Big Data Analytics in Healthcare

Yichao Zhang

March 26, 2019

1 Epileptic Seizure Classification

Multi-layer Perceptron

The number of "trainable" parameters in the model and number of floating-point computations (add/subtraction/multiplication/division/negation/exponent take 1 operation)

Fully connected linear layer: $\sum_{i=1}^{N} x_i w_i + b$ for each output, where N is the size of input, so we have $N + 1$ parameters and $N + N = 2N$ computations for each output.

Sigmoid function: $\sigma(x) = \frac{1}{1+e^{-t}}$. It has no parameter, and has 4 computations, include negation, exponent, add and division.

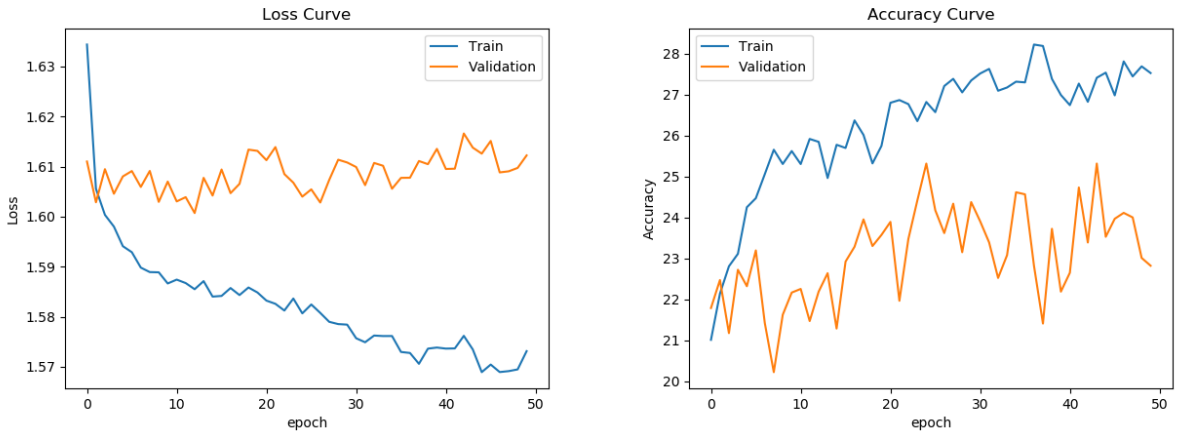
1. Single hidden layer has 178 input, and 16 output, we have $(178 + 1) * 16 = 2864$ parameters, and $2 * 178 * 16 = 5696$ computations;

2. Sigmoid activation function apply on 16 output, we have $4 * 16 = 64$ computations;

3. The output layer has 16 input, and 5 output, we have $(16 + 1) * 5 = 85$ parameters, and $2 * 16 * 5 = 160$ computations.

In total, we have $2864+85 = 2949$ parameters, and $5696+64+160 = 5920$ computations.

