Project-II by Group Ha-Noi

Frederic Ouwehand **Marco Antognini**

Fall 2015

Abstract

This report states our findings and methodology for the image classification problem in Machine Learning using the Histogram of Oriented Gradients (HOG) and the *Overfeat* Convolutional Network-based image features. We compare how the basic and the common advanced methods such as Support Vector Machines (SVM), Random Forests and Neural Networks compete in a binary and a multi classification task. We conclude with two prediction models and their respective expected error when classifying unseen images.

1 Introduction

The goal of this project is to predict whether an image contains a given object or not using machine learning techniques we have seen in class. For this particular real world problem, we focus on the identification of three kinds of objects: horses, planes and cars. If an image doesn't depict any of those categories, we consider it to belong to a fourth category "other". Moreover, we only consider images that contain exclusively one of the mentioned objects.

We focus on two separate but highly related tasks. On the one hand, we want to separate images that contain either a horse, a plane or a car from the ones that don't contain any of these objects. On the other hand, we are interested in classifying the images into the four above-mentioned groups.

In the remaining of this section we present the data on which our classification system is built upon as well as how we measure the expected error rate. We then quickly explain how we used a Principal Component Analysis (PCA) and why it was useful for our task. Next, we present the different methods used to classify images for the binary task and discuss their respective performance. Using these initial observations, we continue the discussion with the multi-class predictions and show how we can combine different classifiers to build more complex classification systems. In the following section we highlight a few implementation details and then report the result of our final classification systems. Finally, we summarise our findings and conclusions.

1.1 Data Description

To accomplish our goals we are given two set of features extracted from actual images for both training and testing data sets. For the training set we also have at our disposal the original 6,000 images from which the two sets of features were extracted in addition to the labels associated with the images. It is important to note that we don't have access to the original pixel representation of the images in the testing set and therefore we cannot extract additional features, say by applying contour detection or image skeletonisation, nor apply some unorthodox method, such as using well-known search engines to identify the source of the images.

The first set of features we have access to is histogram of oriented gradients (HOG), a feature descriptor based on oriented colour gradients and used initially to detect pedestrians in [*HOG*] and was later used for more diverse object detection in static images. For this project, it corresponds to a 5,408-dimensional feature vector per image sample that was generated using [*PMT*] with 17 spacial bin and 8 orientation bins.

As shown in Figure 1, using this descriptor on images with few details can reveal the most important traits of the image but fails to produce a meaningful interpretation for more complex images, at least from the point of view of human beings.

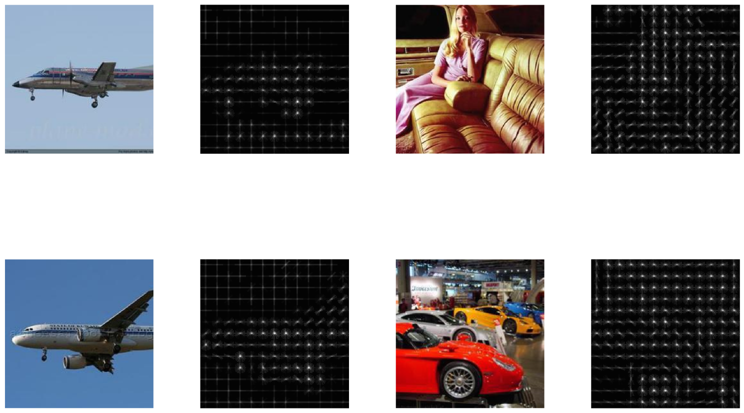


Figure 1: Visualisation of HOG features for simple and complex images

The second available feature set is based on the data extracted using OverFeat, a Convolutional Network-based image classifier and feature extractor that was trained on the ImageNet dataset as described in [*OverFit*]. For this feature set, we have for each of the 6,000 training images a very sparse representation of 36,865 data points and, unlike the previous set of features, cannot be trivially represented graphically.

Since the OverFeat features were generated using a Convolutional Neural Network we will refer to it as CNN in the remaining of the report.

