Atrial Fibrillation Classification from a Short Single Lead ECG Recording

Arsi Ikäheimonen and Maryam Kiashemshaki

December 2020

1 Introduction

Atrial fibrillation (AF) is an irregular heart rhythm that can lead to stroke, heart failure and increases the risk of dementia. AF can be symptomatic or asymptomatic. Asymptomatic, silent AF, occurs frequently[1, 2]. AF may be episodic that makes AF detection problematic. Therefore, continuous monitoring of heartbeat with a portable device might be beneficial for the AF detection[3]. However, discriminating AF from a single-lead electrocardiogram is a challenging task due to the noise. The convolutional neural networks (CNN) approach is a useful tool for the classification of ECG recordings [1, 7]. However, using CNN needs relevant hand-craft architects. In 2017 PhysioNet/CinC provided a Challenge to classify ECG signals into four categories: atrial fibrillation, normal sinus rhythm, other rhythms, or too noisy for classification. More than 60 groups of researchers participated in the challenge. The measured F1-score of classification performance ranged from 0.25 to 0.83 [1].

In this project, we used ECG recordings data provided by PhysioNet/CinC Challenge 2017 to classify ECGs into those four categories. The data were converted from a 1D time series to 2D scalogram images (frequency domain representation), using the wavelet transform (WV). We trained a CNN on the pre-processed data to classify the signals. The classifier reached an accuracy of 0.65 (unweighted F1-score) when evaluated using the test data.

This report is organized as follows: in section 2, we introduce the data set. In section 3, various neural networks are investigated. Section 4 gives the results. Section 5 provides discussion, and in section 6, we conclude what we have learned in this project. In addition, in the appendix, we put some filter and feature maps and details of the model.

All the code used in the project is stored on git hub repository `https://github.com/ArgonSilicon/AI_in_health_technologies_project_work`.

2 The data set and pre-processing

The data set contains 8528 single-lead ECG recordings with a length between $9s$ and just over $60s$. The dominant class in the data set is the normal rhythm class (60%). The second largest group belongs to other rhythms, which contains 28% of the data. Atrial fibrillation class has 9% of the data, and noisy class with 3% has the lowest share.

From the ECG time series, we could deduce that the normal ECG rhythm has relatively evenly spaced R-peaks, while the AF rhythm is irregular. The other rhythm is not easy to distinguish from the normal, while noisy recordings are easy to detect.

Since the raw data was already in Matlab format, we decided to do the pre-processing using Matlab. We started the pre-processing by dividing it into three categories: the test set, the training set, and the validation set. The test set contains 20% of the total data. The remaining 80% of the data is divided by 80% / 20% for the validation set and the rest for the training set. Since the data is unbalanced, we used data augmentation to make the data set even. For this purpose, we took subsamples of $9s$ from the original train/validation data to make the size of each class 4040 samples. The final train/validation data set contains 16100 ECG recordings, while test data contains 1706 samples (unbalanced classes).

In the next step, we replaced the outlying data points (the points having absolute values greater than three standard deviations) from the time series with interpolated values. We also normalized the time-series to have values inside the interval of $[-1, 1]$. After that, we upsampled the time series using a sampling frequency of $600Hz$, expanding the upper limit of the wavelet transform frequency range to $300Hz$. Then, we transformed the time-series using the continuous wavelet transform (CWT) with Morse wavelets, symmetry parameter (gamma) of 3, and time-bandwidth product of 60. Figure 1 shows the example of the unprocessed ECG time-series and the corresponding scalograms.

From the scalogram, we can identify individual R-peaks as high frequency spikes. At lower frequencies (around $1Hz$), we can detect continuous bands that were produced by the full ECG complex (this can be thought of as a frequency component enveloping the whole PQRTS-complex). We decided to use these frequency domain representations as the training data for our convolution neural network classifier. The pre-processed data sets were stored in Google Drive folders to be accessed when training and testing the classifiers on Google Colab.

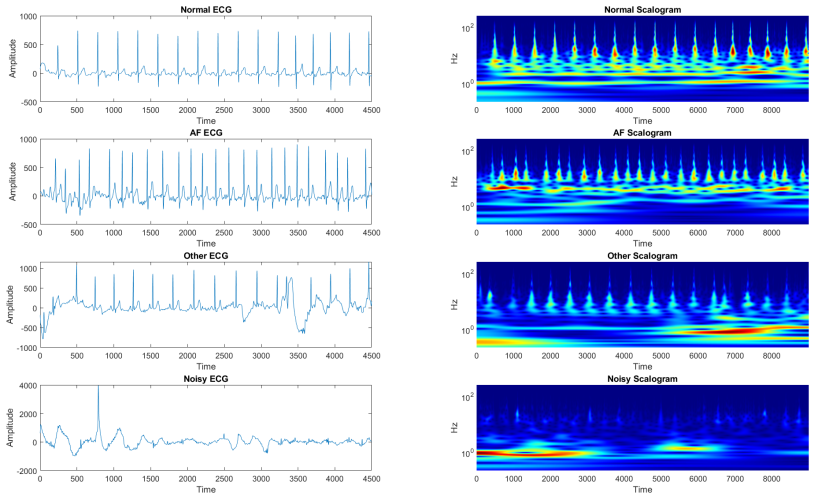

Figure 1: Examples of each class of the ECG time series and corresponding scalograms.

