Project-II by Group AGRA

Smriti Jain
EPFL
smriti.jain@epfl.ch

Vidit Vidit
EPFL
vidit.vidit@epfl.ch

Abstract

In this report, we present our Image Classification System which has been trained to identify objects in the images namely, Airplane, Car and Horse. The system can do Binary Classification and Multi-Class Classification. The Binary Classification predicts whether the image has aforementioned objects or not.The Multi-Class Classification predicts the particular type of object in the image and if image doesn't have any of the mentioned objects it is labeled as "Other" Class.

The report is divided into different sections. We start with description of our data-set and it's visualization. The later sections talk about the models used for Binary and Multi-Class Classification and we conclude with the summary of our results.

1 Data Description

We have a data-set of 6000 images, which consists of 964 images labeled as Class 1(Airplane), 1162 images labeled as Class 2(Car), 1492 images labeled as Class 3(Horse) and 2382 images labeled as Class 4(Others). Along with these labels we have two different pre-computed feature vectors for each image. They are Histogram of Oriented Gradients(HOG) features and Convolutional Neural Network(CNN) features. The HOG feature vectors are of size 5408 and CNN are of size 36865.

The HOG features are created by dividing images(231×231) into blocks of 17×17 pixels. For each block histogram of gradient is created with 8 orientation bins. Each block is represented by 32 values after normalization of histogram with adjacent histograms. Hence the length of our feature vector is $5408(13 \times 13 \times 32)$. The CNN features are extracted using OverFeat software [5]. We didn't look into much details on how these features were extracted.

2 Data Visualization and Pre-processing

Using *hogDraw* function we could visualize our HOG Features. We could clearly see the that HOG features captures the oriented gradients in the images and are distinct for the four classes. We normalize our data to center it. We didn't have any visualization function for CNN features. So, we plotted the histogram for few images and got an idea of the range in which values of the feature lie. The CNN features are highly sparse.

While browsing through the images we could see few images were mislabeled, and mostly in Class4. These images will act as noise for our model and will result in high test error if our model tries to fit them. We tried to identify these mislabeled images using HOG features as we understood them better. We took mean value of the features for each Class 1,2 and 3 and compared the euclidean distance with the features of the images in Class4. But this method had a loophole as our objects were not centered in the image so, mean of HOG feature vectors were not giving any global information of the image. So we continued with the mis-labels in our data-set and tried not to over-fit our models.

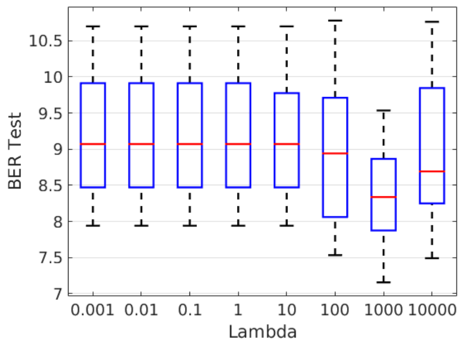
The Balanced Error Rate(BER) is taken as performance measure for our models. This function tells us how our classification model performs on all the individual classes. If we have unequal number of training data for the classes, which is true in our case, we might end up getting biased error predictions with $0 - 1 loss$. For the example given in project web page, $0 - 1 loss$ is 10% but BER will give it as 50% error, hence it is unbiased.

3 Binary Classification

3.1 Classification Models

3.1.1 Logistic Regression

In order to have a baseline for our classification models, we decided to train Logistic Regression on HOG and CNN features. Without any regularization, we observed BER for the CNN features to be $9.19(\pm 0.97)$ and for HOG feature to be $27.88(\pm 3.75)\%$. Since BER for CNN features was much lower than HOG, we proceeded with our analysis using CNN features only. As we have large feature dimension w.r.t the number of training samples, to avoid over-fitting we used L2 regularizer. The fig:1(a) shows the BER Test of 10-Fold Cross Validation(CV) with varying regularizer λ from 10^{-3} to 10^4. We can see for $\lambda = 1000$ best BER, $8.38(\pm 1.03)\%$ is obtained and taken as our baseline. With this baseline with understood that the linear model with CNN feature should be able to improve the BER on Test data.



