BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TÜRK İŞARET DİLİ ALFABESİNİN DERİN ÖĞRENME YÖNTEMİ İLE SINIFLANDIRILMASI

ZEREN BERNA KIN

YÜKSEK LİSANS TEZİ
2019

TÜRK İŞARET DİLİ ALFABESİNİN DERİN ÖĞRENME YÖNTEMİ İLE SINIFLANDIRILMASI

CLASSIFICATION OF TURKISH SIGN LANGUAGE ALPHABET WITH DEEP LEARNING METHOD

ZEREN BERNA KIN

Başkent Üniversitesi
Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin
ELEKTRİK-ELEKTRONİK Mühendisliği Anabilim Dalı İçin Öngördüğü
YÜKSEK LİSANS TEZİ
olarak hazırlanmıştır.

2019

“**Türk İşaret Dili Alfabesinin Derin Öğrenme Yöntemi ile Sınıflandırılması**” başlıklı bu çalışma, jürimiz tarafından, 23/01/2019 tarihinde, **ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI 'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan(Danışman)	Prof. Dr. Hamit Erdem
Üye	Dr. Öğr. Üyesi Selda Güney
Üye	Dr. Öğr. Üyesi Ökkeş Tolga Altınöz

ONAY

01/2019

Prof. Dr. Ömer Faruk Elaldı

Fen Bilimleri Enstitüsü Müdürü

TEŞEKKÜR

Tez çalışmalarım boyunca katkılarından dolayı, Sayın Prof. Dr. Hamit Erdem'e (tez danışmanı) çalışmalarımın her aşamasında karşılaşılan güçlüklerin aşılmasında her zaman yardımcı ve yol gösterici olduğu için, nişanlım Irmak Köseoğlu'na, annem Sıdıka Kın'a, babam Selahattin Kın'a, ablam Bengül Kın Yalçın'a ve ağabeyim Boran Can Yalçın'a bu süreç boyunca yanımda olup, her zaman beni destekledikleri için, Argela ailesine ve tüm çalışma arkadaşlarıma tez çalışmalarım boyunca gösterdikleri anlayış için teşekkürlerimi borç bilirim.

ÖZ

TÜRK İŞARET DİLİ ALFABESİNİN DERİN ÖĞRENME YÖNTEMİ İLE SINIFLANDIRILMASI

Zeren Berna Kın

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Günümüzde işitme engeli olan insanların, kendi aralarında anlaşmak için kullandıkları işaret dilinin çok az insan tarafından biliniyor olmasından dolayı günlük yaşantılarında iletişim kurmak konusunda bir takım sıkıntılar yaşadıkları bilinmektedir. İşitme engeli olan insanlarla duyabilen insanların arasındaki bu iletişim engelini azaltmak için pek çok akademik çalışma yapılmıştır ve günümüzde bu konuyla ilgili çalışmalar hala sürmektedir. Son zamanlarda makine öğrenmesi, derin öğrenme alanında da bu konuda çalışmalar yapılmaktadır. Bu tez çalışmasında, 29 Türkçe işaret dili alfabesi karakterleri ve metin yazmak için tanımlanmış 3 karakterden oluşan veri seti, her bir sınıf için 1500 adet görüntü kaydedilerek oluşturulmuştur. Önceden eğitilmiş bir konvolüsyonel yapay sinir ağ modeli kullanılarak transfer öğrenme metodu ile sistem, Türkçe işaret dili alfebesinden ve özel karakterlerden oluşturulan veri seti ile eğitilmiştir. Eğitim sonrası web kamera ile seçili alanda gösterilen işaretin gerçek zamanlı olarak tanımlanması ve bu yol ile bir kelime ya da cümle oluşturup kaydedilmesi sağlanmıştır. Bu çalışmada kullanılan model, eğitim ve tanımlama işlemleri python dili kullanılarak yazılıma dönüştürülmüştür. Kullanılan önceden eğitilmiş konvolüsyonel sinir ağı modelinin başarısı test edilip yorumlanmıştır. Performans kriterlerine göre başarı %90 olarak elde edilmiştir.

ANAHTAR SÖZCÜKLER: Türkçe İşaret Dili Alfabesi, İşaret Dili Tanımlama, Örüntü Tanıma, Derin Öğrenme, Konvolüsyonel Yapay Sinir Ağları, Gerçek zamanlı Nesne Tanımlama, Öğrenim Transferi

Danışman: Prof. Dr. Hamit Erdem, Başkent Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü

ABSTRACT

CLASSIFICATION OF TURKISH SIGN LANGUAGE ALPHABET WITH DEEP LEARNING METHOD

Zeren Berna Kın

Master of Science

Department of Electrical and Electronics Engineering

Today, it is known that people with hearing impairments have some difficulties while communicating in their daily lives as the sign language they use to communicate with each other is known to very few people. Many academic studies have been carried out to reduce this communication barrier between people who has hearing impairments and the ones who has not, and the studies on this issue are still ongoing. In the fields of machine learning and deep learning studies are being carried out on this subject. In this thesis, The 29 Turkish sign language alphabet characters and 3-character data set for writing text were created by recording 1500 images for each class. The system has been trained with a data set consisting of Turkish sign language alpahbet and 3 special characters by using the transfer learning method with a pre-trained convolutional neural network model. After the training, it was ensured that the signal displayed in the selected area was defined in real time and thus, creation of a word or formation of a sentence which will then be saved was made possible. The training and identification procedures and the model which has been used in this study were converted into software using Python programming language. The success of the pre-trained convolutional neural network model has been tested and interpreted.According to the performance criteria, the success rate is %90.

KEYWORDS: Turkish Sign Language Alphabet, Sign Language Recognition, Pattern Recognition, Deep Learning, Convolutional Neural Network, Real-Time Recognition, Transfer Learning

Supervisor: Prof. Dr. Hamit Erdem, Başkent Üniversitesi, Department of Electrical and Electronics Engineering

İÇİNDEKİLER LİSTESİ

Sayfa

İÇİNDEKİLER LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

ÇİZELGELER LİSTESİ

Sayfa

SİMGELER VE KISALTMALAR LİSTESİ

TİD	Türkçe İşaret Dili
TÜİK	Türkiye İstatistik Kurumu
KYSN	Konvolüsyonel Yapay Sinir Ağları
YSA	Yapay Sinir Ağları

1. GİRİŞ

İletişim kurma ihtiyacı sosyal bir varlık olan insan için temel bir ihtiyaçtır. Sosyolojik açıdan bu ihtiyaç insanın ait olduğu toplumun bir parçası olmayı istemesinden doğar. Vücut dili gibi iletişim kurmayı evrenselleştiren bir seçenek olsa da kullanılan dillerin farklılığı bile iletişim kurmayı zorlaştırabilir. Bu durumda yalnızca işitme engeli olan ve yakın çevrelerinde bulunan insanlardan oluşan ve dolayısıyla daha az kişi tarafından bilinen işaret dilini ele alırsak, işitme engelliler için iletişim kurmanın ne kadar zorlaştığını anlayabiliriz.

İşitme engelliler sosyal yaşantılarında iletişim kurmaktan geçen her konuda çeşitli zorluklar yaşamaktadır. Genelde işitme engeli olan insanlar, duyabilen insanlarla vücut dilini kullanarak veya yazarak iletişime geçmeyi tercih etmektedir. İşitme engelliler kendi sesleri duyamadıkları için bu eksiklikten dolayı yazılı metinleri de anlamakta da güçlük çekerler. Bu sebeple yazılı iletişim, işitme engellilerin iletişim kurmasında pratik olmayan ve yetersiz bir alternatiftir. Bu sebeple işitme engelliler bu zorluğu azaltmak için işaret dilini kullanmaktadır. İşaret dili kişiden kişiye göre değişmeyen ve yoruma açık olmayan bir dil olsa da her dil de olduğu gibi ülkeden ülkeye değişmektedir. Örneğin Amerika'da ya da Hindistan'da kullanılan işaret dili alfabesi (TİD) Türkçe işaret dili alfebesinden farklıdır. Bu sebeple Türkiye'de yaşayan işitme engelliler kendi aralarında TİD kullanarak anlaşabilse de yabancı bir işitme engelliyle bu dili kullanarak anlaşamayabilir. Türkiye'de TİD yaygın olarak kullanılır.

Dünya Sağlık Örgütü ve Dünya Bankası Grubu tarafından hazırlanan "Dünya Engellilik Raporu"na göre, dünyada 600 milyon işitme engelli birey var. Başbakanlık Özürlüler İdaresi Başkanlığı ise 2002'de yapılan son araştırmaya göre Türkiye'de toplam 8,5 milyon engelli olduğunu ve bu sayının yüzde 22'sinin işitme ve konuşma özürlü olduğunu açıklamıştır [48]. Bu sayıya bakıldığında Türkiye'de işitme engelli sayısının genel nufüsa oranla oldukça fazla insanın işitme ve konuşma engelli olduğunu görebiliriz.

Tercüman ile iletişim de işitme engelli insanlar ile iletişim kurmakta tercih edilebilir. Fakat bu pahalı bir alternatif olduğu için herkes tarafından kullanılamaz ve

dolayısıyla yetersiz kalır. Bu sebeple bu dili otomatik olarak algılayabilecek ve tanımlayabilecek bir sisteme ihtiyaç vardır.

İşitme engeli olan insanlarla duyabilen insanlar arasında bu iletişim kurma güçlüğünü azaltmak için teknolojiden yararlanarak bir takım çalışmalar yapılmıştır. İşaret dilini tanımada Amerikan, Arap, Hint, Yunan, Alman, Pakistan, İspanyol, Meksika, Vietnam, Arjantin, Arnavutluk, İtalyan, Tayvan, Pers, Avustralya, Kore, Fars, Çek, Rus , Türk işaret dilleri gibi çeşitli işaret dillerinde çeşitli yöntemler kullanılarak işaret dili tanıma çalışmaları bulmak mümkündür. Bu çalışmaların çoğu Amerikan işaret dili üzerine yapılmıştır.

