Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images

Srikanth Tammina[*]

[*] Electrical Engineering, Indian Institute of Technology, Hyderabad

DOI: 10.29322/IJSRP.9.10.2019.p9420
http://dx.doi.org/10.29322/IJSRP.9.10.2019.p9420

Abstract— Traditionally, data mining algorithms and machine learning algorithms are engineered to approach the problems in isolation. These algorithms are employed to train the model in separation on a specific feature space and same distribution. Depending on the business case, a model is trained by applying a machine learning algorithm for a specific task. A widespread assumption in the field of machine learning is that training data and test data must have identical feature spaces with the underlying distribution. On the contrary, in real world this assumption may not hold and thus models need to be rebuilt from the scratch if features and distribution changes. It is an arduous process to collect related training data and rebuild the models. In such cases, Transferring of Knowledge or transfer learning from disparate domains would be desirable. Transfer learning is a method of reusing a pre-trained model knowledge for another task. Transfer learning can be used for classification, regression and clustering problems. This paper uses one of the pre-trained models – VGG - 16 with Deep Convolutional Neural Network to classify images.

Index Terms—Deep Convolutional Neural Network, Image classification, Machine learning, Transfer learning, VGG – 16.

I. INTRODUCTION

Humans have an intrinsic skill to transfer knowledge across different activities. The knowledge that we acquire while performing one specific activity, we operate in the same manner to solve related activities. “Transfer” is a cognitive practice whereby a learner’s mastery of knowledge or skills in one context enables them to apply that knowledge or skill in a different context. Machine learning and deep learning algorithms have been conventionally designed to work for a specific feature-space distribution. Once the feature-space distribution changes, models need to be redesign from scratch, and it is also cumbersome task to gather the required training data. Consequently, since deep learning models during training need enough labelled data. Hence it is nearly impossible to create a machine learning based model for a target domain which consists of very few labelled data for supervised learning. In such cases, Transfer learning would significantly improve the performance of learning. The main idea behind transfer learning is to borrow labelled data or knowledge extracted from some related domains to help a machine learning algorithm to achieve greater performance in the domain of interest.

Figure 1 shows the contrast between the processes of conventional machine learning and transfer learning. As we can see in a conventional machine learning, it tries to learn each disparate task separately with different learning system, while transfer learning tries to extract the knowledge from previous source tasks to a target tasks where the latter has very few labelled data for supervised learning.

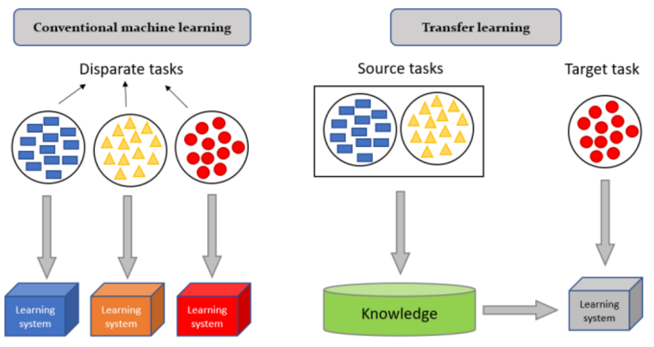


Fig. 1. Comparative diagram of Learning Processes between Conventional Machine Learning and Transfer Learning

Example – If objective of problem is to identify objects in images within a constrained domain space of a Region1 (R1). We collect the data set for domain space (R1) and train a machine learning model to extract features and predict on the test data. Consequently, we tune it to perform better on test data points from the same domain space of Region1 (R1). The main drawback of using traditional deep learning algorithms is, it underperforms when we do not supply with enough training data for required tasks in domain space Region1 (R1). In a different scenario, if we need to detect objects from images in a completely disparate Region2 (R2), then we should be able to apply the model, which is trained for R1, but we face performance degradation and models do not predict accurately in reality. This is where transfer learning can be used, transfer learning makes use of pre trained models which are trained on previously learned tasks and apply them onto newer, related tasks. Because of the scarcity of dataset in Region2, and as model is biased towards training data of Region1(R1), we encounter high error rate. In the case of image classification, certain low-level features, such as edges, shapes, lightning, can be shared across tasks to enable the knowledge transfer among tasks.

