Jurnal Ilmu Komputer dan Informasi (Journal of Computer Science and Information)
16/1 (2023), 69-75. DOI: http://dx.doi.org/10.21609/jiki.v16i1.1118

Embedded Deep Learning System for Classification of Car Make and Model

Ari Wibisono, Hanif Arief Wisesa, Satria Bagus Wicaksono, Puteri Khatya Fahira
[1] Faculty of Computer Science
Universitas Indonesia Depok, Indonesia

E-mail: ari.w@cs.ui.ac.id

Abstract

Automatic car make, and model classification is essential to support activities of intelligent traffic systems in urban areas, such as surveillance, traffic information collection, statistics, etc. In order to classify this data, we need an embedded system approach for real-time car recognition. Many approaches could be made, from image processing to machine learning. Recently, the development of the Convolutional Neural Network has spurred various research in the Area. ResNet, Inception, DenseNet, and NasNet are some of the most commonly used Neural Network based method that is used to classify images. In this research, we utilize pre-processing and cropping technique to maximize the quality of dataset. Several deep learning networks are going to be compared in classifying vehicle make and model in the Stanford dataset. The dataset contains 196 different labels. Several evaluation metrics are used to compare the performance of the methods. From the experiment, the InceptionV3 method achieved the best performance of the AUROC ratio for training the dataset under 50 epochs. Other methods that achieve a high AUROC value tends to have a higher computational time. Real-time simulations have shown that the embedded system is capable of classifying a 100 % success rate for six concurrent users.
Keywords: *Embedded System Classification, Embedded Deep Learning, Car Classification*

1. Introduction

Convolutional Neural Networks has changed the way we conduct image and video processing. The algorithm contains multiple layers, which include convolutional layers, pooling layers, and fully connected layers. The supervised algorithm learns patterns in the image by convoluting each layer with the convolution kernels. As mentioned previously, one of the usages of the CNN method is to classify images from a given dataset [4]. Throughout the years, various research has attempted to improve the method by modifying the layers within the method. In this section, variations of CNN and its use will be examined. There are five different variations of CNN that will be examined in this study: DenseNet, Inception (V2 & V3), NasNetMobile, and ResNet.

The automatic car makes and model classification is essential to support Intelligent Traffic Systems [1]. The classification process is useful for activities such as surveillance, traffic information collection, and statistics in an urban area. Various research has been conducted to classify car make and model [2][3].

Convolutional Neural Networks (CNN) have revolutionized the approach of image and video processing. Since ImageNet classification was conducted using Deep Convolutional Neural Networks [4], the CNN and other similarly based methods were used to classify image data. A study [5] combines CNN with Recurrent Neural Networks (RNN) to create a framework that could classify multiple labels when given an image. Zoph et al. developed the NasNet to find the optimum convolutional layer to be implemented to the CIFAR-10 and the ImageNet datasets [6]. The method achieved a low error rate when tested with the datasets. The study [7] also tested their developed model on the CIFAR-10 and ImageNet datasets. The authors of the study developed the Dense Convolutional Network to solve degradation issues in a Deep Neural Network. In order to reduce computational costs, the study [8] developed a Deep Neural Network that is based on several design principles, that involves the reduction of the kernel dimension.

In this paper, the Neural Network-based methods will be compared using the Stanford Car Dataset. The methods that are going to be compared in this study are DenseNet, Inception (V2 & V3), NasNetMobile, and ResNet. The performance of each method will be evaluated using several metrics. This comparison aims to find the method that has the best tradeoff between training time, accuracy, and embedded system implementation.

The paper is organized as follows. The first section will give a brief introduction to several implementations of CNN based methods. Afterward, the Convolutional Neural Networks section will describe the Neural Network-based methods that will be compared in this study. The third section will describe the process of the experiment from pre-processing to the evaluation methods. The following section will provide the experiment results of this study. The final section will briefly summarize and conclude the findings of this paper.

