

**EGE ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**YAPAY ZEKÂ YÖNTEMLERİ (3+0)**

**2022-2023 BAHAR YARIYILI**

**PROJE-2 RAPORU**

**TESLİM TARİHİ**

17/06/2023

**HAZIRLAYANLAR**

05200000045, Şevval GÖNÜL

İçindekiler

[1) Problemin Tanımı 2](#_Toc137929423)

[2) Araştırma (Ön çalışma) 2](#_Toc137929424)

[3) Kullanılan Ortam, Yöntem ve Kütüphaneler 4](#_Toc137929425)

[4) Önerilen Yöntem 4](#_Toc137929426)

[5) Deneysel Çalışmalar 6](#_Toc137929427)

[6) Sonuç 9](#_Toc137929428)

[8) Ek 1 9](#_Toc137929429)

[9) Ek 2 10](#_Toc137929430)

[10) Ek 3 10](#_Toc137929431)

[11) Ek 4 11](#_Toc137929432)

[11.a Kümeleme Algoritmaları Performans Ölçütleri 11](#_Toc137929433)

[11.b Derin Öğrenmede Transformer Model 11](#_Toc137929434)

[12) Kaynakça 12](#_Toc137929435)

[13) Öz değerlendirme Tablosu 12](#_Toc137929436)

# Problemin Tanımı

German Traffic Signs Recognition trafik işaretlerinin görüntülerinden verilen bir görüntünün hangi trafik işaretine karşılık geldiğini doğru bir şekilde tahmin etmeyi amaçlar.

GTSRB, Almanya'da kullanılan trafik işaretlerini içeren geniş bir veri setine sahiptir. Veri seti, farklı trafik işaretlerinin çeşitli varyasyonlarını içerir, örneğin hız sınırları, duraklama işaretleri, yol işaretleri ve daha fazlası. Bu veri seti, gerçek dünya koşullarında karşılaşılan trafik işaretlerini temsil etmek için çeşitli yol ve hava koşullarında çekilmiş görüntüler içerir.

GTSRB'deki problem, veri setindeki görüntülerden özelliklerin çıkarılması, bu özelliklerin bir derin öğrenme modeline verilmesi ve modelin verilen görüntüye en iyi şekilde karşılık gelen trafik işaretini tahmin etmesidir. Bu, çok sınıflı bir sınıflandırma problemidir, çünkü farklı trafik işaretlerinin birçok sınıfı vardır ve modelin doğru sınıfı tahmin etmesi gerekmektedir.

# Araştırma (Ön çalışma)

<https://www.youtube.com/watch?v=Y1qxI-Df4Lk> 🡪 Convolutional Neural Networks | CNN

Convolutional Neural Network (CNN), derin öğrenme alanında yaygın olarak kullanılan bir yapay sinir ağı türüdür. Özellikle görüntü ve video gibi 2B veya 3B veri setlerinin analizi ve işlenmesi için etkili bir şekilde kullanılır. CNN'ler, geleneksel yapay sinir ağlarına kıyasla özellik çıkarımı ve sınıflandırma görevlerinde daha başarılı sonuçlar elde etmek için tasarlanmıştır.

Feature Scaling kısmında görselin önemli tüm özellikleri çıkartılır.

Kernel = Filter = Feature Detector : Kernel, bir CNN'in konvolüsyon katmanında kullanılan bir matris veya filtre olarak adlandırılır. Konvolüsyon işlemi, bir girdi veri üzerinde kernel'in her bir öğesiyle eleman bazında çarpma yapılması ve sonuçların toplanmasıyla gerçekleştirilir. Videoda bu matematiksel hesaplama 5x5 lik bir input üzerinde 3x3 lük bir kernel ile gösterilmiştir. Bu işlem, girdi verinin belirli bir bölgesini analiz etmeye yarar. Her filtre için uygulanan konvolüsyon işlemi sonucunda bir adet feature map(özellik haritası) oluşur. Oluşacak olan feature map’in boyutu O = |input – kernel| + 1 formülü ile bulunabilir. Örneğin bu videoda kullanılan örnekte O = |5-3|+1 = 3x3 lük bir feature map oluşur. Bu formül birer adım kaydırmalar için geçerlidir. Kernel, özellik çıkarımında önemli bir rol oynar. Örneğin, görüntü işlemede, bir kenar tespiti yapmak için kenar tespitme filtresi veya bir bulanıklık etkisi yaratmak için bulanıklık filtresi gibi farklı kernel türleri kullanılabilir.

Birden fazla filtre kullanıldığında, her bir filtre için bir feature map oluşur. Her feature map, farklı özellikleri temsil eder ve girdi verinin farklı yönlerini, desenlerini veya özelliklerini yakalar. Bu sayede, CNN, çok çeşitli özellikleri tanımlayabilir ve öğrenebilir.

