KOCAELİ ÜNİVERSİTESİ
Trafik İşareti Tespiti

Ramazan KAPLANER

Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi

181307024@kocaeli.edu.tr

Özet

Bu çalışmada, otonom araçları yollarımıza getirmek için sağlam ve güvenilir trafik işareti tespiti gereklidir. Görüntü işleme tekniklerini ve yapay zekanın temellerini kullanarak, trafik işaretlerini otomatik olarak tanımak için derin öğrenmeye dayalı modeller oluşturup bu modelleri başarıya ulaştırmak bu projenin amacıdır.

1.Giriş

Araç teknolojisinin gelişmesiyle birlikte otomotiv sektörleri hızlı bir şekilde büyüdü. Bu büyümenin sonucunda araç sayısı artmış ve trafik karmaşık hale gelmiştir. Trafikteki sürücüleri bilgilendirmek ve trafiğin sağlıklı bir şekilde ilerlemesini sağlamak için trafik işaretleri kullanılır. Trafik işaretleri, trafikte bulunan insanların can güvenliklerini korumak için yardımcı olmaktadırlar. Trafik levhalar zamanla eskiye bilir veya olumsuz hava koşulları sebebi ile insanlar tarafından görülemeyebilir. Bu sebeple kazalara neden olabilir.

Kazaların önüne geçmek için otonom araç teknolojisinde trafik işareti tespiti çok önemli bir rol almaktadır.

2. Veri Setinin Toplanması

Trafik işaretlerinin makine öğrenmesi modelleriyle tanınmasını amaçlayan çalışmalarda kullanılmak üzere internet ortamında açık erişime sahip birçok veri seti bulunmaktadır. Bu projede Veri setini kendimiz oluşturmak için Google images üzerinden BeautifulSoup ve Selenium kütüphaneleri ile web scraping yaparak verileri kategoriler halinde indirme işlemi yapıldı.

Resim 1. Trafik işaretleri

2.1 Veri Seti

Projeye başlamadan önce uygun veri setinin belirlenmesi önemli bir rol almaktadır. Aynı şekilde veri setini toplamak için kullanılacak araçlar da önceden belirlenmesi önemlidir. Saha araştırmasından ve kaynak fazlalığından dolayı projede selenium ve beautifulsoup kütüphaneleri kullanmayı uygun buldum.

3. Yazılım Mimarisi

Makine öğrenmesi modellerini oluşturmak için birçok yazılım dili vardır ve bunların en popüleri çok fazla makine öğrenimi kütüphanesi barındırması ve açık kaynak kodlu olması sebebiyle Python'dur.

3.1 PyCharm

PyCharm, çapraz platform bir Python geliştirme ortamı'dir. Kod analizleri, grafiksel hata ayıklamacısı, versiyon kontrol sistemi ile entegre ve Django ile Python web geliştirmeleri yapılmasını sağlamaktadır.

PhpCharm kullanmamın sebebi içinde barındırdığı araçlar ve kolay kod analizleri ile birlikte hızlı ve ayakları yere basar bir geliştirme yapmamıza olanak sağlaması.

3.2 Web scraping

Projede web scraping yapmak için selenium ve beautifulsoup kütüphaneleri kullanılmıştır. Beautiful Soup, HTML ve XML belgelerini ayrıştırmak için Selenium ise Tarayıcıya özgü bir sürücü aracılığıyla iletişim kurar ve onu kontrol etmemi sağlıyor.

3.2.1 BeautifulSoup

Beautiful Soup, HTML ve XML belgelerini ayrıştırmak için bir Python paketidir. HTML'den veri ayıklamak için kullanılabilen ayrıştırılmış sayfalar için bir ayrıştırma ağacı oluşturur ve bu, web kazıma için yararlıdır.

3.2.2 Selenium

Selenium, tarayıcı otomasyonunu desteklemeyi amaçlayan bir dizi araç ve kitaplık için açık kaynaklı bir şemsiye projedir. Test komut dosyası dili öğrenmeye gerek kalmadan işlevsel testler yazmak için bir oynatma aracı sağlar.

3.2.3 OpenCV

OpenCV gerçek-zamanlı bilgisayar görüsü uygulamalarında kullanılan açık kaynaklı kütüphane. İlk olarak Intel tarafından geliştirilmiş, daha sonra Willow Garage ve sonra Itseez tarafından sürdürüldü. Bu kütüphane çoklu platform ve BSD lisansı altında açık kaynaklı bir yazılımdır.

