# SI2021


***MNIST and Tensorboard hands on***

***Paul Rodriguez, PhD***
***SDSC***

# Overview: MNIST w/noise, and tensorboard

MNIST is too easy, so I switched labels to make learning harder

Tasks:

1 Try adding L1 regularization in classification layer (a penalty $\boldsymbol{\lambda}^{*}\boldsymbol{\Sigma}$ |W| is added to Loss)

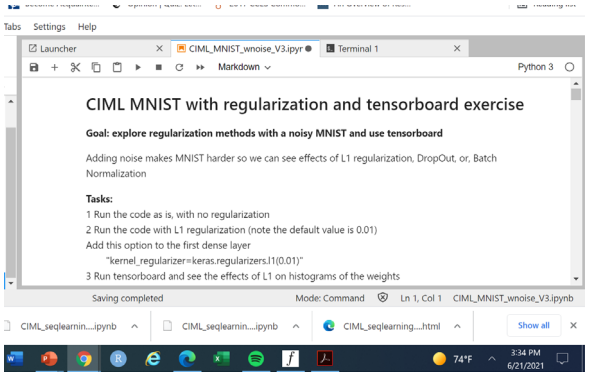
2 View results in tensorboard

Optional, try adding these and comparing performance in tensorboard:

dropout layer (drop % of activations of a layer)

batchNormalization (center and scale activations of a layer)

1 Read the instructions and glance at the code sections




Notice there is a “logs” directory set up for tensorflow

Notice there is a new callback function to log information

Next cell sets up directory for tensorboard logs

```
[1]: #Set up the location for tensorflow logs
import datetime, os
logdir  = os.path.join("logs",
                datetime.datetime.now().strftime("%Y%m%d-%H%M%S")+'_'+ \
                       'test')   #<<<<<----- you can add a comment to describe the test
print('using',logdir,' for logs')
```

```
5]: from tensorflow.keras.callbacks import EarlyStopping
ES_function = EarlyStopping(monitor='val_loss', mode='min',  patience=20)
                    #or try: monitor='val_accuracy', mode='max', patience=1)

tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)
      #Use this to get performance info, like memory usage:  profile_batch = '5,20')

fit_history=mymodel.fit(X_train, Y_train, #validation_split=0.20,
          validation_data=(X_test,Y_test),
          batch_size=64, epochs=100, verbose=1,callbacks=[ES_function,tensorboard_callback])
```

2 Run notebook , following instructions

On Expanse, we set up tensorboard to use reverse proxy as follows:

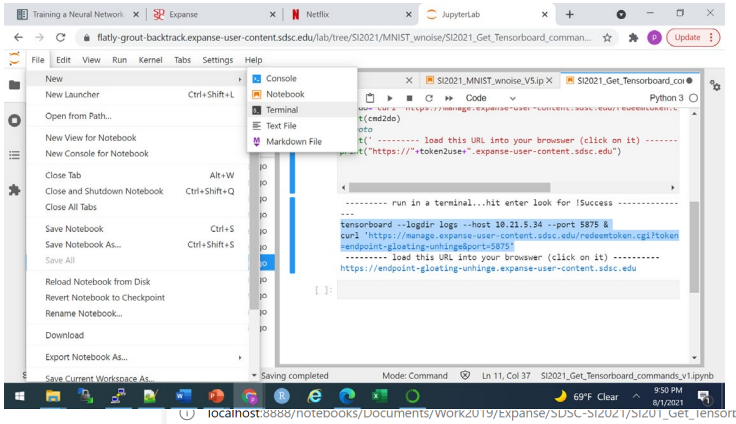
1. Open a terminal window, cd to SI2021 4.1b DL session

2. Open SI2021_Get_Tensorboard_commands notebook and run it

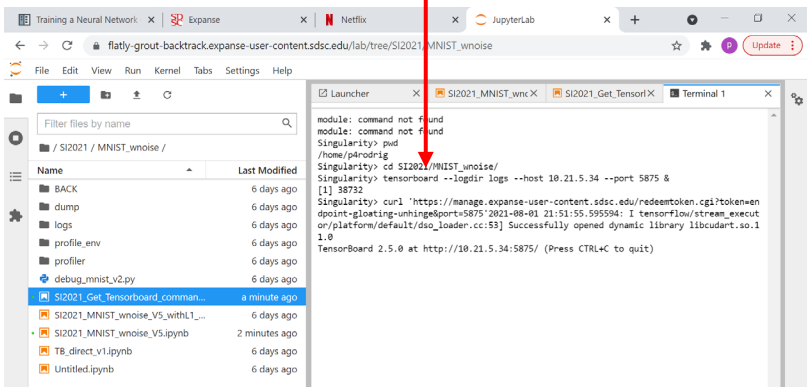
It should print out a ‘tensorboard’ command and ‘curl’ command, and a URL

