Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 6, No. 3, August 2021, Pp. 249-258

Early detection of covid-19 patient's survivability based on the image of lung x-ray image using deep neural networks

Hilmy Bahy Hakim[1], Fitri Utaminingrum [*2], Agung Setia Budi[3]
Computer Vision Research Groups, Faculty of Computer Science, Brawijaya University, Indonesia[1,2]
Faculty of Computer Science, Brawijaya University, Indonesia[3]

Article Info
Keywords:
Covid-19, Lung, X-Ray, Image Processing, CNN, Deep Neural Networks

Article history:
Received: May 21, 2021
Accepted: June 18, 2021
Published: August 31, 2021

Cite:
Bahy Hakim, H., Utaminingrum, F., & Setia Budi, A. (2021). Early Detection of COVID-19 Patient's Survavibility Based On The Image Of Lung X-Ray Image Using Deep Neural Networks. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 6(3). https://doi.org/10.22219/kinetik.v6i3.1265

*Corresponding author
Fitri Utaminingrum
E-mail address:
f3_ningrum@ub.ac.id

Abstract
SARS-CoV-2 causes an infection called COVID-19, which is caused by a new coronavirus. One of the symptoms that are dangerous to the patients is developing pneumonia in their lungs. To detect pneumonia symptoms, one of the newest methods is using CNN (Convolution Neural Networks). The problem is when able to detect pneumonia, the patient's survivability, which knowing this will be helpful to decide the priority for each patient, is still in question. The CNN used to classify the patient's future condition in this research but met some major problems: the dataset is very few and unbalanced. The dataset for this research was taken from https://www.kaggle.com/bachrr/covid-chest-xray. However, we are not using all the images from the dataset. Only the data have value in 'survival' columns, 116 images in total, 94 of them will be used for the training dataset. The image augmentation was used to multiply the training data, resulted in 1429 images, and class weight was applied to prevent miscalculation on minority class. 6 CNN architectures were used to find the best model. The result VGG19 architecturehas the best overall accuracy, in training, it has 80% accuracy, 89% accuracyinvalidation, and 82% *f1* score accuracy on classifying the testing dataset means the best model if looking for accuracy on prediction, but this cost a prediction time that longest compared to other CNN architectures. MobileNet is the fastest, but it cost much worse on prediction accuracy, only 55%. The ResNet50 model has balanced prediction accuracy/time. It got 77% *f1* accuracy and 8.49 seconds of prediction time, 9 seconds less than VGG19.

1. Introduction

Coronaviruses are a form of the virus that can cause infection and, as a result, sickness, ranging from the common cold to serious diseases such as Severe Acute Respiratory Syndrome (SARS) and Middle East Respiratory Syndrome (MERS). SARS-CoV-2 causes an infection called COVID-19, which is caused by a new coronavirus. According to a World Health Organization (WHO) report, COVID-19 viruses, such as SARS, cause open holes in the lungs [1]. In December 2019, an outbreak of SARS-CoV-2 (formerly 2019-nCoV) infection was discovered in Wuhan, Hubei Province, China [2][3]. The patients that suffered the disease that went into the Intensive Care Unit had pneumonia. High cytocines were recorded in the x-ray of the patient's lung images, making them have a lesser 'black area' [4].

One of the newest methods for detecting pneumonia symptoms is using CNN (Convolution Neural Networks) to help extract features from the x-ray images. Then the extracted feature is used to classify the images. In the research involving 5000 images of chest X-Ray, CNN proved to gain high accuracy in classification tasks between normal images and pneumonia images, gained 74%-94% accuracy [5]. As the pandemic started, this method was applied to detect COVID-19 infection that caused pneumonia symptoms and classify the COVID-19 infected pneumonia cases with other pneumonia cases. The research using a merged dataset with 8474 datasets that only 445 of which are COVID-19 cases that caused an imbalanced dataset. With applying weight, the model can generate training accuracy up to 94% and classification accuracy of 95% with the vgg16 model [6]. Besides the VGG16 model, Residual Network also was used similar research to classify normal lungs with pneumonia lung to find the best model for the task [7].

Based on those research, we want to improve the detection to a more advanced level, not only detecting those infected with COVID-19 or not but also supplying more information about the patients. In this research, we want to make a classification system that can predict the survivability condition of the patients so that the doctor could decide the priority for each patient. It is important because the death rate of those who developed pneumonia and were sent into the Intensive Care Unit is still high, around 23% in the US [8].

A dataset supplied the information about the final condition of the patients with pneumonia cases (mostly because of COVID-19) [9]. The metadata had survival columns that provide useful information to our research, but many of them were left blank. We use only the data that have value in 'survival' columns. The problem with the dataset is theamount

is so small, 116 images total that match our requirement to becoming dataset of this research, and when splittinginto two classes for classification purposes, the dataset is also imbalanced, around 1:4. These problems will lead the CNNmodel into a massive over-fitting problem, reducing the training, validation, and testing accuracy.