Stride (adım büyüklüğü), konvolüsyon işlemi sırasında filtre veya kernel'in girdi veri üzerinde kaydırıldığı mesafeyi ifade eder. Stride değeri, her bir kaydırma adımının kaç piksel olduğunu belirler. Videoda stride değeri 2 olarak belirlenmiştir ve bu durumda feature map 2x2 olarak oluşur. O = (i-k)/2 + 1

Padding, CNN'de özellikle kenar piksellerinin özelliklerini korumak ve çıktı verinin boyutunu kontrol etmek için kullanılır. (Border Problem Solver)

Pooling (havuzlama), Convolutional Neural Network (CNN) mimarisindeki bir katmandır. Havuzlama katmanı, özellik haritasındaki özellikleri özetlemek ve boyutunu azaltmak amacıyla kullanılır. Havuzlama işlemi, girdi verinin belirli bir bölgesini (genellikle küçük bir pencere veya filtre) alır ve bu bölgenin bir özetini çıkarır. Bu özet, genellikle maksimum değer (max pooling) veya ortalama değer (average pooling) olarak belirlenir. Havuzlama işlemi, girdi verinin her bir özelliğini kendi bölgesine yerleştirir ve en önemli özelliği veya ortalama değeri alarak çıktı veriyi oluşturur. Videoda 2x2 max pooling ve average pooling uygulanmıştır. 26x26 görsel pooling ile 13x13’e düşürülmüştür.

Flatten, Convolutional Neural Network (CNN) mimarisindeki bir işlem veya katmandır. Flatten işlemi, veri yapısını düzleştirerek 1D vektör haline getirir. Convolutional ve pooling katmanlarından oluşan bir CNN, sonuç olarak 3D veya 2D özellik haritaları üretir. Ancak, genellikle sonraki tamamen bağlı (fully connected) katmanlara giriş olarak düzleştirilmiş veri beklenir. Bu noktada, Flatten işlemi devreye girer. Flatten işlemi, özellik haritalarını düzleştirerek tek boyutlu bir vektör elde etmek için kullanılır.

<https://www.youtube.com/watch?v=rxSmwM7z0_4> 🡪 Fully Connected Layer in CNN

Fully Connected Layer, Convolutional layer ve pooling layer’da görselin özelliklerini çıkarttıktan sonra görseli belirli kategorilere sınıflandırdığımız katmandır. Flatten array, fully connected layer’a girdi olarak verilir. Fully connected(tamamen bağlı) denmesinin sebebi bir katmandaki nöronların kendisinden önceki ve sonraki katmanlarda bulunan tüm nöronlara bağlı olmasından kaynaklanır. Nöronlar arasındaki bağlantılar ağırlık olarak adlandırılır. En son katmanda sınıflandıracağımız kategori sayısı kadar nöron bulunur. Aktivasyon fonksiyonları, tamamen bağlı katmanın çıktılarını sınırlar ve non-lineerlik ekler. Binary sınıflandırma gerçekleştiriyorsak sigmoid, çok sınıflı bir sınıflandırma gerçekleştiriyorsak softmax aktivasyon fonksiyonları kullanılabilir.

<https://www.youtube.com/watch?v=VF4BDE7uqY0> 🡪 CNN architecture

CNN mimarisini özetleyen güzel bir video.

<https://www.youtube.com/watch?v=J1jhfAw5Uvo>

Keras’taki Conv2D, MaxPooling2D ve Dense sırasıyla convolutional layer, pooling layer ve fully connected layer’a karşılık gelir. Sequential model, çeşitli katmanları ardışık bir şekilde ekleyerek bir sinir ağı oluşturmayı sağlar. Bu modeli kullanarak CNN mimarisindeki katmanları ardışık olarak ekleyebiliriz.

<https://keras.io/api/losses/>

Keras’ta tanımlı olan loss fonksiyonları ve bunları nasıl kullanabileceğimizi anlatıyor.

<https://keras.io/api/optimizers/>

Keras’ta tanımlı optimizer’ları anlatıyor

# Kullanılan Ortam, Yöntem ve Kütüphaneler

Ortam : Jupyter Notebook, Programlama dili : Python, Kütüphaneler :

* Pandas
* Numpy
* Scikit-learn
* OpenCV
* Matplotlib
* Tensorflow
* Keras
* PIL
* Neural Plot
* OS
* Random
* Scikit-image

# Önerilen Yöntem

**CNN Modelİ:**

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'))

