**CNN-Assignment1\_PART A**

**GPU vs CPU**

We started running our models on local machines with no GPU support. Initially, we ran the model for 100 epochs and it took nearly 22 hours to complete. As we had AMD Radeon graphics on our machine, we tried using Radeon Open Compute (ROCm) which runs on Ubuntu. We wear not able to allocate GPUs on virtual machines and did not succeed in this. Later we ran the same experiment on AWS p2-xlarge instance which has GPU support and 75G of memory and it completed in 40 mins. We can clearly observe the difference between these two runs and conclude that these algorithms can run with much more speed on GPU machines.

**CALLBACK**

A callback is a set of functions to be applied at given stages of the training procedure. We can use callbacks to get a view on internal states and statistics of the model during training. You can pass a list of callbacks (as the keyword argument callbacks) to the .fit() method of the Sequential or Model classes. The relevant methods of the callbacks will then be called at each stage of the training.

Callback example:

keras.callbacks.Callback()

Arguments of the callback method

on\_epoch\_begin: called at the beginning of every epoch.

on\_epoch\_end: called at the end of every epoch.

on\_batch\_begin: called at the beginning of every batch.

on\_batch\_end: called at the end of every batch.

on\_train\_begin: called at the beginning of model training.

on\_train\_end: called at the end of model training.

There are 11 types of Callback functions in Keras

|  |  |
| --- | --- |
| TYPE | FUNCTION |
| BaseLogger | keras.callbacks.BaseLogger() |
| TerminateOnNaN | keras.callbacks.TerminateOnNaN() |
| ProgbarLogger | keras.callbacks.ProgbarLogger(count\_mode='samples') |
| History | keras.callbacks.History() |
| ModelCheckpoint | keras.callbacks.ModelCheckpoint(filepath, monitor='val\_loss', verbose=0, save\_best\_only=False, save\_weights\_only=False, mode='auto', period=1) |
| EarlyStopping | keras.callbacks.EarlyStopping(monitor='val\_loss', min\_delta=0, patience=0, verbose=0, mode='auto') |
| RemoteMonitor | keras.callbacks.RemoteMonitor(root='http://localhost:9000', path='/publish/epoch/end/', field='data', headers=None) |
| LearningRateScheduler | keras.callbacks.LearningRateScheduler(schedule, verbose=0) |
| ReduceLROnPlateau | keras.callbacks.ReduceLROnPlateau(monitor='val\_loss', factor=0.1, patience=10, verbose=0, mode='auto', epsilon=0.0001, cooldown=0, min\_lr=0) |
| CSVLogger | keras.callbacks.CSVLogger(filename, separator=',', append=False) |
| LambdaCallback | keras.callbacks.LambdaCallback(on\_epoch\_begin=None, on\_epoch\_end=None, on\_batch\_begin=None, on\_batch\_end=None, on\_train\_begin=None, on\_train\_end=None) |
| TensorBoard | keras.callbacks.TensorBoard(log\_dir='./logs', histogram\_freq=0, batch\_size=32, write\_graph=True, write\_grads=False, write\_images=False, embeddings\_freq=0, embeddings\_layer\_names=None, embeddings\_metadata=None) |

**CALLBACK IMPLEMENTATIONS:**

1. **CALLBACK()**

model\_info = model.fit(x\_train, y\_train,

batch\_size=batch\_size, nb\_epoch=epochs,

validation\_data = (x\_test, y\_test),

verbose=0, callbacks=[keras.callbacks.CallBack ()])

Output – We were able to add extra user control like logging after each Arguments

1. **BaseLogger()**

model\_info = model.fit(x\_train, y\_train,

batch\_size=batch\_size, nb\_epoch=epochs,

validation\_data = (x\_test, y\_test),

verbose=0, callbacks=[keras.callbacks.BaseLogger ()])

Output – This is the default callback function and did not show any change

1. **TerminateOnNAN()**

model\_info = model.fit(x\_train, y\_train,

batch\_size=batch\_size, nb\_epoch=epochs,

validation\_data = (x\_test, y\_test),

verbose=0, callbacks=[keras.callbacks.TerminateOnNAaN ()])

Output – It stops when loss of any epochs reaches zero. As in our model this does not happen there is no change in the output.

1. **Tensorboard()**

[TensorBoard](https://www.tensorflow.org/get_started/summaries_and_tensorboard) is a visualization tool provided with TensorFlow.

