Project 1: Classification

CSCI 737: Pattern Recognition

Anushree Das (ad1707),
Nishi Parameshwara (np9447),
Omkar Sarde (os4802)

March 23 2021

1 Design

Our classification model is designed as given below:

- Parse the .inkml files using xml Element tree to get the annotation and labels.
- Perform pre-processing on the stroke information obtained after parsing the files. The data pre-processing steps were followed as given in [4]. We performed only three of the pre-processsing steps, namely, removal of duplicates stroke points, size normalization of the stroke points and resampling of the stroke points.
- Perform feature extraction to extract relevant features from the stroke information after pre-processing. After performing pre-processing and obtaining valid stroke information, we performed feature extraction on it as outlined by [4]. These methods were normalized y-coordinate, vicinity slope and curvature. This step gives us a 3-dimensional vector for stroke point.
- Train with SVM classifier and Random Forest classifier on the training data set with the ground truth file.
- Evaluate and test with SVM classifier and Random Forest classifier on the test data set.

2 Pre-processing and Features

In this classification model, the following pre-processing steps described in [4] have been performed:

- Duplicate point filtering
  Remove redundant and repeated points which are not needed in the stroke data points. We need to remove it as as to generalize the model and not confuse it as the duplicate points do not convey any important information.
- Size normalization
  Scales coordinate values between 0 and 1 for y axis and just modified

the x-axis with respect to the aspect ratio of the image file. Due to the formula used for normalization, we might come across mathematical error especially, when there is not much stroke coordinates as it can lead to divide by 0 error. For this reason, we modify the formula to consider this edge case when the length of the points is less.

- Resampling
  To ensure that all the points are equidistant from each other in time as well as space along the original trajectory of the symbol, we need to resample each symbol to 30 points. So, we must calculate that for each symbol to 30 points, each stroke must have a fixed set of points and adjust accordingly. Then we try to fit the points to get the perfect curve after resampling.

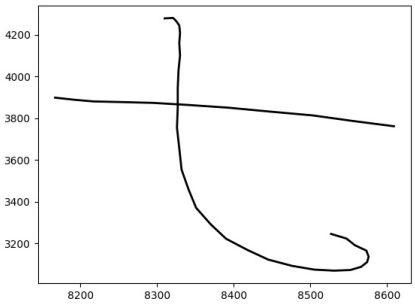

Figure 1: Sample Symbol before pre-processing

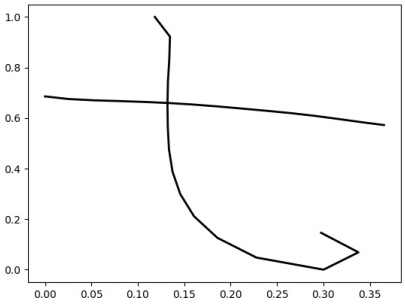

Figure 2: Sample Symbol after pre-processing

After pre-processing, we use the updated stroke information to extract features. These features are all mentioned in [4] i.e. normalized y-coordinate, vicinity of slope and curvature. As these three features will be calculated for each of point of every symbol and we have 30 points per symbols, so 90 features will be extracted for each symbol in our datafile.

1. Normalized y-coordinate
   Size normalization obtained during pre-processing is used as onr of the features.

2. Vicinity of slope
   Tangent between the straight line joining points $x(t-2), y(t-2)$ and $x(t+2), y(t+2)$ and horizontally across point $x(t-2), y(t-2)$. Consider the figure 3 from [4] to understand this angle. Here, tangent of alpha is the vicinity slope.

3. Curvature
   Tangent between the straight line joining points $x(t2), y(t2)$ and $x(t), y(t)$ and the straight line joining point $x(t), y(t)$ and point $(x(t+2), y(t+2))$. Represented as Beta in Figure 3.

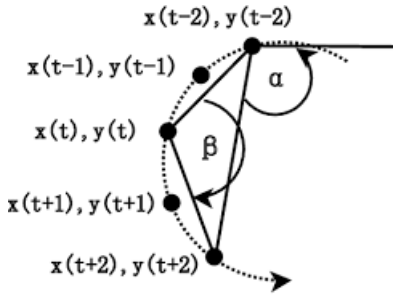

Figure 2. Slope(α) and curvature(β)

Figure 3: Vicinity Slope and Curvature

3 Classifiers

In this classification program, we have utilized two classifiers.

1. Support Vector Machines
2. Random Forest Classifier

3.1 Random Forest Classifier

A random forest fits a number of decision tree classifiers by taking samples of data set. It uses averaging to increase accuracy and reduce over fitting [3]. It is made up of a large number of decision trees that work as an ensemble. Each individual tree in the random forest gives its own class for prediction and the majority vote decides the winner.

