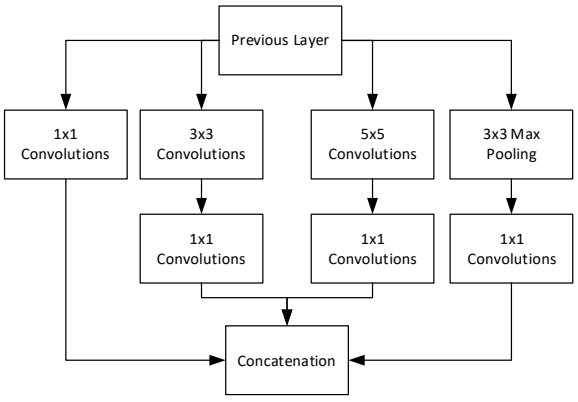

Figure 3. Inception Module

DOI: https://doi.org/10.29207/resti.v7i1.4446

2.5 Residual Network (ResNet)

ResNet uses a modified block from the convolution block of CNN or VGG. Residual blocks are formed to overcome the vanishing gradient problem in convolution-based models. Figure 4 shows one residual block on ResNet.

Weight layer consists of convolution process, including batch normalization and ReLu activation. A skip connection/shortcut maintains the gradient can be calculated in backpropagation which is difficult to do with pure convolution on CNN or VGG. As a result, ResNet can run well on models with many layers, even now ResNet101 with 101 layers shows a good accuracy without the problem of vanishing gradients (see Figure 1).

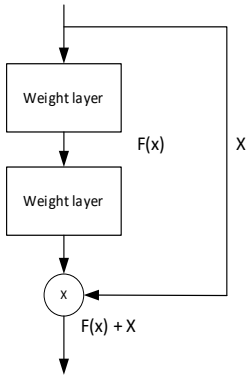

Figure 4. Residual Block Architecture

Vanishing gradient limits the number of convolution layers. With residual network blocks, ResNet can have many layers, for example 50 and 150 in ResNet 50 and ResNet101, respectively. In this study as a comparison is ResNet18 which is the shortest version of ResNet. Although it has many layers, ResNet is very fast because it only performs the convolution process on the residual side, while on CNN and VGG convolution is carried out on the main line (Figure 4).

2.6 MobileNet

This model was originally used to detect objects. Google researchers were trying to replace the heavy CNN with a new model that could run on phones, namely MobileNet. This research uses MobileNet version 2 where pointwise and depth-wise convolution in version 1 is still used and added two new layers, namely liner bottleneck and shortcut connections. Figure 5 shows depth wise convolution which is keep the channels and point-wise convolution that merge the channels.

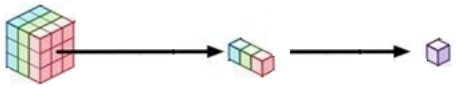

Figure 5. Depth-Wise Convolution (left arrow) and Point-Wise Convolution (right arrow)

Previous research shows MobileNetV2 improves 5% accuracy over the previous version. The liner bottleneck and shortcut connection work together to avoid the nonlinearity that breaks feature maps [21].

2.7 ShuffleNet

ShuffleNet is a method to reduce the computational complexity of a system. As the name implies, this method applies shuffle and point-wise group convolution. This method is 13 times faster than the original Deep Learning model, namely AlexNet [8], [12], [22]. Figure 6 shows the main block of ShuffleNet. Shuffle performs switching from one channel to another in the channel shuffle.

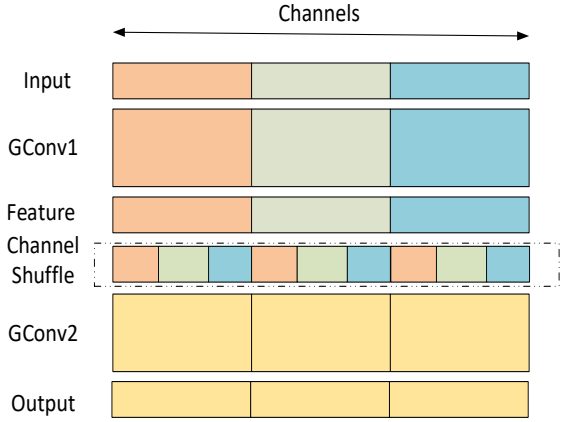

Figure 6. Channel Shuffle Structure in ShuffleNet

Previous study showed that channel shuffle had a lower error than no-channel shuffle [8]. This model is fast and light that it can compete with MobileNetV2. Much effort has been made to improve the accuracy of ShuffleNet, hence, this method also appropriate not only on mobile devices implementation [12], [22].

