ISSN 2278-3091

Volume 9, No.3, May - June 2020

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse52932020.pdf

https://doi.org/10.30534/ijatcse/2020/52932020

Landmark Classification Service Using Convolutional Neural Network and Kubernetes

Indra Prasetya Aji[1], Gede Putra Kusuma[2]

[1]Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 11480, indra.aji@binus.ac.id

[2]Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 11480, inegara@binus.edu

ABSTRACT

The use of Deep Learning for the image classification process has developed rapidly, one of which is in a method called Convolutional Neural Network (CNN). One use of the CNN method is to classify images. This study uses the CNN method to classify image datasets that have landmark categories. The results show that each model has an accuracy level with the highest value of 95% and the lowest value of 92%. The method for this research to produce a service that can be used by other services and or can be implemented in other resources so that the classification function can be used or developed. We show that our deployment method has good results seen with ResNet50 Model using 1200 concurrent users which response rates were smaller by 4,92% using autoscaling pods with one minimum pod and 6 maximum pods that had an error value of 26,58% compared with the best result from the other method.

Key words : Object Classification, Deep Learning, Convolutional Neural Network, Docker Image, Kubernetes.

1. INTRODUCTION

Landmark is a visual symbol that identifies a building form based on a particular visual form that is strong because it has a characteristic and is not owned by other regions. Landmark classifications can also be categorized as image classifications. Image classification is one of the problems in Computer Vision and is often used to detect objects in an image. Classification of the image itself is arguably a very difficult job to do by a computer. One way to simplify computer performance in classification is to implement Deep Learning techniques using the Convolutional Neural Network (CNN) method.

Deep Learning is one of the sub-areas of the Machine Learning method where the algorithm used is based on artificial neural network structures. There are a lot of machine learning techniques that shown to be effective to classify images, but even among those method, neural network seems to have the most robust performance, recent survey conducted by [1] shows that Artificial Neural Network (ANN) have the best result. However just using a simple ANN proved to be not enough, as with different approaches and complexity of the images, this method lag behind with the other machine learning method such as kNN as shown by [2] and SVM method shown by [3]. In recent years Deep Learning has shown exceptional performance. This is largely influenced by stronger computational factors, large datasets, and techniques for training deeper networks [4]. Basically, the CNN method is a more effective artificial neural network architecture for image classification. CNN can make the image learning function more efficient to implement.

The main problem in image classification techniques is thelevel of accuracy of the processing results and the amount of computer resources. The level of accuracy is influenced by several factors such as image diversity, image size, andhardware used. Several methods of Deep Learning optimization have been mentioned by[5], but it still cannot be determined which optimization has the best level of accuracy and the most time efficient. A computer resource will greatly affect the level of performance of the implemented CNN method. The greater or higher the resource, the better CNN method performance.

The selection of this landmark classification method requires comparison with the CNN classification method that has been done in previous studies. Followed by determining the appropriate deployment method for implementing the selected classification method, then the performance of the image classification method using CNN can be measured. The performance measurement will determine the best CNN model which can then be used for other applications.

The deployed model is accessed using a web service. This web service API will be used to test the model at the performance measurement stage. Performance measurements will be carried out in stages on the number of concurrent users. The results of performance measurements will show the model with the best performance compared to the number of concurrent users.

2. RELATED WORKS

There are developments in systems that explore Computer Vision such as Face Recognition, Image Recognition, Malware Classification, and the introduction of certain patterns that have proven a new discovery that can facilitate work in many fields. Implementation of Deep Learning on these problems is very appropriate and effective.

The use of CNN with SGD optimization conducted by [6] is used to classify temple imagery. There are 6 classes with 100 epoch training that get 86.28% accuracy. In the study also mentioned when epoch 50, the machine has gotten optimal results seen from the testing accuracy and the time used to train the model where the training accuracy obtained is 98.99% and validation accuracy reaches 85.57% with the architecture used is AlexNet.

