




# IMAGE CLASSIFICATION USING CNN AND ITS VARIANTS

PAYAL BHATIA  
RANJAN SINHA  
PIYUSH PILARE  
SHANTANU SINGH  
BIKRAMADITYA GHOSH



# CAPSTONE PROJECT REPORT

PAYAL BHATIA

RANJAN SINHA

PIYUSH PILARE

SHANTANU SINGH

BIKRAMADITYA GHOSH

**Abstract-** There has been a tremendous increase in the applications of computer vision. The applications of computer vision range from smart cameras and video surveillance to robotics, transportation and more. In our project, we have implemented deep learning algorithms like Residual Neural Network and VGG16 to classify images as correctly as possible.

**Problem Summary-** The main goal of our project was to come up with a classification model which would be best able to differentiate between the six classes of images given in the dataset. The dataset provided to us had 25k+ images belonging to six different classes. The image type and class labels are given below for reference.

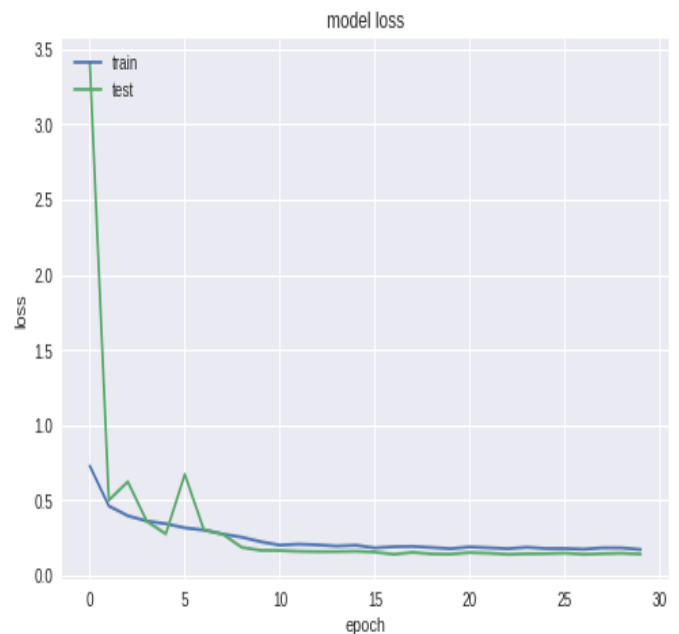
IMAGE TYPE	CLASS LABEL
Buildings	0
Forest	1
Glacier	2
Mountain	3
Sea	4
Street	5

The given dataset had 17034 images in training dataset and had 7301 images in test dataset.

**Line of thought-** Initially our main focus was to see how a simple convolutional neural network model would perform on the dataset. Following this we decided to tweak our baseline model by changing the activation function to see whether it would be an improvement on our previous model. The results of our model then prompted us to make use of transfer learning techniques which implement residual neural network model and VGG16 model.

**Interesting findings-** One of the interesting things we saw was that our Residual Neural Network based transfer learning classification model did not perform according to our expectations and this might have happened because while using transfer learning (via ResNet) we must ensure that the distribution of the training data which our pre-trained model has used must be like the data that we are going to face during test time or at least don't vary too much. Secondly, the number of training data for transfer learning involving the implementation of ResNet as source model should be in a way that we don't overfit the model.

The VGG-16 model performed the best among all the three models and we will be support our claims by showing the plot between loss and epoch for VGG-16 model.



We can very well see that the model minimizes loss following the completion of five epochs.

## I. INTRODUCTION

**Problem Definition-** The problem given to us involved the creation of a classification model which would be able to classify images belonging to six different classes as efficiently as possible. Our dataset had 25k+ images and this was divided into training and test datasets. The training dataset had 17034 images whereas the test dataset had 7301 images. The six classes along with their class labels are displayed in the table below for reference.

TYPE OF IMAGE	LABEL
Buildings	0
Forest	1
Glaciers	2
Mountain	3
Sea	4
Street	5

Our main aim was to come up with a classification model which would be able to classify images according to their correct classes as efficiently as possible and we chose 90% accuracy as our threshold value for a classification model to be retained for further use.