Amerikan işaret dili üzerinde, sensör eldiveni ve çok katmanlı yapay sinir ağı kullanılarak [1], saklı Markov modeli kullanarak [2], gradyanların histogramı metodu ve destek vektör makinesi yapay sinir ağı kullanarak [3], sıçrama hareketini algılayan sensör verilerini ve destek vektör makinesi yapay sinir ağı kullanarak [4], polar dönüşüm histogramı ve seyrek otomatik kodalyıcı ve derin sinir ağı hibrit yapısı kullanarak [5] ve derinlik ve renk üzerinden özellik çıkaran konvolüsyonel yapay sinir ağı kullanarak [6] işaret dilini tanıma çalışmaları yapılmıştır. Arap işaret dili üzerine, Öklit uzaklık hesaplamalarını kullanarak [7], gradyanların histogramı, kanonik korelasyon analizi ve rastgele Forest sınıflandırıcı kullanarak [8], Kinect hareket sensörü ve Saklı Markov modeli kullanarak [9], ANFIS bulanık mantık ağı kullanarak [10], işaret dilini tanıma çalışmaları yapılmıştır. Hint işaret dili üzerine, derinlik sensörü kullanılarak [11], eklem açısal yer değiştirme haritaları ve konvolüsyonel yapay sinir ağları kullanılarak [12] işaret dilini tanıma çalışmaları yapılmıştır. Yunan işaret dili üzerine, sensör eldiveni kullanarak [13], derin öğrenme algoritmaları kullanarak [14] işaret dili tanıma üzerine çalışmalar yapılmıştır. Alman işaret dili üzerine, saklı Markov modeli kullanarak [15] işaret dili tanıma çalışmaları yapılmıştır. Pakistan işaret dili üzerine, K-En yakın Komşu hesaplama yöntemi kullanarak [16], destek vektör makinesi kullanarak [17] işaret dili tanıma çalışmaları yapılmıştır. İspanyol işaret dili üzerine, sensör eldiveni ve sinyal işleme teknikleri kullanılarak [18] işaret dili tanıma çalışmaları yapılmıştır. Meksika işaret dili üzerine, Kinect hareket sensörü kullanarak [19] işaret dili tanıma çalışmaları yapılmıştır. Vietnam işaret dili üzerine, destek vektör makinesi yapay sinir ağı ve saklı markov modeli kullanılarak

[20], mikroelektronik mekanik sistem iel tasarlanan ivmeölçer eldiven ve bulanık mantık kullanılarak [21] işaret dili tanıma çalışmaları yapılmıştır. Arjantin işaret dili üzerine, öz-düzenleyici haritalar kullanarak [22] işaret dilini tanıma çalışmaları yapılmıştır. Arnavutluk işaret dili üzerine, Kinect hareket sensörü ve öklit uzaklığı hesaplamaları kullanılarak [23] işaret dilini tanımala çalışmaları yapılmıştır. İtalyan işaret dili üzerine, destek vektör makinesi yapay sinir ağı ve saklı Markov modeli kullanarak [24] işaret dili tanıma çalışmaları yapılmıştır. Tayvan işaret dili üzerine, Kinect hareket sensörü saklı markov modeli ve desket vektör makinesi kullanarak [25] işaret dili tanıma çalışmaları yapılmıştır. Pers işaret dili üzerine, k-en yakın komşu hesaplamaları kullanarak [26] işaret dili tanıma çalışmaları yapılmıştır. Avustralya işaret dili üzerine, saklı markov modeli kullanarak [27] işaret dili tanıma çalışmaları yapılmıştır. Kore işaret dili üzerine, Bulanık mantık ve saklı markov modeli kullanarak [28], veri eldiveni ve bulanık mantık kullarak [29] işaret dili tanıma çalışmaları yapılmıştır. Fars İşaret dili üzerine, fourier katsayı genliği hesaplamaları ve çok katmanlı yapay sinir ağı kullanarak [30] işaret dili tanıma çalışmaları yapılmıştır. Brezilya işaret dili üzerine derinlik ve renk sensörleri kullanarak [31] işaret dili tanıma çalışmaları yapılmıştır. Çek, Rus ve Türk işaret dilleri için, saklı markov modeli kullanarak sesten işaret diline, işaret dilinden sese çevirim yapabilen hem görme hem işitme engellilerin kullanabileceği bir sistem tasarlanmıştır [32]. İşaret dili tanıma için yapılan çalışmalar göz önünde bulundurulduğunda, çalışmalarda kullanılan yöntemlerden en yaygın olanları sensor verilerin işlenmesi, saklı markov modeli, destek vekör makinesi, matematiksel analizler, bulanık mantık, çok katmanlı yapay sinir ağları ve derin yapay sinir ağlarıdır.

Derin Yapay Sinir Ağları, günümüzde yapay sinir ağlarının özelleştirilmiş ve obje tanımlama işlerinde yapay sinir ağlarında olan özellik çıkarma maliyetini ortadan kaldırmış çok katmanlı yapay sinir ağlarıdır. Derin öğrenme, derin yapay sinir ağları kullanılarak yapılan makine öğrenmesi olarak tanımlanabilir.

Bir sınıflandırma problemi olan TİD tanıma için de çeşitli yöntemler kullanılarak işaret dili tanıma çalışmaları yapılmıştır. TİD için saklı markov modeli ve k-en yakın komşu hesaplamaları kullanarak [34], genelleştirilmiş hough dönüşümü yöntemi kullanarak %93 başarıyla [33], bileklik tabanlı kontur özelliği ve ten rengi bulma

yöntemi kullanarak %99.31 başarı elde edilen [35] , renk ve derinlik sensörü kullanarak %90 başarı elde edilen [36], veri eldiveni kullanarak [37], yalnız sesli harfler için sinyal işleme ve çok katmanlı yapay sinir ağı kullanarak %80 başarı elde edilen[38] işaret dili tanıma çalışmaları mevcuttur ancak TİD için derin öğrenme yapay sinir ağı kullanılarak yapılmış gerçek zamanlı TİD alfabesinin tamamını algılayabilen bir çalışma yoktur. Bu çalışmların pek çoğunda sınıflanırma için özellik çıkarımı yapılmıştır. Bu özelliklerin hiç verilmeden öğrenilmesi derin yapay sinir ağları ile mümkündür. Buna ek olarak, yapılan çalışmalarda kullanılan veri setleri hakkında yeterli açıklama yapılmamış ve kullanılan veri setleri herhangi bir yerde paylaşılmamıştır.

İşaret dili tanımada derin öğrenme yapılarından konvolüsyonel yapay sinir ağları [39,6,69,70] sınırlı Boltzman makineleri [40], tekrarlayan sinir ağları kullanılarak[41] yapılmış çalışmalar mevcuttur.

Görüntü üzerine yapılan çalışmalarda konvolüsyonel yapay sinir ağları oldukça başarılıdır ancak derin öğrenme algoritmalarını kullanmak için büyük veri setine ve bu veri setini işleyebilecek güçlü donanımlara ihtiyaç duyulmaktadır. Transfer öğrenme metodu kullanılarak önceden geniş bir veri setiyle eğitilmiş bir modelin öğrenme kabiliyetini kendi veri setimizi kullanarak aktarmak mümkündür. Bu yöntem kullanılarak çiçek [43] ve köpek [42] çeşitlerini sınıflandıran çalışmalar mevcuttur. İşare dilinde ise bu yöntem kullanılarak amerikan işaret dilini tanıma için çalışmalar yapılmıştır [69, 70]. Böylece kendi büyük veri setimizi zamandan ve işlemden tasarruf ederek sınıflandırmak mümkündür.

Bu tezde önerilen sistemde, 29 TİD alafabesi karakteri için her bir karakter için 1500 tane görüntü dosyasından oluşan veri seti kaydedilmiştir. Buna ek olarak bir kelime veya bir cümle yazmaya olanak sağlamak için işaretin gösterildiği alanın arka planı için 1500 adet görüntüden olşan bir sınıf tanımlanmıştır. Yanlış yazılan karakteri silme ve kelimeler arasına boşluk koyabilmek için iki özel tanımlı karakterden oluşan 1500’er görüntüden oluşan iki sınıf eklenmiştir.

Veri setinde toplam 32 sınıf ve 48000 adet görüntü bulunmaktadır.Bu veri seti kullanılarak daha önce eğitilmiş bir konvolüsyonel yapay sinir ağı modeli kullanılarak TİD alfabesi için kaydedilmiş görüntülerden oluşan veri seti üzerinde

öğrenmeyi transfer etme metodu kullanılarak eğitim yapılmış, sistem eğitildikten sonra gerçek zamanlı bu işaretlerin algılanıp tanımlanması ve bu yolla bir kelime veya cümle oluşturulması sağlanmıştır.

Yapılan çalışmada JetBrains PyCharm IDE 2017 3.1 Professional Edition ortamında Python dili kullanılarak bilgisayar tabanlı yazılım geliştirilmiştir. İşletim sistemi olarak Microsoft Windows 10 Pro kullanılmıştır. Yazılım geliştirmek için kullanılan bilgisyarın işlemcisi Intel(R) Core(TM) i7-6600u CPU @ 2.60GHz 2.81GHz dir. Sistem x64 tabanlı işlemciye sahiptir ve 8,00GB RAM e sahiptir.

