



Teknoloji Fakültesi

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Türk İşaret Dili Tanıma

ve Metne Dönüştürme:

İngilizce Çeviri Destekli

Uygulama

BİTİRME PROJESİ

RAPORU

Bilgisayar Mühendisliği Bölümü

DANIŞMAN

Prof. Dr. Serhat ÖZEKES

İSTANBUL, 2025

MARMARA ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Öğrencileri Ayça Nilay ÖZOĞUL, Merve HASASLARI, Serra K.TALASLI tarafından “Türk İşaret Dili Tanıma ve Metne Dönüşürme: İngilizce Çeviri Destekli Uygulama” başlıklı proje çalışması, 19.06.2025 tarihinde savunulmuş ve jüri üyeleri tarafından başarılı bulunmuştur.

Jüri Üyeleri

Prof. Dr. Serhat ÖZEKES
Marmara Üniversitesi
Dr. Öğr. Üyesi Timur İNAN
Marmara Üniversitesi
Arş. Gör. Merve PINAR
Marmara Üniversitesi

(Danışman)

(İMZА).....

(Üye)

(İMZА).....

(Üye)

(İMZА).....

ÖNSÖZ

Proje çalışmamız süresince karşılaştığım bütün problemlerde, sabırla yardım ve bilgilerini esirgemeyen, tüm desteğini sonuna kadar yanında hissettiğim değerli hocam, sayın Prof. Dr. Serhat ÖZEKES'e en içten teşekkürlerimi sunarım.

Bu proje çalışması fikrinin oluşması ve ortaya çıkışmasındaki önerisi ve desteginden dolayı değerli hocam Prof. Dr. Serhat ÖZEKES'e teşekkür ederim.

Proje çalışmam sırasında maddi ve manevi desteklerini esirgemeyen okul içerisinde ve okul dışında her zaman yanında olan değerli çalışma arkadaşlarım ve değerli hocam Prof. Dr. Serhat ÖZEKES'e sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

- 1. GİRİŞ**
 - 1.1. Proje Çalışmasının Amacı ve Önemi**
- 2. LİTERATÜR TARAMASI**
 - 2.1. Türk İşaret Dili Tanıma Performansının İyileştirilmesi İçin Transfer Öğrenme Yaklaşımı**
 - 2.2. Türk İşaret Dili Tanımada Video Transformer Network Kullanımı**
 - 2.3. Convolutional Neural Network ve LSTM ile Türk İşaret Dili Tanıma**
 - 2.4. İki Akışlı S3D ile Mobil Tabanlı İşaret Dili Tanıma**
 - 2.5. BosphorusSign22k Veri Seti ile Türk İşaret Dili Tanıma**
 - 2.6. Türk İşaret Dili Tanımada Çok Modlu Veri Seti Kullanımı**
- 3. METOT VE YÖNTEMLERİ**
 - 3.1. Veri Seti**
 - 3.2. Veri Ön İşleme Adımları**
 - 3.3. İskelet Çıkarımı Yöntemleri**
 - 3.4. Mediapipe ile İskelet Çıkarımı**
 - 3.5. Model Mimarisi**
 - 3.5.1. Long Short-Term Memory (LSTM)**
 - 3.5.2. Model Birleşimi**
 - 3.5.3. Eğitim ve Test Aşamaları**
- 4. BULGULAR VE TARTIŞMA**
 - 4.1. Veri Seti Özellikleri**
 - 4.2. Veri Ön İşleme**
 - 4.3. İskelet Çıkarımı**
 - 4.4. Model Mimarisi**
- 5. KULLANILAN TEKNOLOJİLER**
- 6. PROJE AKIŞI**
 - 6.1. Kullanıcı Arayüzü (Frontend)**

6.2. Sunucu ve Model Entegrasyonu (Backend)

6.3. Şifre Sıfırlama ve E-posta Entegrasyonu

7. GERÇEKLEŞTİRİLEN TEKNİK ÇALIŞMALAR

8. SİSTEM GEREKSİNİMLERİ

9. SONUÇ VE TARTIŞMA

KAYNAKLAR

ÖZET

Türk İşaret Dili Tanıma ve Metne Dönüştürme: İngilizce Çeviri Destekli Uygulama

Bu proje çalışmasında, Türk İşaret Dili (TİD) kullanılarak işaret dili tanıma ve metne çevirme işlemleri gerçekleştirilmiştir. Geliştirilen sistem, gerçek zamanlı olarak işaret dili hareketlerini algılayarak bu hareketleri metin formatına dönüştürme yeteneğine sahiptir. Amacı, işaret dili kullanıcıları ile Türkçe bilmeyenler arasında iletişimini kolaylaştırmaktır.

Metne dönüştürülen işaret dili ifadeleri, ardından İngilizce'ye çevrilecektir. Sonuç olarak, işaret dili tanıma ve çeviri süreçlerinin daha hızlı ve doğru bir şekilde gerçekleştirilmesi sağlanarak, işaret dili kullanıcılarının günlük yaşamlarında daha fazla erişilebilirlik sunulacaktır.

Haziran, 2025

Öğrenciler

ABSTRACT

Turkish Sign Language Recognition and Text Conversion: English Translation Supported Application

In this study, sign language, which is an essential communication tool for individuals with hearing and speech impairments, is examined in detail. Sign language is not just a visual communication method but a complex system that includes hand shapes, movements, orientations, speed, facial expressions, body posture, and gestures. These elements work together to convey meaning and context, forming a unique linguistic structure. However, the lack of widespread knowledge and proficiency in sign language creates significant challenges for hearing-impaired individuals, particularly in accessing education, healthcare, employment, and social interaction.

Linguistic studies have revealed that sign languages exhibit unique characteristics that differ from spoken languages and vary between countries and cultures. For example, Turkish Sign Language (TİD), which is commonly used by hearing-impaired individuals in Turkey, has its own grammatical structure distinct from both spoken Turkish and other sign languages worldwide. This diversity among sign languages often results in communication difficulties, emphasizing the need for systems capable of automatically translating between different sign languages.

Recent advancements in artificial intelligence (AI) and deep learning (DL) have introduced new opportunities for the automatic recognition and translation of sign languages. Modern sign language recognition systems utilize computer vision, image processing, and natural language processing (NLP) techniques to analyze hand movements and facial expressions, converting them into written or spoken text. These systems hold significant potential to eliminate communication barriers faced by hearing-impaired individuals, thus promoting social inclusion.

Various approaches have been developed to recognize sign languages automatically, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformer models. Despite these efforts, challenges remain in capturing spatial

and temporal features simultaneously and accurately detecting subtle variations in hand shapes and facial expressions. Addressing these challenges requires robust and well-generalized models that can operate efficiently in real-world scenarios.

The primary goal of this research is to develop an innovative model for automatic recognition of Turkish Sign Language (TİD) using advanced deep learning techniques. The study aims to achieve high recognition accuracy while maintaining real-time performance, allowing effective interaction between hearing-impaired individuals and digital platforms. The model will integrate advanced methods such as 3D convolutional networks, attention mechanisms, and multimodal data fusion to accurately capture the spatial and temporal aspects of TİD gestures. Additionally, the use of transfer learning and data augmentation techniques will enhance the model's generalization and reduce overfitting.

Furthermore, the project will include an English translation feature, allowing recognized Turkish Sign Language to be converted into English text. This feature aims to bridge communication between Turkish sign language users and English-speaking individuals, promoting cross-linguistic accessibility.

The expected outcome of this research is to develop a robust and efficient application named "*Turkish Sign Language Recognition and Text Conversion: English Translation Supported Application*". This system will be applicable in various real-world contexts, such as public services, healthcare, education, and social media platforms. By minimizing the need for human interpreters, the system aims to empower hearing-impaired individuals to communicate more freely and independently, promoting social inclusion and equality.

June, 2025

Students

KISALTMALAR

CUDA : Compute Unified Device Architecture

GPU : Graphical Processing Unit

CPU : Central Processing Unit

AUTSL : Large Scale Dataset for Turkish Sign Language

YOLOv8 : You Only Look Once versiyon 8

CNN : Konvolüsyonel Sinir Ağı (Convolutional Neural Network)

LSTM : Uzun Kısa Süreli Bellek (Long Short-Term Memory)

Mediapipe : Google tarafından geliştirilen bir makine öğrenimi çözüm paketi

NLP : Neuro Linguistic Programming

SL-GCN: Spatial-Temporal Graph Convolutional Network, işaret dili tanıma gibi zaman serisi verilerinde uzamsal ve zamansal ilişkileri öğrenmek için kullanılan bir derin öğrenme modeli.

DANN: Domain-Adversarial Neural Network, farklı veri kümeleri (kaynak ve hedef alanlar) arasında öğrenmeyi genelleştirmek için kullanılan bir adaptasyon tekniği.

DSBN: Domain-Specific Batch Normalization, farklı veri dağılımlarına sahip veri kümeleri için ayrı ayrı normalizasyon yapan bir yöntem.

JAN: Joint Adaptation Network, özellik dağılımlarını hizalayarak transfer öğrenmeyi iyileştiren bir yöntem.

MCC: Minimum Class Confusion, sınıf belirsizliğini en aza indirgeyerek modelin daha iyi genelleşmesini sağlayan bir yöntem.

RGB: Red, Green, Blue. Görüntü ve video verilerinde kullanılan temel renk uzayı.

BLSTM: Bidirectional Long Short-Term Memory, dizisel veriler için çift yönlü bilgi işleyebilen gelişmiş bir RNN modeli.