Learning Curves for MLP Model



(a) Loss curves (b) Accuracy curves

Figure 1: Learning curves

Confusion Matrix for MLP Model

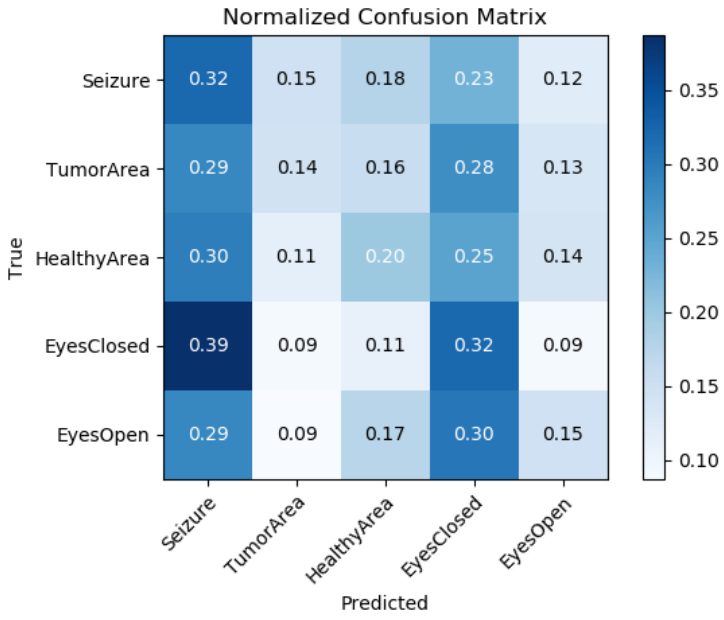


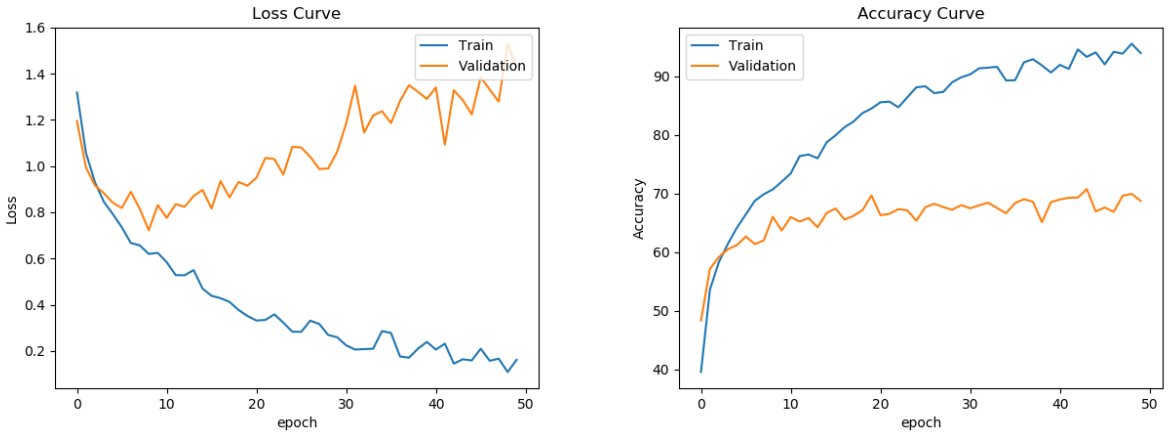
Figure 2: Confusion Matrix

Architecture and Results

Here we use 4 fully connected hidden layers, each layer has 256 units and followed by a Relu function, rather than Sigmoid function. The last output layer has 5 units.

We use this architecture is because the baseline model under-fit the data, so we apply a larger and deeper architecture to fit the data better. And the Relu function has no gradient vanishing problem when $x > 0$.

According to the learning curves below, the improved model decreases the loss from 1.6 to 0.8, raises the accuracy from 25 to 65, and raises the diagonal elements tremendously in the confusion matrix.



(a) Loss curves

(b) Accuracy curves

Figure 3: Learning curves

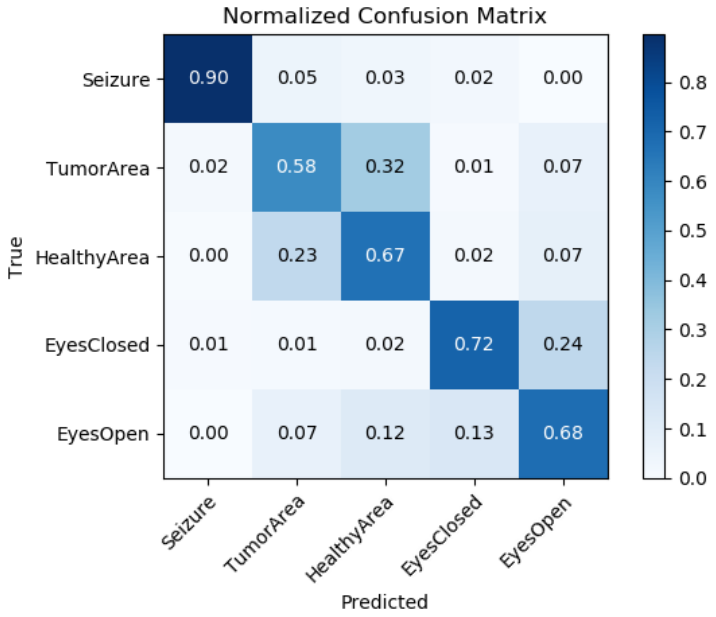


Figure 4: Confusion Matrix

Convolutional Neural Network (CNN)

The number of "trainable" parameters in the model and number of floating-point computations (add/subtraction/multiplication/division/negation/exponent take 1 operation)

1-D CNN:

kernel_size: $K = 5$, there are 5 parameters w_i for each kernel.

