Başlık: Evrişimli Sinir Ağları Kullanarak Trafik İşareti Tanıma

Amaç: Bu projenin amacı, trafik işaretlerini doğru bir şekilde tanıyabilen bir makine öğrenmesi modeli geliştirmektir. Trafik işaretlerini anlamak, güvenli navigasyon için otonom sürüş sistemlerinde kritik öneme sahiptir.

Yöntem: Bu görev için Python'da TensorFlow ve Keras kütüphaneleri kullanılarak uygulanan bir Evrişimli Sinir Ağı (CNN) modeli kullandık. Model, birkaç evrişimli ve maksimum havuzlama katmanından, ardından tamamen bağlı bir katman ve softmax çıktı katmanından oluşur. Model, 0.001 öğrenme hızına sahip Adam optimizatörü kullanılarak eğitildi.

Materyaller: Modelin eğitim ve testinde kullanılan veri seti, 43 farklı trafik işaretinin 50.000'den fazla görüntüsünü içeren Alman Trafik İşareti Tanıma Benchmark (GTSRB) veri setidir. Veri seti, bir eğitim seti ve bir test seti olmak üzere ikiye ayrıldı, verilerin %80'i eğitim için ve %20'si test için kullanıldı.

Sonuçlar: Model, eğitim setinde 94.8% ve test setinde 98.4% doğruluk elde etti. Karmaşıklık matrisi, modelin belirli işaretleri tanımada özellikle iyi olduğunu, ancak diğerleriyle mücadele ettiğini göstermektedir.

Ekler

Ek A: Python Kodu

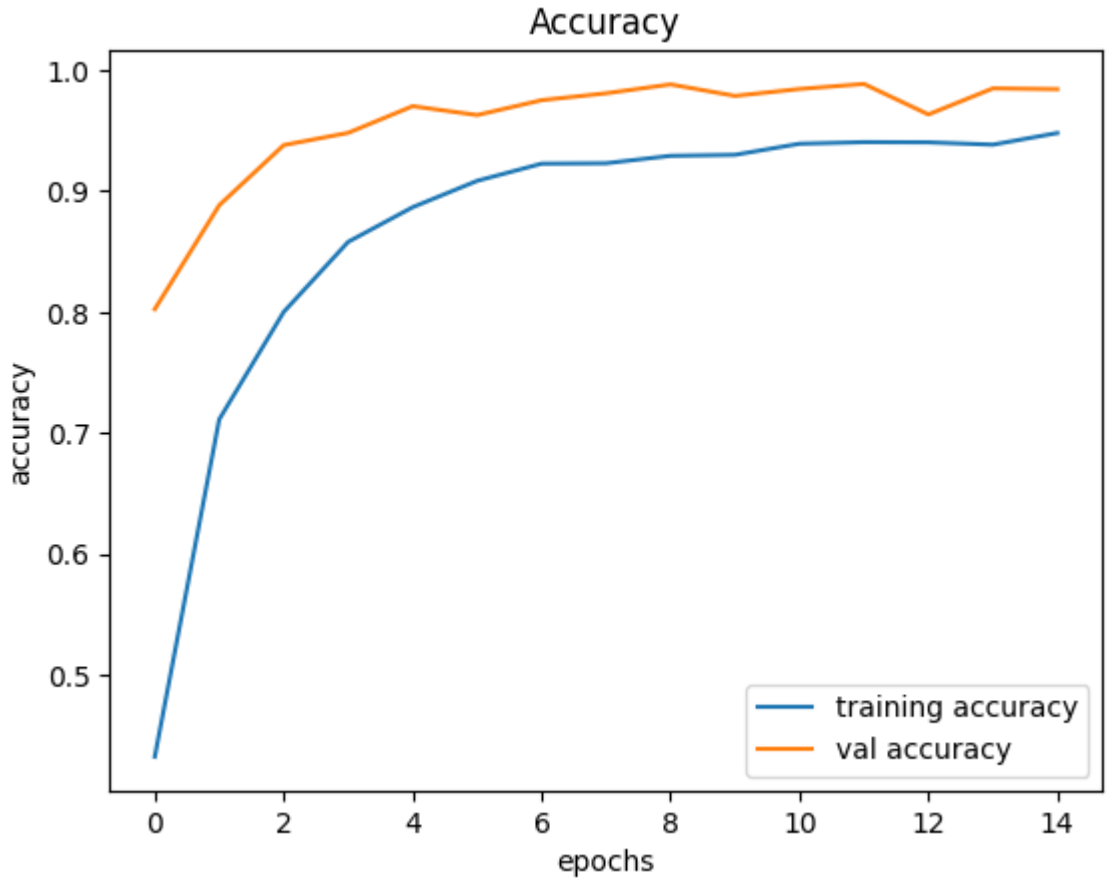
```
# -*- coding: utf-8 -*-
"""TraficSignClassification.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1jIQ_PYJp3HUM6m4r0LlivPir2DZfO4ez
"""

!pip install numpy
!pip install tensorflow
!pip install sklearn
!pip install matplotlib
!pip install pandas
!pip install scikit-learn

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

