TEXTURE IMAGE CLASSIFICATION AND ANALYSIS

PROJECT REPORT

Submitted to the **MANONMANIAM SUNDARANAR UNIVERSITY** in practical fulfillment of the requirements for the award of the **Degree of Bachelor of Computer Application.**

SUBMITTED BY

K.Alagu Mahalakshmi 20222071401203

Under the guidance of

Dr. S. Anitha M.C.A., M.Phil., Ph.D.,

Head & Assistant professor

Department of Computer Application

GOVINDAMMAL ADITANAR COLLEGE FOR WOMEN

TIRUCHENDUR-628215

APRIL-2025

GOVINDAMMAL ADITANAR COLLEGE FOR WOMEN

TIRUCHENDUR-628215

CERTIFICATE

This is to certify that the project entitled "TEXTURE IMAGE CLASSIFICATION AND ANALYSIS".

SUBMITTED BY

K.Alagu Mahalakshmi 20222071401203

Is a partial fulfillment of the requirements for the B.C.A., Degree course in Computer Application (2024-2025) of the **MANONMANIYAM SUNDARANAR UNIVERSITY** is done under my guidance and it is the original work of the candidates.

Signature of the H.O.D(i/c).

(Dr. S. Anitha)

Signature of the guide

(Dr. S. Anitha)

External Examiner

1.

2.

GOVINDAMMAL ADITANAR COLLEGE FOR WOMEN

TIRUCHENDUR-628215

DECLARATION BY THE CANDIDATES

K.Alagu Mahalakshmi 20222071401203

We are the final year students of Department of Computer Application, Govindammal Aditanar College For women, assure that the project entitled "TEXTURE IMAGE CLASSIFICATION AND ANALYSIS" is a bonafide record done by us for the Bachelor of Computer Application in the academic year (2024-2025).

(K.Alagu Mahalakshmi)

Signature of the Guide

(Dr. S. Anitha)

Signature of the candidates

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We feel honored to thanking our principal **Dr. P. Jeyanthi, M.Sc., M.Phil., Ph.D., P.G.D.C.A.,** who had been influential in carrying their project work.

We wish to express our gratitude to **Dr. S. Anitha, M.C.A., M.Phil., Ph.D.,** H.O.D(i/c) of Department of Computer Application for helping us immensely by all means and encouraging us for the successful completion of our project.

We consider ourselves very fortunate in having **Dr. S. Anitha, M.C.A., M.Phil., Ph.D.,** H.O.D(i/c) of department of computer Application as our project guide.

Our special thanks go to all the staff members of our Computer Application Department. Then our graceful thanks to the lab assistants **Mrs. V. Essakkiammal, M.Com., DCA.,** and **Ms.T. Rakasana, M.Sc.,** for their kind support and timely help to complete our project.

Finally, we would like to thank our parents and friends for their help and encouragement at various stages of our project completion successfully.

CONTENTS

CONTENTS	PAGE NUMBER
1. Synopsis	1
2. System Configuration	
2.1 Hardware Specification	2
2.2 Software Specification	2
2.2.1 Software Description	2
3. System Analysis	
3.1 Existing System	6
3.2 Proposed System	6
4. System Design	
4.1 Logical Design	9
4.1.1 Architectural Diagram	9
5. System Testing	11
6. Source Code	13
7. Sample Output	40
8. Conclusion	44
9. Future Enhancement	45
10. Bibliography	46

SYNOPSIS

1.SYNOPSIS

Texture image classification is a crucial task in various applications, including biomedical image analysis, material science, robotics. The increasing complexity of texture image necessitates the development of robust and efficient classification algorithms. This study aims to investigate the performance of three machine learning algorithms - Convolutional Neural Network (CNN), K-Nearest Neighbour (KNN) and Support Vector Machine (SVM) in classifying texture images. KNN is a simple and effective algorithm used to classify images based on similarity to train samples. SVM is explored for ability to handle high data and effectively create decision boundaries in complex texture datasets, particularly through the use of non-linear kernels. CNN is an advance level image classification algorithm. The motivation behind this research is to identify the most effective algorithm for texture image classification, as existing methods often struggle with variations in scale, rotation and illumination. The experimental results shows that CNN out performs the others two algorithm, achieving an accuracy of 92.5%, followed by SVM (85.2%) and KNN (78.5%). This superior performance of CNN can be attributed to its ability to learn hierarchical representation of textures, which are robust to variations in appearance.

The finding of this study demonstrate the effectiveness of CNN in texture image classification, highlighting its potential application in various fields. The result prove that CNN is the best algorithm for texture image classification, offering improved classification accuracy and robustness to variation in texture image.

SYSTEM CONFIGURATION

2.SYSTEM CONFIGURATION

2.1 HARDWARE SPECIFICATION

Processor : Lenova Desktop Pro G1 MT Ci37100 Processor

RAM : 4G RAM

Monitor : 18.5”

Keyboard : Lenovo

Mouse : Lenovo

Printer : Laser Printer(HP Lazer zet p2014)

Hard disk : 1 TBHDD

2.2 SOFTWARE SPECIFICATION

Front end Tool : MYSQL

Back end Tool : Python

Operating System : Windows 10

2.2.1 SOFTWARE DESCRIPTION

INTRODUCTION TO PYTHON:

- Python is widely used general purpose, high level programming language. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.
- Python is a programming language that lets you work quickly and integrate system more efficiently.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages. Python runs on an interpreter system, meaning that code can be executed as soon it is written. This means that prototyping can be very quick. Python can be treated in a procedure way, an object-oriented way or a function way.

HISTORY:

- Python is a widely used general-purpose, high level programming language. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.
- Python has syntax that allows developers to write programs with fewer lines than other programming languages.

OVERVIEW:

- Python is a high level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English key words frequently where as other languages used punctuation and it has fewer syntactical constructions than other languages. Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming language which often uses semicolons or parenthesis.
- Python relies on indentation, using whitespaces, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

CHARACTERISTICS OF PYTHON:

The target of python is write a program once and then run this program on multiple operating systems.

- **EASY TO CODE:**

  Python is a high-level programming language. Python is very easy to learn the language as compared to other language like C, C#, Javascript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

- **FREE AND OPEN SOURCE:**

  Python language is freely available at the official website and you can download it from the given download link below click on the download python keyword.

- **OBJECT-ORIENTED LANGUAGE:**

  One of the key features of python is object oriented programming language. Python supports object oriented language and concepts of classes, object, encapsulation, etc.,

- **HIGH-LEVEL LANGUAGE:**

  Python is a high-level language. When we write program in python, we do not need to remember the system architecture, nor do we need to memory.

- **EXTENSIBLE FEATURES:**

  Python is an extensible language. We can write as some python code into C or C++ language and also we can compile that code in C/C++ language.

- **PYTHON IS PORTABLE LANGUAGE:**

  Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platform such as LINUX, UNIX and MAC then we do not need to change it, we can run this code on any platform.

- **INTERPRETED LANGUAGE:**

  Python is a interpreted language because python code is executed line by line at a time. Like other languages C, C++, Java, etc,. There is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called Bytecode.

- **DYNAMICALLY TYPED LANGUAGE:**

  Python is a dynamically typed language. That means the type (for example -int, double, long, etc,.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of layer.

INTRODUCTION TO HTML:

- **HTML** stands for Hypertext Markup Language. It is used to design web pages using a markup language. HTML is a combination of Hypertext and Markup language. Hypertext defines the link between web pages. A markup language is used to define the text document within the tag which defines the structure of web pages. This language is used to annotate text so that a machine can understand it and manipulate text accordingly. Most markup languages are human-readable. The language uses tags to define what manipulation has to be done on the text.

INTRODUCTION TO CSS:

- Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of a document written in HTML or XML. CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

- CSS is among the core languages of the open web and is standardized across Web browsers according to W3C specifications. Previously, the development of various parts of CSS specification was done synchronously, which allowed the versioning of the latest recommendations. You might have heard about CSS1, CSS2.1, or even CSS3. There will never be a CSS3 or a CSS4; rather, everything is now CSS without a version number.

APPLICATION OF TEXTURE IMAGE CLASSIFICATION:

- Medical Imaging
- Remote Sensing
- Industrial Inspection
- Document Analysis
- Environment Modeling
- Content-Based Image Retrieval

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1. EXISTING SYSTEM

In Existing System, there are many ways to classify an image. The best way is using algorithms "Convolutional Neural Networks (CNN)", "K-Nearest Neighbors (KNN)", "Support Vector Machine (SVM)".

1) **Convolutional Neural Networks (CNN):**

- Automatically extracts hierarchical features directly from raw images.
- Handles variations in texture caused by scale, rotation and illumination.
- Suitable for complex tasks like defect detection and medical imaging.
- Ideal for large datasets with diverse texture pattern

2) **K-Nearest Neighbors (KNN):**

- Classifies texture based on similarity in feature space.
- Relies on handcrafted texture feature like GLCM or LBP.
- Best for small datasets and simple classification tasks.
- Easy to implement but computationally expensive for large datasets.

