Computer Vision II : Project 2

Khushboo Mandlecha & Rutvij Dhotey

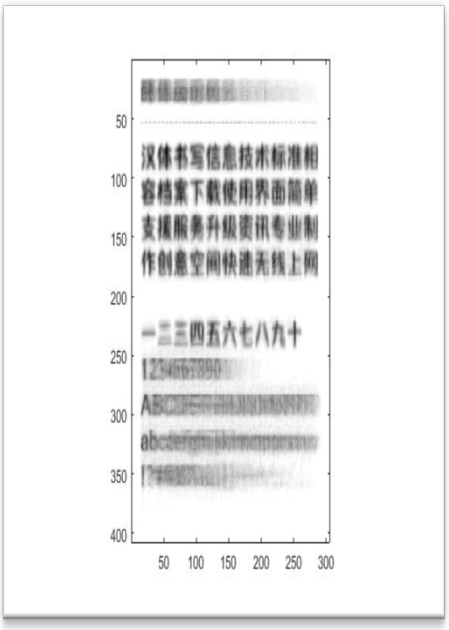
Summary of work:

The idea here is to identify Chinese digits in various fonts from 1 to 10. Before diving into the project statement we went through the example tutorial, practiced, ran and understood the code and how CNN works. After doing that we did the following.

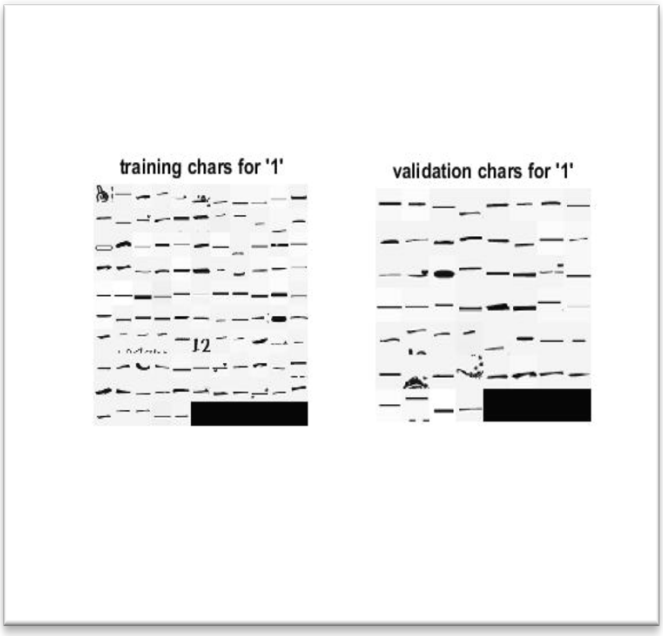
We have our data set where we have 156 images for all digits from 1 to 10. Some of it will become our training data and rest will become the test data. We trained our training dataset using 3 types of CNN combinations. We first implemented the CNN method as described in the oxford tutorial. After that we gave it our sample test data. Here our sample data is the digits on the mahjong tiles. Once we implemented this we tried it with two variations of CNN. After training our data with these two variations we again tested with the mahjong tiles dataset. We have listed all our results in the end. In the end we mentioned what we learnt from the project.

Crop, Align and Pad Characters:

The images given to us were fairly uniform. We considered using a general approach .The row index 212-245 mainly contained the number row of that specific font. The Numbers 1- 10 generally started from column 13 in the increments of 25. That is exactly what we did and we got a pretty decent dataset of extracted images that were stored in the folders Cropped/i where i is between 1:10. We got mean of the images as shown below. We are using a 32*32 image size. The above figure shows why we have used 212-245 as our cropping dimensions



Mean Image for all Data Sets



Initial Training and Test Data for digit1

Training and Test Data:

We split the images into training and test using modulo operation. Here while putting images in the database we put every 1st and 2nd image as training and 3rd image as test. So basically 2/3rd of data is training and 1/3rd data is test. We initially configured the CNN using two layers taking help from the oxford CNN tutorial. We trained our training set by continuously increasing the number epochs. We used our test data for mahjong tiles from 1 to 9.

Effect of Continuous Increase in number of Epochs:

We started training the data with 10 epochs and then added more epochs. We realized that with each epoch our validation and training success rate goes up and after like 40 Epochs, the rate is saturated and cannot improve any further.