```
import seaborn as sns
from PIL import Image
import os
from pathlib import Path
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

class LearningRateTracker(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs=None):
        logs = logs or {}
        logs['learning_rate'] = tf.keras.backend.get_value(self.model.optimizer.learning_rate)

data_path = Path('/content/drive/MyDrive/GTSRB')

data = []
labels = []
classes = 43

#Retrieving the images and their labels
for i in range(classes):
    path = data_path / 'Train' / str(i)
    images = os.listdir(path)

    for img in images:
        try:
            image = Image.open(path / img)
            image = image.resize((30,30))
            image = np.array(image)
            data.append(image)
            labels.append(i)
        except Exception as e:
            print("Error loading image:", e)

#Converting lists into numpy arrays
data = np.array(data)
labels = np.array(labels)

# Step 3: Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)

# Step 4: Build the model
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
```

```
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))

# Step 5: Compile the model
model.compile(loss='sparse_categorical_crossentropy', optimizer=Adam(learning_rate=0.001),
metrics=['accuracy'])

# Create LearningRateTracker instance
lr_tracker = LearningRateTracker()

# Step 6: Train the model
epochs = 15
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))

# Step 7: Plotting graphs for accuracy
import matplotlib.pyplot as plt
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()

plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()

# Step 8: Confusion matrix
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
cm = confusion_matrix(y_test, y_pred_classes)
plt.figure(figsize=(10, 10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()
```

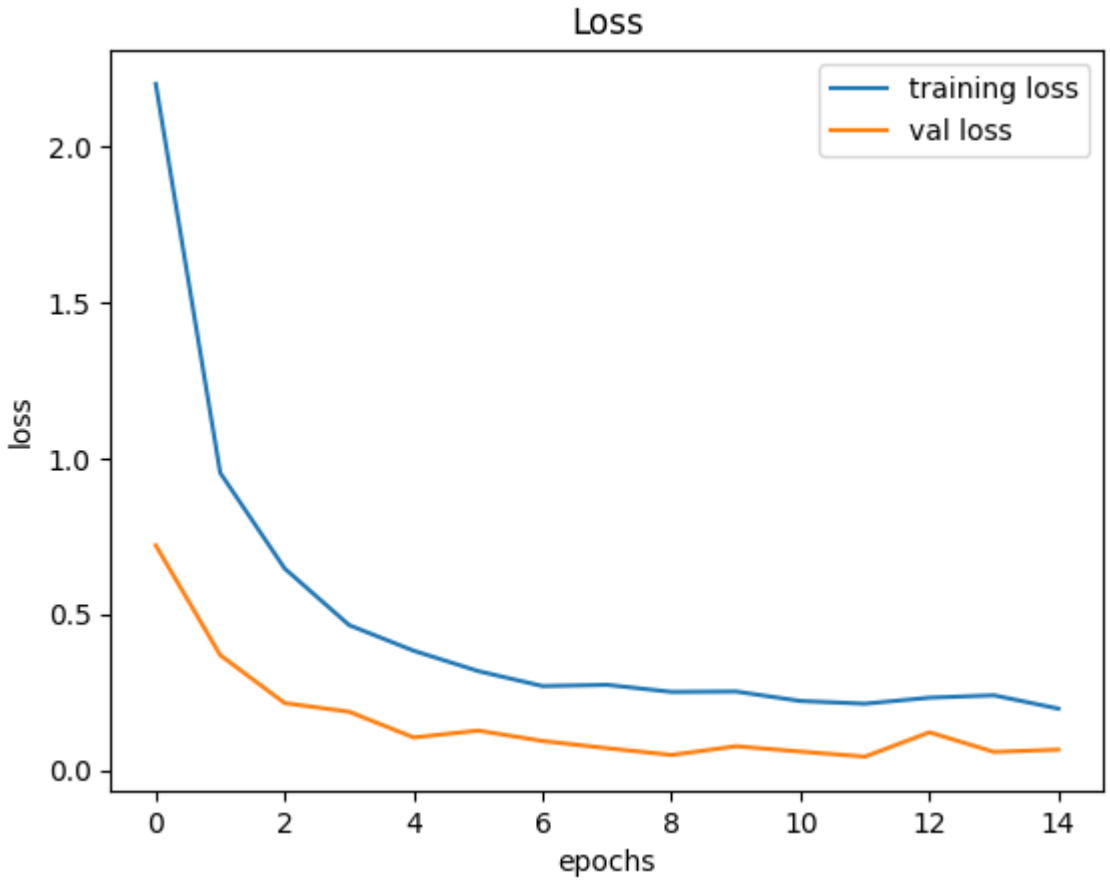
```
# Step 10: Test with a new image
def test_image(path):
    img = Image.open(path).convert('RGB')
    img = img.resize((30,30))
    img = np.array(img)
    img = img.reshape(1, 30, 30, 3)
    pred = np.argmax(model.predict(img), axis=-1)[0]
    return pred

