ECG Classification Using Two Class SVM and DNN

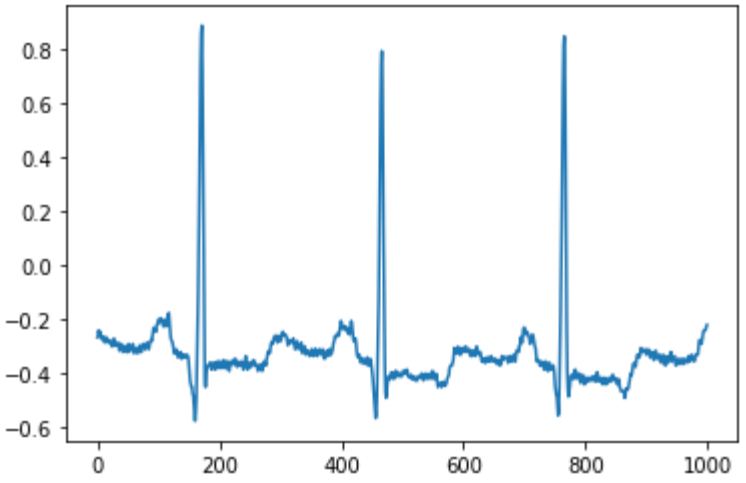
CORY DAVIS

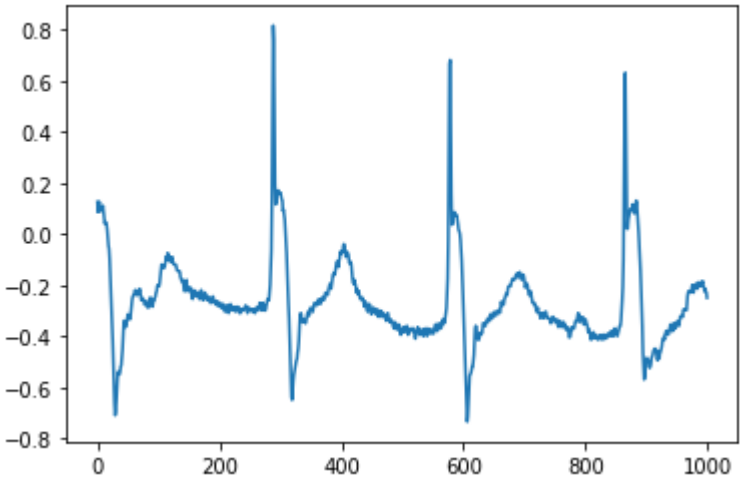
SAYLI APHALE

Contents

Introduction

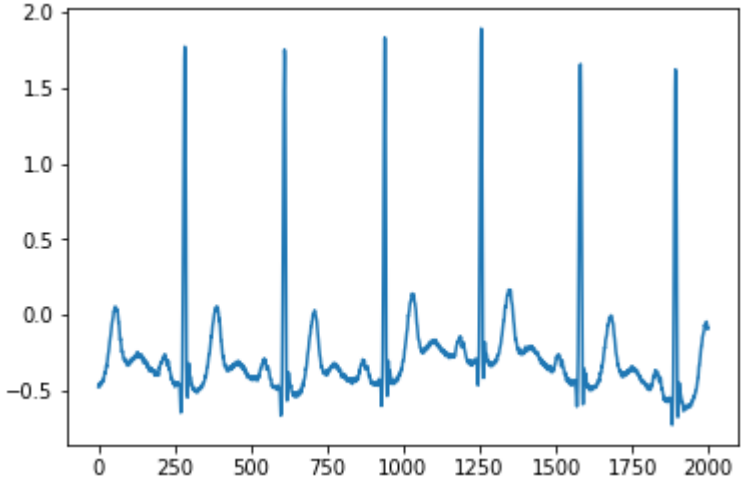
- Electrocardiograms (ECGs) are test to record heartbeats
- Measure electrical pulses of heartbeats
- Cardiologist reads the recordings to detect abnormalities and diagnose heart conditions
- MIT-BIH Arrhythmia Dataset
  - 48 recordings – 30 minutes per recording
  - Approximately 112,000 total heartbeats
  - Annotations of each peak provides beat type

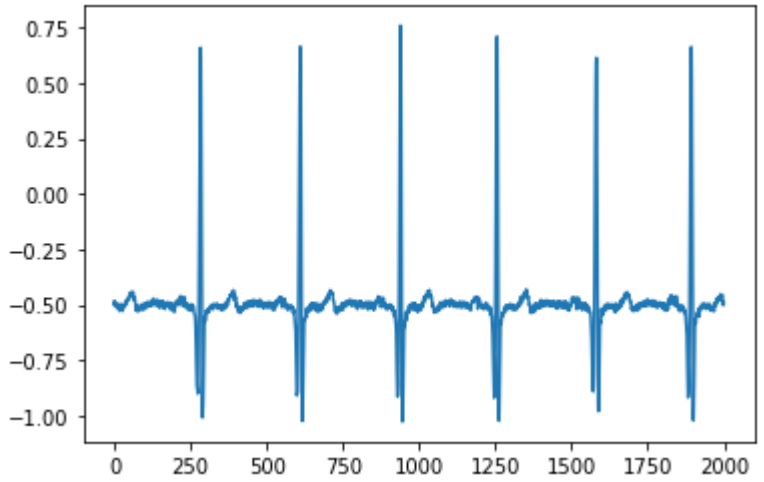




Introduction

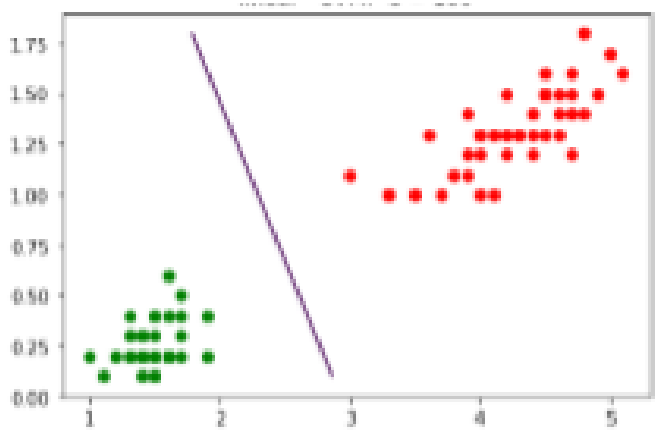
- Preprocessing
- Data contains low-frequency variation
  - Baseline wander or drift
- Data passes through a filter to remove this
- Allows for better wave inspection and accurate data processing
- Pulses are isolated and resampled to length 200
- Processed using FFT

