

Cat VS Dog Classifier

Image Classification

Vishal Verma | Data Science

# Problem Statement

An image processing company is trying to build a model for classifying the pets (dogs and cats). As the artificial intelligence evolving day by day we asked for build a deep learning model for classifying the given image whether cat or dog.

For this we need to build two models as mentioned below:

1. Build a CNN model for Dog VS Cat classification from scratch

2. Use existing CNN model and Transfer learn for Cat vs Dog classification

3. Build an autoencoder model for feature compression and use the features for matching

## Datatset Details

Dataset consists of images of cats and dogs. The dataset is provided in two parts there is Training set which consists of 20000 images and the training set which has 5000 images.

The dataset is normally distributed I.E both the classes dogs and cats have equal number of images 10000 each

The images in dataset are normal I.E any sort of augmentation has not been done on the images.

## Softwares used

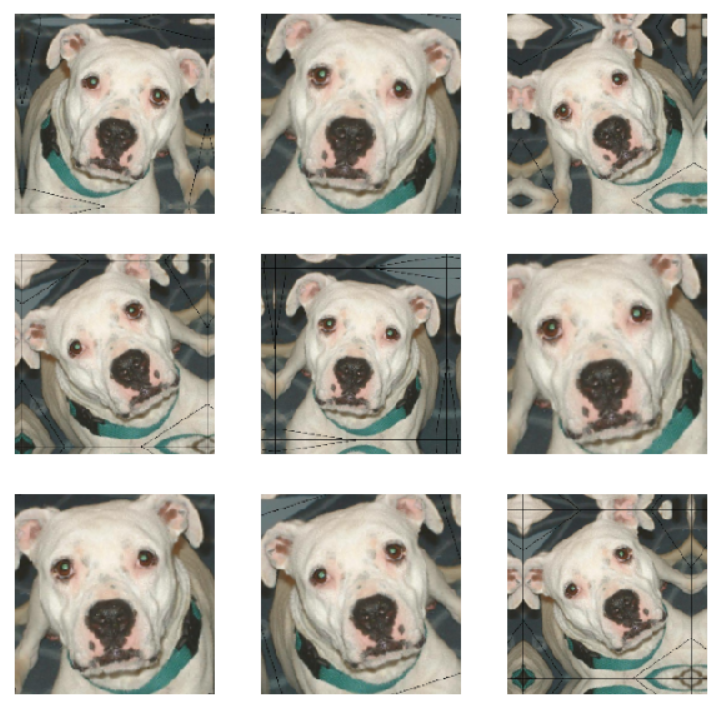
* R studio
* Jupyter Notebook
* Python
* R

## Data-Preprocessing Steps

For the images in the dataset I first trained the models without any preprocessing which resulted in over fitting of the model. Which means it was performing really well for the training set but not for the test set.

There was a huge gap between the accuracies of both the sets so we applied some preprocessing to the images.

I used image augmentation on our images while importing the data from the directory instead of applying the augmentation during the training of the model as while using that approach the model was taking a lot of time for training as while taking the input it was also applying the augmentation so for that reason I used data augmentation while importing the data from the directory.



As shown in above image these augmentation were applied to the images randomly so that there is no overfitting in the data.

