# Innovation/Update

Aman Bhansali
B20ME010
bhansali.1@iitj.ac.in

Indian Institute of Technology Jodhpur

The major issue with the model was its time complexity as the model takes approximately 1 hour to run for each pass and say if convergence occurs after 50 passes the time complexity would approximately be 50 hours on a desktop, also accuracy of the model could be increased further and it could be made more robust/reliable for bad/fewer data. This may be due to because we are applying RNN directly on the raw pixel data due to which the model takes more passes to converge. As instructed we cannot change the features that is cannot remove noise/normalize the input image.

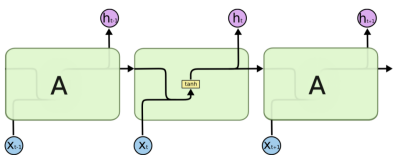
So, there could be two updates one is to decrease complexity and one for making the model work accurately for small data and data with noise. For, decreasing the time limit we can use a CNN network before the RNN network. This could help as CNN takes less time to process and using this we can extract good features vectors and feed it to the further RNN network. This way we would not be pre-processing the data directly now also we would be directly feeding the raw data. Also, as we discussed that we won't be doing any kind of segmentation so this would also follow the same and hence could be applied. We will be applying a fully connected convolution network. We will be applying a convolution network and at each step, we get the activation functions as a feature map then we use pooling and the same for many steps.

Though it may be thought that using an FCN before RNN would rather increase much more time but using this which is quite fast we would extract the most important features and then send it to MDRNN would take fewer passes and if passes decrease the time complexity would reduced by a significant amount.
Now, discussing in deeper this basically is a network that does convolution, pooling and at each step, it would try to reduce the size. By continuous pooling we would at the end get a single layer and now up-scaling it as per the input size would give good features. But, now say the pool we get at the last step we scale it and get a pool the same as the size of the previous one. Now, using this pool and the actual pool we could again get the layer that was present at that stage. Doing the same we would obtain some layers (here layers refer to as image dimension at a particular stage). Now, comparing these layers we get at each layer, we could get the best one. Now feeding this to our RNN model would take less time thus, one of our major problems could be solved.

Now discussing the other method in this we could make our model robust towards noisy images and also could get accuracy when our train data set is small. For this we will be doing the following steps:

1. As images contain pixel values that are used while applying any algorithms on them. The point is that I want my pixel values to be positive (explained further) and for this, I will be using the sigmoid function. Sigmoid function when applied on it would give a value between 0 and 1 and we will just multiply these values to 255 and get a new set of pixel values.

2. Now, we will apply RNN, LSTM as shown in the picture




   This $x_{t-1}, x_t and x_{t+1}$ would be passed through a sigmoid function when we come at their stage and then the result of sigmoid obtained again be multiplied with the original feature and then processed further.

3. Now, doing this we are going deeper into it and the better would be our classification that be it would get more accurate knowledge of the i/p data provided. Since we are passing it through the sigmoid we are basically converting it between 0 and 1 and then its multiplication would give a better result while moving further. This may increase our time complexity but would work for fewer train data and noisy images too.

   The logic behind this is the fact that the one which has larger value would get its pixel value scaled and it will show its value of presence it the data set. This would let the model understand in a much better way. Also, it could be done that is we use tanh and then take the average of the results and then work further on it. The purpose of changing the raw data to values between 0 to 255 is that if in case we use tanh it would get positive values and thus we would get values between 0 and 1.

   These could be the two approaches for the above two problems stated. Their combined effect could somewhat increase time but can be tried. Since applying it for me was a tough task on big data, also its huge complexity was an issue I believe that the model would perform better for the conditions mentioned. I applied the first approach on the MNIST data-set but the results obtained were not much convincing as this dataset was some straightforward so good results were not obtained.

References

1. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=9266005

2. https://ieeexplore.ieee.org/abstract/document/7814070/