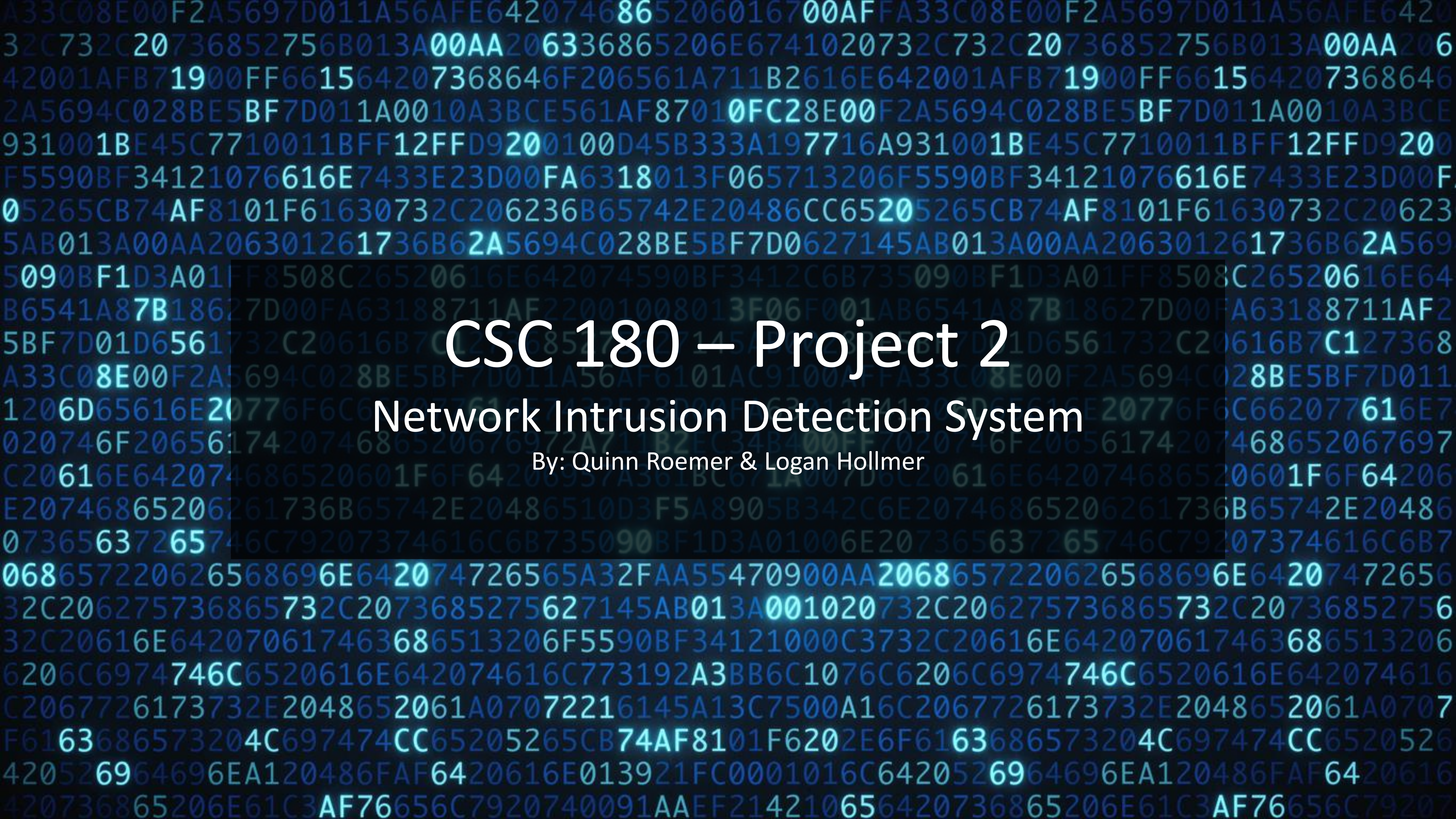
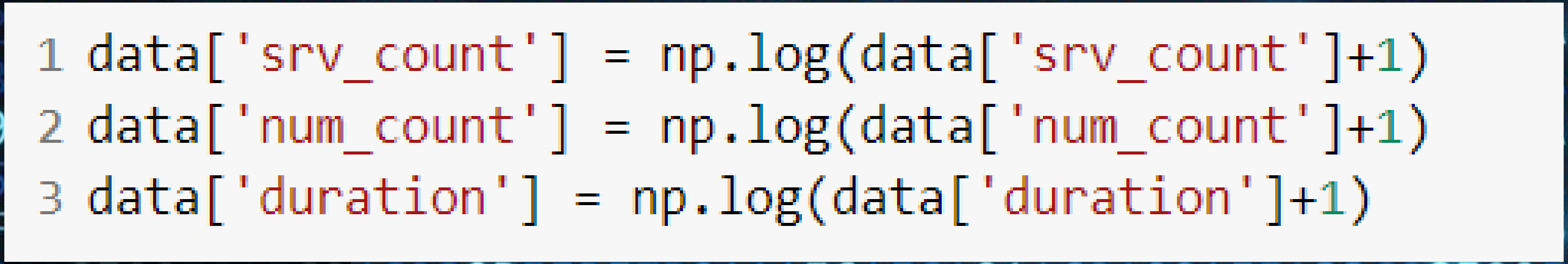