Finally, the label distribution is reported in Figure 2. As we can see, the training data is not equally distributed among classes: a large majority of the data belongs to the category ”other”.

While examining some of the misclassifications we noticed that a few images classified in the *other* category were in fact horses, cars or planes. We identified about fifty misclassified images in the training set, which accounts for less than 1% of the data and therefore should not significantly impact our predictions.

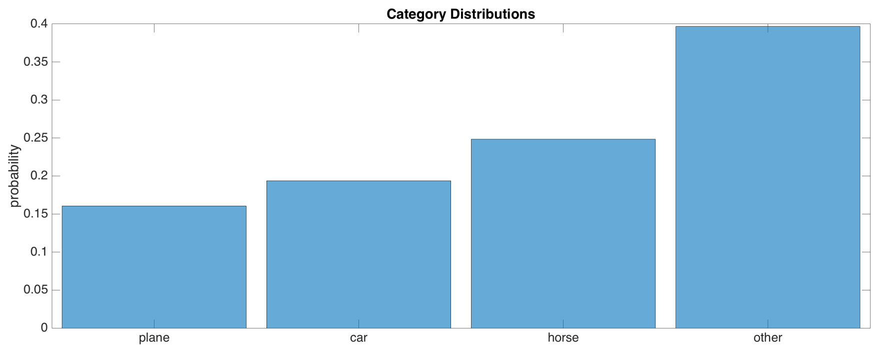


Figure 2: Data distribution for the training data

1.2 Error Metric

While 0-1 loss can give some information about the accuracy of a system, this error metric doesn’t take into account the distribution of the data among classification groups. If the distributions of the data for each class are equivalent then this is not an issue. However, as we have previously shown, our data is not well distributed among classes: e.g. the images of planes account only for 16% of our training data. Hence, we need to measure our misclassification error more carefully in order to be resistant to the unbalanced nature of the data.

To solve this issue, we used the Balanced Error Rate (BER), whose formula is shown in Equation (1), that basically produces an average of the error per class and therefore reports a meaningful information even when 40% of our training data represent images in the category ”other”. This means that if our predictor is utterly bad at categorizing cars then the resulting BER will be high, even if it is good at predicting horses and planes.

$$BER = \frac{1}{C} \sum_{c=1}^{C} \left[\frac{1}{N_c} \sum_{n=1}^{N} (y_n = c) (y_n \neq \hat{y}_n) \right] \tag{1}$$

In Equation (1), C denotes the number of classes, N_c is the number of examples in class c, y_n represents the ground truth and $\hat{n}_n$ its corresponding prediction for sample n out of N. For example, associating an image in a random binary or quaternary class will result in a BER close to 50% and 75%, respectively.

1.3 Principal Component Analysis (PCA)

Before thinking about applying any prediction model, we realised that the dimensionality of the provided features would be an issue. For HOG features, while still computationally possible, using the full set of features would result in a computational time blow-up with some methods such as Logistic Regression. As for the CNN features, naively using the full set of data would simply not be possible with regular computers, without mentioning theoretical issue due to the curse of dimensionality.

We therefore need a way to reduce the dimensionality of our data space. In order to select the right features from the data and leave out less relevant information, we decided to use Principal Component Analysis (PCA) to compress the data.

In order to tune the compression/loss ratio, we plotted the sum-of-squares distortion J as introduced in Section 12.1 of [*Bishop*]. We can infer from Figure 3 that reducing the dimensionality to $M \approx 400$ for HOG features results in a distortion of less that 10%. In practice, we have notice that reducing the dimensionality even further still yields good results.