3.2.3 Roboflow

Roboflow bilgisayarlı görü uygulamaları için görüntü etiketlemeye yarayan çevrimiçi bir platformdur. Projede verileri etiketlemek için roboflow kullanılmıştır.

4. Model Açıklamaları

4.1 Yapay Sinir Ağları

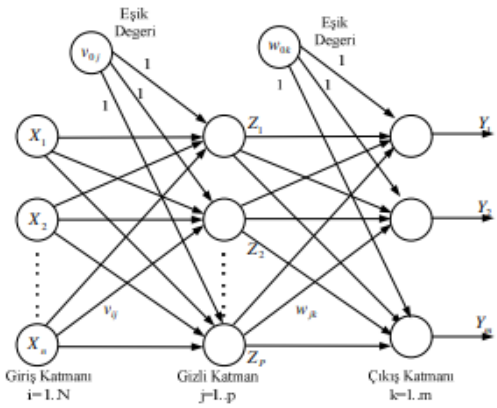
Yapay sinir ağları, biyolojik sinir ağlarından esinlenilerek ortaya çıkarılan ve biyolojik sinir ağlarına benzer bazı performans özellikleri içeren bir bilgi işleme sistemidir (Fausett,1994:3). Basit bir şekilde insan beyninin çalışma şeklini taklit eden YSA'lar veriden öğrenebilme, genelleme yapabilme, sınırsız sayıda değişkenle çalışabilme vb. birçok önemli özelliğe sahiptir.

Yapay Sinir Ağlarında Öğrenme Süreci**;**

Genel anlamda, bir sinir ağı birden fazla nörondan oluşur. Burada her nöron, bazı girdilere dayalı bir çıktıya ulaşmak için bir aktivasyon fonksiyonu ile doğrusal bir fonksiyonu hesaplar. Hesaplamada kullanılan aktivasyon fonksiyonu doğrusallığı bozmak için tasarlanmıştır. Hesaplanan çıktı, önem seviyesini temsil eden bir ağırlığa bağlıdır ve sonraki katmanda hesaplamalar için kullanılır. Bu hesaplamalar, nihai bir çıktıya ulaşılana kadar ağın tüm mimarisi boyunca gerçekleştirilir. Hesaplanan çıktılar, daha sonra hesaplama sürecini yeniden başlatmak ve ağın parametrelerini güncellemek için geri yayılımla kullanılırlar.

4.1.1 Çok katmanlı ileri beslemeli yapay sinir ağları (MLP)

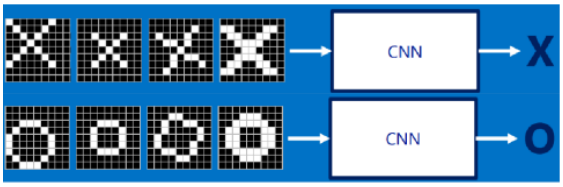
MLP ağlarında nöronlar katmanlar şeklinde organize edilmiştir. MLP'de ilk katman girdi katmanıdır. Girdi katmanı, çözülmesi istenilen probleme ilişkin bilgilerin YSA'ya alınmasını sağlar. Diğer katman ise ağ içerisinde işlenen bilginin dışarıya iletildiği çıktı katmanıdır. Girdi ve çıktı katmanlarının arasında yer alan katmana ise gizli katman adı verilir. MLP ağlarında birden fazla gizli katman da bulunabilir. Şekil 2, tipik bir MLP ağının yapısını göstermektedir.



