DH302 PROJECT PRESENTATION

HEART RATE ESTIMATION FROM PPG SIGNALS

GROUP MEMBERS

Kunal Randad (20D070049)

Sanika Padegaonkar (20D070069)

Sanket Potdar (20D070071)

Settipally Nithish Kumar Reddy (20D070072)

Siddhant Das (20D070075)

DATASETS USED

MTHS Dataset

BIDMC Dataset

HEART RATE MEASURING: PULSE OXIMETRY VS PPG

Pulse oximetry measures pulse rate and oxygen levels using two light wavelengths (red and infrared) on the finger or earlobe for quick results.

Photoplethysmography (PPG) uses light to gauge blood volume changes in tissue, offering a broader range of physiological measurements, including heart rate, respiratory rate, blood pressure, and oxygen saturation.

PPG is versatile but less accurate; pulse oximetry is highly accurate for heart rate and oxygen saturation.

Fig: Heart rate measuring using pulse oximetry

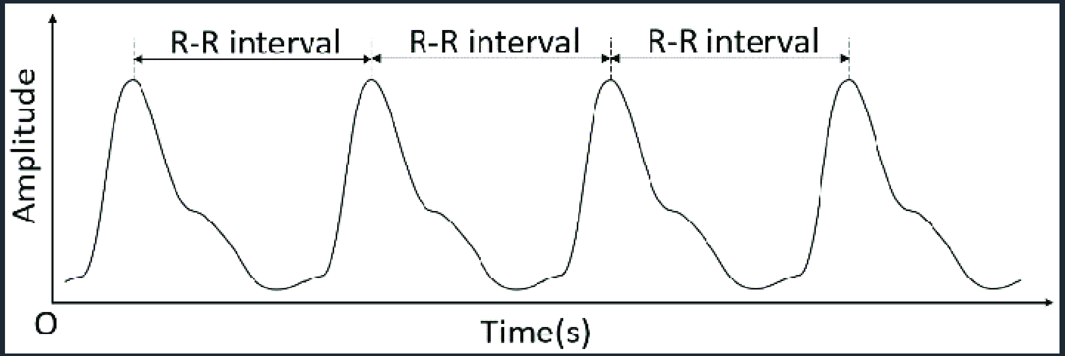


Fig: PPG signal with heartbeat rate R-R time interval

BIDMC DATASET

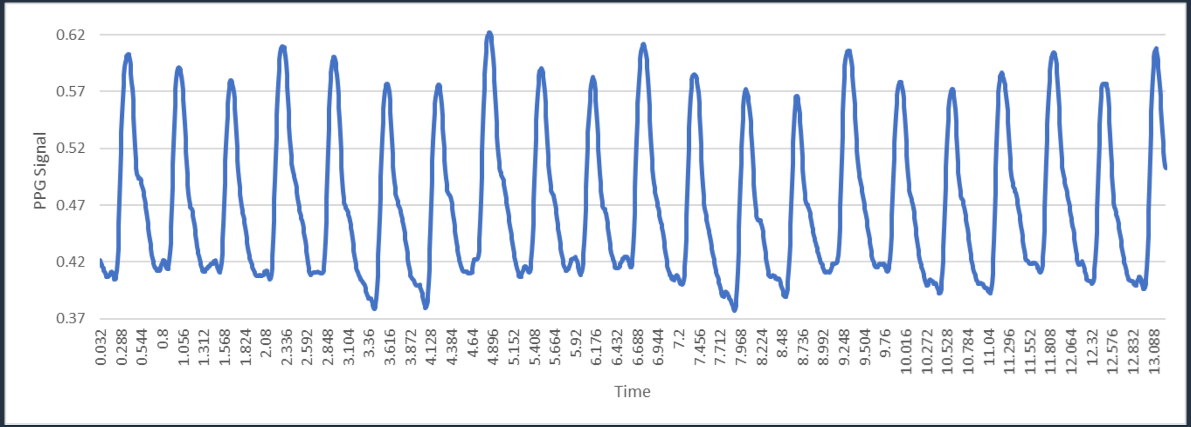


Fig: A sampled PPG Signal for 13s.

- Data from critically-ill patients during hospital care at the Beth Israel Deaconess Medical Centre (Boston, MA, USA).
- It contains following recordings from 53 patients, each 8 minute in duration.
  - Physiological signals: PPG, Impedance Respiratory signal, and Electrocardiogram (ECG). (Sampled at 125 Hz)
  - Physiological parameters: Heart Rate (HR), Respiratory Rate (RR) and SpO2. (Sampled at 1 Hz)
  - Manual breath annotations by two trained annotators.

MTHS DATASET

- A dataset from 62 patients (35 men and 27 women) that contains both hearth rate and SpO2 labels sampled at 1 Hz.
- The PPG signal is acquired at 30 FPS using a smartphone's camera.
- **Inputs:** PPG recordings obtained from the RGB camera of an iPhone 5s at 30fps
- **Outputs/Ground Truths:** HR and SPo2 levels obtained using a pulse oximeter (M70).

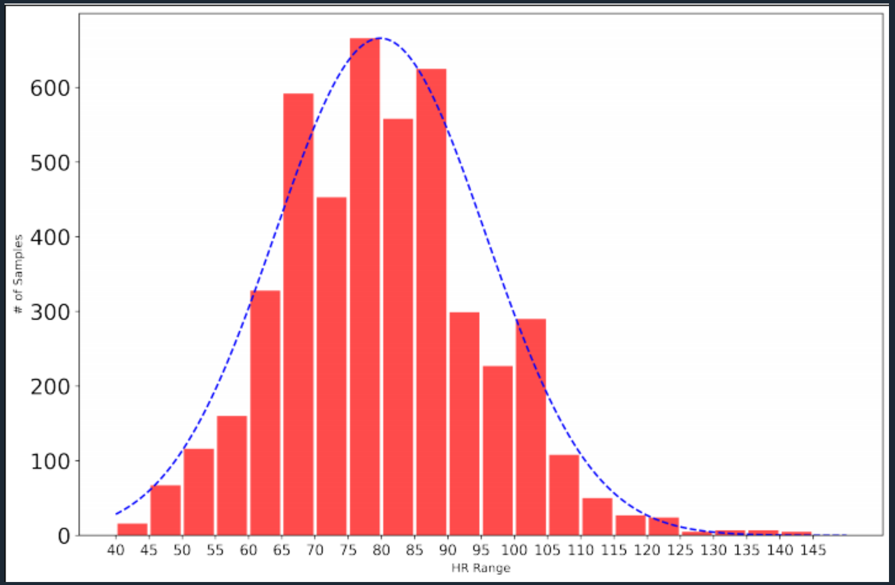


Fig: Heart rate distribution for the MTHS dataset.