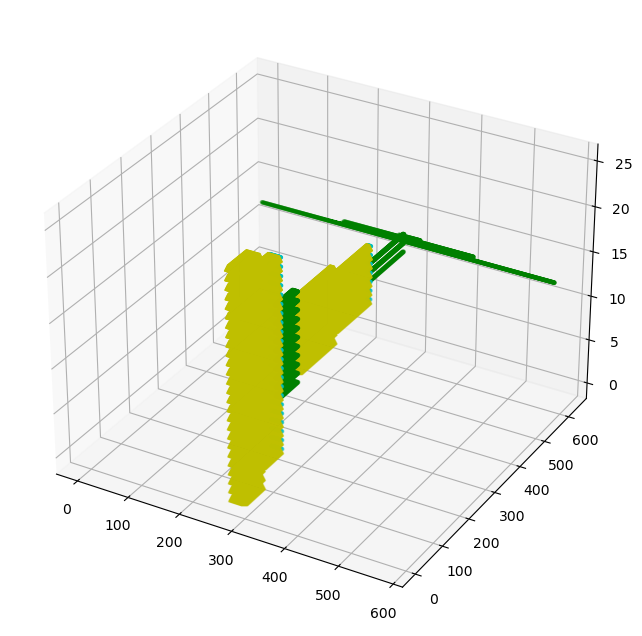
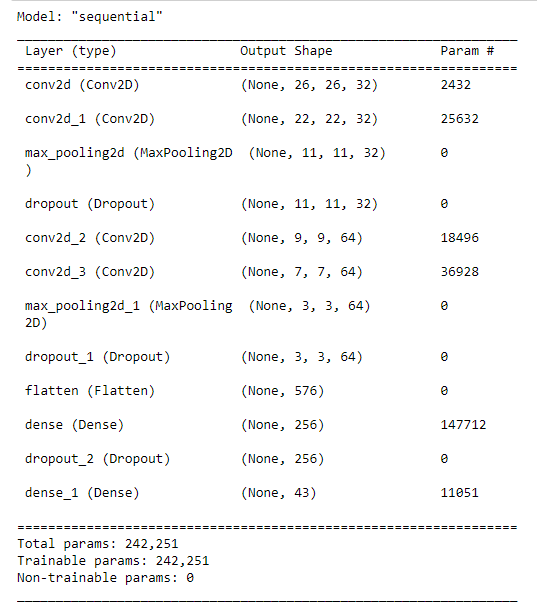
model.add(Dropout(rate=0.5))

# 43 sınıfımız olduğu için son katman bu sınıf sayısıyla eşleşmeli:

model.add(Dense(43, activation='softmax'))

**Açıklaması:**

1. İlk katman, 32 filtreli ve 5x5 boyutunda bir evrişim katmanıdır. ReLU aktivasyon fonksiyonu kullanılarak çıktıları hesaplanır. Giriş şekli, eğitim veri setinin şekline uygun olarak belirlenir.
2. İkinci katman, 32 filtreli ve 5x5 boyutunda bir evrişim katmanıdır. Yine ReLU aktivasyon fonksiyonu kullanılarak çıktıları hesaplanır.
3. MaxPooling katmanı, 2x2 boyutunda bir maksimum havuzlama işlemi uygular. Bu, giriş verilerinin boyutunu küçültmeye yardımcı olur.
4. Dropout katmanı, %25 oranında giriş birimlerini rastgele sıfıra eşitleyerek aşırı uyumu engeller.
5. Daha karmaşık özelliklerin çıkarılması için 64 filtreli ve 3x3 boyutunda bir evrişim katmanı eklenir. Yine ReLU aktivasyon fonksiyonu kullanılır.
6. Yine aynı özellikleri daha da güçlendirmek için ikinci bir 64 filtreli ve 3x3 boyutunda evrişim katmanı eklenir.
7. MaxPooling katmanı, boyutları daha da küçültmek için 2x2 boyutunda bir maksimum havuzlama işlemi uygular.
8. Dropout katmanı, aşırı uyumu önlemek için giriş birimlerinin %25'ini rastgele sıfıra eşitler.
9. Flatten katmanı, evrişim katmanlarından elde edilen özellik haritalarını düzleştirerek vektör haline getirir.
10. Tam bağlantılı (Dense) katman, 256 nöron içerir ve ReLU aktivasyon fonksiyonu kullanır.
11. Dropout katmanı, aşırı uyumu önlemek için giriş birimlerinin %50'sini rastgele sıfıra eşitler.
12. Son olarak, çıkış sınıf sayısına (43) uygun olarak bir Dense katmanı eklenir ve softmax aktivasyon fonksiyonu kullanılır. Bu, modelin her bir sınıf için olasılık dağılımı üretmesini sağlar.



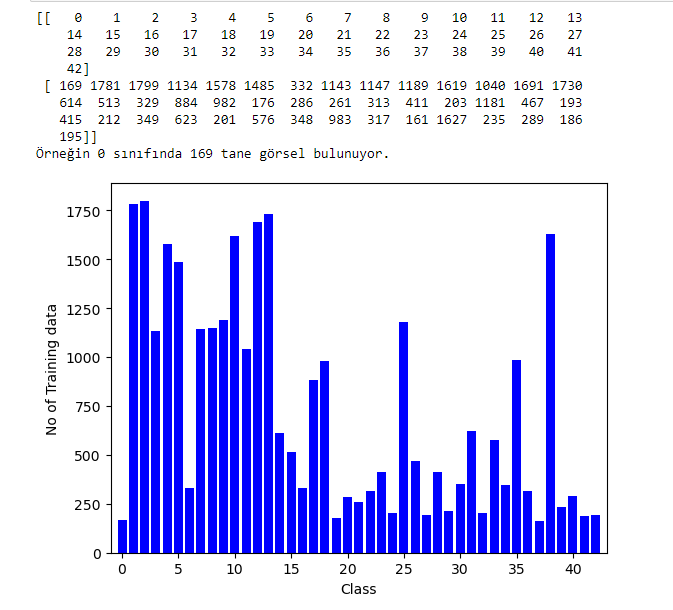
# 5) Deneysel Çalışmalar

Veri setini kaggle’dan elde ettim.