II. Background and Related Work

The study of transfer learning is motivated by the fact that people can intelligently apply knowledge learned previously to solve new problems faster or with better solutions [1]. The fundamental motivation for transfer learning in the field of machine learning was discussed in a NIPS-95 workshop on "Learning to Learn" [2], which focused on the need for lifelong machine learning methods that retain and reuse previously learned knowledge. A learning technique to transfer learning is the multi-task learning framework [3], which tries to learn multiple tasks simultaneously even when they are different. A typical approach for multi-task learning is to uncover the common (latent) features that can benefit each task. In 2005, the years after the NIPS-05 workshop, the Broad Agency Announcement (BAA) 05-29 of Defense Advanced Research Projects Agency (DARPA)'s Information Processing Technology Office (IPTO) 2 gave a new mission of transfer learning: the ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks. In this definition, transfer learning aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task. In contrast to multi-task learning, rather than learning all the source and target tasks simultaneously, transfer learning cares most about the target task. The roles of the source and target tasks are no longer symmetric in transfer learning.

III. System Model

We designed a CNN model by using python language. We used open-source software library called TensorFlow which is widely used for machine learning applications such as neural network and used Keras, which works as wrapper and it is a high-level neural network library that's built on top of TensorFlow. Jupyter notebook is used for development environment.

IV. Problem Statement

In this paper we will be working on an image classification problem with the restriction of having a small number of training samples per category. The data archive consists of images of dogs and cats and objective of the task is to build a heuristic and robust model that can categorize images into bins of cats and dogs separately.

V. Solution

Initially, we will be using Convolutional neural network where the input images are passed through a series of layers i.e. convolutional, pooling, flattening and fully connected layers, and then the output of CNN is generated which classify images. After building CNN models from the scratch, then we will try to fine tune the model by using image augmentation technique. Consequently, we will employ one of the pretrained model – VGG-16 to classify image and check accuracy for training data and validation data.

A. Convolutional Neural Network

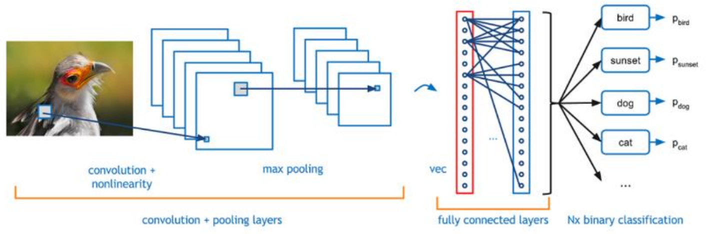


Fig. 2. Architecture of Convolutional neural network

Convolutional neural network is a type of artificial neural network that uses multiple perceptron that analyze image inputs and have learnable weights and bases to several parts of images and able to segregate each other. One advantage of using Convolutional Neural Network is it leverages the use of local spatial coherence in the input images, which allow them to have fewer weights as some parameters are shared. This process is clearly efficient in terms of memory and complexity. The basic building blocks of convolutional neural network are as follows:

a. *Convolution Layer* – In convolutional layer, a matrix named kernel is passed over the input matrix to create a feature map for the next layer. We execute a mathematical operation called convolution by sliding the Kernel matrix over the input matrix. At every location, an element wise matrix multiplication is performed and sums the result onto the feature map. Convolution is a specialized kind of linear operation which is widely used in variety of domains including image processing, statistics, physics. Convolution can be applied over more than 1 axis. If we have a 2-Dimensional image input, I, and a 2-Dimensional kernel filter, K, the convoluted image is calculated as follows:

$$S(i,j) = \sum_{m}\sum_{n} I(m,n)k(i-m,j-n)$$

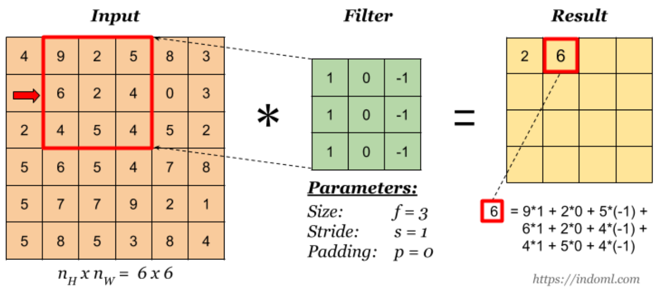


Fig. 3. Element wise matrix multiplication and summation of the results onto feature map in convolutional layer.

b. Non-linear activation functions (ReLU) – Activation function is a node that comes after convolutional layer and the activation function is the nonlinear transformation that we do over the input signal. The rectified linear unit activation function (ReLU) is a piecewise linear function that will output the input if is positive, otherwise it will output zero.

