HOMEWORK 5
Kelly “Scott” Sims

# *Q1.2b*

Trainable Parameters

- Layer 1 – Input Layer
  - 178 Inputs | No Trainable weights
- Layer 2 – Hidden Layer
  - 16 Neurons | 178 Weights per Neuron | 16 Bias Terms
- Layer 3 – Output Layer
  - 5 Neurons | 16 Weights per Neuron | 5 Bias Terms

Layer 2 = (16 * 178) + 16 = 2864
Layer 3 = (5 * 16 ) + 5 = 85

**Total Trainable parameters = 2949**

FLOPs
When a new observation is fed forward through the model, each feature is multiplied by a subsequent weight in the first hidden layer

$$x_i * w_i$$

Since there are 178 input features, there are 178 weights, that’s 178 multiplication operations for a single neuron. The product of those ops are then summed together with a subsequent bias term

$$z = x_1 * w_1 + x_2 * w_2 + ..... x_{178} * w_{178} + b$$

That’s 178 addition ops. That output of that is then fed into a sigmoid activation function

$$\theta = \frac{1}{1 + e^{-z}}$$

Where there is an exponential op, addition op, and division op => 3 FLOPs for the activation function. The same logic applies for the output layer except no activation function

**Hidden layer**
16 neurons * (178 weight multiplication + 178 addition ops + 3 ops in activation function) = 5744
**Output layer**
5 neurons * (16 weight multiplication + 16 addition ops) = 160

**Total FLOPs for One Observation = 5904**

# *Q1.2c*

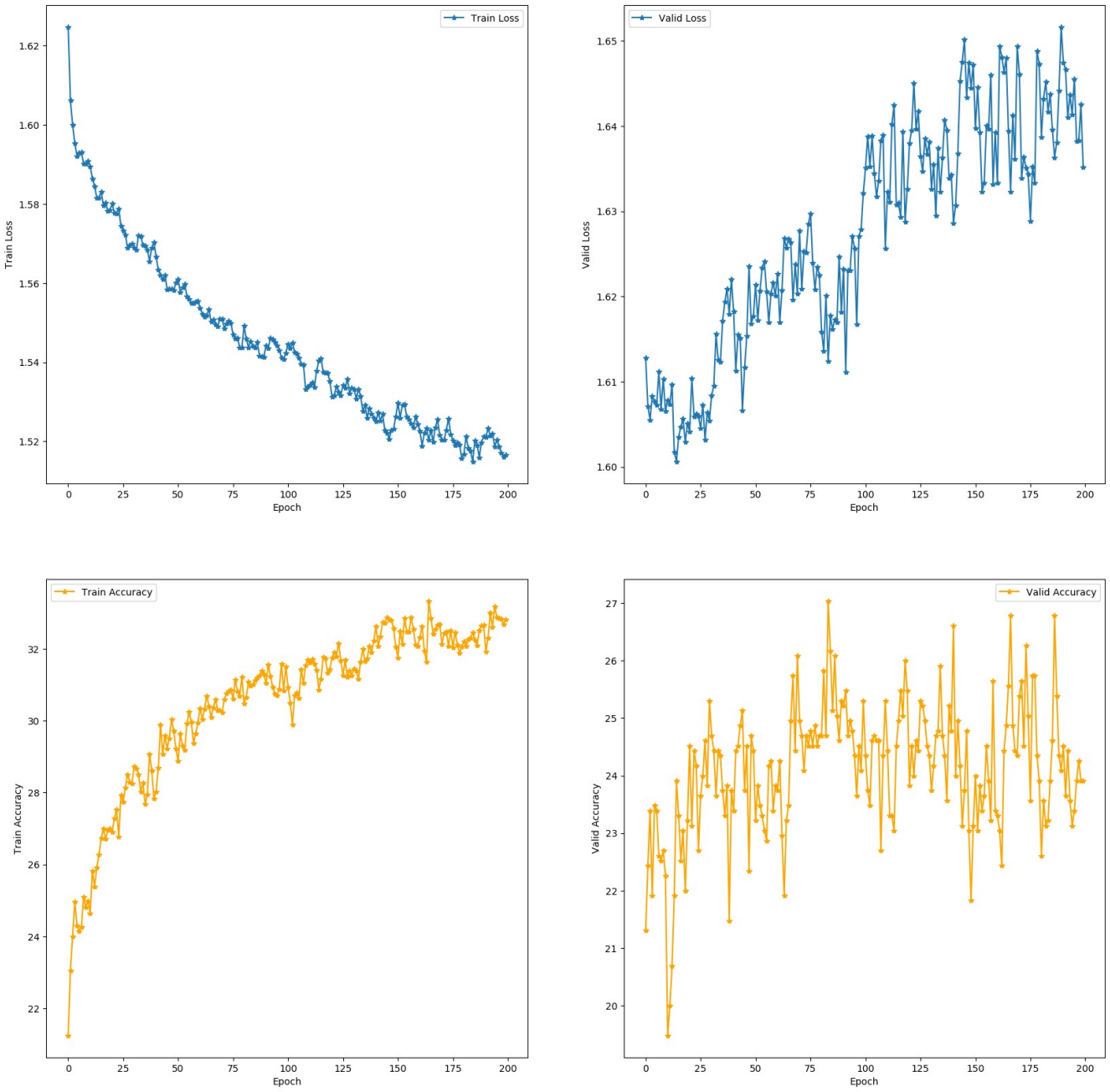



*Illustration 1: (Upper Left) Training Loss; (Upper Right) Validation Loss; (Lower Left) Training Accuracy; (Lower Right) Validation Accuracy*

# *Q1.2d*

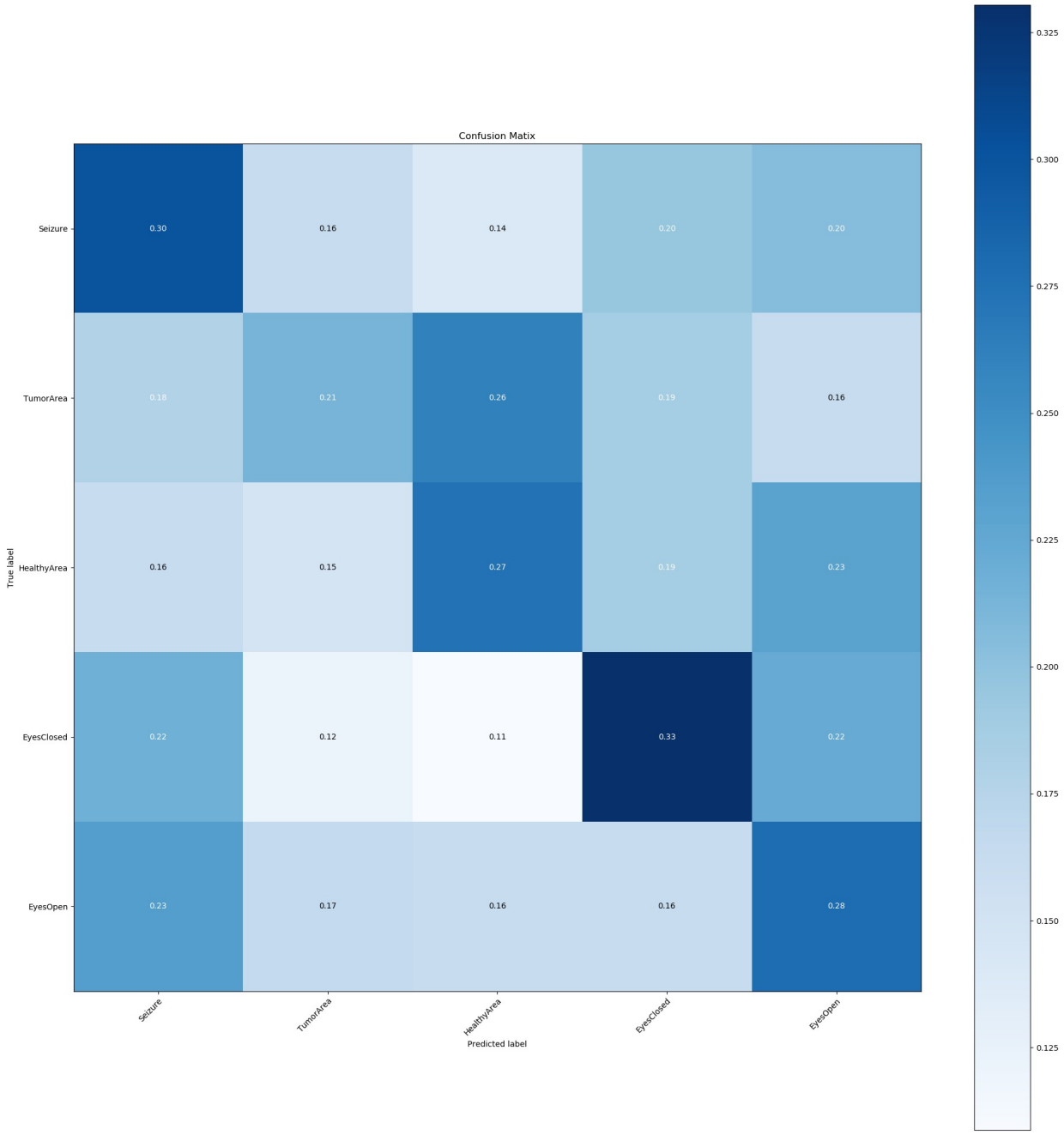



*Illustration 2: MLP Confusion Matrix*

# *Q1.2e*

***Architecture***

- ***Hidden Layer 1 – 178 Inputs | 100 Outputs | ReLu Activation***
- ***Hidden Layer 2 – 100 Inputs | 50 Outputs | ReLu Activation***
- ***Hidden Layer 3 – 50 Inputs | 24 Outputs | ReLu Activation***
- ***Hidden Layer 4 – 24 Inputs | 12 Outputs | Sigmoid Activation***
- ***Output Layer – 12 Inputs | 5 Outputs | Logits***