Şekil1: Çok katmanlı ileri beslemeli yapay sinir ağı

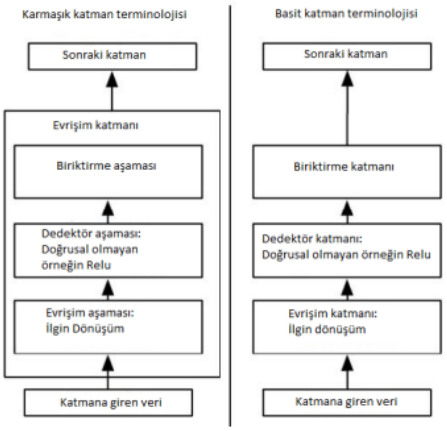
4.2 Derin Öğrenme

Evrişimli sinir ağları, sıradan sinir ağlarına çok benzer, eğitimleri sırasında bilgi öğrenebilen ağırlıkları ve bias değerleri olan nöronlardan oluşurlar. Her nöron bazı girdiler alır ve matematiksel işlemler gerçekleştirir. Tüm ağ, tek bir farklılaştırılabilir puanlama fonksiyonunu ifade eder: bir uçtaki (giriş) görüntü piksellerinin içeriğinden diğer uçtaki (çıktı) karşılık gelen sınıfı veya sonucu tanımlayan puanlayan bir ağ modelidir.



Şekil4.2: CNN girişi ve çıkışı

Bir CNN, bu katmanların her birinin fonksiyonlar yoluyla bir aktivasyon hacmini yenisine dönüştürdüğü bir katman dizisine sahip olmasıyla karakterize edilir ve bu katmanlardan birkaçı söz konusu katmanlar içindeki ve arasındaki yapılandırma parametrelerini değiştirerek kullanıldığında derin öğrenme gerçekleşir.



Şekil 4.3: Basit katmanlar ve terminolojisi

4.2.1 Evrişim Katmanı

Evrişimli süreç için her filtre boyut olarak küçüktür (genişlik ve yükseklik olarak), hatta giriş boyutunun (görüntü) toplam derinliği boyunca uzanır. Örneğin, bir ConvNet'in ilk katmanındaki tipik bir filtre 5x5x3 boyutunda olabilir (yani, 5 piksel genişliğinde ve yüksekliğinde ve renk kanalları nedeniyle 3 katman derinliğinde - RGB). İleri yayılım sırasında, her bir filtre, herhangi bir pozisyondaki filtre girişleri ile giriş arasındaki noktaları hesaplamak için giriş hacminin genişliği ve yüksekliği (eşit derinlik) boyunca kaydırılır.

4.2.2 Aktivasyon Katmanı

Bir sinir ağındaki tüm katmanların doğrusal olmaması nedeniyle, sinir ağındaki nöronların her birinin değerlerini hesapladıktan sonra, bu değerler bir aktivasyon fonksiyonundan geçirilir. Yapay bir sinir ağı temelde matrislerin çarpılması ve eklenmesinden oluşur. Sadece bu doğrusal hesaplamaları kullanıyor olsaydık, onları birbirinin üzerine istifleyebilirdik ve bu çok derin bir ağ olmazdı. Bu nedenle, doğrusal olmayan aktivasyon fonksiyonları genellikle ağın her katmanında kullanılır.

4.2.3 Havuzlama Katmanı

Havuzlama katmanı (çözünürlük azaltma katmanı olarak da bilinir), alıcı alanında daha az değer üreten bir dizi değeri birleştirir veya gruplandırır. Alıcı alanınızın boyutuna (örn. 2 x 2) ve Şekil 4.14'te gösterildiği gibi gruplama işlemine göre yapılandırılabilir. Tipik olarak havuzlama işlemi (max-pooling veya averagepooling), üst üste binmeyen belirli adımlarla kaydırılan bloklarda meydana gelir

4.2.3 Tam Bağlı (Fully-Connected) Katman

Veriler evrişim katmanlarından geçtikten sonra, sonunda yeterince küçük bir özellik haritası oluşur ve içerik tek boyutlu bir vektöre sıkıştırılır. Bu noktaya kadar, veriler sınıf tanımada belirli bir kesinlik derecesi için zaten yeterlidir. Bununla birlikte, karmaşıklık ve hassasiyet açısından daha iyi bir sonuç elde etmek için, modelin sonunda, çıkış nöronunun tüm nöronlara bağlandığı çok katmanlı sinir ağlarına benzer şekilde, tamamen bağlı bir nöral katman kullanılacaktır. Bağlantıların girdisi ve ağırlığı, geri yayılım yöntemi kullanılarak güncellenir.