3) **Support Vector Machines (SVM):**

- Classifies texture using high dimensional handcrafted features.
- Robust to overfitting and effective for small datasets
- Utilizes kernel functions to separate complex texture classes.
- Commonly used in defect detection, material analysis and medical imaging.

3.2. PROPOSED SYSTEM

A Image classification and analysis can be designed using a combination of various algorithms and techniques such as CNN,KNN,SVM.Here is a proposed system for Image classification.

1) **Input**
   - **Data type:** Texture image from diverse domains such as medical imaging, material inspection, or agriculture

- **Dataset Example:** Brodatz Texture dataset, KTH-TIPS, Outex Dataset

2) Preprocessing

- **Resizing:** Standardize image dimensions (e.g.,128 pixels) to ensure uniformity.
- **Normalization:** Scale pixel intensity values to[0,1] or [-1,1] for faster convergence.
- **Data Augmentation:** Rotate, flip, zoom and adjust brightness to simulate real world variations.
- **Noise Reduction:** Use Gaussian or median filters to reduce Norse while preserving texture details.

3) Representation

- **Deep Features(CNN)**

  1. Extract hierarchical features (edges, patterns) Using pre-trained CNN Models (e.g., VGG, ResNet)
  2. Fine -tune the -CNN model on Texture specific data

- **Handcrafted features**

  1. **Statistical features:** Gray -level co-occurrence Matrix(GLCM) for Texture contract, homogeneity and energy
  2. **Structural Features:** Local binary patterns (LBP) for micro-patterns in textures.
  3. **Frequency Features:** Gabor fitters and wavelet transform to capture texture pattern at various scale.

4) Output

- **Primary output:**
  1. **Class labels:** Texture categories e.g., rough, smooth, coarse, fine.
  2. **Probabilities:** Confidence score for each class.

- **Secondary Outputs**
  1. **Visual Insights:** Headmaps or Saliency maps to explain classification decision.

2. **Performance metrics:** Accuracy recall and F1 score for evaluating model performance.

5) **Optimization**

- **CNN Optimization:**

1. **Loss function:** cross-Entropy loss for multi - class classification
2. **Optimizer:** Adam or SGD with momentum for efficient learning
3. **Learning Rate scheduling:** Reduce learning rate on plateau to prevent overfitting.
4. **Regularization:** Dropout and L2 regularization to avoid overfitting.

- **KNN optimization:**

1. **Distance metric:** Experiments with Euclidean, Manhatton or Minkowski distance to improve classification accuracy.
2. **Hyperparameter Turning:** Optimize the number of neighbour (K) using grid search of cross validation.

- **SVM optimization:**

1. **Kernel selection:** Experiment with linear, polynomial or RBF kernels to capture complex decision boundaries.
2. **Regularization parameters (C):** Balance margin maximization and classification accuracy.
3. **Gamma (RBF kernel):** Fine -tune to control influence of each support vector.

SYSTEM DESIGN

4. SYSTEM DESIGN

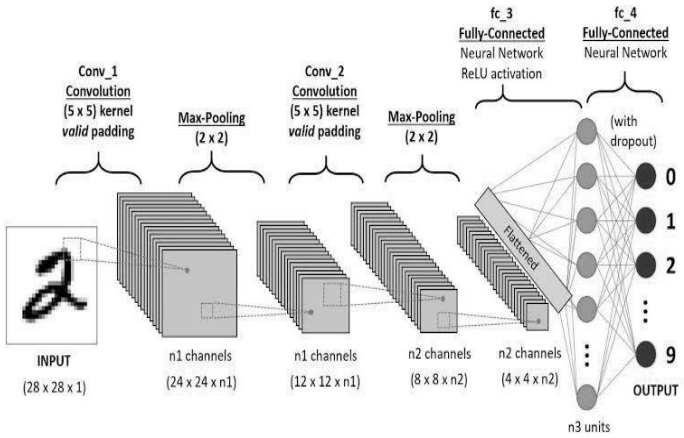
4.1. LOGICAL DESIGN

- System Design is the process of new systems or to replace or complement an existing system. System Design involves first logical design and then physical construction of the system.
- It contains the details specification for the new system. It describes its features. Outputs and inputs all in a manner that meets the project requirements.

4.1.1. ARCHITECTURAL DIAGRAM

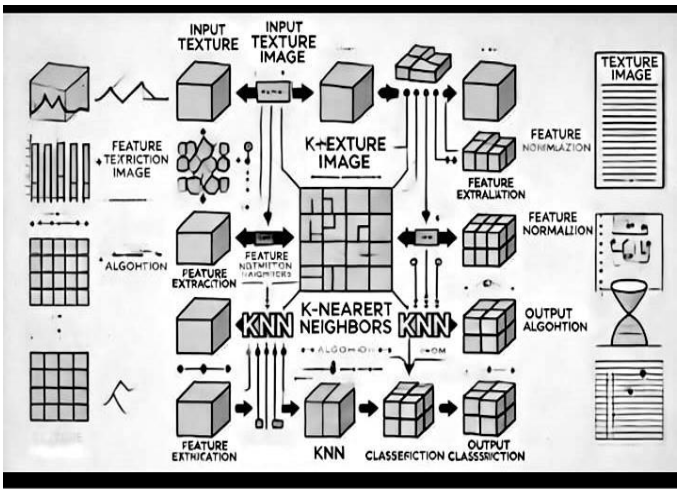
Convolutional Neural Network (CNN):

A Convolutional Neural Network (CNN) is a deep learning model used for image classification. It processes images by detecting important features like edges and patterns through multiple layers. First, convolution layers extract features, then pooling layers reduce the size while keeping important details. Finally, fully connected layers classify the images into different categories. CNNs are widely used in computer vision because they can automatically learn and recognize patterns, making them highly effective for texture image classification



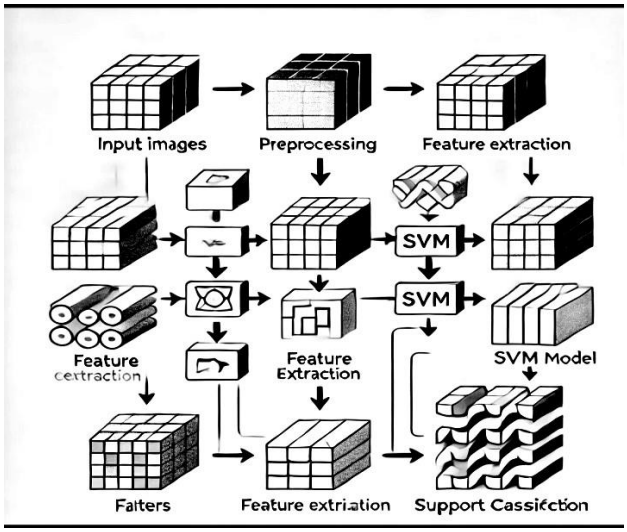
K-Nearest Neighbors (KNN):

The K-Nearest Neighbors (KNN) algorithm is a simple machine learning method used for classification. It works by comparing a new image to existing labeled images and finding the K most similar ones based on features like texture patterns. The image is then classified based on the majority label of its nearest neighbors. KNN does not require training; instead, it useful for texture image classification, where patterns and similarities play a key role in identifying different categories.



Support Vector Machine (SVM):

The Support Vector Machine (SVM) is a machine learning algorithm used for classification. It works by finding the best boundary (hyperplane) that separates different categories in the data. SVM analyzes features from images and determines the optimal way to distinguish between different classes. It is especially useful for texture image classification because it can handle complex patterns and high dimensional data efficiently.



SYSTEM TESTING

5. SYSTEM TESTING

Here are the steps for system testing for Texture image classification and analysis using Convolutional Neural Networks (CNN), K-Nearest Neighbors (KNN) and Support Vector Machine (SVM):

Step 1: Test Environment Setup

Set up the test environment, including the operating system, hardware,and software dependencies.

Step2: Functionality Testing

Test the system's functionality, including:

- Image input and processing.
- Classification using CNN, KNN and SVM.
- Analysis and feature extraction.
- Algorithm selection and switching.

Step 3: Performance Testing

Test the system's performance, including:

- Execution time for each algorithm.
- Accuracy and precision of classification.
- Scalability with large datasets and high-resolution images.

Step 4: Usability Testing

Test the system's usability, including:

- User interface and user experience.
- Documentation and help resources.
- Error handling and feedback.

Step 5: Security Testing

Test the system's security, including:

- Data storage and transmission security.
- Access control and authentication.
- Vulnerability to attacks and exploits.

Step 6: Compatibility Testing

Test the system's compatibility with different:

- Operating system (Windows, Linux, macOS).
- Hardware configuration (CPU, GPU, RAM).
- Software dependencies (libraries, frameworks).

Step 7: Regression Testing

Test the system's functionality and performance after making changes or updates.

Step 8: System Deployment Testing

Test the system's deployment, including:

- Installation and setup.
- Configuration and customization.
- Integration with other systems or tools.

Step 9: System Maintenance Testing

Test the system's maintenance, including:

- Backup and recovery.
- Updates and patches.
- Troubleshooting and debugging.