To prevent this, the image augmentation technique and adding a dropout layer could be the solutions. Research that classifies the face dataset using CNN, containing only 500 images, able to gained a good accuracy of 58% with help from the augmentation technique by flipping the images [10]. More aggressive augmentation techniques are also applicableto improve training and classification for a much lesser dataset, for example, if the dataset was lower than 100 [11]. Because the dataset used in this research is fewer than that and suffers from imbalance, class weight and augmentation with more parameters such as rotation will be applied when training the model to improve the performance further. We also use six different models, VGG16, VGG19, ResNet50, ResNet101, ResNet152, and MobileNet, to find the most suitable model with the best outcome.

2. Research Method

2.1 Proposed Method

The proposed methods consist of five main parts such as illustrated in Figure 1. There are inputs in the form o an image dataset of a chest x-ray. The pre-processing stage consists of augmenting the image, so the dataset becomes larger in number. The CNN Training process is a process to train the CNN models, VGG16, VGG19, ResNet50, ResNet101, ResNet152, and MobileNet. CNN Testing is a process to classify the testing dataset using a trained CNN model. The final step is to display the output of model accuracy and the testing accuracy alongside their training and classification time in second.

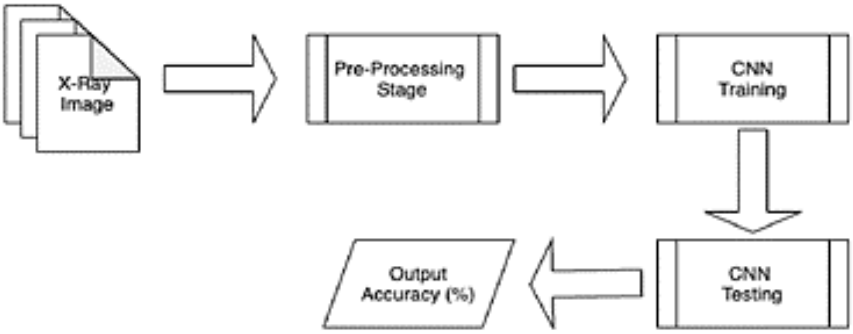

Figure 1. The Block System of Proposed Method

2.2 Pre-Processing

The data used 116 lung images that suffered pneumonia with various causes (mostly COVID-19). Every image has a parameter called survivability. The images are then split into two classes based on the value of the parameter. From each class, 20% of the images that belong to COVID-19 patients, 22 images are taken into the “testing images” directory, these images will be used to testing the process later that must be hidden from the training process, 94 images are taken into “training images” directory.

All datasets that belong to the training directory are augmented before being used for the CNN training process. In supervised learning, dataset augmentation is the practice of applying a wide range of domain-specific transformations to synthetically expand a training set, already become standard practice [12]. In this research, we are using ImageDataGenerator API by Keras [13] to generate more images depending on what hyperparameters we are using. Those parameters could be viewed in Table 1.

Table 1. Image Data Generator Parameter

Variable	Value	Before Augmentation	After Augmentation
Horizontal Flip	True	94	1429
Brightness Range	0.15 … 1.5	94	1429
Rotation Range	0-20 degree	94	1429
Fill Mode	Nearest	94	1429

Augmentation process creating multiple images based on the original image and the combination of Image Data Generator Parameter values. The sample of original images and the result of the augmentation process is shown in Figure 2, the leftmost images are the original image, and the augmented images are in the right of it. ImgeDataGenerator fromKeras was used in this augmentation process. This API works by taking the original image as input, then generating

multiple images. The appearance of the augmented images depends on what parameter that used. In this research, we use the parameter in Table 1, and this process will be looped over by a certain number called a batch. In this research, we set the batch to 15, so the API will generating around 15 augmented images per one original image, the three examples of theaugmented presented in Figure 2 right. A result is a growing number of the training dataset from 94 images to 1429 images divided into two classes. Class 0, where the survivor lung images belong to has 1083 images, and Class 1, where the lung images belong to those who didn't survive, has 346 images.

Figure 2. Original Images of Lung Images and The Result of Augmentation Process

Original Images vary in size and format, so resizing and reformatting are committed so we can simplify the system to receive input images of the same size and format. These images are reformatted into .jpg (Joint Photographic Group) because it is smaller in size than .png, which could save some memory usage used in the training process. The resolution of the images will be resized into 100x100 pixels, applied to all images both on training or testing directory. The images are then loaded into NumPy arrays, the standard representation for numerical data, and the purpose is to enable efficient implementation of numerical computations in a high-level language [14].