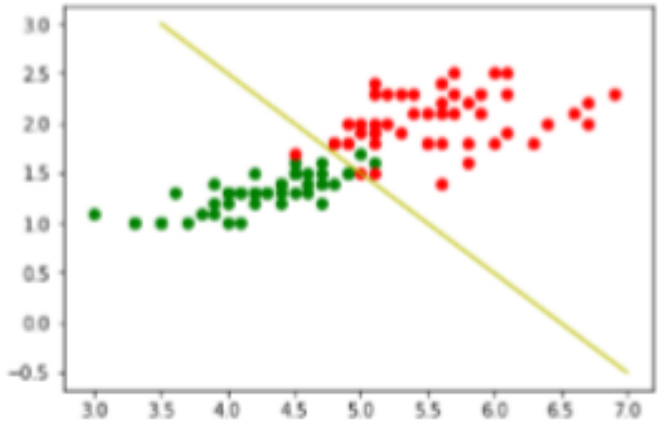




Background: Support Vector Machine

- ❑ Supervised machine learning method for data classification
- ❑ This work uses two-class classification
- ❑ Goal of SVM is to determine hyperplane that correctly divides training data
- ❑ Clearly separated 2-D data is easy
- ❑ What about non-perfectly separable data?
- ❑ Hard SVM
- ❑ Soft SVM



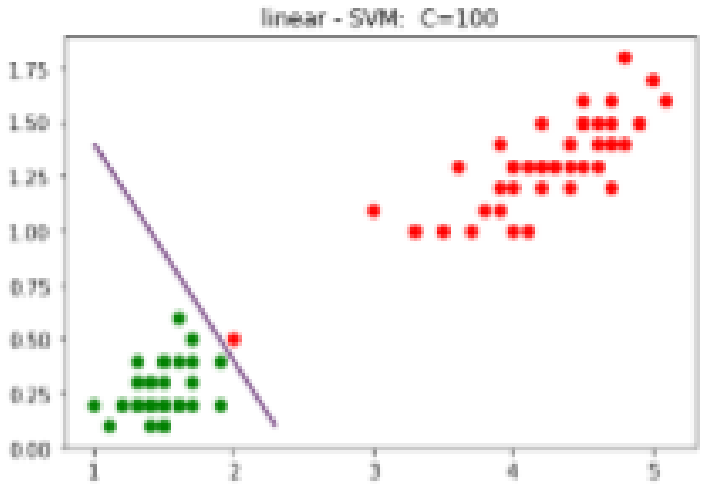


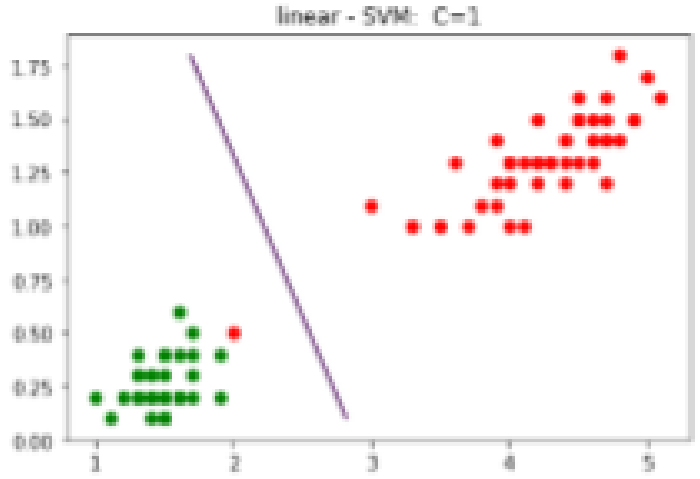
Background: Support Vector Machine

- C – optimization Coefficient

$$\begin{aligned} \min_{\beta,c,\eta} \quad & \beta^T\beta + C\sum_{i=1}^{n}\eta_i \\ \text{Subject to} \quad & y_i(\beta^T x_i + c) \geq 1 - \eta_i, \quad \text{for } i = 1, 2, \ldots, n \\ \text{and} \quad & \eta_i \geq 0, \quad \text{for } i = 1, 2, \ldots, n. \end{aligned}$$

