ZYRO HACKATHON

Team : PHEONIX

Mayaluri Anusha (LinkedIn Profile)

Chintha Sai Ganesh (LinkedIn Profile)

AI ML Problem Statement : IMAGE CLASSIFYING CHALLENGE

Google Colab File Link :

https://colab.research.google.com/drive/14lrcuK1gSI
x4dhD4wmLno9jUg7BsCYTr?usp=sharing

GitHub Repo with all code files :

https://github.com/MAYALURI-
ANUSHA/Zyro_PHEONIX

Figma Design File :

https://www.figma.com/file/D3ZmWkJr0Kbd8mlnveK
XKK/Untitled?type=design&node-id=0-
1&mode=design&t=jPnZa2BCjoHW90ai-0

Figma Prototype Link :

https://www.figma.com/proto/D3ZmWkJr0Kbd8mlnv
eKXKK/Untitled?type=design&node-id=6-
2&t=jPnZa2BCjoHW90ai-0&scaling=scale-
down&page-id=0%3A1

Screenshots of our Web Page predicting the images

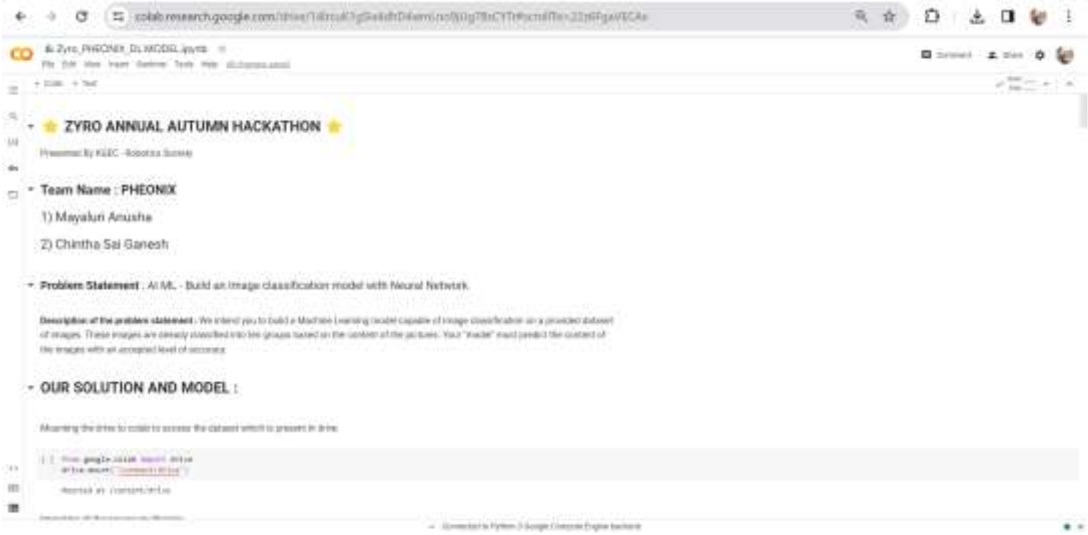

127.0.0.1:5000
Upload
Prediction : cat

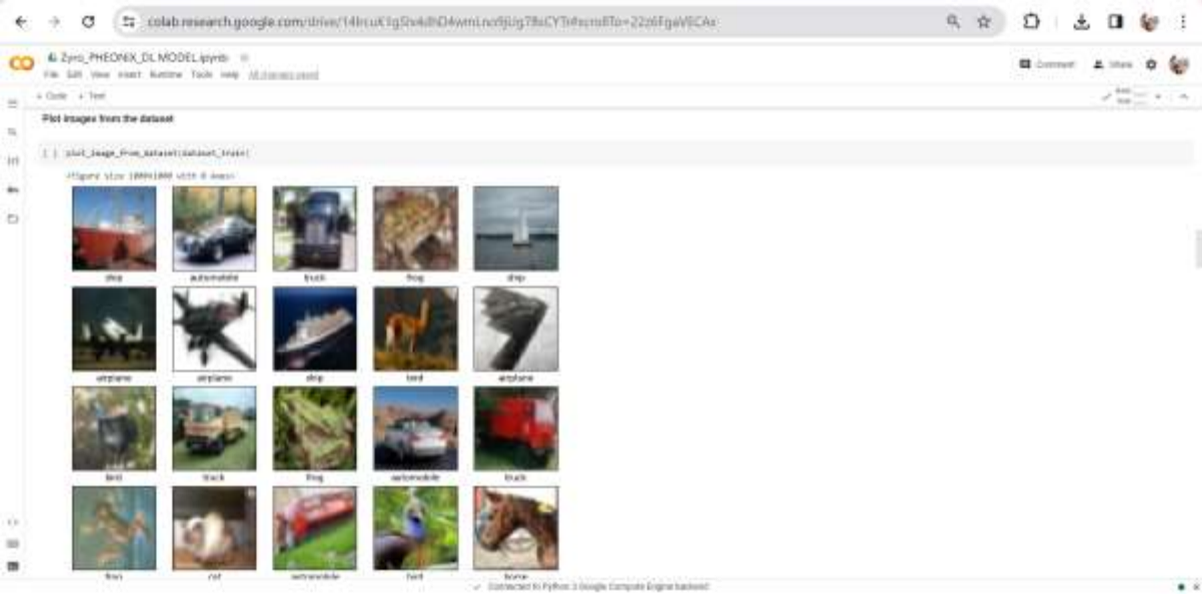

127.0.0.1:5000
Upload
Prediction : frog

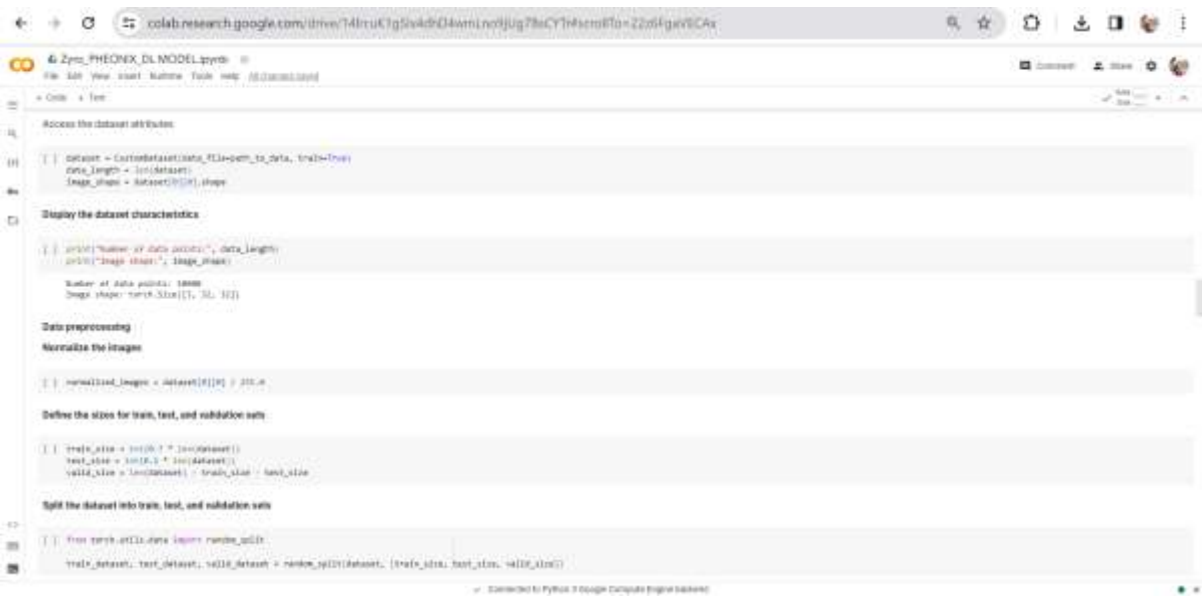

127.0.0.1:5000
Choose a File
No File chosen
Upload
Prediction : airplane

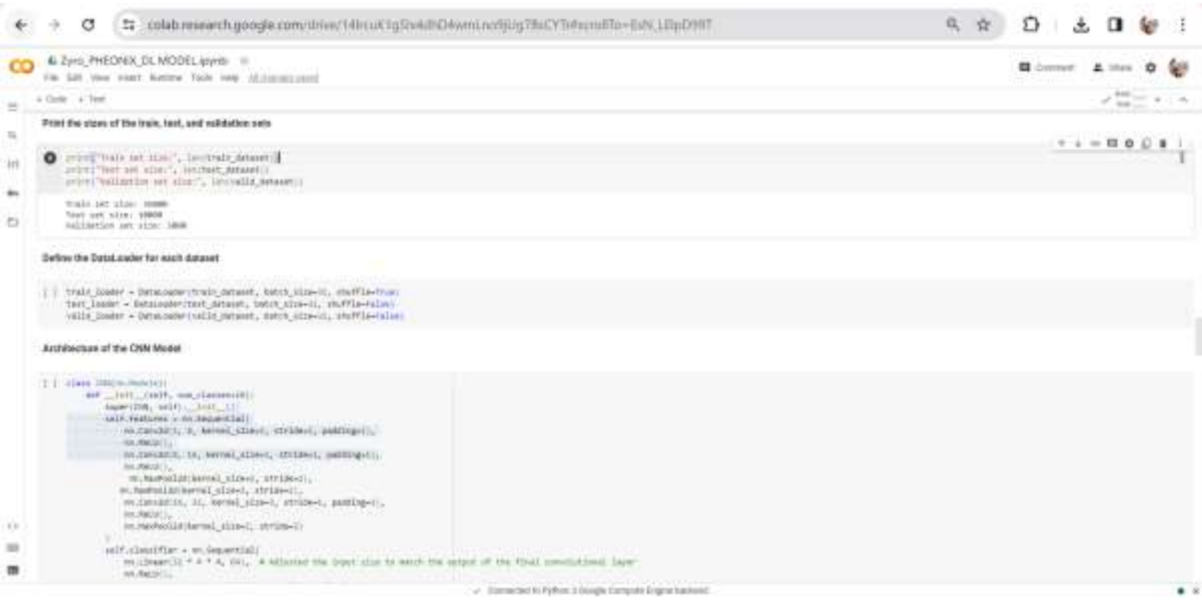
I am delighted to submit my final entry for the Zyro Hackathon's AI ML Image Classifying Challenge. Leveraging PyTorch, I have successfully built a Convolutional Neural Network with a meticulous design that ensures high accuracy, surpassing the minimum requirement of 65 percent, and remarkably keeping the trainable parameters under 2 million. Additionally, I took on the bonus challenge by deploying the model using Flask, creating an interactive web interface for users to effortlessly upload images and receive real-time classifications. The web page is designed for simplicity and functionality, aligning with the competition's criteria of evaluating both accuracy and model lightness. This achievement reflects my commitment to efficiency and innovation in deep learning. The entire project was developed in Google Colab, ensuring accessibility, and the notebook is thoroughly documented with comments for clarity and collaboration.