2. Convolutional Neural Networks

Convolutional Neural Networks has changed the way we conduct image and video processing. The algorithm contains multiple layers, which include convolutional layers, pooling layers, and fully connected layers. The supervised algorithm learns patterns in the image by convoluting each layer with the convolution kernels. As mentioned previously, one of the usages of the CNN method is to classify images from a given dataset [4]. Throughout the years, various research has attempted to improve the method by modifying the layers within the method. In this section, variations of CNN and its use will be examined. There are five different variations of CNN that will be examined in this study: DenseNet, Inception (V2 & V3), NasNetMobile, and ResNet

A. Inception

In the original Inception (Inception V1), have a structure of 22 layers. The Inception has several design principles that affect the performance of the classification and the computational time [8]. The first design concept is that it avoids bottlenecks in the networks. It attempts to avoid bottlenecks by reducing the size gradually, starting from the input and to the output. Also, the representation of higher dimensions is less challenging to process locally. This results in a faster training process. Furthermore, using a lower-dimensional embedding does for spatial formation does not cause any loss in data representation. Therefore, the Dimensional representation in specific layers can be reduced without losing much information during the reduction process. Finally, the method balances the width and depth of the Network. This is done so that the optimal width and depth of the method is achieved, without using unnecessary extensions to improve the computational cost.

In this method, the construction of a block is formed by filters with different dimensions and sizes. The blocks could capture different features of an image in various positions. One main point is that the Network is not built by merely stacking convolutional kernels with various dimensions. Thus, this variation of deep Learning has the potential to decrease the computational cost.

B. Deep Residual Network

Deep Residual Networks are first proposed by He et al. [9]. The objective of this method is to minimize the degradation problem that occurs in various Neural Network methods. The degradation problem occurs when the layers inside the Neural Network structure becomes too deep. Therefore, with the deep layers of the Neural Network, the information might not be conveyed adequately, which decreases the accuracy of the model. Also, the assumption that every layer requires to have information from the previous layers used by the Feed Forward Network contributes to the decrease in the accuracy. In order to solve this issue, He et al. created a network that has layers f(n), which utilize input from several previous layers f(n – x). This process can be seen in Figure 1.

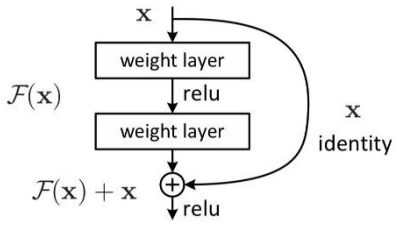

Figure. 1. Residual Learning proposed by He et al. [7].

C. DenseNet

DenseNet attempts to solve the same problem faced by Deep Residual Networks, which is the degradation problem. In this method, the approach is to connect every layer, which differs from the Residual Network that allows "hopping" between each Network [7]. Therefore, every layer can utilize various inputs from different layers. The output feature maps are concatenated with the incoming feature maps. The DenseNet has the same characteristics as the ResNet, in which the deeper the Neural Network layer, the more accurate the accuracy.

D. NasNet-Mobile

NasNet Mobile is a variant of Neural Network that is mainly utilized for image classification. NasNet Mobile utilized the Neural Architecture to determine the optimum

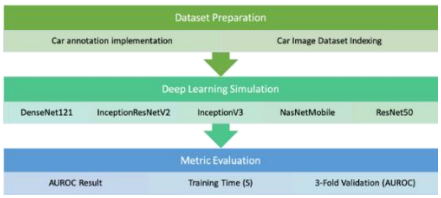

Figure. 2. Data processing steps in this research.

Figure. 3. Example of a cropped image.

convolutional layers for the dataset, to improve the performance of the classification [6]. Besides, the method could also reduce the computational cost required to process the data. The method can be used on mobile platforms by varying the combination of Normal and Reduction layers. Without modifying the dimensions of the image, the Normal layer obtains the feature maps of the input. Meanwhile, the Reduction layer reduces the feature map by a factor of two.

3. Methodology

In this research, there are three primary processes that will be conducted to classify car make and model: data preparation, deep learning simulation, and metric evaluation. The process conducted in this research can be seen in Figure 1.

