TRAKYA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİLGİSAYAR PROJESİ (BAHAR DÖNEMİ)

İŞARET DİLİ ÇEVİRMENİ

Proje Yöneticisi: Rembiye KANDEMİR

Proje Grubu

Samet TOPRAK

Caner AKIN

Edirne, 2023

İÇİNDEKİLER **Sayfa**

İÇİNDEKİLER **Sayfa**

KISALTMA LİSTESİ

VT	Veri tabanı
GB	Gigabyte
.txt	Metin belgesi
LSTM	Long-short-term memory
AI	Artifical Intelligence

ŞEKİL LİSTESİ **Sayfa**

ŞEKİL LİSTESİ **Sayfa**

ÖNSÖZ

Bu proje çalışmasında Türkiye'de yaşayan konuşma ve duyma engelli bireylerin yaşam şartlarını, sosyal yaşantılarını olumlu yönde etkilemek ve duygu, düşüncelerini daha kolay ve hızlı şekilde karşısındaki kişiye aktarmasını sağlamasını adına yapılmış çalışmadır.

Öncelikle proje konusunu seçerken ilgi alanlarımızı göz önünde bulundurup, kaynak bulmamızda destek sağlayan, çeşitli proje örnekleriyle bize yol gösteren proje danışmanımız Dr. Öğr. Üyesi Rembiye KANDEMİR'e teşekkürlerimizi borç biliriz.

ÖZET

Sosyal hayatımız için iletişim büyük bir önem taşır. Duymakta zorluk çeken insanlar için ise iletişim işaret dili vasıtasıyla gerçekleştirir. Bu durumda olan bireyler ile iletişim kurabilmemiz için, özellikle doktor muayene ya da iş görüşme gibi zor ve önemli konular üzerinde anlaşabilmemiz için işaret dili tercümanlarına ihtiyaç duyulur.

Bu durumu en kolay hale getirebilmek için alternatif olarak bir tercüme uygulaması yapma kararı alınmıştır. Uygulama; kullanıcının kamera karşısına geçerek kamera yardımı ile işaret dilinde yaptığı konuşmayı ya da kelimeleri yapay zekâ yardımı ile Türkçe dilinde seslendirmeyi amaçlamış bir projedir.

ABSTRACT

Communication is so important for for our social life. For people who have difficulty hearing, communication takes place through sign language. In order for us to communicate with individuals in this situation, we need sign language interpreters to be able to agree on difficult and important issues like a doctor’s examination or a job interwiev.

In order to maket his situation the easiest, we decided to make a translation application is a project that aims to make the user speak speak in sign language with the help of the camera by standing in front of the camera or to vocalize the words in Turkish language with the help of artificial intelligence.

1. GİRİŞ

Türkiye'de 89.043 kişi işitme engelli, 55.480 kişi de konuşma engellidir. Yaklaşık olarak 320.000 kişi işaret dilini aktif olarak kullanıyor. Fakat yüksek nüfusa karşın işaret dilinin kullanılabileceği ortamlar çok kısıtlı. İşaret dilini kullanan insanların iletişime geçebilmesi için tercümanlara ihtiyaç duymaları işaret dilini kullanan insanları zora sokmakta.

Günümüz teknolojisinin sunduğu imkanlar ile tercümanlık işini basitleştirmek ve iletişimi biraz olsa daha basite indirgemek için çeşitli çeviri uygulaması geliştirilmiştir. Alternatif olarak "İşaret Dili Çevirmeni" uygulaması geliştirmeye karar verdik. Bu uygulama sayesinde görme ve işitsel engelli bireyler ile daha kolay iletişim sağlanabilecek, yöntem olarak sesten veya yazıdansa görsellere çeviri yapılarak iletişim sağlanacaktır.

1.1 Önceden Yapılmış Benzer Çalışmalar

Önceden bu konuyla ilgili yapılan uygulamalar bulunmaktadır.

1.1.1 Sesgoritma

Sesgoritma ismini taşıyan bu uygulama, konuşma engelli bireyin yaptığı işaret dili hareketlerini telefonunuzun kamerasını kullanarak hangi işaretin yapıldığını anlayıp hareket ile birlikte eş zamanlı olarak seslendirmekte ve dışarıdaki konuşma seslerini yazıya çevirerek konuşma veya işitme engellilerin çevresindeki insanlarla rahatça iletişim kurabilmesini sağlıyor. Sesgoritma projesi oldukça geniş kapsamlı bir yapay zeka araştırma, geliştirme ve ürün projesi.Proje içersinde en çok tercih edilen programlama dili olan Python ve OpenCV Kütüphaneleri kullanılmıştır. Proje geliştiricisi Arda MAVİ, 18 yaşında Bu proje macerasına 2017 yılında TÜBİTAK yarışmasına katılarak başlamıştır.[1]

1.1.2 Türk İşaret Dili Çevirmeni

Türk İşaret Dili Çevirmeni uygulamasının amacı, işitme engelli bireylerin sadece işaret dili bilenlerle değil, herkesle iletişim kurmasını kolaylaştırmaktır. Bunu yaparken sadece Türkiye'de değil işitme engelli bireylerin yurt dışında da karşılaştıkları sorunlar göz önünde bulundurularak İngilizce dil seçeneği ile de model oluşturma ve eğitim aşamaları gerçekleştirilmiştir. Çalışmaya eklenen gerçek zamanlı çeviri özelliği ile bireylerin anlık iletişim kurması sağlanmıştır. Bu sayede işitme engelli bireylerin yaşadığı iletişim sorunları büyük ölçüde azalabilecektir. Çalışmada mevcut veri setlerinin yetersiz olması nedeniyle videolar ile veri setleri oluşturulmasına ve en çok tercih edilen programlama dili olan Python kullanılmasına karar verilmiştir. İlk olarak VGG16, OpenCV kütüphaneleri kullanılarak model oluşturma ve eğitim aşamaları gerçekleştirilmiş. Modelin oluşum ve eğitim aşamalarında MediaPipe, OpenCV, Matplotlib, Numpy, Scikit-Learn, Pandas ve Os kütüphaneleri kullanılmış. Atılım Üniversitesi öğrencileri olarak bir araya gelen Ecem Balkaya, Seden Gülay Oral, Eylül Uçan projeyi geliştirmişlerdir.[2]

2. İŞARET DİLİNİN TARİHSEL GELİŞİMİ

Dünya Sağlık Örgütü'nün (DSÖ) verilerine göre, dünya genelinde 466 milyon işitme engelli yaşıyor. Bu kişilerin 34 milyonunu ise 15 yaşın altındaki çocuklar oluşturuyor. Tarihin her döneminde işitme engelli bireyler iletişim kurabilmek için birbirinden farklı işaret dili geliştirmişlerdir. Fransa'da 1770'li yıllarda işitme engellilerin kullandığı el hareketleri, grameri olan bir dil olarak kabul edilmiş ve okullarda öğretilmeye başlanmıştır. Daha sonra bu yöntem bir Fransız işaret dili bilimcisi tarafından Amerika'ya taşınmış ve orada 1817'de Thomas Gallaudet tarafından sadece işitme engellilere eğitim veren, ilk işaret dili öğreten okul 1817'de Thomas Gallaudet tarafından ilk işitme engelli okulu Amerika'da kurulmuştur. (Şimdiki adıyla, Gallaudet University).

3. METODLAR VE YÖNTEMLER

3.1 Kullanılabilecek Web Tabanlı Makine Öğrenme Araçları

Makine öğrenimi modelleri, girdi verilerinden örüntüler ve ilişkiler çıkararak, tahminler yapabilir, sınıflandırma yapabilir veya başka bir şekilde veri işleyebilirler. Web tabanlı makine öğrenme araçları, girdi verilerinden örüntüler ve ilişkiler çıkararak, tahminler yapabilir, sınıflandırma yapabilir veya başka bir şekilde veri işleyebilirler. Bu web tabanlı makine öğrenme araçları arasından seçim yapmamız gereklidir. Yapılan projenin gerekliliklerini en iyi karşılayan makine web tabanlı öğrenme aracı araştırılır.

3.1.1 Tensorflow.js

TensorFlow tarafından geliştirilen bir JavaScript kütüphanesidir. Tarayıcı üzerinde çalışır ve özelleştirilmiş makine öğrenimi modelleri oluşturmanıza olanak tanır. Görüntü, ses ve sensör verileri gibi çeşitli girdiler kullanabilir. Yüksek esneklik sunar ve daha gelişmiş projeler için kullanışlıdır.[3]

3.1.2 Runway ML

Görüntü, ses ve metin gibi farklı girdi verilerini kullanarak özelleştirilmiş makine öğrenimi modeller oluşturulmasına olanak tanır. Sanat, tasarım ve yaratıcı projeler için kullanılabilir. Hem ücretsiz hem de ücretli planlar sunar.[4]

3.1.3 Wekinator

Özelleştirilmiş makine öğrenimi modelleri oluşturmak için kullanılan bir araçtır. Sensör verileri, MIDI verileri ve diğer veriler gibi farklı girdileri işleyebilir. Özellikle etkileşimli müzik ve görsel performanslar gibi uygulamalar için kullanışlıdır.[5]

3.1.4 Teachable Machine

Google tarafından geliştirilen bir web tabanlı araçtır. Görüntü, ses ve hareket gibi girdileri kullanarak özelleştirilmiş makine öğrenimi modelleri oluşturulmasına olanak tanır. Kolay kullanımı ve hızlı sonuçlarıyla dikkat çeker.[6]

3.1.4.1 Denenen Web Tabanlı Makine Öğrenme Aracı (Teachable Machine)

Projemiz için gerekli olan görüntü işleme, özellikle de el işaretlerinin tanınması, karmaşık bir işlemdir ve bu nedenle özel bir model oluşturmak gerekir. Bu, özelleştirilmiş bir makine öğrenimi modeli oluşturmanın önemli olduğu anlamına gelir ve Teachable Machine bu amaç için kullanışlı bir araçtır. Teachable Machine, el işaretleri gibi benzersiz nesneleri tanımak için kullanılabilen farklı algoritmaları destekler. Bu algoritmalar arasında Evrişimli Sinir Ağları (Convolutional Neural Networks- CNNs), Destek Vektör Makineleri (Support Vector Machines - SVMs) ve Karar Ağaçları (Decision Trees) gibi çeşitli makine öğrenimi modelleri bulunur. Sonuç olarak, el işaretlerini algılamak için Teachable Machine kullanılarak model oluşturulup çeşitli denemeler sonucunda, oluşturulan modelin doğruluk oranının oldukça düşük olduğu için farklı yöntemlere başvurulmuştur.

3.2 Kullanılabilecek Makine Öğrenme Araçları

Makine öğrenme araçları, bilgisayar sistemlerinin verilerden öğrenme yeteneğine sahip olduğu bir yapay zekâ alanıdır. Bu araçlar, çeşitli programlama dilleri ve kütüphanelerle birlikte gelir. Python programlama dili TensorFlow, Keras gibi kütüphaneler sıkça kullanılır. Makine öğrenme araçları, kullanıcıların deneyim seviyelerine uygun soyutlama sağlar. Bu makine öğrenme araçları arasından seçim yapmamız gereklidir. Yapılan projenin gerekliliklerini en iyi karşılayan makine öğrenme aracı araştırılır.

3.2.1 Faster R-CNN

Faster R-CNN (Region-based Convolutional Neural Network), nesne algılaması için sıklıkla kullanılan bir modeldir. İkili bir aşamalı yaklaşım kullanır ve bölgesel öneriler üretir. Daha sonra bu öneriler üzerinde sınıflandırma ve konumlandırma yapar.[7]

3.2.2 RetinaNet

RetinaNet, zayıf ve güçlü özellikleri birleştiren bir nesne algılama modelidir. Farklı ölçeklerde özellik haritalarını kullanarak nesne tespiti yapar. Aynı zamanda, düşük özellikli ölçeklerde kaybolan küçük nesneleri de tespit edebilir. .[8]

3.2.3 YOLOv4

YOLOv4, YOLOv5'ten önceki bir versiyonudur ve nesne algılama ve sınıflandırma için kullanılan bir derin öğrenme modelidir. YOLOv4, daha önceki YOLO versiyonlarından daha hızlı ve daha doğru sonuçlar elde etmek için çeşitli geliştirmeler içerir. .[9]

3.2.4 EfficientDet

EfficientDet, hafif ve verimli nesne algılama modeli için tasarlanmış bir yöntemdir. Bu yöntem, birlikte öğrenme ve ölçek çubuğu kapsamında birden çok boyutu bir araya getirerek, daha küçük ve daha hızlı modeller oluşturmayı hedefler. .[10]

3.2.5 YOLOv5

YOLOv5, nesne algılama, sınıflandırma ve segmentasyon görevlerinde kullanılan bir derin öğrenme modelidir. YOLOv5, PyTorch kütüphanesi üzerinde geliştirilmiştir ve önceki YOLO versiyonlarından daha hızlı ve daha doğru sonuçlar elde etmek için çeşitli yenilikler içerir. YOLOv5, önceden eğitilmiş bir derin öğrenme modeli olarak, evrişimli sinir ağları (Convolutional Neural Networks- CNN) temelinde çalışır. Ölçeklenebilir bir mimariye sahiptir ve farklı veri setleri ve uygulamalar için özelleştirilebilir. YOLOv5, temel modelin yanı sıra çeşitli boyutlarda ve derinliklerdeki model varyasyonlarını da içerir.[11]

3.3 Seçilen Eğitim Aracı (YOLOv5)

Geliştirmek istediğimiz uygulamada gerçek zamanlı olarak kamera kullanarak tahmin yapılması gerektiğinden diğer araçlara göre YOLOv5, Gerçek zamanlı uygulamalarda kullanılabilecek hızlı bir performans sağlar. Yapılan denemeler sonucunda diğer araçlarla oluşturulan modellerin doğruluk oranı karşılaştırıldığında en yüksek doğruluk oranı YOLOv5 ile elde edildi. Kullanımı diğer araçlara göre daha kolay ve oldukça geniş bir topluluğa sahip. Topluluğun oluşturduğu erişilebilen birçok proje mevcut olması bu aracın kullanılması için avantaj sağlıyor. Bu durumlar göz önünde bulundurularak makine öğrenme aracı olarak YOLOv5 seçilmiştir.