4.2.4 Adam Optimizasyonu

Adam optimize edici (Adaptive Moment Estimation) derin sinir ağı eğitimi için özel olarak tasarlanmış uyarlanabilir bir öğrenme hızı optimizasyon algoritmasıdır

4.2.5 Softmax Fonksiyonu

Bu fonksiyon bir sinir ağının son katmanında kullanılan bir aktivasyon fonksiyonudur. Bu fonksiyon tam bağlı bir sinir ağının çıktısının olasılık dağılımını vermez ancak bu fonksiyonu kullanmak sinir ağının çıkışının olasılık dağılımını hesaplamaya imkan tanır. Bu fonksiyon, her bir nöronun değerinin

[0,1] arasına çekilmesini sağlayarak, nöronların sonuçlarında bir normalizasyon gerçekleştirir ve girdi görüntüsünün hangi sınıfa ait olma olasılığının belirlenmesine izin verir

5. Kod Açıklamaları

5.1 Web scraping ile verilerin indirilmesi

Gerekli kütüphaneler projeye ekliyoruz

```
import bs4
import requests
from selenium import webdriver
import os
import time
```

Görüntüleri kaydetmek ve indirmek için with open komutunu kullanıyoruz.

```
def download_image(url, folder_name, num):
    # write image to file
    reponse = requests.get(url)
    if reponse.status_code == 200:
        with open(os.path.join(folder_name, str(num) + ".jpg"), 'wb') as
            file.write(reponse.content)
```

Chrome driverin yolunu verip çalıştırıyoruz.

```
chromePath = r'C:\Users\ramaz\Desktop\driver\chromedriver
driver = webdriver.Chrome(chromePath)

#one_way
search_URL="https://www.google.com/search?q=nature+traffi
#no_parking
```

Driverın get komutu ile verileri indirileceği adrese gitmesini sağlıyoruz

```
driver.get(search_URL)
```

Google images indirme yapacağımız için scroll önemli bir hale geliyor. Burada scrooll u aşağı doğru hareket ettirerekkodumuzun verilerin hepsini taramasını sağlıyoruz.

```
SCROLL_PAUSE_TIME = 1

for i in range(6):
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(SCROLL_PAUSE_TIME)
    new_height = driver.execute_script("return document.body.scrollHeight")
```

İmport ettiğimi bs4 kütüphanesinden BeautifulSoup özelliğini kullanarak sayfaları ayrıştırıyoruz.

```
page_html = driver.page_source
pageSoup = bs4.BeautifulSoup(page_html, 'html.parser')
containers = pageSoup.findAll('div', {'class': "isv-r PNCib MSM1fd BUooTd"})
```

Ayrıştırdığımız sayfaların xpath lerini alıp for döngüsüne sokuyoruz. Burada reklamlar gibi bozuk dataların gelmemesi için önceden ön hazırlık yapıp reklamların xpath lerini öğrenip almaması için if methodu kullanıyoruz.

```
for i in range(1, len_containers + 1):
    if i == 25 or i==50 or i==75: #or i==170 or i==315 or i==50 or i==75 or i==172 or i
        # i == 25 or i==170 or i==315 or i==50 or i==75 or i==172 or i==197 or i==222 o

        continue


    xPath ="""//*[@id="islrg"]/div[1]/div[{}]""".format(i)
    previewImageXPath = """//*[@id="islrg"]/div[1]/div[{}]/a[1]/div[1]/img""".format(i)

    previewImageElement = driver.find_element(by=By.XPATH, value=previewImageXPath)
    previewImageURL = previewImageElement.get_attribute("src")
    #print("preview URL", previewImageURL)
    # print(xPath)
    driver.find_element(by=By.XPATH, value=xPath).click()
    # time.sleep(3)
    timeStarted =time.time()
```

Devamında xpathleri alınmış divlerin içinden src etiketli linkleri alıyoruz. Bu linkleri önceden yazdığımız indirme fonksiyonuna yollayarak resimlerin indirmelerini tamamlamış oluyoruz.

```
while True:
    imageElement = driver.find_element(by=By.XPATH,value="""//*[@id="Sva75c"]/div/div/
    imageURL = imageElement.get_attribute('src')
    if imageURL != previewImageURL:
        break
    else:
        currentTime = time.time()
        if currentTime - timeStarted > 10:
            print("Zaman aşımı yeni dosyaya geç")
            break
try:
    download_image(imageURL, folder_name, i)
    print("İndirilen: {} Toplam: {}  URL: {}".format(i, len_containers + 1, imageURL))
except:
    print("Resim İndirilemedi:{} sonrakini indirmeye devam ediyor".format(i))
```

5.2 Veri İşlemleri

İlk olarak veri işlemlerini yapacağımız modülü oluşturuyoruz. Oluşturacağımız modülde iki adet fonksiyon bulunacaktır. Birincisi modeli eğiteceğimiz veriyi hazır hale getirmemizi sağlayacak “set_train_data” fonksiyonu, ikincisi ise modelimizi test edeceğimiz veriyi hazır hale getirmemizi sağlayacak “set_test_data” fonksiyonu.