Each output is $\sum_{i=1}^{K} x_i w_i$, there are $K + K - 1 = 2K - 1 = 9$ computations.

padding: $P = 0$

stride: $S = 1$

1st CNN layer:

Width: $W = 178$, which is in_size

Output_size: $O = int((W - K + 2 * P)/S) + 1 = (178 - 5)/1 + 1 = 174$, there are $174 * 9 = 1566$ computations for each channel.

in_channels $= 1$, out_channels $= 6$, so we have $6*5 = 30$ parameters and $6*174*9 = 9396$ computations.

1st Relu function:

Suppose each Relu takes 1 computation, we have $6 * 174 = 1044$ computations here.

1st Max Pooling Layer:

kernel_size = 2,

Output_size = int(174 /2) = 87, we need 87 comparisons for each channel, suppose each pooling takes 1 computation, we have $6 * 87 = 522$ computations.

2nd CNN layer:

Width: $W = 87$, which is in_size

Output_size: $O = int((W - K + 2 * P)/S) + 1 = (87 - 5)/1 + 1 = 83$, there are $83 * 9 = 747$ computations for each channel.

in_channels = 6, out_channels = 16, so we have $16 * 5 = 80$ parameters and $16 * 83 * 9 = 11952$ computations.

2nd Relu function:

Suppose each Relu takes 1 computation, we have $16 * 83 = 1328$ computations here.

2st Max Pooling Layer:

kernel_size = 2,

Output_size = int(83 /2) = 41, we need 41 comparisons for each channel, suppose each pooling takes 1 computation, we have $16 * 41 = 656$ computations.

FC-128:

$16 * 41 = 656$ input, and 128 output, we have $(656 + 1) * 128 = 84096$ parameters, and $2 * 656 * 128 = 167936$ computations;

3rd Relu function:

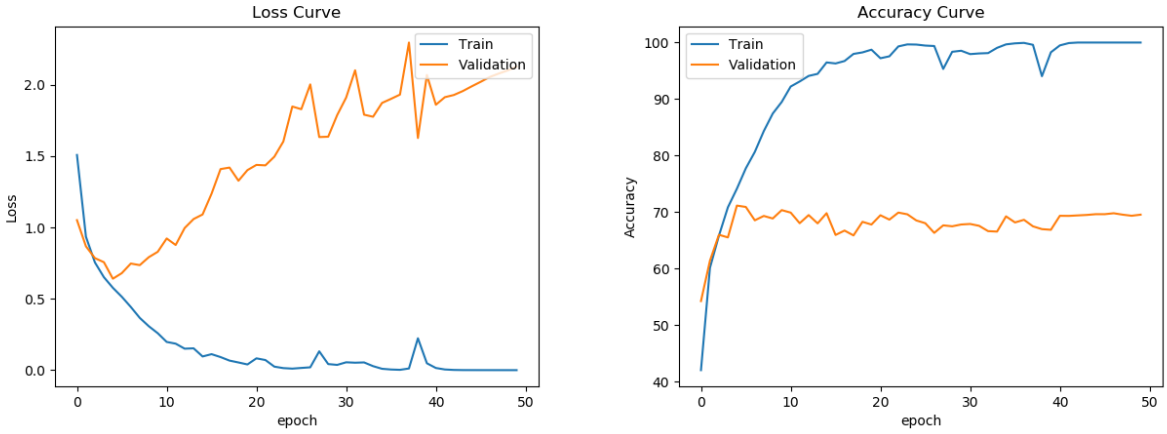
128 computations.

FC-5:

128 input, and 5 output, we have $(128 + 1) * 5 = 645$ parameters, and $2 * 128 * 5 = 1280$ computations.

In total, we have, $30 + 80 + 84096 + 645 = 84851$ parameters, and $9396 + 1044 + 522 + 11952 + 1328 + 656 + 167936 + 128 + 1280 = 194242$ computations. Tedious!

