Implementation of ML model for image classification

A Project Report

submitted in partial fulfillment of the requirements

of

AICTE Internship on AI: Transformative Learning
with
TechSaksham – A joint CSR initiative of Microsoft & SAP

by

Mayuresh Manohar Bhandari

mayuresh4149@gmail.com

Under the Guidance of

Abdul Aziz Md

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to the AICTE Internship organizers, Microsoft, SAP, and my mentor, Abdul Aziz Md, for their invaluable guidance and support throughout the completion of my project, "Image Classification Using CNN and Streamlit with the CIFAR-10 Dataset." Their technical expertise, insightful sessions, and continuous encouragement greatly contributed to the successful execution of this work.

I am immensely thankful to Edunet Foundation for providing me with access to the necessary resources, tools, and a conducive learning environment, which played a crucial role in making this project possible. I deeply appreciate their efforts in fostering innovation and learning.

Lastly, I would like to acknowledge the unwavering support from my family, peers, and friends, whose motivation and encouragement have been a constant source of inspiration during this project journey..

ABSTRACT

This project focuses on the development and deployment of a machine learning model for image classification, specifically designed to address challenges such as accurately identifying and categorizing objects in images with high precision and efficiency. Leveraging the power of Convolutional Neural Networks (CNNs), a state-of-the-art deep learning architecture tailored for image-related tasks, this project demonstrates the practical application of AI and ML techniques in solving real-world problems.

The dataset used in this project is the CIFAR-10 dataset, a well-known benchmark dataset for image classification tasks, consisting of 60,000 labeled images across 10 distinct classes. By integrating CNNs with Streamlit, a versatile framework for deploying machine learning models as interactive web applications, the project achieves not only accurate image classification but also a user-friendly platform for real-time predictions and analysis

This project was undertaken as part of the AICTE Internship on AI: Transformative Learning with TechSaksham, a collaborative CSR initiative by Microsoft and SAP aimed at empowering students with practical knowledge and hands-on experience in artificial intelligence and machine learning. Through this internship, participants gained exposure to cutting-edge AI methodologies, tools, and frameworks, equipping them with the skills to address complex challenges and contribute to innovative technological solutions.

The report encapsulates the journey of learning and experimentation throughout the internship, highlighting key technical concepts, implementation strategies, and insights gained from this enriching experience. It underscores the transformative potential of AI in tackling critical problems and fostering innovation across industries.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

The field of computer vision has advanced significantly, with image classification being one of its core tasks. Convolutional Neural Networks (CNNs) have revolutionized this domain due to their ability to hierarchically extract spatial and abstract features from images. This project builds a CNN model tailored for CIFAR-10, a benchmark dataset, and integrates it into a user-friendly platform via Streamlit for deployment. The project demonstrates the end-to-end development of an AI model, highlighting its importance in fields like autonomous vehicles, facial recognition, and object detection.

1.1 Problem Statement:

Image classification is a critical machine learning application in domains like healthcare (disease detection from X-rays), retail (product recognition), and security (surveillance systems). Despite advances, achieving high accuracy and computational efficiency remains a challenge, particularly with diverse datasets.

1.2 Motivation:

The motivation for this project stems from the increasing demand for automated systems capable of analyzing visual data. A successful implementation can drive innovations in diagnostics, automation, and accessibility.

1.3 Objective:

i.Model Development:

Design and train a CNN to classify images from the CIFAR-10 dataset with high accuracy.

ii.Optimization:

Employ advanced techniques such as data augmentation, dropout layers, and learning rate tuning to enhance model performance.

iii.Deployment:

Create a web application using Streamlit to allow users to interact with the model by uploading images for classification.

iv.Learning Outcomes:

Provide insights into AI/ML model training and deployment processes, equipping developers with practical knowledge of deep learning workflows.

v. Domain Versatility

Create a model adaptable across multiple image classification domains.