3.3.1 YOLOv5 Modelleri

3.3.1.1 YOLOv5s

En küçük ve en hafif modeldir. Hızlı çalışır, ancak diğer modellere göre daha az hassas sonuçlar verebilir.

3.3.1.2 YOLOv5m

Orta boyutlu bir modeldir. Daha iyi algılama performansı sunar ve hızlıdır.

3.3.1.3 YOLOv5l

Daha büyük bir modeldir ve daha fazla parametre içerir. Bu model daha yüksek çözünürlüklerde daha iyi performans gösterir.

3.3.1.4 YOLOv5x

En büyük ve en karmaşık modeldir. Diğer modellere göre daha yüksek çözünürlükte nesne algılama yapabilir ve daha hassas sonuçlar üretebilir. Ancak, daha fazla hesaplama gücü gerektirir ve daha yavaş çalışır.

Model	size (pixels)	mAP^{val} 50-95	mAP^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

Şekil 3.1 YOLO modelleri.

3.4 Yararlanılan Kütüphaneler

3.4.1 OpenCV

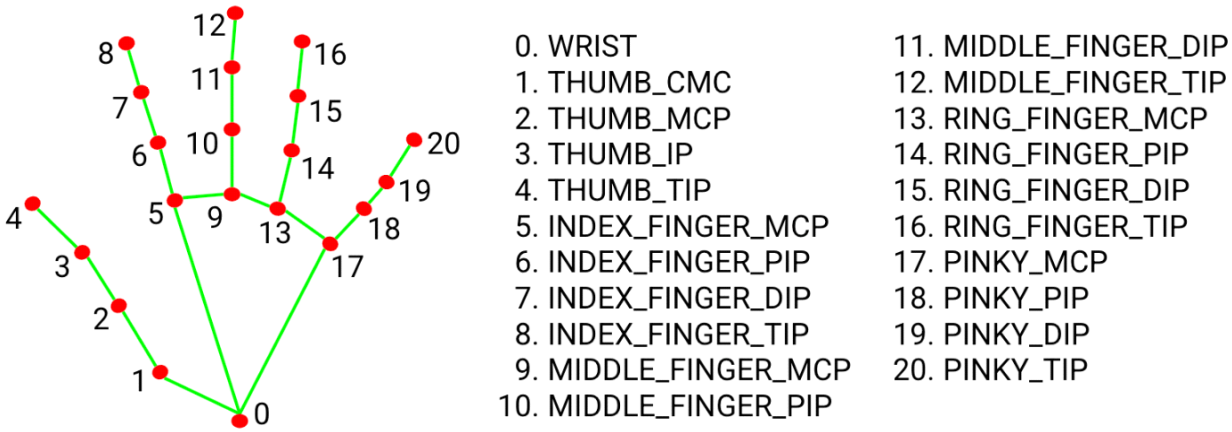
Bilgisayarla görü, makine öğrenimi, görüntü işleme, video analizi gibi uygulamalar için kullanılan devasa bir açık kaynak kodlu kütüphanedir. Gerçek zamanlı işlemlerde oldukça önemli bir rol oynar. OpenCV sayesinde web kameraları, video ve resim dosyaları bilgisayar tarafından analiz eder işlenilebilir hale getirir.[12]

3.4.2 NumPy

Dizilerle çalışmak için kullanılan bir Python kütüphanesidir. Ayrıca doğrusal cebir ve matrisler alanında çalışmak için de gerekli işlevlere sahiptir. Mediapipe kütüphanesi üzerinden elde ettiğimiz elin koordinatları ile daha rahat analiz yapmamızı sağlamıştır.[13]

3.4.3 MediaPipe

Mediapipe: Google tarafından geliştirilmiş bir makine öğrenme kütüphanesidir. Uygulama portföyü içerisinde yüz, vücut pozu ve el algılama gibi metodlar vardır.[14]



Şekil 3.2 Mediapipe'ın algıladığı elde koordinatlarını döndürdüğü noktalar.

3.4.4 TensorFlow

TensorFlow: çoklu makine öğrenimi, derin öğrenme algoritmaları ve modellerini bir araya getirir. Python'u makine öğrenimi için kullanmanıza olanak tanır ve uygulamalar oluşturmak için bir front end API sunar. Bu uygulamaları yürütmek ve yüksek performansın keyfini çıkarmak için C++ ile TensorFlow'u kullanabilir.[15]

3.4.5 Optimizer Modelleri

Derin öğrenme modellerinde, optimize edici (optimization algorithms) kullanılarak modelin hata fonksiyonunun minimum değerine ulaşmasını hedefler. Bu optimizasyon algoritmaları, modelin ağırlık ve bias (yanlılık) parametrelerini güncelleyerek hata fonksiyonunun minimumuna yaklaşır.

3.4.5.1 Gradient Descent (GD)

En basit optimizasyon algoritmasıdır. Hata fonksiyonunun negatif gradyanı kullanılarak ağırlık ve bias parametreleri günceller. GD, büyük veri setleri için yavaş çalışabilir.[16]

3.4.5.2 Stochastic Gradient Descent (SGD)

GD'nin hızlandırılmış bir versiyonudur. Veri seti rastgele örneklerle küçük gruplara (batch) bölünür ve her bir batch üzerinden GD uygulanır.[16]

3.4.5.3 Mini-batch Gradient Descent

Veri seti büyük olduğunda, aşırı uyum (overfitting) riskini azaltmak için SGD'nin bir varyasyonu olan mini-batch GD kullanılır.[16]

3.4.5.4 Adagrad

Bu optimizasyon algoritması, ağırlık ve bias parametrelerinin güncellenmesi için özelleştirilmiş bir öğrenme oranı kullanır. Adagrad, nadir görülen özellikleri olan veri setleri için daha uygun olabilir.[17]

3.4.5.5 RMSProp

Adagrad gibi bir öğrenme oranı tabanlı optimize edicidir, ancak ağırlık parametrelerinin geçmiş karelerinin üstel hareketli ortalamalarını kullanır. Bu, Adagrad'ın aşırı uyuma (overfitting) yol açabilecek öğrenme oranını aşırı azaltmasının önüne geçer.[18]

3.4.5.6 Adam

SGD'nin hızlandırılmış bir versiyonudur. Adam, momentum ve ikinci momenti birleştirerek parametrelerin güncellenmesini sağlar. Bu sayede, daha hızlı ve daha verimli bir optimizasyon sağlar.[19]

3.4.5.7 AdamW

Adam optimizasyon algoritmasının bir genişletmesidir. AdamW, ağırlık bozulmasının (weight decay) önlenmesine yardımcı olmak için ek bir terim ekler. Bu durumda, AdamW algoritması, büyük ağırlıkları azaltarak ağırlık bozulmasını önlemeye çalışır.[19]

3.4.6 PIL (Python Imaging Library)

Python programlama dilinde görüntü işleme işlemleri gerçekleştirmek için kullanılan bir kütüphanedir. PIL, çeşitli görüntü formatlarını destekler ve bu formatlardaki resimleri yükleyebilir ve kaydedebilir. Ayrıca, resimler üzerinde çeşitli dönüşümler, boyutlandırma, kırpma, döndürme, renk düzenleme gibi işlemleri yapmanıza olanak sağlar.[20]

3.4.7 Matplotlib

Matplotlib, çeşitli grafik türleri oluşturmak ve veri analizi sonuçlarını görselleştirmek için kullanılır. 2D ve 3D grafikleri, çizgi grafiklerini, dağılım grafiklerini, histogramları, çubuk grafiklerini, pasta grafiklerini ve daha fazlasını oluşturmanızı sağlar. Bu grafikler, veri setlerinin özelliklerini, ilişkilerini ve dağılımlarını görselleştirerek veri analizi süreçlerine yardımcı olur.[21]

3.4.8 PyTorch

PyTorch, makine öğrenimi ve derin öğrenme için kullanılan açık kaynaklı bir Python kütüphanesidir. PyTorch, tensör hesaplamaları ve otomatik gradyan hesaplaması gibi temel işlevleri sağlar ve aynı zamanda derin sinir ağları (Deep Neural Networks- DNN) oluşturmak, eğitmek ve değerlendirmek için gelişmiş bir araç seti sunar.[22]

Şekil 3.3 PyTorch

3.4.9 GTTS (Google Text-to-Speech)

Python programlama dilinde metni sese dönüştürmek için kullanılan bir kütüphanedir. Google'ın metinden sese dönüştürme API'ını kullanarak çalışır. Bir metin girdisi alır ve bu metni sesli bir çıktıya dönüştürür. Çıktı, kullanıcının belirleyebileceği bir ses dosyası olarak kaydedilebilir veya doğrudan çalınabilir. Bu, metni sesli olarak duyurmak veya sesli yanıtlar üretmek gibi birçok uygulama için kullanışlıdır.[23]

3.4.10 Jupyter NoteBook

Jupyter Notebook, Python ve diğer programlama dilleriyle etkileşimli hesaplama, veri analizi, veri görselleştirme, makine öğrenimi ve araştırma çalışmaları yapmak için kullanılan bir web tabanlı bir uygulamadır. Jupyter Notebook, projeleri, kodları, metinleri, denklemleri, grafikleri ve diğer medya türlerini içeren belgeler oluşturmanızı sağlar.[24]

3.5 Veri Setleri

Veri setlerindeki fotoğrafların net, düzenli, ayırt edilebilir vb. olması makine öğrenmesi için oldukça önemlidir. Modelin doğruluk oranının yüksek olması için birçok yöntem denenmiştir.

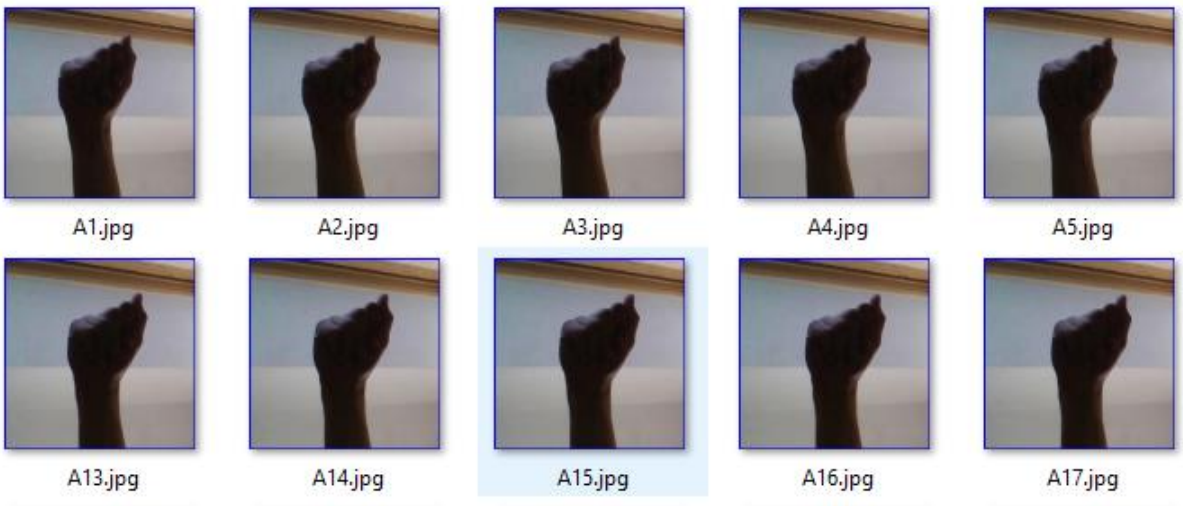
3.5.1 Denenen Veri Setleri

3.5.1.1 Kaggle Harf Veri Seti

Kaggle bünyesinde makine öğrenmesi, veri analizi, yapay zekâ, istatistik, veri görselleştirme, matematik bilimleri gibi birçok veri seti barındıran global çaplı bir web sitedir.

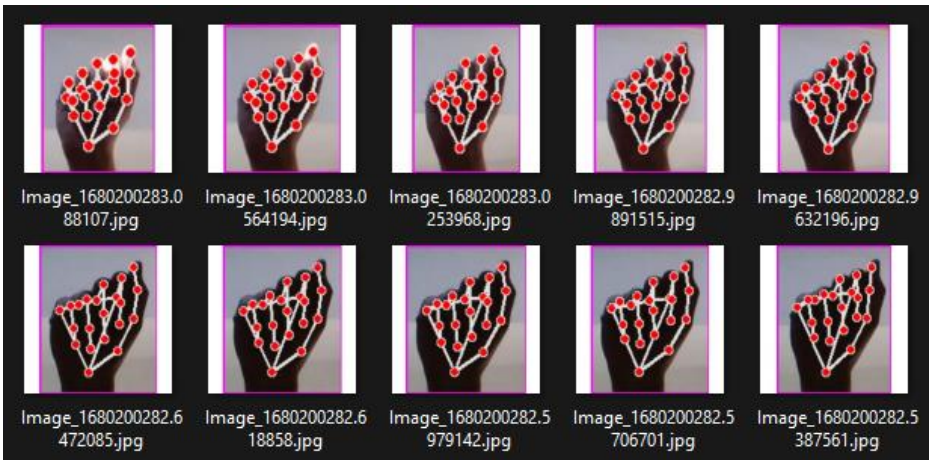
İçerisinde kullanıcıların kendi paylaştığı veri kümeleri, bu verilerin analizini paylaştığı kaynak kodlar gibi içeriklere sahip.