Fig: The setup for data collection

DATASET VISUALIZATION

- Frequency of PPG Signals: 30Hz
- Number of PPG Samples: 1800

- Frequency of vital measures: 1Hz
- Number of vital samples: 60

- Time = 60s

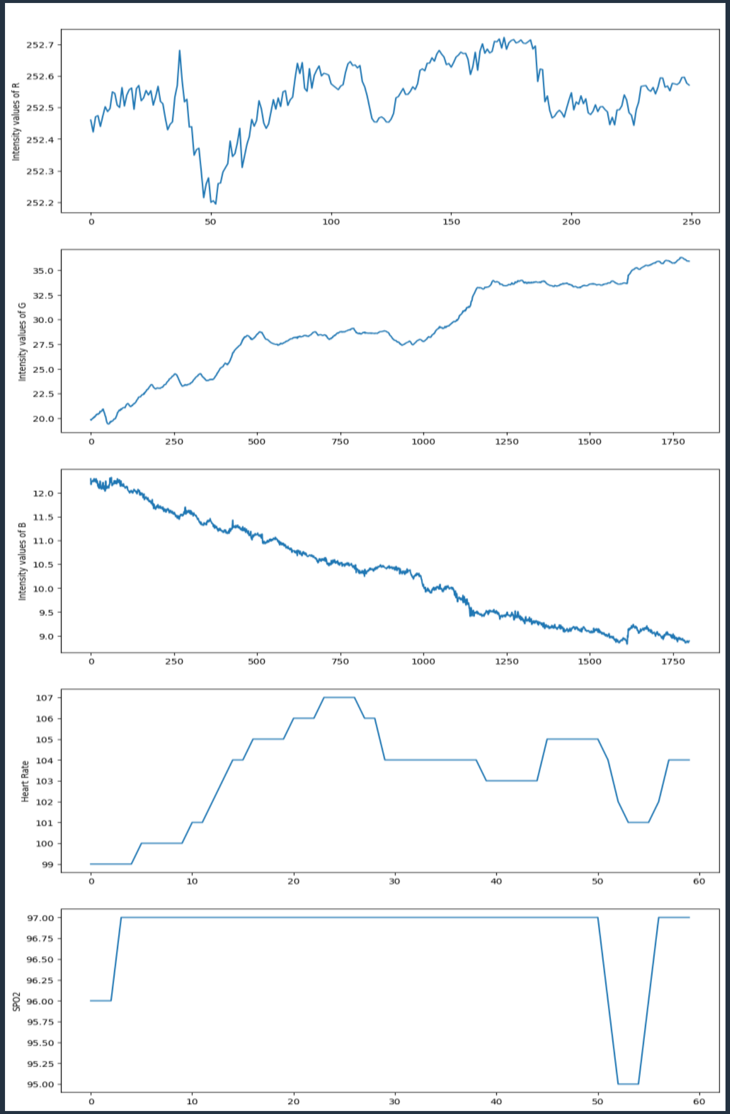


Fig: Patient ID: 50

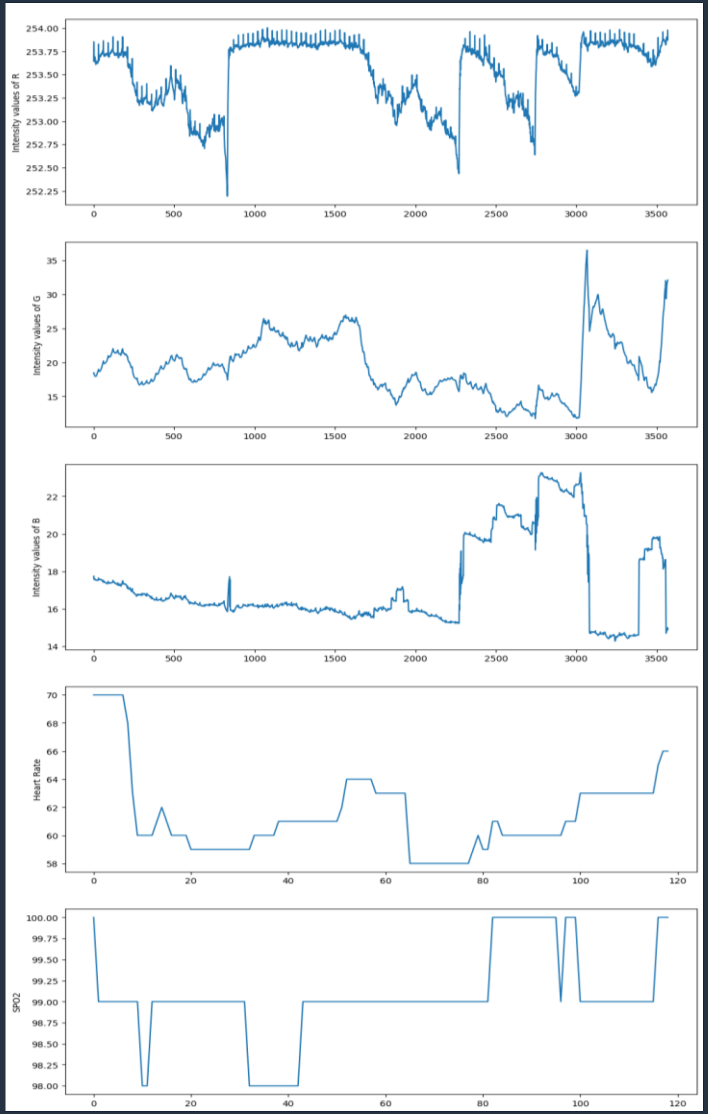


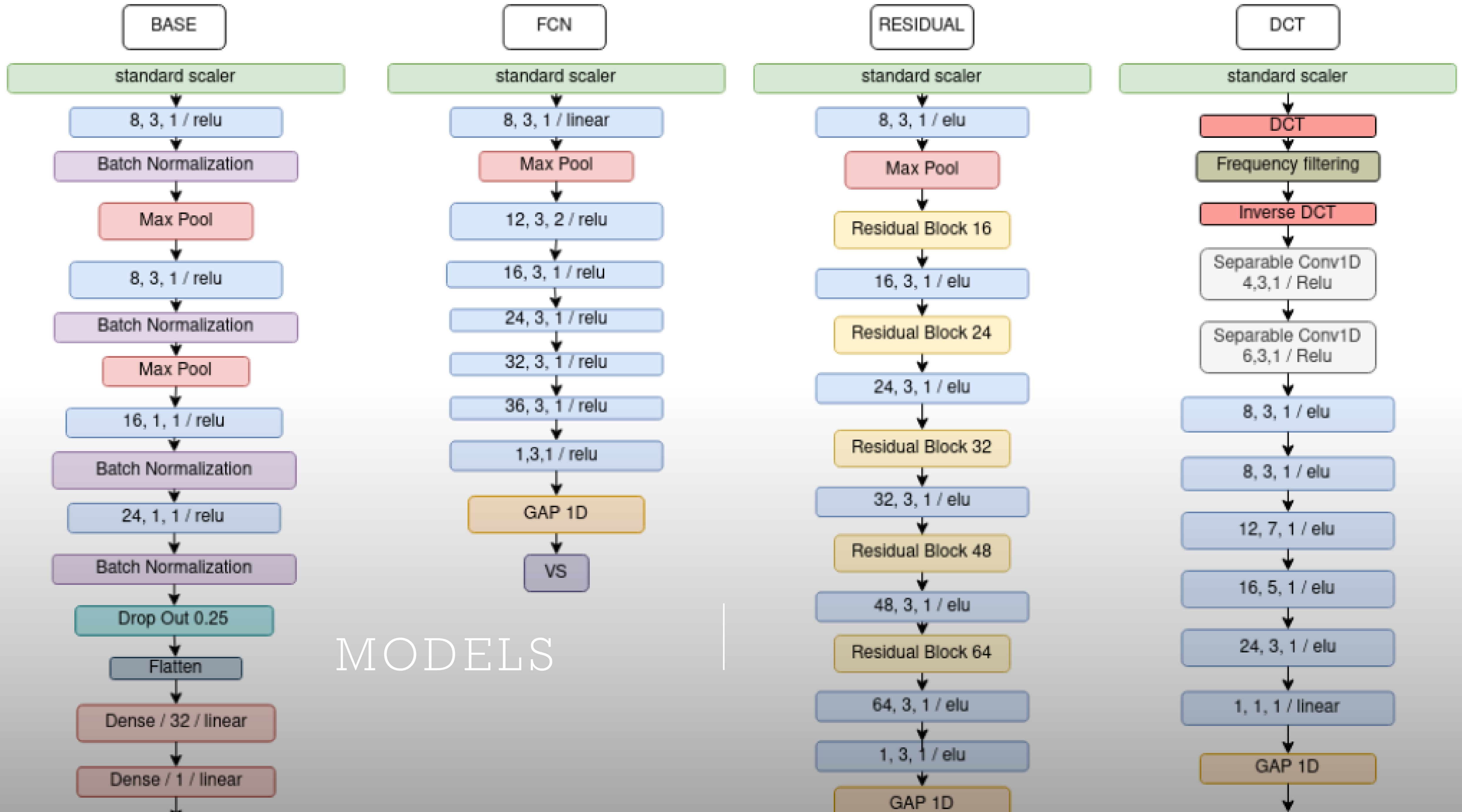
Fig: Patient ID: 23

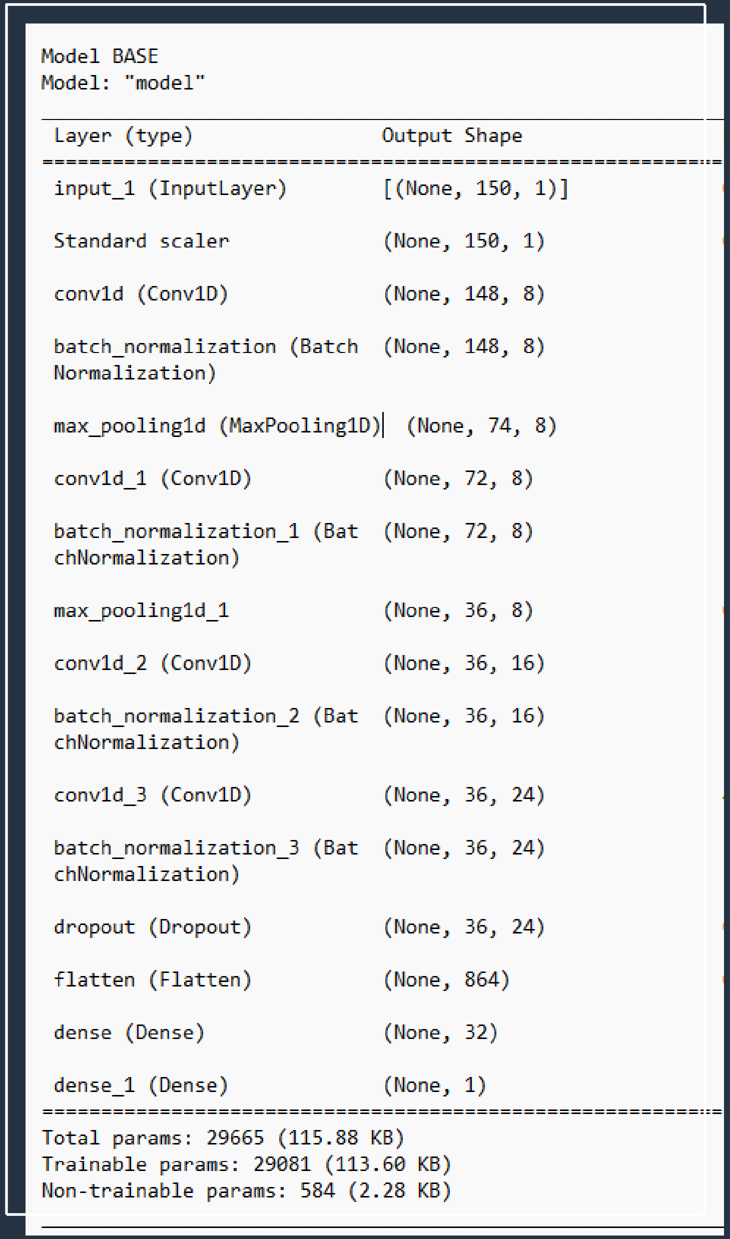
DATALOADER

- Create PPG input (red channel only or all three channels)
    - Down sample to 15 Hz using downsample_factor
        - 2 for MTHS & 8 for BIDMC : 15 samples per second
    - Split into sequences of 10 seconds of signals : 150 samples
- Create target (HR or SpO2)
    - Output of the 10 second gives 1 output: Average heart rate for those 10 seconds

  150 samples input => 1 output heart beat
- Split into train, validation, and test sets
    - 80% train, 15% validation, 20% test
- Return
    - Train, validation, and test sets for PPG input and target

MODELS


BASE
standard scaler
8, 3, 1 / relu
Batch Normalization
Max Pool
8, 3, 1 / relu
Batch Normalization
Max Pool
16, 1, 1 / relu
Batch Normalization
24, 1, 1 / relu
Batch Normalization
Drop Out 0.25
Flatten
Dense / 32 / linear
Dense / 1 / linear
FCN
standard scaler
8, 3, 1 / linear
Max Pool
12, 3, 2 / relu
16, 3, 1 / relu
24, 3, 1 / relu
32, 3, 1 / relu
36, 3, 1 / relu
1,3,1 / relu
GAP 1D
VS
RESIDUAL
standard scaler
8, 3, 1 / elu
Max Pool
Residual Block 16
16, 3, 1 / elu
Residual Block 24
24, 3, 1 / elu
Residual Block 32
32, 3, 1 / elu
Residual Block 48
48, 3, 1 / elu
Residual Block 64
64, 3, 1 / elu
1, 3, 1 / elu
GAP 1D
DCT
standard scaler
DCT
Frequency filtering