2.8 SqueezeNet

SqueezeNet as the name implies intends to squeeze the model to produce a compact model so that it can be implemented on embedded devices. SqueezedNet's parameter count is 50x smaller than AlexNet [6], [23], [24]. Figure 7 shows the Squeeze Structure in SqueezeNet.

Basically, the 'fire' part of SqueezeNet is convolution with the difference in filter size between the left and right sides, namely 1x1 and 3x3. This section re-expands the feature to be repeatedly squeezed back on the next layer eight times.

There are three strategies to design the SqueezeNet architecture, including: i) replace 3x3 filter with 1x1 filter, ii) decreasing the number of input channels to 3x3 filter, and iii) down sampling late in order to generate large activation maps in the convolution section. However, currently strategies (i) and (ii) have been

DOI: https://doi.org/10.29207/resti.v7i1.4446

shown to reduce convolution parameters while maintaining accuracy.

After the dropout process, the convolution is carried out again before entering SoftMax to carry out the classification process. This model is still rarely developed further, so it is necessary to find a way to increase the accuracy to be on par with other models without sacrificing speed and other aspects that are suitable for devices with limited resources, such as size and number of parameters. Only some modification in bypassing each block that was proposed by previous researchers, termed Vanilla SqueezeNet [7]. This model has successfully been implemented in a vehicle.

The more parameters, the more computations required. However, with the development of parallel processors, metrics in the form of floating operations (FLOPs) are no longer relevant, what is needed is the inference speed of the designed model. Sometimes even though the number of parameters is large because they are run in parallel, the results are faster than those with few parameters but serially.

For embedded devices, this method is the main choice where wiring requires computational efficiency. All we have to do is create the right training data to produce an accurate SqueezeNet model. Several terms need to be understood in formulating a SqueezeNet model. As shown in Figure 7, the parameters that need to be considered are filter size, number of filters, stride, dilation factor, padding, padding value, weight learn factor, weightL2factor, bias learn rate factor and bias L2 factor. Each layer in Figure 7 should be defined its parameter in a simple form.

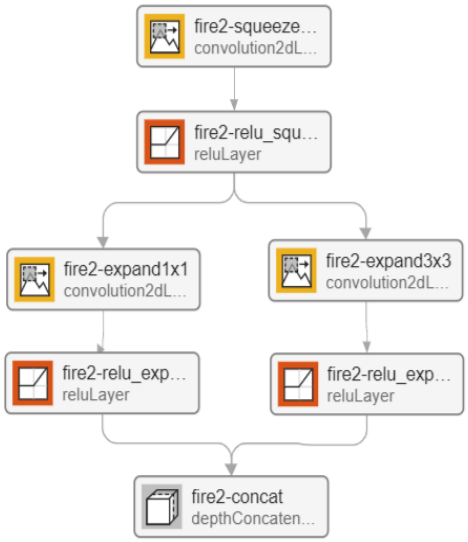

Figure 7. Squeeze Block Structure

SqueezeNet has the fire block consisting of one 'fire' which is squeezed into 1x1 feeding into an expand layer consisting of mixing of 1x1 and 3x3 convolution filters. The basic principle is that SqueezeNet maintains the number of channels in the model to keep the parameter as small as possible that important for devices with limited computational resources implementation. This model will be used in the proposed model as the main network basis. The weakness of this model i.e., low accuracy, will be compensated by other method, discussed in the following section.

2.4 Proposed Model

Figure 8 shows the proposed model. The rationale of the proposed model is based on the fast of SqueezeNet and the accuracy of ResNet18.

The proposed model uses eight fire blocks to squeeze the blocks that maintains the channels not exceeding 128. Two residual blocks are attached to the end of the fire block. The two residual blocks are inspired by ResNet18. Such block performs the filtering process with convolution but in the remaining part (see Figure 4 about the ResNet block).

