ECG IMAGES CLASSIFICATION

Final Report

Mumtaz Murat ARIK

1. Problem statement

We want to predict behaviour of the EKG signals. Categories are normal beat, supraventricular ectopic beat, ventricular ectopic beat, and fusion beat.

Our client is healthcare providers and hospitals. Some patients need to be kept under observation and those patients are connected to EKG machines. It is impossible to put a person 7/24 to watch them. It will be better if we can obtain and analyze EKG data simultaneously to check if everything is in normal condition.

By analyzing EKG data we can inform healthcare providers immediately so they can take immediate action. Seconds might become important to save one's life.

2. Dataset

I will be using data from a Kaggle project. The source link:

https://www.kaggle.com/analiviafr/ecg-images

The original data is coming from Physionet's MIT-BIH Arrhythmia Dataset. There are 109445 samples in the dataset. Image Resolution is 196x128. They classified the data in five categories. The number of pictures in each category are as follow:

N (Normal beat) : 90.589

S (Supraventricular ectopic beat) : 2.779

V (Ventricular ectopic beat) : 7.236

F (Fusion beat) : 803

Q (Unknown beat) : 8.038

I will use a convolutional neural network to solve this problem.

My deliverables will be the Jupyter notebook that includes codes and some notes. I will also report my method and findings in a final report.

3. Exploratory Data Analysis

This data set is very clean. I plotted 6 images from each category as an example. Below, example figures are shown in Figure 1, Figure 2, Figure 3, Figure 4, and Figure 5 .