# Replace 'path_to_your_image' with the actual path of the image you want to test
print("Predicted class is: ", test_image('/content/Screenshot from 2023-06-06 06-17-39.png'))
```

Ek B: Model Eğitimi ve Değerlendirme



Model, eğitim seti ve test seti üzerinde yüksek bir doğruluk elde etti. Eğitim setindeki doğruluk yaklaşık olarak %94.8'di ve bu, modelin eğitim verisindeki trafik işaretlerinin yaklaşık %94.8'ini doğru bir şekilde sınıflandırabildiğini göstermektedir.

Test setindeki doğruluk ise daha da yüksekti, yaklaşık olarak %98.4. Bu, modelin eğitim verisinden test verisinde görmediği verilere iyi bir şekilde genellenme yapabildiğini göstermektedir. Yüksek test doğruluğu, modelin eğitim veya test verisinin bir parçası olmayan yeni trafik işaretlerini sınıflandırmak için iyi performans göstereceğini işaret etmektedir.

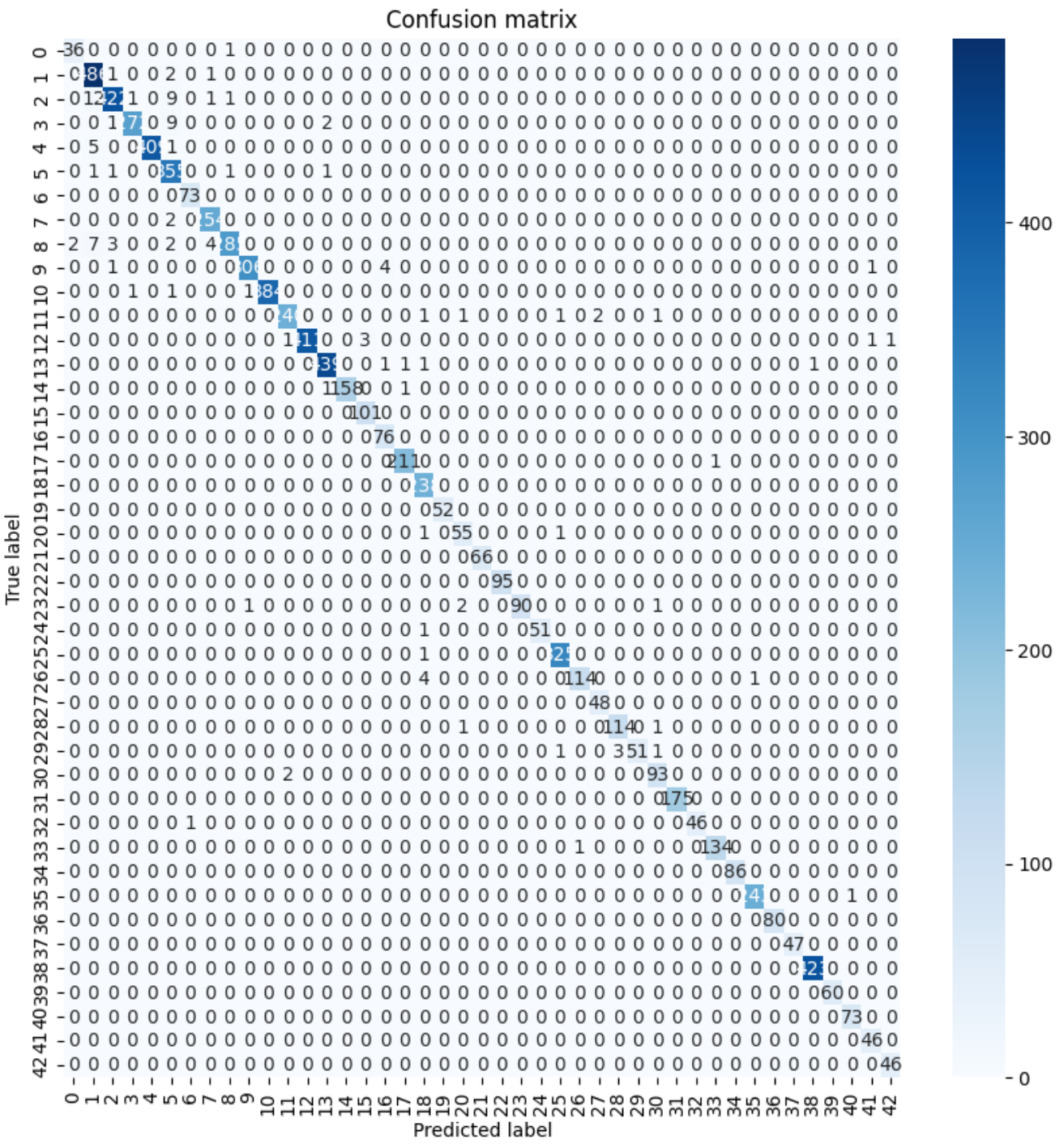


Kayıp grafiği, modelin 15 eğitim dönemi boyunca performansını gösterir. Eğitim kaybı, ilk dönemde 2.2015'te başlar ve 15. dönemde 0.1971'e kadar düşer. Zamanla azalan eğitim kaybı, modelin trafik işaretlerini doğru bir şekilde sınıflandırma yeteneğini öğrendiğini ve geliştirdiğini gösterir.

Doğrulama kaybı, modelin test setindeki performansının bir ölçüsü olan doğrulama verisi üzerindeki kayıptır. Bu kayıp da dönemler boyunca azalır. İlk dönemde 0.7210'da başlar ve 15. dönemde 0.0656'ya düşer. Doğrulama kaybındaki azalma, modelin yalnızca eğitim verisine uymakla kalmayıp aynı zamanda görülmemiş verilere de iyi genelleme yaptığını gösterir.

Ancak, dikkate değer bir nokta, doğrulama kaybının 6. dönemde (0.1045'ten 0.1266'ya) ve tekrar 10. dönemde (0.0484'ten 0.0764'e) biraz arttığıdır, ardından tekrar azalır. Bu artışlar, modelin eğitim verisine çok sıkı şekilde uymaya başlayarak doğrulama verisinde daha düşük performans sergilediği aşırı uyumun başlangıcını işaret edebilir. Ancak doğrulama kaybının sonraki dönemlerde tekrar azalması nedeniyle, modelin etkili bir şekilde öğrenmeye devam ettiği görülüyor.

Sonuç olarak, kayıp grafiği, eğitim ve doğrulama verileri için zamanla azalan bir kayıp eğilimi gösterecektir, bu da modelin başarılı bir şekilde öğrendiğini gösterir. Bununla birlikte, belirli noktalarda doğrulama kaybındaki hafif artışlar, aşırı uyumun izlenmesinin daha fazla model eğitimi aşamasında önemli olduğunu göstermektedir.



