Int. J. Advance Soft Compu. Appl, Vol. 10, No. 2, July 2018
ISSN 2074-8523

GPU Based Image Classification using Convolutional Neural Network Chicken Dishes Classification

Hafizhan Aliady, Dina Tri Utari

Department of Statistics, Universitas Islam Indonesia
e-mail: 14611225@students.uii.ac.id
Department of Statistics, Universitas Islam Indonesia
e-mail: dina.t.utari@uii.ac.id

Abstract

The internet makes human life easier, one of them is in ordering food. By only using smartphones and opening food delivery features, in the application, consumers can order food from restaurants that collaborate with the e-commerce. The case study that we did was classifying the types of chicken dishes namely "ayam geprek", "ayam goreng", and "ayam bakar" using CNN method. CNN method is a popular method for image classification. This research is doing image classification using CNN method based on GPU. From several models built, obtained the best model with an accuracy of 99% and training speed about 233 seconds. While for comparison of CNN method based on CPU and GPU obtained the conclusion that the most rapid training process using GPU.

Keywords: *classification, chicken dishes, image, CNN, GPU.*

1 Introduction

In this digital era, the internet has a huge influence on human life. Along with the development of the era, now humans do not need to bother in searching for information so that people are getting rich in information. With the internet, humans will be greatly helped in terms of ease and speed of delivery, delivery, and acceptance of information. From this ease, many companies can market their products online or known as e-commerce. Many e-commerce in Indonesia is expanding its business in other fields. One of the service features developed is food delivery such as food delivery in a restaurant. Only by using smartphones and opening food delivery features within the application, consumers can order food from restaurants that collaborate with the e-commerce.

Food delivery business is not only developed in Indonesia. According to a study from McKinsey, based on a six-month study covering 16 countries around the world, provides insight into the rapidly changing state of today's market. The food delivery business is changing rapidly as new online platforms are racing for markets and customers across America, Asia, Europe and the Middle East. Although this type of business is new, food delivery is attractive because it's high valuation, even worth more than $ 1 billion.

The case that just happened in Indonesia is there was one customer of food delivery which complains because the food delivered is fallen apart, but what she ordered is *ayam geprek*. *Ayam geprek* is a fried chicken which crushed with *sambal*, so this customer was disappointed to the food delivery service. This is the customer's ignorance of the form of food ordered and the mismatch between the food image in the app and the real one.

Based on the above background, the authors are interested to make a classification of chicken dishes using the Convolutional Neural Network (CNN). Where CNN is a classification method used for digital image classification.

Table 1: Training Time CPU versus GPU

Batch Size	**Training Time CPU**	**Training Time GPU**	**GPU Speed Up**
64 images	64 s	7.5 s	8.5x
128 images	124 s	14.5 s	8.5x
256 images	257 s	28.5 s	9.0x

From research conducted by NVIDIA Deep Learning Institute about GPU acceleration on image classification using CNN [9], obtained image classification results done on GPU have training time faster than training time on CPU. Therefore, this study raised the computational theme using GPU to find the best CNN model in doing chicken dishes classification. Determination of the best CNN model can be seen from the results of high accuracy and the fastest training time.

The organization of this paper is as follows; Section 2 outlines the previous related works on CNN. Section 3 presents the problem formulation states what this research would do and methods of the research are discussed in Section 4. Finally, the experimental results and conclusions are presented in Section 5 and 6 respectively.

2 Related Work

During the last years, several works have been investigated to improve the performance of neural networks on the GPU. In 2006, Chellapilla et. al [3] have released the CNN to the GPU. They presented three independent approaches namely CPU no BLAS, CPU with BLAS, and GPU for high-performance implementation of CNN for document processing. Experimental results on real-world recognition problems show that dramatic speedups of up 3.00X on the CPU and 4.11X on the GPU.

Lahabar et al. [7] have made one of the first neural networks for the new unified shader architecture using CUDA. They implement a GPU on two applications intensive data pattern recognition operations: Parzen Window and ANN. The algorithms exploit high computing power of the GPU and provide the fast performance of the algorithms. Implementation of GPU on Parzen Window can compute the conditional or posterior probability in about 14 microseconds. Similarly, the ANN GPU can perform one-period batch training with 56 thousand samples in about 200 milliseconds.