Sistemin başarısı performans kriterlerine göre test edilmiş sonuçlar yorumlanmıştır. Veri setinin %80'i eğitim, %10'u doğrulama kalan %10'u ise test için kullanılmıştır. Bunun sonucunda sistemin veri setinin eğitimde kullanılmayan %10'luk test için ayırılan kısmında test başarısı %99.9 olarak ölçülmüştür. Buna ek olarak 2 erkek, 2 kadın kullanıcıdan oluşan 4 farklı kullanıcıdan tüm alfabe için alınan her karakter için 10 adet görüntüden oluşmaktadır. Kullanıcılar daha önce bu alfabeyi bilmeyen ve kullanmamış kişilerdir. Farklı kullanıcılardan tarafından alınan görüntülerden oluşan bu test veri seti oluşturulurken, her kullanıcı için 320 adet görüntü, toplamda 1280 adet görüntü kaydedilmiştir. Kullanıcılardan alınan bu test veri seti üzerinde sistemin başarısı test edilmiş. Sonuçları değerlendirilmiştir. Kullanıclar üzerinde yapılan test sonucunda sistemin başarısı tüm karakterler genelinde %90 olarak ölçülmüştür.

Önerilen çalışmanın amacı, kullanıcı tarafından seçilmiş alanda gösterilen işaret dili alfabesi ve tanımlanmış özel karakterlerin gerçek zamanlı olarak algılanabilmesi ve bu yolla bu karakterleri gösterek bir kelime veya bir cümle yazılmasıdır. Tasarlanan sistemin işitme engellilerle iletişim ve TİD eğitiminde kullanılarak fayda sağlaması hedeflenmektedir.

Toplam 6 bölümden oluşan bu tez çalışmasında;

Giriş bölümünde, tez çalışmasıyla alakalı genel bilgi verilmiştir.

Bölüm 2'de TİD hakkında genel bilgiler verilmiştir.TİD'in geçmişi ve gelişimi, TİD kullanıcı profili ve işaret dili alfabesi özelliklerine yer verilmiştir.

Bölüm 3'te makine öğrenmesi ve sinir ağları hakkında genel bilgiler verildikten sonra, konvolüsyonel sinir ağları, KYSA'nın katmanları ve kullanım alanları üzerinde durulmuştur.

Bölüm 4'te tasarımı yapılan sistem hakında bilgi verilmiştir. Sistem tasarımında kullanılan veri seti, veri setinin oluşturulması, algoritma ve sistemde kullanılan alt birimler, çalışma prensibi üzerine bilgiler verilmiştir.

Bölüm 5'de sistemin başarısını ölçmek için yapılmış testler hakkında bilgiler verilmiş, kullanılan yöntemler üzerine durulmuş ve sonuçlar grafiksel olarak yorumlanmıştır.

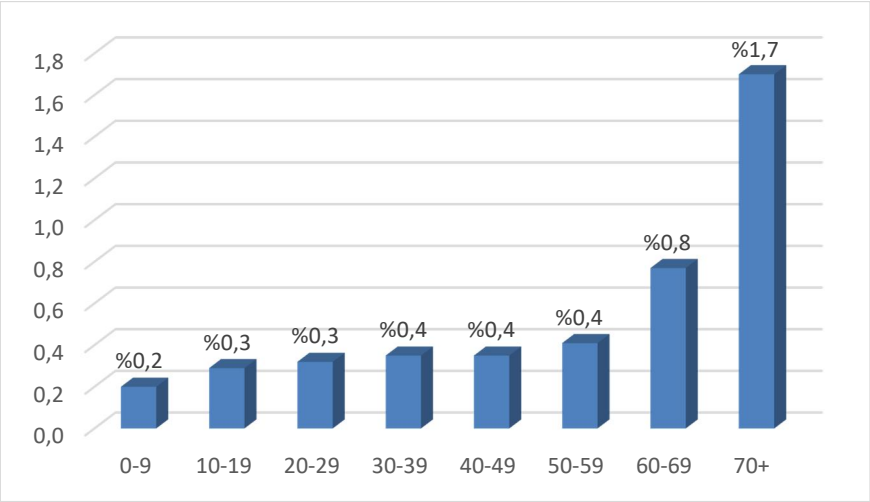
Son bölüm olan altıncı bölümde ise, bu tez çalışmasının sonucunda elde edilenler yorumlanmış ve öneriler ve gelecek çalışmalarla yapılması hedeflenenler hakkında bilgi verilmiştir.

2. TÜRKÇE İŞARET DİLİ

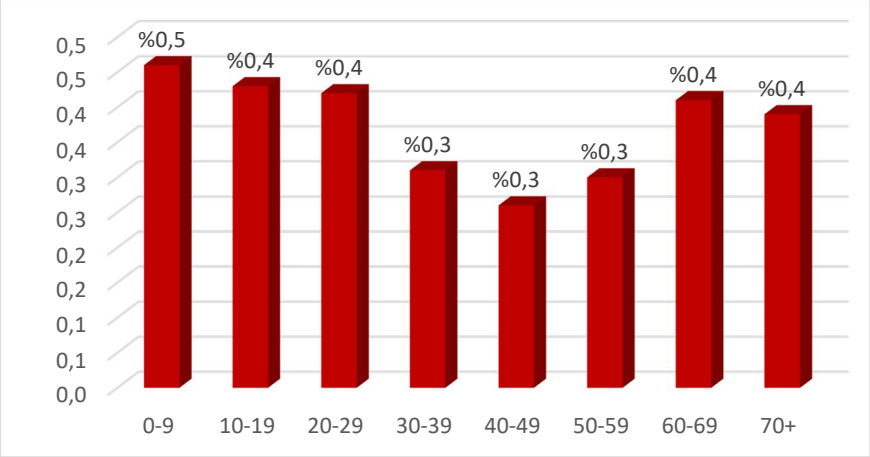
Türk İşaret Dili ya da TİD Türkiye'de işitme engelliler tarafından kullanılan dildir. Diğer işaret dillerinde olduğu gibi TİD' de Türkçe'nin dilbilimsel yapısından farklı, kendine özgü bir dilbilimsel yapısı vardır. TİD ile ilgili ilk bilimsel araştırmalar Ulrike Zeshan ve Hasan Dikyuva tarafından yapılmıştır [46,47]. TİD'in tarihinin bu araştırmalar soncunda çok eskilere dayandığı bilinmektedir. TİD hakkında TDK'da Milli Eğitim Bakanlığı tarafından 2015 tarihinde yayımlanan "Türk İşaret Dili Sözlüğü" dışında görsel bir sözlük veya yazılı bir arşiv bulunmamaktadır [48]. Bu sözlük işitme engellilerin günlük konuşmalarında en sık kullandıkları 2607 kelimeden oluşmaktadır. TİD için dijital ortamda kullanılabilecek kaynaklar Boğaziçi Üniversitesi Bilgisayar Mühendisliği Algısal Zeka Laboratuvarı'nda yürütülen "İşaret Dili Eğitmeni" projesi kapsamında ortaya çıkan TİD sözlüğü [51] ve Başak ve Serdar Uludağ tarafından oluşturulan ve oluşturulurken Milli Eğitim Bakanlığı'nın ve Türk Dil Kurumu'nun en güncel Türk İşaret Dili sözlüklerini kaynak olarak kullanan İşaretçe- Görüntülü Türk İşaret Dili Sözlüğüdür [52].

2.1. Türkiye'de TİD Kullanıcıları

Türkiye İstatistik Kurumu'nun verilerine göre Türkiye'de 89,043 kişi (53,543'i erkek 35,500'i kadın) işitme engelli, ve 55,480 kişi de (34,672 erkek, 20,808 kadın) konuşma engellidir [49]. Şekil 2.1 de TÜİK, Türkiye Engelliler Araştırması, 2002'ye göre işitme ve konuşma engelli nüfus yüzdesinin yaşa göre dağılımını gösteren şekillerden görüleceği gibi işitme engelli kişilerin yüzdesinin yaş ile beraber arttığını görebiliriz.



Şekil 2.1 Türkiye'de işitme engelliler nüfusu yaşa göre dağılımı [49]



Şekil 2.2 Türkiye'de dil ve konuşma engelliler nüfusu yaşa göre dağılımı [49]

Türkiye işaret dili eğitimi konusunda diğer ülkelere nazaran geride kalmaktadır. TİD eğitimi veren yeterli sayıda okul olmaması ve eğitim için gerekli kaynak ve materyallerin sağlanamaması TİD öğrenme ve kullanımının azalmasına sebep olmaktadır.

Bununla beraber konuda faliyet gösteren ve hizmet veren dernek ve kuruluşların sayısıda ger geçen gün artmaktadır. Bu dernek ve cemiyetler Türkiye İşitme Engelliler Milli Federasyonu altında toplanmıştır ve işitme engelliler için bir takım sosyal, kültürel aktiviteler düzenlemektedirler. Bunun dışında TRT'de ve pek çok ulusal televizyon kanalında işitme engelliler için yayınlar yapılmaktadır.

2.2. Türkçe İşaret Dilinin Tarihi ve Gelişimi

TİD, birçok işaret dilinden eski bir işaret dilidir. 16. ve 17. yüzyıllardan itibaren Osmanlı Devleti'nde pek çok makamda işaret dilinin kullanıldığına dair kayıtlar vardır, ancak II. Abdülhamit zamanında İstanbul'da açılan sağırlar okulu TİD'in en az 120 yıllık bir tarihe dayandığının bir göstergesidir [46]. Günümüzde kullanılan Türk İşaret Dili'nin Osmanlı'daki işaret diline dayanıyor olduğu henüz kanıtlanmış değildir [50].