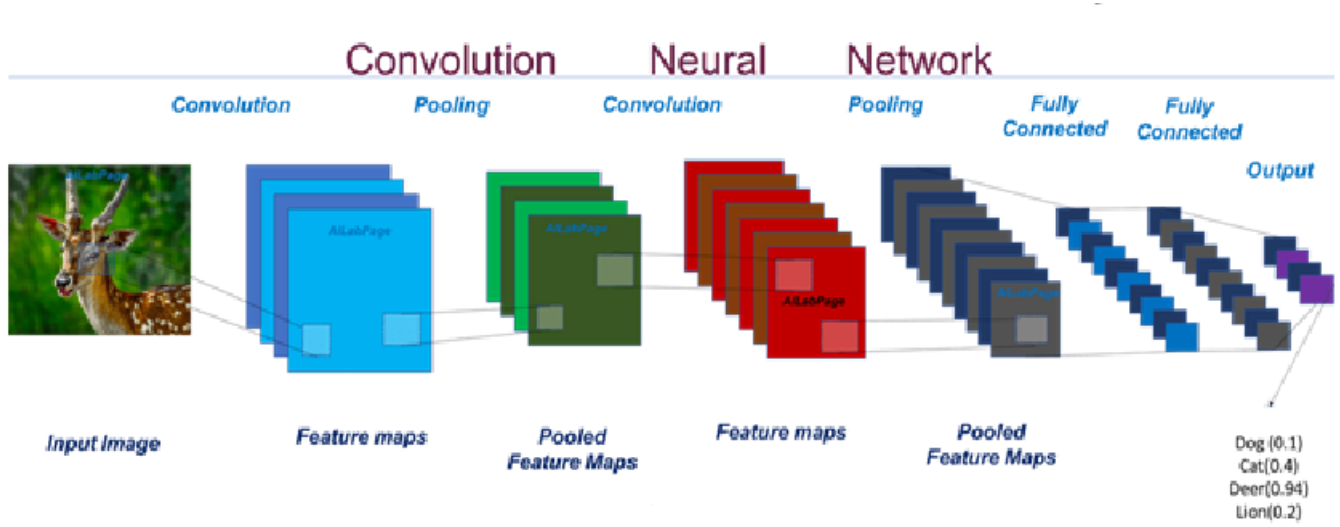
- Higher values of C creates a strict, and more unstable separation
- Lower values of C creates a less restrictive and more stable fit.
- Range of C values must be tested for peak accuracy





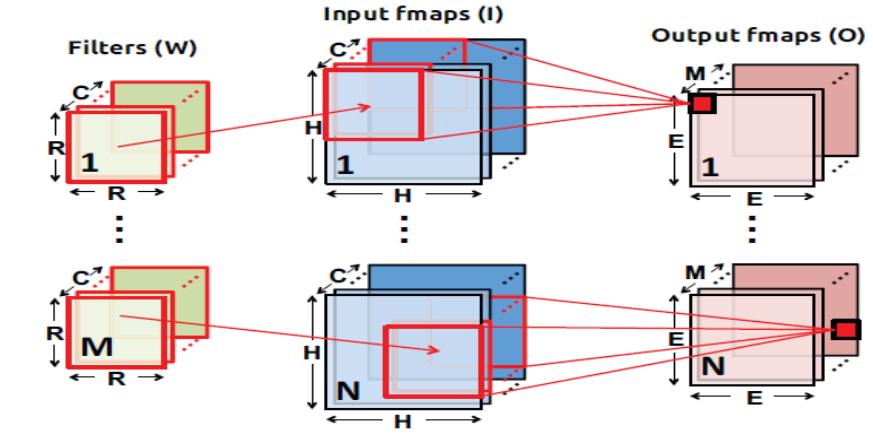
Background of Neural Networks

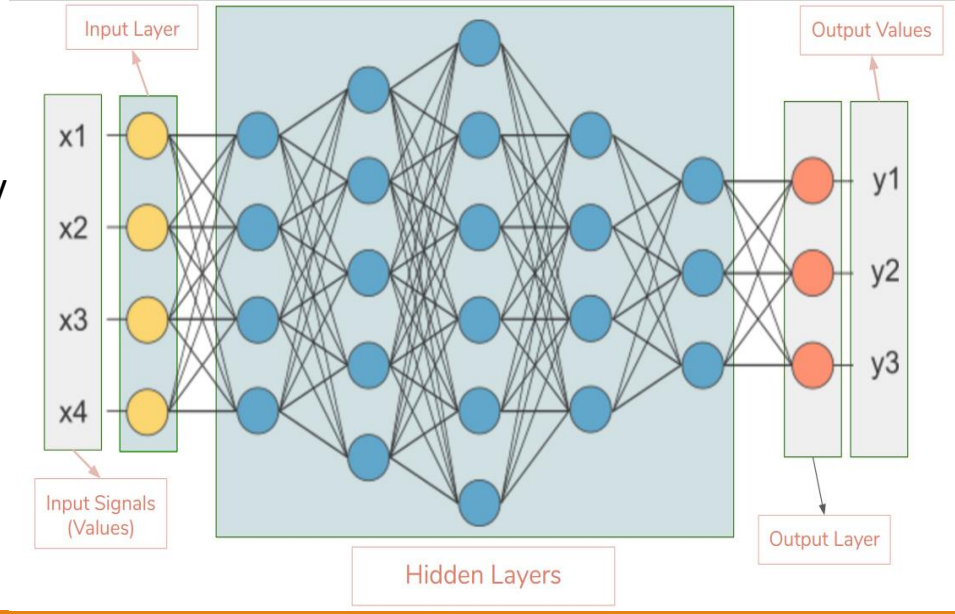
- ❑ Neural Networks Designed to recognize patterns
- ❑ Convolution Neural Networks are constructed by stacking Multiple
- ❑Convolutional Neural Networks are constructed by stacking Multiple computational layers, each layer provides a higher layer of abstraction of Input data –Feature map.
- ❑ Performance of a CNN depends on the density of the network i.e. the type and number of layers used to design.
- ❑ Types of CNN – VGG-16, ResNet50, Xception, etc
- ❑ Types of layers in CNN
  - ▪Input Layer
  - ▪Hidden layer
  - ▪Output layer
- ❑ Hidden layer carrier out feature extraction



Background of Neural Networks

- Hidden layers in CNN:
  - Convolution layer : Uses Matrix filter to detect patterns in image
  - Activation Function : Rectification pf the feature map of the image
  - Pooling layer : Identify edges, corners, etc along with down sampling
  - Flatten layer: To convert 2-D array into single linear vector
  - Fully connected layer : Classification of images