Kaggle içerisinde birçok veri seti ile karşılaşılmıştır. Bulduğumuz veri setlerinde genellikle harf başına düşen fotoğraf sayısı oldukça fazladır. Bunlardan birine örnek verecek olursak harf başına düşen 4000 adet fotoğraf bulunmaktadır.[25]



Şekil 3.4 Kaggle veri seti.

Fotoğrafların görüntü kalitesinin güzel olmaması, fotoğrafların karanlık olması, yapılan el işaretinin ayırt edici olmaması gibi sebeplerden dolayı MediaPipe kütüphanesi kullanılarak veri setine landmarklar eklenmiştir. Yapılan geliştirme sonucunda veri setinin daha doğru bir model oluşturması amaçlanmıştır.



Şekil 3.5 Landmark eklenmiş Kaggle veri seti.

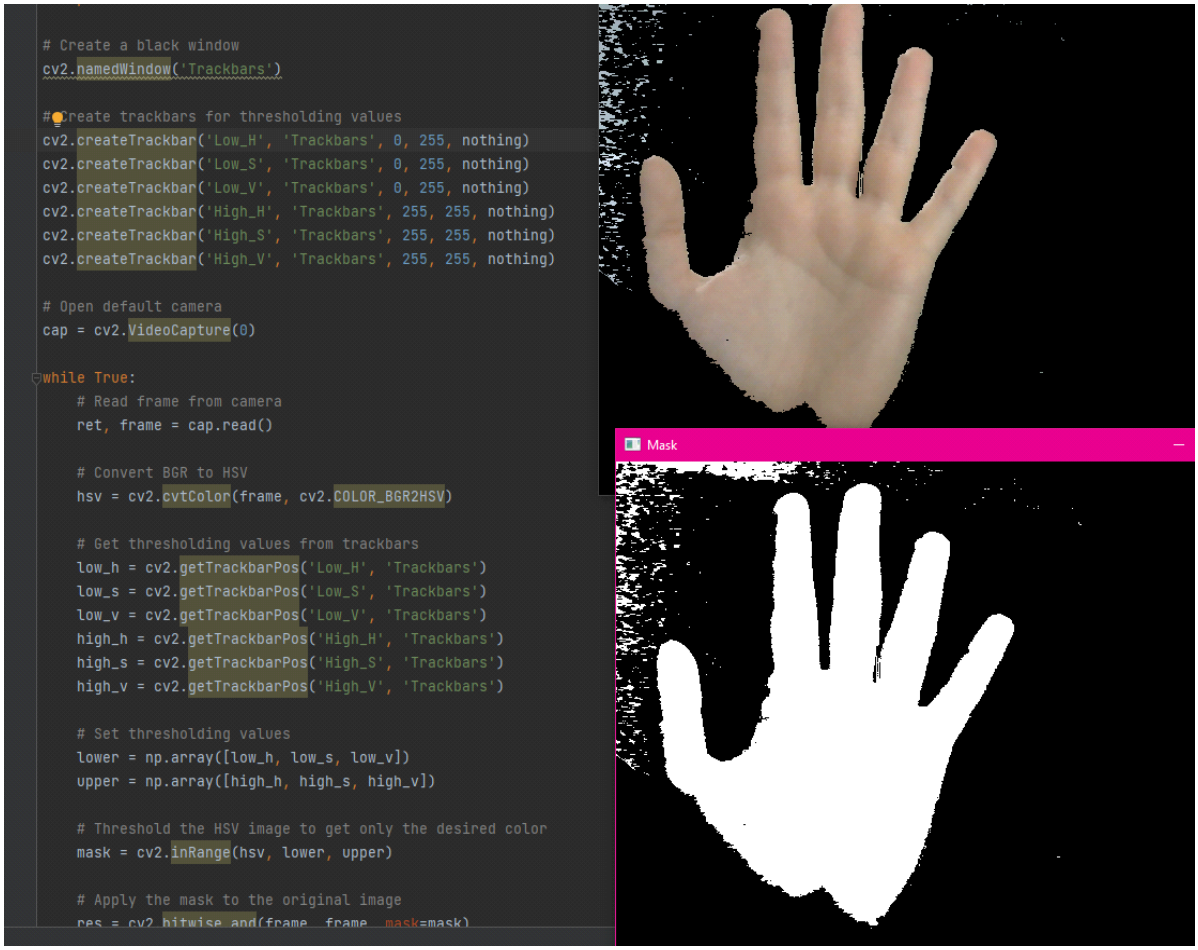
Yapılan tüm geliştirmelerin ardından landmark eklenmiş veri seti kullanılmıştır. Makine öğrenmesi, web tabanlı Teachable Machine kullanılarak gerçekleştirilmiştir. Oluşan model ile çeşitli testler yapılmıştır. Yapılan testlerin sonucunda doğruluk oranının oldukça düşük olduğu tespit edilmiştir. Bu yüzden farklı yöntemlere başvurulmuştur.

```
else:
    imgCrop = res[y - offset:y + h + offset, x - offset:x + w + offset]
    aspectRatio = h / w
    if aspectRatio > 1:
        k = imgSize / h
        wCal = math.ceil(k * w)
        imgResize = cv2.resize(imgCrop, (wCal, imgSize))
        wGap = math.ceil((300 - wCal) / 2)
        imgWhite[:, wGap:wCal + wGap] = imgResize
        f = io.StringIO()
        with redirect_stdout(f):
            prediction, index = classifier.getPrediction(imgWhite)
        # print(prediction,index)
        # print(labels[index])
        harf_ekle(labels[index])
        tahmin_et()

    else:
        k = imgSize / w
        hCal = math.ceil(k * h)
        imgResize = cv2.resize(imgCrop, (imgSize, hCal))
        hGap = math.ceil((300 - hCal) / 2)
        imgWhite[hGap:hCal + hGap, :] = imgResize
        f = io.StringIO()
        with redirect_stdout(f):
            prediction, index = classifier.getPrediction(imgWhite)
        # print(prediction,index)
        # print(labels[index])
        harf_ekle(labels[index])
        tahmin_et()

    cv2.imshow("ImageCrop", imgCrop)
    cv2.imshow("ImageWhite", imgWhite)
```

Şekil 3.6 Mediapipe'tan yararlanarak eli "z" indeksi fark etmeksizin 300x300'e yeniden şekillendiren ve eğitilen modele tahminde bulunan kod.

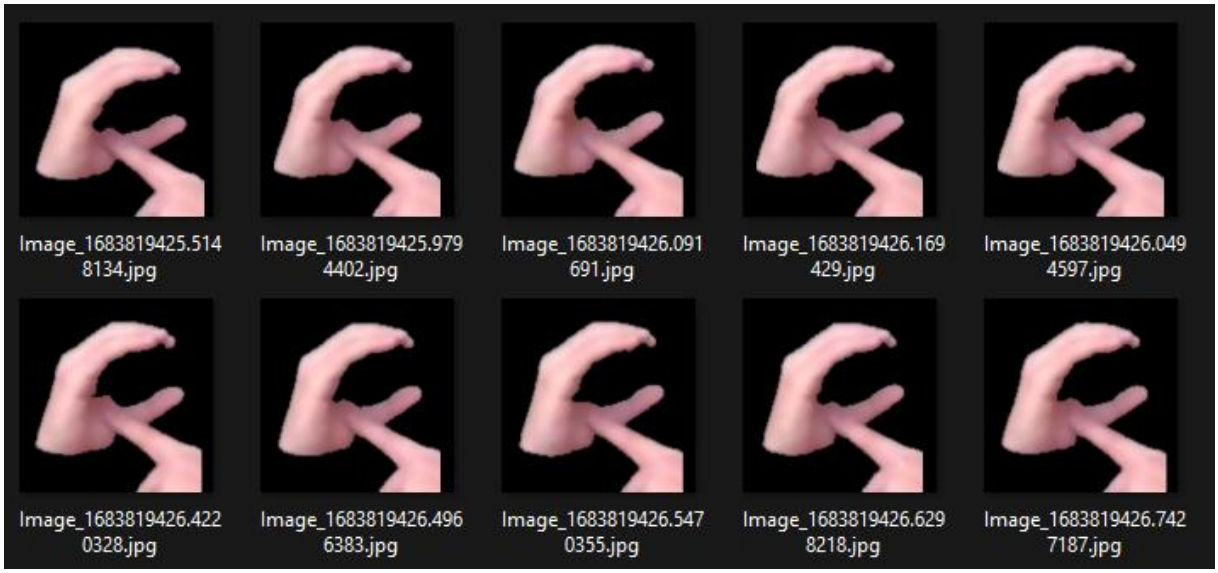
3.5.1.2 Maskeleme Yöntemiyle Oluşturulan Veri Seti

Modelin doğruluk oranının daha iyi olması için veri setinin tekrar oluşturulmasına karar verilmiştir. Veri seti fotoğrafları çekilirken renk ve maskeleme ayarlarını değiştirilmesini sağlayan algoritma eklenmiştir. Bu sayede eli odak noktası haline getirerek maskeleme yapılmıştır. Elin arka planı siyah gözükecek şekilde fotoğraflar çekilmiştir. El işaretlerinin renk ve arka plana bağlı olmadan makine öğrenmesi hedeflenmiştir.



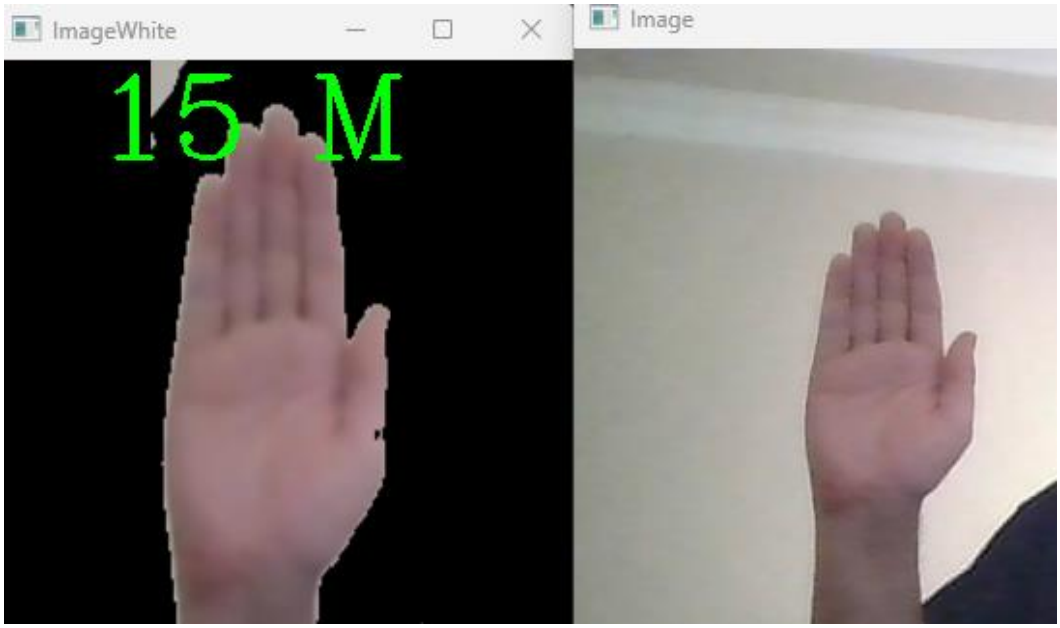

```
# Create a black window
cv2.namedWindow('Trackbars')

#Create trackbars for thresholding values
cv2.createTrackbar('Low_H', 'Trackbars', 0, 255, nothing)
cv2.createTrackbar('Low_S', 'Trackbars', 0, 255, nothing)
cv2.createTrackbar('Low_V', 'Trackbars', 0, 255, nothing)
cv2.createTrackbar('High_H', 'Trackbars', 255, 255, nothing)
cv2.createTrackbar('High_S', 'Trackbars', 255, 255, nothing)
cv2.createTrackbar('High_V', 'Trackbars', 255, 255, nothing)

# Open default camera
cap = cv2.VideoCapture(0)

while True:
    # Read frame from camera
    ret, frame = cap.read()

    # Convert BGR to HSV
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Get thresholding values from trackbars
    low_h = cv2.getTrackbarPos('Low_H', 'Trackbars')
    low_s = cv2.getTrackbarPos('Low_S', 'Trackbars')
    low_v = cv2.getTrackbarPos('Low_V', 'Trackbars')
    high_h = cv2.getTrackbarPos('High_H', 'Trackbars')
    high_s = cv2.getTrackbarPos('High_S', 'Trackbars')
    high_v = cv2.getTrackbarPos('High_V', 'Trackbars')

    # Set thresholding values
    lower = np.array([low_h, low_s, low_v])
    upper = np.array([high_h, high_s, high_v])

    # Threshold the HSV image to get only the desired color
    mask = cv2.inRange(hsv, lower, upper)

    # Apply the mask to the original image
    res = cv2.bitwise_and(frame, frame, mask=mask)
```

Şekil 3.7 Maskeleme yöntemiyle fotoğrafların çekilmesini sağlayan algoritma.



Şekil 3.8 Maskeleme yöntemiyle oluşturulan veri seti.