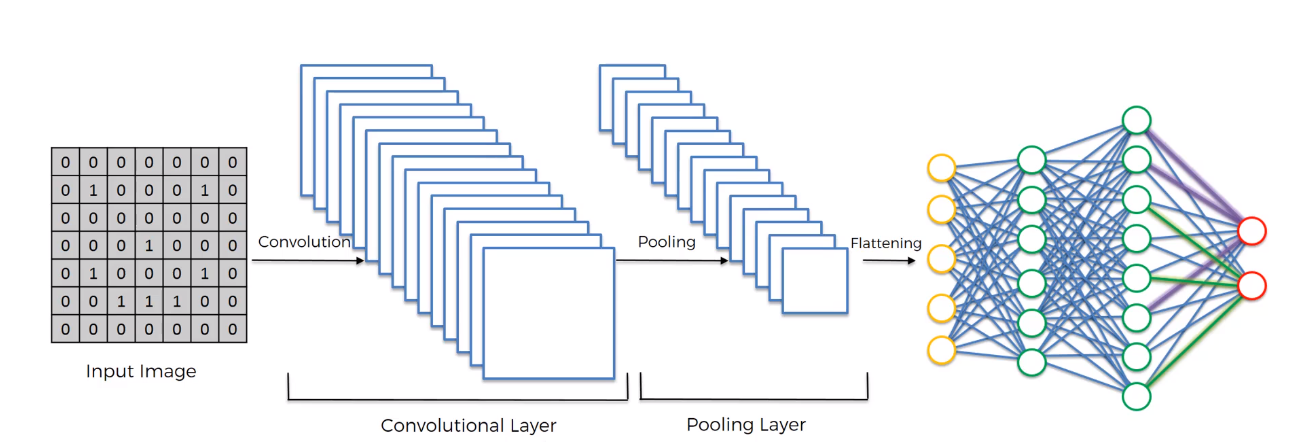
The Augmentation that have been applied are listed below:

* Horizontal Flip
* Sheer Range
* Zoom Range
* Scaling

## CNN Model Explanation

CNN is Convolution Neural Network, they are used for any type of image classification problems

In CNN, it takes the image in the form of matrix where each pixel in the image has some numeric value and for colored images it have 3 channels of RGB whereas for black and white images there is only 1 channel.



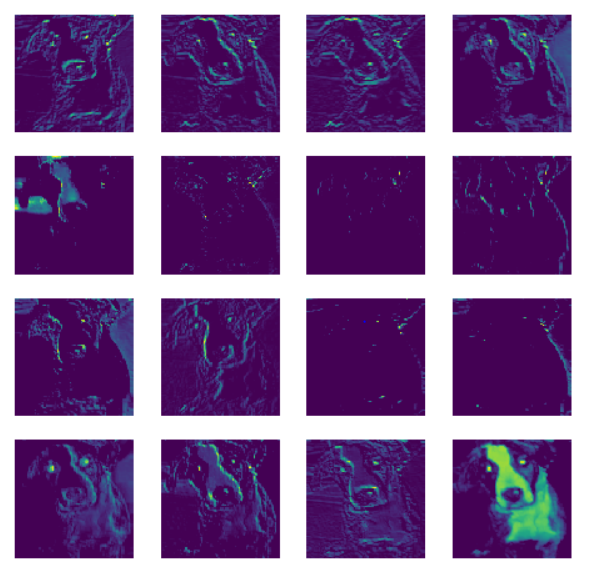
CNN take the Image as a matrix with different channels for RGB and a filter is applied to the image, those filters are used to identify features.

There are multiple filters which are used and only the filters which give a lot of important features are used, some filters such as

* Sharpen
* Blur
* Edge detect
* Edge enhance and many more

The First Layer for our Convolution Neural Network(CNN) is Convolution Layer

* In Convolution layer we perform a convolution operation on the image.
* We get the image as a matrix with 3 channels then multiple filters taken by the layer are multiplied with the image matrix.
* Once this operation is done it generates a Feature Map for the model, which have important features of the image.
* When this operation takes place on multiple channels with multiple filters generates a number of feature maps which are collectively known as convolution layer.