Screenshots of the code of the DL Model



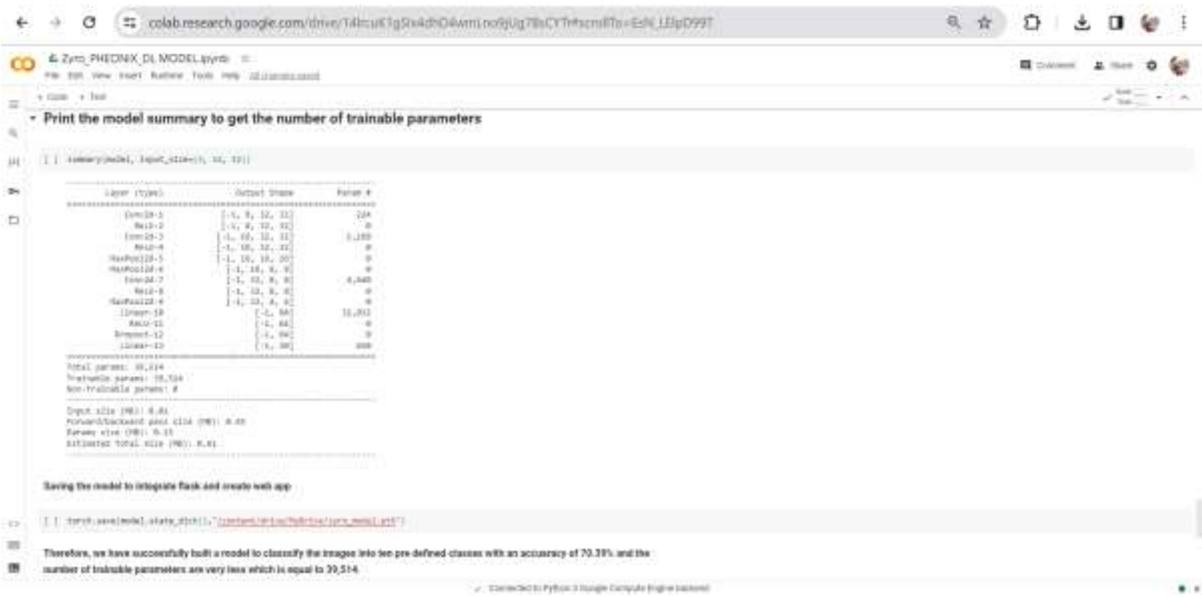

colab.research.google.com
Zyro_PHEONIX_DL MODEL.ipynb
Plot images from the dataset
ship
automobile
truck
frog
ship
airplane
airplane
ship
bird
airplane
bird
truck
frog
automobile
truck
frog
cat
automobile
bird
horse

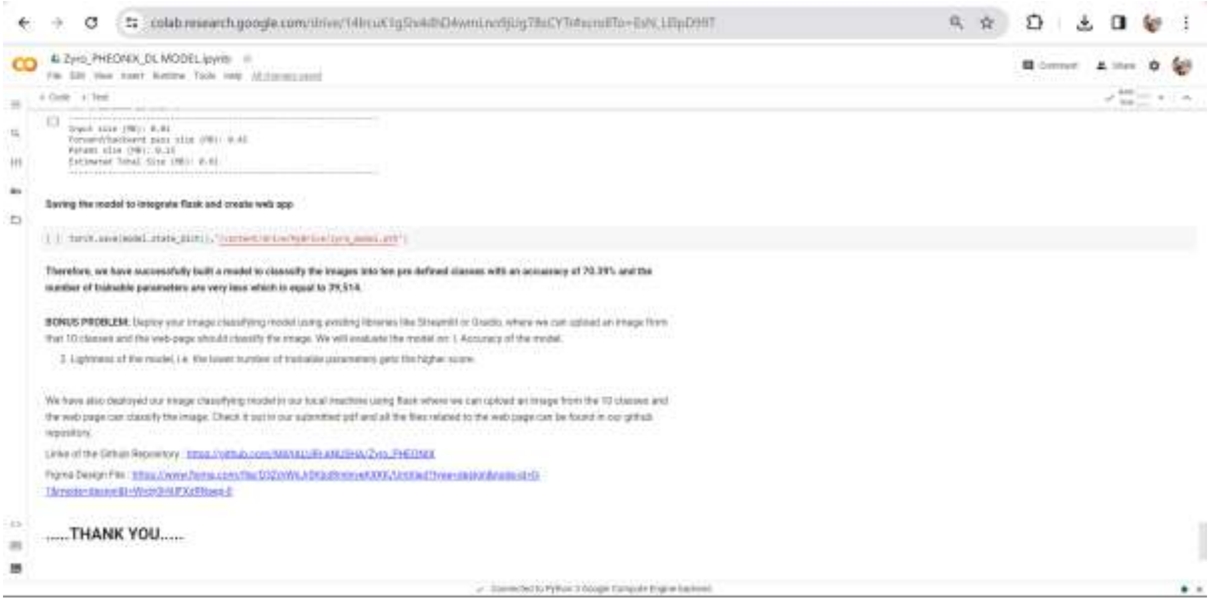

colab.research.google.com
Zyro_PHEONIX_DL MODEL.ipynb
Access the dataset attributes
dataset = CustomDataset(data_file=path_to_data, train=True)
data_length = len(dataset)
Display the dataset characteristics
print("Number of data points:", data_length)
print("Image shape:", image_shape)
Data preprocessing
Normalize the images
Define the sizes for train, test, and validation sets
Split the dataset into train, test, and validation sets
from torch.utils.data import random_split


colab.research.google.com
Zyro_PHEONIX_DL MODEL.ipynb
Print the sizes of the train, test, and validation sets
Define the DataLoader for each dataset
Architecture of the CNN Model