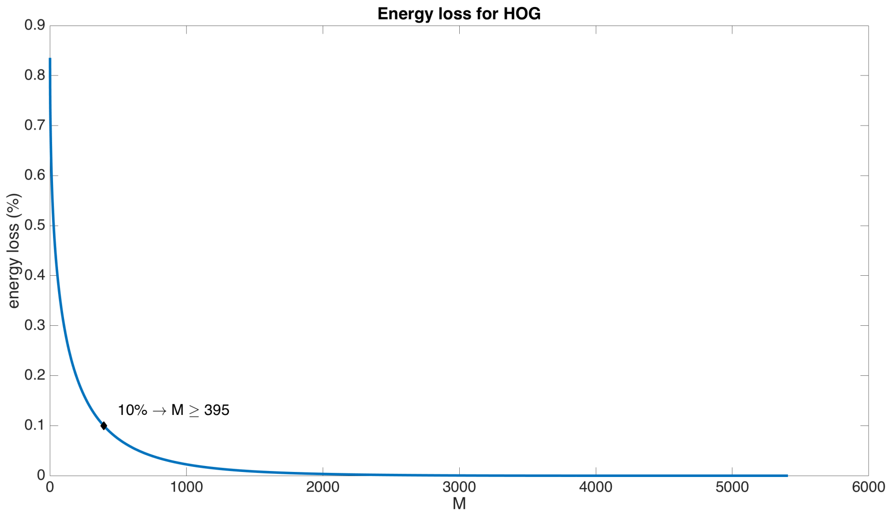


Figure 3: Energy Loss of the HOG matrix

As for CNN features, since its dimensionality exceeds the number of samples at our disposal, we have to use a trick in order to compute the eigenvectors as detailed in Section 12.1.4 of [*Bishop*]. Beside the implementation issue for high-dimensional data, the curve shown in Figure 4 is pessimistic:

since we only have 6,000 image samples, 30,865 of the 36,865 eigenvalues, that could exist were we to have 30,865 additional samples, are null. Therefore the distortion curve let us think that we need at least $M \approx 4000$ in order to retain 90% of the information. However, in practice we have noted that using $M = 200$, or even smaller, could achieve good results.

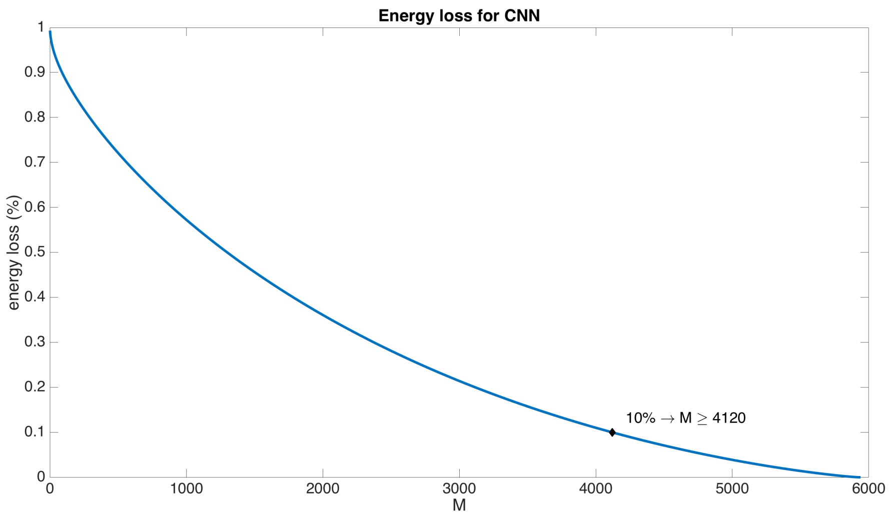


Figure 4: Energy Loss of the CNN matrix

2 Binary classification

Several methods were evaluated on the dataset using the training/validation methodology with a 0.7/0.3 ratio. The evaluation was repeated 20 times with a different random permutation of the images and produces an approximation of the Balanced Error Rate (BER) and a variance for each method. A full comparison is shown in Figure 5, in which each method is shown with its error on the validation set. Note that methods use exclusively the training set to compute their models. Hence, hyper parameters such as C for SVM or gamma for the RBF kernel are computed using the training data which may be split into training/testing pieces in the case of K-Fold. The intuition is that our final error estimate should not be computed using data that has been used to tune our models.