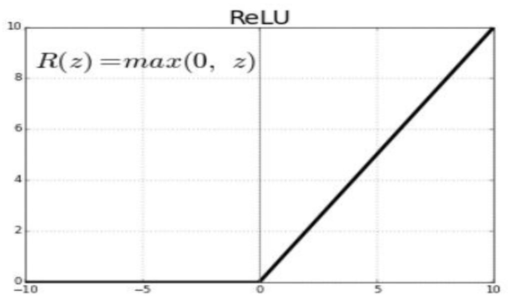


Fig. 4. ReLU activation function

c. Pooling Layer – The drawback of the feature map output of convolutional layer is that it records the precise position of features in the input. This means during cropping, rotation or any other minor changes to the input image will completely results in a different feature map. To counter this problem, we approach down sampling of convolutional layers. Down sampling can be achieved by applying a pooling layer after nonlinearity layer. Pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change.

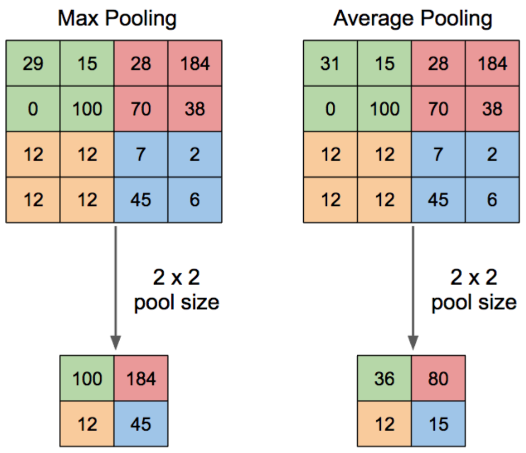


Fig. 5. Types of pooling

d. Fully Connected Layer - At the end of a convolutional neural network, the output of the last Pooling Layer acts as input to the Fully Connected Layer. There can be one or more of these layers. Fully connected means that every node in the first layer is connected to every node in the second layer.

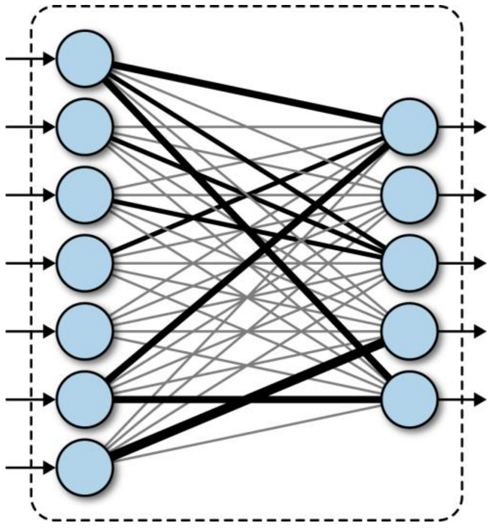


Fig. 6. Fully connected layer – Image shows every single node in the first layer is connected to every node in the second layer