A. Data Preparation

The dataset used in this research is the Stanford Car Dataset [10]. The main challenge of the Stanford Car Dataset is that there are many labels in the dataset, but for each label, there is not much training data contained in it. Annotation of the car image dataset is conducted before it is fed into the deep learning network. The first step in this annotation process is to crop each image in the dataset so that the image consists mostly of the car. This is done to minimize the noise (irrelevant parts of the image) in the image. The example of this cropping process can be seen in Figure 2.

After each image in the dataset have been cropped accordingly, the next step is to index each image according to their correct labels. In this dataset, there are 196 different labels. Each label represents the car make, model, and model year. For instance, the label Dodge Challenger SRT8 2011 is the example of a label the consists of the make (Dodge), the model (Challenger SRT8), and

Table 1. CSV Data Structure

Image	Finding Labels	Car Label 1	.	Car Label N
1.jpg	Car Label 1	1	.	0
…	…	0	.	0
M.jpg	Car Label N	0	.	1

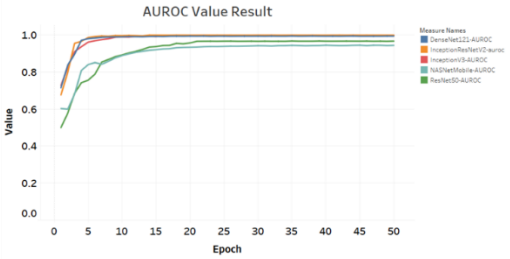

Figure. 4. AUROC Value results from the five different methods in various Epochs (training).

the model year (2011). The dataset is represented in a CSV file, which contains the image index, finding a label, and car labels. The structure of the CSV file can be seen in Table 1. After each image has been indexed in the CSV file, it is fed into the Network to create a new training model from the dataset.

B. Deep Learning Simulation

In this research, the comparison between popular Neural Network methods for image classification in terms of its training capabilities is conducted. The Neural Network methods that are compared in this research are DenseNet121, InceptionResNetV2, InceptionV3, NasNetMobile, and ResNet50. The basic principles of these methods were explained in the previous section.

C. Evaluation Metrics

Several evaluation metrics were utilized in this research. The first evaluation matrix that we used is the Area Under the Receiver Operating Characteristics (AUROC) Curve. The AUCROC curve is comprised of the Area Under The Curve (AUC) and the Receiver Operating Characteristics (ROC) curve [11]. In this evaluation matrix, the higher the Area AUC, the better the classification performance of the classifier. The method measures the True Positive Rate (TPR) and False Positive Rate (FPR) and plots them on the x and y axes, respectively. NasNetMobile and the ResNet50 failed to reach the AUROC Value of 1, although the methods managed to achieve the Value of AUROC ranging between 0.9 and 0.97. The experiment results presented in graphics can be seen in Figure 4.

From the best performing method, which is the InceptionV3 method, the AUROC Value is measured in every fold in the Cross-Validation

evaluation. In Figure 5, it can be seen that the curve is similar throughout all the different sets of folds. Therefore, it can be inferred that the method is consistent even when being presented with a different set of training and testing dataset. Another factor that is evaluated in this study is the training time of the model. As with all other methods, the lower the training time of the model, the more efficient the computation is. This research will evaluate the relationship between training time and the accuracy of the model.

The final evaluation is Cross- Validation. In the Cross-Validation method, the data will be divided into testing and training data [12]. During the evaluation, the classifier will be tested using the data which it has not seen before (data which is not included for the training process). One of the goals of this evaluation method is to avoid the overfitting problem to occur on the created model.

4. Experiment & Results

A. Experiment Setup

The implementation of the Neural Network models used the Keras Neural Network library with the TensorFlow library. We utilize pre-trained imagenet model to train the dataset. The training experiment was conducted using the NVIDIA® DGX-1. The resource is limited by the administrator. We only utilize 1 GPU (32 GB) to train the data. The experiment was conducted with a maximum of 50 Epoch for each training process. In the testing process, we utilize Jetson Nano for our embedded system. It has 128 CUDA Cores, 4 Core arm CPU, Memory 4 GB. We simulate two power modes, 5 watt and 10-watt power modes in Jetson Nano. Also, we make a variation in the number of processes.