2.1 Path to the best model

Logistic Regression is chosen as a baseline method for this classification task because of its simple and intuitive theoretical foundations. However, it relies on the optimization of the problem directly defined by the observations which is computationally expensive when the data samples have a high dimensionality such as the HOG and CNN features. The principal component algorithm was used to project the high frequency features on the principal axis found in our data in order to reduce the dimensionality of the HOG and CNN features to 15 and 20 respectively. These parameters were chosen after a 10-Fold cross-validation among other candidate values and were selected to minimize the expected error. We used the gradient descent method with line search to find an optimal solution. The penalized logistic regression method on the HOG and CNN features after applying the PCA algorithm is shown in the 1st and the 7th column of Figure 5 under the labels *log reg HOG* and *log reg CNN* and the BER is 28% and of 11% respectively. Polynomial features transformations $\Phi = \left[x_n^{0.5}, x_n^2, x_n^3\right]$ did not improve significantly the model.

Neural network are powerful since they can model any kind of function using internal neurons and many libraries propose fast implementations iteratively absorbing the data in small batches such as [*DLT*]. The Neural Network method performed slightly better than the logistic regression on the HOG feature but worse on the CNN feature with respectively 25% and 12%. Overfitting was prevented by setting a low number of internal nodes in our network but few tuning was done on

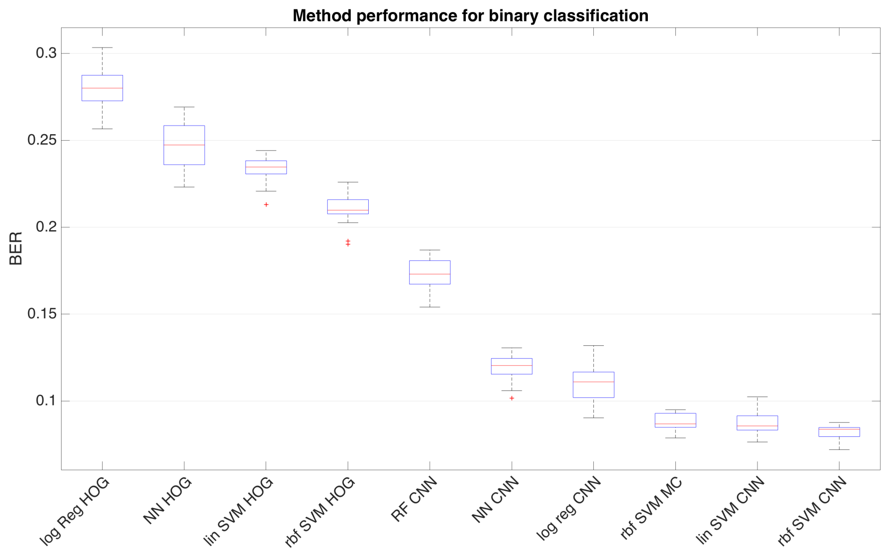


Figure 5: Evaluating the methods for the Binary Task

the model parameters since we realized that the Neural networks are extremely sensitive to their initialization which makes them hard to tune finely.

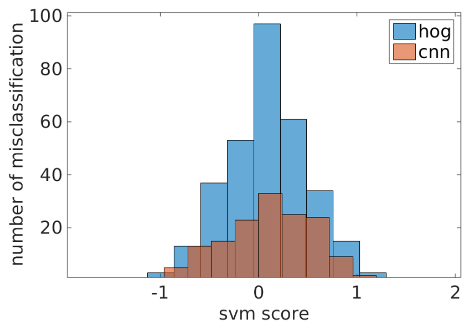
We preferred to investigate more stable models such as the Support Vector Machine model which does have the computational advantage of working on the dual problem since the number of HOG and CNN features is equivalent for HOG and six times bigger for CNN than the number of data samples. After training our model, the kernel for the prediction task is computed using the few training points associated with the support vectors and a final confidence score is produced for each unseen data point. Values between -1 and 1 correspond to points that lie inside the margin in one or the other side of it and the absolute values bigger than 1 are points for which the method is more confident about. Figure 6(a) shows that very few images are misclassified by SVM when their score is above or below 1. Compared to the Neural Network, the Linear kernel offers a 5% and 28% improvement for the HOG and CNN features respectively with a low variance in both cases. To reduce the computational load, the PCA algorithm is applied to CNN before SVM. We chose to select only the first 150 principal components from the CNN features matrix since the SVM algorithm performs equivalently well than with more principal components in this specific case.