B. VGG-16 Model –

VGG-16 Architecture –

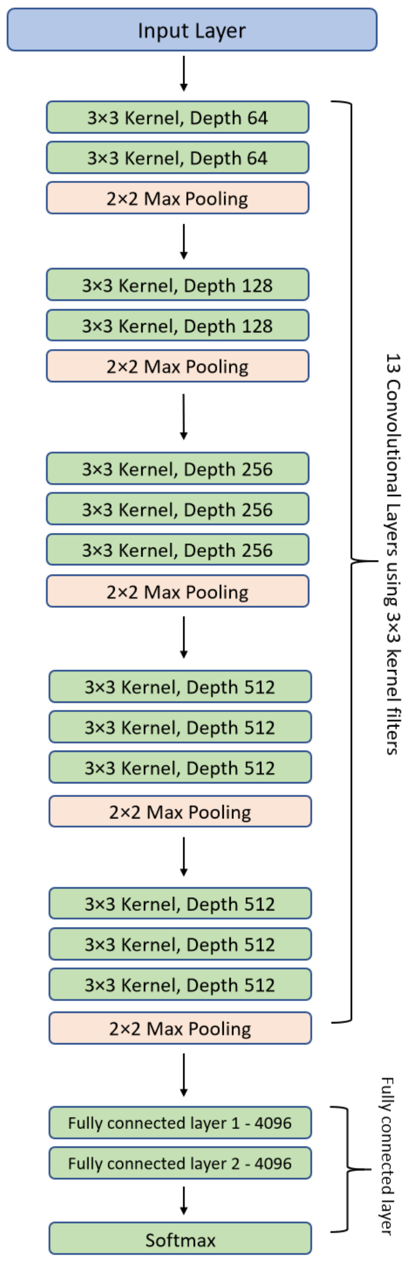


Fig. 7. VGG-16 model architecture – 13 convolutional layers and 2 Fully connected layers and 1 SoftMax classifier

VGG-16 - Karen Simonyan and Andrew Zisserman introduced VGG-16 architecture in 2014 in their paper Very Deep Convolutional Network for Large Scale Image Recognition. Karen and Andrew created a 16-layer network comprised of convolutional and fully connected layers. Using only 3×3 convolutional layers stacked on top of each other for simplicity.

The precise structure of the VGG-16 network shown in Figure. 7. is as follows:

- The first and second convolutional layers are comprised of 64 feature kernel filters and size of the filter is 3×3. As input image (RGB image with depth 3) passed into first and second convolutional layer, dimensions changes to 224x224x64. Then the resulting output is passed to max pooling layer with a stride of 2.
- The third and fourth convolutional layers are of 124 feature kernel filters and size of filter is 3×3. These two layers are followed by a max pooling layer with stride 2 and the resulting output will be reduced to 56x56x128.
- The fifth, sixth and seventh layers are convolutional layers with kernel size 3×3. All three use 256 feature maps. These layers are followed by a max pooling layer with stride 2.
- Eighth to thirteen are two sets of convolutional layers with kernel size 3×3. All these sets of convolutional layers have 512 kernel filters. These layers are followed by max pooling layer with stride of 1.
- Fourteen and fifteen layers are fully connected hidden layers of 4096 units followed by a softmax output layer (Sixteenth layer) of 1000 units.

VI. Leveraging Transfer Learning with Pretrained Models

ImageNet is a research project to develop a large database of images with annotations e.g. images and their labels. Pretrained models like InceptionV1, Inception V2, VGG-16 and VGG-19 are already trained on ImageNet which comprises of disparate categories of images. These models are built from scratch and trained by using high GPU's over millions of images consisting of thousands of image categories. As the model is trained on huge dataset, it has learned a good representation of low level features like spatial, edges, rotation, lighting, shapes and these features can be shared across to enable the knowledge transfer and act as a feature extractor for new images in different computer vision problems. These new images might be of completely different categories from the source dataset, but the pretrained model should still be able to extract relevant features from these images based on the principles of transfer learning. In this paper we will unleash the power of transfer learning by using pretrained model - VGG-16 as an effective feature extractor to classify dog vs cat even with fewer training images.

VII. Dataset

The training archive contains cumulatively 25000 images of dogs and cats. In this paper our objective is to build a robust image classification model with restrictions of consisting of few training samples of dogs and cats (binary classification problem). We need to downscale the number of images to add constraint on the input images. We took 5000 data images of cats and dogs out of 25000 images for training and 2000 images for validation. We leverage pretrained VGG-16 model as a feature extractor and transfer low-level features, such as edges, corners, rotation and learn new level features specific to the target problem which is to classify the images either as a dog or cat.