1This is the output from our convolution layer

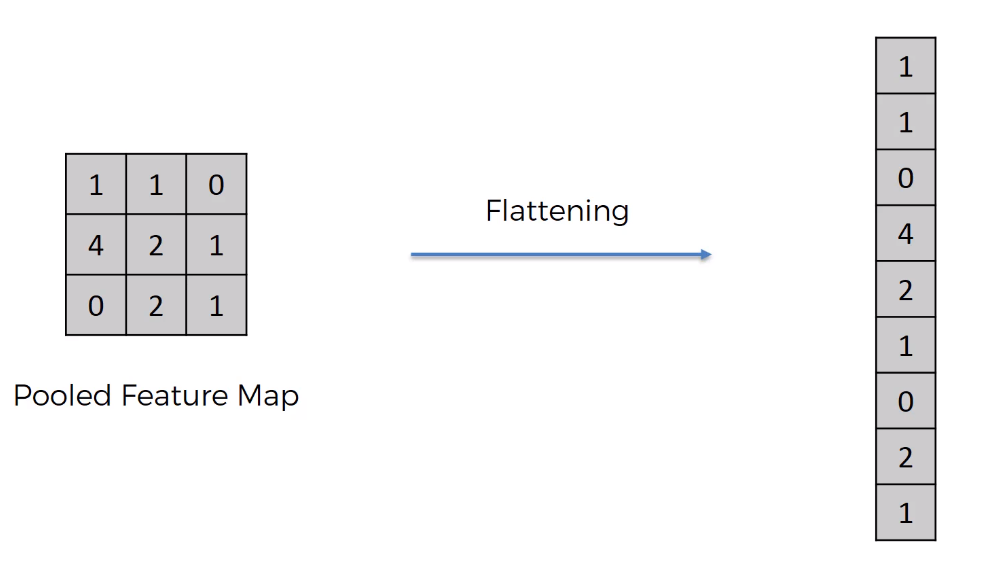
Once we get this output it goes on to the next layer of Max Pooling

* In max pooling layer we try to compress the feature maps generated in the previous layer so that only important features are used for the image classification
* In max pooling layer we select a small matrix or filter which moves over the feature map to select the max value out of all the pixels in that particular filter
* Once that is done we get a compressed size of image with more important features.

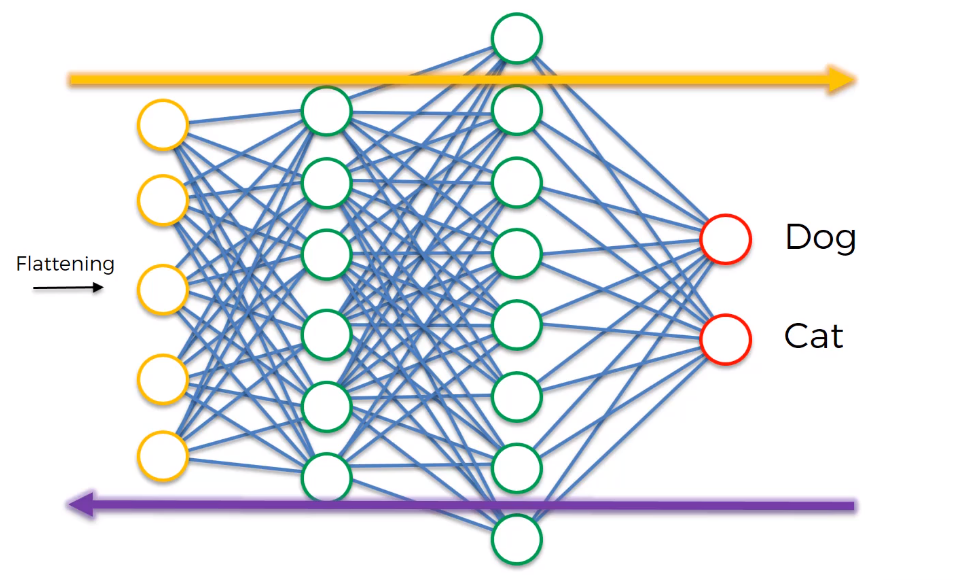


Above Picture is an example of how multiple filters are used in the convolution layer and then how max pooling is used to compress the features

The Next layer that is used is Flatten layer

* Flatten layer is used to flatten the output of the max pooling layer so that it can be further feeded to the Fully connected neural networks
* The fully connected neural networks won’t be able to handle matrix that’s why flatten layer is used to create an array from those matrix as shown below:
* 

After the flatten layer any no. of fully connected layers can be used which helps you to classify whether the given image is a dog or cat.



Similar to as shown in the image multiple fully connected layers are used which will go through a series of forward and back propagation to get the best result.