Jittering:

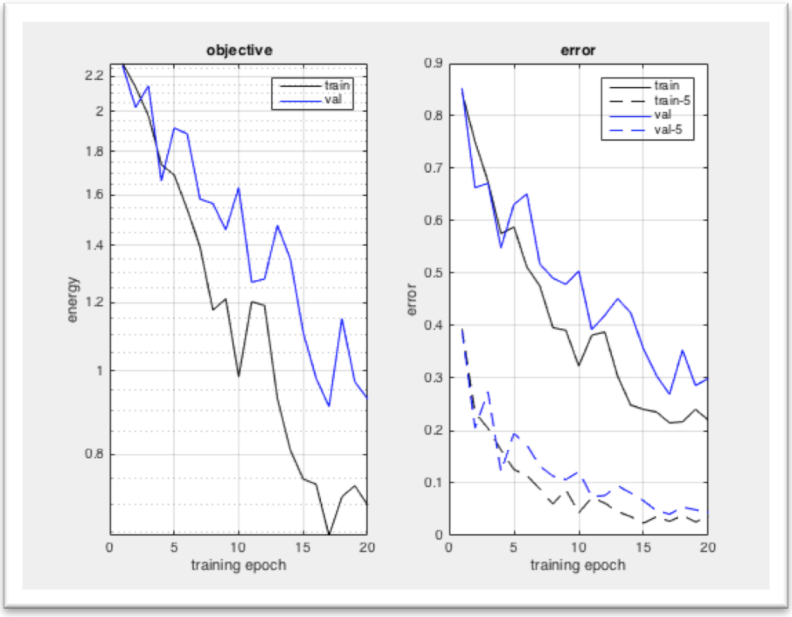
We have followed the jittering provided by the practical in Oxford Tutorial. The jittering in the tutorial states that we have shifted horizontal by +-5 and vertical by +-8 pixels. The main aim of jittering here is to make the data more realistic.

Variations for Training and Testing CNN:

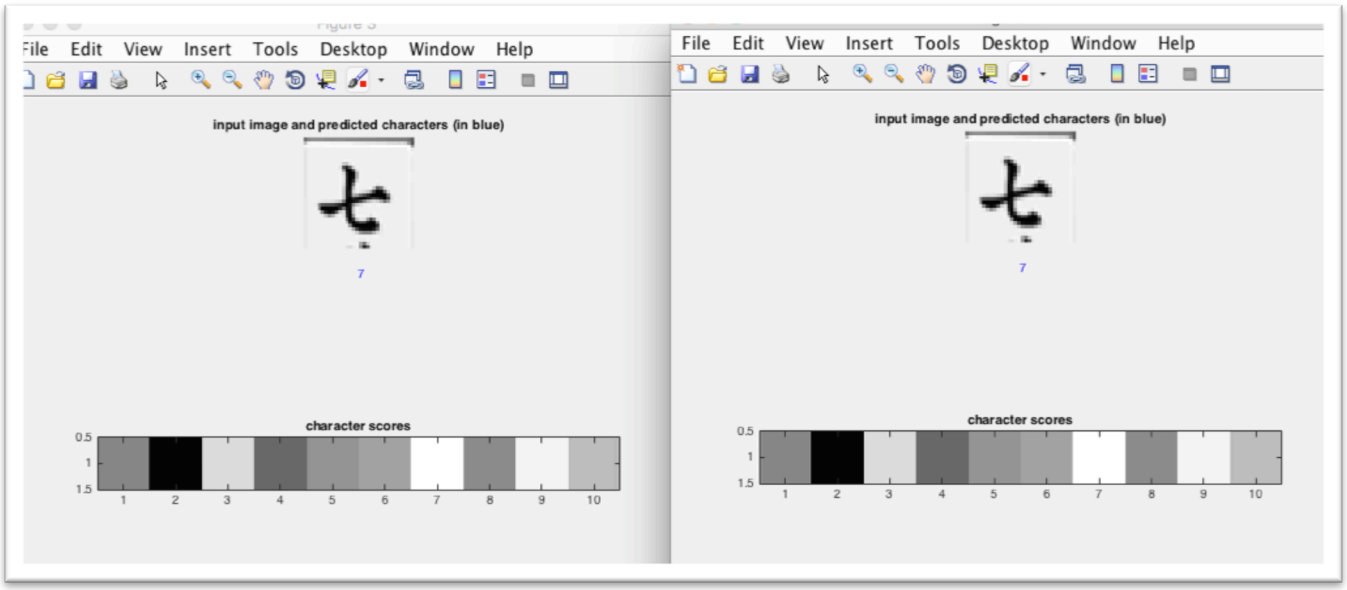
1) Initial CNN Layer:

We used the following set of transformations for our initial variation. This one we have taken from the oxford CNN tutorial. We ran this for 20 epochs. These are the results.

