

-

CLASSIFICATION AND ENHANCEMENT OF SAR IMAGES USING MACHINE LEARNING TECHNIQUES

A PROJECT REPORT

Submitted by

| | |
|-------------------|-----------------------|
| V. KAMALI | (513319104015) |
| R. NISHA | (513319104021) |
| S. RAMYA | (513319104030) |
| L. SHOFICA | (513319104039) |

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING ARNI

ANNA UNIVERSITY: CHENNAI 600 025

JUNE 2023

BONAFIDE CERTIFICATE

Certified that this Project report “**CLASSIFICATION AND ENHANCEMENT OF SAR IMAGES USING MACHINE LEARNING TECHNIQUES**” is the bonafide work of “**V. KAMALI (513319104015), R.NISHA (513319104021), S.RAMYA (513319104030), L.SHOFICA (513319104039)**” who carried out the mini project work under my supervision.

SIGNATURE

SIGNATURE

HEAD OF THE DEPARTMENT

SUPERVISOR

Mr. D. PRABHU, M.E., M.B.A.,
Assistant Professor,
Department of Computer Science and
Engineering,
University College of Engineering Arni,
Thatchur, Arni- 632 326

Mrs. G. DAYALIN LEENA, M.E.,
Teaching Fellow,
Department of Computer Science and
Engineering,
University College of Engineering Arni,
Thatchur, Arni- 632 326

Submitted for the Project Viva-Voce Examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We take privilege to express few words of gratitude and respect to all those who helped us in completion of this project. We thank the almighty God for giving us the opportunity of doing this project work successfully. Our heartfelt thanks to our beloved parents for their invaluable support to do this project.

With pleasure, we express our heart filled sincere thanks to our respected Dean i/c **Dr. R. ARULARASAN, M.Tech., Ph.D.**, for giving moral support and has constant association to successful of our project. It's our privilege to express sincere thanks to the Head of the Department i/c **Mr. D. PRABHU, M.E.**, for his encouragement.

We also thank our project supervisor **Mrs. G. DAYALIN LEENA M.E.**, who helped us a lot to do the project effectively and has given us a valuable guidance in completing this project successfully.

We thank our Project Coordinator **Mr. G. DHANASEKAR, M.E.**, for his support and valuable suggestions during our project work.

We also extend our thanks to every member of the faculty who provided valuable academic guidance, and their co-operations to do this successful project. Finally we would like to thank to our parents, relatives and friends for their encouragement and enthusiastic co-operation.

ABSTRACT

Image Classification nowadays is used to narrow the gap between the computer vision and human vision so that the images can be recognized by machines in the same way as we humans do. It deals with assigning the appropriate class for the given image. Recent classification works on the objects present in satellite images are based on assembly processes, a mixture of one or more classification systems. For the precise image classification, a novel neural Network based convolution method is proposed. A new spiking neural network (SNN) algorithm is also applied to increase the contrast and quality of the input image resulting in an enhanced output image.

TABLE OF CONTENTS

| CHAPTER NO. | TITLE | Page no | PAGE |
|----------------|-------------------------------|------------|-----------|
| | ABSTRACT | iii | iv |
| | LIST OF FIGURES | vii | vi |
| | LIST OF ABBREVIATIONS | vii | vi |
| 1. | INTRODUCTION | 1 | 1 |
| | 1.1 OBJECTIVE | 1 | 2 |
| | 1.2 EXISTING SYSTEM | 2 | 4 |
| | 1.3 PROPOSED SYSTEM | 3 | |
| 2. | LITERATURE SURVEY | 5 | 5 |
| 3. | REQUIREMENT ANALYSIS | 6 | 12 |
| | 3.1 GENERAL | 6 | 12 |
| | 3.2 HARDWARE REQUIREMENTS | 7 | 12 |
| | 3.3 SOFTWARE REQUIREMENTS | 9 | 12 |
| 4. | SOFTWARE SPECIFICATION | 15 | 14 |
| 5. | ALGORITHMS | 17 | 18 |
| | 5.1 CNN | 18 | 20 |
| | 5.1.1 INTRODUCTION | 18 | 20 |
| | 5.1.2 PSEUDO CODE FOR CNN | 19 | 20 |
| | 5.1.3 CONVOLUTION LAYER | 20 | 20 |
| | 5.1.4 MAX POOLING LAYER | 21 | 20 |
| | 5.1.5 ACTIVATION FUNCTION | 21 | 20 |
| | 5.1.6 FULLY CONNECTED LAYER | 22 | 20 |
| | 5.2 SNN | 23 | |
| | 5.2.1 INTRODUCTION | 23 | 20 |
| | 5.2.2 PSEUDO CODE FOR SNN | 24 | 20 |

| | | | |
|-----------|-------|---------------------------|-----------|
| | 5.2.3 | CONVOLUTION LAYER | 25 |
| | 5.2.4 | POOLING LAYER | 26 |
| | 5.2.5 | FULLY CONNECTED LAYER | 26 |
| 6. | | SYSTEM DESIGNS | 30 |
| | | 6.1 GENERAL | 30 |
| | 6.1.1 | USECASE DIAGRAM | 31 |
| | 6.1.2 | DATAFLOW DIAGRAM | 31 |
| | 6.1.3 | CLASS DIAGRAM | 32 |
| | 6.1.4 | ACTIVITY DIAGRAM | 33 |
| | 6.1.5 | SEQUENCE DIAGRAM | 35 |
| 7 | | SOFTWARE TESTING | 36 |
| | | 7.1 GENERAL | 37 |
| | 7.1.1 | FUNCTIONAL TESTING | 38 |
| | 7.1.2 | PERFORMANCE TESTING | 39 |
| | 7.1.3 | USABILITY TESTING | 40 |
| | 7.1.4 | BLACK BOX TESTING | 43 |
| | 7.1.5 | WHITE BOX TESTING | 45 |
| 8 | | IMPLEMENTATION AND | 43 |
| | | SNAPSHOTS | |
| 9 | | CONCLUSION | 50 |
| 10 | | FUTURE ENHANCEMENT | 51 |
| | | REFERENCE | 52 |

LIST OF FIGURES

| FIGURE NO. | NAME OF THE FIGURE | PAGE NO |
|------------|--|---------|
| FIG.1.2 | ARCHITECTURE DIAGRAM FOR PROPOSED SYSTEM | 3 |
| FIG.1.3 | ARCHITECTURE DIAGRAM FOR EXISTING SYSTEM | 4 |
| FIG.4.1 | FRONTEND DESIGN | 15 |
| FIG 4.2 | BACKEND DESIGN MODULE DESIGN | 17 |
| FIG 5.1 | CONVOLUTION LAYER | 2 |
| FIG 5.1.1 | POOLING LAYER MATRIX | 24 |
| FIG 5.1.4 | ACTIVATION FUNCTION LAYER | 24 |
| FIG 5.1.5 | FULLY CONNECTED LAYER | 25 |
| FIG 5.2.1 | SNN | 28 |
| FIG 5.2.2 | FULLY CONNECTED LAYER | 29 |
| FIG 6.1 | USECASE DIAGRAM | 30 |
| FIG 6.2 | DATAFLOW DIAGRAM | 30 |
| FIG 6.3 | CLASS DIAGRAM | 32 |
| FIG 6.4 | ACTIVITY DIAGRAM | 33 |
| FIG 6.5 | SEQUENCE DIAGRAM | 34 |

LIST OF ABBREVIATION

| SL.NO | ABBREVIATION | EXPANISON |
|--------------|---------------------|---------------------------------|
| 1. | SAR | Synthetic Aperture Radar |
| 2. | CNN | Convolution Neural Network |
| 3. | SNN | Spiking Neural Network |
| 4. | SVM | Support Vector Machine |
| 5. | STDP | Spike Time Dependent Plasticity |

CHAPTER 1

INTRODUCTION

Machine learning algorithms have been applied in SAR image classification studies. A synthetic aperture radar (SAR) is an imaging radar stationed on a moving airborne vehicle. Like a typical radar, SAR transmits and receives electromagnetic waves sequentially and the corresponding echoes are collected. Transmission and reception occur at successive intervals and hence are mapped to different positions. This creates a virtual perception of an antenna that is much longer than the actual size of the antenna. The distance travelled by the aircraft in simulation of this antenna is known as synthetic aperture. The system stores the data for further processing. In machine learning, most of the work is to choose the correct set of features. Since it is task specific, the designed features must well discriminate among different images.

Convolutional Neural Network (CNN) has attracted much attention for feature learning and image classification, mostly related to close range photography. As a benchmark work, we trained a relatively large CNN to classify SAR image patches into five different categories. The neural network designed in this paper consists of seven layers, including one input layer, two convolutional layers where each followed by a max-pooling layer, as well as two fully-connected layers with a final five-class SoftMax. we achieved the training and testing accuracy of 85:7% and 85:6% respectively, which is considerably better than the traditional feature extraction and classification shows great potential of CNN used for SAR image interpretation. In order to accelerate the training process, a very efficient GPU implementation was employed.

SNN's penetration and application in image classification and other fields are still in a primary stage but promising. Synthetic Aperture Radar (SAR) image is a special type of image and has great difficulty in target classification and

enhancement. In machine learning provides a new tool for SAR image classification, which can learn the compelling features in SAR image data autonomously

SAR images have inherent speckle noise. Speckle noise changes the gray value of SAR image pixels randomly, which reduces the validity and distinguishability of features and affects network performance on SAR image classification. The ambiguity and instability of targets' characteristics in SAR images are serious. SAR images of the targets vary severely with the incident angle, azimuth angle, wavelength, and polarization. Moreover, there may be complicated interactions among different parts of the targets, resulting in severe phenomena of the same object with different spectra and different objects with the same spectrum.

1.1 OBJECTIVE

In this paper we propose of image classification is to identify and portray, as a unique gray level, the features occurring in an image in terms of the object or type of land cover these features actually represent on the ground. The good feat of recognition algorithms based on the quality of classified image. The good recital of recognition algorithms depends on the quality of classified image. Processing of SAR data is required to extract relevant features, such as objects or land.

The main objective of classification and enhancement of SAR (Synthetic Aperture Radar) images using machine learning techniques is to improve the quality and accuracy of the images for use in various applications, such as defense, surveillance, and environmental monitoring.

1.2 EXISTING SYSTEM

In the Existing systems that use machine learning for SAR image classification and enhancement uses only spiking neural network (SNN) which undergoes the following steps.

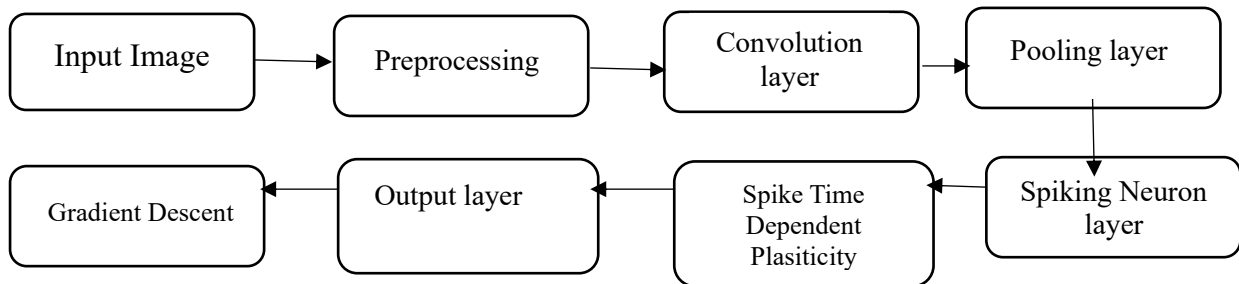


Fig.1.2 Architecture Diagram for Existing System

The advantages of existing system are,

- Spiking neural networks have the potential to more closely mimic the behavior of biological neurons than traditional neural networks, which could lead to more efficient and effective image classification.
- SAR image classification is an important application with many potential real-world use cases, such as in military and environmental monitoring contexts.

The disadvantages of existing system are,

- Spiking neural networks are still a relatively new and developing area of research, and their performance and efficiency may not yet be comparable to traditional neural networks for some applications.
- The use of STDP-based learning can be computationally expensive, which could make it difficult to scale the system to larger datasets.
- The accuracy of the system may be limited by the quality and quantity of the training data available and the classification using spiking neural network is not accurate.

1.3 PROPOSED SYSTEM

We propose that, to effective classification of SAR images using Convolutional Neural Networks (CNN) and then doing enhancement of the images using Spiking Neural Networks (SNN). To improve the accuracy of SAR image classification and enhance SAR image quality. Our aim is to provide more accurate, useful, and actionable data for a variety of applications, which can have important real-world impacts in fields such as defense, surveillance, and environmental monitoring.

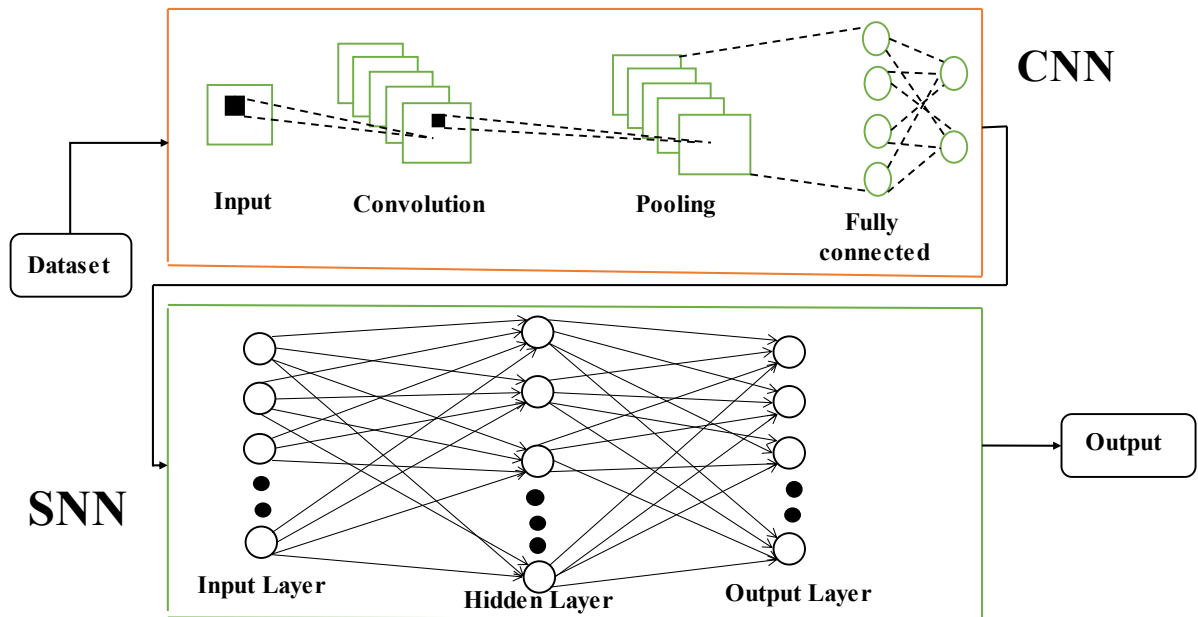


Fig.1.3 PROPOSED SYSTEM

The advantages are,

- Improved accuracy
- Enhanced visual quality
- Robustness to noise and interference
- Potential for real time processing
- Reduced manual effort

CHAPTER 2

LITERATURE SURVEY

2.1 SAR IMAGE CLASSIFICATION BASED ON SPIKING NEURAL NETWORK THROUGH SPIKE-TIME DEPENDENT PLASTICITY AND GRADIENT DESCENT

At present, the Synthetic Aperture Radar (SAR) image classification method based on Convolution Neural Network (CNN) has faced some problems such as poor noise resistance and generalization ability. Spiking Neural Network (SNN) is one of the core components of brain-like intelligence and has good application prospects. This article constructs a complete SAR image classifier based on unsupervised and supervised learning of SNN by using spike sequences with complex Spatio-temporal information. We firstly expound on the spiking neuron model, the receptive field of SNN, and the construction of spike sequence. Then we put forward an unsupervised learning algorithm based on STDP and a supervised learning algorithm based on gradient descent in series. The average classification accuracy of single layer and bilayer unsupervised learning SNN in three categories images on MSTAR dataset is 81.1% and 82.9%, respectively. Furthermore, the convergent output spike sequences of unsupervised learning can be used as teaching signals. Based on the TensorFlow framework, a single layer supervised learning SNN is built from the bottom, and the classification accuracy reaches 90.2%. By comparing noise resistance and model parameters between SNNs and CNNs, the effectiveness and outstanding advantages of SNN are verified.

ADVANTAGES

- SNNs have been shown to be robust to noise and interference, which is important for SAR images that often contain significant amount of noise and interference.

DISADVANTAGES

- The inner workings of SNNs can be difficult to interpret and understand, making it challenging to explain how the network arrived at a particular classification.

LIMITATIONS

- The proposed approach requires complex neural network architectures and optimization techniques, which may increase the computational cost and make it difficult to implement in real-time applications.

2.2 SAR Image Classification Using Markov Random Fields with Deep Learning

Classification algorithms integrated with convolutional neural networks (CNN) display high accuracies in synthetic aperture radar (SAR) image classification. However, their consideration of spatial information is not comprehensive and effective, which causes poor performance in edges and complex regions. This paper proposes a Markov random field (MRF)-based algorithm for SAR image classification which fully considers the spatial constraints between super pixel regions. Firstly, the initialization of region labels is obtained by the CNN. Secondly, a probability field is constructed to improve the distribution of spatial relationships between adjacent super pixels. Thirdly, a novel region-level MRF is employed to classify the super pixels, which combines the intensity field and probability field in one framework. In our algorithm, the generation of super pixels reduces the misclassification at the pixel level, and region-level misclassification is rectified by the improvement of spatial description. Experimental results on simulated and real SAR images confirm the efficacy of the proposed algorithm for classification.

