**Deep Learning and Time Series**

**Deep Learning**

**Coursework 2- ECG-based Anomaly Detection**

|  |  |  |
| --- | --- | --- |
| Hagar Shalabi | 206002 | Autoencoder-based models |
| Mohammed Khalid | 200961 | LSTM-based models |
| Yossuf Yasser | 197032 | CNN-based models |

**Content:**

1. **Introduction**
2. **Data Choice**
3. **Data Pre-processing**
4. **ECG-dataset on Autoencoder-based models comparisons**
5. **ECG-dataset on LSTM-based models comparison**
6. **ECG-dataset on CNN-based models comparison**
7. **Last Comparison of the best models**
8. **Introduction**

ECG-based anomaly detection is used to detect potential health problems one might face and also predict if a person would eventually develop a health issue. A person develops such health issues through many factors and ECG readings would support detecting the issues. Like, an abnormal ECG reading would mean there would be an irregular heartbeat, resulting in a presence of a heart condition, an artery blockage in the lungs, or a heart defect. Normally, the anomaly detector (either autoencoder, LSTM or CNN-based) would differentiate between the normal and the unusual data points.

1. **Data Choice: http://www.timeseriesclassification.com/description.php?Dataset=ECG5000**

The dataset used in based on Electrocardiograms (ECG signals) represent the electrical activity of a person’s heart over 10 seconds per reading. The ECG-based anomaly detection dataset contains 50k rows, each representing a person’s ECG readings. The dataset’s attributes are the ECG reading, the person’s age, ethnicity and other features. Now, we’ll like to see how a person’s ECG readings change according to age, gender, ethnicity, lifestyle, and other factors. Anomaly detection is a process that finds data points that deviate from normal behaviour data points, so this can be applied here to identify abnormal data points (anomalies) in the ECG signal. By and large, the dataset was made to support the detection of abnormal heart rhythms (arrhythmias) and the recordings were made through an ECG device, to also capture the location of the arrhythmia with its type.

1. **Data Pre-processing**
2. *Retrieving ECG-dataset*

The ECG dataset comprises of 2 files of training and testing set, both are read into a dataframe. Then, cleaning of the dataframe is done, like dropping identifier columns and renaming values into the target class to remove weird text from valuable data.

1. *Data Augmentation- Resampling*

To improve any model’s performance, balancing the dataset’s classes is used. This is done by resampling the dataset specifically oversampling. Oversampling aims to balance class distribution by increasing the minority class instances by copying them into the dataset.

1. *Splitting dataset*

The dataset is then split into 80:20 sets for training and testing respectively.

