Heart Attack Prediction using Neural Networks and Transfer Learning

Abstract

The physicians use the patterns of the Electrocardiogram(ECG) signals to diagnose cardiovascular diseases. There has been efforts to make this process automatic through computational models to detect the presence of the heart diseases. This project makes effort to use the ECG signals to predict the Myocardial Infarction(MI) (heart attack). To ensure that ECG classification models are transferable across tasks, there exist previous work to use transfer learning, which allows to extract the feature from the ECG signals which then can be used for different classifications. We extract the feature of ECG signals from the network trained on MIT-BI-heartbeat classification data and then using these features for the detection of MI(heart attack). The task done in project can be summarized as following 1)To classify Electrocardiogram (ECG) signals for the presence of an arrhythmia(MIT-BIH dataset) 2)To predict Electrocardiogram (ECG) signals prone to Myocardial Infarction(Heart Attack) and 3) Apply the knowledge transfer of ECG signals for Myocardial Infarction. After applying and analyzing various models to dataset we were able to achieve 98% average F1 score for heart attack prediction.

1 Introduction

According to the report on health data, the cardiovascular diseases is cause of one-third of total deaths around the world. Therefore, it is important to detect the cardiovascular disease as soon as possible so that the person receives the treatment required. The application of machine learning in healthcare has been tremendous, because machines are faster and accurate than humans. The most important benefit is that, we have limited number of physicians for large population. We need the machines which not only provide fast and less error prone results, but also make the availability of such detection tools to everyone at any time. Nowadays, smart devices collect the tremendous data of heartbeats for every single second of a day. The prediction model can help to detect the cardiovascular diseases and can be built to notify the nearest medical facility automatically.

We are using the PTB-dataset for prediction of MI(heart attack). We have trained the Decision Trees, Support Vector Machines and Convolutional Neural Network models for this prediction task. We find that the Convolutional Neural Networks returns the best results based on the average F1 score on the validation set. The final average F1 score achieved on the testset was 99.5%.Â We want to check if the models trained on different datasets can be transferred across tasks, to do this we try transfer learning by training the MIT-BIH dataset and use it to extract the feature of heartbeats, we train Convolutional Neural Network on MIT-BIH dataset, which classifies the heartbeat into five different class. The model result is 89.4% average F1 Score for heartbeat classification on testset. We extract the feature of signals from this trained model and use it for further training for the task of MI(heart attack) prediction on the PTB-dataset. We use the extracted features to train different models, Naive Bayes, Decision Trees and Support Vector Machines.

The following few sections are designed as, 2) Related work 3) introduction to dataset and methods tried to solve imbalance datasets, 4) Methods and models tried in project with the results achieved on test dataset 5) Findings and comparison of results from different models and Conclusion.

2 Related Work

There has been few efforts before to transfer the knowledge across different tasks using feature extraction of heartbeats [6]. The method proposed in [6] uses complex Convolutional Neural Network architecture consists of 13+ hidden layers to process the MIT-BIH data for heartbeat feature extraction. We train the proposed model in [6], and the details is discussed in section 4.2. The paper[6] also designed a pipeline for pre-processing the ECG signals and extracting the beats from signals from original data. The method is said to be yet effective method for preprocessing ECG signals. Our work uses the same methods for preprocessing the dataset for further training.

3 Dataset and Evaluation

We Below is the details about the Dataset we used for the model training, the methods applied on dataset before training model and Evaluation matrices used to compare different models.

3.1 Dataset for training

We use PhysioNet MIT-BIH Arrhythmia and PTB Diagnostic ECG datasets.

3.1.1 MIT-BIH dataset

The MIT-BIH dataset consists of ECG recordings from 47 different subjects recorded at the sampling rate of 360Hz. Each beat is annotated by at least two cardiologists. We use annotations in this dataset to create five different beat categories in accordance with Association for the Advancement of Medical Instrumentation (AAMI). In this dataset we have 5 classes of beat categories.

3.1.2 PTB Diagnostics dataset

The PTB Diagnostics dataset for prediction of Myocardial Infarction (Heart Attack) consists of ECG records from 290 subjects: 148 diagnosed as MI, 52 healthy control, and the rest are diagnosed with 7 different diseases. In this dataset we have 2 classes.

Both are time series dataset. We considered the 10 second ECG window. Some of the samples are padded with zeros to make its length equal to a predefined fixed length.

3.2 Handling imbalanced dataset

Both of the dataset is very imbalanced distributions of examples for different classes, therefore we tried following methods to address the issue.

3.2.1 SMOTE - Synthetic Minority Oversampling Technique

The SMOTE[5] method first selects a minority class examples “A” at random and finds its k nearest minority class neighbors. The synthetic examples is then created by choosing one of the k nearest neighbors “B” at random and connecting "A” and “B” to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances A and B.

