

T.C

SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

OTONOM SİSTEMLER VE UYGULAMA ALANLARI

TRAFİK İŞARETLERİNİN TESPİTİ

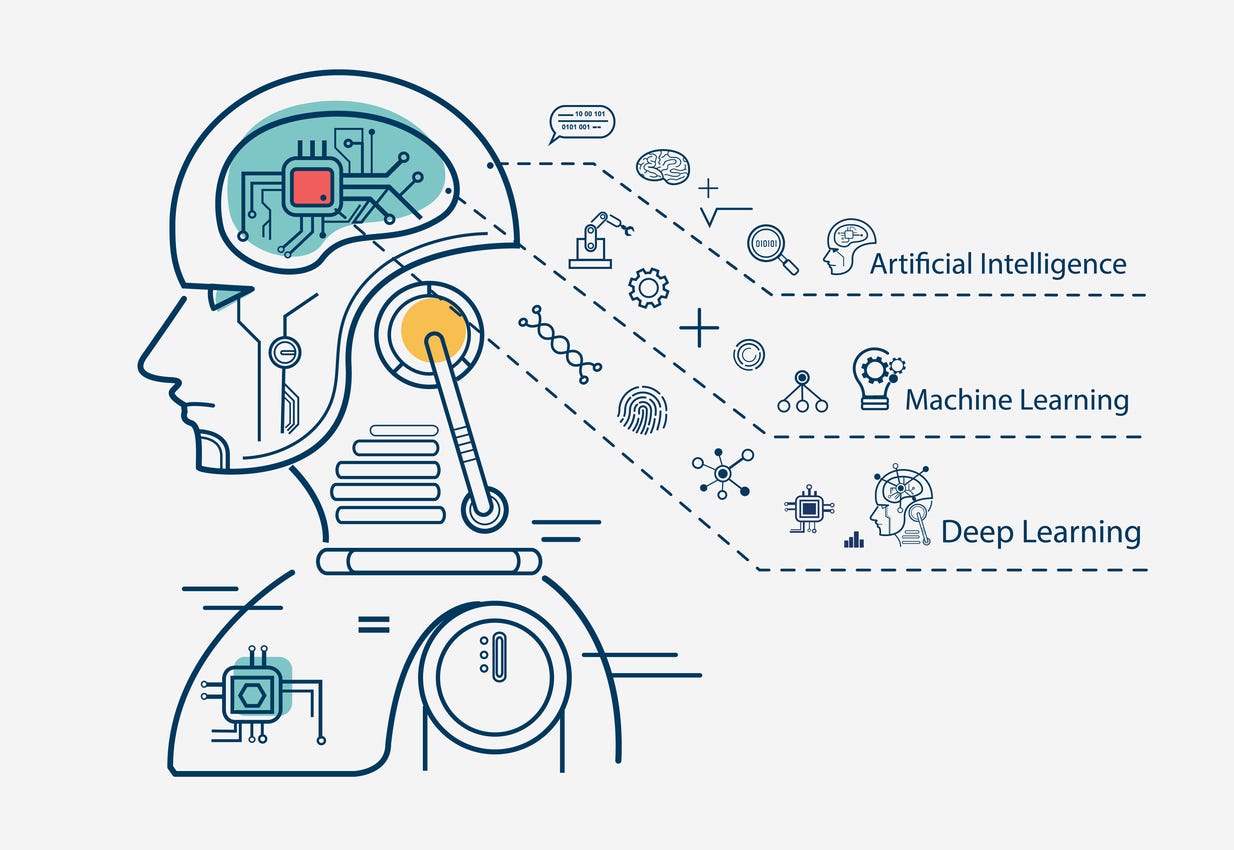
HAZIRLAYANLAR:

SEMİH KAYI 22010103115

SERKAN DELİOĞLAN B200101018

MEHMET BURAK ATILGAN B200101065

1. Derin Öğrenme Modelleri: CNN (Convolutional Neural Networks - Evrişimli Sinir Ağları), trafik işaretlerini tanımak için yaygın olarak kullanılan bir yapay zeka modelidir. Bu modeller, karmaşık özellikleri ve desenleri öğrenerek trafik işaretlerini tanımak için eğitilirler.



1. Veri Kümesi Oluşturma: Bu tür sistemlerin eğitimi için geniş bir veri kümesine ihtiyaç vardır. Farklı açılardan çekilmiş trafik işaretlerini içeren, farklı koşullarda (ışık, hava durumu, yol şartları) çekilmiş geniş ve çeşitli veri setleri kullanılır.



1. Görüntü İşleme Teknikleri: Görüntü işleme algoritmaları, trafik işaretlerini tespit etmek ve ayırt etmek için kullanılır. Renk tanıma, kenar tespiti, özellik çıkarma gibi teknikler işaretleri belirlemek için kullanılabilir.

Görüntü işleme algoritmaları, trafik işaretlerini tanımak ve ayırt etmek için genellikle bir dizi işlemi içerir. Renk tanıma ve kenar tespiti, bu işlemlerin önemli adımlarından ikisidir.

Renk Tanıma: Görüntüdeki renkleri tanımak için, genellikle her pikselin RGB (Kırmızı, Yeşil, Mavi) değerlerini analiz ederiz. Örneğin, bir trafik ışığını tespit etmek için, kırmızı, yeşil ve sarı renkleri algılamak önemlidir. Renk tanıma işlemi genellikle şu adımları içerir:

- Renk Aralıklarının Belirlenmesi: Kırmızı, yeşil, sarı gibi trafik ışığı renklerinin hangi RGB aralıklarına denk geldiği belirlenir.

- Renk Filtreleme: Görüntüde bu renk aralıklarına uygun pikselleri filtrelemek için bir maskelenme veya segmentasyon işlemi uygulanır.

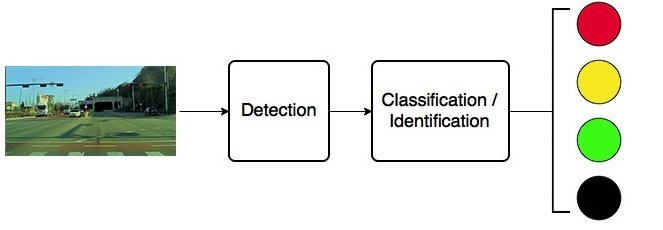
Kenar Tespiti: Görüntüdeki nesnelerin sınırlarını belirlemek veya önemli özellikleri ortaya çıkarmak için kenar tespiti kullanılır. Kenar tespiti genellikle şu yöntemlerle yapılır:

- Sobel ve Canny Filtreleri: Bu filtreler, piksel değerleri arasındaki gradyanı hesaplayarak kenarları tespit eder.

