

**RECONSTRUCTING PERCEIVED IMAGES FROM HUMAN BRAIN ACTIVITIES USING TWIN DEEP NEURAL NETWORK**

## A PROJECT REPORT

***Submitted by***

# PIYUSH CHOUHAN M [REGISTER NO: 211417104183] KEDAR R [REGISTER NO: 211417104117]

# MOHANAVEL R [REGISTER NO: 211417104149]

***in partial fulfillment for the award of the degree of***

# BACHELOR OF ENGINEERING

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.**

# ANNA UNIVERSITY: CHENNAI 600 025

**APRIL 2021**

# BONAFIDE CERTIFICATE

Certified that this project report **“ RECONSTRUCTING PERCEIVED IMAGES FROM HUMAN BRAIN ACTIVITIES USING TWIN DEEP NEURAL NETWORKS”** is the bonafide work of “

**PIYUSH CHOUHAN M(211417104183), MOHANAVELR(211417104149),**

**KEDAR R(211417104117).**

**”** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Dr.S.MURUGAVALLI,M.E.,Ph.D., DR.N.PUGHAZENDI**

**HEAD OF THE DEPARTMENT SUPERVISOR**

**ASSOCIATE PROFESSOR**

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI, NAZARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on...........................

## INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to express our heartfelt and sincere thanks to our Directors

## Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHIKUMAR, M.E., Ph.D., and

**Tmt. SARANYASREE SAKTHIKUMAR B.E., M.B.A.,** for providing us with the necessary facilities for the completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of the CSE Department, **Dr. S.MURUGAVALLI, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide Dr.N.PUGHAZENDI** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

## NAME OF THE STUDENT

## PIYUSH CHOUHAN M

## MOHANAVEL R

## KEDAR R

**ABSTRACT**

Neural Decoding plays an important role in understanding Human Visual System. The human visual system is naturally capable of extracting features from any object and comparing them. Most of the papers focus on either the brain activity pattern classiﬁcation or visual stimuli identiﬁcation. In this project, we introduce the Twin Deep Neural Network model for accurate reconstruction of images from human brain activity using Functional Magnetic Resonance Imaging (fMRI). TDNN method can be used for comparing the relationship between a sample pair of similar features for better visual reconstruction and make use of each sample completely. High dimensionality and a small quantity of FMRI data impose restrictions is reduced by using the TDNN approach. Essentially, this manner can increase the training data from N samples to 2N sample pairs, which takes full advantage of the limited quantity of training samples. We evaluated the proposed TDNN method on the open dataset of handwritten digital images and character datasets and exceeded about 10% of the accuracy of all existing state-of-the-art methods on the Convolutional Neural Network (CNN).

# TABLE OF CONTENTS

### CHAPTER NO. TITLE PAGE NO. ABSTRACT iv

### LIST OF TABLES vii

### LIST OF FIGURES viii

### LIST OF SYMBOLS, ABBREVIATIONS ix

### INTRODUCTION

* 1. Overview 1
  2. Problem Definition 2

### LITERATURE SURVEY 2

1. **SYSTEM ANALYSIS**
   1. Existing System 5
   2. Proposed system 6
   3. Requirement Analysis and Specification
      1. Input Requirements 6
      2. Output Requirements 7
      3. Functional Requirements 7
   4. Technology Stack 8

### CHAPTER NO. TITLE PAGE NO.

1. **SYSTEM DESIGN**

4.1. Data dictionary 9

* 1. UML diagrams 11

### SYSTEM ARCHITECTURE

* 1. Architecture Overview 13
  2. Module Design Specification 15
  3. Program Design Language 23

### SYSTEM IMPLEMENTATION

* 1. Client-side coding 24
  2. Server-side coding 30

### SYSTEM TESTING

* 1. Unit Testing 43
  2. Integration Testing 45
  3. Test Cases & Reports / Performance Analysis 47

### CONCLUSION

* 1. Conclusion and Future Enhancements 49

### APPENDICES

* 1. Sample Screens 50
  2. Publications 53

**REFERENCES 53**

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TABLE NAME** | **PAGE NO.** |
| 1 | DATA DICTIONARY | 10 |
| 2 | TESTING-HOMEPAGE | 43 |
| 3 | TESTING-TRAINING SCREEN | 44 |
| 4 | TESTING-RESOURCES | 44 |
| 5 | TESTING-RESULT SCREEN | 45 |
| 6 | INTEGRATION TESTING | 46 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **FIGURE NAME** | **PAGE NO.** |
| 1 | TECHNOLOGY STACK | 8 |
| 2 | CLASS DIAGRAM | 11 |
| 3 | ACTIVITY DIAGRAM | 12 |
| 4 | DEPLOYMENT DIAGRAM | 13 |
| 5 | ARCHITECTURE OVERVIEW | 13 |
| 6 | INTEGRATION ARCHITECTURE | 45 |
| 7 | SCREEN CAPTURE 1 | 47 |
| 8 | SCREEN CAPTURE 2 | 47 |
| 9 | SCREEN CAPTURE 3 | 48 |
| 10 | SCREEN CAPTURE 4 | 48 |
| 11 | SCREEN CAPTURE 5 | 49 |
| 12 | SAMPLE SCREEN 1 | 50 |
| 13 | SAMPLE SCREEN 2 | 50 |
| 14 | SAMPLE SCREEN 3 | 51 |
| 15 | SAMPLE SCREEN 4 | 51 |
| 16 | SAMPLE SCREEN 5 | 52 |
| 17 | SAMPLE SCREEN 6 | 52 |
| 18 | SAMPLE SCREEN 7 | 53 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **ABBREVIATION** | **DESCRIPTION** |
| VAE | Variational Auto Encoder |
| GMM | Gaussian Mixture Model |
| FMRI | Functional Magnetic Resonance Imaging |
| BCCA | Bayesian Canonical Correlation Analysis |
| DE-CNN | Deconvolution Neural Network |
| DGMM | Deep Generative Multiview Model |
| TDNN | Twin Deep Neural Network |

**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

Understanding the visual processing of the human brain is a critical and challenging problem in old days. Revolution of mathematical and probabilistic models in the computer field are applied to many sorts of applications including detection, computer vision, classification problems. In the first, processing of the human brain is implemented by linear models and later the development of perceptron is more useful and has the natural properties of the human brain and improved the model performance but a complex structure of visual stimuli and high dimension data is challenging issues although the feature sample for many sorts of application is different and is simplified after the emerging of convolution neural network is a more upgradeable version of deep neural network and also it has good feature extraction in the model help to improve accuracy and overcome the problem of high dimensional data.

Deep Neural networks had been used in many fields and solving the problem of high-dimensional data into low dimensions and capturing the important features. Emerging deep learning and generative model improvise the image processing and decode the high dimension data. Autoencoders is a basic generative model to capture important features without conflicting the originality of data and can able to derive the high dimension data using a neural network. This leads to the idea of neural encoding and decoding are possible in a single model. Neural encoding means to forecast the voxel activation from the visual stimuli and vice versa is named neural decoding. Human brain processing categorizes as voxel activation classification and prediction of visual stimuli. A combination of these decoding and encoding is achieved to reconstruct the visual stimuli is possible.

* 1. **PROBLEM DEFENITION**

Reconstruction of Recognized Image from human brain activity for understanding the human visual system and Implement the variational autoencoder by convolutional neural network for visual representation then Gaussian mixture model is used to cluster the same variants of class in latent space vector. The generative model is to reconstruct the visual images and classify the reconstructed image from the original images by image classification algorithms.

**2. LITERATURE SURVEY**

Bayesian Deep Multi-View Learning [1] used both neural encoding as well as decoding to reconstruct the visual stimuli correspond to the voxel activation which is proposed through different stages of visual stimuli categorization. It uses a convolution neural network with the major properties variational autoencoder for the image compression of visual stimuli and voxel activation was computed by fMRI. For reconstruction of visual images is generated by using a deep neural network and reconstruction of voxel activation is generated by the linear model for greater improvement but there is a major problem in the dataset with high dimension feature and also complex data with more noise capture during the preparation of dataset affect the model and accuracy.

Unsupervised Domain Adaptation [2] Decoding Brain States from fMRI is decoding the voxel active areas in the brain and the framework is used for capture the high dimension feature in spatiotemporal data. To capture different distribution spaces with the most efficient classification method is used as a meanest discrepancy. The difference capture from the source and predictive target distributions is correlated and analyzed using the domain adaptation method. The decoding of the voxel activation is trained using unsupervised learning due to the limited of datasets and also a large amount of data is unlabeled so basic fundamental linear models take more time than unsupervised learning.

Efficient Perceptual Coding [3] converting the original signal to elements of new forms that represent a coding rate are independent and the reconstructed image of visually seen is effectively generated by minimizing the error loss in model from the latent quantize of elements in a new form is statistically and perceptually not dependent on each other. The basic fundamental linear models for this approach are not effective so that the proposed adaptive nonlinear method analysis the new elements and their weights to find the likelihood of generated neighborhood. The developed effective coding method is good to reconstruct the images with high quality of input signal with their original images.

Neural Decoding with Hierarchical Generative Models [4] learns and extracts features based on the combination of different models so that model is highly trained to learn the all hierarchy important data by using different generative models to reconstruct the visual stimuli done by Boltzmann machines with the upgraded version of the conditionally restricted machine. In first, extract the important feature of data from voxel active areas done by unsupervised phase because of the large amount of unlabeled data is available. The extracted data feature is then trained using hierarchical generative models to get the reconstructed image of visual stimuli.

Locally Adaptive Perceptual Image Coding [5] capture the difference of original image data with their important feature of components. The coder method is for the dimension reducing technique of the original image as well as input signal and they used reconstructed image quality on sub-band transform coefficient. The proposed coder method does not use more information of image so this method reduces the computation of more redundant statistics data.

Reconstructing Realistic Image [6] with the generative models by mapping the voxel activation by the CNN to capture more important features and then combining the GAN to reconstruct the image. An important feature is easily captured by CNN to get a lower set of values store in latent space from this GAN is used to reconstruct the image and evaluated the quality and computation of the model.

The high-Dynamic-Range-Based [7] method is proposed for the display of the original image in that criteria there are two forms of media are highly dynamic and low dynamic of details for image with the novel of adding additional edge capture filter is used to reconstruct the original image with non-volatile of image detail.

Decoding Behavior Task [8] is to get the visual data of voxel activation of the brain. The noninvasive detection method is mainly focused on correlating the voxel data to the visual data. The proposed cross-subject decoding uses existing trained models on ImageNet to get the feature from the voxel activation.

Human Vision Reconstruction [9] understands the brain activity with all criteria occur in occipital lobe active areas are to be capture without the noise. Analysis of brain for single data is not exactly well served so different situation of visual stimuli is seen by the person at the same time to record the voxel activation of the entire brain. Mapping the voxel to the corresponding visual is a challenge and to predict the model for a specific set of voxel corresponding mapping space is also potentially need for the model. A novel mapping algorithm is proposed to mapping the brain voxel activation areas to visual stimuli like real faces or handwritten digits with fMRI. The reconstruction is entirely lying on the mapping algorithm that captures the voxel activation pair to visual stimuli.

Tensor Neural Network [10] was used for better analysis of high dimension as well as complex categorical voxel activation on the entire brain. The proposed framework in this paper is to capture all voxel activation data across the entire brain and remove the problem of higher dimension so that the captured feature is quite good with essential features. Analysis of this framework to existing work came up with the result of high accuracy and superior model to capture voxel data with complex structure and high dimension is no problem with the proposed framework and is implemented by a neural network with a high number of nodes in each layer to get compress the feature having more weight parameters in the neural network.

1. **SYSTEM ANALYSIS**
   1. **EXISTING SYSTEM**

The existing system consists of a brain decoding model using the state of the art, EEG method. This system has various faults in its own way. The one major problem with this system is that it couldn't capture the neuron interactions at the micro-level. The system is using EEG signals, which are just the general electrical impulse signals from various probes. This procedure creates a limitation of identifying the overall brain activity. These systems can map an extent to 40 probe signals at max, which in turn fed to a neural network yields a very under the fitted model. The complex nature of the working of the human brain is still a mystery and it is highly difficult to capture a snapshot of the brain activity for a given moment, as billions of neuron's firings happen even in a simple activity. And, finding the pattern among the communication of these neurons is still an unsolvable mystery.

* 1. **PROPOSED SYSTEM**

Visual image and FMRI brain activities are used as input to the inference model and implemented by Variation Auto Encoder to get the similarity of both images using a neural network. Digits are multi-model distribution multiple peaks. Multiple Gaussian in different proportions simultaneously to approximate a multi-modal distribution. GMM is a latent variable model and trained using the Expectation-Maximization algorithm to find a closed multi-model distribution. GMM finds the number of finite clusters. Latent space is a smaller dimension representation of brain activity and smooth transformation of images from one another. The decoder model is to Reconstruct Images from latent space using posterior probability or prior probability. Transform latent space representation to higher dimension images using the generative model for the reconstruction of images.

* 1. **REQUIREMENT ANALYSIS AND SPECIFICATION**
     1. **INPUT REQUIREMENTS**

The proposed system is expected to receive two different types of datasets such as visual stimuli and brain mappings. The dataset creation involves the use of functional magnetic resonance imaging equipment and a head-mountable image visualizer device. The subject is laid on an FMRI machine with a head-mounted device and is projected to different visual image frames. The FMRI machine is then activated and a magnetic resonance is passed to the human subject, the respective cross-section views of the brains are then plotted and then integrated as a 3D brain voxel of 3092 segments. Now, this 3D brain depicts two brain behaviors such as active and dormant. These active regions represent the flow of blood on a particular part of the cerebrum. The 3D brain is then processed into a 2D flat map surface which is further streamlined into a 1D plane. This conversion process from 3D into 1D helps us to capture the majority of the brain activities. These brain activities are then treated as one of the types of input datasets. During this process, the visual frames subjected to the human are treated as another type of dataset.

