



# Week 2 Progress Report

Presenter - Matt Behnke

Mentor - Dr. Guo



## Tools that I started with

- Python 3 & Jupyter Notebook
- Tensorflow + Keras
- Pandas
- Sklearn

```
In [3]: # The libraries that I am using
import sys
import sklearn
import keras
import pandas
import numpy
import os

print('Python: {}'.format(sys.version))
print("Keras: {}".format(keras.__version__))
print("Pandas: {}".format(pandas.__version__))
print("Sklearn: {}".format(sklearn.__version__))
print("Numpy: {}".format(numpy.__version__))
```

Using TensorFlow backend.



## Reading in Data using Porosity\_IndexData2.csv

- Used Porosity\_IndexData2.csv to map the Status (1 = bad, 0 = good) to the files in the porosity datasets
- First converted to csv file (primarily for my ease of use)
- Added Desc, Status, & Status2 so each attribute is labeled
- However, ultimately disregarded Desc, size(mm) & Status2
  - Status2 was the same as status, so I chose to disregard it as it was redundant
  - Desc is just a nominal description of status, which is good for readability of new users, but not integral for a CNN
  - size(mm) seemed to be an extra qualifier for bad statuses, since it did not applied to good I did not include it because then good models would be training with a NaN data

Index	T	TP	X	Y	YP	Z	ZP	Layer	Desc	Status	size(mm)	Status2
1	0	0	0	0	0	0	0	1	bad	1	0.4185	1
2	0	1524	0	1	936	0	0	1	bad	1	0.3316	1
3	0	3049	0	3	872	0	0	1	bad	1	0.3418	1



## Reading in Data using Porosity\_IndexData2.csv (Cont.)

- Creating a mapping from a porosity dataset's filename to it's status
- This title name will later be replaced by the porosity dataset's corresponding data

```
# Create a title from data points in Porosity File
def createTitleList(t, tp, x, y, yp, z, zp, layer, status):
    if(str(z) == '0'):
        title = "t" + str(t) + "p" + str(tp) + "_x" + str(x) + "_y" + str(y) + "p" + str(yp) + "_z" + str(z) + "_layer" + str(layer)
    else:
        title = "t" + str(t) + "p" + str(tp) + "_x" + str(x) + "_y" + str(y) + "p" + str(yp) + "_z" + str(z) + "p" + str(zp) + "_" + str(layer)
    titleStatusList[title] = status
```

```
# Create every title from porosity file and map to it's status/result
for index in range(rows):
    createTitleList(PorosityFile.loc[index, 'T'], PorosityFile.loc[index, 'TP'], PorosityFile.loc[index, 'X'], PorosityFile.loc[index, 'Y'], PorosityFile.loc[index, 'Z'], PorosityFile.loc[index, 'ZP'], PorosityFile.loc[index, 'L'], PorosityFile.loc[index, 'S'])
```



# Reading in 300W-30ipm-4rpm\_CSV\_Pyro - converted

1. Iterate through every file in the directory and record filename
2. Using the filename, try in index the map and get its corresponding status
  - a. If the index does not exist, raise a error, that instead adds to a list containing files that did not correspond to the earlier function creating filenames
3. If the filename exists in the map, check if it's status is either 1 or 0
  - a. If the status is either 1 or 0
    - i. Add to inData and OutData (the data to use for training and predictions)
  - b. Otherwise
    - i. Note that the file has an 'invalid' status, ignore it, but print which file it was with its value to the console

```
Here are the files not found, 382 in total
t0p0000_0_0p0000_0_layer1.csv
t100p5_x0_y58p08_z6p12_layer13.csv
t104p0_x0_y0p0000_z6p63_layer14.csv
t106p2_x0_y29p04_z6p63_layer14.csv
```

```
** Invalid Label value of -1, skipping to next value **
Encoding t297p2_x0_y0p0000_z19p38_layer39.csv...
Encoded file t297p2_x0_y0p0000_z19p38_layer39.csv
```