CNN with a standard structure by [7] classifies the landmarks and landmarks using Deep Local Feature (DELF) as a pipeline. DELF is used to sort or filter the dataset. The basis of the model they use is ResNet50. Datasets entered by their method will be filtered again to fewer that meet their eligibility categories.

Research by [8]classify landmarks by using the Network Architectures of Hybrid Auto-Encoder (HAE). Their architecture focuses on making the training algorithm slightly different from the structure of CNN in general. The level of accuracy generated by their architecture is slightly below CNN.The classification of landmarks, especially in India by[9] is done by making an architecture consisting of Graph-based Visual Saliency (GBVS) and combined with kNN and Random Forest Classification. The model they use is Inception ResNet V2. The architecture they make is tested on their own method called Average Ensemble has higher results when tested on the CNN method.

The use of DCNN architecture by[10] was carried out to improve accuracy in landmark classification. They patched the PlaNet model because the model has geolocation parameters that help in the prediction process. The level of accuracy differs depending on the number of patches made on the model.

Landmark classification conducted by [11] by determining the amount of DELF using the Attention mechanism, creating a simple and effective end-to-end similarity learning network for object retrieval by combining 2 subnetworks, Features Extraction to produce local regional features and Object Localization to predict the existence and location of objects. The dataset entered will also be reselected so that each class has the same image similarity pattern. The basis of the model they use is ResNet101.

Research conducted by [12] a classification of special landmarks in Singapore using CNN shows a remarkable result. The model used is NU-LiteNet. The model is an improved model from the SqueezeNet model. Their aim is to reduce the size of the training model file results to be able to be executed (classification) faster on mobile devices.

The dataset cleaning method is used by [13] to get quality images. The accuracy results in the studies mentioned can be seen in Table 1 showing a comparison of CNN methods in the research that has been done.

Table 1: Comparison Table

Author	*Dataset*	*Method*	*Accuration (%)*
[6]	Candi	CNN	86,28
[7]	Oxf5k; Oxf105k; Par6k; Par106k	CNN with DELF	90,0; 88,5; 95,7; 92,8
[8]	Holidays; Oxford5k; Paris6k	Network Architecture of Hybrid Auto-Encoder	89,32; 84,60; 91,90
[9]	Landmark in India	Average Ensemble	90,0
[10]	Paris; Oxford	DCNN	90,63; 76,03
[12]	Landmark in Singapore	CNN with NU-LiteNet	94,65
[11]	Oxf5k; Par5k; Oxf105k; Par106k; OxfShf; ParShf; Ins	CNN with DELF using Attention	91.9; 95.8; 90.4; 93.3; 91.3; 92,4; 78.2
[13]	Clean Google Landmark; Clean Oxford; Clean Paris	CNN	33,01; 66,95; 82,98

3. METHODOLOGY

There are five stages in this research. The first stage is to prepare the dataset used for model training and testing. The scond stage is training process. The structure of the model that has been formed is used to carry out the training stage. The next stage is to test the model, evaluate using web services and performance test. The training and testing process is carried out on a dataset with the same number of epochs and batches. The following is the detailed stage process:

3.1 Dataset Preparation

The evaluation plan that will be carried out is by evaluating the public dataset of Google Landmark Dataset of the model that has been made. It needs to be filtered on a public dataset because there is an imbalance in the number of images in each class. There are classes that have less than 10 images and there are more than 1000 images. In this study we took an average value of 150 from all classes and the number of classes used was 100 classes.

Each class in the dataset will be divided by a certain composition. There are 3 compositions in the dataset, namely data train, data validation, and data testing. Comparison of composition is 60 percent for data train, 20 percent for data validation, and 20 percent for data testing.

3.2 Training Process

The training process is carried out on the data train and data validation. The training results are in the form of a model that has a weight, which is then stored for evaluation and image prediction. The test conducted is a test of the model of training results. The model is tested on test data in which the output is a percentage of overall accuracy. The additional testing process for the landmark dataset used is to make predictions on the inserted image, which provides output in the form of the landmark name of the inserted image.