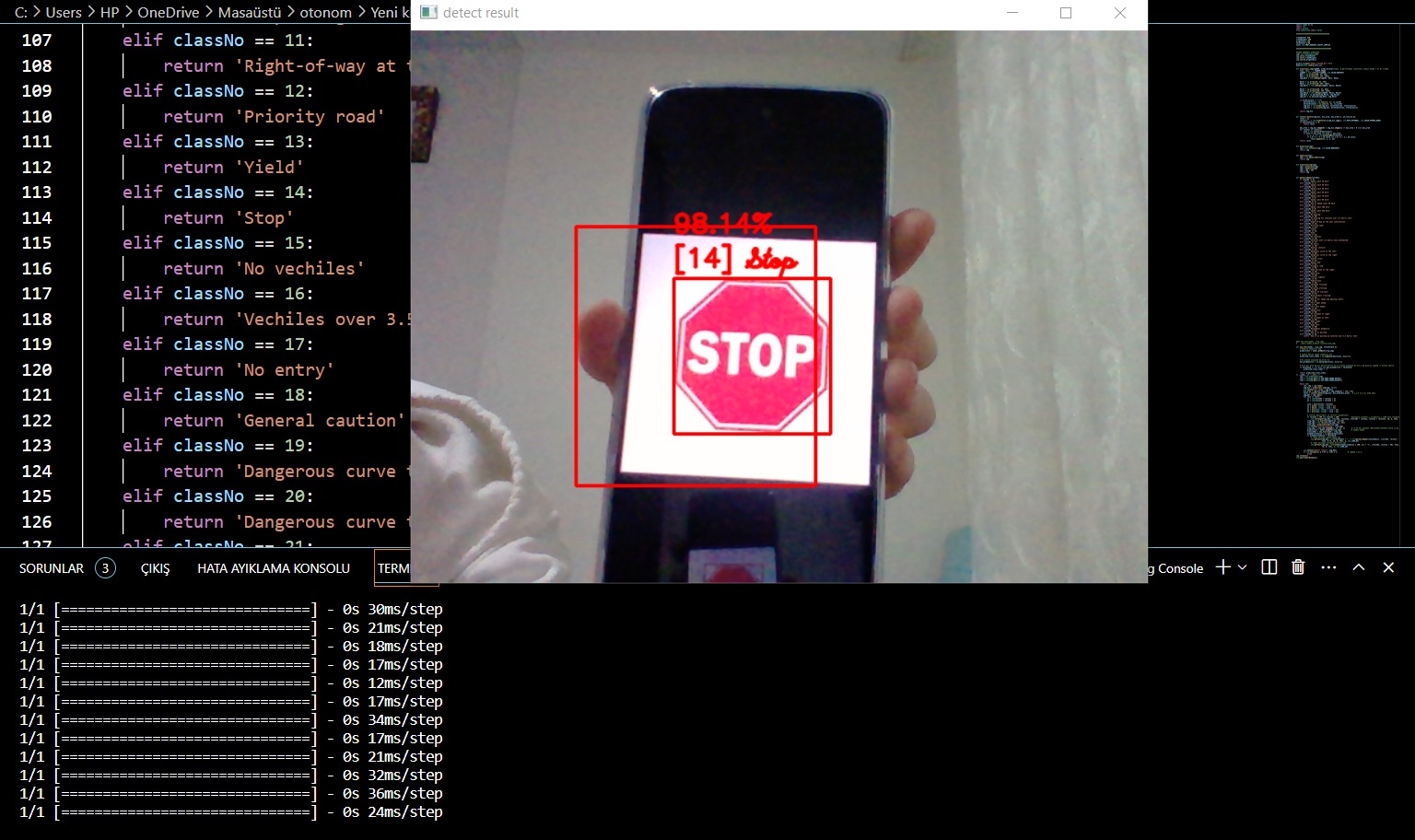
- Laplacian Filtresi: Görüntüdeki piksel değerlerindeki değişiklikleri vurgular ve kenarları belirginleştirir.

- Kenar Belirleme Algoritmaları: Örneğin, Hough dönüşümü gibi algoritmalar, belirli şekilleri veya çizgileri belirlemek için kullanılır.

1. Sınıflandırma ve Tanıma: Eğitilen model, tespit ettiği trafik işaretlerini belirli sınıflara ayırarak (örneğin, dur, ileri git, dönüş yap gibi) tanır ve sınıflandırır.



1. Gerçek Zamanlı Uygulama: Bu sistemler genellikle gerçek zamanlı olarak çalışır, yani bir araç kamerasından gelen görüntüyü anında işler ve sürücüye gerekli uyarıları verebilir.



**KONVOLUSYON SİNİR AĞI MODELİ**

Konvolüsyonel Sinir Ağı (Convolutional Neural Network veya CNN), özellikle görüntü işleme ve tanıma görevlerinde kullanılmak üzere tasarlanmış bir tür yapay sinir ağıdır. CNN'ler, özellikle bilgisayarlı görüş alanında büyük başarı elde etmişlerdir. Temelde şu unsurlardan oluşur.

**Evrişim Katmanları (Convolutional Layers):** Bu katmanlar, görüntüler üzerinde belirli özellikleri (örneğin kenarlar, desenler) öğrenmek için kullanılır. Filtreler (kernels) adı verilen küçük matrisler, görüntü üzerinde kaydırılır ve farklı özellikleri çıkarmak için kullanılır.

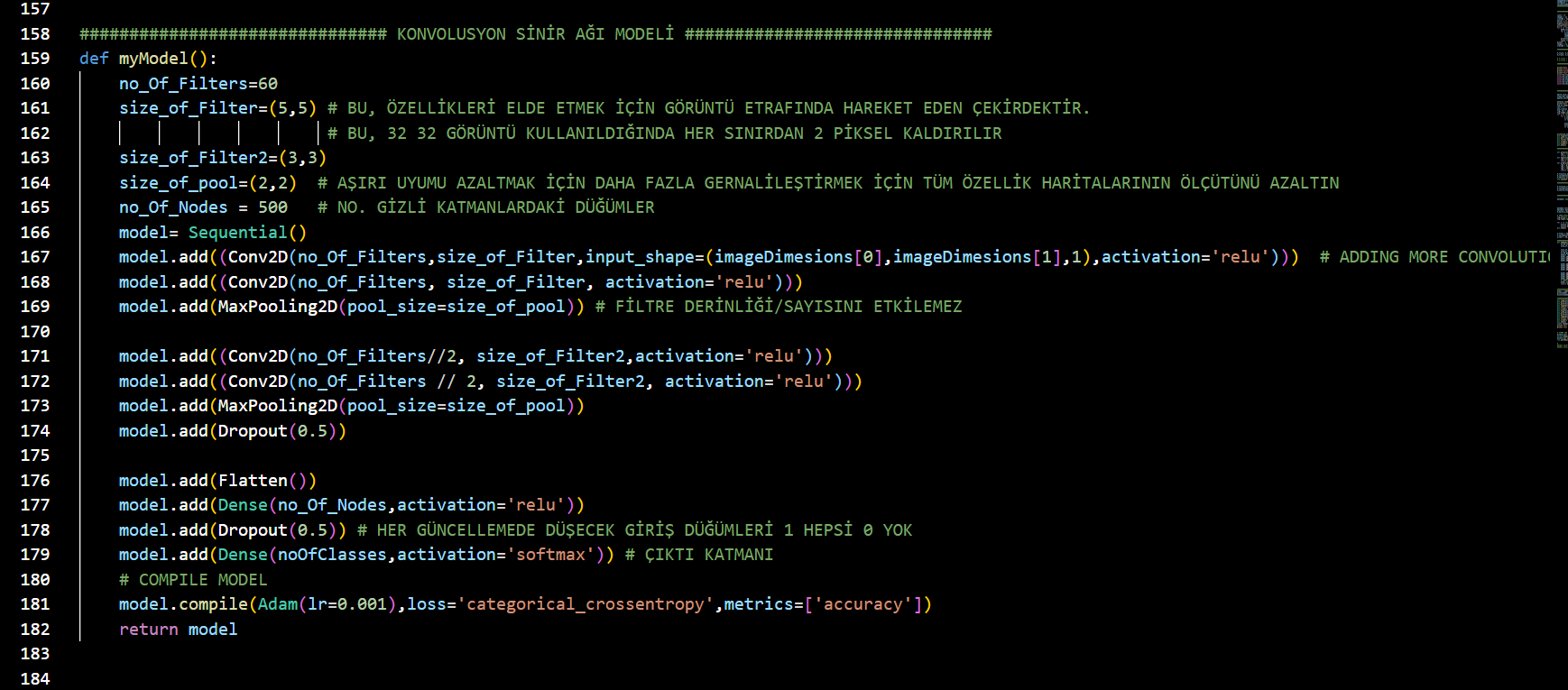
**Havuzlama Katmanları (Pooling Layers):** Bu katmanlar, evrişim katmanlarından elde edilen özellik haritalarının boyutunu küçültmek ve öğrenilen özellikleri ölçeklemek için kullanılır. Tipik olarak, maksimum havuzlama (max pooling) veya ortalama havuzlama (average pooling) kullanılır.

**Tam Bağlantılı Katmanlar (Fully Connected Layers):** Bu katmanlar, önceki katmanlardan elde edilen özellikleri kullanarak sınıflandırma yapmak için kullanılır. Geleneksel yapay sinir ağı katmanlarına benzerdir.

Düşürme (Dropout) Katmanları: Overfitting'i önlemek için kullanılır. Rastgele seçilen nöronları devre dışı bırakarak ağın genelleştirmesine yardımcı olur.

Konvolüsyonel Sinir Ağları, özellikle görüntü sınıflandırma, nesne tespiti, yüz tanıma gibi görsel veri işleme görevlerinde oldukça etkilidir. CNN'ler, veri içerisinde özelliklerin hiyerarşik olarak öğrenilmesine olanak tanıyan bir mimariye sahiptir, bu da onları özellikle karmaşık ve büyük boyutlu veri setleriyle başa çıkabilen güçlü modeller haline getirir.

**KONVOLÜSYON SİNİR AĞI MODELİ KODUMUZ**

****

**1)Fonksiyon Başlatma:**

def myModel():

* 'myModel' adında bir fonksiyon tanımlanmıştır. Bu fonksiyon, bir evrişimli sinir ağı modeli oluşturur.