The Polynomial kernels with degree 2 and 3 give slightly better classifications but the Radial Basis Function (RBF) clearly outperforms the Linear kernel by a 10% and a 2% reduction of the BER when applied on the HOG and CNN features respectively. We used the Gaussian type of RBF Kernel defined by $k(x, x') = e^{-\gamma \|x-x'\|^2}$ where γ is inversely proportional to the variance of the Gaussian distribution $\gamma \approx \frac{1}{2\sigma^2}$. We had to take a very big σ since the high dimensionality of the data points make them appear to be very far from each other.

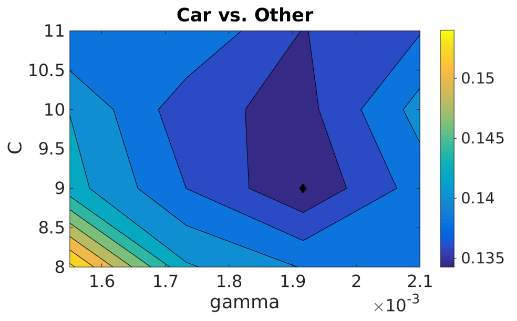
Anticipating the next multiclass predictions, Figure 6(b) shows that $\gamma = 1.9^{-3}$ and $C = 9$ are the best parameters for the RBF SVM model applied on the HOG feature for predicting the car label versus the other labels. Similar cross validation on the hyper-parameters were computed for finding the optimal parameters of the Support Vector Machine algorithm for the other classes. The SVM method with a RBF kernel is shown in Figure 5 in column 4 and 10 under the labels *rbf svm hog* and *rbf svm cnn*, respectively.

Random Forests is an efficient method which can be applied on the CNN feature without any preprocessing. It is shown under the label *RF (Random Forest) CNN* and achieves a BER score of 17.3% by averaging the predictions of 180 trees with a maximal depth of 7.

Our best and final model for the binary classification task is SVM with the RBF kernel on the CNN feature with $\gamma = 2.3^{-4}$ and $C = 3.25$ after applying the PCA algorithm on the 150 biggest singular



(a) Correlation between the SVM score and the misclassification rate



(b) Cross-validating the gamma and C parameters for SVM with RBF kernel on hog for the classification car vs other

Figure 6: Validation BER for each method of the binary classification

values. This method has a median BER score of 8.38% with a 0.52% difference between the 75th and the 25th quantile and a mean of 8.18%.

One could argue that the information lying in the HOG and CNN features may be combined to build a better model. However, decision trees or advanced methods such as Adaboost were not useful in combining our HOG and CNN models since their error rates highly differ. Models using the HOG feature range from a BER of 28% to a BER of 21% and the models based on the CNN feature score between 17% and 8.37%. However, we know from Figure 6(a) that the error rate of the SVM models is inversely proportional to the absolute value of their output. Based on this observation, we tried to combine the SVM models and to make them vote by taking the prediction of the SVM method which outputs the maximum absolute value for a given data point. Figure 5 shows this combination method in the 8th column under the label *rbf svm MC (Maximum Confidence)* with a BER of 8.7% which is greater than the RBF SVM on the CNN features.

