International Journal of Engineering & Technology, 7 (3.15) (2018) 95-98

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Evaluation of Pre-Trained Convolutional Neural Network Models for Object Recognition

M. Zabir[1]*, N. Fazira[2], Zaidah Ibrahim[3*] and Nurbaity Sabri[4]

[1,2,3]Faculty of Computer and Mathematical Sciences, Universiti of Teknologi MARA (UiTM), Shah Alam, Selangor, Malaysia
[4] Faculty of Computer and Mathematical Sciences, Universiti of Teknologi MARA (UiTM), Campus Jasin, Melaka, Malaysia
**Corresponding author E-mail: tsonouichi@gmail.com*

Abstract

This paper aims to evaluate the accuracy performance of pre-trained Convolutional Neural Network (CNN) models, namely AlexNet and GoogLeNet accompanied by one custom CNN. AlexNet and GoogLeNet have been proven for their good capabilities as these network models had entered ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and produce relatively good results. The evaluation results in this research are based on the accuracy, loss and time taken of the training and validation processes. The dataset used is Caltech101 by California Institute of Technology (Caltech) that contains 101 object categories. The result reveals that custom CNN architecture produces 91.05% accuracy whereas AlexNet and GoogLeNet achieve similar accuracy which is 99.65%. GoogLeNet consistency arrives at an early training stage and provides minimum error function compared to the other two models.

Keywords: *CNN;AlexNet;GoogLeNet;Caltech101.*

1. Introduction

Convolutional Neural Network (CNN) is an artificial neural network that has been used to analyze images in machine learning. CNN was inspired by biological process on the connectivity patterns that neurons follow to deal with visual cortex. CNN uses variation of the layers that does various things that is necessary to perform image processing to reduce and minimalize the pre-processing. Hence, CNN can be concluded as relatively little usage of pre-processing other than other image processing.

There are lots of network models to be used to train the computer to identify objects in an image. There is a competition held every year from 2010 until 2016, ImageNet Large Scale Visual Recognition Challenge (ILSVRC), hosted by ImageNet. The purpose of the competition is to challenge the network models from any organization. The winner of the competition will be announced as the best network model for that year without any argument [1].For this study, we had used two types of pre-trained CNN model which are AlexNet and GoogLeNet that are famously used and attended ImageNet Large Scale Visual Recognition Challenge (ILSVRC) where it used the datasets from ImageNet. This study is also accompanied with the custom CNN.

Custom CNN architecture is customized according to the researcher's needs. AlexNet uses layers property that consists of 25 layers. Meanwhile, GoogLeNet has Inception module that helps in reducing the parameters by using small convolutions layers [2]. The result from these models will determine the best technique for the image classification. The data that we are using is Caltech101 by California Institute of Technology (Caltech) for computational visual datasets. This dataset contains 101 object categories. Hence, we will train the network models and use the selected network models and monitor their accuracy, loss and time taken. From that point we will seek which is the best among all.

MATLAB R2017b will be used as the platform to run the CNN to train the network models. Unfortunately, the machine used for the experiment does not have a GPU that supported CUDA. Thus, it uses a single CPU to run the whole process. This report will also explain the necessary and optional steps to perform the transfer learning for object recognition.

2. Convolutional Neural Network

2.1. Custom CNN

The bulk of the custom CNN turns the input image into sets of features. The last few layers are used to perform object recognition. The layer themselves comes with many parameters known as weights. The weights determine how the layers behave when the data is passing through them. The value of those weight are determined by training the network for unknown data. The behavior of the network is learnt from these training data.

The images are introduced for each category to our network. The layers are defined and coded using Matlab 2018a. With this, the network creating and training are being tested out by showing them various new images to identify their accuracy. Fig. 1 shows the typical CNN architecture that consists of various combinations of layers [3]. The convolutional layer extracts features of an image by using a filter that stride over the input image producing a feature map. Max pooling layer calculates that maximum value of the feature map. Neighbouring max pooling accepts input from feature maps that are shifted by more than one row or columns. This operation reduces the dimension of the feature map. The fully-connected layer performs the recognition process.