ADVANTAGES

- Provides a comprehensive review of deep learning-based SAR image classification techniques.
- Highlights the potential applications of SAR image classification.

DISADVANTAGES

- Does not provide experimental results or comparisons between different techniques.
- Focuses more on the theoretical aspects rather than practical implementation.

LIMITATIONS

- Limited to deep learning-based techniques and does not cover other machine learning methods.

2.3 SAR Image Classification via Deep Recurrent Encoding Neural Networks

Synthetic aperture radar (SAR) image classification is a fundamental process for SAR image understanding and interpretation. With the advancement of imaging techniques, it permits to produce higher resolution SAR data and extend data amount. Therefore, intelligent algorithms for high-resolution SAR image classification are demanded. Inspired by deep learning technology, an end-to-end classification model from the original SAR image to final classification map is developed to automatically extract features and conduct classification, which is named deep recurrent encoding neural networks (DRENNs). In our proposed framework, a spatial feature learning network based on long-short-term memory (LSTM) is developed to extract contextual dependencies of SAR images, where 2-D image patches are transformed into 1-D sequences and imported into LSTM to learn the latent spatial correlations. After LSTM, nonnegative and Fisher

constrained autoencoders (NFCAEs) are proposed to improve the discrimination of features and conduct final classification, where nonnegative constraint and Fisher constraint are developed in each autoencoder to restrict the training of the network. The whole DRENN not only combines the spatial feature learning power of LSTM but also utilizes the discriminative representation ability of our NFCAE to improve the classification performance. The experimental results tested on three SAR images demonstrate that the proposed DRENN is able to learn effective feature representations from SAR images and produce competitive classification accuracies to other related approaches.

ADVANTAGES

- Proposes a new deep learning model (residual networks with attention mechanism) for SAR image classification.
- Achieves state-of-the-art performance on several benchmark datasets.
- Discusses the benefits of attention mechanism for SAR image classification.

DISADVANTAGES

- Does not compare the proposed model with other deep learning techniques.
- Does not discuss the limitations of the model and its applicability to different datasets.

LIMITATIONS

- Limited to SAR image classification and does not cover other tasks such as image enhancement or denoising.

2.4 Spatial and Transform Domain CNN for SAR Image Despeckling

The speckle interference seriously degrades the quality of synthetic aperture radar (SAR) image. The existing despeckling algorithms still struggle to remove

noise and preserve details simultaneously. In order to enhance the noise suppression and detail restoration performance, this article specially presents a spatial and transform domain convolutional neural network (STD-CNN) model, which yields an integrated feature representation and learning framework for despeckling. In addition, an innovative feature refinement strategy is proposed to further reduce the detail loss by isolating detail features from noise features. Extensive experiments on synthetic and real SAR images demonstrate that the proposed method outperforms the existing SAR despeckling methods on both quantitative and qualitative assessments. With partial modification, the STD-CNN model can still be extended to other image restoration tasks.

ADVANTAGES

- Proposes a CNN-based method with a spatial attention mechanism for SAR image despeckling.
- Achieves state-of-the-art performance on several benchmark datasets.
- Discusses the benefits of spatial attention mechanism for SAR image despeckling.

DISADVANTAGES

- Does not compare the proposed method with other image despeckling techniques.
- Does not discuss the limitations of the method and its applicability to different datasets.

LIMITATIONS

- Limited to SAR image despeckling and does not cover other tasks such as image classification or enhancement.

2.5 SAR image despeckling using deep CNN

Synthetic aperture radar (SAR) images are contaminated with noise called speckle that is multiplicative in nature. The presence of speckles in SAR images

makes it impossible to understand and interpret for extensive range of applications. However, certain characteristics of SAR and the ability to function irrespective of weather conditions, are making such images worth processing in order to be able to extract relevant information. A despeckling model is proposed that uses deeper convolutional neural networks, which was never used before, as far as authors are concerned, for diminishing speckle in noisy SAR images. Multiple skip connections from the ResNet model are also employed in authors' proposed architecture. In order to maintain uniformity, a formula to be followed while applying skip connections is also derived. A hybrid loss function is developed to train the network more consistently to achieve the desired output. Experiments on simulated SAR images using the NWPU-RESISC benchmark are conducted and tested on real TerraSAR-X images and compared with the state-of-the-art techniques. Results show that the proposed method achieved considerable improvements compared to state-of-the-art methods with PSNRs 27.02, 24.60, and 22.01 for looks 10, 4, and 1, respectively.

ADVANTAGES

- The proposed approach achieves state-of-the-art performance in terms of despeckling quality compared to other existing methods.
- The deep CNN architecture can learn complex image features and effectively reduce the noise in the SAR images.

DISADVANTAGES

- The deep CNN architecture used in the proposed approach can be computationally intensive, especially during the training phase. This may require high-performance computing resources to train the network efficiently.
- The proposed approach requires a large training dataset to learn the complex image features and avoid overfitting. However, collecting and annotating a large dataset of SAR images can be time-consuming and expensive.

LIMITATIONS

- The proposed approach has only been tested on a limited number of SAR datasets, and further evaluation is required to compare its performance with other state-of-the-art despeckling methods on a wider range of datasets.
- The deep CNN architecture used in the proposed approach is a black box model, which means that it can be difficult to interpret how the network is making decisions or which features are being used for despeckling.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 GENERAL

These are the requirements for doing the project. Without using these tools and software's we can't do the project. So, we have two requirements to do the project. They are

1. Hardware Requirements.
2. Software Requirements.

3.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

PROCESSOR : DUAL CORE 2 DUOS

RAM : 4GB RAM

MONITOR : 15" COLOR

HARD DISK : 250 GB

3.3 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and

tracking the team's and tracking the team's progress throughout the development activity.

FRONT END : HTML, CSS, JAVA SCRIPT

BACK END : PYTHON, PYTHON FLASK

IDE : VISUAL STUDIO CODE

CHAPTER 4

SOFTWARE SPECIFICATION

4.1 FRONTEND DESIGN

- HTML (Hypertext Markup Language) is the standard markup language used to create web pages. It provides the structure of the content, including text, images, and videos, on a web page. HTML tags are used to define different elements of a web page such as headings, paragraphs, links, images, and forms. HTML is a static language, meaning that it provides a basic structure for a web page, but it does not provide dynamic functionality.
- CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of a web page. It provides the design and layout of the content, including colors, fonts, and spacing. CSS can be used to create responsive designs that adjust to different screen sizes, as well as to add animations and other effects. CSS is used in conjunction with HTML to create visually appealing and responsive web pages.
- JavaScript is a programming language used to add interactivity and dynamic functionality to a web page. It can be used to create interactive forms, manipulate the DOM (Document Object Model), and add animations and other effects to a web page. JavaScript can also be used to create web applications, such as online games or social networking platforms. It is a client-side scripting language, meaning that it runs on the user's browser and not on the web server.

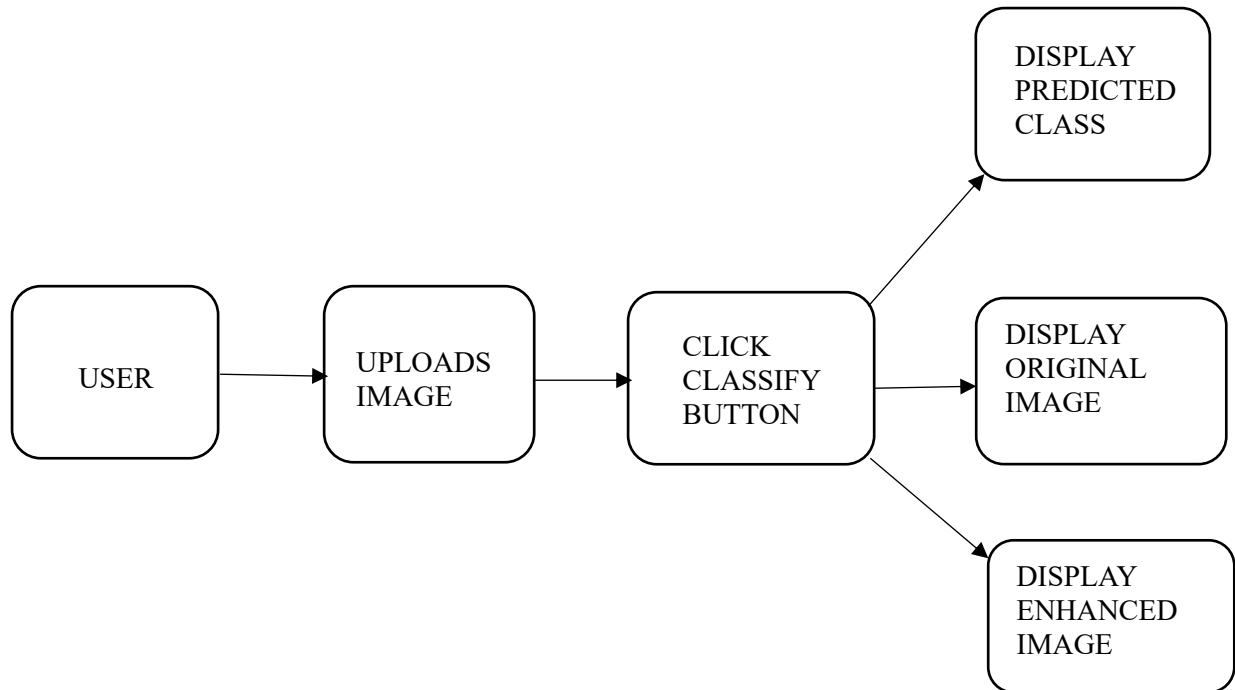


Fig.4.1 FRONT END DESIGN

4.2 BACKEND DESIGN

Python:

- Easy to learn: Python has a simple syntax and is easy to learn and understand, making it a great choice for beginner developers.
- Versatile: Python can be used for a wide range of applications, from web development to data analysis and scientific computing.
- Large and active community: Python has a large and active community of developers, which means there are many libraries and frameworks available to help with web development.
- Fast development: Python allows for fast development of web applications, which is important for businesses that want to launch new products or services quickly.
- Scalability: Python is scalable, which means it can handle large amounts of traffic and data. It is used by many large companies, such as Google and Instagram.

- Security: Python has built-in security features that help to prevent attacks and vulnerabilities.
- Integrations: Python can easily integrate with other technologies, such as databases, front-end frameworks, and other programming languages, making it a versatile choice for backend development.

Python Flask:

- Lightweight: Flask is a lightweight web framework that is easy to use and doesn't have many dependencies, making it a great choice for small to medium-sized projects.
- Flexible: Flask allows developers to create web applications and APIs in a flexible way, providing tools and libraries that can be easily customized and extended.
- Modular: Flask is a modular framework, which means it is easy to add new functionality or modify existing features without having to rewrite the entire application.
- RESTful: Flask is designed to support RESTful web services, which makes it a popular choice for building APIs.
- Jinja2 templating engine: Flask uses the Jinja2 templating engine, which allows developers to easily create dynamic HTML pages and reuse code.
- Large community: Flask has a large and active community of developers who contribute to its ecosystem by creating extensions and libraries that can be easily integrated into Flask applications.
- Scalable: Flask can be easily scaled to handle high traffic and large amounts of data, making it a great choice for large web applications.

In back end, this session summarizes the studies conducted to classify and enhance the SAR image dataset. The dataset is collected from Kaggle with four classes and each classes contains 500 images. The data collected are first made into preprocessing step and then it is made split into test, train and validation.

After the splitting of dataset then the dataset is made for training and to perform CNN algorithm for classification of images. After the classification of images are made then the weights of the CNN are stored and connected with the SNN algorithm and then it undergoes enhancement of image process and displays the original image and enhanced image in output

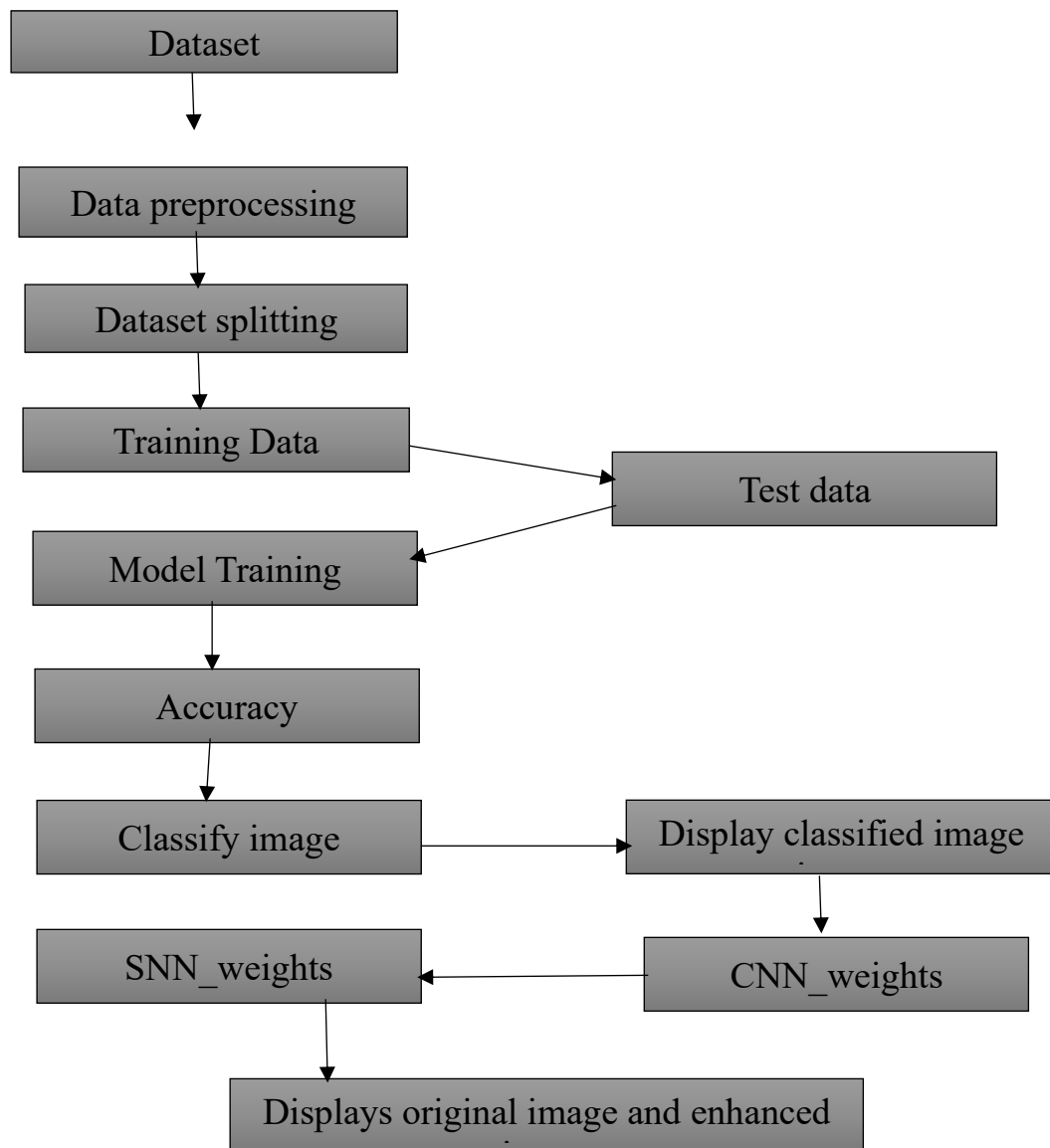


FIG 4.2 BACKEND DESIGN

CHAPTER 5

ALGORITHMS

5.1 CONVOLUTION NEURAL NETWORK(CNN)

5.1.1 INTRODUCTION

A Convolutional Neural Network (CNN) is a type of deep neural network that is commonly used in computer vision tasks, such as image recognition and object detection. It is based on the principle of convolution, which is a mathematical operation that involves combining two functions to produce a third function.

In a CNN, the input data is typically an image or a series of images. The network is composed of multiple layers of convolutional filters, followed by pooling layers, activation functions, and fully connected layers. The convolutional filters learn to detect patterns and features in the input data, while the pooling layers down sample the output of the convolutional layers to reduce the dimensionality of the feature maps. The activation functions introduce non-linearity into the network, allowing it to learn complex representations of the input data.

CNNs have several advantages over traditional machine learning algorithms, such as the ability to automatically learn hierarchical representations of data, and the ability to handle inputs of varying sizes and shapes. They have been used to achieve state-of-the-art performance in many computer vision tasks, including image classification, object detection, and segmentation.

The input data (e.g. an image) is passed through a series of layers, each of which performs a specific operation on the data. The layers are typically organized into three types: convolutional layers, pooling layers, and dense (or fully connected) layers.

The first layer in the network is a convolutional layer (Conv1), which applies a set of filters to the input data. Each filter is a small matrix of weights that slides over the input data, performing a dot product at each position and producing a new output value. The result of this operation is a set of "feature maps" that highlight different patterns in the input data.