SOURCE CODE

HOME PAGE

Main_app.py

```
from flask import Flask,render_template
app=Flask(__name__)
@app.route('/')
def home():
    return render_template('main_index.html')
if __name__=='__main__':
    app.run(port=5009,debug=True)
```

Main_index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTD-8">
        <title><center>IMAGE CLASSIFICATION AND ANALYSIS</center></title>
    </head>
    <style>
    body{
        font-family: Arial, Helvetica, sans-serif;
        text-align: center;
        background: linear-gradient(135deg,#001f3f,#0074D9,#00c6ff);
        height: 20cm;
        margin: 0;
        padding: 0;
    }
    .container{
        display: flex;
        align-items: center;
        justify-content: space-between;
        gap:50px;
        padding: 20px;
    }
    .content{
        text-align: center;
        flex-grow: 1;
    }
    h1{
        color: #FFFFFF;
        font-weight: bold;
        text-shadow: 2px 2px 8px rgba(0,255,255,0.8);
        margin-top: 50px;
    }
    ul{
        list-style-type:none;
        padding: 0;
```

```
}
li{
    margin: 20px 0;
}
a{
    display: inline-block;
    padding:  12px  24px;
    font-size: 18px;
    text-decoration: none;
    font-weight: bold;
    color: #FFFFFF;
    background:linear-gradient(135deg,#0074D9,#00c6ff);
    border: 2px solid #00FFFF;
    box-shadow: 0 0 10px rgba(0,255,255,0.8);
    transition: all 0.3s ease-in-out;
    border-radius: 8px;
    transition: 0.3s;
}
a:hover{
    background:linear-gradient(135deg,#00c6ff,#0074D9)
}
.left-image{
    position: absolute;
    top: 60%;
    left: 30px;
    transform: translateY(-50%);
    width: 350px;
    max-height: 350px;
    object-fit: contain;
}
.right-image{
    position: absolute;
    top: 60%;
    right: 40px;
    transform: translateY(-50%);
    width: 350px;
    max-height: 350px;
    object-fit: contain;
}
h1{
    font-size:  1.2cm;
    color: #D9D9D9;
    font-weight: 500;
    text-shadow: 1px 1px 5px rgba(255,255,255,0.5);
}
h3{
    font-size: 0.8cm;
}
</style>
<body>
```

```
    <img src="{{url_for('static',filename='uploads/ICA.jpg')}}" class="left-image" width="350" height="350">
    <div class="container">
    <div class="content">
    <h1><center>IMAGE CLASSIFICATION AND ANALYSIS</center></h1>
    <h3><center>Choose A Classification Algorithm</center></h3>
    <br>
    <center>
      <div style="text-align: center;margin-top: 20px;">
        <h2><a id="knn-btn" class="button" href="http://127.0.0.1:5001" style="text-decoration: none; font-size: 24px; font-weight: bold; color:black;">KNN CLASSIFICATION</a></h2>
        <h2><a id="cnn-btn" class="button" href="http://127.0.0.1:5002" style="text-decoration: none; font-size: 24px; font-weight: bold; color:black;">CNN CLASSIFICATION</a></h2>
        <h2><a id="svm-btn" class="button" href="http://127.0.0.1:5003" style="text-decoration: none; font-size: 24px; font-weight: bold; color:black;">SVM CLASSIFICATION</a></h2>
        <h2><a id="graph-btn" class="button" href="http://127.0.0.1:5010" style="text-decoration: none; font-size: 24px; font-weight: bold; color: black;">VIEW GRAPH</a></h2>
      </div>
    </center>
    </div>
    <br>
    <br>
    <img src="{{url_for('static',filename='uploads/OCI2.jpg')}}" class="right-image" width="350" height="350">
    </div>
  </body>
</html>
```

GRAPH

Graph.py

```
from flask import Flask,render_template
app=Flask(__name__)
@app.route('/')
def home():
  return render_template('graph.html')
if __name__=='__main__':
  app.run(port=5010,debug=True)
```

Graph.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
    <meta charset="UTD-8">
    <title><center></center></title>
</head>
<style>
body{
    background: linear-gradient(#00008B,#87CEFA);
    height:20cm;
    max-width: 80%;
    margin: auto;
    overflow-x: hidden;
    font-size: 16px;
}
.container{
    width: 70%;
    margin: auto;
    padding: 10px;
}
 .center-image{
    position: absolute;
    top: 55%;
    left: 240px;
    transform: translateY(-50%);
    width: 750px;
    max-height: 750px;
    object-fit: contain;
}
h1{
    font-weight: bold;
    color:chocolate;
    font-size: 2.5rem;
}
.btn-primary {
        background-color: #FFD700;
        border: none;
        padding: 10px 20px;
        color: black;
        font-size: 16px;
        border-radius: 5px;
        cursor: pointer;
        font-weight: bold;
        backdrop-filter: blur(10px);
}
.btn-primary:hover {
        background: rgba(108, 117, 125, 0.4);
        border: 1px solid rgba(108, 117, 125, 0.4);
}
</style>
<body>
    <h1><center>HISTOGRAM</center></h1>
    <div class="container">
```

```
    <center>
    <img    src="{{url_for('static',filename='uploads/graph5.jpg')}}"      class="center-image"
    width="750" height="750">
    </center>

    <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
     <br><br><br><br><br><br><br><br>
    <center><a href="http://127.0.0.1:5009/" class="btn btn-primary">Home</a></center>
    </div>
  </body>
  </html>
```

CNN

app.py

```
from flask import Flask, request, render_template
from torchvision import transforms
from PIL import Image
import torch
import torch.nn as nn
import os
import time
import torchvision

# Define CNN architecture
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3)
        self.fc1 = nn.Linear(64 * 14 * 14, 128)
        self.fc2 = nn.Linear(128, 3)  # 3 classes: Cat, Dog, Snake

    def forward(self, x):
        x = torch.relu(self.conv1(x))
        x = self.pool(x)
        x = torch.relu(self.conv2(x))
        x = self.pool(x)
        x = x.view(-1, 64 * 14 * 14)
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Load the trained model
model = CNN()
model.load_state_dict(torch.load('E:/iiiBCA/MAJOR PROJECT1/cnn_project/model/cnn_m
odel.pth'))
model.eval()
```

```
# Class labels
class_labels = ['Cat', 'Dog', 'Snake']

# Image transformation
transform = transforms.Compose([
    transforms.Resize((64, 64)),
    transforms.ToTensor()
])

# Flask app setup
app = Flask(__name__)

# Set up the upload folder path
UPLOAD_FOLDER = 'E:/III BCA/MAJOR PROJECT1/cnn_project/static/uploads'
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)  # Create the directory if it does not exist

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Prediction function
def predict_image(image_path):
    image = Image.open(image_path).convert('RGB')
    image = transform(image).unsqueeze(0)
    with torch.no_grad():
        start_time = time.time()
        outputs = model(image)
        _, predicted = torch.max(outputs, 1)
        end_time = time.time()
        prediction_time = end_time - start_time
    return class_labels[predicted.item()], prediction_time

# Calculate accuracy function
def calculate_accuracy(loader):
    correct = 0
    total = 0
    with torch.no_grad():
        for images, labels in loader:
            outputs = model(images)
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
    return 100 * correct / total

# Dataset paths
dataset_path_train = 'E:/iii BCA/MAJOR PROJECT1/Dataset'
dataset_path_test = 'E:/iii BCA/MAJOR PROJECT1/Dataset'

# Ensure the dataset paths exist
if not os.path.exists(dataset_path_train):
```

```
    raise FileNotFoundError(f"Training dataset path does not exist: {dataset_path_train}")
if not os.path.exists(dataset_path_test):
    raise FileNotFoundError(f"Testing dataset path does not exist: {dataset_path_test}")

# Calculate accuracies before running the server
train_accuracy = calculate_accuracy(train_loader)
test_accuracy = calculate_accuracy(test_loader)
train_loader = torch.utils.data.DataLoader(
    torchvision.datasets.ImageFolder(dataset_path_train, transform=transform),
    batch_size=16, shuffle=True
)
test_loader = torch.utils.data.DataLoader(
    torchvision.datasets.ImageFolder(dataset_path_test, transform=transform),
    batch_size=16, shuffle=False
)
@app.route('/', methods=['GET', 'POST'])
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    file = request.files['image']
    if file:
        # Safely create the full file path
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)

        # Ensure the file name is safe and valid
        file.save(file_path)

        # Make the prediction
        result, prediction_time = predict_image(file_path)

        # Return the result to the user

Return render_template('result.html', result=result, image=file.filename, time=prediction_tim
e, train_accuracy=train_accuracy, test_accuracy=test_accuracy)

if __name___== '__main__':
    app.run(port=5002, debug=True)
```

cnn_training.py

```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
import os
import time
```

```
# Define CNN architecture
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3)
        self.fc1 = nn.Linear(64 * 14 * 14, 128)
        self.fc2 = nn.Linear(128, 3)  # 3 classes: Cat, Dog, Snake

    def forward(self, x):
        x = torch.relu(self.conv1(x))
        x = self.pool(x)
        x = torch.relu(self.conv2(x))
        x = self.pool(x)
        x = x.view(-1, 64 * 14 * 14)
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Dataset path
dataset_path = 'E:/iii BCA/MAJOR PROJECT1/Dataset/Animals'

# Data transformation
transform = transforms.Compose([
    transforms.Resize((64, 64)),
    transforms.ToTensor()
])

# Check if paths exist
train_dir = os.path.join(dataset_path)
test_dir = os.path.join(dataset_path)
if not os.path.exists(train_dir):
    raise FileNotFoundError(f"Training directory not found: {train_dir}")
if not os.path.exists(test_dir):
    raise FileNotFoundError(f"Testing directory not found: {test_dir}")

# Load the training dataset
train_data = datasets.ImageFolder(train_dir, transform=transform)
train_loader = DataLoader(train_data, batch_size=16, shuffle=True)

# Load the testing dataset
test_data = datasets.ImageFolder(test_dir, transform=transform)
test_loader = DataLoader(test_data, batch_size=16, shuffle=False)

# Initialize model, loss, and optimizer
model = CNN()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