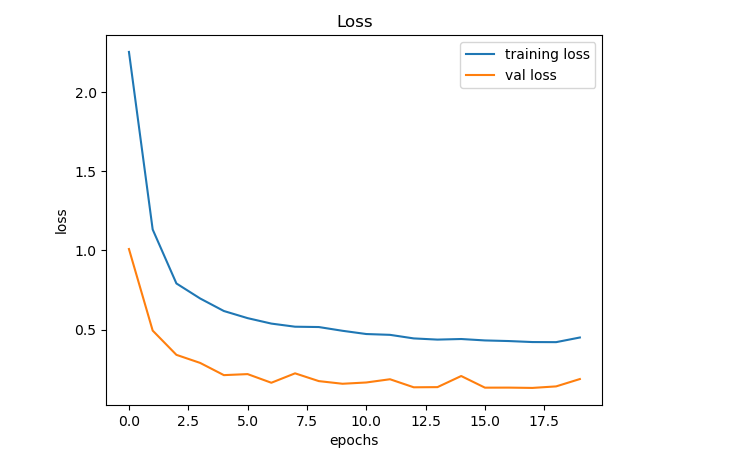
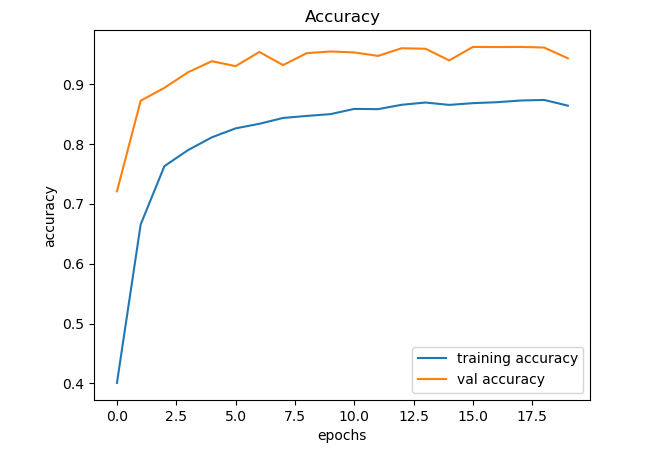
<https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign>

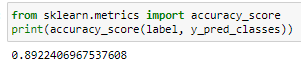
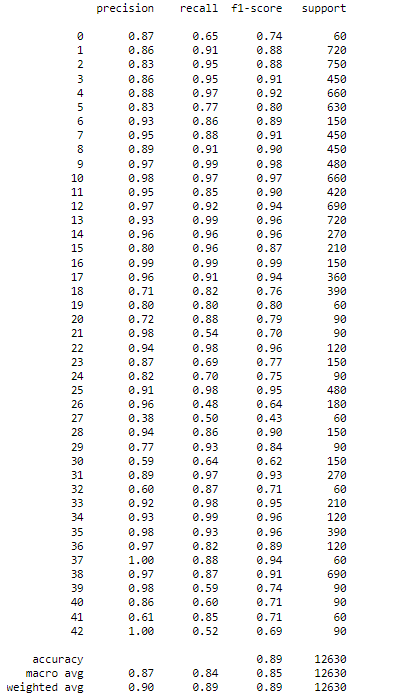
Veri seti ‘archive’ adında bir klasörde bulunuyor. Meta Test ve Train adında üç klasörden ve aynı isimli 3 csv dosyası bulunuyor. 43 farklı trafik işareti sınıfı var ve Train klasörünün içerisinde 0-1-2.. gibi alt dizinlerde herbir sınıfa ait görseller bulunuyor. 39209 adet görsel eğitim için sağlanıyor. Ben bu görsellerin %80 inini eğitim %20 sini validasyon için ayırdım. Görselleştirmeleri daha iyi gösterebilmek için Jupyter Notebook üzerinde çalıştım. (Basic Descriptive Analysis başlığı altında her sınıftan kaç görsel bulunduğunu bulabilirsiniz.) Test klasöründe ise test için 12630 adet görsel bulunuyor. Modelin oluşturulması ve eğitimi tamamlandıktan sonra test.csv dosyasından dosya yolunu alıp bu klasördeki verilerle test işlemi gerçekleştiriliyor.