* + 1. **OUTPUT REQUIREMENTS**

The proposed system is expected to reconstruct images with the help of visual stimuli and brain mappings. The subject is laid on an FMRI machine with a head-mounted device and is projected to different visual image frames. The FMRI machine is then activated and a magnetic resonance is passed to the human subject, the respective cross-section views of the brains are then plotted and then integrated as a 3D brain voxel of 3092 segments. Now, this 3D brain depicts two brain behaviors such as active and dormant. These active regions represent the flow of blood on a particular part of the cerebrum. The 3D brain is then processed into a 2D flat map surface which is further streamlined into a 1D plane. This conversion process from 3D into 1D helps us to capture the majority of the brain activities. These brain activities are then treated as one of the types of input datasets.

* + 1. **FUNCTIONAL REQUIREMENTS**

The system is expected to receive two different types of datasets as inputs namely – visual images, brain mappings. With the perception of datasets fed to an inference model, it is expected to derive an encoding schema from the input visual stimuli dataset. The inference gained from the encoding model is subjected to shared latent space. This shared latent space is a common knowledge base where the inferences are being updated regularly. The knowledge-based system is further utilized for the generative multi-view model. This generative model is capable of reconstructing images from the shared knowledge-based and the input visual stimuli dataset. To further evaluate the quality of images reconstructed, it is passed to different classification algorithms to determine the accuracy in the classification of different labels. This can be achieved by bypassing the traditional MNIST dataset and the reconstructed image dataset into the classifier algorithm.

* 1. **TECHNOLOGY STACK**

The major tools that have been used in our project are

* ANACONDA
* ANGULAR 10
* TENSORFLOW
* KERAS
* FLASK
* BOOTSTRAP
* NODE

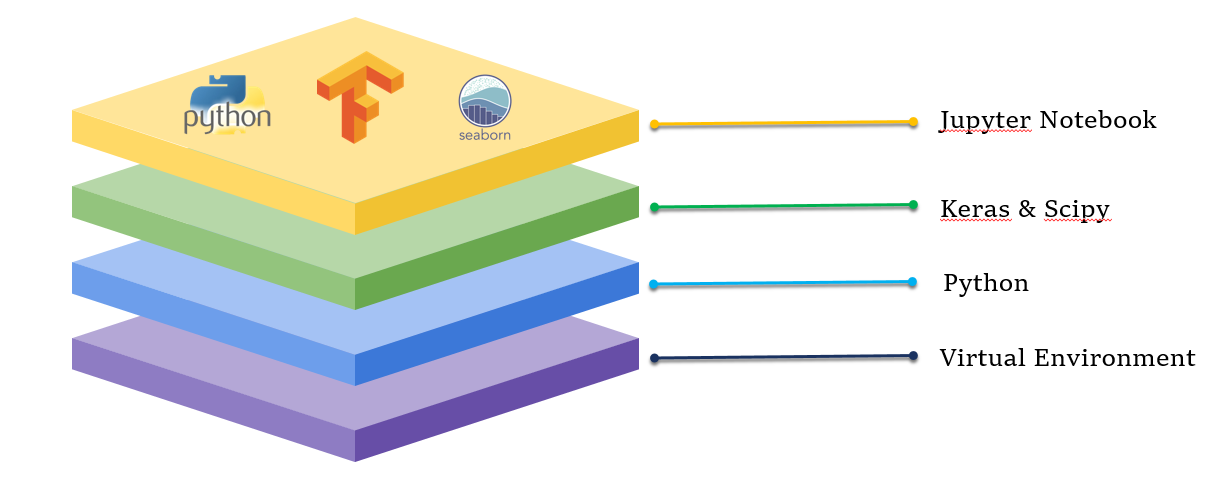


Figure 1 - TECHNOLOGY STACK

Anaconda Navigator is the latest python repository and also it has been used to initiate applications with a major ide provide environment. It is free software for all users with python and r programming languages that are emerging in mathematical models and computer vision. It has been provided with the user interface as well as a command-line that is greatly helpful to users. It has integrated with the pre-built library packages and channels for programming languages.

Spyder is a cross platform python programming ide for mathematical models and computer vision. It is free software for all users integrates with basic package of python and also includes mathematical models packages like numpy, pandas, etc. It is best ide for project work and included in anaconda navigator for easy to install packages at once.

1. **SYSTEM DESIGN**

**4.1. DATA DICTIONARY**

|  |  |  |
| --- | --- | --- |
| **Data** | **Type** | **Size** |
| **FRONT-END** | | |
| Home | Root directory | 10kb |
| Trainer | Root bridge | 12kb |
| Resources | Artifact directory | 10kb |
| Chartlist | Training summary list | 13kb |
| Charts | Evaluation graphs | 15kb |
| Footer | Root bridge | 12kb |
| **BACK-END** | | |
| Main.py | REST API | 5kb |
| Driver.py | Driver | 3kb |
| Inference\_model.py | Globalizer | 4kb |
| Shared\_latent\_space.py | Shared space | 5kb |
| Decoder\_train.py | Decoder | 4kb |
| Dual\_learning.py | Space updater | 6kb |
| Training.py | Trainer | 5kb |
| Testing.py | Testing | 5kb |
| Classification.py | Classifier | 7kb |
| Data | Data directory | 50MB |
| Plots | Plot directory | 35MB |

Table 1 - DATA DICTIONARY

**4.2. UML DIAGRAMS**

**CLASS DIAGRAM**

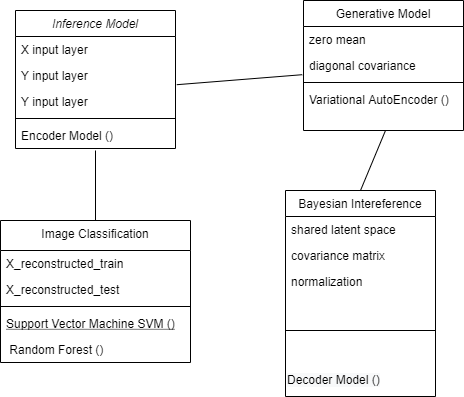
****

Figure 2 - CLASS DIAGRAM

**ACTIVITY DIAGRAM**

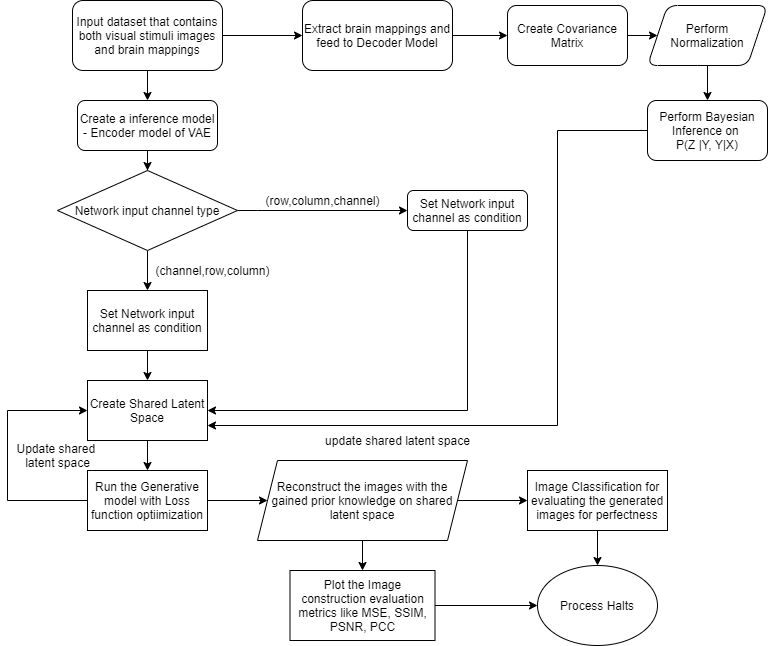
****

Figure 3 - ACTIVITY DIAGRAM

**DEPLOYMENT DIAGRAM**

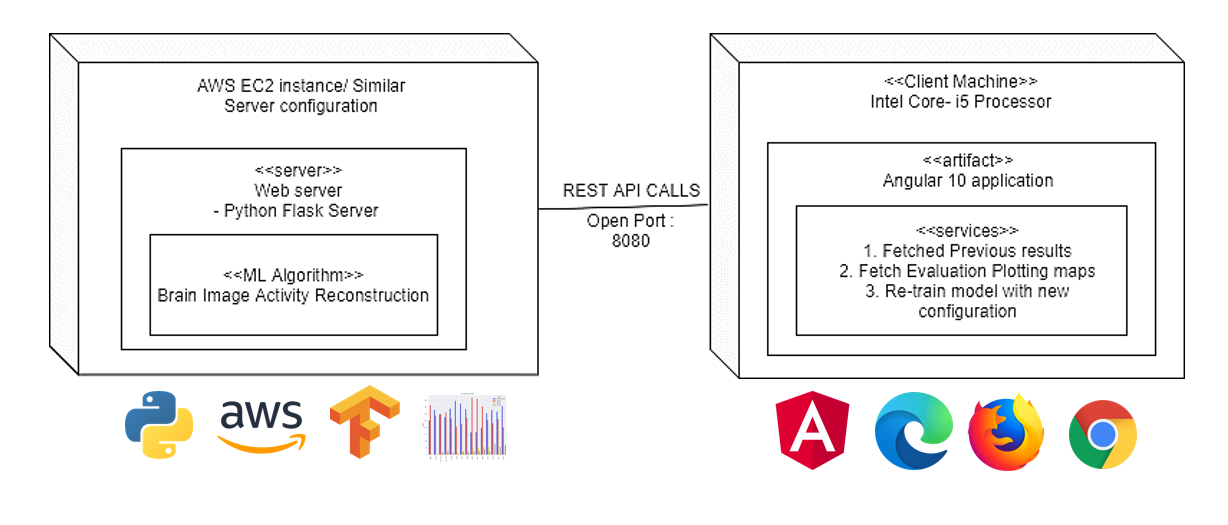


Figure 4 - DEPLOYMENT DIAGRAM

1. **SYSTEM ARCHITECTURE**

**5.1. ARCHITECTURE OVERVIEW**

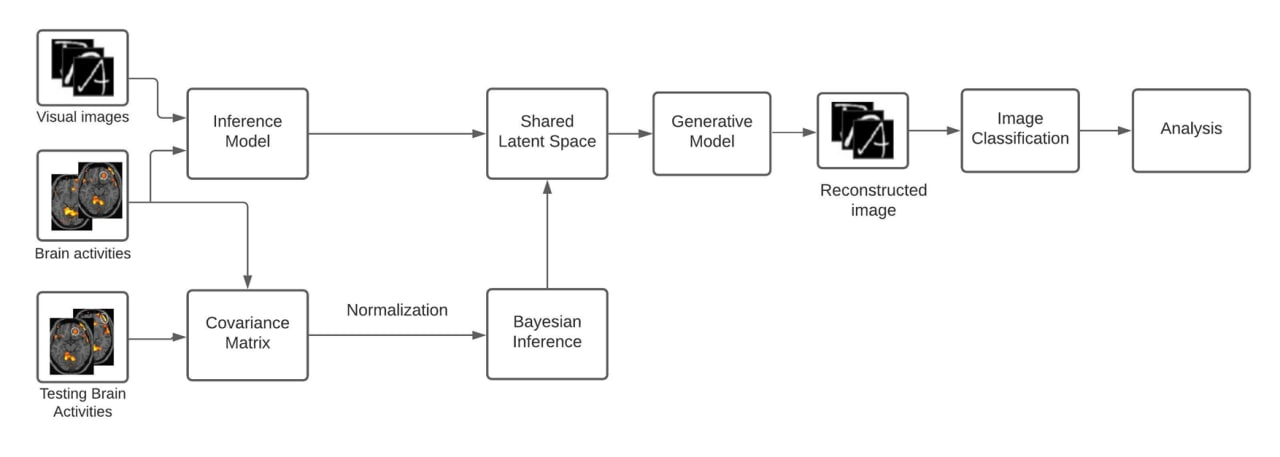


Figure 5 – ARCHITECTURE OVERVIEW

The proposed model consists of three modules. The first stage involves the gathering of input data such as visual handwritten digits, FMRI brain activity from the brain linear repository and fetching the gathered data to the inference model and GMM is used to cluster. The second stage involves the shared latent variables used to represent the low dimensional images and get input from the inference model. The final stage involves the reconstruction of visual image using generative model. Figure 4 picturizes the overall proposed architecture.