Learning Curves and Confusion Matrix for CNN Model



(a) Loss curves

(b) Accuracy curves

Figure 5: Learning curves

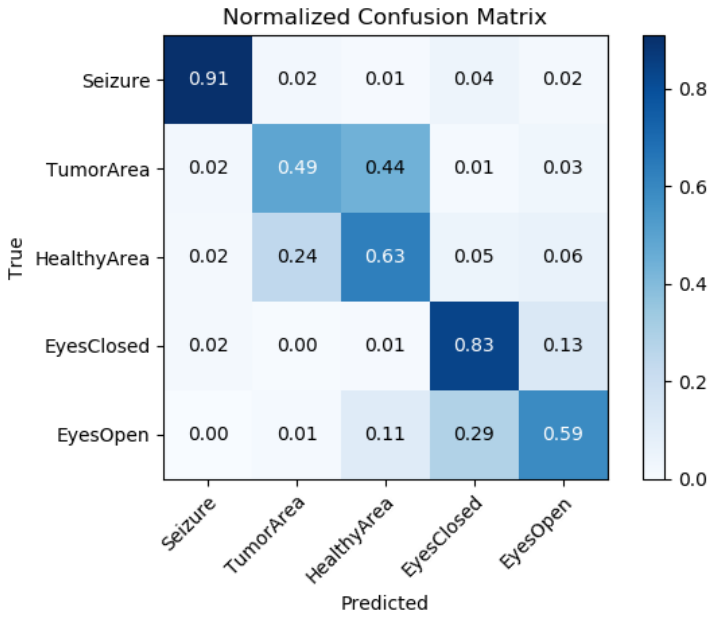


Figure 6: Confusion Matrix

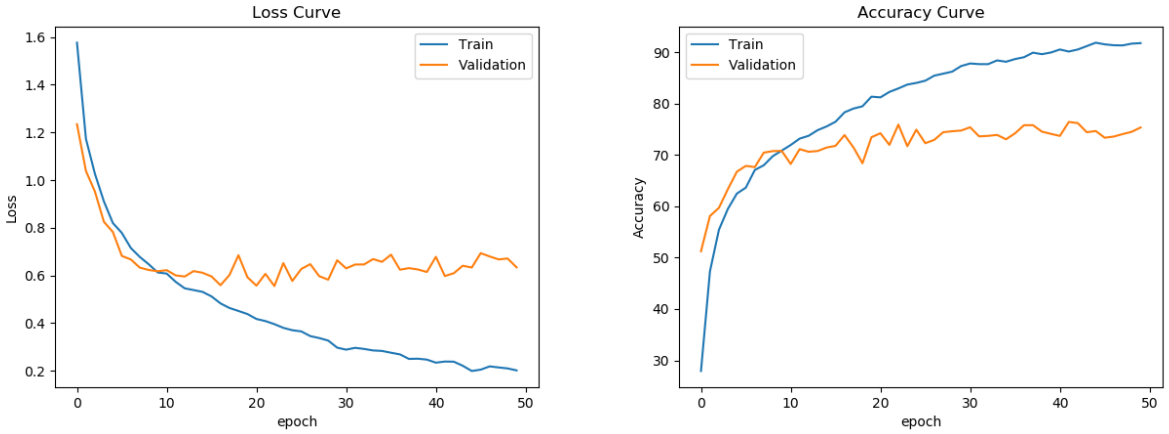
Architecture and Results

My architecture is modified from the baseline CNN model:

Both conv1 and conv2 are followed by Dropout layer(p = 0.2) for regularization.

At the end, we use FC-128, FC-64 and FC-5 to make our model more complex. Both FC-128, FC-64 are followed by Dropout layer($p = 0.2$) for regularization, use FC-5 as output.

The learning curves are shown below. Although the loss and accuracy for the best model are similar, the improved model decreases the overfitting obviously.



(a) Loss curves (b) Accuracy curves

Figure 7: Learning curves

The confusion matrix shows that, the improved model raises the diagonal elements and makes the matrix more symmetric. Especially get a better performance between TumorArea and HealthyArea.