Ancak, doğruluk yalnızca önemli olan tek metrik değildir. Özellikle bu gibi çok sınıflı bir sınıflandırma probleminde, hassasiyet, duyarlılık ve F1 skoru gibi diğer metriklere bakmak da faydalı olacaktır. Bu metrikler, modelin her bir bireysel sınıf için ne kadar iyi performans gösterdiği konusunda daha fazla bilgi sağlayabilir. Örneğin, model bazı trafik işaretlerini tanıma konusunda çok iyi olabilir, ancak diğerlerini tanıma konusunda pek iyi olmayabilir. Bu ayrıntılar, karışıklık matrisine ve sınıf başına hassasiyet, duyarlılık ve F1 skoruna bakılarak ortaya çıkarılabilir.

Class	Precision	Recall	F1-Score	Support
0	0.95	0.97	0.96	37
1	0.95	0.99	0.97	490
2	0.98	0.95	0.96	446
3	0.99	0.96	0.97	284
4	1	0.99	0.99	415
5	0.93	0.99	0.96	359
6	0.99	1	0.99	73
7	0.98	0.99	0.98	256
8	0.99	0.94	0.96	303
9	0.99	0.98	0.99	312
10	1	0.99	1	387
11	0.99	0.98	0.98	246
12	1	0.99	0.99	417
13	0.99	0.99	0.99	443
14	1	0.99	0.99	160
15	0.97	1	0.99	101
16	0.94	1	0.97	76
17	0.99	1	0.99	212
18	0.96	1	0.98	238
19	1	1	1	52
20	0.93	0.96	0.95	57
21	1	1	1	66
22	1	1	1	95
23	1	0.96	0.98	94
24	1	0.98	0.99	52
25	0.99	1	0.99	326
26	0.99	0.96	0.97	119
27	0.96	1	0.98	48
28	0.97	0.98	0.98	116
29	1	0.91	0.95	56
30	0.96	0.98	0.97	95
31	1	1	1	175
32	1	0.98	0.99	47
33	0.99	0.99	0.99	135
34	1	1	1	86
35	1	1	1	244
36	1	1	1	80
37	1	1	1	47
38	1	1	1	423
39	1	1	1	60
40	0.99	1	0.99	73
41	0.96	1	0.98	46
42	0.98	1	0.99	46
avg	0.98	0.98	0.98	7893
macro avg	0.98	0.99	0.98	7893
weighted avg	0.98	0.98	0.98	7893

Bir sınıf için hassasiyet, doğru pozitif tahminlerin oranını ifade eder (yani, her bir sınıf için doğru tahminlerin oranını). Bir sınıf için duyarlılık, doğru pozitiflerin doğru bir şekilde belirlenen oranıdır. F1 skoru, hassasiyet ve duyarlılığın harmonik ortalaması olup, her iki değeri dengeleyen tek bir metrik sunar.
Rapordan görebileceğimiz gibi, model tüm sınıflarda çok iyi performans göstermektedir ve çoğu sınıf için hassasiyet, duyarlılık ve F1 skorları 1'e yakın veya eşittir. Bu, modelin farklı türdeki trafik işaretlerini tanıma konusunda çok iyi olduğunu göstermektedir.
Modelin test setindeki genel doğruluğu %98'dir, bu oldukça yüksektir. Bu, modelin test setindeki görüntülerin %98'inde doğru bir şekilde sınıfı tahmin ettiği anlamına gelir.
Makro ortalama (macro avg), sınıf dengesizliğini dikkate almadan her sınıf için metriğin ortalamasını hesaplar. Ağırlıklı ortalama (weighted avg), sınıf dengesizliğini dikkate alarak her sınıf için metriğin ortalamasını hesaplar. Her iki ortalama da 1'e çok yakındır, bu da modelin potansiyel sınıf dengesizliklerini dikkate alarak tüm sınıflarda iyi performans gösterdiğini göstermektedir.