**2)Hiperparametre Tanımlama:**

no\_Of\_Filters=60

size\_of\_Filter=(5,5)

size\_of\_Filter2=(3,3)

size\_of\_pool=(2,2)

no\_Of\_Nodes = 500

* Modelimizin hiper parametreleri tanımlanmıştır. Filtre sayıları, filtre boyutları, havuzlama boyutları ve gizli katmandaki düğüm sayısı gibi.

**3)Model Oluşturma:**

model = Sequential()

* Keras'ın 'Sequential' modeli başlatılmıştır. Bu, modelin katmanlarını sırasıyla eklemek için kullanılır.

**4)Evrişimli Katmanlar:**

model.add(Conv2D(no\_Of\_Filters, size\_of\_Filter, input\_shape=(imageDimesions[0], imageDimesions[1], 1), activation='relu'))

model.add(Conv2D(no\_Of\_Filters, size\_of\_Filter, activation='relu'))

model.add(MaxPooling2D(pool\_size=size\_of\_pool))

* İlk iki evrişimli katmanlar eklenir. Bu katmanlar, görüntü üzerinde özellikleri çıkarmak için kullanılır. ReLU aktivasyon fonksiyonu kullanılmıştır.
* İlk katmana giriş şekli '(imageDimesions[0], imageDimesions[1], 1)' olarak belirlenmiştir.

**5)Evrişimli Katmanlar (Devam):**

model.add(Conv2D(no\_Of\_Filters//2, size\_of\_Filter2, activation='relu'))

model.add(Conv2D(no\_Of\_Filters//2, size\_of\_Filter2, activation='relu'))

model.add(MaxPooling2D(pool\_size=size\_of\_pool))

model.add(Dropout(0.5))

* İkinci iki evrişimli katmanlar ve bir maksimum havuzlama katmanı eklenir.
* Dropout katmanı, ağın genelleştirmesine yardımcı olmak için eklenmiştir. Dropout, belirli bir olasılıkla rastgele düğümleri devre dışı bırakır.

**6)Flatten ve Tam Bağlantılı Katmanlar:**

model.add(Flatten())

model.add(Dense(no\_Of\_Nodes, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(noOfClasses, activation='softmax'))

* Flatten katmanı, özellik haritalarını düzleştirerek bir giriş vektörü oluşturur.
* İki tam bağlantılı katman eklenir. İlk katman 500 düğüme sahiptir ve ReLU aktivasyon fonksiyonu kullanır. Dropout kullanılarak overfitting azaltılmaya çalışılır.
* Çıkış katmanı, sınıflandırma probleminde kullanılan softmax aktivasyon fonksiyonuna sahiptir.

**7)Model Derleme:**

model.compile(Adam(lr=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])

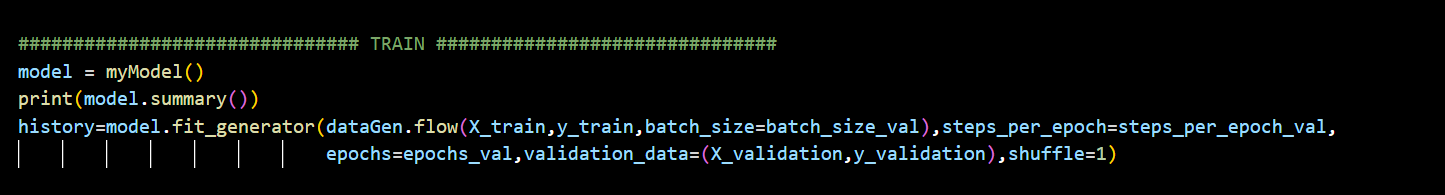
* Model derlenir. Adam optimizer, categorical crossentropy kaybını minimize etmeye çalışacaktır.
* Doğruluk (accuracy) metriği kullanılarak modelin performansı ölçülür.

**8)Modeli Döndürme:**

return model

* Oluşturulan model, fonksiyon tarafından döndürülür.

**TRAIN**



**1)Modelin Oluşturulması:**

model = myModel()

* Önceki açıklamada tanımlanan myModel fonksiyonu kullanılarak bir model oluşturulur.

**2)Model Özetinin Yazdırılması:**

print(model.summary())

* Oluşturulan modelin özetini ekrana yazdırır. Modelin katmanları, çıkış şekli, toplam parametre sayısı gibi bilgileri içerir.

**3)Eğitim Verilerinin Oluşturulması ve Modelin Eğitimi:**

history = model.fit\_generator(dataGen.flow(X\_train, y\_train, batch\_size=batch\_size\_val),

steps\_per\_epoch=steps\_per\_epoch\_val,

epochs=epochs\_val,

validation\_data=(X\_validation, y\_validation),

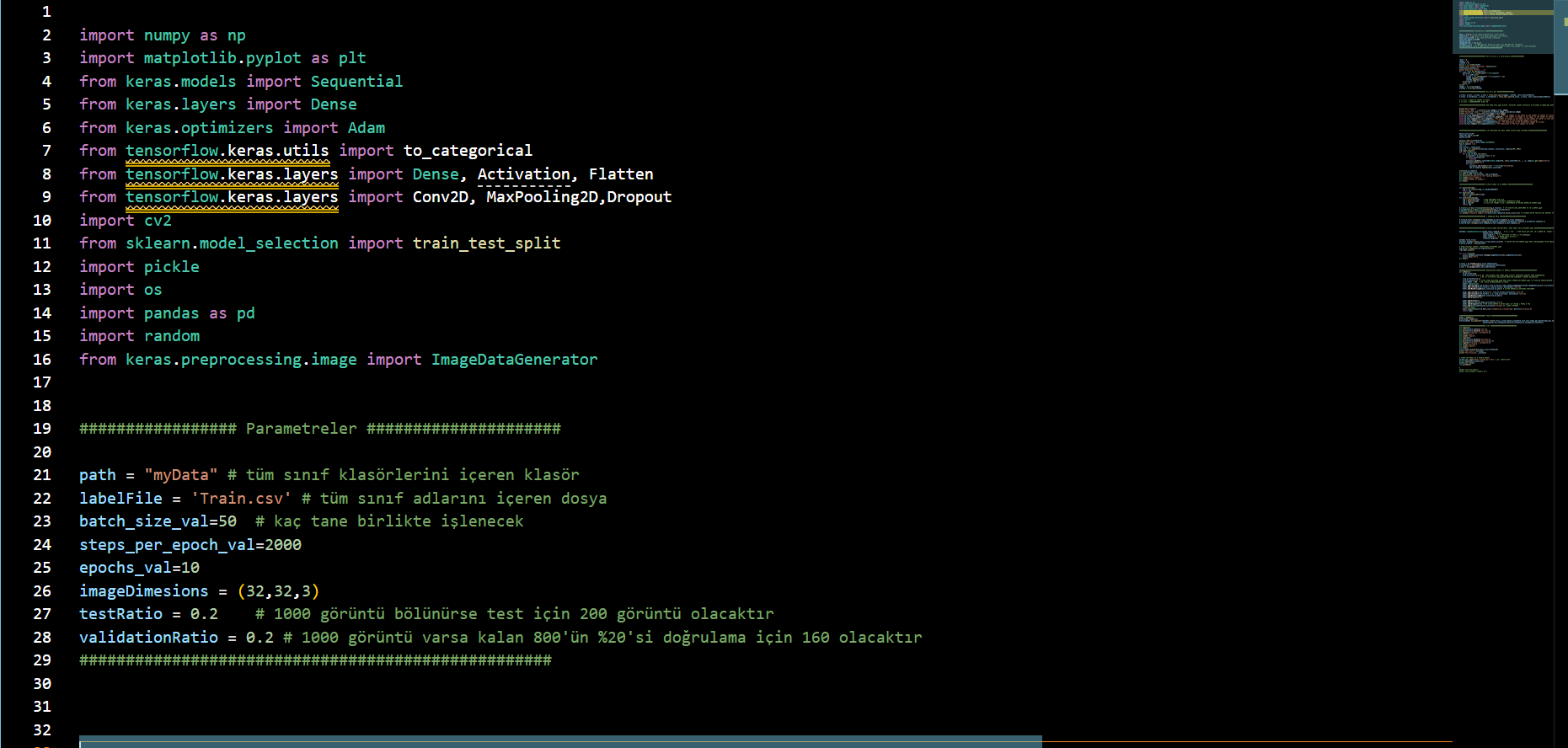
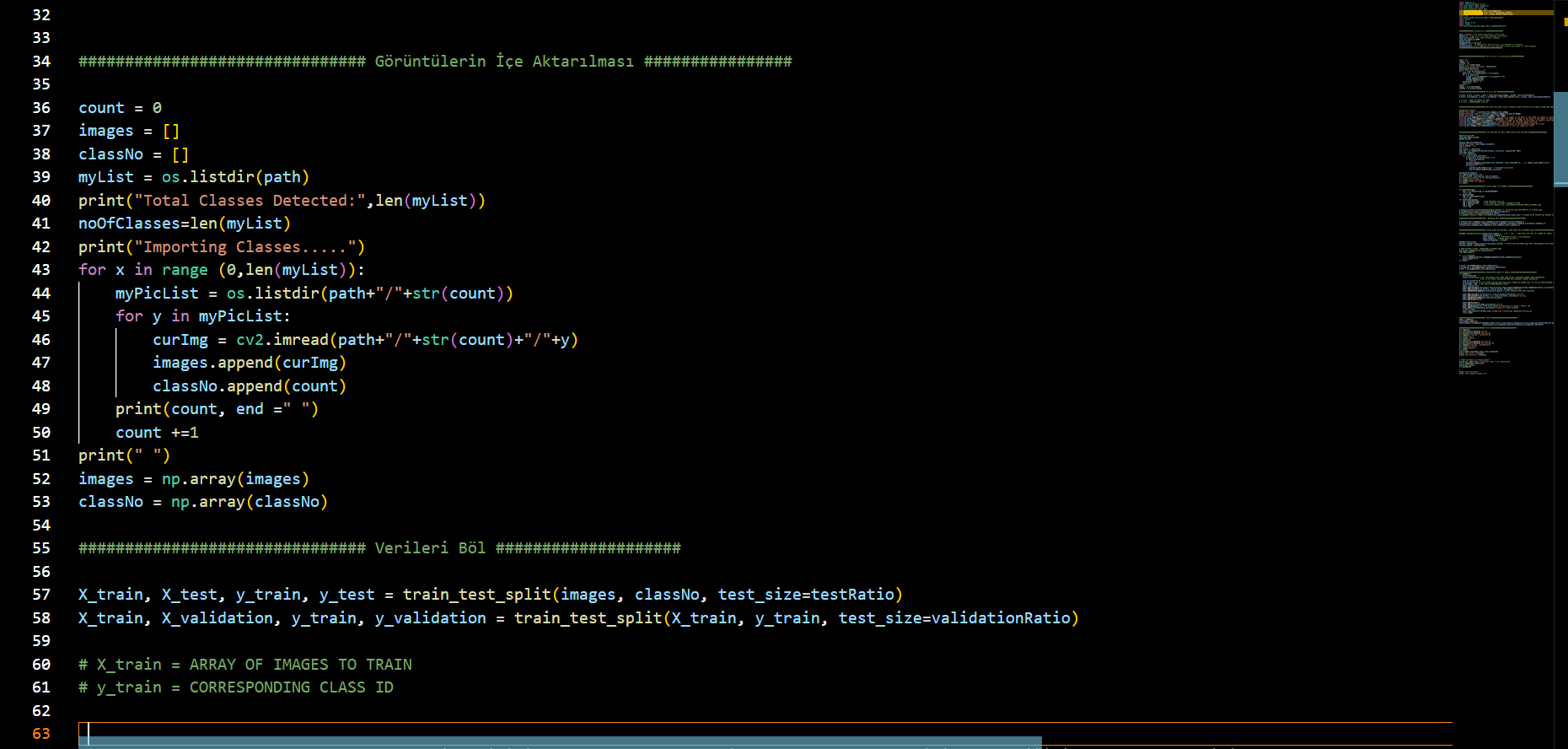
shuffle=1)