The last layer output layer:

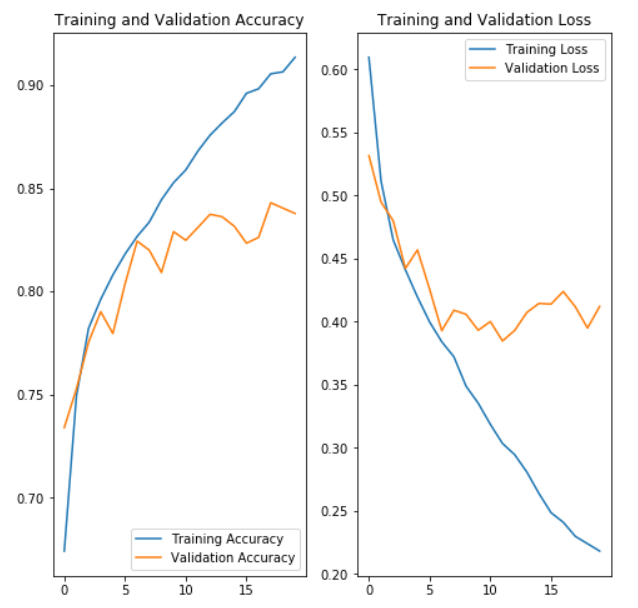
* We have used a output layer with only 1 neuron as the output with a sigmoid activation function
* Sigmoid activation function gives the value of 0 if the value is in between 0 – 0.5 and 1 as the output if the value is above o.5
* As this was enough to classify the image we have used only one neuron in the output layer

### CNN Model From scratch

We created a CNN model from scratch

It consisted of 2 sets of Convolution and max pooling layer followed by a flatten layer and then two fully connected layers out of which the last one was output layer with only one neuron

We trained the model on the dataset and validation data as the test set for 20 epochs and were able to achieve the final accuracy of 83%



These were the results that we obtained when creating a CNN model from scratch

## Transfer Learning Procedure

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

In transfer learning we take a pre-trained model which is trained on a particular task for a lot of time and provide a greater accuracy. So as it is trained in some different data it has a different output layer

So what we do is while we import this pre trained model from the library we don’t include the output layer which is used to identify 1000’s of object.

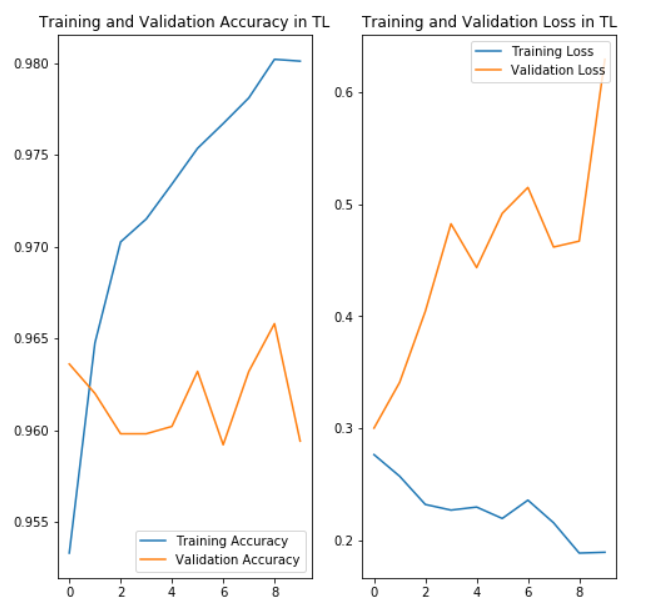
And we add our own output layer with number of neurons as required by us.

### Building a model by using Transfer learning

For building a model for transfer learning we have used MobileNetV2 model which is one of the pre-trained image net model used for classification of 1000’s of objects.

Steps included in building a model with transfer learning

* First we imported the MobileNetV2 model from the keras library without the output layer as we only need to classify between dogs and cats.
* Then we freeze the weights and filters of MobileNetV2 model as we don’t want to train the model again as it is already trained.
* Now we add one flatten layer which will flatten the output of MobileNetV2 model
* And add one fully connected output layer with only one neuron with sigmoid as the activation function
* Now we create a model with MobileNetV2 layers as input and our layers as the output
* Then we just import the data and train the model for the data set.

