Deep Learning architectures for ECG classification

Javid Guliyev [1] DSAI, French-Azerbaijani University
Mahir Israyilov [2] DSAI, French-Azerbaijani University

Abstract

This paper is about Deep Learning models for Time series. We used two types of Deep Learning architectures for classification. Convolutional Neural Network and Recurrent Neural Network. We constructed and made comparison between deep learning models via the EGG200 dataset. We shared the results in the experimental results section of the paper. We also implemented robust models for classification between normal and abnormal ECG signals. Our model can differentiate with high accuracy between normal and abnormal ECG signals.

1. Introduction

The Deep Learning Specialization is a foundational program that will prepare us to participate in the development of cutting-edge AI technology by helping us comprehend the capabilities, problems, and repercussions of deep learning. We have trained neural network architectures including Convolutional Neural Networks, Recurrent Neural Networks, LSTMs, and Transformers, as well as how to improve them with tactics like Dropout, BatchNorm, and more in this Specialization. We'll use Python, TensorFlow and Keras to grasp theoretical principles and their industry applications, tackling real-world problems including speech recognition, music synthesis, chatbots, machine translation, natural language processing, and more. Many sectors are being transformed by artificial intelligence.

[1]Data Science and Artificial Intelligence, French-Azerbaijani University, Baku, Azerbaijan [2]Data Science and Artificial Intelligence, French-Azerbaijani University, Baku, Azerbaijan. Correspondence to: Javid Guliyev <javid.guliyev2@ufaz.az>, Mahir Israyilov <mahir.israyilov@ufaz.az>.

2. Convolutional Neural Network and Recurrent Neural Network

2.1. Convolutional Neural Network

A Convolutional Neural Network, or CNN, is a type of artificial neural network that has been widely utilized for image analysis. Although image analysis has been CNN's most widely used application. It can also be used to solve various data analysis and categorization issues. A CNN can be thought of as an artificial neural network with some form of specialty for picking out or detecting patterns and making sense of them. CNN's ability to discover patterns is what makes it so valuable for picture analysis. So, if a CNN is just a type of artificial neural network, what sets it apart from a typical multi-layer perception or natural language processing system? A CNN, on the other hand, has hidden layers known as convolutional layers, and these layers are what distinguishes a CNN. Now, CNNs can and usually do have non-convolutional layers as well, but the convolutional layers are the foundation of a CNN. A convolutional layer receives data, transforms it in some way, and then sends the transform input to the next layer, much like any other layer. This transformation is a convolution operation when using a convolutional layer. Patterns and visuals can be created using convolutional neural networks. The convolutional layers can detect patterns more precisely. We must indicate the number of filters that each convolutional layer should have. It's important to remember that these filters are what detect the patterns.

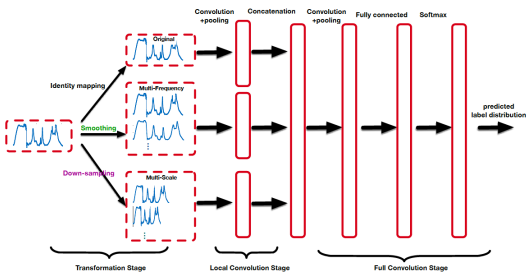

Figure 1. Convolutional Neural Network

2.2. Recurrent Neural Network

Recurrent Neural Network also known as RNN is used mainly for natural language processing tasks. Convolutional Neural Networks are mainly for images, But Recurrent Neural Networks are mainly for NLP (Natural Language Processing). A Recurrent Neural Network (RNN) is a form of artificial neural network that works with time series or sequential data. These deep learning algorithms are often employed for ordinal or temporal issues like language translation, natural language processing (nlp), speech recognition, and image captioning, and they're used in popular apps like Siri, voice search, and Google Translate. What does it mean? Google has Recurrent Neural Network embedded into it where, when we type in a sentence "not interested at", it will auto complete with "this time". If we say "we will let you know if it changes" it will also say "in the future". So this saves our time. It will write the sentence for us. Recurrent neural networks, like feedforward and convolutional neural networks (CNNs), learn from training input. They are distinguished by their "memory," which allows them to impact current input and output by using knowledge from previous inputs. While typical deep neural networks presume that inputs and outputs are independent of one another, recurrent neural networks' output is reliant on the sequence's prior elements. While future occurrences may be useful in determining a sequence's output, unidirectional recurrent neural networks cannot account for them in their predictions.