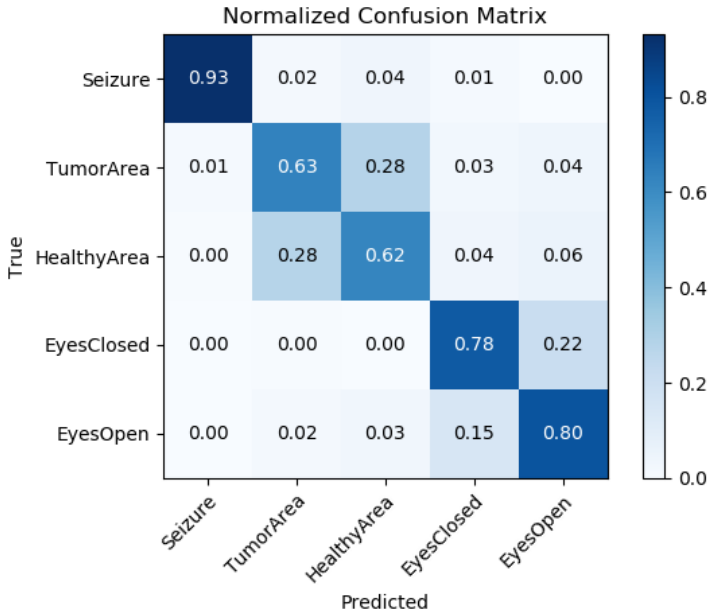


Figure 8: Confusion Matrix

1.4 Recurrent Neural Network (RNN)

The number of "trainable" parameters in the model and number of floating-point computations (add/subtraction/multiplication/division/negation/exponent take 1 operation)

GRU, 1 layer:

Width: $W = 1$, for each $\mathbf{x}^t$,

Sequence Length: $L = 178$,

Each sequential input is a $W * L$ matrix.

hidden_size: $H = 16$

When we get a row vector $\mathbf{x}^t$ as input:

$\mathbf{z}^t = \sigma(\mathbf{U}^z\mathbf{x}^t + \mathbf{W}^z\mathbf{h}^{t-1} + \mathbf{b}^z)$

Parameters: $(W + H + 1)H = (1 + 16 + 1) * 16 = 288$

Computations: each output element has $2(W + H) = 2(1 + 16) = 34$ computations in quote, and 4 computations for sigmoid. So $(34 + 4)H = (34 + 4) * 16 = 608$.

$\mathbf{r}^t = \sigma(\mathbf{U}^r\mathbf{x}^t + \mathbf{W}^r\mathbf{h}^{t-1} + \mathbf{b}^r)$

Parameters: 288

Computations: 608

$\tilde{\mathbf{h}}^t = \tanh(\mathbf{U}^h\mathbf{x}^t + \mathbf{r}^t * \mathbf{W}^h\mathbf{h}^{t-1} + \mathbf{b}^h)$

Parameters: 288

Computations: each output element has $2(W + H) + 1 = 35$ computations in quote, $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ has 1 computation for e^x, 2 computations for e^{-x}, and another 3 computations to combine them, so 6 computations for tanh. Then we have $(35 + 6) * 16 = 656$ computations.

$\mathbf{h}^t = (1 - \mathbf{z}^t) * \mathbf{h}^{t-1} + \mathbf{z}^t * \tilde{\mathbf{h}}^t$

Computations: each output element has 4 computations, thus $4 * 16 = 64$ computations.

In total for each $\mathbf{x}^t$ in GRU:

Parameters: $288 + 288 + 288 = 864$

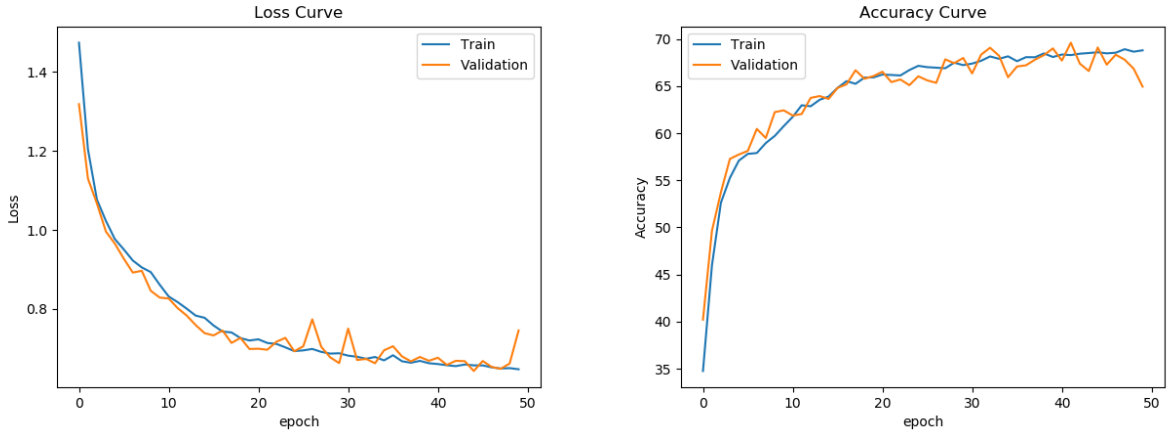
Computations: $608 + 608 + 656 + 64 = 1936$

Consider the sequence length $L = 178$, we have $1936 * 178 = 344608$ computations in GRU.