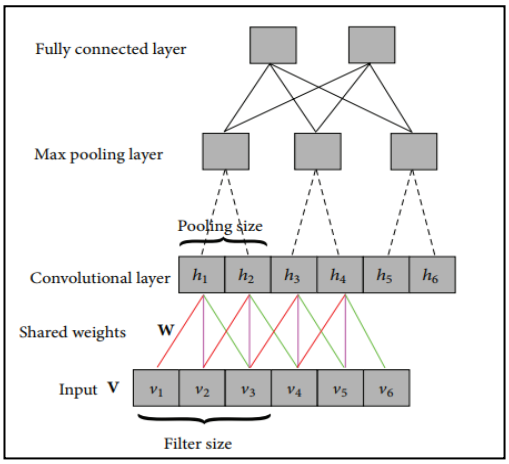

Fig. 1: CNN Architecture

2.2. AlexNet

AlexNet proclaimed to have trained more than a million images and is able to classify the images to 1000 object categories. Hence, it has learnt rich variety of feature representations for a wide range of images. Due to that, it is considered as one good alternative for feature extractions [4]. AlexNet uses layers property that comprises of 25 layers. There are 8 layers for learnable weights, 5 convolutional layers and 3 fully connected layers. To set up the network in Matlab is by typing the following statement:

net = alexnet()

Fig. 2 shows the details of all the layers of AlexNet.

```
25x1 Layer array with layers:

   1   'data'     Image Input                  227x227x3 images with 'zerocenter' normalization
   2   'conv1'    Convolution                  96 11x11x3 convolutions with stride [4  4] and padding [0  0]
   3   'relu1'    ReLU                         ReLU
   4   'norm1'    Cross Channel Normalization  cross channel normalization with 5 channels per element
   5   'pool1'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0]
   6   'conv2'    Convolution                  256 5x5x48 convolutions with stride [1  1] and padding [2  2]
   7   'relu2'    ReLU                         ReLU
   8   'norm2'    Cross Channel Normalization  cross channel normalization with 5 channels per element
   9   'pool2'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0]
   10  'conv3'    Convolution                  384 3x3x256 convolutions with stride [1  1] and padding [1  1]
   11  'relu3'    ReLU                         ReLU
   12  'conv4'    Convolution                  384 3x3x192 convolutions with stride [1  1] and padding [1  1]
   13  'relu4'    ReLU                         ReLU
   14  'conv5'    Convolution                  256 3x3x192 convolutions with stride [1  1] and padding [1  1]
   15  'relu5'    ReLU                         ReLU
   16  'pool5'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0]
   17  'fc6'      Fully Connected              4096 fully connected layer
   18  'relu6'    ReLU                         ReLU
   19  'drop6'    Dropout                      50% dropout
   20  'fc7'      Fully Connected              4096 fully connected layer
   21  'relu7'    ReLU                         ReLU
   22  'drop7'    Dropout                      50% dropout
   23  'fc8'      Fully Connected              1000 fully connected layer
   24  'prob'     Softmax                      softmax
   25  'output'   Classification Output        crossentropyex with 'tench', 'goldfish', and 998 other classes
```

Fig.2: AlexNet Layers.

The 23rd and 25th layer is configured for 1000 classes which AlexNet was originally pre-trained. The last three layers are fine-tuned and specify again to cater around a new dataset. The changes of the last three layers are necessary for such a small dataset to be trained as the learning parameters can be transferred from the previous training processes to a specific dataset.

2.2. GoogLeNet

Similar to AlexNet, GoogLeNet also proclaimed to have been trained with more than a million images and is able to classify 1000 object categories as they took the same challenge. Inception is similar with GoogLeNet but is has more layers [4]. The purpose of the deep layers of Inception is to increase the width of the training parameters which leads to better classification results but the training process is longer compared to GoogLeNet [5]. GoogLeNet introduces Inception module which drastically reduces the parameters by using small convolutions layers as shown in Figure 3. Top Inception module is the naïve version whereas the bottom is the Inception module with dimensionality reduction.