```
# Training loop
total_train_time = 0
for epoch in range(10):
    total_loss = 0
    epoch_start_time = time.time()
    for images, labels in train_loader:
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
    epoch_end_time = time.time()
    epoch_train_time = epoch_end_time - epoch_start_time
    total_train_time += epoch_train_time
    print(f'Epoch {epoch+1}, Loss: {total_loss:.4f}, Time: {epoch_train_time:.2f} seconds')

# Calculate training accuracy
def calculate_accuracy(loader):
    correct = 0
    total = 0
    with torch.no_grad():
        for images, labels in loader:
            outputs = model(images)
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
    return 100 * correct / total

# Training accuracy
train_accuracy = calculate_accuracy(train_loader)
print(f'Training Accuracy: {train_accuracy:.2f}%')

# Testing accuracy
test_accuracy = calculate_accuracy(test_loader)
print(f'Testing Accuracy: {test_accuracy:.2f}%')

# Save the trained model
torch.save(model.state_dict(), 'model/cnn_model.pth')
print("Model saved successfully.")
print(f'Total training time: {total_train_time:.4f} seconds')
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Image Classification Using CNN</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
      body {
          background-image: url('/static/uploads/3409297.jpg'); /* Use the correct path to the
image */
          background-size: cover; /* Ensure the background image covers the entire screen */
          background-position: center top;
          background-repeat: no-repeat;
          height:100vh; /* Correct height unit */
          width:100vw;
          margin: 0;
          font-family: 'Great Vibes', cursive;
          display: flex;
          justify-content: center;
          align-items: center;
      }
      h2 {
          font-family: Georgia, 'Times New Roman', Times, serif;
          font-size: 2.5rem;
          background: -webkit-linear-gradient(#ffffff, #cccccc);
          background-clip: text;
          -webkit-text-fill-color: transparent;
      }
      h3{
          font-size: 2.0rem;
      }
      .upload-box {
          width: 500px; /* Set width */
           /* Set height equal to width */
          padding: 20px;
          text-align: center;
          margin-top: 10px;
          backdrop-filter: blur(10px);
          background: rgba(255, 255, 255, 0.2);
          border-radius: 10px;
          display: flex;
          flex-direction: column;
          justify-content: center;
          align-items: center;
          border: #fff9f9;
          border-style: dashed;
      }
      .upload-box input {
          margin-top: 15px;
      }
      .image-preview {
          margin-top: 15px;
      }
```

```
        .img-thumbnail {
            max-height: 200px;
        }
        .btn-primary {
            background: rgba(0, 123, 255, 0.2);
            border: 1px solid rgba(0, 123, 255, 0.2);
            color: #007bff;
            font-weight: bold;
            backdrop-filter: blur(10px);
        }
        .btn-primary:hover {
            background: rgba(0, 123, 255, 0.4);
            border: 1px solid rgba(0, 123, 255, 0.4);
        }
        .btn-glassy {
            background: rgba(255, 255, 255, 0.2);
            border: 1px solid rgba(255, 255, 255, 0.2);
            color: #007bff;
            font-weight: bold;
            backdrop-filter: blur(10px);
            border-radius: 10px;
        }
        .btn-glassy:hover {
            background: rgba(255, 255, 255, 0.4);
            border: 1px solid rgba(255, 255, 255, 0.4);
        }
        #selectedImage {
            width: 200px;
            height: 200px;
            object-fit: cover;
            display: block;
            margin: 0 auto;
            font-size: 1.5rem;
        }
        input{
            font-size: 1.2rem;
        }
    </style>
</head>
<body>
    <div class="container mt-5">
        <h2 class="text-center">Image Classification Using CNN</h2>
        <form action="/predict" method="POST" enctype="multipart/form-data">
            <center>
                 
            <div class="upload-box">
                <label for="image" class="form-label"><h3>Select an Image</h3></label>
                <input type="file" name="image" id="image" class="form-control" required>
                <div class="image-preview" id="image-preview">
                    <img id="selectedImage" alt="Selected Image">
```

```
            </div>
            <button type="submit" class="btn btn-primary btn-glassy mt-3">Classify Image</button>
        </div>
        </center>
      </form>
   </div>
   <script>
      document.getElementById("image").addEventListener("change", function() {
         const file = this.files[0];
         if (file) {
            const reader = new FileReader();
            reader.onload = function(e) {
               const imagePreview = document.getElementById("image-preview");
               const imgElement = document.createElement("img");
               imgElement.src = e.target.result;
               imgElement.classList.add("img-thumbnail");
               imagePreview.innerHTML = "";
               imagePreview.appendChild(imgElement);
            }
            reader.readAsDataURL(file);
         }
      });
   </script>
</body>
</html>
```

result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Classification Result</title>
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
   <link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">
   <style>
      body {
         background-image: url('/static/uploads/bg8.jpg'); /* Use the correct path to the image */
         background-size: cover; /* Ensure the background image covers the entire screen */
         background-position: center top;
         background-repeat: no-repeat;
         height: 100vh; /* Correct height unit */
         width: 90vw;
         margin: 0;
         font-family: 'Great Vibes', cursive;
```

```
    display: flex;
    justify-content: center;
    align-items: center;
}
.result-box {
    width: 600px; /* Set width */
    height: 450px; /* Set height equal to width */
    padding: 20px;
    text-align: center;
    margin-top: 10px;
    backdrop-filter: blur(10px);
    background: rgba(255, 255, 255, 0.2);
    border-radius: 10px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
.result-box img {
    max-width: 100%; /* Ensure the image fits within the box */
    max-height: 50%; /* Limit height to avoid overflowing */
    margin-bottom: 10px;
    align-items: center;
}
.btn-secondary {
    background: rgba(108, 117, 125, 0.2);
    border: 1px solid rgba(108, 117, 125, 0.2);
    color: white;
    font-weight: bold;
    backdrop-filter: blur(10px);
}
.btn-secondary:hover {
    background: rgba(108, 117, 125, 0.4);
    border: 1px solid rgba(108, 117, 125, 0.4);
}
.btn-primary{
    background: rgba(108, 117, 125, 0.2);
    border: 1px solid rgba(108, 117, 125, 0.2);
    color: white;
    font-weight: bold;
    backdrop-filter: blur(10px);
}
.btn-primary:hover {
    background: rgba(108, 117, 125, 0.4);
    border: 1px solid rgba(108, 117, 125, 0.4);
}
h1 {
    font-family: 'Pacifico', cursive;
    font-weight: 700;
    background: -webkit-linear-gradient(#178a99, #ffffff);
```

```
            background-clip: text;
            -webkit-text-fill-color: transparent;
            text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
        }
    </style>
</head>
<body>
<div class="container mt-5">

    <center><h1 class="text-center">Classification Result</h1></center>
    <center>
            
    <div class="result-box">
        <h4>Predicted Class: {{ result }}</h4>
        <img src="{{ url_for('static', filename='uploads/' + image) }}" alt="Uploaded Image"
class="img-fluid">
        <p><strong>Prediction Time:</strong> {{ time }} seconds</p>
        <p><strong>Training Accuracy:</strong> {{ 100 }}%</p>
        <p><strong>Testing Accuracy:</strong> {{ test_accuracy }}%</p>
        <center><a href="/" class="btn btn-secondary">Back to Upload</a>
        <a href="http://127.0.0.1:5009/" class="btn btn-primary">Home</a></center>
    </div>
    </center>
</div>
</body>
</html>
```

