A PROJECT REPORT
ON

‘Self Supervised Deep Neural Network for Automated Detection of Heart Valve Diseases using Time-Frequency Analysis of Heart Sound Recordings’

BY

Hari Krishna Dhamodaran
2019A4PS1263H

Under the supervision of
Dr Rajesh Kumar Tripathy

Prepared on completion of the
Laboratory Project Course No. EEE F366

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
April 2023

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me with the possibility to complete this report. I would also like to acknowledge with much appreciation the crucial role of the staff of Dr.Rajesh Kumar Tripathy, the faculty mentor who supervises my project, who gave the necessary materials, guidance and various technical insights to proceed with the project "**Self Supervised Deep Neural Network for Automated Detection of Heart Valve Diseases using Time-Frequency Analysis of Heart Sound Recordings**".

I would also like to extend my gratitude to the management of BITS Pilani for providing me with a wonderful opportunity to explore fields of my own interest with much freedom, which would prove to be really valuable for shaping a fruitful career for myself. Last but not the least, I would also like to thank my parents and friends who have continuously supported and motivated me towards pursuing this project.

Birla Institute of Technology and Science-Pilani,

Hyderabad Campus

Certificate

This is to certify that the project report entitled "**Self Supervised Deep Neural Network for Automated Detection of Heart Valve Diseases using Time-Frequency Analysis of Heart Sound Recordings"** submitted by Mr Hari Krishna D(ID No. 2019A4PS1263H) in the completion of the End Semester requirements of the course EEE F366, Lab Project Course, embodies the work done by him under my supervision and guidance.

Date: 25-04-2023

(Dr.Rajesh Kumar Tripathy)

BITS- Pilani, Hyderabad Campus

ABSTRACT

For this implementation of Self Supervised Neural Networks,the inbuilt CIFAR dataset was used and the Model was built on Google Colab.

The **SimSiam** system which was proposed in Exploring Simple Siamese Representation Learning is implemented for this project.

Further in the coming days,to improve the performance and efficiency of the model the Discrete Wavelet Transformation (DWT) will be used on the images and then the 4 modes of each image in both the datasets will be used to train the model with a goal to increase the cosine similarity between them and to decrease the overall loss of the model.

After Testing this model on CIFAR,the model will be implemented on the Heart Sound Dataset and further changes to improve the model's accuracy on this dataset.

Introduction:

What is a Self-Supervised Neural Network?

The Self-Supervised Learning Models obtain the supervisory signals from the data itself and use the underlying structure in the data for training the model. The Technique of Self Supervised Learning involves predicting any unobserved hidden part of the input from any observed or unhidden part of the input.

SSL has various use cases in NLP, Computer Vision etc. In NLP, we can hide a part of the sentence and predict the hidden words. We can also predict past or future frames in a video(hidden data) from the current ones(observed data).

Self-Supervised Neural Networks for Computer Vision:

For Computer Vision Tasks using Self Supervised Learning, we use an Energy based Contrastive Learning approach where we construct pairs of incompatible images 'x' and 'y' and adjust the model parameters so that the corresponding output energy is Large. In most cases, Cosine Similarity is used as the energy method.

Our Model takes two images, x and y, as inputs, and if 'x' and 'y' are slightly distorted versions of the same image, the model is trained to produce low energy output if x and y are completely different from each other. For e.g.:An image of a house and its cropped version will produce a low energy output, while an image of a house and an image of a car will produce high energy output.