Fig. 8. train images of dogs and cats. Total training images = 5000, Total validation images = 2000

VIII. Analysis

As discussed earlier, first we will build a basic convolutional neural network from scratch, train it on training image dataset and evaluate the model. Later we will improve the accuracy using image augmentation technique. Finally, we will leverage the pretrained model VGG-16 which is already trained on a huge dataset with diverse range of categories to extract features and classify images.

All the evaluation metrics will be compared in later stage.

I. ***Basic Convolutional neural network*** – The architecture of CNN comprises of three convolutional layers with kernel size 3×3 and activation function as ReLU. This layer coupled with 2-Dimensional max pooling of size 2×2. The first three convolutional layers act as feature extraction. The output of the layer is fed to dense fully connected layer where the final prediction of either cat or dog is done.

Below are the model metrics after running basic CNN model – Model accuracy and loss

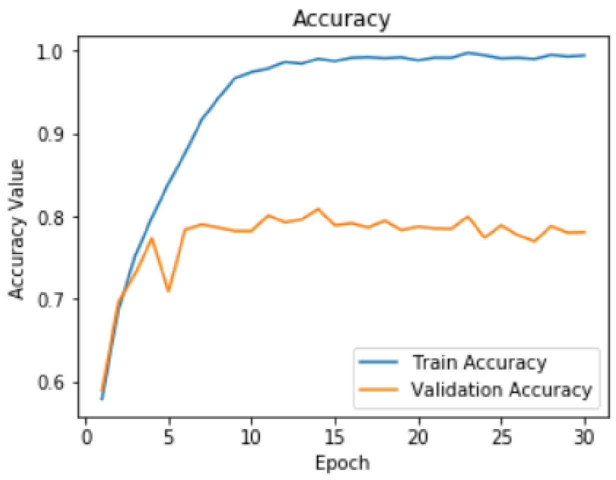


Fig. 9. Training and validation accuracy after using CNN

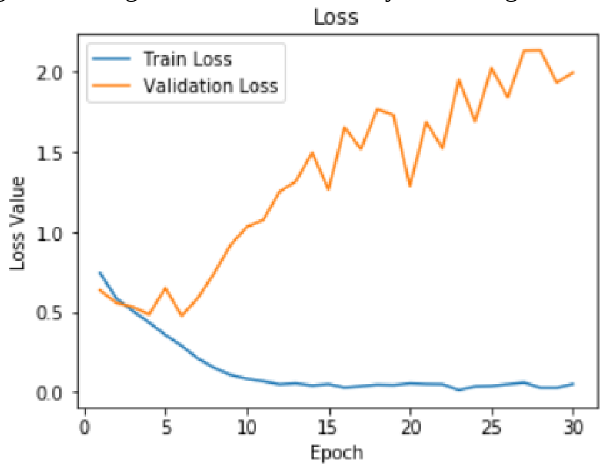


Fig. 10. Training and validation loss after using CNN

II. ***Convolutional neural network with image*** augmentation - Keras uses ImageDataGenerator() to quickly set-up python generators that automatically turn image files into preprocessed tensors that can be fed directly into models during training. Image transformation operations are applied on the training dataset such as rotation, translation and zooming to produce new versions of existing images. We input these new images into our model during training. It performs the following functions for us easily:

```
train_datagen = ImageDataGenerator( rescale=1./255,
                                    zoom_range=0.4,
                                    rotation_range=50,
                                    width_shift_range=0.3,
                                    height_shift_range=0.3,
                                    shear_range=0.2,
                                    horizontal_flip=True,
                                    fill_mode='nearest')

val_datagen = ImageDataGenerator(rescale=1./255)
```

- Decode the JPEG content to RGB grids of pixels.
- Convert these into floating-point tensors.
- Zooming the image randomly by a factor of 0.3 using the zoom_range parameter.

- ➢ Rotating the image randomly by 50 degrees using the rotation_range parameter.
- ➢ Translating the image randomly horizontally or vertically by a 0.2 factor of the image's width or height using the width_shift_range and the height_shift_range parameters.
- ➢ Applying shear-based transformations randomly using the shear_range parameter
- ➢ Rescale the pixel values (between 0 and 255) to the [0, 1] interval – It normalizes the image pixel values to have 0 mean and standard deviation of 1.