KNN

app.py

```
from flask import Flask,render_template,request,url_for
import os
import cv2
import time
import numpy as np
from werkzeug.utils import secure_filename
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import ntpath
app=Flask(__name__, static_folder='static')
UPLOAD_FOLDER="E:/iii BCA/MAJOR PROJECT1/knn_project/static/uploads"
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)
app.config["UPLOAD_FOLDER"]=UPLOAD_FOLDER
app.config["ALLOWED_EXTENSIONS"]={'png','jpg','jpeg','gif','jfif'}
app.config['MAX_CONTENT_LENGTH']=16*1024*1024
dataset_path="E:/iii BCA/MAJOR PROJECT1/Dataset"
```

```
categories=["Cat","Dog","Snake"]
image_size=(64,64)
data,labels,ids=[],[],[]
try:
    test_folder=os.path.join(dataset_path,categories[0])
    test_files=os.listdir(test_folder)
    print(f"Test: os.listdir() works. Found {len(test_files)} files in {categories[0]}")
except AttributeError as e:
    print("Error:o 'os.listdir() is not working. Possible module override issue")
    raise e
for category in categories:
    folder_path=os.path.join(dataset_path,category)
    print(f"Processing category:",{category})
    for filename in os.listdir(folder_path)[:100]:
        if not(filename.lower().endswith('.jpg') or filename.lower().endswith('.png')):
            continue
        print(f"Processing file:{filename}")
        img_path=os.path.join(folder_path,filename)
        img=cv2.imread(img_path)
        if img is not None:
            img=cv2.resize(img,image_size)
            img_gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            img_normalized=img_gray/255.0
            data.append(img_normalized.flatten())
            labels.append(category)
            image_id=filename.split('_')[0]
            ids.append(image_id)
        else:
            print(f"Failed to load{filename}")
image_id=filename.split('_')[0]
ids.append(image_id)
print(f"Data length:{len(data)}")
print(f"Label length:{labels[:5]}")
label_encoder=LabelEncoder()
encoded_labels=label_encoder.fit_transform(labels)
X_train,X_test,y_train,y_test=train_test_split(data,encoded_labels,test_size=0.2,random_stat
e=42)
knn=KNeighborsClassifier(n_neighbors=5)
def train_model():
    start_time=time.time()
    print("Training Model...")
    knn.fit(X_train,y_train)
    y_train_pred=knn.predict(X_train)
    train_accuracy=accuracy_score(y_train,y_train_pred)
    y_test_pred=knn.predict(X_test)
    test_accuracy=accuracy_score(y_test,y_test_pred)
    training_time=time.time()-start_time
    print(f"Total time:{training_time:.2f} seconds")
    return train_accuracy,test_accuracy,training_time
```

```
def predict(image_path):
    img=cv2.imread(image_path)
    img_resized=cv2.resize(img,image_size)
    img_gray=cv2.cvtColor(img_resized,cv2.COLOR_BGR2GRAY)
    img_normalized=img_gray/255.0
    img_flattened=img_normalized.flatten()
    prediction=knn.predict([img_flattened])
    predicted_label=label_encoder.inverse_transform(prediction)[0]
    allowed_classes=["Cat","Dog","Snake"]
    if predicted_label not in allowed_classes:
        return "Unknown Category"
    return predicted_label
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.',1)[1].lower() in
app.config["ALLOWED_EXTENSIONS"]

@app.route("/",methods=["GET","POST"])
def index():
    training_time=None
    test_accuracy=None
    train_accuracy=None
    predicted_label=None
    if request.method=="POST":
        if "image" not in request.files:
            print(f"No file part in request")
            return "No file uploaded",400
        file=request.files['image']
        print(f"File received:{file.filename}")
        if file.filename=="":
            print(f"No selected file")
            return "No selected file",400
        print(f"File received:{file.filename}")
        if file and allowed_file(file.filename):
            filename=secure_filename(file.filename)
            filepath=os.path.join(app.config["UPLOAD_FOLDER"],filename)
            file.save(filepath)
            if os.path.exists(filepath):
                print(f"File successfully saved at:{filepath}")
            else:
                print(f"File saving failed")
            predicted_label=classify_image_based_on_id(filename)
            print(f"File saved at:{filepath}")
            image_id=filename.split('_')[0]
            train_accuracy,test_accuracy,training_time=train_model()
            return
render_template("result.html",uploaded_image=filename,predicted_label=predicted_label,trai
n_accuracy=train_accuracy,test_accuracy=test_accuracy,training_time=training_time)
    return render_template("index.html")
if __name__=="__main__":
    if not os.path.exists(UPLOAD_FOLDER):
```

```
    os.makedirs(UPLOAD_FOLDER)
train_model()
app.run(port=5001,debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <title>IMAGE CLASSIFICATION USING KNN</title>

    <style>
        body {
            background-image: url('/static/uploads/3409297.jpg'); /* Use the correct path to the image */
            background-size: cover; /* Ensure the background image covers the entire screen */
            background-position: center top;
            background-repeat: no-repeat;
            height: 100vh; /* Correct height unit */
            width: 100vw;
            margin: 0;
            font-family: 'Great Vibes', cursive;
            display: flex;
            justify-content: center;
            align-items: center;
        }
        h2 {
            font-family: Georgia, 'Times New Roman', Times, serif;
            font-size: 2.5rem;
            background: -webkit-linear-gradient(#ffffff, #cccccc);
            background-clip: text;
            -webkit-text-fill-color: transparent;
        }
        h3{
            font-size: 2.0rem;
        }
        .upload-box {
            width: 500px; /* Set width */
            padding: 20px;
            text-align: center;
            backdrop-filter: blur(10px);
            background: rgba(255, 255, 255, 0.2);
            border-radius: 10px;
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            border: #fff9f9;
```

```
            border-style: dashed;
        }
        .upload-box input {
            margin-top: 15px;
        }
        .image-preview {
            margin-top: 15px;
        }
        .img-thumbnail {
            max-height: 200px;
        }
        .btn-primary {
            background: rgba(0, 123, 255, 0.2);
            border: 1px solid rgba(0, 123, 255, 0.2);
            color: #007bff;
            font-weight: bold;
            backdrop-filter: blur(10px);
        }
        .btn-primary:hover {
            background: rgba(0, 123, 255, 0.4);
            border: 1px solid rgba(0, 123, 255, 0.4);
        }
        .btn-glassy {
            background: rgba(255, 255, 255, 0.2);
            border: 1px solid rgba(255, 255, 255, 0.2);
            color: #007bff;
            font-weight: bold;
            backdrop-filter: blur(10px);
            border-radius: 10px;
        }
        .btn-glassy:hover {
            background: rgba(255, 255, 255, 0.4);
            border: 1px solid rgba(255, 255, 255, 0.4);
        }
        #selectedImage {
            width: 200px;
            height: 200px;
            object-fit: cover;
            display: block;
            margin: 0 auto;
            font-size: 1.5rem;
        }
        input{
            font-size: 1.2rem;
        }
    </style>
</head>
<body>
    <center>
    <h2 class="text-center">Image Classification Using KNN</h2>
```

```
    <div class="upload-box">
        <form action="/" method="POST" enctype="multipart/form-data">
            <label for="image" class="form-label"><h3>Select an Image</h3></label>
            <input type="file" name="image" id="image" accept="image/* onchange="previewI
             mage(event)" />
            <div class="image-preview" id="imagePreview">
                <img id="selectedImage" alt="Selected Image">
            </div>
            <button type="submit" class="btn btn-primary btn-glassy mt-3">Classify
Image</button>
        </form>
    </div>
    </center>
    <script>
        function previewImage(event) {
            const file = event.target.files[0];
            if (file) {
                const reader = new FileReader();
                reader.onload = function(e) {
                    const imageElement = document.getElementById('selectedImage');
                    imageElement.src = e.target.result;
                    document.getElementById('imagePreview').style.display = 'block';
                };
                reader.readAsDataURL(file);
            }
        }
    </script>
</body>
</html>
```

result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prediction Result</title>
    <link                                    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <link              href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap"
rel="stylesheet">
    <style>
        body {
            background-image: url('/static/uploads/bg8.jpg'); /* Use the correct path to the image
*/
            background-size: cover; /* Ensure the background image covers the entire screen */
            background-position: center top;
            background-repeat: no-repeat;
            height: 100vh; /* Correct height unit */
```

```
    width: 90vw;
    margin: 0;
    font-family: 'Great Vibes', cursive;
    display: flex;
    justify-content: center;
    align-items: center;
}
.result-box {
    width: 600px; /* Set width */
    height: 450px; /* Set height equal to width */
    padding: 20px;
    text-align: center;
    margin-top: 10px;
    backdrop-filter: blur(10px);
    background: rgba(255, 255, 255, 0.2);
    border-radius: 10px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
.result-box img {
    max-width: 100%; /* Ensure the image fits within the box */
    max-height: 50%; /* Limit height to avoid overflowing */
    margin-bottom: 10px;
    align-items: center;
}
.btn-secondary {
    background: rgba(108, 117, 125, 0.2);
    border: 1px solid rgba(108, 117, 125, 0.2);
    color: white;
    font-weight: bold;
    backdrop-filter: blur(10px);
}
.btn-secondary:hover {
    background: rgba(108, 117, 125, 0.4);
    border: 1px solid rgba(108, 117, 125, 0.4);
}
.btn-primary {
    background: rgba(108, 117, 125, 0.2);
    border: 1px solid rgba(108, 117, 125, 0.2);
    color: white;
    font-weight: bold;
    backdrop-filter: blur(10px);
}
.btn-primary:hover {
    background: rgba(108, 117, 125, 0.4);
    border: 1px solid rgba(108, 117, 125, 0.4);
}
h1 {
```

```
            font-family: 'Pacifico', cursive;
            font-weight: 700;
            background: -webkit-linear-gradient(#178a99, #ffffff);
            background-clip: text;
            -webkit-text-fill-color: transparent;
            text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
        }
    </style>
</head>
<body>
<div class="container mt-5">
          