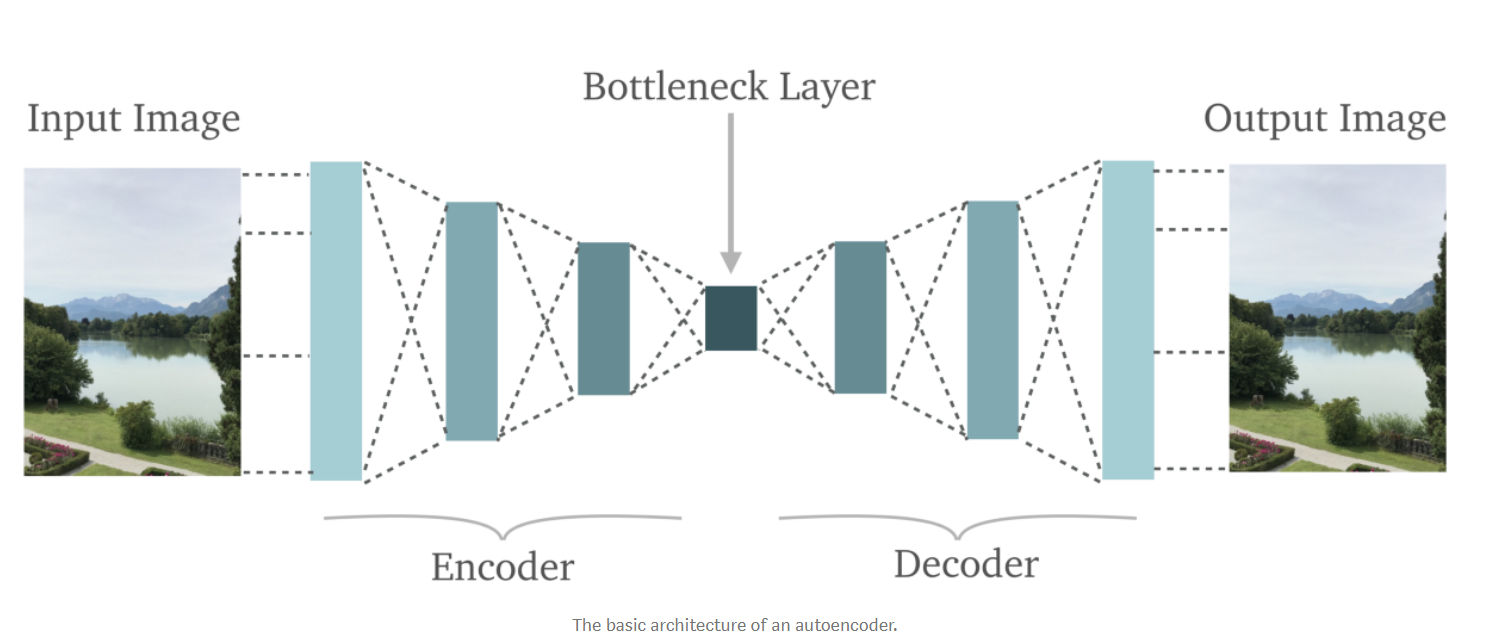


We trained the model on the dataset and validation data as the test set for 10 epochs and were able to achieve the final accuracy of 96%

## Build an autoencoder using CNNs

Auto encoder are the Neural network which are used to train the model in such a way that they try to reproduce the image that is given as the input while compressing it and then again extracting the information.