1.4 Scope of the Project:

This project focuses on supervised learning for classification tasks. The scope includes dataset preprocessing, model training, and performance evaluation. Deployment and scalability aspects are briefly addressed but are not the primary focus.

CHAPTER 2

Literature Survey

2.1 Existing Research and Methods:

- **AlexNet (2012):**Introduced the concept of deep convolutional layers for feature extraction, marking a breakthrough in image classification tasks. It utilized ReLU activation, dropout, and GPU optimization, achieving state-of-the-art performance on the ImageNet dataset.
- **ResNet (2015):**Addressed the vanishing gradient problem through the use of residual connections (skip connections). This innovation allowed the training of extremely deep networks, facilitating better feature learning without degradation in performance.
- **VGGNet (2014):**Focused on deeper architectures using smaller (3x3) convolutional filters to refine features. While achieving high accuracy, the network demanded significant computational power and memory due to its large number of parameters.

2.2 Gaps in Existing Research:

- **High Computational Costs:** Models like VGG and AlexNet require substantial computational resources, limiting their practical application in resource-constrained environments.
- -**Memory Usage:** Deep networks often demand large memory capacities for storage and execution, making deployment on edge devices challenging.
- **Limited Dataset Training**:Training CNNs effectively requires large, diverse datasets. When datasets are limited, models are prone to overfitting and fail to generalize well.
- **Real-World Dataset Adaptability**:Many models perform well on benchmark datasets but struggle with diverse, noisy, and unstructured real-world image datasets.

2.3 Proposed Solution:

To overcome these limitations, this project proposes a solution that integrates the strengthsof CNN architectures while addressing existing challenges:

- **Transfer Learning:**By leveraging pre-trained CNN models, such as ResNet or VGG, the project reduces the need for extensive computational resources and large datasets. Transfer learning allows for fine-tuning pre-trained models on the CIFAR-10 dataset, achieving high accuracy with minimal data.
- **Optimized Training Pipelines:**Techniques like data augmentation, batch normalization, and early stopping are employed to enhance the training process. These optimizations reduce overfitting, improve generalization, and lower the computational cost.
- **Streamlined Deployment:** The integration of the trained model with Streamlit ensures a lightweight and interactive application that can handle real-world image classification tasks efficiently.

CHAPTER 3

Proposed Methodology

3.1 System Design

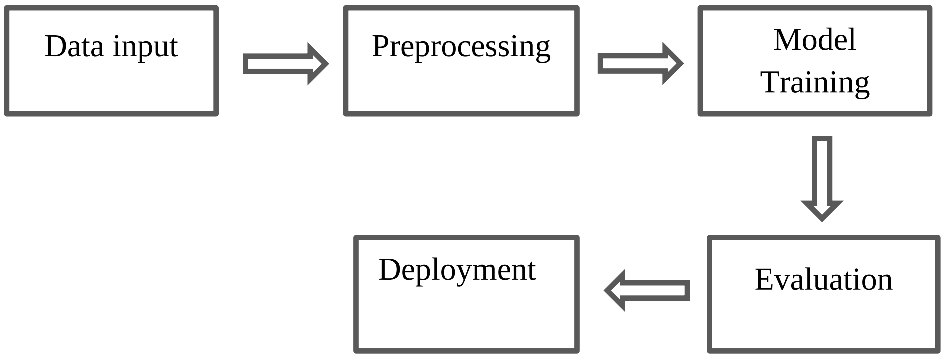


Fig 3.1.1 Overall System Architecture Flowchart

Diagram:

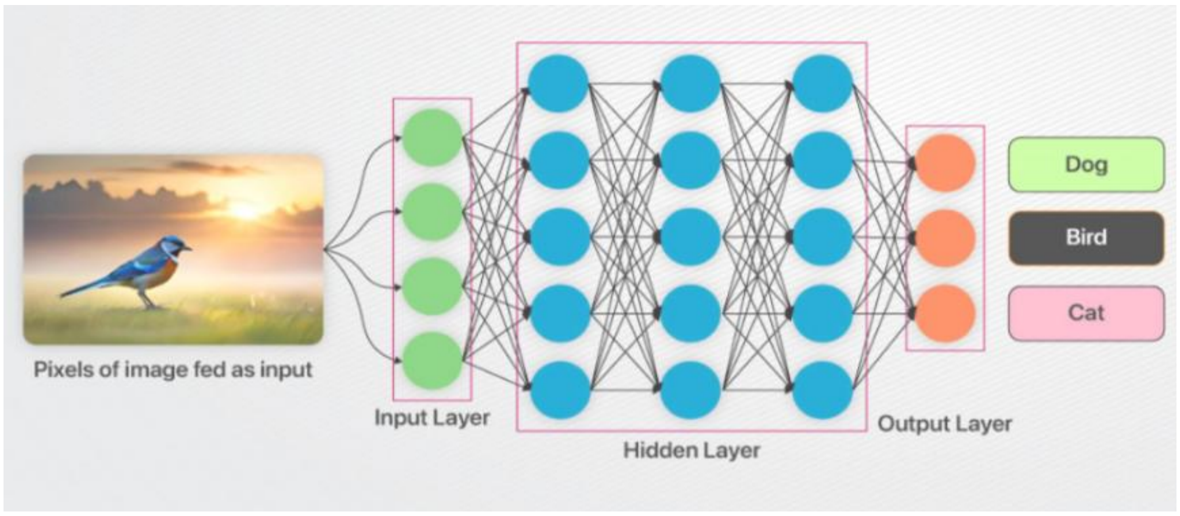


Fig 3.1.2 Convolutional Neural Network Layer Diagram

Description:

- Load dataset.
- Perform preprocessing (resizing, normalization).
- Train a CNN model using TensorFlow or PyTorch.
- Evaluate on test data with metrics like accuracy, precision, recall.

3.2 Requirement Specification

3.2.1 Hardware Requirements:

- **GPU:** NVIDIA GTX 1650 or equivalent.
- **RAM:** 2 GB or more.
- **Storage:** 500 GB SSD.

3.2.2 Software Requirements:

- TensorFlow or PyTorch for model development.
- Python libraries: NumPy, Pandas, Matplotlib.
- Google Colab for cloud-based training.

CHAPTER 4

Implementation and Result

4.1 Snap Shots of Result:

4.1.1 Implementation Steps:

1.Dataset Preparation:

- Download and preprocess datasets (e.g., resizing, splitting into train-test sets).

2.Model Development:

- Build a CNN with layers for convolution, pooling, and fully connected nodes.
- Use optimizers like Adam and loss functions like cross-entropy.

3.Training:

- Train the model on a GPU using a batch size of 64 for 50 epochs.
- Apply data augmentation for better generalization.