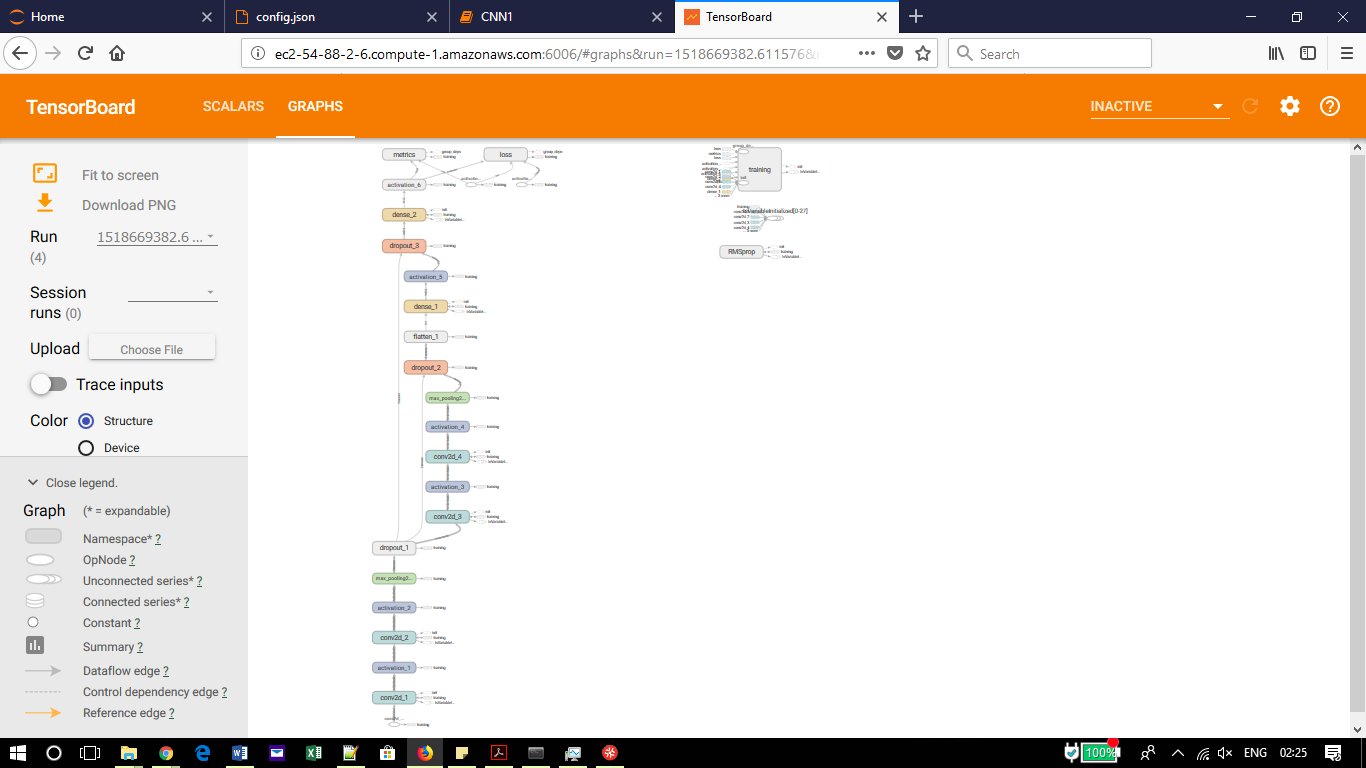
This callback writes a log for TensorBoard, which allows you to visualize dynamic graphs of your training and test metrics, as well as activation histograms for the different layers in your model.

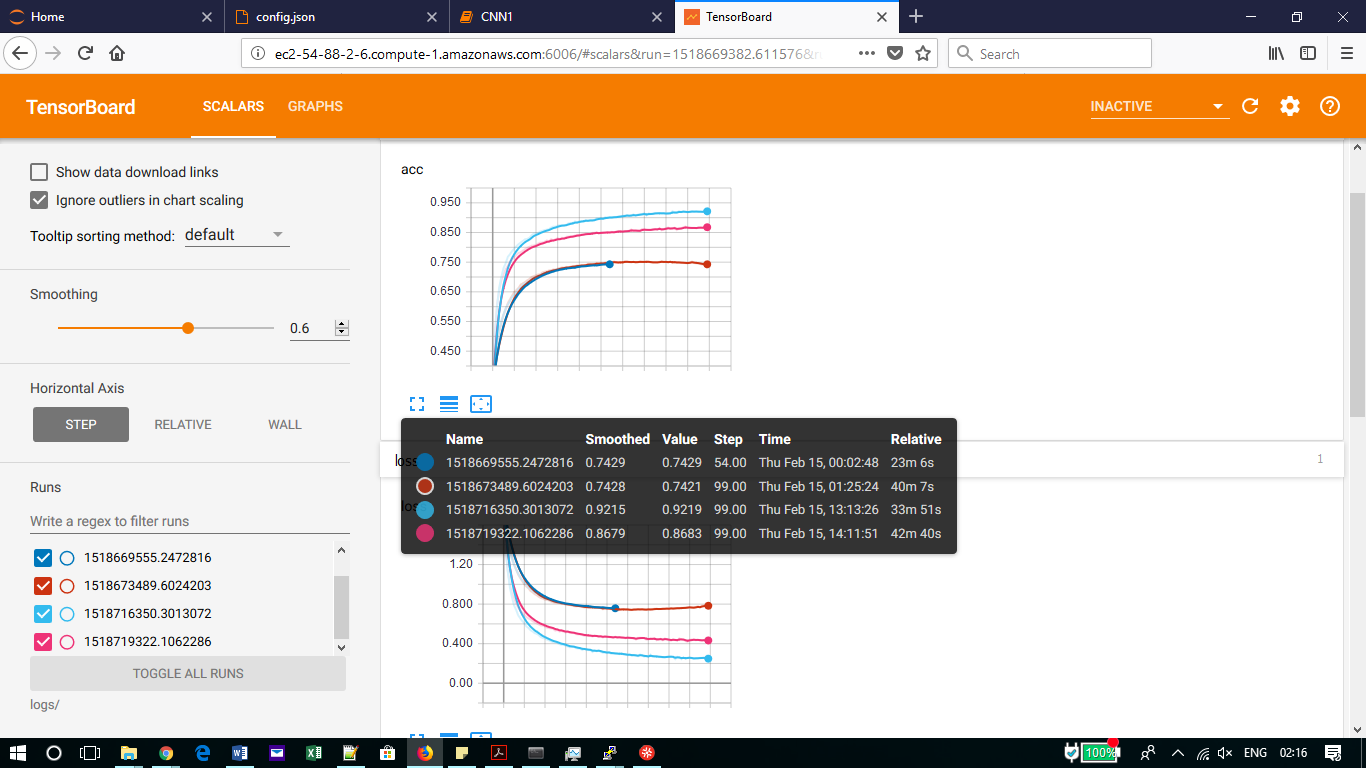
If you have installed TensorFlow with pip, you should be able to launch TensorBoard from the command line:

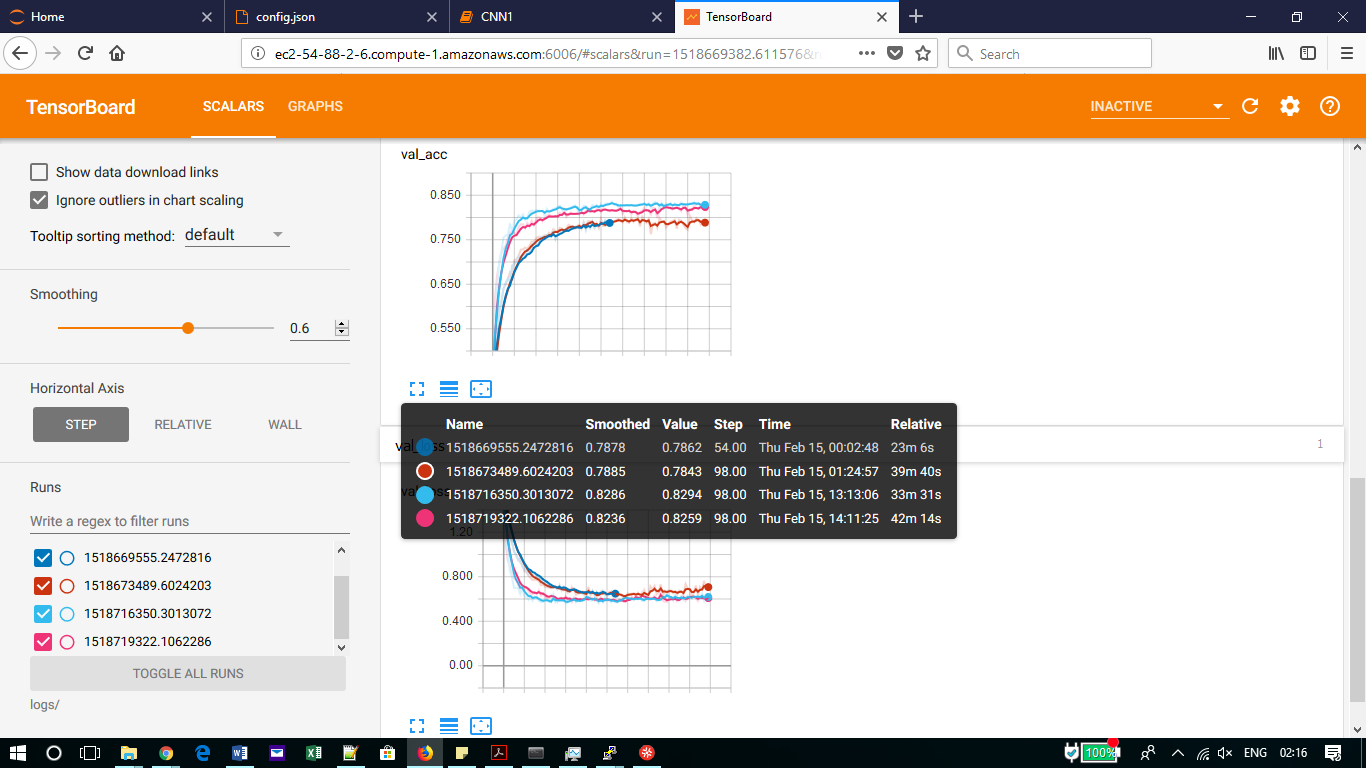
tensorboard --logdir=/full\_path\_to\_your\_logs

tensorboard = TensorBoard(log\_dir="logs/{}".format(time()),histogram\_freq=0,write\_graph=True, write\_images=True)

Output: A new visualization window opens where we can see the progress of CNN according to each epoc. Below are the screen shots of the same:







**Conclusion**

We have tried different model with different parameters and as we can see that the blue is the best model.