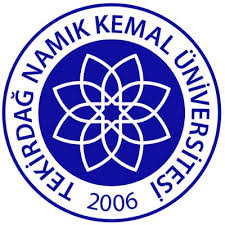
**T.C**

**TEKİRDAĞ NAMIK KEMAL ÜNİVERSİTESİ**

**ÇORLU MÜHENDİSLİK FAKÜLTESİ**

****

**B****İLGİSAYAR MÜHENDİSLİĞİ PROJE-II DERSİ BİTİRME**

**ÇALIŞMASI**

**OTONOM ARAÇLAR İÇİN TRAFİK IŞIKLARININ SINIFLANDIRILMASI**

**Ad ve Soyad :YAĞMUR KAHYA**

**No :2170656025**

**DANIŞMAN**

**Dr. Öğr. Üyesi Erkan ÖZHAN**

**ÖZET**

**Projenin Amacı :** Yapay Zeka ve teknoloji dünyasında birçok araştırmacı ve Tesla, Uber, Google, Mercedes-Benz, Toyota, Ford, Audi gibi büyük şirketler otonom araçlar ve sürücüsüz arabalar üzerinde çalışıyor. Dolayısıyla bu teknolojide doğruluğa ulaşmak için araçların trafik işaretlerini yorumlayabilmesi ve buna göre kararlar alabilmesi gerekmektedir.

Yolcuların seyahat için tamamen araçlara güvenebilmesi için otonom araçların trafik işaretlerini, kurallarını anlaması ve bunlara uyması gerekir.

Bu projede otonom araçların bu trafik işaretlerini anlaması için trafik işaretleri sınıflandırma işleminin yapılması amaçlanmıştır.

Trafik işaretleri sınıflandırması, bir trafik işaretinin hangi sınıfa ait olduğunu belirleme işlemidir.

**Projenin İlerleyişi:** Bu Python projesinde, görüntüde bulunan trafik işaretlerini farklı kategorilere ayırabilen derin bir sinir ağı modeli oluşturulur. Bu model ile tüm otonom araçlar için çok önemli bir görev olan trafik işaretlerini okuyup anlayabilmesi amaçlanmıştır.

**İÇİNDEKİLER**

**Sayfa**

**ÖZET…………………………………………………………………………………….….ii**

**İÇİNDEKİLER……………………………………………………………………………..iii**

**ÖNSÖZ…………………………………………………..………………………………….iv**

**1. GİRİŞ………………………………………….………………………………………….1**

1.1.Projeni̇nTanımı..**………………………………………………………………………..1**

**2. MATERYAL ve YÖNTEM………………………………………………………………1**

2.1. Proje Gereksi̇ni̇mleri̇…………………………………………………………………..1

2.2.Keras……………………………………………………………………………………1

2.3.Matplotlib………………………………………………………………………………2

2.4.Scikit-Learn…………………………………………………………………………….2

2.5.Pandas…………………………………………………………………………………..2

2.6.PIL………………………………………………………………………………………2

2.7.Tkinter…………………………………………………………………………………..2

**3.PROJE KODLARI…….**…………..………………………………………………………3

3.1.Dataser………………………………………………………………………………….3

3.2.CNN Model……………………………………………………………………………..4

3.3..ModelinEğitilmesi………………………………………………………………..…….5

3.4.Modelin Test Edilmesi………………………………………………………………….7

**4.Trafffic Sings Classifier GUI.…………………………………………………………….8**

5**. TARTIŞMA ve SONUÇ………………………………………………………………….11**

**5. KAYNAKÇA…………………………………………………………………………….12**

**ÖNSÖZ**

Proje çalışmamızın planlanmasında, yürütülmesinde ilgi ve desteğini esirgemeyen, Ekan Özhan ’a, Bilgisayar Mühendisliği Bölüm Öğretim Elemanlarına ve Tekirdağ Namık Kemal Üniversitesi’ne en içten teşekkürlerimizi sunarız.

Ayrıca eğitimimiz süresince bize her konuda tam destek veren ailemize ve bize değerli bilgiler kazandıran tüm hocalarımıza saygı ve sevgilerimizi sunarız.

**Tarih Ad ve Soyadlar**

**Yağmur Kahya**

**1. GİRİŞ**

Bu projede, görüntüde bulunan trafik işaretlerini farklı kategorilere ayırabilen derin bir sinir ağı modeli oluşturacağız. Bu model ile tüm otonom araçlar için çok önemli bir görev olan trafik işaretlerini okuyup anlayabiliyoruz.

**Kaggle ‘ da bulunan GTSRB - German Traffic Sign Recognition Benchmark dataseti kullanılmıştır.**

* 1. **Projeni̇n Tanımı**

Yapay Zeka ve teknoloji dünyasında birçok araştırmacı ve Tesla, Uber, Google, Mercedes-Benz, Toyota, Ford, Audi gibi büyük şirketler otonom araçlar ve sürücüsüz arabalar üzerinde çalışıyor. Dolayısıyla bu teknolojide doğruluğa ulaşmak için araçların trafik işaretlerini yorumlayabilmesi ve buna göre kararlar alabilmesi gerekmektedir.