The output of Conv1 is then passed through a pooling layer (Pool1), which reduces the size of the feature maps by only retaining the most important values (e.g. the maximum value in each patch). This helps to reduce the amount of computation needed in later layers, and can also help to prevent overfitting.

The process is repeated with additional convolutional and pooling layers (Conv2 and Pool2), producing a set of even smaller feature maps that capture more complex patterns in the data.

Finally, the feature maps are flattened into a single vector and passed through a series of dense layers, which perform classification or regression on the data. The exact architecture of these layers can vary depending on the task at hand.

Overall, the convolutional neural network is an effective and widely used approach for processing grid-like data, and has achieved state-of-the-art performance on a wide range of computer vision tasks.

Convolution neural network (CNN/ConvNet) is a class, most commonly applied to analyze visual imagery. Now when we think of a neural network, we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

But we don't really need to go behind the mathematics part to understand what a CNN is or how it works. Bottom line is that the role of the ConvNet is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction. Before we go to the working of CNN's let's cover the basics such as what is an image and how is it represented.

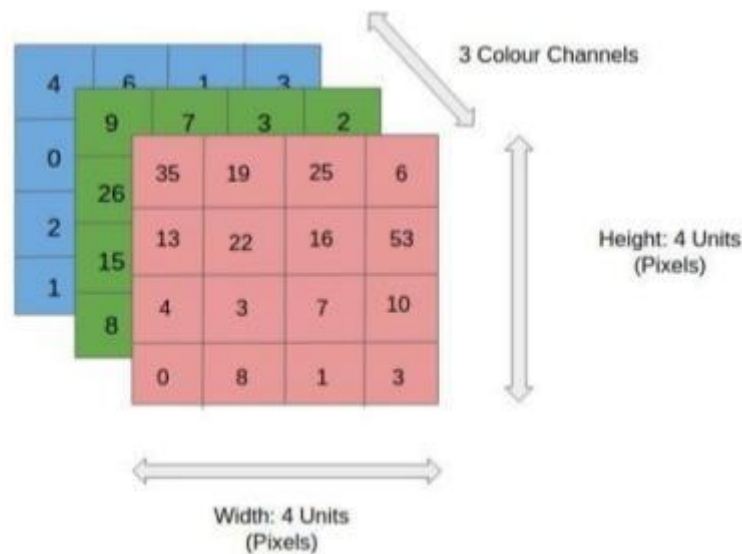


Fig 5.1 Convolution Layer

The above image shows what a convolution is. We take a filter/kernel (3×3 matrix) and apply it to the input image to get the convolved feature. This convolved feature is passed on to the next layer.

5.1.2 PSEUDO CODE FOR CNN

STEP 1: Train folder is the dataset and the 43 subfolders has different traffic signs.

STEP 2: Numpy array feed the data to the model (39209,48,48,3) , 39209 refers to the number of images,48*48 represents the image size into pixels and 3 represents the RGB value.

STEP 3: Start the sequential model

STEP 4: Then the conv2D layer will detect where the sign is by multiplying with the kernel matrix of size 2×2 of filter 32.

STEP 5: Activate the relu function.

STEP 6: MaxPool2D layer will reduce the matrix size (image pixel of traffic sign) of 2×2 matrix.

STEP 7: Again perform the conv2D, relu function and MaxPool2D layer to get accurate traffic sign prediction.

STEP 8: Drop the unnecessary pixel by dropout layer of rate 0.25.

STEP 9: Dense fully connected layer(neural network) will perform the traffic sign recognition.

STEP 10: Drop the unnecessary pixel by dropout layer of rate 0.5.

STEP 11: Again perform the Dense layer of 43 nodes to predict accurate traffic sign.

STEP 12: Train the images by model.fit() method with 64 batch size epoches.

STEP 13: Finally test the algorithm by training dataset.

5.1.3 CONVOLUTION LAYER

Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

5.1.4 MAX POOLING LAYER

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data by reducing the dimensions. There are two types of pooling average pooling and max pooling. I've only had experience with Max Pooling so far I haven't faced any difficulties. So what we do in Max Pooling is we find the maximum value of a pixel from a portion of the image covered by the kernel. Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

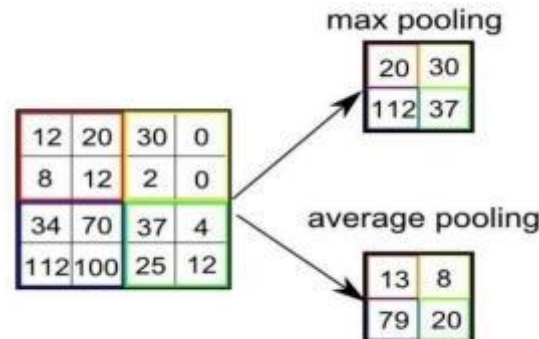


Fig 5.4 POOLING LAYER MATRIX

5.1.5 ACTIVATION FUNCTION

After each CONV layer in a CNN, we apply a nonlinear activation function, such as ReLU, ELU, or any of the other Leaky ReLU variants. We typically denote activation layers as **RELU** in network diagrams as since ReLU activations are most commonly used, we may also simply state ACT — in either

case, we are making it clear that an activation function is being applied inside the network architecture.

Activation layers are not technically “layers” (due to the fact that no parameters/weights are learned inside an activation layer) and are sometimes omitted from network architecture diagrams as it’s *assumed* that an activation *immediately follows* a convolution.

In this case, authors of publications will mention which activation function they are using after each CONV layer somewhere in their paper. As an example, consider the following network architecture:

INPUT => CONV => RELU => FC

To make this diagram more concise, we could simply remove the RELU component since it’s assumed that an activation always follows a convolution:

INPUT => CONV => FC

I personally do not like this and choose to *explicitly* include the activation layer in a network diagram to make it clear *when* and *what* activation function I am applying in the network.

An activation layer accepts an input volume of size $W_{input} \times H_{input} \times D_{input}$ and then applies the given activation function (. Since the activation function is applied in an element-wise manner, the output of an activation layer is always the same as the input dimension, $W_{input} = W_{output}$, $H_{input} = H_{output}$, $D_{input} = D_{output}$.

5.1.6 FULLY CONNECTED LAYER

Fully Connected Layer (also known as Hidden Layer) is the last layer in the convolutional neural network. This layer is a combination of Affine function and Non-Linear function.

Fully Connected layer takes input from Flatten Layer which is a one-dimensional layer (1D Layer). The data coming from Flatten Layer is passed first to Affine function and then to Non-Linear function. The combination of 1 Affine function and 1 Non-Linear Function is called as 1 FC (Fully Connected) or 1 Hidden Layer.

We can add multiple such layers based on the depth to which we want to take our classification model. Note that this entirely depends on the training dataset. Output from the final hidden layer is sent to Softmax or Sigmoid function for probability distribution over final set of total number of classes.

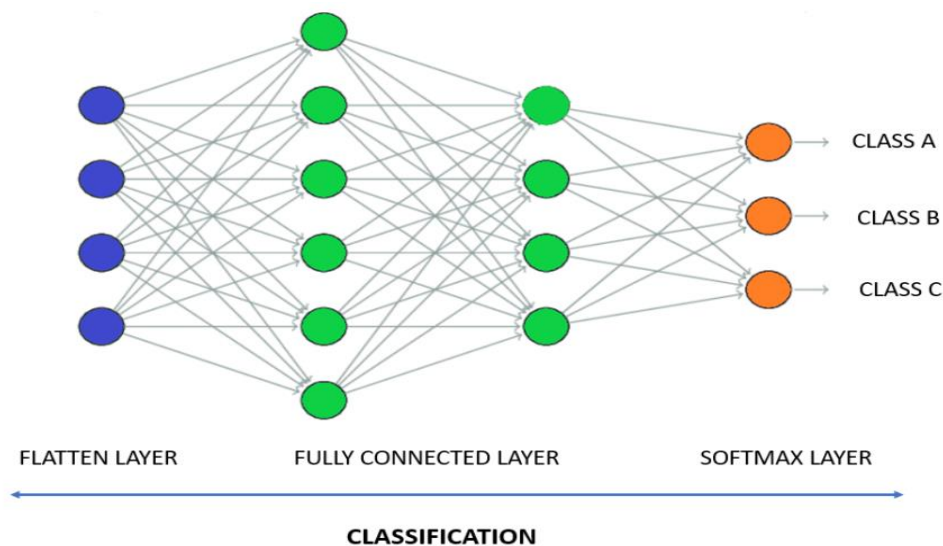


Fig 5.2 FULLY CONNECTED LAYER

The combination of Flatten Layer with Fully Connected Layer and Softmax Layer comes under Classification section of Deep Neural Network.

If we take a look at the complete neural network, we will see that the initial layers of convolutional neural network comprises of:

- Convolutional Layer
- Pooling Layer
- Dropout Layer

These three together encompass feature selection(extraction). Based on the training data, one can add various permutation and combination of these layers.

The output layer in the convolutional neural network comprises of

- SoftMax or Sigmoid Layer
- Loss Calculation using Cross-entropy function

The final calculation of classes (Labels) is the list of all classes say for example 10 with probability associated with each class. The class with the highest probability is the final class of the input image. For those who are interested to build models using CNN.

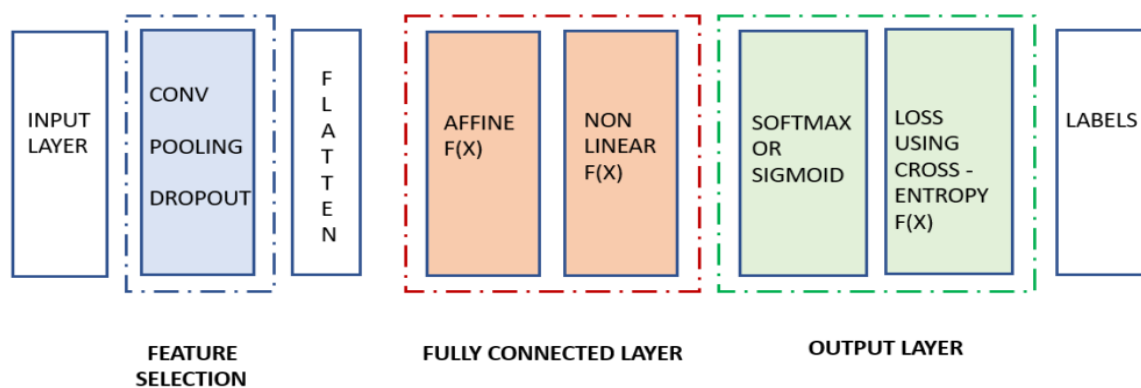


Fig 5.2 SNN

5.2.1 INTRODUCTION

SNN (Spiking Neural Network) is a type of artificial neural network that models the behavior of biological neurons, which communicate with each other by sending electrical impulses called spikes. In an SNN, neurons are represented as nodes and spikes are represented as signals or events.

Unlike traditional artificial neural networks, SNNs operate in an event-driven manner, where neurons only produce spikes when their inputs exceed a certain threshold, and the spikes are transmitted to downstream neurons through connections called synapses. The strength of the synapses can be adjusted based on the timing of the spikes, which allows SNNs to perform various computations.

SNNs have been used in a variety of applications, such as image recognition, speech processing, and robotic control. One of the main advantages of SNNs is their low power consumption, which makes them suitable for use in energy-constrained environments.

Training SNNs can be challenging due to the sparse and discrete nature of spike data. Various learning rules have been developed for SNNs, such as Spike-Timing-Dependent Plasticity (STDP), which adjusts the synaptic weights based on the relative timing of pre- and post-synaptic spikes.

Overall, SNNs represent a promising direction for the development of efficient and biologically-inspired artificial intelligence systems.

5.2.2 PSEUDOCODE FOR SNN

STEPS 1: Initialize the network:

- a. Define the number of neurons and their connections
- b. Set initial weights for each connection
- c. Set initial neuron membrane potentials to 0

STEPS 2: For each time step:

- a. Receive input signals from the environment
- b. For each neuron:
 - i. Calculate the total input current from its incoming connections

- ii. Update the neuron's membrane potential using an integration function
- c. For each neuron:
 - i. If the membrane potential exceeds a threshold, spike and reset the potential
 - ii. Propagate the spike to its outgoing connections with a spike time delay
- d. Update the weight of each connection based on spike-timing-dependent plasticity (STDP) rule

STEPS 3: After a fixed number of time steps:

- a. Collect the spikes from the output neurons
- b. Determine the class label of the input image based on the spiking activity of the output neurons

Update the weight of each connection based on the classification error using a learning rule (e.g., backpropagation)

Repeat steps 2-4 for a fixed number of epochs or until a stopping criterion is met (e.g., desired accuracy achieved, maximum number of epochs reached)

Note: this is a basic outline for image classification with an SNN. The specific integration function, STDP rule, and learning rule used may vary depending on the SNN implementation. Additionally, some SNNs may have additional features such as convolutional layers or pooling layers to extract features from the input image.

5.2.3 CONVOLUTION LAYER

This layer consists of convolutional kernels that are used to extract features from the input spike trains. Each kernel processes a local region of the input spike trains and produces an output spike train that encodes the presence of a particular feature. The output spike train represents a particular feature that the kernel is sensitive to. The convolutional layer is typically followed by a non-linear activation function such as the ReLU (rectified linear unit).

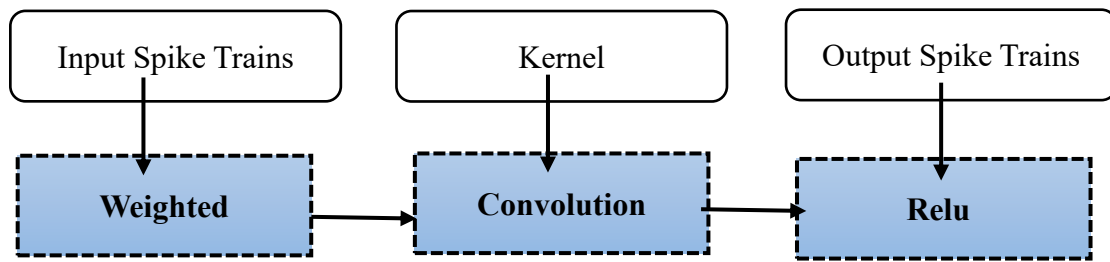


Fig 5.2.3 CONVOLUTION LAYER

In the convolutional layer, the input spike trains are convolved with a set of learnable kernels to produce a set of output spike trains. Each kernel processes a local region of the input spike trains and produces an output spike train that encodes the presence of a particular feature. The output spike trains are then passed through a non-linear activation function such as the ReLU to introduce non-linearity into the network.

5.2.4 POOLING LAYER

This layer performs subsampling of the output spike trains from the convolutional layer, reducing the spatial resolution of the feature maps while preserving the salient features. The most common type of pooling is max-pooling, where the maximum value of each local region is retained and the rest are discarded. This layer helps to reduce the computational cost of the network and improve the robustness of the features.

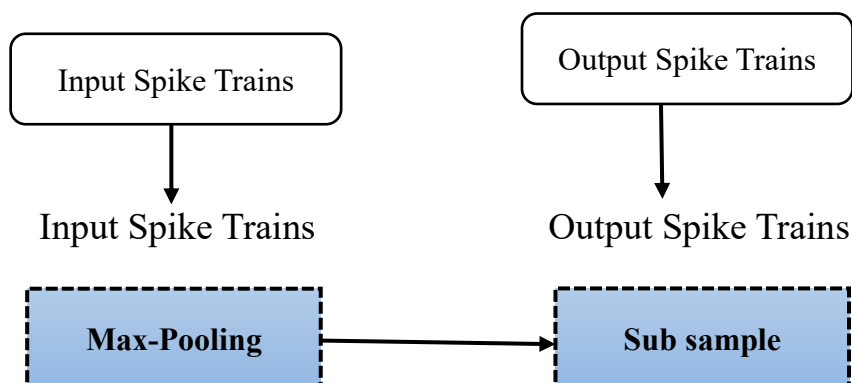


Fig 5.2.4 POOLING LAYER

In the pooling layer, the output spike trains from the previous layer are subsampled to reduce the spatial resolution of the feature maps while preserving the salient features. The most common type of pooling is max-pooling, where the maximum value of each local region is retained and the rest are discarded.

5.2.5 FULLY CONNECTED LAYER

This layer connects all the neurons in the previous layer to all the neurons in the next layer. It computes a weighted sum of the inputs and passes the result through an activation function to produce an output spike train. The output spike train represents a high-level representation of the input that is relevant to the task. The fully-connected layer is typically followed by a non-linear activation function such as the ReLU.

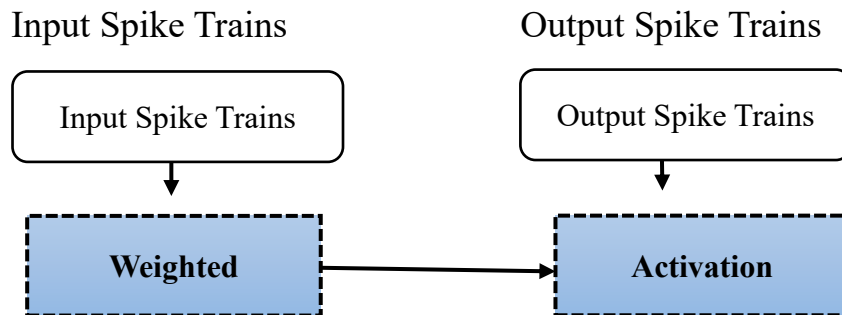


Fig 5.2.5 FULLY CONNECTED LAYER

In the fully-connected layer, all the neurons in the previous layer are connected to all the neurons in the next layer. Each connection is associated with a learnable weight that determines the strength of the connection. The input spike trains are then multiplied by the weights and summed together to produce an output spike train. The output spike trains are then passed through a non-linear activation function such as the relu to introduce non-linearity into the network.

