**Assignment 3**

**Name: Aloukik Aditya Student\_ID: 1115290**

**Question 1:** Load the dataset for all three classes and resize each image to (32 x 32). Apply the required preprocessing steps to employ the data into Machine Learning / Deep Learning algorithms.

**Solution:**

According to question I have successfully loaded all the images into single dataset. Basically, I have created a list i.e. data and labels. Data contains all three images which are abdomen, chest and head. Total images in dataset are 1500 (500 for each class). List of labels contain labels of each class which are 0,1,2. The size of the image has been set to 32 by 32. Glob library helped in reading images. The following code will explain everything.



Each for loop will attain images of each class, label them and then merge into database. Test\_train\_split function will split data randomly into training and testing dataset.

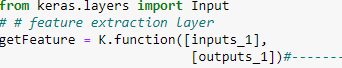
**Question 2:** Construct a Convolutional Neural Network (CNN) architecture from scratch to extract features from the images. (HINT: Extract features for train and test set separately. Extract the features constructed by the convolutional layers from an intermediate dense layer. Please refrain from using any pre-trained model for implementing this step)

**Solution:**

I have created 3 feature layer, 2 dense layer or fully connected layer. Last layer won’t be used in our model because that layer is for classification. Only first four layers will be used to extract the features from images. Keras sequential model is used in the image below.



We have to run the model, so that we can extract features from the images. So, the model has been compiled. I have set epoch to 5, the performance can be increased by increasing the epochs. After getting features from CNN, I will feed the image data into RF and KNN.



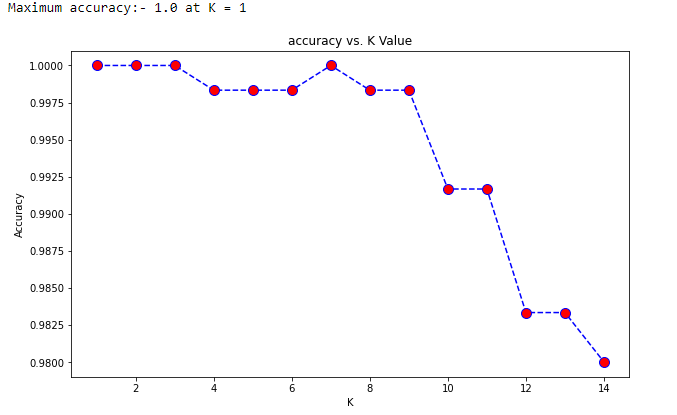
**Question 3** Apply the K-Nearest Neighbor (KNN) algorithm to the extracted features from CNN and find the optimal value of K. The value of K can be considered as [3, 5, 7, 9]. Determine the performance of the model using an appropriate performance metric. Draw a graph of K values and their corresponding performance in order to represent your results.

**Solution:**

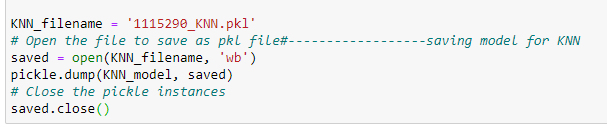
I have used K nearest sklearn libraries to implement this step. As you can see in the above image. Knn\_model is running into the for loop. And optimal value of k is recorded afterwards.



List of accuracy was created, that records all the accuracy and value of K can be traced by the index of the list. In the above image the max accuracy is on 1, 2, 3. The for loop contains all values which were given (1,3,5,7).



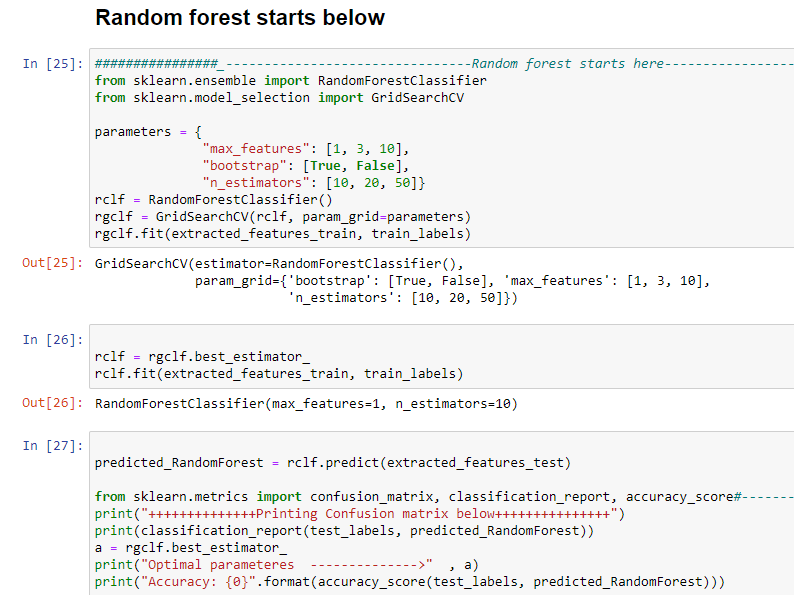
Also, models are saved using pickle libraries. And loaded in the ending of the code. Code for pickle saving is shown below