**Background work-** As part of our data preparation we are using the Image Data Generator class to generate batches of tensor image data with real-time data augmentation. The data will be looped over in batches. In this phase of data preparation we have taken 20% of our training data to be part of our validation data. The validation dataset helps us to assess whether a model is overfitting or underfitting in nature. The performance of our model on the validation data gives us an idea as to how well our model is performing. We have also scaled our data by a factor of  $1./255$  as the RGB content in our images take values ranging between 0-255 and at a given learning rate these values are too high for our model to process. This process helps the RGB coefficient values to be restricted between 0 and 1 thereby eliminating the previously mentioned problem which arose from using RGB coefficient values ranging between 0 and 255. The image is allowed to randomly rotate between 0-15 degrees. The value of shear range i.e. shearing transformation intensity is taken as 0.2. The width shift and height shift values are fixed at 0.1.

**Approach—** Firstly as part of our data preparation we are using the Image Data Generator class to generate batches of tensor image data with real-time data augmentation. The data will be looped over in batches. In this phase of data preparation we have taken 20% of our training data to be part of our validation data. The validation dataset helps us to assess whether a model is overfitting or underfitting in nature. The performance of our model on the validation data gives us an idea as to how well our model is performing. We have also rescaled our data by a factor of  $1./255$  as the RGB content in our images take values ranging between 0-255 and at a given learning rate these values are too high for our model to process. This process helps the RGB coefficient values to be restricted between 0 and 1 thereby eliminating the previously mentioned problem which arose from using RGB coefficient values ranging between 0 and 255. Secondly we tried applying a baseline convolutional neural network model on our dataset. Following this we decided to tweak our baseline model by changing the activation function from RELU (Rectified Linear Activation Unit) to Leaky RELU to see whether it would be an improvement on our previous baseline model. The results of our model then prompted us to make use of transfer learning techniques which implement residual neural network model and VGG16 model. The transfer learning technique which used Residual Neural Network was found to be inferior to the transfer learning technique which utilized VGG16 model. The weights used in the techniques involving transfer learning are called ‘imagenet’. During the compilation of both the above models we are taking the loss function to be based on categorical cross-entropy and we are assessing the performance of our model by using accuracy scores as our metrics. We also used Adaptive Moment Estimation to update the network weights iteratively in the training data instead of Stochastic Gradient Descent as it leads to Vanishing Gradient problem. Based on the performance of our various models, we saw that the model which implemented the transfer learning algorithm via VGG16 model proved to be the most efficient, giving us accuracy scores as high as 94%. Thus we decided to use our transfer learning enabled VGG16 model as the solution for the given problem statement.

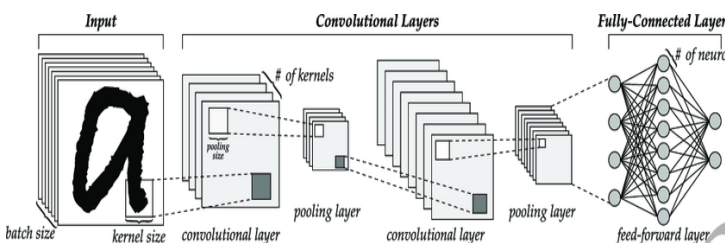
## II. DETAILS OF APPROACH

In our project we have mainly tried three different approaches with regards to our classification model. The first model is the simple convolution neural network based classification model. The second model is the transfer learning enabled Residual Neural Network model. The third and final model is also a transfer learning enabled VGG16 model. The workings of each model will be explained in this segment.