2.3 Implementation of CNN

CNN is a Deep Learning architecture based on the ability of humans to recognize objects and their surroundings quickly. If computers can recognize or classify objects and their surroundings using low-level features, they can use a series of convolutional layers to build a more abstract concept of what they know. As a result, Convolutional Neural Network is the name given to this network structure [15]. By using Deep Learning, we can eliminate building a feature extraction model because it can automatically manipulate features [16]. Each process consists of three layers, namely: convolution layer, activation function layer, and pooling layer [17]. CNN architecture comprises of a series of convolutional layers with activation functions and a continuation of the previous structure. The image is smaller, but it is more detailed. Following the convolutional and pooling layers, a feed-forward neural network with fully connected layers and an activation function is added, followed by a predictive layer with a softmax function that displays the estimated probability [18]. CNN architecture is displayed in Figure 3.

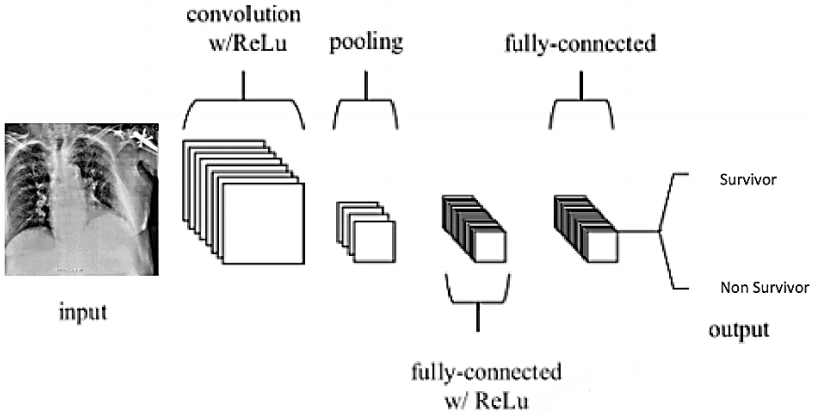

Figure 3. Convolutional Neural Network Architecture

Convolution is a mathematical operation on two functions (*f* and *g*) that results in a third function that expresses how the shape is changed by the other. Convolution, in other words, is a method of applying a filter to one function and obtaining the resulting function. The concept of convolution is widely used in a variety of computer sciencefields. Because

we're dealing with images in computer vision, the original function is a 2D image, and the filter we wantto apply is also 2D. Convolution operation for a 2D image (*H*) and 2D filter (kernel) *F* could be viewed in Equation 1.

$$G[i,j] = \sum_{u=-k}^{k} \sum_{u=-k}^{k} H[u,v]F[i-u,j-v] \quad (1)$$

2.3.1 VGG Architecture

In their paper, K. Simonyan and A. Zisserman from the University of Oxford proposed the VGG16 convolutional neural network model [19]. In ImageNet testing, a dataset of over 14 million images belonging to 1000 classes, the model achieves 92.7 percent, top-5 in test accuracy. It was one of the most well-known models submitted to the 2014 ILSVRC. It outperforms AlexNet by sequentially replacing large kernel-size filters (11 and 5 in the first and second convolutional layers) with multiple three by three kernel-size filters. VGG16 had been trained for weeks on NVIDIA Titan Black GPUs. VGG16 has 16 layers and 138 million parameters. Meanwhile, VGG19 has 19 layers and 144 million parameters. The VGG16 and VGG19 architecture could be viewed in Figure 4

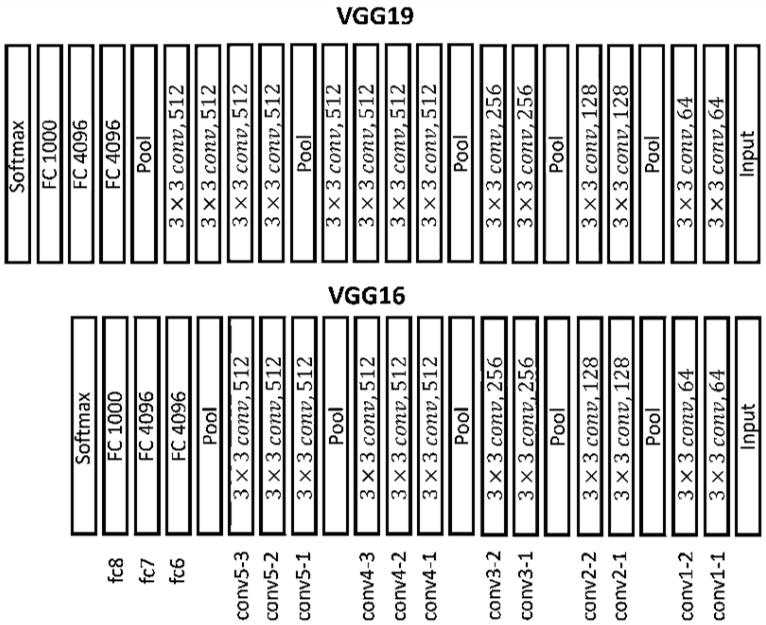

Figure 4. VGG16 and VGG19 Architecture

2.3.2 ResNet Architecture

Residual Networks (ResNets) are deep convolutional networks in which the basic idea is to use shortcut connections to skip blocks of convolutional layers [20]. The two numbers (ResNetXX) means the number of layers. Therefore this ResNet50 using 50 layers and having over 23 million trainable parameters, ResNet101 using 101 layers and having over 44 million trainable parameters, and ResNet152 using 152 layers and having over 60 million trainable parameters. ResNet's fundamental building block is a Residual block, which is repeated throughout the network. This residual block is illustrated in Figure 5.

