Project Report: Image Classification using Convolutional Neural Networks (CNNs)

Name : Lakshita Vishhnu

Introduction

This project involves building an image classification system using Convolutional Neural Networks (CNNs). CNNs are a class of deep neural networks that have proven highly effective for image-related tasks. This project aims to provide hands-on experience with deep learning, image processing, and model development.

Dataset

Utilized the CIFAR-10 dataset, which is a collection of 60,000 32x32 color images in 10 different classes. The dataset is divided into 50,000 training images and 10,000 testing images. The 10 classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

Data Preprocessing

Preprocessing steps are crucial for improving the model's performance and generalization. The following preprocessing steps were applied:

1. **Resizing**: Ensuring all images are of uniform size (32x32 pixels).
2. **Normalization**: Normalizing the pixel values to a range of 0 to 1.
3. **Data Augmentation**: Applying random transformations (like rotation, zoom, horizontal flip) to increase the diversity of the training set.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Data Augmentation

```
datagen = ImageDataGenerator(
   rescale=1.0/255.0,
   rotation_range=15,
   width_shift_range=0.1,
   height_shift_range=0.1,
   horizontal_flip=True
)
```

Model Architecture

We implemented a CNN architecture inspired by VGGNet, a well-known deep learning model for image classification. The architecture includes:

1. **Convolutional Layers**: Extract features from the input images using filters.

2. **Max-Pooling Layers**: Reduce the spatial dimensions of the feature maps.

3. **Fully Connected Layers**: Perform classification based on the extracted features.

4. **Activation Functions**: Use ReLU for non-linearity and Softmax for the output layer.

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Model Training

The model was trained using the following parameters:

Optimizer: Adam

Loss Function: Categorical Crossentropy

Metrics: Accuracy

Epochs: 25

Batch Size: 64

```python
history = model.fit(datagen.flow(x_train, y_train, batch_size=64),
                    epochs=25, validation_data=(x_test, y_test))
```

Model Evaluation

The trained model was evaluated on the test dataset. The evaluation metrics included:

Accuracy: Overall accuracy of the model.

Precision: The precision for each class.

Recall: The recall for each class.

F1-Score: The F1-score for each class.

```python
from sklearn.metrics import classification_report

y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

report = classification_report(y_true, y_pred_classes, target_names=class_names)
print(report)
```

Deployment Instructions

1. **Environment Setup**: Ensure you have Python and necessary libraries installed (TensorFlow, Flask, etc.).

2. **Model Saving** : Save the trained model in the `.keras` format.

3. **Flask Application**: Create a Flask web application for deploying the model.

```python
# Save the model
model.save('cnn_model.keras')
```

Flask Application

Here is the simplified Flask application code for deploying the model:

app.py:

```python
from flask import Flask, request, render_template_string
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

app = Flask(__name__)

# Load the model
model = load_model('cnn_model.keras')

# HTML templates with CSS
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</title>
<style>
  body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #e6f7ff;
    margin: 0;
  }
  .container {
    text-align: center;
    background: #ffffff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  }
  h1 {
    color: #333333;
  }
  form {
    margin-top: 20px;
  }
  input[type="file"] {
    padding: 10px;
    border: 1px solid #dddddd;
    border-radius: 4px;
    background: #ffffff;
  }
  button {
```

```
            padding: 10px 20px;
            margin-top: 10px;
            background: #99ccff;
            border: none;
            color: #ffffff;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background: #80bfff;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Image Classification using CNN</h1>
        <form action="/predict" method="post" enctype="multipart/form-data">
            <input type="file" name="file" accept="image/*">
            <button type="submit">Predict</button>
        </form>
    </div>
</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>
<style>
  body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #e6f7ff;
    margin: 0;
  }
  .container {
    text-align: center;
    background: #ffffff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  }
  h1 {
    color: #333333;
  }
  p {
    font-size: 1.2em;
    color: #666666;
  }
  a {
    display: inline-block;
    margin-top: 20px;
    padding: 10px 20px;
    background: #99ccff;
    color: #ffffff;
```