Yolcuların seyahat için tamamen araçlara güvenebilmesi için otonom araçların trafik işaretlerini, kurallarını anlaması ve bunlara uyması gerekir.

Bu projede otonom araçların bu trafik işaretlerini anlaması için trafik işaretleri sınıflandırma işleminin yapılması amaçlanmıştır.

**2. MATERYAL VE YÖNTEM**

**2.1. Proje Gereksi̇ni̇mleri̇**

**Bu projede Keras, Matplotlib, Scikit-learn, Pandas, PIL ve görüntü sınıflandırması kullanılmıştır.**

GTSRB - German Traffic Sign Recognition Benchmark dataseti kullanılmıştır.

**2.2.Keras**

Keras, neredeyse her tür derin öğrenme modelini tanımlamak ve eğitmek için uygun bir yol sağlayan Python için bir derin öğrenme kütüphanesidir. Keras, **Tensorflow** , **Theano** ve **CNTK** üzerinde çalışabilen Python ile yazılmış bir üst düzey sinir ağları API’sıdır.

İçerdiği çok fazla işlevsel fonksiyon sayesinde Keras kolayca bir derin öğrenme modeli oluşturmamızı ve onu eğitmemizi sağlıyor. Bu nedenle derin öğrenmeye yeni başlayanlara önerilen kütüphanelerin başında Keras geliyor.

**2.3.Matplotlib**

Matplotlib; veri görselleştirmesinde kullandığımız temel python kütüphanesidir. 2 ve 3 boyutlu çizimler yapmamızı sağlar. Matplotlib genelde 2 boyutlu çizimlerde kullanılırken, 3 boyutlu çizimlerde başka kütüphanelerden yararlanılır.

**2.4.Scikit-Learn**

Scikit-learn, veri bilimi ve machine learning için en yaygın kullanılan Python paketlerinden biridir. Birçok işlemi gerçekleştirmenizi sağlar ve çeşitli algoritmalar sağlar. Scikit-learn ayrıca sınıfları, yöntemleri ve işlevleri ile kullanılan algoritmaların arka planıyla ilgili belgeler sunar.

**2.5.Pandas**

Pandas, “ilişkisel” ve “etiketli” verilerle çalışmayı kolay ve sezgisel hale getirmek için tasarlanmış hızlı, esnek ve etkileyici veri yapıları sağlayan bir Python paketidir. Python’da pratik, gerçek dünya veri analizi yapmak için temel yapı taşı olmayı hedefler. Ayriyeten, her dilde mevcut olan en güçlü ve esnek açık kaynak veri analizi / manipülasyon aracı olmak gibi daha geniş bir amaca sahiptir.

**2.6.PIL**

Python Image Library , yani Python Resim Kütüphanesi, Pythonda image işlemlerini kolayca yapabilmek için geliştirilmiş kütüphanedir .

**2.7.Tkinter**

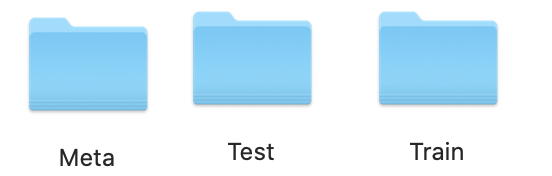
Tkinter, Python'un fiili standart GUI (Grafik Kullanıcı Arayüzü) paketidir.

Python, Tkinter ile birlikte kullanıldığında GUI uygulamaları hızlı ve kolay bir şekilde oluşturulur. Tkinter, Tk GUI araç setine güçlü bir nesne yönelimli arayüz sağlar. Python2’de “Tkinter”, Python3’te “tkinter” modülü ile temsil edilir.

Tkinter kullanarak bir GUI uygulaması oluşturmak kolaydır. Tek yapmamız gereken aşağıdaki adımları gerçekleştirmektir;

* Tkinter modülünü uygulamaya import edin.
* GUI uygulama ana penceresini oluşturun.
* GUI uygulamasına yukarıda belirtilen widget’lardan birini veya daha fazlasını ekleyin.
* Kullanıcı tarafından tetiklenen her etkinliğe karşı harekete geçmek için ana etkinlik döngüsünü girin.