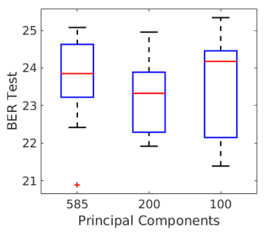
(a) Boxplot of BER vs Lambda

Figure 1: Logistic Regression with CNN feature for Binary Classification

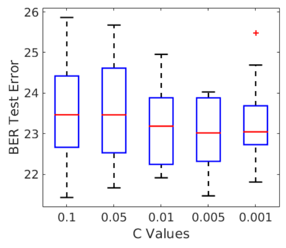
3.1.2 Support Vector Machine

We chose to start with Support Vector Machine(SVM) because we wanted to build upon our baseline. It was prudent to check the results with first linear SVM before trying other techniques, as our baseline suggested linear model should be able to classify the data. The linear kernel SVM has only one hyper-parameter to optimize so it was easier to apply grid search for it. We started with linear kernel function and had to tune hyper-parameter C, also called Box Constraint. The MATLAB's *svmtrain* and *svmclassify* functions were used for the implementation. The choice of C affects the margin around the decision boundary. The large C, does hard classification and tries to avoid mis-classification by having smaller margin. Whereas, small C tends to increase margin in turn allowing mis-classifications. SVM model training was done, firstly using HOG features and then CNN features.

HOG: We normalized our HOG features and used linear kernel function for SVM[3]. Initially we kept all the 5408 features for the training. In this setting the convergence was not achieved even when maximum iteration was increased from 15000 to 250000 and C was reduced to as low as 0.01, to do soft classification. This suggests that SVM is not able to find a decision hyper-plane with these features. We reduced our feature space dimensions using PCA. The reduced features from PCA does not guarantee to give better result as less important Principal Components(PCs) may be useful to classify image better but we went on with it and found it's working for these features. We varied the number of PCs and using 10-fold validation, keeping C at 0.01 looked into the BER. The fig:2(a) shows the BER with different PCs considered. We can see it's lower when first 200 PCs are used. Now we start optimizing our hyper-parameter C using grid search. It was seen that for the value of C > 0.1 no convergence was achieved. The fig: 2(b) shows BER with varying the C values for 200 PCs. The lowest BER 22.5% is attained for C = 0.005. We didn't delve more into this feature as



(a) Boxplot of BER vs Number of PCs

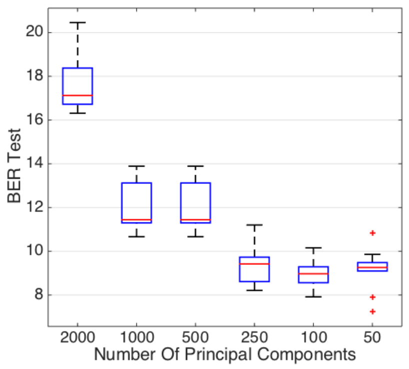


(b) Boxplot of BER vs C value

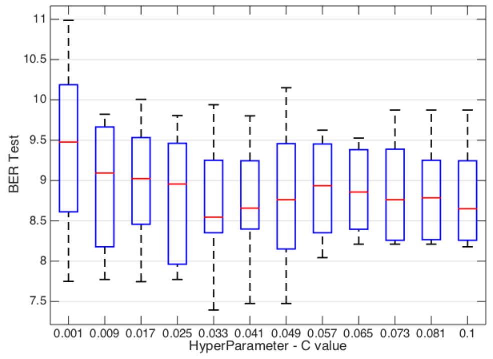
Figure 2: HOG Features With SVM Binary Classification

CNN features discussed in following part were presenting better results.