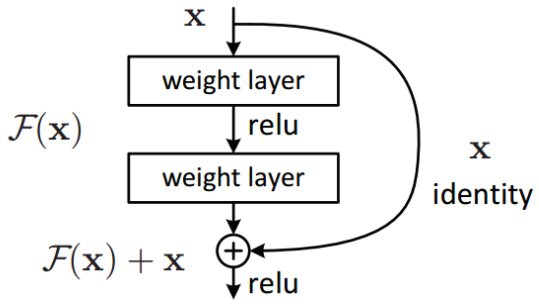

Figure 5. Residual Block on ResNets Architecture

2.3.3 MobileNet Architecture

MobileNet uses Depthwise separable convolutions. Compared to a network with regular convolutions of the same depth in the nets, it significantly reduces the number of parameters. As a result, lightweight deep neural networks are created [21]. As shown in Figure 6, the main difference between MobileNet and traditional CNN architecture is that instead of a single 3x3 convolution layer (left), batch normalization and ReLU are used. Mobile Nets (right) split the convolution into a 3x3 depth-wise convolution and a 1x1 pointwise convolution. MobileNet at default settings has 4.2 million parameters and could be reduced by tuning the width hyperparameter.

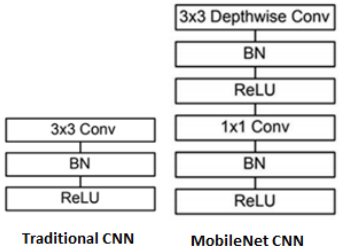

Figure 6. MobileNet Architecture (Right) Compared to Traditional CNN (Left)

2.3.4 Softmax Classifier

The cross-entropy loss is used by the Softmax classifier [22]. The Softmax classifier gets its name from the softmax function, which squashes raw class scores into normalized positive values that sum to one to apply thecross-entropy loss. The softmax activation formula could be viewed in Equation 2.

$$f(x_i) = \frac{\exp(x_i)}{\sum_j \exp\left(x_j\right)} \tag{2}$$

Where:
$f(x_i)$ = the result of the softmax activation function
X_i = class number i (i=1,2...) in this research we are using 2 class
j = the value of the vector

2.3.5 Class Weight

One of the main problems with the dataset that we are using for this research is a class imbalance [23]. Class 0, where the survivor's lungs belong to has a higher occurrence compared to class 1. It will greatly affect prediction tasks because most machine learning assumes that data between classes are evenly distributed. Sklearn libraries have 'class_weight' [24][25] that will be helpful to handle such a problem by manually assign each class weight value using the formula at Equation 3.

$$w_j = n\_samples \,/\, (n\_classes * n\_samplesj) \tag{3}$$

Where:
w_j is the weight for each class (j signifies the class)
n_samples is the total number of samples or rows in the dataset
n_classes is the total number of unique classes in the target
n_samplesj is the total number of rows of the respective class

2.4 Hardware & System Configuration

Every CNN run has to be done in the same device and software specification to get the result fairly. For this research, the setup, software, and hardware used for the experiment as displayed in Table 2.

Table 2. Hardware & System Configuration

Variable	Specification
Processor	Core i7 4500U @ 1.80 GHz
RAM	8 GB DDR3
GPU	NVIDIA GT 750M 2 GB of VRAM
Operating System	Microsoft Windows 10 Pro
Virtual Environment for Python Code	Anaconda with Python 3
Software Used for Training the CNN	Tensorflow GPU 1.15

3. Results and Discussion

The first step is to train the CNN model using various architecture models. In this research, the training process will run 6 times in total to train the model using 6 different architectures, VGG16, VGG19, ResNet50, ResNet101, ResNet152, and MobileNet. The training process involves 1071 samples and validation on 358 samples and runs 15 epochs for each model architecture with class weight value applied. This training process is a loss and accuracy graph on both the training and validation process. A loss curve is one of the most commonlyused plots to debug a neural network [26]. It provides an overview of the training process as well as the network's learning trajectory. The higher the loss, the model will be worse. An accuracy curve is another commonly used curve tounderstand the progress of Neural Networks, and the higher accuracy indicates a better model. The loss graph on training could be viewed in Figure 7, and the validation loss graph in Figure 9. The training accuracy graph could be viewed in Figure 8, and the validation accuracy in Figure 10. The training time is also provided in Figure 11 because it is also one of the performance parameters used to know which architecture has the best performance.

