Image Classification Using CNN Architectures*

*Note: Image classification is done on Natural Scenes around the world from Kaggle

Alishba Tahir
School of Electronic Engineering and Computer Science (SEECS))
National University of Science and Technology (NUST)
Islamabad, Pakistan
atahir.mscs19seecs@seecs.edu.pk

***Abstract*—Image classification of the data set, based on Natural Scenes around the world, has been done using multiple CNN models/architectures.**

***Index Terms*—VGG16, InceptionResNetV2, Hyper Parameter Training**

I. Introduction

Convolutional neural network (CNN), also known as deep learning neural network, is a process of applying multiple algorithms on unstructured data (mostly popular on images) in an efficient way. These are known to be much faster than Multi-layer Neural Network. They have become most popular in the field of computer vision where some CNNS have become benchmarks for visual application problems, such as classification. They have also been found to be successful in Natural Language Processing for its efficient processing on text classification.

CNNs can find patterns on images such as shapes, segments, faces or eyes (also used in object detection). CNNs, unlike classic computer vision algorithms, exceed in time proficiency that it can operate on raw images (without needing pre-processing) and give results in real time.

CNNs are feed-forward neural network, which is often 20 or 30 layers long, or even longer. With a special kind of layer known as the convolutional layer which performs convolutions on input set of uni or multi dimensional arrays. They use receptive fields and shared weights for their faster processing in comparison to classic neural networks.

A few (3-4) convolutional layers can recognize hand written digits while approximately 25 layers can distinguish faces or objects. The CNNs are designed to mirror a human visual cortex structure where an ensemble of layers process an incoming image and progressively are trained to learn more complex or detailed images.

These convolutional layers are stacked up on each other, recognizing and detecting features whose level of refinement increases as the number of layers. The initial layers detect basic features while the later layers are used to detect higher level features.

Reusing existing models is called **transfer learning**. By invoking the above stated facts we can train just the top layers of any existing model with already trained weights to our data set. This is conditioned to that the model was trained on a similar data set and/or the trained model is used for a similar application to which it was originally trained for.

There have been many models developed for classification purposes. From Alexnet 2012**??**, Vgg 2014**??**, GoogleNet 2014, Resnet 2015, SENet 2017 to many more complex models being made by computer vision experts all around the world.

I have applied some pre existing classification CNN models to classify natural image scenes around the world (of 6 different classes). The further sections detail the models implemented with different hyper-parameters.

II. Methodology

I have used Vgg16 and InceptionResNetV2 on multiple hyper-parameters. This is varying from augmented data to real instances of data, with multiple optimizers with adaptive or static learning rates.

The best accuracy was provided by vgg16. Figure 1 is the Network diagram of the model used by me to classify Natural scenes around the world into 6 classes.

The best accuracy on prediction data I got was 89.138% using vgg16 model, resizing the input images to (90,90,3), 10 epochs, stocastic gradient descent optimizer, learning rate 0.001, decay 1e-7, momentum 0.9, loss categorical cross entropy and metrics accuracy. When I tried to use augmented data (specifically on the fly augmentation) the accuracy dropped to 85% while all other parameters were the same.

III. Results

A compination of CNN models with different hyper-parameters were tuned, including but not limited to vgg16, InceptionResNetV2, With optimizers were sgd, adam rmsprop. Adaptive learning rate was also used in conjunction to adam and rmsprop separately. Augmented data used was Keras on the fly with training data augmentation values target size (150,150), batch size 16 32 64, shuffle true, class mode categorical and interpolation nearest. Early stopping is a very powerful tool albeit I discovered after having seen many multiple training and validation accuracy graphs.

The best accuracy was with vgg16 with 89% accuracy and hyper parameters discussed in the above section.