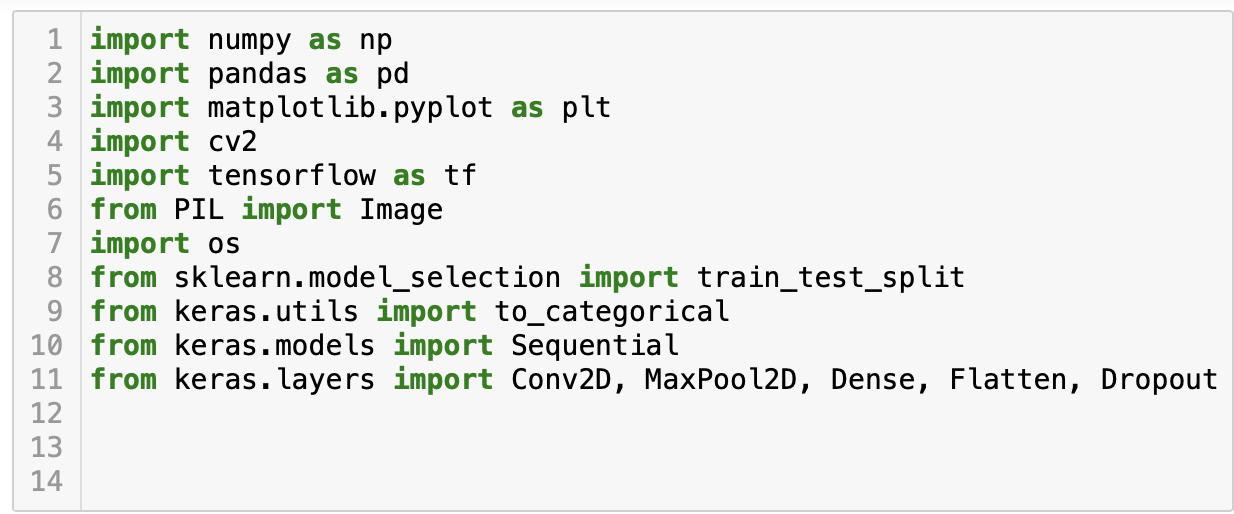
**3.PROJE KODLARI**



**3.1.Dataset**

Train klasörü her biri farklı bir sınıfı temsil eden 43 klasör içerir. Klasörün aralığı 0 ile 42 arasındadır. İşletim sistemi modülünün yardımıyla, tüm sınıfları yineler ve veri ve etiketler listesine görüntüleri ve ilgili etiketleri ekleriz.

PIL kitaplığı, görüntü içeriğini bir diziye açmak için kullanılır.



Tüm görüntüleri ve etiketlerini listelere (veriler ve etiketler) kaydettik.

Modeli beslemek için listeyi numpy dizilerine dönüştürmemiz gerekiyor.

Verinin şekli (39209, 30, 30, 3), yani 30×30 piksel boyutunda 39.209 görüntü olduğu ve son 3'ü verinin renkli görüntüler (RGB değeri) içerdiği anlamına gelir.

Sklearn paketiyle, eğitim ve test verilerini bölmek için train\_test\_split () yöntemini kullanıyoruz.

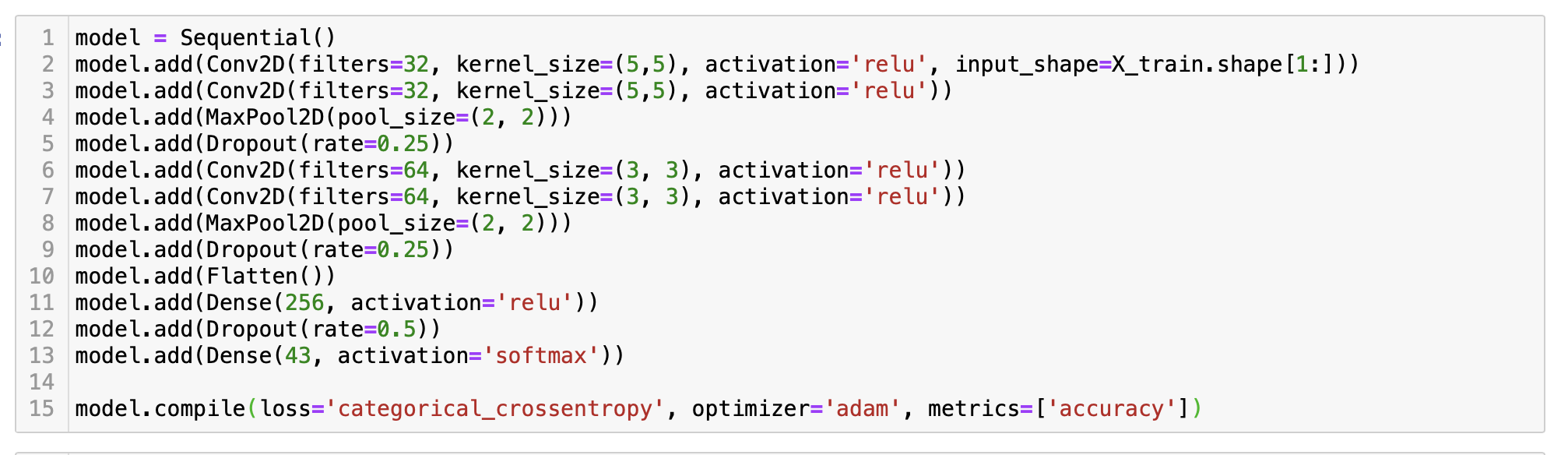
Keras.utils paketinden, y\_train ve t\_test'te bulunan etiketleri one-hot encodinge dönüştürmek için to\_categorical yöntemini kullanıyoruz.

**3.2.CNN Model**

Görüntüleri kendi kategorilerine göre sınıflandırmak için bir CNN modeli oluşturacağız.