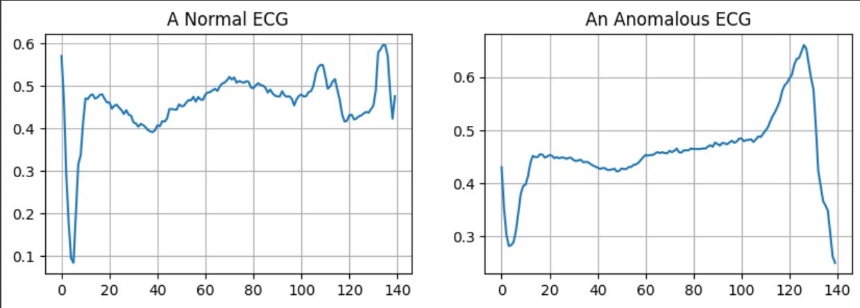
1. *Normalizing dataset*

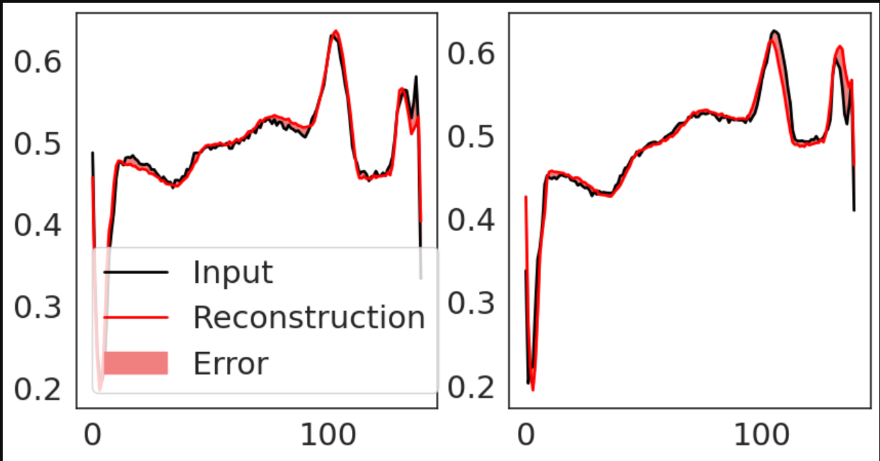
All the sets are standardized into the ranges between 0 and 1.

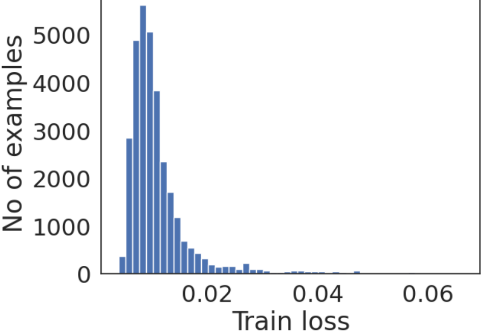
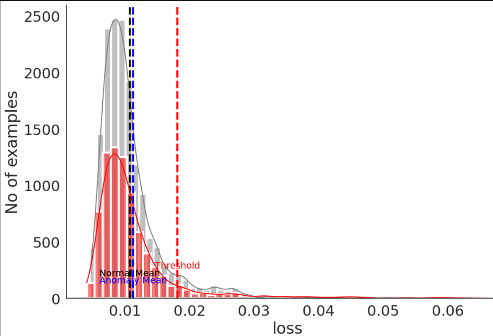
1. **ECG-dataset on Autoencoder-based models comparisons**

**Autoencoder code files- https://drive.google.com/drive/folders/1u4G2I7Sqxb\_a0zV9RDWXX9vQsFzWMF3-?usp=sharing**

Autoencoders are self-supervised neural networks, which means that they have no labels to be trained on, instead they generate their own labels from the training set during training. That being that autoencoders have shown massive success with finding anomalies in time series by encoding (compressing) the input sequence and measuring the difference between the actual and the predicted data, through decoding (decompressing) the data. Accordingly, the autoencoders can predict the next value of the time-series solely from the known data, developing a predictive model.

**ECG-dataset with 2 classes (Normal and Anomalous ECG signals):**

****** ***Figure 1, Examples of instances in the 2 class dataset (row 0)***

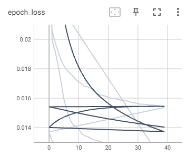
***Figure 2, Examples of instances in the 2 class dataset with reconstruction errors***

***Figure 3, The overview of the data***

* **Model used here was inspired by: https://keras.io/examples/timeseries/timeseries\_anomaly\_detection/**
* Constants: epochs=40, batch\_size=128, loss=mae, learning\_rate=0.01
* Variants with Tanh Activation Function at the model’s output layer with different optimizers

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Adam** | **RMSprop(BEST)** | **SGD** |
| Loss curve | * Robust | * Unstable | - **🡪** Robust |
| Loss curve with training only |  |  |  |
| Metrics |  |  |  |
| Threshold: 0.0182  Accuracy:0.51413  Precision: 0.5068  Recall:0.9303 | Threshold: 0.0212  Accuracy:0.5218  Precision: 0.5111  Recall:0.9313 | Threshold: 0.0190  Accuracy:0.5098  Precision: 0.5045  Recall:0.920 |

