2110581 BIOINFORMATIC I

Final Report

# non-coding RNA Classification

Pakkapon Wattanawaha 6130391021

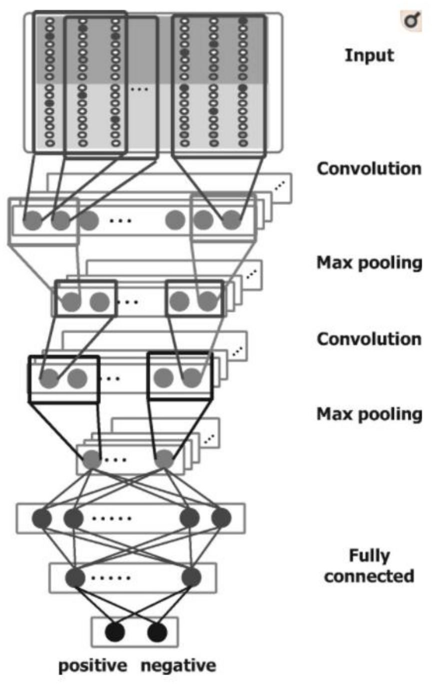
Sathianpong Trangcasanchai 6131050221

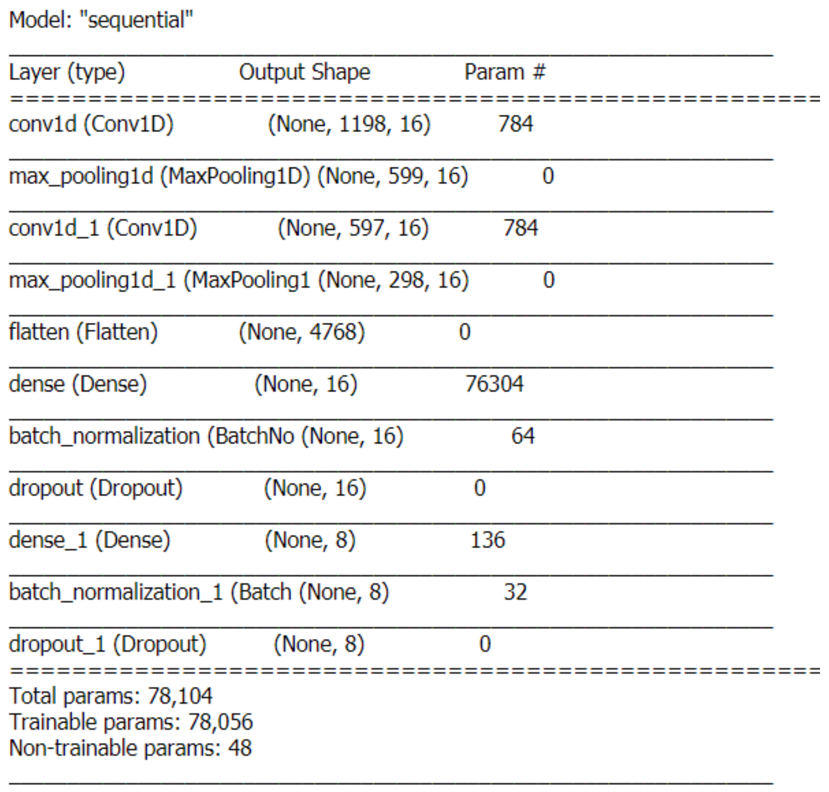
Supawit Sutthiboriban 6130535121

## Report

Our code written in Python consists of 3 parts. First part is to convert data from input FASTA file format to ncRNA matrix that represent pairwise alignment with gaps of two sequences. Second part is to generate ncRNApair data and ncRNApair label from ncRNA matrix from part one. The last part is to use the ncRNA data along with its label to train the neural network to recognize the pattern of data.

In the third part of deep learning, we deploy TensorFlow library to build our convolutional neural network. Our network applies a one-dimensional CNN to accurate clustering of ncRNA sequences, we developed a new CNN-based method for classification of pairwise alignments of ncRNA sequences. The architecture of model is described as the following.