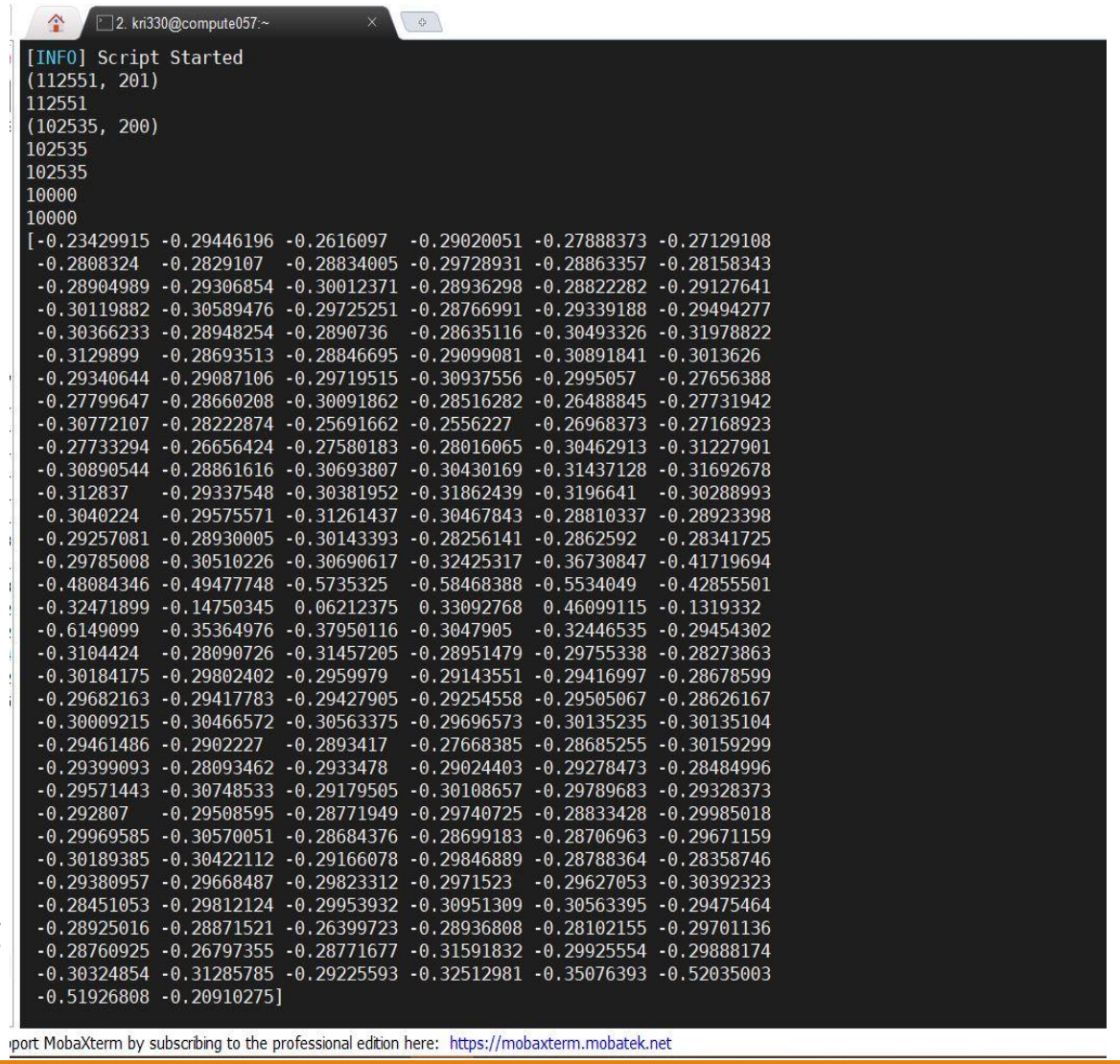




- CNN Decoder Ring states dimensions of the Input Feature maps, Filters , Output
- Today CNN are designed to be very dense to obtain : High performance Accuracy
- Time required to train dense CNN is large: Large Number of Layers & Dataset
- Training time can be sizable but the inference time should be lowest
- CNN application are user response based.
- Applications of CNN : Medical, Finance, Infrastructure