3.2.2 Weighted loss on validation data

Another approach we tried to handle imbalanced datasets is to penalize the minority class with greater value compared to majority class, which is weighted loss on validation data.

3.2.3 Data Augmentation

We were not familiar with the theory of many augmentation techniques used on time series data, we stuck with the basic ones because we also can’t be sure about the correct label for signal after augmentation. We chose two forms of augmentation, Reverse the signal and adding Noise to the signal for varying scales.

3.3 Hyper parameters and Evaluation Criteria

We have used Stratified 5-fold cross validation to select the hyper parameters like the depth of the decision tree, C parameter, kernel in Support Vector Machine.

For deep learning models on the MIT-BIH data we use 80:20 split for the train and cross validation set. For the PTB the models use 64% for training 16% for validation and 20% for testing.

4 Methods

Below is the list of approaches we tried for prediction of heart attack.

4.1 Supervised Learning using PTB Dataset

The models described in this subsection are trained on the PTB dataset. The objective is to predict whether a person is having an MI(heart attack or not). The training set was broken into train and test using the ratio 80:20. There was no test set for this dataset. All the models were trained on sklearn [3].

The dataset is imbalanced. We cannot rely on accuracy alone. We used the average F1 score of 5 fold cross validation error split we mention in section 3.3 to compare different models we trained. The same technique is used for tuning the hyper parameters for a given model. We started with a basic model, gaussian naive bayes to classify whether a patient is having a heart attack. The dataset is continuous so we assume the data follows the gaussian distribution. The Table 4.1 shows the results achieved for this model. The results make sense because Naive bayes makes lots of assumptions like feature independence and the underlying distribution.

Next we chose decision trees for this classification task. We tried to tune the maximum height of the decision trees. We again used the cross validation methods mentioned in section 3.2 to select a best model. We found that a maximum depth of 18 gives the highest average F1-score. With 18 as the maximum height of the tree and the minimum number of examples at any given node as 2, we get the following results as shown in Table 4.2.

	Precision	**Recall**	**F1 Score**
0	0.41	0.82	0.55
1	0.89	0.54	0.67

Table 4.1: Naive Bayes result

	Precision	**Recall**	**F1 Score**
0	0.87	0.85	0.86
1	0.94	0.95	0.95

Table 4.2: Decision Trees result

	Precision	**Recall**	**F1 Score**
0	0.95	0.94	0.94
1	0.98	0.98	0.98

Table 4.3: Support Vector Machine result

	Precision	**Recall**	**F1 Score**
0	0.99	0.99	0.99
1	1.00	0.99	1.00

Table 4.4: Convolutional Neural Network

Finally, we chose SVM for the classification task. We tried to tune the C value and kernel of the SVM. The following kernels were used. RBF, Polynomial(Only a degree of 2), Sigmoid kernel, No kernel(Soft Margin SVM with no kernel trick). We tried the following C values 1,16,64,100. We found that an RBF kernel with a C value of 100 resulted in the lowest validation error. As 100 was on the boundary of our range, to ensure we did not miss out on “good" hyper parameters. We tried 105 and 110. The best model still chose RBF kernel and a C value of 100. Training with the whole dataset and a RBF kernel. The results on the test set are shown in Table 4.3

We chose the following Architecture in figure 4.1 based on experimentation by running the experiments on the validation set which was created by further fragmenting the training data to in the ratio 80:20. We trained for a total of 20 epochs using a batch size of 32. The optimizer used for this network is the Adam optimizer with a learning rate of 0.001. The

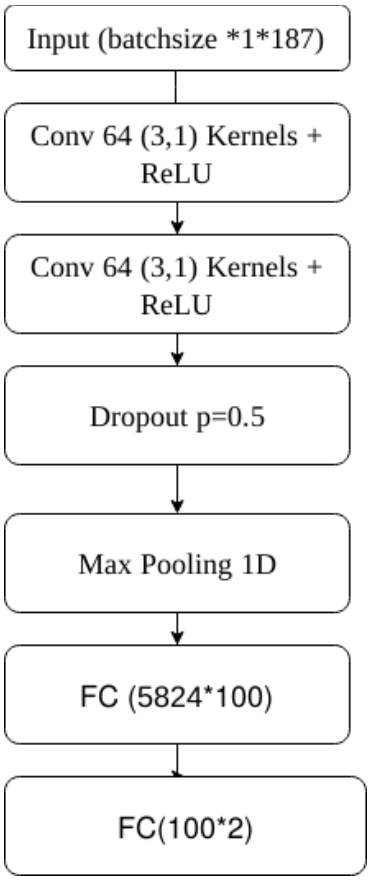

Figure 4.1: CNN for heart attack prediction

results are shown in the Table 4.4. Unsurprisingly, CNN performs better than all the models trained till now and this metric was based on the average F1 score reported by every model trained so far. We believe that the case because the different filters learn particular parts of a signal which helps identify the irregularities in a heartbeat, whereas SVM tried to find a maximal hyperplane separating the positive and negative examples. This CNN model was trained using Keras [2]