FC-5: 16 input, and 5 output, we have $(16 + 1) * 5 = 85$ parameters, and $2 * 16 * 5 = 160$ computations.

In total, $864 + 85 = 949$ parameters, and $344608 + 160 = 344768$ computations.

Learning Curves and Confusion Matrix for RNN Model



(a) Loss curves

(b) Accuracy curves

Figure 9: Learning curves

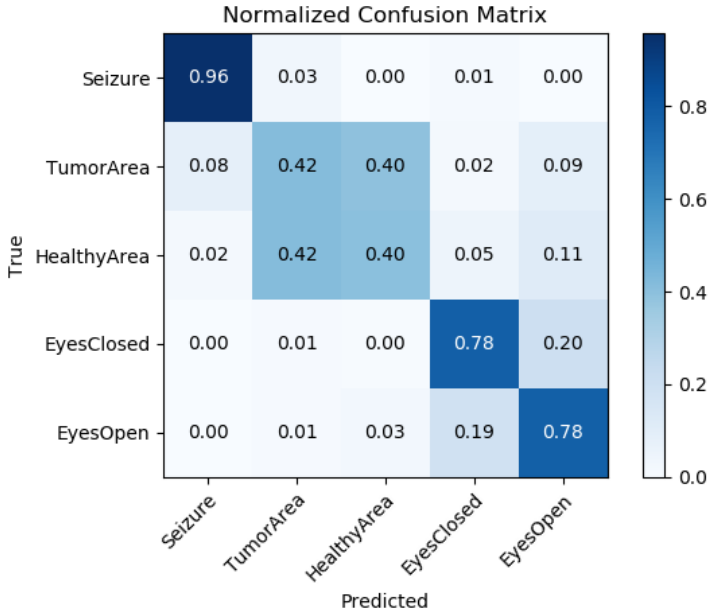
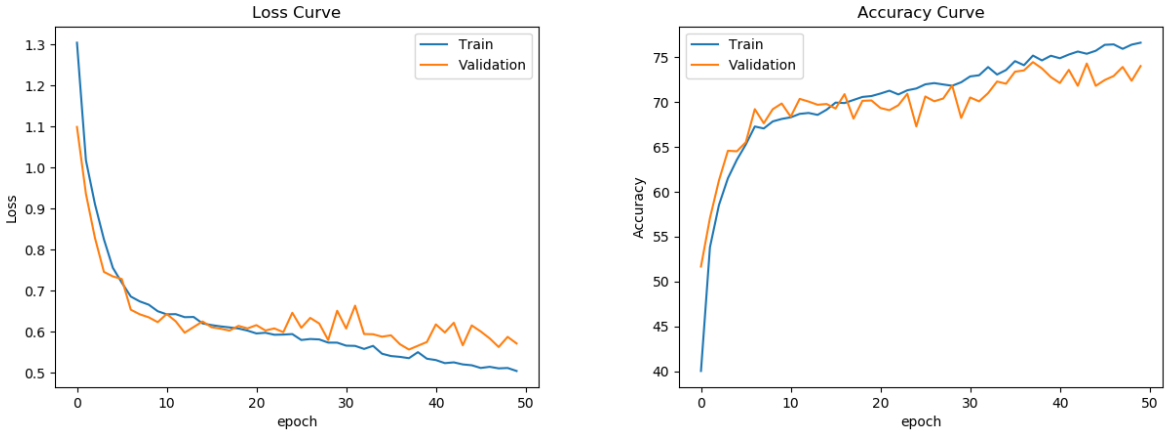


Figure 10: Confusion Matrix

Architecture and Results

We raise the hidden_size from 16 to 32 to take more bottle neck features from RNN, and increase the number of layers from 1 to 2 to raise the complexity of model. In addition, we insert a fully connected layer with FC-16, followed by a Relu activation function, before the FC-5 output layer.

The new architecture decreases the Loss from 0.7 to 0.6, and raises the accuracy from less than 70 to more than 70. As shown in learning curves below.



(a) Loss curves

(b) Accuracy curves

Figure 11: Learning curves

And the improved RNN raises the diagonal elements in confusion matrix. Similarly with improved CNN, we get a better performance between TumorArea and HealthyArea.