This architecture was chosen because the original model suffered from high bias. It wasn't capable of generalizing to the training data nor the validation data. That is because the model stepped down too quickly in dimensionality $178 \rightarrow 16 \rightarrow 5$. This didn't allow the model to learn anything about the data. To counteract this, 3 extra hidden layers were added to control the descent in dimensionality. ReLU activation was utilized in the inner 3 hidden layers with sigmoid activation being utilized for the last hidden layer. ReLU was used in earlier layers to prevent any vanishing gradient problem during back propagation. Since the output of a sigmoid is 0 or 1, during training, one risks mulitplying a small number via a small number. This as a consequence would result in an even smaller number, forcing the gradients to "vanish". The result of this model is >90% training accuracy and >70% validation accuracy. The model is suffering, now, from high variance [over fitting], which could be counteracted via regularization.

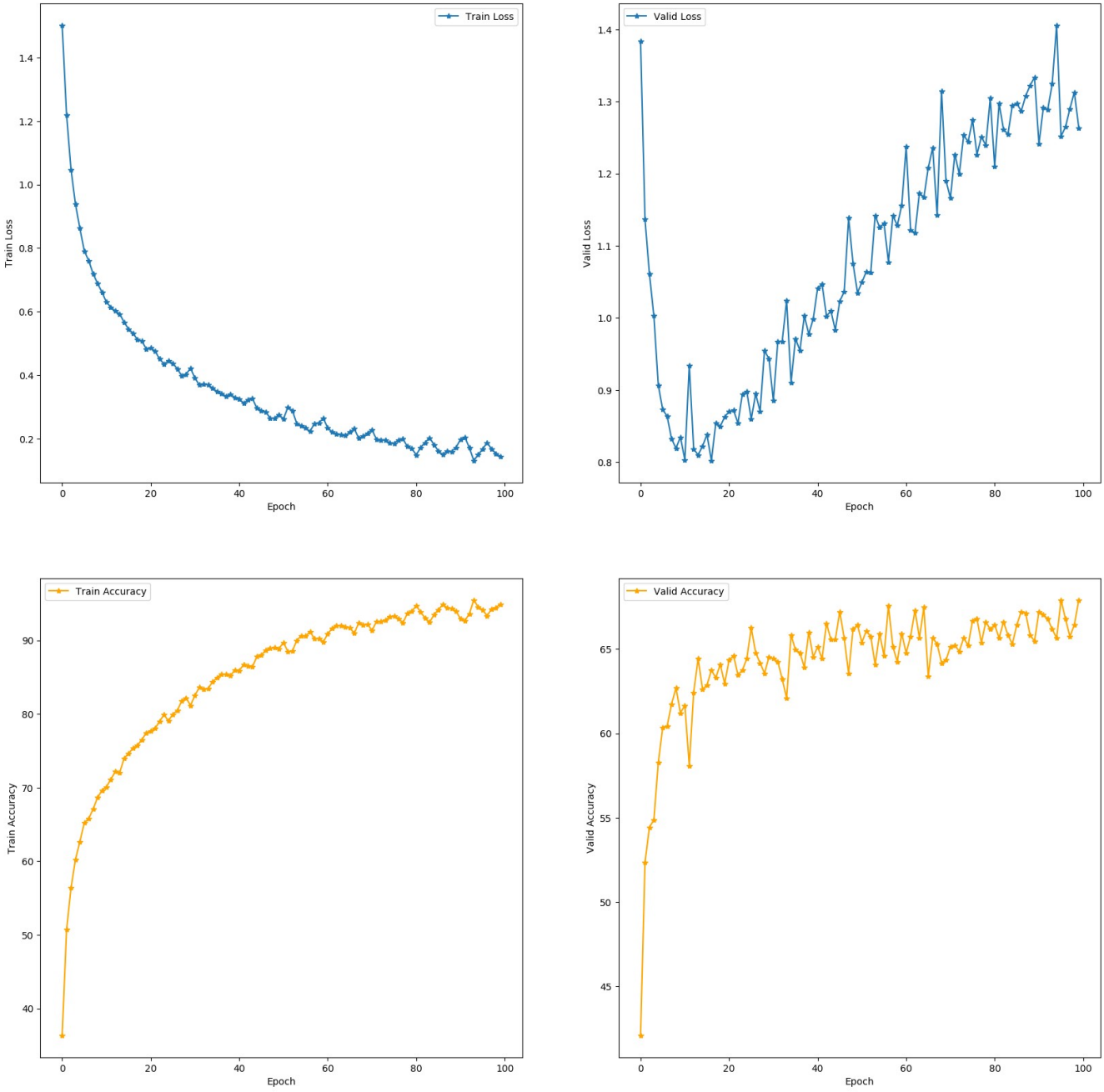



*Illustration 3: (Upper Left) Training Loss; (Upper Right) Validation Loss; (Lower Left) Training Accuracy; (Lower Right) Validation Accuracy*

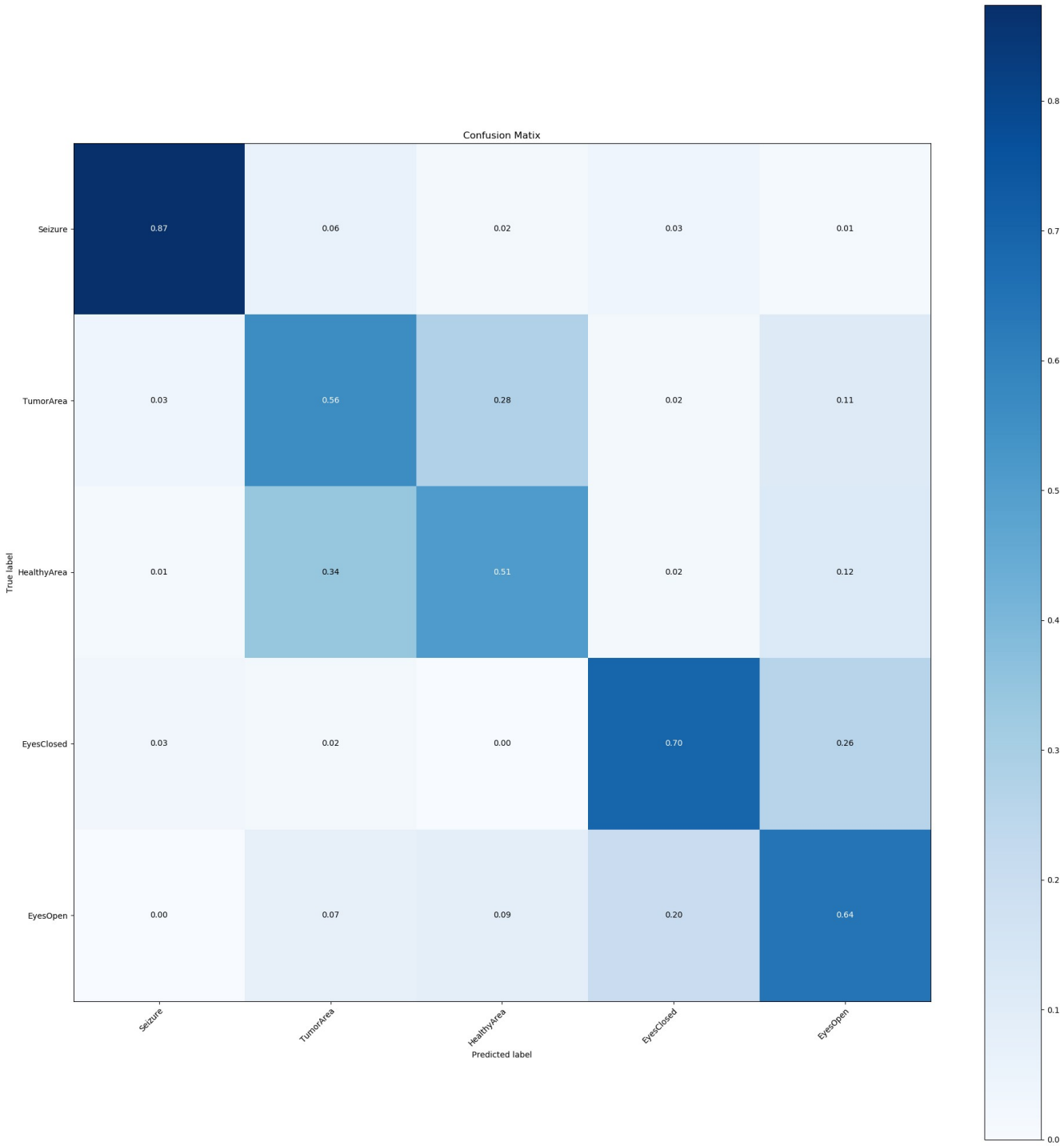



*Illustration 4: Custom MLP Confusion Matrix*

# *Q1.3b*

Conv1 → 5x1 filter | Stride 1 | 6 outputs
* 6 – (5x1) filters learned = ((5x1 + 1) * 6) = 36