3 The model and the training

After creating the training, validation, and test set, we defined the model. First, we used the transfer learning method with several pre-trained networks (ResNet50, Xception, and EfficientNetB0), with pre-trained weights. We also tried to train the networks from scratch, using the training data. It turned out that these methods yielded rather poor results. We suppose that the networks' kernel sizes were too small to detect the consecutive R-peaks in scalograms. Therefore, we designed a simple CNN with a larger convolution kernel size and trained it on the data. We deduced that the kernel size 9×9 had a receptive field large enough to cover the consequent R-peaks from the scalograms. The simple CNN model consists of 3 convolution layers with batch normalization and max-pooling layers after each of them. In addition to convolution layers, global average pooling and three dense layers, and a prediction layer was added to the network. Figure 12 shows the model summary in detail.

We trained the network using an early stopping hook (no epoch limit), the learning rate of 0.0001, Adam-optimizer, and sparse categorical cross entropy-loss. We find out that the model learns the training set well, while the training accuracy/loss was not so good (figure 2), indicating the model overfitting. We used dropout layers (for the convolution and the dense layers), an early stopping hook, and L_1 regularization to overcome the overfitting. In addition, the data augmentation (random shifting horizontally and vertically) applied on the data between the epochs. After testing the model performance with the test set, we

chose the most simple model, as it performs as well as the more complex ones. Figure 3 shows the selected model training/validation accuracy for the selected model. There is less overfitting than before, and the training and validation accuracy are better aligned.

Figure 2: Training and validation accuracy and loss, no regularization

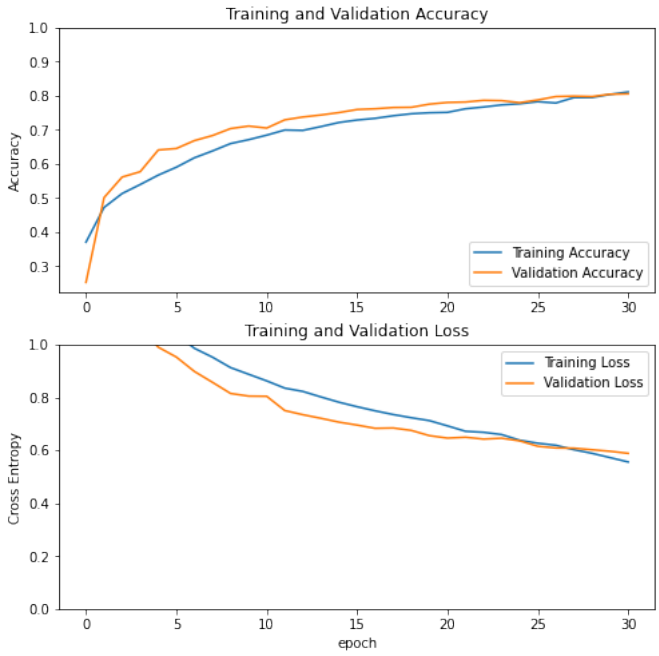

Figure 3: Training and validation accuracy and loss with regularization

4 Results

For the model performance evaluation, we used the model to classify the images on the test set. Figure 4 displays the confusion matrix, showing how well the model did on each class on the test set. Figure 5 shows the multi-label ROC curves for each class. Table 1 demonstrates the classification performance, which is measured using the precision, recall, and F_1 scores of each class (Normal, AF, Others, and Noisy).