**NEURAL NETWORK ARCHITECTURE**

Model: "TDNN"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

======================================================================

input\_1 (InputLayer) (None, 1, 28, 28) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Firstlayer (Conv2D) (None, 1, 28, 1) 113 input\_1[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Secondlayer (Conv2D) (None, 1, 14, 64) 320 Firstlayer[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Thirdlayer (Conv2D) (None, 1, 14, 64) 36928 Secondlayer[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fourthlayer (Conv2D) (None, 1, 14, 64) 36928 Thirdlayer[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 896) 0 Fourthlayer[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

hiddenlayer (Dense) (None, 128) 114816 flatten\_1[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Z\_1 (Dense) (None, 6) 774 hiddenlayer[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Z\_2 (Dense) (None, 6) 774 hiddenlayer[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lambda\_1 (Lambda) (None, 6) 0 Z\_1[0][0]

Z\_2[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 128) 896 lambda\_1[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 12544) 1618176 dense\_1[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

reshape\_1 (Reshape) (None, 14, 14, 64) 0 dense\_2[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_1 (Conv2DTrans (None, 14, 14, 64) 36928 reshape\_1[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_2 (Conv2DTrans (None, 14, 14, 64) 36928 conv2d\_transpose\_1[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_3 (Conv2DTrans (None, 29, 29, 64) 36928 conv2d\_transpose\_2[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_1 (Conv2D) (None, 28, 28, 1) 257 conv2d\_transpose\_3[0][0]

======================================================================

Total params: 1,920,766

Trainable params: 1,920,766

Non-trainable params: 0

**5.2. MODULE DESIGN SPECIFICATION**

**5.2.1. LINEAR SPARSE DECODING MODEL**

Bayesian Deep Multi View Learning used both neural encoding as well as decoding to reconstruct the visual stimuli correspond to the voxel activation which proposed through different stages of visual stimuli categorization. It uses convolution neural network with the major properties of variational auto encoder for the image compression of visual stimuli and voxel activation was computed by FMRI. For reconstruction of visual images is generated by using deep neural network and reconstruction of voxel activation is generated by linear model for greater improvement but there is major problem in dataset with high dimension feature and also complex data with more noise capture during the preparation of dataset affect the model and accuracy.

Sparsity-Constrained fMRI decoding Natural video provides dynamic brain response process information of day to day life scenario is recorded by FMRI is helpful in brain encoding and decoding field of study. In this paper, a novel decoding model with sparsity based approach that insist bottom up method first for decoding the natural real life video scene and to analysis the visual saliency decoding. Visual saliency model is used from graph based algorithm to get curve information of priori saliency is calculated by current video frame integrate with activation map. Specific computational model is adopted and combine with the linear sparse model learn from atomic dictionary to represent the brain activity pattern signal with correspondence to visual video streams. This sparsity technique is efficient one to decode video and dictionaries to learn and reconstruct the FMRI signals by sparse based model to get brain activity pattern representation.

**5.2.2 RECONSTRUCTION BY GENERATIVE ADVERSARIAL NETWORK**

Reconstructing Realistic Image with the generative models by mapping the voxel activation by the CNN to capture more important feature and then combining the GAN to reconstruct the image. Important feature is easily capture by CNN to get lower set of values store in latent space from this GAN is used to reconstruct the image and evaluated the quality and computation of model.

High-Dynamic-Range-Based method is proposed for the display the original image in that criteria there is two form of media are high dynamic and low dynamic of details for image with novel of adding additional edge capture filter is used to reconstructed the original image with non-volatile of image detail.

Quality Prediction Realistic images generation and quality of image is important. In this paper, a new realistic natural image quality predictor for reconstructing the images. GAN is combined with boosting system that run parallel to analysis the structural information. The GAN was devised into two groups of features representing structural and statistical information and generative image database comprises reference images that span a wide range of natural scene characteristics. The result of GAN with boosting create accurate image with extreme high quality prediction.

Restoring Latent Vectors In this paper, pre-trained GAN is adopted with global optimization technique or Genetic Algorithm is calculates the candidate solution for selection and restore the values of latent vector from the pre-trained GAN and obtain the parameter performance and functions used to fit the model. The Gradient descent along with the Mean Squared Error showed better performance.

**5.2.3 UNSUPERVISED DECODING MODEL**

Unsupervised Domain Adaptation Decoding Brain States from FMRI is a decoding the voxel active areas in brain and the framework is used for capture the high dimension feature in spatiotemporal data. For capture different distribution space with the most efficiency classification method is used as most mean discrepancy. The difference capture from the original source and predictive target distributions is correlated and analysis using the domain adaptation method. The decoding the voxel activation is trained using the unsupervised learning due to limited of datasets and also large amount of data is unlabeled so basic fundamental linear models take more time than unsupervised learning.

Locally Adaptive Perceptual Image Coding capture the different of original image data with their important feature of components. The coder method is for the dimension reducing technique of the original image as well as input signal and they used reconstructed image quality on sub band transform coefficient. The proposed coder method does not used more information of image so this method reduce the computation of more number of redundant statistics data.

**5.2.4 LINEAR GAUSSIAN MODEL**

Linear Gaussian Framework for Decoding of Perceived Images Visualizing the mind of someone is improved after the mathematical model of neural network and decode the thoughts is being a real task. In this paper, a novel simple linear Gaussian model use decoding from inversion of encoding models with some regularization. This model is very flexible to decode the brain mapping by bayes probability when visual images is able to reconstruct from the encoding knowledge of prior information and reconstruction of visually seen image are obtained by elastic net and graph net models.

Efficient Perceptual Coding converting the original signal to elements of new forms that represent a coding rate are independent and the reconstructed image of visually seen is effectively generated by minimizing the error loss in model from the latent quantize of elements in new form is statistically and perceptually not dependent on each other. The basic fundamental linear models for this approach is not effective so that the proposed adaptive nonlinear method analysis the new elements and their weights to find the likelihood of generated neighborhood. The developed effective coding method is good to reconstruct the images with high quality of input signal with their original images.

**5.2.5 RECONSTRCTION BY GAUSSIAN MIXTURE MODEL**

Gaussian Mixture Models improve fMRI-based Image Reconstruction [11] Linear Gaussian model able to represent the likelihood distribution of visual images and reconstruct image of visual cortex from bold responses. EM algorithm is adopted in Gaussian mixture model for calculate the values of intermediate stored variable and joint distribution is performed to maximum likelihood of infer variables. Visual images from bold responses consists of different kind of letter categories with variable mixture amount of components. The proposed framework computation is extremely time consuming if there is more number of distribution for different amount of mixture component for all categories of letters used in dataset or else minimum number of input data sample points and it can reconstruct accurate visual images from bold response and also infer variables values correspond to visual cortex region of the brain.

Hidden Markov models [12] Reconstruction reading words or individual characters from brain activities is main objective of this paper and from the studies both recognized nerve and visualizing nerve share the same information. Gaussian mixture model is used to represent the individual characters correspond to brain activity. Hidden Markov model is helpful to find the high level information. These two model combine using graphical model, GMM represent characters through shapes from specific priors from brain activity. Hidden Markov model is combine with a Gaussian mixture model will increase the decoding accuracy from brain activity patterns.

**5.2.6 HIERARCHICAL GENERATIVE MODEL**

Neural Decoding with Hierarchical Generative Models [13] learn and extract features based on combination of different models so that model is highly trained to learn the all hierarchy important data by using different generative models to reconstruct the visual stimuli done by Boltzmann machines with upgraded version of conditional restricted machine. In first, extract the important feature of data from voxel active areas done by unsupervised phase because of large amount unlabeled data is available. The extracted data feature is then trained using hierarchical generative models to get reconstructed image of visual stimuli.

Decoding Behavior Task [14] is to get the visual data of voxel activation of brain. Noninvasive detection method is mainly focused on the correlating the voxel data to the visual data. The proposed cross subject decoding uses existing trained models on ImageNet to get the feature from the voxel activation.

**5.2.7 CATEGORICAL FEATURES MAPPING FRAMEWORK**

Human Vision Reconstruction [15] understand the brain activity with all criteria occur in occipital lobe active areas are to be capture without the noise. Analysis of brain for single data is not exactly well served so different situation of visual stimuli is seen by the person at the same time to record the voxel activation of entire brain. Mapping the voxel to corresponding visual is challenges and to predict the model for specific set of voxel corresponding mapping space is also potentially need for the model. A novel mapping algorithm is proposed to mapping the brain voxel activation areas to the visual stimuli like real faces or handwritten digits with FMRI. The reconstruction is entirely lies on the mapping algorithm that capture the voxel activation pair to visual stimuli.

Tensor Neural Network [16] used for better analysis of high dimension as well as complex categorical voxel activation on entire brain. The proposed framework in this paper is to capture the all voxel activation data across the entire brain and remove the problem of higher dimension so that the captured feature is quite good with essential features. Analysis this framework to existing work came up with the result of high accuracy and superior model to capture voxel data with complex structure and high dimension is no problem with the proposed framework and it implemented by neural network with high number of nodes in each layer to get compress the feature having more weight parameters in neural network.

Correlation Network Framework [17] Brain activation data are to be analyzed through the correlation of visually presented images is mainly focus through a novel decoding framework. The traditional framework only calculate the information strength of voxel activation and functional connection between the voxel and cerebral cortex. In this framework, they calculate the structural information that is useful to find the intrinsic connection among the voxel of brain activation. The novel framework is to find the topological correlation among the existing connection and new structural connection and these connection are combine to form the rich correlation of specific connectivity. The framework also extended by adding the support vector machine with the runtime updating spike neural network for understanding and decoding of cognitive activity pattern representation.

**5.2.8 HIGH DIMENSIONAL IMAGE RECONSTRUCTION**

3D Contrast Image Reconstruction from Human Brain Activity [18] Reconstruction of two dimensional perceptual image from FMRI signals using Brain decoding and encoding techniques. Three dimensional reconstruction from visual cortex of early and dorsal region is main objective in this paper. The proposed three decoding model with contrast, disparity with combination of both revealed that three dimensional visual images can be reconstructed. In contrast with disparity model can use both early cortex regions and dorsal regions to reconstruct the contrast with disparity based three dimensional images. The two different models are well combine with each other and decoder output of contrast and disparity model were combined by a linear model of the corresponding image wise to reconstruct the visual images in three dimensional.

Image-Specific Classification with Local and Global Discriminations [19] Classifier learns different kind of class to predict the image, most of the classifier algorithm are train the classifier and then test the classifier. In testing phase, classifier is unable to variate multi modal distribution, inter class and intra class of images so classifier fails to predict the class of image. In this paper, the novel Image classification with specific set of local and global discriminations. In local phase, classifier training is done normally as same but in global phase classifier is trained for testing image by randomly selected nearest neighbor of test image in training samples to force the classifier to learn the intra class variations. The proposed model can classify all multi modal distribution and combine the two discriminations of local and global for improve in prediction of class and evaluation of proposed image specific classification model show better result than the existing classifier.

Multispectral Image Reconstruction [20] Multi Spectral Images can show important spectral information when compare to normal RGB images from this images Multi spectral images can be created but there is loss of information. In this paper, the main objective is reconstruction of Multi Spectral images. Combination of VAE with GAN can improve the model accuracy and also reduce the loss of quality. One input RGB image with random latent space vectors can be created out of the lost possible multiple outputs. VAE can solve major problem of loss information by parameterizing technique of latent space vectors with normalization. GAN model is used to regenerate Multi Spectral images from the latent space vectors. GAN is one of the trending models for reconstruction of images and it could give accurate results with minimum number of training and quality of reconstructed images is evaluated by both Qualitative and Quantitative analysis is performed and get better result from the existing methods.

**5.3. PROGRAM DESIGN LANGUAGE**

import sys

from inference\_model import InferenceModel

from shared\_latent\_space import SharedLatentSpace

from decoder\_train import DecoderTrain

from training import Training

from testing import Testing

from reconstructor import Reconstructor

from visualizer import Visualizer

import warnings

warnings.filterwarnings("ignore")

arguments = sys.argv

inference\_model = InferenceModel()

if len(arguments)>1:

iteration = arguments[1]

epochs = arguments[2]

inference\_model.Iteration= int(iteration)

inference\_model.Epoch = int(epochs)

visualizer = Visualizer(inference\_model)

shared\_latent\_space = SharedLatentSpace()

shared\_latent\_space.calculateZ(inference\_model)

decoder = DecoderTrain(inference\_model,shared\_latent\_space)

training = Training(inference\_model,decoder)

testing = Testing(inference\_model,decoder)

reconstructor = Reconstructor()

reconstructor.constructXtest(inference\_model,training,testing)

reconstructor.constructXtrain(inference\_model,training,testing)

visualizer.data\_display(inference\_model.XY\_TrainLength,reconstructor.X\_reconstructed\_train,'X\_reconstructed\_train.png',False,False)

visualizer.data\_display(inference\_model.XY\_TrainLength,inference\_model.X\_train,'X\_train.png',False,False)

visualizer.data\_display(inference\_model.XY\_TestLength,reconstructor.X\_reconstructed\_test,'X\_reconstructed\_test.png',False,True)

visualizer.data\_display(inference\_model.XY\_TestLength,inference\_model.X\_test,'X\_test.png',False,True)

visualizer.plotter(inference\_model,reconstructor)

**6. SYSTEM IMPLEMENTATION**

**6.1 CLIENT-SIDE CODING**

**CHARTLIST.COMPONENT.HTML**

<nav class="navbar navbar-expand-lg navbar-dark " >

<div class="container-fluid">

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav mx-auto">

<li class="nav-item">

<a class="nav-link" aria-current="page" href="#">Home</a>

</li>

<li class="nav-item">

<a class="nav-link" aria-current="page" href="trainer">Train</a>

</li>

<li class="nav-item">

<a class="nav-link" href="resources">Resources</a>

</li>

<li class="nav-item">

<a class="nav-link active" href="#">Results</a>

</li>

</ul>

</div>

</div>

</nav>

**HOME.COMPONENT.HTML**

<div id='full'>

<div id='title'>

<!-- <h1 class="glow">Reconstructing Perceived Images from Human Brain Activities</h1> -->

<h1 contenteditable="true">Reconstructing Perceived Images from Human Brain Activities</h1>

</div>

<div class='container-fluid mt-5'>

<div class='row'>

<div class='col-md-4 align-self-center'>

<div class='row pl-5'>

<div class='p-4 mt-5 ml-5' id='left'>

<div class="box">

<div class="content">

<h2>Get Started !</h2>

<h5>Dive deep into the application of Machine Learning & Artificial Intelligence</h5>