Gerekli modüllerin yüklenmesi

```
import os
import cv2
import numpy as np
import pandas as pd
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

Eğitim verilerinin oluşacağı fonksiyon

```
def set_train_data(train_data_dir, num_classes,
                   resize_col=32, resize_row=32,
                   test_size=0.2, random_state=1):
```

Fonksiyon parametreleri;

- **num_classes:** Veri setindeki sınıf(label) sayısı
- **train_data_dir:** Eğitim verilerin bulunduğu konum
- **resize_row, resize_col:** Tüm resimlerin aynı boyutta olmasını sağlamak amacıyla hepsi yeniden boyutlandırılacaktır.
- **test_size**: Validasyon verisi bölme oranı.
- **random_state**: Fonksiyon bölme işlemini gerçekleştirirken verileri veri seti içerisinden rastgele seçecektir.

Resim matrislerinin ve etiketlerin tutulması için boş bir dizi açtım.

```
data = []
labels = []
train_files = os.listdir(train_data_dir)
```

Her bir dosya yolunu dolaşacak döngümüzde dosya yolunun adı aynı zamanda sınıf numaramız olduğundan dosyadan tüm resimlerin yolunu alıyoruz.

```
train_files = os.listdir(train_data_dir)
for classes in train_files:

    classname = str(classes)
    path = os.path.join(train_data_dir, classname)
    images = os.listdir(path)
```

Tüm resimleri dolaşacak döngümüzde resim okunarak bir diziye aktarılıyor ve yeniden boyutlandırılıyor daha sonra ilgili dizilere ekleniyor.

```
for image in images:

    image_path = os.path.join(path, image)
    image_array = cv2.imread(image_path)
    image_array = cv2.resize(image_array, (resize_row, resize_col))
    data.append(image_array)
    labels.append(classname)
```

Dizilerin numpy dizilerine dönüştürüyoruz

```
data = np.array(data)
labels = np.array(labels)
```

Resim verilerinin train ve validation verisi olarak bölünmesi işlemi

```
x_train, x_val, y_train, y_val =
    train_test_split(
    data, labels, test_size=test_size,
    random_state=random_state)
```

Sınıfların kategorik matris hale getirilmesi

```
y_train = to_categorical(y_train, num_classes)
y_val = to_categorical(y_val, num_classes)
```

Test verilerinin oluşacağı fonksiyon

```
def set_train_data(train_data_dir, num_classes,
                   resize_col=32, resize_row=32,
                   test_size=0.2, random_state=1):
```

Fonksiyon parametreleri şunlardır;

- **test_data_dir:** Test verilerinin bulunduğu konum
- **train_info_dir:** Test verileriyle ilgili bilgilerin bulunduğu csv dosyasının konumu

Csv dosyasının okunması

```
y_test = pd.read_csv(test_info_dir)
```

Resimlerin dosya yollarının ve sınıf bilgilerinin csv den okunması

```
images = y_test['filename'].values
```

```
y_test = y_test["class"].values
```

Resim matrislerinin tutulacağı dizi

```
x_test = []
```

Test klasöründe bulunan tüm resimleri dolaşacak döngü

```
for image in images:

    image_path = os.path.join(dataset_dir, image)

    image_array = cv2.imread(image_path)
```

Verileri okuduktan sonra boyutları

```
(5592, 32, 32, 3)
(5592, 13)
(1399, 32, 32, 3)
(1399, 13)
(1621, 32, 32, 3)
(1621, 13)
```

4.3 Model İşlemleri

4.3.1 CNN Modeli

Gerekli modülleri yüklenmesi

```
from tensorflow.keras.layers import Dense, Dropout, Flat
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization,
from tensorflow.keras.models import Sequential
import os
```

Modelin oluşacağı fonksiyon