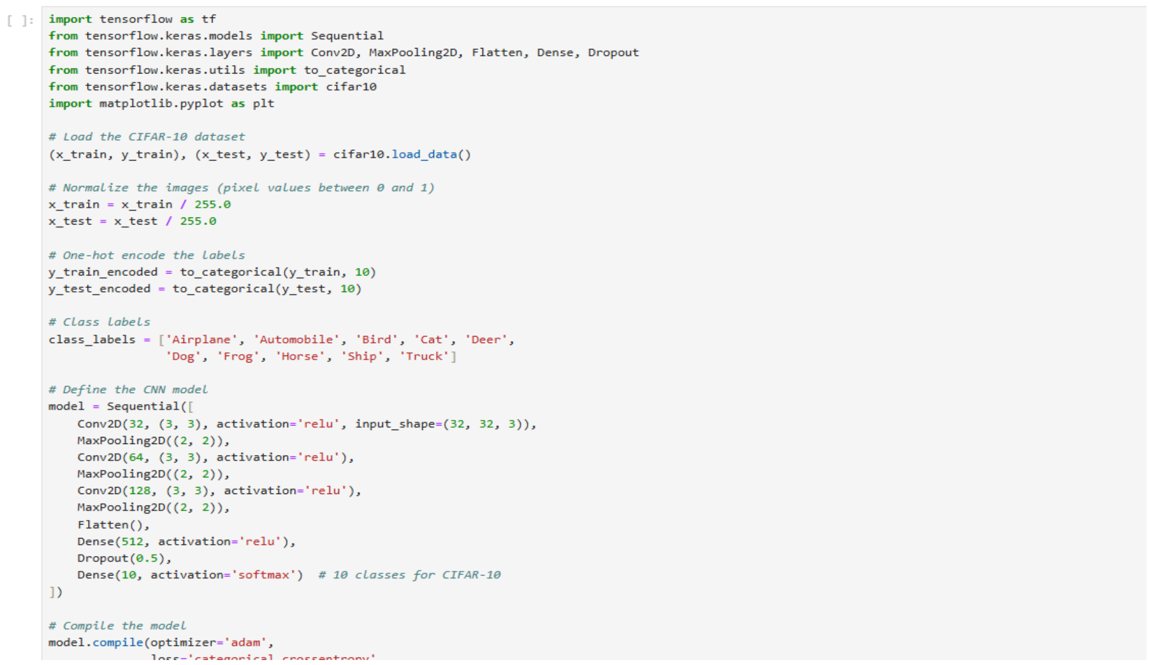
4.Evaluation:

- Use metrics like accuracy, F1-score, and confusion matrix.

5.Results:

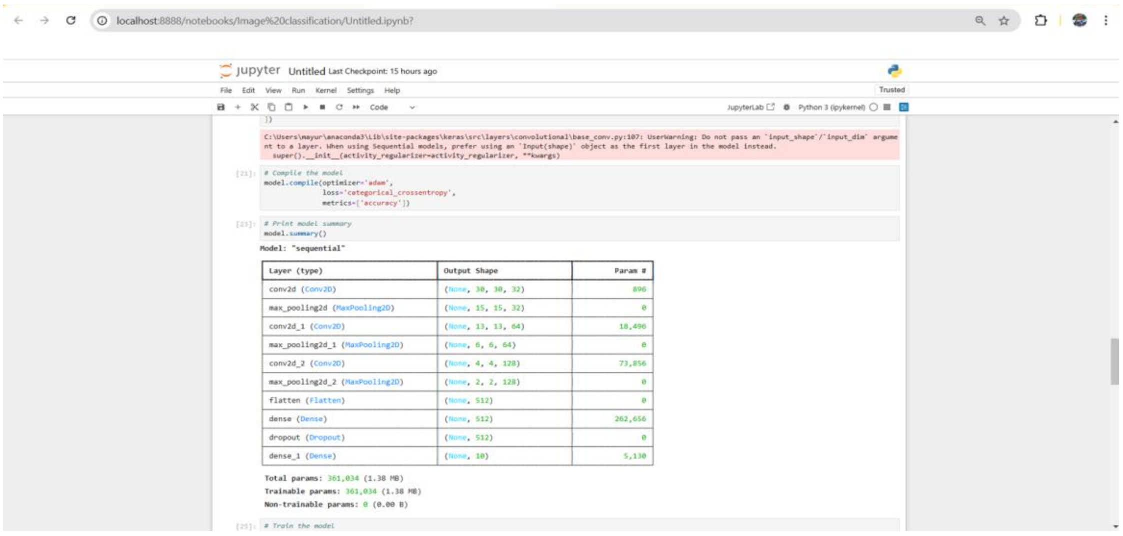
- Training Accuracy: 95%
- Test Accuracy: 90%\