1953 yılında Milli Eğitim Bakanlığı tarafından çıkarılmış bir kanun ile işaret dilinin okullarda kullanımı yasaklanmıştır. İşaret dilinin yerini sözel eğitimin almış ve sözel eğitimin erken yaşta işitme engellilerin konuşmasına sağlayabileceği düşünülmüştür. Fakat sonra bu eğitim metodunun yanlış olduğunun farkına varılıp bu yasak 2005 yılında kaldırılmıştır.

TİD'in geliştirilmesi ve toplumda yaygınlaştırılması ve farkındalık yaratılması için günümüzde çalıştaylar düzenlenmektedir. Buna rağmen şuanda bu konuda yapılan dilbilimsel çalışmalar ve mevcut TİD sözlükleri tam ve yeterli değildir. Bu konuda yapılan çalışmaların artması beklenmedir.

2.3. Türkçe İşaret Dili ve Alfabesinin Özellikleri

TİD, Türkçe'den türetildiği için Türkçe ile benzerlik gösterse de dilbilimsel açıdan farklı özelllikleri olan farklı bir dildir. TİD kelimelerden oluşan bir dildir. Genelde belirlenmiş bu kelimeleri kullanarak bir cümle oluşturma yoluyla işitme engelliler kendilerini ifade ederler. Bunun yanı sıra, kelimlerin yeterli olmadığı bazı durumlarda ya da özel isimler için harf işaretleri kullanılarak heceleme yöntemi kullanılmaktadır. Türkçe'de yer alan her harfin TİD alfabesinde bir karşılığı vardır. Tüm harfler tek tek el işaretleri ile ifade edilebilmektedir. Şekil 2.3'te TİD Alfabesi verilmiştir.

A	B	C	Ç
			
D	**E**	**F**	**G**
			
Ğ	**H**	**I**	**İ**
			
J	**K**	**L**	**M**
			
N	**O**	**Ö**	**P**
			
R	**S**	**Ş**	**T**
			
U	**Ü**	**V**	**Y**
			
Z			
			

Şekil 2.3 Türkçe işaret dili alfabesi

3. MAKİNE ÖĞRENMESİ

Makine öğrenmesi bir sistemin verilerden öğrenmesine imkan veren yapay zekanın bir şeklidir. Makine öğrenmesi, verileri tanımlamak ve veri ile ilgili sonuçları tahmin etmek için tekrarlı bir şekilde veriden öğrenen algoritmalar kullanır. Algoritmalar bu eğitim verilerini alır, bu verilere sayesinde daha kesin sonuçlar üreten modeller üretmek mümkündür. Sistemin eğitimi tamamlandktan sonra, eğitilen modele bir girdi verildiğinde anlamlı bir çıktı üretmesi beklenir. Buna örnek olarak, bir tahmin algoritması ile bir tahmin modeli oluştulup, oluşturulan bu model eğitildikten sonra, tahmin modeline veri sağlandığında, modelden eğitim verilerine dayanarak bir tahmin almamız verebilir.

3.1. Yapay Zeka ve Makine Öğrenmesinin Tarihi

Yapay zeka alanında yapılan çalışmalar 1950'lere kadar uzanır. Makine öğrenmesinin yapay zekanın alt çalışma alanı olduğu söylenebilir. Bir IBM araştırmacısı olan Arthur Lee Samuels, en eski makine öğrenme programı olan dama oynamak için kendi kendine öğrenme programını geliştirmiştir. Makine öğrenmesine geniş bir perspektiften bakabilmek için Şekil 3.1'e bakabiliriz. Buradan da görülebileceği gibi makine öğrenmesi, yapay zekanın; derin öğrenme de makine öğrenmesinin bir alt kırımı olarak tanımlanabilir.

Şekil 3.1 Yapay zeka ve makine öğrenmesine genel bakış

3.2. Makine Öğrenmesinde Büyük Verinin Önemi

Büyük veri, büyük veri hacmine sahip veri olarak tanımlanabilir. Bir makine öğrenim modelinin doğruluğu, büyük veriler ile eğitilmişse önemli ölçüde artabilir. Makine öğrenmesinde öğrenme sürecinde uygulanacak doğru veri setinim oluşturulması önemlidir. Makine öğrenmesinde büyük veriler sistemleri eğitirken sistemin ürettiiği çıkışların doğruluğu arttırmaya yardımcı olabilmektedir.

3.3. Makine Öğrenmesine Yaklaşımlar

3.3.1. Denetimli öğrenme

Denetimli öğrenme tipik olarak belirlenmiş bir veri kümesi ve bu verilerin nasıl sınıflandırıldığına dair belli bir anlayışla başlar. Denetimli öğrenme, bir analitik işlemine uygulanabilecek verilerdeki kalıpları bulmayı amaçlar.Bu veri, verilerin anlamını tanımlayan özelliklere sahiptir. Örneğin, milyonlarca hayvan imgesi olabilir ve her bir hayvanın ne olduğuna ilişkin bir açıklama içerebilir ve daha sonra bir hayvanı diğerinden ayıran bir makine öğrenme uygulaması oluşturabilirsiniz.

3.3.2. Denetimsiz öğrenme

Denetimsiz öğrenme, problemin etiketlenmemiş büyük miktarda veri gerektirmesi durumunda en uygun yöntemdir. Bu verinin arkasındaki anlamı anlamak, bulduğu örüntülere veya kümelere göre verileri sınıflandırabilmeye dayanan anlamı anlamaya başlayabilen algoritmalar gerektirir.

3.4. Yapay Sinir Ağları

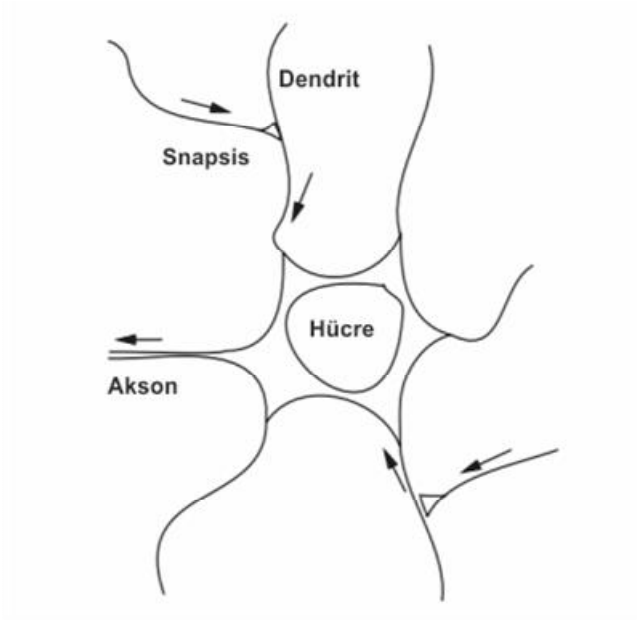
3.4.1. Genel bilgiler

YSA insan beyninin en temel işlevi olan öğrenme kabiliyetini bilgisayarlar için gerçekleyen sitemler olarak tanımlanabilir [53].

Öğrenme işlevi, insan beyninin biyolojik nöronundan esinlenerek tasarlanmıştır. Biyolojik sinir hücreleri birbirleriyle iletişim kurmak için sinapsleri kullanılırlar. Biyolojik Sinir Hücresi Şekil 3.2 İle gösterilmiştir. İşlenmiş bilgiler axon adı verilen yapı vasıtaysıyla diğer hücrelere gönderilir. Yapay sinir ağları birbirine bağlantılı

işlem elemanlarından meydana gelir ve kurulmuş her bağlantı bir ağırlık değerine sahiptir. YSA'nın öğrendiği bilgi ağırlık değerlerinde saklıdır ve tüm ağa yayılır.

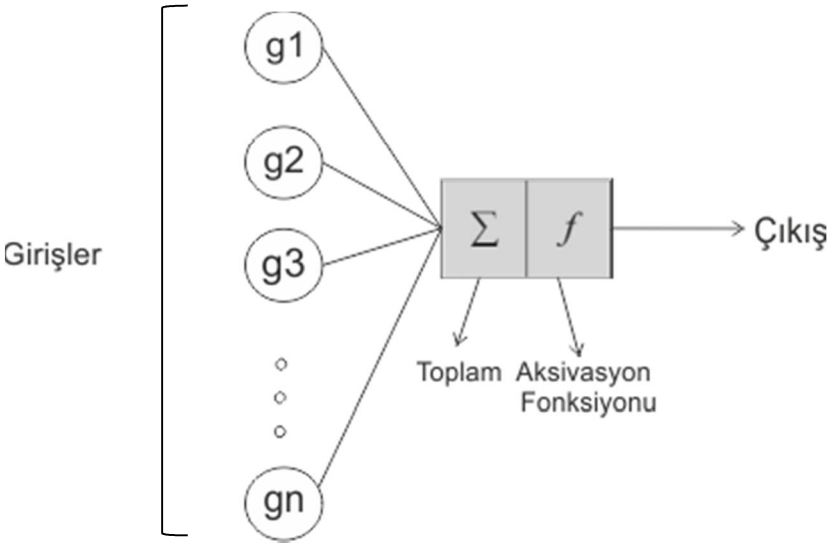
Biyolojik nöron ile benzer şekilde, yapay sinir hücreleri bilgileri toplama fonksiyonu vasıtasıyla toplar, bu toplamı bir aktivasyon fonksiyonundan geçirerek anlamlı bir çıktı üretir ve bu çıktıyı ağın bağlantıları ile ağın diğer elemanları ile paylaşır. Bu aşamalarda ihtiyaca göre değişik toplama ve aktivasyon fonksiyonları kullanılabilir [53].