Inverse DCT
Separable Conv1D 4,3,1 / Relu
Separable Conv1D 6,3,1 / Relu
8, 3, 1 / elu
8, 3, 1 / elu
12, 7, 1 / elu
16, 5, 1 / elu
24, 3, 1 / elu
1, 1, 1 / linear
GAP 1D



```
Model BASE
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape
=================================================================
 input_1 (InputLayer)        [(None, 150, 1)]

 Standard scaler             (None, 150, 1)

 conv1d (Conv1D)             (None, 148, 8)

 batch_normalization (Batch  (None, 148, 8)
 Normalization)

 max_pooling1d (MaxPooling1D)|  (None, 74, 8)

 conv1d_1 (Conv1D)           (None, 72, 8)

 batch_normalization_1 (Bat  (None, 72, 8)
 chNormalization)

 max_pooling1d_1             (None, 36, 8)

 conv1d_2 (Conv1D)           (None, 36, 16)

 batch_normalization_2 (Bat  (None, 36, 16)
 chNormalization)

 conv1d_3 (Conv1D)           (None, 36, 24)

 batch_normalization_3 (Bat  (None, 36, 24)
 chNormalization)

 dropout (Dropout)           (None, 36, 24)

 flatten (Flatten)           (None, 864)

 dense (Dense)               (None, 32)

 dense_1 (Dense)             (None, 1)

=================================================================
Total params: 29665 (115.88 KB)
Trainable params: 29081 (113.60 KB)
Non-trainable params: 584 (2.28 KB)
_________________________________________________________________
```

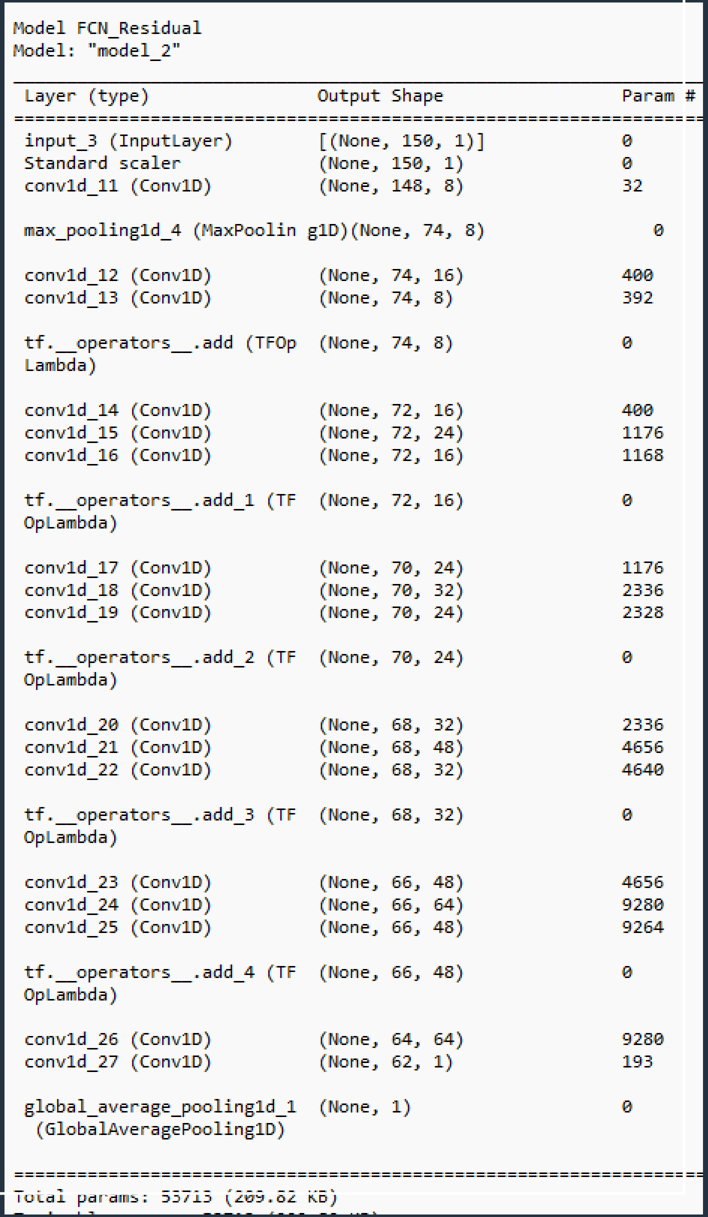
BASE MODEL

Standard scaler layer: Normalize the input , making it more amenable for training

Convolutional Layers: Feature Extraction, Translation Invariance

Batch Normalization: accelerate model training by reducing internal covariate shift, which, in turn, helps in faster convergence.