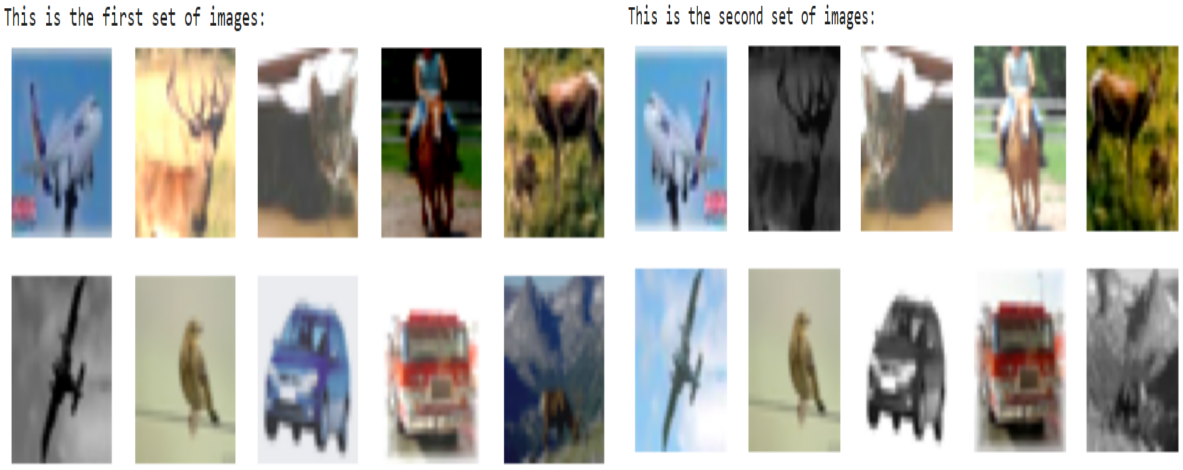
Problem Statement and Architecture used:

For this implementation of Self Supervised Neural Networks, the inbuilt CIFAR dataset was used, and the Model was built on Google Colab

The **SimSiam** system proposed in Exploring Simple Siamese Representation Learning is implemented for this project.

Architecture Brief:

1. First, we Create Two Different versions of the same Dataset with the help of a Stochastic Data Modification and Augmentation Pipeline with random initialization seeds.
2. We First build our **Encoder** by taking the Resnet50 Architecture without the classification head and using it as the **Backbone,** and then we connect a shallow, fully connected layer (**Projection Head)** on top of it.
3. We then build our **Predictor/Decoder,** which is another shallow, fully connected layer having an AutoEncoder-like structure. The output of the encoder model is passed onto the Predictor, which makes the Final Predictions of our Model.
4. We then train our model(encoder) to maximize the negative cosine similarity between the two versions of our dataset.



Note: We can Notice the Differences between the two datasets and can see that they are the same image with only minor differences like cropping and lighting.

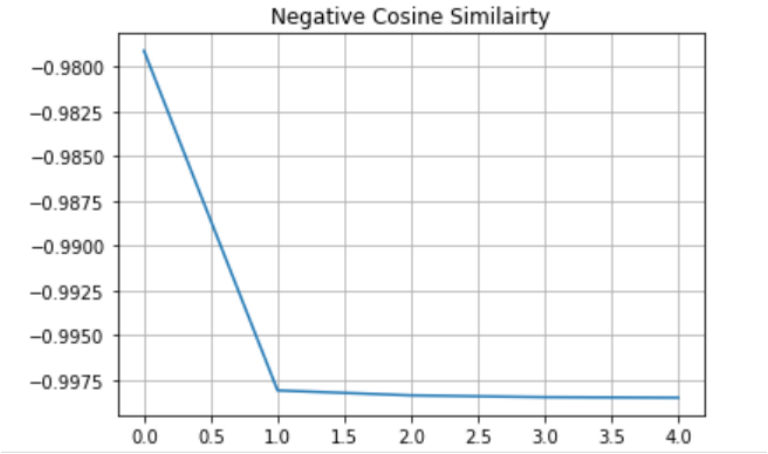
Visualizing the Architecture:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        multiple                  0

 resnet50 (Functional)       (None, 2048)              23587712

 dense (Dense)               (None, 2048)              4196352

 dense_1 (Dense)             (None, 2048)              4194304

 batch_normalization (BatchN  (None, 2048)             8192
 ormalization)