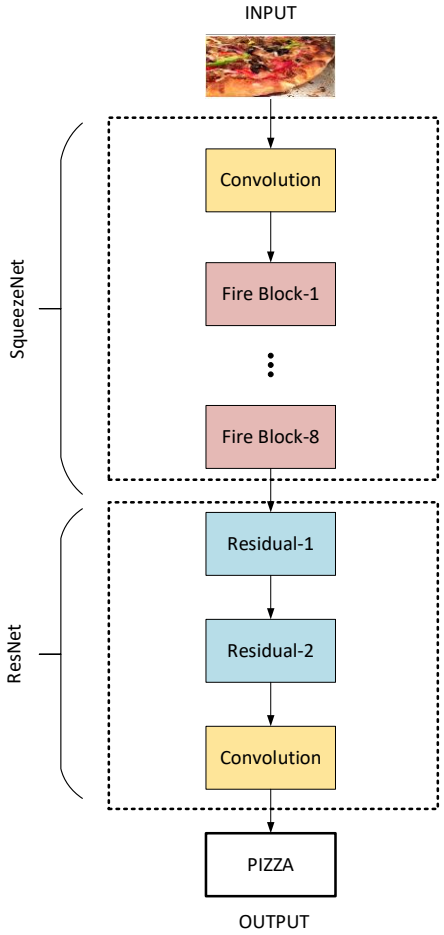

Figure 8. Res-SqueezeNet

Skip connection on the ResNet18 speeds up the computing process while avoiding the vanishing gradient problem that often occurs in convolution-based

DOI: https://doi.org/10.29207/resti.v7i1.4446

models i.e., CNN [10], [12], [24]–[26] and VGG [6], [14], [27] and limits the number of layers.

Since the proposed model does the prediction, the result is the class of the predicted image, for example in the example above is the pizza class. For other problems such as object detection and semantic segmentation, modifications are needed. For example, for semantic segmentation, it is necessary to have a module to break the image into small parts and then classify the small parts based on certain pixel classes. Meanwhile, for object detection, it uses a box that shows the class of an object. This paper only focuses on object classification.

The input layer has a function to make the input size equal to the model size, which is 227x227 with Red, Green, and Blue (RGB) channels.

To produce better accuracy, several the residual block can be added, but of course it will have a big size (file size and number of parameters). Here we use only two different residual blocks including a branch with two residual blocks and a branch with a faster skip connection.

2.5 Datasets

This study used food image datasets from Mathworks [28]. This dataset consists of 978 photographs of food in nine classes (caesar_salad, caprese_salad, french_fries, greek_salad, hamburger, hot_dog, pizza, sashimi, and sushi).

To ensure the proposed model can be run well, the network analyzer tools was used (Figure 9). A dot in network analyzer represents the layer and the arrows show the connection. If there is an error, e.g., missing layer size and other parameter values, the model cannot be trained. Network analyzer also shows the parameters e.g., number of layers, activation, learnable layer, warnings, errors, etc. After there is no warning and error, the training process can be done using a proper data training.

The code used in this paper was based on: https://www.mathworks.com/help/deeplearning/gs/get-started-with-transfer-learning.html. There are some important modifications in the code for the proposed model (Res-SqueezeNet). The data should be prepared appropriately.

Some baseline methods have different input size standards. Therefore, in the input layer, it is necessary to process the image input has the same size with the input layer of the model. The size of input model has been set following the pretrained model (227-by-227, 224-by-224, 299-by-299, 256-by-256, and 331-by-331).

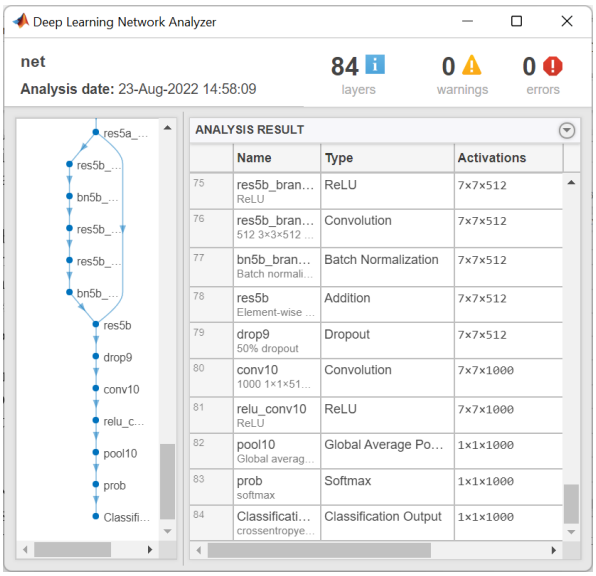

Figure 9. Network Analysis Result of Proposed Model

3. Results and Discussions

After processing the datasets with data training and validation, the training process has done for all models. Figure 10 shows the training process of the proposed models. After 6 epochs with 108 iterations, the accuracy calculation showed 84.59% accuracy and mIoU value of 0.6312.