CNN: We trained our SVM with linear kernel function initially, keeping all the CNN features. We could see that with all the features, train BER was 0% and test BER mean for 10-fold CV was $8.3(\pm 1.0638)\%$. We varied C from 0.001 to 10 but train and test BER still remained same, probably because hyperplane found is linearly separating our data. We proceed ahead to see if reducing number features and then tuning hyper-parameter can improve our result compared to when all the features are considered. We applied PCA and tested with different number of PCs, keeping C at 0.01. The fig: 3(a) shows BER Test for 10-fold CV.



(a) Boxplot of BER vs Number of PCs



(b) Boxplot of BER vs C value

Figure 3: CNN Features With SVM Binary Classification

Here again we see with lesser PCs we are able to get low BER, suggesting that few PCs have the information for objects in the image. We varied number of PCs from $2000(93\%$ of variance) to 50. We take 100 PCs which has lowest BER $9.03(\pm 0.667)\%$ and proceed to optimize the C values for it. The fig: 3(b) shows BER Test for 10-fold Cross Validation keeping first 100 PCs for the CNN feature for different values of C. The best value obtained is for C = 0.025 which has BER as $8.82(\pm 0.76)\%$. We could see variance is high as we have mis-labels and it can be possible that in some fold we are getting number of them in our test set. We tried changing the seed for CV but couldn't improve our results.

Finally we compare our best model with 100 PCs and the model when all the features were considered. The 100 PCs model has higher BER than All features one. Hence the All Features model appears to be best with BER $8.30(\pm 1.0638)\%$, as our best model learned using SVM. This value is close to our baseline. We only considered CNN features hereon for our further analysis as they describe objects in the images better than the HOG features.

3.1.3 Neural Network Using CNN Features

As we saw that Linear SVM is able to classify using CNN features. So, there is less of a point to move to a more complex model by adding non linearity as in Neural Networks(NN). But we still tried using NN for binary classification. We considered tuning 7 parameters as mentioned in Table 1. All the other parameters are set to their default values.

We used PCA for dimensionality reduction, as it was computationally intensive to tune the parameters when working with all the features. The Figure 4(a) shows Average BER with different number of Principle Components for 5-fold cross validation and with the parameters, as mentioned in Table 1. The best we achieved was with all the features. But using all the features is computationally expensive. So, we chose second best which has an average BER of 9.61% with least variance(when using 100 PCs).

No. of hidden layers	1
Size of hidden layer	50
learning rate	2
momentum	0.5
Dropout	0.5
Epochs	30
Batch Size	500
L2 regularization constant	0

Table 1: Parameter Values for deciding Number of PC's

We proceeded with 100 PCs for the analysis. As, neural networks are very sensitive to initialization. So, we fixed the seed(random state) to avoid random initialization. We fixed the momentum to its default value of 0.5. In order to avoid over-fitting the noise, we wanted some kind of regularization. Dropout factor randomly omits some proportion of hidden units from the hidden layers during training to avoid overfitting on the training data[6]. We tried out [0,0.4,0.5,0.6] values of dropout. But 0.5 gave us the best result. So we heuristically decided the value of dropout as 0.5. We set the value of L2 regularization as 0 as we thought dropout will handle this issue. Number of epochs and batch size were kept same as shown in Table 1.

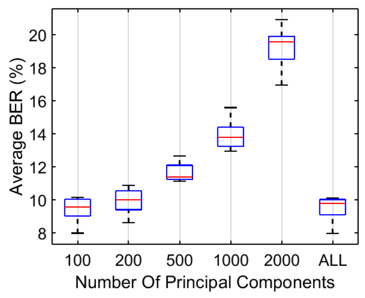
For finding optimal number of hidden layers, we compared between 1 hidden layer with 50 units and 2 hidden layers with 30,20 units respectively. The comparison box plot shown in Fig 4(b) clearly indicate that there is not much difference in the BER for two cases. So there is no point in increasing hidden layers. So, we took single hidden layer. For deciding the size of hidden layer, we plotted Figure 4(c) Average BER when varying number of nodes from 10 to 100 for 5 fold cross validation . As optimal layer size should be between the input layer size(100-in our case) and output layer size(2). Rest of parameters are same as show in table 1. We got average BER of $9.6584(\pm 0.5746)\%$(with size of hidden layer as 10). We also did the same thing with learning rate by varying it from 0.5 to 4 at interval of 0.5, shown in Figure 4(d). And we got learning rate of 2 as the best one.

