Derin Öğrenme Teknikleri Kullanılarak Braille Alfabesi Okuma

Muhammet Zahit Aydın
190201039

Fatih Güner
190201074

Kerem Karataş
190201076

Murat Karakurt
190201052

Ahmet Burhan Bulut
190201006

***Abstract*— Proje seçimi olarak Derin öğrenme teknikleri kullanılarak Körler alfabesi(Braille) okuma konusunu seçtik. Bu konuyu seçmemizin sebebi nesne tespitinin ilgi çekici olması ve YOLOv8 kullanmak istememizdir. verisetimiz Roboflow üzerinde bir verisettir. Bu verisetimizde resimler eğitim verisi, validasyon verisi ve test verisi diye 3'e ayrılmıştır. Yani hangi resimlerin hangi veri türünden olduğu etiketlenmiştir. %70 eğitim, %15 validasyon, %15 test verisi olarak ayrılmıştır. Resimler 640x640'lık şekle önişlemeyle yeniden boyutlandırılmıştır. Bize verilen bir körler alfabesi içeren resmin algılanmasını YOLO v8 kullanarak modeli eğittik. Nesne tespiti için kullanılan doğruluk metriği mAP50 veya mAP50-95' dir. mAP50'nin değeri 0.95 çıkmıştır. mAP50-95'in değeri daha hassas bir ölçüm olduğu için 0.75 gibi bir değerle bizi karşıladı. Epochs sayısını 25 olarak belirledik. 25 epoch'dan sonra modelin mAP50 değeri nerdeyse artış göstermedi hatta bazı düşüşler oldu. Bu da overfitting durumuna girdiğini gösteriyor. O yüzden 25'te tuttuk ve yeterli bir doğruluk elde ettik. Modelimiz yaklaşık yarım saatte eğitildi.**

Keywords— Object Detection, YOLO, Braille, Deep Learning

I. GİRİŞ

Günümüzde derin öğrenme teknikleri, çeşitli uygulamalarda büyük başarı elde etmiştir. Bunlardan biri de Braille alfabesi okuma alanında kullanılmaktadır. Braille alfabesi, görme engellilerin dokunarak okuyabileceği bir alfabedir ve bu alfabede yer alan noktaların kombinasyonu harfleri ve sayıları ifade etmektedir. Derin öğrenme teknikleri, Braille alfabesi okuma konusunda büyük bir potansiyel sunar, çünkü bu teknikler, büyük miktarda veriyi işleyebilir ve karmaşık desenleri tanıyabilir.

Bu makalede, derin öğrenme tekniklerinin Braille alfabesi okuma konusunda nasıl kullanılabileceğine dair bir inceleme yapacağız. İlk olarak, Braille alfabesi hakkında genel bir bilgi vererek başlayalım.

- Braille Alfabesi:

Braille alfabesi, görme engellilerin dokunarak okuyabileceği bir alfabedir. Bu alfabede, 6 noktadan oluşan bir matris kullanılır ve bu noktaların kombinasyonu harfleri, sayıları ve diğer sembolleri ifade eder. Braille alfabesi, Louis Braille tarafından 1824 yılında geliştirilmiştir ve hala yaygın bir şekilde kullanılmaktadır.

Braille alfabesi okumak, görme engelliler için hayati önem taşır. Ancak, Braille alfabesi okumak zorlu bir beceridir ve bu beceriyi kazanmak için yıllarca eğitim gerektirir. Bu nedenle, Braille alfabesi okuma sürecini hızlandırmak ve kolaylaştırmak için derin öğrenme teknikleri kullanılabilmektedir.

- Derin Öğrenme Teknikleri:

Derin öğrenme teknikleri, büyük miktarda veri kullanarak karmaşık desenleri tanıyan bir makine öğrenme yöntemidir. Derin öğrenme teknikleri, genellikle yapay sinir ağları olarak adlandırılan çok katmanlı ağlarda kullanılır. Bu ağlar, birçok katmandan oluşur ve her katman, bir önceki katmanın çıktısını alır ve daha karmaşık desenler tanımak için bu çıktıları işler.

Derin öğrenme teknikleri, yüksek doğruluk oranları ve diğer makine öğrenme tekniklerinden daha iyi performans gösterme eğilimindedir. Bu nedenle, Braille alfabesi okuma konusunda da kullanılabilirler.

- Braille Alfabesi Okuma için Derin Öğrenme Teknikleri:

Braille alfabesi okuma için derin öğrenme teknikleri, öncelikle büyük bir veri kümesine ihtiyaç duyarlar. Bu veri kümesi, Braille alfabesi karakterlerini içeren bir veri kümesidir. Veri kümesi, Braille alfabesi karakterlerinin farklı boyutlarda, farklı yazı tiplerinde, farklı pozisyonlarda ve farklı ışık koşullarında çekilmiş resimlerini içermelidir. Bu veri kümesi, derin öğrenme modelinin Braille alfabesi karakterlerinin çeşitli varyasyonlarını tanımasına yardımcı olur.