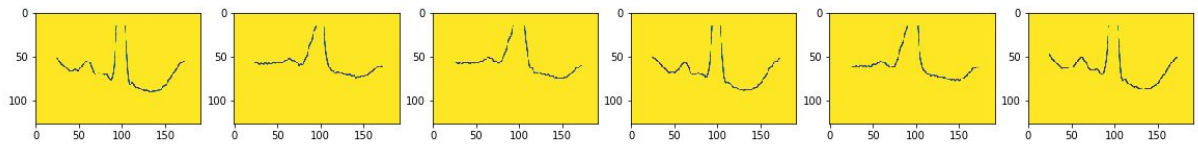

Figure 1. F Fusion beat images

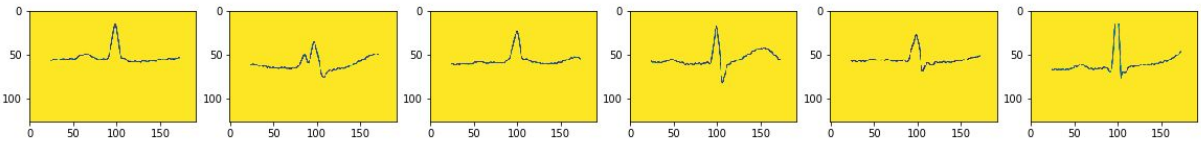

Figure 2. N Normal beat images

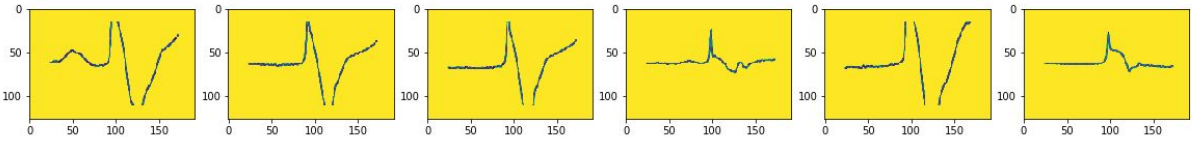

Figure 3. Q Unknown beat images

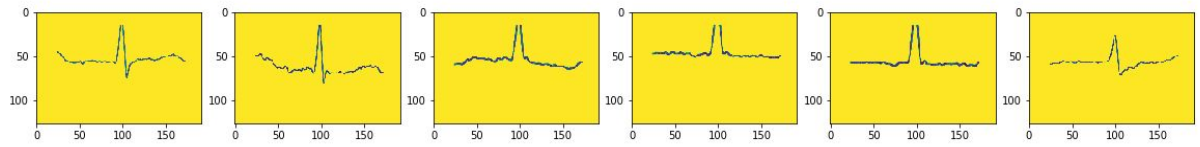

Figure 4. Supraventricular ectopic beat images

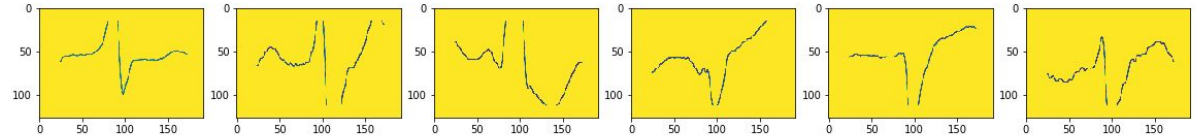

Figure 5. Ventricular ectopic beat images

All images have 3 channel information. I plotted one of the images' Red, Green, and Blue values separately in Figure 6.

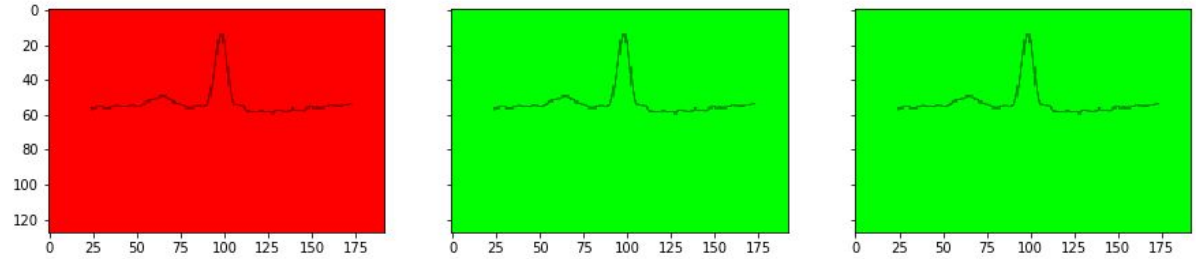

Figure 6. RGB images of one image

In the model training, I will use only one channel.

4. Data Wrangling

This dataset has no separate train and test sets. All images are categorized under its own subfolder with the same name. These folders are in the "../input/ecg-images/MITBIH_img" folder. For model training we need to split the dataset and also if you are working on Kaggle you have to copy your files under "../working" subfolder. I chose a 0.2 split ratio. For this reason I created "../working/test" and "../working/train" subfolders on Kaggle.. These test and train folders have 5 subfolders. From each categorical subfolder("../input/ecg-images/MITBIH_img/ "), the first 20% of the images are copied to its related test subfolders and the rest are copied to related train subfolders.

After splitting the dataset, the image numbers in each file are shown Table 1.

Table 1. Image numbers in each train and test subfolders

Class	Folder Name	Train(../working/train/)	Test(../working/test/)	Total(../input/ecg-images/MITBIH_img/)
Supraventricular Ectopic Beat	S	2223	556	2779
Normal Beat	N	72471	18118	90589
Ventricular ectopic Beat	V	5789	1447	7236
Unknown Beat	Q	6430	1608	8038
Fusion Beat	F	642	161	803

5. Predictive Model

Predictive models are run at Kaggle servers. GPU is used. GPU is faster than CPU.

Convolutional Neural Network(CNN) is used for classification. I used the PyTorch package. PyTorch package is more flexible than other packages.

5.1. Model DataLoader

Original image resolution is 196x128 pixels. Image resolution is reduced to 120x120 pixels. I used one channel and I normalized the images with mean=0.5 and std =0.5. The batch size is 32.

5.2. Model Architect

I tried 4 different CNN models and compared them. Confusion matrix and scores are reported after each model.