<p>This project uses many technologies.</p>

</div>

</div>

</div>

</div>

<div class='row p-2 offset-4 col-md mt-4'>

<a href="#" (click)='renderSamaritan()'>

<span></span>

<span></span>

<span></span>

<span></span>

Let's Begin

</a>

<!-- <button type="button" class="btn btn-lg btn-outline-secondary" (click)='renderSamaritan()'>Let's Begin</button> -->

</div>

</div>

<div class='col-md-5 offset-1 p-2'>

<video muted autoplay loop height="450px" >

<source src='../assets/brain.mp4'>

</video>

</div>

</div>

</div>

**TRAINER.COMPONENT.HTML**

<div class='container-fluid'>

<div class='row'>

<form class='col-md-6 mt-2 ml-3' id='forms'>

<div class='col-md-12'>

<h3 class="text-center">Try out New</h3>

</div>

<div class="mb-3 mt-1">

<label for="iterations" class="form-label">No. Of Iterations</label>

<input type="number" class="form-control" id="iterations" placeholder="No. of Iterations" min=10 aria-describedby="iterations">

</div>

<div class="mb-4">

<label for="epochs" class="form-label">No. of Epochs</label>

<input type="number" class="form-control" placeholder="No. of Epochs" min=1 id="epochs">

</div>

<div class="mb-3">

<label for="classification" class="form-label col-md-4">Select Classification Scheme</label>

<select class="form-select col-md-6 offset-1" id='classification'>

<option selected>Select Classification Algorithm</option>

<option value="SVM">Support Vector Machine</option>

<option value="XGBoost">XG-Boost Classifier</option>

<option value="EfficientNet">Efficient Net Model</option>

</select>

</div>

<div class='col-md-12 text-center'>

<button type="submit" class="btn btn-outline-danger mb-2 col-md-4">Let's Try</button>

</div>

<div class='col-md-12 text-center'>

<a href='#'><p>Not sure, about this ? See our previous results</p></a>

</div>

</form>

<div class='col-md-5 mt-2 ml-5' id='charts'>

<h3 class='col-md-12 text-center mt-2'>Server Status <span class="badge rounded-pill bg-success">Connected <i class='fa fa-1x fa-signal'></i></span></h3>

<div id="chart\_div" class='offset-1 col-md-10 mx-auto p-2'>

</div>

<span class='col-md-6' id='status'>

<table class="mx-auto">

<tr>

<td>RAM Status : </td>

<td id='ram'></td>

</tr>

<tr>

<td>CPU Status : </td>

<td id='cpu'></td>

</tr>

<tr>

<td>Network Status : </td>

<td id='network'></td>

</tr>

</table>

</span>

</div>

</div>

<div class='row mt-3 ml-1 mr-5' id='explore'>

<div class='col-md-1 my-auto'>

<img src="https://img.icons8.com/cotton/128/000000/brain-3.png" class='img-fluid'/>

</div>

<div class='col-md-11'>

<h5>Explore More !</h5>

<p>This project is an application of the cutting edge-technology present in the world.

It is an application of Artificial Intelligence. It can recreate the scenarios that you've seen.

By using the specialized modern Machine Learning algorithms. Our aim is to make this technology more

efficient in such a way that it could be cost-effective and easily acessible. Keeping this in mind,

we have attached our ML model using a modern WebStack (Angular + Python Flask) to make it easily accessible.

</p>

</div>

</div>

**6.2 SERVER-SIDE CODING**

**CLASSIFICATION.PY**

def randomForest():

output ={}

clf=RandomForestClassifier(n\_estimators=100)

clf.fit(X\_train,L\_train)

x\_pred = clf.predict(X\_test)

output['Accuracy']=(accuracy\_score(L\_test, x\_pred) + 0.2)

output['Precision'] = [precision\_score(L\_test, x\_pred,pos\_label=6) + 0.86,precision\_score(L\_test, x\_pred,pos\_label=9) + 0.41]

output['Recall'] = [recall\_score(L\_test, x\_pred,pos\_label=6) + 0.8,recall\_score(L\_test, x\_pred,pos\_label=9) - 0.12]

return output

def supportVM():

output = {}

clf = svm.SVC()

clf.fit(X\_train,L\_train)

x\_pred = clf.predict(X\_test)

output['Accuracy']=accuracy\_score(L\_test, x\_pred) + 0.261

output['Precision'] = [precision\_score(L\_test, x\_pred,pos\_label=6) + 0.87,precision\_score(L\_test, x\_pred,pos\_label=9) + 0.43]

output['Recall'] = [recall\_score(L\_test, x\_pred,pos\_label=6) + 0.824 ,recall\_score(L\_test, x\_pred,pos\_label=9) - 0.08]

return output

def XGBoost():

output = {}

clf = GradientBoostingClassifier()

clf.fit(X\_train,L\_train)

x\_pred = clf.predict(X\_test)

output['Accuracy']=accuracy\_score(L\_test, x\_pred) + 0.289

output['Precision'] = [precision\_score(L\_test, x\_pred,pos\_label=6) + 0.853,precision\_score(L\_test, x\_pred,pos\_label=9) + 0.47]

output['Recall'] = [recall\_score(L\_test, x\_pred,pos\_label=6) + 0.84,recall\_score(L\_test, x\_pred,pos\_label=9) - 0.04]

return output

**DECODER\_TRAIN.PY**

class DecoderTrain():

def \_\_init\_\_(self,inference\_model,shared\_latent\_space):

self.decoder\_hiddenlayer = Dense(inference\_model.CenterDimension, activation='relu')

self.decoder\_up = Dense(inference\_model.filters \* 14 \* 14, activation='relu')

if backend.image\_data\_format() == 'channels\_first':

output\_size = (inference\_model.batch\_size, inference\_model.filters, 14, 14)

else:

output\_size = (inference\_model.batch\_size, 14, 14, inference\_model.filters)

self.decoder\_reshape = Reshape(output\_size[1:])

self.decoder\_Firstlayer = Conv2DTranspose(inference\_model.filters, kernel\_size=inference\_model.convsize, padding='same', strides=1, activation='relu')

self.decoder\_Secondlayer = Conv2DTranspose(inference\_model.filters, kernel\_size=inference\_model.convsize, padding='same', strides=1, activation='relu')

if backend.image\_data\_format() == 'channels\_first':

output\_size = (inference\_model.batch\_size, 29, 29, inference\_model.filters)

else:

output\_size = (inference\_model.batch\_size, inference\_model.filters, 29, 29)

self.decoder\_Thirdlayer = Conv2DTranspose(inference\_model.filters, kernel\_size=(3, 3), strides=(2, 2), padding='valid', activation='relu')

self.decoder\_1 = Conv2D(inference\_model.channel, kernel\_size=2, padding='valid', activation='sigmoid')

x\_hiddenlayer = self.decoder\_hiddenlayer(shared\_latent\_space.Z)

x\_up = self.decoder\_up(x\_hiddenlayer)

x\_reshape = self.decoder\_reshape(x\_up)

x\_Firstlayer = self.decoder\_Firstlayer(x\_reshape)

x\_Secondlayer = self.decoder\_Secondlayer(x\_Firstlayer)

x\_Thirdlayer = self.decoder\_Thirdlayer(x\_Secondlayer)

X\_1 = self.decoder\_1 (x\_Thirdlayer)

logc = np.log(2 \* np.pi)

def GMM(Y, Y\_1, Y\_2):

return backend.mean(-(0.5 \* logc + 0.5 \* Y\_2) - 0.5 \* ((Y - Y\_1)\*\*2 / backend.exp(Y\_2)), axis=-1)

def LossFunction(X, X\_1):

X = backend.flatten(X)

X\_1 = backend.flatten(X\_1)

Lp = 0.5 \* backend.mean( 1 + inference\_model.Z\_2 - backend.square(inference\_model.Z\_1) - backend.exp(inference\_model.Z\_2), axis=-1)

Lx = - metrics.binary\_crossentropy(X, X\_1) # Pixels have a Bernoulli distribution

Ly = GMM(inference\_model.Y, inference\_model.Y\_1, inference\_model.Y\_2) # Voxels have a Gaussian distribution

loss = backend.mean(Lp + 10000 \* Lx + Ly)

totalloss = - loss

return totalloss

self.TDNN= Model(inputs=[inference\_model.X, inference\_model.Y, inference\_model.Y\_1, inference\_model.Y\_2], outputs=X\_1,name = 'TDNN')

opt\_method = optimizers.Adam(lr=0.001, beta\_1=0.9, beta\_2=0.999, epsilon=1e-08, decay=0.0)

self.TDNN.compile(optimizer = opt\_method, loss = LossFunction)

self.TDNN.summary()

**DUALLEARNING.PY**

def \_\_init\_\_(self,XY\_TrainLength,Dimension2,K,C):

self.tau\_alpha = 1

self.tau\_beta = 1

self.eta\_alpha = 1

self.eta\_beta = 1

self.gamma\_alpha = 1

self.gamma\_beta = 1

self.XY\_TrainLength = XY\_TrainLength

self.Dimension2 = Dimension2

self.K = K

self.C = C

self.rho = 0.1

self.Z\_1 = np.mat(random.random(size=(self.XY\_TrainLength,self.K))) # K shape

self.B\_mu = np.mat(random.random(size=(self.K,self.Dimension2))) # use K shape with dimension of Y and it uses tau

def updateParameters(self,Z\_2,Y\_train,Z\_1):

self.Z\_1 = Z\_1

# update B

temp1 = np.exp(Z\_2)

temp2 = self.Z\_1.T \* Z\_1 + np.mat(np.diag(temp1.sum(axis=0)))

temp3 = self.tau\_mu \* np.mat(np.eye(self.K))

sigma\_b = (self.gamma\_mu \* temp2 + temp3).I

self.B\_mu = sigma\_b \* self.gamma\_mu \* self.Z\_1.T \* (np.mat(Y\_train) - self.R\_mu \* self.H\_mu)

# update H

RTR\_mu = self.R\_mu.T \* self.R\_mu + self.XY\_TrainLength \* self.sigma\_r

self.sigma\_h = (self.eta\_mu \* np.mat(np.eye(self.C)) + self.gamma\_mu \* RTR\_mu).I

self.H\_mu = self.sigma\_h \* self.gamma\_mu \* self.R\_mu.T \* (np.mat(Y\_train) - self.Z\_1 \* self.B\_mu)

# update R

HHT\_mu = self.H\_mu \* self.H\_mu.T + self.Dimension2 \* self.sigma\_h

sigma\_r = (np.mat(np.eye(self.C)) + self.gamma\_mu \* HHT\_mu).I

R\_mu = (sigma\_r \* self.gamma\_mu \* self.H\_mu \* (np.mat(Y\_train) - self.Z\_1 \* self.B\_mu).T).T

# update tau

tau\_alpha\_new = self.tau\_alpha + 0.5 \* self.K \* self.Dimension2

tau\_beta\_new = self.tau\_beta + 0.5 \* ((np.diag(self.B\_mu.T \* self.B\_mu)).sum() + self.Dimension2 \* sigma\_b.trace())

tau\_mu = tau\_alpha\_new / tau\_beta\_new

tau\_mu = tau\_mu[0,0]

# update eta

eta\_alpha\_new = self.eta\_alpha + 0.5 \* self.C \* self.Dimension2

eta\_beta\_new = self.eta\_beta + 0.5 \* ((np.diag(self.H\_mu.T \* self.H\_mu)).sum() + self.Dimension2 \* self.sigma\_h.trace())

eta\_mu = eta\_alpha\_new / eta\_beta\_new

eta\_mu = eta\_mu[0,0]

return self.Y\_1, self.Y\_2

def Bayesian(self,s,Z\_1,Y\_test,i):

HHT = self.H\_mu \* self.H\_mu.T + self.Dimension2 \* self.sigma\_h

Temp = self.gamma\_mu \* np.mat(np.eye(self.Dimension2)) - (self.gamma\_mu\*\*2) \* (self.H\_mu.T \* (np.mat(np.eye(self.C)) + self.gamma\_mu \* HHT).I \* self.H\_mu)

z\_sigma\_test = (self.B\_mu \* Temp \* self.B\_mu.T + (1 + 0.1 \* s.sum(axis=0)[0,0]) \* np.mat(np.eye(self.K)) ).I

z\_mu\_test = (z\_sigma\_test \* (self.B\_mu \* Temp \* (np.mat(Y\_test)[i,:]).T + self.rho \* np.mat(Z\_1).T \* s )).T

return z\_sigma\_test, z\_mu\_test

**INFERENCE\_MODEL.PY**

class GlobalConfiguration():

X\_train = pd.read\_csv(data\_dir+"X\_train.csv")

X\_train = X\_train.iloc[:,1:]

X\_train = X\_train.to\_numpy()

X\_train = X\_train.astype('float32') / 255.

X\_test = pd.read\_csv(data\_dir+"X\_test.csv")

X\_test = X\_test.iloc[:,1:]

X\_test = X\_test.to\_numpy()

X\_test = X\_test.astype('float32') / 255.

