**Street View House Numbers Image Recognition**

**1.0 Preprocessing**

The original data was split into training and testing with a ratio of 7.3:2.7. For each image in the original dataset, I converted it from red, green, and blue (RGB) to grayscale, using a weight of 0.3, 0.6 and 0.1 on the RGB value respectively. Then I scaled each pixel value in the image to be in the range of 0 to 1. The original pixel values are from 0 to 255, which may be harder for the neural network to deal with. I applied a simple linear scale by dividing by the max value of 255.

**2.0 Methods**

**SVC (RBF):** This is a classification model that employs supervised learning, and constructs hyperplane(s) to try to effectively separate the classes. Some benefits of using SVC to classify images are: they tend to perform well on image datasets, have a decreased risk of over-fitting, and can scale upwards to data of increased dimensions. The ‘RBF’ SVC outperforms the linear kernel, due to the data not being linearly separable.

**Logistic Regression:** Although primarily a binary classifier, the logistic regression model can be converted to perform multi-classification, by using ‘one-vs-rest’ regression for each class. The result is *n* logistic regression models corresponding to *n* total classes.

**K-Nearest Neighbours:** This classifier determines its predictions by finding k-nearest training points relative to each instance and averages the nearest k points. There is only one hyperparameter for this method (# of neighbors) and explicitly uses past data to classify new data.

**Random Forest:** This is an ensemble method and operates by training the data on multiple decision trees with a specified depth, returning the class that occurs most frequently. Overfitting is not as prevalent and tend to garner a lower variance compared to a single decision tree. This model tends to perform well by simply tweaking the tree count hyperparameter and is a good base model to begin with.

**Ensemble Stacking:** This is an ensemble method that aims to further increase model performance in terms of accuracy. It uses the prediction(s) of combined classifiers and employs another classifier to predict based on the results of the combined classifiers. I will use this to boost the accuracy of our base model(s) after fine-tuning their hyperparameters.

**2.1 Code Explanation:**

The notebook is split up into various sections. The first section applies pre-processing as discussed in report section 1.0, with the addition of under sampling. This is due to the training dataset being imbalanced, as many samples belong to classes 1, 2, and 3. As a result, the classes were under sampled to match the number of samples in class 10, achieving an even distribution.

Section two of the notebook consists of grid searches and random searches for optimal parameters, specifically for the random forest algorithm and SVC using RBF kernel. Predictions on the test data are then made for each of the models. A stacking ensemble model using SVC and random forest (voting classifier) was included in the list of tested classifiers. The hyperparameters that the search returned for random forest achieved a test accuracy of 64%. Ultimately, these hyperparameters were not used in favor of simply increasing the number of estimators and keeping the max depth and minimum samples split at the default value. Ideally, I would search for more hyperparameters. However, runtimes increased significantly as more hyperparameters were included in the search. The grid search for appropriate gamma and C values for the SVC gave a test accuracy of approximately 68% when gamma = 0.00001 and C = 10. However, I found that I could further increase test accuracy by having an even smaller gamma of 0.000001. Lowering the gamma value more than this value resulted in diminishing returns and a decrease in test accuracy. Refer to Table 2-11 below for the hyperparameters searched for the corresponding models, as well as the associated code. Hyperparameter searching was not done for KNN and logistic regression due to significant runtimes, although principal component analysis was applied to reduce dimensionality. I also output a confusion matrix for our predictions to see the details of the misclassified digits.

The third section in the notebook is where I aim to improve prediction accuracy on the test data by applying a stacking ensemble method using both the SVC and random forest classifiers. I utilize a voting classifier to determine the final prediction.

Table 2-11. List of hyperparameters used for each model

|  |  |  |
| --- | --- | --- |
| **Model** | **Hyperparameters Searched** | **Final Hyperparameters** |
| Random Forest | N\_estimators = [500, 700, 900], max\_features = “auto”, max\_depth = [1,5,10], min\_samples\_split = [3, 5], min\_samples\_leaf = [1, 2, 4], bootstrap = [True, False] | N\_estimators = 1500, max\_features = “auto”, min\_samples\_leaf = 1 |
| Random Forest w/ Rand Search | N\_estimarors = 900, max\_features = “auto”, min\_samples\_leaf = 1, max\_depth = 10, min\_samples\_split = 5 |
| K-Nearest Neighbors | None | N\_neighbors = 2 |
| SVC RBF | Gamma = [0.00001, 0.0001, 0.1], C = [1, 10] | Gamma = 0.000001, C = 10 |
| Logistic Regression | None | Multi\_class = “ovr”, solver = “lbfgs” |
| Stacking SVC with Random Forest | None | Voting = “hard” |

**2.12 Model Performance Comparison:**

Table 2-12. Comparison of Test Accuracy for Various Models

|  |  |  |
| --- | --- | --- |
| **Model** | **Test Accuracy** | **Comments for future recommendation** |
| Random Forest | 71% | Highest accuracy out of tested classifiers, slightly better than SVC |
| Random Forest w/ Rand Search | 64% | Likely need to add more parameters |
| K-Nearest Neighbors | 46% | May have overlap between classes, also a bit noisy due to images including overlapping digits |
| SVC RBF | 71% | Try testing different C-values |
| Logistic Regression | 17% | Try an increasing number of features (image size) |
| Stacking SVC with Random Forest | 71% | Could improve performance by adding another decent-performing model, with weight(s) |

I expected SVC RBF to perform well in comparison to the other base models, as a radial basis function with a very small gamma value tends to draw lower variance, with the tradeoff of higher bias. The SVC can separate the classes with hyperplanes, and depending on the choice of C value, is able to eliminate samples that fall too close to the margin between two classes. The biggest challenge with this dataset is the presence of multiple images where there is more than one digit, many of which are cut-off parts of different numbers. This is more difficult to train due to many training samples containing other image’s pixels, compared to a simpler dataset such as MNIST. This might be the reason why logistic regression did not perform well on this dataset. One possible solution would be to include the extra set of images to increase the number of samples available for training, and to be able to oversample the training data rather than under sample.

**2.13 Future Improvements:**

Due to long runtimes, performing grid search / random search on all models was not feasible. Thus, there is still room for improvement in various models, but I did not achieve those due to limited computation capacity of home laptops. The optimal scenario would be to test a multitude of different hyperparameters for all models to achieve the best performances for them. Inclusion of the extra set provided, combined with the current training set likely would have increased accuracy for the models as well. Accuracy could be potentially increased even further if I performed stacking SVC w/ random forest, and maybe a convolutional neural network prediction on top of the ensemble.