Very basic CNN has at least one convolutional layer. With the help of this layer we try to understand features in the image independently from its location in the image. Typical kernel sizes are 3*3 and 5*5. I will use 3*3.

After the convolutional layer people use an activation layer. Idea of the activation layer is to introduce non-linearity to the model. Neurons in real life work in a similar way. They are activated upon a specific threshold value. I use the Relu function which makes all negatives values zero.

In a very basic CNN after activation layer, people use a pooling layer which helps to reduce the size of the input image and computational cost.

We can repeat this combination(convolution , activation, and pooling layer) in a CNN. Of course there are other layers other than these three layers, but we can think that this is the basic.

The last step is the flattening layer. All outputs will be aligned in a single layer. This single layer will be connected to another single layer which has a number of nodes same as class numbers.

One can use an activation layer and single layers in these last steps.

Another layer is a dropout layer, which will make some nodes 0 with a given probability. It is shown that these improve accuracy of the model. I insert a dropout before the flattening layer.

5.2.1. First Model

My first model is a very basic model composed of two convolutional layers. Architect is as follow:

- First Convolutional Layer: 4 convolutional filters with kernel size (F*F) = 3*3,

  stride (S) = 1, padding (P) = 1.

Aim of convolutional layers is to detect features in the given image. In this layer we have 3*3 filter. Input image size is 120*120 and output image size is also 120*120:

$$\text{Output} = \frac{I+F+P+P}{S} + 1 = \frac{120-3+1+1}{1} + 1 = 120$$

Our network will learn 4 filters in this process

- Activation function: Rectified linear unit (ReLU)

After the coming layer after convolutional is the activation layer. f(x) = x^+ = max(0,x). This relatively new activation function has strong biological and mathematical motivations and it is the most popular one nowadays. Its advantages are biological plausibility, sparse activation, better gradient propagation, efficient computation, scale-invariant. (Ref: https://en.wikipedia.org/wiki/Rectifier_(neural_networks))

- Max pooling with kernel size 2

The problem with output of a convolutional layer is it is sensitive to locations of the features in the input image. One solution to this sensitivity is pooling. The next layer after the activation layer is pooling. Two common pooling methods are average and maximum pooling. We used max pooling. The kernel size is 2*2.

After max pooling, data size reduced from 120*120 to 60*60.

- Second Convolutional Layer: 4 convolutional filters with kernel size=3*3, stride=1, padding=1

Number of inputs and outputs are the same. So our network will continue to learn the same filters in this layer.

- Activation function: Rectified linear unit (ReLU)

We used the same activation function.

- Max pooling with kernel size 2

After this Second max pooling layer input size reduced from 60*60 to 30*30.

- Dropout(p=0.5)

Neural networks are inclined to overfit a training dataset. For this reason some nodes are dropped randomly. In this layer we assign to 0 to nodes with a probability of 0.5 to overcome overfitting.

- Linear layer: input 4*30*30 output: 5 (Because we have 5 classes)

In this layer all coming input is converted to one single layer. We have 5 classes so this flattened single layer is connected to the last layer which has 5 nodes.

Criterion is CrossEntropyLoss. This criterion combines nn.LogSoftmax() and nn.NLLLoss() in one single class. This criterion is a good choice for us because we do classification. (Ref:https://pytorch.org/docs/master/generated/torch.nn.CrossEntropyLoss.html?highlight=cross entropyloss#torch.nn.CrossEntropyLoss)

Optimizer is stochastic gradient descent(SGD) with learning rate=0.001 and momentum=0.9. Number of epochs is 5. These models are trained on the Kaggle gpu. During training after 300 steps, Loss and Accuracy are printed on the screen. Confusion matrix, recall, precision, and F1-score are printed on the screen after each epoch. The last confusion matrix after 5 epoch is given in Table 2.