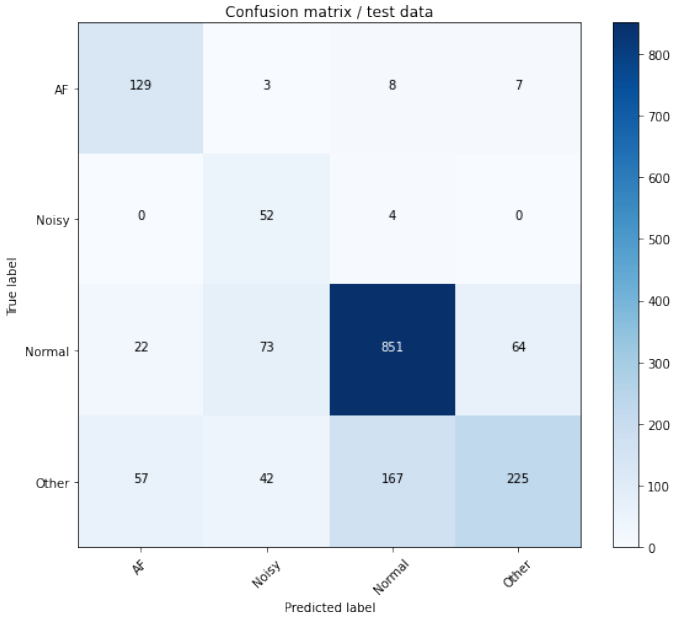

Figure 4: Confusion matrix test data

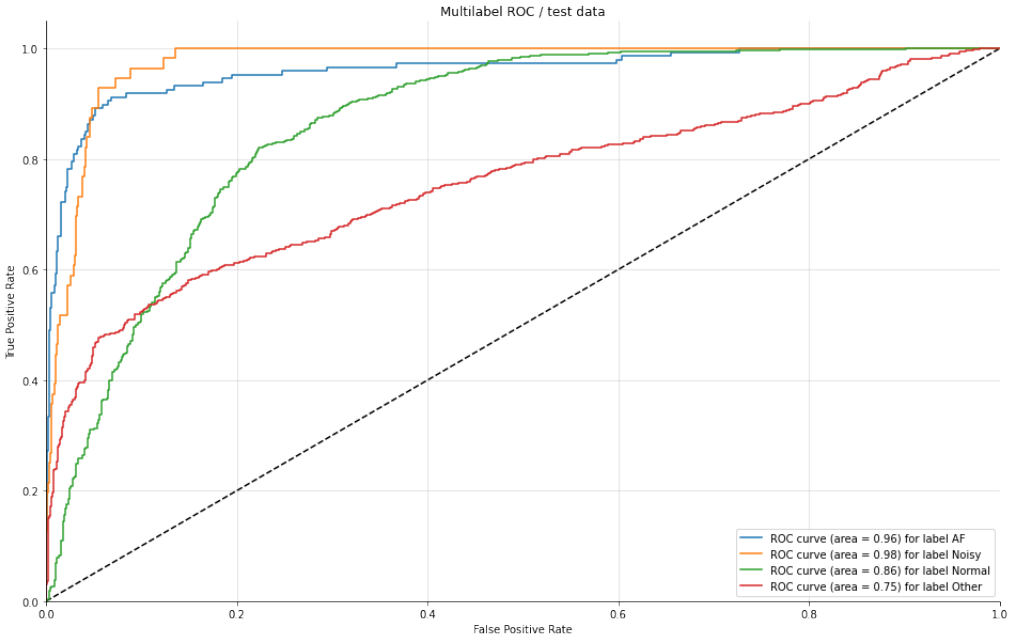

Figure 5: Multilabel ROC curves

	precision	**recall**	f_1 **score**	**support**
AF	0.62	0.88	0.73	147
Noisy	0.31	0.93	0.46	56
Normal	0.83	0.84	0.83	1010
Other	0.76	0.46	0.57	491
accuracy			0.74	1704
macro avg	0.63	0.78	0.65	1704
weighted avg	0.77	0.74	0.74	1704

Table 1: Classification performance

From the evaluation metrics, we notice that the model has a rather poor precision for Noisy label detection and poor recall for Other label detection. The classifier miss classifies some data with Normal and Other labels as Noisy. The data belonging to Other class is also miss classified as Normal, AF, or Noisy. These deficiencies brought the unweighted F_1 score down when compared to validation results. The Normal class has the highest precision (also highest support), while AF, Noisy, and Other has relatively good recall values.

5 Discussion

The presented results in previous section illustrate that the simple model is more effective than the more complex ones. We speculate, that this is because the more complicated models are prone to overfitting due to too small training data set. However, we were impressed with how well the model performs without extensive feature engineering efforts. Some of the high scoring competition submissions have used some arduous feature engineering, and traditional machine learning methods (for example, Random Forest, and AdaBoost) for classifying the data [6] [5]. The good results were also obtained with the neural network classifiers combining CNN layers for the representation learning and LSTM layers for the sequence learning [4] [7]. However, since we did not have access to the test data, we could not compare our results directly with the challenge submission.

To achieve better classification results, we could try several methods. These methods include data pre-processing and feature engineering, improving the neural network model, and acquiring more training more data. In the pre-processing phase, the unwanted frequencies could be removed from the signal. For the classifying purposes, the most important ECG frequencies are around $20Hz$, while movement related artifacts occur at low frequencies (sub $0.5Hz$), and noise occurs at higher frequencies ($50Hz$ and higher) [**?**]. Therefore, a passband filtering around the frequency of $20Hz$ could be tried to remove the

unimportant information from the data. However, the frequencies related to movement and noise are also indicative for Noisy class detection. Some of the competition submissions acquired high scores by using LSTM networks with CNN's. We could have also extended our model by adding those at the top of our CNN layers. The LSTM networks could be added on top of 1D convolution or 2D convolution network. Since the data set was fixed, one way of getting more training data could be randomly subsampling the original time-series. Since the series are of varying length, we decided to use 9-second subsamples to keep the training data consistent. Other option would have been to first compute the wavelet transforms, and then cut the time-series randomly in 9-second sequences. That way, we could have induced more variation in to the training data. We could also have padded the time series with zeros to create sequences of consistent lengths.