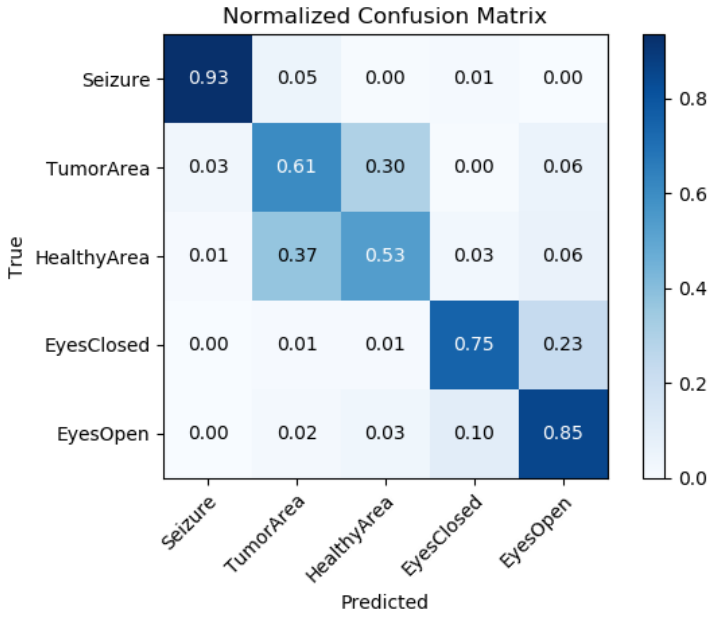


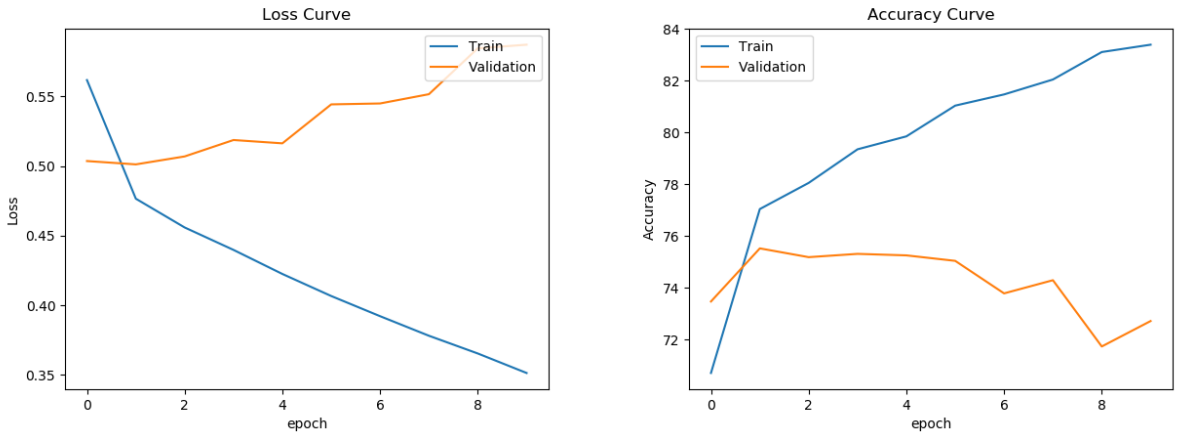
Figure 12: Confusion Matrix

2 Mortality Prediction with RNN

Building Model

Baseline Model

Below is the learning curves and confusion matrix for the baseline model: FC-32, 1-layer GRU-16, FC-2. The best loss is close to 0.5 and best accuracy is about 75.



(a) Loss curves

(b) Accuracy curves

Figure 13: Learning curves

However, the confusion matrix below looks bad and a lot of 0 label has not been predicted. That means the baseline model has a bias to label 1.