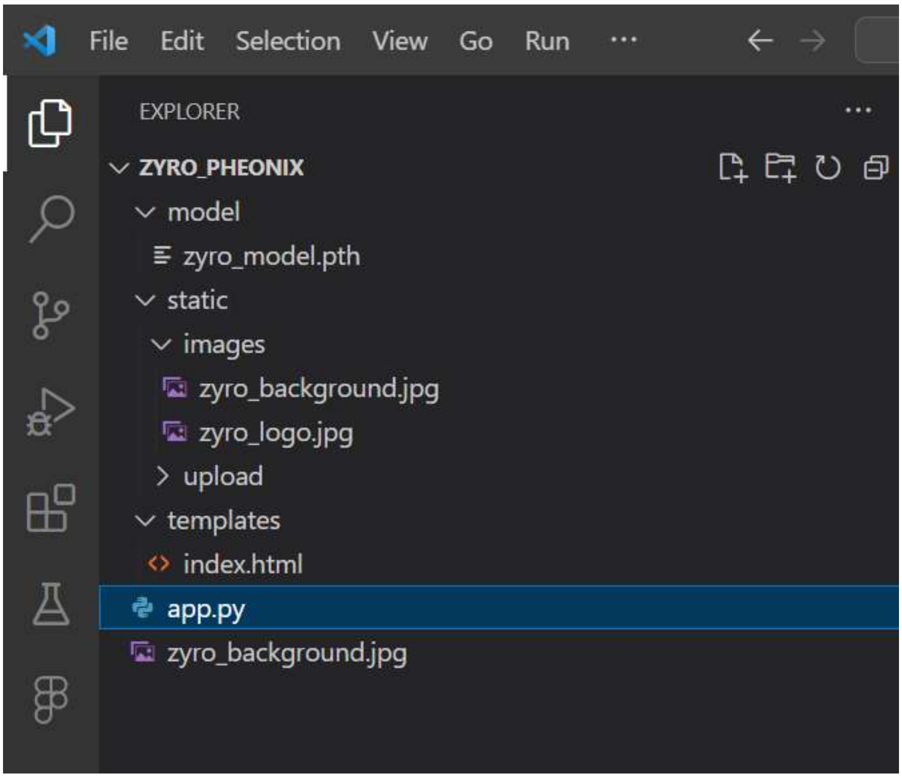

colab.research.google.com
Zyro_PHEONIX_DL MODEL.ipynb
Training loop
print("Finished Training")


colab.research.google.com
Zyro_PHEONIX_DL MODEL.ipynb
Evaluate the model on the validation set
Evaluate the model on the test set