```
def build_model(input_shape, num_classes,
                kernel_size=(3, 3),
                pool_size=(2, 2),
                dropout_rate=0.3):
```

Fonksiyon parametreleri;

- **input_shape:** Resim matrislerinin boyutu
- **num_classes:** Sınıf sayısı
- **kernel_size:** Resme uygulanacak filtre boyutu. Varsayılan (3, 3)
- **pool_size:** Katmana uygulanacak altörnekleme boyutu. Varsayılan (2, 2)
- **dropout_rate:** Modelden çıkarılacak nöronların oranı. Varsayılan (0,3)

```
model = Sequential()

model.add(Conv2D(32, kernel_size=kernel_size, padding='same',
                 input_shape=input_shape))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=pool_size))
model.add(Dropout(dropout_rate))
```

```
model.add(Conv2D(64, kernel_size=kernel_size, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=pool_size))
model.add(Dropout(dropout_rate))
```

```
model.add(Conv2D(128, kernel_size=kernel_size, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=pool_size))
model.add(Dropout(dropout_rate))

model.add(Flatten())

model.add(Dense(128))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(dropout_rate))

model.add(Dense(num_classes, activation="softmax"))

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

return model
```

Modeli oluşturup derledikten sonra detaylarını görmek için

```
model = build_model((32,32,3),13)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 batch_normalization (BatchN  (None, 32, 32, 32)       128
 ormalization)

 activation (Activation)     (None, 32, 32, 32)        0

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )
```

Model eğitim fonksiyonu

```
def train_model(model,
                x_train, y_train,
                x_val, y_val,
                batch_size=16,
                epochs=10):
```

Fonksiyon parametreleri şu şekilde;

- **model**: Eğitilecek model

- **x_train, y_train**: Eğitim verileri
- **x_val, y_val**: Validasyon verileri
- **batch_size**: Her bir iterasyonda alınacak veri sayısı
- **epochs:** Eğitim iterasyon sayısı

Fonksiyonun dönüş değeri olarak modeli ve model verilerini içeren "history" verisi gönderilmektedir.

```
history = model.fit(x_train, y_train,
                    batch_size=batch_size, epochs=epochs,
                    validation_data=(x_val, y_val))
return model, history
```

Veri Görselleştirme fonksiyonu

```
def visualize_history(hist):
```

Fonksiyon parametre olarak sadece model historysini almaktadır.

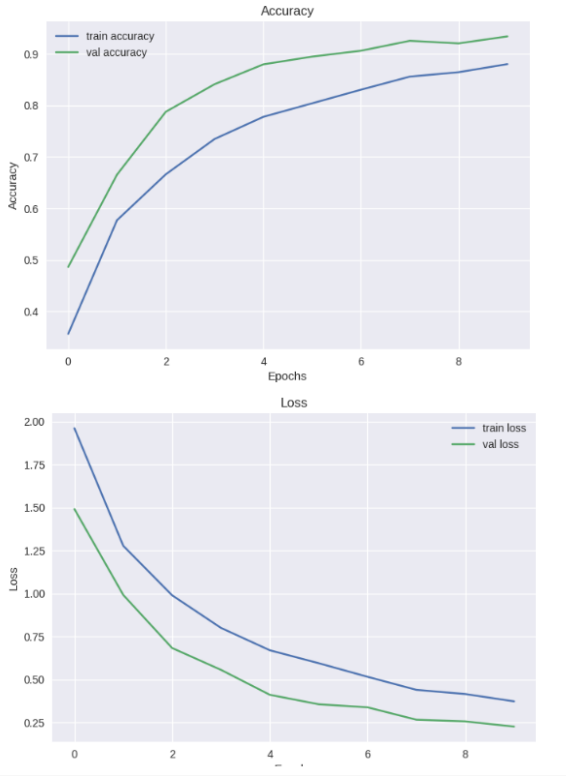
```
histo = hist.history

plt.style.use('default')
plt.style.use('seaborn')

plt.figure(0)
plt.plot(histo['accuracy'], label='train accuracy')
plt.plot(histo['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.savefig('train_accuracy')

plt.figure(1)
plt.plot(histo['loss'], label='train loss')
plt.plot(histo['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.savefig('train_loss')
```

Çalıştırma Aşaması

Modelin kurulması ve eğitiminin yapılması

