Offline Signature Verification System

Data Preprocessing

We get the original data from ICDAR 2011 Signature Verification Competition. The whole dataset contains the signatures of 67 person. There are 12 pictures with real signature, and 12 pictures with fake signature for each person. So the total number of pictures is 1608. We use 80% of the data as the training dataset, 10% is validation dataset, and the remain 10% is test dataset.

Feature Extraction

At first, pixel values of images are extracted as features to be used in KNN(pixel). However, the results are bad, we only get 17% accuracy on test dataset. After that, we got inspiration from Prof. Downey, after talking with him, we found that we can use transfer learning to help use extract the features of the images, thus we don't have to build our own convolutional neural network. And VGGNet is utilized to get new representative features of the images. This new set of features is a vector with 4096 elements.

Model Training

1) Nearest Neighbor

Based on the features with VGG representation, we tried L1 distance and L2 distance for 1-nearest neighbor, the accuracies are the same, 84.26%, because we always find the 1-nearest neighbor, so it doesn't matter how we calculate the distance. Next, we tried 3-nearest neighbor and the final estimation result is decided by the majority among the three nearest neighbors, the accuracy is a little bit smaller, 82.95%. We think the reason could be that the second and the third nearest neighbor bring same estimation, which is different from the estimation made by the nearest one. In this way, the estimation may be misguided.

2) Neural Networks

The inputs to the neural networks are still based on the features with VGG representation. We use one-hot encoding method to encode the labels and create the validation and test datasets. Softmax activation function is used for the output layer and cross entropy is used to calculate the cost.

When size of batch=20, the accuracy and loss changing tendency is shown in the picture below:

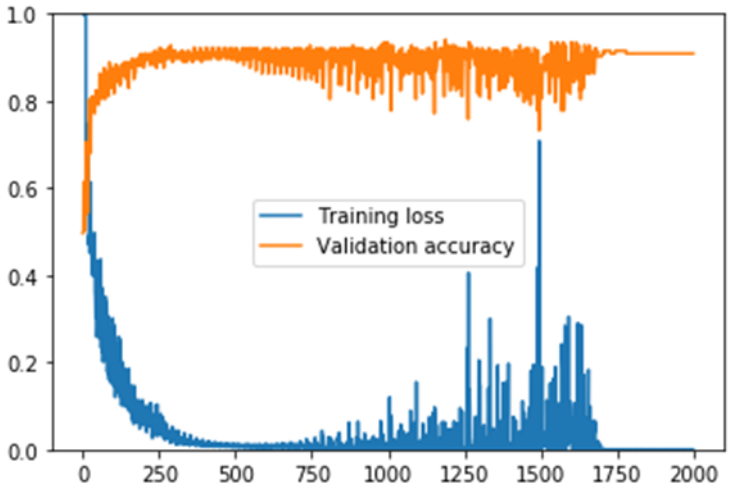

Figure 1: The learning curve (training loss and validation accuracy) when size of batch is 20

When size of batch=10, the accuracy and loss changing tendency is shown in the picture below:

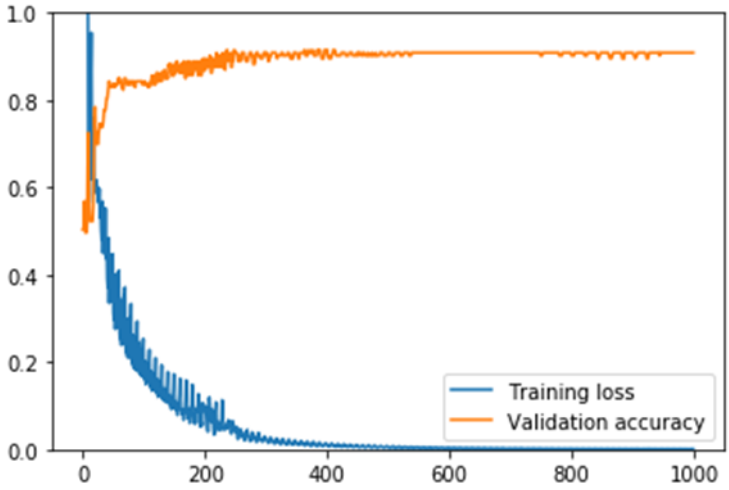

Figure 2: The learning curve (training loss and validation accuracy) when size of batch is 10

From the above two pictures, we might see that the curve become smoother as the size of batch is smaller. The reason might be that the smaller size of batch means the less possibility to be trapped in local minimum and thus shown unstable and accuracy and loss. And the test accuracy for size of batch=10 is 94.77%, which is pretty high.

3) Support Vector Machine

The input to SVM is still based on the features with VGG representation.

Because our dataset is not linear inseparable, so we didn't use kernel function to build the model. And we use Hinge Loss as our loss function, it was defined as the below equation:

$$\frac{1}{n}\sum \max\left(0,1 - y_i(wx_i + b)\right) + \alpha\|w\|^2$$

And α is called the soft regularization parameters. In our project, we investigate the influence of different soft regularization parameters.