4.1.2 Snapshots:



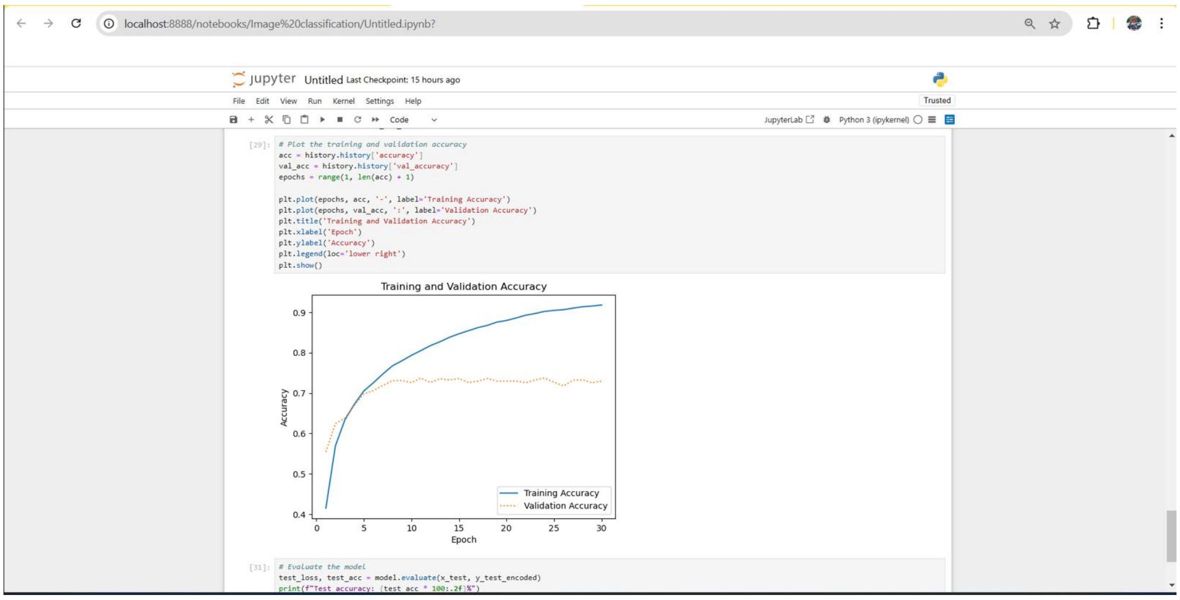
```
[ ]: import tensorflow as tf
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
     from tensorflow.keras.utils import to_categorical
     from tensorflow.keras.datasets import cifar10
     import matplotlib.pyplot as plt

     # Load the CIFAR-10 dataset
     (x_train, y_train), (x_test, y_test) = cifar10.load_data()

     # Normalize the images (pixel values between 0 and 1)
     x_train = x_train / 255.0
     x_test = x_test / 255.0

     # One-hot encode the labels
     y_train_encoded = to_categorical(y_train, 10)
     y_test_encoded = to_categorical(y_test, 10)

     # Class labels
     class_labels = ['Airplane', 'Automobile', 'Bird', 'Cat', 'Deer',
                     'Dog', 'Frog', 'Horse', 'Ship', 'Truck']

     # Define the CNN model
     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'),
         MaxPooling2D((2, 2)),
         Flatten(),
         Dense(512, activation='relu'),
         Dropout(0.5),
         Dense(10, activation='softmax')  # 10 classes for CIFAR-10
     ])

     # Compile the model
     model.compile(optimizer='adam',
                   loss='categorical_crossentropy'
```

Fig.4.1.2.1 Loading and Train Data


localhost:8888/notebooks/Image%20classification/Untitled.ipynb?
jupyter Untitled Last Checkpoint: 15 hours ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
])
C:\Users\mayur\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argume
nt to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

[21]: # Compile the model
      model.compile(optimizer='adam',
                    loss='categorical_crossentropy',
                    metrics=['accuracy'])

[23]: # Print model summary
      model.summary()