The Best Neural Network model for us was with Size of hidden layer as 10 and rest of the parameters kept same as Table 1. This gave us $9.6584(\pm 0.5746)\%$.

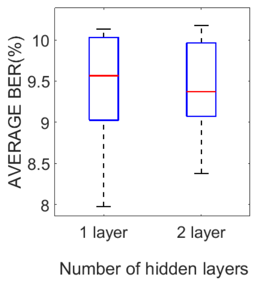
3.1.4 Random Forest with CNN Dataset

Why Random Forest Tree based models are robust to outliers. As our dataset consists of some mislabeled samples, which can be considered as outliers. So, in this case random forest can act as powerful binary classification model. Random forest is an ensemble method which fits many decision trees on different bootstrapped samples of dataset and then averages all the estimates to reduce the high variance of single decision tree classifier. We used Random forest because it to some extent overcomes the problems in decision trees. Decision trees is unstable bacause slight change in input causes change in tree structure, hence high variance in error. Thus decision trees try to overfit on the data. Random forest overcomes this problem of over fitting by reducing variance by averaging the predictions from ensemble of trees.

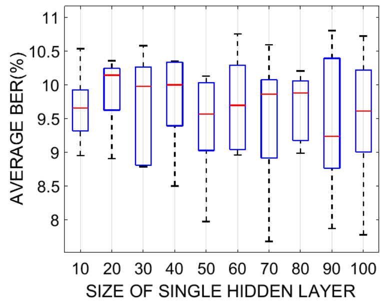
Processed Data:Random forest works best if proper features are used. Thus, we considered dimensionality reduction using PCA. To decide the number of Principle Components(PCs) , the correct



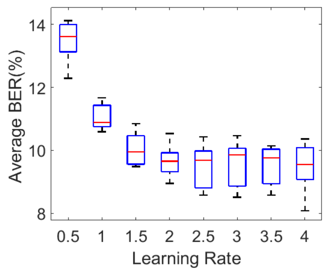
(a) Average BER(5-folds) vs Number of Principal Components



(b) Comparison between single hidden layer and 2 hidden layer



(c) Average BER(5-folds) vs Size of single hidden layer



(d) Average BER(5-folds) vs Learning Rate

Figure 4: Neural Network for Binary Classification

approach will be to find the best model for each set of PCs used and then compare the best models against the number of PCs used. But this process is time-consuming. So we decide number of PC's to be 500. We applied Random Forest model on reduced dataset of dimension 6000×500.

Parameters Tuning We considered parameters mentioned in Table 2 for tuning. The plan for finding the optimal parameters is as follows:- 1).Initially we heuristically and using some basic understanding of parameters decided search space for all those parameters. Then we used randomized search on parameters for 50 iterations with 10 fold cross validation to find those parameters which gave least BER and least variance across all folds. The code for this is submitted.

Max_dept	[3,4,5,6,7,8,9,10]
Min_sample_split	[4,5,6,7,8,9]
Min_sample_leaf	[4,5,6,7,8,9]
Max_features	[D,sqrt(D),log2(D)]
Num_trees	[20]
bootstrap	[True]
Loss Function	[”gini” , ”entropy”]

Table 2: Parameter Search Space

Search Space For Parameters The search space for different parameters is shown in Table 2. Increases in Max_depth can cause overfitting. Therefore, we restricted it to 10 or less. Python sklearn [1] sets it default value to None(which means tree can infinitely grow until it finds pure leaf nodes this can clearly cause overfitting to noise of our data). Hence we restricted the search space from 3 to 10. Similarly tuning Min_sample_split and Min_sample_leaf is also important because their very low value can cause overfitting and their very high value can cause underfitting. So, to avoid overfitting to our noisy dataset, we set their search space as [4,9].