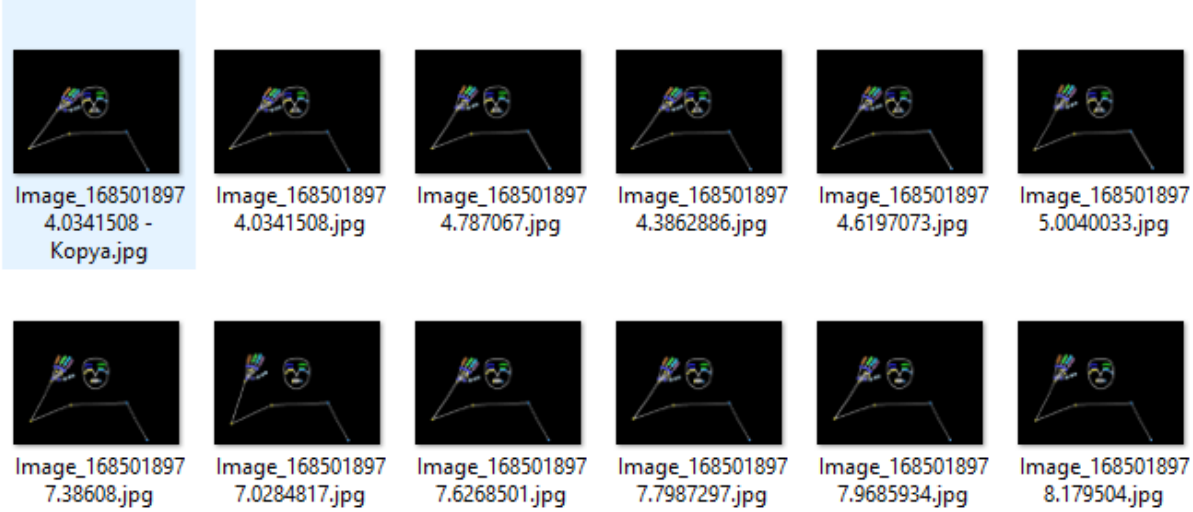
Maskeleme yöntemiyle oluşturulan veri seti kullanılarak Teachable Machine üzerinden eğitim modeli oluşturulmuştur. Oluşturulan model koda entegre edildikten sonra testler yapılmıştır. Yapılan testler sonucunda uygulamanın yaptığı tahminlerin çok fazla yanılması sonucunda modelin doğruluk oranının yeterli olmadığına karar verilmiştir.



Şekil 3.9 “B” harfini “M” algılayarak yanlış tahminde bulunan uygulama.

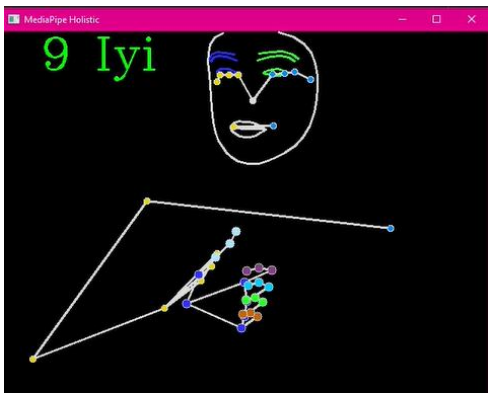
3.5.1.3 Vücut Hatlarından Yararlanılarak Oluşturulan Veri Seti

Vücut hatlarından yararlanmak ve arka plan, ışık vb. dış etkenlerden etkilenmemesi için MediaPipe kütüphanesi kullanılmıştır. Siyah bir arka plan üzerine eklenen landmarkların gözükeceği şekilde veri seti oluşturulmuştur. Sadece vücut hatları ve ellerin gözüktüğü landmarkların daha net ve ayırt edilebilir gözüktüğünden doğruluk oranı arttırılması hedeflenmiştir.



Şekil 3.10 Vücut hatlarından yararlanarak oluşturulan veri seti.

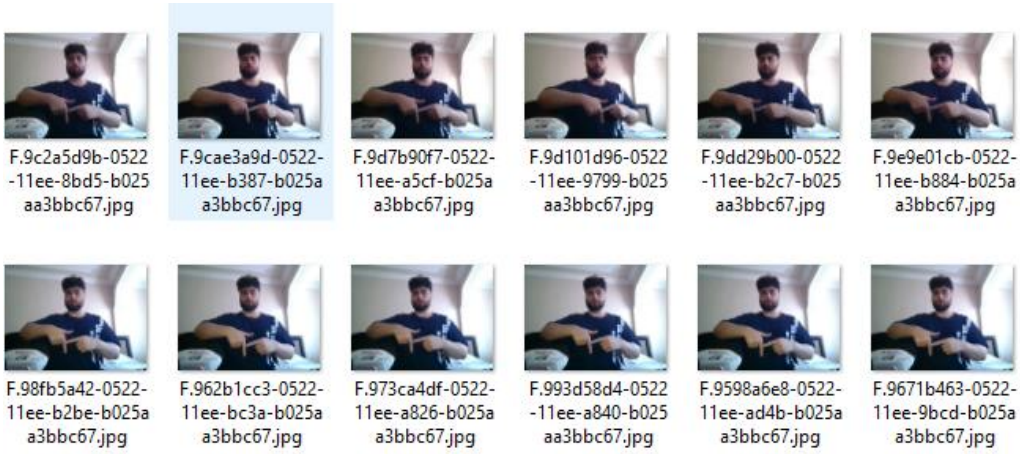
Oluşturulan veri seti ile web tabanlı Teachable Machine üzerinden eğitim modeli oluşturulmuştur. Modeli kendi oluşturduğumuz koda entegre ettikten sonra çeşitli testler yapılmıştır. Yapılan testlerin sonucunda tahminlerin doğruluk oranının oldukça düşük olduğu tespit edilmiştir.



Şekil 3.11 Oluşturulan model kullanılarak yapılan örnek bir tahmin.

3.5.2 Kullanılan Veri Seti

Anlık olarak veri seti toplamamızı sağlayabilen bir algoritma geliştirildi [4.2]. Oluşturulan algoritma sayesinde 640x480 piksellik fotoğraflar elde edildi. Veri seti içerisinde her harf ve kelime için 30 adet fotoğraf bulunmakta. Bu veri setini kullanmamızın sebepleri, fotoğraflar ışık, arka plan, el açıları gibi dış etken çeşitliliğinin detaylı olması, fotoğrafların yükseklik ve genişlik olarak birbirleri ile aynı olması, tüm fotoğraflara düzenli ve sıralı olarak isimlendirilme verilmiş olması. Bu sebepler projemiz için yeterli buluğundan, oluşturduğumuz bu veri seti kullanılmıştır.

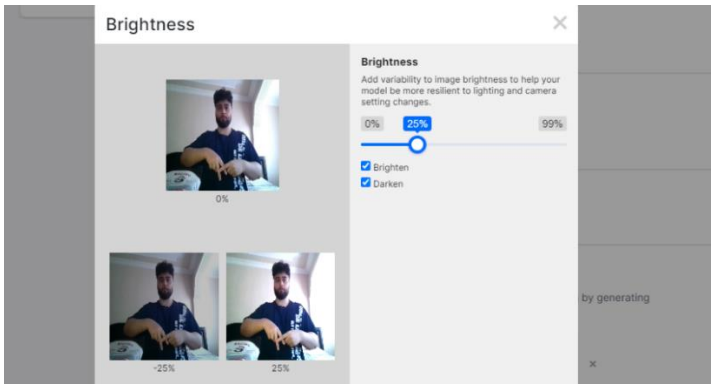


Şekil 3.12 Kullanılan harf veri seti örneği.

.

3.5.2.1 Kullanılan Veri Setinin Geliştirilmesi

Projede kullanılmak üzere seçilen veri setinin modelin öğretilmesinde daha yüksek bir doğruluk oranı yakalamak adına harf ve kelime başına düşen fotoğraf sayısının artırılmasına karar verildi. Fotoğrafların çeşitlendirilmesi için Roboflow sitesi kullanılarak, her bir fotoğrafın parlaklık değeri yüzde yirmi beş oranında arttırıldı ve azaltıldı bu sayede bir adet fotoğrafın iki farklı parlaklıktaki hali veri setine eklendi. Bu yöntem sayesinde her harf ve kelime için 90 adet fotoğraf sayısı elde edildi.[21]



Şekil 3.13 Parlaklık değeri değiştirilerek arttırılan veri seti.

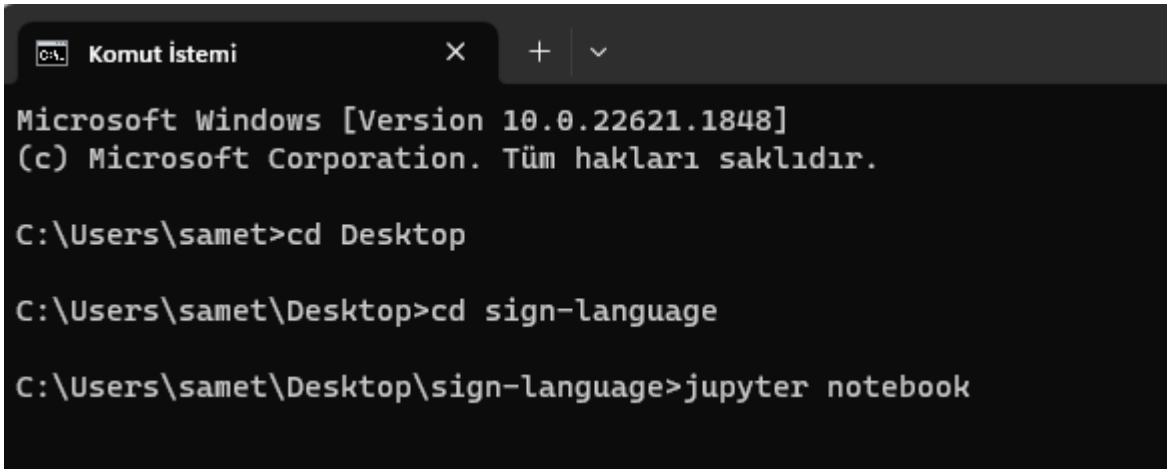
4. GELİŞTİRİLEN UYGULAMA

4.1 Jupyter NoteBook'un Kurulumu ve Kullanılması

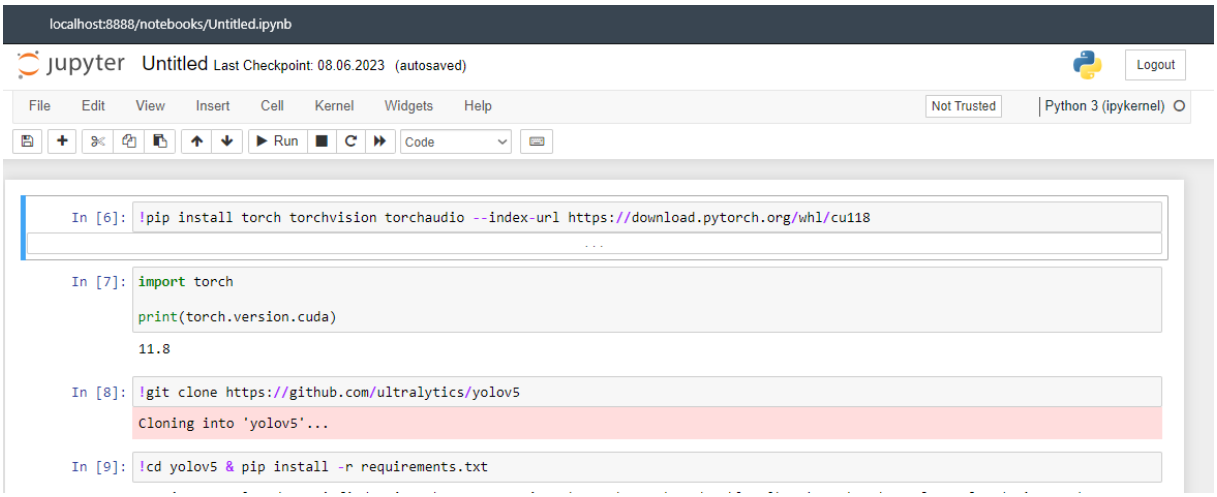
Python'un başarıyla kurulmasının ardından, Jupyter Notebook'u kurmak için pip adlı bir Python paket yöneticisi kullanacağız.[24]

```
C:\Users\PC>pip install jupyter
Collecting jupyter
  Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Collecting notebook
  Downloading notebook-6.5.4-py3-none-any.whl (529 kB)
     ---------------------------------------- 529.8/529.8 kB 1.1 MB/s eta 0:00:00
Collecting qtconsole
  Downloading qtconsole-5.4.3-py3-none-any.whl (121 kB)
     ---------------------------------------- 121.9/121.9 kB 1.4 MB/s eta 0:00:00
Collecting jupyter-console
  Downloading jupyter_console-6.6.3-py3-none-any.whl (24 kB)
Collecting nbconvert
  Downloading nbconvert-7.6.0-py3-none-any.whl (290 kB)
     ---------------------------------------- 290.4/290.4 kB 1.2 MB/s eta 0:00:00
Collecting ipykernel
  Downloading ipykernel-6.23.2-py3-none-any.whl (152 kB)
     ---------------------------------------- 152.8/152.8 kB 1.3 MB/s eta 0:00:00
```

Şekil 4.1 Jupyter NoteBook'un kurulmasını.



Şekil 4.2 Projenin kurulu olduğu yerde Jupyter'in açılması.



Şekil 4.3 8888 portundan bağlanılarak açılan Jupyter Notebook kullanıma hazır hale getirilmiştir.