For implementation, users are advised to add epochs and iterations considering that the image under the gradient still shows an improvement. However, six epochs is good enough to compare the six models in this paper.

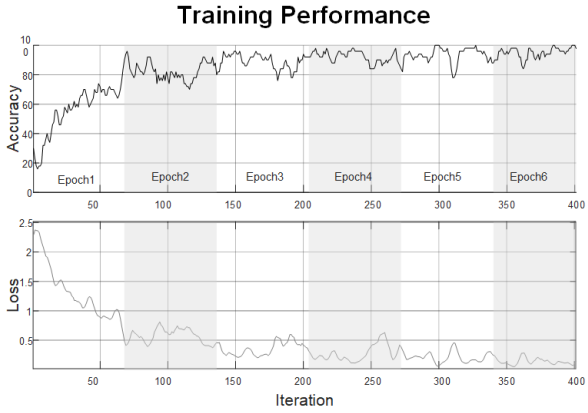

Figure 10. Training Result of Proposed Model

Table 1 shows the performance of the six models. The proposed model (Res-SqueezeNet) showed better accuracy than the SqueezeNet (increase 2%) and keep the inference time in 8 second which is better than ResNet18. The attachment of Residual block in the end of Res-SqueezeNet layer improved the accuracy. The training process is fast, less than five minutes with an i5

DOI: https://doi.org/10.29207/resti.v7i1.4446

processor and an Nvidia GeForce MX130 GPU that runs on the Windows 11 operating system.

GoogLeNet in Table 1 shows best performance in overall accuracy calculation, but the best mIoU goes to MobileNetV2. As seen in equation 3, mIoU is more accurate in measuring the accuracy performance of a model and has become a standard metric for accuracy calculations.

Table 1. Models Performance

No	Model	Accuracy/ *mIoU***	Inference (s)	Size (Mb)
1.	SqueezeNet	0.8253/*0.5405*	8.127	**2.61**
2.	ShuffleNet	0.8493/*0.6388*	**7.281**	3.25
3.	Google Net	0.8801/*0.6735*	8.603	21.3
4.	ResNet18	0.8733/*0.6463*	11.581	39.77
5.	MobileNetV2	0.8767/***0.6884***	13.692	25.2
6.	Res-SqueezeNet*	0.8459/*0.6312*	8.460	44.28

Results in bold text represent the best value
* The proposed model
** Results in italic represent mIoU value

The problem of the proposed models is the size of network (44.28 Mb), higher than ResNet and 20 times higher than SqueezeNet; hence, this model does not appropriate for embedded system, but can be run on notebook/laptop, PC, tablet, etc. However, the characteristics of residual network can maintain the speed of inference process (8.460 second) as well as keep the vanishing gradient problem of ordinary convolution layer (the characteristic of ResNet). The Residual block in the proposed model increase the size of network. However, the residual network has distinctive characteristics that are quite good in solving semantic segmentation problems, for example DeepLab which uses ResNet as its main component.

For implementation, an Open Neural Network Exchange (ONNX) module can be used to generate code for other programming languages such as Python, Caffe, and others.

4. Conclusion

Current research on Deep Learning is not just about accuracy. Apart from the transparency of the Deep Learning model, one aspect that needs to be considered in Deep Learning research is the ease of implementation, especially on devices with limited computational resources. The use of residual block in the end of SqueezeNet layer before SoftMax layer give a higher accuracy result. Also, the characteristic of ResNet keep the model running fast that suitable for semantic segmentation, e.g., inside the DeepLab. The accuracy of proposed model (Res-SqueezeNet) was 2% higher than the SqueezeNet and faster than ResNet. The Res-SqueezeNet was also almost as accurate as ShuffleNet. In the future, the study will focus on reducing the size of the proposed model and implement the proposed method in semantic segmentation.

Acknowledgment

The authors thanks to Universitas Islam 45 Bekasi and Universitas Bhayangkara Jakarta Raya for the support of the research.

Reference

[1] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 10691–10700, 2019.

[2] Mathworks, "Pretrained Deep Neural Networks," 2022. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html. [Accessed: 21-Aug-2022].

[3] M. Z. Alom *et al.*, "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," 2018.

[4] R. Zhang, G. Li, M. Li, and L. Wang, "Fusion of images and point clouds for the semantic segmentation of large-scale 3D scenes based on deep learning," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 143, no. April, pp. 85–96, 2018.