6 Conclusion

In this project, we learned how to construct and implement deep learning algorithms for classifying ECG recordings using convolution neural network architecture. We learned to pre-process the data sets for classification using wavelet transform and frequency domain representations, turning the 1D-time-series classification into a 2D image classification task. We adapted the CNN model to learn the important features from the data automatically, and validated the results using different techniques. Furthermore, We were impressed how good results the classifier obtained using the unbalanced, small, and noisy data set. We believe, that it would be an easy task to further improve the results by modifying the CNN model.

Appendix

Visualising the learned weights can help us to find how well our network has learned the features. Inspecting the figures, we notice that the filters have learned to detect vertical and horizontal lines. Figures 6 , 7 and 8 show filter maps for each convolution layer.

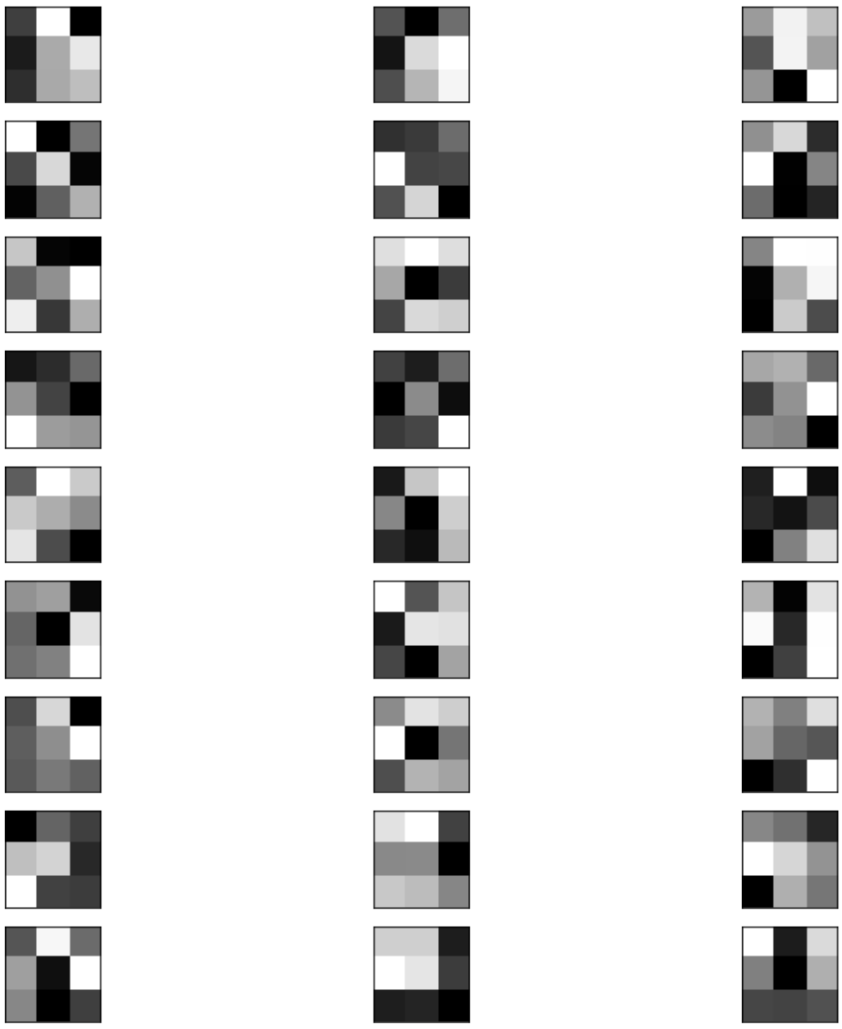

Figure 6: 1.st convolution layer filter map

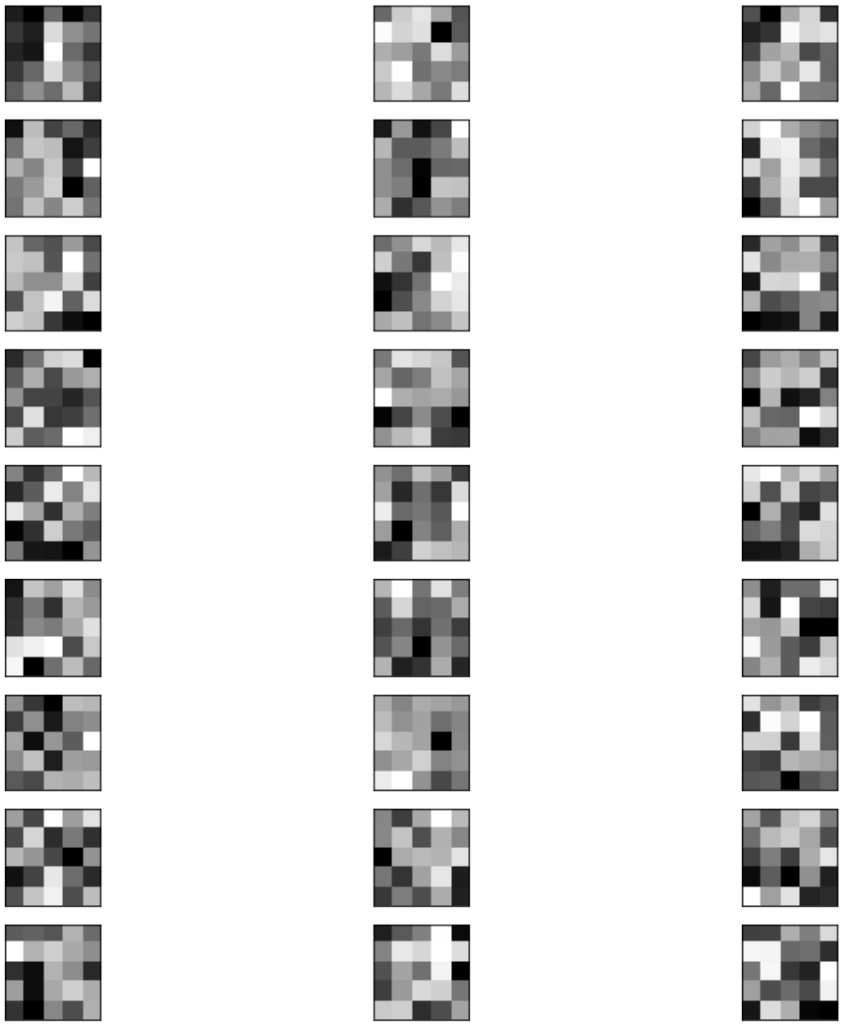

Figure 7: 2.st convolution layer filter map

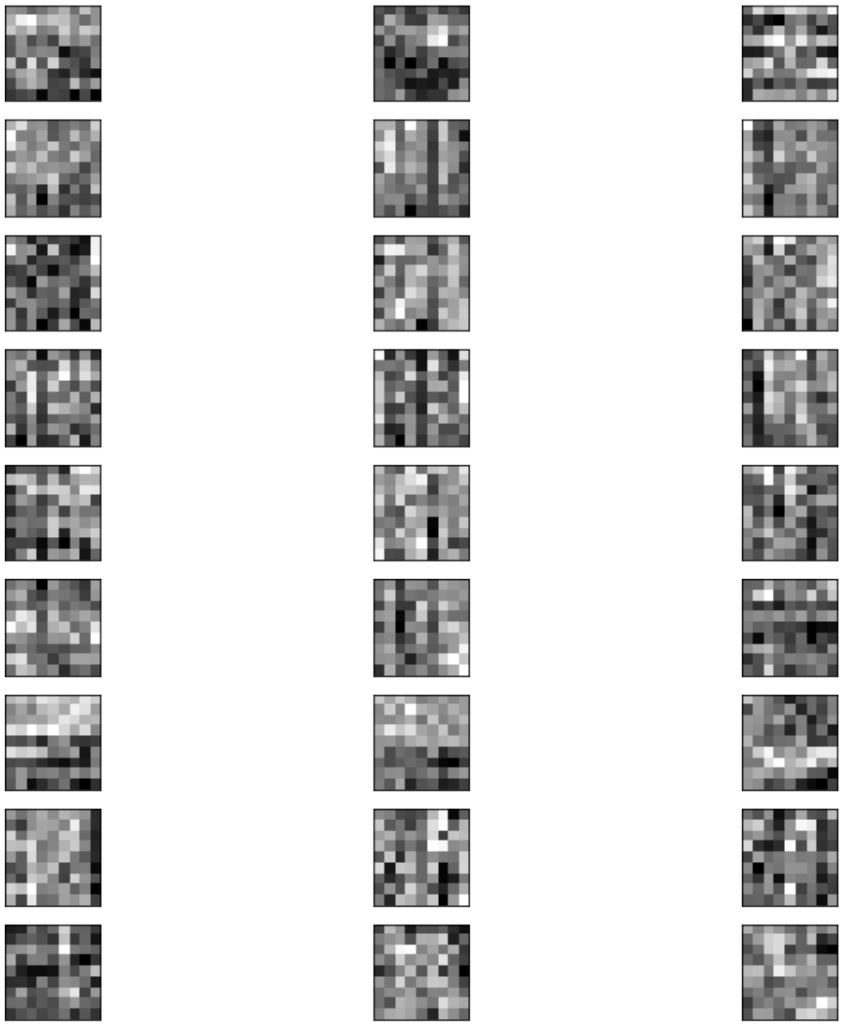

Figure 8: 3.st convolution layer filter map

By visualising feature maps for a specific input image, we can gain some understanding of what features our CNN detects. It seems, that the network can detect consequent R-R peaks, and low frequencies as well from ECG time series frequency presentations. Figures 9 , 10 and 11 show feature maps for the first, second and third convolution layer, respectively.