Şekil 3.2 Biyolojik sinir hücresi – nöron [54]

- Tek katmanlı öğrenme- perseptron

Perseptron tek katmandan oluşan en basit sinir ağ yapısıdır. Perseptron, 1958'de Frank Rosenbatt tarafından bulunmuştur. Bu ağ yapısı yanlızca giriş ve çıkış katmanlarından oluşur. Eğitilebilecek tek bir yapay sinir hücresine sahiptir. Şekil 3.3 ile perseptron yapısı verilmiştir.



Şekil 3.3 Persepton yapısı

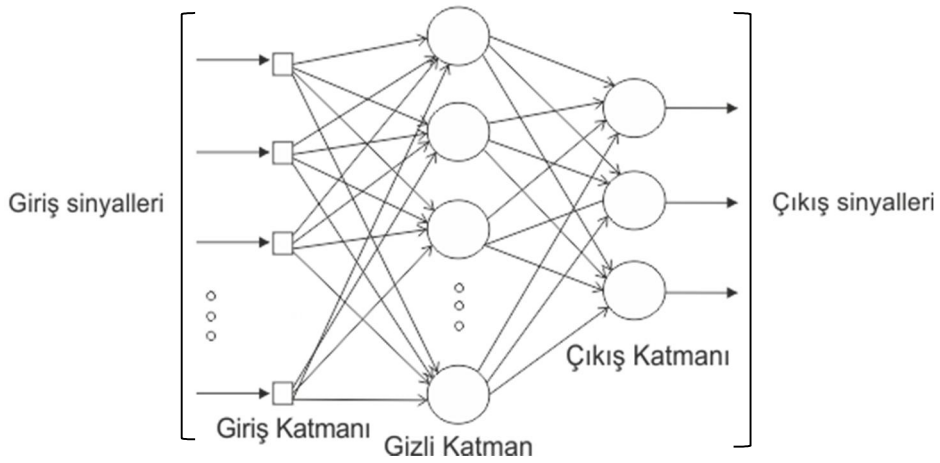
Bir yapay sinir ağının, girdilerden doğru çıktıları üretebilmesi için ağırlık değerlerinin doğru olarak belirlenmesi gerekir. Sistemin doğru çıktıları üretmesi için doğru ağırlıkların belirlenmesi işlemi, sistemin eğitilmesi ile yapılır. Bu ağırlık değerleri başlangıçta rastgele değerler olarak atanır ve ağın öğrenme kuralına göre güncellenirken doğru ağırlık değerleri saptanır [53].

Yapay sinir ağının öğrenmesi için problem uygun ağ modelinin seçilmesi önemli rol oynar. Ağ modeli oluşturulurken; ağın topolojisi, kullanılan toplama ve aktivasyon fonksiyonunun özelliği, öğrenme stratejisi ve kuralları göz önünde bulundurulmalıdır [53].

Perseptron yapısı hata tabanlı öğrenme yapar. Giriş değerleri başlangıçta rastgele verilen ağırlık değerleriyle bir çıkış üretir. Üretilen bu çıktı değeriyle, beklenen değer arasındaki fark yani hata değerine göre ağırlıklar güncellenir ve kabul edilebilir hata değerine ulaşılana kadar bu adımlar güncellenmiş ağırlık değerleriyle tekrarlanır. Bu yapının basit sınıflandırma ve kümeleme problemlerini çözerken, problemler zorlaştıkça başarısız olduğu görülmüştür. Perseptron modeliyle yalnızca lineer olarak ayrıştırılabilen sınıflar ayrıştırılabilirken, XOR problem gibi lineer olarak ayrıştırılamayan problemlerin çözülememesi yapay sinir ağları alanında uzun bir dönem çalışmaların durmasına sebep olmuştur [57].

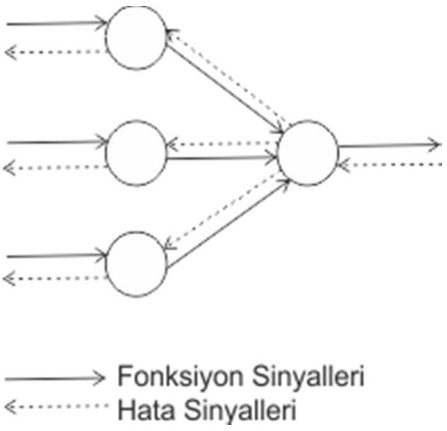
- Çok katmanlı öğrenme- çok katmanlı perseptron

1980'li yıllarda yapay sinir ağları çok katmanlı perseptron yapısı ile tekrar popüler hale gelmiştir. Bu yapısı temelde 3 kısımdan oluşur. Bunlar giriş katmanı, gizli katman ve çıkış katmanıdır. Şekil 3.4 ile bir çok katmanlı ağ yapısı gösterilmiştir.



Şekil 3.4 Çok katmanlı perseptron ağ yapısı

Perceptron yapısında kullanılan gizli katman sayısı ve kullanılacak nöron sayısı problemin yapısına göre deneme yanılma yöntemiyle saptanır. Çok katmanlı perseptron yapısında da hata tabanlı öğrenme yapılır. Çok katmanlı perseptron yapısında hata sinyalleri Şekil 3.5'de gösterildiği gibi geri yayılım ile tüm ağa yayılır ve bu hata değerlerine göre ağırlıklar yeni değerleriyle güncellenir [57].



Şekil 3.5 Hata sinyalinin geri yayılımı

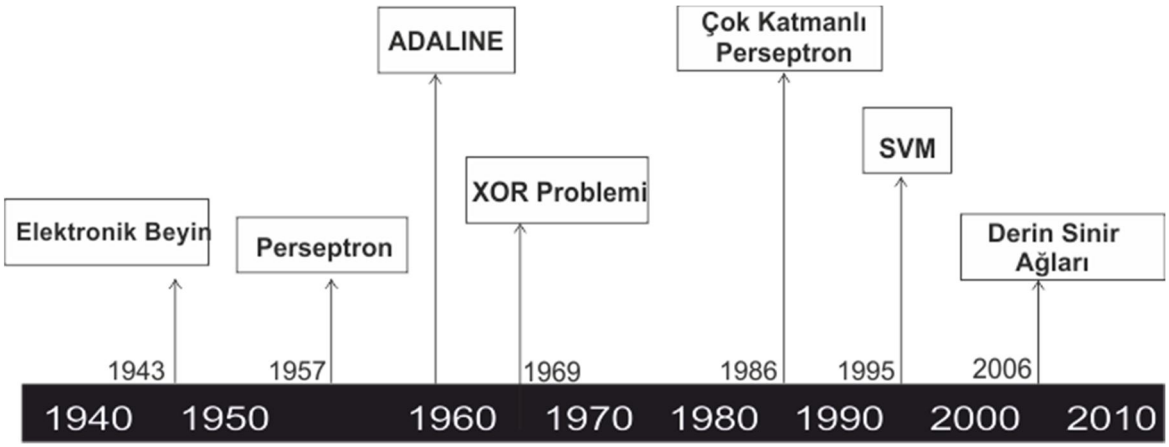
3.4.2. YSA'larının kullanıldığı çalışmalar

Yapay sinir ağlarının sınıflandırma, kümeleme, örüntü tanıma, veri sıkıştırma optimizasyon, veri madenciliği, optimum rota belirleme, iş çizelgesi hazırlama gibi günlük hayatımızda kullandığımız pek çok uygulama için oldukça başarılı bir teknik olduğu söylenebilir.

3.4.3. YSA'nın tarihsel gelişimi

Yapay Sinir Ağları alanında ilk çalışmanın S. McCulloch ve W. Pitts tarafından 1940'lı yıllarda yapılan elektronik beyin çalışması olduğu bilinmektedir. Şekil 3.6 ile gösterilen yapay sinir ağlarının tarihsel gelişimine bakıldığında, 1970 yılının bu alanda yapılan çalışmalarda bir dönüm noktası olduğu görülmektedir. Bu tarihten önce bir çok araştırmanın yapılmış ve 1969 yılında XOR probleminin çözülememesi nedeni ile araştırmalar durmuştur. 1970 yılından sonra XOR probleminin çözülmesi sonucunda yapay sinir ağlarına olan ilgi yeniden alevlenmiştir. İzleyen yıllar içinde birbirinden farklı yeni model geliştirilmiştir [55].

Derin öğrenme ve Derin Yapay Sinir Ağları ile ilgili çalışmalar G. Hinton ve S.Ruslan tarafında 2006 yılında başlamış ve halen günümüzde bu konuyla ilgili çalışmalar devam etmektedir [56].



Şekil 3.6 Yapay sinir ağlarının tarihsel gelişimi

3.5. Derin Öğrenme

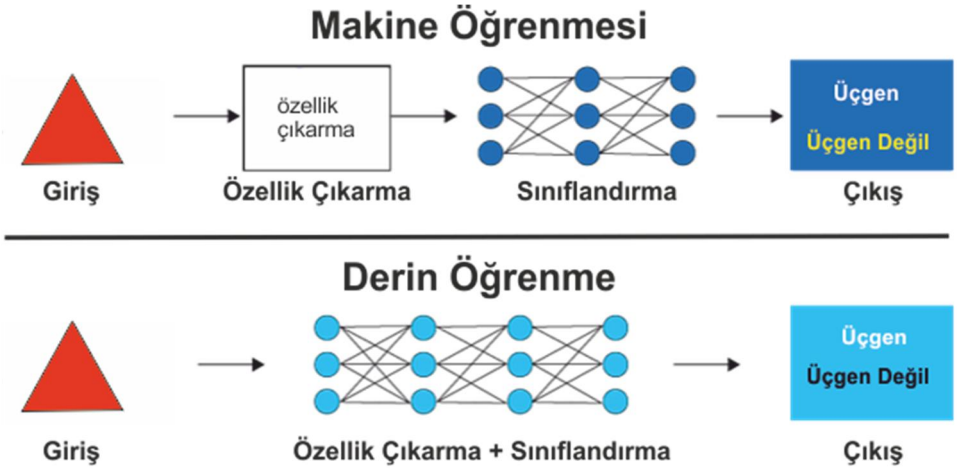
3.5.1. Genel bilgiler

Derin öğrenme, insan beyninin karmaşıkk problemleri çözmek için kullandığı yöntem ve kabiliyetlerinin örnek olarak alan, büyük miktarda veriden faydalanarak özellik çıkarma, sınıflandırma ve dönüştürme işlemlerini yapma yeteneğine sahip bir makine öğrenmesi tekniğidir. Derin öğrenme, büyük miktarda denetimsiz veri kullanarak sınıflandırma problemlerinde özellik çıkarımı maliyetini ortadan kaldırmış, yapay sinir ağlarının özelleştirilmiş pek çok gizli katmandan ve işlem elemanından oluşan bir çeşididir.