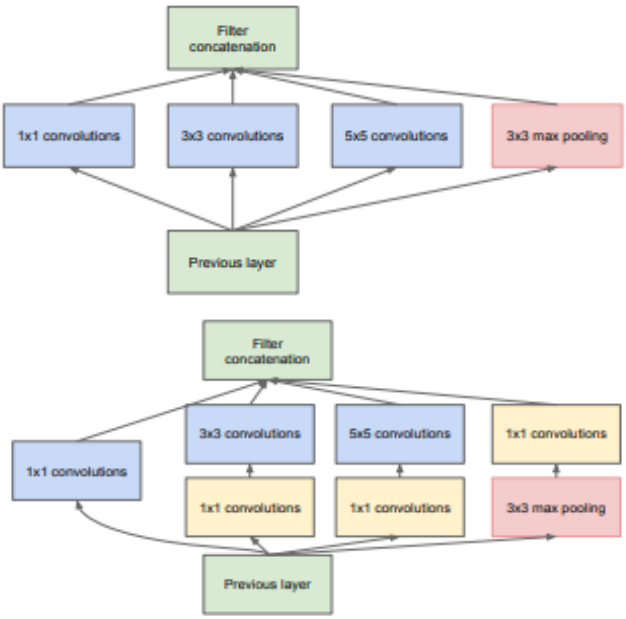

Fig.3: Inception Module of GoogLeNet [5].

The network can be used in Matlab by typing the following command.

net = GoogLeNet()

GoogLeNet produces a feature called LayerGraph where it can be seen in Figure 4. Similar to AlexNet, GoogLeNet still requires fine-tuning in classification layers to cater a new dataset. Fig. 4 shows the architecture of GoogLeNet [6].

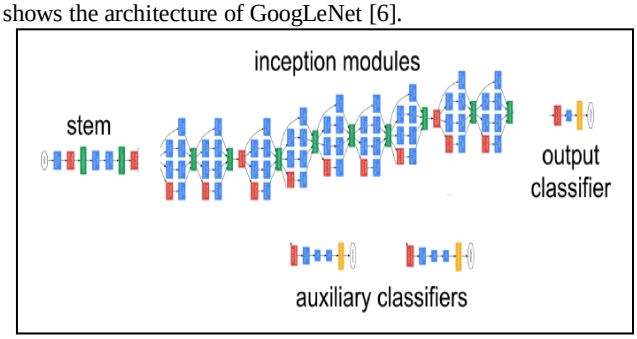

Fig.4: GoogLeNet Architecture [6].

One of the advantages of this architecture is that it allows the increase in the number of units at each stage significantly with less computational complexity. GoogLeNet takes up more time for the training process compared to AlexNet due to its parameters and learning ability. Nevertheless, the accuracy is much higher compared to AlexNet [7].

Evaluation of pre-trained CNN models has been performed by other research groups to determine which model produces better results in terms of their accuracy and speed. Fig. 5 indicates that GoogLeNet achieve better result compared to AlexNet [8][9]. GoogLeNet is said to give better results as compared to AlexNet due to its network that are designed with more layers which can help in extracting more features for classification [10].

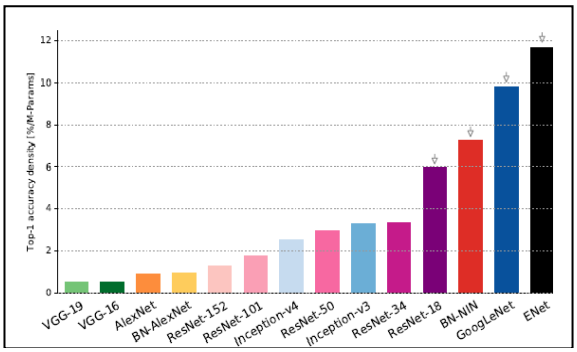

Fig.5: Top Accuracy among Neural Network Model [8][9].