 re_lu (ReLU)                (None, 2048)              0

 dense_2 (Dense)             (None, 2048)              4194304

 batch_normalization_1 (Batc  (None, 2048)             8192
 hNormalization)

 input_4 (InputLayer)        multiple                  0

 dense_3 (Dense)             (None, 512)               1048576

 re_lu_1 (ReLU)              (None, 512)               0

 batch_normalization_2 (Batc  (None, 512)              2048
 hNormalization)

 dense_4 (Dense)             (None, 2048)              1050624

=================================================================
Total params: 38,290,304
Trainable params: 38,227,968
Non-trainable params: 62,336
_________________________________________________________________
```


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        multiple                  0

 resnet50 (Functional)       (None, 2048)              23587712

 dense (Dense)               (None, 2048)              4196352

 dense_1 (Dense)             (None, 2048)              4194304

 batch_normalization (BatchN  (None, 2048)             8192
 ormalization)

 re_lu (ReLU)                (None, 2048)              0

 dense_2 (Dense)             (None, 2048)              4194304

 batch_normalization_1 (Batc  (None, 2048)             8192
 hNormalization)

 input_4 (InputLayer)        multiple                  0

 dense_3 (Dense)             (None, 512)               1048576

 re_lu_1 (ReLU)              (None, 512)               0

 batch_normalization_2 (Batc  (None, 512)              2048
 hNormalization)

 dense_4 (Dense)             (None, 2048)              1050624

=================================================================
Total params: 38,290,304
Trainable params: 38,227,968
Non-trainable params: 62,336
```


The Model was Trained with the two modified versions of the dataset for 5 Epochs and compiled with SGD as the optimiser. The Negative Cosine similarity was Tracked for the 5 Epochs, and the negative similarity steadily decreased with every Epoch, as expected. The Training Results are displayed below:

```
Epoch 1/5
782/782 [==============================] - 96s 101ms/step - loss: -0.9791
Epoch 2/5
782/782 [==============================] - 78s 100ms/step - loss: -0.9981
Epoch 3/5
782/782 [==============================] - 79s 101ms/step - loss: -0.9983
Epoch 4/5
782/782 [==============================] - 79s 101ms/step - loss: -0.9984
Epoch 5/5
782/782 [==============================] - 79s 101ms/step - loss: -0.9985
```



Note: Our Objective is to Maximise the Cosine similarity between the two versions of the Dataset as they are just modified versions of the same image. So we Minimise the Negative Cosine similarity, which is the same as Maximising Cosine Similarity.

Testing and Evaluation Results:

For Evaluating the Model, we build a Linear Classifier built using the frozen features of the trained Backbone Model(Resnet50) and connect it to a Dense Layer with the Softmax activation function to make the final predictions. This Linear Model is then used to evaluate the classifier on unseen images. We Train the Linear Model on the original unmodified training dataset and validate our model by making predictions on the unmodified test dataset and calculating the accuracy and loss.

```
Epoch 1/5
782/782 [==============================] - 17s 18ms/step - loss: 13525.1660 - accuracy: 0.1169 - val_loss: 15346.0430 - val_accuracy: 0.1186
Epoch 2/5
782/782 [==============================] - 14s 18ms/step - loss: 7966.3750 - accuracy: 0.1425 - val_loss: 4788.8965 - val_accuracy: 0.1548
Epoch 3/5
782/782 [==============================] - 14s 17ms/step - loss: 3042.1628 - accuracy: 0.1839 - val_loss: 3461.6433 - val_accuracy: 0.1097
Epoch 4/5
782/782 [==============================] - 13s 17ms/step - loss: 779.3660 - accuracy: 0.2661 - val_loss: 454.8718 - val_accuracy: 0.2908
Epoch 5/5
782/782 [==============================] - 13s 17ms/step - loss: 361.5571 - accuracy: 0.3254 - val_loss: 319.7362 - val_accuracy: 0.3419
157/157 [==============================] - 2s 13ms/step - loss: 319.7362 - accuracy: 0.3419
```


```
Epoch 1/5
782/782 [==============================] - 17s 18ms/step - loss: 13525.1660 - accuracy: 0.1169 - val_loss: 15346.0430 - val_accuracy: 0.1186
Epoch 2/5
782/782 [==============================] - 14s 18ms/step - loss: 7966.3750 - accuracy: 0.1425 - val_loss: 4788.8965 - val_accuracy: 0.1548
Epoch 3/5
782/782 [==============================] - 14s 17ms/step - loss: 3042.1628 - accuracy: 0.1839 - val_loss: 3461.6433 - val_accuracy: 0.1097
Epoch 4/5
782/782 [==============================] - 13s 17ms/step - loss: 779.3660 - accuracy: 0.2661 - val_loss: 454.8718 - val_accuracy: 0.2908
Epoch 5/5
782/782 [==============================] - 13s 17ms/step - loss: 361.5571 - accuracy: 0.3254 - val_loss: 319.7362 - val_accuracy: 0.3419
157/157 [==============================] - 2s 13ms/step - loss: 319.7362 - accuracy: 0.3419
```


```
print("Test accuracy: {:.2f}%".format(test_acc[1] * 100))
```

```
Test accuracy: 34.19%
```