CSC 180 – Project 2

Network Intrusion Detection System

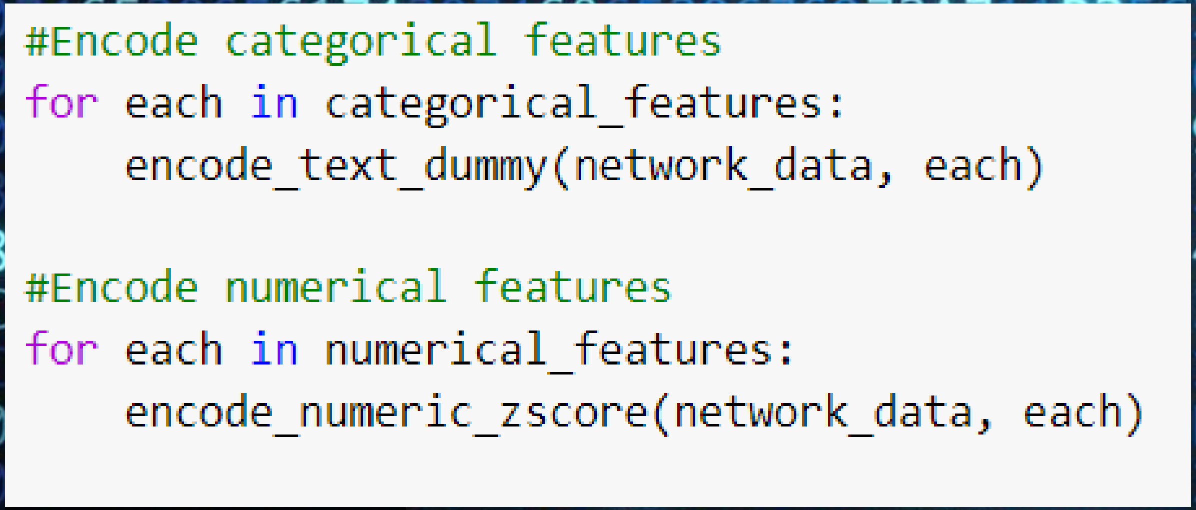
By: Quinn Roemer & Logan Hollmer

Model/Code Design



```
data['srv_count'] = np.log(data['srv_count']+1)
data['num_count'] = np.log(data['num_count']+1)
data['duration'] = np.log(data['duration']+1)
```

Logan's Code



```
#Encode categorical features
for each in categorical_features:
    encode_text_dummy(network_data, each)

#Encode numerical features
for each in numerical_features:
    encode_numeric_zscore(network_data, each)
```