4.2 Transfer Learning

Every deep neural network in this section was trained using the MIT-BIH dataset. There was a separate test set. We split the training dataset into train and validation set using the split ratio of 80:20. This 20% split helped us to choose the hyperparameters like learning rate and model selection for the task of transfer learning.

	Precision	**Recall**	**F1 Score**
0	0.80	0.83	0.81
1	0.06	0.03	0.04
2	0.07	0.07	0.07
3	0.02	0.01	0.01
4	0.07	0.08	0.08

Table 4.5: Result - CNN-Paper model on MIT-BIH

4.2.1 Training on MIT-BIH Dataset

This was the hardest part of the project. The dataset was heavily imbalanced. To contrast the scale of imbalance, the minority class had 641 classes and the majority class had 72471. Now we started with the following architecture mentioned in the paper[6]. When we trained the model performed poorly on examples belonging to the other classes. The testset results can be found in the Table 4.5. We did not train further than this because of the poor generalization on the validation set and as well as the testset provided. We used PyTorch for training the network[1]. This model is same as the one mentioned in the paper [6]. It is very deep and very complex. We tried different learning rates, we could never achieve the scores reported in the paper.

Now we explored weighted loss function penalization as mentioned in the section 3.2. As the architecture mentioned in the paper did not work. We chose the one is figure 4.1. To keep the figure short and readable we omitted the average pooling layer i.e After every Conv -> ReLU -> BatchNorm block, there is an average pooling layer with a kernel size of 2. The architecture remained the same. The weights are initialized 5 times using different seeds and with each initialization the model is trained for 5 epochs. The model with the highest average F1 score is selected and training is resumed using the weights after the 5 epochs. We used early stopping with a patience interval of 5 epochs. The criterion of early stopping was based on the improvement of average F1 score on all the classes. We used a batch size of 512 and the Adam optimizer to perform weight updates. The loss function used was the CrossEntropyLoss. This loss function in PyTorch applies the softmax to the raw outputs and then calculates the loss, gradients are calculated as if there is a softmax layer at the end. For

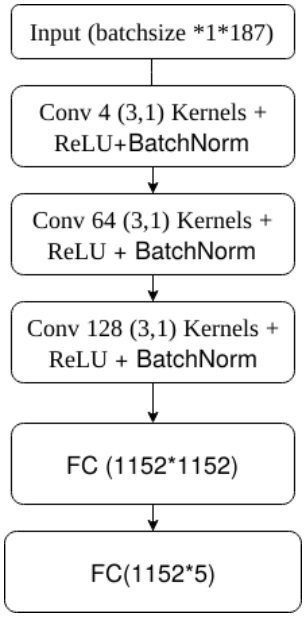

Fig 4.2 CNN for heartbeat classification

	Precision	Recall	F1 Score
0	0.99	0.95	0.97
1	0.44	0.85	0.58
2	0.93	0.85	0.89
3	0.45	0.91	0.60
4	0.95	0.99	0.97

Table 4.6 CNN without augmentation

	Precision	Recall	F1 Score
0	0.99	0.95	0.97
1	0.58	0.84	0.69
2	0.84	0.97	0.90
3	0.50	0.78	0.61
4	0.92	0.99	0.95

Table 4.7 CNN with augmentation

this reason we did not have a softmax layer at the end of our network. The results from the testset are shown in the Table 4.6.

Now we wanted to see the effect of augmentation on this network. To train deep neural network, We chose the techniques mentioned in section 3.2. Everything mentioned above stays the same except the dataset. The results are shown in the following Table 4.7.

4.2.2 Transferring knowledge to the task of PTB dataset

Now we see whether we can transfer the knowledge from the MIT-BIH dataset to PTB dataset. We choose the neural network with the highest average F1 score (the one trained using augmented dataset) as our feature extracting network. We feed the PTB data through the network trained on MIT-BIH data. The dimensions are the same, so no preprocessing was necessary. We used the outputs from the final convolutional layer(the layer just before the dense layer) as features for training ML models. Instead of human hand picking features, we can use the MIT-BIH network as a feature extractor. We trained three ML models as mentioned in section 4.1, but using the features extracted from the network trained in section 4.2.1. The ML models use the same methodologies specified in section 4.1. Only the data changes. SVM outperforms as always. The best model has the validation accuracy of 0.99 and the results of the model can be found in the Table 4.8. The recall is 1, meaning the model never misses a case where a person is having a heart attack. In medical fields recall is more important than precision. A false positive only wastes the time of a person, while missing a person having a heart attack can be devastating to all parties involved. From the below Table 4.8 it is evident that we have trained a network which can act a feature extractor across different tasks in the same domain.