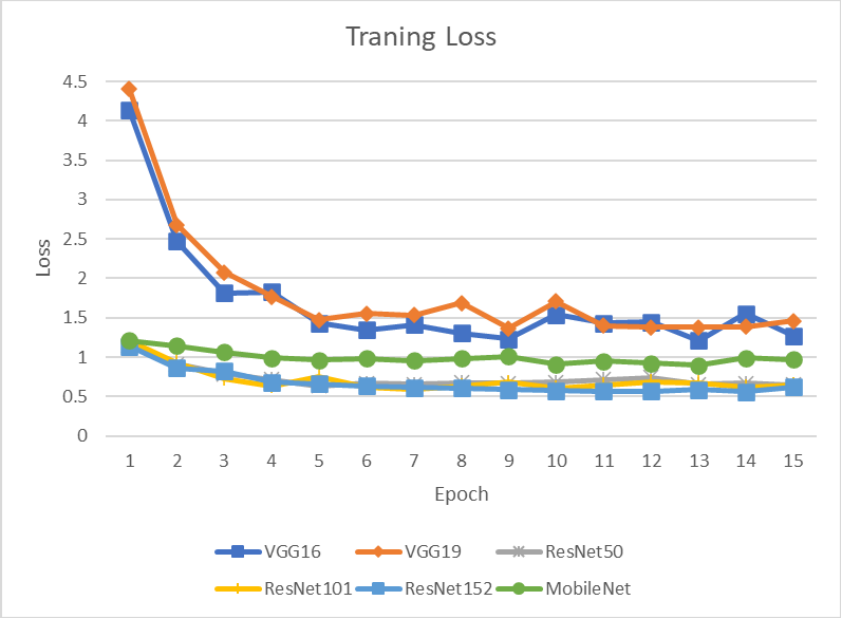

Figure 7. Training Loss Graph on Each CNN Architecture

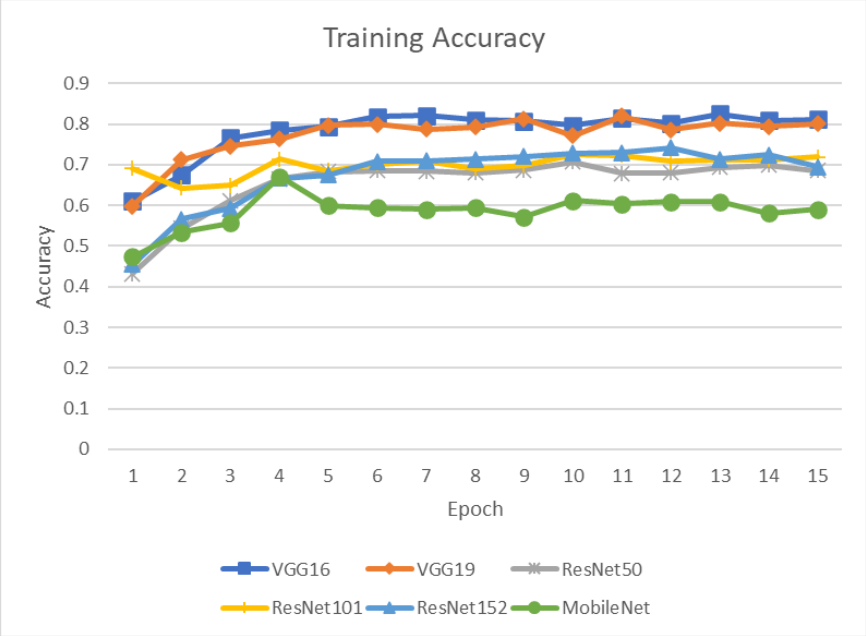

Figure 8. Training Accuracy Graph on Each CNN Architecture

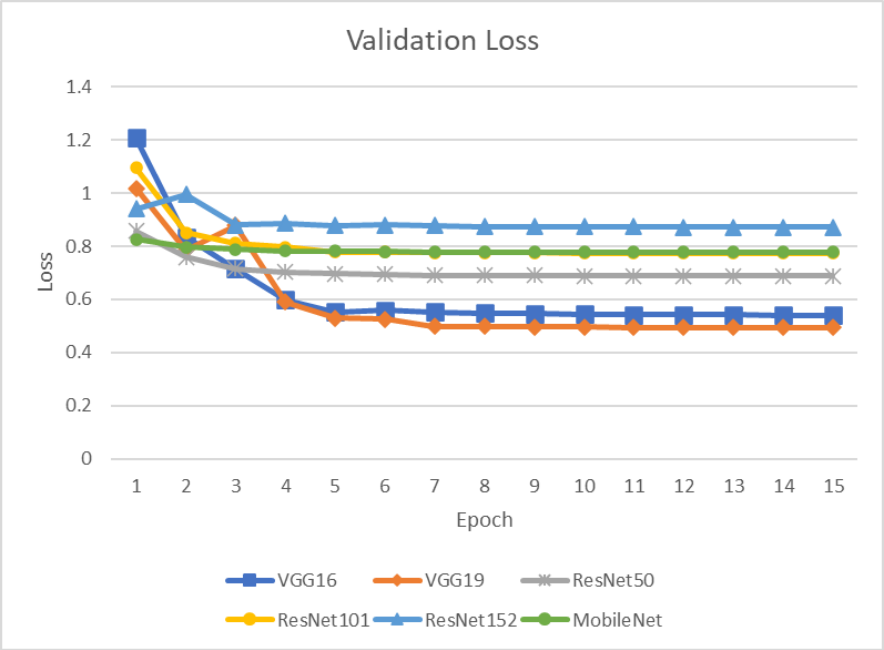

Figure 9. Validation Loss Graph on Each CNN Architecture

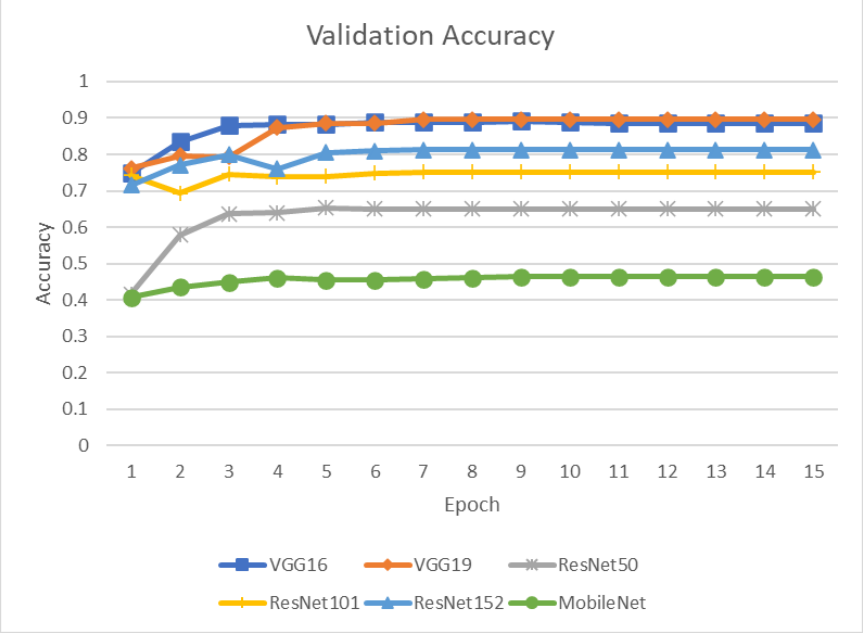

Figure 10. Validation Accuracy Graph on Each CNN Architecture

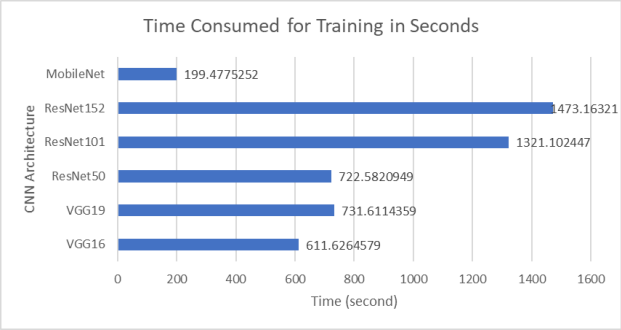

Figure 11. Training Time to Run 15 Epochs on Each CNN Architecture

The result in VGGNet (VGG16 and VGG19) has the best training accuracy (81% on VGG16, 80% on VGG19) and validation accuracy above 80% (88% on VGG16 at epoch 15, 89% on VGG19 at epoch 15) and also having considerable shorter training time compared to some model except MobileNet, but MobileNet much lower Accuracy and higher Loss compared to VGGNet. However, those VGGNet model has low validation loss but higher training loss (but also high training accuracy) compared to the other model. The normal condition is that if the model has high accuracy, then the loss must be lower. The loss tracks the prediction's inverse confidence. A high Loss and high Accuracy score indicate that the model is less confident in its predictions, even when making correct ones.

The next step is to run the model to predict the testing dataset. This dataset is hidden from the training process, containing 22 images, split into two classes. Class 0, called 'survive' containing 17 images, and class 1 called 'not survived' containing 5 images, the output is a classification report. The classification report displayed in Table 3, showing precision Equation 4, recall Equation 5, f1 value Equation 6, support (the number of images that belong to a class or whole dataset), Accuracy Equation 7, and this research also displayed classification time to measure performance. If we need to find a balance between Precision and Recall and there is an uneven class distribution, the F1 score might be a better measure to use [27].