For the testing purpose, we build simple API server based on flask. Then we load the best model from our training and testing evaluation which is the result from the DGX-1. We load the model using Jetson Nano. The Jetson Nano serve as a web API server, we test the API using another computer with this specifications Intel i7-4702 MQ,16 GB of RAM, and 512 GB SSD. The automation of API request is conducted by using JMeter [13]. The JMeter is conFigured to increase the number of concurrent user iteratively. The experiment results are described in Figure. 9 and Figure. 10.

B. Experiment Result

From the experiment results, it can be seen that the InceptionV3 Network achieved a near- perfect AUROC value after the 8th Epoch. From then onwards, the method achieved an AUROC value of 1. This is followed shortly by the DenseNet121.

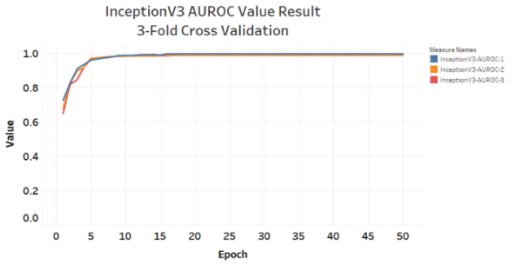

Figure. 5. AUROC Value in each fold of the InceptionV3 method in various Epochs (training).

Similar to the DenseNet121, the inceptionResNetv2 Network has a steep incline in AUROC Value in the first couple of Epoch. However, afterward, the increase of the AUROC Value gradually settles until it reaches the Value of 1 after 10 Epochs. In this experiment, the NasNetMobile and the ResNet50 failed to reach the AUROC Value of 1, although, the methods managed to achieve the Value of AUROC ranging between 0.9 and 0.97. The experiment results presented in graphics can be seen in Figure 4.

From the best performing method, which is the InceptionV3 method, the AUROC Value is measured in every fold in the Cross-Validation evaluation. In Figure 5, it can be seen that the curve is similar throughout all the different sets of folds. Therefore, it can be inferred that the method is inconsistent even when being presented with a different set of training and testing datasets.

From the validation loss, it can be seen that most of the methods have a relatively low loss in the first few Epochs. The ResNet50 method has a high loss in the initial Epochs, but it gradually declines after a few more Epochs. The trend of the validation loss experiment is similar to the trend in the AUROC value experiment. Both the NasNetMobile and the ResNet50 failed to reach a 0-validation loss value in the experiment after 50 Epochs. Meanwhile, the InceptionV3, DenseNet121, and InceptionResNetV2 managed to reach the 0-validation loss value, as seen in Figure 6. The DenseNet121 method has the least fluctuations in the Validation loss experiment. This means that it has the least variations of incorrect observations in various Epochs.

The result of the validation loss experiment in each fold of the InceptionV3 method can be seen in Figure 7. From the experiment, the validation loss from the first fold fluctuates, especially between the 5 and 15 Epochs. This means that there is a fluctuation in the incorrect classifying during those Epochs. The first fold reaches an almost zero validation loss value. For the second and third fold, while The validation loss results almost reach 0; it is actually twice larger than the first fold validation loss.

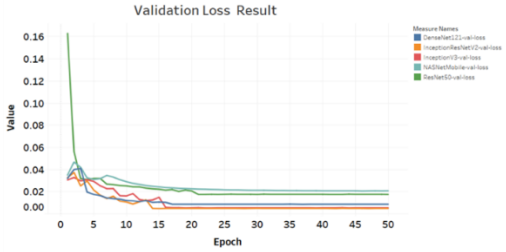

Figure. 6. Validation loss value from five different methods in various Epochs (training).

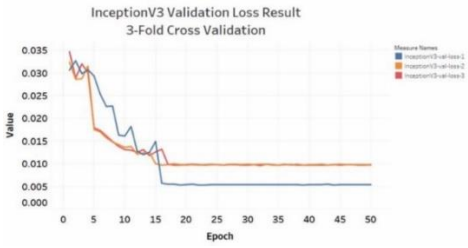

Figure. 7. Validation loss value in each fold of the InceptionV3 methods in various Epochs (training).