3. Results And Analysis

In this experiment, Caltech 101 dataset that consists of 101 categories with 40 – 800 images per category was used [11]. Fig. 6 illustrates some sample images from Caltech 101 dataset. For custom CNN, the accuracy rate is 91.05%, with overall process duration of 1 minute and 30 seconds. It started to achieve consistency after the 10th iteration of validation whereas the loss function gradually decreases until it reaches the 30th iteration. Figure 7 visualizes the training progress of Caltech101 using custom CNN.

Fig.6: Sample images from Caltech 101 dataset [11].

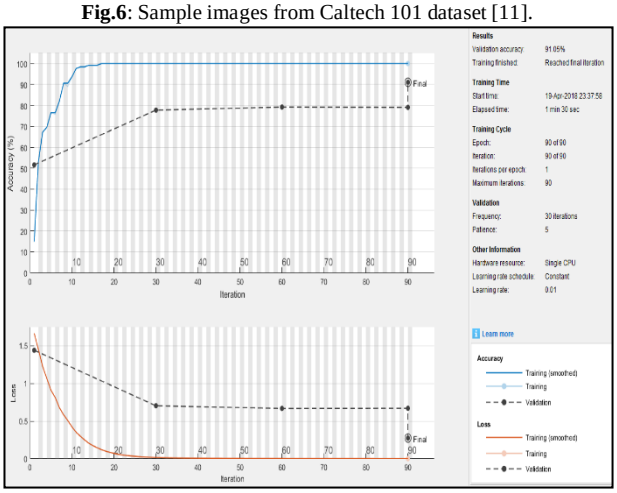

Fig.7: Training progress of Caltech101 using custom CNN.

By looking at Figure 7, we can see that after the 10th epoch, the consistency of the accuracy remains throughout the whole process until it reaches the 90th epochs of training for the above model. Meanwhile in AlexNet, the classification accuracy reached 99.65% in 54 minutes. Top graph denotes the classification accuracy for validation data while bottom graph denotes loss function for validation and training data as shown in Figure 8. AlexNet rapidly increases its accuracy at the beginning and maintain in a steady state after the 60th iteration. The error function decreases to a stable level as the iteration closes to 80.

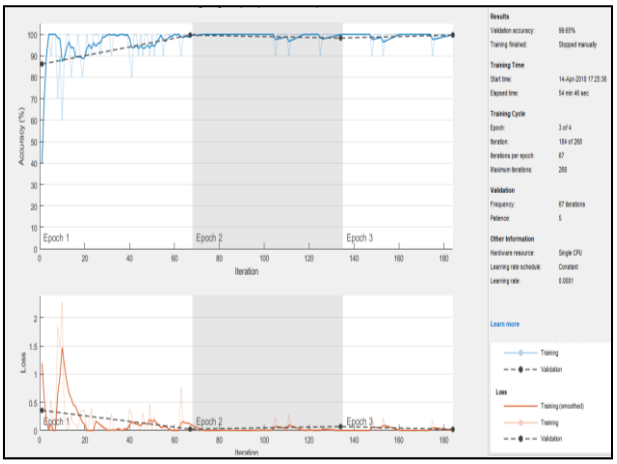

Fig. 8: Training Progress of Caltech101 using AlexNet.

GoogLeNet yields similar accuracy which is 99.65% at 99th iterations and maintain the consistency of accuracy starting at the 12th iteration, and the rate is increasing as the iteration increases. Fig. 9 indicates the GoogLeNet's training result.

Initializing image normalization.