Eğitim veri seti:

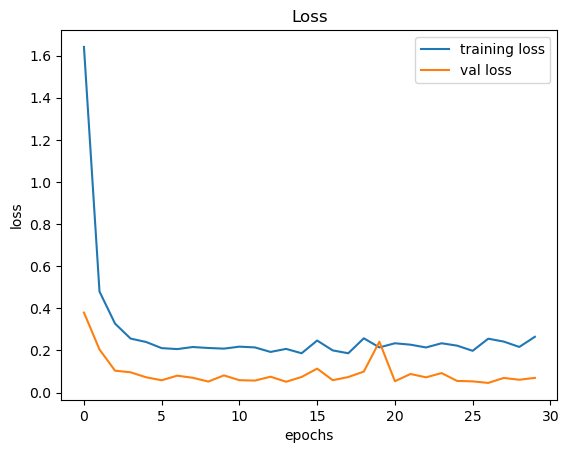
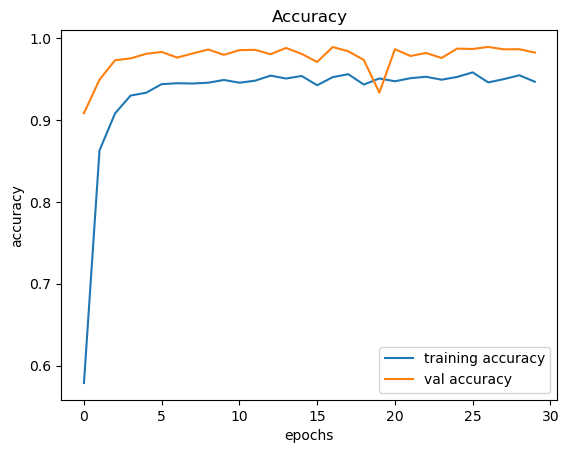


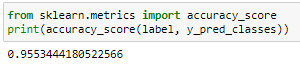
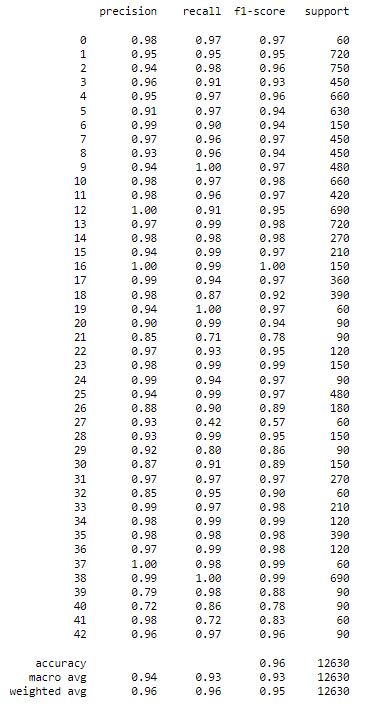
Epoch sayısı 20 iken sonuçlar:





Epoch Sayısı 30 iken sonuçlar :





**Karşılaştırma ve Yorum**

Epoch sayısını arttırıp modeli eğittiğimde test verisindeki değerlendirme metriklerinde olumlu yönde artışlar yaşandığını gözlemledim. Fakat 30 epoch ile eğitim yaparken grafikleri gözlemlediğimde eğitim verisinin doğruluk değeri yükselirken validasyon verisinin doğruluk değerinin düştüğünü gördüm. Bu overfitting(aşırı uyum) sorunun meydana gelmiş olabileceğini gösterir. Epoch sayısını arttırmak, modelin daha fazla veriye erişmesini ve daha iyi öğrenmesini sağlayabilir. Ancak, aşırı uyum (overfitting) riskini arttırabilir. Aşırı uyum, modelin eğitim verisine çok iyi uyum sağlaması ancak genelleme yaparken performansının düşmesi durumudur. Modelimde kullandığım Dropout katmanları sayesinde model bu durumu aşmış gibi görünüyor. Test verisi üzerinde modelin doğruluk değerinin %89 dan %95 e çıktığı görülüyor ancak yükleyeceğim dökümanda epoch sayısını overfitting durumundan kaçınmak için daha düşük bir değerde bırakacağım.

Model performansını daha kesin bir şekilde arttırmak için ise veri ön işleme teknikleri kullandım ve çok başarılı sonuçlar elde ettim. Sonuçları Ek 1 başlığı altında bulabilirsiniz.