Max Pooling Layer: Dimension Reduction: managing computational complexity and limiting overfitting.



```
Model FCN_Residual
Model: "model_2"

 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 150, 1)]          0
 Standard scaler             (None, 150, 1)            0
 conv1d_11 (Conv1D)          (None, 148, 8)            32

 max_pooling1d_4 (MaxPoolin g1D)(None, 74, 8)             0

 conv1d_12 (Conv1D)          (None, 74, 16)            400
 conv1d_13 (Conv1D)          (None, 74, 8)             392

 tf.__operators__.add (TFOp  (None, 74, 8)             0
 Lambda)

 conv1d_14 (Conv1D)          (None, 72, 16)            400
 conv1d_15 (Conv1D)          (None, 72, 24)            1176
 conv1d_16 (Conv1D)          (None, 72, 16)            1168

 tf.__operators__.add_1 (TF  (None, 72, 16)            0
 OpLambda)

 conv1d_17 (Conv1D)          (None, 70, 24)            1176
 conv1d_18 (Conv1D)          (None, 70, 32)            2336
 conv1d_19 (Conv1D)          (None, 70, 24)            2328

 tf.__operators__.add_2 (TF  (None, 70, 24)            0
 OpLambda)

 conv1d_20 (Conv1D)          (None, 68, 32)            2336
 conv1d_21 (Conv1D)          (None, 68, 48)            4656
 conv1d_22 (Conv1D)          (None, 68, 32)            4640

 tf.__operators__.add_3 (TF  (None, 68, 32)            0
 OpLambda)

 conv1d_23 (Conv1D)          (None, 66, 48)            4656
 conv1d_24 (Conv1D)          (None, 66, 64)            9280
 conv1d_25 (Conv1D)          (None, 66, 48)            9264

 tf.__operators__.add_4 (TF  (None, 66, 48)            0
 OpLambda)

 conv1d_26 (Conv1D)          (None, 64, 64)            9280
 conv1d_27 (Conv1D)          (None, 62, 1)             193

 global_average_pooling1d_1  (None, 1)                 0
  (GlobalAveragePooling1D)

=================================================================
Total params: 53713 (209.82 KB)
```

FCN RESIDUAL

Residual Blocks:

Mitigating Vanishing Gradients: Residual blocks address the vanishing gradient problem by introducing skip connections. These connections allow gradients to flow more easily during training, which is especially beneficial for deep networks.

Improved Training: Residual connections make training deep networks more efficient and result in faster convergence. They enable the training of very deep neural networks without suffering from gradient-related issues.

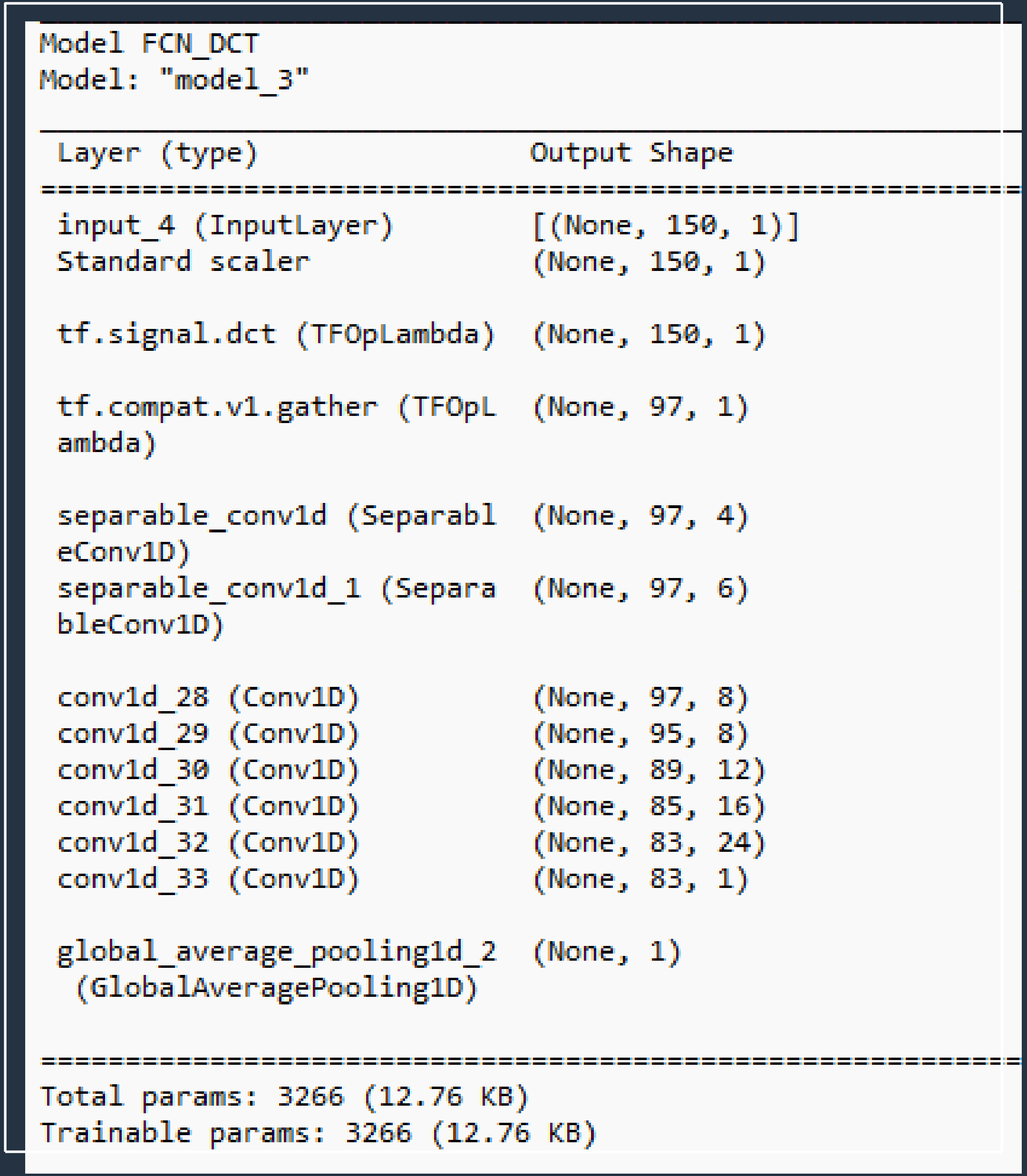
FCN DCT

DCT Layer:

Signal Compression: DCT (Discrete Cosine Transform) is efficient in representing the signal in a compact manner. It condenses the information in the input sequence while preserving essential features.

