

CIML

***MNIST and Tensorboard hands on
Paul Rodriguez, PhD
SDSC***



Overview: MNIST w/noise, and tensorboard

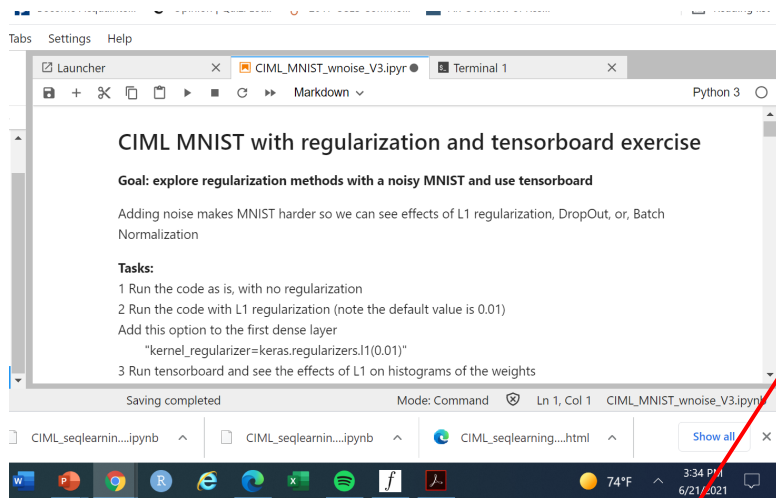
MNIST is too easy, so switch labels to make learning harder

Try an L1 regularization (a penalty $\lambda * \sum |W|$ is added to Loss) on the classification layer

View results in tensorboard

Optional: dropout layer (drop % of activations of a layer)

batchNormalization (center and scale activations of a layer)



Read the instructions and glance at the code sections

Notice there is a log directory name 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')
```

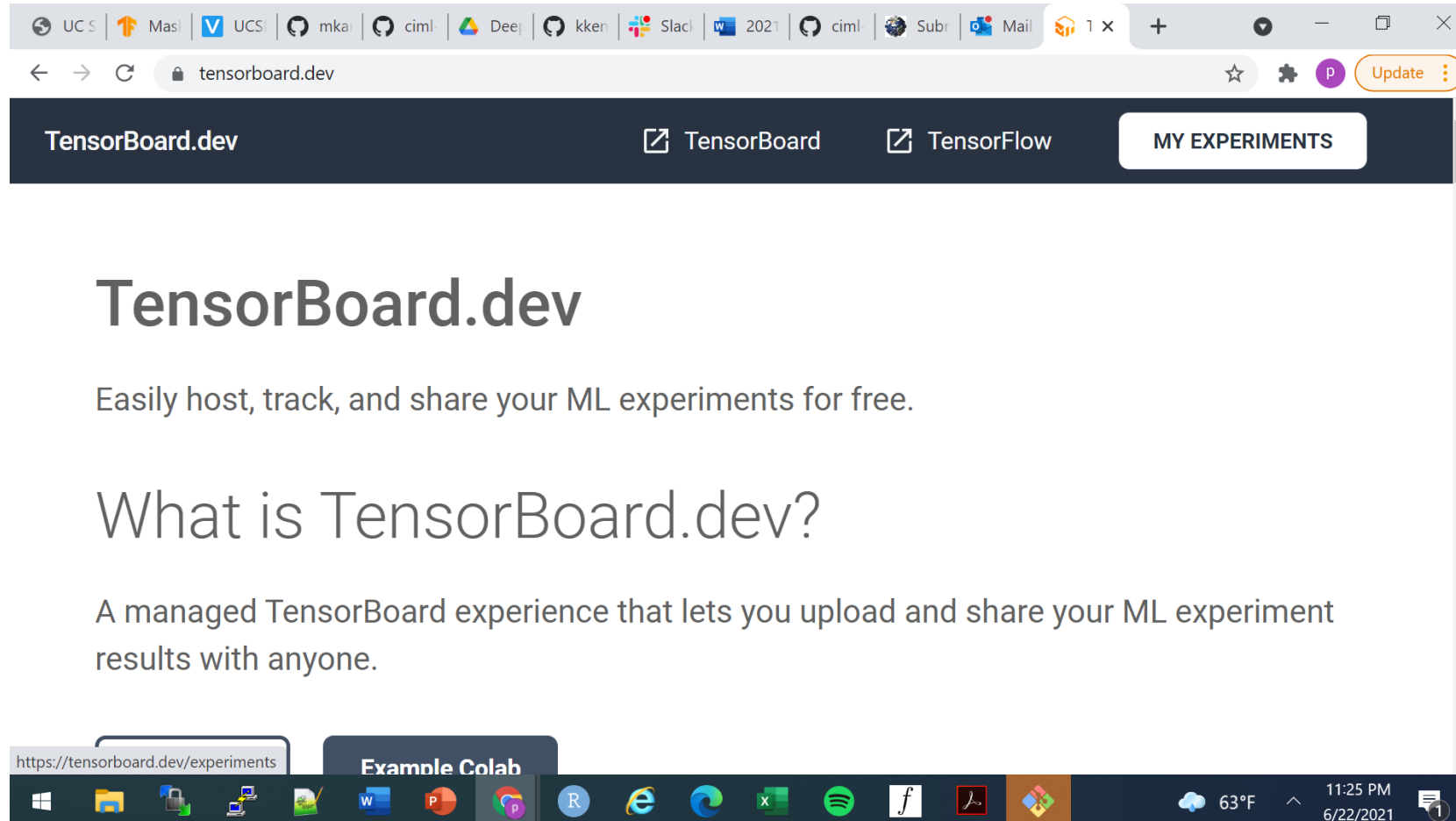
```
5]: from tensorflow.keras.callbacks import EarlyStopping
ES_function = EarlyStopping(monitor='val_loss', mode='min', patience=20)
#or try: monitor='val_accuracy', mode='max', patience=1)