Before the model can be used for evaluation, the model needs to be trained using CNN method with several different model architecture such as MobileNet, MobileNet(V2), ResNet50, ResNet50V2, and VGG16. Each model will use certain optimizer technique in order to get the output class for classification, the optimizer that we will use are Adam, Adamax, Adagrad, Nadam, and RMSProp. Model training process developed using python and deep learning framework keras. The model training is processed using Google Collaboration, the specification of the server is using GPU model Tesla K80 and 12 GB RAM.

The step of making a model into a function, using all the libraries we successfully imported before. We can determine the base model, for example the base model chosen is the base model VGG16 application. The addition of the Global Average Pooling (GAP) layer and the dense layer is also carried out with the aim of determining how many classes we will be training to be indexed when preprocessing the image and increasing accuracy. Optimization variables are also needed so that when the model is computational, the array values in the image have values that make it easier for machines to process them. Optimization has a variable called the learning rate which the value of the learning rate will greatly assist the calculation process that regulates the magnitude of the learning rate.

3.3 Model Testing

In machine learning fields and in particular the problem of statistical classification, Confusion Matrix is a specific table that allows visualization of an algorithm's performance. Classification can be done by matching the models that have been made. Matching is done on test data in each class.

The results and predictions obtained can be grouped into several conditions, namely true positive and actual positive which means that the data is indeed in accordance with the classification. The true positive condition on the actual negative data means that the data is not included in the exact classification. True negative conditions on positive actual data mean that the data should not be in the classification in question. True negative conditions on negative actual data mean that the data cannot be classified in certain classifications.

Evaluation results of each method used will be measured using a confusion matrix table. The data in the confusion matrix is the basis for the accuracy of each data. Confusion matrix table has reports in the form of precision, recall, and f1-score for each class, as well as accuracy, macro average, and weighted average for the whole class.

3.4 Web Service Evaluation Plan

We used Flask to create a rest API as a link to the image classification results.. The previously trained model will be loaded by the Flask framework, then the structure as well as the results of parameters and weights of the model will be ready to be used for the classification process. The API endpoints that have been generated can be used to be implemented on the gateway API. Making of the API gateway is to use the Spring Boot framework.The function of the API gateway is as access to authorize it so that other applications can use the image classification that has been created.

The next evaluation is to conduct a performance test to measure the CNN model through the API web service that has been deployed to Kubernetes. This evaluation includes comparisons between different CNN models, with various performance measurement benchmarks such as concurrent users, latency and response time from the network.

Performance tests will be carried out using the JMeter application. The number of APIs is made according to the number of models to be tested. The value of the number of users that will make requests simultaneously or called concurrent users on the API is done in stages from 150 and a multiple of 150 up to a maximum value of 1200. Inputs received by the API are in the form of data base64 encoding of an image.

3.5 Performance Measurement

The performance test will test each model through the gateway API.Performance measurements are also carried out on the model through the API that has been made. The API was deployed to Kubernetes system on Google Cloud Platform, the server specification is 22 GB RAM and 6 CPU cores using Intel Skylake technology. Measurements were made using the JMeter application, each model will be distinguished through labels to be tested one by one. Each label has a different API url and from each label will produce a report with data including average, min, max, std. dev, error, throughput and Kb / Sec.

Average is the average time taken by all samples to run a certain label. Min is the shortest time taken by a sample for a particular label. Max is the longest time taken by a sample for a particular label. Std. Dev shows a series of cases that deviate from the average value of sample response time. The lower this value the more consistent the data flow from network. Std. A good dev should be less than equal to half the average time for a label. Error (%) is the percentage of failed requests per label. Throughput is the number of requests processed per unit of time by the server. This time is calculated from the beginning of the first sample to the end of the last sample. Bigger throughput is better. KB / Sec shows the amount of data downloaded from the server during the performance test.