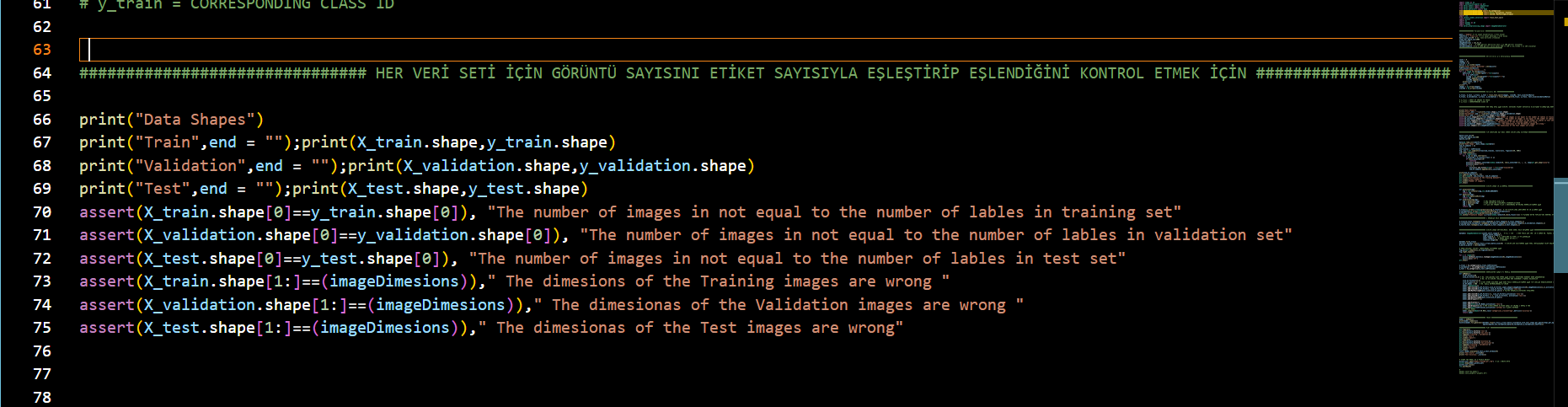
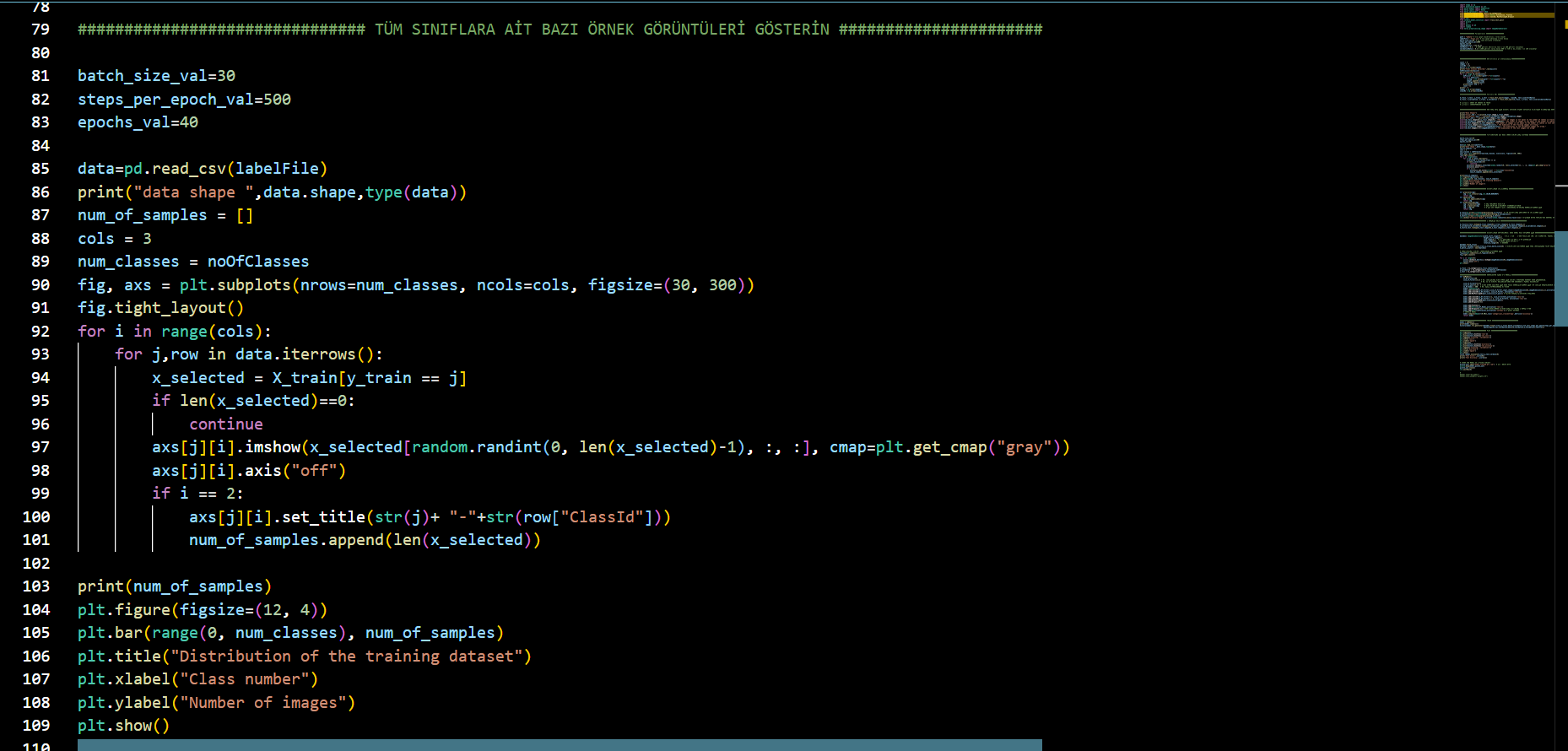
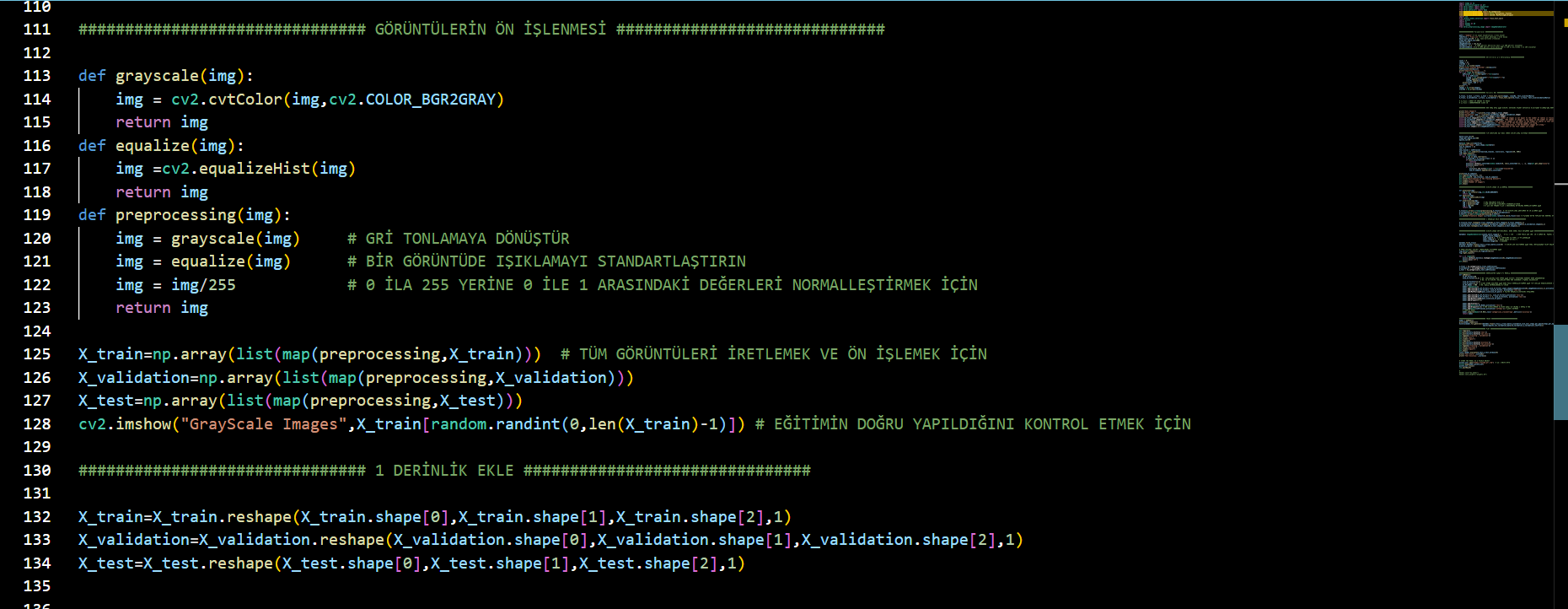
* 'fit\_generator' fonksiyonu, modelin eğitimini gerçekleştirir. Bu, veri artırma (data augmentation) ile birlikte kullanılabilen bir yöntemdir.
* 'dataGen.flow' fonksiyonu, veri artırma için kullanılır. X\_train ve y\_train verilerini kullanarak eğitim verisi üretir.
* 'steps\_per\_epoch\_val', her bir eğitim epoch'unda kaç adımın gerçekleştirileceğini belirtir.
* 'epochs\_val', eğitim epoch sayısını belirtir.
* 'validation\_data' parametresi, eğitim sırasında doğrulama için kullanılan veri setini belirtir.
* 'shuffle=1' parametresi, her epoch öncesi verilerin karıştırılmasını sağlar.