VTN (Video Transformer Network): Video tabanlı görevlerde uzun vadeli bağımlılıkları modellemek için Transformer mimarisini kullanan bir ağ türü.

S3D: Separable 3D Convolution, 3D evrişimleri ayırtırarak hesaplama maliyetini düşüren bir video analiz yöntemi.

IDT (Improved Dense Trajectories): Hareket tabanlı özellik çıkarımı yapan bir video analizi yöntemi.

3D Residual Networks (MC3): 2D ve 3D evrişimleri birleştirerek hareket bilgisini yakalayan bir sinir ağı modeli.

GCN (Graph Convolutional Network): Grafik yapılarındaki verileri işlemek için kullanılan bir sinir ağı modeli.

I3D (Inflated 3D ConvNet): 2D konvolüsyonları 3D'ye genişleterek video tabanlı derin öğrenme modellerini geliştiren bir yöntem.

CSL (Chinese Sign Language Dataset): Çince İşaret Dili tanıma için kullanılan büyük ölçekli bir veri seti.

ŞEKİL LİSTESİ

Şekil 3.2.1 Durağan Kare Temizliği	15
Şekil 3.2.2 Çözünürlük Ölçeklemesi	16
Şekil 3.2.3 Bozuk Video Tespiti	16
Şekil 3.2.4 Kısa Video Tespiti	17
Şekil 3.2.5 Düşük Çözünürlüklü Video Tespiti	17
Şekil 3.3.1 Yolo algoritmasının görselleştirilmesi	18
Şekil 3.3.2 MediaPipe algoritmasının görselleştirilmesi	20
Şekil 3.4.1 Zaman Serisi Yapısı	21
Şekil 3.5.1 Girdi Özellikleri	23
Şekil 3.5.2 LSTM Katmanları	24
Şekil 3.5.3 Havuzlama ve Tam Bağlantılı Katman	25
Şekil 3.5.4 Çıkış Katman	25
Şekil 3.5.5 Hiperparametre Optimizasyonu (Grid Search)	25
Şekil 6.1.1 Başlangıç Sayfası	29
Şekil 6.1.2 Ana Sayfa	29
Şekil 6.1.3 Giriş Sayfası	30
Şekil 6.1.4 Kayıt Sayfası	30
Şekil 6.1.5 İşlem Sayfası	31
Şekil 6.1.6 Responsive Tasarım	31
Şekil 6.3 Şifre Yenileme Maili	33

TABLO LİSTESİ

Tablo 3.1 Veri setinin özelliklerı 13

Tablo 8 Sistem Gereksinimleri 35

1. GİRİŞ

İşaret dili, duyma ve konuşma engelli insanların dış dünyayla iletişim kurması için en önemli araçlardan biridir. Coğunlukla görsel bir iletişim tarzı olarak değerlendirilse de işaret dili yalnızca el hareketlerinden ibaret değildir; el yönleri ve hızları, yüz ifadeleri, postür ve mimikler gibi çeşitli komponentler, bu dili oluşturan temel unsurlar arasında bulunmaktadır [1]. İşaret dilini etkin şekilde kullanabilen birey sayısının kısıtlı olması, işitme engelli toplulukların eğitim, istihdam, sağlık hizmetleri ve sosyal etkileşim gibi temel sahalarda dezavantajlı konuma düşmelerine sebep olmaktadır (World Federation of the Deaf, 2023).

Dilbilimsel çalışmalar, her dilin kendine has bir işaret dili olduğunu ve bu dillerin de kendine özgü fonolojik, morfolojik ve sentaktik özelliklere sahip olduğunu göstermektedir [3]. Türk İşaret Dili (TİD), Türkiye'deki işitme engelli bireylerin anadilidir ve diğer işaret dillerinden farklı bir dilbilgisi yapısına sahiptir [4]. Bu durum, işitme ve konuşma engelli insanların farklı diller konuşulan ülkelere seyahat etmeleri veya farklı kültürlerden bireylerle iletişim kurmaları sırasında zorluklarla karşılaşmalarına yol açmaktadır. Küresel bir işaret dili ölçütünün olmaması, yurt dışından gelen işitme engelli bireyler arasındaki iletişimini oldukça zorlaştırmaktadır. Bu bağlamda, işaret dilleri arasında otomatik çeviri yapabilecek teknolojilere duyulan gereksinim zamanla artmaktadır [5].

Son yıllarda yapay zekâ (YZ) ve derin öğrenme (DL) temelli teknolojiler, işaret dili tanıma sistemlerinin geliştirilmesinde önemli bir gelişme kaydetmiştir. Gelişmiş görüntü işleme ve doğal dil işleme yöntemleri sayesinde, kameralar aracılığıyla kaydedilen işaret dili hareketleri otomatik olarak analiz edilerek metne veya farklı dillere çevrilebilmektedir [6]. Bu tür teknolojiler, işitme engelli bireylerin iletişim duvarlarını aşmalarına, bilgiye ulaşımlarının kolaylaştırılmasına ve toplumsal yaşama daha aktif katılımlarına imkan sağlamaktadır [7].

Bu çalışmada, Türk İşaret Dili'nin (TİD) otomatik tanınmasını sağlamak amacıyla son teknoloji derin öğrenme yaklaşımlarının incelenmesi ile daha yüksek doğruluk oranına ve kullanıcı dostu yapıya sahip bir model geliştirilmesi hedeflenmektedir. Modelin, kamu ve özel sektör işlemlerinde işaret dili tercümanı ihtiyacını azaltarak işitme engelli bireylerin sosyal ve profesyonel hayatı daha aktif katılım sağlama amacını gerçekleştirmektedir. Bu

sayede, işaret dili ile iletişimim evrensel ölçekte daha erişilebilir duruma getirilmesi ve işitme engelli bireylerin uluslararası etkileşimlerinin kolaylaştırılması hedeflenmektedir.

2. LİTERATÜR TARAMASI

2. 1 Türk İşaret Dili Tanıma Performansının İyileştirilmesi İçin Transfer Öğrenme Yaklaşımı

Türk İşaret Dili tanımda kısıtlı veri setlerinin başarısını artırmak amacıyla transfer öğrenme yöntemleri uygulanmıştır. AUTSL ve BSign22k veri setleri birleştirilerek ortak bir çalışma alanı oluşturulmuştur. Çalışmada SL-GCN, DANN, DSBN, JAN ve MCC gibi modeller kullanılmıştır. Karma yöntemlerle %98,8 doğruluk oranına ulaşılmıştır [8].

2. 2 Türk İşaret Dili Tanımada Video Transformer Network Kullanımı

AUTSL veri seti üzerinde VTN modeli kullanılmıştır. İnsan vücut noktaları ve el hareketleri gibi ek bilgilerle model doğruluğu %92,92'ye ulaşmıştır. RGB verisi kullanıldığından ise doğruluk oranı %82 olarak hesaplanmıştır [9].

2. 3. Convolutional Neural Network ve LSTM ile Türk İşaret Dili Tanıma

CNN ve LSTM modelleri, geniş çaplı AUTSL veri setinde kullanılmıştır. BLSTM ve Attention mekanizmaları ile %95,95 doğruluk oranına ulaşılmıştır. Kullanıcıdan bağımsız testlerde doğruluk %62,02 olarak elde edilmiştir [10].

2. 4. İki Akışlı S3D ile Mobil Tabanlı İşaret Dili Tanıma

RGB ve iskelet verilerini kullanan İki Akışlı S3D modeli ile AUTSL veri setinde %93,75 doğruluk oranına ulaşılmıştır. LSA64 veri setinde ise %99,38 doğruluk elde edilmiştir [11].

2. 5 BosphorusSign22k Veri Seti ile Türk İşaret Dili Tanıma

BosphorusSign22k veri seti, 744 işaret içermekte olup Microsoft Kinect v2 ile kaydedilmiştir. IDT ve MC3 modelleriyle doğruluklar sırasıyla %88,53 ve %78,85 olarak rapor edilmiştir [12].

2.6 Türk İşaret Dili Tanımada Çok Modlu Veri Seti Kullanımı

AUTSL veri seti üzerinde GCN, I3D ve BLSTM gibi modellerle %98,42 (RGB Track) ve %98,53 (RGB+D Track) doğruluk oranlarına ulaşılmıştır [13].

3. METOT VE YÖNTEMLERİ

3. 1 Veri Seti

Bu çalışmada, Türk İşaret Dili (TİD) tanıma üzerine yapılan analizlerde AUTSL veri seti kullanılmaktadır. AUTSL, Türk İşaret Dili için hazırlanmış kapsamlı bir veri setidir. İçerisinde 226 farklı işaret bulunmakta olup toplamda 38.336 örnek içermektedir. Bu veri seti, 43 farklı kişinin gerçekleştirdiği işaretlerden oluşmakta olup, her işaret için ortalama 170 örnekle geniş bir çeşitlilik sunmaktadır. AUTSL, içeriği örnek sayısı açısından, CSL veri setinden sonra en büyük ikinci veri seti olma özelliğine sahiptir [14].