In 2012, Krizhevsky et al. [6] have made the ImageNet classification with deep convolutional networks with convolution operation using GPU application. A deep convolutional neural network has been trained, dividing the object into 1000 different classes from 1.2 million images with high resolution and a very effective speed. Obtained a very low classification error rate (0.15) and the success of the classification of the convolutional neural network were observed.

3 Problem Formulations

This research is to classify chicken dishes based on three classification class: *ayam geprek*, *ayam goreng*, and *ayam bakar*. Since the purpose of this case study is to obtain a CNN model with a high degree of accuracy, the problem is how the best CNN model architecture should be built. In addition, we also want to get the model with the fastest training time based on the GPU being used.

4 Method

4.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is a neural network is used specifically to process the data structured grid, one of them in the form of image two dimensional. The convolution process is a linear algebra operation that multiplies the matrix of the filter on the image to be processed. The process is called a layer of convolution and is one of many types of layers that can own in one network.

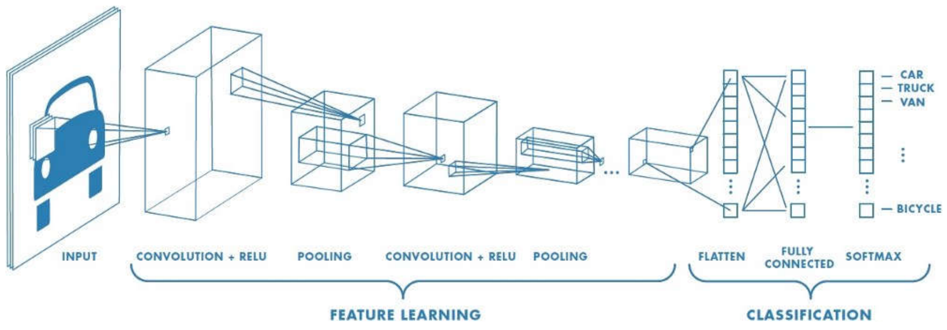

Fig. 1. CNN Architecture[1]

Convolution layer is the main layer and most important to use. Another type of layer that can be used is pooling layer. The pooling layer is used to extract the average value or maximum value of the pixel portion of the image. Each input layer entered has different volumes and is represented by depth, height, and width. The amount to be obtained may vary depending on the filtration result of the previous layer and the number of filters used.

4.1.1 Operation of Convolution

The convolution operation is an operation on two functions of a real-valued argument [5]. This operation applies the output function as feature map of the input image. These inputs and outputs can be seen as two real-valued arguments. Formally a convolution operation can be written with the following formula,

$$s(t) = (x * w)(t) \tag{1}$$

The function $s(t)$ gives a single output of feature map; the first argument is the input which is the x and the second argument w as the kernel or filter. If we see the input as a two-dimensional image, then we can assume t as pixel and replace it with i and j. Therefore, operations for convolution to inputs with more than one dimension can be written as follows,

$$S(i,j) = (K * I)(i,j) = \sum \sum I(i-m, j-n)K(m,n) \tag{2}$$

The equation (2) is the basic calculation in the convolution operation where i and j are the pixels of the image. The calculations are commutative and appear K as the kernel, I as input and the kernel can be reversed relative to the input. Alternatively, the convolution operation can be seen as the matrix multiplication between the input image and the kernel where the output can be computed with the dot product [10].

[1] https://www.mathworks.com

4.1.2 Convolutional Layer

The convolutional layer consists of neurons arranged in such a way that it forms a filter of length and height (pixels). For example, the first layer on the feature extraction layer is usually a 5x5x3 convolution layer, 5 pixels length, 5 pixels high and 3 or 3 pieces thick according to the channel of the image. All three of these filters will be shifted to all parts of the image. Each shift will be performed dot operation between the input and value of the filter to produce an output or commonly referred to as the activation map or feature map.

Stride is a parameter that determines how many filter shifts. If the value of stride is 1, then the convolutional filter will shift as much as 1 pixel horizontally and then vertically. In the illustration in Fig. 2, the stride used is 2. The smaller the stride it will be the more detailed information we get from an input but require more computation when compared with the large stride. But keep in mind that by using a small stride we will not always get a good performance.