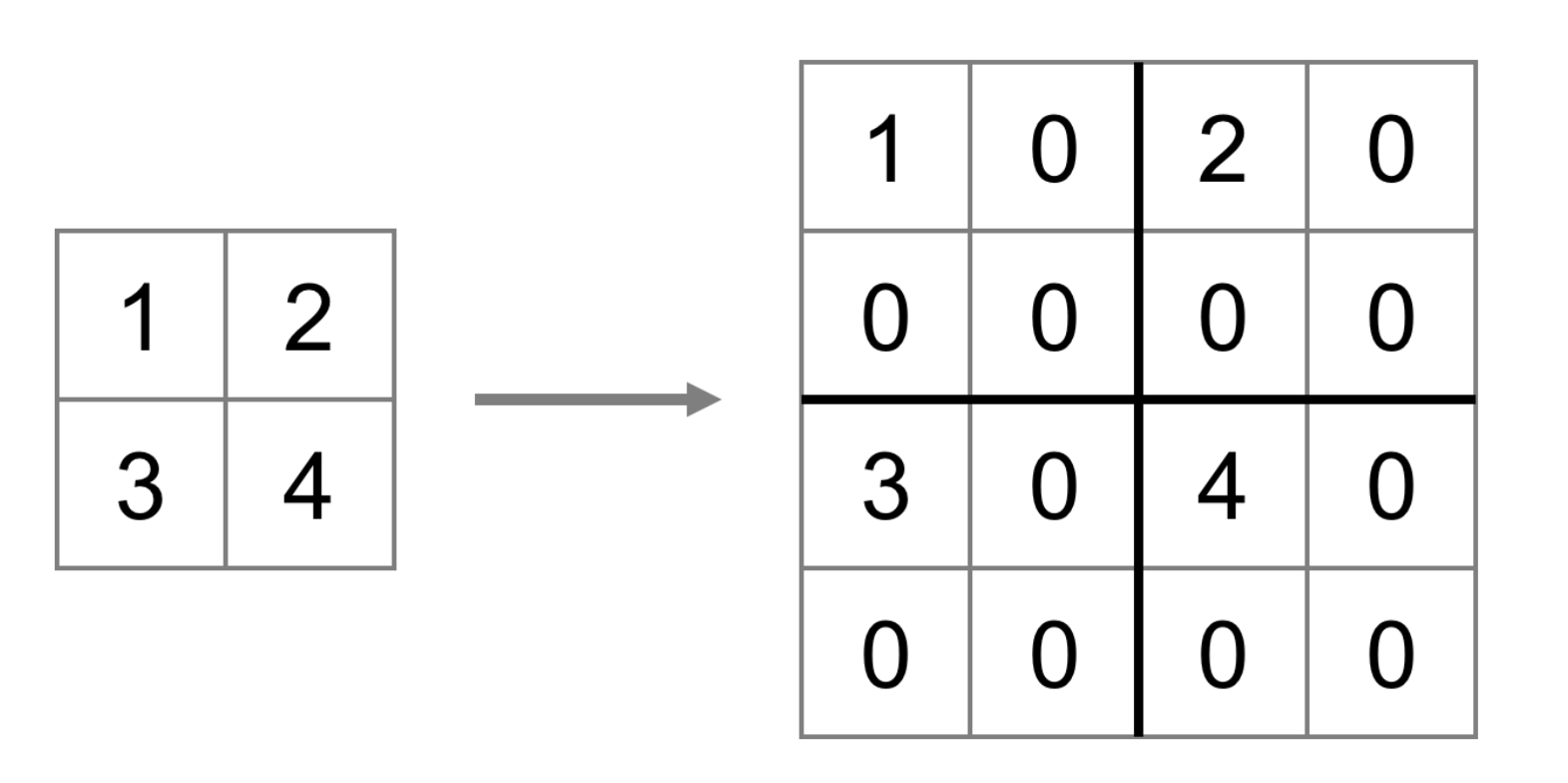
While doing so when the neural network try to produce the same result the features that are important and filters that are giving us the valuable information are kept.



For Building our auto encoder we used convolution layer, max pooling layer and up sampling layer.