**Simple CNN Model-** A convolutional neural network (CNN) is a class of deep, artificial neural networks that found application mostly in image and sound analysis. CNN usually analyses a part or region of the sample which is called receptive field. What differentiates CNN from other neural networks is that for each receptive field the weights are shared among all neurons in filter of convolutional layer. In that way feature detection is independent from spatial position of the feature in the feature field. High level architecture of convolutional neural network is the same as for other neural networks i.e. input - multiple hidden layers – output. The two most commonly used architectures are **Linear stacking** and **Inception**. Linear stacking is a type of architecture where one layer is stacked on top of the other in series. The output of one layer is the input of the next layer. Inception is a method of stacking some layers in parallel and the output of all those layers is concatenated in the next layer. We have chosen the Linear stacking architecture in our project. In this model we are using **Adam** optimizer to update the weights in an iterative manner. Mathematically speaking It is denoted by a ‘\*’ sign as shown below:-

$$[f * g](t) = \int_0^t f(\tau)g(t - \tau)d\tau$$

The first function is the multidimensional array of inputs and the second function (called Kernel) is the multidimensional array of parameters to be adapted. Simple structure of CNN:-



The functionality of the various layers of the network are described below:-

### Convolutional-

This layer applies convolution operation to the input of the receptive field and passes the result to the next layer. Usually comprises of several filters that detect different features. First convolutional layer usually detects simple features. Next convolutional layer will detect more complex element and so on.

### Pooling-

This layer collects the output of a cluster of neurons and combines it into single neuron of the next layer. Pooling can use different aggregation functions (max, average, etc.) to choose the value that represents the cluster. Pooling layer reduces size of the spatial representation (therefore the number of parameters) but retains translational invariance. It helps to control overfitting.

### Dense-

This layer has properties of traditional perceptron neural network in which every neuron of one layer is connected to every neuron in the next layer. Dense layers are the last layers that perform the final classification.

### Activation-

This layer introduces non-linearity to the representation through resetting to zero negative input from the convolution layer. Among different activation functions Leaky RELU proved to be the most reliable.

### Dropout-

This layer helps to overcome overfitting through dropping out a random set of activations and setting them to zero. The network trained with dropout layer must develop more redundancy i.e. classify even if some neurons are deactivated. Dropout layer is only used in the training phase.

### Flatten-

This layer converts the three dimensional input into one dimensional feature vector. Flatten layer removes spatial information (not needed at this step) and passes output to the next dense layer where it can be processed for classification.

**Residual Neural Network-** As per what we have seen so far, increasing the depth should increase the accuracy of the network, as long as over-fitting is taken care of. But the problem with increased depth is that the signal required to change the weights, which arises from the end of the network by comparing ground-truth and prediction becomes very small at the earlier layers, because of increased depth. It essentially means that earlier layers are almost negligibly learned. This is called vanishing gradient. The second problem with training the deeper networks is, performing the optimization on huge parameter space and therefore naively adding the layers leading to higher training error. Residual networks allow training of such deep networks by constructing the network through modules called residual models as shown in the figure. This is called degradation problem. The intuition around why it works can be seen as follows:

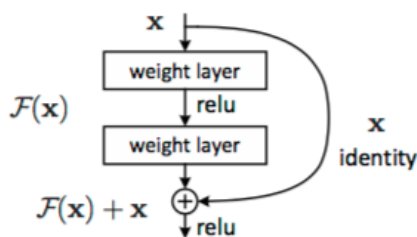


Figure 2. Residual learning: a building block.

Imagine a network, A which produces  $x$  amount of training error. Construct a network B by adding few layers on top of A and put parameter values in those layers in such a way that they do nothing to the outputs from A. Let's call the additional layer as C. This would mean the same  $x$  amount of training error for the new network. So while training network B, the training error should not be above the training error of A. And since it **DOES** happen, the only reason is that learning the identity mapping (doing nothing to inputs and just copying as it is) with the added layers-C is not a trivial problem, which the solver does not achieve. To solve this, the module shown above creates a direct path between the input and output to the module implying an identity mapping and the added layer-C just need to learn the features on top of already available input. Since C is learning only the residual, the whole module is called residual module.