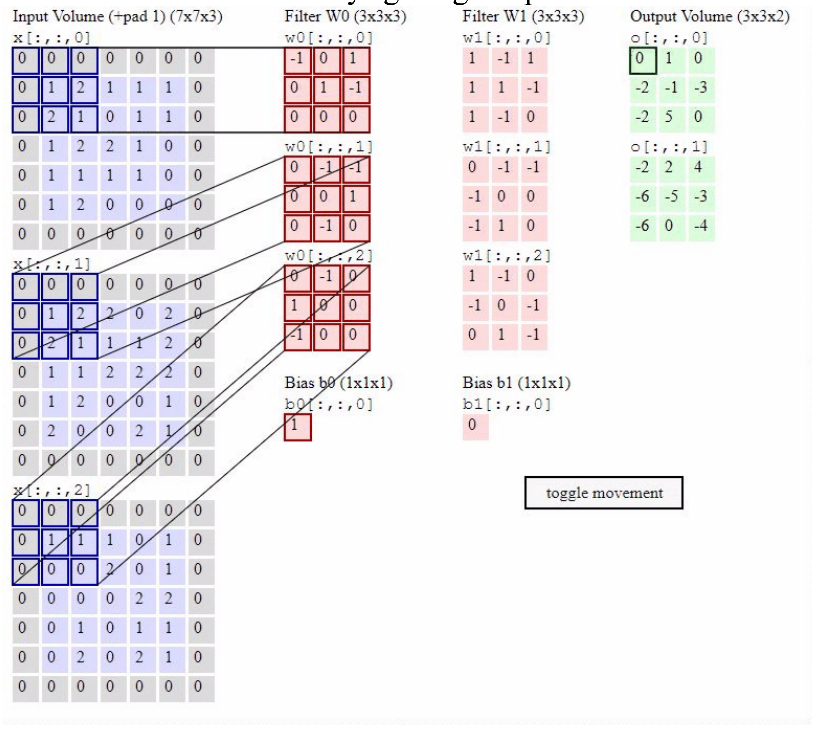

Fig. 2. Feature Map[2]

Padding or Zero Padding is a parameter that determines the number of pixels (containing the value 0) to be added on each side of the input. It is used in order to manipulate the output dimensions of the convolutional layer (feature map).

[2] http://cs231n.github.io/convolutional-networks/

The purpose of padding usage is the dimension of the output of the convolutional layer is always smaller than its input (except the use of a 1x1 filter with stride 1). This output will be reused as input from the convolutional layer so that more information is wasted. Using padding, we can adjust the output dimension to remain the same as the input dimension or at least not drastically reduced. So we can use convolutional layers so that more features are successfully extracted. Padding is also able to improve the performance of the model because the convolutional filter will focus on the actual information that is between the zero padding. In Fig. 2, the dimension of the actual input is 5x5, if done convolution with 3x3 filter and stride of 2, it will get feature map with size 2x2. But if we add zero padding as much as 1, then the resulting feature map is 3x3 (more information is generated). To calculate the dimension of the feature map we can use the formula as below,

$$output = \frac{W - N + 2P}{S} + 1 \tag{3}$$

where W is length/high of input, N is length/high of the filter, P is padding, and S is stride.

Pooling layers usually reside after convolutional layer. In principle, the pooling layer consists of a filter with a certain size and stride that will shift across the entire feature map area. Pooling commonly used is max pooling and average pooling. For example, if we use max pooling 2x2 with stride 2, then at each filter shift, the maximum value in the 2x2 pixel area will be selected, while the average pooling will select the average value.

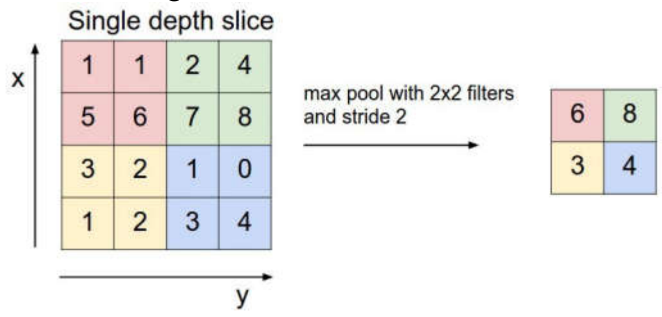

Fig. 3. Max Pooling[3]

The purpose of using pooling layer is to reduce the dimension of the feature map (down sampling), thus accelerating the computation because the parameters to be updated are less and overfitting.