Quinn's Code

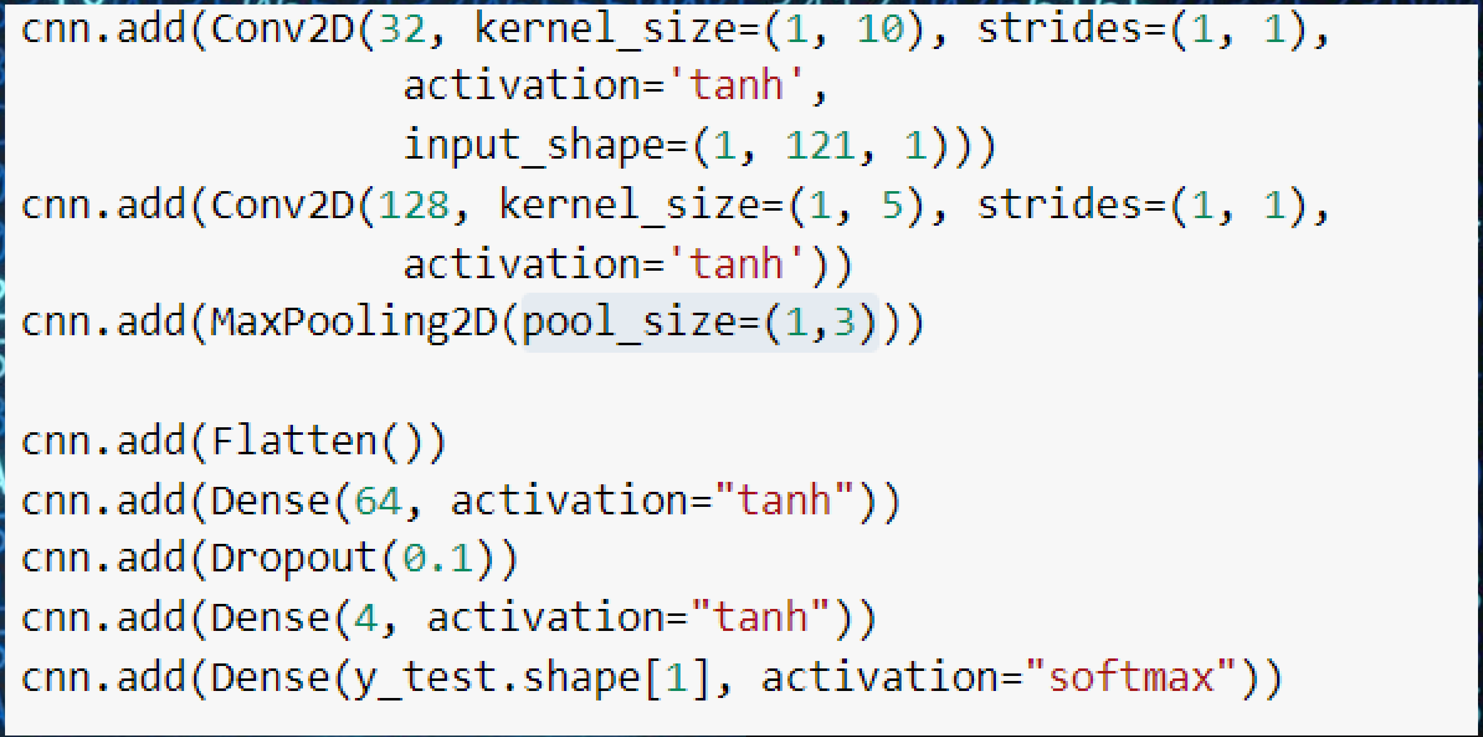
Data/Code Differences

- 10% dataset vs 100% dataset
- Log normalization into z-score vs z-score

Model/Code Design

Model differences

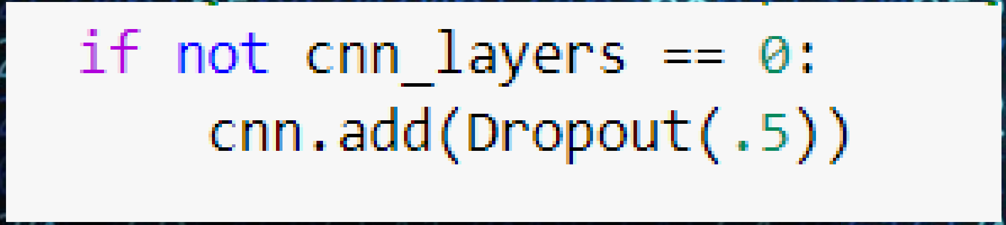
- Quinn trained CNN and fully connected networks
- Logan train CNN, fully connected networks, fully connected networks using regularization, and fully connected networks using featured from logistic regression.
- Quinn created more dropout layers in his CNN networks.
- Logan created more diverse CNN networks styles.



```
cnn.add(Conv2D(32, kernel_size=(1, 10), strides=(1, 1),
                activation='tanh',
                input_shape=(1, 121, 1)))
cnn.add(Conv2D(128, kernel_size=(1, 5), strides=(1, 1),
                activation='tanh'))
cnn.add(MaxPooling2D(pool_size=(1,3)))

cnn.add(Flatten())
cnn.add(Dense(64, activation="tanh"))
cnn.add(Dropout(0.1))
cnn.add(Dense(4, activation="tanh"))
cnn.add(Dense(y_test.shape[1], activation="softmax"))
```