So we are going to use this residual neural network as a base on top of which we will run our classification model. For efficient performance we are using Batch Normalization, a technique for improving the performance and stability of artificial neural networks. It is used to normalize the input layer by adjusting and scaling the activations. It can mitigate the problem of internal covariate shift, where parameter initialization and changes in the distribution of the inputs of each layer affects the learning rate of the network. It uses a global average pooling followed by the classification layer. We will be using the pre-trained model approach in transfer learning. This approach comprises of the following steps:-

**Select Source Model.** A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.

**Reuse Model.** The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modelling technique used.

**Tune:** Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

In our case, the source model is the ResNet and it will be reused as the starting point for our multi class classification convolutional neural network model which makes use of Early Stoppage algorithm and ReduceLROnPlateau algorithm. The former is a type of regularization technique which is used to avoid overfitting and the latter reduces the learning rate once it stagnates after reaching a certain level.

However some of the problems that we faced while using transfer learning (via ResNet) was that the distribution of the training data which our pre-trained model has used must be like the data that we are going to face during test time or at least don't vary too much. Second, the number of training data for transfer learning involving the implementation of ResNet as source model should be in a way that we don't overfit the model.

**VGG16-** For our third approach we will be implementing the pre-trained model approach in transfer learning. The only thing different in this approach is that we are using VGG16 model as our source model instead of ResNet.

VGG network is characterized by its simplicity, using only 3x3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier.

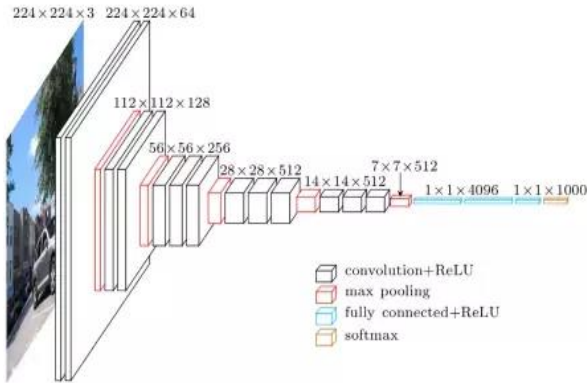


Fig 3: VGG-16 block diagram

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig 4:VGG-16 architecture visualization

VGG Net consists of 16 convolutional layers and is very appealing because of its very uniform architecture. It only performs 3x3\(\times\)33x3 convolutions and 2x22\(\times\)22x2 pooling all the way through.

As mentioned before, models for image classification that result from a transfer learning approach based on pre-trained convolutional neural networks are usually composed of two parts:

Convolutional base, which performs feature extraction.

Classifier, which classifies the input image based on the features extracted by the convolutional base.

In our transfer learning approach we have used the VGG-16 model as our convolutional base and then we have used a linearly stacked convolutional network to classify the input images as correctly as possible and this classification is done on the basis of the features extracted by the convolutional base, in this case, the VGG-16 model.

Now, we focus on the classifier part, we must start by saying that different approaches can be followed to build the classifier. Some of the most popular are:

**Fully-connected layers.** For image classification problems, the standard approach is to use a stack of fully-connected layers followed by a softmax activated layer. The softmax layer outputs the probability distribution over each possible class label and then we just need to classify the image according to the most probable class.

**Global average pooling.** A different approach, based on global average pooling, is proposed by Lin et al. (2013). In this approach, instead of adding fully connected layers on top of the convolutional base, we add a global average pooling layer and feed its output directly into the softmax activated layer.

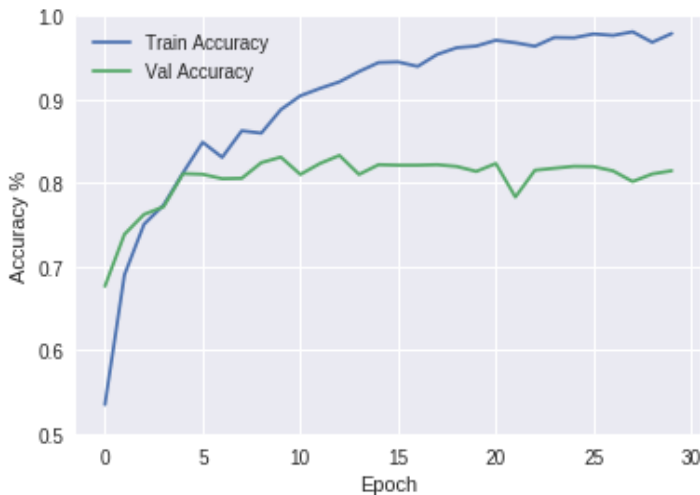
**Linear support vector machines.** Linear support vector machines (SVM) is another possible approach. According to Tang (2013), we can improve classification accuracy by training a linear SVM classifier on the features extracted by the convolutional base. Further details about the advantages and disadvantages of the SVM approach.