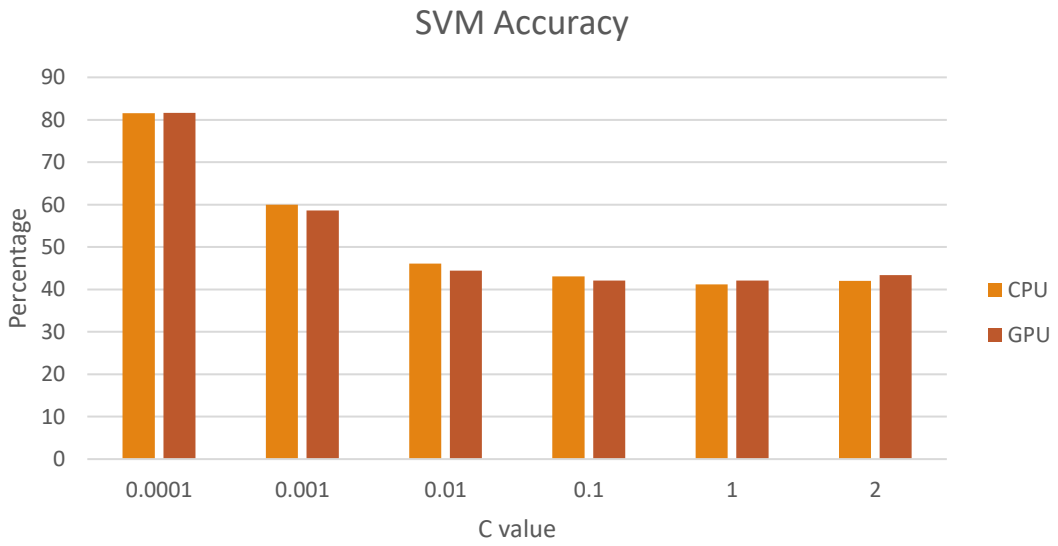
Design of Implemented Sequential Model

- Model : Keras Sequential
  - Total 16 layers
  - Convolution 1-D : 6 layers (input layer)
  - MaxPooling 1-D : 2 layers
  - GlobalMaxPooling : 1 layer
  - Activation Function : ReLu
  - Dense (fully connected) : 3 layers
  - Before feeding input into Dense layer it is Flattened
- Model is then compiled, fitted and testing is performed
- Two sets of data:
  - Set 1: 20,000 beats for Training, 2,000 beats for testing
  - Set 2: 65,000 beats for Training, 10,000 beats for testing
- Model run on CPU and GPU Nodes



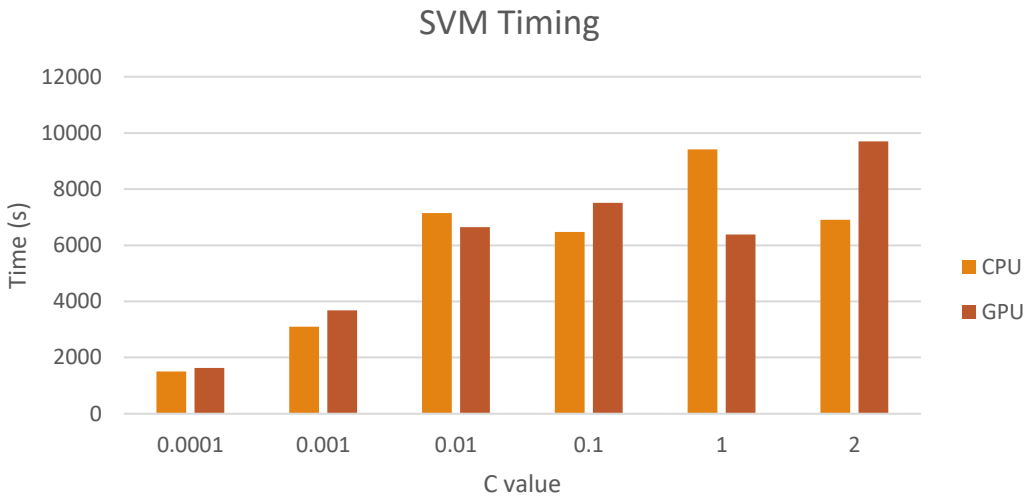
SVM Results

- Tested using C values: 0.0001, 0.001, 0.01, 0.1, 1.0, 2.0
- C value – 0.0001 yielded 81% across CPU and GPU
- 0.5% difference in accuracy in favor of CPU
- By far the best accuracy
- Harder SVM yielded accuracy less than a coin flip
- The variation of “normal” heartbeats amongst patients, lends towards softer SVM



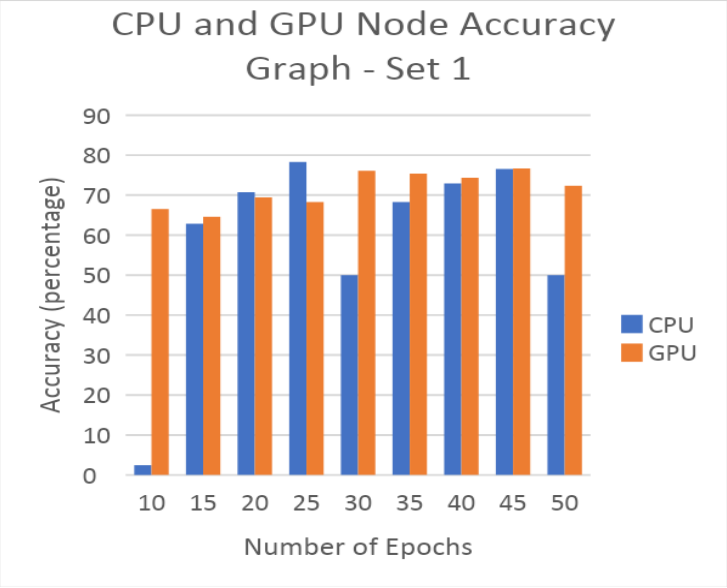
SVM Results

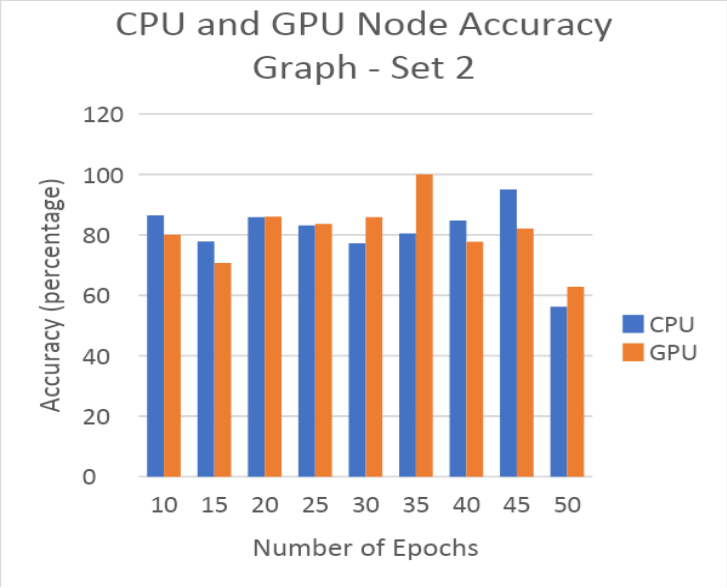
- Tested using C values: 0.0001, 0.001, 0.01, 0.1, 1.0, 2.0
- the GPU took 2.7% longer than the CPU to process the data
- C value – 0.0001 was the fastest to process at about 1600s
- Processing time peaked at 9600s for GPU/C value – 2.0