Conv2 → 5x1 filter | Stride 1 | 16 outputs
* 16 – (6x5x1) filters learned = ((6x5x1 + 1) * 16) = 496

Fc1 → 128 neurons plus bias term | Flattened Output from previous layer = 16 * 41 = 656
* (128 x 656) + 128 = 84096

Fc2 → 5 neurons plus bias term | 128 Input features
* (128 x 5) + 5 = 645

**Total Trainable Parameters = 85,273**

FLOPs
Conv1 → 1 convolution sample = 5 multiplications & 5 addition ops
Conv1 Stride → stride of 1 = convolve sample 178 – (length of filter (5)- 1) = 174 times
Conv Ops 6 filters →(5 + 5) * 174 * 6 = 10,440
Conv1 Activation relu = 1 comparison op for 6 filters = 10,440 + 6 = 10,446
Pool1 → Stride 2 | size 2 = (174 /2) = 87 comparison ops * 6 filters = 522
Conv1 Block FLOPS = 522 + 10,446 = **10,968**

Conv2 → 1 convolution sample = 5 multiplications & 5 addition ops
Conv1 Stride → stride of 1 = convolve sample 87 – (length of filter (5)- 1) = 83 times
Conv Ops 16 filters →(5 + 5) * 83 * 16 = 13,280
Conv2 Activation relu = 1 comparison op for 16 filters = 13,280 + 16 = 13,296
Pool2 → Stride 2 | size 2 = floor(83 /2) = 41 comparison ops * 16 filters = 656
Conv2 Block FLOPs = 656 + 13296 = **13952**FC1 – 128 neurons x 656 inputs = 83968 multiplication ops & 83968 addition ops = 167936
FC1 activation relu = 128 comparison ops
FC1 FLOPs = 128 + 167936 = **168064**

FC2 – 5 neurons x 128 inputs = 640 multiplication ops & 640 addition ops = 1280
FC2 no activation
FC2 FLOPs = **1280**

**Total FLOPs** = 10968 + 13952 + 168064 + 1280 = **194264 per observation**

# *Q1.3c*

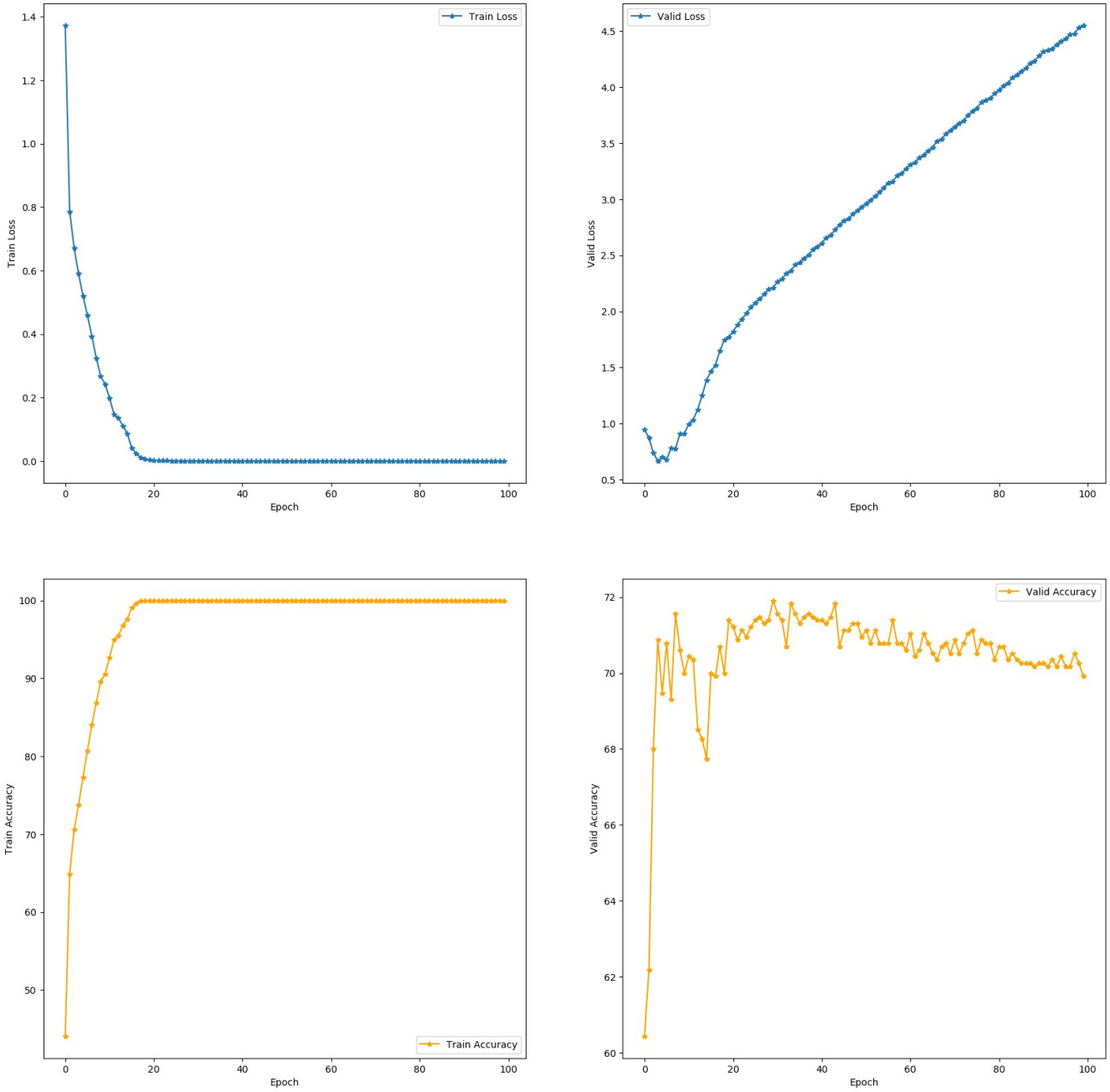



*Illustration 5: (Upper Left) Training Loss; (Upper Right) Validation Loss; (Lower Left) Training Accuracy; (Lower Right) Validation Accuracy*

