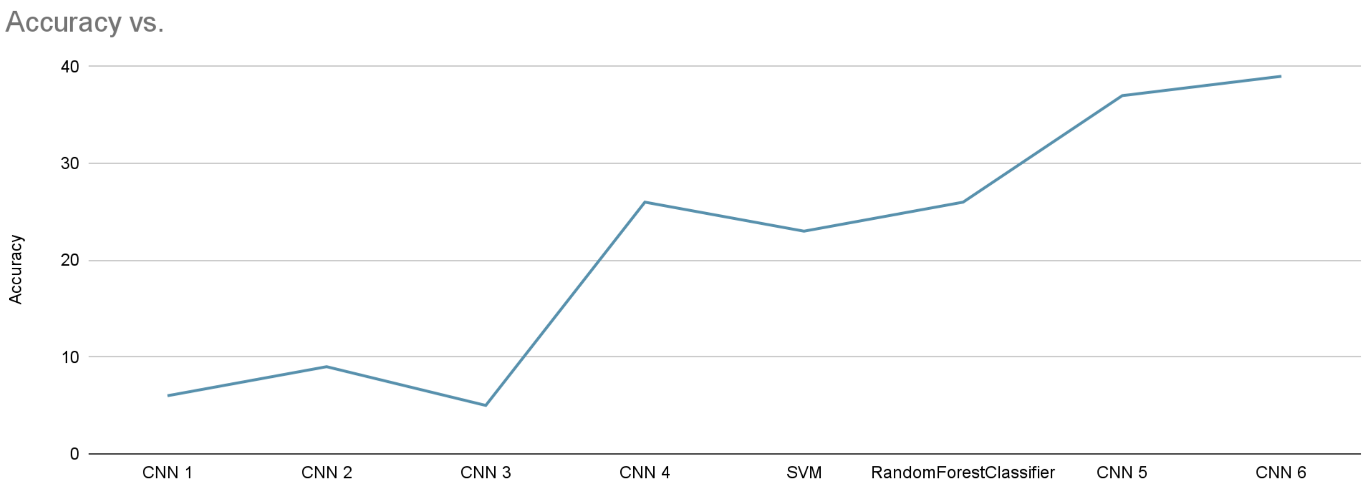
Deepam Sarmah
2020050
deepam20050@iiitd.ac.in

- **Observations**:
  - I observed that the target labels either belong to categories 0 to 24. There are in total 25 categories. This seemed to be a classification problem. Also, as there were only two classes I observed that this would be a binary classification problem.
  - I also noticed that the dataset was big and so a GPU would be required.
- **Methodologies**:
  - As it was my first time using only the sk-learn library for ML I read up the documentation of sk-learn's different multiclass classification algorithms. These included GradientBoostingClassifier, RandomForestClassifier, LinearDiscriminant Analysis, CNNs, QuadraticDiscriminantAnalysis etc.
  - For each such classifier I wrote code to first divide the training dataset into a training set and a validation set.
  - Next I used the training features and training labels to fit my model using any of the classifiers mentioned above.
  - Post that I used the predict function of those classifiers to predict the outcome on the test set.
  - I then created a pandas dataframe to save my output of Id and target which consisted of Ids of the test data and the predicted labels of the test set using the classifiers.
  - I then saved the output into a csv file for submission into Kaggle..
- **Experiments**:
  - I initially tried out a simple CNN but it didn't give me good accuracy. I got only 6%.
  - I then ran the algorithm for 100 epochs for testing purposes and found that accuracy increased to 9%.
  - I changed the optimizer from adam to rmsprop and then ran the algorithm. It gave me 26% accuracy.
  - I also implemented a SVM and tested it. It gave me only 23% accuracy.
  - I changed it a bit and implemented RandomForestClassifier and it gave me 26% accuracy.
  - Finally I made changes to the convolution matrix and tweaked it to (2, 2) and also experimented with the activation function. This gave me accuracy of 39%
- **Results**: (Accuracy) I submitted 33 different models and for some of them the accuracy was
  - CNN 1 : 6%
  - CNN 2: 9%
  - CNN 3: 5%
  - CNN 4: 26%
  - SVM : 23%

  - RandomForestClassifier: 26%
  - CNN 5: 37%
  - CNN 6: 39%
- **Analysis**:



- **Inferences**:
  - There is a lot of hyper parameter tuning that goes into making a good CNN.
  - SVMs are good for binary classification. But when it comes to multiclass classification they fail miserably. In such cases CNNs are widely used.
  - Random forests may not be the best choice for image classification because they may not capture all the relevant information in an image. CNNs can automatically learn features from raw image data which makes them superior to RandomForestClassifiers.