Veri kümesi hazırlandıktan sonra, bir yapay sinir ağı tasarlanır. Bu ağ, veri kümesindeki resimlerin girişlerini alacak ve Braille alfabesi karakterlerinin sınıflandırmasını yapacak. Yapay sinir ağı, birçok katman ve birçok nöron içerir. Bu nöronlar, verileri işler ve sonuç olarak, hangi Braille alfabesi karakteri olduğunu belirler.

Daha sonra, eğitim süreci başlar. Yapay sinir ağı, veri kümesindeki resimleri kullanarak öğrenir. Her bir öğrenme adımında, ağın çıktısı gerçek etiketlerle karşılaştırılır ve ağın hatalarını belirleyen bir kayıp fonksiyonu kullanılır. Bu

hatalar, ağın geriye doğru yayılımıyla düzeltilir ve ağ, daha doğru sonuçlar üretmek için yeniden eğitilir.

Projeyi Python kullanarak Google Colab(Google Colaboratory) üzerinden yaptık. Google Colab bize yapay zeka ve derin öğrenme projeleri için üzerinde kolayca ortak bir biçimde çalışabileceğimiz, bulut tabanlı, notebook çalışma alanlarını kullanmaya dayalı bir programlama ortamı sunar. Jupyter Notebook'tan farkı ondan daha fazla kullanışlı özelliklerinin olması ve bulut sunucularında barındırılmasıdır.

Aslında bize bir docker image veriyor, yani bize bir bilgisayar tahsis ediliyor gibi düşünebiliriz. Ram sıkıntısı, gpu sıkıntısı olmaması için biz geliştiricilere harika bir ortam sunuyor. Bunun dışında tpu(tensor processing unit) desteği de sunabiliyor. Ayrıca gerekli olan birçok kütüphane de kuruludur. Google colab ile Roboflow birbirine bağlanabilen yapılar. Verisetimizin bulunduğu Roboflow üzerinden Google Colab notebook'u oluşturduk. Roboflow görüntülerde ve videolarda nesneleri görebilen yazılımlara güç veren bir platformdur. Roboflow, veri yüklemeyi, hazırlamayı, etiketlemeyi eğitimi ve dağıtımı kolaylaştırarak bilgisayar görüş modellerini oluşturmamıza yardmcı olur. 100 milyon etiketli ve açıklamalı görüntüden oluşan 100.000'den fazla veri seti Roboflow üzerinde yönetilmektedir (1). Şimdi ise YOLO'yu tanıtalım. Açık kaynaktır. YOLO CNN kullanarak nesne tespiti yapan bir algoritmadır. "You Only Look Once" demektir. Bunun sebebi ise algoritmanın nesne tespitini oldukça hızlı bir şekilde ve tek seferde yapabiliyor olmasıdır.

YOLO aloritmasının diğer algoritmalaradan daa hızlı olmasının sebebi resmin tamamını tek seferde nöral bir ağdan geçiriyor olmasıdır. YOLO algoritması görünütler üzerinde tespit ettiği nesnelerin çevresini bounding box(sınırlayıcı kutu) ile çevreler. YOLO kendisine giridi olarak verilen görüntüyü NXN'lik ızgaralara böler. Bu ızgaralar 5x5, 9x9, 17x17... olabilir. Her ızgara kendi içerisinde nesne olup olmadığını ve nesne var olduğunu düşünüyorsa merkez noktasının kendi alanında olup olmadığını düşünür. Nesnenin mrekez noktasına sahip olduğuna karar verenızgara o nesnenin sınıfını, yüksekliğini ve genişliğini bulup o nesnenin çevresine sınırlayıcı kutu çizmeldir. Bİrden fazla ızgara, nesnenin kendi içerisinde olduğunu düşünebilir. Bu durumda ekranda gereksiz sınırlayıcı kutular oluşur. Bütün sınırlayıcı kutuların güven skoru vardır. Bu durumu engellemek için Non-Maximum Supression algoritması kullanılır, bu algoritma görüntü üzerinde tespit edilen nesneler için çizilen sınırlayıcı kutulardan güven skor değeri en yüksek olanı çizer (2). Birden çok versiyonu vardır. Bizim kullandığımız versiyonu ise YOLOv8'dir.