One of Logan's CNN models

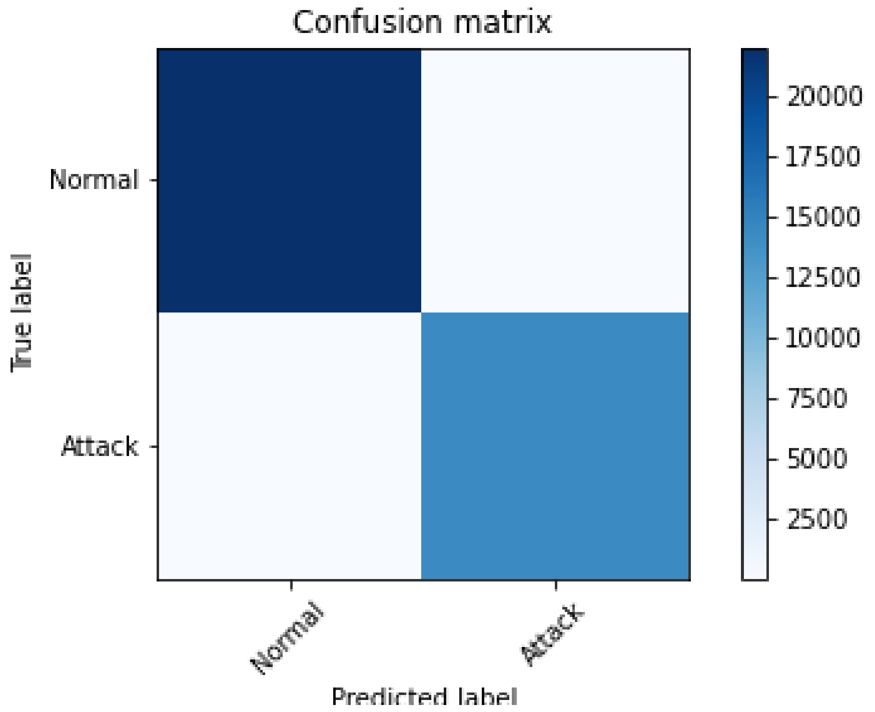


```
if not cnn_layers == 0:
    cnn.add(Dropout(.5))
```