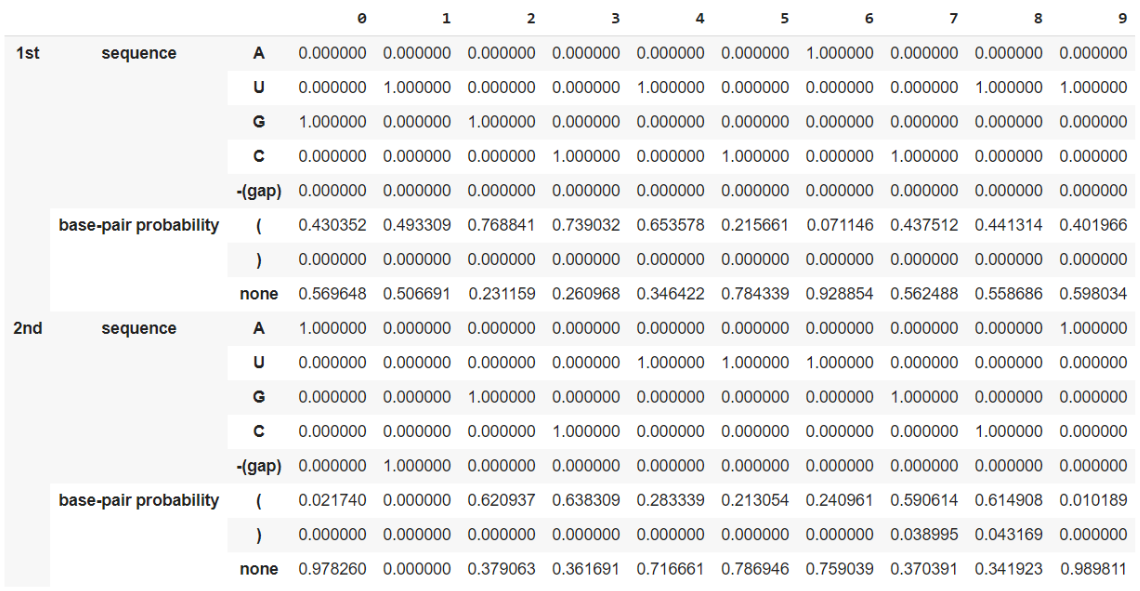
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 1198, 16)          784
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 599, 16)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 597, 16)           784
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 298, 16)           0
_________________________________________________________________
flatten (Flatten)            (None, 4768)              0
_________________________________________________________________
dense (Dense)                (None, 16)                76304
_________________________________________________________________
batch_normalization (BatchNo (None, 16)                64
_________________________________________________________________
dropout (Dropout)            (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 136
_________________________________________________________________
batch_normalization_1 (Batch (None, 8)                 32
_________________________________________________________________
dropout_1 (Dropout)          (None, 8)                 0
=================================================================
Total params: 78,104
Trainable params: 78,056
Non-trainable params: 48
_________________________________________________________________
```

***From the model we explained in progress report 3***, we improved the model by changing the activation function in Fully-connected part of our model from Rectified Linear Unit(ReLu) to Exponential Linear Unit(Elu). The reason of changing to Elu is that we do a research and found that ReLu might suffer from dying neurons. It has proven to be better than ReLU and its variants like Leaky-ReLU(LReLU) and Parameterized-ReLU(PReLU). Using ELU leads to a lower training times and and higher accuracies in Neural Networks as compared to ReLU, and its variants.

The results are the following

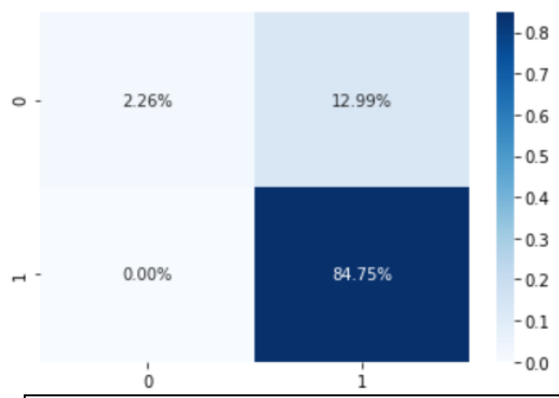
– After the change of activation function to Elu, the accuracy of the model on validation dataset is improved from around 0.9 to 0.95.
– The accuracy on test dataset is slightly improved as well.
– The overall learning time of the model is slightly decreased.

This is the sample of the input for the CNN obtained by One-hot encoding the pair of sequence ncRNA and concatenating the three-dimensional vector for representing secondary-structure information in column *i* consists of left-side base-pairing probability $p^{left}_i$, right-side base-pairing probability $P^{right}_i$ and unpaired probability $p^{unpaired}_i$.

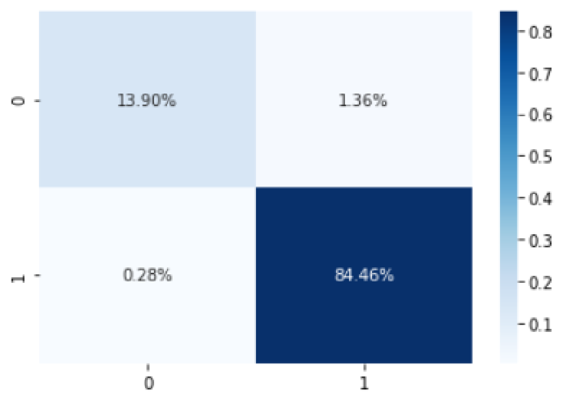


| | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1st | sequence | A | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| | | U | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 |
| | | G | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| | | C | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| | | -(gap) | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| | base-pair probability | ( | 0.430352 | 0.493309 | 0.768841 | 0.739032 | 0.653578 | 0.215661 | 0.071146 | 0.437512 | 0.441314 | 0.401966 |
| | | ) | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| | | none | 0.569648 | 0.506691 | 0.231159 | 0.260968 | 0.346422 | 0.784339 | 0.928854 | 0.562488 | 0.558686 | 0.598034 |
| 2nd | sequence | A | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| | | U | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| | | G | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| | | C | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 |
| | | -(gap) | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| | base-pair probability | ( | 0.021740 | 0.000000 | 0.620937 | 0.638309 | 0.283339 | 0.213054 | 0.240961 | 0.590614 | 0.614908 | 0.010189 |
| | | ) | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.038995 | 0.043169 | 0.000000 |
| | | none | 0.978260 | 0.000000 | 0.379063 | 0.361691 | 0.716661 | 0.786946 | 0.759039 | 0.370391 | 0.341923 | 0.989811 |

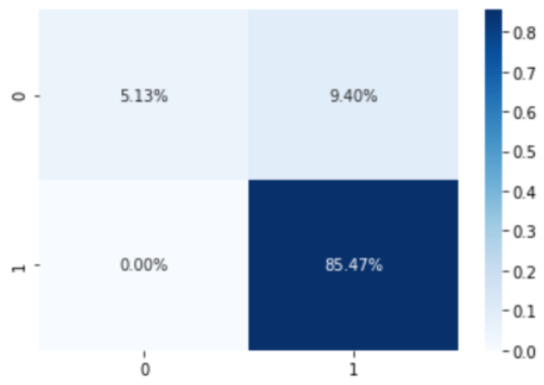
Then, we compile our model using ‘Adam’ optimizer and ‘Accuracy’ for a metrics. We the train the model with 10 epochs and get the final accuracy on training around 90% when using Relu activation and around 98% when using Elu activation. The comparison is sown below. We also deploy confusion_matrix and accuracy_score module from sklearn.metrics to evaluate the model’s performance.



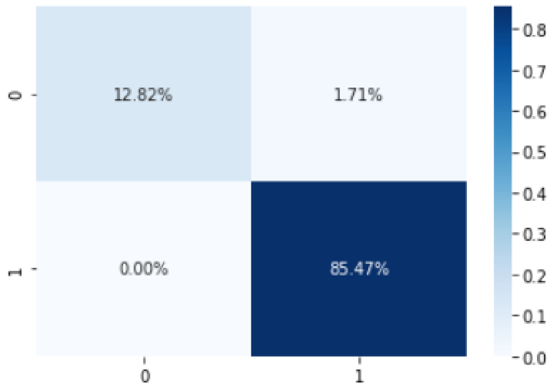
Confusion matrix for predicting ***training set*** when using ***ReLu*** activation function



Confusion matrix for predicting ***training set*** after changing to ***ELu*** activation function



Confusion matrix for predicting ***test set*** when using ***ReLu*** activation function



Confusion matrix for predicting ***test set*** when changing to ***Elu*** activation function

The performance of each method was evaluated as follows. For classification accuracy, given the prediction of one-dimensional CNN for a pair of ncRNA sequences in the test data, the prediction is defined as true positive (TP) if the pair is labeled as the positive class and the prediction is positive. In the same manner, false positives (FPs), true negatives (TNs) and false negatives (FNs) are defined if the pair is labeled as the negative class, but the prediction is positive, the pair is labeled as the negative class and the prediction is negative, or the pair is labeled as the positive class but the prediction is negative, respectively.

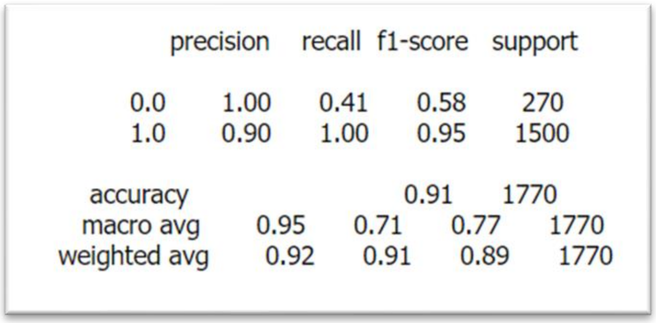
$$\text{Accuracy} = \frac{\#\text{TP} + \#\text{TN}}{\#\text{TP} + \#\text{FP} + \#\text{TN} + \#\text{FN}}$$

$$\text{Precision} = \frac{\#\#\text{TP}}{\#\text{TP} + \#\#\text{FP}}$$

$$\text{Recall} = \frac{\#\#\text{TP}}{\#\#\text{TP} + \#\#\text{FN}}$$

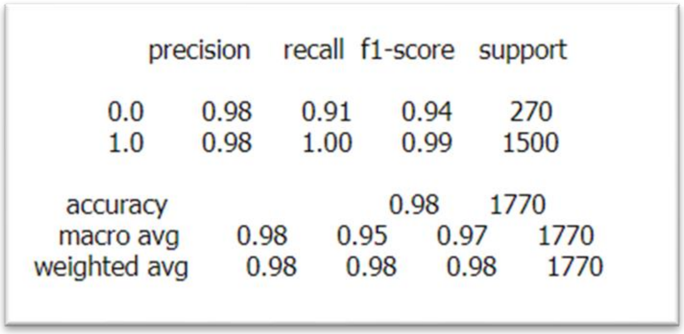
$$F - \text{value} = \frac{2\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

We calculated those measurement using sklearn.metrics.classification_report and this is the result of training set and test set when using different activation functions.



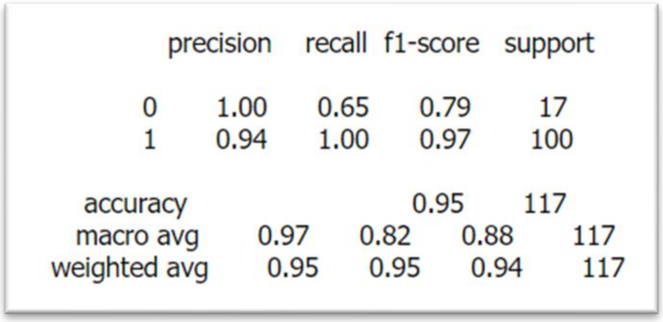
```
              precision    recall  f1-score   support