Neural Network Results

- Accuracy
  - Set 1- CPU Node: Range : 2.45% - 78.3%, Max at epoch =25
  - Set 2- CPU Node: Range : 56.21% - 95.05%, Max at epoch = 45
  - CPU Node: Decrease in correct percentage at epoch = 30 after initial rise
  - Set 1- GPU Node: Range : 64.6% – 76.64% , Max at epoch = 45
  - Set 2- GPU Node: Range : 68.20% - 100.0%, Max at epoch = 35
  - GPU Node : Set 1: Epoch range 30-45 is ideal as the variation is limited & For set 2 epoch range 20-35 is giving best outcome





Neural Network Results

Accuracy : CPU Node

2. kri330@compute057:~

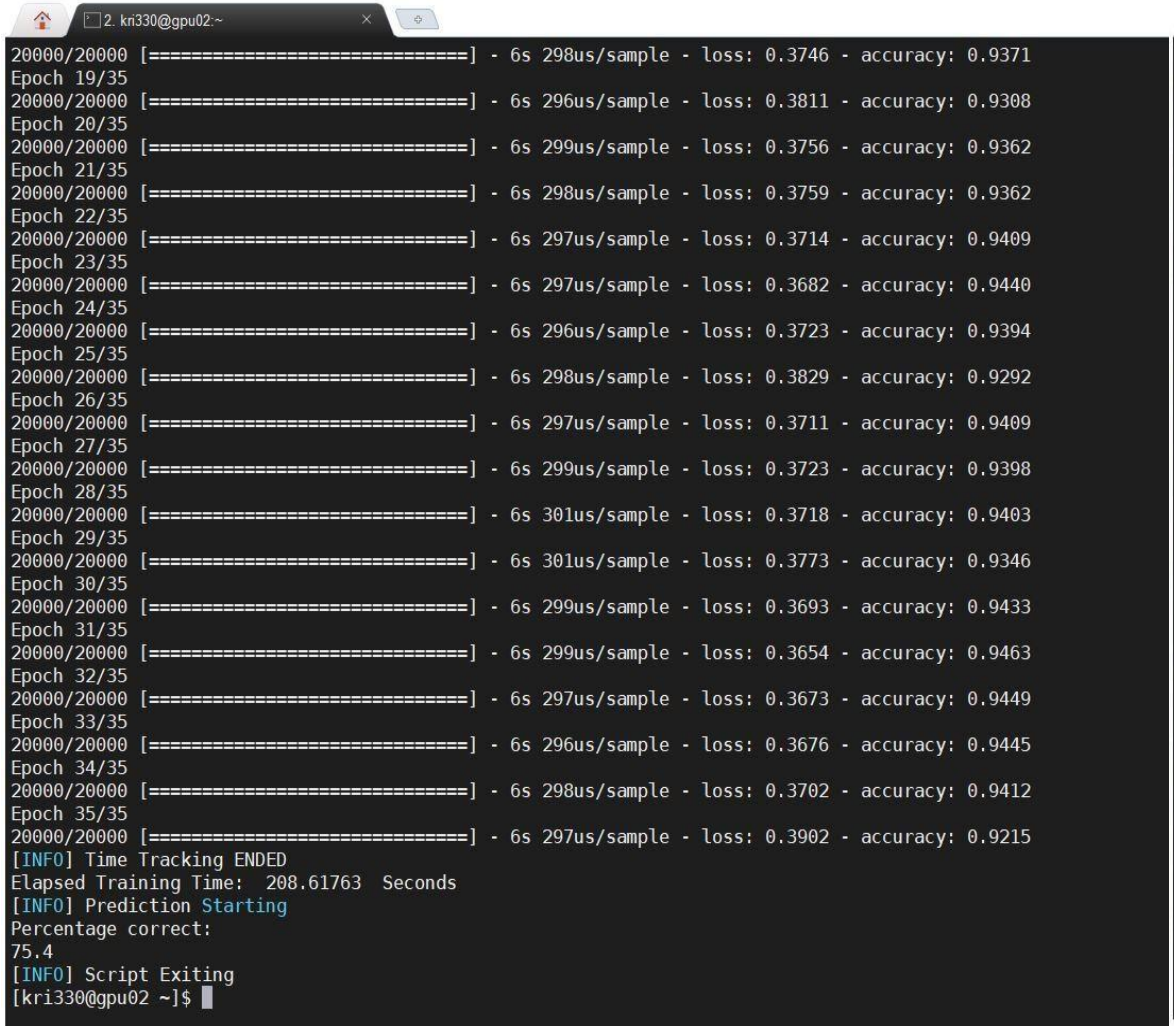
```
20000/20000 [==============================] - 5s 259us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 14/30
20000/20000 [==============================] - 5s 260us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 15/30
20000/20000 [==============================] - 5s 260us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 16/30
20000/20000 [==============================] - 5s 260us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 17/30
20000/20000 [==============================] - 5s 260us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 18/30
20000/20000 [==============================] - 5s 259us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 19/30
20000/20000 [==============================] - 5s 259us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 20/30
20000/20000 [==============================] - 5s 259us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 21/30
20000/20000 [==============================] - 5s 260us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 22/30
20000/20000 [==============================] - 5s 258us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 23/30
20000/20000 [==============================] - 5s 259us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 24/30
20000/20000 [==============================] - 5s 259us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 25/30
20000/20000 [==============================] - 5s 258us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 26/30
20000/20000 [==============================] - 5s 259us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 27/30
20000/20000 [==============================] - 5s 258us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 28/30
20000/20000 [==============================] - 5s 259us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 29/30
20000/20000 [==============================] - 5s 259us/sample - loss: 0.8133 - accuracy: 0.5000
Epoch 30/30
20000/20000 [==============================] - 5s 259us/sample - loss: 0.8133 - accuracy: 0.5000
[INFO] Time Tracking ENDED
Elapsed Training Time:  156.09425  Seconds
[INFO] Prediction Starting
Percentage correct:
50.0
[INFO] Script Exiting
(tensorflow-cpu) [kri330@compute057 ~]$
```