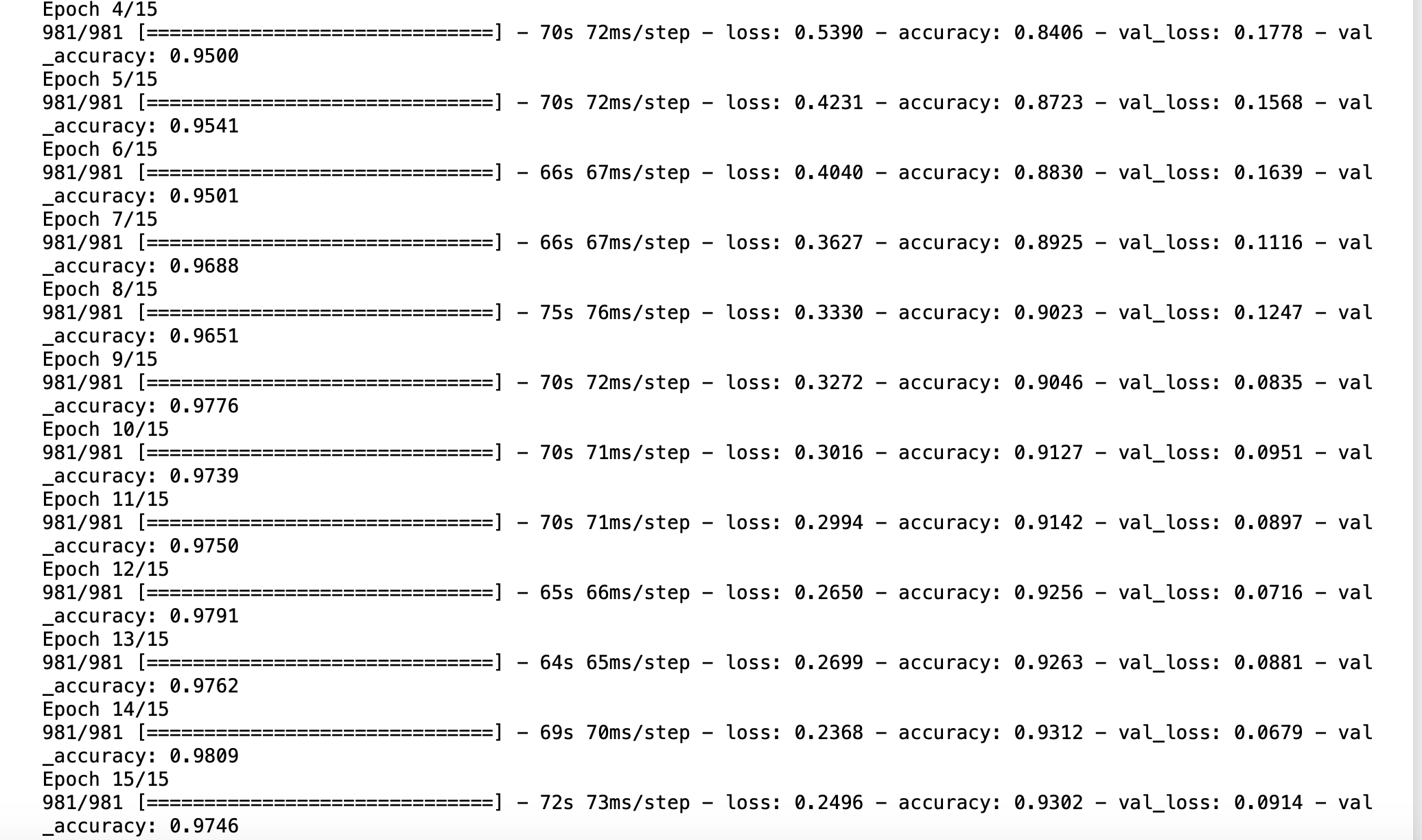
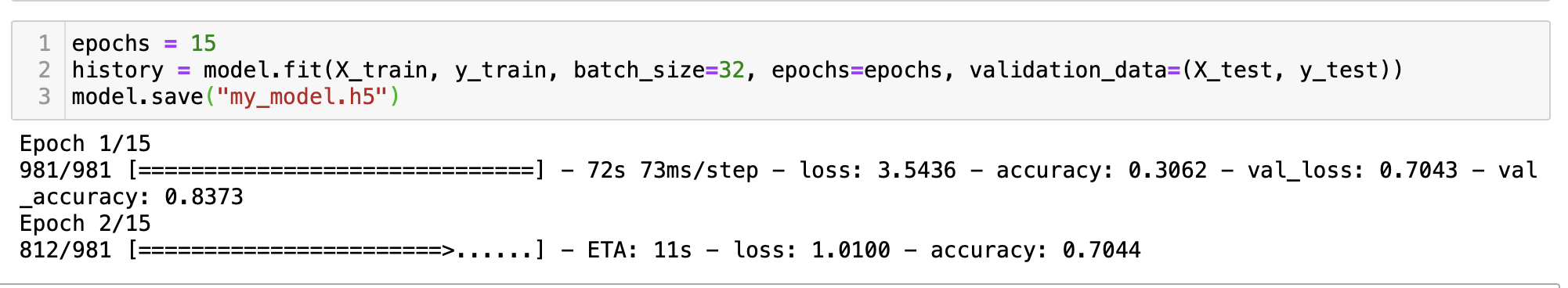
Modelin mimarisi;