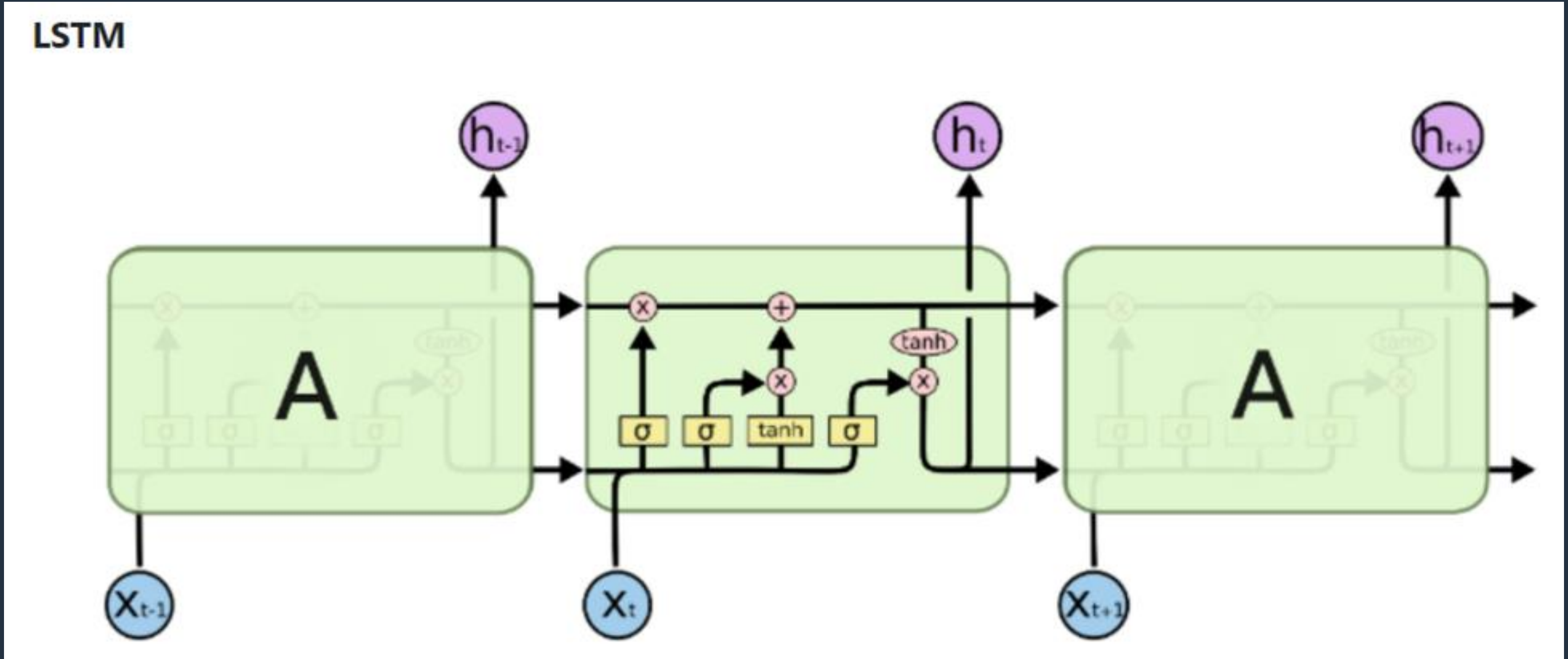
Reducing Dimensionality: DCT can help in dimensionality reduction, making the input data more manageable while retaining relevant information.

Feature Extraction: DCT can highlight relevant frequency components in the signal, aiding in feature extraction.



```
Model FCN_DCT
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape
=================================================================
 input_4 (InputLayer)        [(None, 150, 1)]
 Standard scaler             (None, 150, 1)

 tf.signal.dct (TFOpLambda)  (None, 150, 1)

 tf.compat.v1.gather (TFOpL  (None, 97, 1)
 ambda)

 separable_conv1d (Separabl  (None, 97, 4)
 eConv1D)
 separable_conv1d_1 (Separa  (None, 97, 6)
 bleConv1D)

 conv1d_28 (Conv1D)          (None, 97, 8)
 conv1d_29 (Conv1D)          (None, 95, 8)
 conv1d_30 (Conv1D)          (None, 89, 12)
 conv1d_31 (Conv1D)          (None, 85, 16)
 conv1d_32 (Conv1D)          (None, 83, 24)
 conv1d_33 (Conv1D)          (None, 83, 1)

 global_average_pooling1d_2  (None, 1)
  (GlobalAveragePooling1D)

=================================================================
Total params: 3266 (12.76 KB)
Trainable params: 3266 (12.76 KB)
```

LSTM | Long Short-term Memory



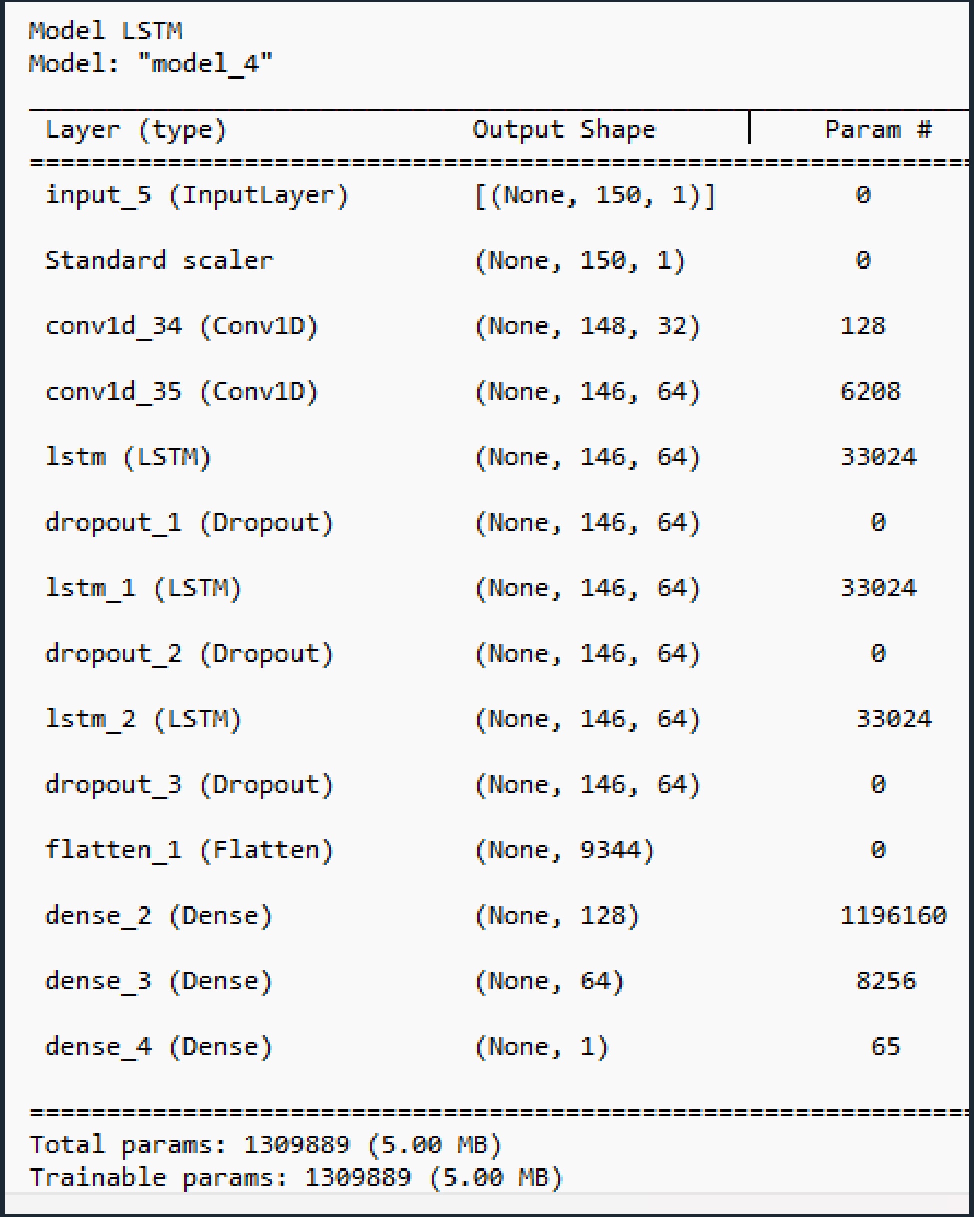
The repeating module in an LSTM contains four interacting layers.

LSTM

LSTM layers:

LSTMs are capable of capturing long-term dependencies in sequential data. They maintain an internal state that can store and retrieve information over extended time intervals.

Vanishing Gradient Mitigation: LSTMs mitigate the vanishing gradient problem, making them suitable for training deep networks. This is achieved through the gating mechanisms (forget, input, and output gates) that control information flow.