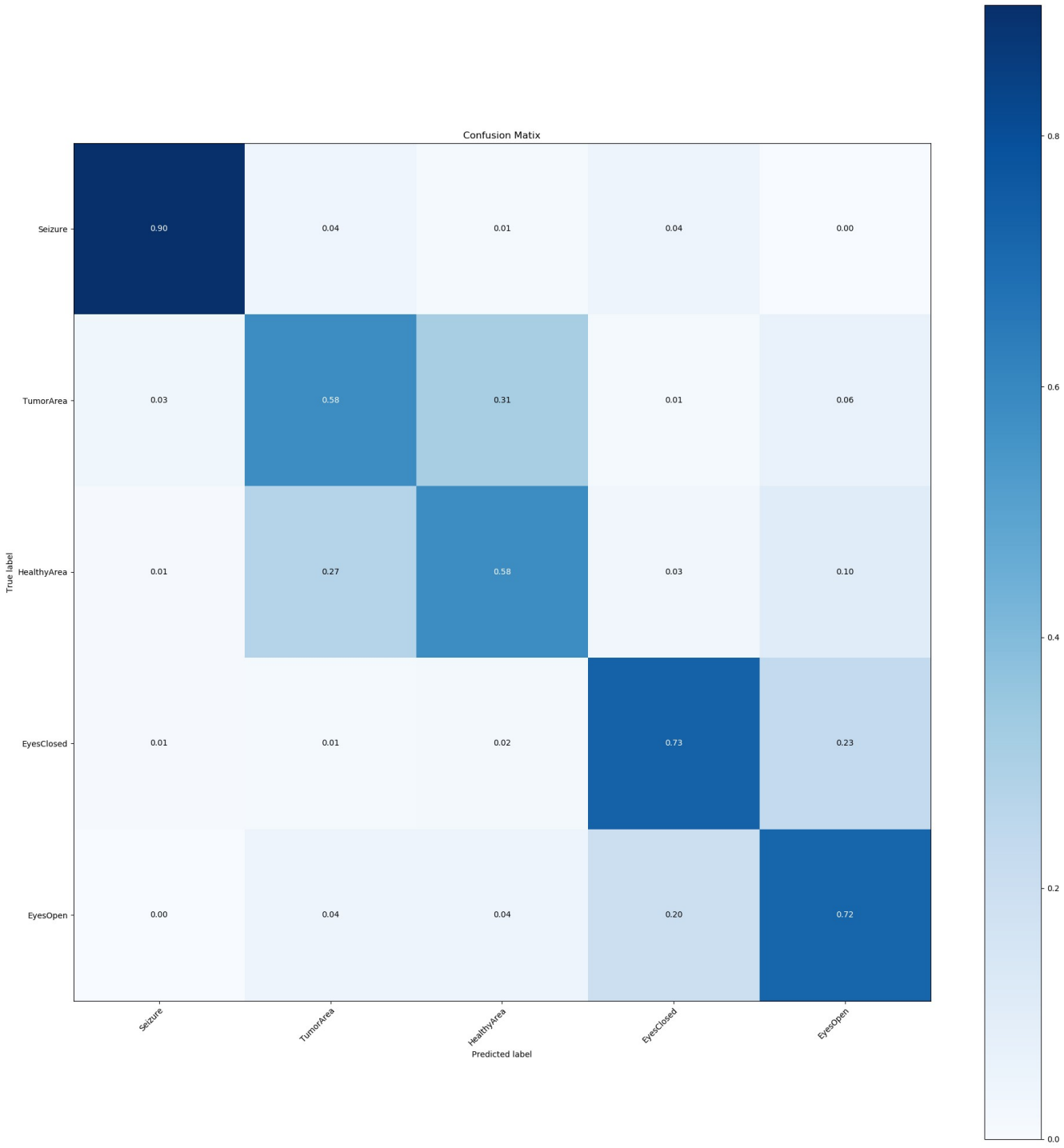



*Illustration 6: CNN Confusion Matrix*

## *Q1.3d*

The original CNN Architecture suffered from the same issue as the MLP, it dropped in dimensionality too quickly all the while growing the parameters abruptly. A successful CNN follows the trend of monotonically reducing output dimension while monotonically increasing number of parameters with each Convolutional block. To mimic this, 5 convolutional blocks were instituted and the kernel size was reduced from 5 to 3. With each output for a convolutional block, the output dimension is decreasing, but the number of parameters are increasing via number of filters (out channels) per each block. This resulted in better precision and recall.

Each conv block used relu activation and max pooling
ConvBlock1(3, 3) → ConvBlock2(15, 3) → ConvBlock3(20, 3) → ConvBlock4(30, 3) → ConvBlock5(100, 3)

Where the first number is the number of filters for that ConvBlock and the second number is the kernel_size

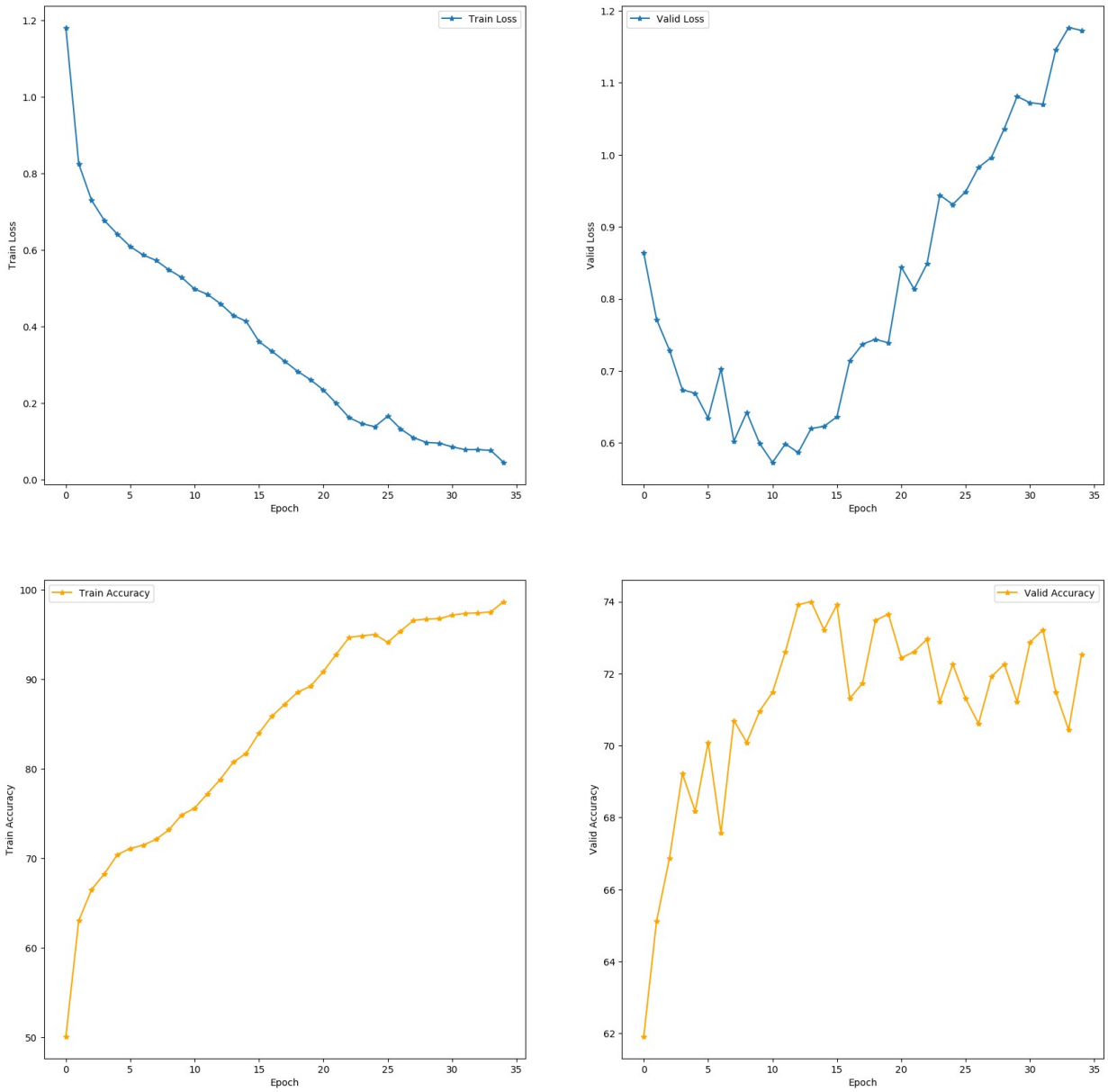
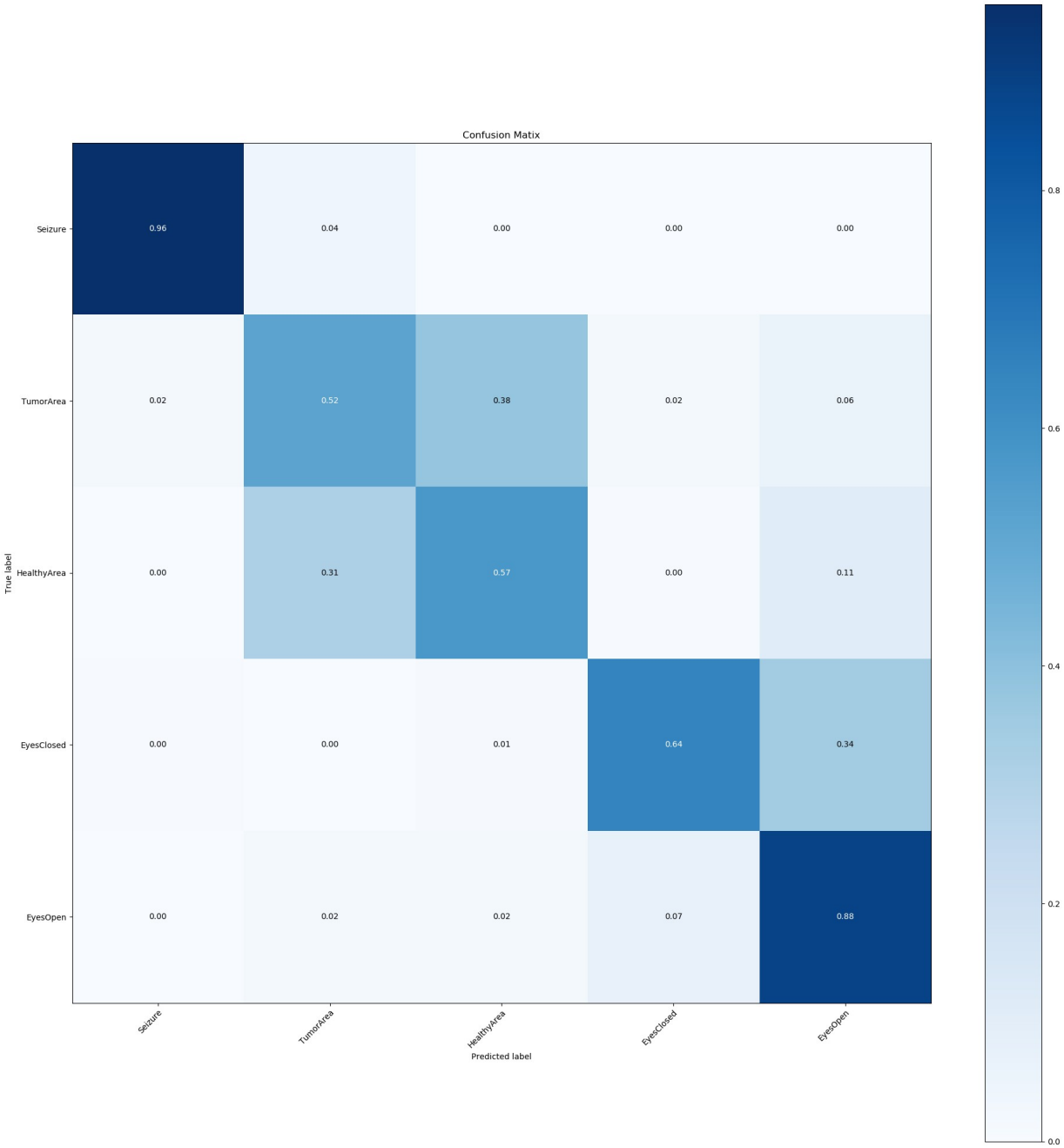