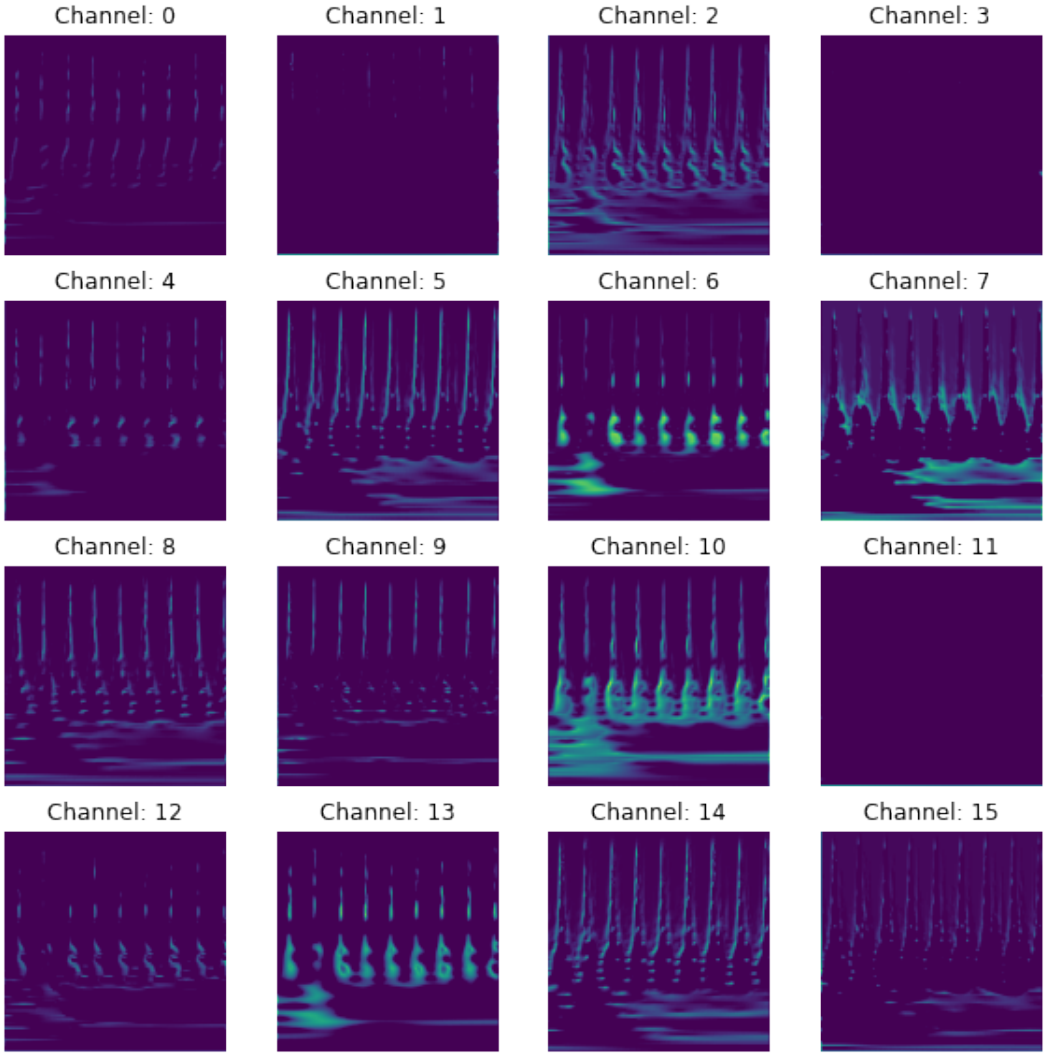

Figure 9: 1:st convolution layer feature map example

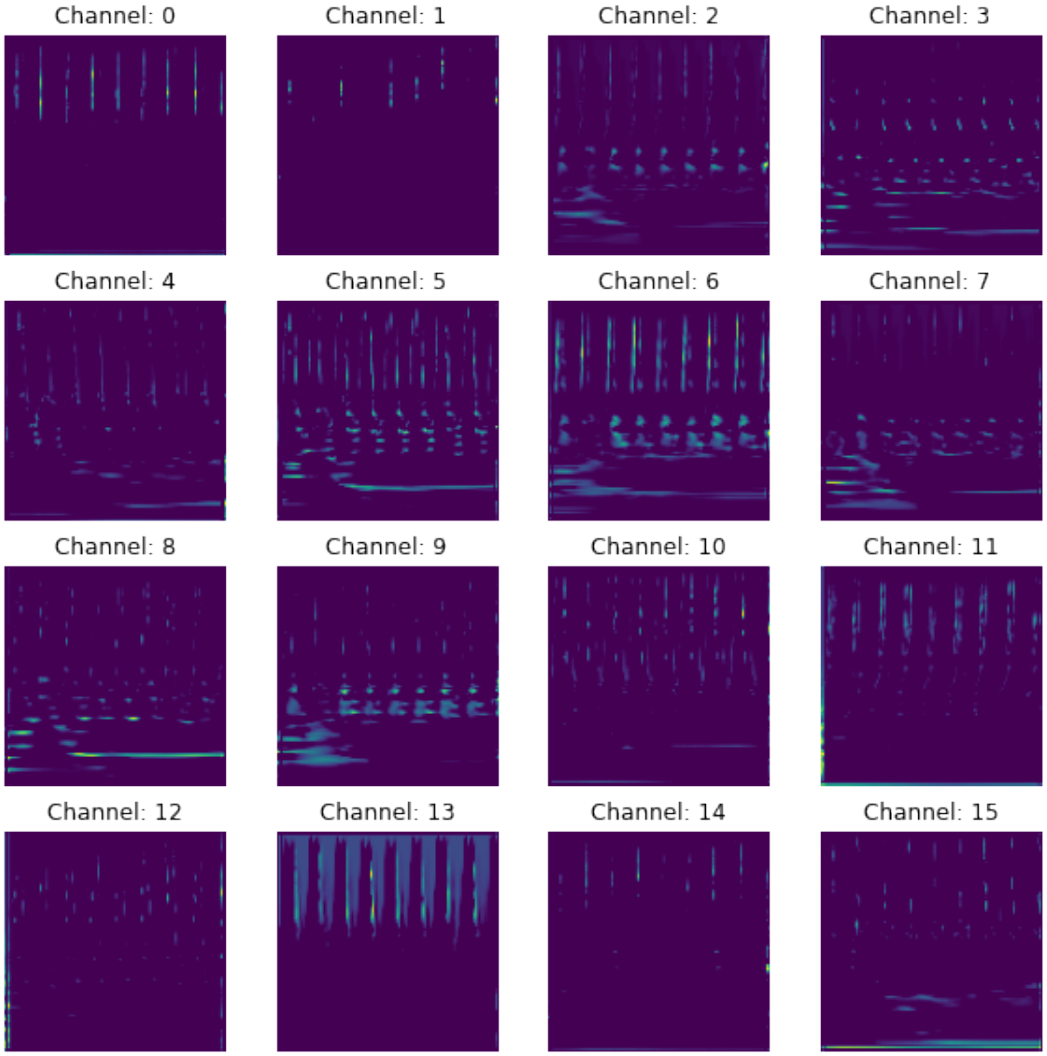

Figure 10: 2:nd convolution layer feature map example

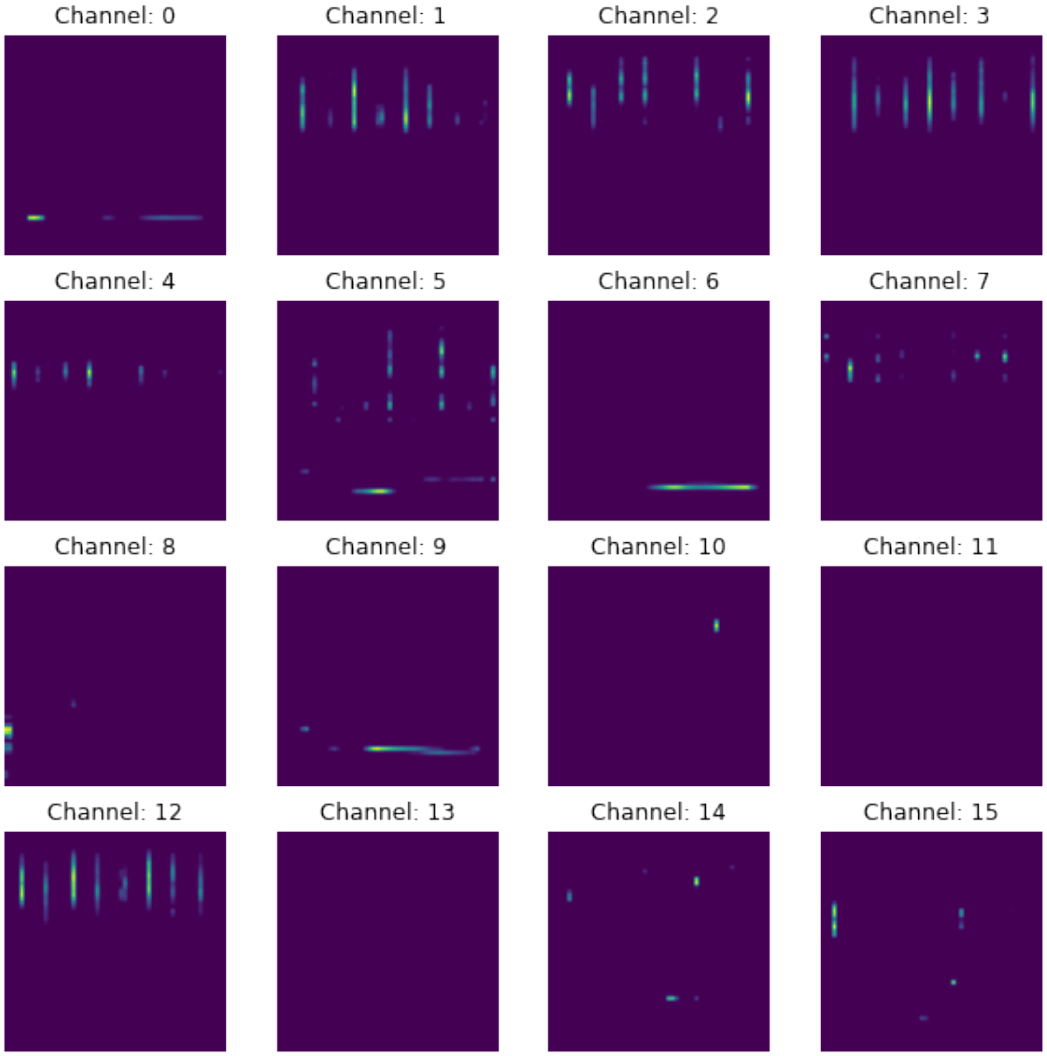

Figure 11: 3:rd convolution layer feature map example

Figure 12 illustrates the model summery for more detail.

```
(224, 224, 3)
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
rescaling_2 (Rescaling)      (None, 224, 224, 3)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 224, 224, 32)      896
_________________________________________________________________
spatial_dropout2d_6 (Spatial (None, 224, 224, 32)      0
_________________________________________________________________
batch_normalization_6 (Batch (None, 224, 224, 32)      128
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 112, 112, 64)      51264
_________________________________________________________________
spatial_dropout2d_7 (Spatial (None, 112, 112, 64)      0
_________________________________________________________________
batch_normalization_7 (Batch (None, 112, 112, 64)      256