3 Multi-class classification

Based on our previous findings, we decided to solve the multi-class problem by testing three main strategies. First, we applied the best known method for binary classification to independently identify images from the *car*, *horse* and *plane* categories to generate three confidence scores. As Figure 7 shows, using our previously build SVM model with RBF kernel, we were able, after carefully tuning the different parameters of the models, to detect whether a car is present in an image with an accuracy superior to 97%; detecting planes or horses is slightly harder with an accuracy of 95% and 93%, respectively and finally the accuracy of identifying images from the category *other* is down to 91%. Those results were obtained by using the CNN features. However, when we used the HOG features instead we achieved an accuracy several magnitude lower but with similar relative performance which means that cars are consistently simpler to identify with both set of features.

With that in mind, we devised a manual decision tree that, given the prediction scores for labels *car*, *plane* and *horse*, will decide to which of the four classes an image belongs to; i.e. if the score for car is higher than 1, the image is considered to be a car, else if the score for plane is higher that 1, the images falls into the category of planes, and if the confidence to be a horse is more than 1, the image is categorised as one, and otherwise we assume the image is none of those three categories. Using this simple decision tree based on the prior accuracy of each binary classifier, we were able to reach a BER of 7.8% as depicts in the *SVM + Manual Tree* column of Figure 9. This strategy does very few errors for images of planes, cars or horses since such as classification implies that a score returned by the SVM classifier is greater than the threshold of 1, meaning that the classifier is very confident in its decision. Figure 8 confirms than most of the false positives are objects from class *plane*, *car* and *horse* classified as *other* with a decision threshold of 1. Tuning further this decision boundary and setting it to 0.7 improved the results to an expected BER of 7.3% as reported in Figure 9 by the column *SVM + Manual Tree 2*.

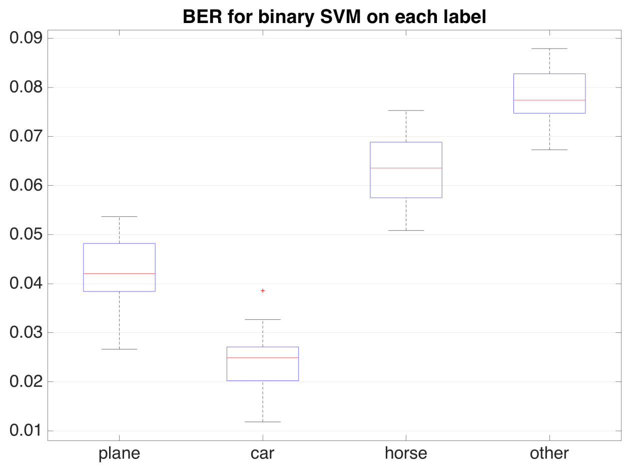


Figure 7: Evaluating the RBF SVM methods for the diverse binary classifications

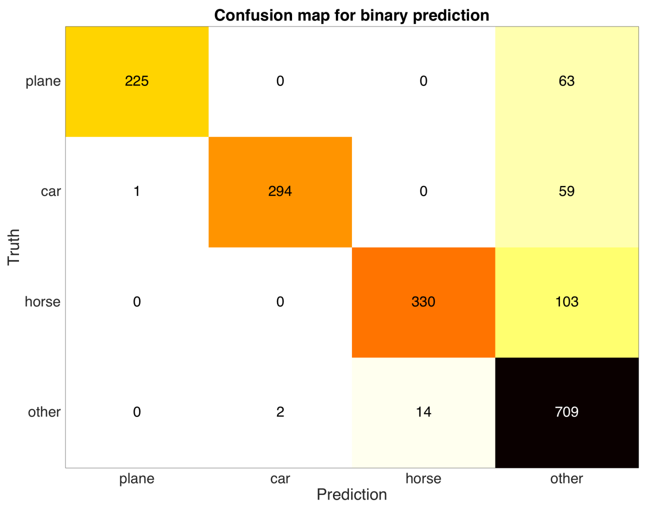


Figure 8: Confusion matrix for SVM + Manual Tree model

Additionally, we compared this approach against the Matlab implementation of SVM for multi-class identification, namely *fitcecoc* with the *'SVM'* learner. Thanks to its auto-tuning ability, we were able to achieve an error of 8.6% as reported in Figure 9 in the column named *SVM Matlab*. We expect, however, that the error could be slightly improved with this model by manually tuning its parameters instead of fully relying on the heuristic used by Matlab.