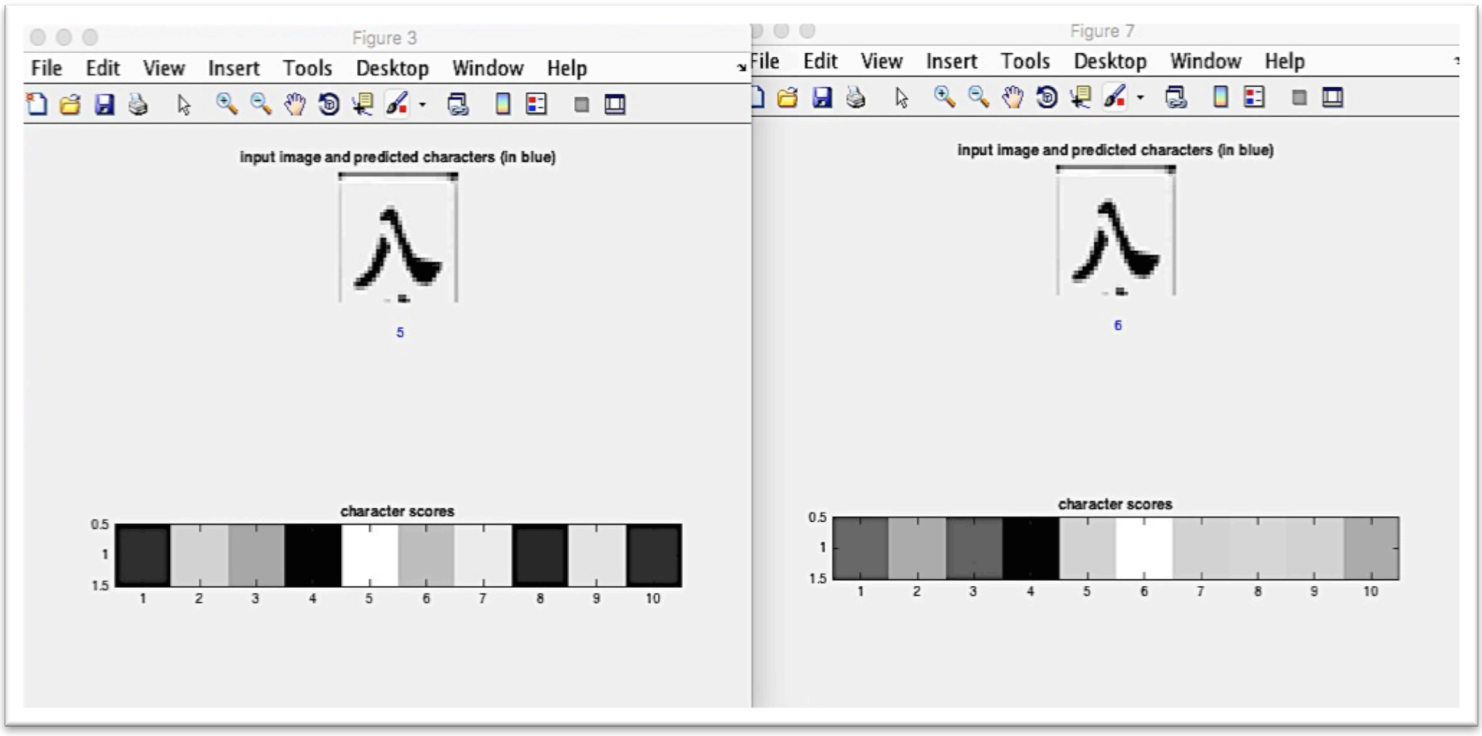
conv->pool->conv->pool->conv->relu->conv->softmax .