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{4}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{5}$$

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall} \tag{6}$$

$$\text{Accuracy} = \frac{F1_{\text{class 1}} + F1_{\text{class 2}} + \cdots + F1_{\text{classN}}}{Number\ of\ Classes} \tag{7}$$

Because of the unbalanced dataset problem, we are also displayed macro average and weighted average. Macro average means calculates precision, recall, *f1* by class but not including weights for aggregation. Weighted average means calculate precision, recall, *f1* for each class, then adding weights value depending on the class [28]. An example of the macro calculation, the f1 macro calculation displayed on Equation 8. The example of the weighted calculation, the f1 weighted calculation, could be viewed at Equation 9.

$$\text{Macro AVG} = F1_{\text{class 1}} + F1_{\text{class 2}} + \cdots + F1_{\text{classN}} \tag{8}$$

$$\text{Weighted AVG} = F1_{\text{class 1}} * W_1 + F1_{\text{class 2}} * W_2 + \cdots + F1_{\text{classN}} * W_N \tag{9}$$

Table 3. The Result of Classification on Testing Dataset

Model Architecture		Precision	Recall	F1	Support	Time (s)
VGG16	0 ('Survive')	0.90	0.53	0.67	17	17.52
	1 ('Not Survived')	0.33	0.80	0.47	5	
	Accuracy			0.59	22	
	Macro AVG	0.62	0.66	0.57	22	
	Weighted AVG	0.77	0.59	0.62	22	
VGG19	0 ('Survive')	1.00	0.76	0.87	17	17.71
	1 ('Not Survived')	0.56	1.00	0.71	5	
	Accuracy			0.82	22	
	Macro AVG	0.78	0.88	0.79	22	
	Weighted AVG	0.90	0.82	0.83	22	
ResNet50	0 ('Survive')	0.93	0.76	0.84	17	8.49
	1 ('Not Survived')	0.50	0.80	0.62	5	
	Accuracy			0.77	22	
	Macro AVG	0.71	0.78	0.73	22	
	Weighted AVG	0.83	0.77	0.79	22	
ResNet101	0 ('Survive')	0.83	0.88	0.86	17	11.2
	1 ('Not Survived')	0.50	0.40	0.44	5	
	Accuracy			0.77	22	

	Macro AVG	0.67	0.64	0.65	22	
	Weighted AVG	0.76	0.77	0.76	22	
ResNet152	0 (‘Survive’)	0.80	0.71	0.75	17	13.85
	1 (‘Not Survived’)	0.29	0.40	0.33	5	
	Accuracy			0.64	22	
	Macro AVG	0.54	0.55	0.54	22	
	Weighted AVG	0.68	0.64	0.66	22	
MobileNet	0 (‘Survive’)	1.00	0.41	0.58	17	5.34
	1 (‘Not Survived’)	0.33	1.00	0.50	5	
	Accuracy			0.55	22	
	Macro AVG	0.67	0.71	0.54	22	
	Weighted AVG	0.85	0.55	0.56	22	

From the result displayed in Table 3, the best *f1* score belongs to VGG19 architecture with the highest *f1* accuracy of 82%, which means the best model is looking for accuracy on prediction, but this costs a longer prediction time than other CNN architectures. MobileNet is the fastest, but it cost much worse on prediction accuracy, only 55%. The ResNet50 model has balanced prediction accuracy/time. It got 77%, not too bad accuracy, and 8.49 seconds of prediction time, 9 seconds less than VGG19.

4. Conclusion

This research is to classify the condition of COVID-19 patients that develop pneumonia symptoms based on their survivability using a Convolutional Neural Network. The problem encountered in this research is that the dataset is too few and unbalance. Image augmentation is used to multiply the original dataset containing 94 training images to 1429 images divided into 346 and 1083 images each. Class Weight is applied in this research to address the class imbalance problem instead of making equal images on both classes using augmentation. The classification task used CNN with 6 different architectures to find the best performance between them, and the result is VGG19 has the best overall accuracy. Training has 80% accuracy, 89% accuracy invalidation, and 82% *f1* score accuracy on classifying the testing dataset. VGG19 has relatively less time-consuming training steps, but in the classification, using the testing dataset, it has the most time-consuming compared to all the other architecture. ResNet50 has balanced prediction accuracy and time, got 77% on classifying testing dataset, and is 9 seconds less time-consuming than VGG19. MobileNet, however, has the fastest training and testing time, but the accuracy is unreliable.

References

[1] W. Zhang, “Imaging changes of severe COVID-19 pneumonia in advanced stage,” *Intensive Care Med.*, vol. 46, no. 5, pp. 841–843, 2020. https://doi.org/10.1007/s00134-020-05990-y

[2] H. A. Sidharta, F. Filipi, R. A. Jaskandi, and F. Nugroho, “Design and Implementation LETS (Low Power Cluster Server) for Sustaining UMKM during Pandemic,” *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 4, pp. 77–82, 2021. https://doi.org/10.22219/kinetik.v6i1.1162

[3] Y. Xu *et al.*, “Characteristics of pediatric SARS-CoV-2 infection and potential evidence for persistent fecal viral shedding,” *Nat. Med.*, vol. 26, no. 4, pp. 502–505, 2020. https://doi.org/10.1038/s41591-020-0817-4