```
input_shape = x_train.shape[1:]
model = build_model
(input_shape, num_classes)
model, history = train_model
(model, x_train, y_train, x_val, y_val)
```

Epoch/Doğruluk Oranı Grafiği



Epoch/Loss oranı grafiği

4.3.2 MLP Modeli

```
class MLPMixerLayer(layers.Layer):
    def __init__(self, num_patches, hidden_units,
        super(MLPMixerLayer, self).__init__(*args,

        self.mlp1 = keras.Sequential(
            [
                layers.Dense(units=num_patches),
                tfa.layers.GELU(),
                layers.Dense(units=num_patches),
                layers.Dropout(rate=dropout_rate),
            ]
```

6. Karşılaşılan Sorunlar

Projeye başlarken önce Colab üzerinden geliştirmeye başlamıştım ama selenium webdriver modülünü colab a eklerken path ve izin hataları ile karşılaştım. Bende projemi pycharm taşıdım.

Verileri otomatik çekerken araya giren reklamlar veri çekme işleminin yarıda kesilmesine sebep oluyordu. Bunun önüne geçmek için reklamların xpath lerini alıp onları koda dahil etmedim.

OpenCV ile verileri ön isleme yaparken cv2.error: OpenCV(4.5.5) hatası ile karşılaştım.

7. Sonuçlar

Veriler web scraping yöntemi ile Google görsellerden indirildi. Veriler kategoriler halinde ayrıştırılıp işlenmeye hazır hale getirildi.

CNN Modelinin Eğitim sonunda %93 civarında bir doğruluk oranı elde edilmiştir.

8.Kaynaklar

[1] https://medium.com/analytics-vidhya/create-your-own-real-image-dataset-with-python-deep-learning-b2576b63da1e

[2] https://www.ibm.com/docs/sl/scdli/1.2.0?topic=dataset-images-object-classification

[3] https://www.analyticsvidhya.com/blog/2021/05/create-your-own-image-dataset-using-opencv-in-machine-learning/

[4] https://towardsdatascience.com/loading-custom-image-dataset-for-deep-learning-models-part-1-d64fa7aaeca6

[5] https://stackoverflow.com/questions/22304500/multiple-or-condition-in-python

[6] https://betterprogramming.pub/stop-using-or-to-check-multiple-conditions-in-python-404d31f2b569

[7] https://www.geeksforgeeks.org/check-multiple-conditions-in-if-statement-python/

[8] https://stackoverflow.com/questions/22304500/multiple-or-condition-in-python

[9] https://github.com/ivangrov/Downloading_Google_Images/blob/main/webscraping_google_images/webscraping_google_images.py

[10] https://www.thepythoncode.com/article/download-web-page-images-python

[11] https://www.thepythoncode.com/code/download-web-page-images-python#download_images_js

[12] https://www.youtube.com/watch?v=stIxEKR7o-c

[13] https://www.technopat.net/sosyal/konu/python-ile-sitedeki-resimleri-indirme.1674149/

[14] http://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.34181&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

[15] https://unsplash.com/s/photos/traffic-signs

[16] https://www.youtube.com/watch?v=Yt6Gay8nuy0

[17] https://www.ijert.org/traffic-sign-detection-and-recognition-using-image-processing

[18] https://data-flair.training/blogs/python-project-traffic-signs-recognition/

[19] https://scikit-learn.org/stable/

[20] https://seaborn.pydata.org/

[21] https://www.kaggle.com/datasets/ajaypalsinghlo/world-happiness-report-2021

[22] https://www.youtube.com/watch?v=ONOK5F3qXCg

[23] https://www.youtube.com/watch?v=cuJR9w5z050

[24] https://www.geeksforgeeks.org/top-7-image-processing-project-ideas-for-beginners/

[25] https://arslanev.medium.com/makine-%C3%B6%C4%9Frenmesi-knn-k-nearest-neighbors-algoritmas%C4%B1-bdfb688d7c5f

[26] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[27] https://en.wikipedia.org/wiki/Vision_transformer

[28] https://tr.wikipedia.org/wiki/CNN

[29] https://github.com/afaq-ahmad/Traffic-Sign-Recognition--HSV-SVM--BelgiumTSC/blob/master/code/1.HOG_with_svm_classifier.ipynb

[30] https://datafiction.github.io/docs/dl/traffic-sign/Traffic_Sign_Classifier/