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 56, 56, 64)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 56, 56, 128)       663680
_________________________________________________________________
spatial_dropout2d_8 (Spatial (None, 56, 56, 128)       0
_________________________________________________________________
batch_normalization_8 (Batch (None, 56, 56, 128)       512
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 28, 28, 128)       0
_________________________________________________________________
global_average_pooling2d_2 ( (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 512)               66048
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_9 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_10 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_8 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_11 (Dense)             (None, 4)                 516
=================================================================
Total params: 947,524
Trainable params: 947,076
Non-trainable params: 448
```

Figure 12: The model summary

References

[1] Af classification from a short single lead ecg recording - the physionet computing in cardiology challenge 2017.

[2] Asymptomatic atrial fibrillation.

[3] Alivecor kardiamobile, personal ekg monitor, 2020. Accessed: 2020-12-29.

[4] Masun Nabhan Homsi Philip Warrick. Cardiacarrhythmia detection from ecg combining convolutional and long short-term memory networks. In *2017 Computing in Cardiology (CinC)*, pages 1–4. IEEE, 2017.

[5] Ayan Mukherjee Rohan Banerjee Anirban Dutta ChoudhuryRituraj Singh Arijit Ukil Soma Bandyopadhyay Arpan Pal Dr Sundeep Khandelwal Shreyasi Datta, Chetanya Puri. Identifying normal, af and other abnormal ecg rhythms using a cascadedbinary classifier. In *2017 Computing in Cardiology (CinC)*, pages 1–4. IEEE, 2017.

[6] Daniel Castro Paulo Félix Tomás Teijeiro, Constantino A. García. Arrhythmia classification from the abductive interpretation of shortsingle-lead ecg records. In *2017 Computing in Cardiology (CinC)*, pages 1–4. IEEE, 2017.

[7] Martin Zihlmann, Dmytro Perekrestenko, and Michael Tschannen. Convolutional recurrent neural networks for electrocardiogram classification. In *2017 Computing in Cardiology (CinC)*, pages 1–4. IEEE, 2017.