# Standardizing Data using SKLearn

- Puts data within a minimum and maximum size
  - Typically between 1 and 0
- Just so that data is scaled to unit size
- Helps normally distribute the data
- I used this in a prior Neural Network project, so I thought I would apply it again

```
# Standardize Data
from sklearn.preprocessing import StandardScaler
print(inData[5:6])
print(inData.shape)
scalers = {}
for i in range(inData.shape[1]):
    scalers[i] = StandardScaler()
    inData[:, i, :] = scalers[i].fit_transform(inData[:, i, :])
```



# Splitting Test & Train Data

- Typically just splitting data so a model does not become overfitted to its own dataset
  - Using it's own data not trained on so it has data it can trial on and determine accuracy
  - Train (80%), Test (10%), Valid (10%)
    - Not much significance, just arbitrary values that can be further tuned later
- Then I started running into problems =(

```
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(inData, outData, test_size = 0.2)
X_val, X_test, y_val, y_test = train_test_split(X_val, y_val, test_size = 0.5)
X_train.shape
```

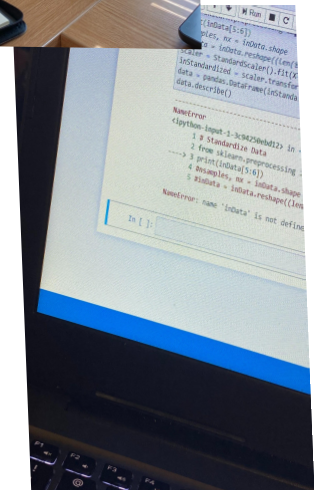
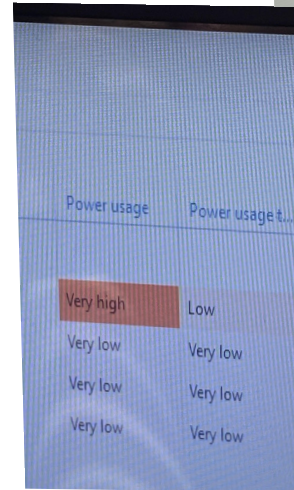
```
(940, 479, 753)
```

# Splitting Test & Train Data (Cont.)

```
X_train.shape
```

```
(940, 479, 753)
```

- 940 Instances of matrices of 479 Rows x 753 Columns
  - Hence, 1 Instance has 360,687 data points (yikes)
  - So the dataset in whole has 339,045,780 data points (yikes.. again)
- This caused my laptop to not be too happy...





# Building Model (After some successful loads)

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Activation, Dropout, BatchNormalization, Flatten
from keras.wrappers.scikit_learn import KerasClassifier
from keras.optimizers import Adam
def initializeModel():
    # Get sequential keras model
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(479, 753, 1)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

```
model = initializeModel()
```

```
import numpy as np
X_train2 = np.expand_dims(X_train, -1)
y_train2 = np.expand_dims(y_train, -1)
#X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))
#X_val = np.reshape(X_val, (X_val.shape[0], 1, X_val.shape[1]))
history = model.fit(X_train2, y_train2,
                    epochs=10,
                    verbose=1,
                    )
```

- Very generic model, just to try and get the data to begin to train
- Occasionally ran, but had same errors (performance) when trying to train


```
In [ ]: import numpy as np
X_train = np.expand_dims(X_train, -1)
y_train = np.expand_dims(y_train, -1)
#X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))
#X_val = np.reshape(X_val, (X_val.shape[0], 1, X_val.shape[1]))
history = model.fit(X_train, y_train,
                    epochs=10,
                    verbose=1,
                    )
```

WARNING:tensorflow:From C:\Users\behn\anaconda3\lib\site-packages\keras\backend\tensorflow\_backend.py:422: The name tf.global\_variables is deprecated. Please use tf.compat.v1.global\_variables instead.

Epoch 1/10



# Moving Forward / Questions (That I have for you and you have for me =) )

- Using google colab 
  - Cloud Based, which could help fight some of the performance/data issues I have with my laptop
  - Also has jupyter notebook formatting, so it won't be too much a re-work (I hope)
- Trying to be more clever/creative with how I am attempting to train the data
- Goals next week (tentative)
  - Have model training and tested with different types of models for comparison
  - Involves figuring out performance problems (proving to be a big hurdle)
  - Getting files that were excluded on naming parameters back into dataset
- Any campus machines to putty into (I have other friends at university they have done this with, for fast training)
- Any other suggestions for moving forward to training a model with such data