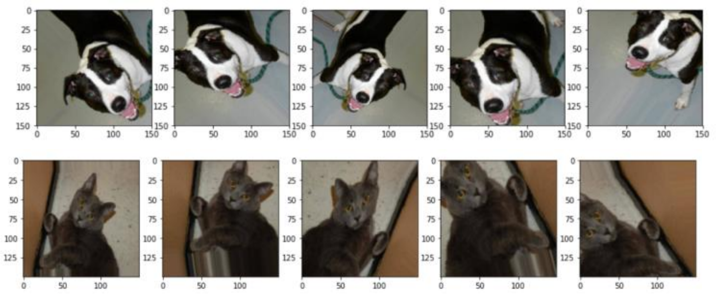


Fig. 11. Images are populated with zoom, rotation, height, width after applying ImageDataGenerator().

The model built by using convolutional neural network in I is trained on small dataset of images; therefore, it didn't predict well and ended up overfitting after 3 epochs (See Fig. 9 for reference). The idea of image augmentation is to add small variations without damaging the central object so that the neural network is more robust to these kinds of real world variations. Hence, by using image augmentation we populate these new images (fine tuned by using various parameters like zoom, rotation, width and height change) into our training dataset and train the model.

Below are the model metrics – Model accuracy and loss after running fine tuning CNN with image augmentation

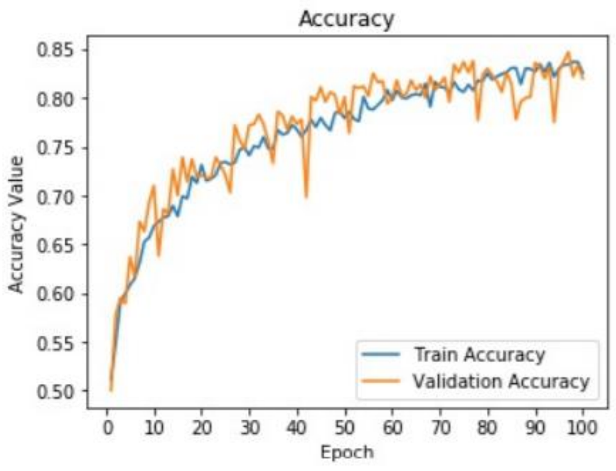


Fig. 11. Training and validation accuracy plot by applying CNN with image augmentation

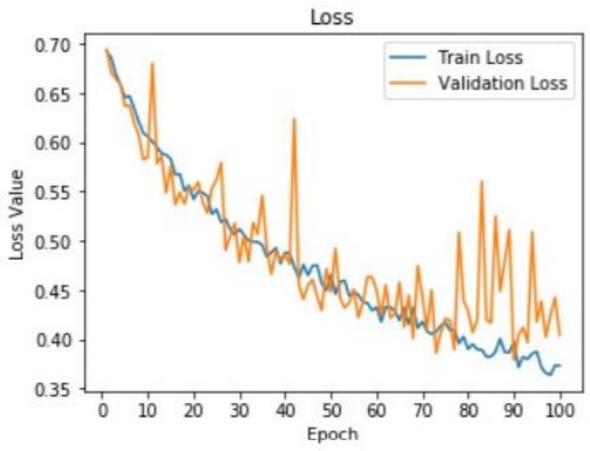
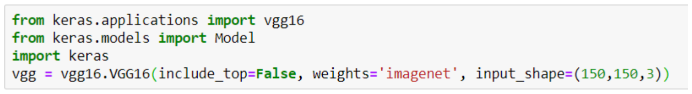


Fig. 12. Training and validation loss metric plot resulted by applying CNN with image augmentation.

III. ***Pretrained Convolutional neural network model as a feature extractor with image augmentation*** – Here we will implement VGG-16 pretrained model which is trained on the imagenet weights to extract features and feed this output to new classifier to classify images.



```
from keras.applications import vgg16
from keras.models import Model
import keras
vgg = vgg16.VGG16(include_top=False, weights='imagenet', input_shape=(150,150,3))
```

Above is the code to call VGG-16 pretrained model. We need to include weights = 'imagenet' to fetch VGG-16 model which is trained on the imagenet dataset. It is important to set include_top = False to avoid downloading the fully connected layers of the pretrained model. We need to add our own classifier as pretrained model classifier has more than 2 classes whereas our objective is to classify the image into 2 classes (dog or cat). After the convolutional layers of the pretrained model extract low level image features such as edges, lines and blobs, the fully connected layer then classify them into 2 categories.