[3] http://cs231n.github.io/convolutional-networks

4.1.3 Fully-Connected Layer

The featured map generated from the feature extraction layer is still a multidimensional array, so it must either flatten or reshape the feature map into a vector to be used as input from the fully-connected layer.

The Fully-Connected layer is a layer in which all the activation neurons of the previous layer are all connected with neurons in the next layer just like regular neural networks. Each activation of the previous layer needs to be converted into one-dimensional data before it can be connected to all the neurons in the layer.

The Fully-Connected layer is usually used in the multi perceptron layer method and aims to process the data so that it can be classified. The difference between the fully-connected layer and convolution layers is that neurons in the convolution layer connect only to specific regions of the input, while the Fully-Connected layer has an overall connected neuron. However, both layers still operate dot products, so the function is not so different [4].

4.2 Compute Unified Device Architecture (CUDA)

GPU (Graphic Processing Unit) [9] is a special processor for 3D graphics parts of the microprocessor. GPU popularized by NVIDIA, NVIDIA also developed a technology called CUDA (Compute Unified Device Architecture). CUDA (Compute Unified Device Architecture) is hardware and software architecture to manage computation in parallel on GPU hardware. Each CUDA-enabled GPU device can act as a parallel data computing device in bulk with a large amount of memory. This allows for much greater performance compared to traditional CPU. This performance improvement is caused by different GPU hardware designs with CPU. Where multicore CPUs provide large cache and perform full x86 instructions on each core, while smaller GPU cores are intended for throughput floating point.

This CUDA architecture allows software developers to create programs running on NVIDIA-made GPUs with syntax similar to the familiar C syntax. The CUDA architecture consists of three basic parts that help system developers to take advantage of the efficient computing capabilities of the graphics card. CUDA divides the device into grids, blocks, and threads in a hierarchical structure as in the Fig. 4. Because there are a number of threads in a block and the number of blocks in a grid and a number of grids in a single GPU, so the parallel process is achieved using a very large hierarchical architecture.

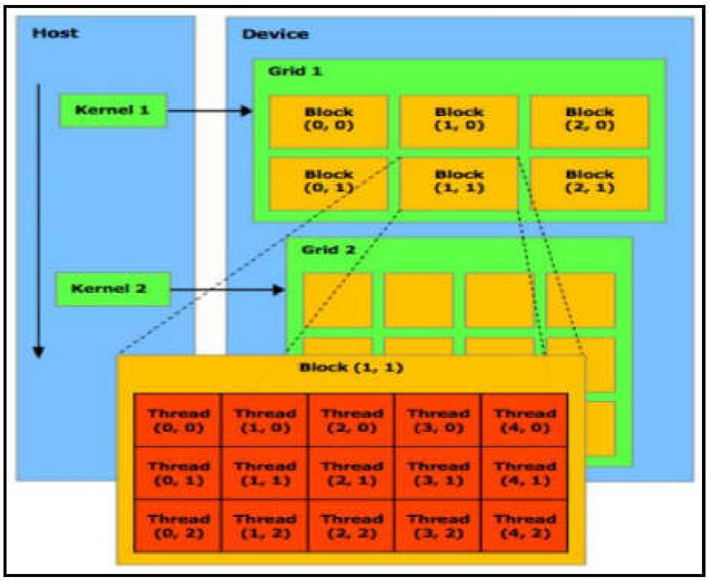

Fig. 4. CUDA Architecture[4]

CUDA is commonly used for graphical programming such as image and video processing digitally like image segmentation, image quality changes, pattern recognition can be applied to various needs in the real world [1].

The structure of the convolutional neural networks allows parallel programming as described in section 4.1. Because of this feature, applications using CNN algorithm can be done by parallel calculation in CUDA programming in computers with NVIDIA supported GPU.

Some of the methods using the GPU platform are [2, 8, and 11]. Although these studies are different in terms of methods and subjects, they are similar in terms of using GPU to develop the methods they developed using the CNN algorithm.

5 Experimental Results and Discussions

5.1 Dataset

Data that used in this case is taken from google images, using batch image downloader. We have 3 categories in this case, which is *ayam geprek, ayam goreng,* and *ayam bakar*. So we download about 120 images per category where 300 images for training and 60 images to validating the models.