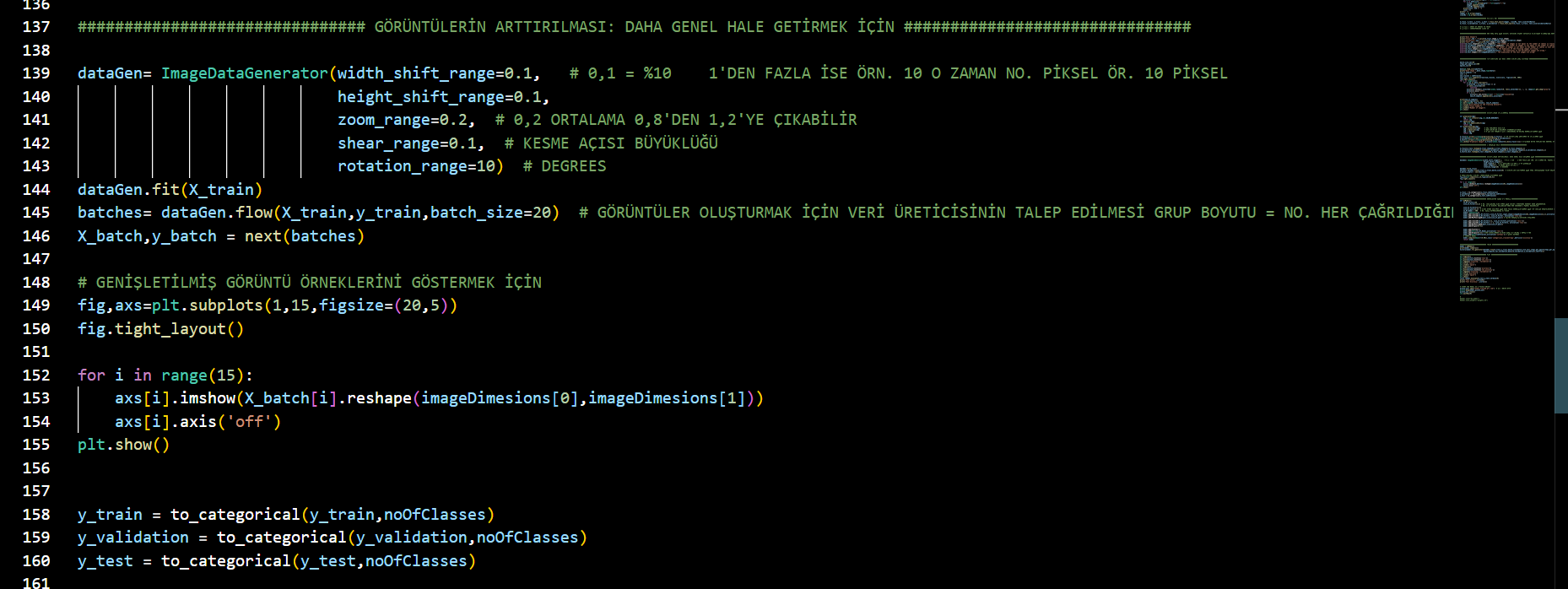
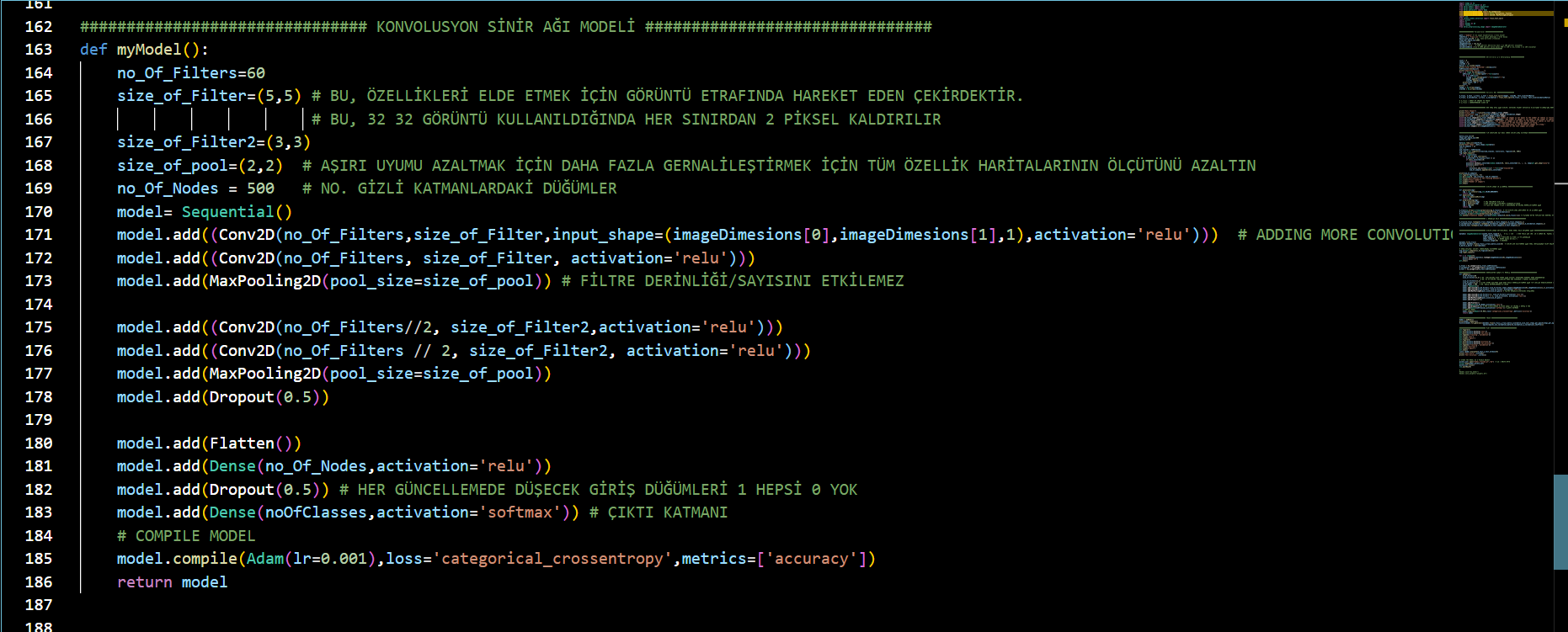
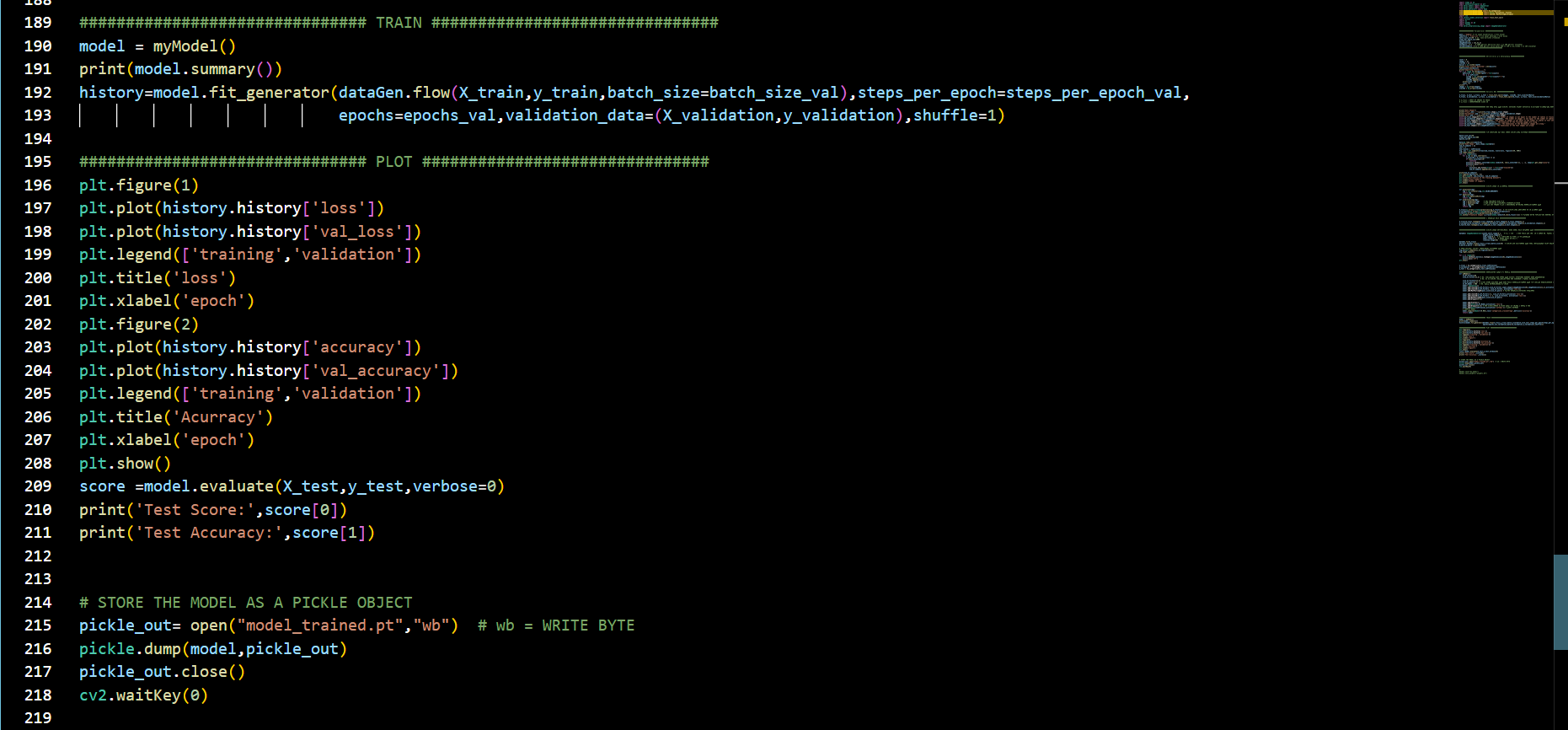
**4)Eğitim Geçmişinin Kaydedilmesi:**