As Convolution layer and max pooling are already explained we’ll just go to Up-Sampling layer

In Up sampling layer we do just the opposite of max pooling or down sampling. Where we take up a matrix or filter from the image and create an image of greater size using those pixel values which are surrounded by 0 value in the layer as shown below



We built the auto encoder using those layers and then trained the model on the training set and tried to reproduce the result of those images as shown below:



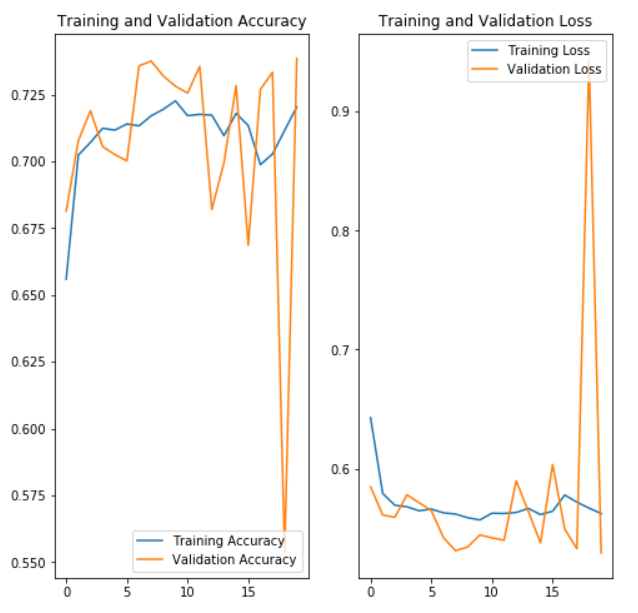
No we will move forward to use the Encoder layers to use as feature and predict the images

### Use encoder output as features and build a classifier

Steps involved in creating the auto encoder classifier

* First we created a model which had same layers as the encoder part of our auto encoder and a flatten layer and fully connected layer.
* Then we copied the weights of our autoencoder model to the newly created model.
* After the weights were copied we just set those layers of our model to trainable as false. As we didn’t want to change the features that our encoder part is creating
* Then simply we train our model on training data set and validation inputs.

Using the Auto encoder classifier we got multiple set of filters and features which were used to classify the images.



We trained the model on the dataset and validation data as the test set for 20 epochs and were able to achieve the final accuracy of 74%

## comparison of three concepts

Comparing the three concepts I would divide the comparison on three parts:

1. **Time Taken to Train the Model:**

The time taken for the **CNN** model was pretty high as it was being trained from scratch so it took a lot of time to get trained and also provided a decent accuracy

For **Transfer** **learning** it was pretty fast to get trained as the trainable parameters for the model were pretty less and provided a good accuracy as it was pre-trained

For **Auto** **Encoder** based model was also decent fast in training the model but the accuracy provided by it was not that good maybe if we would have trained for more number of epochs could have reached good number.

1. **Accuracy Provided by the Model**

As we have already mentioned the **CNN** model gave us an accuracy of **83**%, **Transfer** **learning** model an Accuracy of **96%** and the **auto** **encoder** model an accuracy of **74%**

So based on the accuracy provided transfer learning model stands at the top.

1. **Areas of Improvement**

Now talking about where these model can be improved if we look closely the **accuracy VS Validation accuracy** graphs for the all the model it is evident in the **CNN** model and **Transfer** **learning** model the gap between both the accuracy is increasing which states that if train them more it could lead to over fitting well in the case of **transfer** **learning** it is already 96% accurate so we don’t need to train it but **CNN** model would lead to over fitting if trained for more no. of epochs.

No talking about **Auto** **encoder** model both the accuracy and validation accuracy are showing an upward trend so it is evident that if we train it for more number of epochs in both the cases I.E auto encoder part as well as the full model part it can give us better results.

## summary

After working on all the three concepts of image classification. I got to understand how different layers in convolution neural network works what all models can be used to do image classification and how data augmentation plays an important role to reduce over fitting of the models.

# Instructions to run the code

## Python

* Open Jupyter notebook in the directory where you have stored the solution file
* Change the working directory to get the data from the dataset provided
* Run the cells one by one using shift + enter.

## R

* Open R studio
* Open the R solution file
* Change the working directory to get the data from the dataset provided
* Run the commands in R studio one by one using ctrl+enter