4.2.3 Transfer Knowledge Using a Binarized Dataset

We wanted to see if binarizing the MIT-BIH dataset i.e converting it into a dataset with two classes having an Arrhythmia and not having an Arrhythmia can act as an efficient feature extractor. We used the CNN architecture and training was the same as the process mentioned in section 4.1. We just trained the SVM using the same criteria mentioned in section 4.2.2. The results are shown in Table 4.9. The validation error of this was around 0.98 with C value of 16 and an RBF kernel. We also tried to apply binary MIT-BIH dataset on LSTMs. The LSTMS are advanced RNNs that learn order dependence with sequence data, as the ECG signals are 10 second windows of sequence data, we have made use of LSTMs for

	Precision	Recall	F1 Score
0	0.99	0.99	0.99
1	0.99	1.00	1.00

Table 4.8 CNN on 5 class-MIT-BIH

	Precision	Recall	F1 Score
0	0.98	0.97	0.98
1	0.99	0.99	0.99

Table 4.9 CNN + SVM PTB prediction

	Precision	Recall	F1 Score
0	0.98	0.99	0.98
1	0.94	0.88	0.91

Table 4.10 LSTM on 5 class-MIT-BIH

	Precision	Recall	F1 Score
0	0.31	0.99	0.47
1	0.97	0.16	0.27

Table 4.11 LSTM + SVM PTB prediction

classification of the 5 class data set converted to binary Arrhythmia classification(MIT-BIH). Since the order of the signal matters, we did not choose Bi-Directional LSTMs. A basic LSTM layer with 200 neurons followed by a 20% dropout layer and a fully connected output layer was sufficient to achieve good performance on the MIT-BIH dataset's test data. The LSTM model performed comparatively good on MIT-BIH classification(in Table 4.10), However, testing the features extracted from the MIT-BIH dataset on the PTB dataset using transfer learning achieved average of 37% F1 score, in Table 4.11.

5 Conclusion

We compare the models with and without transfer learning on the PTB dataset. The best model trained using transfer learning(among decision trees, SVM, GNB) has the average F1 score of 0.99 on the validation set i.e SVM with a rbf kernel. Whereas the best model trained on the raw features of the PTB dataset has an average F1 score of 0.98. We can see that the network trained on the MIT-BIH dataset can act as an efficient feature extractor. Even the SVM model trained from the features of the binarized dataset gives a validation accuracy of 0.98 comparable to the model on raw trained data. We chose SVM from transferred features as the best model. The results can be found in figure 4.8.

To summarize, we learned a lot of things. Especially choosing the hyperparameters of the models. Ensuring reproducible models(This was especially tricky with PyTorch). Moreover, we learnt about thinking about back-propagation as a computational graph of chain rule. Learned using deep learning libraries like PyTorch and Keras. Handling Imbalance in training data. Learned about weighted losses and data augmentation. Finally throughout the training process we tried to follow good guidelines(Without peeking).

References

[1] PyTorch: An Imperative Style, High-Performance Deep Learning Library by Paszke, Adam and Gross, Sam and Massa, Francisco and Lerer, Adam and Bradbury, James and Chanan, Gregory and Killeen, Trevor and Lin, Zeming and Gimelshein, Natalia and Antiga, Luca and Desmaison, Alban and Kopf, Andreas and Yang, Edward and DeVito, Zachary and Raison, Martin and Tejani, Alykhan and Chilamkurthy, Sasank and Steiner, Benoit and Fang, Lu and Bai, Junjie and Chintala, Soumith

[2] Keras by Chollet, Francois and others

[3] Scikit-learn: Machine Learning in Python by Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E.

[4] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. *Array programming with NumPy*. Nature 585, 357-362 (2020). DOI: 0.1038/s41586-020-2649-2. (Publisher link)

[5] SMOTE: Synthetic Minority over-Sampling Technique by Chawla, Nitesh V. and Bowyer, Kevin W. and Hall, Lawrence O. and Kegelmeyer, W. Philip

[6] ECG Heartbeat Classification: A Deep Transferable Representation by Kachuee, Mohammad and Fazeli, Shayan and Sarrafzadeh, Majid