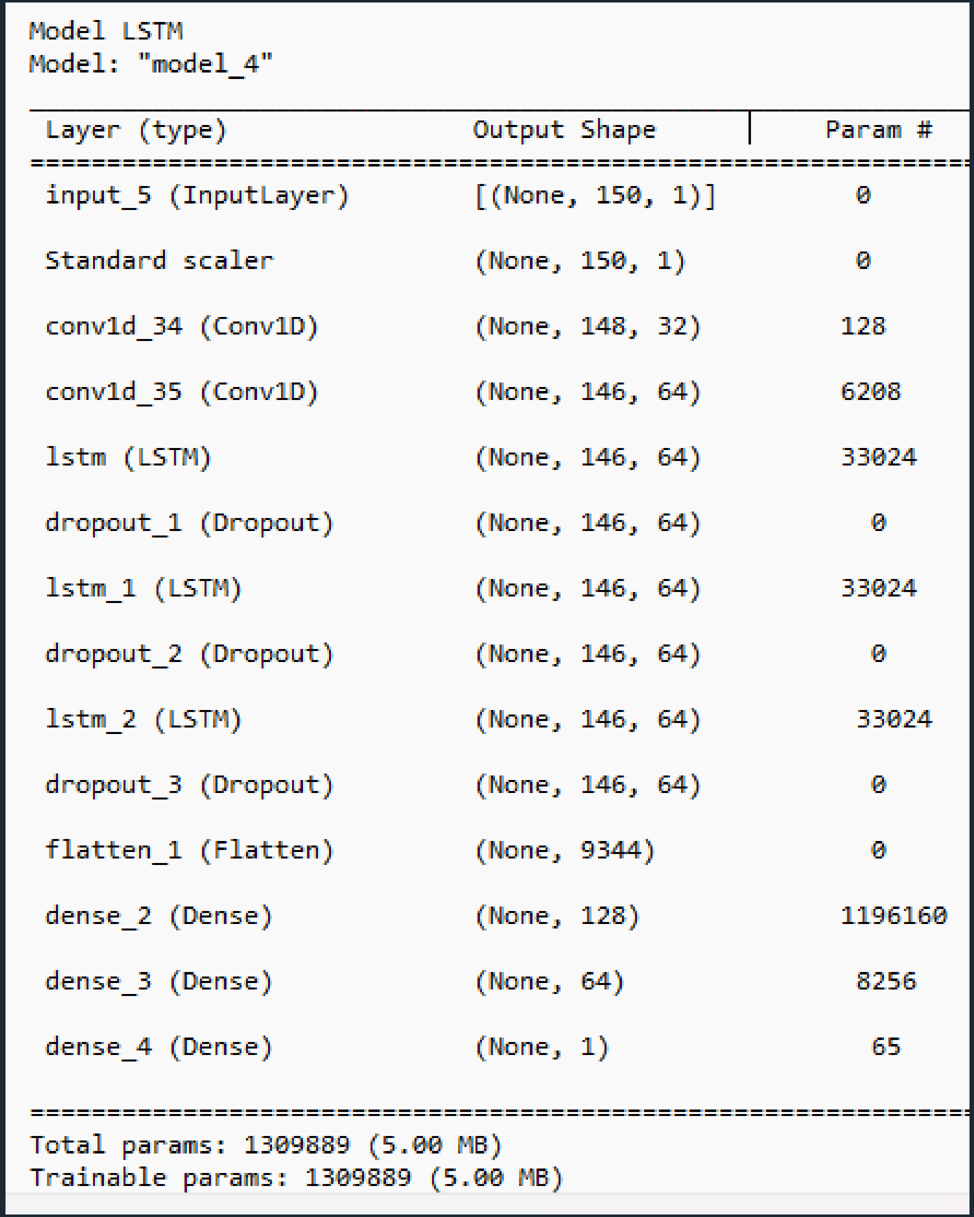
```
Model LSTM
Model: "model_4"

 Layer (type)                Output Shape             Param #
=============================================================
 input_5 (InputLayer)        [(None, 150, 1)]          0

 Standard scaler             (None, 150, 1)            0

 conv1d_34 (Conv1D)          (None, 148, 32)           128

 conv1d_35 (Conv1D)          (None, 146, 64)           6208

 lstm (LSTM)                 (None, 146, 64)           33024

 dropout_1 (Dropout)         (None, 146, 64)           0

 lstm_1 (LSTM)               (None, 146, 64)           33024

 dropout_2 (Dropout)         (None, 146, 64)           0

 lstm_2 (LSTM)               (None, 146, 64)           33024

 dropout_3 (Dropout)         (None, 146, 64)           0

 flatten_1 (Flatten)         (None, 9344)              0

 dense_2 (Dense)             (None, 128)               1196160

 dense_3 (Dense)             (None, 64)                8256

 dense_4 (Dense)             (None, 1)                 65

=============================================================
Total params: 1309889 (5.00 MB)
Trainable params: 1309889 (5.00 MB)
```

LSTM

Drop out layers:

Regularization: Dropout layers introduce a form of regularization by randomly deactivating a portion of neurons during training. This helps prevent overfitting and improves model generalization.

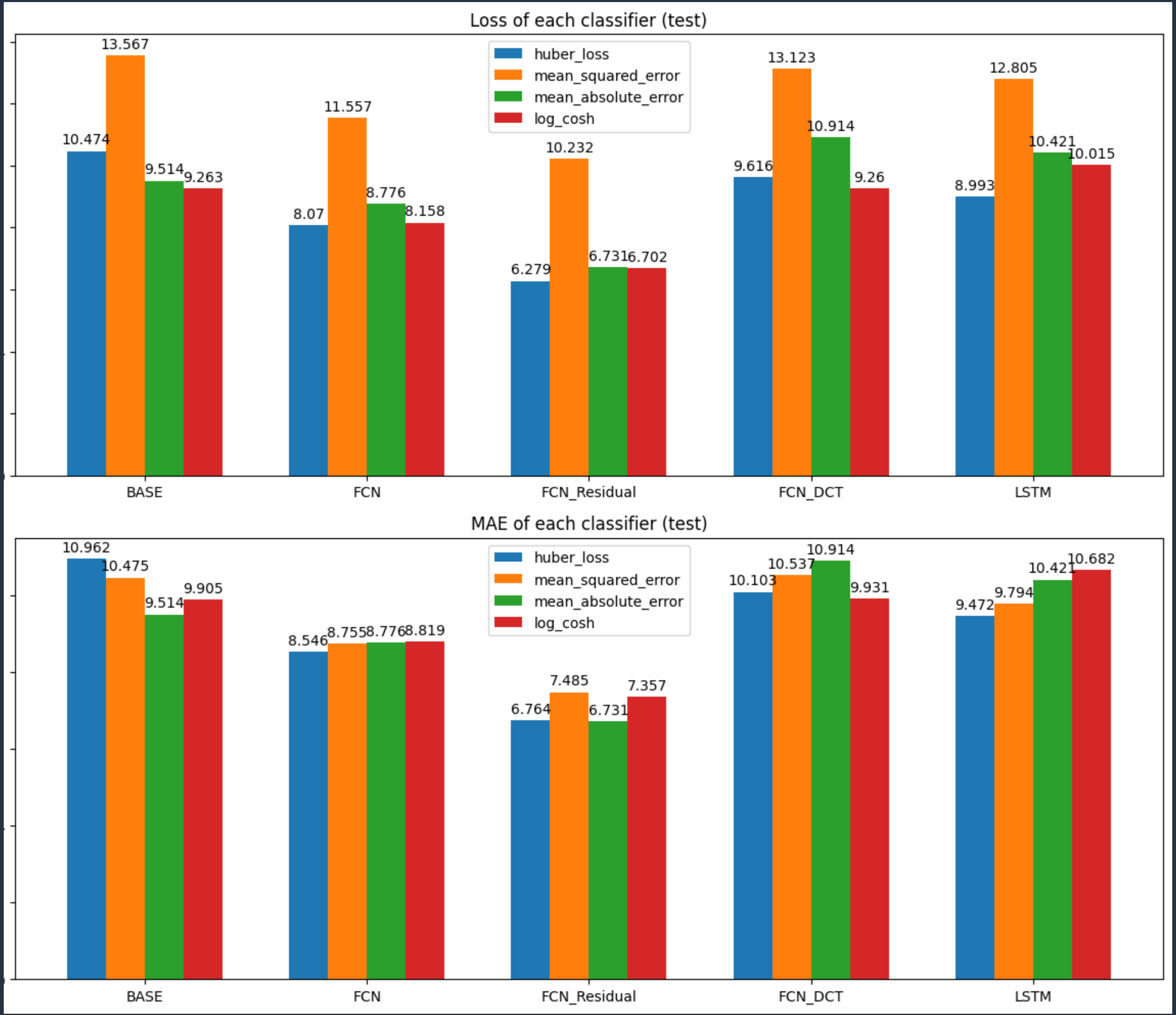
Reducing Co-Adaptation: Dropout discourages neurons from relying too heavily on specific input features, encouraging them to be more robust and adaptive.



```
Model LSTM
Model: "model_4"

 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 150, 1)]          0

 Standard scaler             (None, 150, 1)            0

 conv1d_34 (Conv1D)          (None, 148, 32)           128

 conv1d_35 (Conv1D)          (None, 146, 64)           6208

 lstm (LSTM)                 (None, 146, 64)           33024

 dropout_1 (Dropout)         (None, 146, 64)           0

 lstm_1 (LSTM)               (None, 146, 64)           33024

 dropout_2 (Dropout)         (None, 146, 64)           0

 lstm_2 (LSTM)               (None, 146, 64)           33024

 dropout_3 (Dropout)         (None, 146, 64)           0

 flatten_1 (Flatten)         (None, 9344)              0

 dense_2 (Dense)             (None, 128)               1196160

 dense_3 (Dense)             (None, 64)                8256

 dense_4 (Dense)             (None, 1)                 65

=================================================================
Total params: 1309889 (5.00 MB)
Trainable params: 1309889 (5.00 MB)
```