When the soft regularization parameter is 0.001, the loss and accuracy changing tendency is shown in the figure 3 and 4 below. The test accuracy is 72.78%.

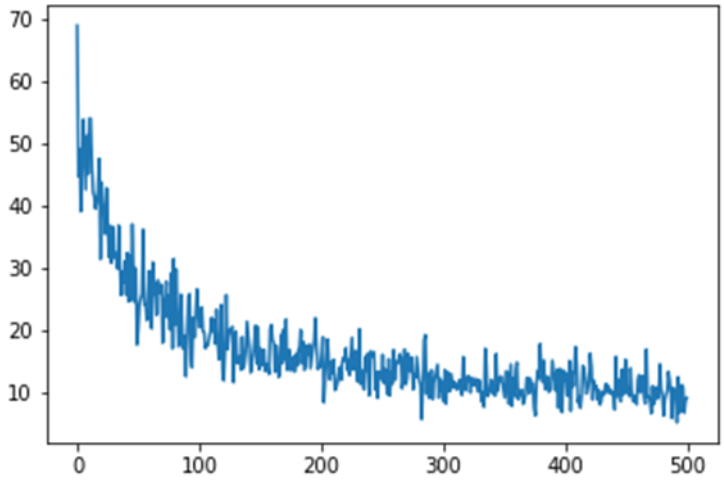

Figure 3: The learning curve (training loss) when soft regularization parameter is 0.001

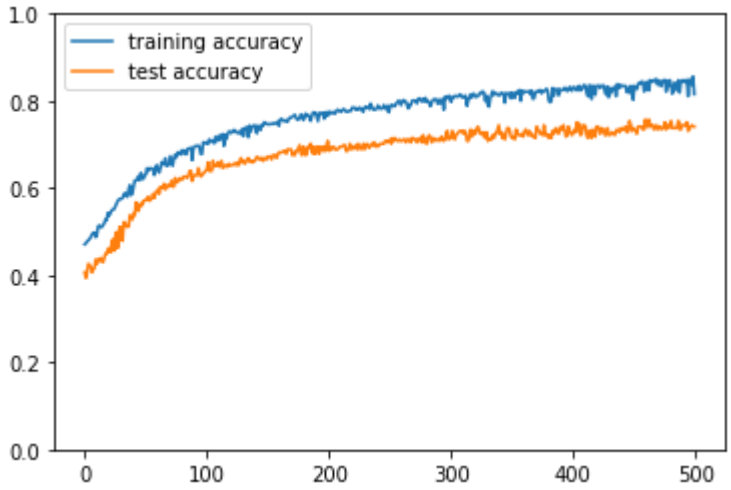

Figure 4: The learning curve (training and test accuracy) when soft regularization parameter is 0.001

When the soft regularization parameter is 0.1, the loss and accuracy changing tendency is shown in the two pictures below. The test accuracy is 81.67%.

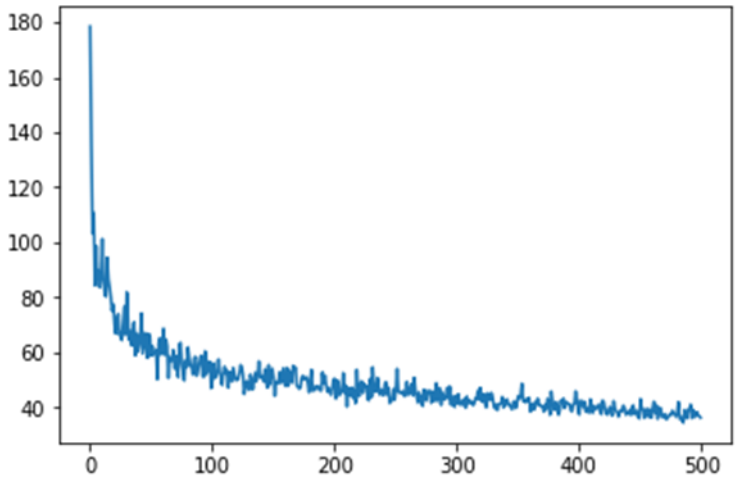

Figure 5: The learning curve (training loss) when soft regularization parameter is 0.1

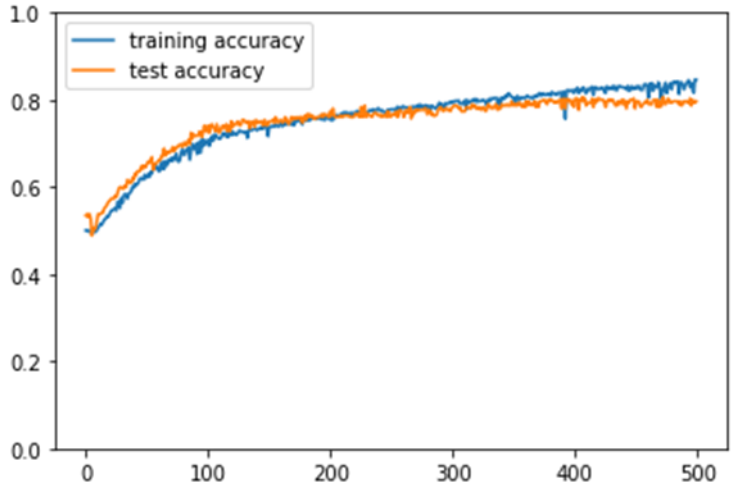

Figure 6: The learning curve (training and test accuracy) when soft regularization parameter is 0.1

When the soft regularization parameter is 1.0, the loss and accuracy changing tendency is shown in the two pictures below: The accuracy is 75.73%.