* 2 Conv2D layer (filter=32, kernel\_size=(5,5), activation=”relu”)
* MaxPool2D layer ( pool\_size=(2,2))
* Dropout layer (rate=0.25)
* 2 Conv2D layer (filter=64, kernel\_size=(3,3), activation=”relu”)
* MaxPool2D layer ( pool\_size=(2,2))
* Dropout layer (rate=0.25)
* Flatten layer to squeeze the layers into 1 dimension
* Dense Fully connected layer (256 nodes, activation=”relu”)
* Dropout layer (rate=0.5)
* Dense layer (43 nodes, activation=”softmax”)

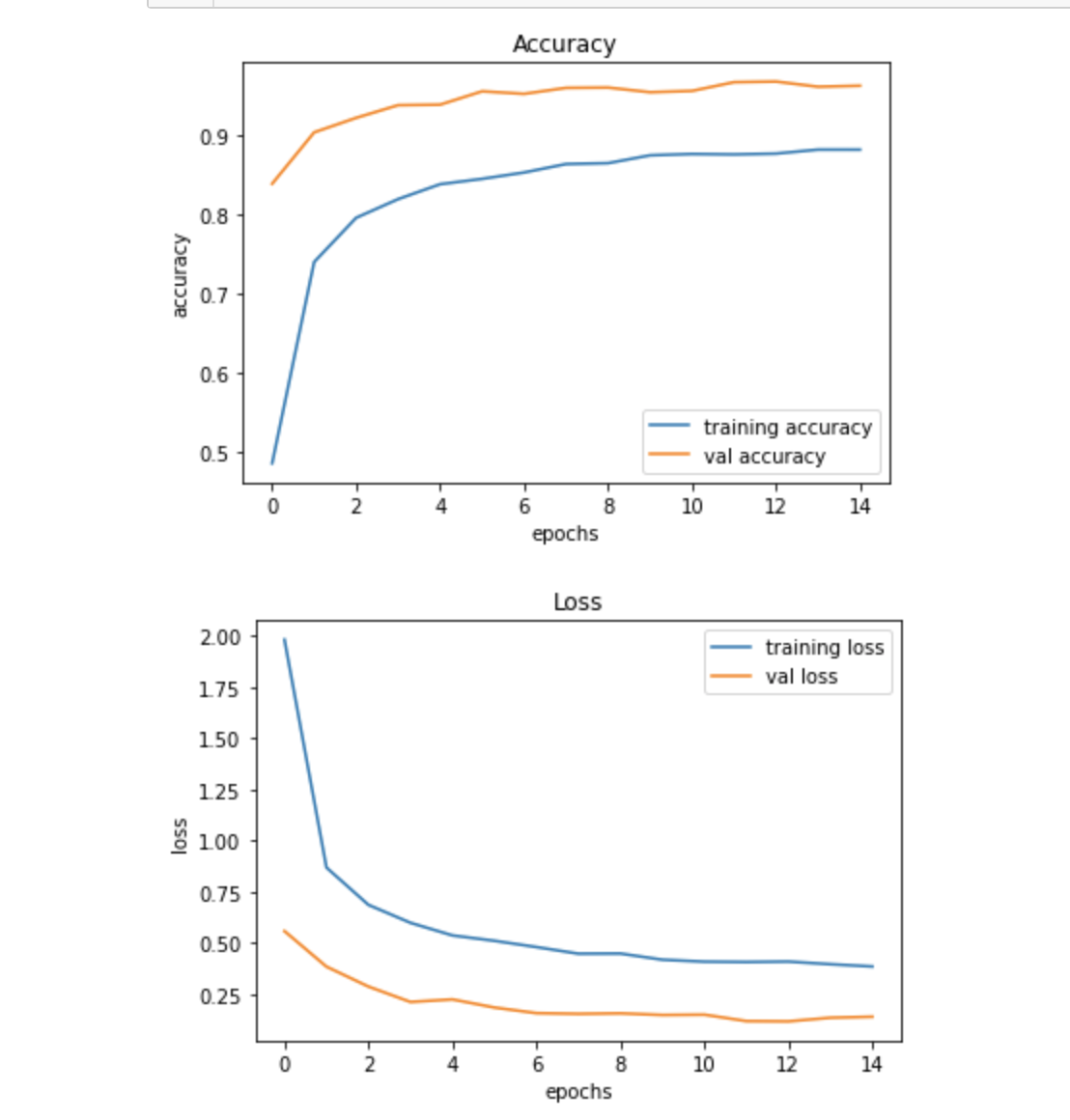
Modeli, iyi performans gösteren Adam optimizer ile derliyoruz ve kategorize edilecek birden fazla sınıfımız olduğu için kayıp “categorical\_crossentropy” oluyor.

3.3.Modelin Eğitilmesi

Model mimarisini oluşturduktan sonra modeli **model.fit()** kullanarak eğitiriz.

****

Modelimiz eğitim veri setinde %93 doğruluk elde etti. Matplotlib ile, doğruluk ve kayıp için grafiği çiziyoruz.

****

3.4.Modelin Test Edilmesi

Veri kümemiz bir test klasörü içerir ve bir test.csv dosyasında görüntü yolu ve ilgili sınıf etiketleriyle ilgili ayrıntılara sahibiz.