LOSS FUNCTIONS USED

- Mean Squared Error (MSE): The most common and simplest loss function. We take the difference between the predicted value and ground truth, square it and average it out across the whole dataset.
- Mean Absolute Error (MAE): We take the difference between the model's prediction and the ground truth, apply the absolute value and average it out over the whole dataset.
- Huber Loss: It is a combination of mean squared error and mean absolute error. This makes it more robust to outliers compared to MSE.
- Log Cosh Loss: log(cosh(x)) works mostly like mean squared error but will not be so strongly affected by the occasional wildly incorrect prediction.



PERFORMANCE EVALUATION

FCN Residual model is better at predicting the target variable than the other models.

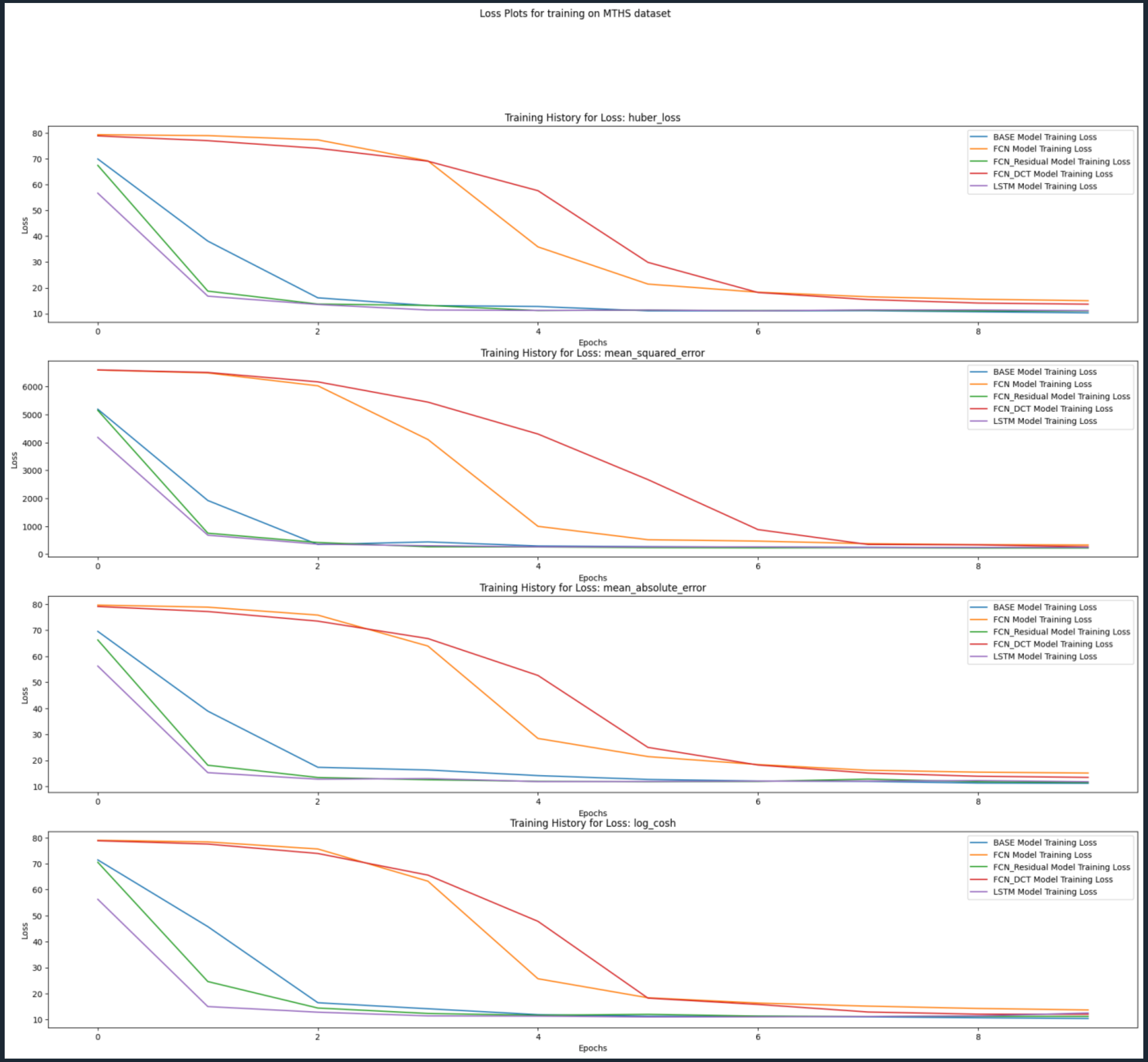
Huber loss gives best performance for almost all the models except our base model

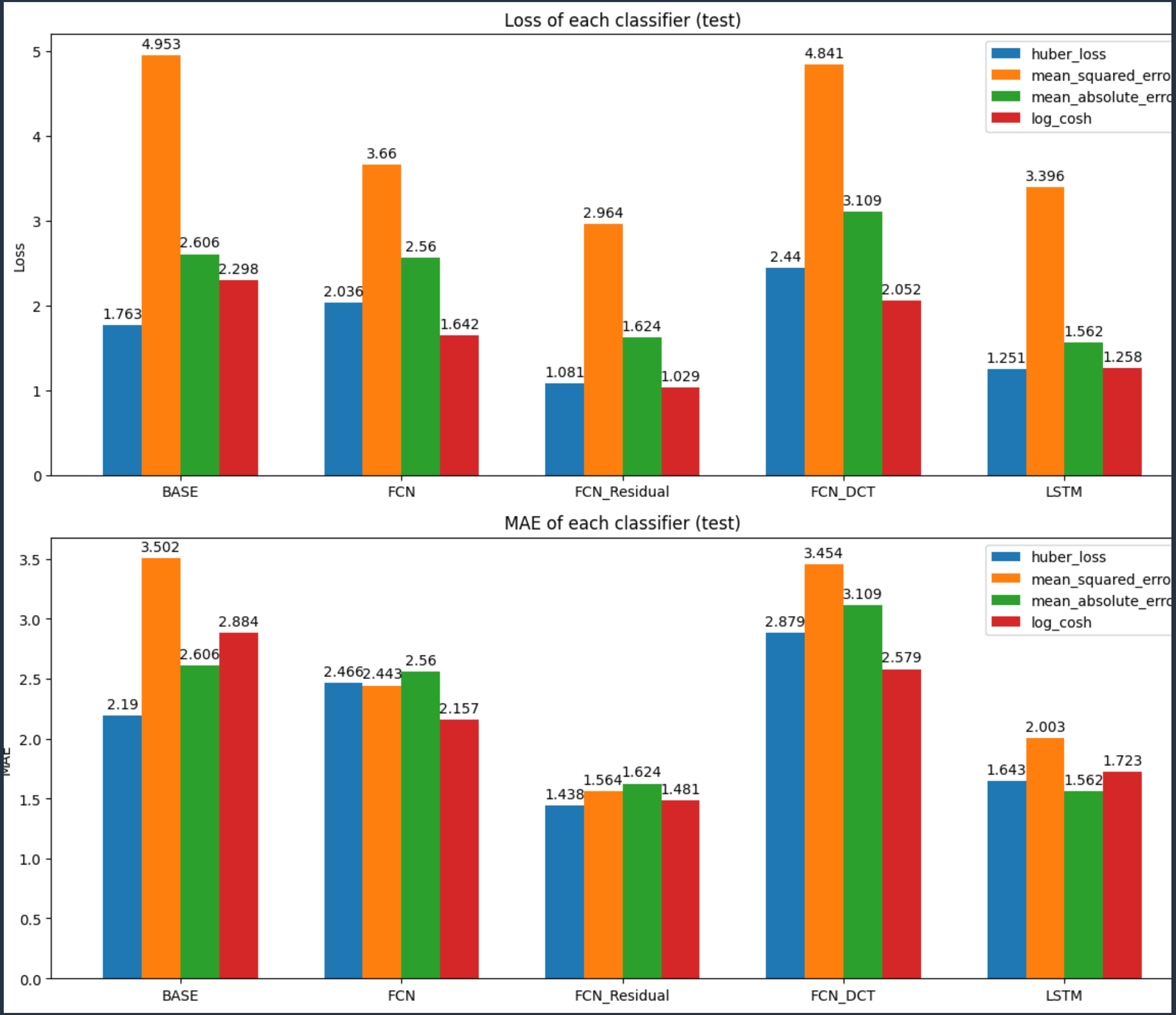
MODEL CONVERGENCE

LSTM model has faster convergence to other models.