Property	Description
Number of signs	226
Number of signers	43
Total samples	38,336
Number of different backgrounds	20
Mean sample per sign	169.6
RGB and depth resolution	512x512
FPS	30

Tablo 3.1 Veri setinin özellikleri

AUTSL'yi diğer büyük ölçekli işaret dili veri setlerinden ayıran en önemli özelliklerden biri, 20 farklı arka plan içermesi ve çeşitli zorlukları barındırmasıdır. Veri setindeki videolar, hem iç hem de dış mekânlarda, farklı ışıklandırma koşullarında çekilmiş olup, gerçek hayat senaryolarına uygun şekilde hazırlanmıştır. Özellikle dış mekân çekimlerinde, rüzgârla hareket eden nesneler, arka planda geçen insanlar ve değişken ışık koşulları gibi dinamik unsurlar yer almaktadır. Bu sayede, geliştirilen modellerin gerçek dünya koşullarına daha iyi uyum sağlaması ve genelleme yapabilmesi hedeflenmektedir [14].

Veri seti oluşturulurken, günlük hayatı en sık kullanılan işaretlerin seçilmesine özellikle dikkat edilmiştir. Ayrıca, el hareketlerinin çeşitliliğini dengeli bir şekilde dağıtmak ve birbirine benzeyen ancak anlamı farklı olan işaretleri de dahil etmek amacıyla titiz bir çalışma yürütülmüştür. Bu süreçte, Türk İşaret Dili eğitmenleriyle iş birliği yapılarak işaretlerin kapsamlı ve temsil edici olmasına

özen gösterilmiştir. Seçilen işaretler, el şekilleri ve hareket dinamikleri açısından geniş bir yelpazeyi kapsamaktadır. Modelin farklı varyasyonları öğrenebilmesini sağlamak amacıyla çaprazlaşdırılmıştır.

Bazı işaretlerde eller birbirini veya yüzü kapatabilmektedir, örneğin:

Ellerin birbirini kapattığı işaretler: *Ayakkabı, Bal...*

Ellerin yüzü kapattığı işaretler: *Beklemek, Üzgün...*

Bazı işaretlerde eller derinlik yönünde hareket etmektedir, örneğin:

İtme, Terzi...

Bazı işaretlerde sağ ve sol el çapraz pozisyonda bulunmaktadır, örneğin:

Yardım, Tehlike...

Ayrıca, veri setinde bileşik işaretler de yer almaktadır. Örneğin, *Hastane* işareteti, *Doktor* ve *Bina* işaretlerinin sıralı ve ardışık şekilde yapılmasıyla oluşmaktadır. Benzer şekilde, *Yemek* ve *Ocak* işaretleri ile oluşturulan *Yemek pişirmek* bileşik işareteti de veri setine dahil edilmiştir.

AUTSL veri setinde yer alan 43 işaretleyici, farklı deneyim düzeyine ve profillere sahiptir:

- 6 kişi Türk İşaret Dili eğitmeni
- 3 kişi Türk İşaret Dili tercümanı
- 1 kişi işitme engelli birey
- 1 kişi işitme engelli ebeveynlerin çocuğu (Coda - Children of Deaf Adults)
- 25 kişi Türk İşaret Dili kursiyeri
- 7 kişi veri setindeki işaretleri öğrenmiş eğitimli işaretleyiciler

İşaretleyicilerin 10'u erkek, 33'ü kadın olup 2'si solaktır. Veri seti, işaretlerin dağılımı açısından dengeli bir yapıya sahiptir. Ancak, bazı işaretleyicilerin örnek sayısı diğerlerine kıyasla daha fazla olabilmektedir. Bunun temel nedeni, aynı işaretleyicilerin farklı kıyafetlerle veya değişen arka planlarda birden fazla kez kayıt altına alınmış olmasıdır. Bu çeşitlilik, modelin farklı ortam ve koşullara daha iyi uyum sağlayabilmesi için bilinçli olarak eklenmiştir [14].

Sonuç olarak, AUTSL veri seti, geniş kapsamı, farklı arka plan çeşitliliği, dinamik çevresel koşulları ve dengeli işaret dağılımı ile Türk İşaret Dili tanıma alanında önemli bir kaynak niteliği taşımaktadır. Bu çalışmada, modelin eğitimi ve değerlendirilmesi süreçlerinde bu veri seti etkin bir şekilde kullanılmıştır.

3. 2 Veri Ön İşleme Adımları

Bu çalışmada, Türk İşaret Dili tanıma amacıyla kullanılan AUTSL veri setine ait video verileri belirli ön işleme adımlarından geçirilerek analiz için uygun hale getirilmiştir. Veri setindeki videoların işlenmesi sürecinde aşağıda sıralanan adımlar takip edilmiştir:

1. Videoların Başında ve Sonunda Boş veya Durağan Karelerin Tespiti ve Kaldırılması:

Veri setindeki videoların başlangıç ve bitiş kısımlarında yer alan boş veya durağan kareler, ardışık kareler arasındaki farklar analiz edilerek tespit edilmiştir. Bu işlemde, OpenCV kütüphanesinin `cv2.absdiff()` fonksiyonu kullanılmış ve belirlenen eşik değerinin altında kalan kareler otomatik olarak kaldırılmıştır. Böylece, videoların yalnızca anlamlı hareketler içeren bölümleri işleme alınmıştır. Böylece, yalnızca işaret dili hareketlerini içeren bölümler veri setinde tutulmuştur.

```
def remove_static_frames(video_path, output_path, threshold=30):
    """Videoanın başında ve sonundaki statik frameleri kaldırır."""
    cap = cv2.VideoCapture(video_path)
    fps = int(cap.get(cv2.CAP_PROP_FPS))
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

    # Video writer oluştur
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

    frames = []
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        frames.append(frame)

    # Baştan ve sondan statik frameleri tespit et
    start_idx = 0
    end_idx = len(frames) - 1

    # Baştan statik frame kontrolü
    for i in range(len(frames)-1):
        diff = cv2.absdiff(frames[i], frames[i+1])
        if np.mean(diff) > threshold:
            start_idx = i
            break

    # Sondan statik frame kontrolü
    for i in range(len(frames)-1, 0, -1):
        diff = cv2.absdiff(frames[i], frames[i-1])
        if np.mean(diff) > threshold:
            end_idx = i
            break

    # Statik olmayan frameleri yaz
    for frame in frames[start_idx:end_idx+1]:
        out.write(frame)

    cap.release()
    out.release()
```

Şekil 3.2.1 Durağan Kare Temizliği

2. Tüm Karelerin Aynı Çözünürlüğe Ölçeklenmesi:

Veri setindeki videoların farklı çözünürlüklerde sahip olması, modelin eğitimi sırasında tutarsızlıklara yol açabileceğiinden tüm kareler belirli bir standart çözünürlüğe uyumlu hale getirilmiştir. Bu çalışmada, tüm videolar

224x224 piksel boyutlarına yeniden ölçeklendirilerek veri kümesinin homojen bir giriş formatına sahip olması sağlanmıştır. Bu işlem, OpenCV kütüphanesinin **cv2.resize()** fonksiyonu kullanılarak gerçekleştirilmiştir.

```
def resize_video(video_path, output_path, target_size=(224, 224)):
    """Videoyu hedef çözünürlüğe ölçekler."""
    cap = cv2.VideoCapture(video_path)
    fps = int(cap.get(cv2.CAP_PROP_FPS))

    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    out = cv2.VideoWriter(output_path, fourcc, fps, target_size)

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        resized_frame = cv2.resize(frame, target_size)
        out.write(resized_frame)

    cap.release()
    out.release()
```

Şekil 3.2.2 Çözünürlük Ölçeklemesi

3. **Bozuk veya Eksik Videoların Tespit Edilmesi:** Veri setinde bazı videolarda eksik kare veya bozuk içerik bulunabileceği göz önüne alınarak tüm videolar **cv2.VideoCapture** yöntemiyle işleme tabi tutulmuş ve düzgün şekilde okunamayan dosyalar tespit edilerek veri kümesinden çıkarılmıştır.

```
def check_video_integrity(video_path):
    """Videoların bütünlüğünü kontrol eder."""
    try:
        cap = cv2.VideoCapture(video_path)
        if not cap.isOpened():
            return False
        ret, frame = cap.read()
        if not ret or frame is None:
            return False
        cap.release()
        return True
    except:
        return False
```

Şekil 3.2.3 Bozuk Video Tespiti

4. **Sıfır Süreli veya Çok Kısa Videoların Filtrelenmesi:** Veri setinde yer alan çok kısa süreli videolar, modelin yeterli bilgiyle eğitilmesini

zorlaştıracak olan filtrelerden滤除后会被丢弃。因此，每一段视频的持续时间，**cv2.CAP_PROP_FPS** 和 **cv2.CAP_PROP_FRAME_COUNT** 将被用来计算出一个预设的阈值（例如1秒），在该阈值之下剩余的数据将被丢弃。这样一来，只有那些包含有价值信息的视频帧将被处理，从而使得模型能够更有效地学习。

```
def check_video_duration(video_path, min_duration=1.0):
    """Video süresini kontrol eder."""
    cap = cv2.VideoCapture(video_path)
    fps = cap.get(cv2.CAP_PROP_FPS)
    frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    duration = frame_count / fps
    cap.release()
    return duration >= min_duration
```

Şekil 3.2.4 Kısa Video Tespiti

5. Düşük Çözünürlüklü veya Bozuk Görüntülü Videoların

Ayıklanması: Düşük çözünürlüklü veya görüntü kalitesi yetersiz olan videolar, işaret dili hareketlerinin doğru şekilde analiz edilebilmesi için veri setinden çıkarılmıştır. Bu amaçla, videoların ilk kareleri incelenmiş ve belirlenen minimum çözünürlük (**224x224 piksel**) altında kalanlar ayıklanmıştır. Ayrıca, bozuk kareler tespit edilerek bu videoların işlenmesi önlenmiş ve yalnızca net ve kaliteli görüntü içeren videoların modele dahil edilmesi sağlanmıştır.

```
def check_frame_quality(video_path, min_resolution=(224, 224)):
    """Frame kalitesini ve çözünürlüğünü kontrol eder."""
    cap = cv2.VideoCapture(video_path)
    ret, frame = cap.read()
    if not ret:
        cap.release()
        return False

    height, width = frame.shape[:2]
    if height < min_resolution[0] or width < min_resolution[1]:
        cap.release()
        return False

    # Bozuk frame kontrolü
    while ret:
        if frame is None or frame.size == 0:
            cap.release()
            return False
        ret, frame = cap.read()

    cap.release()
    return True
```

Şekil 3.2.5 Düşük Çözünürlüklü Video Tespiti

Yukarıda belirtilen veri ön işleme adımları sayesinde, AUTSL veri setindeki verilerin kalite ve tutarlılığı önemli ölçüde artırılmıştır. Bu işlemler sonucunda elde edilen temiz ve standart hale getirilmiş veri kümesi, işaret dili tanıma modelinin daha güvenilir ve verimli bir şekilde eğitilmesine olanak tanımaktadır.

