# Neural Network for MIMIC-III Patients

Brihat Sharma & Neha Goel

***Abstract*—Neural Networks has been producing astonishing results in last few years. Convolution neural networks has been really efficient in image classification task and recently it has also been used in natural language processing such as document classification. Critical ICU MIMIC-III patients was trained using convolution neural network and accuracy of 70% was produced for test dataset.**

## I. Introduction

DOCUMENT classification has always been time and memory consuming task. Traditional machine learning algorithm such as Support Vector Machine, Naive Bayes Classifier has been extensively use to classify documents. Nowadays, with CPU and GPU evolving with speed and memory efficiency, neural network has become the backbone of machine learning. They don't need much feature engineering, and can extract required information by itself using filters.

Convolution neural network was used to classify critical ICU patients' doctor notes. Patient with critical ICU condition still has a surviving chances and the model build using convolution neural network flags the patients which may expire in the hospital itself.

## II. Dataset Description

MIMIC-III is a large publicly available database, which consists of de-identified health related data for about sixty thousand patients, admitted and stayed in critical care units of Beth Isreal Deaconess Medical Center between 2001 and 2012. This database consists of large diverse population of ICU patients.

The dataset consists of text files, where each patients represents a file. MIMIC-III patients database was used to get the required data. Specifically, *SUBJECT_ID* and *TEXT* column was used from the Noteevent Table of the MIMIC-III database. The text column consists of physician notes, imaging reports, laboratory test results, procedures, and many more. All the sensitive patient information are written in the text column and for the document classification task the text column was used for each patients.

## III. Data Process

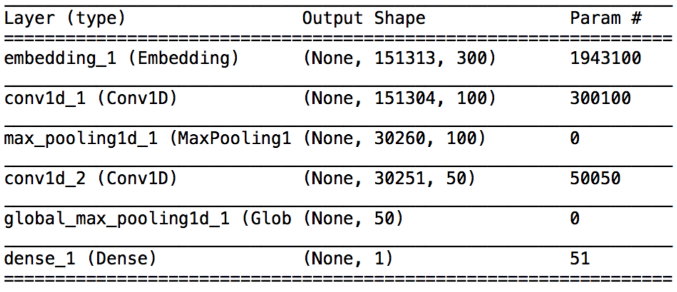
The patient dataset was raw text file where all the information was kept inside the single file for each patient. Initial data process task was to extract features from the raw text file using Apache cTakes. Apache cTakes is a feature extraction software for natural language processing task, specifically for electronic medical record clinical free-text. cTakes extracts medical related information converts them into CUI, these CUI are then found in UMLS website. Working with the CUIS are easier, efficient and at the end CUIs can be searched in UMLS website to find the information on what they actually mean.

Advisor: Dmitiry, Lehrstuhl für Nachrichtentechnik, TUM, WS 2050/2051.

## IV. Baseline Approach

The given dataset consisted of files which represented an individual patient. Hence, each file needed to be classified into either positive class: died or negative class: survived. This was a binary document classification problem, where each document had encoded string representing symptoms of the patient. Initial approach was to read each document, use tokenization to break the document into individual encoded string, and count the number of occurrence of each encoded string. The frequency of each string can be normalized and different machine learning algorithm can be implemented to train the documents, and use the trained modeled on test set to flag the ICU Patients.

## V. Method Description



```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 151313, 300)       1943100
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 151304, 100)       300100
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 30260, 100)        0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 30251, 50)         50050
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
```

Fig. 1. CNN Model Summary

First layer of the Convolution neural network model was embedded layer. Embedded layer takes each CUI and convert them into vectors. This is a slightly different representation of the input data then the regular one-hot representation. Each CUI is represented by 300 vectors, and this vectors are weights of a neural network where the input data is equal to the output data. This process is famously known as word2Vec converter.

Second layer of the model is one dimensional convolution layer, which takes one dimensional input vector. This layer contains number of filter as 100, and strides as 10. The filter will act on 10 element at once. The activation function provided was as relu.

Third layer of the model was max pooling, which is basically feature reduction. Now the input vector will change after the max pooling, which can then again be used for another layer of neural network.

Fourth layer of the model was again one dimensional convolution neural network. For this layer, filter size used was 50 with strides as 10. This layer was followed by global max pooling, which output the maximum weight input vector.

Last layer of the neural network was Dense layer, which had optimizer as rmsprop,loss as binary crossentropy and activation function as sigmoid function. Rmsprop is an optimization on SGD optimizer, where rmsprop adjusts the step size so that step is in the same scale as the gradient. Binary crossentropy is for binary prediction and sigmoid function squeezes the output between 0 and 1.