With Jitter Initial Variation Error rate



Without and With Jitter for 7



Without and With Jitter for 8

Observations for Initial Variation:

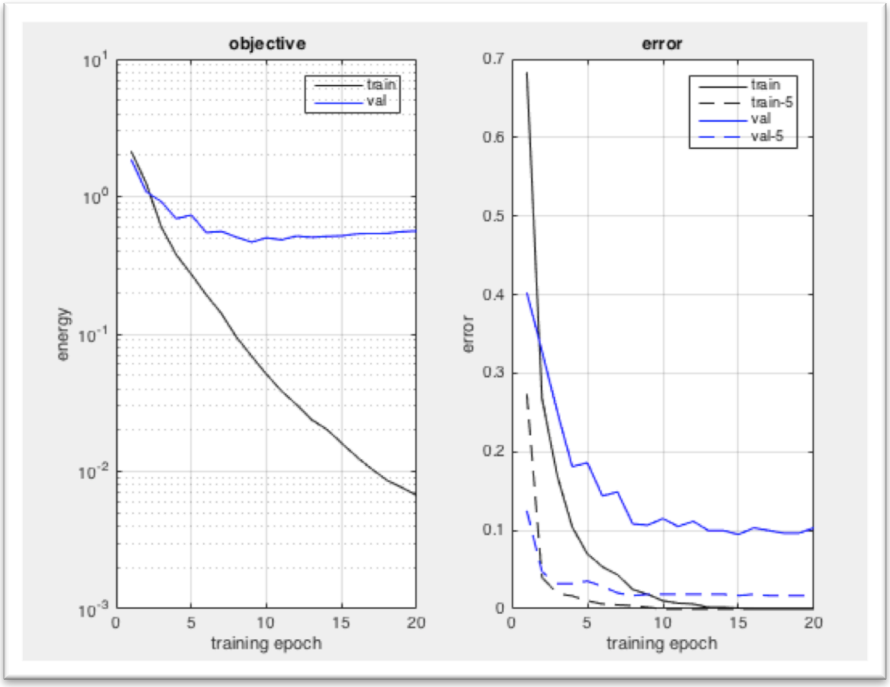
We can see from the above observations that, we get perfect results for 7, however we do not get good results for 8. Both being wrong, the main observations is the scoreboard for with jitter. The score of 8 is much closer to the answer than the score for 8 in without jitter.

---

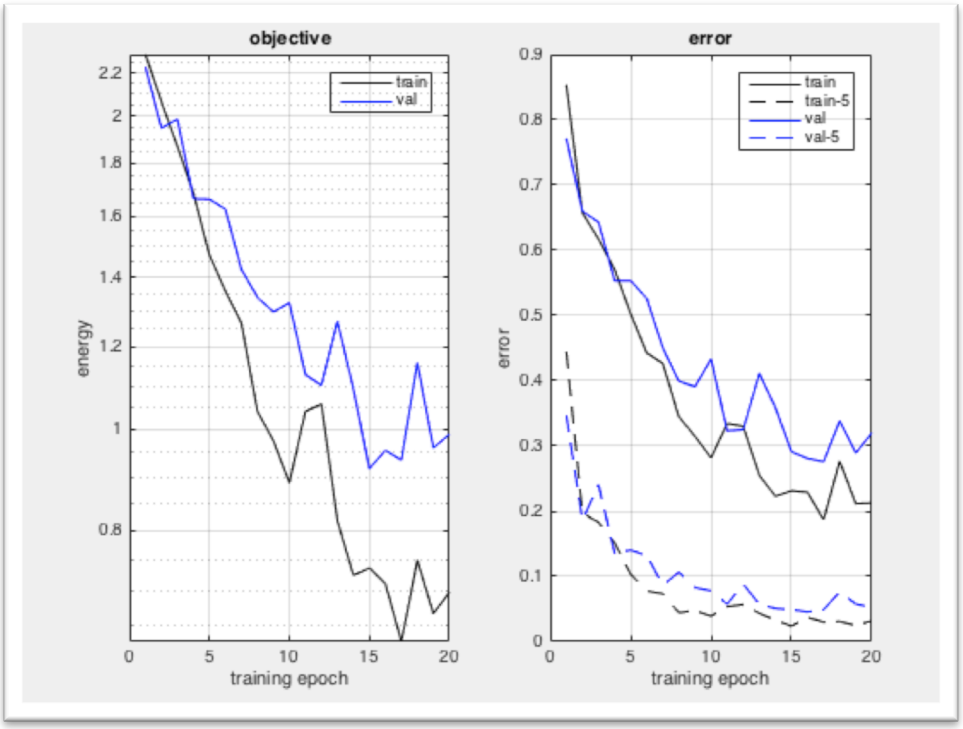
2)Variation1 CNN Layer:

We have used the following layer structure for this variation, in total 11 layers. We ran this for 20 epochs. These are the results