4.2 Veri Seti Oluşturmak İçin Geliştirilen Algoritma

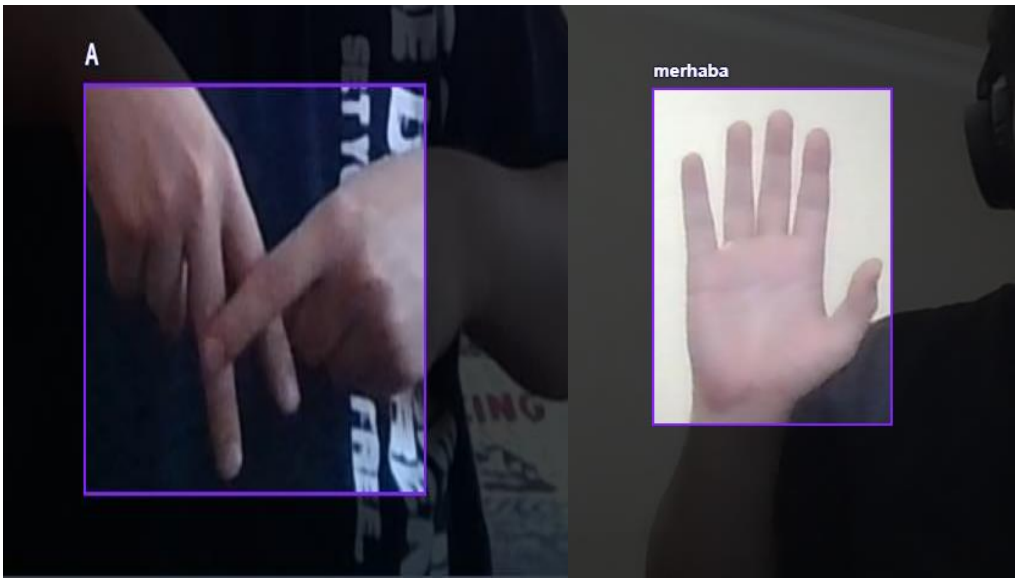
İşaret dilindeki harf ve kelime veri setini oluşturmak için algoritma geliştirilmesine karar verilmiştir. Cv2 kütüphanesinden faydalanarak oluşturulan algoritma ile 640x480 piksellik net ve makinenin algılaya bileceği şekilde veri seti toplanabilmiştir.

```
import cv2

word = 'A'

IMAGES_PATH = os.path.join('Data', word)  # /data/images
label = word
sayac = 0

cap = cv2.VideoCapture(0)
# Loop through labels
while True:
    ret, frame = cap.read()
    cv2.imshow('Image Collection', frame)
    if cv2.waitKey(1) == ord(' '):
        imgname = os.path.join(IMAGES_PATH, label + '.' + str(uuid.uuid1()) + '.jpg')
        cv2.imwrite(imgname, frame)
        sayac += 1
        print(sayac)

cap.release()
cv2.destroyAllWindows()
```

Şekil 4.4 Veri setini oluşturmak için geliştirilen algoritma.

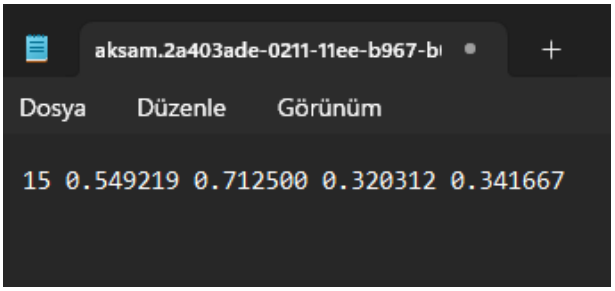
4.3 Veri Setindeki Fotoğrafların Anotasyonlanması ve Sınıflandırılması

Kullanılan veri seti üzerinden el işaretlerinin algılanması için fotoğraflar anotasyonlanmış ve sınıflandırılmıştır. Bu yöntem Roboflow sitesi kullanılarak yapılmıştır. Gerçekleştirilen işlem, fotoğraf üzerinden yapılan el işaretini seçerek anlamına göre sınıflara ayırmaktır.[26]



Şekil 4.5 Anotasyonlama yapılmış veri seti örnekleri.

Tüm fotoğrafların anotasyonlama işlemi yapıldıktan sonra her bir anotasyon için label.txt dosyası Roboflow sitesi tarafından oluşturuldu. Oluşan dosyalar öğretilecek model için kullanılmak üzere indirildi. Ardından dosyalar veri setinin yanına eklendi. İndirilen label.txt dosyaları içeriğinde ilk sırada sınıf sayısı bilgisi gelmektedir. Ardından oluşturulan karenin dört köşe noktasının fotoğraf üzerindeki koordinatları gösterilmektedir.



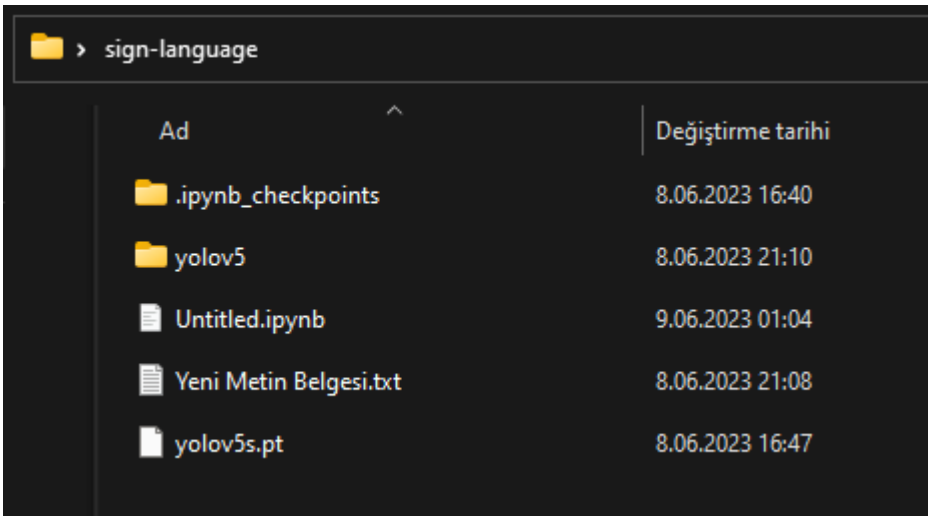
Şekil 4.6 Sınıf sayısı ve anotasyon işleminin koordinatlarının bulunduğu txt dosyası.

4.4 YOLOv5' in Yüklenmesi

Projenin oluşturulacağı yere karar verilmiştir. Ardından karar verilen dosyanın içerisine YOLOv5 Github üzerinden klonlanmıştır. [11]

```
!git clone https://github.com/ultralytics/yolov5

Cloning into 'yolov5'...
```

Şekil 4.7 Github üzerinden YOLOv5'in yüklenmesi.



Şekil 4.8 YOLOv5'in dosyaya yüklenmiş hali.

Ardından yolov5 adlı klasörün içerisindeki "requirements.txt" dosyasının içerisindeki gerekli paketler YOLOv5 ile birlikte çalışması için yüklenmiştir.

```
!cd yolov5 & pip install -r requirements.txt

Requirement already satisfied: gitpython>=3.1.30 i
Requirement already satisfied: matplotlib>=3.3 in
Requirement already satisfied: numpy>=1.18.5 in c:
Requirement already satisfied: opencv-python>=4.1.
Requirement already satisfied: Pillow>=7.1.2 in c:
Requirement already satisfied: psutil in c:\users\
Requirement already satisfied: PyYAML>=5.3.1 in c:
Requirement already satisfied: requests>=2.23.0 in
Requirement already satisfied: scipy>=1.4.1 in c:\
Requirement already satisfied: thop>=0.1.1 in c:\u
Requirement already satisfied: torch>=1.7.0 in c:\
Requirement already satisfied: torchvision>=0.8.1
```

Şekil 4.9 Gerekli paketlerin yüklenmesi.

4.5 Modelin Eğitilmesi (YOLOv5)

Bu adımda, oluşturduğumuz veri setini YOLOv5 kütüphanesi kullanılarak eğitim yapılacaktır.

```
model

AutoShape(
  (model): DetectMultiBackend(
    (model): DetectionModel(
      (model): Sequential(
        (0): Conv(
          (conv): Conv2d(3, 32, kernel_size=(6, 6), stride=(2, 2), padding=(2, 2))
          (act): SiLU(inplace=True)
        )
        (1): Conv(
          (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
          (act): SiLU(inplace=True)
        )
        (2): C3(
          (cv1): Conv(
            (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1))
            (act): SiLU(inplace=True)
          )
          (cv2): Conv(
            (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1))
            (act): SiLU(inplace=True)
          )
          (cv3): Conv(
            (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
            (act): SiLU(inplace=True)
          )
...
        )
      )
    )
  )
)
```

Şekil 4.10 Modelin katman yapısı.

YOLOv5 dosyası içerisinde bulunan "train.py" içerisindeki katmanlar ayarlanarak eğitime hazır hale getirilmiştir. Modelin katman yapısının son hali şekil [4.10]'te gözükmektedir.

Oluşturulan modelin yapısında, YOLOv5'in tüm eğitim dosyaları ve katmanları "train.py" dosyasının içerisinde bulunmaktadır. Verilen tüm veri setini 416x416 şeklinde tekrardan boyutlandırmıştır. Epochs değeri, bir makine öğrenimi modelinin eğitim veri kümesini kaç kez geçeceğini belirten bir parametredir. Modelimizde kullanılan epochs değeri kendi modelimize en uygun şekli deneme yanılma yöntemiyle bulunmuştur. Kendi oluşturduğumuz veri setinin sınıflar ve etiketleri "dataset.yml" olarak model yapısının içine entegre edilmiştir. YOLOv5'in kullanılan modeli "weights" olarak belirtilmiştir. Kendi modelimizde YOLOv5'in "yolov5x" modeli kullanılmıştır. Model kod bloğunun en sonunda belirtilen "workers" değeri ise kaç adet işlemci çekirdeğinin eğitim sürecinde yardım etmesini belirtmektedir.

```
!cd yolov5 && python train.py --img 416 --batch 16 --epochs 100 --data dataset1.yml --weights yolov5x.pt --workers 6
```

Şekil 4.11 Model eğitimini başlatan kod bloğu.

“dataset1.yml” dosyasının içeriğinde bulunan “path” Veri setinin dosya yolunu içerir. “train” modeli eğitmek için kullanılacak olan fotoğraflardır. “val” modelin eğitim sırasında her epochtan sonra bu konumdaki fotoğraflar üzerinden tahminler üreteceği dosyaların yoludur. Ardından yaptığı tahminlerin ne kadar doğru olduğuna bakar ve mAP değerini döndürür. Dolayısıyla epochs sayısı arttıkça doğruluk oranı da artar. “test” model eğitildikten sonra modeli denemek için kullanılan benzer fotoğrafların olduğu dosya yoludur. “nc” modelde bulunan sınıfların adetini gösterir. “names” sınıfların adını içeren diziyi gösterir.

```yaml
path: ../SignLanguageDataSet
test: test/images
train: train/images
val: valid/images

nc: 29
names:
  [
    "A","B","C","C2","D","E","F","G","G2","H","I","I2","J","K","L",...
```

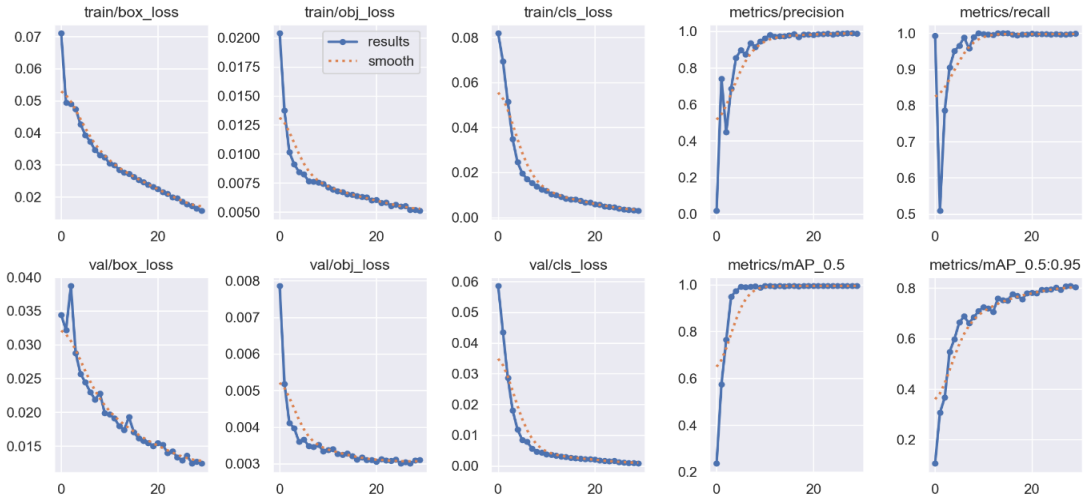
Şekil 4.12 “dataset1.yml” dosyasının içeriği.

```
Fusing layers...
Model summary: 322 layers, 86361826 parameters, 0 gradients, 204.4 GFLOPs
Adding AutoShape...
```

Şekil 4.13 Model katmanları ve parametreler.