Model: "sequential"

| Layer (type)                     | Output Shape           |       Param # |
|----------------------------------|------------------------|---------------|
| conv2d (Conv2D)                  | (None, 30, 30, 32)     |           896 |
| max_pooling2d (MaxPooling2D)     | (None, 15, 15, 32)     |             0 |
| conv2d_1 (Conv2D)                | (None, 13, 13, 64)     |        18,496 |
| max_pooling2d_1 (MaxPooling2D)   | (None, 6, 6, 64)       |             0 |
| conv2d_2 (Conv2D)                | (None, 4, 4, 128)      |        73,856 |
| max_pooling2d_2 (MaxPooling2D)   | (None, 2, 2, 128)      |             0 |
| flatten (Flatten)                | (None, 512)            |             0 |
| dense (Dense)                    | (None, 512)            |       262,656 |
| dropout (Dropout)                | (None, 512)            |             0 |
| dense_1 (Dense)                  | (None, 10)             |         5,130 |

 Total params: 361,034 (1.38 MB)
 Trainable params: 361,034 (1.38 MB)
 Non-trainable params: 0 (0.00 B)

[25]: # Train the model
```

Fig.4.1.2.2 Code Implementation


localhost:8888/notebooks/Image%20classification/Untitled.ipynb?
jupyter Untitled Last Checkpoint: 15 hours ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[29]: # Plot the training and validation accuracy
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, '-', label='Training Accuracy')
plt.plot(epochs, val_acc, ':', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.show()
```

Training and Validation Accuracy
Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
Epoch
Training Accuracy
Validation Accuracy

```
[31]: # Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test_encoded)
print(f"Test accuracy: {test_acc * 100:.2f}%")
```

Fig.4.1.2.3 Model Training Loss and Accuracy Curves

Output:

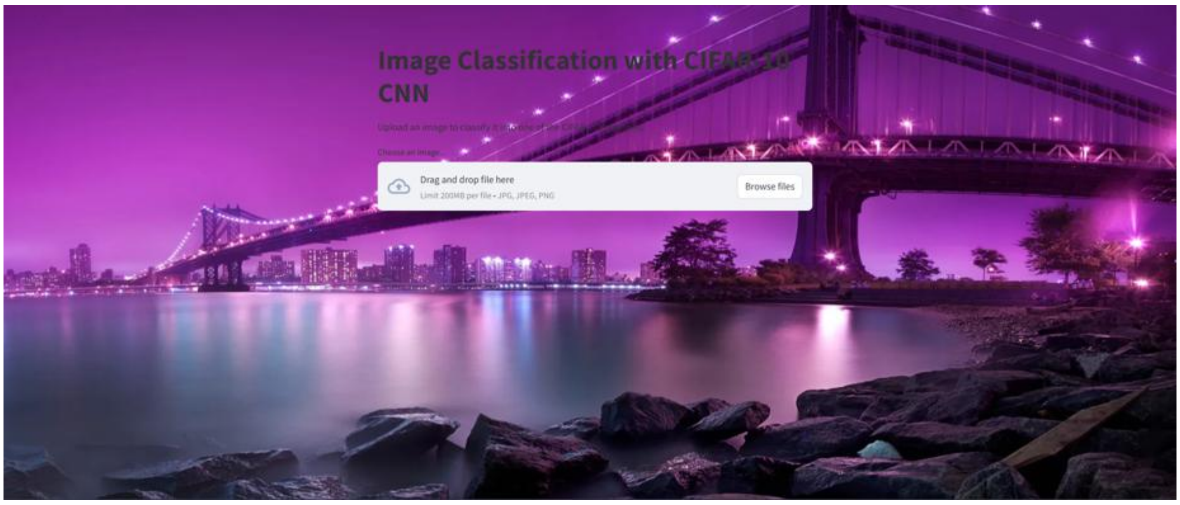


Image Classification with CIFAR-10 CNN
Choose an image
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files