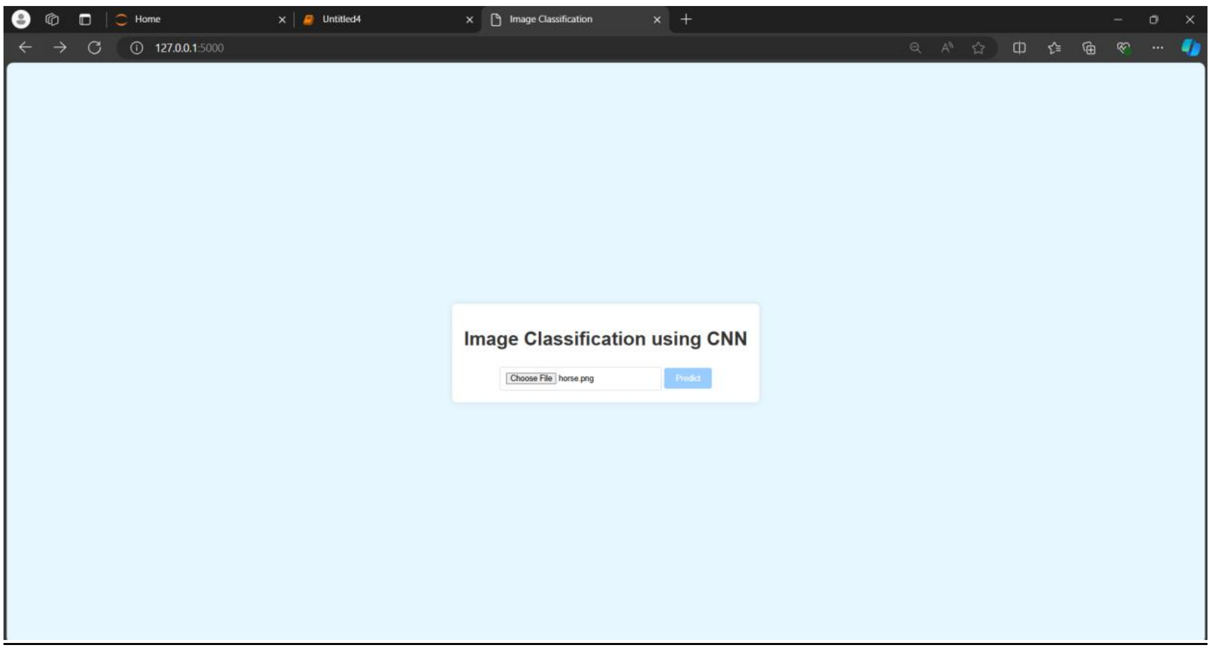
```
            text-decoration: none;
            border-radius: 4px;
        }
        a:hover {
            background: #80bfff;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Prediction Result</h1>
        <p>Predicted Class: {{ prediction }}</p>
        <a href="/">Classify another image</a>
    </div>
</body>
</html>
"""

@app.route('/')
def index():
    return render_template_string(index_html)

@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return "No file part", 400
    file = request.files['file']
    if file.filename == '':
        return "No selected file", 400
    if file:
        img = image.load_img(file, target_size=(32, 32))
        img = image.img_to_array(img)
        img = np.expand_dims(img, axis=0)
        img = img / 255.0

        prediction = model.predict(img)
        class_idx = np.argmax(prediction)
        return render_template_string(result_html, prediction=class_idx)

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

Conclusion

This project demonstrated the complete workflow of building, training, and deploying an image classification system using CNNs. The hands-on experience gained through this project is valuable for understanding deep learning and its applications in image processing.

By following the steps outlined in this report, you can build and deploy a robust image classification system using CNNs.

Final Webpage Screenshot:


Home
Untitled4
Image Classification
127.0.0.1:5000
Image Classification using CNN
Choose File
horse.png
Predict