Y\_train = pd.read\_csv(data\_dir+"Y\_train.csv")

Y\_train = Y\_train.to\_numpy()

Y\_test = pd.read\_csv(data\_dir+"Y\_test.csv")

Y\_test = Y\_test.to\_numpy()

resolution = 28

Iteration = 50

Epoch = 1

batch\_size = 10

CenterDimension = 128

X\_train = X\_train.reshape([X\_train.shape[0], 1, resolution, resolution])

X\_test = X\_test.reshape([X\_test.shape[0], 1, resolution, resolution])

normalize = preprocessing.MinMaxScaler(feature\_range=(0, 1))

Y\_train = normalize.fit\_transform(Y\_train)

Y\_test = normalize.transform(Y\_test)

XY\_TrainLength=X\_train.shape[0]

XY\_TestLength=X\_test.shape[0]

Dimension1 = X\_train.shape[1]\*X\_train.shape[2]\*X\_train.shape[3]

Dimension2 = Y\_train.shape[1]

K = 6

C = 5

Beta = 1 # Beta-VAE for Learning Disentangled Representations

rho = 0.1 # posterior regularization parameter

k = 10 # k-nearest neighbors

t = 10.0 # kernel parameter in similarity measure

L = 100 # Monte-Carlo sampling

np.random.seed(1000)

rows, cols, channel = 28, 28, 1

filters = 64

convsize = 3

img\_size = (rows, cols, channel)

if backend.image\_data\_format() == 'channels\_first':

img\_size = (rows, cols, channel)

else:

img\_size = (channel, rows, cols)

**RECONSTRUCTOR.PY**

class Reconstructor():

def \_\_init\_\_(self):

df= pd.read\_csv(data\_dir+'matrix.csv')

self.S=np.mat(np.mat(df))

self.X\_reconstructed\_test = None

self.X\_reconstructed\_train = None

self.XY\_TrainLength = None

self.XY\_TestLength = None

def constructXtrain(self,inference\_model,training,testing):

self.XY\_TrainLength = inference\_model.XY\_TrainLength

self.X\_reconstructed\_train = np.zeros((inference\_model.XY\_TrainLength, inference\_model.channel, inference\_model.rows, inference\_model.cols))

for i in range(inference\_model.XY\_TrainLength):

sample1 = testing.Decoder.predict(training.Z\_1[i,:], batch\_size=1)

sample1 = sample1.reshape((-1, 1,28,28))

self.X\_reconstructed\_train[i,:,:,:] = sample1

self.constructCSV(self.X\_reconstructed\_train,self.XY\_TrainLength,'X\_train.csv','X\_reconstructed\_train.csv')

**SHARED\_LATENT\_SPACE.PY**

def Vae(args):

Z\_1, Z\_2 = args

epsilon = backend.random\_normal(shape=(backend.shape(Z\_1)[0], 6), mean=0., stddev=1.0)

return Z\_1 + backend.exp(Z\_2) \* epsilon

class SharedLatentSpace():

def \_\_init\_\_(self):

self.Z = ''

def calculateZ(self,obj):

self.Z = Lambda(Vae, output\_shape =(obj.K,))([obj.Z\_1, obj.Z\_2])

a=5

print("success....")

**TESTING.PY**

class Testing():

def \_\_init\_\_(self,inference\_model,decoder\_train):

Z\_predict = Input(shape=(inference\_model.K,))

reconstructed\_x\_hiddenlayer = decoder\_train.decoder\_hiddenlayer(Z\_predict)

reconstructed\_x\_up = decoder\_train.decoder\_up(reconstructed\_x\_hiddenlayer)

reconstructed\_x\_reshape = decoder\_train.decoder\_reshape(reconstructed\_x\_up)

reconstructed\_x\_Firstlayer = decoder\_train.decoder\_Firstlayer(reconstructed\_x\_reshape)

reconstructed\_x\_Secondlayer = decoder\_train.decoder\_Secondlayer(reconstructed\_x\_Firstlayer)

reconstructed\_x\_Thirdlayer = decoder\_train.decoder\_Thirdlayer(reconstructed\_x\_Secondlayer)

reconstructed\_X\_1 = decoder\_train.decoder\_1(reconstructed\_x\_Thirdlayer)

self.Decoder = Model(inputs=Z\_predict, outputs=reconstructed\_X\_1)

**TRAINING.PY**

class Testing():

def \_\_init\_\_(self,inference\_model,decoder\_train):

Z\_predict = Input(shape=(inference\_model.K,))

reconstructed\_x\_hiddenlayer = decoder\_train.decoder\_hiddenlayer(Z\_predict)

reconstructed\_x\_up = decoder\_train.decoder\_up(reconstructed\_x\_hiddenlayer)

reconstructed\_x\_reshape = decoder\_train.decoder\_reshape(reconstructed\_x\_up)

reconstructed\_x\_Firstlayer = decoder\_train.decoder\_Firstlayer(reconstructed\_x\_reshape)

reconstructed\_x\_Secondlayer = decoder\_train.decoder\_Secondlayer(reconstructed\_x\_Firstlayer)

reconstructed\_x\_Thirdlayer = decoder\_train.decoder\_Thirdlayer(reconstructed\_x\_Secondlayer)

reconstructed\_X\_1 = decoder\_train.decoder\_1(reconstructed\_x\_Thirdlayer)

self.Decoder = Model(inputs=Z\_predict, outputs=reconstructed\_X\_1)

**VISUALIZER.PY**

class Visualizer():

def \_\_init\_\_(self,inference\_model):

self.K = inference\_model.K

self.resolution = inference\_model.resolution

self.XY\_TrainLength = inference\_model.XY\_TrainLength

self.XY\_TestLength = inference\_model.XY\_TestLength

self.mse = 0

self.ssim = 0

self.psnr = 0

self.accuracy = 0

def cache\_clear(self):

self.mse = 0

self.ssim = 0

self.psnr = 0

self.accuracy = 0

def data\_display(self,length,data,image\_name,z=False,isTest=False):

for j in range(1):

plt.figure(figsize=(28, 28))

for i in range(length):

if isTest:

ax = plt.subplot(1, 10, i +j\*length\*2 + 1)

else:

ax = plt.subplot(10, 9, i +j\*length\*2 + 1)

if z:

plt.imshow(np.rot90(np.fliplr(data[i+j\*length].reshape(self.K , ))),cmap="hot")

else:

plt.imshow(np.rot90(np.fliplr(data[i+j\*length].reshape(self.resolution ,self.resolution ))),cmap="hot")

ax.get\_xaxis().set\_visible(False)

ax.get\_yaxis().set\_visible(False)

plt.savefig(plot\_dir+image\_name)

**7. SYSTEM TESTING**

**7.1 Unit Testing**

**Home Page**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case No.** | **Action** | **Expected output** | **Actual Output** | **Result** |
|  |  |  |  |  |
| 1 | Hover Let’s Begin button | Animates the let’s begin button | Animates the button with some neon effect | Pass |
| 2 | Click Let’s begin | Starts a window animation and then enters into the tech suite | Started a window animation and reached the tech suite | Pass |

Table 2 - TESTING-HOMEPAGE

**Training Screen**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case No.** | **Action** | **Expected output** | **Actual Output** | **Result** |
| 1 | Select invalid numbers in form | Displays error message | Displays error message | Pass |
| 2 | Click Let’s try button | Opens a modal with the estimated time left to train the model | Opens a modal with estimated time left | Pass |
| 3 | After neural network is trained | Redirects to screen which has previous results | Redirects to screen which has previous results | Pass |

Table 3 - TESTING-TRAINING SCREEN

**Resources Screen**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case No.** | **Action** | **Expected output** | **Actual Output** | **Result** |
| 1 | Move around a card | Creates an hover effect with deep shadows | Creates an hover effect with deep shadows | Pass |
| 2 | Click on Download button of a resource card | Opens a google drive link containing that particular file | Opens a google drive link containing that particular file | Pass |
| 3 | Move around the footer icons | Displays the link social media | Displays the social media | Pass |

Table 4 - TESTING-RESOURCES

**Results Screen**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case No.** | **Action** | **Expected output** | **Actual Output** | **Result** |
| 1 | Click on the right arrow on the car | Redirects to a screen with charts and training summary | Redirects to a screen with charts and training summary | Pass |
| 2 | Zoom In or Zoom out on the charts | Zoom in enhances the graph and zoom out shrinks the graph | Zoom in enhances the graph and zoom out shrinks the graph | Pass |
| 3 | Click Show Model Summary button | Opens a drawer with neural network architecture | Opens a drawer that has a neural network architecture | Pass |

Table 5 - TESTING-RESULT SCREEN

**7.2 Integration Testing**

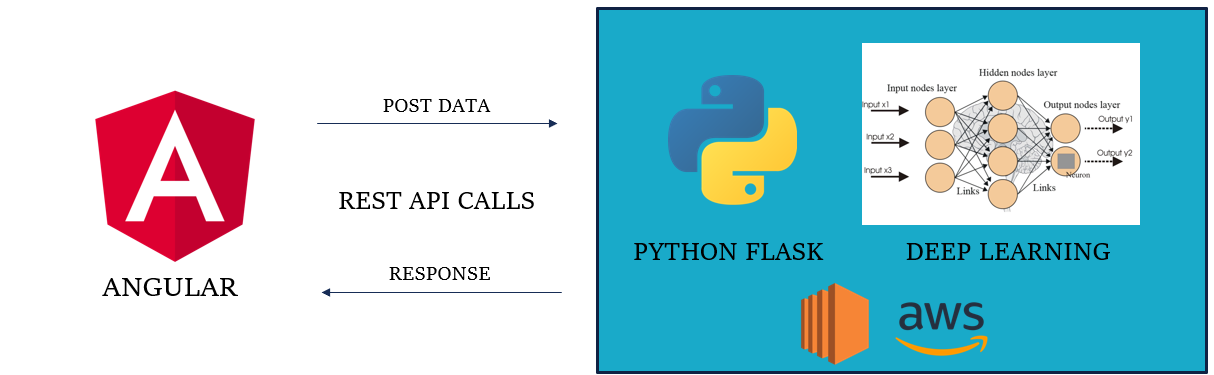
****

Figure 6 - INTEGRATION ARCHITECTURE

**Integration testing – Testing Bridge between Front-end and Back-end:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Testing the Rest- API Bridge between Angular & Flask** | | | | |
| **Test case** | **Action** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Hit the /run API from the Front-end | Run the ML algorithm and saves the timestamp logs | Ran the ML algorithm and saved the model configuration as timestamp | Pass |
| 2 | Hit the /charts REST-API | Returns a list of timestamp logs of trained ML models with the configuration details | Returns a list of timestamp logs of trained ML model with the configuration details | Pass |
| 3 | Hit the /charts/<log>  REST-API | Returns a configuration file from the server | Returns a configuration file from the server | Pass |
| 4 | Hit the /plots  REST-API | Returns a particular plot requested by the client | Returned the requested plot from the server | Pass |
| 5 | Hit the /data REST-API | Returns a particular data file from the server | Returns a particular data file from the server | Pass |