4. RESULTS

To test Google Landmark Dataset, the method used in the study was carried out on 5 models and each model was carried out using 5 optimizations. The models produced with these conditions are 25 different models. The following is a table of training results with conditions of epoch 150 and batch size 300.

Table 2: Using Adamax Optimizer

Optimizer Adamax			
Model	***Time to Train (second)***	***Validation(%)***	***Test(%)***
MobileNet	1259,57	93,33	94,56
MobileNet(V2)	1321	92,11	93,22
Resnet50	1382,71	94,44	95,67
Resnet50V2	1431,03	92,5	92,72
VGG16	1333,5	89,06	90

Table 3: Using Adam Optimizer

Optimizer Adam			
Model	***Time to Train (second)***	***Validation(%)***	***Test(%)***
MobileNet	1327,49	94,44	95,56
MobileNet(V2)	1415,29	93,44	94,44
Resnet50	1751,82	92,39	94,11
Resnet50V2	1421,03	90,5	92,44
VGG16	1747,34	92,67	94,28

Table 4: Using Adagrad Optimizer

Optimizer Adagrad			
Model	***Time to Train (second)***	***Validation(%)***	***Test(%)***
MobileNet	1200,8	85,98	86,83
MobileNet(V2)	1250,3	81,83	82,94
Resnet50	1349,23	91,56	93,22
Resnet50V2	1404,59	83,72	85,72
VGG16	1324,06	85,94	86,33

Table 5: Using Nadam Optimizer

Optimizer Nadam			
Model	***Time to Train (second)***	***Validation(%)***	***Test(%)***
MobileNet	1218,21	93,94	95,33
MobileNet(V2)	1329,54	92,94	93,72
Resnet50	1401,21	95,22	95,28
Resnet50V2	1588,32	91,89	92
VGG16	1707,37	92,61	93,61

Table 6: Using RMSProp Optimizer

Optimizer RMSProp			
Model	***Time to Train (second)***	***Validation(%)***	***Test(%)***
MobileNet	1314,27	93,83	94,78
MobileNet(V2)	1417,43	92,94	93,39
Resnet50	1712,9	90,89	91,06
Resnet50V2	1645,53	91,78	92,44
VGG16	1762,29	70,06	72,39

Based on the accuracy comparison results table, the best accuracy value obtained tested on the test dataset is the ResNet50 model with Adamax optimization with an accuracy value of 95.67% as shown in Table 2. The best accuracy value obtained tested on the validation dataset is the ResNet50 model with Adam optimization with an accuracy value of 95.22% as shown in Table 3.

The worst accuracy value obtained that was tested against the test dataset and also validation is the VGG16 model with RMSprop optimization with the accuracy value shown in Table 6. The accuracy value in Adagrad and Nadam optimization can be seen in Table 4 and Table 5 which have various accuracy values.The cause of the low accuracy value of a model shown in the data above is due to the unstable accuracy of the data train compared to the data validation at each epoch. Other factors that cause are unstable and high loss in the data train compared to data validation at each epoch.

The graph of the difference in accuracy and loss values between the test data against the validation data that occurs in the model with the worst accuracy value of the validation data and the test data can be seen in Figure 1. The graph shown is very different from the graph from the model that has the best accuracy value for the test data in Figure 2 and for the data validation in Figure 3.

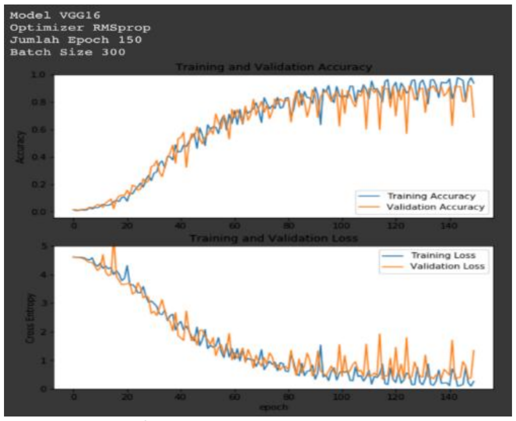

Figure 1: Graph for accuracy and loss (VGG16, RMSprop Optimizer)