**Comments:**

* The highest accuracy was done by the model variant with the RMSprop optimizer, 0.1 more than the second highest variant, with Adam as its optimizer.
* Theoretically, this is plausible because RMSprop faster to change its directions to reach the minimum.
* Also, the RMSprop-based variant is the most precise and has the highest True Positive Rate (TPR) relative to the others by a small percentage.
* On the other hand, the training of the SGD variant is robust relative to RMS variant, as shown above, the SGD variant has a good balance between bias and variance.
* ***(Worst optimizer)SGD***🡪***Adam***🡪***RMSprop(Best optimizer)***
* ******Regardless of all that, the highest accuracy is still bad, despite data augmentation being done. 🡪 Sigmoid activation function is then done on the same dataset.
* ***Note: Tensorboard was done but its results are off.***

***(I won’t rely on it, but it’s in my files!)***

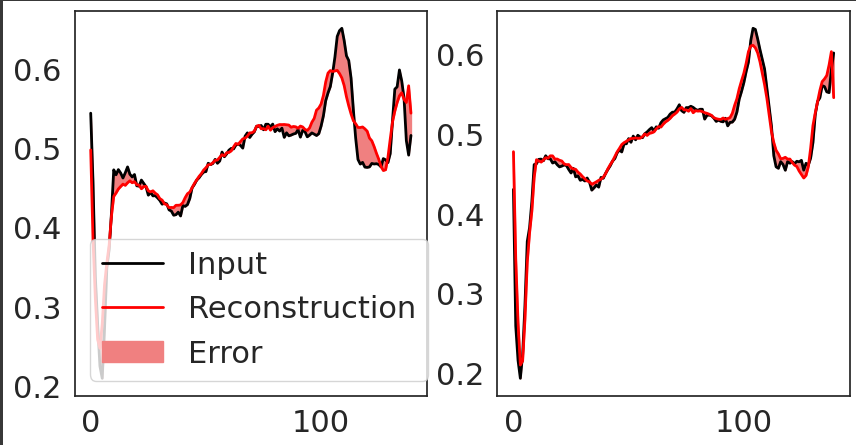
* Variants with Sigmoid Activation Function at the model’s output layer with different optimizers

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Adam** | **RMSprop(BEST)** | **SGD** |
| Loss curve |  |  |  |
| Loss curve with training only |  |  |  |
| Metrics |  |  |  |
| Threshold: 0.01753  Accuracy: 0.5068  Precision: 0.5028  Recall:0.9207 | Threshold: 0.01850  Accuracy: 0.5079  Precision: 0.5034  Recall:0.9227 | Threshold: 0.01733  Accuracy: 0.5061  Precision: 0.5025  Recall:0.9135 |

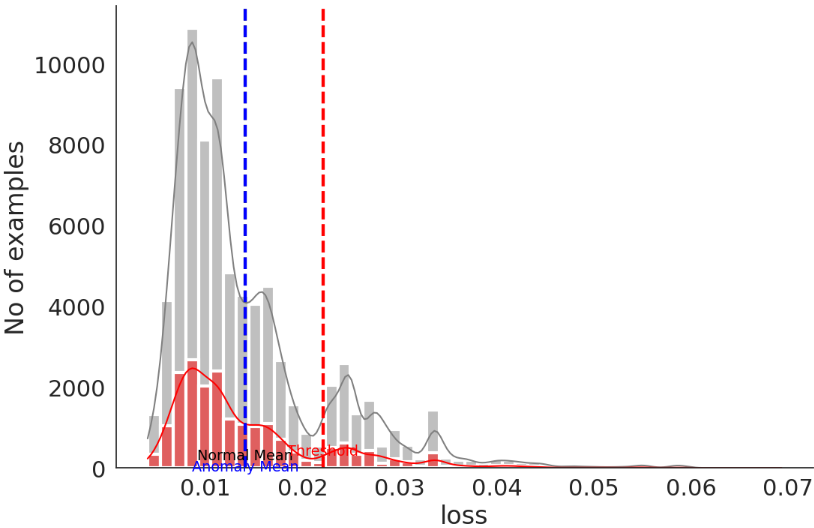
**Comments:**

* Similar to the Tanh activation function variant, the highest accuracy was done by the model variant with the RMSprop optimizer, 0.1 more than the second highest variant, with Adam as its optimizer.
* Also, the RMSprop-based variant is the most precise and has the highest True Positive Rate (TPR) relative to the others by a small percentage.
* On the other hand, the training of the SGD variant is robust relative to RMS variant, as shown above, the SGD variant has a good balance between bias and variance.
* Regardless of all that, the highest accuracy is still bad, despite data augmentation being done. 🡪 Changing the dataset to a bigger ECG dataset