- save computational time and resources during the training process.

Although FCN-residual exhibits lower loss after a significant number of epochs

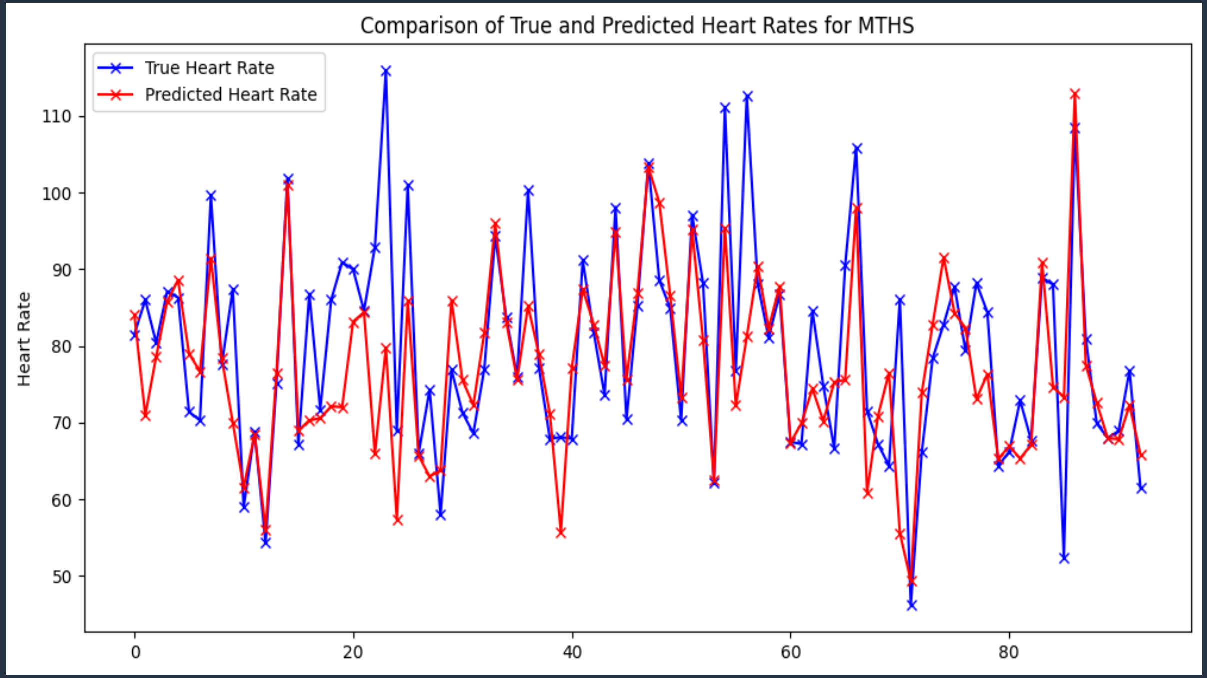




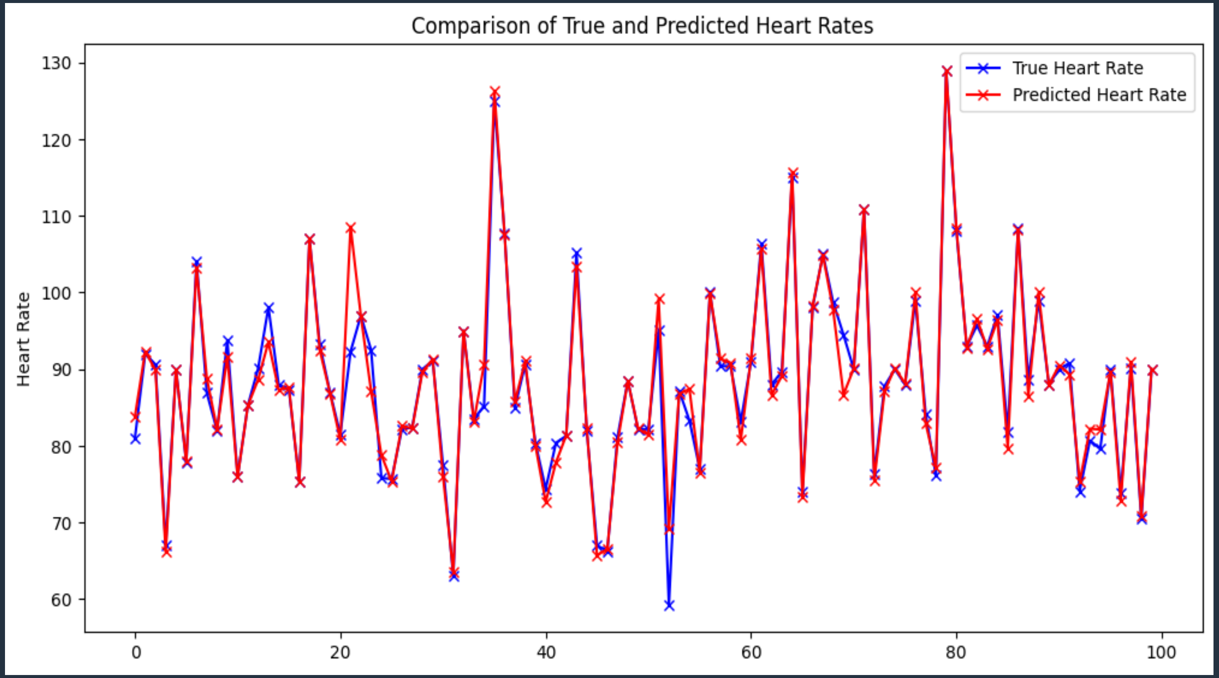
BIDMC RESULTS

We observe that all the loss values are less than those compared to our MTHS dataset

COMPARISION OF HEART RATE GROUND TRUTHS



MTHS dataset



BIDMC dataset

NOVEL EXPERIMENTS

- LSTM Model : We added LSTM architecture and evaluated it on all 4 losses which performed comparable to previous models but converged faster.
- BIDM Dataset: We accommodated the models to train on BIDMC dataset which performed better due to the methodology used to acquire dataset which is more accurate and also more number of datapoints due to large duration of each reading.
- Plotting: We created a custom bar graph plotting function adaptable to any number of losses and models for an efficient inclusion of new models and losses.
- Data Augmentation: We can introduce some noise into our existing signal and add it to our dataset to make the model robust to noise from ambient light and also add more datapoints

COMPARISON OF LOSSES AND MODELS

MTHS DATASET

Loss \model	BASE	FCN	FCT resid-ual	FCT DCT	LSTM
Huber loss	10.47	8.07	6.28	9.61	8.9
MSE	13.56	11.55	10.23	13.12	12.8
MAE	9.51	8.77	6.73	10.91	10.42
Log cosh	9.26	8.15	6.70	9.26	10.01

BIDMC DATASET

Loss \model	BASE	FCN	FCT residual	FCT DCT	LSTM
Huber loss	1.76	2.04	1.08	2.44	1.25
MSE	4.95	3.66	2.96	4.84	3.40
MAE	2.61	2.56	1.62	3.11	1.56
Log cosh	2.30	1.64	1.03	2.05	1.26

THANK YOU

References:

[1] Taha Samavati, Mahdi Farvardin , Aboozar Ghaffari, et al. "*Efficient Deep Learning-based Estimation of the Vital Signs on Smartphones*." arXiv:2204.08989

[2] Images on slide 1, 2, 3 ,4, 21 are from Google.