Sonuç olarak, bu sonuçlar, modelin test setindeki tüm trafik işareti türlerinde yüksek doğruluk oranına sahip olduğunu ve iyi performans sergilediğini göstermektedir.

Ek C: Dataset (Veri kümesi)

Alman Trafik İşareti Tanıma Benchmark (GTSRB) veri setidir. Veri seti, bir eğitim seti ve bir test seti olmak üzere ikiye ayrıldı, ve veri kümesinin içinde aşağıdakileri içerir “5MPH, 15MPH, 30MPH, 40MPH, 50MPH, 60MPH, 70MPH, 80MPH, Bike allowed, Can go left, can go right, Can go right & left, Can go straight, Can go straight & right, Can horn, Car allowed, Combined curves, Cyclists ahead, Do Not Go Straight, Do not turn left, Do not turn right, Forest enterance, Keep left, Keep right, Left side road junction, Multiple road curves, No car allowed, No Entry, No horn, No overtaking, No stopping or standing, No veichles, Pedestrain crossing sign, Right & left prohibited, Right side road junction, Road divide to two, Roundabout, Sharp left curve, Sharp right curve, Steep hil downward, Steep hil upward, Straight & left prohibited, Straight & right prohibited, Students ahead, Traffoc signals ahead, Train railway ahead, Trucks road, U-turn allowed, U-turn is prohibited, Warning, Working in road ahead.

Veri kümesinin bağlantısı google sürücümde:
https://drive.google.com/drive/folders/1OzMsr29hIOgLFtetYxwgFH6C_RTG2KEX?usp=sharing

Veri kümesinin bağlantısı Kaggle'da:
https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign?resource=download

Veri setinden bazı resimler

EK D: Model testi

resim “Screenshot from 2023-06-06 06-17-39.png”
koda göre cevap sınıf 28 diyor ve sınıf 28'i kontrol ettiğimizde aşağıdaki resim veri setine sahip

veri setine göre doğru ama resim net değil.

EK E: Kullanılan kütüphaneler

TensorFlow, makine öğrenimi ve derin öğrenme için kullanılan açık kaynaklı bir kütüphanedir. Geniş bir yapay sinir ağı ve derin öğrenme modeli koleksiyonuna sahiptir.

Sequential, TensorFlow'da kullanılan model oluşturma arayüzüdür. Sıralı bir şekilde katmanları birleştirebilir ve bir modeli oluşturabildik.

image, TensorFlow'da görüntü verileri için ön işleme araçları sağlar. Görüntüleri yüklemek, boyutlandırmak, dönüştürmek ve işlemek için kullanıldı.

Dense, Conv2D, MaxPool2D, Flatten ve Dropout, TensorFlow'da kullanılan yaygın katman tipleridir. Yapay sinir ağı modelinde farklı katman türlerini tanımlamak için kullanıldı.