3.5.2. Makine öğrenmesinden farklılıklar

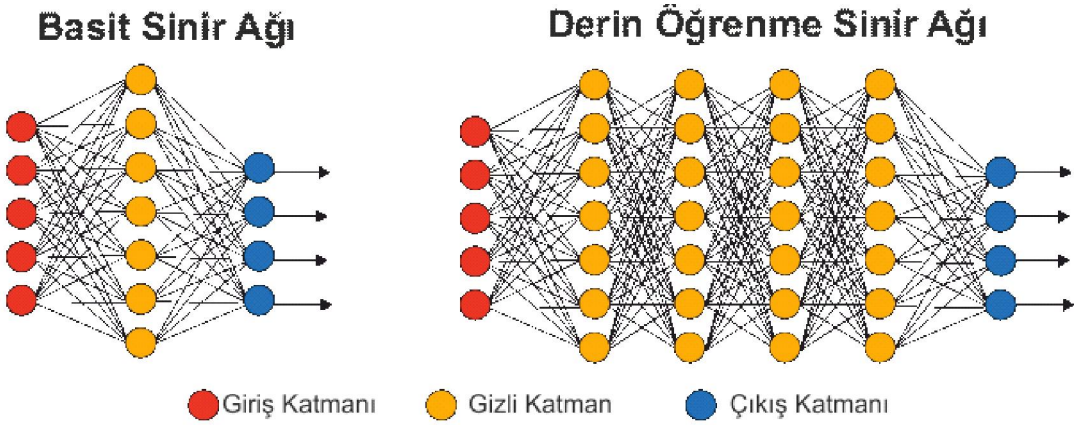
Geleneksel makine öğrenmesi ile derin öğrenme arasındaki en temel fark otomatik özellik çıkarımı kabiliyetidir. Bu özellik çıkarımının yapılabilmesi için derin öğrenme algortimaları büyük bir veri setine ihtiyaç duyar. Şekil 3.7'de de görülebileceği gibİ geleneksel makine öğrenme modelinde sınıflandırma yapmadan önce her sınıfın özelliğinin belirtilmesi gerekirken, derin öğrenme de bu özellikler otomatik olarak çıkartılır ve öğrenilir. Başka bir deyişle, derin öğrenme de denetimsiz öğrenme yapma kabiliyeti vardır.

Çok katmanlı perseptron yapısında, ikiden fazla gizli katman kullanılmazken, derin öğrenme sinir ağ yapısında çokça gizli katman kullanılır. Her iki ağ yapısıda hata tabanlı öğrenme yapsada, çok katmanlı perseptron yapısında denetimli öğrenme yapılırken, derin öğrenmede denetimsiz öğrenme yapılabilir.



Şekil 3.7 Makine öğrenmesi ve derin öğrenmenin farkı

Derin öğrenme ağ yapısını basit bir yapay sinir ağından ayıran en temel özellik Şekil 3.8'den görülebileceği gibi birden fazla gizli katmana sahip olması ve daha karmaşık bir ağ yapısına sahip olmasıdır.



Şekil 3.8 Basit sinir ağıyla derin öğrenme sinir ağı karşılaştırması [71]

Çizelge 3.1'de Makine öğrenmesi ve derin öğrenmenin veri bağımlılığı, donanımsal bağımlılıklar, özellik çıkarma ve uygulama süresi açısından karşılaştırılması yapılmıştır.Derin öğrenme algoritmalarının, standart bir makine öğrenmesine kıyasla iyi performans gösterebilmesi için çok sayıda veriye ve güçlü donanımlara ihtiyaç duyduğu ve uygulama süresinin buna bağlı olarak çok daha uzun sürdüğü görülmektedir. Tüm bu bağımlılıklar dezavantaj olarak düşünülebilir

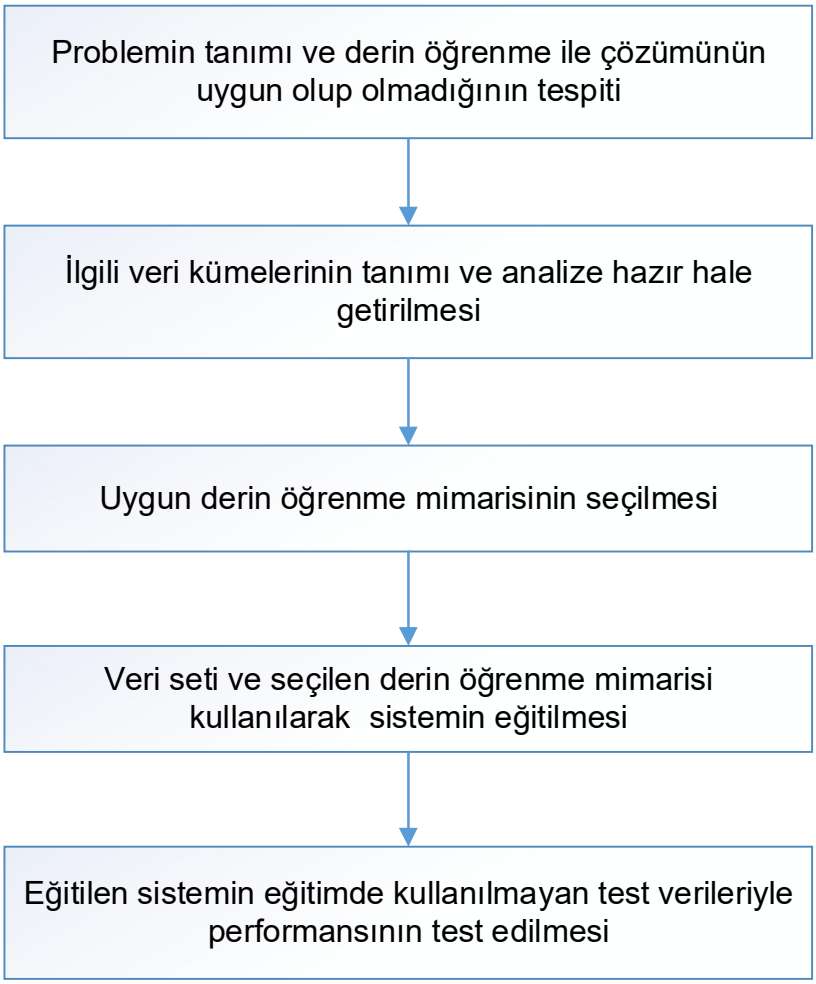
fakat bu bağımlılıklar derin öğrenme algoritmalarının en önemli avantajı olan özellik çıkarma kabiliyetinin maliyeti olarak oluşmuştur.

Çizelge 3.1 Makine öğrenmesi ve Derin öğrenme karşılaştırması

Karşılaştırma Parametresi	Makine Öğrenmesi	Derin Öğrenme
Veri Bağımlılığı	Küçük / orta ölçekli veri kümesinde mükemmel performans	Büyük bir veri setinde mükemmel performans Büyük bir veri setinde mükemmel performans
Donanımsal Bağımlılıklar	Düşük kaliteli bir makinede çalışabilir.	Güçlü bir makine gerektirir, tercihen GPU: çok sayıda matris çarpımı gerçekleştirme kabiliyetine sahip bir donanıma ihtiyaç duyar.
Özellik çıkarma	Verilerin temsil ettiği özellikleri bilmeye ihtiyaç duyar.	Verileri temsil eden en iyi özelliği bilmeye ihtiyaç duymaz.
Uygulama Süresi	Birkaç dakikadan saatlere kadar sürebilir.	Haftalara kadar sürebilir.Bunun sebebi yapay sinir ağının önemli miktarda ağırlık hesaplamasının gerekmesidir.

3.5.3. Derin öğrenme süreçleri

Her problem derin öğrenme ile çözülmeye uygun olmayabilir. Bunun için öncelikle problemin belirlenmesi ve derin öğrenme algoritmalarının kullanımına uygunluğunun değerlendirilmesi gerekir. Eğer uygun ise derin öğrenme için belirli süreçlerin sırasıyla yapılması gerekir. Derin Öğrenme süreçleri Şekil 3.9'da verilmiştir.



Şekil 3.9 Derin öğrenmenin süreçleri [58]

3.5.4. Derin öğrenme için kullanılabilecek işletim sistemleri

Derin öğrenme için farklı işletim sistemleri kullanılanbilir. Windows, Linux, Mac OSX, Android ya da iOS işletim sistemleri derin öğrenme çalışmaları için kullanılabilir. Bu tez çalışmasında Windows işletim sistemi kullanılmıştır.

3.5.5. Derin öğrenme için yaygın olarak kullanılan programlama dilleri

Derin öğrenme çalışmaları için çeşitli programlama dilleri kullanılsada bunlardan en yaygın olarak kullanılanları Python, C++, Java, MATLAB programlama dilleridir. Bu tez çalışmasında Python programlama dili kullanılmıştır.