[5] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[6] A. A. Ardakani, A. R. Kanafi, U. R. Acharya, N. Khadem, and A. Mohammadi, "Application of deep learning technique to manage COVID-19 in routine clinical practice using CT images: Results of 10 convolutional neural networks," *Computers in Biology and Medicine*, vol. 121, no. April, p. 103795, 2020.

[7] H. J. Lee, I. Ullah, W. Wan, Y. Gao, and Z. Fang, "Real-Time vehicle make and model recognition with the residual squeezenet architecture," *Sensors (Switzerland)*, vol. 19, no. 5, 2019.

[8] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016.

[10] A. Victor Ikechukwu, S. Murali, R. Deepu, and R. C. Shivamurthy, "ResNet-50 vs VGG-19 vs training from scratch: A comparative analysis of the segmentation and classification of Pneumonia from chest X-ray images," *Global Transitions Proceedings*, vol. 2, no. 2, pp. 375–381, 2021.

[11] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *Computer Vision and Pattern Recognition*, 2017.

[12] N. Ma, X. Zhang, H. T. Zheng, and J. Sun, "Shufflenet V2: Practical guidelines for efficient cnn architecture design," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11218 LNCS, pp. 122–138, 2018.

[13] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic," *Measurement: Journal of the International Measurement Confederation*, vol. 167, no. July 2020, p. 108288, 2021.

[14] 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.

[15] Z. Benbahria, M. F. Smiej, I. Sebari, and H. Hajji, "Land cover

DOI: https://doi.org/10.29207/resti.v7i1.4446

intelligent mapping using transfer learning and semantic segmentation," *7th Mediterranean Congress of Telecommunications 2019, CMT 2019*, pp. 1–5, 2019.

[16] M. Imad, O. Doukhi, and D. J. Lee, "Transfer learning based semantic segmentation for 3d object detection from point cloud," *Sensors*, vol. 21, no. 12, pp. 1–15, 2021.

[17] J. Yang, Y. Q. Zhao, and J. C. W. Chan, "Learning and Transferring Deep Joint Spectral-Spatial Features for Hyperspectral Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 8, pp. 4729–4742, 2017.

[18] I. Z. Matovinovic, S. Loncaric, J. Lo, M. Heisler, and M. Sarunic, "Transfer learning with U-net type model for automatic segmentation of three retinal layers in optical coherence tomography images," *International Symposium on Image and Signal Processing and Analysis, ISPA*, vol. 2019-Septe, pp. 49–53, 2019.

[19] M. Á. Molina, G. Asencio-Cortés, J. C. Riquelme, and F. Martínez-Álvarez, "A Preliminary Study on Deep Transfer Learning Applied to Image Classification for Small Datasets," *Advances in Intelligent Systems and Computing*, vol. 1268 AISC, pp. 741–750, 2021.

[20] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into Deep Learning," *arXiv preprint arXiv:2106.11342*, 2021.

[21] N. D. Trung, T. T. Ngoc, and H. X. Huynh, "Automated Pneumonia Detection in X-Ray Images Via Depthwise Separable Convolution Based Learning," pp. 1–8, 2020.

[22] K. A. Laube and A. Zell, "ShuffleNASNets: Efficient CNN models through modified Efficient Neural Architecture Search," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2019-July, 2019.

[23] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," pp. 1–13, 2016.

[24] J. Wang, P. Lv, H. Wang, and C. Shi, "SAR-U-Net: Squeeze-and-excitation block and atrous spatial pyramid pooling based residual U-Net for automatic liver segmentation in Computed Tomography," *Computer Methods and Programs in Biomedicine*, vol. 208, pp. 1–25, 2021.

[25] P. K. Gadosey *et al.*, "SD-UNET: Stripping down U-net for segmentation of biomedical images on platforms with low computational budgets," *Diagnostics*, vol. 10, no. 2, 2020.

[26] I. L. Kharisma, R. T. Handayanto, and D. A. Dewi, "Face Mask Detection In The Covid-19 Pandemic Era by Implementing Convolutional Neural Network and Pre-Trained CNN Models," in *2021 IEEE 7th International Conference on Computing, Engineering and Design (ICCED)*, 2021, pp. 1–6.

[27] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[28] Mathworks, "Data Sets for Deep Learning," 2022. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/data-sets-for-deep-learning.html. [Accessed: 21-Aug-2022].

DOI: https://doi.org/10.29207/resti.v7i1.4446