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])
```

Tensorboard has no reverse-proxy on Expanse, so we can use public tensorboard

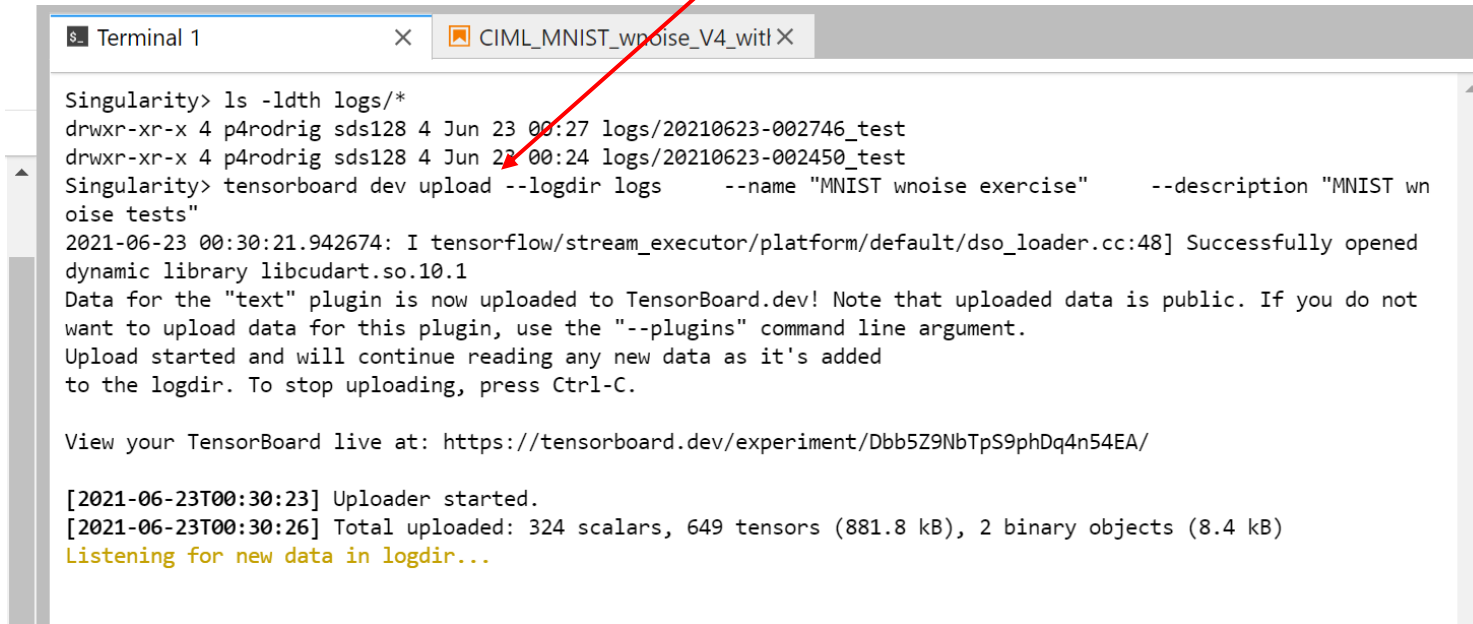
(In future, we will get a more secure set up on Expanse)



From a terminal window (inside the notebook) , upload log files to tensorboard.dev

