Following is a submission on **Digit recognizer competition**:

Here's the link: https://www.kaggle.com/c/digit-recognizer

I could not upload the logs in the google form due to size constraints, so I have uploaded them to the drive and here's the link to the complete folder.

I have trained different models, compared them on TensorBoard. I have saved the logs of all the models I tried. I tried to explain what I understood from the graphs. I took this competition because of its dataset and I could train many models in less amount of time.

Here's what I did

- Loaded the dataset.
- Compared the number of different classes >> all classes are almost equally abundant in dataset.
- Normalizing the image, reshaping the image, converted the class labels to one-hot vectors.
- Splitting the training, validations data into 9:1.
- We started by randomly taking a network and training on it, with the help of this network we decided the number of epochs, to be used in next model.
- The previous model gave a minimum to the validation error values close to 8 epochs. So for next set of models we will take 8 epochs as standard.

- I then took a loop over conv_layer= [1,2,3], layer_size= [32,64,128], and dense_layer=[0,1,2];
- Comparing the scalars of these models in TensorBoard.
- On comparing models with 1, 2, 3 conv layers keeping rest same, 2conv layers performed best. When talking about performance, I'm right now considering validation error primarily and then validation accuracy.
- So we observed high bias in all 1 conv layer models, we noticed that 1 conv layer isn't enough to recognize the spatial features of the images.
- Also all the 3 conv layer models were having higher validation errors, and lower validation accuracy, but they were not saturating, so it is possible they might perform better on increasing the number of epochs.
- We also noticed that 32 layers of filter are not performing well compared to 64 and 128, and 0 dense layer is also giving higher validation error.

- These 4 model performed better compared to rest in this loop :
    - 2conv-64n-1dense; 2conv-64n-2dense; 2conv-128n-1dense; 2conv-128n-2dense;
    - 2conv-64n-1d: performs the best;
    - but it looks like that 2conv-64n-2d: also might perform better on increasing the number of epochs.

Again, I loop over conv_layer= [2, 3], layer_size= [64,128], and dense_layer= [1,2,3] but this time with 16 epochs.

- Initially we had a doubt that 3conv layer might perform better on increasing the number of epochs, it is now rejected as the 3conv layer model still performs poorly compared to the 2conv layer model.
- We observe that 3 dense layer also performs poorly. 1 or 2 dense layers are enough to find relation between the spatial features recognized by the conv layers.
- We are again are left with 4 models to compare:
    - 2conv-64n-1dense; 2conv-64n-2dense; 2conv-128n-1dense; 2conv-128n-2dense;
    - Order of Val loss:
    - 2conv-128n-2dense> 2conv-64n-2dense> 2conv-128n-1dense>~ 2conv-64n-1dense

- Now we are convinced that 2 conv layers are better compared to 1 and 3 conv layers. Also 32 filters are not sufficient.
- 2conv-128n-1dense and 2conv-64n-1dense emerge as the better models.
- Until now we were using 3x3 filter, again trying 2conv-128n-1dense and 2conv-64n-1dense with 5x5 layers.
- 2conv-128n-(5x5)-1dense outperforms all other tested models.

On submission the model gives an accuracy of 98.771%