Post Mid Sem Improvisations:

To improve the performance of the SSL Model,the discrete wavelet transform was applied to the images.DWT layer was added to the Neural network and the model was trained and tested on the Dataset.Even Though DWT was successful in increasing the accuracy of the model to over 40%,the accuracy was still low and the performance had to be improved.Following are the model architecture and results of the DWT based Model:

```
model = tf.keras.Sequential(
       [
         # Input and backbone.
         tf.keras.layers.Input((CROP_TO_DWT, CROP_TO_DWT, 3)),
         tf.keras.layers.InputLayer(input_shape = (WIDTH_DWT,HEIGHT_DWT,3)),
         tf.keras.layers.Conv2D(1, 3,activation='relu',padding="same",input_shape=input_shape_gs[1:]),
         DWT.DWT(name="haar",concat=0),
         #tf.keras.layers.Conv2D(4, 1024),
         tf.keras.layers.Conv2D(3, 4,activation='relu',padding="same",input_shape=input_shape[1:]),
         tf.keras.applications.ResNet50(input_shape=(WIDTH, HEIGHT , 3),weights=None, include_top=False, pooling='avg'),
         tf.keras.layers.Dense(PROJECT_DIM),

         # Projection head.
         tf.keras.layers.Dense(PROJECT_DIM,use_bias=False,kernel_regularizer=regularizers.l2(WEIGHT_DECAY)),
         tf.keras.layers.BatchNormalization(),
         tf.keras.layers.ReLU(),
         tf.keras.layers.Dense(PROJECT_DIM,use_bias=False,kernel_regularizer=regularizers.l2(WEIGHT_DECAY)),
         tf.keras.layers.BatchNormalization()
       ]
```

```
Epoch 1/6
782/782 [==============================] - 195s 246ms/step - loss: 640.0529 - accuracy: 0.1820 - val_loss: 137.1362 - val_accuracy: 0.2648
Epoch 2/6
782/782 [==============================] - 176s 225ms/step - loss: 69.6294 - accuracy: 0.3448 - val_loss: 36.5943 - val_accuracy: 0.3993
Epoch 3/6
782/782 [==============================] - 160s 205ms/step - loss: 36.2232 - accuracy: 0.3977 - val_loss: 36.5821 - val_accuracy: 0.3994
Epoch 4/6
782/782 [==============================] - 160s 205ms/step - loss: 36.0701 - accuracy: 0.4022 - val_loss: 36.5821 - val_accuracy: 0.3994
Epoch 5/6
782/782 [==============================] - 160s 205ms/step - loss: 36.2313 - accuracy: 0.3976 - val_loss: 36.5821 - val_accuracy: 0.3994
Epoch 6/6
782/782 [==============================] - 177s 226ms/step - loss: 36.2420 - accuracy: 0.3985 - val_loss: 36.5821 - val_accuracy: 0.3994
157/157 [==============================] - 27s 171ms/step - loss: 36.5821 - accuracy: 0.3994
```