3. 3 İskelet Çıkarımı Yöntemleri

İskelet çıkarma işlemi için öncelikle videolar karelere (frame) ayrılmaktadır. Çünkü iskelet çıkarma işlemi, videolar yerine tekil kareler üzerinde çalışıldığı zaman daha doğru sonuçlar vermektedir. Bunun nedenleri şunlardır:

- **Anlık Tespit:** Video yerine her kareyi ayrı ayrı işleyerek, hareket bulanıklığını ve modelin sürekli değişen pozisyonlara uyma zorunluluğunu ortadan kaldırır.
- **Bağımsız İşleme:** Tekil kareler üzerinde yapılan analizler, ardışık kareler arasındaki olası hata durumunu minimize eder.
- **Paralel İşlemeye Uygunluk:** Kare bazlı işlemde, GPU veya çoklu iş parçacığı kullanımıyla hızlandırılabilir.

Bu durumda iskelet çıkarma işlemi için farklı yöntemler denenmiş ve aşağıdaki sonuçlara ulaşılmıştır:

YOLO ile İskelet Çıkarımı

YOLO (You Only Look Once), nesne tespiti için kullanılan bir derin öğrenme modelidir. İşaret dili tanıma sürecinde, el hareketlerini algılamak için YOLO'dan faydalananabileceğini düşünülmüştür. Ancak:

- YOLO bütün vücuda odaklanırken, el ve parmak detayları yeterince iyi tespit edilmemiştir.
- İşaret dili için parmak hareketleri kritik öneme sahip iken, bu yöntemin kullanımı uygun görülmemektedir.



Şekil 3.3.1 Yolo Algoritmasının Görüntülenmesi

OpenPose ile İskelet Çıkarımı

OpenPose; yüz, vücut ve ellerin iskelet noktalarını çıkarabilen gelişmiş bir kütüphanedir. İşaret dili tanıma sürecinde kullanımı açısından avantajlı durumları şunlardır:

- Tüm vücut hareketlerini ve parmak detaylarını tespiti.
- İskelet bazlı hareket analizlerinde yüksek doğruluk sunmak.

Fakat, OpenPose'un kullanımı sırasında şu zorluklarla meydana gelmektedir:

- Kurulum Süreci Karmaşıktır: CUDA, cuDNN, CMake, Visual Studio gibi ek yazılımlar gerektirmektedir.
- Sürüm Uyumsuzlukları: Çeşitli donanım ve yazılım kombinasyonlarında uyumluluk sorunları yaşanmaktadır.
- Zaman Kaybı: Uzun kurulum ve yapılandırma süreçleri nedeni ile pratik kullanım açısından verimli bulunmamaktadır.

Bu nedenlerden OpenPose'un kullanımı tercih edilmemektedir.

MediaPipe Hands ve MediaPipe Body ile İskelet Çıkarımı

MediaPipe, Google tarafından geliştirilen, hafif ve GPU destekli bir kütüphanedir. İşaret dili tanıma sürecinde vücut ve el hareketleri birlikte önemli olduğundan, MediaPipe'in Hands (Eller) ve Body (Vücut) modülleri birlikte kullanılmaktadır.

Neden MediaPipe Tercih Edildi?

- Kurulumu Kolaydır: Python kütüphanesi olarak hızlıca entegre edilebilmektedir.
- GPU Desteği Sunar: Gerçek zamanlı analiz için hız avantajı sağlamaktadır.
- El ve Parmakları Detaylı Tespiti: Her el için 21 anahtar nokta belirler ve bu noktaların x, y, z koordinatlarını JSON formatında kaydetmek mümkündür.



Şekil 3.3.2 MediaPipe Algoritmasının Görselleştirilmesi

Veri Kaydı ve Kullanımı

- Frame Bazlı İşleme: Videolar belirli adımlarla karelere ayrıldıktan sonra, her kare için iskelet verileri çıkarılmıştır.
- Paralel İşleme: Çoklu iş parçacığı kullanılarak hızlandırılmıştır.
- Analiz İçin JSON Formatı: Elde edilen koordinatlar, işaret dili tanıma modeline giriş olarak verilmek üzere JSON formatı ile saklanmıştır.

Denenen bu yöntemler arasında MediaPipe Hands ve MediaPipe Body, kurulum kolaylığı, gerçek zamanlı çalışabilmesi ve detaylı el takibi sunması nedeniyle işaret dili tanıma sürecinde en uygun yöntem olarak seçilmiştir.

3. 4 Mediapipe ile İskelet Çıkarımı

1. Koordinatların Elde Edilmesi:

Videolardan elde edilen her bir karedede, MediPipe yardımıyla elin 21 anahtar noktası (landmark) saptanmıştır. Bu noktalar X, Y ve Z koordinat bilgilerini içermektedir.

- X ve Y koordinatları elin konumunu belirtir.
- Z koordinatı, derinlik bilgisini içerir ve elin kameraya olan uzaklığını belirtir.

2. Koordinatların Normalizasyonu:

Elde edilen koordinatlar, her kullanıcının el yapısı ve pozisyonuna göre değişiklik gösterebileğinden, normalizasyon(standartlaştırma) işlemi uygulanmıştır.

- Min-Max Normalizasyonu: Koordinatlar [0,1] aralığına ölçeklendirilmiştir.
- Merkezleme: Elin merkezi (bilek noktası) referans alınarak koordinatlar yeniden belirlenmiştir.

3. Zaman Serisi Yapısının Elde Edilmesi:

Videolardan çıkarılan her frame(kare), bir zaman serisi oluşturacak şekilde sıraya alınmıştır. Ardışık karelerden elde edilen koordinat verileri, modelin zamansal tutarlılığı öğrenebilmesi için birleştirilmiştir.

```
def process_image(image_path, pose, hands):
    try:
        image = cv2.imread(image_path)
        if image is None:
            raise Exception("Görüntü okunamadı")
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        # Poz ve el algılama
        pose_results = pose.process(image_rgb)
        hands_results = hands.process(image_rgb)

        if not pose_results.pose_landmarks and not hands_results.multi_hand_landmarks:
            raise Exception("İskelet ve eller tespit edilemedi")

        # Görüntüye çizimler ekleme
        annotated_image = image.copy()

        # Vücut iskeleti çizimi
        if pose_results.pose_landmarks:
            mp_drawing.draw_landmarks(
                annotated_image, pose_results.pose_landmarks,
                mp_pose.POSE_CONNECTIONS,
                landmark_drawing_spec=mp_drawing.DrawingSpec(
                    color=(0, 255, 0), thickness=2, circle_radius=2)
            )

        # El iskeleti çizimi (farklı renkte)
        if hands_results.multi_hand_landmarks:
            for hand_landmarks in hands_results.multi_hand_landmarks:
                mp_drawing.draw_landmarks(
                    annotated_image, hand_landmarks, mp_hands.HAND_CONNECTIONS,
                    landmark_drawing_spec=mp_drawing.DrawingSpec(
                        color=(255, 0, 0), thickness=2, circle_radius=2)
                )

    return annotated_image, True

except Exception as e:
    return None, False
```

Şekil 3.4.1 Zaman Serisi Yapısı

3. 5 Model Mimarisi

Bu model, el hareketlerinin zamansal analizine odaklanan iki katmanlı çift yönlü LSTM (Bidirectional LSTM) yapısı üzerine kurulmuştur. Görsel özellik çıkarımı doğrudan modelin giriş verisinde hazır olarak sağlandığı varsayılmakta, bu nedenle CNN yerine, önceden çıkarılmış görsel özellikler LSTM'e doğrudan beslenmektedir.

Bidirectional LSTM (BiLSTM) Nedir?