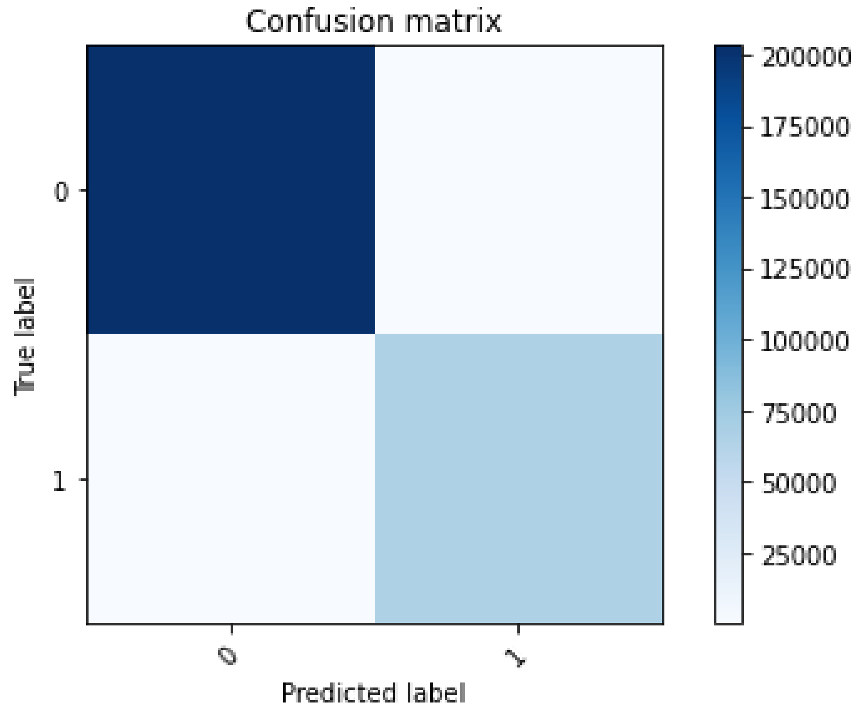
Quinn's additional dropout layers

Findings/Results

Quinn’s top model: F1 score: 0.99871



Logan’s top model: F1 score: 0.99961



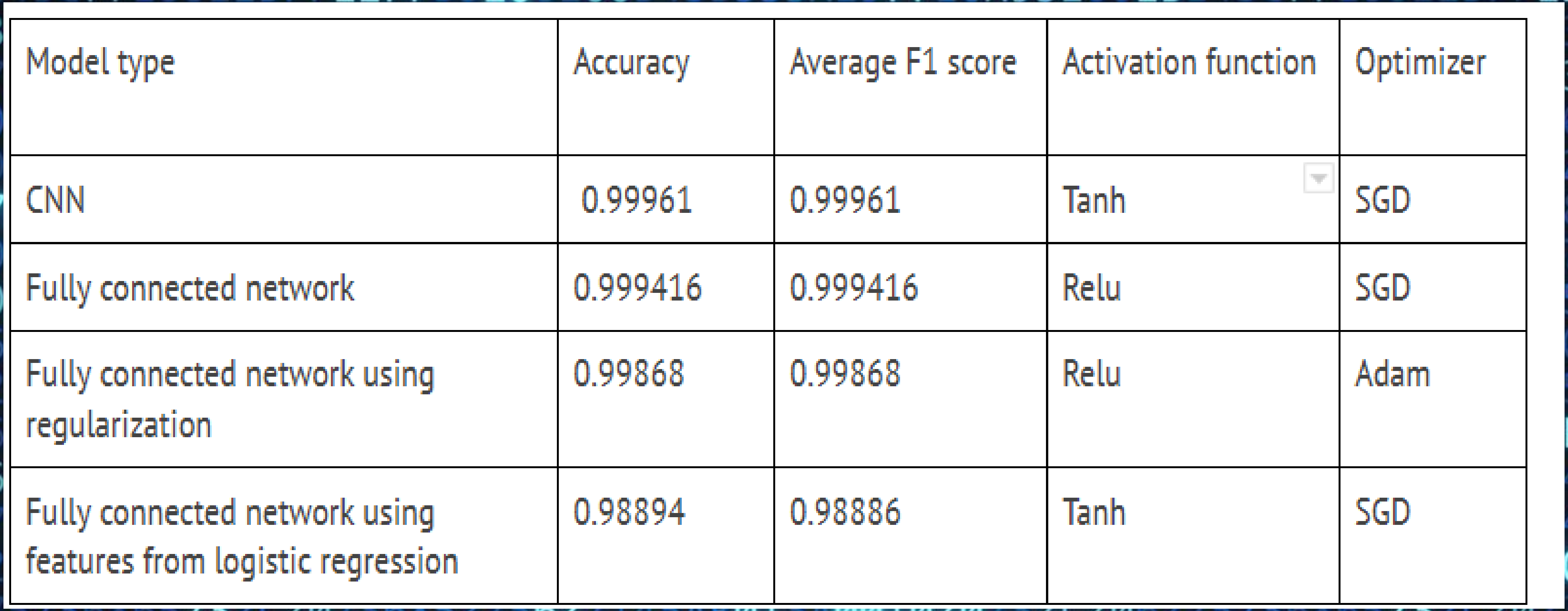
Model Settings

- 5 hidden layers decreasing by a power of 2 from 64 to 4
- Optimizer: Adam
- Activation function: sigmoid (on all layers except the last)

Model Settings

- Convo layer, 32 features, kernel_size=(1, 10) and strides=(1, 1)
- Convo layer, 128 features, kernel_size=(1, 5) and strides=(1, 1)
- Max pooling layer with pool_size=(1,3)
- Flatten() into a dense layer with 64 neurons
- Dropout layer set to 0.1, into dense layer with 4 neurons.

Findings/Results



Model type	Accuracy	Average F1 score	Activation function	Optimizer
CNN	0.99961	0.99961	Tanh	SGD
Fully connected network	0.999416	0.999416	Relu	SGD
Fully connected network using regularization	0.99868	0.99868	Relu	Adam
Fully connected network using features from logistic regression	0.98894	0.98886	Tanh	SGD

Logan’s top model from each network type

Task Division, Challenges Encountered, and what we Learned

Task Division

- Both group members created fully functional programs to pre-process data, train, and test our models.
- Quinn created the majority of the report and worked on the notebook.
- Logan created the presentation, report, and created the base of the notebook.

Challenges Encountered

- Training time was a problem we ran into, especially for Logan who used the whole dataset. With training in the upwards of 2 hours for CNN models. This was solved by creating automatic scripts to train overnight.
- We struggled to find the best architecture especially for the CNN models. We tried both using established model such as ones used for MNIST. But we also tried custom models inspired by existing techniques.

What we Learned

- More data is a good thing! Using the larger dataset seemed to be a big factor.
- Sometimes more than 1 normalization technique can be used.
- Different types of data often call for different normalization techniques.
- There is a lot a variation to CNN models and some “unique” models can perform very well.