Below is the block diagram of our model architecture

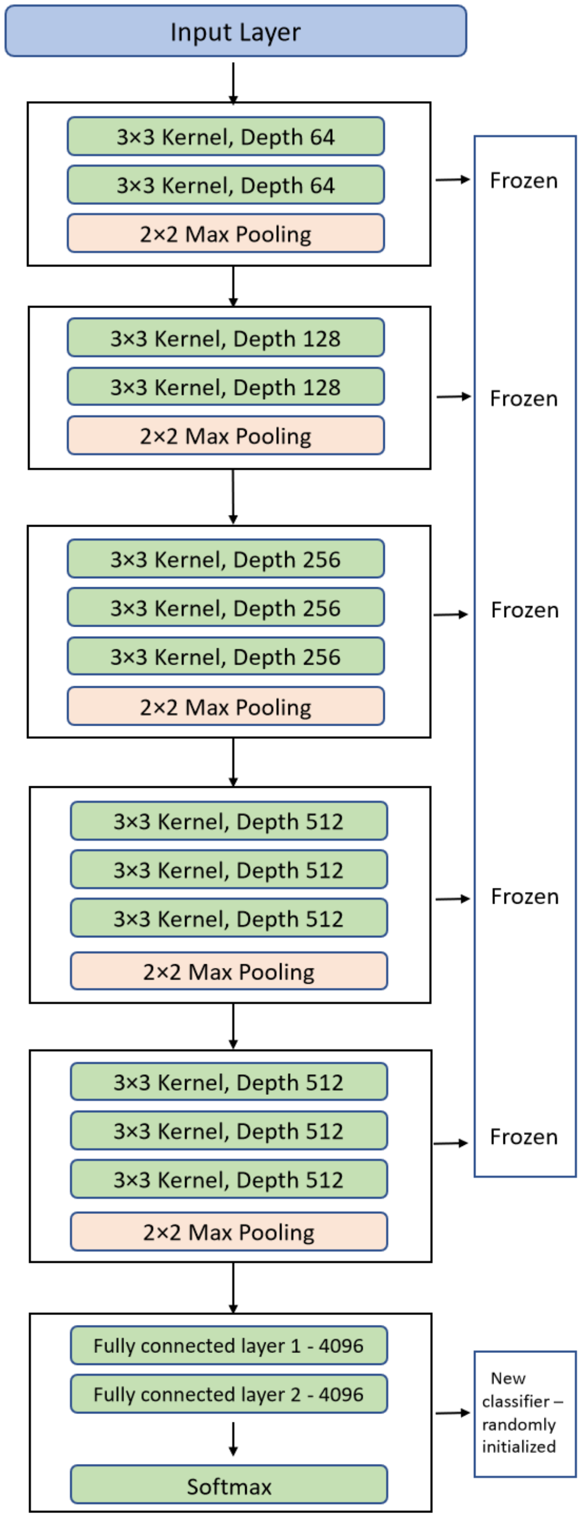


Fig. 13. Block diagram shows the architecture of VGG-16. To extract feature vectors from VGG-16 model, weights of all 5 convolutional blocks are frozen and resulted output is given to new classifier.

Below are the model metrics – Model accuracy and loss after fine tuning the CNN with pretrained model and image augmentation.