3. cut & paste commands into terminal window

4. Click on URL



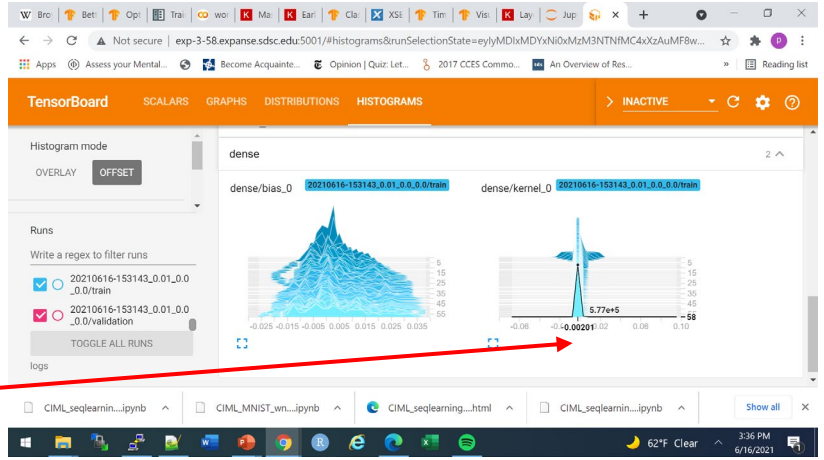

cut & paste



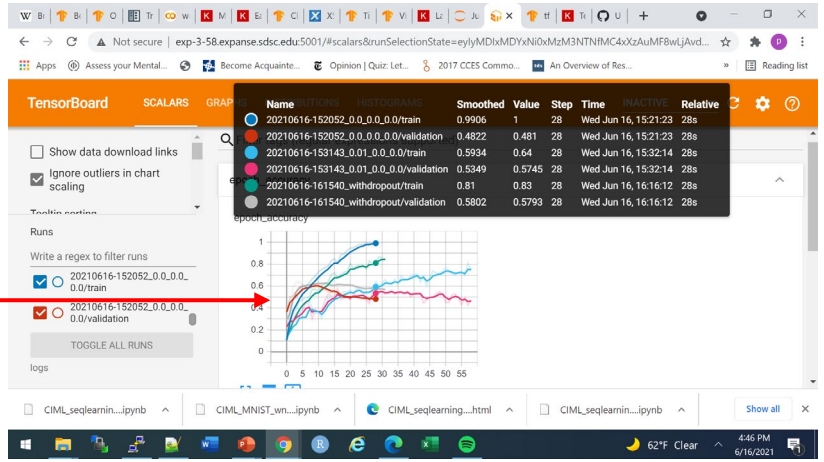
Run the notebook;

1 Find the <<<< ----  comments to change/add regularization

2 In tensorboard, toggle the runs to just see the  logs with L1 regularization, then check out the weight histograms dense layer




3
If you run several times you can get all the performances plotted together (depending on what log data you toggle )



NOTE on optional tasks:

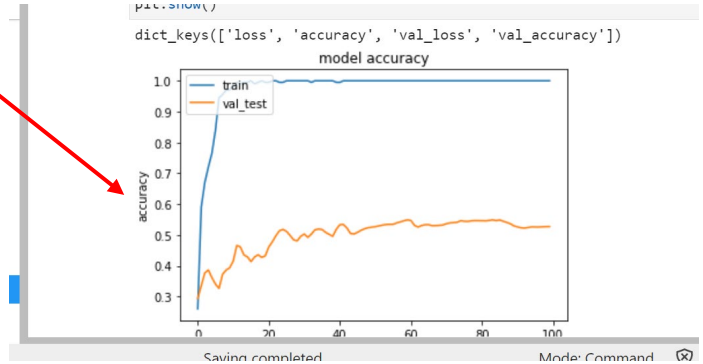
BatchNormaliztion seems to learn the training set very fast and avoid the inverted U for accuracy

```
numfilters = 64
mymodel.add(Convolution2D(numfilters, (3,3), strides=1,  data_format="channels_last", activation='relu'
mymodel.add(Convolution2D(numfilters, (3,3), strides=1,  data_format="channels_last", activation='relu'
mymodel.add(MaxPooling2D(pool_size=(2,2),strides=2,data_format="channels_last"))
mymodel.add(Flatten())

#----- add final classification layers
mymodel.add(Dense(64, activation='relu'))   #<<<<---- Add the L1 reglzer option here
#mymodel.add(Dropout(0.50))
mymodel.add(BatchNormalization(axis=-1))

mymodel.add(Dense(10, activation='softmax'))

print('added layers to model')
```



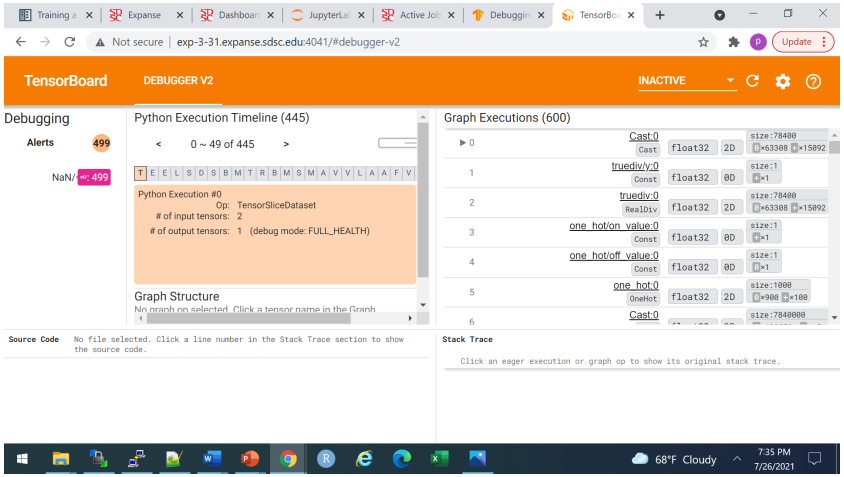
PAUSE

EXTRA NOTE:

Tensorboard has a ‘debugger’ option

The debugger will ‘alert’ you to errors, such as NaNs in you calculations.  Most useful if you are building custom functions.

It requires a command in your Python code to ‘dump’ information into the logs




tf.debugging.experimental.enable_dump_debug_info( ….

EXTRA NOTE:

Tensorboard has a ‘profiler’ plugin

The profile option has performance information for some part of the training iterations

On Expanse it requires installation of virtual environment (ask for details if interested)