*Illustration 3: (Upper Left) Training Loss; (Upper Right) Validation Loss; (Lower Left) Training Accuracy; (Lower Right) Validation Accuracy*


Confusion Matix
Seizure
TumorArea
HealthyArea
EyesClosed
EyesOpen
True label
Predicted label
0.96 0.04 0.00 0.00 0.00
0.02 0.52 0.38 0.02 0.06
0.00 0.31 0.57 0.00 0.11
0.00 0.00 0.01 0.64 0.34
0.00 0.02 0.02 0.07 0.88
0.8
0.6
0.4
0.2
0.0

# *Q1.4b*

GRU cell → 16 Hidden units | 178 inputs | 3 Feed Forward Neural Networks in a Cell
GRU cell params = #of NNs * (#ofHidden units (#ofHidden units + 1) + #ofHidden units)
1 GRU cell = 3 * ( 16 (16 + 178) + 16) = 9,360
2 GRU cells = 2 * 9360 = 18720

FC1 → 5 Neurons | 32 Inputs | plus bias
FC1 params = (5 * 32) + 5 = 165

**Total Trainable Parameters = 165 + 18720 = 18885**

FLOPs
A FC in the first GRU has 16 neurons and 178 inputs and tanh activation = (e^z – e^-z) / (e^z + e^-z) or 7 FLOPs and sigmoid activation or 3 FLOPs
FC = 16 neurons * (178 multiplication ops + 178 addition ops) = 5696
3sigmoid + 5696 FC input + 5696 FC hidden state = 11395

$$z_t = \sigma(W_{iz} x_t + b_{iz} + W_{hz} h_{(t-1)} + b_{hz})$$

3sigmoid + 5696 FC input + 5696 FC hidden state = 11395

$$n_t = \tanh(W_{¿} x_t + b_{¿} + r_t * (W_{hn} h_{(t-1)} + b_{hn}))$$

7tanh + 5696 FC input + 5696 FC hidden state = 11399

$$h_t = (1 - z_t) * n_t + z_t * h_{(t-1)}$$

4 * 16 = 64
FLOPS GRU CELL 1 = 34253

A FC in the second GRU has 16 neurons and 16 inputs
FC = 16neurons * (16 multiplication ops + 16 addition ops) = 512
rt = 3sigmoid + 512 FC input + 512 FC hidden state = 1027
zt = 3sigmoid + 512 FC input + 512 FC hidden state = 1027
nt = 7sigmoid + 512 FC input + 512 FC hidden state = 1031
ht = 4 * 16 = 64
FLOPS GRU CELL 2 = 3149

Final FC layer has 5 neurons with 32 inputs and 5 bias
FC = (5 * 32) + 5 = 165

**TOTAL FLOPs = 37567**

# *Q1.4c*

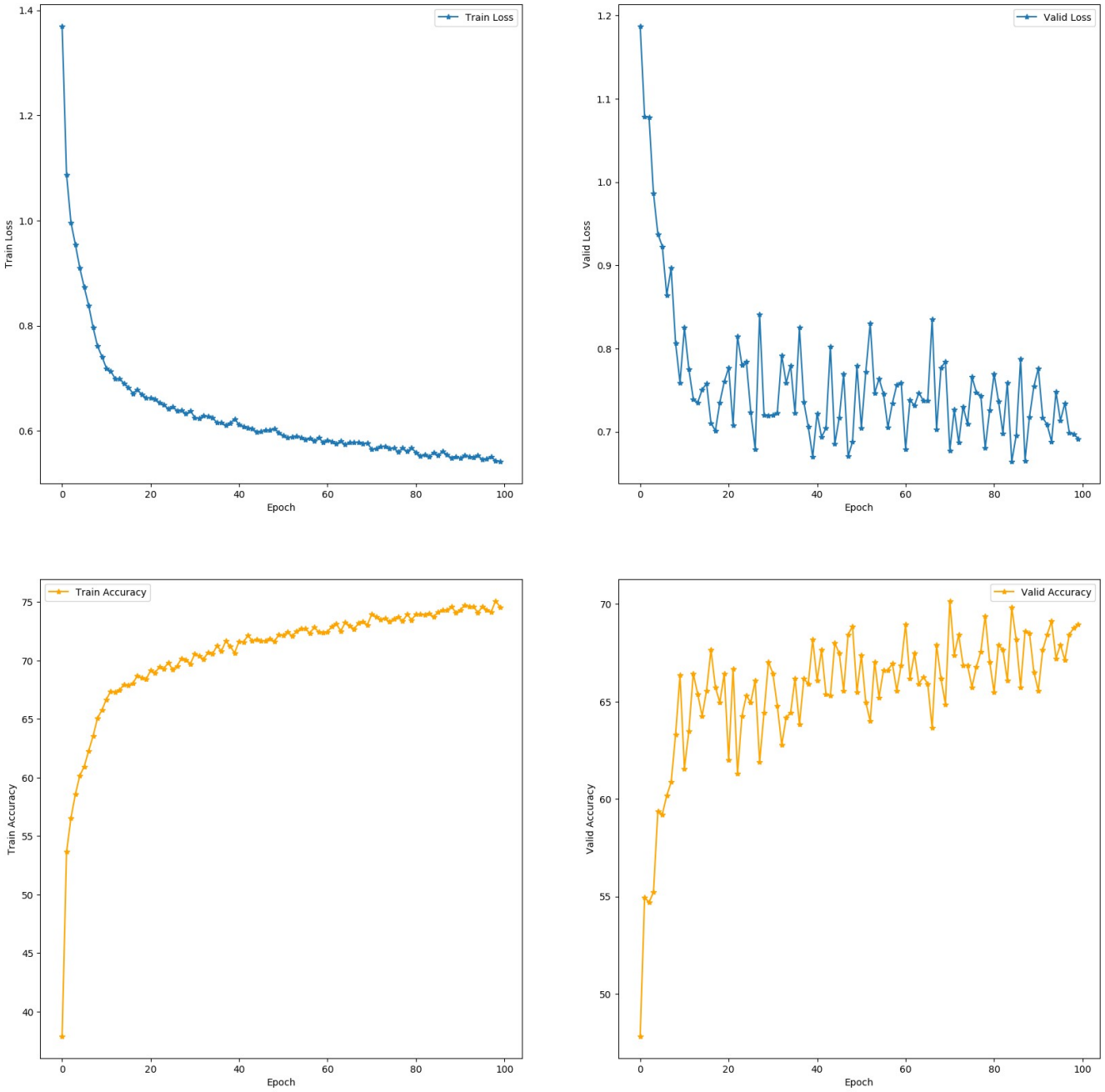



*Illustration 4: (Upper Left) Training Loss; (Upper Right) Validation Loss; (Lower Left) Training Accuracy; (Lower Right) Validation Accuracy*