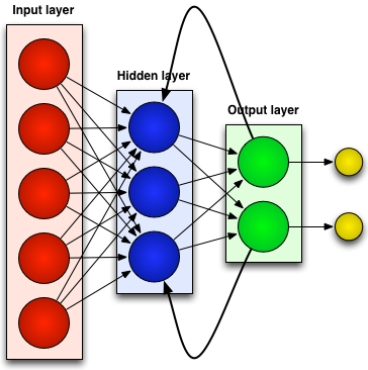

Figure 2. Recurrent Neural Network

3. Time Series Data and Regularization Techniques

3.1. Time Series Data

In mathematics, a time series is a series of data points indexed in chronological order. As a rule, a time series is a sequence drawn at the same time at separate points. Thus, it is a sequence of discrete-time data. A set of observations gathered through repeated measurements over time is known as Time Series data. If we plot the points on a graph, time will always be one of the axes.

Because time is a component of everything observable, Time Series data may be found everywhere. Sensors and systems are always releasing a continuous stream of time series data as our world becomes increasingly instrumented. Such information can be used in a variety of sectors. Using some examples, we can put this into context.

Time series data can be useful for:

Tracking daily, hourly, or weekly weather data Tracking changes in application performance Medical devices to visualize vitals in real time Tracking network logs EEG and ECG signals obtained during sleep Smartphone GPS and signal strength measurements of urban travellers Electricity demand for a city and etc.

3.2. Regularization Techniques

There are also some regularization techniques. For example, we can say that L2 and L1 regularization, Dropout, Data augmentation, Early stopping. In our models, we used some of regularization techniques. Dropout and Early Stopping, L1 regularization. One of the more intriguing sorts of regularization techniques is Dropout. The dropout cancels randomly neurons from the model and makes model more robust. It also generates excellent results, and as a result, it is the most often used regularization technique in deep learning. Early stopping is a cross-validation approach in which a portion of the training set is used as the validation set. When we notice that the model's performance on the validation set is deteriorating, we immediately cease training it. This is referred to as "early stopping."

4. Experimental Results

4.1. Setup

Experimental Setup. Our experiments are processed on the Google Colab platform. The parameters of Google Colab are NVIDIA K80 / T4 GPU, GPU Memory 12GB. The GPU Memory Clock is 0.82 GHz and has 4.1 TFLOPS performance, 12GB RAM storage.
On Model parameter side
All models using categorical crossentropy loss function. The validation test split is 0.3, batch size 30, epoc= 500, adam optimizer and callbacks. The model checkpoint callback are given to all models while training. This callback saves best version of you model and you can load it.

4.2. Dataset

Our dataset is ECG200 dataset. This dataset collected by R. Olszewski. And the dataset contains 2 classes one of

them is normal and the other one is ischemia. This classes represents the labels for the heart beat. We will face some problems when we will use the dataset.
The one of the main problems is the dataset is not equally distributed between classes. The sample count was too small. 69 for class 1 and and 31 for the class 2.

4.3. Proposed models

Our problem is a classification problem. So we have two main options for making the classification. RNN and CNN. We have tried both of the architectures. We have created 8 different models. and evaluated all of them. In the end, we have done regularization techniques.
We proposed that we will try 3 different convolution1D models and will select the better accuracy one. And then we'll do regularization techniques to increase model performance and decrease model overfilling.

4.3.1. CNN MODEL

In figure 3 we can see our first model. We have 3 conv layer. Before activation function we do Batch Normalization. with this layers we can standardize data that flows from the model. With this model we get 82% accuracy and loss is shown in figure 4.