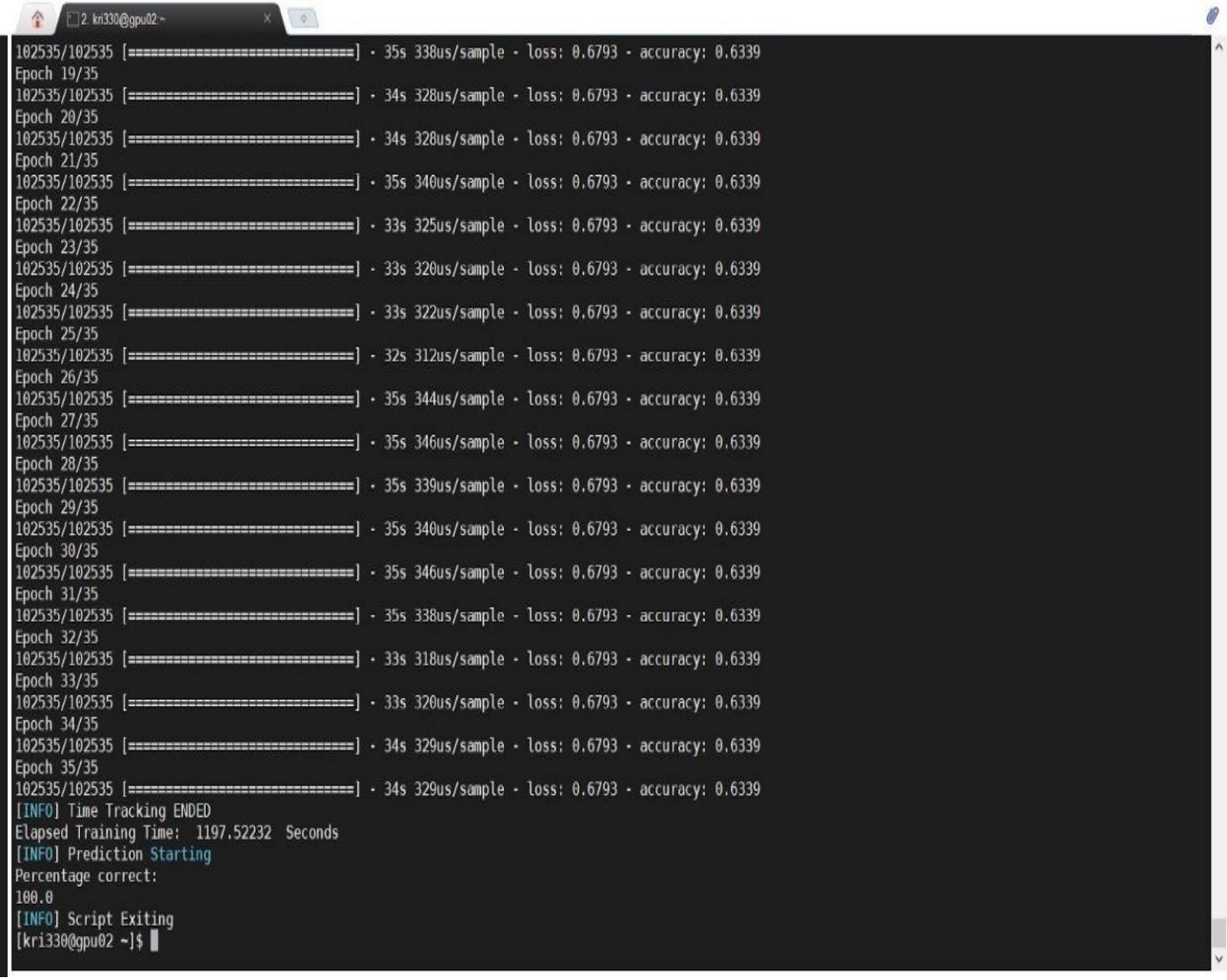
2. kri330@compute031:~

```
102535/102535 [==============================] - 30s 291us/sample - loss: 0.4506 - accuracy: 0.8616
Epoch 29/45
102535/102535 [==============================] - 30s 290us/sample - loss: 0.4492 - accuracy: 0.8633
Epoch 30/45
102535/102535 [==============================] - 30s 291us/sample - loss: 0.4487 - accuracy: 0.8639
Epoch 31/45
102535/102535 [==============================] - 30s 289us/sample - loss: 0.4932 - accuracy: 0.8194
Epoch 32/45
102535/102535 [==============================] - 30s 290us/sample - loss: 0.4991 - accuracy: 0.8137
Epoch 33/45
102535/102535 [==============================] - 30s 290us/sample - loss: 0.5674 - accuracy: 0.7455
Epoch 34/45
102535/102535 [==============================] - 30s 291us/sample - loss: 0.5565 - accuracy: 0.7566
Epoch 35/45
102535/102535 [==============================] - 30s 290us/sample - loss: 0.5125 - accuracy: 0.8006
Epoch 36/45
102535/102535 [==============================] - 30s 289us/sample - loss: 0.5043 - accuracy: 0.8087
Epoch 37/45
102535/102535 [==============================] - 30s 290us/sample - loss: 0.5300 - accuracy: 0.7829
Epoch 38/45
102535/102535 [==============================] - 30s 290us/sample - loss: 0.5728 - accuracy: 0.7404
Epoch 39/45
102535/102535 [==============================] - 30s 289us/sample - loss: 0.6061 - accuracy: 0.7071
Epoch 40/45
102535/102535 [==============================] - 30s 290us/sample - loss: 0.6348 - accuracy: 0.6784
Epoch 41/45
102535/102535 [==============================] - 30s 290us/sample - loss: 0.6055 - accuracy: 0.7078
Epoch 42/45
102535/102535 [==============================] - 30s 291us/sample - loss: 0.6180 - accuracy: 0.6952
Epoch 43/45
102535/102535 [==============================] - 30s 290us/sample - loss: 0.5940 - accuracy: 0.7192
Epoch 44/45
102535/102535 [==============================] - 30s 290us/sample - loss: 0.6395 - accuracy: 0.6738
Epoch 45/45
102535/102535 [==============================] - 30s 290us/sample - loss: 0.6454 - accuracy: 0.6678
[INFO] Time Tracking ENDED
Elapsed Training Time:  1340.40457  Seconds
[INFO] Prediction Starting
Percentage correct:
95.05
[INFO] Script Exiting
(tensorflow-cpu) [kri330@compute031 ~]$
```