[4] http://www.nvdia.com/object/cuda_home

Fig. 5. Training Dataset

5.2 Models

In this research, we used 3 models of CNN algorithm, each of their parameters is different so we can compare training time and validation accuracy to compare between models. Then we find the best models from there.

Table 2: Model Comparison

Layer	Model 1			Model 2			Model 3		
	Parameters	Size	Activation	Parameters	Size	Activation	Parameters	Size	Activation
Convolution	64 Filter	4x4	ReLu	128 Filter	3x3	ReLu	64 Filter	3x3	ReLu
Convolution	32 Filter	3x3	ReLu	128 Filter	3x3	ReLu	64 Filter	3x3	ReLu
Pooling	Average Pooling	4x4	ReLu	Max Pooling	2x2	ReLu	Max Pooling	2x2	ReLu
Convolution	32 Filter	3x3	ReLu	64 Filter	3x3	ReLu	128 Filter	3x3	ReLu
Convolution	32 Filter	3x3	ReLu	64 Filter	3x3	ReLu	128 Filter	3x3	ReLu
Convolution	32 Filter	3x3	ReLu	64 Filter	3x3	ReLu	128 Filter	3x3	ReLu
Pooling	Average Pooling	2x2	ReLu	Max Pooling	2x2	ReLu	Max Pooling	2x2	ReLu
Flatten Layer									
Fully Conected	256 Neuron		ReLu	256 Neuron		ReLu	512 Neuron		ReLu
Fully Conected	256 Neuron		ReLu	256 Neuron		ReLu	512 Neuron		ReLu
Fully Conected	128 Neuron		ReLu	128 Neuron		ReLu	256 Neuron		ReLu
Fully Conected	128 Neuron		ReLu	128 Neuron		ReLu	256 Neuron		ReLu
Fully Conected	-	-	-	-	-	-	128 Neuron		ReLu
Fully Conected	128 Neuron		TanH	128 Neuron		TanH	64 Neuron		TanH
Output Layer	3 Neuron		Softmax	3 Neuron		Softmax	3 Neuron		Softmax

On the first model, we use 5 convolution layer and 2 pooling layer which is using average pooling and the size of the pool is 4x4 and 2x2. Then we use 5 fully connected layers of the neural network and a single output layer with 3 nodes for each category.

Then on the second models, we use 5 convolution layer, but a number of the filter is different from the first model, the number of second models is decreasing from a high number of kernel filter to low number. Then the pooling layer that we use in this second model is max pooling with a dimension of the pool is 2x2. Then we use 5 fully connected layers of the neural network and a single output layer with 3 nodes for each category.

And then for the third models, we use 5 convolution layer too, but the number of filter kernel is increasing between layer. Then the pooling layer that we use in this third model is max pooling with a dimension of the pool is 2x2. Then we use 6 fully connected layers of the neural network and a single output layer with 3 nodes for each category.

5.3 Result

In this training process, we using 150 epoch, with 30 batch size and RMSPROP (Root Mean Square Propagation) for gradient descent algorithm with default parameters. So the hardware that we use in this training is GPU NVIDIA GTX 1060 3Gb GDDR5. The language that we use is R programing language with Rstudio and the library is Keras with Tensorflow backend. So the training process we can see in figure 6 below.

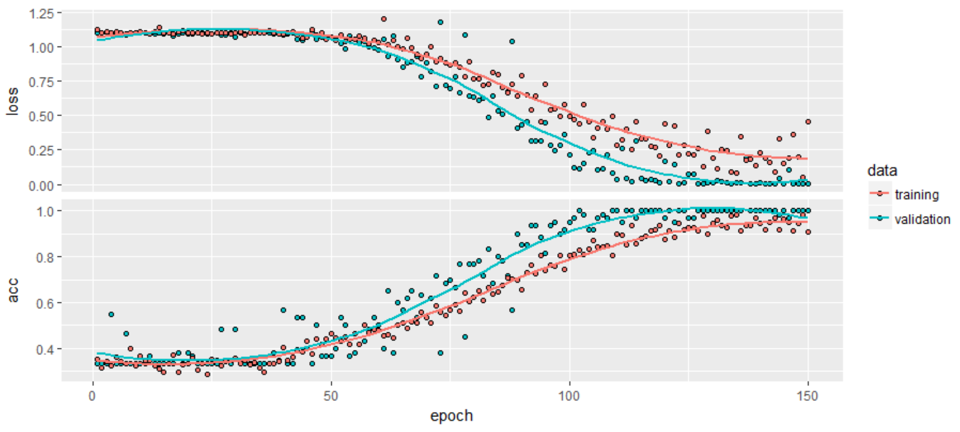

Fig. 6. Training Dataset for First Model

On the first model, we can see on the figure above. The model can predict better and in training process, the first model reaches the highest accuracy in 93%. From the graph, we can conclude the first models need more epoch to get the best model. Then this training progress consumes about 1.9 minutes or around 115 sec. Then for the second models, we can see on the figure below.