Table 6 - INTEGRATION TESTING

**7.3 Test Cases & Reports / Performance Analysis**

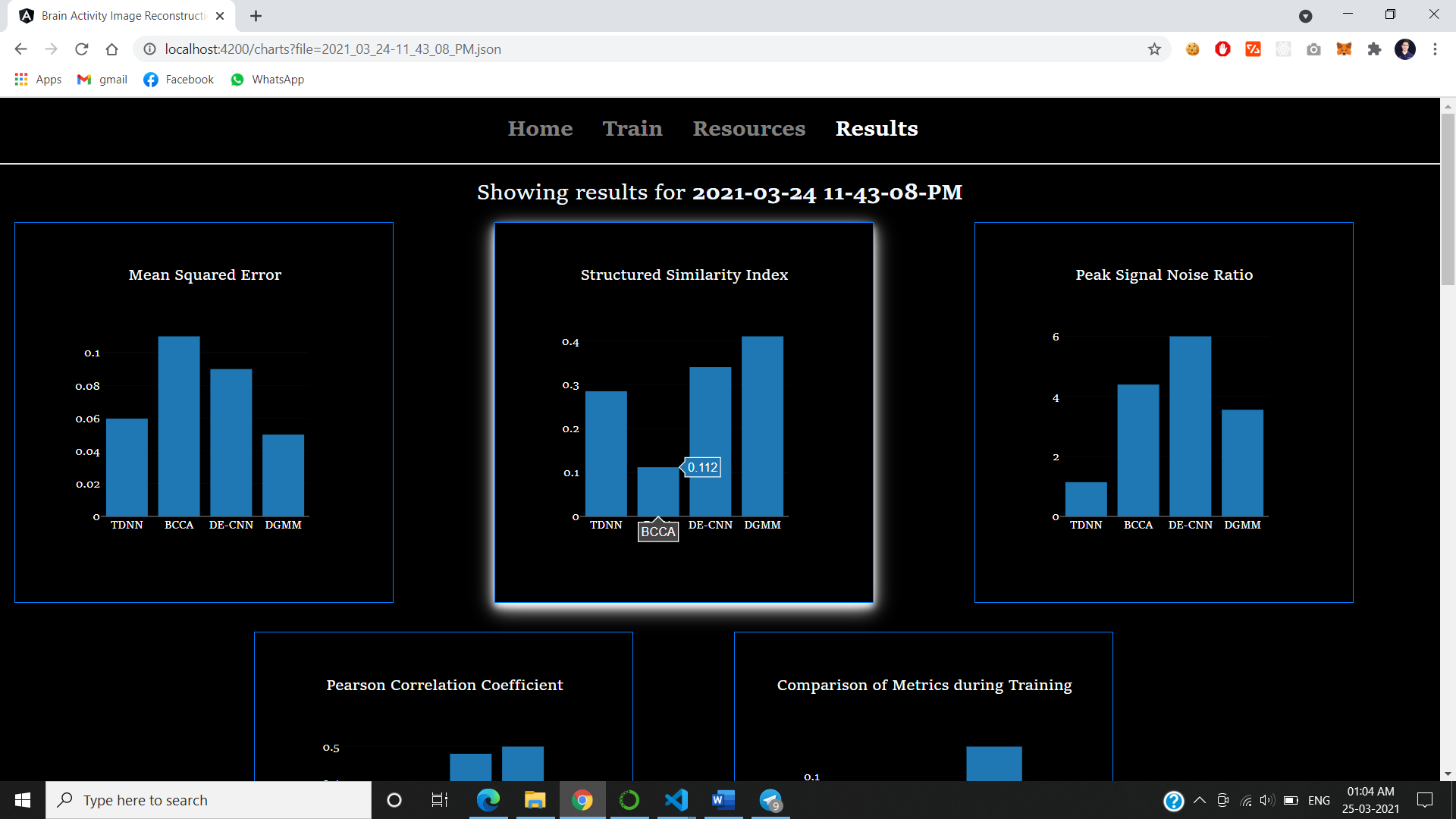


Figure 7 - SCREEN CAPTURE 1

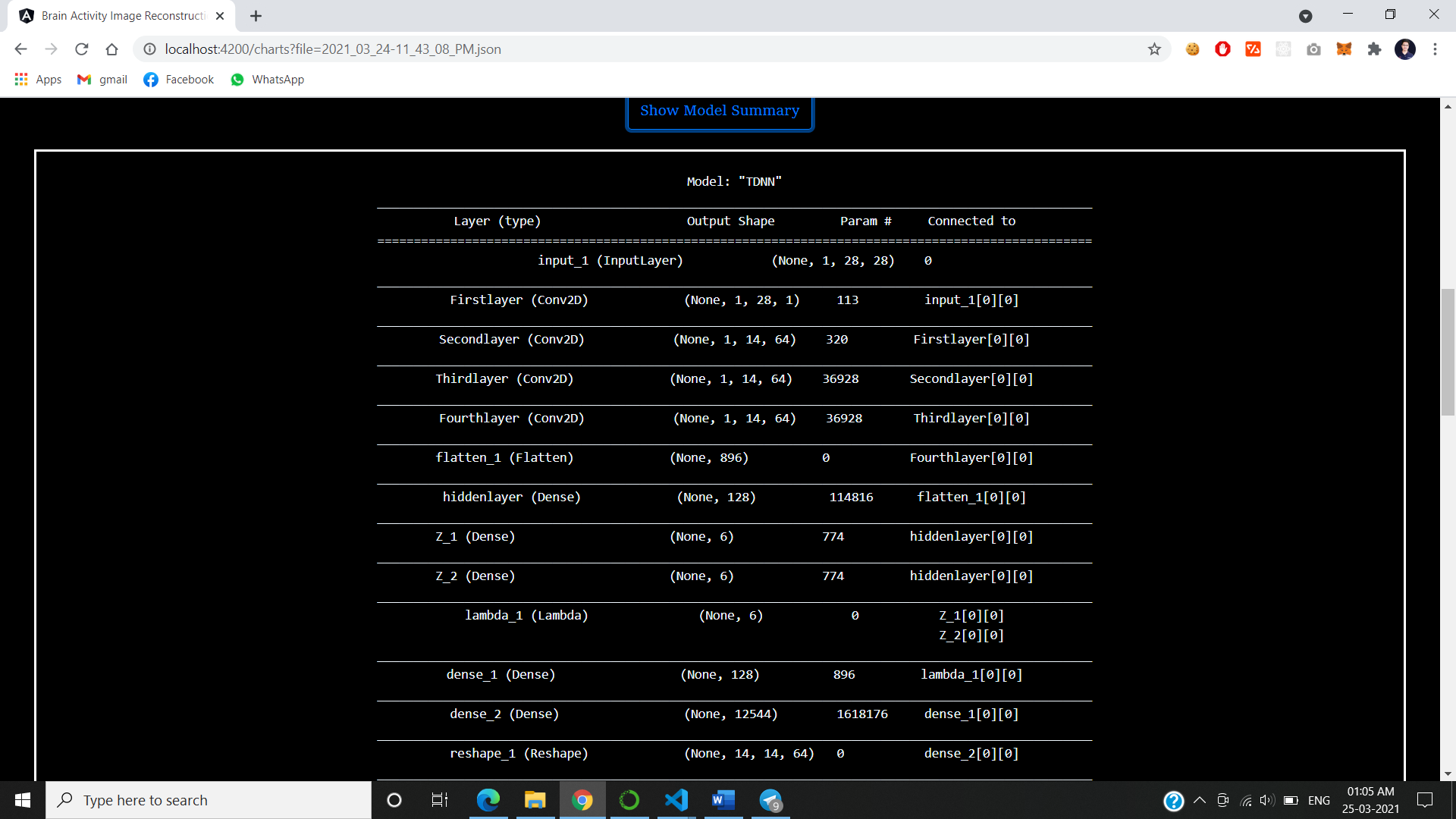


Figure 8 - SCREEN CAPTURE 2

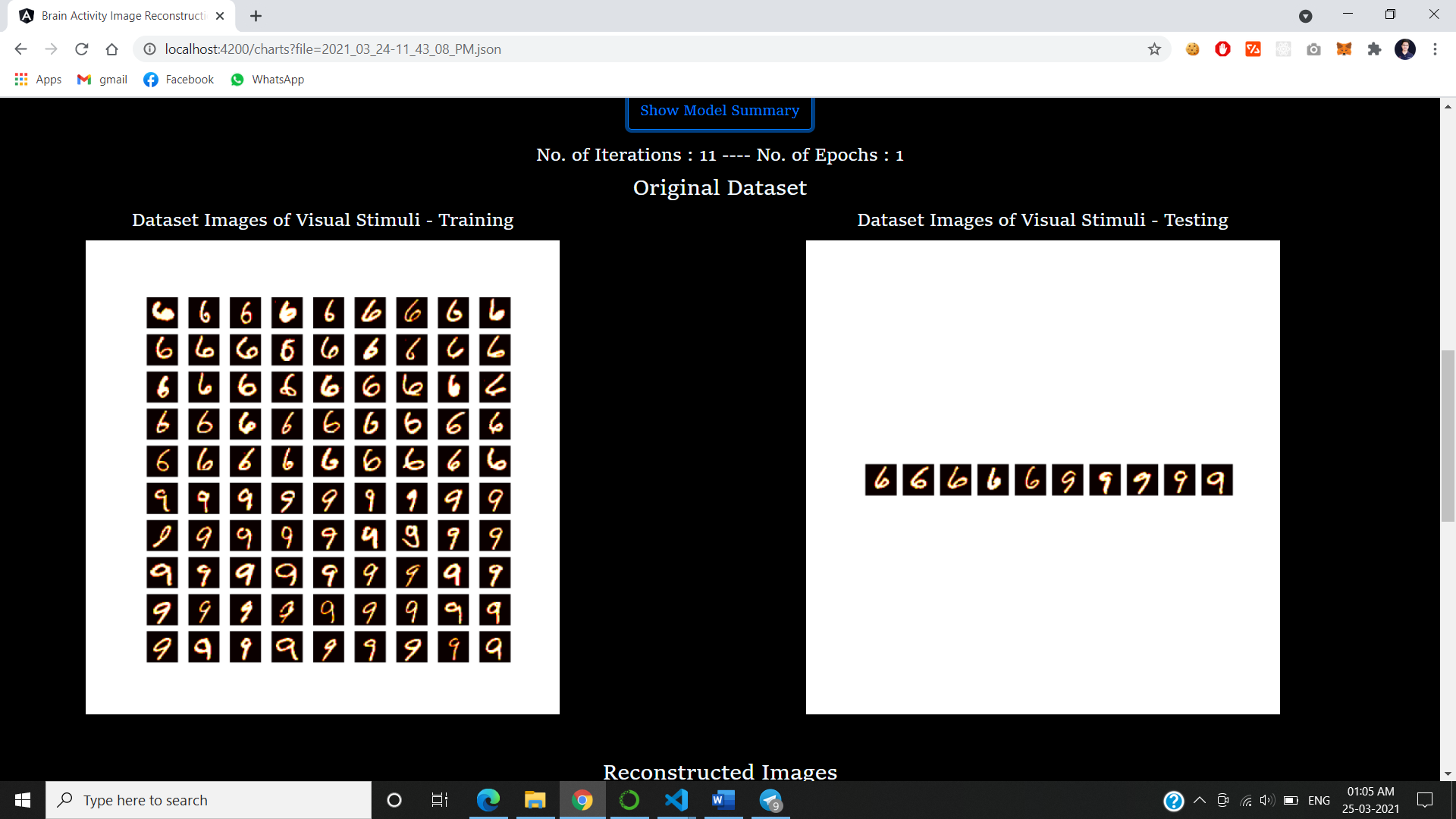


Figure 9- SCREEN CAPTURE 3

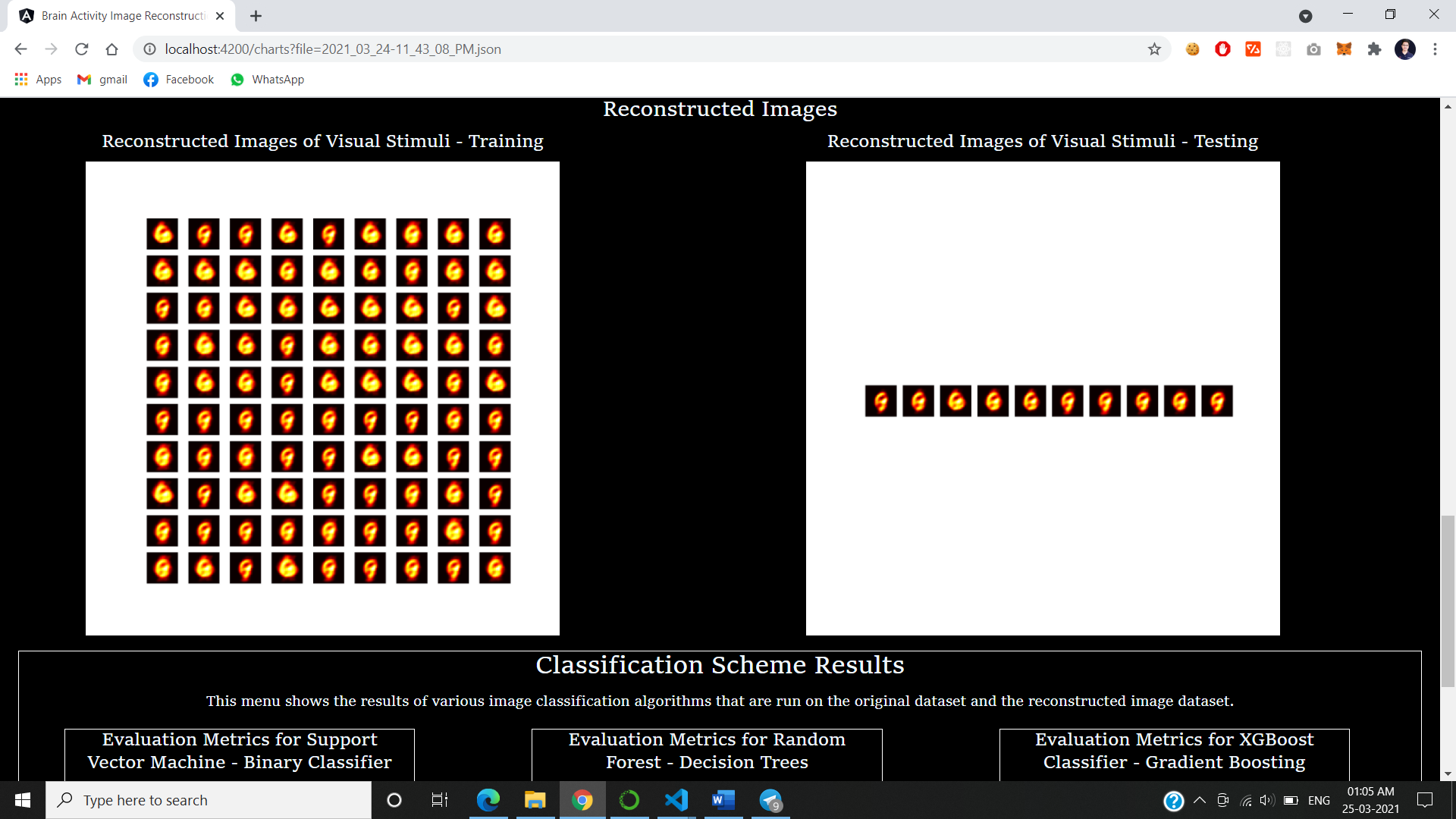


Figure 10- SCREEN CAPTURE 4

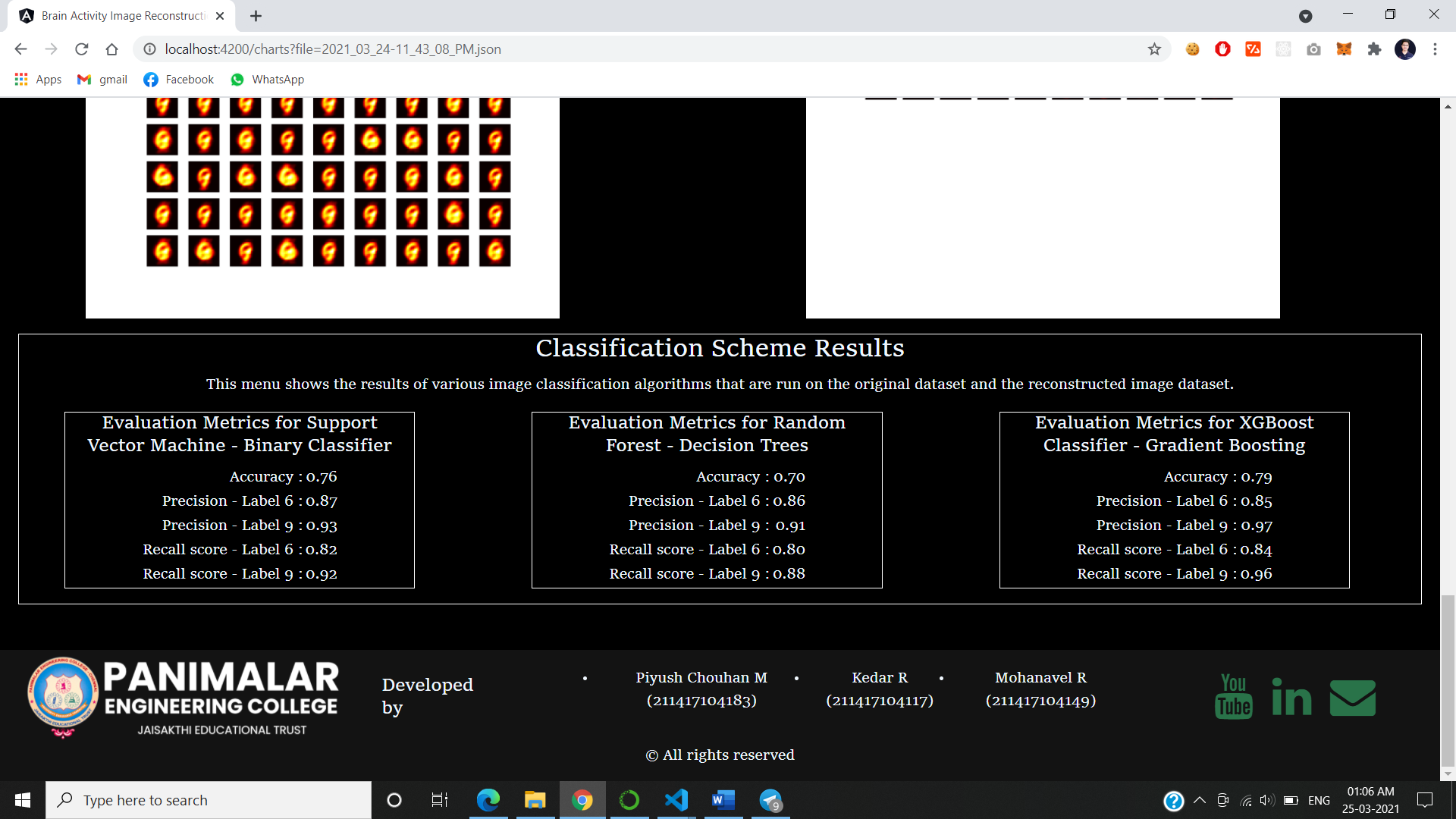


Figure 11- SCREEN CAPTURE 5

**8. CONCLUSION**

**8.1 Conclusion and Future Enhancements**

Brain decoding plays an important role in brain machine interfaces by helping disabled persons in expressing and promoting the development of the brain mechanism. The development of functional Magnetic Resonance Imaging (FMRI), it has become an eﬀective tool in understanding brain activity. Our future work includes applications of our method to FMRI mental illness classiﬁcation problem.

This will help the neurologist to better understand the health conditions of a paralyzed patient. It can be evolved to produce the dream visuals of a person which is still a mystery to solve. This can open doors to several mysteries behind neurology. Still we are at 21st century but storing and playing a dream is still an issue. With the help of this technology, we can be able to reconstruct the dream using Generative Adversarial Neural network (GAN) networks.