```
$ tensorboard dev upload --logdir logs \  
  --name "(optional) My latest experiment" \  
  --description "(optional) Simple comparison of several hyperparameters"
```

go to folder with log directory and enter this command



The image shows a terminal window with two tabs: 'Terminal 1' and 'CIML_MNIST_wnoise_V4_witl'. The terminal output shows the user running 'ls -ldth logs/*' which lists two log directories. Then, the user runs the 'tensorboard dev upload' command with the specified name and description. The terminal shows a successful upload message from TensorFlow, indicating that data is now being uploaded to TensorBoard.dev. It also provides a URL to view the experiments live. Finally, it shows status messages from the uploader, including the total amount of data uploaded (324 scalars, 649 tensors, and 2 binary objects) and a message stating it is listening for new data.

```
Singularity> ls -ldth logs/*  
drwxr-xr-x 4 p4rodrig sds128 4 Jun 23 00:27 logs/20210623-002746_test  
drwxr-xr-x 4 p4rodrig sds128 4 Jun 23 00:24 logs/20210623-002450_test  
Singularity> tensorboard dev upload --logdir logs      --name "MNIST wnoise exercise"      --description "MNIST wnoise tests"  
2021-06-23 00:30:21.942674: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcudart.so.10.1  
Data for the "text" plugin is now uploaded to TensorBoard.dev! Note that uploaded data is public. If you do not want to upload data for this plugin, use the "--plugins" command line argument.  
Upload started and will continue reading any new data as it's added to the logdir. To stop uploading, press Ctrl-C.  
  
View your TensorBoard live at: https://tensorboard.dev/experiment/Dbb5Z9NbTpS9phDq4n54EA/  
  
[2021-06-23T00:30:23] Uploader started.  
[2021-06-23T00:30:26] Total uploaded: 324 scalars, 649 tensors (881.8 kB), 2 binary objects (8.4 kB)  
Listening for new data in logdir...
```

Also use these to clean up uploaded experiments:

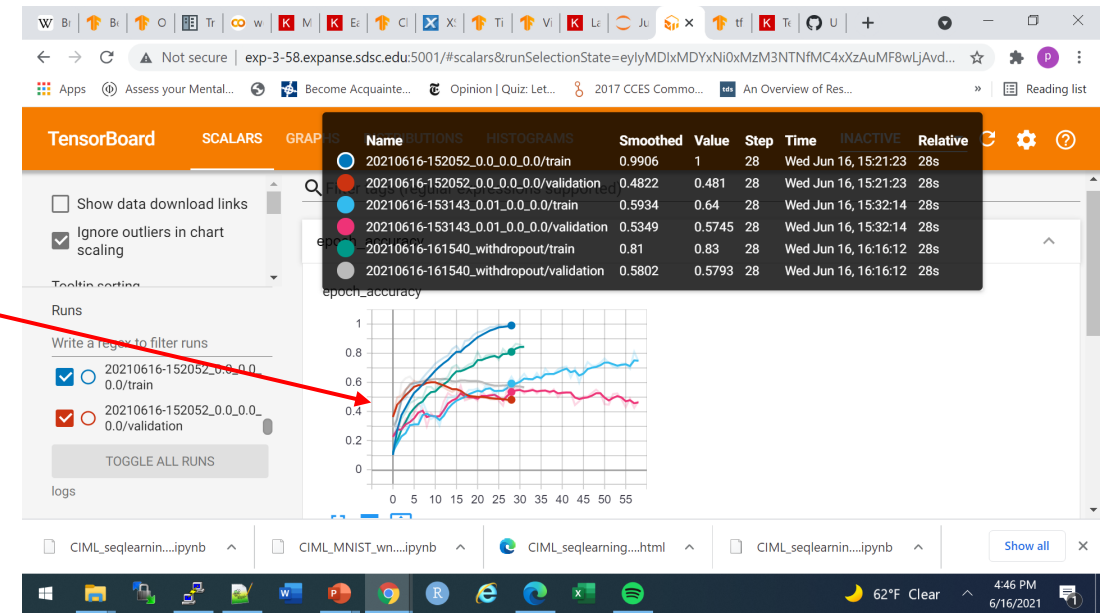
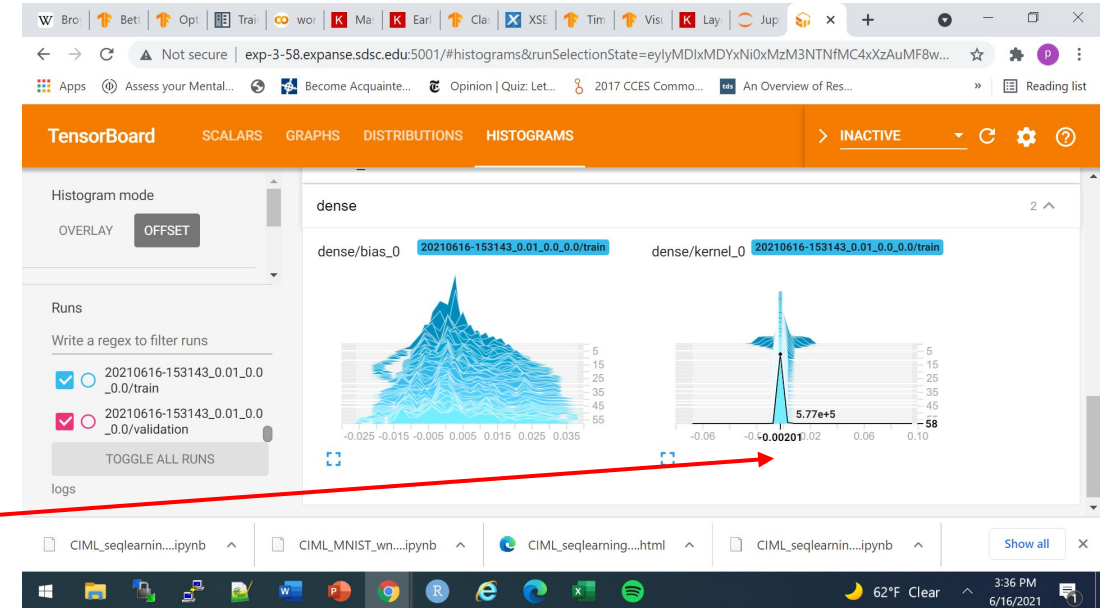
```
$tensorboard dev list
```