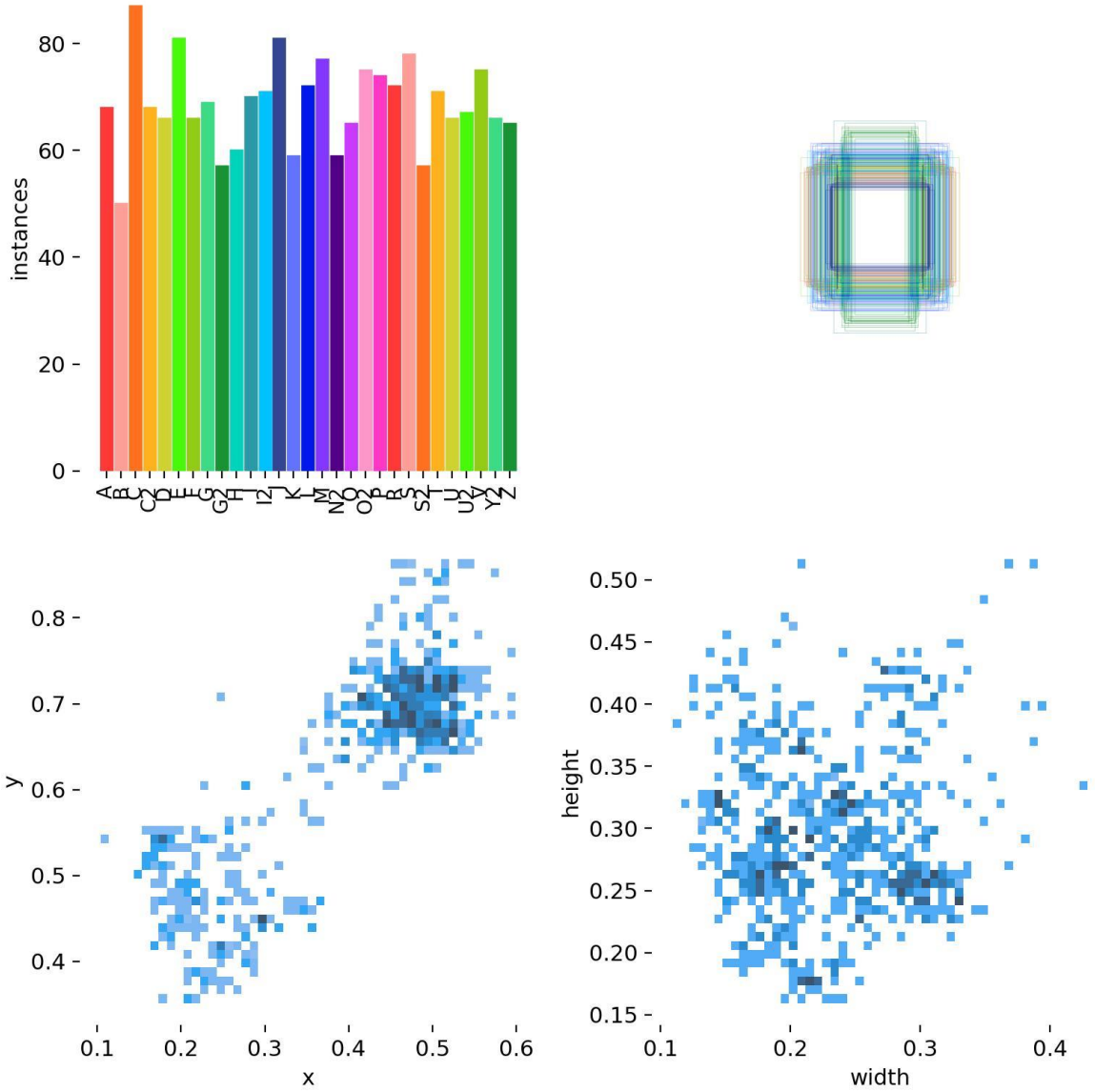
4.5.1 Harfler İçin Eğitilen Model

Harfler için olan modelin epoch değeri 30'a ayarlandıktan sonra şekil [4.11] deki kod bloğu çalıştırılarak model eğitime başlamıştır.



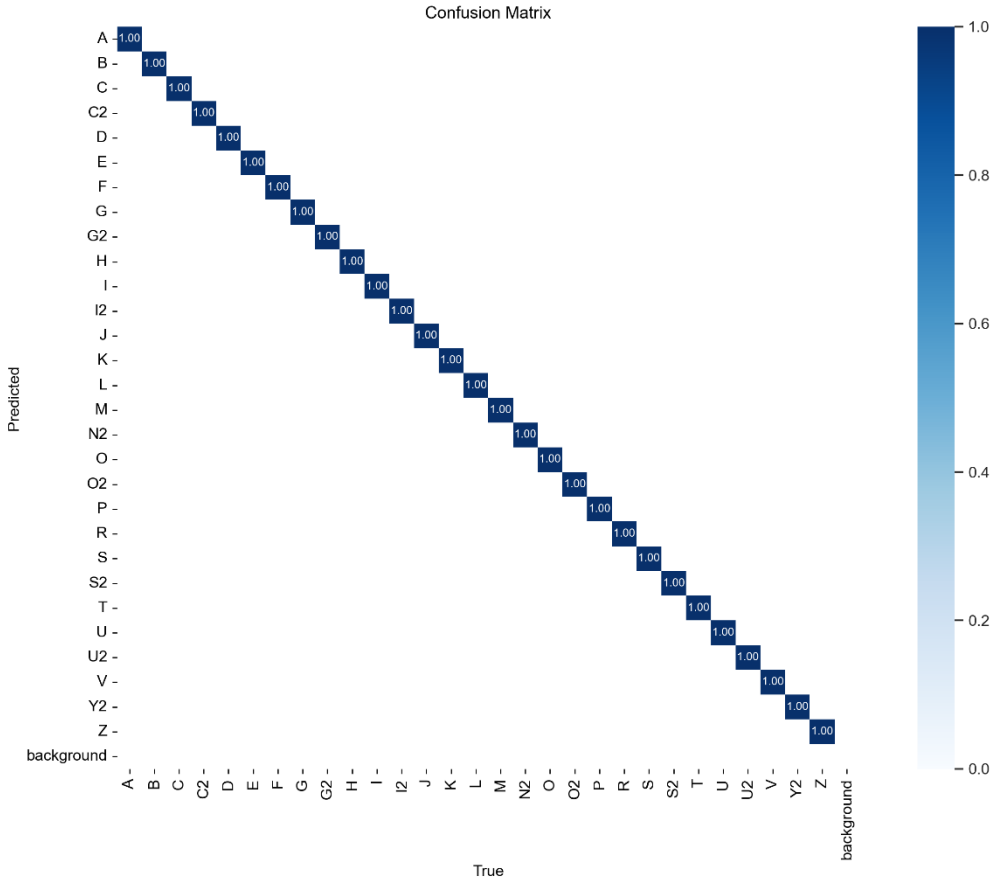
Şekil 4.14 Harfler için eğitilen modelin sonuç grafikleri. (results.png)

"train/box_loss", sınırlayıcı kutu tahminlerini ne kadar iyi yapabildiğini gösterir. "train/obj_loss", modelin nesne varlığını doğru bir şekilde tespit etme yeteneğini yansıtır. "train/cls_loss", modelin sınıfları doğru bir şekilde sınıflandırma yeteneğini temsil eder. "val/box_loss" bu grafik, modelin nesne sınırlayıcı kutularının doğruluğunu değerlendirmek için kullanılan doğrulama (validation) veri kümesi üzerindeki kayıp değerini gösterir. "val/obj_loss" modelin nesne varlığı (objectness) tahminlerinin doğruluğunu değerlendirmek için kullanılan doğrulama veri kümesi üzerindeki kayıp değerini temsil eder. "val/cls_loss", modelin doğrulama (validation) veri kümesinde sınıf tahminlerinin doğruluğunu değerlendirmek için kullanılan kayıp değeridir. "metrics/precision", modelin hassasiyet (precision) performansını gösterir. "metrics/recall", modelin geri çağırma (recall) performansını gösterir. Geri çağırma, gerçek pozitiflerin doğru bir şekilde tespit edilme oranını ifade eder. "metrics/mAP_0.5" modelin ortalama hassasiyetin (mean average precision) belirli bir eşik değeri (threshold) kullanarak nasıl performans gösterdiğini gösterir. "metrics/mAP_0.5:0.95" modelin ortalama hassasiyetin belirli bir eşik aralığında nasıl performans gösterdiğini gösterir. Bu durumda, 0.5 ile 0.95 arasındaki farklı eşik değerleri altında hesaplanan hassasiyetin ortalamasıdır.



Şekil 4.15 Harfler için eğitilen modelin sınıflar ve anotasyonlanan fotoğraflar hakkında bilgi içeren grafikler (labels.jpg).

Karmaşıklık matrisi, bir sınıflandırma veya tahmin modelinin performansını değerlendirmek için kullanılan bir araçtır. Şekilde [4.16] görüldüğü üzere gerçek ve tahmin edilen sınıflar arasındaki ilişkiyi göstermektedir.



Şekil 4.16 Harfler için karmaşıklık matrisi.

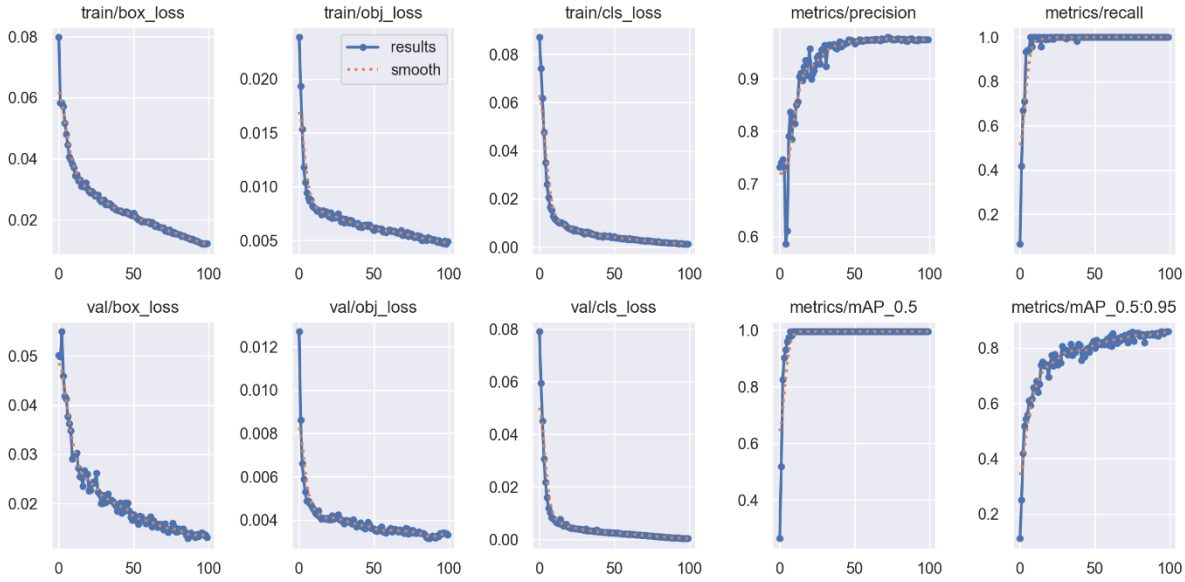
epoch,	train/box_loss,	train/obj_loss,	train/cls_loss,	metrics/precision,	metrics/recall,	metrics/mAP_0.5,metrics/mAP_0.5:0.95,
0,	0.071056,	0.02039,	0.081956,	0.017895,	0.99342,	0.23672, 0.10574,
1,	0.049425,	0.013733,	0.069361,	0.74138,	0.5089,	0.57457, 0.30673,
2,	0.049057,	0.010152,	0.051427,	0.44818,	0.78695,	0.7655, 0.36691,
3,	0.04744,	0.0091261,	0.034778,	0.68592,	0.90591,	0.94951, 0.54762,
4,	0.042571,	0.0084611,	0.02462,	0.85425,	0.9507,	0.97399, 0.59854,
5,	0.039309,	0.0082618,	0.01956,	0.89685,	0.96552,	0.99089, 0.66427,
6,	0.037208,	0.0076555,	0.017124,	0.87283,	0.98849,	0.99086, 0.68821,
7,	0.034614,	0.0076311,	0.015302,	0.9365,	0.95821,	0.99266, 0.66161,
8,	0.03305,	0.0075357,	0.013804,	0.91501,	0.98964,	0.99417, 0.68507,
9,	0.03233,	0.0074471,	0.012525,	0.94524,	1,	0.98761, 0.7093,
10,	0.030478,	0.0071402,	0.011804,	0.96283,	0.99824,	0.995, 0.72557,
11,	0.029863,	0.0069441,	0.010396,	0.98151,	0.99667,	0.995, 0.71936,
12,	0.028462,	0.0067976,	0.0099166,	0.97117,	0.99585,	0.99454, 0.70589,
13,	0.027661,	0.0067239,	0.0091103,	0.97259,	1,	0.995, 0.75953,
14,	0.027167,	0.0065493,	0.0083351,	0.97358,	1,	0.99371, 0.7525,
15,	0.026267,	0.0064915,	0.008091,	0.97862,	1,	0.995, 0.7516,
16,	0.025243,	0.0063919,	0.008041,	0.98667,	0.99709,	0.995, 0.77714,
17,	0.024633,	0.0063136,	0.007432,	0.96803,	0.99445,	0.99393, 0.77005,
18,	0.023761,	0.0062673,	0.0066008,	0.984,	0.99719,	0.995, 0.75664,
19,	0.023169,	0.0060329,	0.0066593,	0.98404,	0.99635,	0.995, 0.78001,
20,	0.022432,	0.0060462,	0.005854,	0.97982,	0.99857,	0.995, 0.78214,
21,	0.021553,	0.0057846,	0.0057447,	0.9849,	0.99779,	0.995, 0.77929,
22,	0.021018,	0.0058246,	0.0049757,	0.98468,	0.99776,	0.995, 0.79373,
23,	0.019898,	0.0055381,	0.0046517,	0.98888,	0.99816,	0.995, 0.79342,
24,	0.019578,	0.0056539,	0.0045997,	0.9843,	0.99665,	0.995, 0.79588,
25,	0.018512,	0.0054976,	0.0039034,	0.98883,	0.99785,	0.995, 0.80254,
26,	0.017763,	0.0055417,	0.0034959,	0.9884,	0.99668,	0.995, 0.79371,
27,	0.017135,	0.0052,	0.0033347,	0.98971,	0.99728,	0.995, 0.80729,
28,	0.016363,	0.0052141,	0.0032736,	0.98957,	0.99818,	0.995, 0.80883,
29,	0.015648,	0.0051156,	0.0029684,	0.98831,	0.99913,	0.995, 0.80336,