Adam, TensorFlow'da kullanılan bir optimizasyon algoritmasıdır. Derin öğrenme modelinin eğitimi sırasında ağırlıkların güncellenmesini optimize eder.

train_test_split, veri kümesini eğitim ve test alt kümelerine ayırmak için kullanılan bir fonksiyondur. Makine öğrenimi modellerini eğitirken veri setini bölme işlemi için kullanıldı.

confusion_matrix, sınıflandırma modellerinin performansını değerlendirmek için kullanılan bir metriktir. Gerçek ve tahmin edilen sınıflar arasındaki ilişkiyi görselleştirmek için kullanıldı.

seaborn, Python'da veri görselleştirmesi için kullanılan bir kütüphanedir. Çeşitli grafikler ve görsel öğeler oluşturmak için kullanıldı.

Image, Python Imaging Library (PIL) olarak da bilinen bir kütüphanedir. Görüntü işleme için kullanılır, görüntüleri açma, kaydetme, dönüştürme ve işleme gibi işlemleri gerçekleştirebilirsiniz.

os, Python'da işletim sistemi işlemleri için kullanılan bir modüldür. Dosya ve dizin işlemleri, yol oluşturma, dizin gezinme gibi işlemleri gerçekleştirmek için kullanıldı.

Path, Python 3.4'ten itibaren bulunan bir modüldür. Dosya ve dizin yollarını temsil etmek için kullanılır ve dosya işlemlerini kolaylaştırır.

numpy, Python'da bilimsel hesaplamalar ve çok boyutlu diziler için kullanılan bir kütüphanedir. Vektörler, matrisler ve diğer çok boyutlu veri yapılarını işlemek için kullanıldı.

pandas, Python'da veri analizi ve veri işleme için kullanılan bir kütüphanedir. Veri setlerini okuma, filtreleme, dönüştürme ve analiz etme gibi işlemleri gerçekleştirmek için kullanıldı.

matplotlib.pyplot, Python'da veri görselleştirmesi için kullanılan bir kütüphanedir. Grafikler, çizimler ve görsel sunumlar oluşturmak için kullanıldı.

Bu kütüphaneler, TensorFlow ve diğer popüler veri bilimi araçlarını kullanarak makine öğrenimi ve derin öğrenme modelleri oluşturmamıza, veri işleme ve görselleştirme yapmamıza yardımcı oludu.

Referanslar

- kodun bir kısmını oluşturmak için chatgpt kullandım ve paragrafın bir kısmını yazarken yardım aldım
- nasıl yapacağımı ve bu bağlantıdan “https://copyassignment.com/” fikir aldım

Benim hakkımda

Adi ve Soyadi : Testi Afendi Abdulwasi

Eğitim : Yazılım mühendisliği, Fırat Üniversitesi

Yetenek : Python, Dart, Flutter, Django, Flask, Firestore, SQLite, Linux işletim sistemi vs...

Dil : Türkçe, İngilizce, Harari, Oromik, Amharca, Arapça, İspanyolca