    <center><h1 class="text-center">Prediction Result</h1></center>
    <center>
    <div class="result-box">
        <img src="{{url_for('static', filename='uploads/' + uploaded_image)}}" alt="Uploaded
        Image" class="img-fluid">
        <p><strong>Predicted Label:</strong> {{predicted_label}}</p>
        <p><strong>Training Accuracy:</strong>{{"%.2f"|format(train_accuracy*100)}}%</p>
        <p><strong>Testing Accuracy:</strong> {{test_accuracy*100|round(2)}}%</p>
        <p><strong>Total Time:</strong> {{training_time|round(2)}} Seconds</p>
        <center><a href="/" class="btn btn-secondary">Go Back</a>
   <a href=http://127.0.0.1:5009/ class="btn btn primary">Home</a></ce
nter>
    </div>
    </center>
</div>
</body>
</html>
```

SVM

app.py

```
from flask import Flask,render_template,request
import os
import cv2
import numpy as np
import time
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
app=Flask(__name__)
dataset_path="E:/iii BCA/MAJOR PROJECT1/Dataset"
upload_folder=os.path.join('svm_project','static','uploads')
os.makedirs(upload_folder,exist_ok=True)
def load_data():
    data=[]
    labels=[]
```

```
    for label_folder in os.listdir(dataset_path):
        label_path=os.path.join(dataset_path,label_folder)
        if os.path.isdir(label_path):
            for img_file in os.listdir(label_path):
                img_path=os.path.join(label_path,img_file)
                img=cv2.imread(img_path)
                if img is not None:
                    img=cv2.resize(img,(64,64))
                    img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
                    data.append(img.flatten())
                    labels.append(label_folder)
    return np.array(data),np.array(labels)
X,y=load_data()
le=LabelEncoder()
y_encoded=le.fit_transform(y)
X_train,X_test,y_train,y_test=train_test_split(X,y_encoded,test_size=0.2,random_state=42)
start_time=time.time()
svm_model=SVC(kernel='linear')
svm_model.fit(X_train,y_train)
training_time=time.time()-start_time
train_accuracy=accuracy_score(y_train,svm_model.predict(X_train))
test_accuracy=accuracy_score(y_test,svm_model.predict(X_test))
@app.route("/",methods=["GET","POST"])
def index():
    if request.method=="POST":
        file=request.files["image"]
        if file:
            img_path=os.path.join(upload_folder,file.filename)
            file.save(img_path)
            print(f"Image saved at:{img_path}")
            image_id=file.filename.split("_")[0]
            img=cv2.imread(img_path)
            img=cv2.resize(img,(64,64))
            img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            img_flattened=img.flatten().reshape(1,-1)
            prediction=svm_model.predict(img_flattened)
            predicted_label=le.inverse_transform(prediction)[0]
render_template("result.html",uploaded_image=file.filename,predicted_label=predicted_label
,train_accuracy=train_accuracy,test_accuracy=test_accuracy,training_time=training_time)
    return render_template("index.html")
if __name__=="__main__":
    app.run(port=5003,debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
```

```
    <title>IMAGE CLASSIFICATION USING SVM</title>
    <style>
        body {
            background-image: url('/static/uploads/3409297.jpg'); /* Use the correct path to the image */
            background-size: cover; /* Ensure the background image covers the entire screen */
            background-position: center top;
            background-repeat: no-repeat;
            height: 100vh; /* Correct height unit */
            width: 100vw;
            margin: 0;
            font-family: 'Great Vibes', cursive;
            display: flex;
            justify-content: center;
            align-items: center;
        }
        h2 {
            font-family: Georgia, 'Times New Roman', Times, serif;
            font-size: 2.5rem;
            background: -webkit-linear-gradient(#ffffff, #cccccc);
            background-clip: text;
            -webkit-text-fill-color: transparent;
        }
        h3{
            font-size: 2.0rem;
        }
        .upload-box {
            width: 500px; /* Set width */
            padding: 20px;
            text-align: center;
            backdrop-filter: blur(10px);
            background: rgba(255, 255, 255, 0.2);
            border-radius: 10px;
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            border: #fff9f9;
            border-style: dashed;
        }
        .upload-box input {
            margin-top: 15px;
        }
        .image-preview {
            margin-top: 15px;
        }
        .img-thumbnail {
            max-height: 200px;
        }
        .btn-primary {
```

```
            background: rgba(0, 123, 255, 0.2);
            border: 1px solid rgba(0, 123, 255, 0.2);
            color: #007bff;
            font-weight: bold;
            backdrop-filter: blur(10px);
        }
        .btn-primary:hover {
            background: rgba(0, 123, 255, 0.4);
            border: 1px solid rgba(0, 123, 255, 0.4);
        }
        .btn-glassy {
            background: rgba(255, 255, 255, 0.2);
            border: 1px solid rgba(255, 255, 255, 0.2);
            color: #007bff;
            font-weight: bold;
            backdrop-filter: blur(10px);
            border-radius: 10px;
        }
        .btn-glassy:hover {
            background: rgba(255, 255, 255, 0.4);
            border: 1px solid rgba(255, 255, 255, 0.4);
        }
        #selectedImage {
            width: 200px;
            height: 200px;
            object-fit: cover;
            display: block;
            margin: 0 auto;
            font-size: 1.5rem;
        }
        input{
            font-size: 1.2rem;
        }
    </style>
</head>
<body>
    <center>
    <h2 class="text-center">Image Classification Using SVM</h2>
    <div class="upload-box">
        <form action="/" method="POST" enctype="multipart/form-data">
            <label for="image" class="form-label"><h3>Select an Image</h3></label>
             <input type="file" name="image" id="image" accept="image/*" onchange="preview
             Image(event)" />
            <div class="image-preview" id="imagePreview">
                <img id="selectedImage" alt="Selected Image">
            </div>
            <button type="submit" class="btn btn-primary btn-glassy mt-3">Classify
Image</button>
        </form>
    </div>
```

```
    </center>
    <script>
        function previewImage(event) {
            const file = event.target.files[0];
            if (file) {
                const reader = new FileReader();
                reader.onload = function(e) {
                    const imageElement = document.getElementById('selectedImage');
                    imageElement.src = e.target.result;
                    document.getElementById('imagePreview').style.display = 'block';
                };
                reader.readAsDataURL(file);
            }
        }
    </script>
</body>
</html>
```

result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prediction Result</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">
    <style>
        body {
            background-image: url('/static/uploads/bg8.jpg'); /* Use the correct path to the image */
            background-size: cover; /* Ensure the background image covers the entire screen */
            background-position: center top;
            background-repeat: no-repeat;
            height: 100vh; /* Correct height unit */
            width: 90vw;
            margin: 0;
            font-family: 'Great Vibes', cursive;
            display: flex;
            justify-content: center;
            align-items: center;
        }
        .result-box {
            width: 600px; /* Set width */
            height: 450px; /* Set height equal to width */
            padding: 20px;
            text-align: center;
```

```
            margin-top: 10px;
            backdrop-filter: blur(10px);
            background: rgba(255, 255, 255, 0.2);
            border-radius: 10px;
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
        }
        .result-box img {
            max-width: 100%; /* Ensure the image fits within the box */
            max-height: 50%; /* Limit height to avoid overflowing */
            margin-bottom: 10px;
            align-items: center;
        }
        .btn-secondary {
            background: rgba(108, 117, 125, 0.2);
            border: 1px solid rgba(108, 117, 125, 0.2);
            color: white;
            font-weight: bold;
            backdrop-filter: blur(10px);
        }
        .btn-secondary:hover {
            background: rgba(108, 117, 125, 0.4);
            border: 1px solid rgba(108, 117, 125, 0.4);
        }
        .btn-primary {
            background: rgba(108, 117, 125, 0.2);
            border: 1px solid rgba(108, 117, 125, 0.2);
            color: white;
            font-weight: bold;
            backdrop-filter: blur(10px);
        }
        .btn-primary:hover {
            background: rgba(108, 117, 125, 0.4);
            border: 1px solid rgba(108, 117, 125, 0.4);
        }
        h1 {
            font-family: 'Pacifico', cursive;
            font-weight: 700;
            background: -webkit-linear-gradient(#178a99, #ffffff);
            background-clip: text;
            -webkit-text-fill-color: transparent;
            text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
        }
    </style>
</head>
<body>
<div class="container mt-5">
          
```