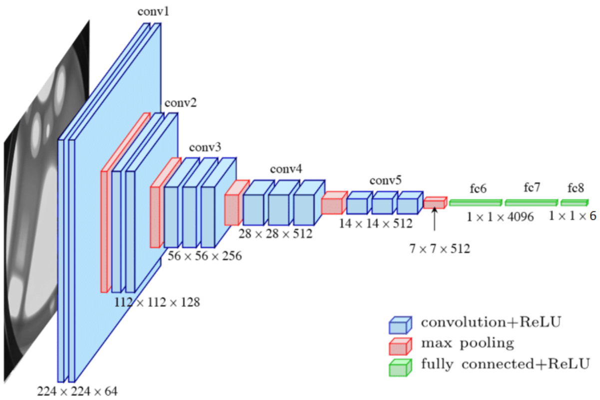


Fig. 1. VGG16 Architecture

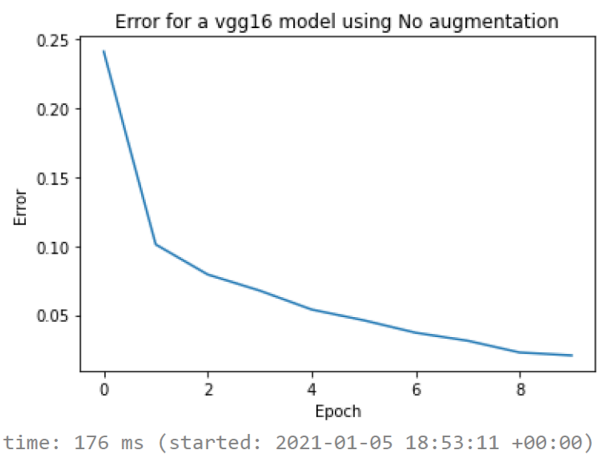


Fig. 2. Train error

IV. Discussion and Conclusion

The reason I think hyper-parameter tuning or augmented data didn't increase the accuracy of the data predicted was because the models were imported or transferred with already enhanced weights. My hyperparameter tuning did not then help much with the cause.

V. Github Repository Link

My Github Repository Link: https://github.com/A-Janj/Natural-Image-Classification-with-Different-CNN-Models

TABLE I
Table Type Styles

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

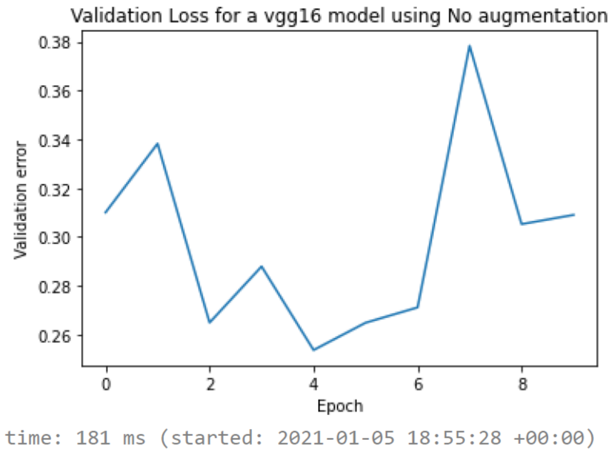


Fig. 3. validation error

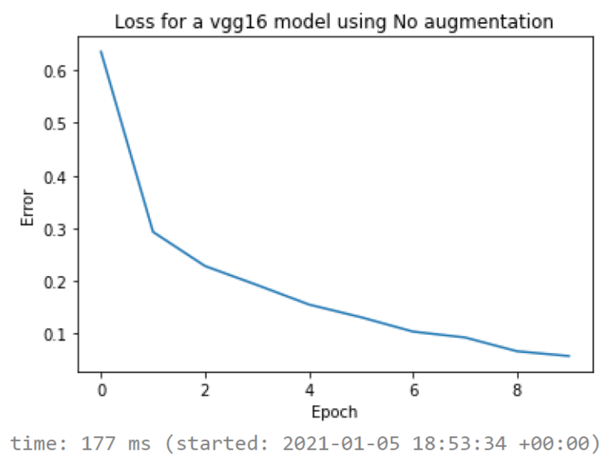


Fig. 4. Train loss

Acknowledgment

All praise to Almighty ALLAH who created the heavens and the earth, and who gave me the ability to learn. The class of Computer Vision is grateful to Dr.Moazam Fraz for having the wisdom of assigning the task so we could learn more about computer vision. Teacher's Assistant of the courses were extremely helpful. I am obliged to Syed Nauyan Rashid and Asim Khan. I am indebted to my friends who helped me whenever I faced obstacles to complete the task.