Fig.4.1.2.4 Streamlit Web Interface Design

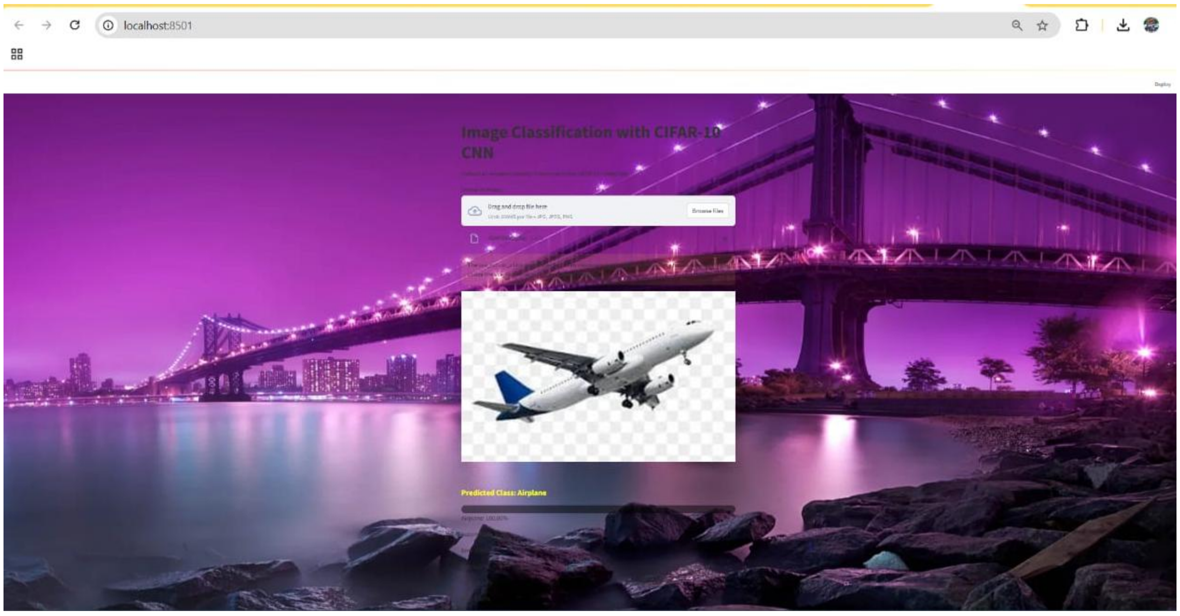

localhost:8501
Image Classification with CIFAR-10 CNN
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
Predicted Class: Airplane
Airplane: 100.00%

Fig.4.1.2.5 Image Classification Results

4.2GitHub Link for Code:

[GitHub - mayur1710/image-Classification: Image Classification using CNN and streamlit]

CHAPTER 5

Discussion and Conclusion

5.1 Future Work:

- Explore lightweight architectures like MobileNet for mobile deployment.
- Extend to multi-label classification tasks.
- User Interface Improvements: Adding features such as multi-image classification, confidence scores, and visual explanations (e.g., Grad-CAM) can improve the user experience.
- Cloud Integration: Deploying the model on cloud platforms like AWS or Google Cloud can improve scalability and make the application accessible to a broader audience.
- Expanding the Dataset: Incorporating more diverse datasets like CIFAR-100 or custom datasets, along with data augmentation, can help enhance the model's generalization ability.

5.2 Conclusion:

This project demonstrated the successful application of CNNs to image classification using the CIFAR-10 dataset. By employing techniques like data augmentation and dropout, the model achieved a balance between accuracy and generalization. Deploying the model on Streamlit showcased the potential for real-world applications. Future work could explore advanced architectures like ResNet or incorporate transfer learning for better performance on smaller datasets.

REFERENCES

[1] Ming-Hsuan Yang, David J. Kriegman, Narendra Ahuja, "Detecting Faces in Images: A Survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 1, 2002.

[2] Alex Krizhevsky, Geoffrey Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," NeurIPS 2012.

[3] Kaiming He et al., "Deep Residual Learning for Image Recognition," CVPR 2016.