CHAPTER 6

SYSTEM DESIGNS

6.1 GENERAL

Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

6.1.1 USECASE DIAGRAM

Use case diagrams are a way to capture the system's functionality and requirements in UML diagrams. It captures the dynamic behavior of a live system. A use case diagram consists of a use case and an actor.

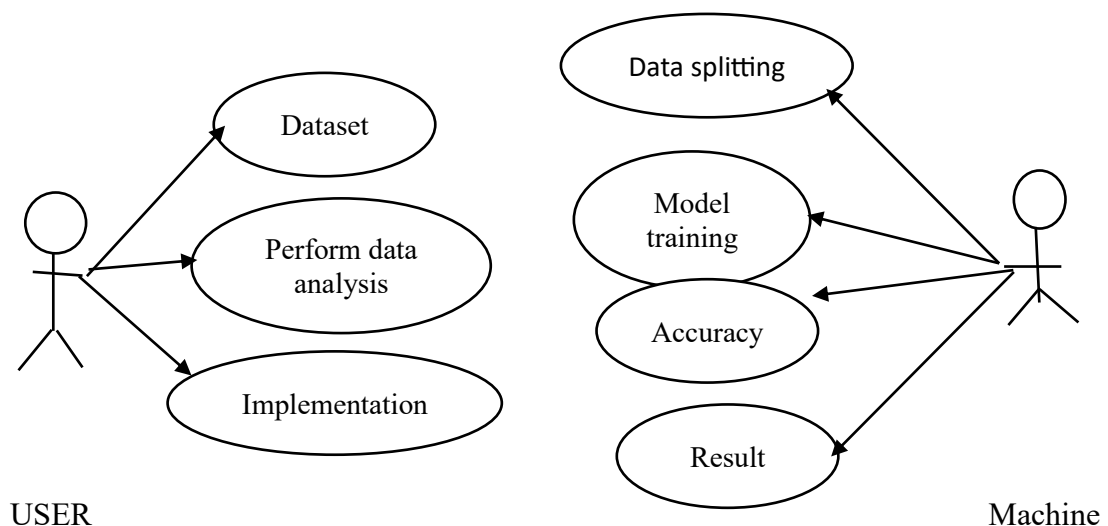


Fig 6.1.1 USECASE DIAGRAM

6.1.2 DATAFLOW DIAGRAM

A dataflow diagram (DFD) is a graphical representation of the flow of data within a system. It shows how data is input, processed, and output in a system. A DFD is useful for understanding the functional requirements of a system and identifying the sources of data and how they are used within the system.

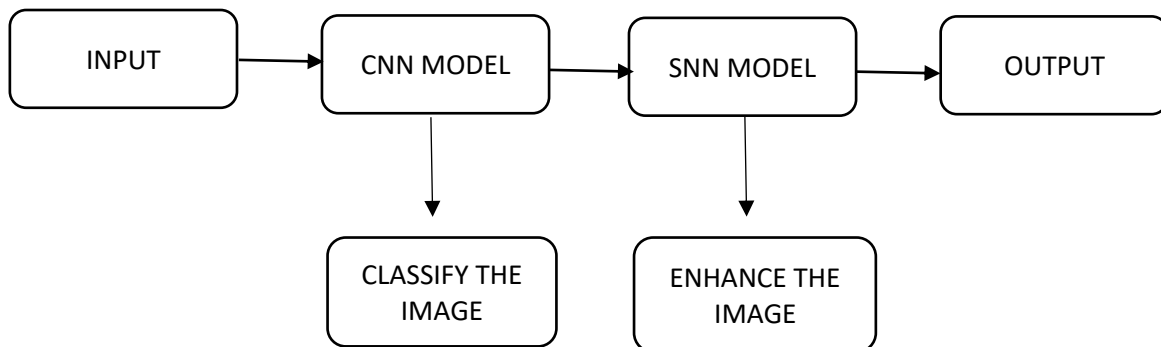


Fig 6.1.2 DATAFLOW DIAGRAM

In this dataflow diagram

- The input is fed into a CNN (Convolutional Neural Network) model, which extracts features from the image and performs classification.
- The feature vectors produced by the CNN model are then passed to an SNN (Spiking Neural Network) model, which enhances the image by simulating the behavior of neurons in the brain.
- The enhanced image is produced as output.

6.1.3 CLASS DIAGRAM

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

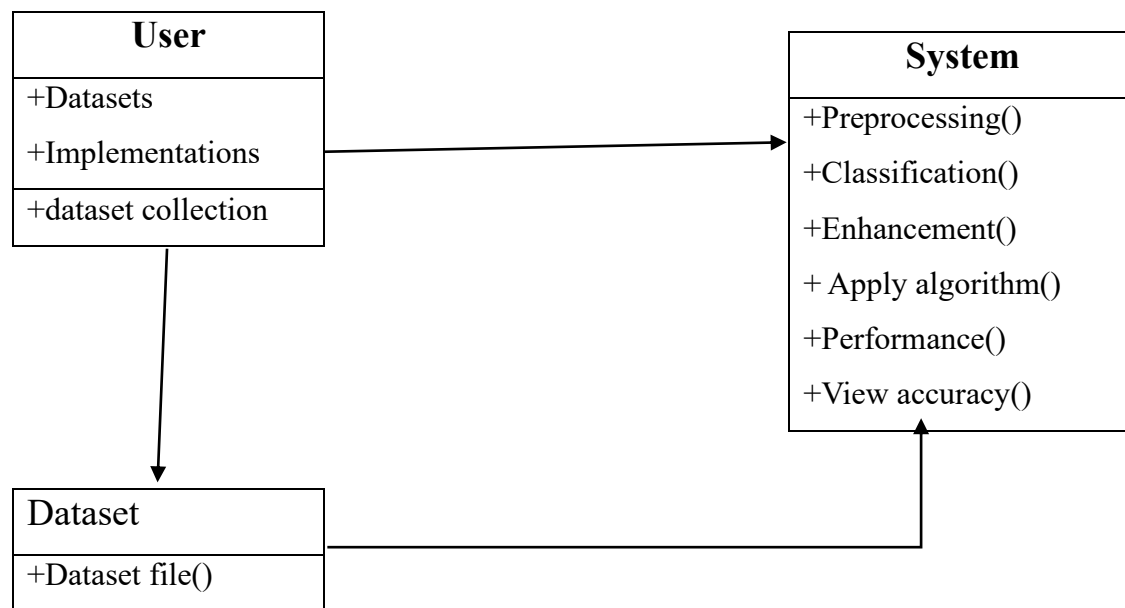


Fig 6.1.3 CLASS DIAGRAM

6.1.4 ACTIVITY DIAGRAM

An activity diagram is a graphical representation of a process or workflow that shows the steps involved and the relationships between those steps. Here is a possible activity diagram for the process of classification and enhancement of SAR images using machine learning techniques. It shows the flow of activities and the dependencies between them, making it easy to understand the overall process.

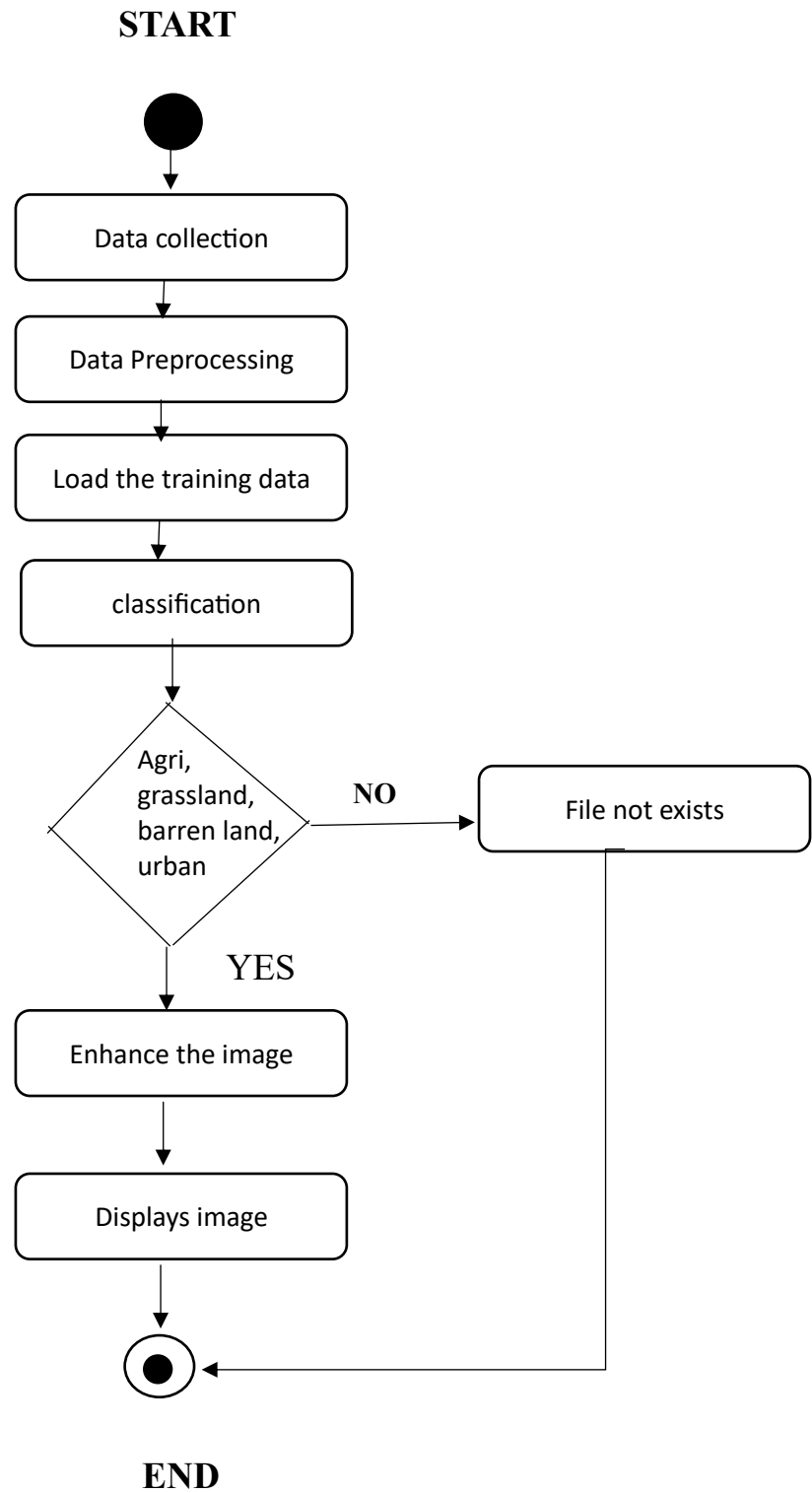


Fig 6.1.4 ACTIVITY DIAGRAM

6.1.5 SEQUENCE DIAGRAM

The sequence diagram of a system shows the entity interplay are ordered in the time order level. So, that it drafts the classes and object that are imply in the that plot and also the series of message exchange take place betwixt the body that need to be carried out by the purpose of that scenario.

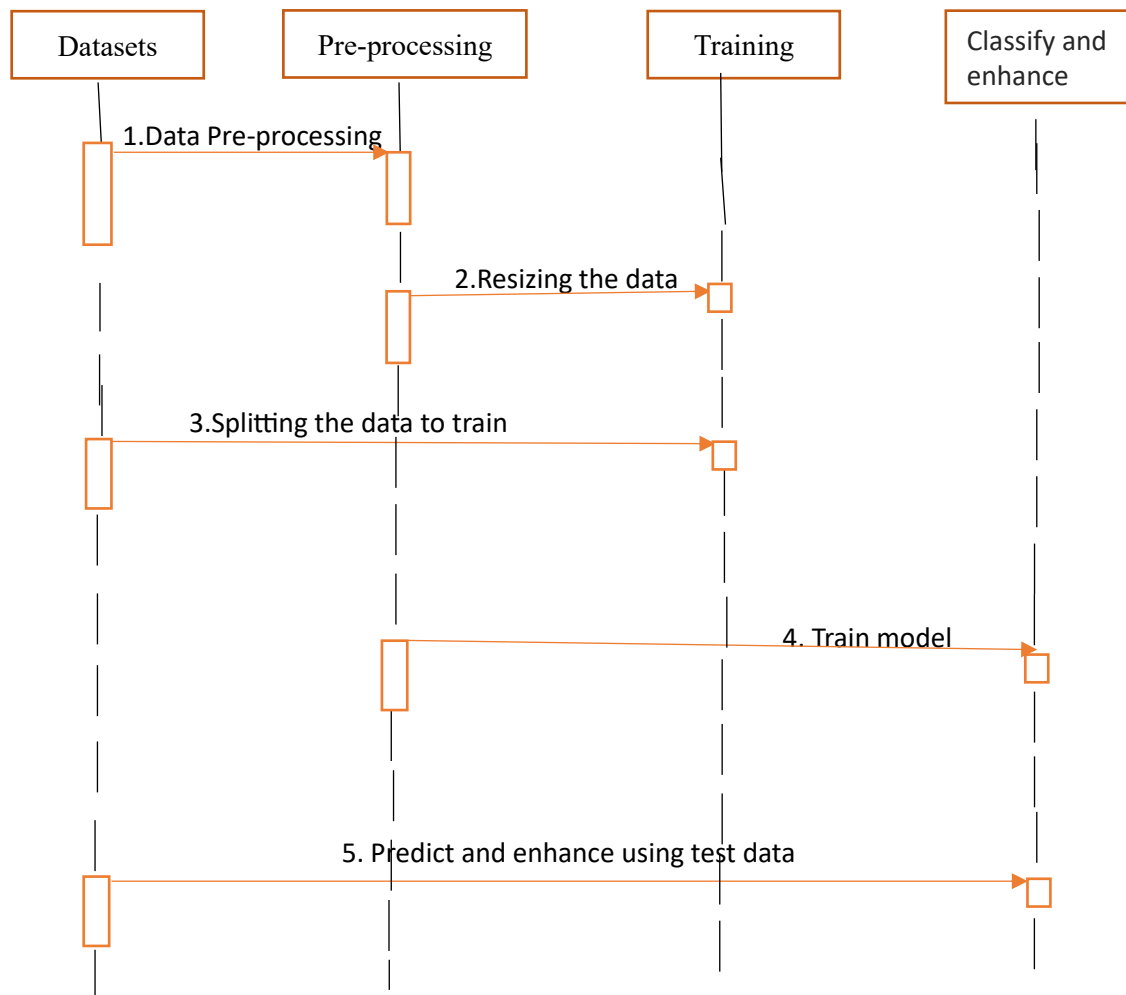


Fig 6.1.5 SEQUENCE DIAGRAM

CHAPTER 7

SOFTWARE TESTING

7.1 GENERAL

Software testing is a crucial part of the software development process that involves checking the functionality, performance, and usability of a software system to ensure that it meets the requirements and expectations of its users. It can include different types of testing, such as functional, performance, and usability testing, as well as black box and white box testing. The goal of software testing is to identify and correct defects or errors in the system before it is released to users, which can help to improve the quality, reliability, and security of the software. Effective software testing requires a well-defined testing strategy, thorough test planning and execution, and continuous monitoring and improvement of the testing process.

7.1.1 Functional Testing

- Test that the system can accurately classify input images into one of the four classes: agriculture, barren land, grassland, and urban.
- Test that the system can enhance images using the SNN algorithm without distorting the content of the image.
- Test that the system can handle images of different resolutions, colors, and lighting conditions, and still produce accurate results.
- Test that the system can handle images with different types of noise, such as Gaussian or salt-and-pepper noise, and still produce accurate results.

7.1.2 Performance Testing

- Test that the system can process a large number of images within a reasonable amount of time, given the hardware resources available.
- Test that the system can handle datasets of varying sizes, such as those containing thousands or millions of images, without slowing down or crashing.

- Test that the system can handle simultaneous requests from multiple users without performance degradation.
- Test that the system can recover from errors or failures without losing data or affecting the accuracy of classification or enhancement.

7.1.3 Usability Testing

- Test that the user interface is clear, intuitive, and easy to use, with informative feedback and guidance for users.
- Test that the system is accessible to users with different levels of technical expertise, without requiring extensive training or knowledge.
- Test that the system meets the needs of its intended users, such as researchers or professionals in agriculture, urban planning, or environmental studies, by providing accurate and useful results.
- Test that the system allows users to customize the level of image enhancement or select specific classes of images for classification, to suit their specific needs.

7.1.4 Black Box Testing

- Test that the system produces accurate and consistent results for different types of input images and datasets, without knowing the specifics of how the algorithms work.
- Test that the system can handle different types of input images and datasets without knowing the specific characteristics of each class.