* 'history' adlı değişken, modelin eğitim geçmişini içerir. Eğitim sırasında kaydedilen metrik değerleri içerir (örneğin, loss ve accuracy).

**TRAFİK İŞARETLERİNİ ALGILAMA SİSTEMİ EĞİTİM KODUMUZ**

****

****

****

**Trafik İşaretlerinin Tespit Edilmesi**

1.Adım→ Veri Toplama ve Etiketleme

- İlk adım, modelin öğrenmesi için kullanılacak geniş ve çeşitli trafik işareti görüntülerini içeren bir veri setinin oluşturulmasıdır.

- Her görüntüde bulunan trafik işaretleri etiketlenir. Yani, görüntüde hangi trafik işaretinin bulunduğu belirtilir.

2.Adım→ Veri Ön İşleme

- Toplanan veriler, genellikle boyut standartlaştırma, renk düzeltme ve gürültü azaltma gibi ön işleme adımlarından geçirilir. Bu, modelin daha etkili öğrenmesine yardımcı olur.

3.Adım→ Evrişimli Sinir Ağı (CNN) Modelinin Eğitimi

- Eğitim veri seti üzerinde bir evrişimli sinir ağı (CNN) modeli eğitilir. Bu model, görüntülerdeki özellikleri öğrenir ve farklı trafik işaretlerini tanıma yeteneği kazanır.

4.Adım→ Modelin Doğrulanması ve Ayarlanması:

- Eğitim sürecinden sonra model, doğrulama veri seti üzerinde test edilir. Modelin başarı oranı değerlendirilir ve gerekiyorsa hiperparametreler ayarlanarak veya model mimarisi değiştirilerek performansı artırılır.

5.Adım→ Tespit İşlemi:

- Eğitilen model, bir kamera görüntüsündeki trafik işaretlerini tespit etmek için kullanılır.