Uzun Kısa Süreli Bellek (Long Short-Term Memory, LSTM) ağları, geleneksel Tekrarlayan Sinir Ağlarının (RNN) zaman içindeki uzun bağımlılıkları öğrenme konusundaki zorluklarını aşmak için geliştirilmiştir [15]. LSTM hücresi, her zaman adımda bilgi akışını kontrol etmek için üç temel kapıdan oluşur: giriş kapısı it , unutma kapısı (ft) ve çıkış kapısı (ot). Bu kapılar sayesinde hücre, hangi bilgilerin tutulup hangilerinin unutulacağını öğrenebilir. LSTM'nin matematiksel formülasyonu aşağıdaki gibidir [16]:

1. Giriş Kapısı:

$$it = \sigma(Wixt + Uih - 1 + bi)$$

2. Unutma Kapısı:

$$ft = \sigma(Wfxt + Ufh - 1 + bf)$$

3. Çıkış Kapısı:

$$ot = \sigma(Woxt + Uoh - 1 + bo)$$

4. Aday Hücre Durumu:

$$\tilde{C}_t = \tanh(Wcxt + Uch - 1 + bc)$$

5. Hücre Durumu Güncellemesi:

$$C_t = ft \odot C_{t-1} + it \odot \tilde{C}_t$$

6. Gizli Durum (Çıktı):

$$ht = ot \odot \tanh(Ct)$$

Burada:

- xt modelin t . zaman adımındaki girişi,
- $ht - 1$ bir önceki gizli durum,
- Ct hücre durumu,
- $W^*, U^*U_-^* U^*, b^*b_-^* b^*$ ağırlık ve bias matrisleri,
- σ sigmoid aktivasyon fonksiyonu,
- \tanh hiperbolik tanjant fonksiyonu,
- \odot ise eleman bazlı çarpımı (Hadamard çarpımı) ifade eder.

Bidirectional LSTM (BiLSTM) ise bu yapıyı iki yönlü genişleterek hem geçmişten geleceğe ($ht \rightarrow$) hem de gelecekten geçmişe ($ht \leftarrow$) olacak şekilde çalıştırır. Her zaman adımında elde edilen ileri ve geri çıktılar birleştirilir:

$$ht^{\{BiLSTM\}} = [ht \rightarrow; ht \leftarrow]$$

Bu yaklaşım sayesinde model, yalnızca geçmiş bağlamı değil, aynı zamanda gelecekteki bağlamı da dikkate alarak her zaman adımında daha kapsamlı bir temsil oluşturabilir [17]. Özellikle el hareketleri, jestler veya konuşma gibi zamansal örüntülerin analizinde bu yapı, hareketin hem başlangıç hem de bitiş bağlamına erişerek doğruluğu artırabilir.

1. Girdi Özellikleri

- Modelin girişi, her biri görsel bir kareyi temsil eden ve belirli sayıda özelliğe sahip sıralı veri (zaman serisi) olarak tanımlanmıştır.
- Girdi şekli: (zaman_adımı, kare_özellik_sayısı)

```
input_shape = (self.sequence_length, self.frame_features)
inputs = tf.keras.Input(shape=input_shape)
```

Şekil 3.5.1 Girdi Özellikleri

2. Bidirectional LSTM Katmanları

- **İlk Katman:** 128 birimli çift yönlü LSTM katmanı, haretetlerin zamansal örüntülerini her iki yönde öğrenmek üzere yapılandırılmıştır. `return_sequences=True` sayesinde çıktılar bir sonraki LSTM katmanına tam sırayla aktarılır.
- **İkinci Katman:** 64 birimli ikinci çift yönlü LSTM katmanı, zaman içindeki daha derin ilişkileri öğrenir. Her iki katmanda da L2 regularizasyonu uygulanarak aşırı öğrenme (overfitting) önlenmeye çalışılmıştır.
- Her LSTM katmanından sonra sırasıyla:
 - **Batch Normalization:** Öğrenmeyi stabilize eder.
 - **Dropout (0.4):** Rastgele nöronları devre dışı bırakarak aşırı öğrenmeyi önler.

```
# 1. İlk Bidirectional LSTM katmanı (daha küçük)
x = tf.keras.layers.Bidirectional(
    tf.keras.layers.LSTM(128, return_sequences=True,
                         kernel_regularizer=regularizers.l2(1e-3))
)(inputs)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dropout(0.4)(x)

# 2. İkinci Bidirectional LSTM katmanı (daha küçük)
x = tf.keras.layers.Bidirectional(
    tf.keras.layers.LSTM(64, return_sequences=True,
                         kernel_regularizer=regularizers.l2(1e-3))
)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dropout(0.4)(x)
```

Şekil 3.5.2 LSTM Katmanları

3. Havuzlama ve Tam Bağlantılı Katman

- **GlobalAveragePooling1D:** Zaman serisi boyunca ortalama alınarak boyut düşürülür.
- **Dense Katmanı:** 128 nöronlu, ReLU aktivasyon fonksiyonlu tam bağlantılı katman, öğrenilen zamansal özellikleri işleyerek sınıflandırma için uygun forma getirir.
- Ardından yine **Batch Normalization** ve **Dropout (0.4)** uygulanır.

```

# 3. Global Average Pooling
x = tf.keras.layers.GlobalAveragePooling1D()(x)

# 4. Dense Layer
x = tf.keras.layers.Dense(128, activation='relu',
                         kernel_regularizer=regularizers.l2(1e-3))(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dropout(0.4)(x)

```

Şekil 3.5.3 Havuzlama ve Tam Bağlılı Katman

4. Çıkış Katmanı

- Son katman, sınıf sayısı kadar nörona sahip olup **Softmax** aktivasyon fonksiyonu ile sınıflandırma gerçekleştirir.
- Bu yapı sayesinde model, her hareket dizisini belirli bir sınıfa ait olarak etiketleyebilir.

```

# 5. Output Layer
outputs = tf.keras.layers.Dense(self.num_classes, activation='softmax',
                                kernel_regularizer=regularizers.l2(1e-3))(x)

```

Şekil 3.5.4 Çıkış Katman

5. Hiperparametre Optimizasyonu (Grid Search)

Modelin başarımını artırmak için **Grid Search** yöntemiyle farklı hiperparametre kombinasyonları denenmiştir:

- **Dropout oranı**,
- **Öğrenme oranı (learning rate)**,
- **Batch size** gibi parametreler sistematik olarak test edilmiştir.

```

param_grid = {
    'dropout': [0.3, 0.4, 0.5],
    'lr': [0.001, 0.0001],
    'batch_size': [16, 32, 64]
}

```

Şekil 3.5.5 Hiperparametre Optimizasyonu (Grid Search)

Bu aramalar sonucunda en yüksek doğruluk sağlayan parametre kombinasyonu belirlenmiş ve eğitimde bu değerler kullanılmıştır.

3. 5. 3 Eğitim ve Test Aşamaları

1. Veri Bölme:

Veri seti, % 78 eğitim, %12 validation ve %10 test olarak bölünecektir. Eğitim sırasında çapraz doğrulama (k-fold cross-validation) uygulanmıştır.

3. 5. 4 Model Eğitimi

1. Eğitim Parametreleri:

- Optimizasyon Algoritması: Adam (Adaptive Moment Estimation) kullanılmıştır.
- Kayıp (Loss) Fonksiyonu: Kategorik çapraz entropi kaybı (Categorical Cross Entropy) tercih edilmiştir.
- Öğrenme Oranı (Learning Rate): 0.001 ve 0.0001 olarak denenmiştir.
- Batch Boyutu: 16, 32, 64 olarak denenmiştir.
- Epoch Sayısı: 150 epoch boyunca eğitim yapılmıştır.

2. Model Değerlendirme:

- Doğruluk (Accuracy): Modelin sınıflandırma başarısını göstermektedir.
- F1 Skoru: Kesinlik(Precision) ve Duyarlılık(Recall) değerlerinin harmonik ortalamasını hesaplamaktadır..
- Karışıklık Matrisi (Confusion Matrix): Hatalı ve doğru sınıflandırmaları göstermektedir.

3. 5. 5 Deneysel Sonuçlar

Modelin eğitimi ve değerlendirmesi sonucunda elde edilen performans değerleri analiz edilmiştir.

- Başarı Oranı Karşılaştırması: Literatürdeki halihazırda metotlar ile bu çalışma kapsamında oluşturulan modelin performansı karşılaştırılmıştır.
- Hata Analizi: Yanlış sınıflandırma yapılan işaretlerin özellikleri analiz edilecek ve modelin iyileştirilmesi yapılmıştır.
- Gerçek Zamanlı Tanıma: Modelin canlı video akışında performansı test edilmiştir.

4. MODELLEME VE GELİŞTİRME AŞAMALARI

4. 1 Veri Seti Özellikleri

Bu çalışmada, Türk İşaret Dili (TİD) örneği için AUTSL veri seti kullanılmıştır, çeşitli ön işleme adımları gerçekleştirilmektedir. Veri seti, 226 farklı işaret ve toplam olarak 38.336 örnek içermektedir. 43 farklı işaretleyici tarafından sağlanan veri seti, arka plan çeşitliliği ve dinamik çevresel koşullar ile modelin gerçek dünya koşulları içinde daha iyi genelleme yapabilmesini düzenlenmektedir. Türk İşaret Dili eğitmenlerinin işbirliği ile, günlük hayatta sıkça kullanılan işaretler seçilmiş ve el hareketlerinin çeşitliliği dengeli bir biçimde dağıtılmaktadır.

4. 2 Veri Ön İşleme

Veri setinde bulunan videoların işlenme sürecinde, boş veya durağan karelerin tespiti, videoların aynı çözünürlükte ölçüklenmesi ve bozuk olan videoların ayıklanması gibi adımlar uygulanmaktadır. Yapılan bu işlemler sonucunda elde edilen temiz ve standart hale getirilmiş veri kümesi, modelin daha düzgün bir şekilde eğitilmesine olanak tanımaktadır.

4. 3 İskelet Çıkarımı

MediaPipe kütüphanesi ile el ve vücut hareketleri detaylı bir şekilde tespit edilmektedir. Elde edilen koordinatlar ise video kareleri üzerinden çıkarılıp, JSON formatında saklanmaktadır. Bu yöntem ile kurulum kolaylığı ve gerçek zamanlı analiz yapabilme yeteneği nedeniyle tercih edilmektedir.

