***Methods in model creation and model deployment in esp32 board for both 2D(image) and 1D data.***

***-S Theerth Krish***

***CB.SC.U4AIE23071***

**Why ESP-32 Boards?**

ESP32 boards are favoured for model creation and deployment in embedded machine learning applications due to their cost-effectiveness, built-in Wi-Fi and Bluetooth capabilities, and dual-core processing power. They offer sufficient memory for handling lightweight models and support real-time operating systems for efficient task management. The large and active community around ESP32 provides extensive libraries, tools, and support, including TensorFlow Lite for Microcontrollers, simplifying model deployment. Additionally, ESP32 boards are energy-efficient, making them ideal for battery-powered IoT applications. Their compatibility with development environments like Arduino IDE and ESP-IDF, along with their versatility for various applications, further enhances their suitability for edge computing and data processing locally on the device.

**Model creation for 2D (Image) data**

**Data Collection and Preprocessing**

* + Methods for collecting and preparing image data for training
  + Image resizing, normalization, and augmentation techniques

**Model Selection and Training**

* + Overview of CNN architectures suitable for ESP32
  + Training the model on a more powerful system
  + Considerations for model size and complexity

**Model Optimization**

* + Techniques for reducing model size (e.g., pruning, quantization)
  + Use of TensorFlow Lite or other lightweight frameworks

**Model Deployment for 2D Data on ESP32**

Converting and Deploying Models

* + Converting the trained model to a format compatible with ESP32
  + Using TensorFlow Lite Micro or ESP-WHO (for image recognition)

**Running Inference on ESP32**

* + Code examples for loading and running the model on ESP32
  + Performance considerations and optimization tips

**Model Creation for 1D Data**

**Data Collection and Preprocessing**

* + Gathering and preprocessing 1D data (e.g., time-series, sensor data)
  + Feature extraction techniques

**Model Selection and Training**

* + Choosing suitable model architectures (e.g., RNN, LSTM)
  + Training models on larger systems

**Model Optimization**

* + Simplifying models for deployment on ESP32
  + Quantization and other optimization techniques

**Model Deployment for 1D Data on ESP32**

**Converting and Deploying Models**

* + Using TensorFlow Lite Micro for 1D models
  + Converting and uploading the model to ESP32

**Running Inference on ESP32**

* + Code examples for deploying and running 1D models
  + Real-time performance considerations

**Case Studies and Examples**

* Practical examples of deploying models on ESP32 for both 2D and 1D data

**Car Motorcycle object detection Model**

import os

import cv2

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report

**# Function to load and preprocess image data**

def load\_data(data\_dir, img\_size=(64, 64)):

categories = ['car', 'bike']

data = []

labels = []

for category in categories:

path = os.path.join(data\_dir, category)

class\_num = categories.index(category) # 0 for car, 1 for bike

for img in os.listdir(path):

try:

img\_array = cv2.imread(os.path.join(path, img), cv2.IMREAD\_COLOR)

resized\_array = cv2.resize(img\_array, img\_size) # Resize to the given size

data.append(resized\_array)

labels.append(class\_num)

except Exception as e:

pass

data = np.array(data).reshape(-1, img\_size[0] \* img\_size[1] \*

**# Flatten the images**

labels = np.array(labels)

return data, labels

**# Data Collection**

data\_dir = "path/to/car\_bike\_dataset" # Replace with your dataset path

x, y = load\_data(data\_dir)

**# Data Preprocessing**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=31)

scale = StandardScaler()

x\_train = scale.fit\_transform(x\_train)

x\_test = scale.transform(x\_test)

**# Training the Moimport os**

import cv2

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report

**# Function to load and preprocess image data**

def load\_data(data\_dir, img\_size=(64, 64)):

categories = ['car', 'bike']

data = []

labels = []

for category in categories:

path = os.path.join(data\_dir, category)

class\_num = categories.index(category) # 0 for car, 1 for bike

for img in os.listdir(path):

try:

img\_array = cv2.imread(os.path.join(path, img), cv2.IMREAD\_COLOR)

resized\_array = cv2.resize(img\_array, img\_size) # Resize to the given size

data.append(resized\_array)

labels.append(class\_num)

except Exception as e:

pass

data = np.array(data).reshape(-1, img\_size[0] \* img\_size[1] \*

**# Flatten the images**

labels = np.array(labels)

return data, labels

**# Data Collection**

data\_dir = "path/to/car\_bike\_dataset" # Replace with your dataset path

x, y = load\_data(data\_dir)

**# Data Preprocessing**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=31)

scale = StandardScaler()

x\_train = scale.fit\_transform(x\_train)

x\_test = scale.transform(x\_test)

**# Training the Model**

svm\_clf = SVC(kernel='linear', C=1.0)

svm\_clf.fit(x\_train, y\_train)

y\_pred = svm\_clf.predict(x\_test)

**# Evaluating Model Performance**

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"\nPrediction Accuracy: {round(accuracy, 2)}\n")

print("Classification Report:\n", classification\_report(y\_test, y\_pred))del

svm\_clf = SVC(kernel='linear', C=1.0)

svm\_clf.fit(x\_train, y\_train)

y\_pred = svm\_clf.predict(x\_test)

**# Evaluating Model Performance**

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"\nPrediction Accuracy: {round(accuracy, 2)}\n")

print("Classification Report:\n", classification\_report(y\_test, y\_pred))