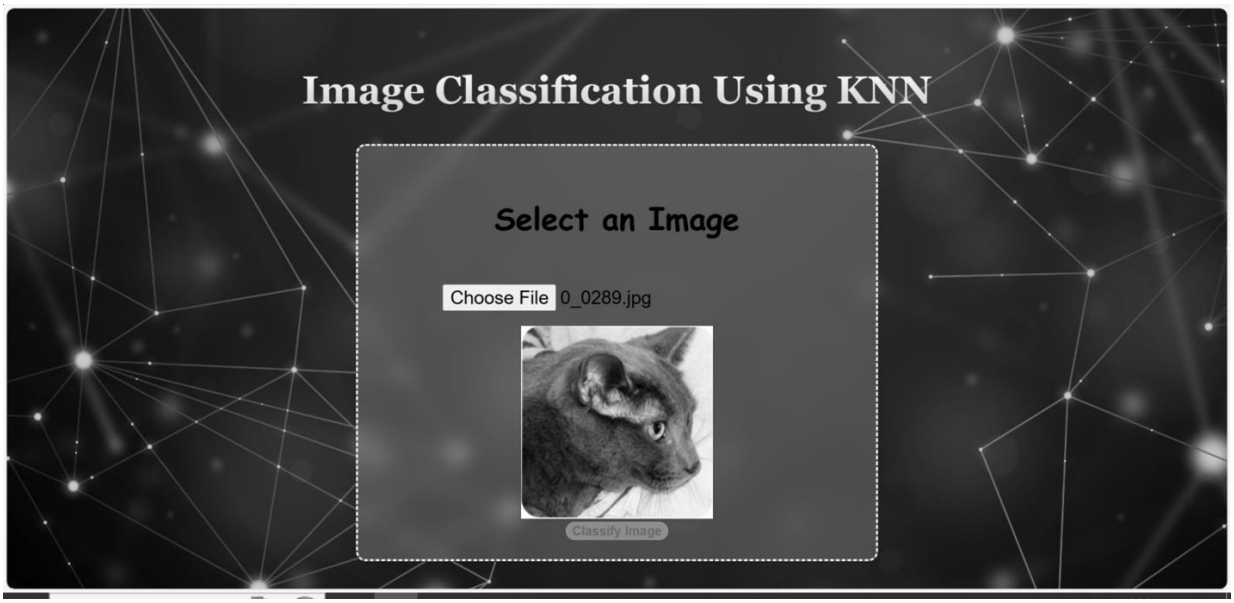
```
    <center><h1 class="text-center">Prediction Result</h1></center>
    <center>
    <div class="result-box">
        <img src="{{url_for('static', filename='uploads/' + uploaded_image)}}" alt="Uploaded
Image" class="img-fluid">
        <p><strong>Predicted Label:</strong> {{predicted_label}}</p>
        <p><strong>Training                                                    Accuracy:</strong>
{{"%.2f"|format(train_accuracy*100)}}%</p>
        <p><strong>Testing Accuracy:</strong> {{test_accuracy*100|round(2)}}%</p>
        <p><strong>Total Time:</strong> {{training_time|round(2)}} Seconds</p>
        <center><a href="/" class="btn btn-secondary">Go Back</a>
           <a          href="http://127.0.0.1:5009/"          class="btn          btn-
primary">Home</a></center>
    </div>
    </center>
</div>
</body>
</html>
```

SAMPLE OUTPUT

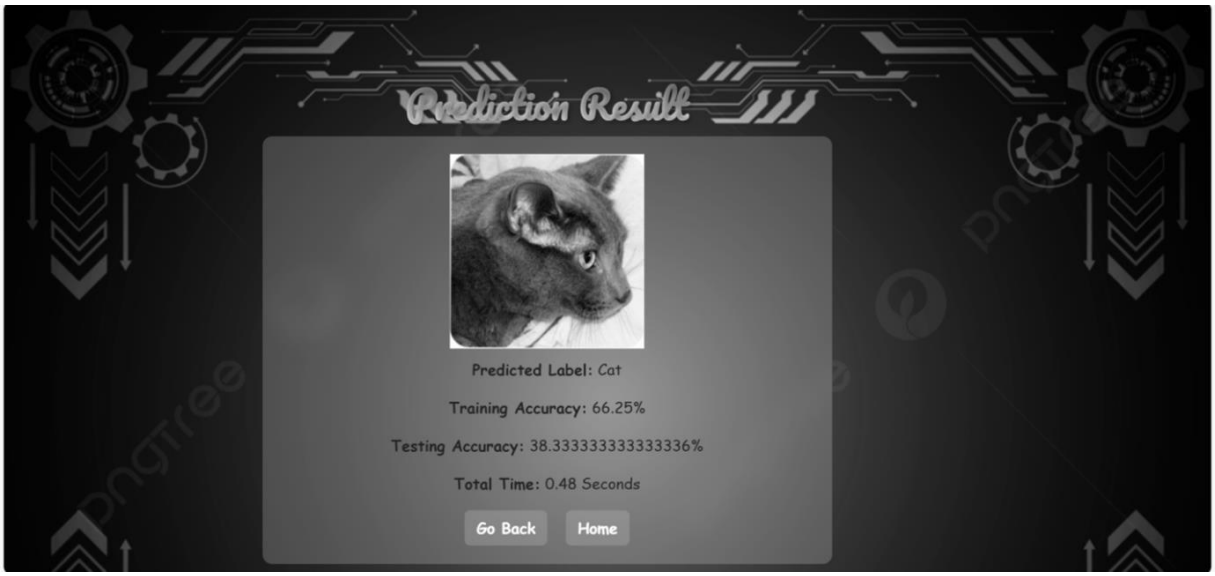
HOME PAGE

KNN (K-Nearest Neighbors)

INDEX.HTML

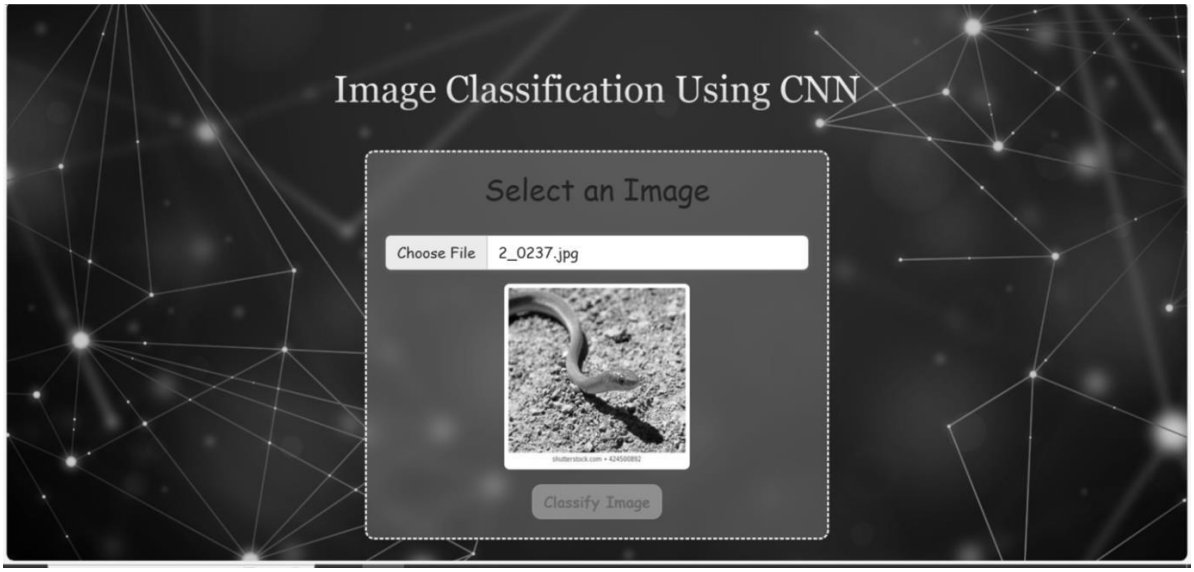


RESULT.HTML

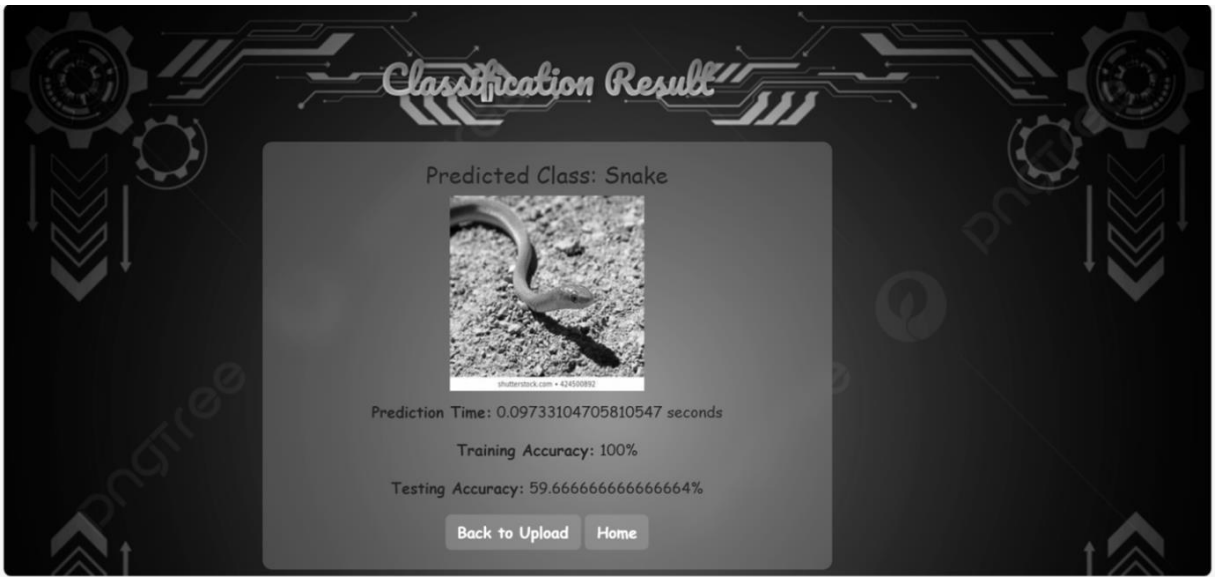


CNN (Convolutional Neural Networks)