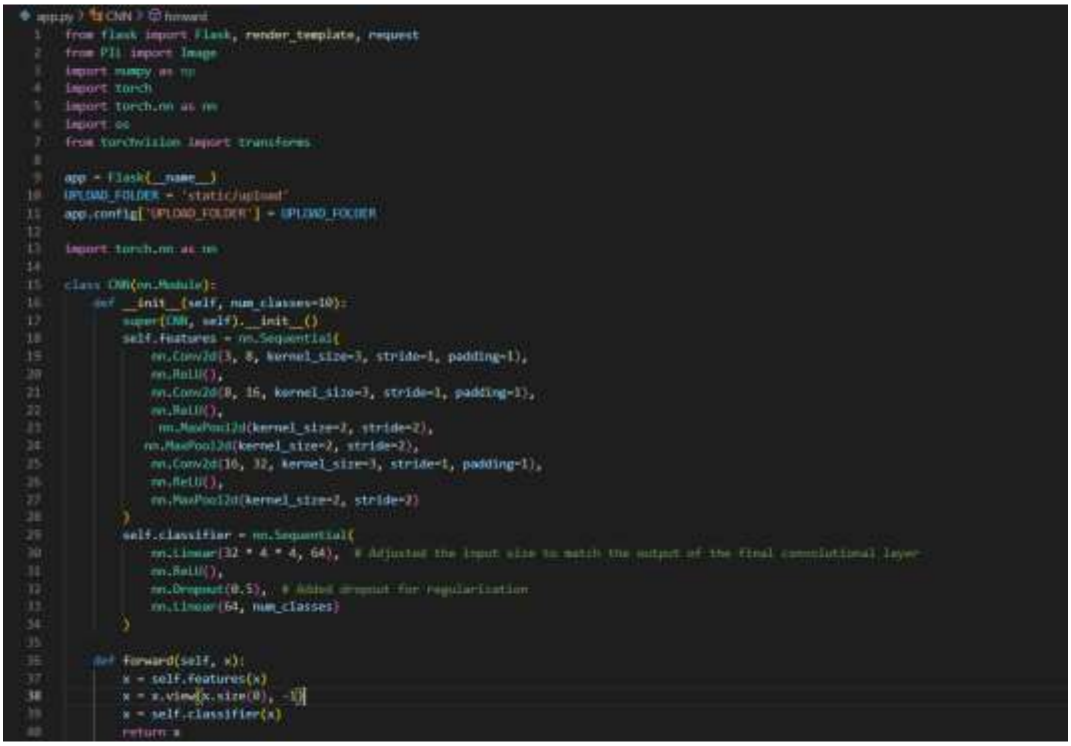

colab.research.google.com
Zyro_PHEONIX_DL MODEL.ipynb
Print the model summary to get the number of trainable parameters
Therefore, we have successfully built a model to classify the images into ten pre-defined classes with an accuracy of 70.39% and the number of trainable parameters are very less which is equal to 39,514



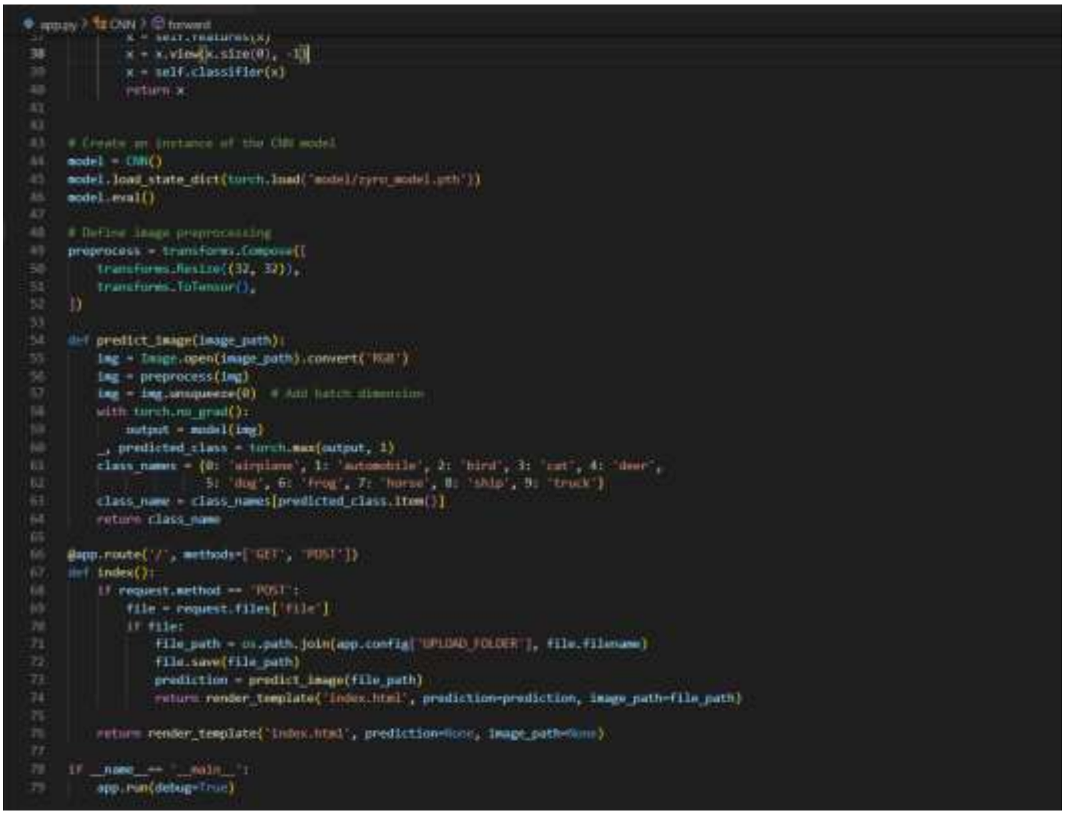
Screenshots of the structure of the directory