[4] C. Huang *et al.*, “Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China,” *Lancet*, vol. 395, no. 10223, pp. 497–506, 2020. https://doi.org/10.1016/S0140-6736(20)30183-5

[5] H. Sharma, J. S. Jain, P. Bansal, and S. Gupta, “Feature extraction and classification of chest X-ray images using CNN to detect pneumonia,” *Proc. Conflu. 2020 - 10th Int. Conf. Cloud Comput. Data Sci. Eng.*, pp. 227–231, 2020. https://doi.org/10.1109/Confluence47617.2020.9057809

[6] M. Heidari, S. Mirniaharikandehei, A. Z. Khuzani, G. Danala, Y. Qiu, and B. Zheng, “Improving the performance of CNN to predict the likelihood of COVID-19 using chest X-ray images with preprocessing algorithms,” *Int. J. Med. Inform.*, vol. 144, no. September, p. 104284, 2020. https://doi.org/10.1016/j.ijmedinf.2020.104284

[7] M. M. Eid and Y. H. Elawady, “Efficient Pneumonia Detection for Chest Radiography Using ResNet-Based SVM,” *Eur. J. Electr. Eng. Comput. Sci.*, vol. 5, no. 1, pp. 1–8, 2021. https://doi.org/10.24018/ejece.2021.5.1.268

[8] P. Quah, A. Li, J. Phua, and J. Phua, “Mortality rates of patients with COVID-19 in the intensive care unit: A systematic review of the emerging literature,” *Crit. Care*, vol. 24, no. 1, pp. 1–4, 2020. https://doi.org/10.1186/s13054-020-03006-1

[9] Bachir, “COVID-19 chest xray,” COVID-19 chest xray, 2020.

[10] K. Pasupa and W. Sunhem, “A comparison between shallow and deep architecture classifiers on small dataset,” *Proc. 2016 8th Int. Conf. Inf. Technol. Electr. Eng. Empower. Technol. Better Futur. ICITEE 2016*, 2017. https://doi.org/10.1109/ICITEED.2016.7863293

[11] Kaladharanalytics, “Covid-19-X-ray-Image-Augmentation-,” 2020.

[12] T. Devries and G. W. Taylor, “Dataset Augmentation In Feature Space,” pp. 1–12, 2017.

[13] J. Terstriep, “Keras Spatial,” 2019. https://doi.org/10.1145/3356471.3365240

[14] B. Usage, “The NumPy Array: A Structure for Efficient Numerical Computation,” pp. 22–30. https://doi.org/10.1109/MCSE.2011.37

[15] A. Nurhopipah and N. A. Larasati, “CNN hyperparameter optimization using random grid coarse-to-fine search for face classification,” vol. 4, 2021. https://doi.org/10.22219/kinetik.v6i1.1185

[16] G. Licea, “Towards supporting Software Engineering using Deep Learning : A case of Software Requirements Classification,” 2017. https://doi.org/10.1109/CONISOFT.2017.00021

[17] R. T. Puteri and F. Utaminingrum, "Micro-sleep detection using combination of haar cascade and convolutional neural network," *ACM Int. Conf. Proceeding Ser.*, pp. 130–135, 2020. https://doi.org/10.1145/3427423.3427433
[18] H. Bi, F. Xu, Z. Wei, Y. Xue, and Z. Xu, "An Active Deep Learning Approach for Minimally Supervised PolSAR Image Classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 9378–9395, 2019. https://doi.org/10.1109/TGRS.2019.2926434
[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., pp. 1–14, 2015.
[20] M. Yamazaki et al., "Yet Another Accelerated SGD: ResNet-50 Training on ImageNet in 74.7 seconds," arXiv, 2019.
[21] A. G. Howard and W. Wang, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2012.
[22] X. Qi and T. Wang, "Comparison of Support Vector Machine and Softmax Classifiers in Computer Vision," pp. 151–155, 2017. https://doi.org/10.1109/ICMCCE.2017.49
[23] X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou, "On the Class Imbalance Problem *," pp. 192–201, 2008. https://doi.org/10.1109/ICNC.2008.871
[24] M. I. N. Zhu *et al.*, "Class Weights Random Forest Algorithm for Processing Class Imbalanced Medical Data," vol. 3536, no. c, 2018. https://doi.org/10.1109/ACCESS.2018.2789428
[25] "Estimate class weights for unbalanced datasets."
[26] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. van den Broeck, "A semantic loss function for deep learning with symbolic knowledge," arXiv, 2017.
[27] A. Yulianto, P. Sukarno, and N. A. Suwastika, "Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset," *J. Phys. Conf. Ser.*, vol. 1192, no. 1, 2019. https://doi.org/10.1088/1742-6596/1192/1/012018
[28] D. Morales, E. Talavera, and B. Remeseiro, "Playing to distraction: towards a robust training of CNN classifiers through visual explanation techniques," 2020.