As the next step the same Self Supervised Neural Network Model was applied on the Heart Diseases Dataset.The Dataset consisted of .wav files(sound signals) which were converted into images using librosa library and were used to train and evaluate the SSL Model.

These were the results:

```
Epoch 1/15
13/13 [==============================] - 17s 823ms/step - loss: 916.3933 - accuracy: 0.1875 - val_loss: 748.3208 - val_accuracy: 0.1950
Epoch 2/15
13/13 [==============================] - 4s 296ms/step - loss: 1117.8206 - accuracy: 0.1925 - val_loss: 1337.4128 - val_accuracy: 0.1750
Epoch 3/15
13/13 [==============================] - 5s 422ms/step - loss: 1261.4952 - accuracy: 0.2150 - val_loss: 954.7783 - val_accuracy: 0.2250
Epoch 4/15
13/13 [==============================] - 5s 385ms/step - loss: 853.2183 - accuracy: 0.1950 - val_loss: 977.4105 - val_accuracy: 0.1950
Epoch 5/15
13/13 [==============================] - 4s 347ms/step - loss: 697.2146 - accuracy: 0.2050 - val_loss: 742.0377 - val_accuracy: 0.1750
Epoch 6/15
13/13 [==============================] - 6s 511ms/step - loss: 722.1813 - accuracy: 0.2212 - val_loss: 906.8786 - val_accuracy: 0.2000
Epoch 7/15
13/13 [==============================] - 4s 294ms/step - loss: 860.6091 - accuracy: 0.2000 - val_loss: 979.4104 - val_accuracy: 0.2000
Epoch 8/15
13/13 [==============================] - 6s 503ms/step - loss: 902.3329 - accuracy: 0.2000 - val_loss: 997.8470 - val_accuracy: 0.2000
Epoch 9/15
13/13 [==============================] - 4s 295ms/step - loss: 913.8474 - accuracy: 0.2000 - val_loss: 1002.5334 - val_accuracy: 0.2000
Epoch 10/15
13/13 [==============================] - 4s 351ms/step - loss: 921.0363 - accuracy: 0.2000 - val_loss: 1003.7246 - val_accuracy: 0.2000
4/4 [==============================] - 1s 305ms/step - loss: 742.0377 - accuracy: 0.1750
```

As we can see the model had very less validation accuracy and accuracy and high validation loss and loss.