# 6) Sonuç

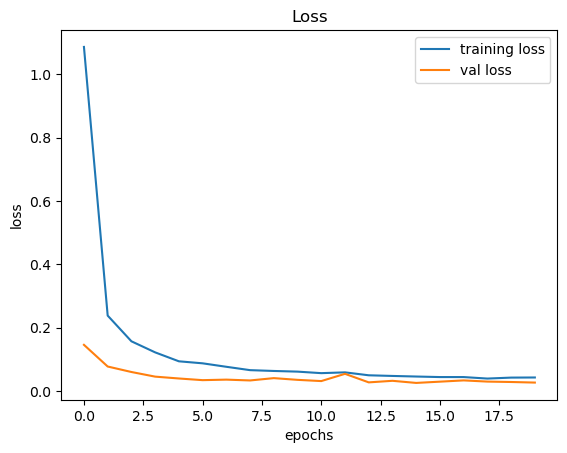
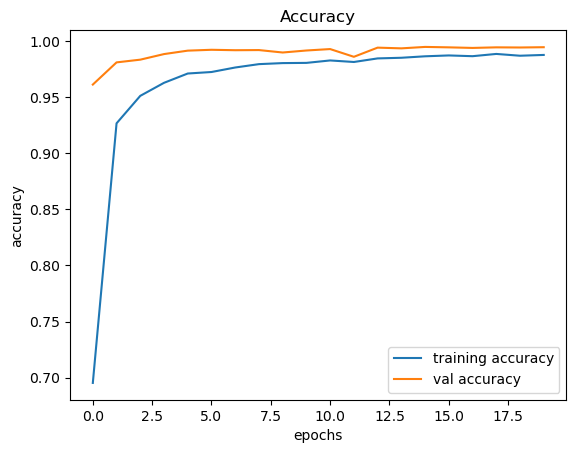
Projenin son halinde (veri ön işleme teknikleri kullanıldıktan sonra) başarılı bir sınıflandırıcı elde ettiğimi düşünüyorum. Elde ettiğim model performans değerlendirme metriklerinin bunu kanıtladığını düşünüyorum.

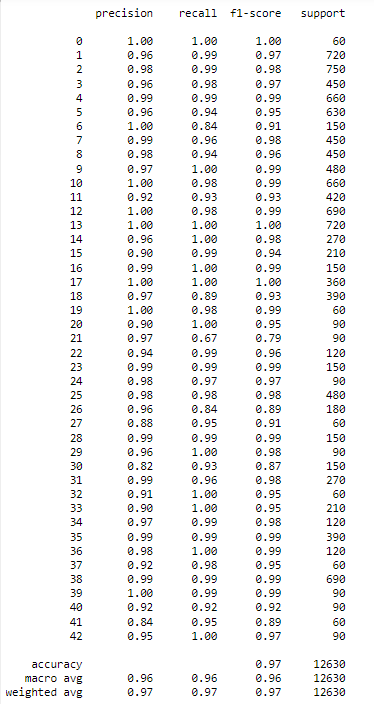
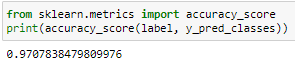
Bu projenin gerçek hayatta kullanılabileceğini ve insanlara yarar sağlayabileceğini düşünüyorum. Örneğin sürücü yardımcı sistemlerinin bir parçası olarak kullanılabilir. Otomobillerde bulunan bir kamera sistemi, trafik işaretlerini tanıyarak sürücüye hız sınırlarını, dur-kalk işaretlerini veya yaya geçitlerini bildirebilir. Bu sayede sürücülerin trafik kurallarına uyması ve güvenli bir şekilde seyahat etmesi sağlanır. Lojistik sektöründe de kullanılabilir. Örneğin, bir kamyonun trafik işaretlerini tanıyarak rotasını belirlemesi ve trafik kurallarına uygun şekilde seyahat etmesi sağlanabilir. Bu, lojistik süreçlerin daha verimli hale gelmesine ve teslimat sürelerinin iyileştirilmesine yardımcı olabilir.

Bu projeyi geliştirirken çok şey öğrendiğimi düşünüyorum. CNN mimarisini derinlemesine öğrenmiş oldum ve ilk defa bir modelini geliştirdim. İlk defa bilgisayarlı görü alanında bir proje gerçekleştirdim ve görüntülerle çalıştım. Projeyi geliştirirken Python’ın çeşitli ve önemli kütüphanelerine hakimiyet kazandım. İlk defa Tensorflow, Keras ve OpenCV gibi kütüphaneleri kullandım. Önceki projelerimize kıyasla büyük ve daha karmaşık bir veri setiyle çalışmayı öğrendim. Projeyi geliştirirken birçok yeni kavram ve konsept ile karşılaştım ve araştırmalar yaptım. Bazı veri ön işleme tekniklerini öğrendim ve bunun model performansı ve başarısı açısından ne kadar önemli olduğunu somut bir şekilde görmüş oldum.

# 8) Ek 1

Başarımı arttırmak için Shuffling (Karıştırma) , Gray Scaling (Gri Tonlama), Local Histogram Equalization (Yerel Histogram Eşitleme) ve Normalization(Normalleştirme) ön işleme tekniklerini kullandım.



Bu yöntemleri uyguladıktan sonra 20 epoch süren eğitimden sonra test verileri için %89 olan doğruluk değeri %97’ye ulaştı ve precision, recall ve f1-score değerlerinde tabloda görüldüğü üzere kayda değer başarı artışı sağlandı.

# 9) Ek 2

Bu projeyi geliştirirken anladığımı kanıtlamak adına pek çok yorum ekledim. Ayrıca Jupyter Notebook kullandığım için koda anlatımlar da ekleyebildim. Açıklamalarımı ve anlatımlarımı Türkçe dilinde ve açıklayıcı olduğunu düşündüğüm bir şekilde yazdım. Sadece literatürden uzaklaşmamak için başlıkları ve değişkenleri İngilizce dilinde yazdım. Ne yazık ki Türkçe kaynak bulmak kolay değil. Bu konuyla ilgili Türkçe bir anlatım isteyen birinin bu projeyi inceleyip konu hakkında birçok şey öğrenebileceğini ve projeyi anlayabileceğini düşünüyorum. Bu şekilde küçük de olsa literatüre bir katkım olduğunu düşünüyorum.