         0.0       1.00      0.41      0.58       270
         1.0       0.90      1.00      0.95      1500

    accuracy                           0.91      1770
   macro avg       0.95      0.71      0.77      1770
weighted avg       0.92      0.91      0.89      1770
```

Classification report for predicting ***training set*** when using ***ReLu*** activation function



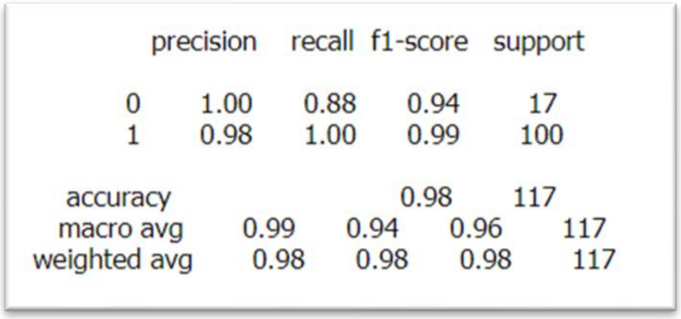
```
              precision    recall  f1-score   support

         0.0       0.98      0.91      0.94       270
         1.0       0.98      1.00      0.99      1500

    accuracy                           0.98      1770
   macro avg       0.98      0.95      0.97      1770
weighted avg       0.98      0.98      0.98      1770
```

Classification report for predicting ***training set*** when using ***ELu*** activation function



```
              precision    recall  f1-score   support

           0       1.00      0.65      0.79        17
           1       0.94      1.00      0.97       100

    accuracy                           0.95       117
   macro avg       0.97      0.82      0.88       117
weighted avg       0.95      0.95      0.94       117
```

Classification report for predicting ***test set*** when using ***ReLu*** activation function

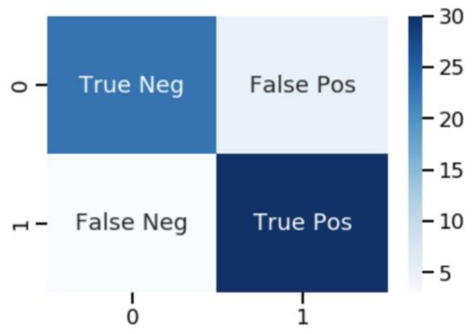


```
              precision    recall  f1-score   support

           0       1.00      0.88      0.94        17
           1       0.98      1.00      0.99       100

    accuracy                           0.98       117
   macro avg       0.99      0.94      0.96       117
weighted avg       0.98      0.98      0.98       117
```

Classification report for predicting ***test set*** when using ***ELu*** activation function

As we see from the confusion matrix and classification report, we can see that after using the exponential linear unit as an activation function the accuracy of both validation and test dataset is improved. Moreover, the False negative and False positive of training and test dataset are decreased according to the confusion matrix shown above.

Source code available on Github :
https://github.com/pakkaponwattanawaha/Bioinfomatics_TermProject



The confusion matrix used in this report is represented in this format.