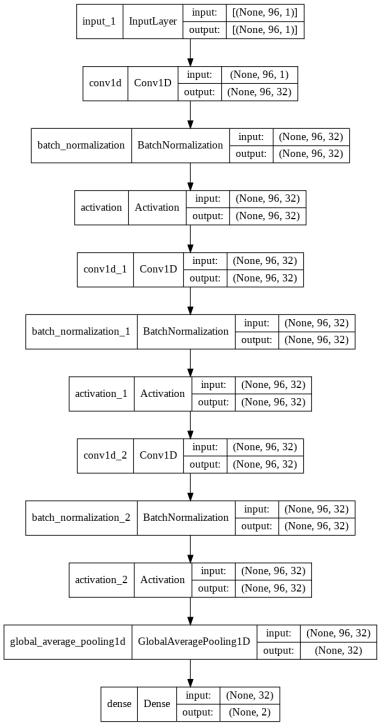

Figure 3. CNN Model

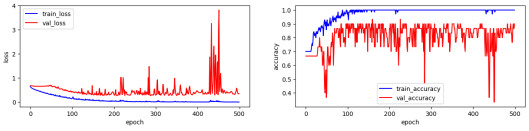

Figure 4. CNN Model Accuracy and Loss

4.3.2. LSTM MODEL

In figure 5 we can see our LSTM model. LSTM is Long short-term memory. it is RNN architecture. We have 3 conv layer. In our model we used 24 neurons and added Dropout layers 0.5 because we don't want to our model overfit. Although we used 0.5 dropout we had overfitting. even with overfitting we have 80% accuracy on test set.

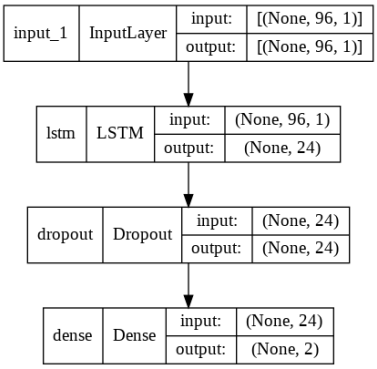

Figure 5. LSTM Model

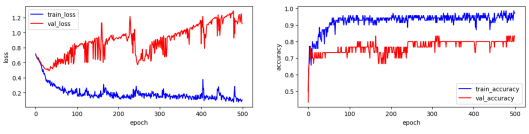

Figure 6. LSTM Model Accuracy and Loss

4.3.3. REGULARIZATION

In figure 7 we show the CNN model that applied with some regularization techniques. We applied 0.5 dropout, l1 regularization, and as you can see in the figure 8 we have good loss and accuracy values compared to older implementations. And figure represents all the model itself. Using this methods we can get 86% accuracy and 0.56 loss on testing set.

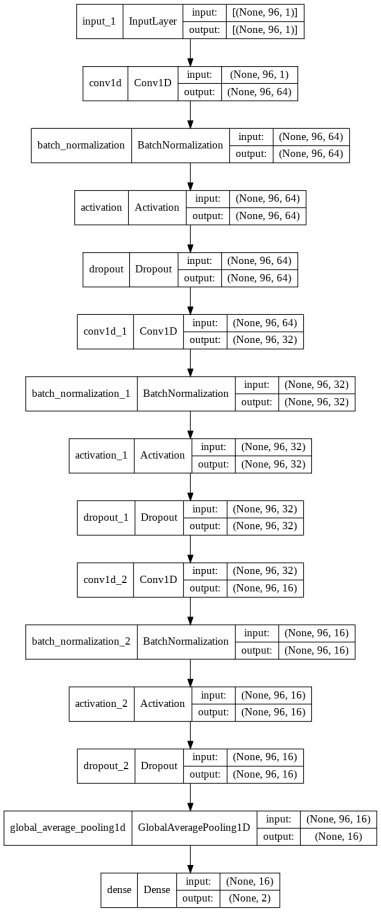

Figure 7. CNN Regularization used Model

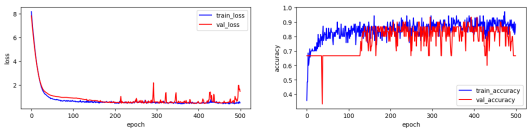

Figure 8. Results

5. Unbalanced dataset

We have seen that the class one is more than 2 times less than class 0; Because of that we used downsampling method to make classes equal. Unfortunately we didn't get good results because of the sample count is too low also. We got 73% accuracy and 1.0 loss.