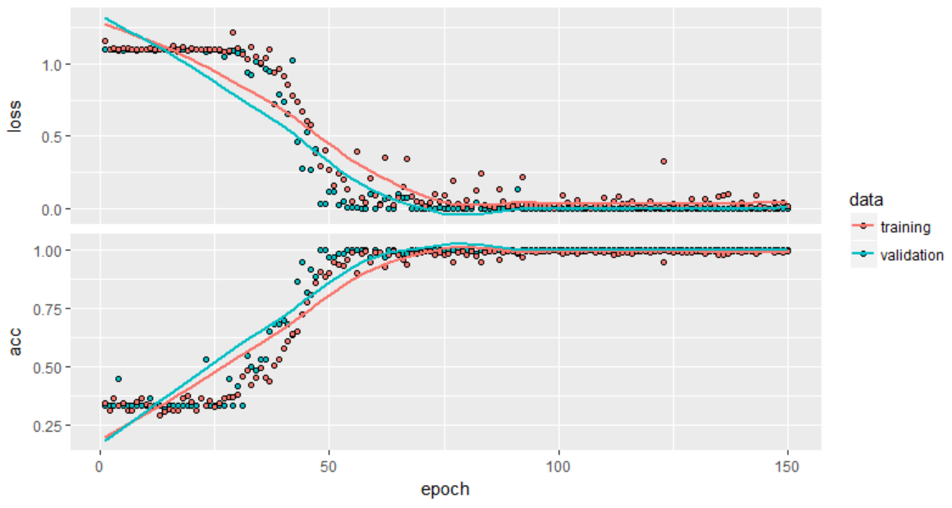

Fig. 7. Training Dataset for the Second Model

On the figure above we can see on training process of second models. The models can reach the highest accuracy at around 50 epoch. And the process of training this model is about 4.41 minutes or around 265 seconds. Although the training time is about 5 minutes to reach 150 epoch, the models reach optimum accuracy at around 50 epoch. So we can say the second models are better than the first model. Then for the third models, we can see on the figure below.

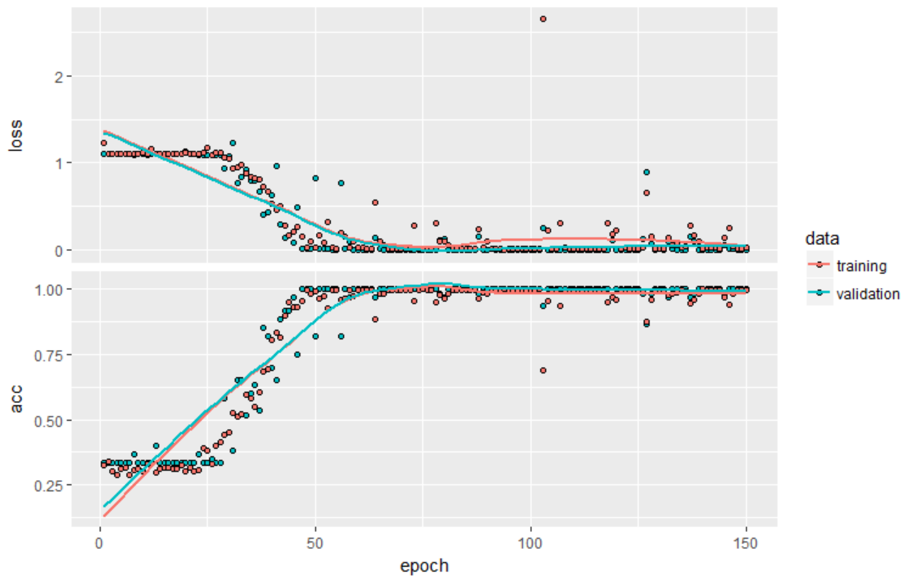

Fig. 8. Training Dataset for Third Model

From the figure above we can see the training progress of the third model. This model can reach an optimum point at around 50 epoch, which is same as the second model, but the training process takes around 3.89 mins or 233 sec. From this, we can say the third model is better than the second model. So we can say from the 3 comparisons above, the third model is the best model because the third model has high accuracy and faster than the second models.