3.5.6. Derin öğrenme için kullanılabilecek kütüphaneler

Derin öğrenme çalışmalarında kullanılan programlama diline göre kullanılabilecek pek çok açık kaynak derin öğrenme kütüphanesi vardır. Bunlar içinde en yaygın olarak kullanılanları Çizelge 3.2 ile verilmiştir.

Çizelge 3.2 Derin öğrenme kütüphanleri

Kütüphane	Arayüz	Sahibi
TensorFlow	Python, C++, Java	Google
Theano	Python,C++	Montreal Institute for Learning Algorithms(MILA)
Caffee	Python, C++, Matlab	Berkeley Vision and Learning Center (BVLC)
Torch/PyTorch	Lua, Python	Ronan Collobert Diğerleri
CNTK	Python, C++	Microsoft
Deeplearning4j	Java, Scala, C	Skymind
MatConvNet	Matlab	Andrea Vedaldi, Karel Lenc

3.5.7. Derin öğrenme mimarileri

- Tekrarlayan sinir ağları

Tekrarlayan sinir ağlarında, gizli katman çıkışını hem bir sonraki katmana hem de tekrar aynı katmana giriş olarak verir. Ağın bu yapısal tekrarlama özelliğinden dolayı ismi tekrarlayan sinir ağıdır. Bu ağ yapısının yapısal özelliği dolayısıyla ses veri setleri üzerinde oldukça başarılı olduğu bilinmektedir. Sınıflandırma problemlerinde kullanılabilir bir derin ağ mimarisidir.

- Sınırlı boltzman makineleri

Sınırlı Boltzman Makineleri veri setindeki olasılık dağılımlarını öğrenebilme kabiliyetine sahiptir. Bu yapay sinir ağı temelde iki katmandan oluşur ve rastlantısal bir yapay sinir ağ yapısına sahiptir. Bu ağ yapısı sınıflandırma problemlerinde kullanılabilir.

- Derin oto-kodlayıcılar

Derin oto-kodlayıcılar girdi verisine en fazla benzerliğe sahip çıkışı üretecek, veri ile ilişkili önemli bilgilere sahip fonksiyonu bulurlar. Derin oto-kodlayıcılar sınıflandırma işlerini yapamamaktadır. Algortimanın kullanım amacı genellikle özellik vektörlerini asgari kayıpla alt örnekleme yapmaktır.

- Konvolüsyonel sinir ağları

Memelilerin görme sistemini örnek alan konvolüsyonel sinir ağı, çok sayıda konvolüsyon işlemi ve örnekleme katmanına sahip ileri beslemeli bir ağ yapısıdır. Konvolüsyonel sinir ağları görüntü sınıflandırma, nesne tanımlama gibi görüntü tabanlı çalışmalarda çok başarılıdır.

Bu tezde görüntü veri seti ile çalışıldığı için uygun görülüp seçilen derin öğrenme mimarisi görütüntü sınıflandırma başarısı sebebiyle konvolüsyonel yapay sinir ağı mimarisidir.

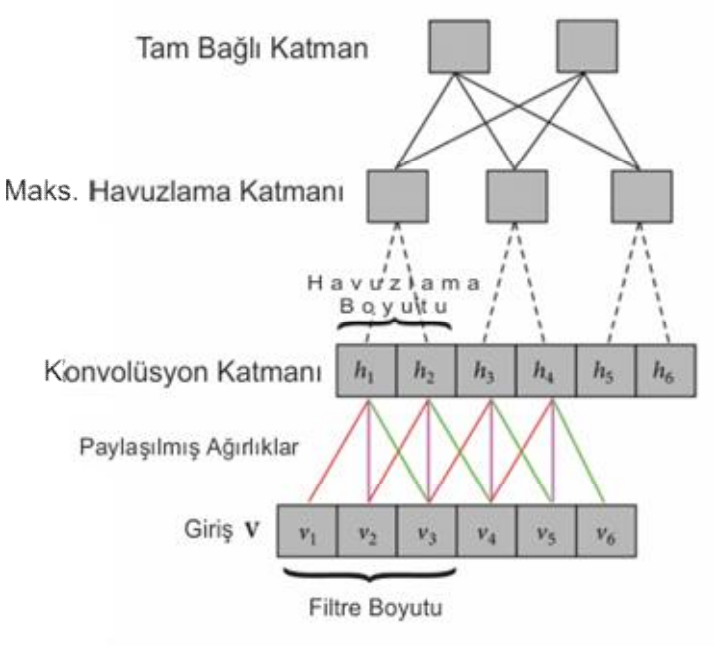
3.6. Konvolüsyonel Sinir Ağları Modeli

3.6.1. Genel bilgiler

İnsanlar bir nesne görüntüsüne baktığında o nesnenin rengi, boyutu, benzer tipteki nesneleri zahmetsizce ayırt edebilir. Aynı görüntülere bilgisayarlar baktığında ise bu görüntüyü bir matris olarak değerlendirir. Görüntüdeki her piksel için matriste bir değer o piksele göre değişmektedir. Konvolüsyonel yapay sinir ağları çok sayıda gizli katmanda bu matrisleri işleyerek farklı özelliklerini algılar ve böylece nesneleri kolay bir şekilde ayırt edebilir.

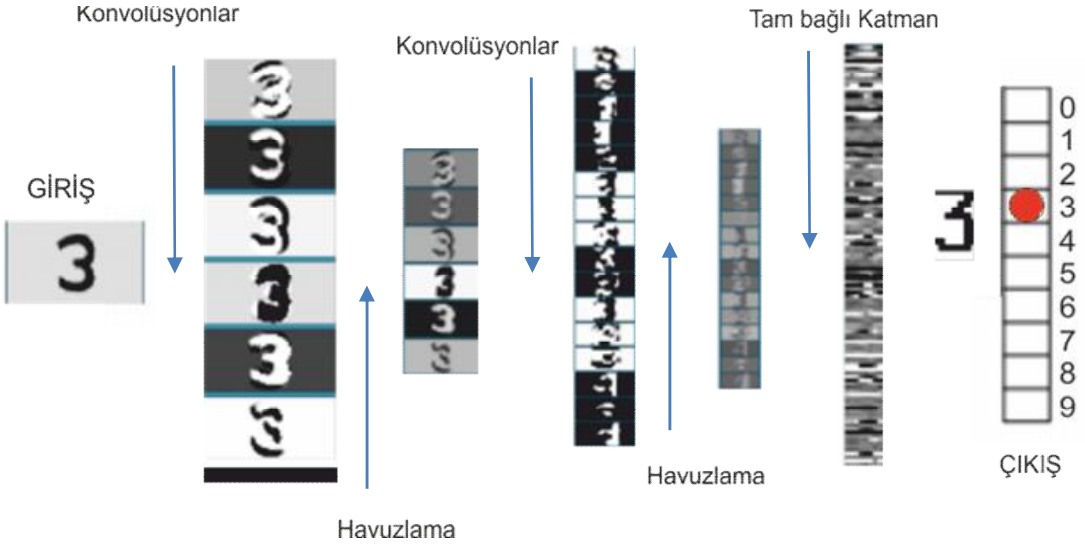
3.6.2. Basit bir KYSA Yapısı ve Katmanları

Konvolüsyonel yapay sinir ağının temel yapısı Şekil 3.10'de verilmiştir [59]. Eğitim için kullanılacak verinin dinamiğine ve yapısına göre bu katmanlar ve kullanılan katmanların sayısı sistemi tasarlayan kişi tarafından değiştirilebilir.



Şekil 3.10 KYSA temel yapısı [59]

Temelde KYSA 3 ana katmandan oluşur.Bunlar uygulama sırasına göre, Konvolüsyon katmanı, Havuzlama katmanı, en son olarak tam bağlantılı katmanıdır. Şekil 3.11 ile bir konvolüsyonel Sinir ağı ile bir görüntünün katmanlarda işlenmesi ve sınıflandırılması gösterilmiştir.



Şekil 3.11 Konvolüsyonel sinir ağı ile bir görüntünün sınıflandırılması [60]

• Konvolüsyon katmanı

Konvolüsyon katmanının birincil amacı giriş olarak verilen görüntülerden özelliklerin çıkarılmasını sağlamaktır. Konvolüsyon pikseller arasındaki uzamsal ilişkiyi koruyarak giriş verisi üzerinden görüntü özelliklerini öğrenir.

Daha önce de bahsedildiği gibi he görüntü piksel değerlerinden oluşan bir matris olarak ifade edilir. Elimizde Şekil 3.12'deki gibi pixel değerleri 1 ve 0 lardan oluşan 5x5 bir görüntü olduğunu varsayalım. KYSA terminolojisinde Şekil 3.13 ile verilmiş 3x3 matris "Filtre", "kernel" ya da "özellik çıkarcı" gibi farklı isimlerle tanımlanır. Bu iki matrise konvolüsyon işlemi özellik çıkarıcı matris görüntü matrisi üzerinde Şekil 3.14'de gösterildiği gibi kaydırılarak nokta çarpımı uygulanması sonucu oluşan Şekil 3.15 ile gösterilen "Konvolüsyon uygulanmış özellik", "Aktivasyon haritası" ya da "Özellik haritası" gibi farklı isimlerle tanımlanan matris elde edilir.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Şekil 3.12 Görüntü matrisi

1	0	1
0	1	0
1	0	1

Şekil 3.13 Özellik çıkarıcı matris

1 x1	0 x0	1 x1	0	0
0 x0	1 x1	0 x0	1	0
1 x1	0 x0	1 x1	1	1
0	0	1	1	0
0	1	1	0	0

Şekil 3.14 Özellik haritasının çıkarılması

4	3	4
2	4	3
2	3	4

Şekil 3.15 Özellik Haritası

Filtreme için kullanılan özellik çıkarıcı matrisin değerlerine göre görüntü üzerinde farklı özellikler çıkartılabilir. KYSA bu filter değerlerini öğretme aşamasında kendi öğrenir. Fakat uygulanacak filtrenin sayısı, boyutu ve ağın yapısı gibi parametrelerin sitemi tasarlayan kişi tarafından belirlenmesi gerekmektedir.