4. 4 Model Mimarisi

Görsel ve zamansal bilgiyi içeren bir model oluşturulmuştur. İki katmanlı Bidirectional LSTM yapısı kullanılmıştır. Eğitim sürecinde Adam optimizasyon algoritması ve çapraz entropi (categorical cross-entropy) kaybı kullanılmıştır. En iyi performans için dropout, öğrenme oranı (lr) ve batch size değerleri Grid Search ile optimize edilmiştir.

5. KULLANILAN TEKNOLOJİLER

Bu projede, hem kullanıcı deneyimini iyileştirmek hem de gerçek zamanlı işaret dili tanıma sistemini etkili biçimde uygulamak amacıyla çeşitli teknolojilerden yararlanılmıştır. Kullanılan başlıca teknolojiler aşağıda başlıklar halinde sunulmuştur:

- Frontend Teknolojileri:

Kullanıcı arayüzü, HTML5, CSS3 ve JavaScript kullanılarak geliştirilmiştir. Bu

sayede kullanıcıların sisteme kolayca erişebilmesi, kamera ile görüntü alınabilmesi ve etkileşimli arayüz kullanımı sağlanmıştır. Tasarımda responsive (duyarlı) yapılar tercih edilerek tüm cihaz türlerine uyumlu hale getirilmiştir.

- Backend Teknolojileri:
Sunucu tarafı işlemlerinde Python programlama dili tercih edilmiş; Flask mikro web çatısı ve WebSocket protokolü ile gerçek zamanlı veri iletişimini sağlanmıştır. WebSocket, video karelerinin gecikmesiz biçimde işlenmesini mümkün kılmıştır.
- Makine Öğrenimi ve Görüntü İşleme:
Görüntü işleme ve işaret tanıma işlemleri için TensorFlow derin öğrenme kütüphanesi ile birlikte MediaPipe ve OpenCV kütüphanelerinden faydalanyılmıştır. MediaPipe, özellikle el ve vücut pozisyonlarını belirlemek için kullanılmış, elde edilen landmark verileri TensorFlow modeline giriş olarak verilmiştir.
- E-posta Servisi:
Şifre sıfırlama işlemleri için üçüncü taraf e-posta servisi olan EmailJS kullanılmıştır. Bu servis sayesinde kullanıcılara otomatik olarak doğrulama kodu gönderimi yapılmaktadır.
- Veri Saklama:
Kullanıcı oturumları, kayıt bilgileri ve geçici veriler tarayıcı tabanlı localStorage sistemi ile yönetilmektedir. Bu yöntem, sunucuda kullanıcı verisi saklamaya gerek kalmadan hızlı ve kullanıcı bazlı veri erişimi sağlamaktadır.
- Yayınlama ve Dağıtım:
Uygulamanın internet üzerinde erişilebilir olması için Vercel veya Netlify gibi modern hosting platformları önerilmektedir. Bu platformlar, statik ve dinamik içerikleri kolaylıkla yayinallyarak projeyi genel kullanıma açma imkânı sunmaktadır.

6. PROJE AKIŞI

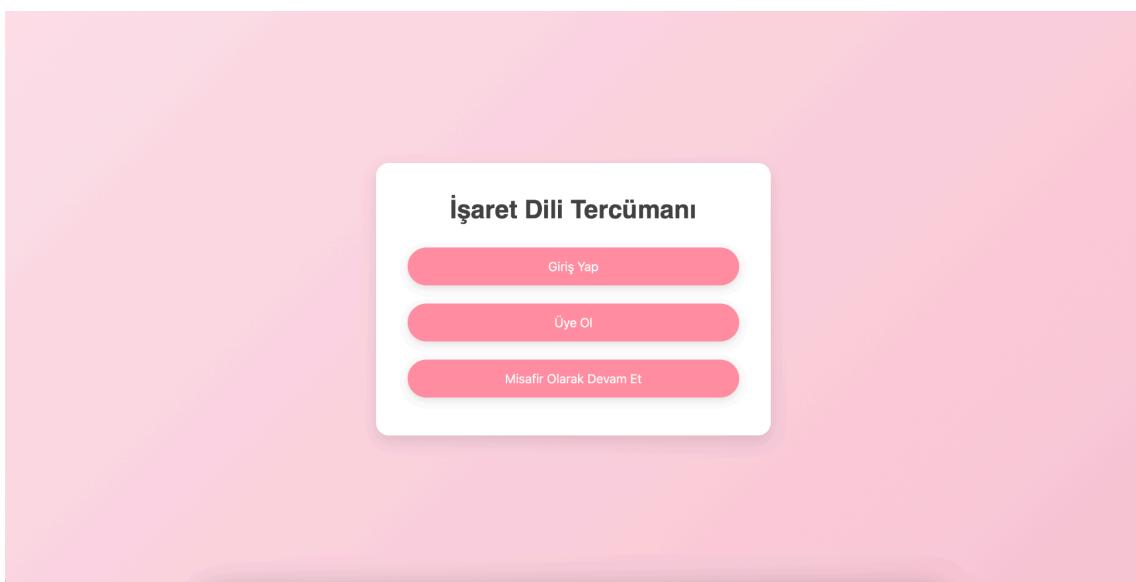
6.1. Kullanıcı Arayüzü (Frontend)

Sistem, kullanıcı dostu ve etkileşimli bir arayüz ile donatılmıştır. Kullanıcılar şu

işlemleri gerçekleştirilmektedir:

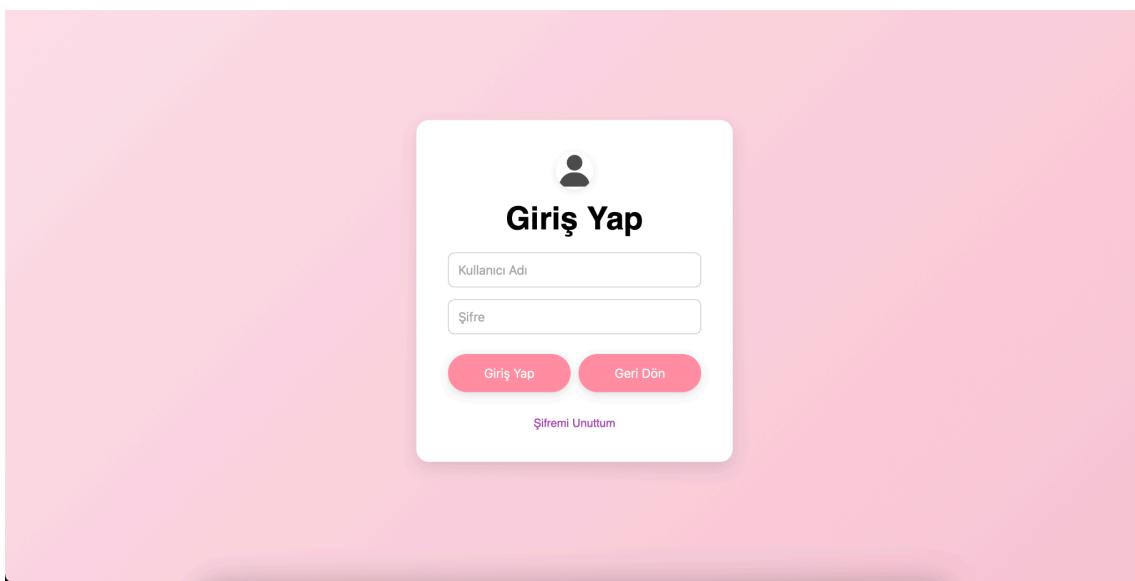


Şekil 6.1.1 Başlangıç Sayfası

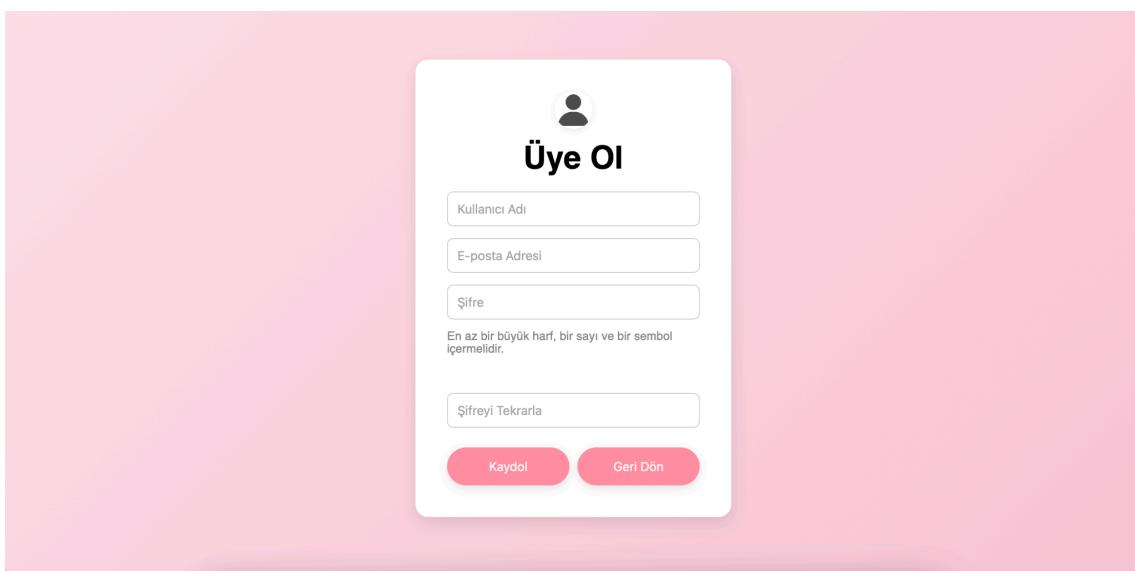


Şekil 6.1.2 Ana Sayfa

- **Kayıt ve Giriş:** Kullanıcılar yeni hesap oluşturabilir veya mevcut hesapları ile giriş yapabilir. Bu süreçte kullanıcı adı, e-posta ve şifre gibi bilgiler localStorage üzerinde saklanmaktadır.