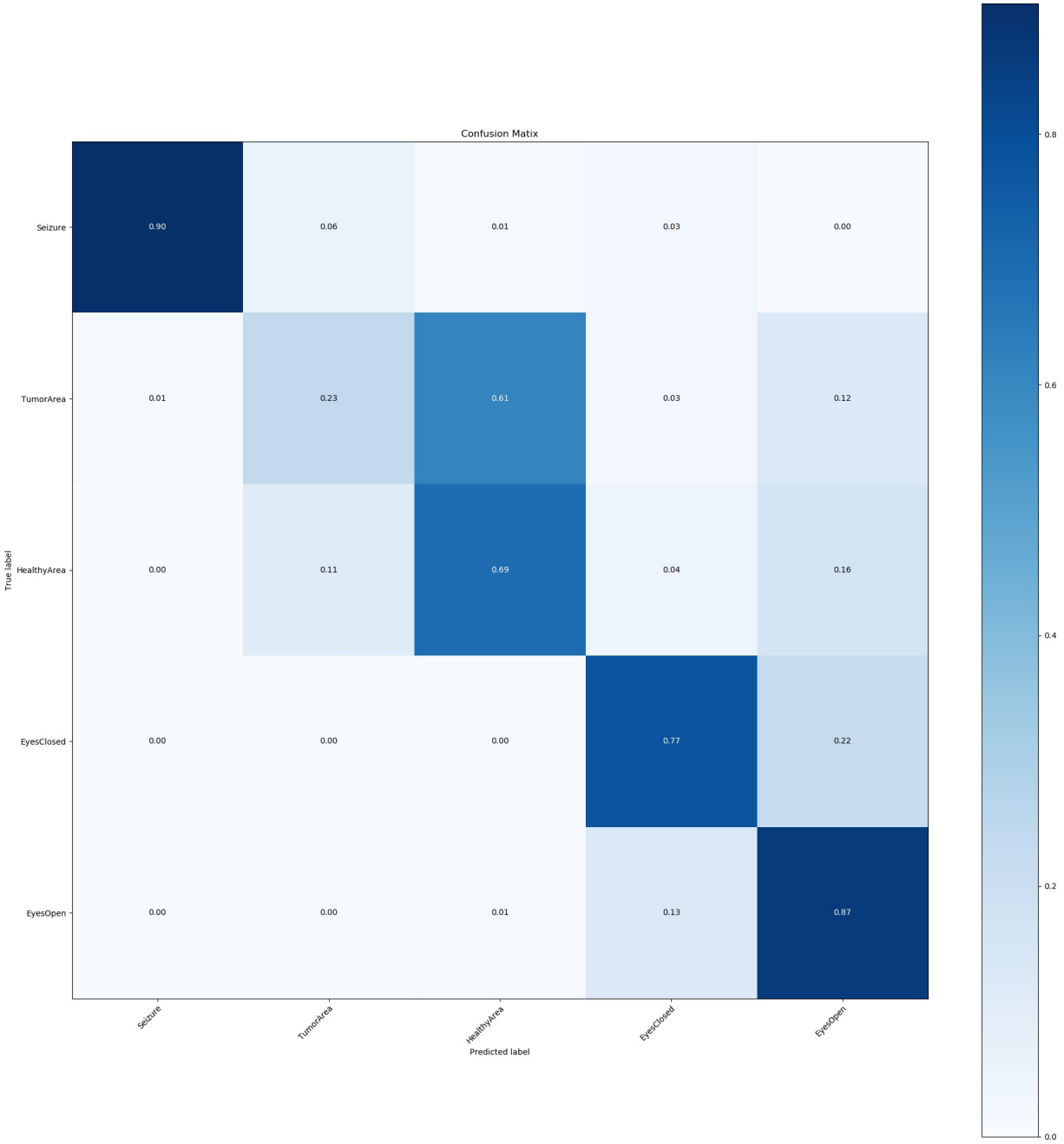



*Illustration 5: RNN Confusion Matrix*

# *Q1.4d*

The original RNN architecture plateaued at around 70% validation accuracy while the train accuracy was only around 75%. This is a sign of potential high bias. To combat this, more layers were added in the GRU in attempts to get the model to learn more complex features. By adding only 2 more GRU layers and stepping down the feature dimension space, the model's train accuracy reached ~80% while the the validation accuracy increased to ~75%. This indicates that more layers would be beneficial, but due to extremely long train times, more layers weren't attempted

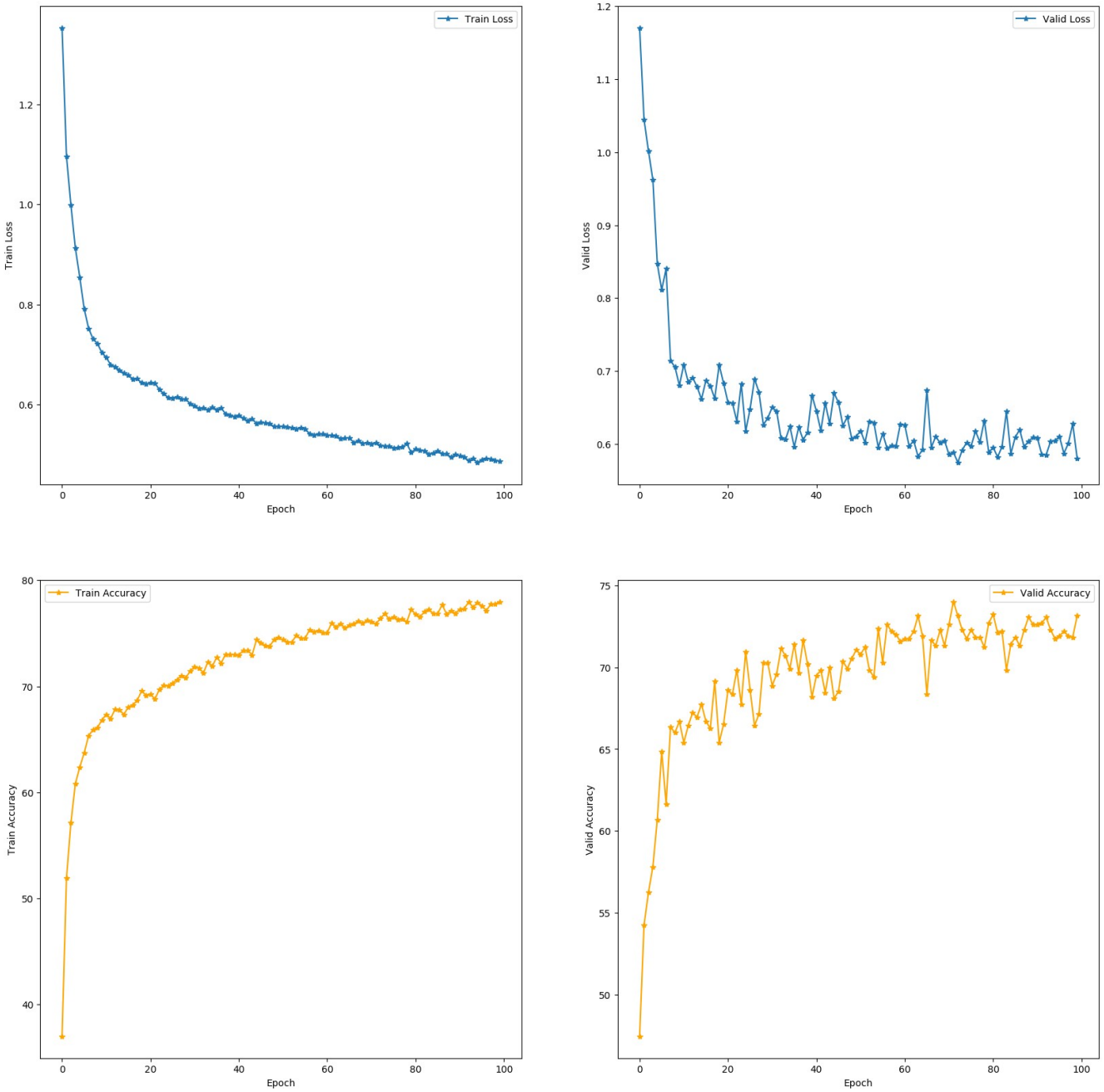



*Illustration 6: (Upper Left) Training Loss; (Upper Right) Validation Loss; (Lower Left) Training Accuracy; (Lower Right) Validation Accuracy*