Şekil 4.17 Harfler için eğitilen modelin sonuçları (results.csv)

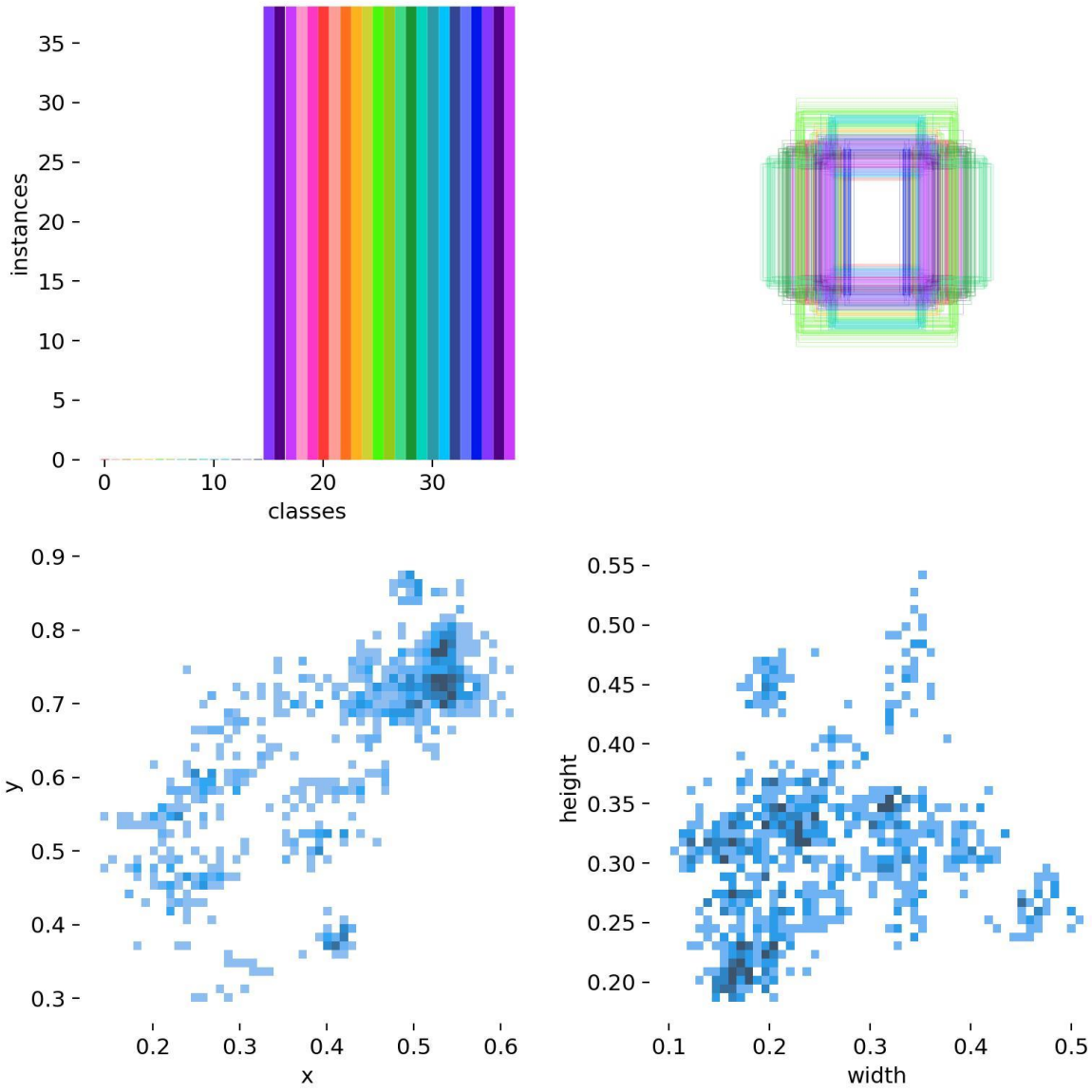
Model verilen epoch değerine ulaştıktan sonra modelin mAP_0.5 ve mAP_0.5:0.95 değerleri kontrol edilir. Eğittiğimiz modele bakılırsa mAP_0.5 değeri 0.995'e, mAP_0.5:0.95 değeri ise 0.80336'a ulaşmıştır. Ulaştığımız değerler modelin doğru eğitildiğini ve doğruluk payının yüksek olacağını göstermektedir.

4.5.2 Kelimeler İçin Eğitilen Model

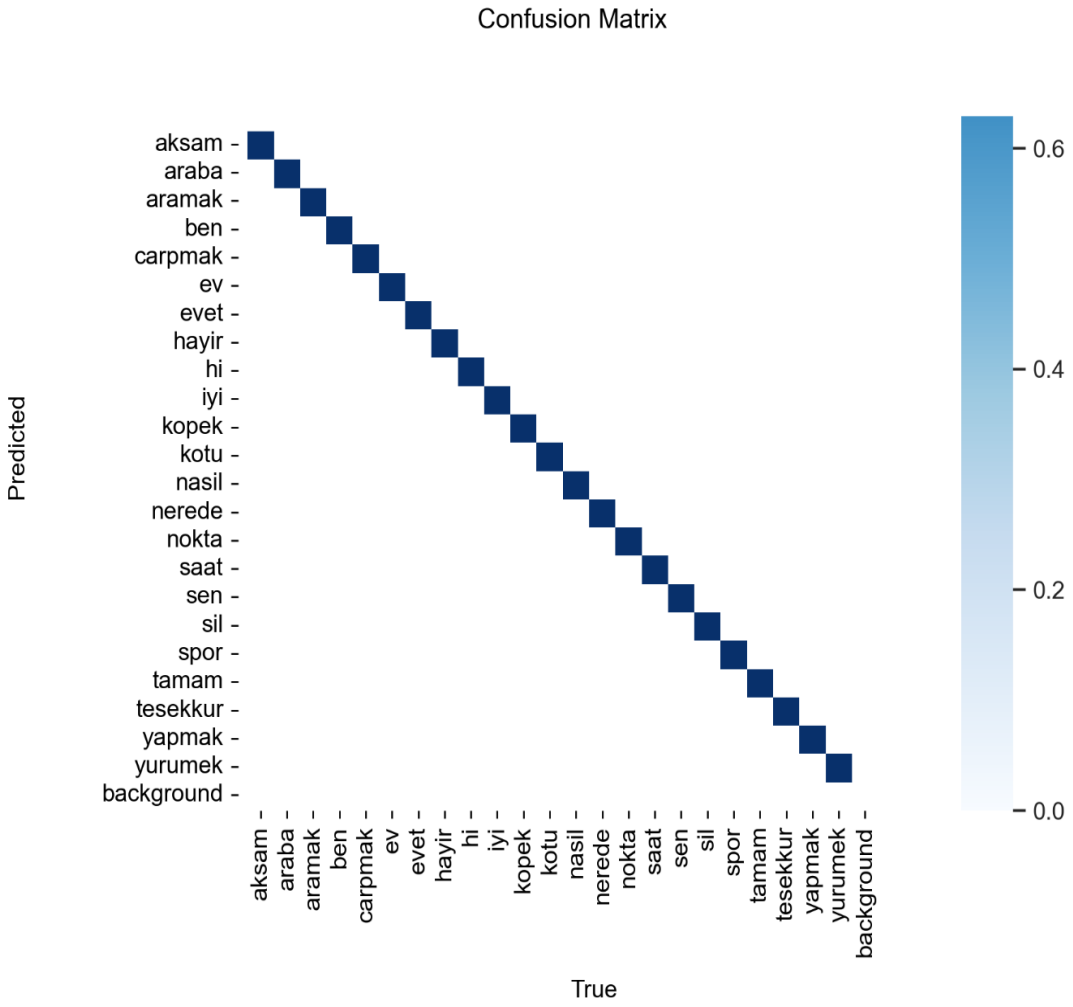
Kelimeler için olan model epoch değeri 100'a ayarlandıktan sonra şekil [4.11] deki gibi model eğitimi başlamıştır.



Şekil 4.18 Kelimeler için eğitilen modelin sonuç grafikleri (results.png).



Şekil 4.19 Kelimeler için eğitilen modelin sınıflar ve anotasyonlanan fotoğraflar hakkında bilgi içeren grafikler (labels.jpg).



Şekil 4.20 Kelimeler için karmaşıklık matrisi.

epoch,	train/box_loss,	train/obj_loss,	train/cls_loss,	metrics/precision,	metrics/recall,	metrics/mAP_0.5,metrics/mAP_0.5:0.95,	
71,	0.016438,	0.0055162,	0.0025709,	0.97529,	1,	0.995,	0.84389,
72,	0.016598,	0.0057663,	0.0028038,	0.97863,	1,	0.995,	0.81906,
73,	0.016121,	0.0054078,	0.0024741,	0.9783,	1,	0.995,	0.85143,
74,	0.016672,	0.0055129,	0.0025129,	0.97625,	1,	0.995,	0.8417,
75,	0.016005,	0.0056297,	0.0024543,	0.97423,	1,	0.995,	0.85798,
76,	0.015633,	0.0054147,	0.0024466,	0.97433,	1,	0.995,	0.8244,
77,	0.0159,	0.0052947,	0.0022605,	0.97404,	1,	0.995,	0.85148,
78,	0.015461,	0.0053784,	0.0022183,	0.97529,	1,	0.995,	0.85358,
79,	0.015407,	0.0054664,	0.002263,	0.97573,	1,	0.995,	0.8537,
80,	0.015238,	0.0052566,	0.0021992,	0.97509,	1,	0.995,	0.84285,
81,	0.015044,	0.0053894,	0.0019679,	0.9735,	1,	0.995,	0.84715,
82,	0.014599,	0.0050265,	0.0019648,	0.9746,	1,	0.995,	0.84827,
83,	0.014712,	0.0051655,	0.0019069,	0.97506,	1,	0.995,	0.81904,
84,	0.014506,	0.0051901,	0.0018599,	0.9724,	1,	0.995,	0.84816,
85,	0.014207,	0.0050291,	0.0018571,	0.97164,	1,	0.995,	0.8503,
86,	0.014073,	0.0053068,	0.0018042,	0.97459,	1,	0.995,	0.84716,
87,	0.013841,	0.0051781,	0.0017247,	0.9751,	1,	0.995,	0.8494,
88,	0.013661,	0.0051721,	0.0016619,	0.97594,	1,	0.995,	0.85143,
89,	0.0137,	0.0050043,	0.001737,	0.97381,	1,	0.995,	0.84704,
90,	0.013476,	0.005131,	0.0017565,	0.97281,	1,	0.995,	0.84929,
91,	0.013103,	0.0050311,	0.001757,	0.97355,	1,	0.995,	0.85359,
92,	0.013052,	0.0048911,	0.0014471,	0.97253,	1,	0.995,	0.84283,
93,	0.012826,	0.0050173,	0.0013948,	0.97481,	1,	0.995,	0.85149,
94,	0.012577,	0.0048329,	0.0015997,	0.97409,	1,	0.995,	0.85906,
95,	0.012285,	0.0048688,	0.0015398,	0.97363,	1,	0.995,	0.85688,
96,	0.012066,	0.0047154,	0.0014092,	0.97448,	1,	0.995,	0.85154,
97,	0.012193,	0.004992,	0.0013512,	0.9737,	1,	0.995,	0.85693,
98,	0.012124,	0.0046832,	0.001319,	0.97432,	1,	0.995,	0.85911,
99,	0.012225,	0.0049331,	0.001277,	0.97409,	1,	0.995,	0.85904,

Şekil 4.21 Kelimeler için eğitilen modelin sonuçları (results.csv)

Kelimeler için eğittiğimiz modelin mAP_0.5 değeri 0.995, mAP_0.5:0.95 değeri ise 0.85904'e yaklaşmıştır. Model kullanılabilecek ölçüde doğruluk oranına yaklaşmıştır.

4.6 Paketlerin ve Modelin Yüklenmesi

Eğitilen modelin kullanılabilmesi için gereken kütüphaneler import edilmiştir. Ardından harfler için (exp9/weights/best.pt) ve kelimeler için (exp15/weights/best.pt) oluşturulan modeller "torch" kütüphanesinden yararlanılarak model ve model2 adlı değişkenlere atanmıştır.

```
import cv2
import numpy as np
from PIL import Image, ImageTk
import tkinter as tk
from gtts import gTTS
import os
import pygame
import uuid
from collections import Counter
import torch
from matplotlib import pyplot as plt

model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5/runs/train/exp9/weights/best.pt', force_reload=True)
model2 = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5/runs/train/exp15/weights/best.pt', force_reload=True)
```

Şekil 4.22 Paketler ve modellerin hazırlanması.