Pandas kullanarak görüntü yolunu ve etiketleri çıkarıyoruz.

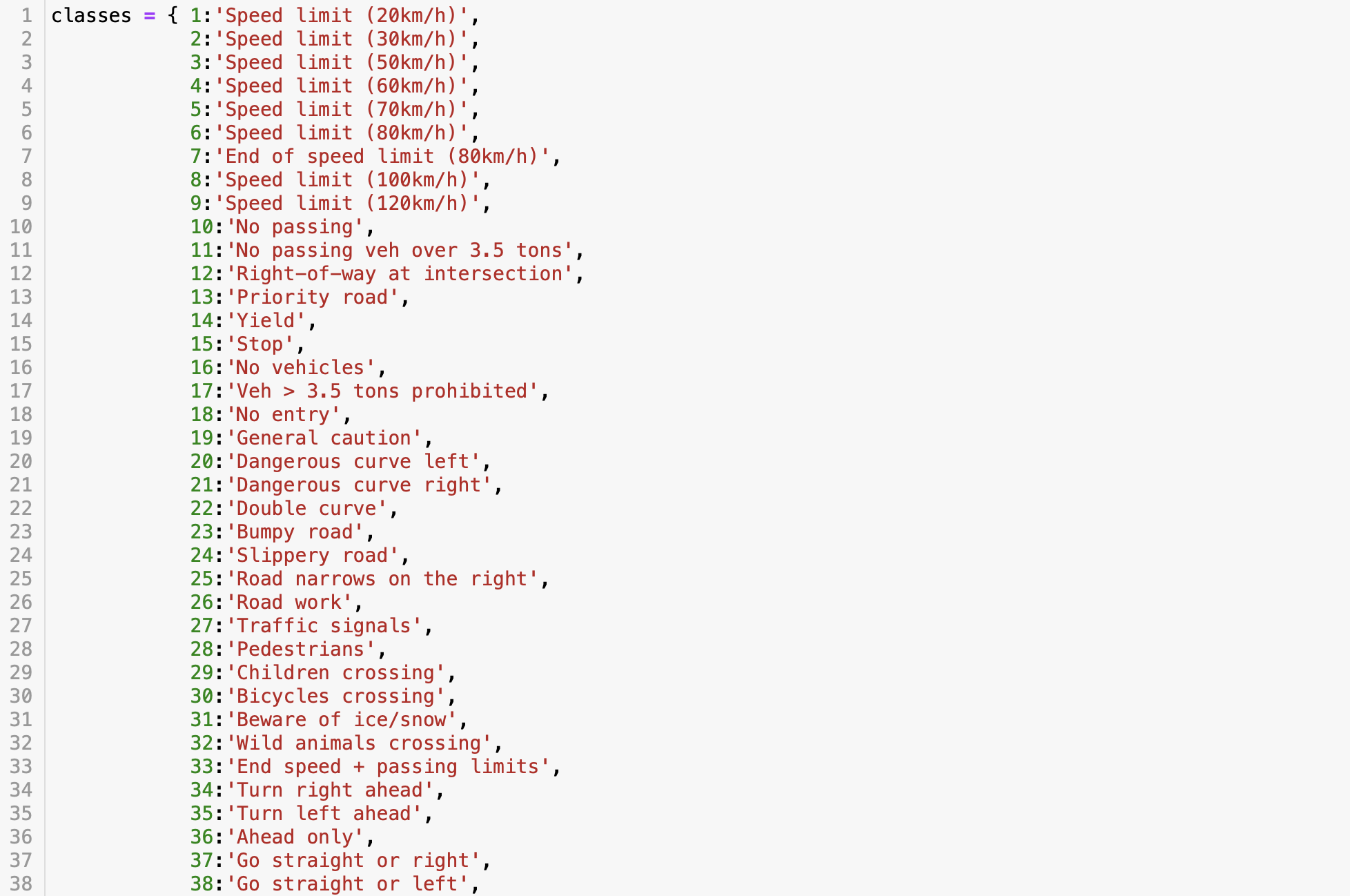
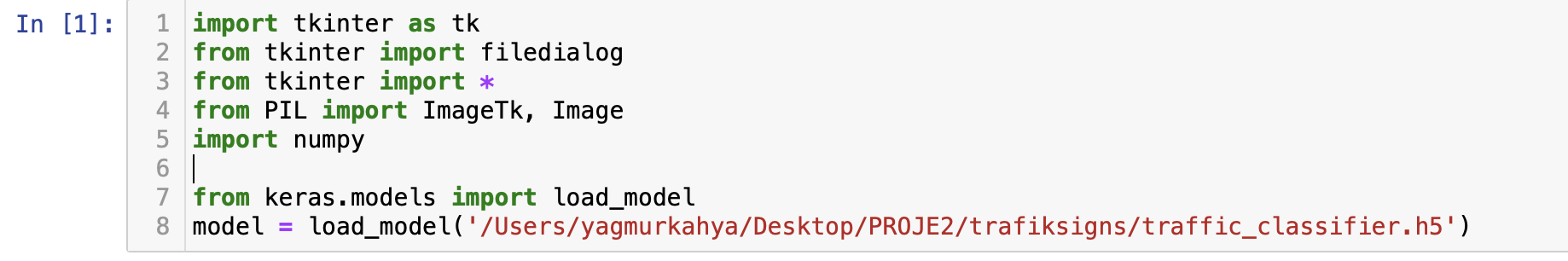
Daha sonra modelin tahmini için resimlerimizi 30×30 piksel olarak yeniden boyutlandırmamız ve tüm resim verilerini içeren bir dizi oluşturmamız gerekiyor.

sklearn.metrics'den accuracy\_score ‘u import ettik ve modelimizin gerçek etiketleri nasıl tahmin ettiğini gözlemledik. Bu modelde %93 doğruluk elde ettik.