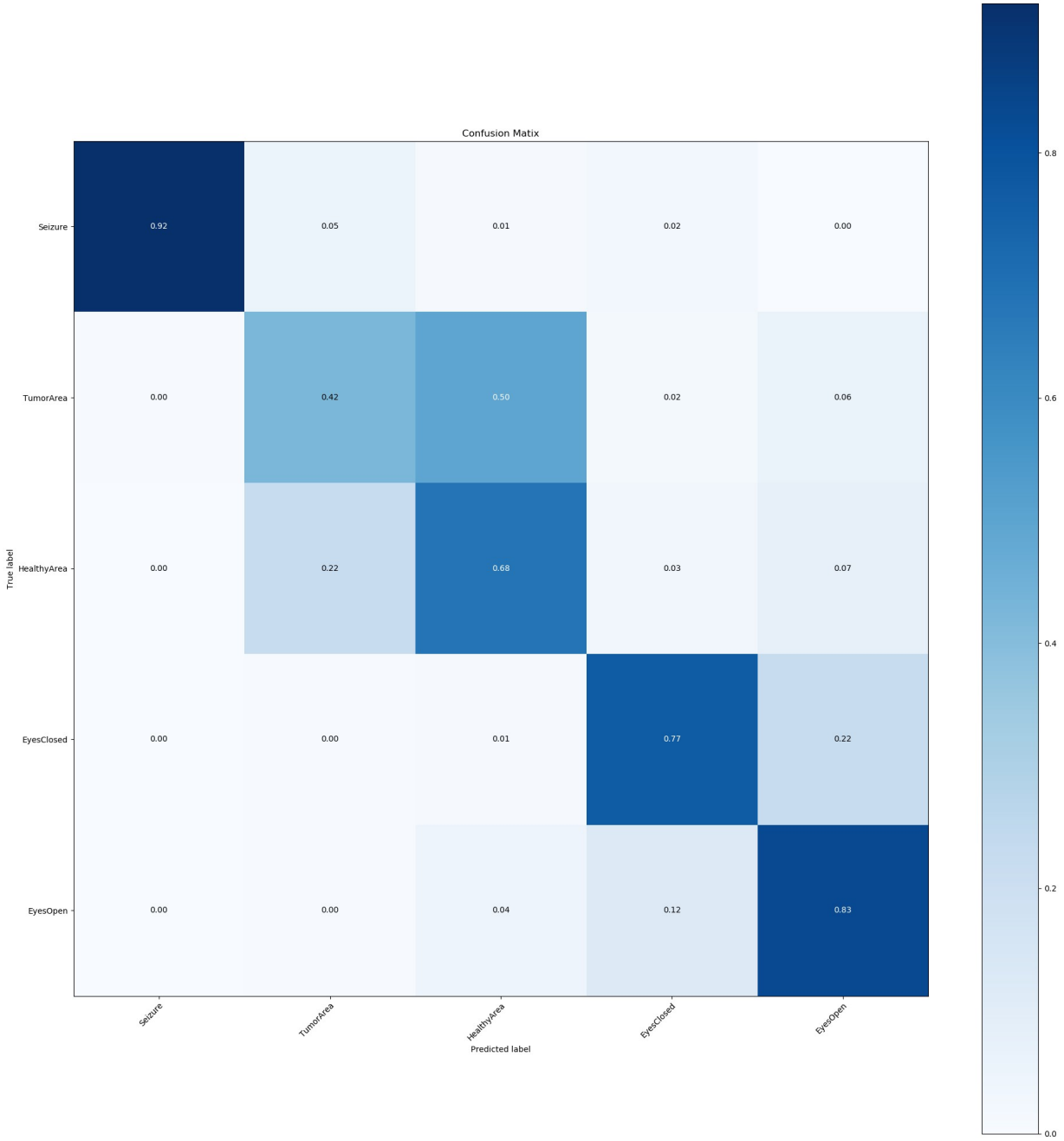



*Illustration 7: Custom RNN Confusion Matrix*

# *Q2.3a*

From the training statistics (Train vs Validation), it is evident the model is suffering from high variance. It is drastically overfitting itself to the random patterns embedded in the training set. It is highlighted in the graphs below around Epoch 10, Training loss drops off drastically while accuracy increases sharply. In contrast to this, around the same epoch, the validation set experiences an increase in velocity of large losses while the accuracy drops tremendously

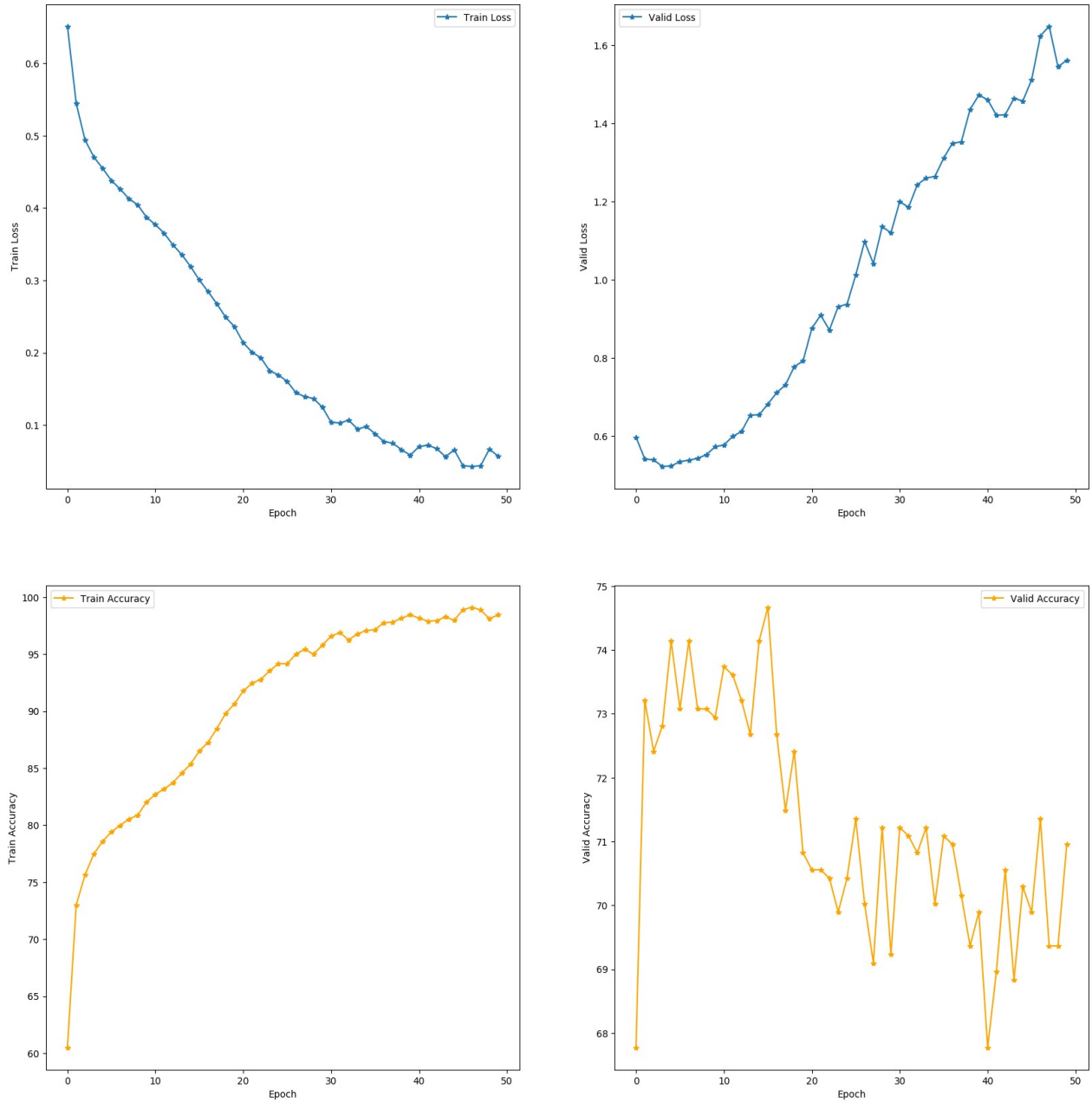



*Illustration 8: (Upper Left) Training Loss; (Upper Right) Validation Loss; (Lower Left) Training Accuracy; (Lower Right) Validation Accuracy*

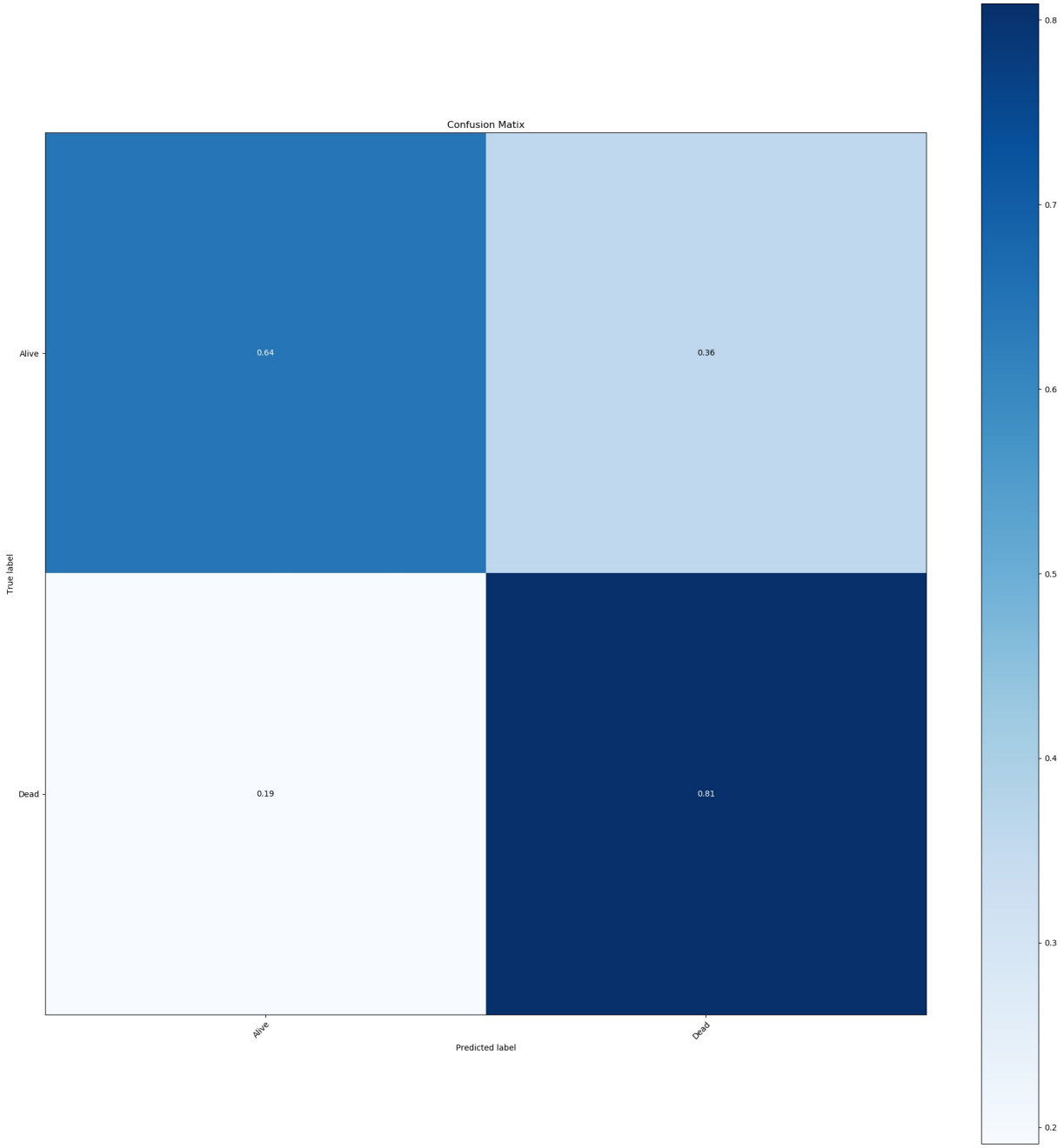



*Illustration 9: Variable RNN Confusion Matrix*

# *Q2.3b*

For this architecture, I added more Fully connected layers before the GRU. I instituted LeakyReLU activation to attenuate the risks of vanishing gradients as well as dropout layers to prevent overfitting. Each layer reduces the tensors in dimension, gradually stepping down output. 4 GRU layers were added in hopes to increase the model's capability of "remembering" important features. Ultimately, this model does perform better than the original architecture, but does suffer from high variance (overfitting). This can be seen in the graphs below where 100% accuracy has been achieved with the training set, but the validation set has plateaued between 71-73%. The model also does a better than decent job at being predicting "dead", but only a little better than guessing with respect to predicting "alive"