# 10) Ek 3

Kaynakçada bulunan son 4 hazır bağlantıdan yararlandım. Farklı projelerden kodları bir araya getirebilmek için konuya hakim olmam gerekiyordu ve parçaların birbirleriyle uyum içerisinde çalışmasını sağladım. Farklı olarak: modelin görselleştirilmesi için kütüphane araştırdım ve ModelPlot kütüphanesini kullandım. Sınıflandırma raporunu ve hata matrisini gösterdim. Epoch sayılarını değiştirerek deneyler yaptım. İşlem sıralamasını kendi mantığıma daha uygun olacak bir şekle getirdim ve değiştirdim. Aradaki farkı daha iyi görebilmek için aynı modeli bahsettiğim veri işleme adımlarına tabii tutmadan ve tutarak eğitip sonuçları karşılaştırdım. Yorumlar ve anlatımlar ekledim.

# 11) Ek 4

## 11.a Kümeleme Algoritmaları Performans Ölçütleri

Denetimsiz öğrenmede etiketlerin bilinmemesi model performansının değerlendirilmesini zorlaştırır fakat iyi bir kümeleme algoritması veri noktalarını birbirine benzerliklerine göre gruplayarak küçük küme içi varyansa ve büyük kümeler arası varyansa sahip olması ile tutarlı bir şekilde değerlendirilebilir.

**Silhouette Coefficient (Silhouette Katsayısı)** **:** Veri noktasının kendi kümesinde diğer veri noktalarıyla olan ortalama uzaklığı ile veri noktasının en yakın komşu kümedeki veri noktalarıyla olan ortalama uzaklığını karşılaştırarak kümeleme algoritmasının sonuçlarını değerlendirmek için kullanılan bir yöntemdir.

Silhouette katsayı 1 ile -1 arasında değer alır. Bir veri noktasının Silhouette Katsayısı 1'e yakınsa, bu veri noktası kendi kümesindeki noktalara daha benzerdir ve komşu kümelere göre daha farklıdır. Yani, iyi bir şekilde tanımlanmış bir kümeleme sonucunu temsil eder. Bir veri noktasının Silhouette Katsayısı -1'e yakınsa, bu veri noktası kendi kümesindeki diğer noktalara göre daha farklıdır ve komşu kümelere daha benzerdir. Bu durumda, veri noktası yanlış bir şekilde kümelenmiş olabilir.

**Davies-Bouldin Indexi** : Bu yöntemde her bir kümenin içindeki veri noktalarının, kendi küme merkezinden olan uzaklıklarının ortalaması alınır. Bu, küme içindeki veri noktalarının yayılımını temsil eder. Her bir küme merkeziyle diğer kümelerin merkezi arasındaki uzaklığın ortalaması alınır. Bu, kümeler arasındaki benzerliği ölçer. Bu metriklerden yararlanılarak her bir kümenin Davies-Bouldin değeri hesaplanır. İdeal olarak, düşük bir Davies-Bouldin İndeksi değeri, iyi tanımlanmış ve birbirinden ayrışmış kümeleri temsil eder. Davies-Bouldin İndeksinin hesaplanması, Silhouette puanlarına göre daha kolaydır ve nokta bazında mesafeler kullanır. Ancak, yalnızca Davies-Bouldin İndeksi'ne dayanarak kesin bir yargıya varmak için dikkatli bir değerlendirme yapılması gerekir.

## 11.b Derin Öğrenmede Transformer Model

Transformer modeli, doğal dil işleme (NLP) ve makine çevirisi gibi görevler için geliştirilmiş bir derin öğrenme mimarisidir. Transformer, özellikle dil modellerinde büyük bir etkiye sahip olan "Attention" mekanizmasını kullanarak dil işleme problemlerini çözmek için tasarlanmıştır. Transformer modeli, geleneksel rekürsif veya evrişimli yapılar yerine tamamen dikkat odaklı bir yaklaşım kullanır. Dikkat (Attention), Transformer modelinin temel bir bileşenidir ve modelin giriş verileri arasındaki ilişkileri belirlemek için kullanılır. Dikkat mekanizması, modelin belirli bir kelimenin diğer kelimelerle olan ilişkisini hesaplamasını sağlayan bir ağırlık mekanizmasıdırTransformer, birçok önemli bileşenden oluşur: Encoder(Kodlayıcı), Decoder(Çözücü), Attention(Dikkat)

Her kodlayıcı katmanının işlevi, girdilerin hangi bölümlerinin birbiriyle alakalı olduğu hakkında bilgi içeren kodlamalar üretmektir. Kodlamalarını bir sonraki kodlayıcı katmanına girdi olarak iletir. Her çözücü katmanı, tüm kodlamaları alarak ve bunların birleştirilmiş bağlamsal bilgilerini bir çıktı dizisi oluşturmak için kullanarak tersini yapar. Bunu başarmak için, her kodlayıcı ve çözücü katmanı bir dikkat mekanizmasından yararlanır.