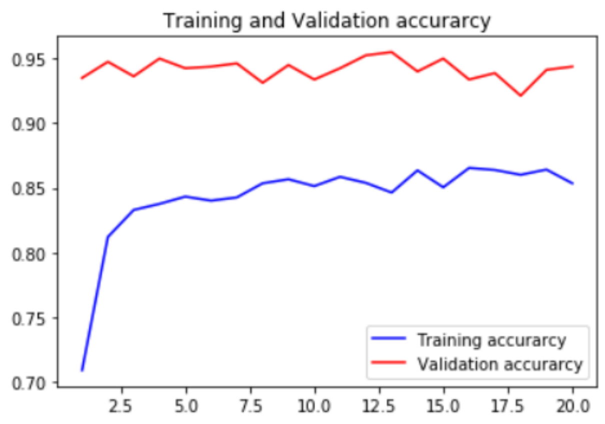


Fig. 14. Training and validation accuracy plot

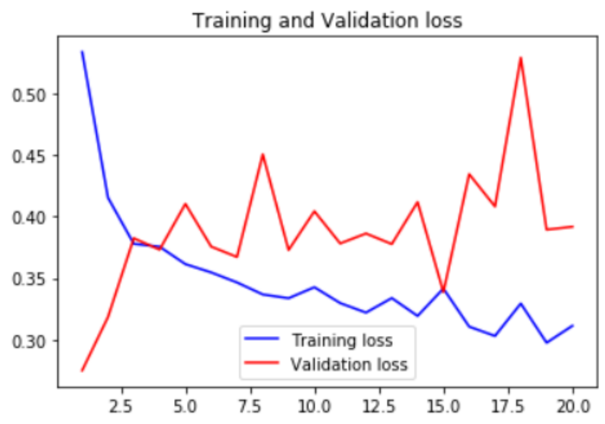


Fig. 15. Training and validation loss

IX. Comparative Results

	Training accuracy	Validation accuracy
Basic Convolutional neural network	98.20%	72.40%
Fine tuning CNN with image augmentation	81.30%	79.20%
Fine tuning CNN with pretrained VGG-16 model and image augmentation	86.50%	95.40%

Fig. 16. Above table shows training and validation accuracy for different neural network models. The first model build using convolutional neural network gives validation accuracy of **72.40 %.** We fine tuned this model with image augmentation and achieved accuracy of **79.20 %.** Lastly, we leverage the use of one of the pretrained models (VGG-16) trained on huge dataset of images and fine-tuned with image augmentation to achieve accuracy of **95.40 %**

REFERENCES

[1] Henry C. Ellis. The Transfer of Learning. The Macmillan Company, New York, 1965

[2] Pan, Sinno Jialin, and Qiang Yang. "A Survey on Transfer Learning."
IEEE Transactions on Knowledge and Data Engineering 22.10 (2010): 1345-359.

[3] Rich Caruana. Multitask learning. Machine Learning, 28(1):41–75, 1997.

[4] Ding, Zhengming, and Yun Fu. "Robust Transfer Metric Learning for Image Classification." IEEE Transactions on Image Processing (2016)

[5] https://www.hackerearth.com/practice/machine-learning/transfer-learning/transfer-learning-intro/tutorial/

[6] Oeslle Lucena, Amadeu Junior: Transfer learning using Convolutional neural networks for face-anti spoofing

[7] https://riptutorial.com/keras/example/32608/transfer-learning-using-keras-and-vgg

[8] https://www.learnopencv.com/keras-tutorial-transfer-learning-using-pre-trained-models/

[9] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

[10] Transfer learning – Lisa torrey and Jude Shavlik - http://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf

[11] https://www.researchgate.net/publication/303600638_A_survey_of_transfer_learning

[12] http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Lecture/transfer%20(v3).pdf

[13] https://www.sciencedirect.com/science/article/pii/S1389041718310933

[14] Jason Yosinski, Jeff Clune, Yoshua Bengio: “How transferable are features in deep neural networks?”, 2014, Advances in Neural Information Processing Systems 27, pages 3320-3328. Dec. 2014; [http://arxiv.org/abs/1411.1792 arXiv:1411.1792].

[15] M.M.H. Mahmud, S.R. Ray, "Transfer Learning Using Kolmogorov Complexity: Basic Theory and Empirical Evaluations", *Proc. 20th Ann. Conf. Neural Information Processing Systems*, pp. 985-992, 2008.

[16] D. Xing, W. Dai, G.-R. Xue, Y. Yu, "Bridged Refinement for Transfer Learning", *Proc. 11th European Conf. Principles and Practice of Knowledge Discovery in Databases*, pp. 324-335, 2007-Sept.

Srikanth Tammina graduated from Indian Institute of Technology, Hyderabad. His research interests include Neural networks, Text classification, Image classification and Natural language processing