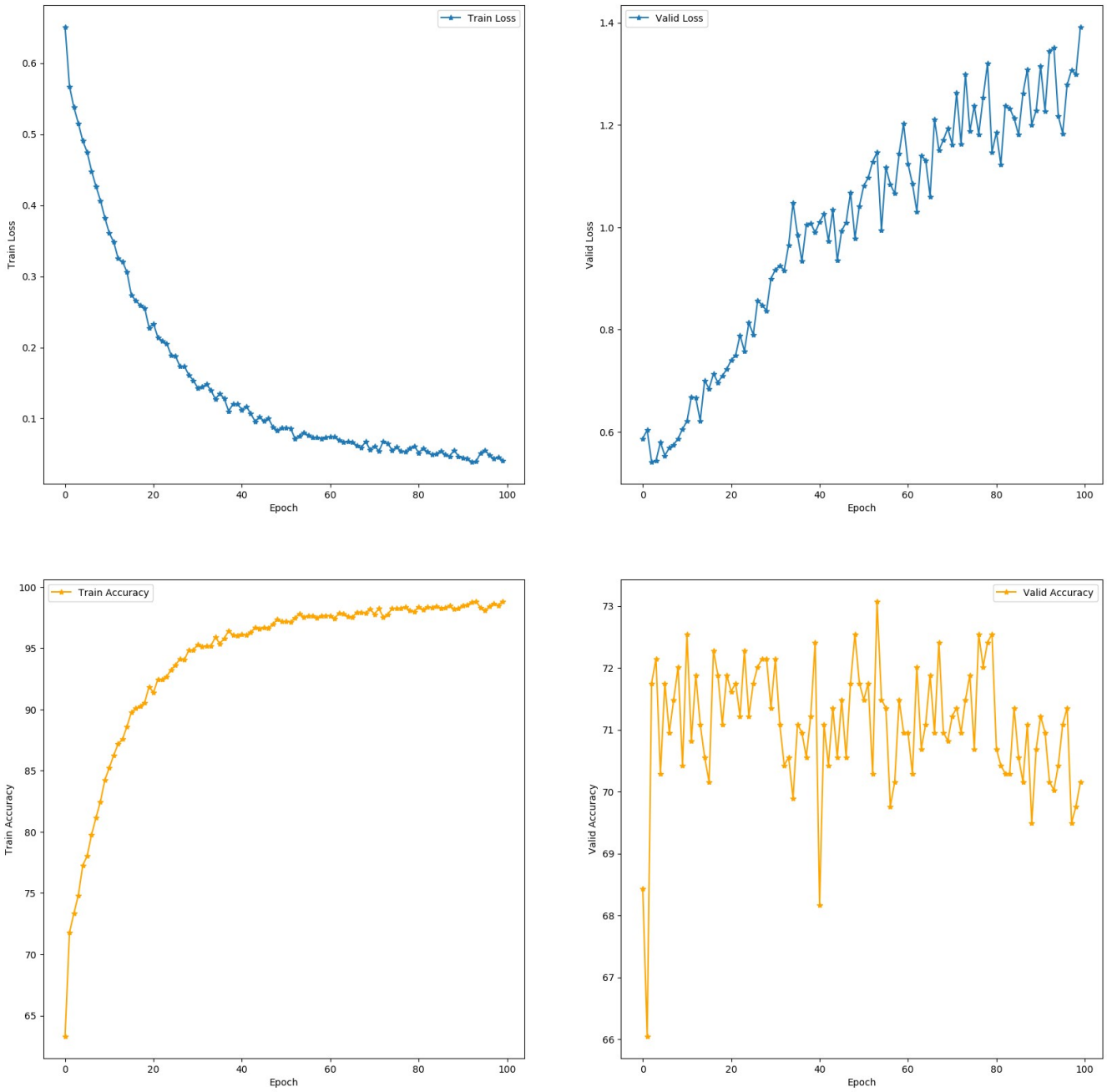



*Illustration 10: (Upper Left) Training Loss; (Upper Right) Validation Loss; (Lower Left) Training Accuracy; (Lower Right) Validation Accuracy*

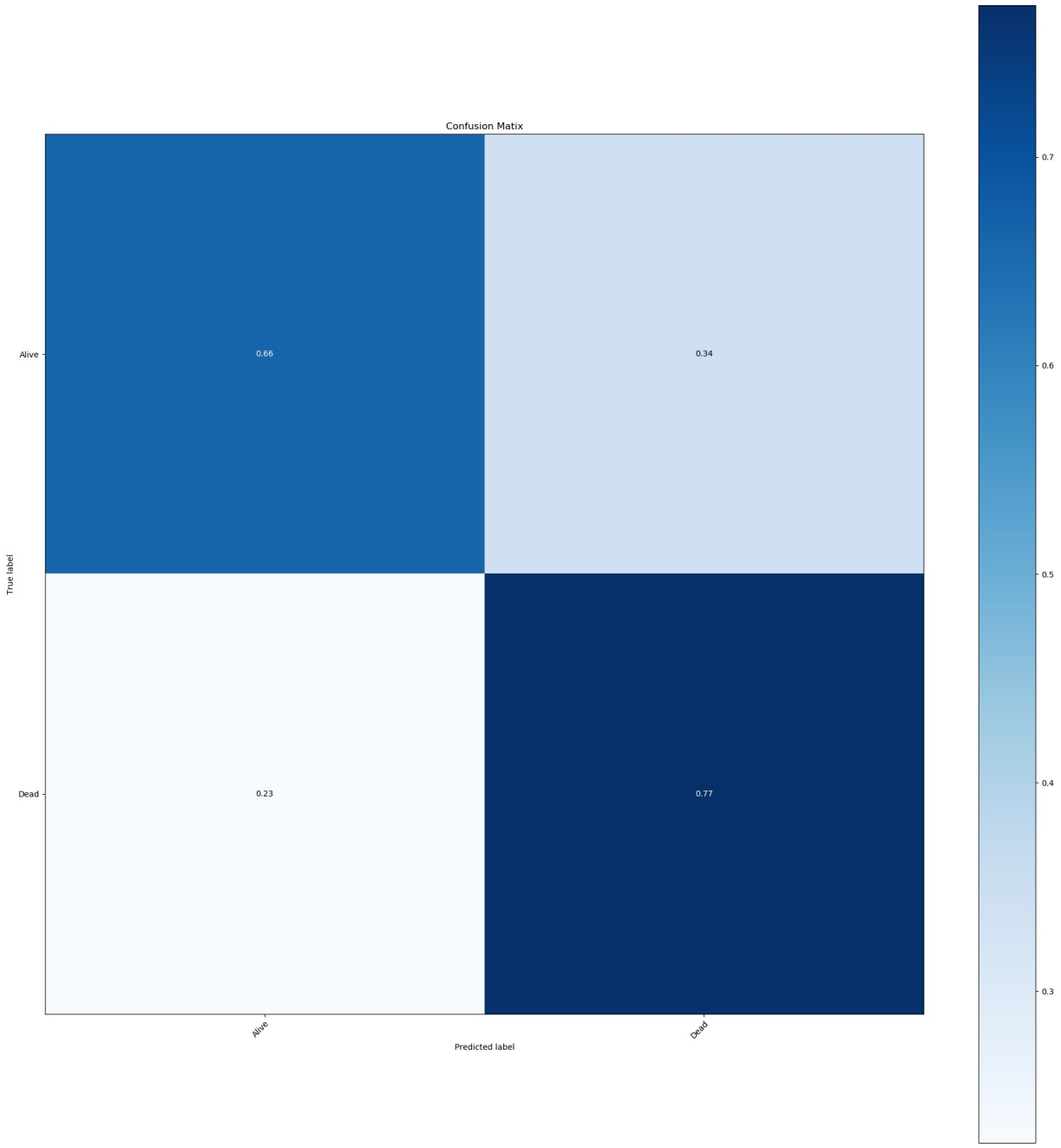



*Illustration 11: Custom Variable RNN Confusion Matrix*