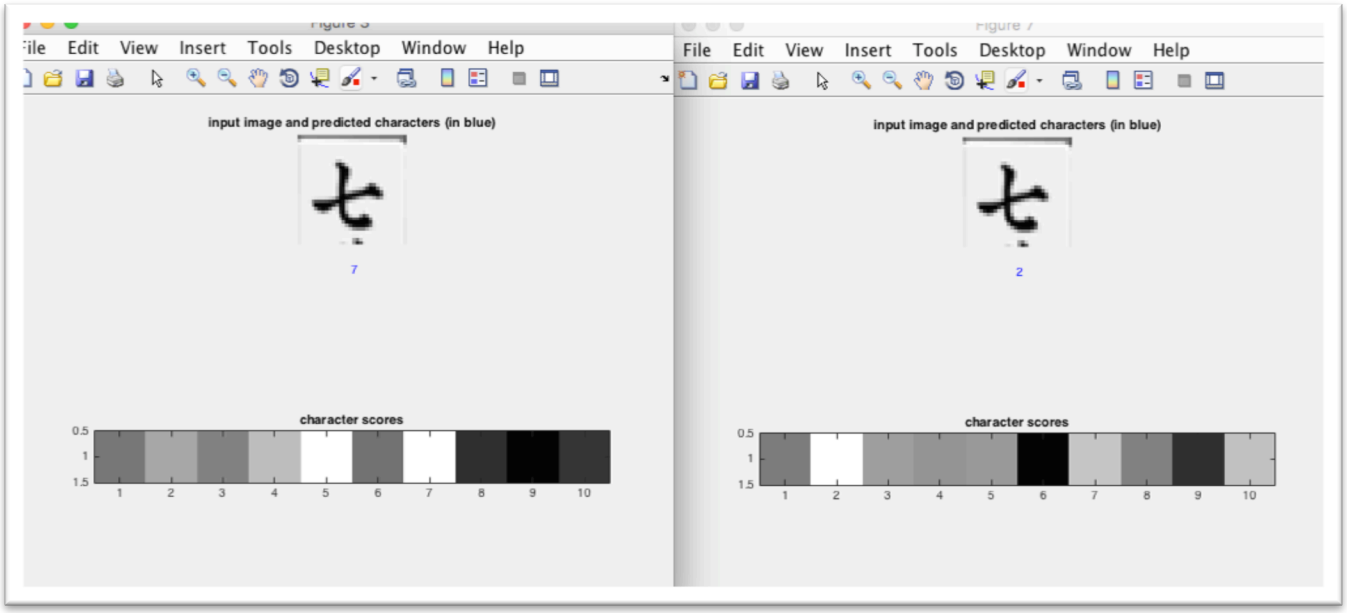
conv->conv->pool->relu->pool->relu->conv->relu->pool->conv->softmax