References

Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). 2012 AlexNet. Adv. Neural Inf. Process. Syst., 1-9.

Yan, Z., Zhang, H., Piramuthu, R., Jagadeesh, V., DeCoste, D., Di, W., Yu, Y. (2014). HD-CNN: Hierarchical deep convolutional neural network for large scale visual recognition. arXiv preprint arXiv:1410.0736.

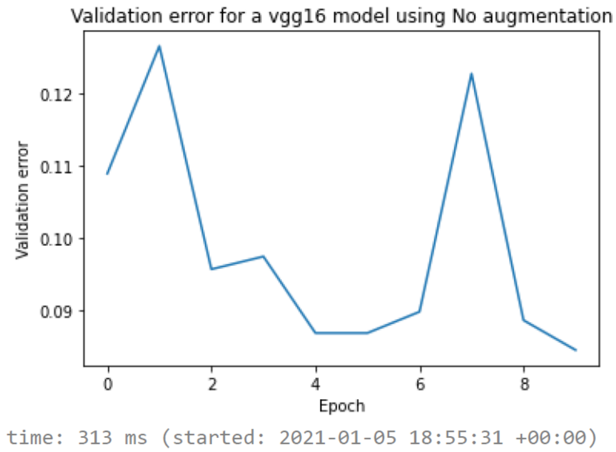
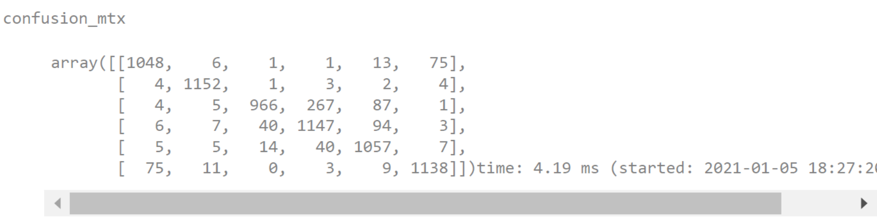


Fig. 5. validation loss



```
confusion_mtx

array([[1048,    6,    1,    1,   13,   75],
       [   4, 1152,    1,    3,    2,    4],
       [   4,    5,  966,  267,   87,    1],
       [   6,    7,   40, 1147,   94,    3],
       [   5,    5,   14,   40, 1057,    7],
       [  75,   11,    0,    3,    9, 1138]])time: 4.19 ms (started: 2021-01-05 18:27:2
```

Fig. 6. Confusion matrix values

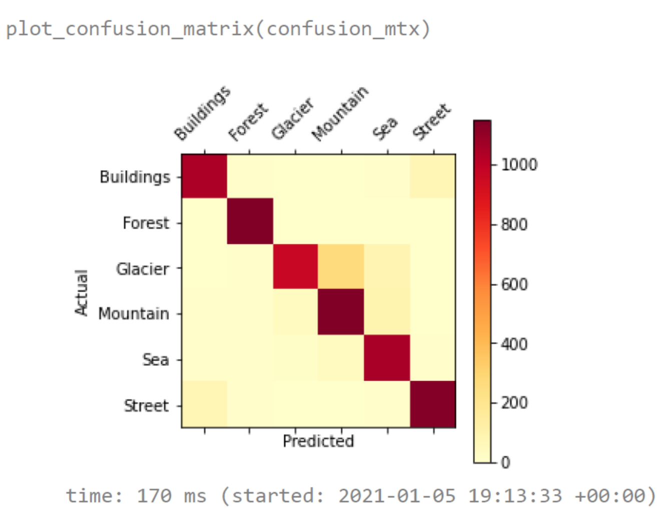


Fig. 7. Confusion Matrix heat map

Fig. 8. Example of a figure caption.