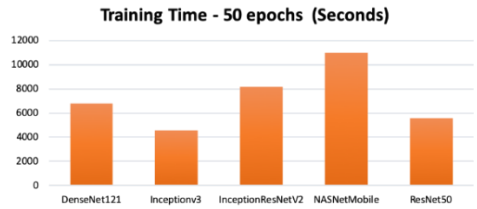

Figure. 8. Training time to reach 50 Epochs for each methods in seconds (training).

In Figure 8, the training time of each method are presented in a chart. From the Figure, it can be seen that the lowest training time is achieved by the InceptionV3 methods, with approximately 4500 seconds of training time. This is followed by the ResNet50 and the DenseNet121 methods, with 5700 and 6900 seconds of training times, respectively. A poor result is achieved by NasNetMobile method, with approximately 11000 seconds of training time. We choose Inception-V3 as a model to be implemented in the edge computer (Jetson Nano).

We choose a model that has good accuracy performance and fair inference time. Based on the benchmark evaluation of S. Bianco et.al [14], MobileNet can achieve the lowest inference time among others. However, it has a drawback in the accuracy result. It is not as good as the InceptionV3 or DenseNet121.Also, the edge computer that we use has 4 GB of GPU memory. It is capable of inference on the Inception V3 or DenseNet121 models.

After the training process is done, we have tried the best performance model that has been produced in

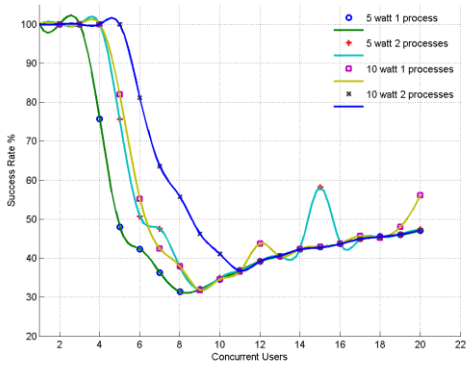

Figure. 9. Success Rate (%) simulation result from various scenarios (testing)

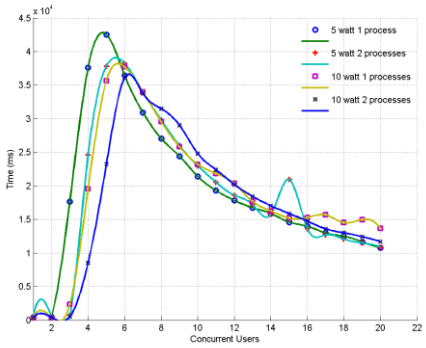

Figure. 10. Time consumption (ms) simulation result (testing)

the training process to be tested in an embedded system environment. Jetson Nano is a tool that we use to do simulations. This device can be easily linked to the camera at an affordable cost. The parameters that we measure in conducting this testing are the success rate and simulation time when the device must serve more than 1 user at a time. The user here is defined as the process of making a vehicle classification request.

Variation of scenarios is carried out, such as variation of power mode on Jetson Nano (5 watts and 10 watts) and variations in the number of processes (1 and 2 processes). Memory owned by Jetson Nano is 4 GB. Therefore the ability to load from the vehicle class can only be done for two processes.

Figure. 9. showed a comparison between the success rate and the number of current users. In general, success rate performance graphs are reduced by the increasing number of concurrent users. The best success rate is obtained by the scenario of 10 watts and two processes. This scenario has a success rate of 100% up to 6 concurrent user requests. In contrast, other processes have decreased the success rate to 80% when the number of concurrent users is 4.

In Figure. 10., it presents a comparison between the average time per process of each user who

makes a request to the system and the number of concurrent users. From the results of the simulation, the 10 watts 2 processes scenario has the best results with an average simulation time of 2,200 ms for four concurrent users. In comparison, the 5 watts 2 processes scenario requires around 3,700 ms. Moreover, the processes which have the most extended simulation time for four concurrent users is a 5 watts 1 process scenario that is above 4,200 ms.