Amongst the three approaches we chose to go with the fully connected layers method of building the classifier.



### III. RESULTS

**Simple CNN-** After running simple convolutional neural network model as our classification model, we observe the following:-



From the above figure(Fig 5) we see that as the number of epochs increase our training accuracy also increases. The validation accuracy increases in the first 5 epochs and after the first 5 epochs, the validation accuracy does not vary significantly with the completion of further epochs.

The training and test accuracies observed are as follows:-

training acc.: 0.9828588716393006

test acc.: 0.8263412646441365

The average of the best 5 training accuracy scores and validation scores are given below:-

training top 5: 0.9828588716393006

val top 5: 0.8263412646441365

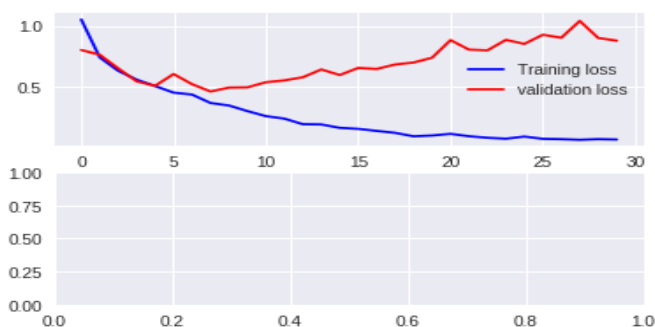


Fig 6: Training loss vs Validation loss

From the given figure we see that after 5 epochs the validation loss increases drastically compared to the training loss and this is a sign that our model is underfitting.

**ResNet-** After running transfer learning with ResNet as the source model, we observe the following:-

The learning rate stagnated after the fourteenth and nineteenth epochs respectively and they are denoted below:-

Epoch 14/30

- 96s - loss: 0.5474 - acc: 0.8072 - val\_loss: 0.9070 - val\_acc: 0.8518

Epoch 00014: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.

Similarly post the 19<sup>th</sup> epoch we see the same thing.

Epoch 19/30

- 94s - loss: 0.4663 - acc: 0.8290 - val\_loss: 0.8223 - val\_acc: 0.8572

Epoch 00019: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.

After the 30<sup>th</sup> epoch we found the following result:-

Epoch 30/30

- 96s - loss: 0.4607 - acc: 0.8308 - val\_loss: 0.7627 - val\_acc: 0.8641

By using ResNet in our transfer learning model, we get roughly an accuracy of 85%. This can be improved further by using VGG-16 as our source model in the transfer learning approach.

**VGG-16-** We have also used VGG-16 as our source model in transfer learning model and the results obtained were a significant improvement from the previous model which utilized ResNet as source model. The results are described in the following page.

It has been observed that the learning rate stagnates for the VGG-16 model after the 8<sup>th</sup>, 16<sup>th</sup> and 20<sup>th</sup> epochs respectively.

Epoch 8/30

- 96s - loss: 0.2777 - acc: 0.9019 - val\_loss: 0.2766 - val\_acc: 0.9107

Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.000100000000474974513.

Epoch 16/30

- 96s - loss: 0.1850 - acc: 0.9342 - val\_loss: 0.1579 - val\_acc: 0.9417

Epoch 00016: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.

Epoch 20/30

- 96s - loss: 0.1804 - acc: 0.9364 - val\_loss: 0.1441 - val\_acc: 0.9517

Epoch 00020: ReduceLROnPlateau reducing learning rate to 1e-05.

Now we will take a look at the accuracy and loss curves. The model accuracy for the VGG-16 model is given below:-



Fig 7: Accuracy vs epoch for train and test data

From the above graph it can be easily seen that as the number of epochs increase the model accuracy for both test and train remain the same barring a few fluctuations for the test data between zero to five epochs.