INDEX.HTML

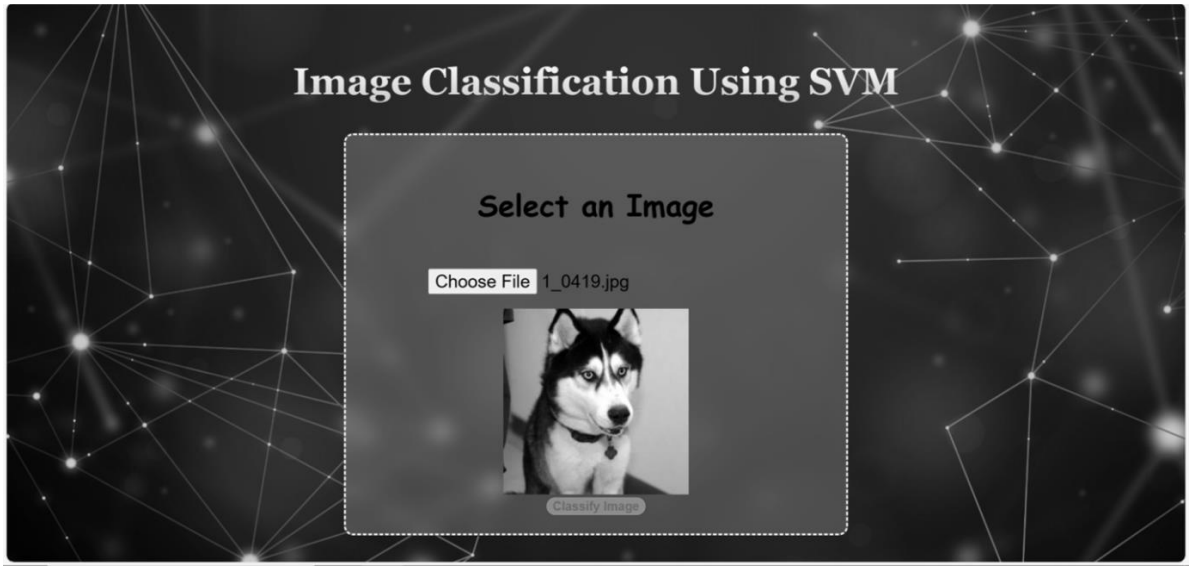


RESULT.HTML

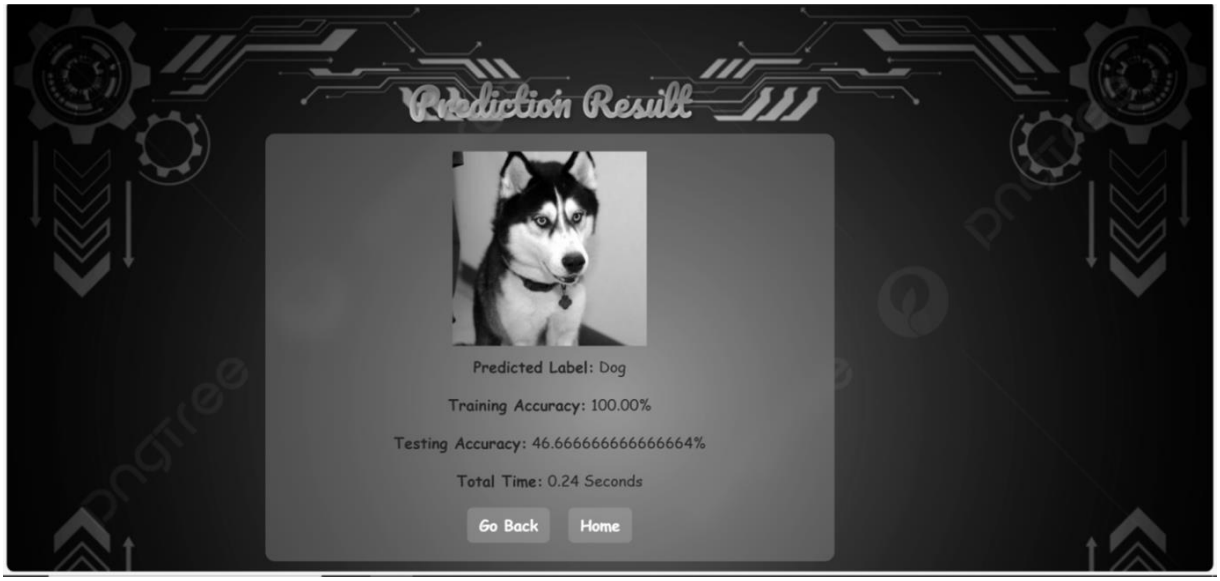


SVM (Support Vector Machine)

INDEX.HTML

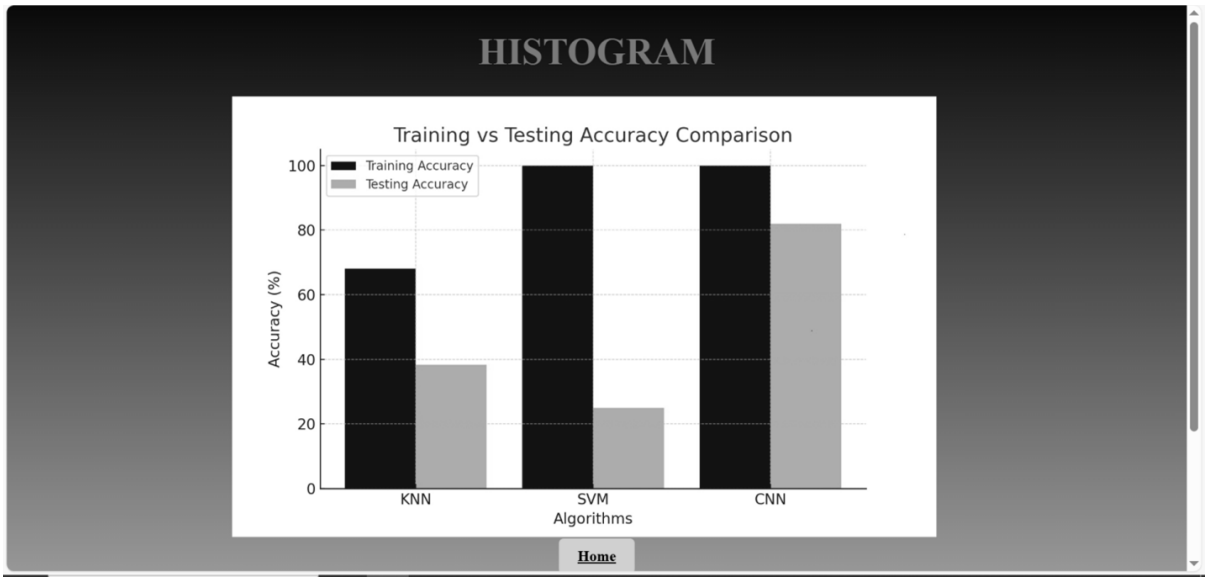


RESULT.HTML



GRAPH

GRAPH.HTML



CONCLUSION

8. CONCLUSION

This project successfully demonstrated the application of machine learning techniques, specifically Convolutional Neural Network (CNN), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM), for texture image classification and analysis. The experimental results showed that CNN outperformed KNN and SVM in terms of classification accuracy, highlighting the effectiveness of deep learning in capturing complex texture patterns. Through a comprehensive comparison of the three models, this study conclusively proved that CNN is the best approach for texture image classification, achieving superior accuracy and robustness. The study also revealed the importance of feature extraction and selection in improving the performance of machine learning models. Overall, this project contributes to the development of efficient and accurate texture image classification system, which can be applied in various fields such as medical imaging, material science, and quality inspection.

FUTURE ENHANCEMENT

9.FUTURE ENHANCEMENT

Future enhancement of this project can be directed towards exploring advanced deep learning architectures, such as ResNet, Inception, or DenseNet, to further improve the classification accuracy. Additionally, techniques like data augmentation, transfer learning, and ensemble methods can be employed to enhance the robustness and generalizability of the model. Another potential direction is to integrate texture analysis with other image analysis techniques, such as object detection or segmentation, to develop a more comprehensive image understanding system. Furthermore, applying texture image classification and analysis to real-world applications, such as medical image analysis, material inspection, or quality control, can be a promising area of future research.

BIBLIOGRAPHY

10. BIBLIOGRAPHY

The following websites were referred during the anylysis and execution phase of the project.

WEBSITES:

YOUTUBE : https://www.youtube.com/

W3SCHOOLS : https://www.w3schools.com/

GITHUB : **https://github.com/**

GEEKSFORGEEKS: **https://www.geeksforgeeks.org/courses**