Next, we looked into simple one-layer, feed-forward Neural Network as we did for the binary task in Section 2, that is we only did a minimal tuning of its inner layer size to avoid overfitting. When applied on the CNN features, we achieved a BER of 9.2% on average on the validation set. If applied to the HOG features, we were only able to get an average error rate of roughly 30%.

4 Implementation details

We implemented the SVM and the PCA algorithm. However, we used the Sequential minimal optimization (SMO) provided during the laboratories for solving the quadratic optimization problem

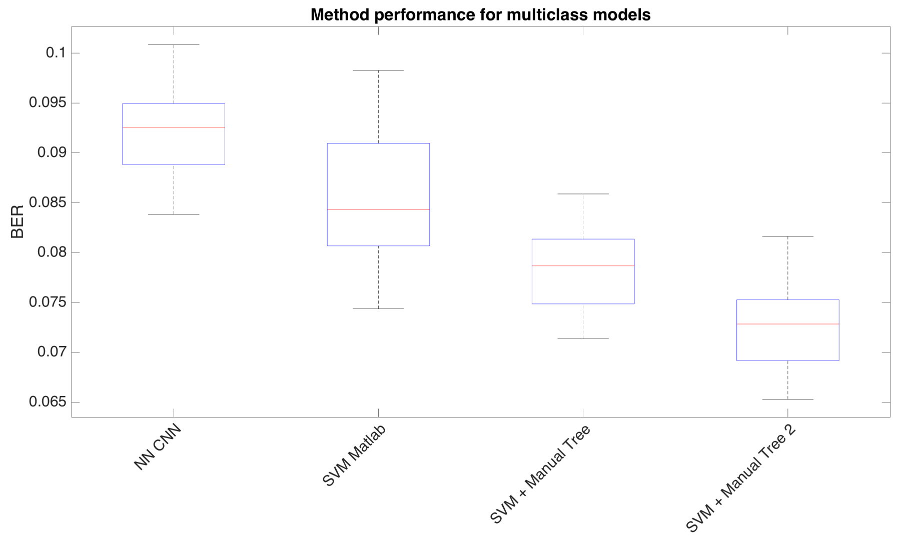


Figure 9: Evaluating the methods for the Multiclass Task

that arises in the SVM algorithm. We also used the Matlab functions such as *svmtrain* for their many options and flexibility in tuning the hyper-parameters.

5 Results

To summarise our results, we achieved for the Binary Task an expected BER of 8.18% when using the CNN features to train a SVM model that rely on a RBF kernel and a 7.3% expected BER for the Multi-class predictions with three binary SVM classifier based on the RBF kernel and a decision tree.

6 Conclusion

Given a labelled dataset of images, several methods were evaluated against baselines in their ability to produce good models with low variances. More sophisticated methods proved useful under some conditions such as reducing the dimensionality of the samples and choosing an appropriate error metric for the evaluation.

We encountered many practical aspects of advanced Machine Learning methods such as finding the appropriate range and sweet spots for each of our parameters for best performance while avoiding the overfitting problem.

References

[*HOG*] Dalal, Navneet and Triggs, Bill ”Histograms of oriented gradients for human detection” International Conference on Computer Vision & Pattern Recognition (CVPR 05), Jun 2005, San Diego, United States. IEEE Computer Society, 1, pp.886893, 2005.

[*OverFeat*] Sermanet, Pierre, et al. ”OverFeat: Integrated recognition, localization and detection using convolutional networks.” arXiv preprint arXiv:1312.6229 (2013).

[*Bishop*] Bishop, Christopher ”Pattern Recognition and Machine Learning”, Springer (2006).

[*PMT*] Piotr Dollár, Piotr’s Computer Vision Matlab Toolbox, `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`

[*DLT*] Palm, Rasmus Berg ”Prediction as a candidate for learning deep hierarchical models of data”, 2012.