**Use CNN to identify digits and letters**

**Abstract:** In lab 6 we learned how to use SVM to identify. Since we what to identify all digits and letters, and SVM performed poorly, that is, spent a great deal of time (nearly 15 minutes) to obtain an accuracy around 82.82%, we want to use CNN to do the same problem and expect better result.

**Keywords:** CNN, digits and letters, EMNIST.

1. **Dataset description**

our data set is still **EMNIST[1]** , and we use the emnist-digits.mat, emnist-letters.mat at first as the demo, then we use emnist-bymerge.mat, which has merged C, I, J, K, L, M, O, P, S, U, V, W, X, Y and Z as one class to get a total 47classes.

If we aim at creating a hand writing prediction app, using 47 classes dataset can provide better performance by judging the capital letters and small letters.

1. **Formulation**

We use valid padding in the convolution layers and use relu in every layer except the last layer using softmax. Since we have 36/47 classes, we use categorical crossentropy as the loss function. (notice that we should use one-hot on the target to apply this loss function) we use adam as the optimizer and adding batch normalization layers which is very important in this case.

1. **Experiment**

We start with the 36 classes dataset in lab6, to verify if there are any problems in our idea. Then we found some misclassification in the confusion matrix, so we tried to make the confusion matrix balance by manually set the class weight. However, it turns out that the balance of confusion matrix will sacrifice the validation accuracy, so we finally decide to train the data without using class weight.

In the main body project.ipynb, we use 47 classes dataset instead of the 36 classes one. It is acceptable for us to witness misclassification between 1:1/18:i/21:l and 9:9/g:g because they look like each other. What surprise us is the misdetection between 15:F and 40:f. After we printed out the figure of F in the dataset, we found that some people wrongly written the F into f. in this case, we gain a validation accuracy of 0.8794.

To acquire better performance, we looped over the number of channels (include the number of channels in convolution layers and dense layers.) to increase the performance. And after finding the best channels number, we can loop over the learning rate of optimizer.

First we loop through nconvs= ([16,32,48]) and nnodes= ([384,512,640]),

Then base on the previous result, we loop through nconvs= ([24,32,40]) and nnodes= ([320,384,448]).

Finally loop through lrates= ([0.5,0.1,0.05,0.01,0.005]) and reach the validation accuracy of 0.8832 with convolution channels = 32, dense layer channels = 384, and learning rate = 0.005.

Moreover, we decide to go further to achieved higher accuracy by adding dropout. we use the same way as above to acquire the best parameters. Finally, by using nconv=48 and nnode=512 , the accuracy reach 0.8845.

1. **Bonus program**

We also wrote a tiny program that can shows the prediction of the input figure by using the project program we wrote. (photo\_to\_2828\_gray.ipynb)

1. **Conclusion**

We save a great amount of time by using CNN instead of SVM to identify the letters and digits. (training 46000 samples in SVM using 30 min, training 46000 samples in CNN using 8 min). And obtain a higher accuracy by using CNN(0.8845) instead of SVM(0.8448). Furthermore, we have a brighter future since we can go deeper to the deep learning region.

To be more specific, go to <https://github.com/jianyupang/iml-project> and verify project.ipynb

1. **Future scope**

In order to increase the accuracy, we also try to use the VGG16 pre-trained program on the new dataset, but the result is not ideal (only around 0.8 accuracy), that might due to the lack of digits and letters training data in the VGG16. (project\_VGG16.ippynb).

We can go deeper by adding more layers in the CNN to get better result if we can access to a GPU and even try many different advance ways listed on the leaderboard of EMNIST.