Son olarak Keras **model.save()** fonksiyonu ile eğittiğimiz modeli kaydediyoruz.

4.Traffic Signs Classifier GUI

Şimdi Tkinter ile trafik işaretleri sınıflandırıcımız için bir grafik kullanıcı arayüzü oluşturacağız. Tkinter, standart python kitaplığındaki bir GUI araç takımıdır.

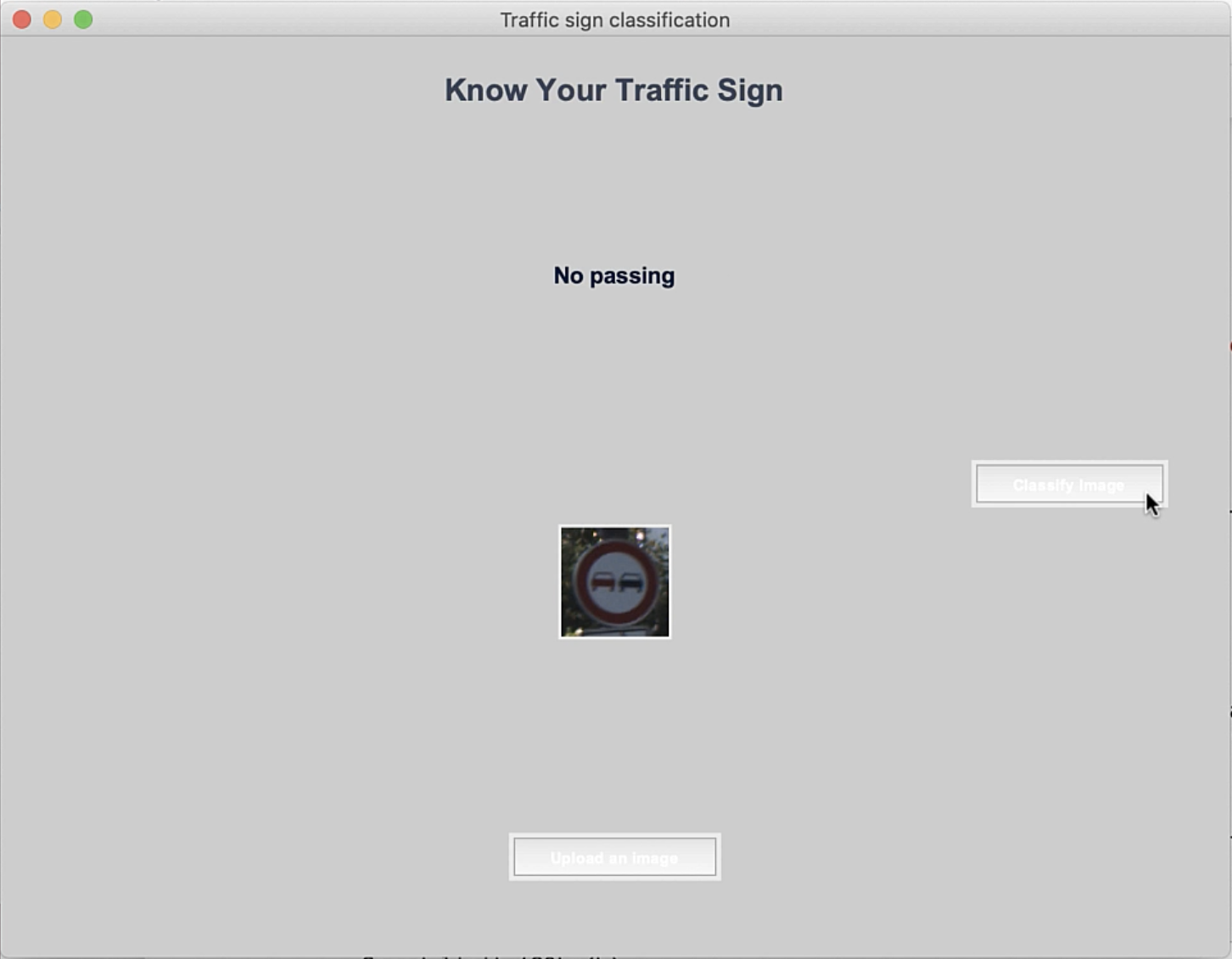
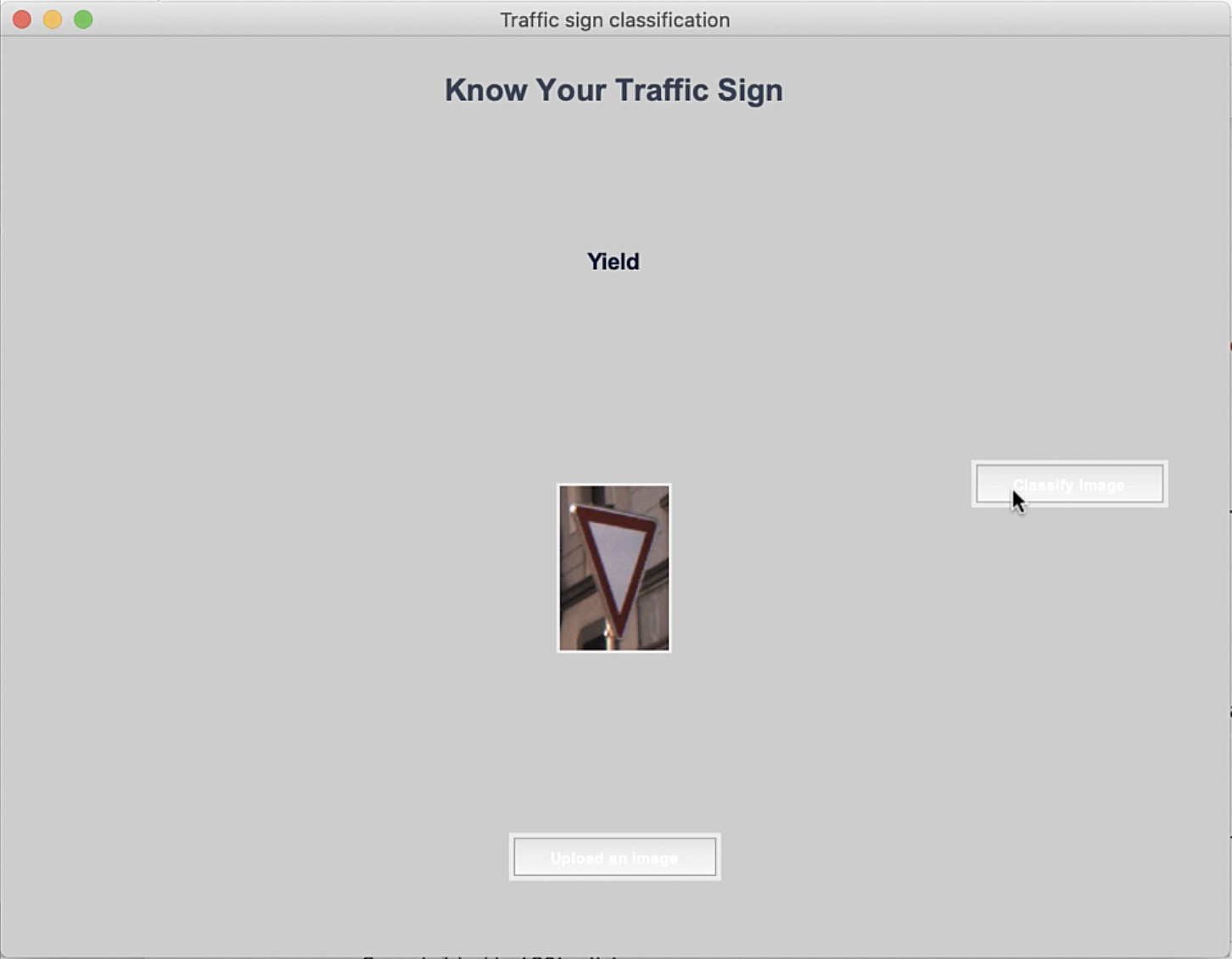