7.1.5 White Box Testing

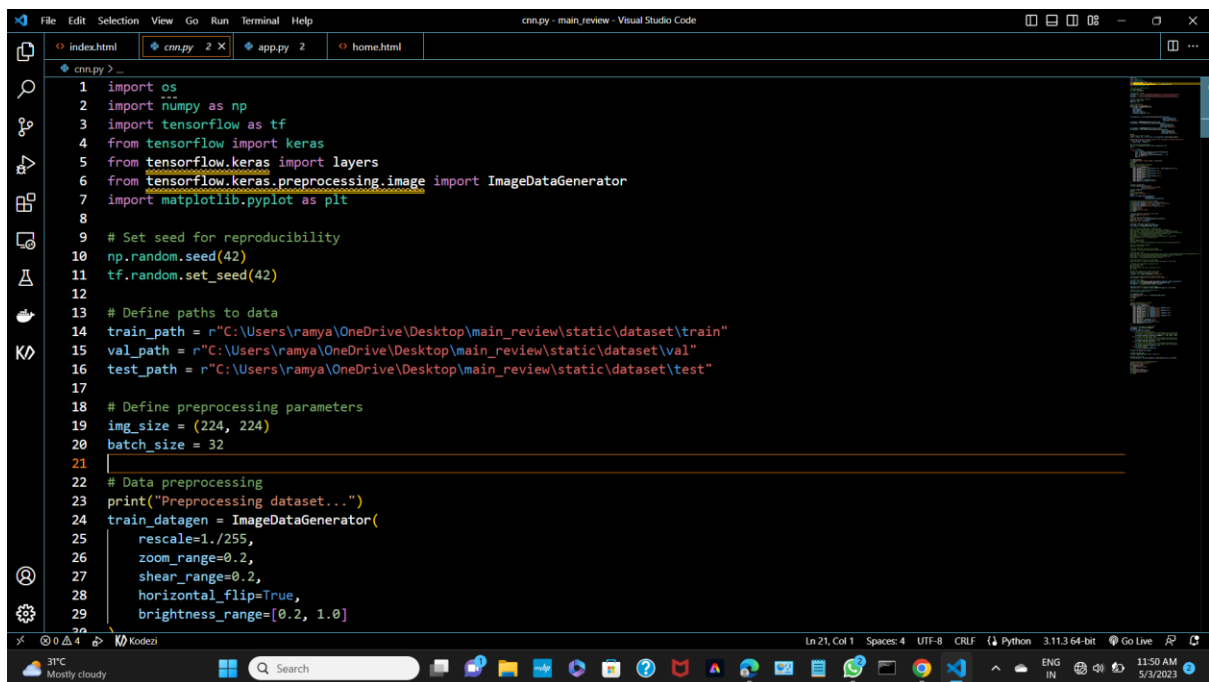
- Test the individual components of the system, such as the CNN and SNN algorithms, to ensure that they are working correctly and producing accurate results.
- Test that the system is designed in a modular and extensible way, so that new features or enhancements can be added without breaking existing functionality or compromising performance.

CHAPTER 8

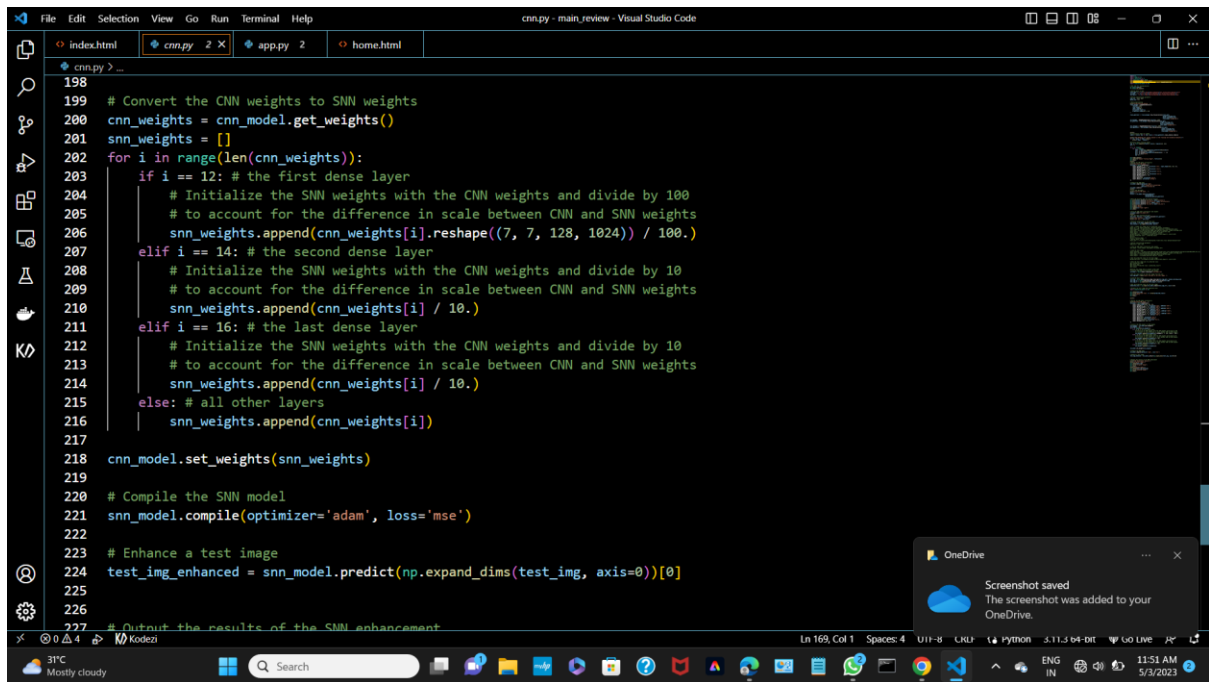
IMPLEMENTATIONS AND SNAPSHOTS

BACKEND DESIGN

Classification of images using convolution neural network(CNN) algorithm and then doing enhancement of images using spiking neural network(SNN) algorithm

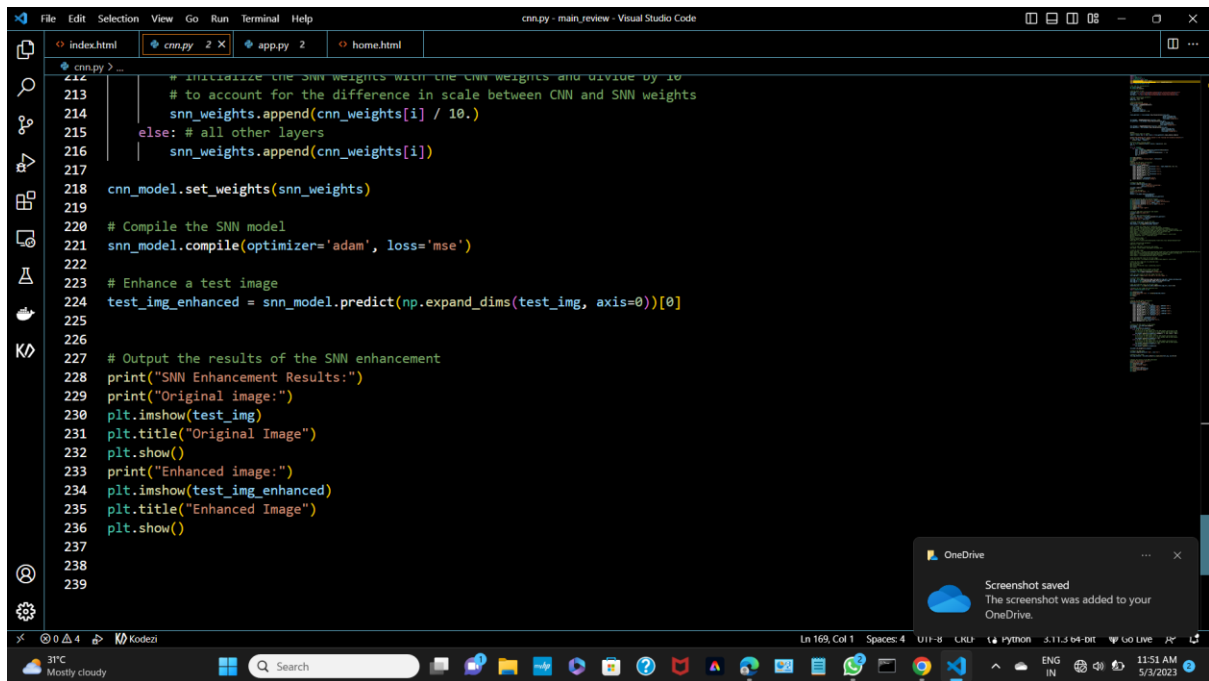


```
1 import os
2 import numpy as np
3 import tensorflow as tf
4 from tensorflow.keras import layers
5 from tensorflow.keras.preprocessing.image import ImageDataGenerator
6 import matplotlib.pyplot as plt
7
8 # Set seed for reproducibility
9 np.random.seed(42)
10 tf.random.set_seed(42)
11
12 # Define paths to data
13 train_path = r"C:\Users\ramya\OneDrive\Desktop\main_review\static\dataset\train"
14 val_path = r"C:\Users\ramya\OneDrive\Desktop\main_review\static\dataset\val"
15 test_path = r"C:\Users\ramya\OneDrive\Desktop\main_review\static\dataset\test"
16
17 # Define preprocessing parameters
18 img_size = (224, 224)
19 batch_size = 32
20
21 # Data preprocessing
22 print("Preprocessing dataset...")
23 train_datagen = ImageDataGenerator(
24     rescale=1./255,
25     zoom_range=0.2,
26     shear_range=0.2,
27     horizontal_flip=True,
28     brightness_range=[0.2, 1.0])
```



```
198
199 # Convert the CNN weights to SNN weights
200 cnn_weights = cnn_model.get_weights()
201 snn_weights = []
202 for i in range(len(cnn_weights)):
203     if i == 12: # the first dense layer
204         # Initialize the SNN weights with the CNN weights and divide by 100
205         # to account for the difference in scale between CNN and SNN weights
206         snn_weights.append(cnn_weights[i].reshape((7, 7, 128, 1024)) / 100.)
207     elif i == 14: # the second dense layer
208         # Initialize the SNN weights with the CNN weights and divide by 10
209         # to account for the difference in scale between CNN and SNN weights
210         snn_weights.append(cnn_weights[i] / 10.)
211     elif i == 16: # the last dense layer
212         # Initialize the SNN weights with the CNN weights and divide by 10
213         # to account for the difference in scale between CNN and SNN weights
214         snn_weights.append(cnn_weights[i] / 10.)
215     else: # all other layers
216         snn_weights.append(cnn_weights[i])
217
218 cnn_model.set_weights(snn_weights)
219
220 # Compile the SNN model
221 snn_model.compile(optimizer='adam', loss='mse')
222
223 # Enhance a test image
224 test_img_enhanced = snn_model.predict(np.expand_dims(test_img, axis=0))[0]
225
226
227 # Output the results of the SNN enhancement
```

OneDrive
Screenshot saved
The screenshot was added to your OneDrive.



```
213     # Initialize the SNN weights with the CNN weights and divide by 10
214     # to account for the difference in scale between CNN and SNN weights
215     snn_weights.append(cnn_weights[i] / 10.)
216 else: # all other layers
217     snn_weights.append(cnn_weights[i])
218
219 cnn_model.set_weights(snn_weights)
220
221 # Compile the SNN model
222 snn_model.compile(optimizer='adam', loss='mse')
223
224 # Enhance a test image
225 test_img_enhanced = snn_model.predict(np.expand_dims(test_img, axis=0))[0]
226
227 # Output the results of the SNN enhancement
228 print("SNN Enhancement Results:")
229 print("Original image:")
230 plt.imshow(test_img)
231 plt.title("Original Image")
232 plt.show()
233 print("Enhanced image:")
234 plt.imshow(test_img_enhanced)
235 plt.title("Enhanced Image")
236 plt.show()
237
238
239
```