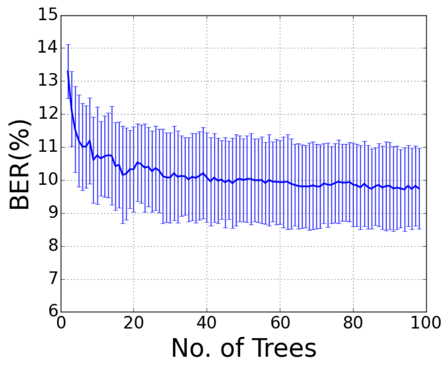
Num_trees indicate the number of decision trees to fit in a forest. Increasing the number of trees generally decreases the test error but converges after some point. We fixed Num_trees to 20 for parameter optimization using Randomized Search just to decrease the computation time. Max_features represent the maximum number of features which are randomly tried out at each split/decision point in order to find the best split. We tried following three values of max_features[D, sqrts(D), log2(D)], where D is the total number of features and tried to find the best for our dataset.

Final Parameters:We used RandomizedSearchCV [2] function of sklearn library[4] to find the optimized parameter by 10 fold cross-validation search over parameter space shown in Table 2 such that BER Error is minimized and variance across folds is reduced. Number of iterations for which parameter are randomly selected is set to 50. The best parameter values which we got in 50 iterations shown in Table 3. These value gave avg BER:$11.5(\pm 0.7)\%$.During this we fixed number of trees to 20.

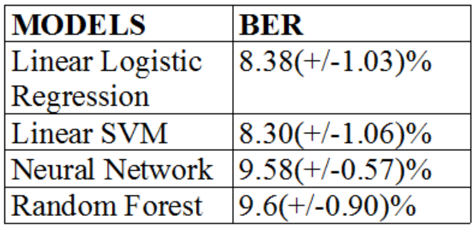
Max_dept	9
Min_sample_split	4
Min_sample_leaf	5
Max_features	D
bootstrap	True
Loss Function	”entropy”

Table 3: Best Parameter Values for 500 PC’s

Using optimized parameters(Table 3), we tried to find the optimal number of trees in a forest. Increasing the number of trees doesnt lead to overfitting. The Figure 5(a) shows that Ber reduces and converges at 99 trees(for 10-fold CV). Hence the best Random Forest model for binary classification is with 99 trees and rest of the parameters mentioned in Table 3. The generalized test error is $9.6(\pm 0.90)\%$.



(a) Average BER(10-folds) vs Number of trees in a forest



MODELS	BER
Linear Logistic Regression	8.38(+/-1.03)%
Linear SVM	8.30(+/-1.06)%
Neural Network	9.58(+/-0.57)%
Random Forest	9.6(+/-0.90)%

(b) Binary Best Model Selection

Figure 5: (a) Random Forest for Binary Class and (b) Compares different Binary Classification models

3.2 Best Model Selection

As presented in the following Table 5(b) we can see SVM All feature model beats all the other models with BER $8.3(\pm 1.0638)\%$ and hence used to make binary predictions on the test data. We expected SVM to perform much better but couldn’t tune the parameters optimally.

4 Multi-Class Classification Models

4.1 Classification Models

Since the CNN features were better in Binary Classification task we focused mainly on them for Multi-Class classification.

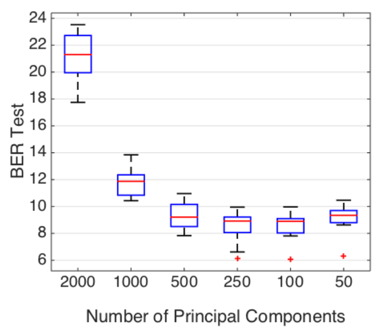
4.1.1 Logistic Regression

Using all CNN features, we couldn’t achieve convergence for Multiclass Logistic Regression model, hence took first 100PCs for the features. This particular number of PCs were taken as they had lower BER among other combinations, as seen in binary classification 3.1.2. We don’t regularize our model because we have already reduced our feature dimension. We did classification using One vs All(OVA) strategy and the test data was given to that class for which sigmoid prediction value was high. The BER test observed was $9.43(\pm0.7643)\%$ which acts baseline for Multi-Class classification models.