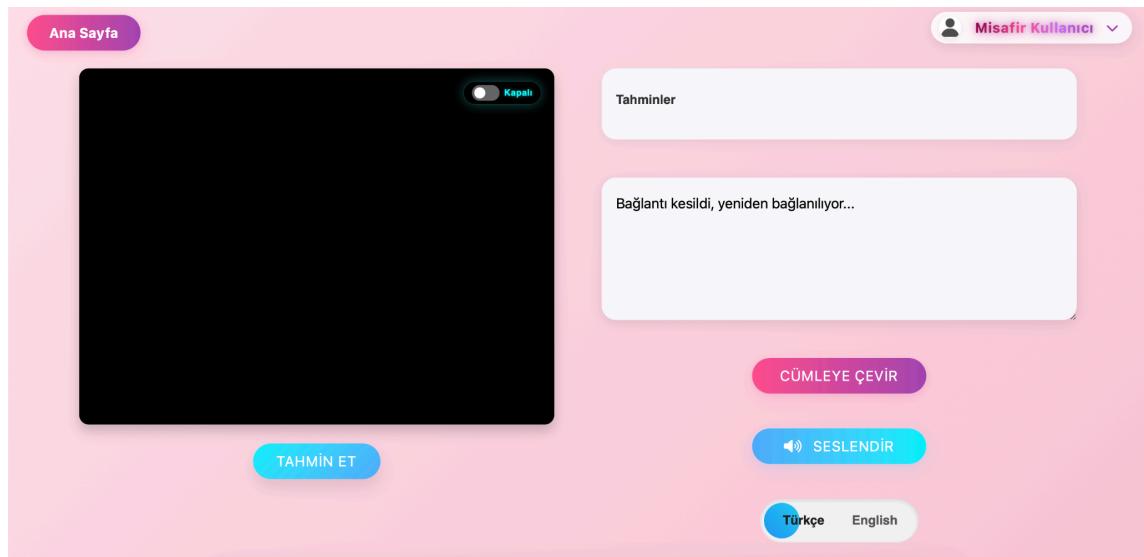
Şekil 6.1.3 Giriş Sayfası



Şekil 6.1.4 Kayıt Sayfası

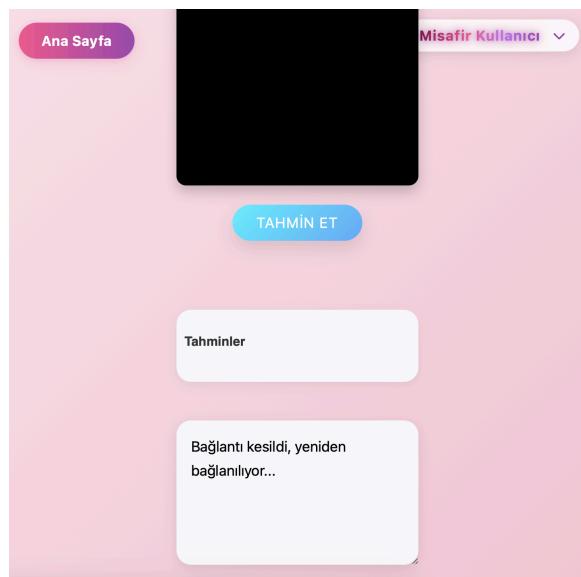
- **Şifre Sıfırlama:** Şifresini unutan kullanıcılar, sistemde kayıtlı e-posta adreslerini girerek doğrulama kodu alabilirler. EmailJS aracılığıyla 6 haneli rastgele bir kod gönderilmekte; bu kodun arayüze girilmesinin ardından kullanıcı yeni şifresini belirleyebilmektedir.
- **Kamera Tabanlı Tanıma:** Kamera üzerinden alınan görüntüler anlık olarak işlenerek, işaret dili hareketleri tanımlanmakta ve tahmin edilen kelimeler

kullanıcıya gösterilmektedir. Kullanıcı isterse bu kelimeleri anlamlı bir cümleye dönüştürebilmektedir.



Şekil 6.1.5 İşlem Sayfası

- Responsive Tasarım: Tüm bu işlemler, farklı ekran boyutlarına uyum sağlayabilecek biçimde tasarlanmıştır. Media query'ler ile desteklenen CSS yapısı sayesinde masaüstü, tablet ve mobil cihazlarda sorunsuz bir kullanıcı deneyimi sağlanmaktadır.



Şekil 6.1.6 Responsive Tasarım

6.2. Sunucu ve Model Entegrasyonu (Backend)

Sistemin arka planında, Python ile geliştirilmiş bir WebSocket sunucusu yer almaktadır. Bu yapı sayesinde, frontend tarafından gönderilen canlı kamera görüntülerini aşağıdaki adımlarla işlenmektedir:

1. Görüntü Alımı: Kamera görüntülerini, WebSocket protokolü üzerinden backend'e aktarılmaktadır.
2. Landmark Çıkarımı: MediaPipe kütüphanesi kullanılarak el ve vücut landmark'ları (anahtar noktalar) belirlenmektedir.
3. Model Tahmini: Elde edilen landmark verileri, TensorFlow tabanlı eğitilmiş bir derin öğrenme modeline aktarılmakta ve model bu veriler üzerinden ilgili işaret kelimesini tahmin etmektedir.
4. Sonuçların Gönderimi: Tahmin edilen kelime, frontend arayüzüne iletilmekte ve kullanıcıya sunulmaktadır.
5. Cümle Oluşturma: Kullanıcı tercihine bağlı olarak, art arda gelen tahminler Gemini API'si yardımıyla birleştirilerek anlamlı bir cümle haline getirilebilmektedir.

6.3. Şifre Sıfırlama ve E-posta Entegrasyonu

Şifre sıfırlama işlemi, EmailJS servisi aracılığıyla gerçekleştirilmiştir. Bu sürecin teknik detayları aşağıdaki gibidir:

- Kullanıcı, e-posta adresini girerek şifre sıfırlama isteğinde bulunur.
- Sistem, EmailJS API'si aracılığıyla bu adrese 6 haneli rastgele bir doğrulama kodu gönderir.
- Kullanıcı bu kodu ilgili giriş alanına girdikten sonra yeni şifresini belirleyebilir.
- EmailJS şablonları dinamik olarak kullanıcı bilgilerine göre özelleştirilmekte, böylece kişiye özel ve güvenli bir sıfırlama süreci sağlanmaktadır.



Şekil 6.3 Şifre Yenileme Maili

Projede kullanıcıların şifrelerini güvenli bir şekilde yenileyebilmelerini sağlamak amacıyla, **EmailJS** servisi ile e-posta tabanlı bir şifre sıfırlama mekanizması uygulanmıştır. Bu yaklaşım, kullanıcı deneyimini artırırken aynı zamanda temel güvenlik standartlarını da karşılamaktadır.

Uygulama Süreci

1. Şifre Yenileme Talebi:

Kullanıcı, giriş ekranında yer alan “Şifremi Unuttum” bağlantısına tıklayarak, sistemde kayıtlı olan e-posta adresini girer.

2. EmailJS ile E-posta Gönderimi:

Kullanıcının girmiş olduğu e-posta adresine, **EmailJS** aracılığıyla tek kullanımlık bir bağlantı içeren şifre sıfırlama e-postası gönderilir. EmailJS, sunucu tarafı kod yazmadan, istemci tarafında SMTP tabanlı e-posta gönderimini mümkün kılarak hızlı bir çözüm sunmaktadır.

3. E-posta Şablonu ve Güvenlik:

Gönderilen e-postada, kullanıcının kimliğini doğrulamasını sağlayacak şekilde özel bir token içeren bağlantı yer almaktadır. Bu bağlantı, belirli bir süre için geçerli olacak şekilde zaman damgası ile desteklenmiştir. E-posta içeriği, EmailJS'in şablon sistemi üzerinden özelleştirilmiştir.

4. Yeni Şifre Belirleme:

Kullanıcı e-posta içerisindeki bağlantıya tıkladığında, sistem tarafından yönlendirilen arayüz üzerinden yeni şifresini girerek işlemi tamamlar. Girilen

yeni şifre, gerekli güvenlik önlemleri (örneğin, minimum karakter sayısı, büyük/küçük harf, özel karakter vb.) sağlandığında sistem tarafından kabul edilir.

Teknik Kazanımlar

- Kullanıcılar sisteme kolayca şifrelerini sıfırlayabilir hâle gelmiştir.
- EmailJS sayesinde, backend geliştirmesine gerek kalmadan doğrudan istemci tarafında işlem tamamlanmıştır.
- Süreç, kullanıcı doğrulamasını destekleyen güvenli ve kullanıcı dostu bir yöntemle uygulanmıştır.