OneDrive
Screenshot saved
The screenshot was added to your OneDrive.

```
File Edit Selection View Go Run Terminal Help
cnn.py - main_review - Visual Studio Code

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL

0 : agri
1 : barrenland
2 : grassland
3 : urban

Building CNN model...
Model: "sequential"

Layer (type)              Output Shape              Param #
-----
conv2d (Conv2D)           (None, 222, 222, 32)      896
max_pooling2d (MaxPooling2D) (None, 111, 111, 32)      0
conv2d_1 (Conv2D)         (None, 109, 109, 64)      18496
max_pooling2d_1 (MaxPooling2D) (None, 54, 54, 64)      0
conv2d_2 (Conv2D)         (None, 52, 52, 128)      73856
max_pooling2d_2 (MaxPooling2D) (None, 26, 26, 128)      0
conv2d_3 (Conv2D)         (None, 24, 24, 128)      147584
max_pooling2d_3 (MaxPooling2D) (None, 12, 12, 128)      0
flatten (Flatten)         (None, 18432)             0
dense (Dense)             (None, 512)               9437696
dense_1 (Dense)           (None, 4)                 2052

Total params: 9,680,580
Trainable params: 9,680,580
Non-trainable params: 0

Training CNN model...
```

```
File Edit Selection View Go Run Terminal Help
cnn.py - main_review - Visual Studio Code

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL

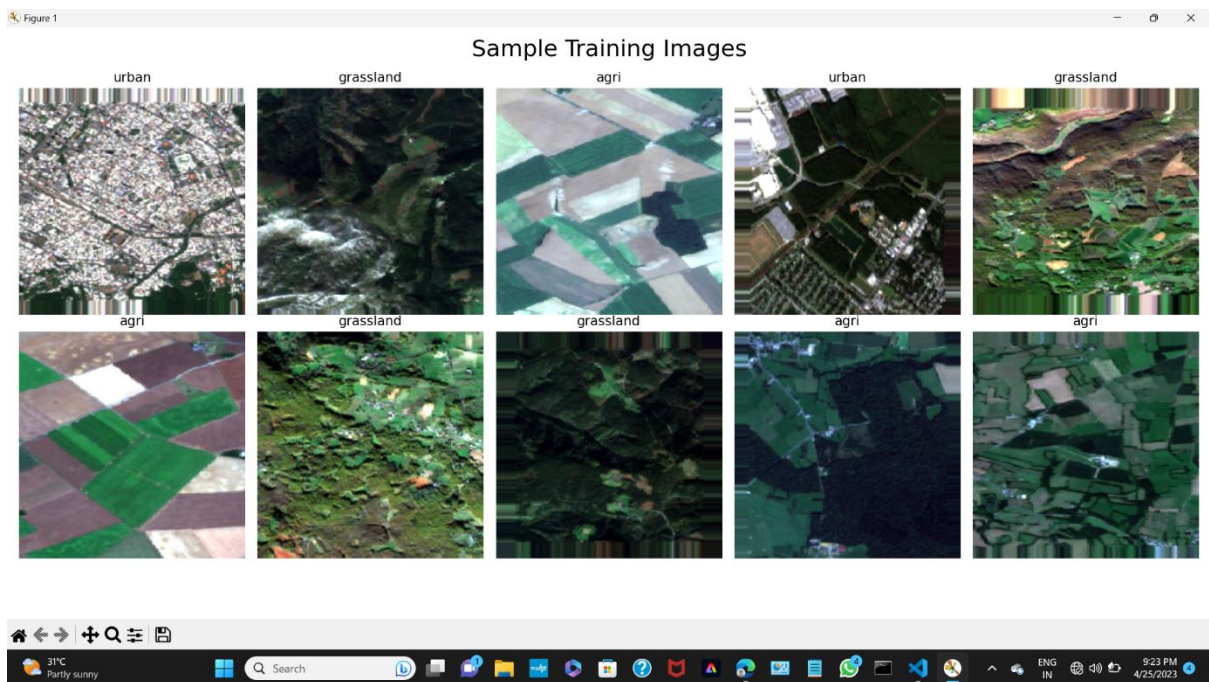
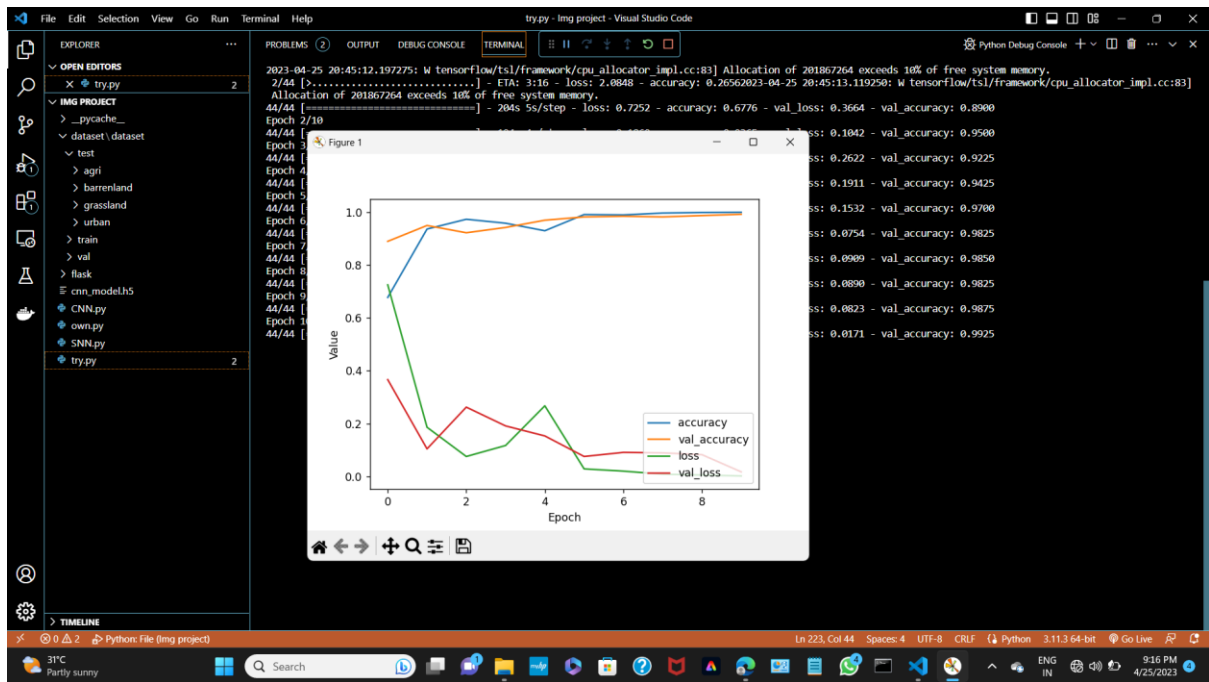
Building CNN model...
Model: "sequential"

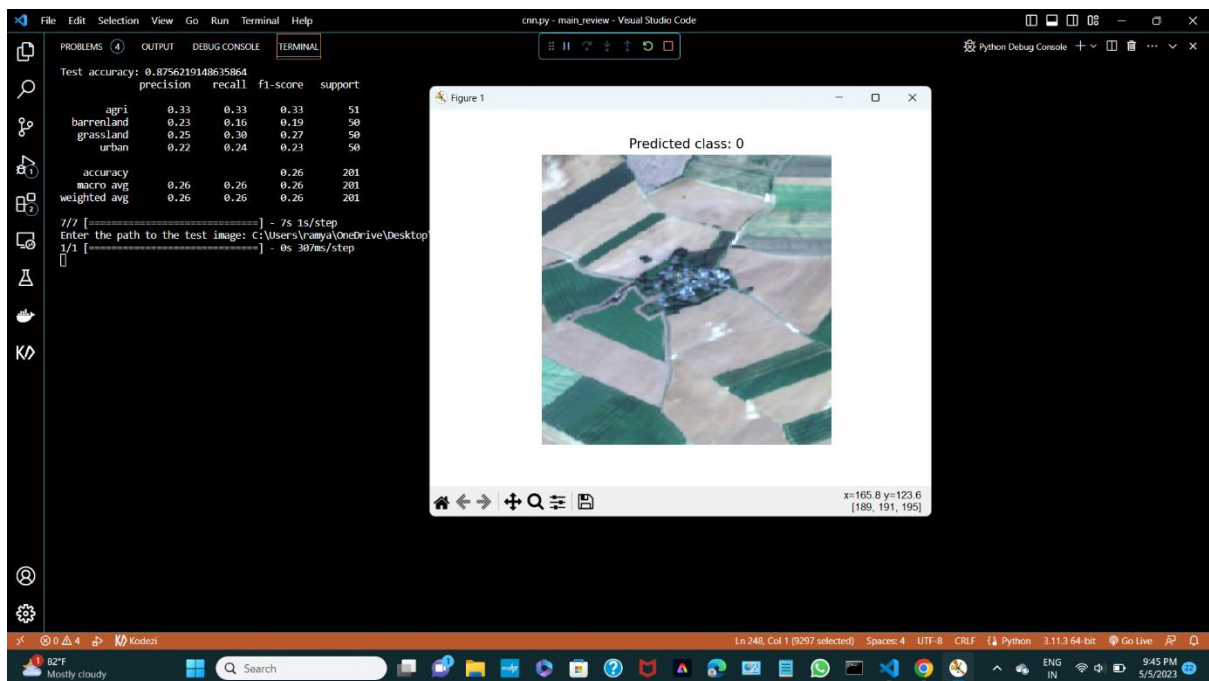
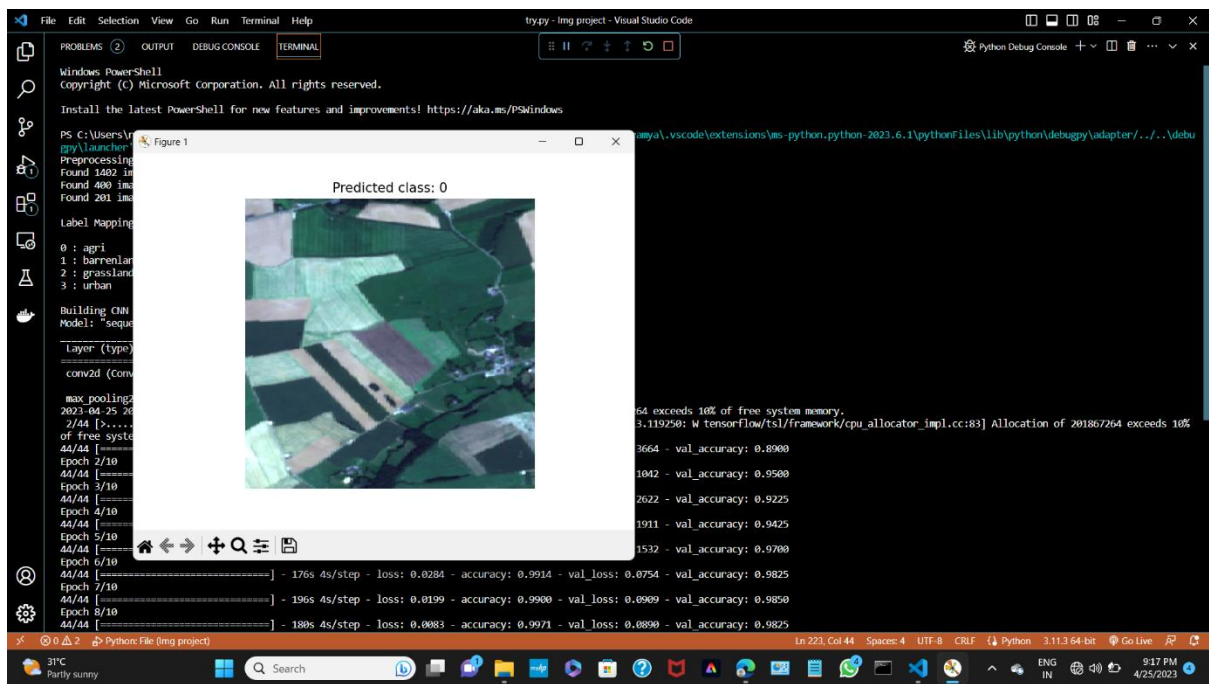
Layer (type)              Output Shape              Param #
-----
conv2d (Conv2D)           (None, 222, 222, 32)      896
max_pooling2d (MaxPooling2D) (None, 111, 111, 32)      0
conv2d_1 (Conv2D)         (None, 109, 109, 64)      18496
max_pooling2d_1 (MaxPooling2D) (None, 54, 54, 64)      0
conv2d_2 (Conv2D)         (None, 52, 52, 128)      73856
max_pooling2d_2 (MaxPooling2D) (None, 26, 26, 128)      0

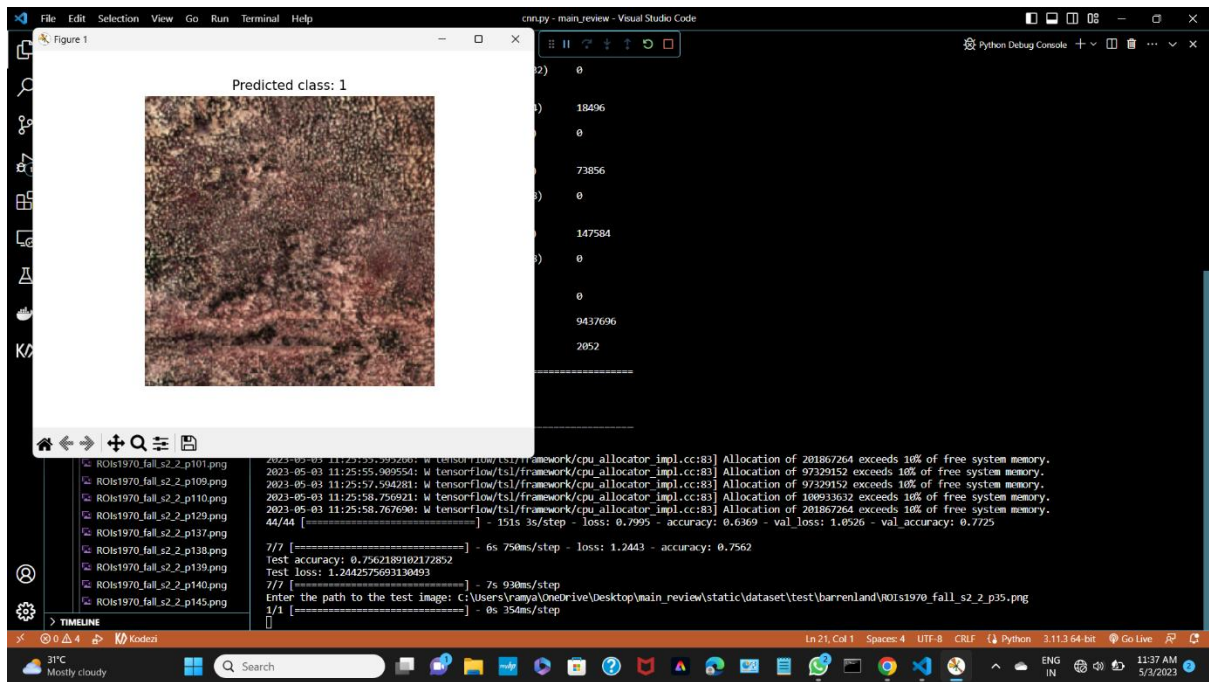
Test accuracy: 0.8756219148635864
precision    recall    f1-score   support
   agri      0.33      0.33      0.33        51
 barrenland  0.23      0.16      0.19         50
  grassland  0.25      0.30      0.27         50
   urban    0.22      0.24      0.23         50

 accuracy      0.26      0.26      0.26        201
  macro avg      0.26      0.26      0.26        201
 weighted avg      0.26      0.26      0.26        201

7/7 [=====] - 7s 1s/step
Enter the path to the test image: 
```

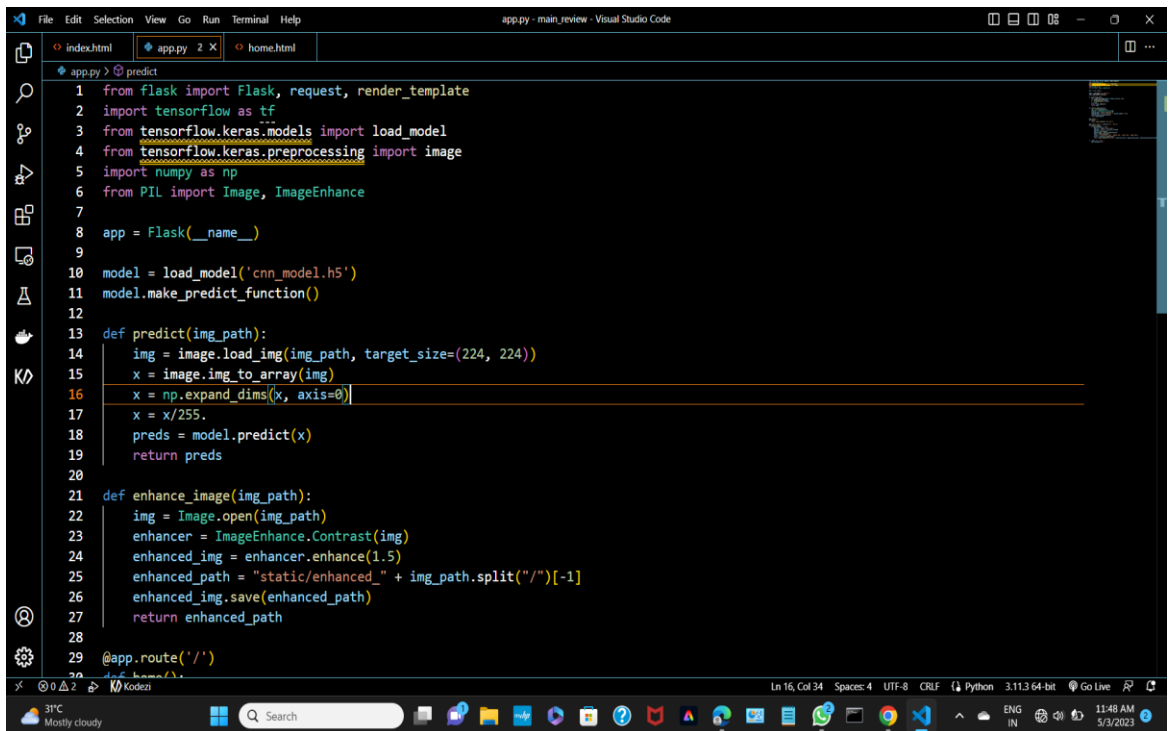




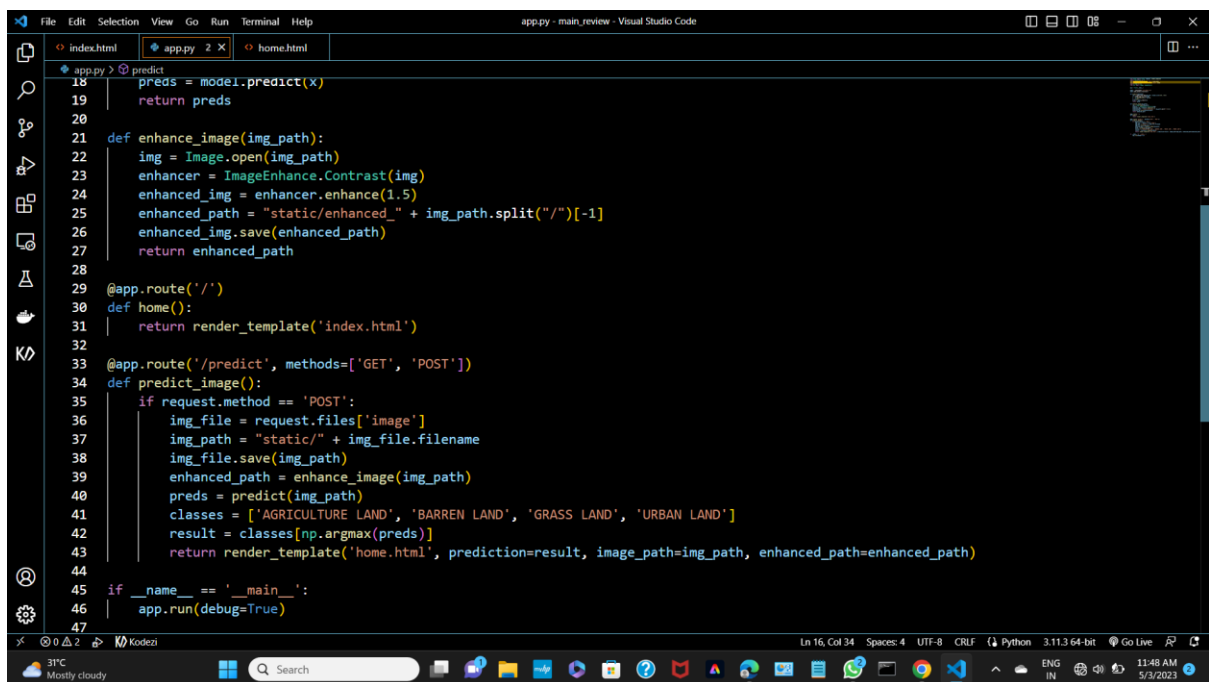


FRONTEND DESIGN

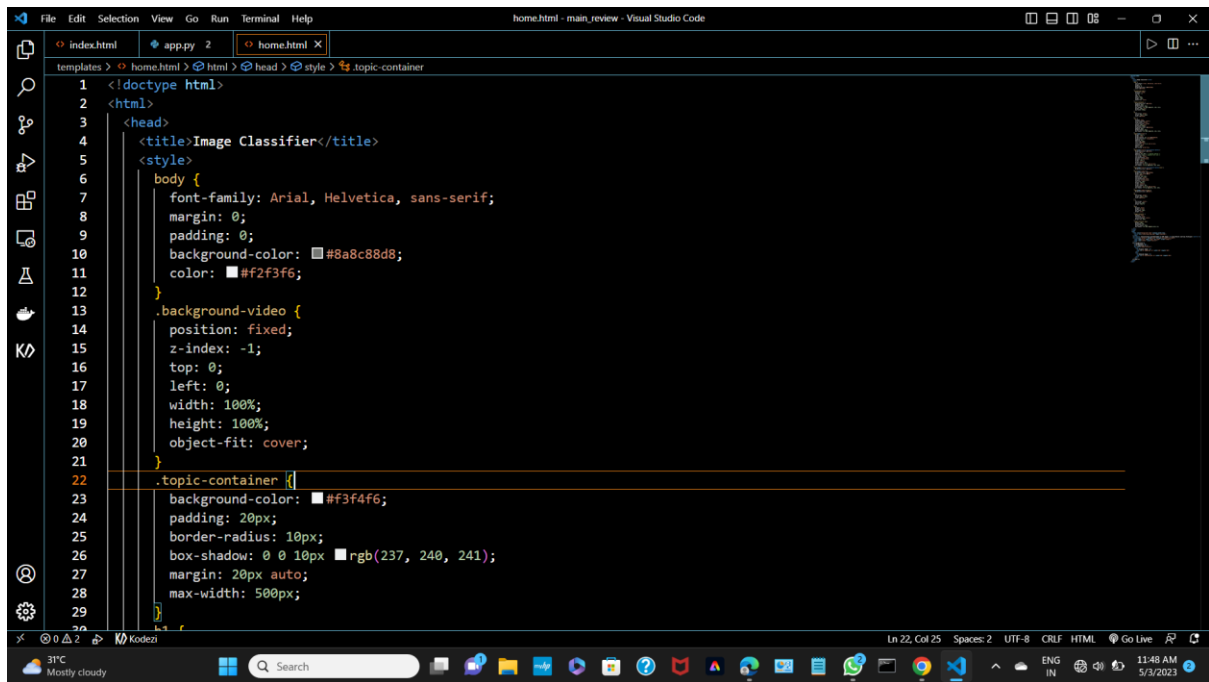
The benefit of using CNNs is their ability to develop an internal representation of a two-dimensional image. This allows the model to learn position and scale of faces in an image. After training the CNN, it can be able to recognize faces in an image. One can effectively use Convolutional Neural Network for image data.