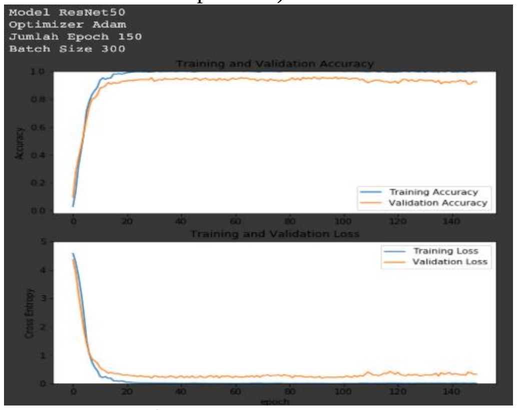

Figure 2: Graph for accuracy and loss (Resnet50, Adam Optimizer)

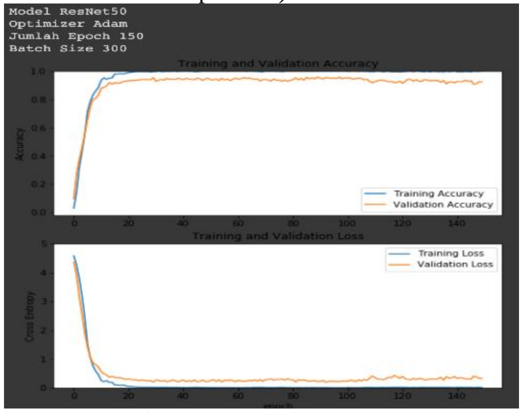

Figure 3: Graph for accuracy and loss (Resnet50, Adamax Optimizer)

Before the training results model can be used through the API for testing, each model needs a service for the classification process with base64 input which is then used as a docker image. Docker image is a file that consists of several instructions that are used to execute some code and make a package so that an application can run. In this case, the docker will run the dockerfile which contains instructions for running the python file where this file when executed will load the model and prepare the service for the classification process needs. The docker image also needs to provide a python environment along with the library needed for classification by giving instructions to the dockerfile, so that when deployed to the server you don't need to install the dependencies. After that, the docker image will be uploaded to Dockerhub which will be used for deployment to Kubernetes.

We used Kubernetes for deployment automation system provided by the Google Cloud Platform. Deployment in Kubernetes will use a docker image that has been created previously. When the deployment is ready, another thing to consider is the use of autoscaling since the tests will be accessed by concurrent users. Autoscaling is a method used in cloud computing, where the amount of computing resources on the server is set automatically based on the load request on a service, this can speed up resources in processing a request. Kubernetes has a pod autoscaling feature where the pod will automatically increase depending on how much deployment is accessed. In the first test, the python pod engine does not use autoscaling with a minimum pod of 2. In the second test, the python pod engine will use autoscaling with a minimum pod of 1 and a maximum of 6.

In performance testing for each model, the method used in the study is carried out on a number of different concurrent users. The performance measurements have 2 ways on how the pod is deployed with concurrent users of 1200,it can be seen in Table 7 and Table 8.

Table 7: Performance Test 1200 Concurrent Users (Autoscale 2 Pods)

	Model				
	VGG16	***MobileNet***	***MobileNet (V2)***	***ResNe t50***	***ResNe t50 -V2***
Average	31470	39572	16868	24049	23593
Min	813	673	684	751	745
Max	81287	87320	48487	54635	5246 6
Std. Dev.	20993.5	21286.23	8093.61	13229.92	12657 .65
Error %	37.83	34.17	31.92	34.25	31.50
Throughput	14.59108	13.56147	24.56097	21.92982	22.42 236
Received Kb/sec	18.88	13.52	28.8	26.96	26.32
Sent Kb/sec	42.76	42.06	78.94	68.02	72.46