İlk olarak Keras kullanarak eğitilmiş "traffic\_classifier.h5" modelini yükledik.

Ardından görüntüyü yüklemek için GUI'yi oluşturuyoruz ve classify() işlevini çağıranları sınıflandırmak için bir düğme kullanılıyor.

classify() işlevi, görüntüyü şekil boyutuna (1, 30, 30, 3) dönüştürüyor. Bunun nedeni, trafik işaretini tahmin etmek için modeli oluştururken kullandığımız boyutun aynısını sağlamamız gerektiğidir.

Sonra sınıfı tahmin ederiz, model.predict\_classes(image) bize ait olduğu sınıfı temsil eden (0-42) arasında bir sayı verir. Sınıf hakkında bilgi almak için sözlüğü kullanırız.



**5. TARTIŞMA ve SONUÇ**

Bu Python projesinde, trafik işaretleri sınıflandırıcısı için bir CNN modeli oluşturduk %93 başarıyla sınıflandırdık ve doğruluk ve kaybımızın zamanla nasıl değiştiğini görselleştirdik.

Tkinter GIU ile sınıflandırma işlemi için arayüz oluşturduk, eğittiğimiz CNN modelini ekledik.

**6.KAYNAKÇA**

**https://www.kaggle.com/GTSRB - German Traffic Sign Recognition Benchmark**

**<https://www.kaggle.com/marinovik/recognizing-traffic-signals-with-keras-cnn>**

**<https://towardsdatascience.com/traffic-sign-recognition-using-deep-neural-networks-6abdb51d8b70>**

**<https://erdincuzun.com/ileri-python/tkinter-gui/>**