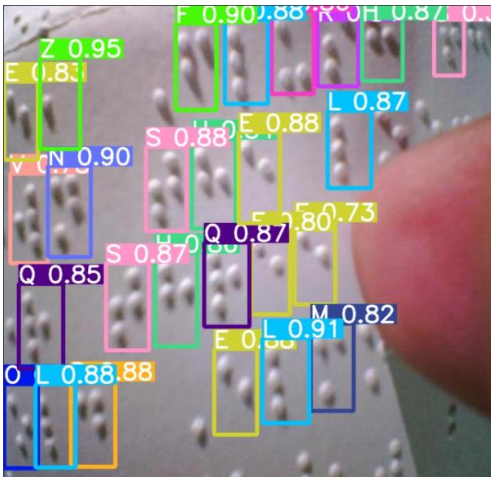
YOLOv8 Ultralytics ile birlikte anılır. Bu şirketin bir ürünüdür. Bu isim de bir paket vardır ve bu paketten YOLO'yu import ederiz. Colob'da eğitim ve doğrulama yaparken yolo'ya parametreler veriyoruz. Eğitim yaparken task=detect dediğimiz parametre nesne tespiti yapacağını, mode=train eğitim modunda olduğunu model=yolov8s.pt bu modeli kullanıp kaydedeceğini, data=location/data.yml burada veri ile ilgili confirgasyonu, epoch=25 bu akdar epoch sayısı olduğunu, imgsz=800 resimleri buradaki değere boyutlandırdığını gösterir.

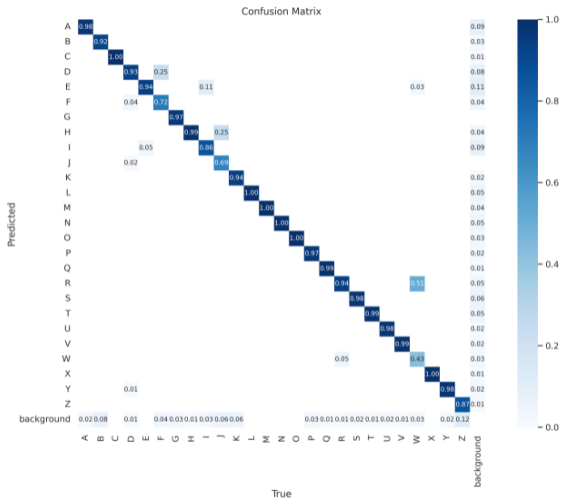
Validasyon için de benzer parametreler vardır. Google Colab haricinde kendi localimize de indirdik. VsCode üzerinde Orada eğittimiz modeli hemen load edip denemeler yapabiliyoruz, bu internet bağlantısı istemiyor ya da api key üzerinden roboflowa deploy ettiğimiz modele ulaşıp denemeler yapabiliyoruz.

II. MODEL PERFORMANSI

Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
all	209	3381	0.886	0.914	0.951	0.714
A	209	233	0.887	0.927	0.94	0.687
B	209	53	0.867	0.858	0.932	0.635
C	209	73	0.911	0.987	0.976	0.76
D	209	194	0.87	0.948	0.971	0.726
E	209	425	0.923	0.948	0.978	0.746
F	209	53	0.815	0.665	0.892	0.682
G	209	59	0.991	0.949	0.993	0.745
H	209	184	0.914	0.973	0.978	0.752
I	209	143	0.82	0.867	0.935	0.709
J	209	16	0.859	0.759	0.902	0.659
K	209	54	0.953	0.926	0.972	0.721
L	209	215	0.917	0.927	0.982	0.735
M	209	59	0.712	0.983	0.892	0.685
N	209	150	0.891	0.98	0.982	0.742
O	209	265	0.943	0.996	0.981	0.747
P	209	58	0.872	0.948	0.943	0.73
Q	209	155	0.975	0.991	0.992	0.743
R	209	173	0.888	0.965	0.969	0.723
S	209	342	0.921	0.953	0.978	0.727
T	209	162	0.914	0.951	0.957	0.736
U	209	50	0.872	0.9	0.958	0.72
V	209	106	0.904	0.982	0.977	0.715
W	209	35	0.827	0.682	0.841	0.639
X	209	39	0.937	1	0.984	0.719
Y	209	61	0.834	0.825	0.918	0.69
Z	209	24	0.817	0.875	0.906	0.702

III. DENEYSEL SONUÇLAR





IV. SONUÇ

Derin öğrenme teknikleri, Braille alfabesi okuma konusunda büyük bir potansiyel sunar. Bu teknikler, verileri işleyerek karmaşık desenleri tanıyabilir ve Braille alfabesi karakterlerini sınıflandırabilir. Bu sayede, görme engellilerin Braille alfabesi okuma sürecini hızlandırabilir ve kolaylaştırabilir. Ancak, daha fazla araştırma ve geliştirme gereklidir, özellikle farklı ışık koşullarında veya farklı pozisyonlarda çekilen resimlerin tanınması gibi zorlukların üstesinden gelmek için.

KAYNAKÇA

[1] https://roboflow.com/

[2] https://smartera.com.tr/gercek-zamanli-nesne-takibireal-time-object-detection-w-yolo-python/

[3] https://universe.roboflow.com/braille-lq5eh/braille-detection

[4] https://docs.ultralytics.com/

[5] https://en.wikipedia.org/wiki/Object_detection

[6] https://www.v7labs.com/blog/yolo-object-detection