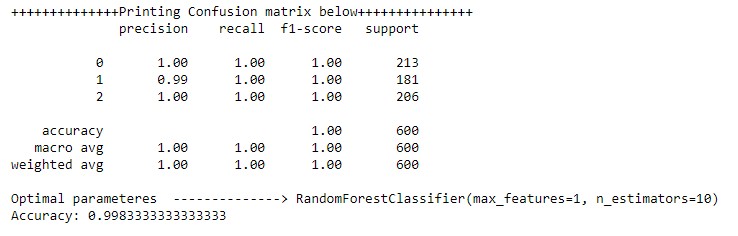
**Question 4:** Apply Random Forest (RF) algorithm to the extracted features from CNN. Tune at least two hyperparameters using random search. Determine the model's optimal performance, the confusion matrix, and the value of hyperparameters producing the optimal performance.

**Solution:**

I have set 3 hyperparameters for random forest which are max\_features, n\_estimator, bootstrap. This section will also tell which parameter performed best. Sklearn libraries are used to perform this step. Gridseachcv is used to find optimized parameters. Basically it will test out all the parameters and selects the best among all.



**CONFUSION MATRIX**



A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

Also, models are saved using pickle libraries. And loaded in the ending of the code. Code for pickle saving is shown below



**Question 5:** Report the performance of each model and explain your results. (eg. overfitting, underfitting, etc.)

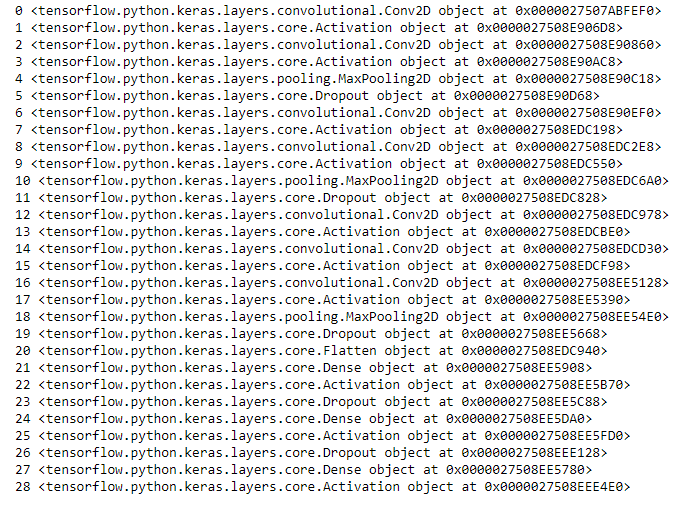
**Solution:**

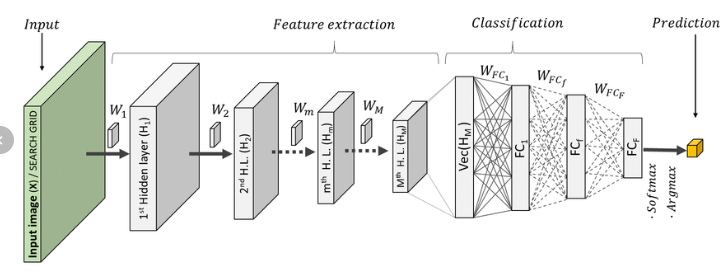
Lastly, the performance of both models was quite same (100 percent). The performance was good because the dataset was small, and images were grayscale. Also, the structure of images was same. Feature extraction played very important role, it highly boosted the models performance.

In our CNN model, there was overfitting when epochs were around 100, basically Overfitting is the case where the overall cost is really small, but the generalization of the model is unreliable. This is due to the model learning “too much” from the training data set.

There was no case of underfitting in our model. Basically, Underfitting is the case where the model has “not learned enough” from the training data, resulting in low generalization and unreliable predictions.

Feature extraction is a really good way to improve model’s performance. These are all the layers of our model. And values from intermediate has been extracted and fed into other models.





Basically, Once the deep learning models are trained, they are used to extract image features from image patches and output values of the max pooling layer in architecture A or the third convolutional layer in architecture B are used as image features, referred to as CNN features,

Comparing KNN and random forest. The algorithms are completely different. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

The random forest is a classification algorithm consisting of many decision trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

[Reference]

[1] Jyostna Devi Bodapati and N. Veeranjaneyulu, “Performance of different Classifiers in non-linear subspace” Proceedings of the International Conference on Signal and Information Processing, 2016.

[2] Veeranjaneyulu N and Jyostna devi Bodapati, “Scene classification using support vector machines with LDA”, Journal of theoretical and applied information technology,2014.

[3] Tara N. Sainath, Abdel-rahman Mohamed, Nrian Kingsbury, Bhuvana Ramabhadram, “Deep convolutional neural networks for LVCSR”, ICASSP, 2013.

[4] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, “Image Net Classification with Deep Convolutional Neural Networks”, NIPS, 2012.

[5] LeCun, Yann, et al. “Gradient-based learning applied to document recognition”, Proceedings of the IEEE 86.11, pp. 2278–2324, 1998.