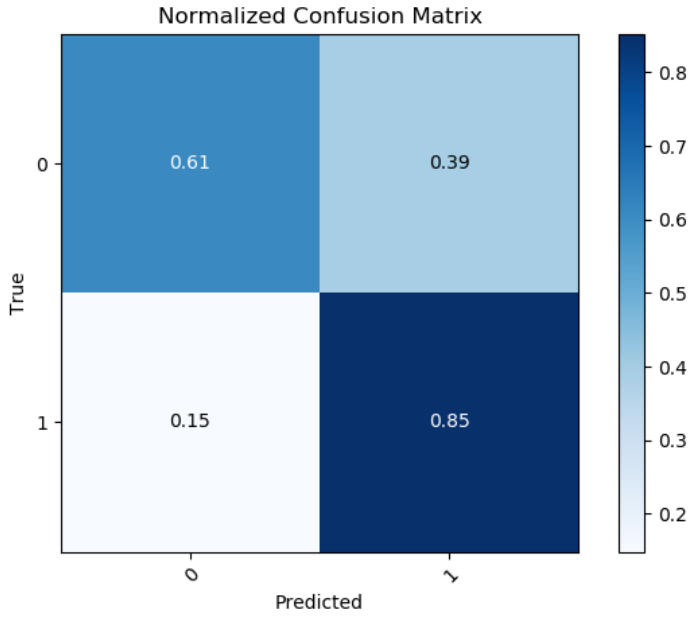


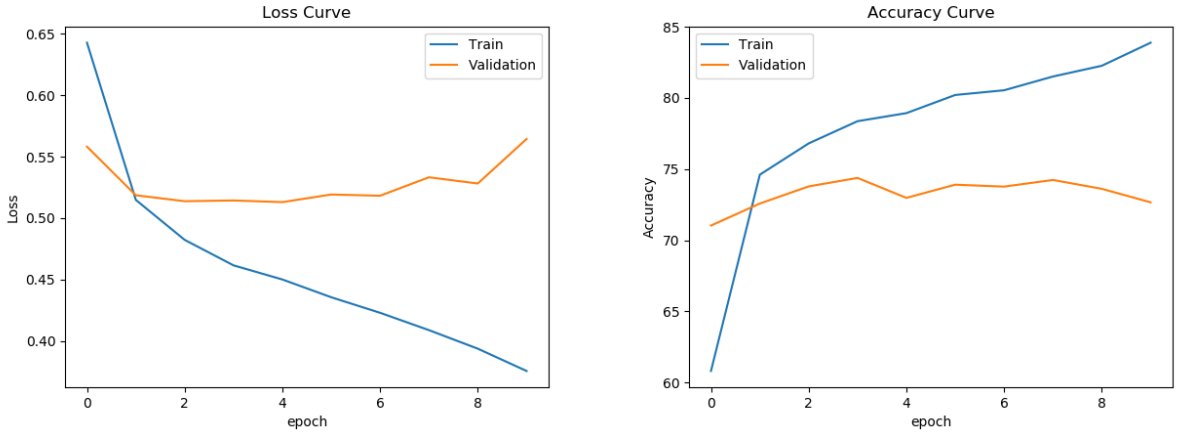
Figure 14: Confusion Matrix, where 0 is dead and 1 is alive

Improved Model

Here is my improved architecture, it is simple, but powerful:

FC-64, 2-layer LSTM-16, FC-8, Relu, FC-2

From the learning curves below, it seems that the new LSTM model performs similar with the baseline GRU model. This is because we use hard label predictions here.



(a) Loss curves

(b) Accuracy curves

Figure 15: Learning curves

However, the confusion matrix shows that our new model has improved much better. Both label 0 and 1 have the comparable accuracy, above 73.

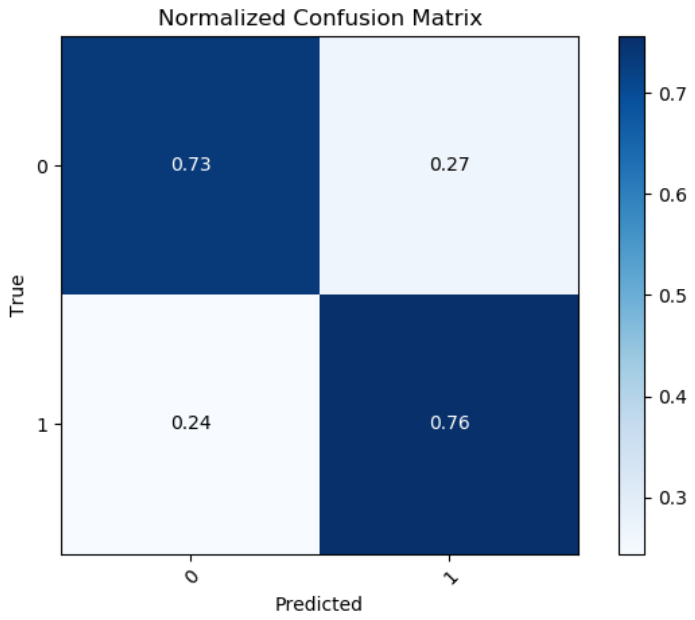


Figure 16: Confusion Matrix, where 0 is dead and 1 is alive

In addition, if we use soft label on our output, the AUC score can be raised to above 0.8.