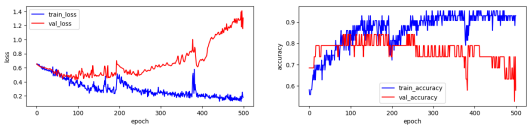

Figure 9. Unbalanced Data

6. Robustness of the evaluation

Because of checking our robustness of a model we Compile and Fit our models 5 times and we calculate standard deviation and mean of all five evaluation. in this image we can see the 3 train and validation , loss and accuracy scores.

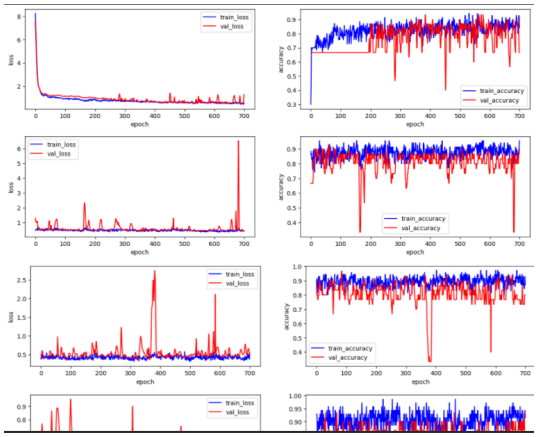

Figure 10. Robustness of the evaluation

7. Visualization

We have implemented Class Activation Map for visually appearance of single sample This map shows the regions of a single sample have been selected by model. In this way, we can better understand our model.
Using confusion matrix can help us visually understanding the prediction of model.

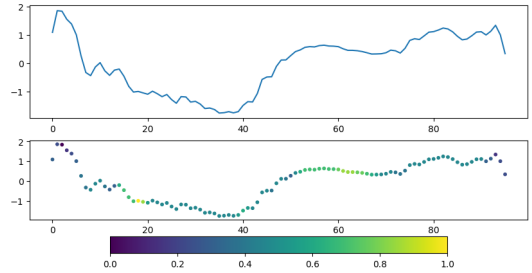

Figure 11. Class Activation Map

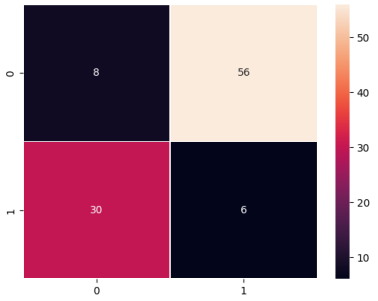

Figure 12. Confusion Matrix

8. Conclusion

The main goal of this project was to implement different model types and explain their behavior on EDG200 dataset. Doing the regularization techniques such as dropout, l1, early stopping and etc. We have taked about the model performances and how to boost model performance on this type of models. we measured the robustness of our models and have implemented visualization techniques to provide a deeper understanding of the dataset and samples from the dataset

9. References

[1] https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network
[2]https://www.ibm.com/cloud/learn/convolutional-neural-networks
[3]https://www.ibm.com/cloud/learn/recurrent-neural-networks
[4]https://www.influxdata.com/what-is-time-series-data/
[5]https://jmread.github.io/talks/Time_Series_AI.pdf
[6]https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/
[7]https://www.youtube.com/watch?v=Y2wfIKQyd1I ab_channel=codebasics
[8]https://miro.medium.com/max/1400/1*H3JzmYY38w-awkqA02rEMQ.png
[9]https://miro.medium.com/max/375/1*chs1MCz2rCK4_dFRLnUEIg.png
[10]https://www.coursera.org/specializations/deep-learning
[1] Time Series Classification Website. (2001). R. Olszewski. http://www.timeseriesclassification.com/description.php?Dataset=ECG200