Transformerlar 2017 yılında Google Brain ekibi tarafından tanıtıldı ve giderek doğal dil işleme (NLP) problemleri için tercih edilen model haline geliyor. Uzun kısa süreli hafıza (LSTM) gibi RNN modellerinin yerini alıyor. Transformerlar, RNN modellerine göre daha paralelizasyona uygun olup, daha büyük veri kümeleri üzerinde eğitim yapmayı mümkün kılar. Bu durum, GPT (Generative Pre-Trained Transformer) gibi önceden eğitilmiş sistemlerin geliştirilmesine yol açtı.

# 12) Kaynakça

<https://towardsdatascience.com/7-evaluation-metrics-for-clustering-algorithms-bdc537ff54d2>

<https://www.youtube.com/watch?v=Y1qxI-Df4Lk>

<https://www.youtube.com/watch?v=rxSmwM7z0_4>

<https://www.youtube.com/watch?v=VF4BDE7uqY0>

<https://www.youtube.com/watch?v=J1jhfAw5Uvo>

<https://keras.io/api/losses/>

<https://keras.io/api/optimizers/>

<https://www.youtube.com/watch?v=qahpZkPlTRM>

https://en.wikipedia.org/wiki/Transformer\_(machine\_learning\_model)

<https://towardsdatascience.com/my-tryst-with-deep-learning-german-traffic-data-set-with-keras-87970dfb18b7>

<https://github.com/maxritter/SDC-Traffic-Sign-Recognition/tree/master>

<https://github.com/mohamedameen93/German-Traffic-Sign-Classification-Using-TensorFlow/blob/master/Traffic_Sign_Classifier.ipynb>

https://github.com/Spidy20/Traffic\_Signs\_WebApp

# 13) Öz değerlendirme Tablosu

**Açıklama kısmında yapıldı, yapılmadı bilgisi ve hangi maddelerin nasıl yapıldığı veya neden yapılamadığı kısaca yazılmalıdır.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **İstenen Madde** | **Var** | **Açıklama** | **Tahmini Not** |
| **1** | **Kapak Sayfası, Problemin Tanımı, Kullanılan Ortam, Yöntem ve Kütüphaneler, Araştırma (10)** |  | Kapak sayfası yapıldı, problem açıklayıcı bir şekilde tanıtıldı, kullanılan ortam ve kütüphane bilgisi verildi. Arkaplandaki çalışma şeklini ve mantığı anlamak ve yorumlayabilmek için yeterli araştırmalar ve ön çalışmalar yapıldı. Gerçekleştirim için videolar izlendi ve teorik anlatım ile bağlantılar kuruldu. | 10 |
| **2** | **Önerilen Yöntem (10)** |  | Kullanılan CNN modelinin kodu, açıklaması, özeti ve 3 boyutlu bir çizimi verildi. | 10 |
| **3** | **Deneysel Çalışmalar (10)** |  | Veri setinin nasıl elde edildiği, dosya düzeni, eğitim ve test verileri sayısı ve her sınıfta kaç görsel bulunduğu bir sütun grafiği ile verildi. Epoch sayısı değiştirilerek deneyler yapıldı ve grafikler yardımıyla anlatıldı. Karşılaştırma ve yorumlar eklendi. | 10 |
| **4** | **Proje Rapor Biçimi, Organizasyonu, Boyutu, Kalitesi, Kaynakça ve atıflar (10)** |  | Önceki projelerimizdeki template kullanıldı, başlıklar ve içindekiler tablosu oluşturularak istenilenler organize edildi. Kaynakça eklendi. Belirtilen boyut aralığında oluşturuldu. | 10 |
| **5** | **Sonuç (10)** |  | Projenin başarısı değerlendirildi, gerçek hayatta nerelerde kullanılabileceğinin bilgisi verildi ve projeyi geliştirirken öğrendiklerim kendi cümlelerim ile aktarıldı. | 10 |
| **6** | **Ek 1: Başarım İyileştirme (10)** |  | Ön işleme teknikleri belirtildi. Sonuçlar grafiklerle gösterildi ve başarıda kayda değer iyileşmeler sağlandı. | 10 |
| **7** | **Ek 2 (10)** |  | Projenin literature katkısı yorumlandı. | 10 |
| **8** | **Ek 3 (10)** |  | Kullanılan kaynaklardan farklılıklar belirtildi. | 10 |
| **9** | **Ek 4 (10): Her madde 5’er puan.** |  | İlk madde için Silhouette katsayısı ve Davies-Bouldin Indexi anlatıldı. İkinci madde için Transformer Model anlatıldı ve günümüzdeki yerine atıfta bulunuldu. | 10 |
| **10** | **Özdeğerlendirme Tablosu (10)** |  | Detaylıca dolduruldu. | 10 |
| **100 üzerinden Toplam Not:** | | | | 100 |