**ECG-dataset with 5 classes (Normal and 4 different heart conditions):**



***Figure 4, Examples of instances in the 2 class dataset (row 0)***

******

***Figure 3, The overview of the data***

* Constants: epochs=20, batch\_size=512, loss=mae, learning\_rate=0.01
* Variants with Tanh Activation Function at the model’s output layer with different optimizers

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Adam** | **RMSprop** | **SGD (BEST)** |
| Loss curve | 🡪 Little unstable | 🡪 Unstable | 🡪Robust (not over/under fitting) |
| Loss curve with training only |  |  |  |
| Metrics | Threshold:0.02679  Accuracy:0.8541  F1 score: 0.9213  Loss:0.0169 | Threshold:0.02679  Accuracy:0.5948  F1 score: 0.7459  Loss:0.0267 | Threshold:0.03927  Accuracy:0.8765  F1 score: 0.9342  Loss:0.049 |

* The highest accuracy was done by the model variant with the SGD optimizer, 0.2% more than the second highest variant, with Adam as its optimizer.
* Theoretically, SGD optimizer is better than Adam in this dataset, solely because it converges faster towards zero at the flat valley, having an overall better generalization.
* Also, the SGD-based variant is the most precise and has the highest True Positive Rate (TPR) relative to the others by a 0.1% (F1 score is a combination of precision and recall, so the higher the F1 score, the most precise the model is).
* Also, the training of the SGD variant is robust relative to Adam variant, as shown above, the SGD variant has a good balance between bias and variance.

***(Worst optimizer)RMSprop🡪Adam🡪SGD(Best optimizer)***

* Variants with Sigmoid Activation Function at the model’s output layer with different optimizers

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Adam** | **RMSprop** | **SGD(BEST)** |
| Loss curve | 🡪 Little unstable | 🡪 Unstable | 🡪Robust (not over/under fitting) |
| Loss curve with training only |  |  |  |
| Metrics | Threshold: 0.02179  Accuracy: 0.8655  F1 score: 0.9279  Loss: 0.0137 | Threshold: 0.02179  Accuracy: 0.6393  F1 score: 0.7799  Loss: 0.0215 | Threshold: 0.03355  Accuracy: 0.8907  F1 score: 0.9422  Loss: 0.0196 |

* The highest accuracy was done by the model variant with the SGD optimizer, 0.25% more than the second highest variant, with Adam as its optimizer.
* Theoretically, SGD optimizer is better than Adam in this dataset, solely because it converges faster towards zero at the flat valley, having an overall better generalization.
* However, the variant with the highest F1 score (the highest precision and recall) is the Adam variant.

To conclude- the best Autoencoder model variant is:

|  |  |  |
| --- | --- | --- |
|  | Tanh AF+SGD optimizer | Sigmoid AF+SGD optimizer(BEST) |
| Accuracy(%) | 87.6% | 89.0% |

* The best autoencoder-based model on this dataset is the SGD optimizer with the Sigmoid activation function.

1. **ECG-dataset on LSTM-based models comparison**

The Long Short-Term Memory (LSTM) model is an unsupervised special case of the Recurrent Neural Network (RNN) class is the Long Short-Term Memory (LSTM). It was constructed to solve the vanishing gradient problem of the RNN. The LSTM model consists of an input, an output, and a forget gate modules. The forget gate was the lately added to reset the network state. The LSTM architecture is made up of memory blocks, which are recurrently linked sub-networks. The memory block's concept is to retain its state over time while regulating information flow via non-linear gating units. The first proposed model of LSTM contained only cells, input and output gates, and it went developed adding a forget gate, and a peephole connection.