Table 8: Performance Test 1200 Concurrent Users (Autoscale 6 Pods)

	Model				
	VGG16	***MobileNet***	***MobileNet (V2)***	***ResNe t50***	***ResNe t50 -V2***
Average	28145	36278	15897	24160	19000
Min	823	661	695	766	735
Max	88910	61246	49698	73136	4735 7
Std. Dev.	19543.36	18478.18	7702.7	16479.82	10656 .72
Error %	32.00	34.33	34.00	26.58	31.50
Throughput	13.41157	19.49033	23.79064	16.27119	25.00 208
Received Kb/sec	15.74	19.35	28.93	17.06	29.26
Sent Kb/sec	43.15	60.29	74.22	56.36	80.8

Representation of the results of the classification of the results of tests conducted on the implementation of the model using the Confusion Matrix. The model that is tested is the model that has the highest validation results on each type of model. The test results on each different model are represented with several values such as accuracy precision, recall, f1-score and support where each model has support 3041 which means the number of images tested. In Table 9 namely the VGG16 model has its accuracy with a value of 0.93 and precision, recall and f1-score which has more or less the same value. Likewise with other tables as in Table 10 and Table 11 with the MobileNet-V1 and MobileNet-V2 models. Both of these models produce scores that are almost the same where MobileNet-V1 is slightly better in terms of accuracy. The same thing also happened in Table 12 and Table 13, namely the ResNet-V1 and ResNet-V2 models where ResNet-V1 has a higher level of accuracy. Then in Figure 4there is a matrix confusion graph that illustrates the accuracy of testing in each class.

Table 9: Confusion Matrix Report Table (VGG16)

Model	**VGG16 – Google Landmark Dataset**			
Optimizer	**Adam**			
Accuracy Score:	**0.9312726076948372**			
	Precision	***Recall***	***f1-score***	***Support***
accuracy			0,93	3041
micro avg	0,93	0,93	0,93	3041
macro avg	0,94	0,93	0,93	3041
weighted avg	0,94	0,93	0,93	3041

Table 10: Confusion Matrix Report Table (MobileNet-V1)

Model	**MobileNet – Google Landmark Dataset**			
Optimizer	**Adam**			
Accuracy Score:	**0.9533048339362052**			
	Precision	***Recall***	***f1-score***	***Support***
accuracy			0,95	3041
micro avg	0,95	0,95	0,95	3041
macro avg	0,96	0,95	0,95	3041
weighted avg	0,96	0,95	0,95	3041

Table 11: Confusion Matrix Report Table (MobileNet-V2)

Model	MobileNetV2 – Google Landmark Dataset			
Optimizer	Adam			
Accuracy Score:	0.9385070700427491			
	Precision	***Recall***	***f1-score***	***Support***
accuracy			0,93	3041
micro avg	0,94	0,94	0,94	3041
macro avg	0,94	0,94	0,94	3041
weighted avg	0,94	0,94	0,94	3041

Table 12: Confusion Matrix Report Table (ResNet50-V1)

Model	**ResNet50 – Google Landmark Dataset**			
Optimizer	**Nadam**			
Accuracy Score:	**0.9519894771456757**			
	Precision	***Recall***	***f1-score***	***Support***
accuracy			0,95	3041
micro avg	0,95	0,95	0,95	3041
macro avg	0,96	0,95	0,95	3041
weighted avg	0,96	0,95	0,95	3041

Table 13: Confusion Matrix Report Table (ResNet50-V2)

Model	**ResNet50V2 – Google Landmark Dataset**			
Optimizer	**Adamax**			
Accuracy Score:	**0.9263400197303518**			
	Precision	***Recall***	***f1-score***	***Support***
accuracy			0,92	3041
micro avg	0,93	0,93	0,93	3041
macro avg	0,93	0,93	0,93	3041
weighted avg	0,93	0,93	0,93	3041