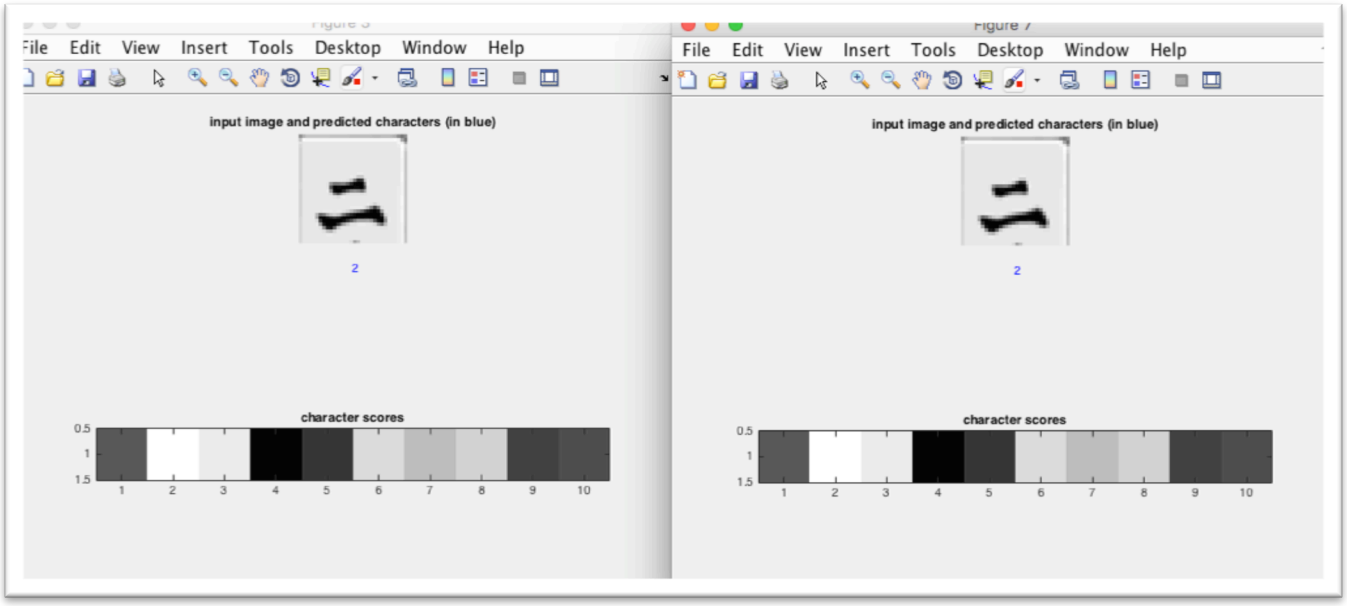
Error Rates for W/0 Jitter: Variation1



Error Rates for W Jitter:Variation1



Variation 1 for digit 7



Variation 1 for digit 2

Observations for Variation1:

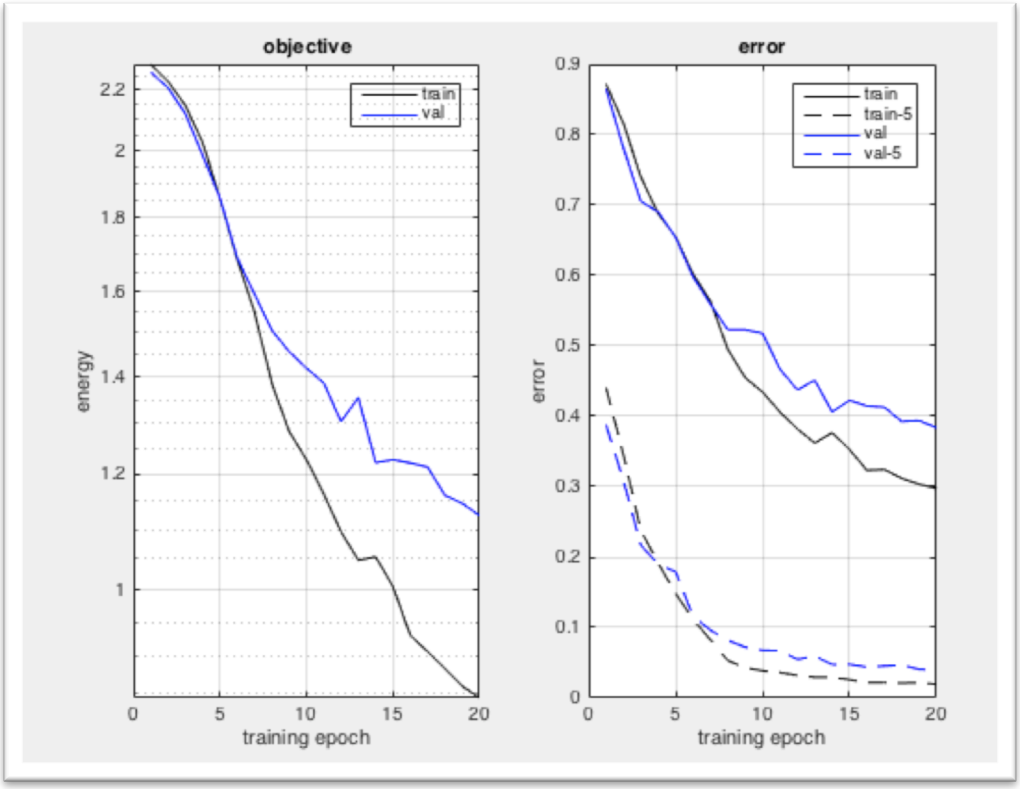
As we can clearly see from the error rates for without jitter, we can confidently say that this variation is the best amongst all three. The variation shows success rates of almost 100 with both training and validation sets. However ,there is one ambiguity with the error rate graph for with jittering. Here jitter has more error rate than without jittering. However, the jittering results for 7 are closer even after a wrong output.