To improve the performance on this Dataset another unsupervised clustering based neural network model was implemented and were trained and tested using the heart diseases dataset.However the unsupervised neural network model also yielded very low accuracy and high loss.Following are the model architecture and results of the Unsupervised Neural Network Model:

```
inp = Input(shape=(Height, Width, 3))
e = Conv2D(4, (3, 3), activation='tanh', padding='same')(inp)
e = MaxPooling2D((2, 2), padding='same')(e)
e = Conv2D(2, (3, 3), activation='tanh', padding='same')(e)
e = MaxPooling2D((2, 2), padding='same')(e)
e = Conv2D(1, (3, 3), activation='tanh', padding='same')(e)
e = MaxPooling2D((2, 2), padding='same')(e)

shape_before_flattening = K.int_shape(e)
encoded = Flatten()(e)
d = Reshape(shape_before_flattening[1:])(encoded)

d = Conv2D(1, (3, 3), activation='tanh', padding='same')(d)
d = UpSampling2D((2, 2))(d)
d = Conv2D(2, (3, 3), activation='tanh', padding='same')(d)
d = UpSampling2D((2, 2))(d)
d = Conv2D(4, (3, 3), activation='tanh', padding='same')(d)
d = UpSampling2D((2, 2))(d)
decoded = Conv2D(1, (3, 3), padding='same')(d)

autoencoder = Model(inputs=inp, outputs=decoded, name='autoencoder')
```

```
8/8 [==============================] - 1s 131ms/step - loss: 2221.9956
Epoch 989/1000
8/8 [==============================] - 1s 127ms/step - loss: 2196.9814
Epoch 990/1000
8/8 [==============================] - 1s 129ms/step - loss: 2189.5269
Epoch 991/1000
8/8 [==============================] - 1s 128ms/step - loss: 2203.7219
Epoch 992/1000
8/8 [==============================] - 1s 132ms/step - loss: 2212.7817
Epoch 993/1000
8/8 [==============================] - 1s 153ms/step - loss: 2208.4839
Epoch 994/1000
8/8 [==============================] - 2s 220ms/step - loss: 2204.8545
Epoch 995/1000
8/8 [==============================] - 2s 227ms/step - loss: 2221.4177
Epoch 996/1000
8/8 [==============================] - 2s 186ms/step - loss: 2213.0950
Epoch 997/1000
8/8 [==============================] - 1s 130ms/step - loss: 2241.9719
Epoch 998/1000
8/8 [==============================] - 1s 130ms/step - loss: 2199.3008
Epoch 999/1000
8/8 [==============================] - 1s 131ms/step - loss: 2219.6045
Epoch 1000/1000
8/8 [==============================] - 1s 129ms/step - loss: 2218.3560
<keras.callbacks.History at 0x7efaad558130>
```

We can see that even after 1000 iterations the loss was very high.

As the next step,a simple Supervised Learning Based CNN Neural Network was built as the main model.Three Different Transforms namely STFT,CWT and Chirplet Transformations were applied on the ecg Image Dataset and the same base CNN Model was trained and evaluated on these three different datasets(STFT Dataset,CWT Dataset and Chirplet Transform Dataset).Then Model Quantization was carried out for the different versions of the model trained on the three different datasets in order to choose the best version of the model with best quantization for deploying in a mobile-app(on an android device).Floating Point 32,Floating Point 16 and Integer 8 were the three different quantizations tested and each of the model's versions model' performance was evaluated for the three different quantizations.Following are the results:

Transformation_Name:	FP32 Quantization Accuracy	FP16 Quantization Accuracy	INT8 Quantization Accuracy
STFT	90.50%	90%	88%
CWT	42%	40%	38.50%
Chirplet	91.50%	91.50%	90%

From the results we can say that the Chirplet Model with FP16 had the best combination of highest accuracy and less size.So this model was finalized and deployed in an android app.The app is connected to the google cloud where the images(with chirplet transformation applied) are present and the app when given an image from the cloud predicts the heart disease associated with the image or if the image corresponds to a healthy heart.

Conclusion:

We can conclude that the Self Supervised Learning method was unsuccessful in giving a good performance on the ECG Dataset.The Unsupervised Neural Network Model was also bad at classifying the ECG images to their corresponding labels.Finally the Supervised Learning model was implemented on the Dataset and as expected it yielded very good results and performed considerably well in classifying the images compared to the Self-Supervised and Unsupervised Models.

References:

Colab Links :
https://colab.research.google.com/drive/1bYtTYL95A7ZTrbSRgAPPnn1oBqGaamaZ#scrollTo=D3Be1ZTgUAIs

https://colab.research.google.com/drive/1uCGfPQLpyrYFQcH6x9OBXn5m0NPv7VeL#scrollTo=WhYv9wgor54O

https://colab.research.google.com/drive/1c0LsW8ISmWtObtet5_aZYsvhiDXzD2yF
Architecture Idea taken from the "*Exploring Simple Siamese Representation Learning*" Paper.
Paper Link: https://arxiv.org/abs/2011.10566