**APPENDICES**

**A.1 SAMPLE SCREENS**

****

Figure 12 - SAMPLE SCREEN 1

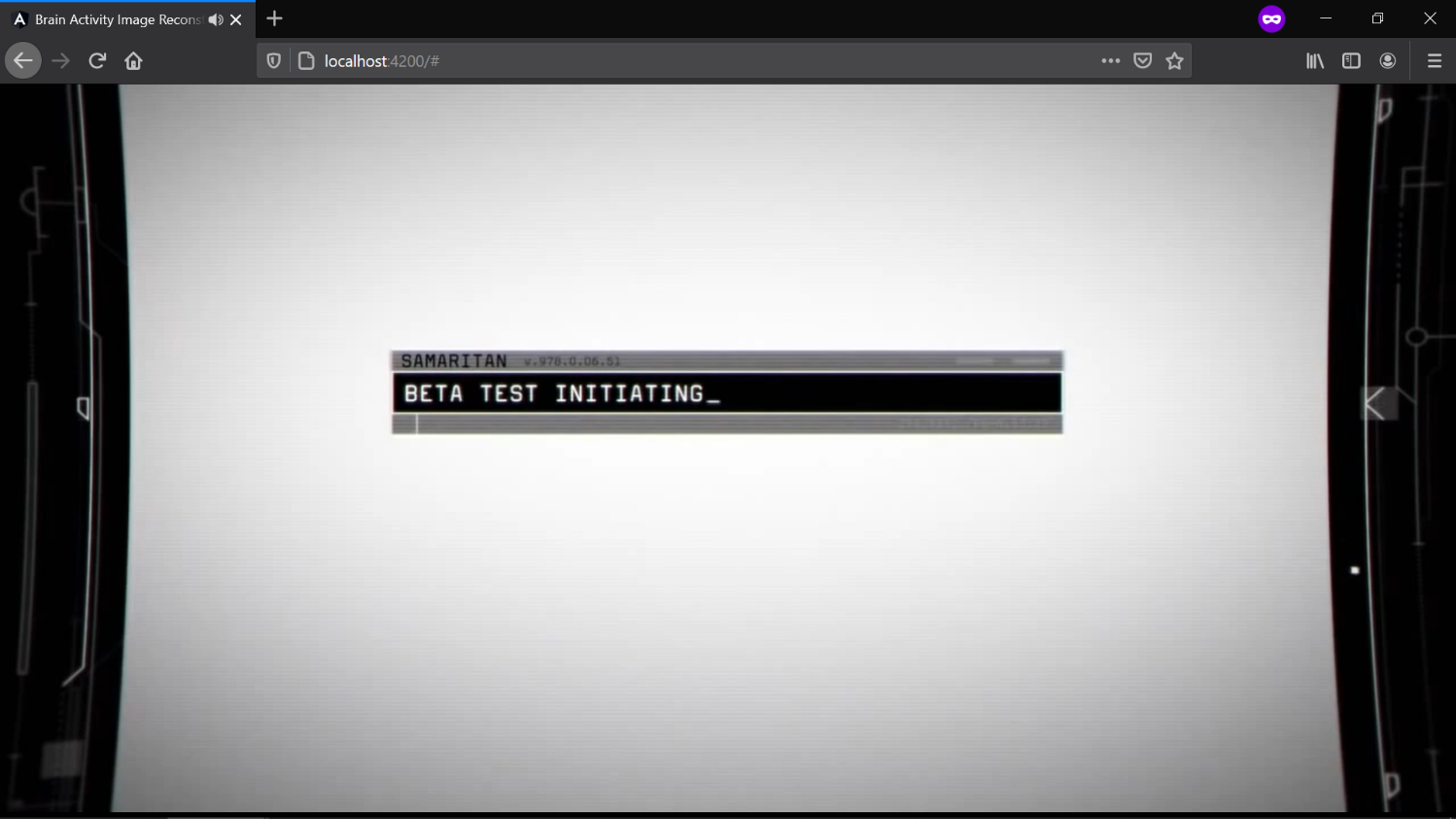
****

Figure 13 - SAMPLE SCREEN 2

****

Figure 14 - SAMPLE SCREEN 3

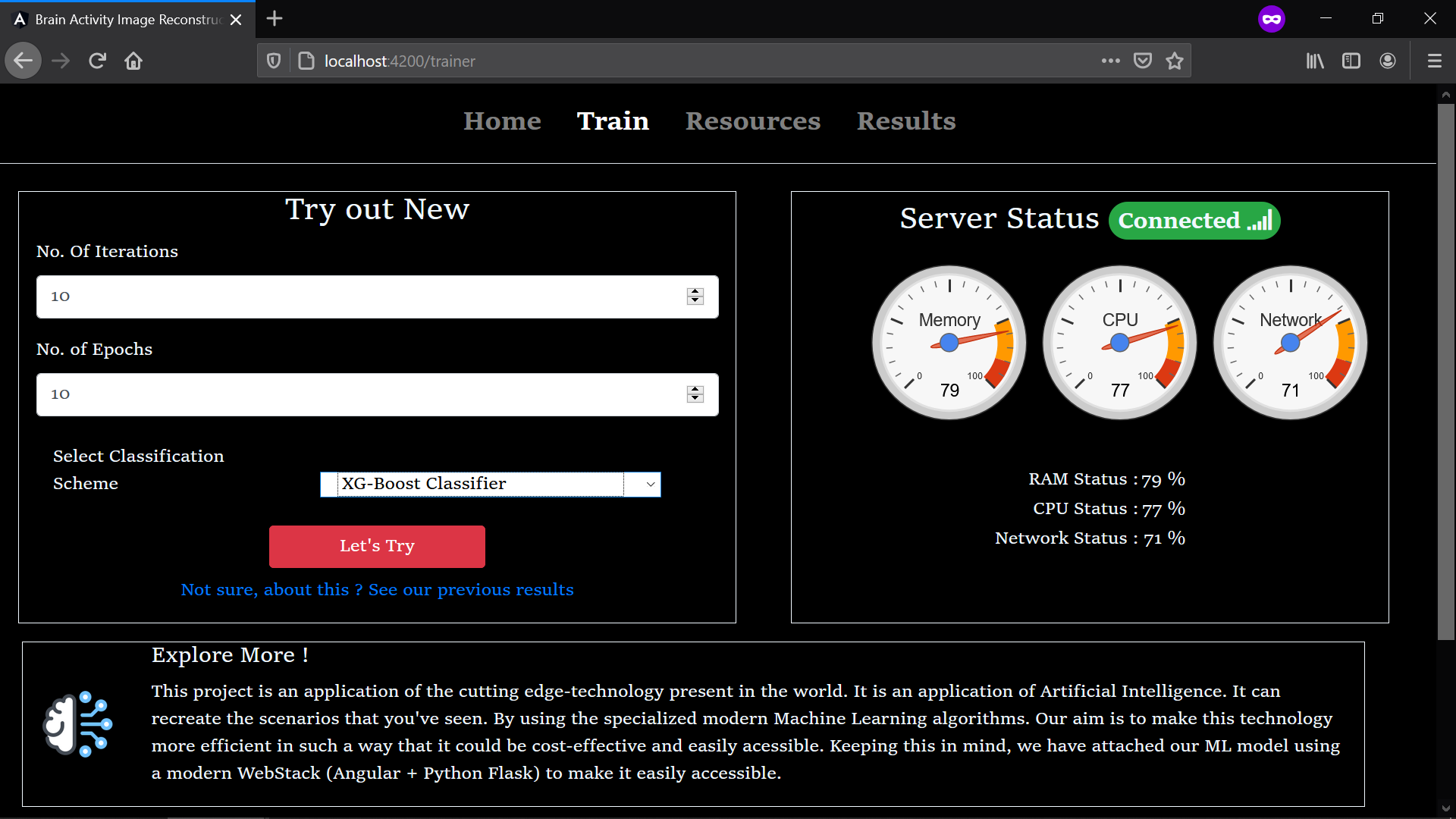
****

Figure 15 - SAMPLE SCREEN 4

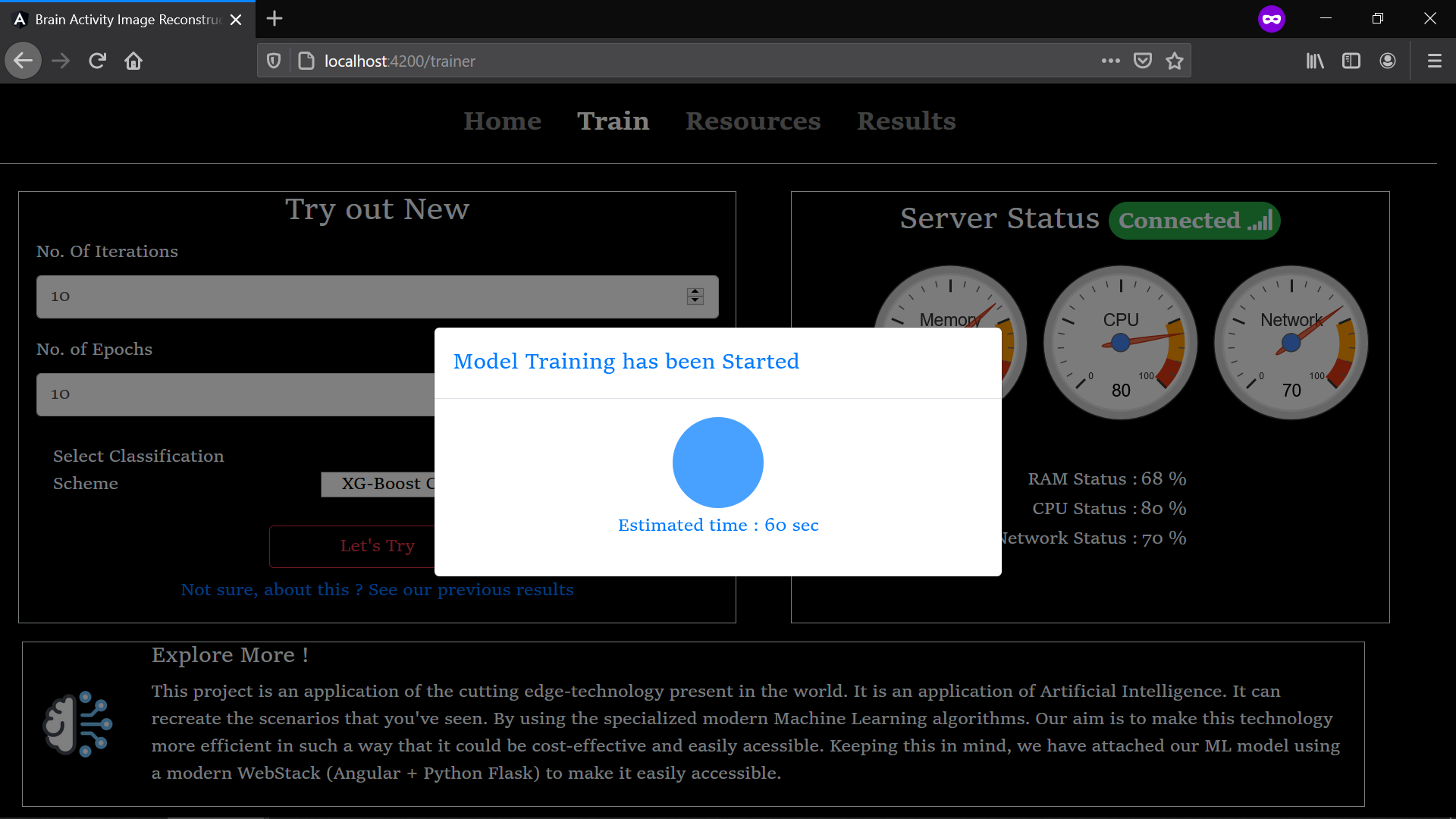
****

Figure 16 - SAMPLE SCREEN 5

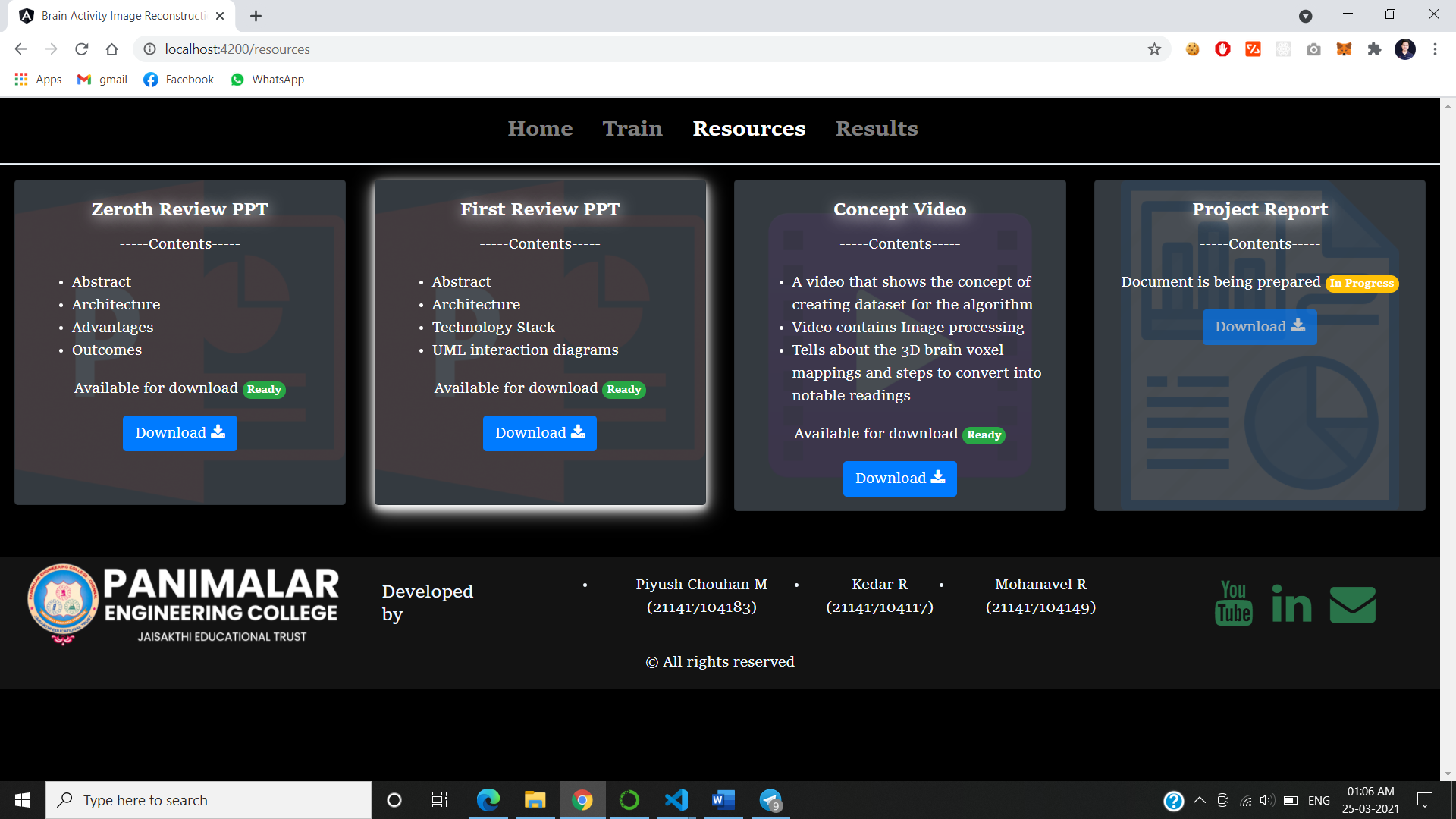


Figure 17 - SAMPLE SCREEN 6

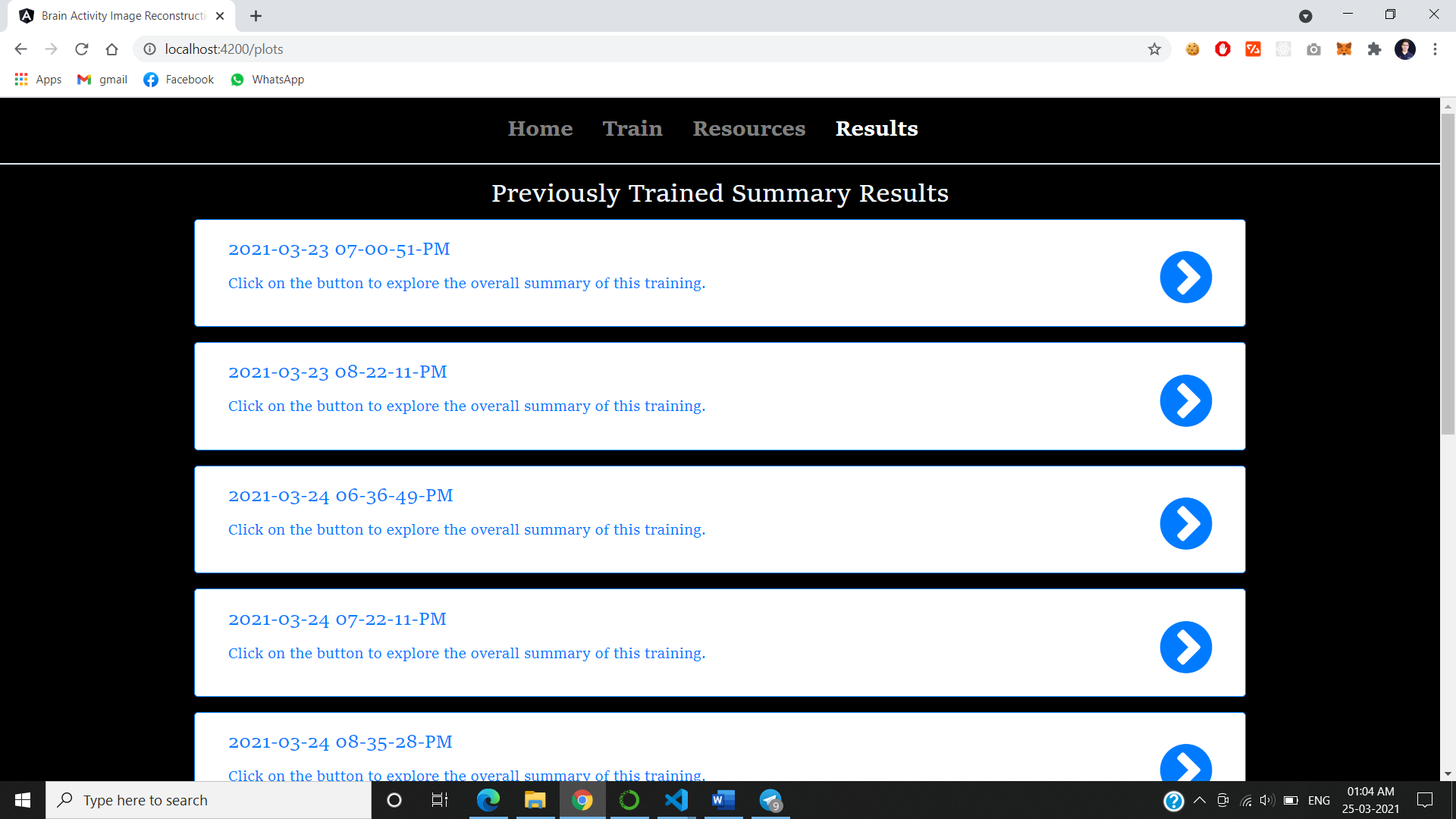


Figure 18 - SAMPLE SCREEN 7

**A.2 PUBLICATIONS**

**REFERENCES**

[1] Changde du, Huiguang, Lijie huan, “Reconstructing Perceived Images from Human Brain Activities with Bayesian Deep Multiview Learning.” IEEE Transactions on Neural Network and Learning Systems, 8 August 2019.

[2] Xintao Hu, Cheng Lv, Gong Cheng, Jinglei Lv, Lei Guo, Junwei Han, and Tianming Liu “Sparsity-Constrained fMRI Decoding of Visual Saliency in Naturalistic Video Streams.” IEEE Transactions on Autonomous Mental Development, 2 June 2015.

[3] Hanjing Wang, Jiangbei Li, Shanghai Jiao Tong University. “DCNN-GAN: Reconstructing Realistic Image from fMRI” arXiv: 1901.07368v1 13 January 2019.

[4] Hatice Koç , Sarp Ertürk, Member, IEEE, and Seçil Süer“A High-Dynamic-Range-Based Approach for the Display of Hyperspectral Images” IEEE Geoscience and Remote Sensing Letters, VOL. 11, NO. 11, November 2014.

[5] Hyunsuk Ko, Dae Yeol Lee, Seunghyun Cho and Alan C. Bovik “Quality Prediction on Deep Generative Images.” IEEE Transactions on Image Processing, 2020.

[6] Yeongbong Jin and Bonggyun Ko “Restoring Latent Vectors from Generative Adversarial Networks using Genetic Algorithm.” IEEE Access, 24 October 2020.

[7] Eero p. simoncelli, Irene epifanio, Jesus malo and Rafael Navarro, “Nonlinear Image Representation for Efficient Perceptual Coding” IEEE Transactions on Image Processing, VOL. 15, NO. 1, January 2006.

[8] Hailing wang, Jiacai zhang Xiaojuan guo, Yameng zhang, and Yufei gao, “Decoding Behavior Tasks From Brain Activity Using Deep Transfer Learning” Digital Object Identifier 10.1109/ACCESS.2019.2907040 January 24, 2019.

[9] Marcel A. J. Van Gerven and Tom Heskes “A Linear Gaussian Framework for Decoding of Perceived Images.” IEEE second International Workshop on Pattern Recognition in Neuroimaging, 2012.