---

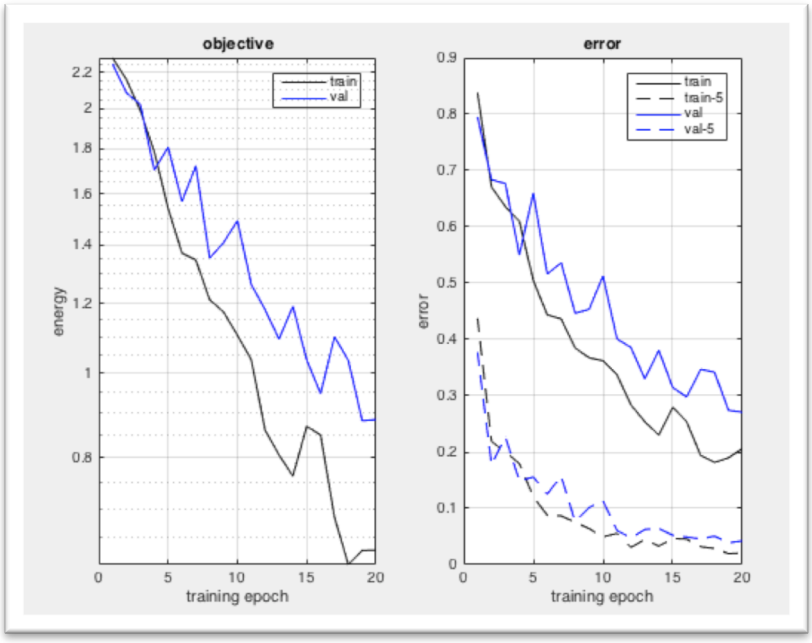
3) Variation2 layers:

We have used the following layer structure for this variation, in total 13 layers. We ran this for 20 epochs. These are the results.

conv->relu->pool->conv->relu->pool->relu->conv->relu->pool->relu->conv->softmax



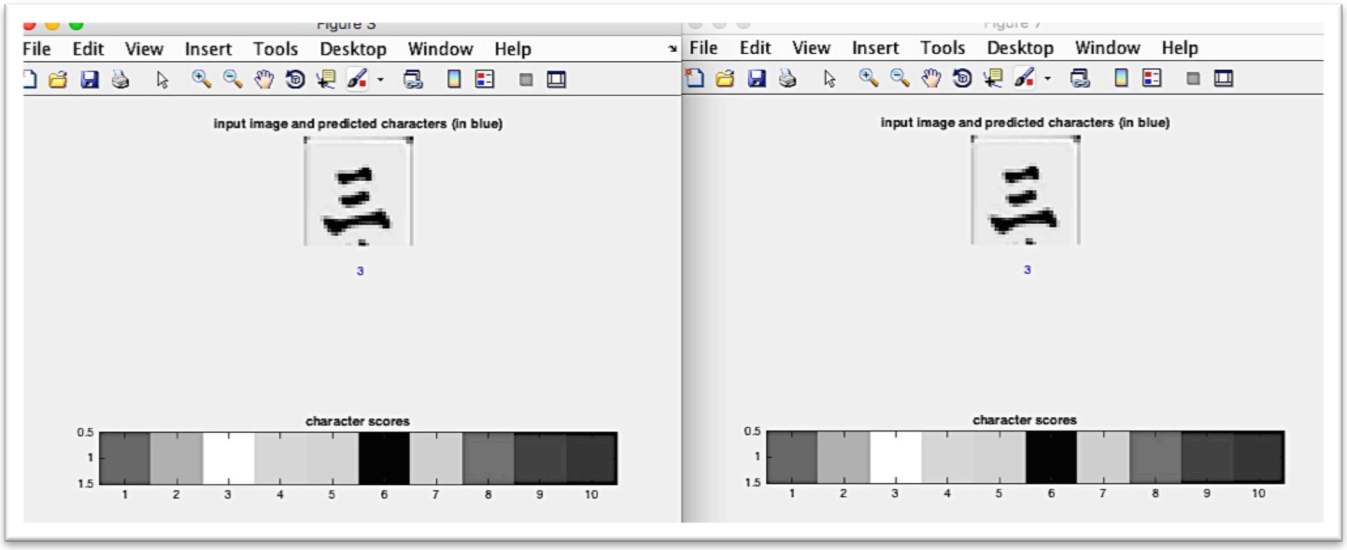
Error Rate for Variation 2 Without Jittering