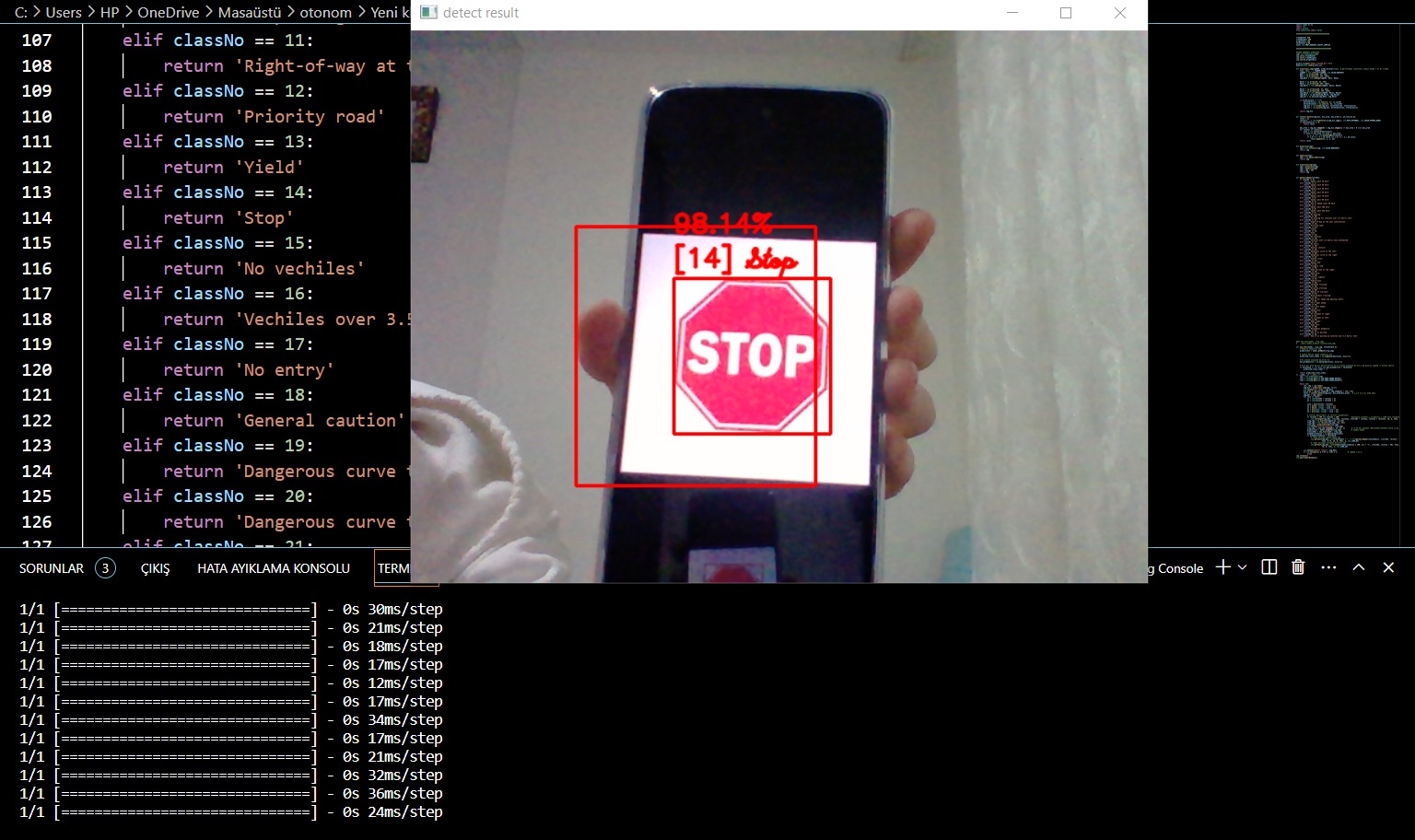
- Görüntü, küçük parçalara bölünür ve bu parçalardaki olası trafik işaretleri model tarafından değerlendirilir.

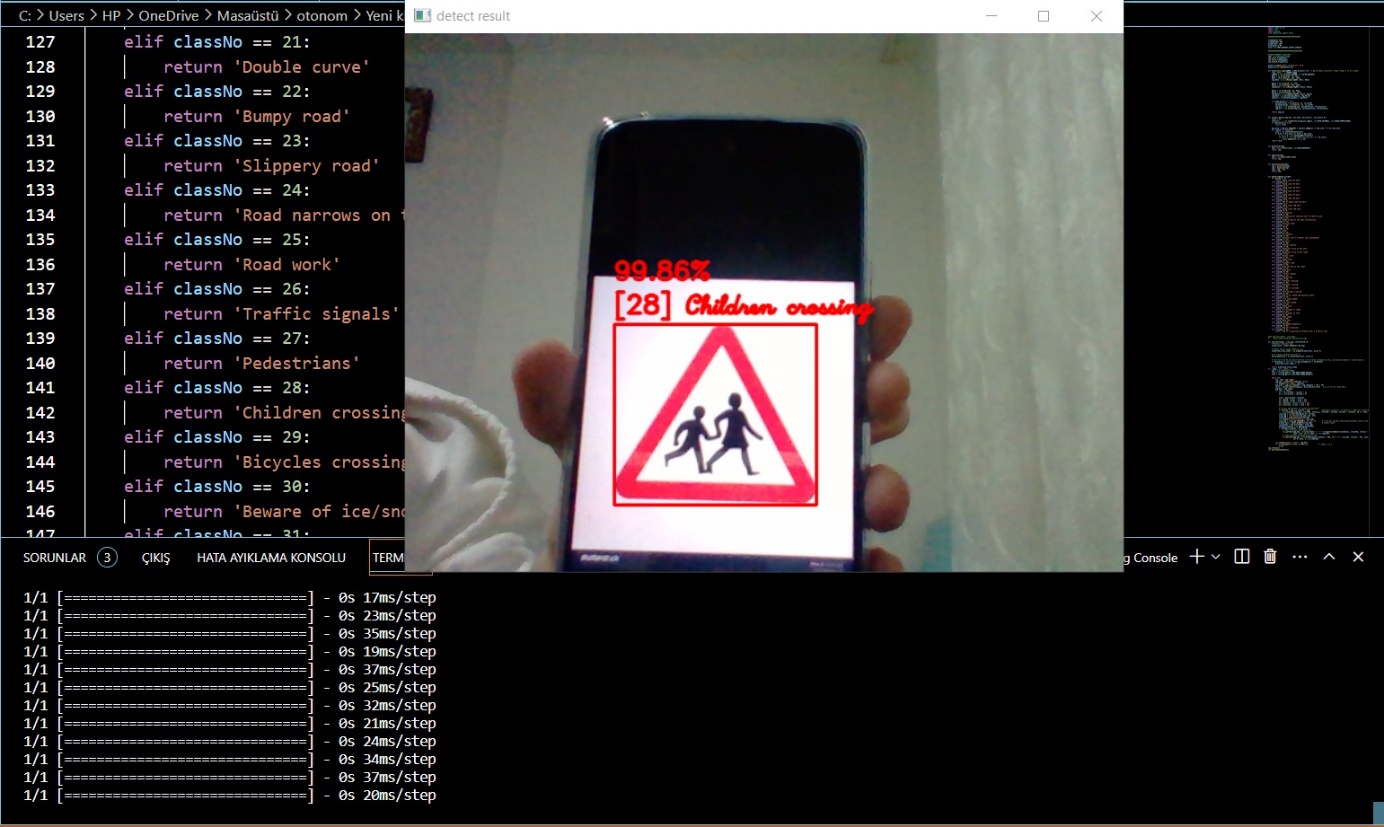
- Eğitilen model, trafik işareti sınıflarını belirleyebilir ve her trafik işareti için konum bilgilerini sağlayabilir.