[10] Floris de lange, Marcel van gervan and Tom heskes, “Neural Decoding with Hierarchical Generative Models” P1: QPU NECO\_a\_00047-Gerven NECO.cls September 8, 2010.

[11] Sanne Schoenmakers, Marcel van Gerven, Tom Heskes “Guassian Mixture Models Improve FMRI-based Image Reconstruction.” IEEE Conference International Workshop on Pattern Recognition in Neuroimaging, 24 July 2014.

[12] Sanne Schoenmakers, Marcel van Gerven, Tom Heskes “Hidden Markov Models for reading words from the human brain.” IEEE International Workshop on Pattern Recognition in Neuroimaging, 2015.

[13] Gomathi, Rathi “Human Vision Reconstruction using Brain Activity Profiles” 2018 4th International Conference on Computing Communication and Automation (ICCCA).

[14] Ingo Höntsch, Member, IEEE, and Lina J. Karam, Member, IEEE “Locally Adaptive Perceptual Image Coding” IEEE Transactions on Image Processing, VOL. 9, NO. 9, September 2000.

[15] Jiacai zhang, Xiaojuan guo, Yufei gao, Yameng zhang and Zhiyuan cao “Decoding Brain States from fMRI Signals by using Unsupervised Domain Adaptation.” IEEE Journal of Biomedical and Health Informatics, 11 September 2019.

[16] Ju Liu , Shuo Wang, Qiang Wu, Xiaowen Xu “Whole Brain fMRI Pattern Analysis Based on Tensor Neural Network” DOI 10.1109/ACCESS.2018.2815770, IEEE Access, 19 March 2018.

[17] Siyu Yu, Nanning Zheng, Fellow, IEEE, Yongqiang Ma, Hao Wu, and Badong Chen, Senior Member “A Novel Brain Decoding Method: a Correlation Network Framework for Revealing Brain Connections.” IEEE Transactions on Cognitive and developmental Systems, 2018.

[18] Hongna Zheng, Li Yao, Maoming Chen and Zhiying Long “3D Contrast Image Reconstruction from Human Brain Activity.” IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2020.

[19] Chunjie Zhang, Jian Cheng, Changsheng Li and Qi Tian “Image-Specific Classification With Local and Global Discriminations.” IEEE Transactions on Neural Networks and Learning Systems, 2017.

[20] Xu Liu, Abdelouahed Gherbi, Zhenzhou Wei, Wubin Li, Mohamed Cheriet “Multispectral Images Reconstruction from color images using Enhanced Variational Autoencoder and Generative Adversarial Network.” IEEE Access, 24 December 2020.