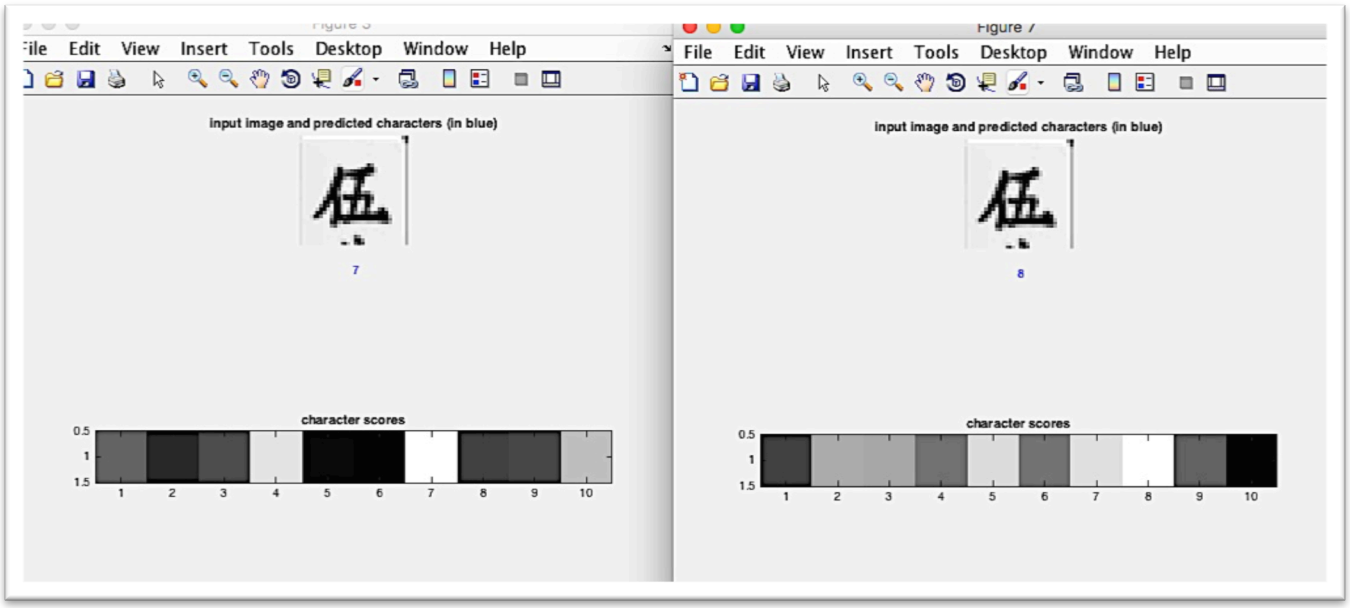
Error Rate for Variation 2 With Jittering

Observations for Variation2:

The observations to take from the following graph are two things, one is the training success rate and the other is the validation success rate. However when we compare this with the training on the part where we apply jittering , we can clearly say that our success rate for training increases. (70 to 80%). We can also see this from the two example results found below. The results for 3 are perfect, however, we can see that the scores, even if they are wrong for 5, are more biased towards 5 in the with Jittering figure.



Without and With Jittering for digit 3



Without and With Jittering for digit 5

Code details:

- **Project2.m**
  Extract Images and put into folders from 1 to 10.

- **ProcessImg.m**
  Put image in the database we have called it “chardatabase.mat”

- **Exercise4.m**
  Train and Test Data

- **initialconvolutioncnn.m:**
  Initial CNN compututaion

- **initialconvolutioncnn_change.m :**
  variation1

- **initialconvolutioncnn_change2.m:**
  variation2

- **decodeImage.m:**
  this file is used from the tutorial.

Conclusion:

Our Strategy in this project was to increase the number of layers. So we started with the initial 9 layer structure available to us from the Practical. Later on, we added more layers(2 Variations of 13 and 11). The observations can be formed from the error graphs. The success rate can be seen as high as 100% from the graphs, and we see that it is increasing then decreasing with the number of layers. However we believe that the best possible permutation of layers is not exactly related to the number, but with how the layers are arranged . For eg. Are they arranged **CONV->POOL->RELU or POOL->RELU->POOL** or any

such combinations. The best variation we got was variation1 as seen from the results. The second observation is related to the jitter factor. Better results (comparative) were obtained using jittering. There was one outlier (Stated in Variation1), however, jitter generally led to increase in success rate .

References:

[1] http://www.robots.ox.ac.uk/~vgg/practicals/cnn/