For our model, random forest classifier taken from ensemble section of scikit- library [2] takes the following parameters: min_samples_split = 3, n_estimators = 50, max_depth_size = 20.

3.2 Support Vector Machines

Support Vector machines are used for high dimensional data. To classify data points, we use hyperplanes. Different points fall onto different sides of the plane. Support vectors are points that are closer. They decide the position of the hyperplane. So, removing them will change the direction.

For our model, support vector machines taken from scikit-learn library [1] takes all the default parameters except gamma: Regularization parameter C = 1.0, kernel = rbf, degree = 3, gamma = auto, shrinking = True, cache size = 200.

4 Experiments and Results

1. We attempted transfer learning using a pre-trained CNN as a feature extractor. To evaluate the approach we utilized a pre-trained VGG-16 to extract features and used a single layer as the final layer to the model. This model converged quickly as seen in 1, in just 9 epochs achieving 78.5% Accuracy.The initial attempt at transfer learning failed as the number of features before the final layers were large (4096) for both the classifiers.

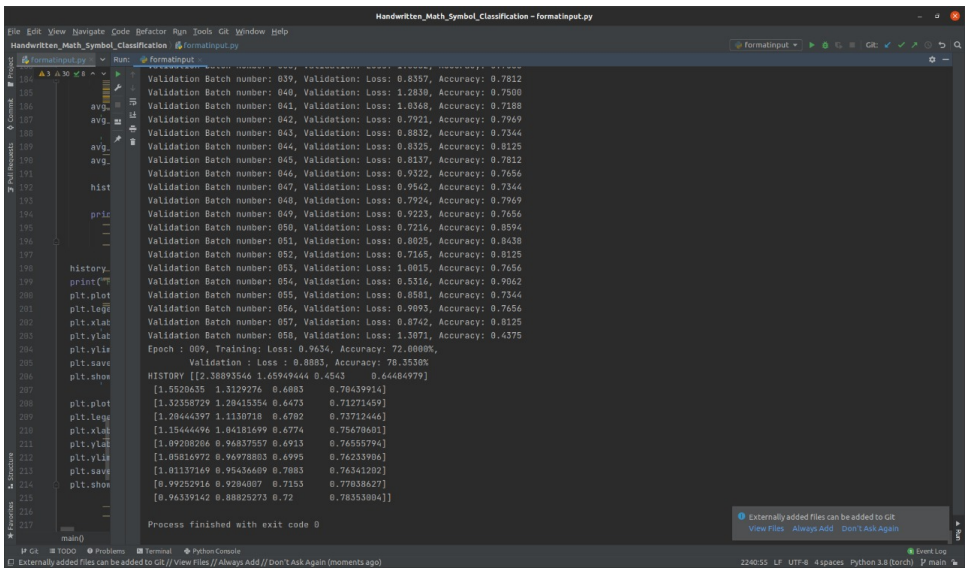

Figure 4: Using CNN as feature Extractor

2. Such vast feature space caused low accuracy in random forest implementation, whereas caused explosion in SVM training times and model sizes necessitating change in approach based on available hardware. Hence, we decided to utilize feature engineering methods described in [4].

3. For training we used features extracted from the entire training symbols dataset, fed through pandas dataframe pipelines to both the classifiers. A hold out set was partitioned from the training set to be used as the validation set. The features from the training set were then used to train both the classifiers.

4. Initially, random forest model took a large amount of time and space to run which even caused memory allocation error message to be displayed. So we tried on Google Colab, which suprisingly, gave the same message. So, we decided to take a closer look in our data and realized while feature-extraction and pre-processing, the new values are stored in a new variable which is increasing the computational overhead. So, we decided to store it in a pickle file.

5. Training Set accuracy (RF): 0.9896324386383009
   Validation Set accuracy (RF): 0.9003708281829419
   Training Set accuracy (SVM): 0.7153443387200584
   Validation Set accuracy (SVM): 0.7124492318559068

6. The predicted output class for testSymbols has been stored in a .csv file. As there is no ground truth file for the same, there is no recognition rate.

7. The Random Forest Classifier outperformed the SVM classifier on the validation set due to the choice of feature engineering methods in [4].

References

[1] [n.d.]. Scikit-Learn Support Vector Machines. https://scikit-learn.org/stable/modules/svm.html/

[2] [n.d.]. Sckit-Learn Random Forest. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[3] [n.d.]. Understanding Random Forest. https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[4] L. Hu and R. Zanibbi. 2011. HMM-Based Recognition of Online Handwritten Mathematical Symbols Using Segmental K-Means Initialization and a Modified Pen-Up/Down Feature. In *2011 International Conference on Document Analysis and Recognition*. 457–462. https://doi.org/10.1109/ICDAR.2011.98