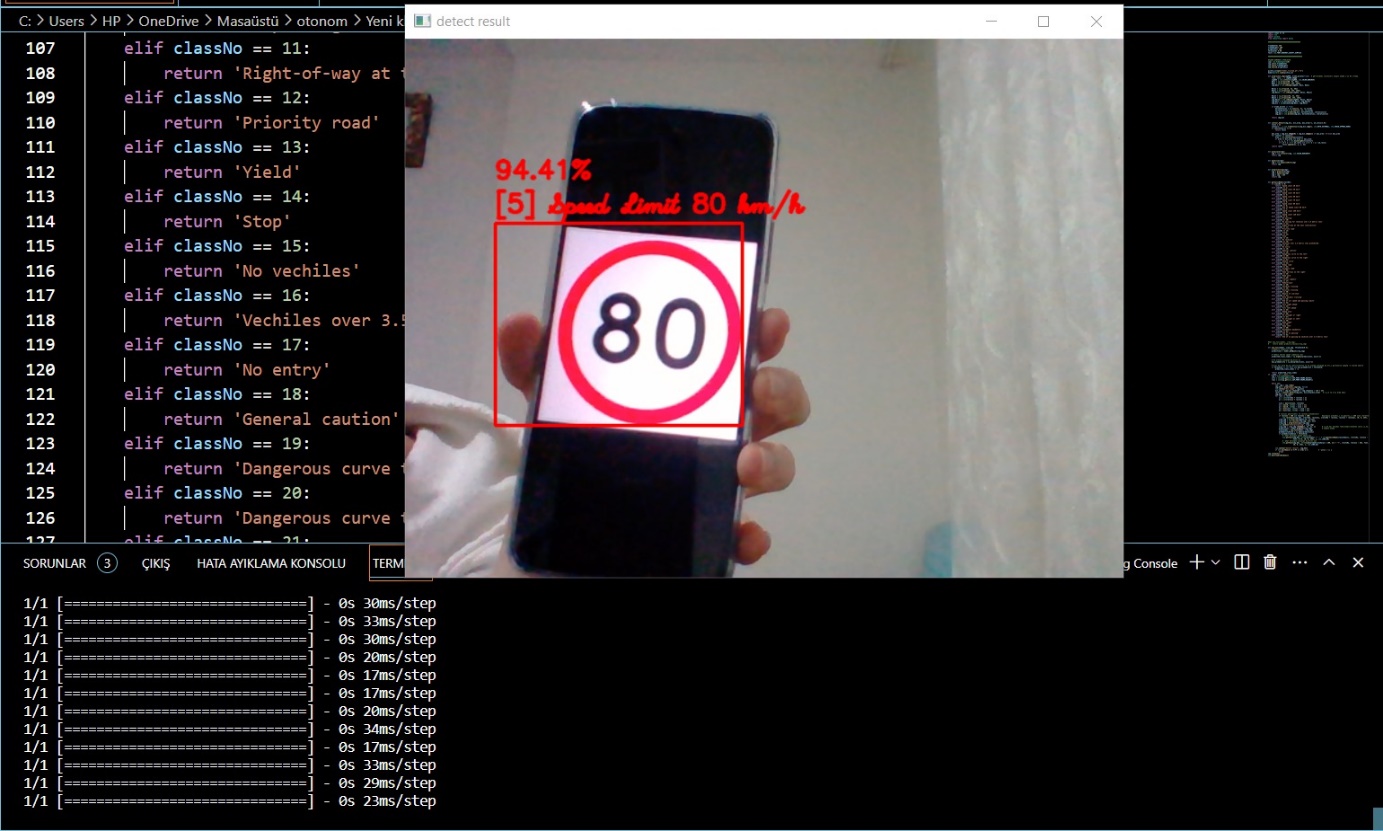
6.Adım→ Sonuçların Görselleştirilmesi

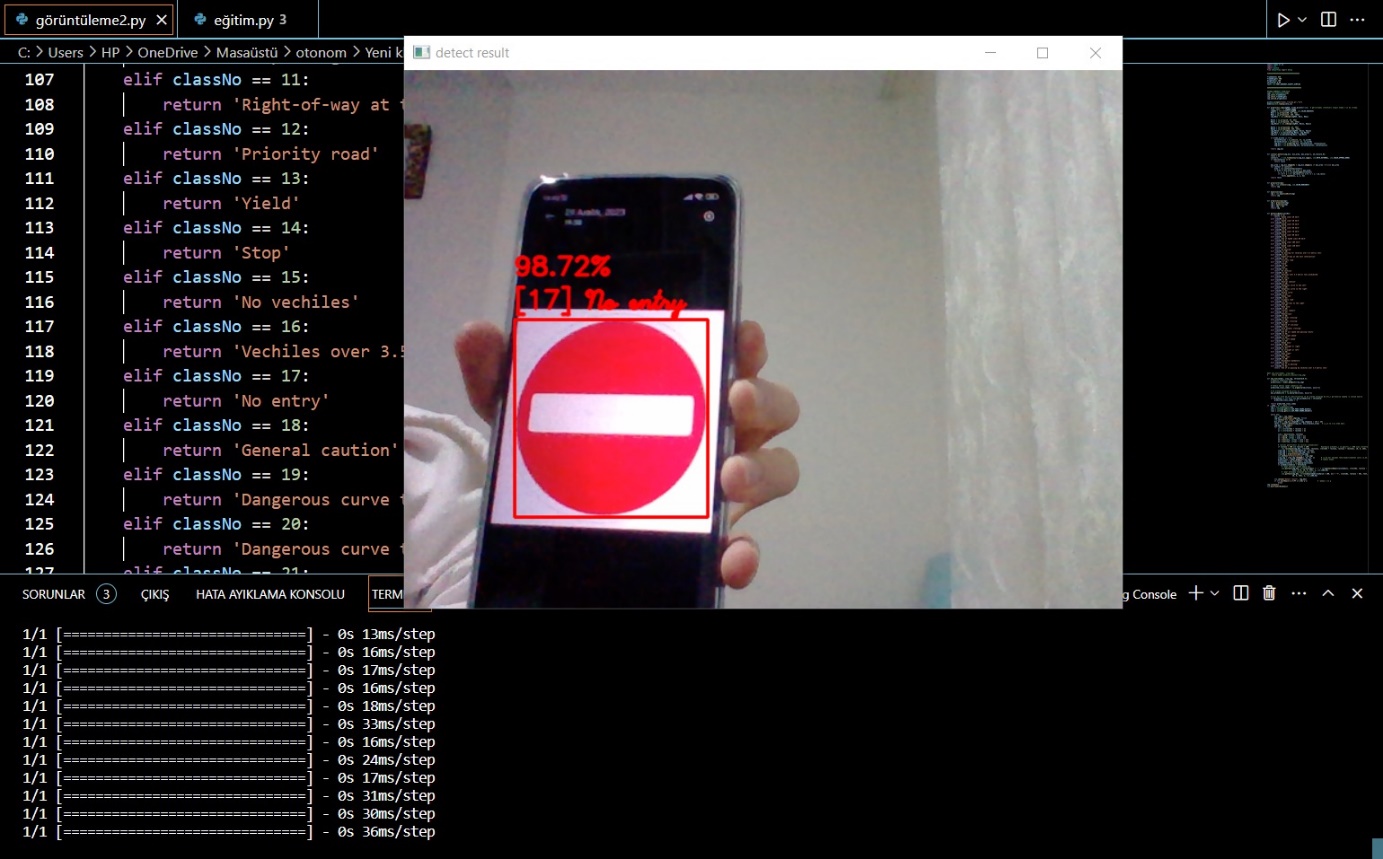
- Tespit edilen trafik işaretleri, genellikle çerçeve içinde vurgulanır veya işaretlenir. Bu, kullanıcının veya sistem operatörünün tespit sonuçlarını görmesine yardımcı olur.

**Çalışmamızdan Örnek Çıktılar**









SORUNLAR VE ÇÖZÜMLERİMİZ

1)Bilgi bulutu içindeki projelerin çoğuna aşina olmak istedik fakat bu durum, kafa karışıklığına yol açtı.

- Bilgi bulutunda projeler arasında gezinirken karmaşık bir durumla karşılaştık. Projeleri kategorize ederek daha anlaşılır bir yapı oluşturmayı düşündük. Bu sayede, belirli kategorilere odaklanarak araştırma ve filtreleme yapabilir, anahtar kriterler belirleyerek projeleri değerlendirdik. Ayrıca, adım adım ilerleyerek önceliklendirme yapmak ve arkadaşlarımızdan yardım almak, bu bilgi yığınını daha yönetilebilir hale getirebildik. Bu adımlar, karmaşık ve geniş bir alandaki bilgiyi daha anlaşılır ve kullanılabilir hale getirmemize yardımcı oldu.

2)Veri setinin oluşturulması ve projeye uygun veri setinin seçimi.

3)Proje için gerekli kütüphaneleri ulaşmak ve nasıl kullanıldığını öğrenmek.

4)Veri setinin bilgisayarda bulunduğu konumu yol olarak kod derleyicisine aktarmak ve çalıştırmak.

5) Belirli bir nesneyi sınıflandırmak için kullanılan model, benzer özelliklere sahip farklı nesneleri aynı sınıfa dahil etmesi.

- Modelin daha fazla çeşitlilik içeren bir veri seti üzerinde eğitilmesi, benzer özelliklere sahip farklı nesneleri ayırt etme konusunda daha etkili olmasını sağladı.

6) Test aşamasında bilgisayar performansının yavaşlaması ve bizi sonuca geç ulaştırması.

- Nvidia'nın CUDA programını kullanarak, GPU'nun verimliliğini artırdık ve işlemleri hızlandırdık. CUDA, Nvidia'nın paralel hesaplama platformu olarak kullanılarak, GPU'nun paralel işleme yeteneklerinden yararlanarak hesaplama süreçlerini daha etkili bir şekilde optimize etmemizi sağladı. Bu sayede, hesaplamaların büyük bir kısmını GPU üzerinde gerçekleştirerek daha hızlı sonuçlar elde ettik ve iş yüklerini daha verimli bir şekilde dağıtarak performansı artırdık.