4.7 Tahmin Algoritmasının Geliştirilmesi

Makineye öğretilmiş model dosyasını baz alarak kelime ve harfler için tahminde bulunacak algoritma geliştirilmiştir.

```
def process_frame(frame):

    if modelBool == True:

        results = model(frame)
    else:
        results = model2(frame)

    if len(results.pandas().xyxy[0]) == 0:
        empty_results_count += 1
    else:
        empty_results_count = 0

    if empty_results_count >= 10:
        last_20_frames.clear()
        empty_results_count = 0

    class_names = results.pandas().xyxy[0]["name"].tolist()
    last_20_frames.extend(class_names)

    if len(last_20_frames) > 30:
        last_20_frames = last_20_frames[-30:]

    counter = Counter(last_20_frames)

    if len(counter) > 0 and len(last_20_frames) == 30:
        most_common_class = counter.most_common(1)[0][0]
        print(most_common_class)
        last_20_frames.clear()
        most_common_class= most_common_class.lower()
        word=most_common_class


        text_box.configure(state=tk.NORMAL)
        text_box.insert(tk.END, word)
        text_box.configure(state=tk.DISABLED)
        firstWord = word

    rendered_frame = np.squeeze(results.render())
    return rendered_frame
```

Şekil 4.23 Tahminde bulunan algoritma.

Geliştirilen algoritma cv2 kütüphanesinden yararlanarak son 20 karenin her biri için tahmin yapar (results = model(frame)). Ardından son 20 tahmin içerisinde en çok tekrar eden sınıfı bulur ve ekrana yazdırır. Buradaki asıl amaç yapılan tahminden emin olmaktır.

4.8 Uygulama Arayüzünün Oluşturulması

Geliştirdiğimiz uygulamanın tüm arka plan işlemleri bitirilmiştir. Kelimeleri ve harfleri yüksek oranda doğru tahmin edebilen uygulama geliştirilmiştir. Bu kısımda kullanıcıların uygulamayı kullanabilmesi için sade ve basit bir arayüz tasarlanmıştır.

4.8.1 Uygulamanın Ekrana Yazdırma Fonksiyonları

Bu kısımda, uygulamanın harf veya kelime tahmininde bulunması için mod değiştirme özelliği eklenmiştir. Kelime ve harflerin arasına boşluk bırakabilme özelliği eklenmiştir. Son harfi veya son kelimeyi silme özellikleri eklenmiştir. Ekranda yazan tüm metnin tamamını temizleme ve sesli şekilde okunmasını sağlayan özellikler eklenmiştir.

```
def change_mod():
    global modelBool
    modelBool = not modelBool

def del_letter():
    text_box.configure(state=tk.NORMAL)
    current_text = text_box.get("1.0", tk.END).strip()  # Mevcut içeriği alır ve boşlukları temizler
    updated_text = current_text[:-1]  # Son harfi siler
    text_box.delete("1.0", tk.END)  # Mevcut içeriği siler
    text_box.insert("1.0", updated_text)  # Güncellenmiş içeriği ekler
    text_box.configure(state=tk.DISABLED)

def delete_last_word():
    text_box.configure(state=tk.NORMAL)
    current_text = text_box.get("1.0", tk.END).strip()  # Mevcut içeriği alır ve boşlukları temizler
    words = current_text.split()  # İçeriği kelimelere böler
    if len(words) > 0:
        words.pop()  # Son kelimeyi listeden çıkarır
    updated_text = " ".join(words)  # Güncellenmiş içeriği yeniden birleştirir
    text_box.delete("1.0", tk.END)  # Mevcut içeriği siler
    text_box.insert("1.0", updated_text)  # Güncellenmiş içeriği ekler
    text_box.configure(state=tk.DISABLED)

def add_space():
    text_box.configure(state=tk.NORMAL)
    current_text = text_box.get("1.0", tk.END).strip()  # Mevcut içeriği alır ve boşlukları temizler
    updated_text = current_text + " "  # Sonuna bir boşluk ekler
    text_box.delete("1.0", tk.END)  # Mevcut içeriği siler
    text_box.insert("1.0", updated_text)  # Güncellenmiş içeriği ekler
    text_box.configure(state=tk.DISABLED)

def button_click_clear():
    text_box.configure(state=tk.NORMAL)
    text_box.delete("1.0", tk.END)  # Mevcut içeriği siler
    text_box.configure(state=tk.DISABLED)


def speak_text():
    text = text_box.get("1.0", tk.END).strip()
    unique_filename = str(uuid.uuid4()) + ".mp3"
    tts = gTTS(text=text, lang='tr')
    tts.save(unique_filename)
```

Şekil 4.24 Uygulama arayüzü özellikleri için yazılan fonksiyonlar.

```
root = tk.Tk()

button_frame = tk.Frame(root)
button_frame.pack(side=tk.BOTTOM)

button_exit = tk.Button(button_frame, text="Mod Değiştir", command=change_mod)
button_exit.pack(side=tk.LEFT, padx=10, pady=10)

button_clear = tk.Button(button_frame, text="Temizle", command=button_click_clear)
button_clear.pack(side=tk.LEFT, padx=10, pady=10)

button_harf_sil = tk.Button(button_frame,text="Harf Sil", command = del_letter)
button_harf_sil.pack(side=tk.LEFT,padx=10,pady=10)

button_clear = tk.Button(button_frame, text="Kelime Sil", command=delete_last_word)
button_clear.pack(side=tk.LEFT, padx=10, pady=10)

button_yaz = tk.Button(button_frame, text="Boşluk", command=add_space)
button_yaz.pack(side=tk.LEFT, padx=10, pady=10)

button_speak = tk.Button(button_frame, text="Seslendir", command=speak_text)
button_speak.pack(side=tk.LEFT, padx=10, pady=10)

button_exit = tk.Button(button_frame, text="Çık", command=button_click_exit)
button_exit.pack(side=tk.LEFT, padx=10, pady=10)

text_box_frame = tk.Frame(root)
text_box_frame.pack(side=tk.BOTTOM, pady=10)

text_box = tk.Text(text_box_frame, height=5)
text_box.pack(fill=tk.BOTH, expand=True)
text_box.configure(state=tk.DISABLED)

label_video = tk.Label(root)
label_video.pack()
```

Şekil 4.25 Uygulama arayüzü özellikleri için eklenen butonlar.

4.8.2 Uygulamaya Sesli Okuma Özelliği Eklenmesi

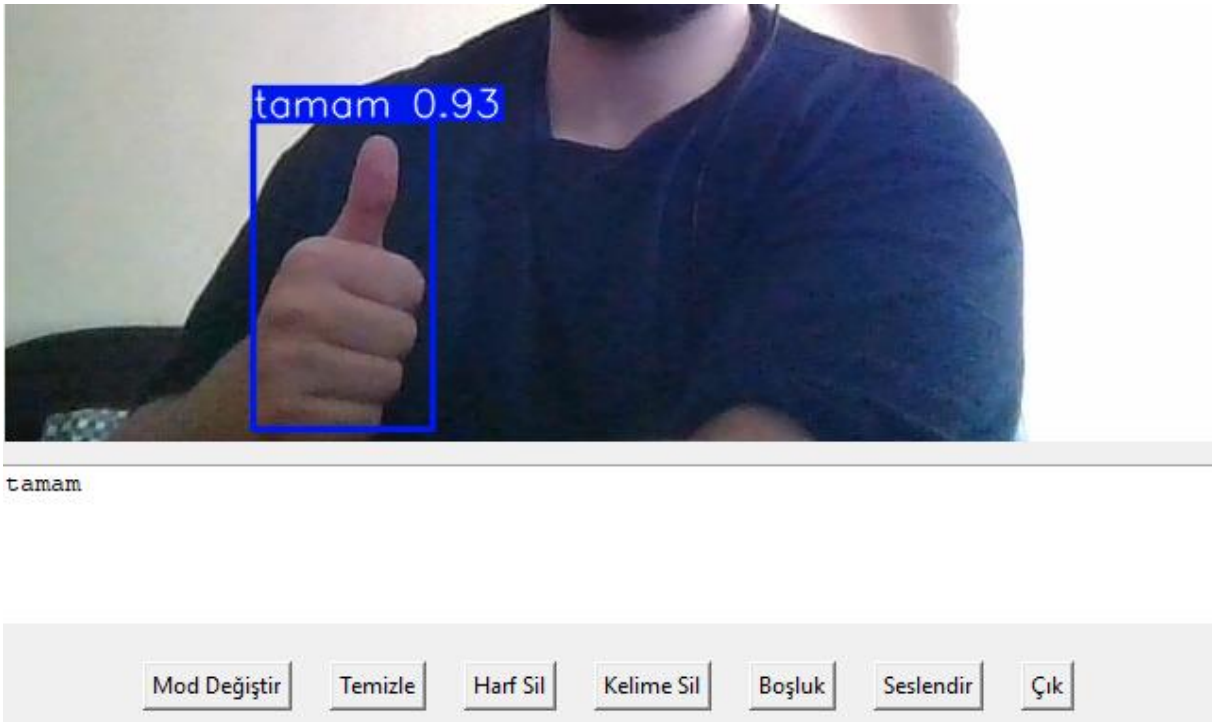
Bu kısımda kullanıcı, ekrana yazdırılan metni sesli şekilde dinleyebilmesi için eklenmiştir. Kod, metni Türkçe dilinde bir ses dosyasına dönüştürür ve benzersiz bir dosya adıyla kaydeder. Bu özellik kullanıcılara daha etkili bir okuma deneyimi sunmak amacıyla kullanılmıştır.

```
def speak_text():
    text = text_box.get("1.0", tk.END).strip()
    unique_filename = str(uuid.uuid4()) + ".mp3"
    tts = gTTS(text=text, lang='tr')
    tts.save(unique_filename)

    # Sesin oynatılması
    pygame.mixer.init()
    pygame.mixer.music.load(unique_filename)
    pygame.mixer.music.play()


def button_click_exit():
    pygame.mixer.init()
    pygame.mixer.music.load("sound.mp3")
    pygame.mixer.music.play()

    # Oluşturulan MP3 dosyalarının silinmesi
    for filename in os.listdir("."):
        if filename.endswith(".mp3"):
            if filename != "sound.mp3":
                os.remove(filename)

    # Programın kapatılması
    video_capture.release()
    cv2.destroyAllWindows()
    root.destroy()
```

Şekil 4.26 Sesli okuma özelliğinin eklendiği kod bloğu.

4.9 Uygulama Arayüzü

Uygulama kullanıcılar için kullanılabilir hale getirilmiştir. Eklenen tüm özellikler doğru ve tutarlı şekilde çalışmaktadır. Basit bir tasarıma sahip olan uygulamamız ile kolayca kelime ve harf modu arasında geçiş yapılabilir bu sayede kullanıcılar iletmek istedikleri cümleyi oluşturabilirler. Geliştirilen uygulama ile engelli bireylerin daha rahat şekilde iletişime geçmeleri sağlanmıştır.



Şekil 4.27 Uygulamanın güncel arayüzü.

5. SONUÇLAR

İşaret dilini tanıyan uygulamamızın işitme ve konuşma engelli bireylerin sosyal yaşantılarını daha iyi standartlara yerleştirmiş, yaşantılarını kolaylaştırmıştır. İşaret diline hâkim olmayan bireyler ile iletişime geçememe sorununu ortadan kaldırmaya yönelik bir uygulama oluşturulmuştur.

Python programlama dili kullanılarak gerekli kütüphane implementasyonları yapılmıştır. Birçok veri seti üzerinde denemeler yapılmış, en uygun olarak tarafımızca oluşturulan veri seti ile makine öğretiminin gerçekleştirilmesine karar verilmiştir. Kendi oluşturduğumuz veri setini çeşitli uygulamalar üzerinde geliştirmiş ve el işaretlerini fotoğraflar üzerinde ne anlama geldikleri etiketlenmiştir. Veri setine yapılan etiketlemeler sonucunda model YOLOv5 kullanılarak oluşturulmuştur. Oluşturulan model sonucunda yüksek bir doğruluk oranı elde edilmiştir. Makinenin anladığı harf ve kelimeleri birleştirmek ve ekrana yazdırmak için bir uygulama geliştirilmiştir. Belirlenen süre içerisinde planlanan takvimle paralel olarak ilerlenmiştir. Geliştirilen uygulama ile hedeflenen sonuca ulaşılmış engelli bireylerin iletişim kurmaları kolaylaştırılmıştır.

KAYNAKLAR

[1] https://ardamavi.github.io/Sesgoritma-TheNewAge/

[2] https://bilisimyildizlari.org.tr/2021/yapay-zeka-ile-isaret-dilini-farkli-dillerde-metne-ceviren-mobil-uygulama

[3] https://www.tensorflow.org/js

[4] https://runwayml.com/

[5] http://www.wekinator.org/

[6] https://teachablemachine.withgoogle.com/

[7] https://arxiv.org/abs/1506.01497

[8] https://arxiv.org/abs/1708.02002

[9] https://github.com/AlexeyAB/darknet

[10] https://arxiv.org/abs/1911.09070

[11] https://ultralytics.com/yolov5

[12] https://pypi.org/project/opencv-python/

[13] https://numpy.org

[14] https://google.github.io/mediapipe/solutions/hands.html

[15] https://www.tensorflow.org

[16] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/SGD

[17] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adagrad

[18] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/RMSprop

[19] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam

[20] https://python-pillow.org/

[21] https://matplotlib.org/

[22] https://pytorch.org/

[23] https://github.com/pndurette/gTTS

ÖZGEÇMİŞ

Ad Soyad	Samet TOPRAK
Doğum yeri	Edirne
Lise	2014-2018
Staj Yaptığı Yerler	-

Ad Soyad	Caner AKIN
Doğum yeri	İstanbul
Lise	2014-2018
Staj Yaptığı Yerler	TOMYA Teknoloji Şirketi