Neural Network Results

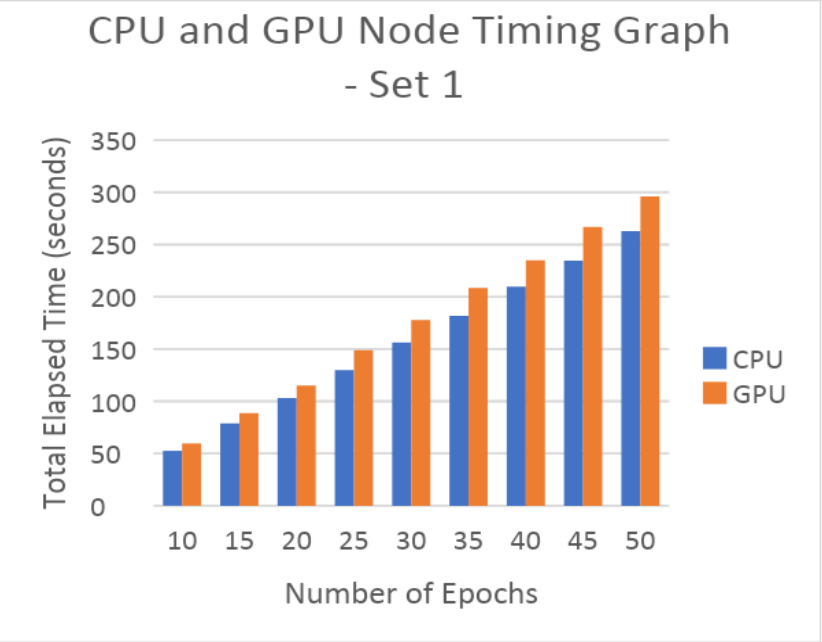
Accuracy : GPU Node

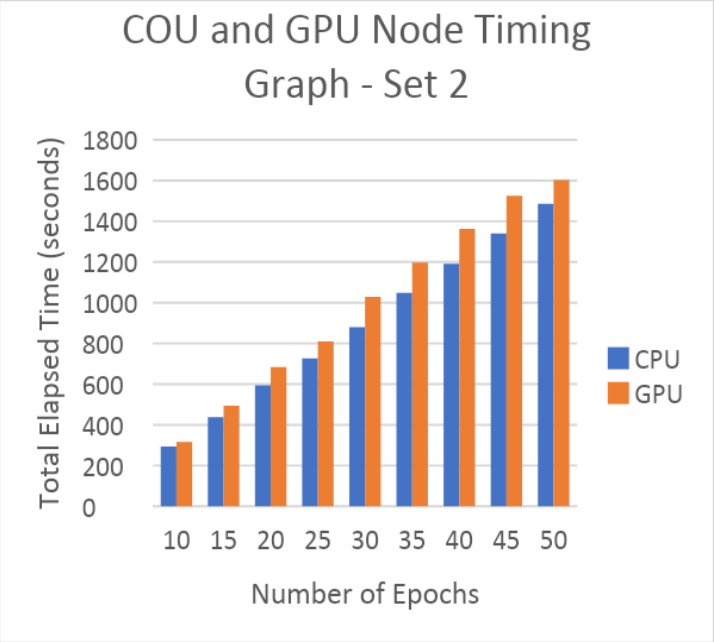




Neural Network Results

- Total Elapsed Time
  - Time in creating and processing the sequential model increases as the number of epochs are incremented
  - With the increase in the training and testing samples there is a rise in time required to process the model
  - GPU takes more time CPU





Comparison CPU and GPU Node

- The time required to process each epoch of the model on CPU node and GPU node is in seconds
- Difference between the time is only a few second more on GPU node than that of CPU node for set 1
- For set 1 data the is not much of a difference in accuracy just 2% and the CPU is performing better than GPU
- On the contrary for large dataset the GPU performs better than CPU by 6 % at lesser number of epochs
- It can be clearly understood that depending on the size of data both the nodes will require different epoch counts to give the best results
- Even though the time consumed is more, the GPU node performs better than the CPU node for the designed model

Conclusion

- There are many ways to perform machine learning.
- This research tests two of those methods for comparison, SVM and CNN.
- SVM is a supervised learning technique that excels at classification of data.
- CNN models a network to recognize patterns between data and excels in this form of classification.
- It is clear from the data that CNN performed significantly better than SVM.
- CNN peaked at 95% whereas the optimal SVM peaked at roughly 82% accurate.
- Using set 2 and 50 epochs, the CNN took 1600s to process
- Shortest time for SVM was also approximately 1600s, while the largest processing time was 9600s
- Typically, a GPU would be expected to process this data faster than a CPU.
- However, in this research the GPU overall took about 3% longer to finish calculations.