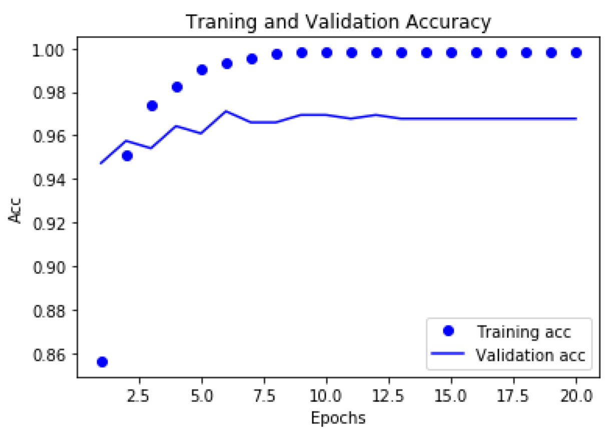


Fig. 2. Convolution Network Analysis Graphs

## VI. Evaluation

**Prediction outcome**

| | | **p** | **n** |
|---|---|---|---|
| **actual value** | **p′** | TP 425 | FN 10 |
| | **n′** | FP 11 | TN 246 |

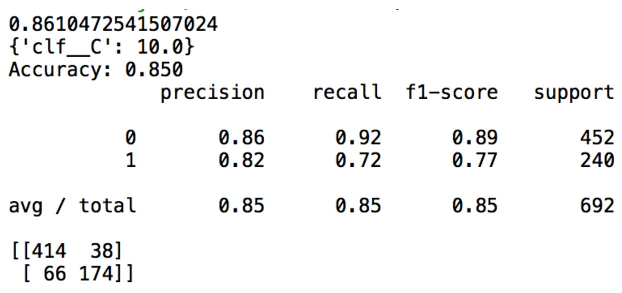
The result of convolution neural network was compared with that of logistic regression. The accuracy when using logistic regression was 86 %, and that of convolution neural network was 96%. From the confusion matrix above we can see that, true positive was 425 compared to that of 414 for logistic regression and true negative was 246 and for logistic regression was 174.

In both cases, model predicted better for positive class. This was predicted before hand since the data for the positive class was more than that of the negative class.

Most of the time convolution neural network stop updating the weights after 10 epoch, this could have been solved if more number of data was used. This would also decrease over fitting and output better accuracy on the test data.

TABLE I
Epochs Analysis for Train and Validation Data's Loss and Accuracy

| EPOCHS | LOSS | ACCURACY | ValLoss | ValAcc. |
|---|---|---|---|---|
| 1 | 0.4033 | 0.8563 | 0.1377 | 0.9473 |
| 2 | 0.1786 | 0.9507 | 0.1317 | 0.9575 |
| 3 | 0.1284 | 0.9739 | 0.2362 | 0.9541 |
| 4 | 0.0842 | 0.9826 | 0.1660 | 0.9643 |
| 5 | 0.0746 | 0.9904 | 0.2359 | 0.9609 |
| 6 | 0.0592 | 0.9937 | 0.2165 | 0.9711 |
| 7 | 0.0434 | 0.9952 | 0.2760 | 0.9660 |
| 8 | 0.0337 | 0.9973 | 0.2940 | 0.9660 |
| 9 | 0.0288 | 0.9982 | 0.2766 | 0.9694 |
| 10 | 0.0288 | 0.9982 | 0.2636 | 0.9694 |
| 11 | 0.0288 | 0.9982 | 0.3122 | 0.9677 |
| 12 | 0.0288 | 0.9982 | 0.2672 | 0.9694 |
| 13 | 0.0288 | 0.9982 | 0.2820 | 0.9677 |
| 14 | 0.0288 | 0.9982 | 0.2820 | 0.9677 |
| 15 | 0.0288 | 0.9982 | 0.2820 | 0.9677 |
| 16 | 0.0288 | 0.9982 | 0.2820 | 0.9677 |
| 17 | 0.0288 | 0.9982 | 0.2820 | 0.9677 |
| 18 | 0.0288 | 0.9982 | 0.2820 | 0.9677 |
| 19 | 0.0288 | 0.9982 | 0.2820 | 0.9677 |
| 20 | 0.0288 | 0.9982 | 0.2820 | 0.9677 |



```
0.8610472541507024
{'clf__C': 10.0}
Accuracy: 0.850
            precision    recall  f1-score   support

          0       0.86      0.92      0.89       452
          1       0.82      0.72      0.77       240

avg / total       0.85      0.85      0.85       692

[[414  38]
 [ 66 174]]
```

Fig. 3. Logistic Regression Analysis

## VII. Discussion

We observed that Convolution Neural Network produce better results with more data and more epochs. Initially we tried with 150 patients and 10 epochs which was giving only 65 accuracy. As we modeled with the whole data set and 20 epochs we could get the accuracy of 96. We also had a comparison analysis with Logistic Regression classification which gave a 85 accuracy thus helping us understand how hidden layers within neural network can increase the efficiency and accuracy of the prediction model.

## VIII. Conclusion

The overall goal of our project was to predict the survival of the patients from the MIMIC data. Our convolution network did a pretty good job with 96 accuracy. We also believe that with increased batch size we could have some really visual clear graphs. In conclusion, this project helped us understand convolution neural network and its hidden layers, and gave us first hand experience working with real clinical dataset.

## IX. Appendix

All team members had access to the code. Each team member ran the code to verify the results and then reimplemented it, using other parameters. The contributors for the project were Brihat Sharma and Neha Goel, with the help of Dr. Dmitriy Dligach, Loyola Unversity Chicago