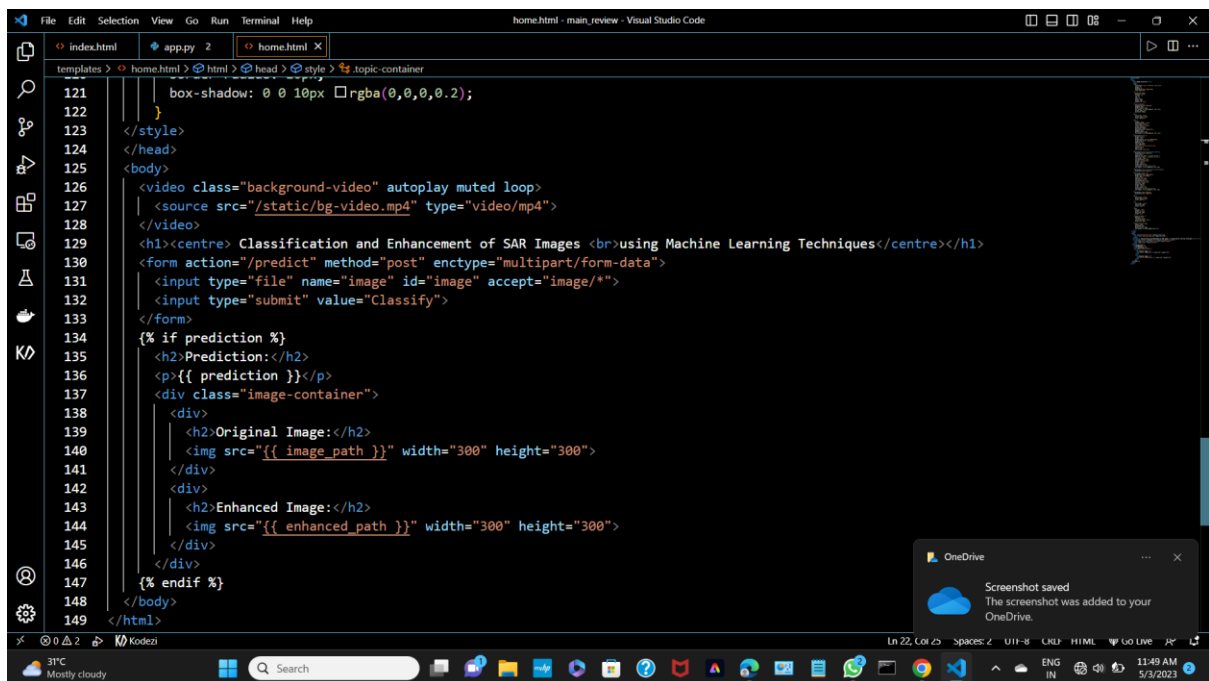
```
1 from flask import Flask, request, render_template
2 import tensorflow as tf
3 from tensorflow.keras.models import load_model
4 from tensorflow.keras.preprocessing import image
5 import numpy as np
6 from PIL import Image, ImageEnhance
7
8 app = Flask(__name__)
9
10 model = load_model('cnn_model.h5')
11 model.make_predict_function()
12
13 def predict(img_path):
14     img = image.load_img(img_path, target_size=(224, 224))
15     x = image.img_to_array(img)
16     x = np.expand_dims(x, axis=0)
17     x = x/255.
18     preds = model.predict(x)
19     return preds
20
21 def enhance_image(img_path):
22     img = Image.open(img_path)
23     enhancer = ImageEnhance.Contrast(img)
24     enhanced_img = enhancer.enhance(1.5)
25     enhanced_path = "static/enhanced_" + img_path.split("/")[-1]
26     enhanced_img.save(enhanced_path)
27     return enhanced_path
28
29 @app.route('/')
30 def home():
31     return render_template('index.html')
32
33 @app.route('/predict', methods=['GET', 'POST'])
34 def predict_image():
35     if request.method == 'POST':
36         img_file = request.files['image']
37         img_path = "static/" + img_file.filename
38         img_file.save(img_path)
39         enhanced_path = enhance_image(img_path)
40         preds = predict(img_path)
41         classes = ['AGRICULTURE LAND', 'BARREN LAND', 'GRASS LAND', 'URBAN LAND']
42         result = classes[np.argmax(preds)]
43         return render_template('home.html', prediction=result, image_path=img_path, enhanced_path=enhanced_path)
44
45 if __name__ == '__main__':
46     app.run(debug=True)
```



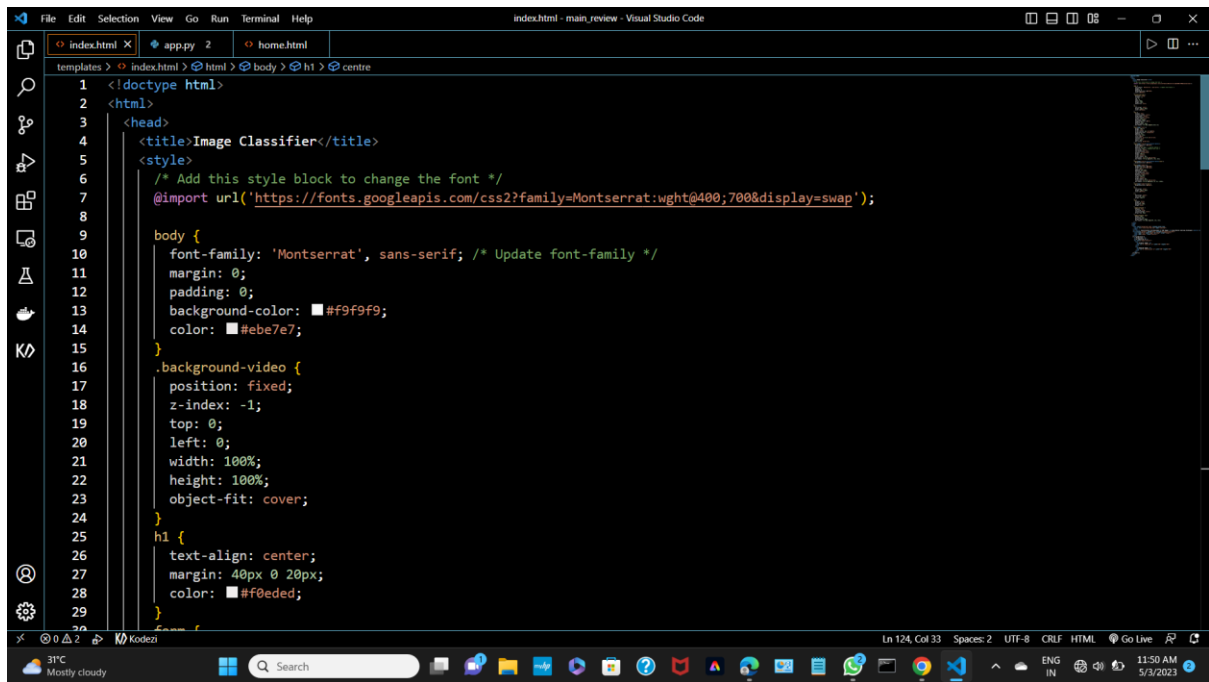
```
18 preds = model.predict(x)
19 return preds
20
21 def enhance_image(img_path):
22     img = Image.open(img_path)
23     enhancer = ImageEnhance.Contrast(img)
24     enhanced_img = enhancer.enhance(1.5)
25     enhanced_path = "static/enhanced_" + img_path.split("/")[-1]
26     enhanced_img.save(enhanced_path)
27     return enhanced_path
28
29 @app.route('/')
30 def home():
31     return render_template('index.html')
32
33 @app.route('/predict', methods=['GET', 'POST'])
34 def predict_image():
35     if request.method == 'POST':
36         img_file = request.files['image']
37         img_path = "static/" + img_file.filename
38         img_file.save(img_path)
39         enhanced_path = enhance_image(img_path)
40         preds = predict(img_path)
41         classes = ['AGRICULTURE LAND', 'BARREN LAND', 'GRASS LAND', 'URBAN LAND']
42         result = classes[np.argmax(preds)]
43         return render_template('home.html', prediction=result, image_path=img_path, enhanced_path=enhanced_path)
44
45 if __name__ == '__main__':
46     app.run(debug=True)
```



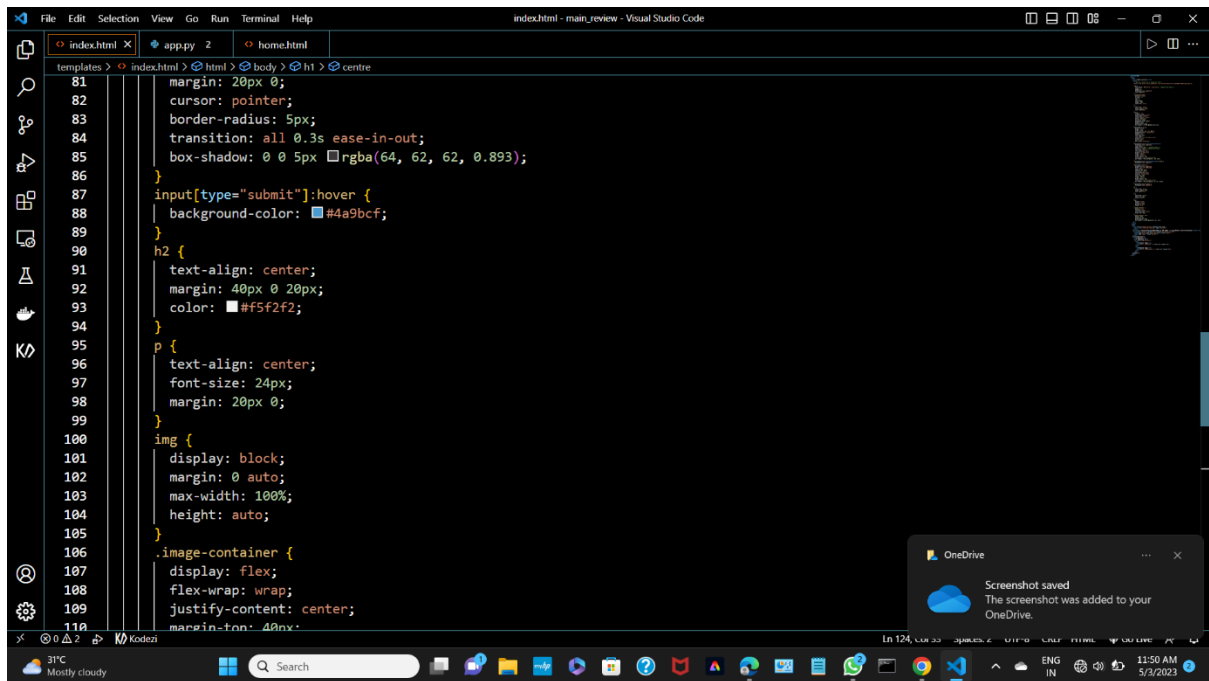
```
1 <doctype html>
2 <html>
3 <head>
4 <title>Image Classifier</title>
5 <style>
6   body {
7     font-family: Arial, Helvetica, sans-serif;
8     margin: 0;
9     padding: 0;
10    background-color: #8a8c88d8;
11    color: #f2f3f6;
12  }
13  .background-video {
14    position: fixed;
15    z-index: -1;
16    top: 0;
17    left: 0;
18    width: 100%;
19    height: 100%;
20    object-fit: cover;
21  }
22  .topic-container {
23    background-color: #f3f4f6;
24    padding: 20px;
25    border-radius: 10px;
26    box-shadow: 0 0 10px #rgb(237, 240, 241);
27    margin: 20px auto;
28    max-width: 500px;
29  }
30 }
```



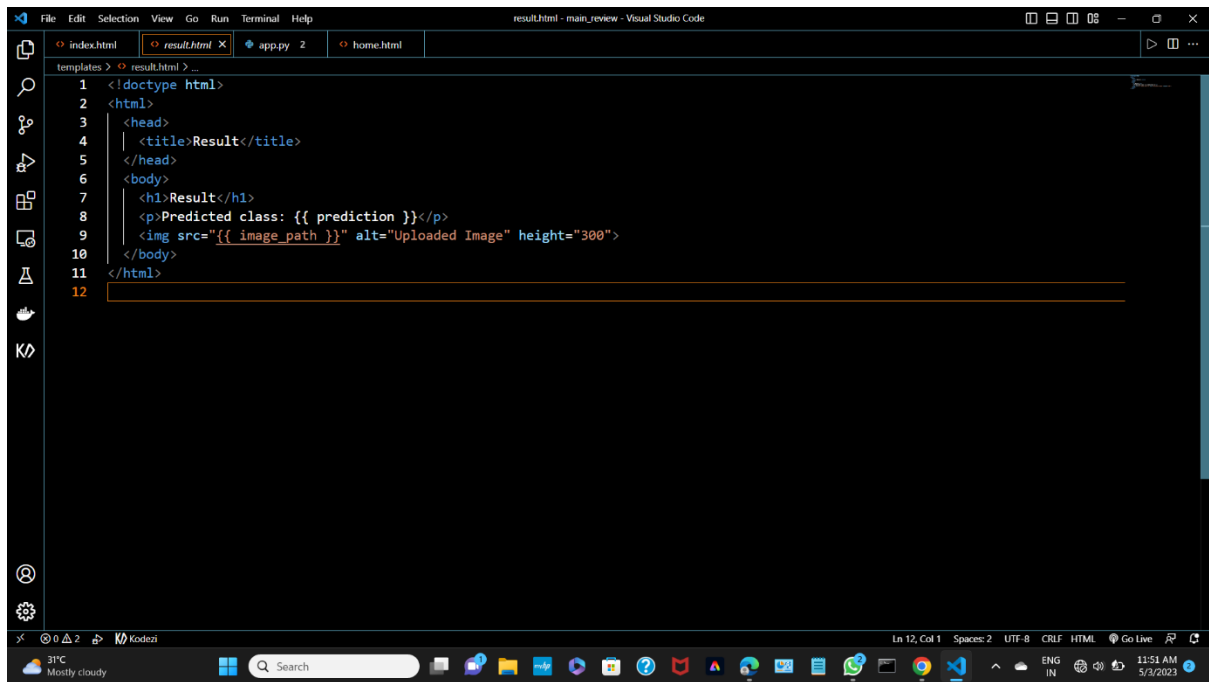
```
121   box-shadow: 0 0 10px #rgba(0,0,0,0.2);
122 }
123 </style>
124 </head>
125 <body>
126 <video class="background-video" autoplay muted loop>
127   <source src="/static/bg-video.mp4" type="video/mp4">
128 </video>
129 <h1><centre> Classification and Enhancement of SAR Images <br>using Machine Learning Techniques</centre></h1>
130 <form action="/predict" method="post" enctype="multipart/form-data">
131   <input type="file" name="image" id="image" accept="image/*">
132   <input type="submit" value="Classify">
133 </form>
134 {% if prediction %}
135 <h2>Prediction:</h2>
136 <p>{{ prediction }}</p>
137 <div class="image-container">
138   <div>
139     <h2>Original Image:</h2>
140     
141   </div>
142   <div>
143     <h2>Enhanced Image:</h2>
144     
145   </div>
146 </div>
147 {% endif %}
148 </body>
149 </html>
```