Table 3: Accuracy and Processing Time

Model	Accuracy (%)	Process Time(s)
1ST Model	93	115
2ND Model	99	265
3RD Model	99	233

From the Table 3 above we can see the accuracy degree and processing time from each model, and then the best model is used to do the training model.

5.4 CPU versus GPU

After we get the best model from comparing 3 models above, then we compare between training and testing process on both devices between CPU and GPU. We compare the time that uses for each device to complete the training and testing process.

Table 4: CPU versus GPU

	CPU	GPU
Training	354	13
Testing	1.9	0.09

At the training process, we decide to reduce the number of epoch because training on CPU it takes too long if using 150 epochs, then we reduce the epoch to 10. We can see in Table 4 above that training process on CPU it takes 354 seconds to complete 10 epoch, and using GPU it takes only 13 seconds. This thing is caused by a number of processing core on CPU and GPU are quietly different. CPU that we use as processing device only has 4 cores. And GPU that use on this training process has 1152 CUDA Cores. Therefore, the training process on GPU so much faster than CPU. As well on the testing process, the processing time on 60 testing data takes only 0.09 seconds on GPU dan 1.9 seconds on CPU.

6 Conclusion

Based on the research that has been done, we can conclude that best model that can predict chicken dishes are the 3rd model. With 99% accuracy and the training speed is 233 seconds. Then the comparison between CPU and GPU on training and testing process on CPU it takes 354 seconds to complete 10 epoch and using GPU it takes only 13 seconds. This thing is caused by a number of processing core on CPU and GPU are quietly different. CPU that we used as processing device only has 4 cores. And GPU that use on this training process has 1152 CUDA Cores. Therefore, the training process on GPU so much faster than CPU. As well on the testing process, the processing time on 60 testing data takes only 0.09 seconds on GPU dan 1.9 seconds on CPU.

ACKNOWLEDGMENTS
The authors would like to thank the Department of Statistics, Islamic University of Indonesia (UII), for their support in data science research.

References

[1] Basuki, A. (2005). Pengolahan Citra Digital Menggunakan Visual Basic 6. Yogyakarta: Graha Ilmu.
[2] Cengil, E, & ÇÕnar, A. (2016). Robust Face Detection with CNN. In *Proceedings of the 4nd ICAT International Conference on Advanced Technology & Sciences (pp. 47-51).* ICAT.
[3] Chellapilla, K., Puri, S., & Simard, P. Y. (2006) High-Performance Convolutional Neural Networks for Document Processing. In *Proc. of the 10th Int. Workshop on Frontiers in Handwriting Recognition*.
[4] Danukusumo, K. P. (2017). Implementasi Deep Learning Menggunakan Convolutional Neural Network untuk Klasifikasi Citra Candi Berbasis GPU (Undergraduate mini-thesis, Universitas Atma Jaya).
[5] Goodfellow, et. al. (2016). Deep Learning. http://www.deeplearningbook.org/.
[6] Krizhevsky, I. Sutskever, & Hinton, G. E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (pp. 1907-1105).*
[7] Lahabar, S., Agrawal, P., & Narayanan, P. J. (2008). High-Performance Pattern Recognition on GPU. In *Proc. of the National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics* (*pp. 154–159).*
[8] Li, H., Lin, Z., Shen, X., Brandt, J., & Hua, G. (2015). A Convolutional Neural Network Cascade for Face Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5325-5334).*
[9] NVIDIA. (2009). NVIDIA CUDA-Programming Guide, http://www.nvidia.com/object/cuda home.html.
[10] Rismiyati. (2016). *Implementasi Convolution Neural Network Untuk Sortasi Mutu Salak Ekspor Berbasis Citra Digital* (Magister thesis, Universitas Gadjah Mada).
[11] Wang, L., Lee, C., Y., Tu, Z., & Lazebnik, S. (2015). Training Deeper Convolutional Networks with Deep Supervision. In *arXivpreprint arXiv:1505.02496.*