Similarly we try to plot the loss vs epoch curve to see how well our model is performing for both training and test data. The plot is given below:-

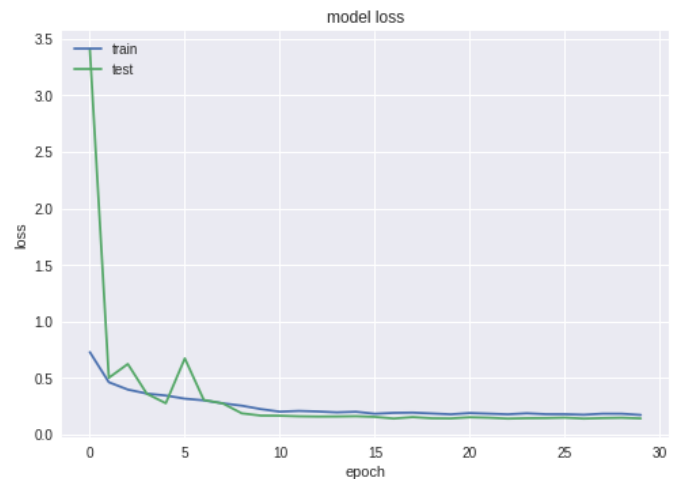


Fig 8: Loss vs Epoch for train and test data

From the above curve we can see that at the beginning there was a huge loss for test case as compared to train case. Upon reviewing the plot we see that the losses are virtually the same after 6 epochs. Prior to the occurrence of 6 epochs, we see that the loss for the test case fluctuates a little bit more as compared to the train case.

After the completion of 30 epochs we got the following result:-

Epoch 30/30

- 96s - loss: 0.1764 - acc: 0.9363 - val\_loss: 0.1442 - val\_acc: 0.9497.

This model is roughly giving us an accuracy of 94%

## IV. SUMMARY AND CONCLUSIONS

In this study an image classifier was developed using three different approaches.

The three approaches are as follows:-

CNN

RESNET

VGG-16



Using only CNN for image classification gave us the lowest accuracy among all the models and this model was also prone to overfitting as validation loss was greater than training loss in this case.

Using ResNet based transfer learning approach, we achieved better results than the previous model but this model also had some flaws. Some of the problems that we faced while using transfer learning (via ResNet) was that the distribution of the training data which our pre-trained model has used must be like the data that we are going to face during test time or at least don't vary too much. Second, the number of training data for transfer learning involving the implementation of ResNet as source model should be in a way that we don't overfit the model.

Finally we applied the VGG-16 based transfer learning algorithm for our classification model and this model turned out to be satisfactory in its performance as it had an accuracy score of 94% which was greater than our accepted threshold value of 90%.

Thus, amongst the three different modelling approaches, we can say hands down that our VGG-16 based transfer learning classification model will perform the best.

## V. CONTRIBUTIONS

The names and contributions of our team-members are listed below:-

Bikramaditya Ghosh- Responsible for making the report, compiling the slides and making simple CNN model using Leaky Relu as activation function.

Payal Bhatia-Responsible for data visualization part in our project and prepared extensively on SOM and CNN models with different activation functions.

Piyush Pilare-Responsible for ResNet and VGG-16 codes along with our presentation.

Ranjan Sinha-Responsible for our power point presentation and did extensive research on how CNN could be used for image classification.

Shantanu Singh-Responsible for ResNet and VGG-16 based transfer learning models.

## VI. REFERENCES

<https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751>

<https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/>

[https://brohrer.github.io/how\\_convolutional\\_neural\\_networks\\_work.html](https://brohrer.github.io/how_convolutional_neural_networks_work.html)

<https://github.com/czapol/Neural-Network-Classification-of-White-Blood-Cell-Images/blob/master/Neural%20Network%20Classification%20of%20White%20Blood%20Cell%20Images%20-%20Capstone%20Project%2020-%20Report%20-%20Springboard.pdf>