Table 2. Confusion Matrix for First Model Train Data

	ACTUAL				
PREDICTIONS	N	Q	S	V	F
	71650	546	1451	1042	373
	150	5781	3	99	3
	234	2	734	35	1
	412	100	35	4567	75
	25	1	0	46	190

Scores are given in Table 3.

Table 3. Score Table for First Model Train Data

Classes	Recall	Precision	F1-Score
N	0.99	0.95	0.97
Q	0.90	0.96	0.93
S	0.33	0.73	0.45
V	0.79	0.88	0.83
F	0.3	0.73	0.42

If one look at F1-scores, will see that N has the highest 0.97 F1-score. The lowest F1-score is 0.42 for F class which has the least number of images. This data set is unbalanced and most of the data is in N class. I believe that if we have more data for other classes, F1-scores will be increased for all classes.

This trained model is tested on the test dataset. The confusion matrix is given at Table 4.

Table 4: Confusion Matrix for First Model Test Data

	ACTUAL				
PREDICTIONS	N	Q	S	V	F
	17959	127	381	299	96
	26	1467	0	43	1
	49	0	169	5	0
	80	14	6	1093	33
	4	0	0	7	31

Test dataset scores are given in Table 5.

Table 5. Score Table for the First Model Test Data

Classes	Recall	Precision	F1-Score
N	0.99	0.95	0.97
Q	0.91	0.95	0.93
S	0.30	0.76	0.43
V	0.76	0.89	0.82
F	0.19	0.74	0.31

First model F1-scores for test dataset is almost the same F1-scores for train dataset. This is a good sign. This model is the best model among four models.

5.2.2. Second Model

This model is almost the same as the first model. For comparison I only changed filter(kernel) numbers in the second convolutional layer(CL). Number of filters is 8 in the second CL.

- First CL : 4 convolutional filters with kernel size = 3*3, stride = 1, padding = 1.
- Activation function: Rectified linear unit (ReLU)
- Max pooling with kernel size 2
- ----
- Second CL: **8** convolutional filters with kernel size = 3*3, stride = 1, padding = 1
- Activation function: Rectified linear unit (ReLU)
- Max pooling with kernel size 2
- Dropout(p=0.5)
- ----
- Linear layer: input 4*30*30 output: 5

Kernels are used for learning features in the image. I am looking at ECG data and I am thinking that 4 filters might not be enough to train my model. So in this second model I only increased my filter numbers to learn more features. Confusion matrix for the second model for the train dataset is given in Table 6.

Table 6. Confusion Matrix for the Second Model Train Data

	ACTUAL				
PREDICTIONS	N	Q	S	V	F
	20164	1870	635	1620	175
	19649	1661	614	1631	165
	16852	1496	503	1275	143
	15620	1388	464	1248	155
	186	15	7	15	4

Scores for the second model for the train dataset is given in Table 7.

Table 7. Score Table for the Second Model Train Data

Classes	Recall	Precision	F1-Score
N	0.28	0.82	0.42
Q	0.26	0.07	0.11
S	0.23	0.02	0.04
V	0.22	0.07	0.10
F	0.01	0.02	0.01

Scores for test data set is not good as train data set. The maximum F1-score is 0.42.

5.2.3. Third Model

Another person worked on this project using Keras package in R platform. Their project is on Kaggle. I try to mimic their project. Third model is the most complicated model among the four models.

Second convolutional layer has 8 filters, the activation function is changed to the Leaky Rectified linear unit. And there is batch normalization. Batch normalization is used to increase speed, performance, and stability of the artificial neural network (https://en.wikipedia.org/wiki/Batch_normalization#cite_note-:0-1) .

- First CL : 4 convolutional filters with kernel size = 3*3, stride = 1, padding = 1.
- Activation function: Rectified linear unit (ReLU)
- Max pooling with kernel size 2
- ------
- Second L: 8 convolutional filters with kernel size = 3*3, stride = 1, padding = 1

- Activation function: Leaky Rectified linear unit (LeakyReLU)
- Batch Normalization
- Max pooling with kernel size 2
- Dropout(p=0.25)
- ------
- Linear layer: input 8*30*30 output: 100
- Activation function: Rectified linear unit (ReLU)
- Dropout(p=0.5)
- Linear layer: input 100 output: 5

Model is trained with 5 epochs. Confusion Matrix is reported in Table 8.