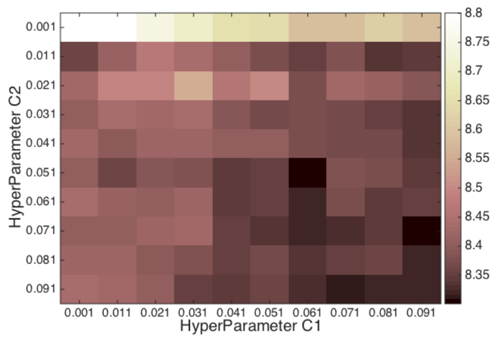
4.1.2 Support Vector Machine

We used OVA strategy to convert binary SVM classifier to Multi-Class one. We trained with taking all the CNN feature after normalization. The value of C was taken to be 0.025 the value we saw in SVM binary classification. The BER Test for 10- Fold CV was observed to be $8.6419(\pm1.043)\%$. But we also varied C but the BER test didn't change and BER train was 0. This behavior was similar to what we saw in binary classification 3.1.2.

We used PCA again for dimensionality reduction, and observed the BER changes as we change number of PCs for training the model. The fig: 6(a) shows BER Test for 10-fold CV for different number of PCs. The best BER $8.495(\pm1.063)$ is obtained for 100 PCs. As we have seen in Binary Classification, only few PCs are sufficient for the objects identification in the image.



(a) Boxplot of BER vs Number of PCs



(b) Heatmap of BER for different HyperParameter, Colorbar represents BER value

Figure 6: CNN Features With SVM Multi-Class Classification

We had kept the value of C=0.025 same for all the models in OVA strategy. But,we saw that most of the error was due to mis-classification in Class3 vs All and Class4 vs All model by looking into the confusion matrix for CV test data. This can be because we have mislabeled images in our data-set and probably most are of Class3. Hence, we need to vary the C depending on the model. Also because, we have unbalanced training data for the different classes it's better to have different hyper-parameters for different models.

We have to do grid search for 4 different hyper-parameters,which is tricky task as it would require lot of computations. We can reduce this grid search by just optimizing C for Class3 vs All and Class4 vs All. The reason behind this was we see BER train and test for Class1 vs All and Class2 vs All to be close. We varied the hyper-parameter C for Class3 vs All(referred as C1) and Class4 vs All(referred as C2) from 0.001 to 0.1. For C$>$ 0.1 convergence was not achieved even max. iteration was increased to 250000. For Class1 vs All and Class2 vs All a value C =0.1 was taken to do hard classification. The fig:6(b) shows the mean BER for 10-fold CV for the stated hyper-parameter setting. The best BER $8.27(\pm0.8302)$ was for C1=0.061 and C2=0.051. The other candidates for C1 and C2 with similar mean were rejected as they had higher variance. This is considered as our best SVM model for multi-class.

4.1.3 Neural Network

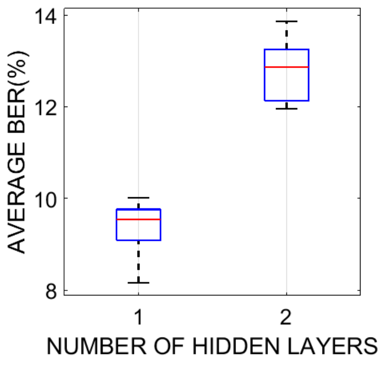
We followed the similar procedure as explained in Section 3.1.3. Initially we decided the number of PC's for dimensionality reduction using the parameters mentioned in Table 1. 100 PC's gave the least BER(with least variation across folds). So, using reduced feature space of 100, We tried checking for number of hidden layers(keeping rest of the parameters same as Table 1). Fig 7(a) clearly indicates that it was better with 1 layer. So, we proceeded with just 1 layer. Similar to the binary case, we decided size of hidden layer and learning rate using 5-fold cross validation, and rest of parameters same as Table 1. Average BER is least in case of a single layer with 90 units and learning rate of 1.5 and rest of the parameters as in Table 1(can not include plots because of space contraint). The best avg BER :$9.5497(\pm 0.3140)\%$.