A number of trials was done over LSTM and Bi-Directional LSTM Manual changing the Optimizer and the number of layers and the dropout. The following was the results of the two highest models from both variations.

* Epochs was fixed as 10.
* Batch size fixed at 64.

|  |  |  |
| --- | --- | --- |
|  | LSTM [TWO layers x 64 cell no dropout Adam optimizer] | Bi-Directional LSTM [TWO layers x 64 cell with dropout Adam optimizer](BEST) |
| Accuracy(%) | 97.69% | 98.09% |

Bi-Directional LSTM shows it to be working better with High dimensional data.

Following using Keras Tuner to tune the hyperparameters showing that:

* Increase the LSTM layers with high dimensional data increases the accuracy.
* Sigmoid activation function affects the LSTM layer better than Relu.
* Increase the learning rate decrease the model training accuracy.

1. **ECG-dataset on CNN-based models comparison**

**https://colab.research.google.com/drive/1wtucss4ZXUNiuXgbrHzqGzchN6fob0yj?usp=sharing**

The deep CNN model is trained on a large dataset of ECG heart rate signals and with myocardial infarction types. The model learns to identify patterns signals that are associated with each type of the disease and then uses this information to make predictions on new ECG signals. We can evaluate it by using normal evaluation methods for classification tasks like accuracy, F1 score and confusion matrix.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Model 1 | Model 2 | Model 3 |
| Number of conv layers | 1 | 2 | 3 |
| Activation function | softmax | softmax | sigmoid |
| Optimizer | RMSprop | Adam | Adamax |
| Learning rate | 0.002 | 0.001 | 0.003 |
| Loss function | categorical\_crossentropy | categorical\_crossentropy | categorical\_crossentropy |
| Accuracy | 99.36% | 98.10% | 99.68% |
| Avg Precision | 99% | 98% | 57% |
| Avg recall | 99% | 98% | 99% |
| Avg f1-score | 99% | 98% | 72% |

Chart

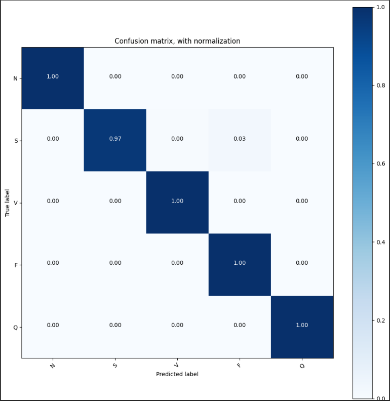
Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generatedChart, line chart

Description automatically generatedChart

Description automatically generatedChart, line chart

Description automatically generated**Model 1 Model 2 Model 3**

Looking at the accuracy metric, Model 3 has the highest accuracy of 99.68%, followed by Model 1 with an accuracy of 99.36%, and Model 2 with an accuracy of 98.10%. This suggests that Model 3 may be the best performing model. However, when looking at the precision, recall, and f1-score metrics, Model 3 appears to perform worse than the other two models. Specifically, Model 3 has a lower average precision and f1-score compared to the other two models. So we can know that Model 3 is not performing as well in correctly identifying the positive class, and may have more false positives or false negatives. It is also important to note that the number of convolutional layers, activation function, optimizer, and learning rate may all impact the performance of the models so the best model is model one even it has only 1 convolution layer but it has RMSprop Optimizer so we can clearly see RMSprop is the best optimizer and softmax is the best activation function for ECG.

1. **Last Comparison of the best models**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Best Autoencoder**  **Sigmoid AF+SGD optimizer** | **Best LSTM Bi-directional LSTM, 2 layers + Adam optimizer** | **Best CNN Sigmoid AF+AdaMax optimizer** |
| **Accuracy(%)** | 89.0% | 98.1% | 96.8% |

-The highest accuracy on the dataset is done by the CNN-based model.

-All the models used the Adam optimizer here, but the CNN-based model used an extension of the Adam optimizer, AdaMax.

-The AdaMax optimizer is much more robust to large gradients and noise, as it does not rely on gradient’s squares, provides a better convergence.