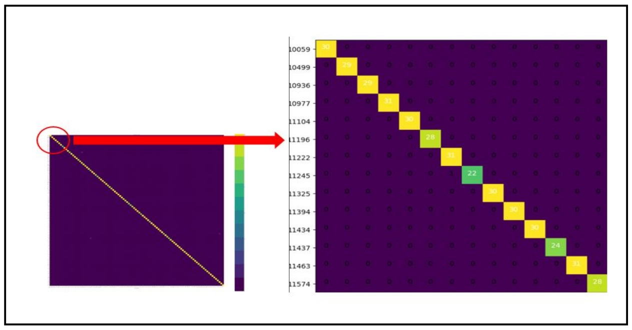
Figure 4: Confusion Matrix Graph (Google Landmark Dataset)

The difference in performance based on the type of deployment for the MobileNet model can be seen in Figure 5. The graph shows that the autoscale technique has a higher error rate than the number of static pods. The same conclusion also applies to the MobileNet-V2 model as shown in Figure 6.

Different results are actually shown in the ResNet50 model shown in Figure 7 Based on the graphs shown, the autoscaling technique in the pod greatly influences the number of errors that occur especially in the increasing value of concurrent users. The same effect also occurs in the ResNet50V2 model as well as the VGG16 model, which is shown in Figure 8 and Figure 9.

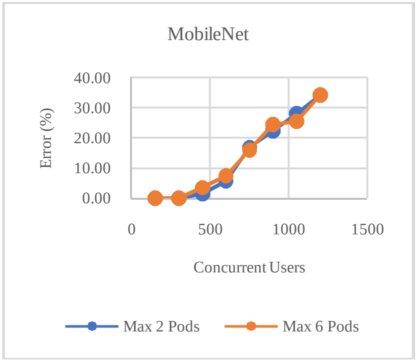

Figure 5: Performance Graph for MobileNet Model

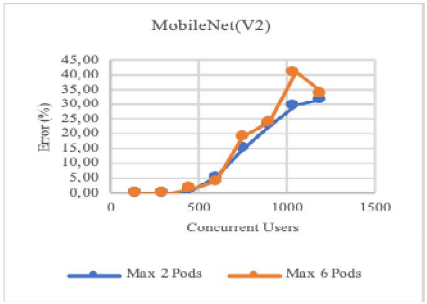

Figure 6: Performance Graph for MobileNet(V2) Model

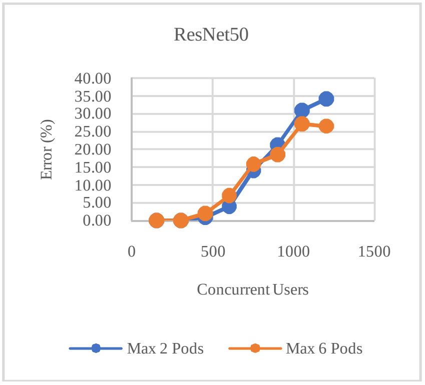

Figure 7: Performance Graph for ResNet50 Model

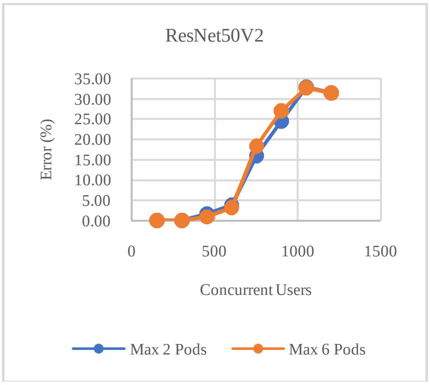

Figure 8: Performance Graph for ResNet50V2 Model

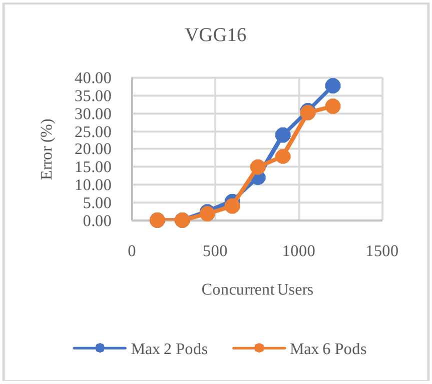

Figure 9: Performance Graph for VGG16 Model

5. CONCLUSION

Based on the discussion and evaluation carried out in the previous chapter, the conclusions that can be drawn from this study are, as follows:

- The best CNN method is to use the ResNet50 model with Adamax optimization with an accuracy value of 95.67%.
- Based on the results of the graph the error value on the difference in concurrent users with the number of pods in each model tested, the best deployment method is to use the deployment autoscaling method with a minimum number of pods 1 and a maximum of 6.
-

- The ResNet50 model has the best performance shown from the small number of errors compared to other models in the autoscaling method on deployment as well as with the most concurrent users.

REFERENCES

1. M. A. Arasi, **Survey of Machine Learning Techniques in Medical Imaging**, Int. J. Adv. Trends Comput. Sci. Eng., vol. 8, no. 5, pp. 2107–2116, Oct. 2019. https://doi.org/10.30534/ijatcse/2019/39852019
2. M. A. Ottom, **Classification of Mushroom Fungi Using Machine Learning Techniques**, Int. J. Adv. Trends Comput. Sci. Eng., vol. 8, no. 5, pp. 2378–2385, Oct. 2019. https://doi.org/10.30534/ijatcse/2019/78852019
3. N. A. Nasharuddin, **Multi-feature Vegetable Recognition using Machine Learning Approach on Leaf Images**, Int. J. Adv. Trends Comput. Sci. Eng., vol. 6, no. 4, pp. 1789–1794, Aug. 2019. https://doi.org/10.30534/ijatcse/2019/110842019
4. K. G. Kim, **Book Review: Deep Learning**, Healthc. Inform. Res., vol. 22, no. 4, p. 351, 2016. https://doi.org/10.4258/hir.2016.22.4.351
5. S. Ruder, **An overview of gradient descent optimization algorithms**, pp. 1–14, Sep. 2016.
6. K. P. Danukusumo, Pranowo, and M. Maslim, **Indonesia ancient temple classification using convolutional neural network**, in 2017 International Conference on Control, Electronics, Renewable Energy and Communications (ICCREC), 2017, pp. 50–54.
7. H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, **Large-Scale Image Retrieval with Attentive Deep Local Features**, in 2017 IEEE International Conference on Computer Vision (ICCV), 2017, vol. 2017-Octob, pp. 3476–3485. https://doi.org/10.1109/ICCV.2017.374
8. J. Hu, R. Ji, H. Liu, S. Zhang, C. Deng, and Q. Tian, **Towards Visual Feature Translation**, pp. 3004–3013, Dec. 2018.
9. A. Kumar, S. Bhowmick, N. Jayanthi, and S. Indu, **Improving Landmark Recognition using Saliency detection and Feature classification**, Nov. 2018.
10. K. B. da Cunha et al., **Patch PlaNet: Landmark Recognition with Patch Classification Using Convolutional Neural Networks**, in 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 2018, pp. 126–133.
11. Z. Chen, W. Zhang, Z. Kuang, and K. Y. K. Wong, **Learning local similarity with spatial relations for object retrieval**, MM 2019 - Proc. 27th ACM Int. Conf. Multimed., pp. 1703–1711, 2019. https://doi.org/10.1145/3343031.3351005
12. C. Termritthikun, S. Kanprachar, and P. Muneesawang, **NU-LiteNet: Mobile Landmark Recognition using Convolutional Neural Networks**, Evaluation, vol. 00, no. c, pp. 1–5, Oct. 2018. https://doi.org/10.37936/ecti-cit.2019131.165074
13. K. Ozaki and S. Yokoo, **Large-scale Landmark Retrieval/Recognition under a Noisy and Diverse Dataset**, Jun. 2019.