4.1.4 Random Forest with CNN Dataset

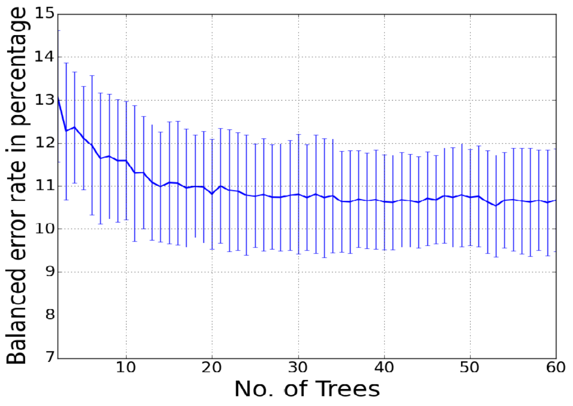
The reason for using Random Forest for multi-class classification is similar to that explained in Section 3.1.4. We used a similar approach for tuning the parameters of Random Forest as we used in Binary classification(see Section 3.1.4). We applied Random forest classifier on reduced 6000×500 CNN data(using PCA). We used Randomized Search with 50 iterations to search for the best parameters over Search Space(similar to the one for binary–Table 2). It took 15680.5 sec to get the Best Parameters shown in Table 4. We tried to find the optimal number of trees in a forest using (Table 4) parameters. Fig 7(b) shows that BER with number of trees for 10-fold CV. **Best Model**: The optimal parameters are mentioned in Table 4 along with number of trees as 99. The generalized BER value with 99 trees is $10.6(\pm 1.16)\%$ (10 fold CV).

Max_dept	7
Min_sample_split	7
Min_sample_leaf	4
Max_features	D(dimension)
bootstrap	True
Loss Function	”entropy”

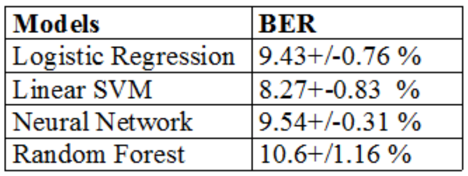
Table 4: Best Parameter Values for 500 PC’s for Multi Class



(a) Comparison between single hidden layer and 2 hidden layer



(b) Average BER(10-folds) vs Number of trees in a forest



Models	**BER**
Logistic Regression	9.43+/-0.76 %
Linear SVM	8.27+-0.83 %
Neural Network	9.54+/-0.31 %
Random Forest	10.6+/1.16 %

(c) Multi-Class Best Model Selection

Figure 7: a). Neural Network for Multi-Class Classification , b). Random Forest for Multi-Class c). Compares different Multi-Class Classification models

4.2 Best Multi Class Model Selection

For the Multi-Class Classification Linear SVM presents better BER of $8.27(\pm 0.8302)$ compared to all the other models, as shown in the Table 7(c).

5 Summary

We can see with the given dataset, Linear Models were sufficient to achieve good BER for the classification tasks. We chose SVM All feature model to make prediction for the binary classification and SVM model with tuned hyperparameters to predict multi-class labels on the test set.

References

[1] Random Forest sklearn python Library. `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`.

[2] Randomised Search sklearn. `http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.RandomizedSearchCV.html#`.

[3] Holger Hähnel. Support vector machines–einführende aspekte eines jungen klassifikationsverfahrens.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.