Epoch	Iteration	Time Elapsed (seconds)	Mini-batch Loss	Validation Loss	Mini-batch Accuracy	Validation Accuracy
1	1	29.80	1.7174	0.6025	30.00%	84.78%
1	2	255.06	0.7999		80.00%	
1	3	278.67	0.0276	0.4695	100.00%	83.04%
1	4	502.19	0.4911		80.00%	
1	5	528.65	0.4580		90.00%	
1	6	552.70	1.0322	0.5135	70.00%	83.04%
1	49	4469.13	0.0370		100.00%	
1	50	4491.62	0.0000		100.00%	
1	51	4513.97	0.0086	0.0254	100.00%	99.31%
1	52	4709.04	0.0010		100.00%	
1	53	4731.29	0.0942		100.00%	
1	54	4753.55	0.0000	0.0260	100.00%	99.65%
1	55	4948.83	0.0144		100.00%	
1	56	4971.14	0.0433		100.00%	
1	57	4993.49	0.0590	0.0264	100.00%	99.65%
1	58	5188.52	0.0000		100.00%	
1	59	5210.78	0.0003		100.00%	
1	60	5233.05	0.0012	0.0272	100.00%	99.65%
1	61	5428.59	0.0105		100.00%	
1	62	5451.06	0.0002		100.00%	
1	63	5474.00	0.4230	0.0258	90.00%	99.65%
2	96	8121.35	0.0027	0.0239	100.00%	99.65%
2	97	8560.19	0.1029		90.00%	
2	98	8582.57	0.0000		100.00%	
2	99	8604.90	0.0001	0.0238	100.00%	99.65%

Fig. 9: Training progress of Caltech101 using GoogLeNet.

From the experimental results we can see that AlexNet starts to achieve consistent accuracy at the 78th iteration. However, GoogLeNet starts to produce accurate results consistently at the 54th iteration. Even though the accuracy is the same for both pre-trained CNN models, GoogLeNet's consistency rate increases after the 6th iteration.

4. Conclusion

GoogLeNet achieves better result compared to AlexNet and custom CNN models. Even though AlexNet and GoogLeNet yield similar accuracy which is 99.65%, GoogLeNet achieves its consistency at an earlier rate. The error function drops early below 10th epoch for GoogLeNet. These leads to better object recognition results by GoogLeNet compared with the other CNN models. Thus, if we do not have millions of data for training, applying pre-trained CNN model can still produce excellent results. Future work is to experiment on other pre-trained CNN models and other datasets to determine which pre-trained CNN model is better for which type of images.

Acknowledgement

The authors would like to thank Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Selangor, for sponsoring this research.

References

[1] Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Summers, R. M. (2016). Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. IEEE Transactions on Medical Imaging, 35(5), 1285–1298.

[2] Rodrigues, L. F., Naldi, M. C., & Mari, J. F. (2017). HEp-2 Cell Image Classification Based on Convolutional Neural Networks. 2017 Workshop of Computer Vision (WVC), 13–18.

[3] Hu, W., Huang, Y., Wei, L., Zhang, F., & Li, H. (2015). Deep convolutional neural networks for hyperspectral image classification. Journal of Sensors, pg no

[4] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D. Rabinovich, A. (2014). Going Deeper with Convolutions. arXiv:1409.4842. http://doi.org/10.1109/CVPR.2015.7298594

[5] B. Baiping, X. Han, & D. Wu. (2018) Real-Time Crowd Density Estimation Based on Convolutional Neural Networks. 2018 International Conference on Intelligent Transportation, Big Data & Smart City, 691-694

[6] Subramanian, S. V. (2018). Deep Convolutional For Tiny ImageNet Classification∗, 17319279.

[7] No, E. (n.d.). A PERFORMANCE ANALYSIS OF CONVOLUTIONAL NEURAL NETWORK MODELS IN SAR TARGET RECOGNITION Naval Aeronautical and Astronautical University , Department of Electronic and Information, (188).

[8] Vu, M., Francophone, U., &Beurton-aimar, M. (2018). Heritage Image Classification by Convolution Neural Networks, 1–6.

[9] Gu, S., & Rigazio, L. (2015). Towards Deep Neural Nettwork Architectures, (2013), 1–9.

[10] Canziani, A., Culurciello, E., & Paszke, A. (2017). Evaluation of neural network architectures for embedded systems. 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 1–4.

[11] L. Fei-Fei, R. Fergus and P. Perona. (2014). Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. IEEE. CVPR 2004, Workshop on Generative-Model Based Vision.