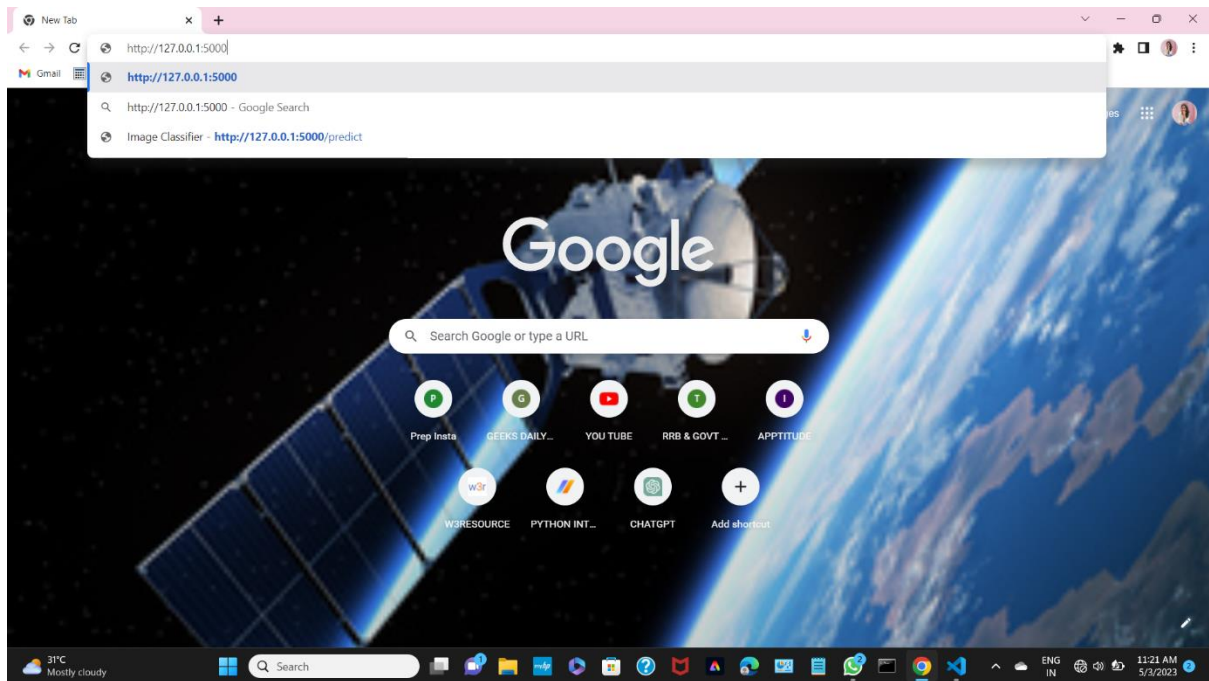
```
1 <!doctype html>
2 <html>
3   <head>
4     <title>Image Classifier</title>
5     <style>
6       /* Add this style block to change the font */
7       @import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=swap');
8
9       body {
10         font-family: 'Montserrat', sans-serif; /* Update font-family */
11         margin: 0;
12         padding: 0;
13         background-color: #f9f9f9;
14         color: #ebe7e7;
15       }
16       .background-video {
17         position: fixed;
18         z-index: -1;
19         top: 0;
20         left: 0;
21         width: 100%;
22         height: 100%;
23         object-fit: cover;
24       }
25       h1 {
26         text-align: center;
27         margin: 40px 0 20px;
28         color: #f0eded;
29       }
30     </style>
31   </head>
32   <body>
```

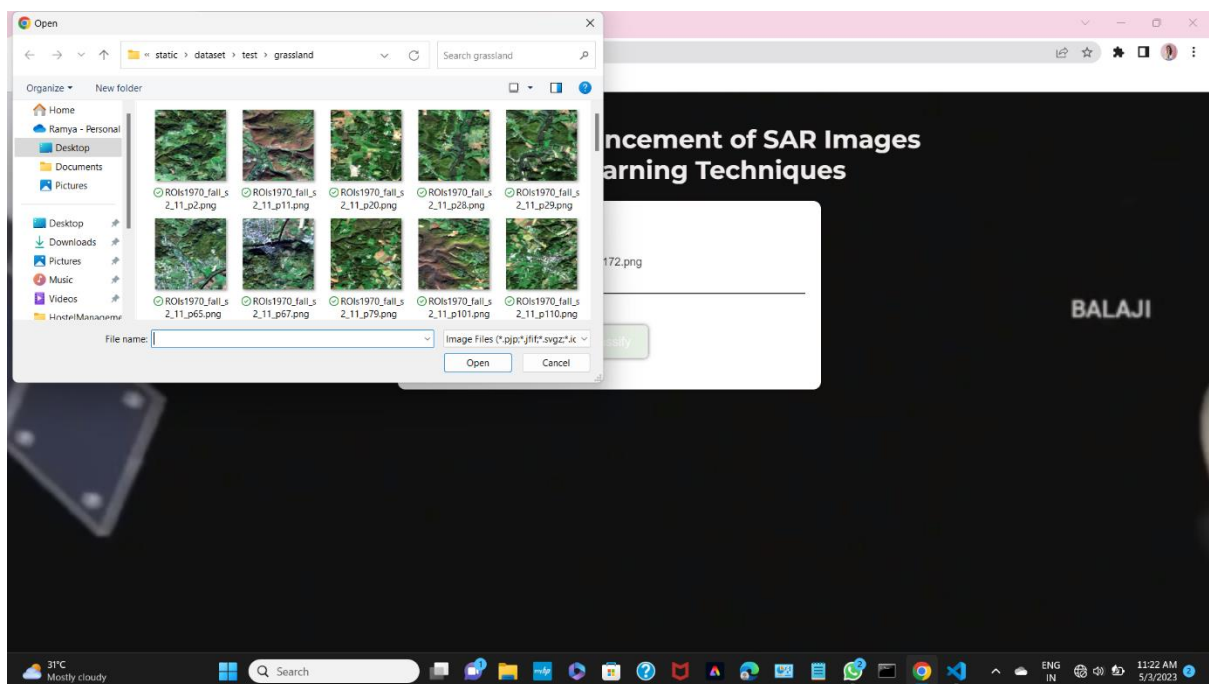
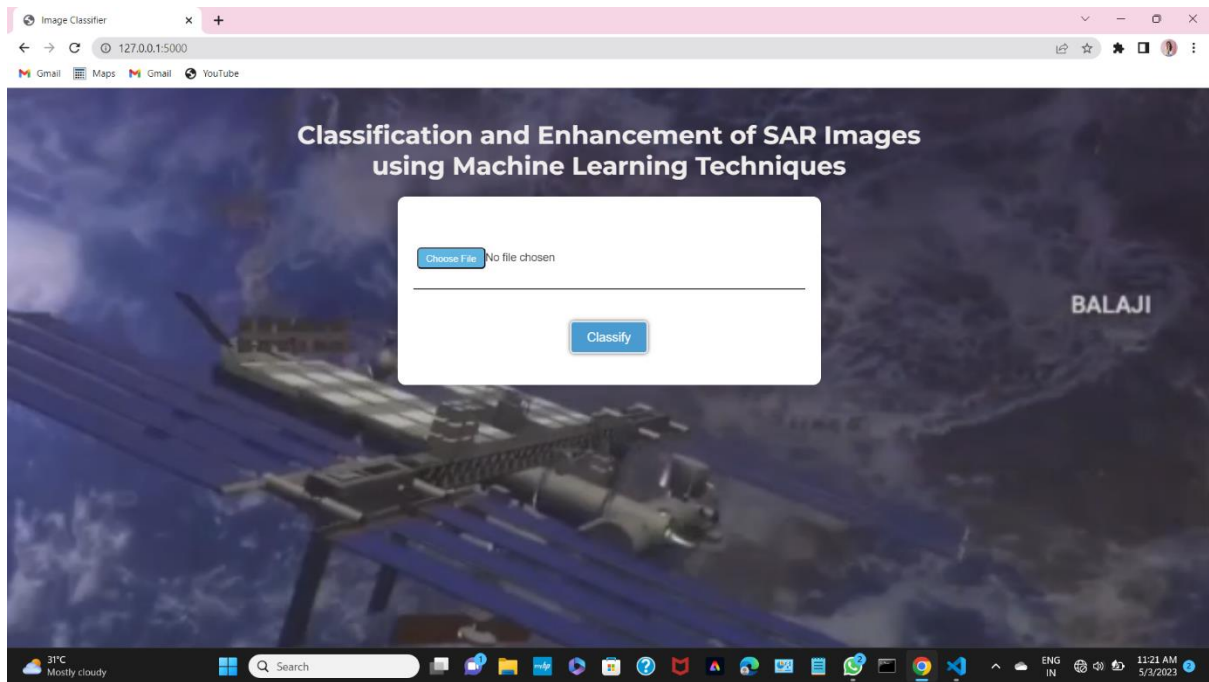


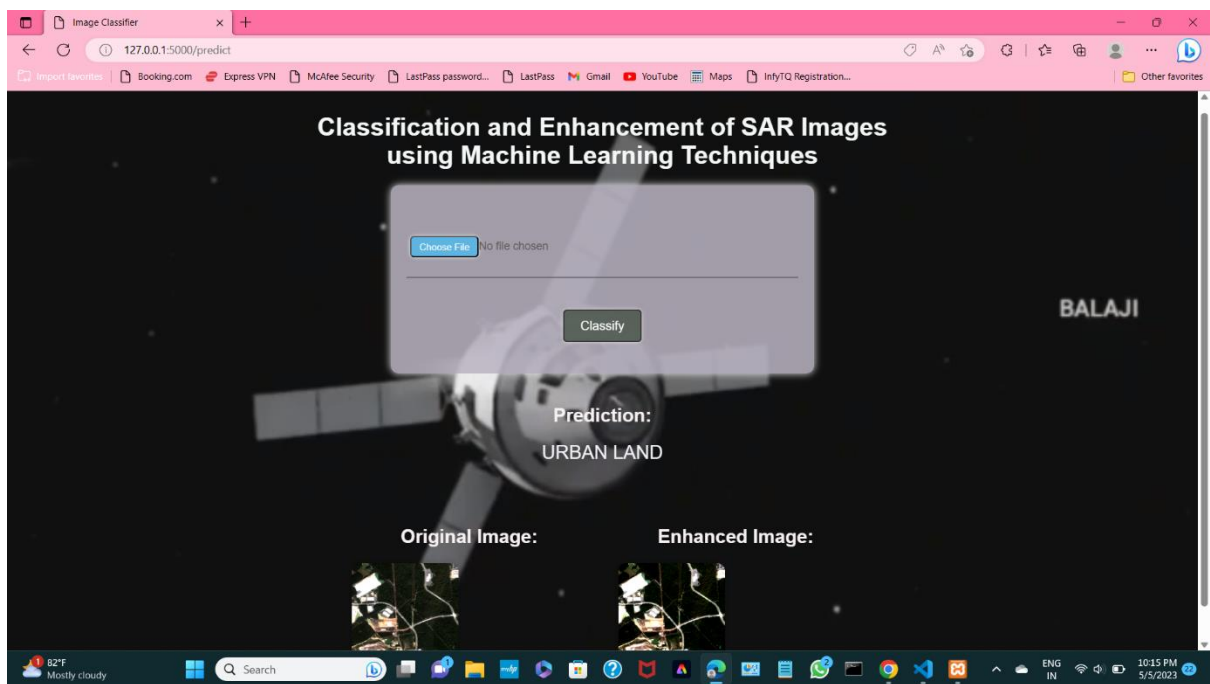
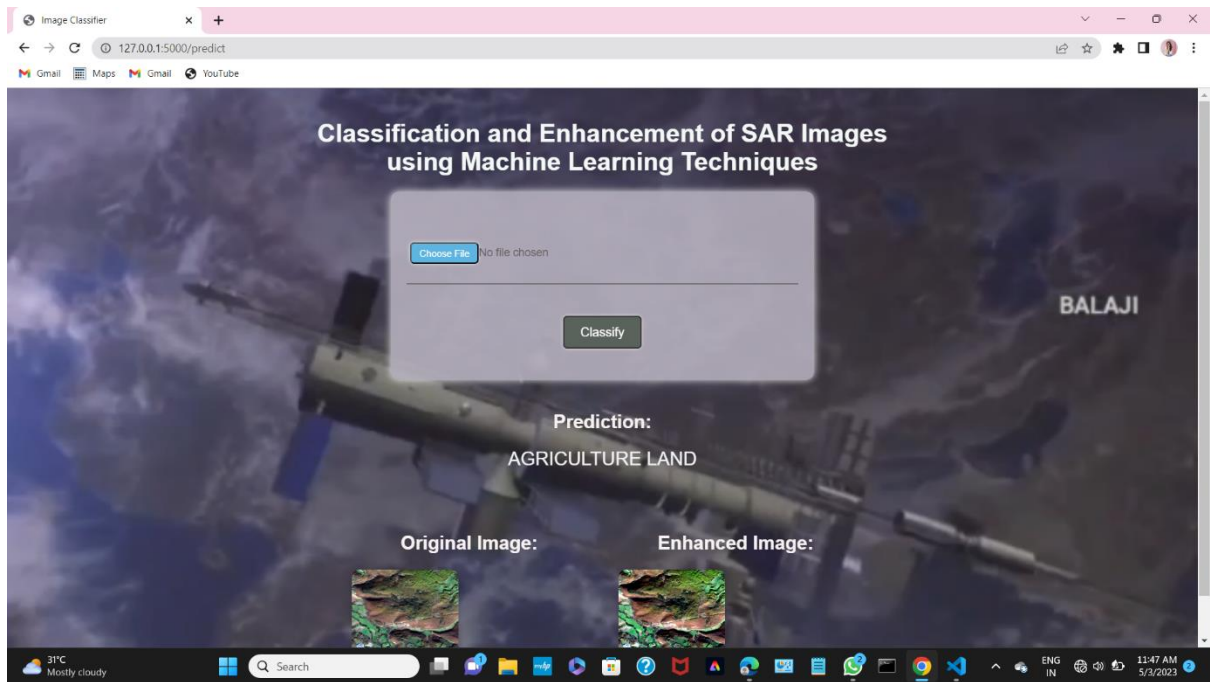
```
81     margin: 20px 0;
82     cursor: pointer;
83     border-radius: 5px;
84     transition: all 0.3s ease-in-out;
85     box-shadow: 0 0 5px rgba(64, 62, 62, 0.893);
86   }
87   input[type="submit"]:hover {
88     background-color: #4a9bcf;
89   }
90   h2 {
91     text-align: center;
92     margin: 40px 0 20px;
93     color: #f5f2f2;
94   }
95   p {
96     text-align: center;
97     font-size: 24px;
98     margin: 20px 0;
99   }
100  img {
101    display: block;
102    margin: 0 auto;
103    max-width: 100%;
104    height: auto;
105  }
106  .image-container {
107    display: flex;
108    flex-wrap: wrap;
109    justify-content: center;
110    margin-top: 40px;
111  }
```



```
1 <!doctype html>
2 <html>
3   <head>
4     <title>Result</title>
5   </head>
6   <body>
7     <h1>Result</h1>
8     <p>Predicted class: {{ prediction }}</p>
9     
10  </body>
11 </html>
12
```







CHAPTER 9

CONCLUSION

In this project, we explored the use of machine learning techniques to classify and enhance SAR images. We started by performing preprocessing on the dataset, which involved noise reduction and speckle filtering. Then, we used convolutional neural networks (CNN) to classify the images into different categories. Finally, we applied a spatial neural network (SNN) to enhance the output images.

The results of our experiments showed that the proposed approach was effective in both classification and enhancement of SAR images. The CNN achieved high accuracy in classifying the images, while the SNN significantly improved the quality of the output images by reducing the noise and enhancing the details.

CHAPTER 10

FUTURE ENHANCEMENT

There are several areas in which this project could be extended or improved. Here are some future enhancements that could be considered:

- Integration of other machine learning techniques: While CNNs are effective in image classification, there are other techniques such as support vector machines (SVM) and random forests that could be explored.
- Use of different enhancement techniques: In addition to SNN, there are other image enhancement techniques such as wavelet transforms and non-local means that could be explored.
- Use of more advanced preprocessing techniques: While we performed basic preprocessing in this project, more advanced techniques such as polarimetric filtering and phase history correction could be considered.
- Incorporation of multi-temporal data: SAR images captured at different times could be combined to produce a time series of images, which could be used to analyze changes over time.
- Use of more diverse datasets: While we used a single dataset in this project, it would be interesting to explore how the proposed approach performs on different datasets with varying characteristics.

Overall, there is still much room for improvement in the field of SAR image analysis using machine learning techniques.

REFERENCES

- [1] W. Mcculloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, *The bulletin of mathematical biophysics* 5 (4) (1943) 115–133. doi:10.1007/BF02478259.
- [2] M. Boschetti, A. Garzelli, and L. Lombardi, "Supervised classification of SAR images using machine learning techniques," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 10, pp. 2314-2323, 2005.
- [3] X. Yang, J. Huang, S. Wang, and J. Liu, "SAR image classification using deep convolutional neural networks with transfer learning," *Remote Sensing*, vol. 9, no. 3, p. 278, 2017.
- [4] Y. Zhu, Q. Du, G. Zhang, Y. Dong, and Y. Wang, "SAR image classification based on deep learning with improved stacked denoising autoencoder," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 1, pp. 310-323, 2017.
- [5] M. J. Arévalo, J. J. Ruiz, and R. Molina, "A review of SAR image classification using machine learning," *Remote Sensing*, vol. 10, no. 2, p. 182, 2018.
- [6] S. Chen, G. Wu, Y. He, and X. Wang, "SAR image enhancement using deep convolutional neural network and multi-scale wavelet transform," *Remote Sensing*, vol. 9, no. 5, p. 449, 2017.

- [7] J. Wei, Y. Wang, X. Li, and W. Wei, "SAR image despeckling via convolutional neural network and non-local means," *Remote Sensing*, vol. 11, no. 10, p. 1154, 2019.
- [8] D. Shrestha, Y. N. Wang, and Y. Zhang, "SAR Image Classification Using Spiking Neural Network with STDP and Backpropagation," in *IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 107-111.
- [9] Y. Zhu, Y. Dong, G. Zhang, Q. Du, and Y. Wang, "SAR image classification using spiking neural networks with STDP and backpropagation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 10, pp. 5878-5889, 2018.
- [10] L. Song, J. Chen, S. Yin, and S. Lu, "SAR Image Classification based on Spiking Neural Networks with STDP and Gradient Descent," in *IEEE International Conference on Computational Science and Engineering (CSE)*, 2020, pp. 138-143.