Based on the AUROC, validation loss, and training time results, the most optimal performer between the Neural Network methods is the InceptionV3 method. This is because it has the best tradeoff between training time and accuracy. As explained in the previous paragraph, it reached 50 Epochs the fastest, even while maintaining a near-perfect AUROC value. Another method with a near- perfect AUROC value, the DenseNet121 and the InceptionResNetV2 was substantially slower than the InceptionV3. The results of this experiment support the argument of the design principles of Inception, which is that the reduction of dimension for the convolutional kernels does not necessarily mean a substantial loss in carried information.

Based on the results of simulation tests on the embedded system environment (Jetson Nano). The best results are obtained with a 10-watt scenario and two processes. Usually, some users only use a 5-watt of 1 process or 10-watt of 1 process. However, with the addition of processes with the same hardware, the performance results obtained are better than one process.

5. Conclusion

In this research, several popular Neural Network- based machine learning methods were compared to classify the make and model of Stanford Car dataset images. The methods that were compared in this study were the DenseNet121, InceptionV3, InceptionResNetv2, ResNet50, and the MobileNasNet. Several matrices were used to evaluate the performance of the methods. These matrices are AUROC, Time Performance, and the Cross-Validation method. Based on the experiment result, the InceptionV3 provides the best tradeoff between AUROC Value and training time compared to other methods. On the other hand, other methods that achieved high AUROC values tend to be very slow during the training process. Also, the embedded result simulation has shown that multiple processes with proper limitations still capable of achieving better performance compared to a single process. In the future, it is hoped that other Neural Network methods could also be compared in the study.

Acknowledgement

We would also like to thank Tokopedia-UI AI research center for the provision of the NVIDIA® DGX-1 used in this experiment.

References

[1] L. Figureueiredo, I. Jesus, J. Machado, J. Ferreira, J. M. de Carvalho, "Towards the development of intelligent transportation systems", Proc. Intell. Transp. Syst., vol. 88, pp. 1206-1211, 2001

[2] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale Convolutional Networks.", In *International Joint Conference on Neural Networks*, pp. 2809-2813, 2011.

[3] Y. Huang, R. Wu, Y. Sun, W. Wang, and X. Ding, "Vehicle logo recognition system based on convolutional neural networks with a pretraining strategy". *IEEE Transactions on Intelligent Transportation Systems*, *16*(4), 1951-1960, 2015.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks.", Advances in neural information processing systems. 2012.

[5] J. Wang, Y. Yang, J. Mao, Z. Huang. "Cnn-rnn: A unified framework for multi-label image classification." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[6] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8697-8710, 2018.

[7] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4700-4708, 2017.

[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, Zbigniew Wojna; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818-2826, 2016.

[9] He, Kaiming et al. "Deep Residual Learning For Image Recognition". 2016 IEEE Conference On Computer Vision And Pattern Recognition (CVPR), 2016. IEEE, doi:10.1109/cvpr.2016.90. Accessed 1 Dec 2018.

[10] J. Krause, M. Stark, D. Jia, F. Li, "3D Object Representations for Fine- Grained Categorization", 4th IEEE Workshop on 3D Representation and Recognition, at ICCV 2013 (3dRR-13). Sydney, Australia, 2013.

[11] C. Brown and H. Davis, "Receiver operating characteristics curves and related decision measures: A tutorial", Chemometrics and Intelligent Laboratory Systems, vol. 80, no. 1, pp. 24-38, 2006. Available: 10.1016/j.chemolab.2005.05.004.

[12] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", in the International Joint Conference on Artifcial Intelligence

(IJCAI), pp. 1137-1143, 1995.
[13] Apache JMeter https://jmeter.apache.org/. Accessed on 2021, June, 20.
[14] Bianco, S., Cadene, R., Celona, L., & Napoletano, P. (2018). Benchmark analysis of representative deep neural network architectures. IEEE access.