```
$tensorboard dev delete --experiment_id GdEEZ0NBQaGdOjqXeijwLw
```


Run the notebook;
Find the <<<< ---- comments to
change/add regularization

In tensorboard, for L1 regularization
logs the weight histogram will be
mostly 0s – right?
(try toggling runs on/off, look for dense
layer)

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



NOTE on extra tasks:

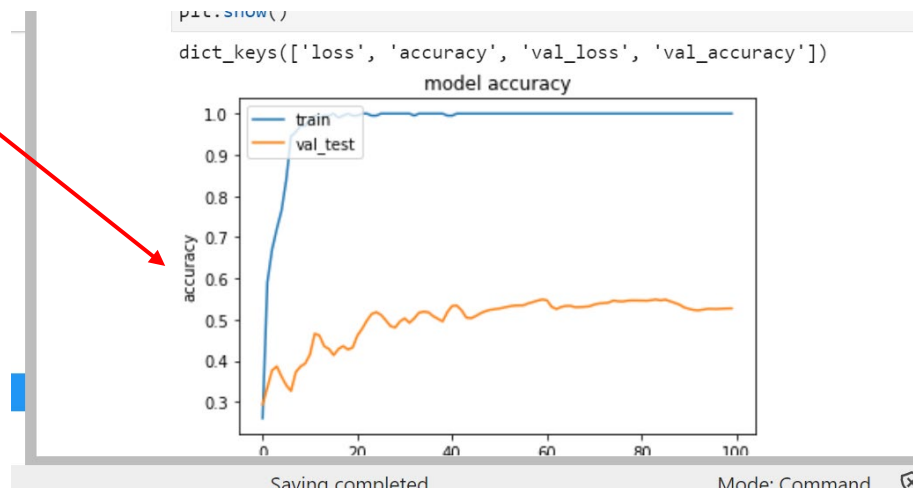
BatchNormalization seems to learn the training set very fast and avoid the inverted U for accuracy

```
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')
```



EXTRA NOTE:

Tensorboard has a 'profiler' plugin

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

On Expanse it does not run in the container – yet.