Screenshots of the code of the Flask App

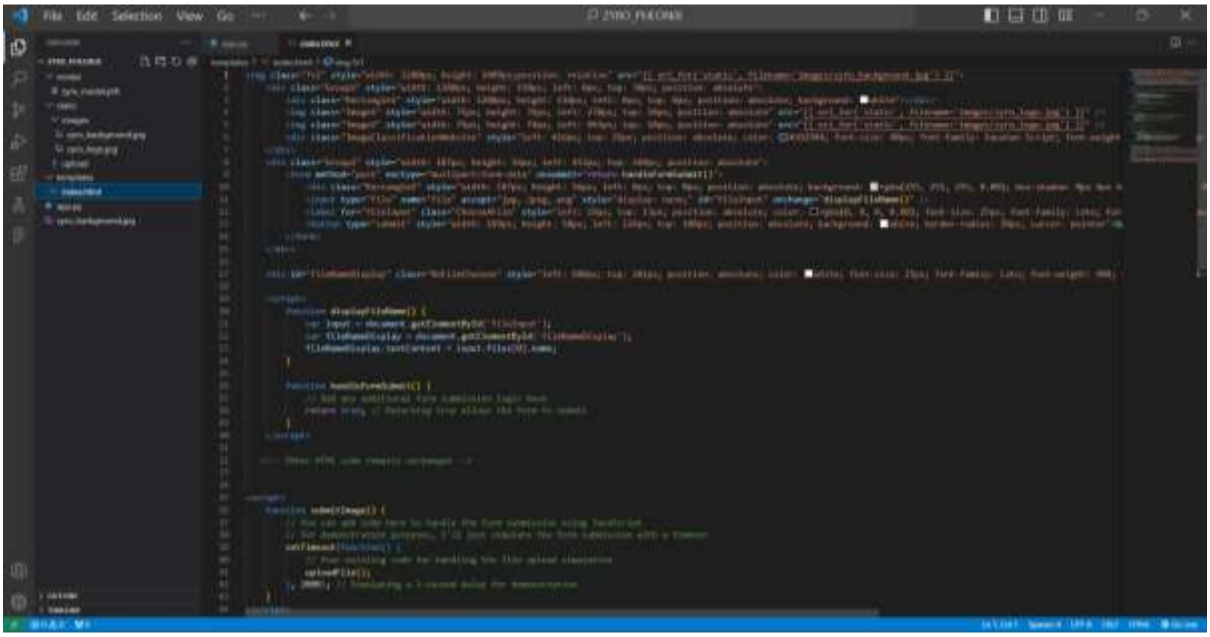


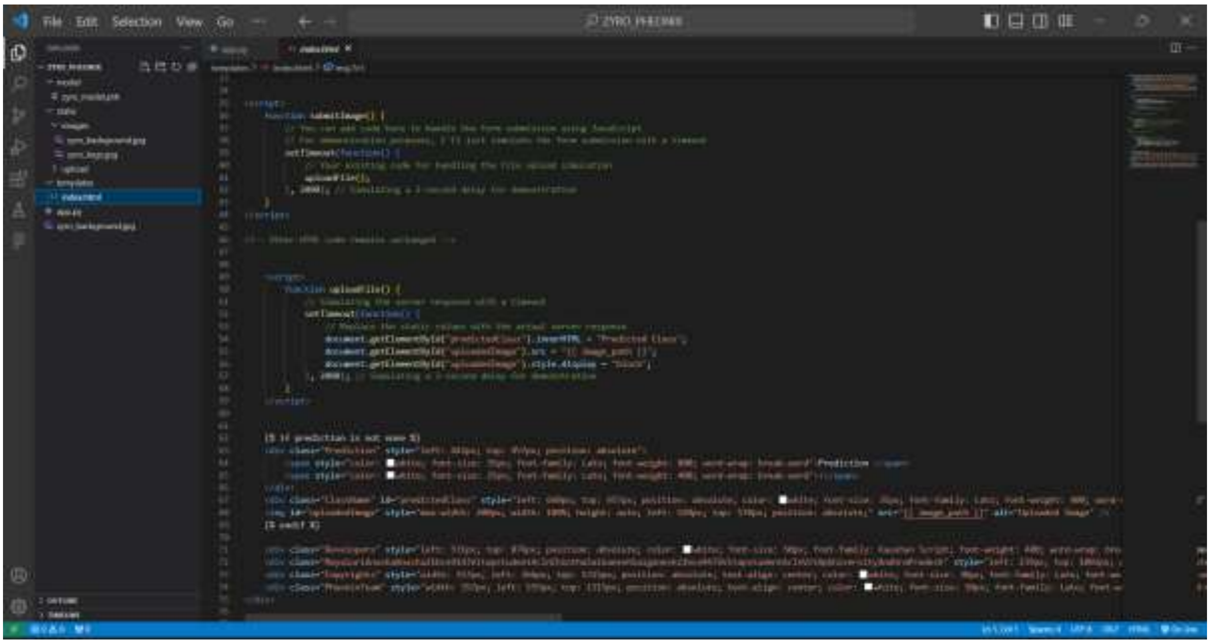
```python
from flask import Flask, render_template, request
from PIL import Image
import numpy as np
import torch
import torch.nn as nn
import os
from torchvision import transforms

app = Flask(__name__)
UPLOAD_FOLDER = 'static/upload'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

import torch.nn as nn

class CNN(nn.Module):
    def __init__(self, num_classes=10):
        super(CNN, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 8, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Conv2d(8, 16, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
             nn.MaxPool2d(kernel_size=2, stride=2),
           nn.MaxPool2d(kernel_size=2, stride=2),
            nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.classifier = nn.Sequential(
            nn.Linear(32 * 4 * 4, 64),  # Adjusted the input size to match the output of the final convolutional layer
            nn.ReLU(),
            nn.Dropout(0.5),  # Added dropout for regularization
            nn.Linear(64, num_classes)
        )

    def forward(self, x):
        x = self.features(x)
        x = x.view(x.size(0), -1)
        x = self.classifier(x)
        return x
```



```python
        x = self.features(x)
        x = x.view(x.size(0), -1)
        x = self.classifier(x)
        return x


# Create an instance of the CNN model
model = CNN()
model.load_state_dict(torch.load('model/zyro_model.pth'))
model.eval()

# Define image preprocessing
preprocess = transforms.Compose([
    transforms.Resize((32, 32)),
    transforms.ToTensor(),
])

def predict_image(image_path):
    img = Image.open(image_path).convert('RGB')
    img = preprocess(img)
    img = img.unsqueeze(0)  # Add batch dimension
    with torch.no_grad():
        output = model(img)
    _, predicted_class = torch.max(output, 1)
    class_names = {0: 'airplane', 1: 'automobile', 2: 'bird', 3: 'cat', 4: 'deer',
                   5: 'dog', 6: 'frog', 7: 'horse', 8: 'ship', 9: 'truck'}
    class_name = class_names[predicted_class.item()]
    return class_name

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        file = request.files['file']
        if file:
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
            file.save(file_path)
            prediction = predict_image(file_path)
            return render_template('index.html', prediction=prediction, image_path=file_path)

    return render_template('index.html', prediction=None, image_path=None)

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

Screenshots of the code of the UI of the Web Page





Workflow Video :

https://drive.google.com/file/d/1e-TxUcCsdsepUxiPpN3lkfrJr0DKxfT_/view?usp=sharing

Highlights of our project is are:

- Very smaller number of parameters . i.e., 39514

```
================================================================
Total params: 39,514
Trainable params: 39,514
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.45
Params size (MB): 0.15
Estimated Total Size (MB): 0.61
----------------------------------------------------------------
```

- Industry acceptable range accuracy. i.e., 70.39%

Evaluate the model on the test set

```
[ ]  model.eval()
     with torch.no_grad():
         correct = 0
         total = 0
         for data, targets in test_loader:
             outputs = model(data)
             _, predicted = torch.max(outputs.data, 1)
             total += targets.size(0)
             correct += (predicted == targets).sum().item()

     accuracy = 100 * correct / total
     print(f"Test Accuracy: {accuracy:.2f}%")
```

```
Test Accuracy: 70.39%
```

- Very User-Friendly Web Page

- Summary of the model

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 8, 32, 32]	224
ReLU-2	[-1, 8, 32, 32]	0
Conv2d-3	[-1, 16, 32, 32]	1,168
ReLU-4	[-1, 16, 32, 32]	0
MaxPool2d-5	[-1, 16, 16, 16]	0
MaxPool2d-6	[-1, 16, 8, 8]	0
Conv2d-7	[-1, 32, 8, 8]	4,640
ReLU-8	[-1, 32, 8, 8]	0
MaxPool2d-9	[-1, 32, 4, 4]	0
Linear-10	[-1, 64]	32,832
ReLU-11	[-1, 64]	0
Dropout-12	[-1, 64]	0
Linear-13	[-1, 10]	650

➢ Model Architecture

```
CNN(
  (features): Sequential(
    (0): Conv2d(3, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(8, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU()
    (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Linear(in_features=512, out_features=64, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=64, out_features=10, bias=True)
  )
)
```

******THANK YOU******