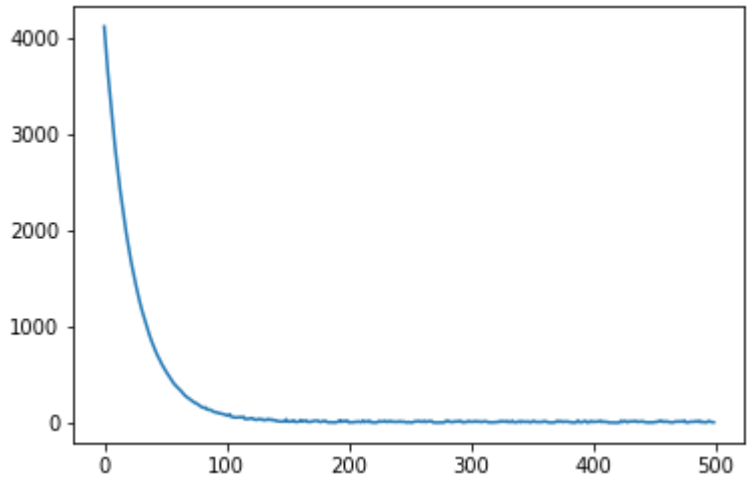

Figure 7: The learning curve (training loss) when soft regularization parameter is 1.0

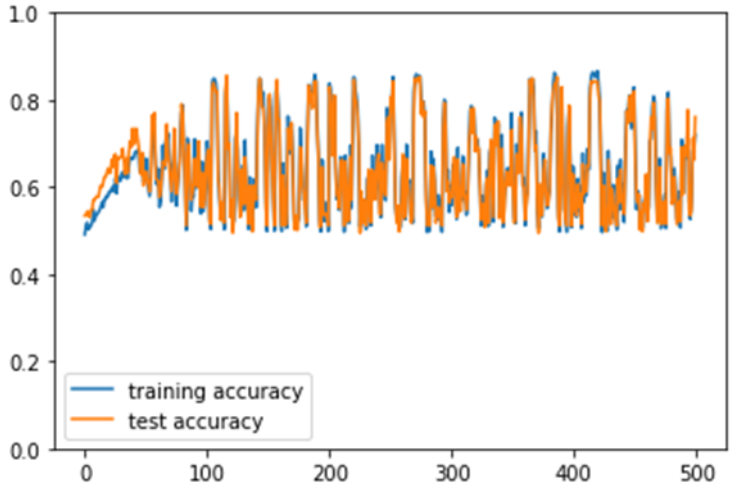

Figure 8: The learning curve (training and test accuracy) when soft regularization parameter is 1.0

From the figures, we can see that a too big or too small soft regularization parameter will both decrease the test accuracy and may result in more extreme fluctuations in loss or test accuracy. This tells us that in order to reach a better performance, we need to balance error and margin, a well-selected soft regularization parameter can help us do that.

Conclusion and future work

At beginning, we just train the model on a small part of the dataset, and we found that the model accuracy is not very high, and it seems that the nearest neighbor got the highest accuracy compared to neural networks. However, when we increased the training dataset and feed more images to the network, it seems that it got better, and reach 94.77% accuracy.

Because for the limit of the time, we didn't try enough machine learning algorithms, and I think that's will be our major future works. For example, we could still try to use Boosting algorithm to classify the images. And we can also try to use other CNN models, e.g, AlexNet, GoogleNet, and see what happens.

Result Table

Algorithm	Accuracy
1NN (Pixel Representation) - L1 Distance	17%
1NN (Pixel Representation) - Cosine Similarity	50%
1NN (VGG Representation) - L1 Distance	84.26%
1NN (VGG Representation) - L2 Distance	84.26%
3NN (VGG Representation) - majority vote	82.95%
Support Vector Machine	81.67%
Neural Networks	94.77%

Members' Work

In this project, Xiangguo Liu mainly works on data pre-processing and k-NN method, Chenghong Lin mainly works on NN and SVM method, and builds the website. Finally the result analysis, discussion and report writing are done cooperatively. Since it's only a two-person project, we two both discussed the realization of each detail, thus a very effective discussion is reached.

Reference:

Alvarez, Gabe, Blue Sheffer, and Morgan Bryant. Offline Signature Verification with Convolutional Neural Networks. Tech. rep., Stanford University, Stanford, 2016.

The dataset source:
http://www.iapr-tc11.org/mediawiki/index.php/ICDAR_2011_Signature_Verification_Competition_(SigComp2011)