7. GERÇEKLEŞTİRİLEN TEKNİK ÇALIŞMALAR

Projenin geliştirme sürecinde aşağıdaki teknik çalışmalar başarıyla tamamlanmıştır:

- **Kullanıcı Yönetimi:** Kayıt olma, giriş yapma, oturum yönetimi ve profil düzenleme işlemleri localStorage kullanılarak güvenli bir şekilde gerçekleştirilmiştir.
- **Gerçek Zamanlı Görüntü İşleme:** WebSocket aracılığıyla frontend'den backend'e gönderilen görüntüler MediaPipe ile analiz edilmiş, ardından TensorFlow modeli ile işaret dili tahminleri gerçekleştirilmiştir.
- **Anlamlı Cümle Oluşturma:** Tahmin edilen işaretlerin mantıksal bağlam içinde birleştirilmesini sağlayan özel bir cümle oluşturma fonksiyonu GPT ve Gemini LLM'leri kullanılarak geliştirilmiştir. Gemini modeli seçilerek projede kullanılmıştır.
- **Responsive Arayüz Tasarımı:** Tüm cihazlarda uyumlu çalışacak şekilde duyarlı tasarım ilkeleri benimsenmiş ve arayüz CSS medya sorguları ile desteklenmiştir.
- **Kullanıcı Deneyimi Geliştirme:** Modern bir tasarım dili ile kullanıcı arayüzü zenginleştirilmiş, hata mesajları ve bilgilendirme ekranlarıyla kullanıcı yönlendirmesi sağlanmıştır.

8. SİSTEM GEREKSİNİMLERİ

Çalışma, sistem özellikleri birbirinden farklı olan üç farklı bilgisayarda gerçekleştirılmıştır. Bu üç sistemin özellikleri aşağıdaki tabloda yer almaktadır:

Özellik	ASUS TUF Gaming F15	MSI GL75 Leopard	MacBook Pro 2020
İşletim Sistemi	Windows 11 Pro (Sürüm 24H2)	Windows 11 Home Single Language (23112)	macOS Sequoia
İşlemci	Intel Core i5-10300H @ 2.50GHz	Intel Core i7-10750H @ 2.60GHz	Apple M1
RAM	32 GB	16 GB	8 GB
Depolama	477 GB	477 GB	245,11 GB
Grafik Kartı	Birden fazla GPU (4 GB belirtilmiş)	Birden fazla GPU (6 GB belirtilmiş)	Apple M1 entegre grafik
Sistem Türü	64 bit, x64 tabanlı	64 bit, x64 tabanlı	Apple Silicon (ARM tabanlı)

Tablo 8 Sistem Gereksinimleri

9. SONUÇ VE TARTIŞMA

Bu çalışmada, Türk İşaret Dili'nin (TİD) gerçek zamanlı olarak tanınması, metne dönüştürülmesi ve ardından İngilizce'ye çevrilmesi hedeflenmiştir; bu kapsamda hem derin öğrenme yöntemleri hem de etkileşimli bir yazılım altyapısı bir arada kullanılmıştır. Proje, iştirme ve konuşma engelli bireylerin hem Türkçe hem de İngilizce konuşan kişilerle daha etkili iletişim kurabilmesini sağlamayı amaçlamaktadır. Geliştirilen sistem; görüntü işleme, iskelet çıkarımı, zaman serisi modelleme ve kullanıcı arayüzü tasarımları gibi çok katmanlı teknik süreçlerin entegre edilmesiyle oluşturulmuştur.

AUTSL veri seti kullanılarak gerçekleştirilen model eğitiminde, veri ön işleme adımları büyük önem arz etmiş ve ham verilerin homojen hale getirilmesiyle modelin doğruluk oranı artırılmıştır. Özellikle, MediaPipe ile elde edilen iskelet koordinatları sayesinde hem ellerin hem de vücutun detaylı bir şekilde analiz edilmesi mümkün olmuştur. Bu yaklaşım, OpenPose gibi daha karmaşık yöntemlerin kurulum zorluklarını ortadan kaldırarak gerçek zamanlı performans açısından önemli avantajlar sağlamıştır.

Model mimarisi olarak kullanılan çift katmanlı Bidirectional LSTM yapısı, işaret dili hareketlerinin zamansal örüntülerini başarılı bir şekilde öğrenmiştir. Grid Search ile yapılan hiperparametre optimizasyonları, öğrenme oranı, dropout oranı ve batch size gibi parametrelerin en uygun kombinasyonunu ortaya koyarak, modelin hem doğruluk hem de genelleme başarısını artırmıştır. Eğitim süreci boyunca kullanılan Adam optimizasyon algoritması ve categorical cross-entropy kaybı sayesinde sınıflandırma performansı dengeli bir şekilde sağlanmıştır.

Model, literatürdeki benzer çalışmalarдан farklı olarak iskelet çıkarımından elde edilen koordinatları kullanarak özgün bir yaklaşım sunmaktadır. Bununla birlikte, bazı işaretlerin özellikle yüzü örten el hareketleri ya da benzer dinamiklere sahip olmaları, yanlış sınıflandırma ihtimalini artırmıştır. Bu nedenle, daha hassas özellik çıkarımı yapan sensörlerle desteklenen multimodal veri setlerinin kullanılması, modelin performansını daha da iyileştirebilir.

Ayrıca, geliştirilen kullanıcı arayüzü ve WebSocket tabanlı sunucu yapısı sayesinde

sistemin gerçek zamanlı çalışması başarıyla sağlanmıştır. Kullanıcılar, canlı kamera görüntüsü üzerinden alınan işaretlerin karşılıklarını anlık olarak görebilmekte ve bu kelimeleri anlamlı cümlelere dönüştürebilmektedir. Gemini gibi modern dil modellerinin entegrasyonu ile, anlam bütünlüğü gözetilerek çeviri kalitesi de iyileştirilmiştir.

Sonuç olarak, bu proje kapsamında geliştirilen sistem, Türk İşaret Dili tanıma ve çeviri teknolojileri açısından hem akademik hem de pratik anlamda önemli katkılar sunmaktadır. Gelecek çalışmalarında, modelin diğer işaret dillerine genişletilmesi, farklı yaş gruplarından veri toplanması, konuşmaya çeviri modülünün eklenmesi ve mobil platformlara entegrasyonu gibi geliştirmelerle daha kapsamlı bir iletişim çözümüne dönüştürülmesi mümkündür.

KAYNAKLAR

- [1] Stokoe, W. C. (1960). Sign language structure: *An outline of the visual communication systems of the American deaf*. Studies in Linguistics: Occasional Papers.
- [2] WFD (World Federation of the Deaf). (2023). *Guidelines for achieving sign language rights*. World Federation of the Deaf
- [3] Brentari, D. (1998). *A prosodic model of sign language phonology*. MIT Press.
- [4] Kurumu, T. D. (2023). iletişim. Nisan, 3, 2023.
- [5] Supalla, T., & Webb, R. (1995). CHAPTER FIFTEEN. *Language, Gesture, and Space*, 333.
- [6] Elakkiya, R., & Sumathi, V. P. (2020). Deep learning techniques for sign language recognition: A review. *Artificial Intelligence Review*, 53(8), 5685-5730.
<https://doi.org/10.1007/s10462-020-09804-3>
- [7] Starner, T., Weaver, J., & Pentland, A. (1998). Real-time American Sign Language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12), 1371-1375.
<https://doi.org/10.1109/34.736314>
- [8] De Coster, G., & Sincan, S. (2023). *Isolated Sign Recognition From RGB Video Using Pose Flow*. arXiv. <https://arxiv.org/pdf/2403.14534>
- [9] De Coster, G. (2021). *Isolated Sign Recognition From RGB Video Using Pose Flow and CVPRW 2021*. ChaLearn.
https://openaccess.thecvf.com/content/CVPR2021W/ChaLearn/html/De_Coster_Isolate_d_Sign_Recognition_From_RGB_Video_Using_Pose_Flow_and_CVPRW_2021_paper.html
- [10] Sincan, S., & Koç, A. (2020). *Deep Learning Approaches to Sign Language Recognition*. IEEE Xplore. <https://ieeexplore.ieee.org/abstract/document/9210578>
- [11] Kocabey, E., & Şahin, F. (2023). *Sign Language Recognition Using Machine*

Learning. ACM Digital Library. <https://dl.acm.org/doi/abs/10.1145/3654522.3654559>

[12] Sincan, S., & Koç, A. (2020). *Large-Scale Sign Language Recognition Using Pose Flow*. arXiv. <https://arxiv.org/abs/2004.01283>

[13] Sincan, S., & De Coster, G. (2021). *Large Scale Signer Independent Isolated Sign Language Recognition*. ChaLearn.

https://openaccess.thecvf.com/content/CVPR2021W/ChaLearn/html/Sincan_ChaLearn_LAP_Large_Scale_Signer_Independent_Isolated_Sign_Language_Recognition_CVPR_W_2021_paper.html

[14] O. M. Sincan and H. Y. Keles, "AUTSL: A Large Scale Multi-Modal Turkish Sign Language Dataset and Baseline Methods," in IEEE Access, vol. 8, pp. 181340-181355, 2020, doi: 10.1109/ACCESS.2020.3028072.

[15] Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. Neural computation, 9(8), 1735-1780.

[16] Olah, C. (2015). *Understanding LSTM Networks*.

<http://colah.github.io/posts/2015-08-Understanding-LSTMs>

[17] Schuster, M., & Paliwal, K. K. (1997). *Bidirectional recurrent neural networks*. IEEE Transactions on Signal Processing, 45(11), 2673–2681.