Table 8. Confusion Matrix for the Third Model Train Data

	ACTUAL				
PREDICTIONS	N	Q	S	V	F
	585	76	22	80	3
	24813	2232	700	2242	253
	2100	166	72	186	19
	26868	2547	850	1985	234
	18105	1409	579	1296	133

Scores are reported in Table 9.

Table 9. Score Table for the Third Model Train Data

Classes	Recall	Precision	F1-Score
N	0.01	0.76	0.02
Q	0.35	0.07	0.12
S	0.03	0.03	0.03
V	0.34	0.06	0.10
F	0.21	0.01	0.01

For the second model the highest f1-score is 0.12 which is worse than second and first model.

5.2.4. Fourth Model

The difference between this model and our simple first model is, in the second convolutional layer kernel size is increased to 5*5. Because I increased kernel size, I increased padding from 1 to 2.

- First CL : 4 convolutional filters with kernel size = 3*3, stride = 1, padding = 1.
- Activation function: Rectified linear unit (ReLU)
- Max pooling with kernel size 2
- ----
- Second L: 4 convolutional filters with kernel size = 5*5, stride = 1, padding = 2
- Activation function: Rectified linear unit (ReLU)
- Max pooling with kernel size 2
- Dropout(p=0.5)
- ----
- Linear layer: input 4*30*30 output: 5

Model is trained with 5 epochs. Confusion Matrix is reported in Table 10.

Table 10. Confusion Matrix for the Fourth Model Train Data

	ACTUAL				
PREDICTION	N	Q	S	V	F
	27781	2426	832	2165	248
	3994	317	133	335	35
	5705	471	164	406	51
	11495	1100	341	1044	117
	23496	2116	753	1839	191

Scores are reported in Table 11.

Table 11. Score Table for the Fourth Model Train Data

Classes	Recall	Precision	F1-Score
N	0.38	0.83	0.52
Q	0.05	0.07	0.06
S	0.07	0.02	0.04
V	0.18	0.07	0.10
F	0.30	0.01	0.01

The highest f1-score is 0.52 for N class. This score is worse than the first model but I still checked model performance on the test dataset.

Confusion Matrix for test data is reported in Table 12.

Table 12. Confusion Matrix for the Fourth Model Test Data

	ACTUAL				
PREDICTIONS	N	Q	S	V	F
	6892	595	224	510	43
	1060	69	30	70	10
	1463	124	52	113	7
	2803	304	71	263	43
	5900	516	179	491	58

Scores for test data are reported in Table 13.

Table 13. Score Table for the Fourth Model Test Data

Classes	Recall	Precision	F1-Score
N	0.38	0.83	0.52
Q	0.69	0.06	0.05
S	0.09	0.03	0.04
V	0.18	0.08	0.11
F	0.36	0.01	0.02

Forth model f1-scores for train and test dataset are similar and they are not better than the first model.

6. Conclusion

I tested four different models. My first is the basic one and gives the best results. Reported scores for classes: N, Q, S, V, F are Recall: 0.99, 0.90, 0.33, 0.79, 0.3; Precision: 0.95, 0.96, 0.73, 0.88,

0.73; F1-Score: 0.97, 0.93, 0.45, 0.83, 0.42. Forth model reported scores for classes: N, Q, S, V, F are Recall: 0.38, 0.04, 0.09, 0.18, 0.36; Precision: 0.83, 0.06, 0.03, 0.08, 0.01; F1-Score: 0.52, 0.05, 0.04, 0.11, 0.02.

Second model and third models' Recall, Precision, and F1-Scores are less than first and forth models' scores.