Aşağıda Çizelge 3.3 ile farklı özellik çıkarıcı matrislerin görüntü üzerinde çıkardığı özellikler için örnekleri görebiliriz.

Çizelge 3.3 Farklı filtrelerin çıkardığı özellik haritaları

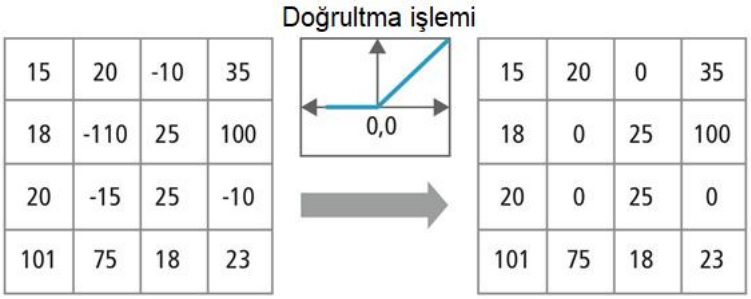
Operasyon	Filtre (Özellik çıkarıcı)	Özellik Haritası
Kimlik	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	
Kenar algılama	$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$	
keskinleştirme	$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$	
Bulanıklaştırma (Normalize edilmiş)	$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	

Konvolüsyon katmanının hemen sonunda düzleştirme başka bir deyişle doğrultma işlemi uygulanır. Bu işlem, konvolüsyonel sinir ağlarının sürecinin ayrı bir bileşeni değildir. Bazı kaynaklarda bu adım ayrı bir katman olarak ifade edilsede aslında bu adımı konvolüsyon katmanının tamamlayıcı adımı olarak düşünebiliriz.

Doğrultucu işlevini uygulamanın amacı, görüntülerimizdeki doğrusal olmayanlığı arttırmaktır. Bunu yapmak istememizin nedeni görüntülerin doğal olarak doğrusal olmamasıdır. Herhangi bir görüntüye baktığınızda, birçok doğrusal olmayan özellik (örneğin pikseller, kenarlıklar, renkler arasındaki geçiş vb.) içerdiğini görürüz.

Böylece konvolüsyonel yapay sinir ağına doğrusal olmayan bir yön vermiş oluruz, çünkü ağın öğrenmesini isteyeceğimiz gerçek dünya verilerinin çoğu doğrusal olmayan verilerdir. ReLU, piksel başına uygulanır ve özellik haritasındaki tüm

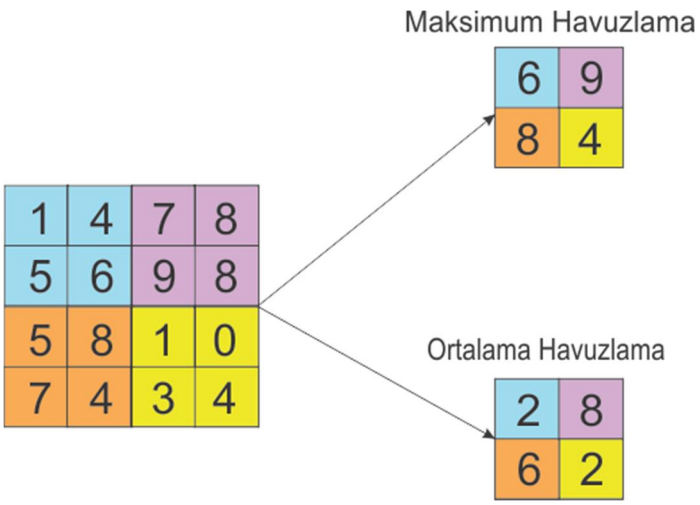
negatif piksel değerlerini sıfır olarak değiştirir.Şekil 3.16' da doğrultma işleminin yapılması gösterilmiştir.



Şekil 3.16 Doğrultma işleminin uygulanması

- Havuzlama katmanı

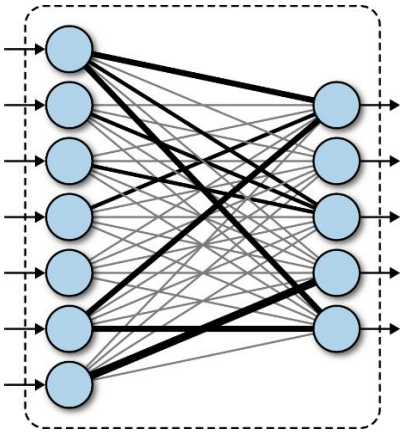
Havuzlama katmanı, ortalama veya maksimum değeri alarak özellik haritalarının boyutunu azaltır. Havuzlama, giriş boyunca bir pencere kaydırıp pencerenin içeriğini bir havuzlama işlevine besleyerek çalışır. Havuzlamanın amacı, ağımızdaki parametre sayısını azaltmak (bu nedenle aşağı örnekleme olarak adlandırılır) ve ölçeklendirme ve yönlendirme değişikliklerini daha değişken hale getirerek öğrenilmiş özellikleri daha sağlam hale getirmektir. Şekil 3.17 ile maksimum ve ortalama havuzlama işleminin nasıl gerçekleştiği gösterilmiştir.



Şekil 3.17 Maksimum ve ortalama havuzlama

- Tam bağlantılı katman

Tam Bağlantılı katman, tam olarak adının ima ettiği şekilde yapılandırılır. Kendisinden önceki katmanın çıktısına tam olarak bağlanır. Tamamen bağlı bir katman, önceki katmandaki tüm nöronları alır (tamamen bağlı, havuzlanmış veya konvolüsyon) ve onu sahip olduğu her bir nörona bağlar. Tamamen bağlı bir katman eklemin amacı, bu özelliklerin doğrusal olmayan kombinasyonlarını öğrenilebilmesidir. Konvolüsyonel ve havuzlayıcı katmanlardan öğrenilen özelliklerin çoğu iyi olabilir, ancak bu özelliklerin kombinasyonları daha da iyi olabilir. Şekil 3.18'de tam bağlı katmanın gösterimi verilmiştir.

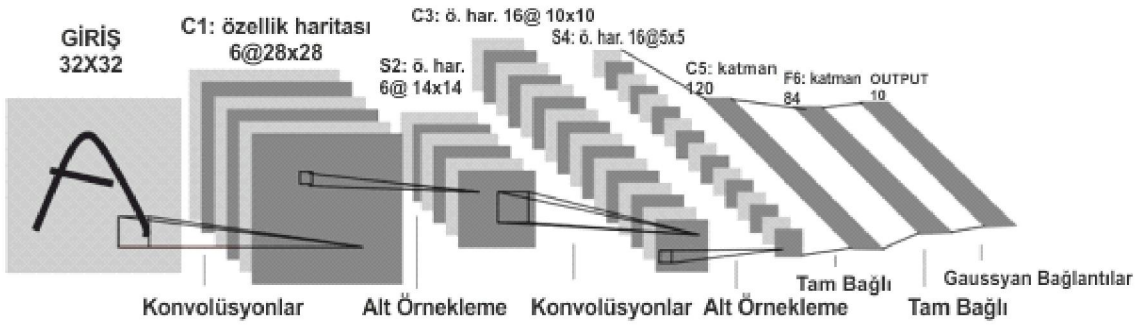


Şekil 3.18 Tam bağlı katman

3.6.3. En yaygın kysa yapıları

- LeNet

Yann Lecun'un LeNet-5 modeli, posta hizmetinde posta kodu tanıma için el yazısı rakamları tanımlamak amacıyla 1998'de geliştirilmiştir [60]. Bu model, bugün bildiğimiz konvolüsyonel yapay sinir ağının ilk başarılı modelidir. Şekil 3.19 ile LeNet yapısı verilmişitr.

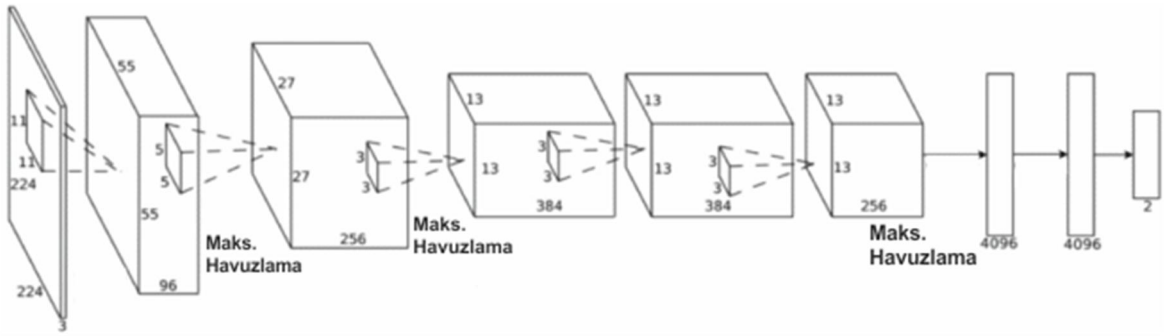


Şekil 3.19 LeNet Yapısı [60]

Bu yapıda alt örnekleme yapmak için ortalama havuzlama yapılır [60].

- AlexNet

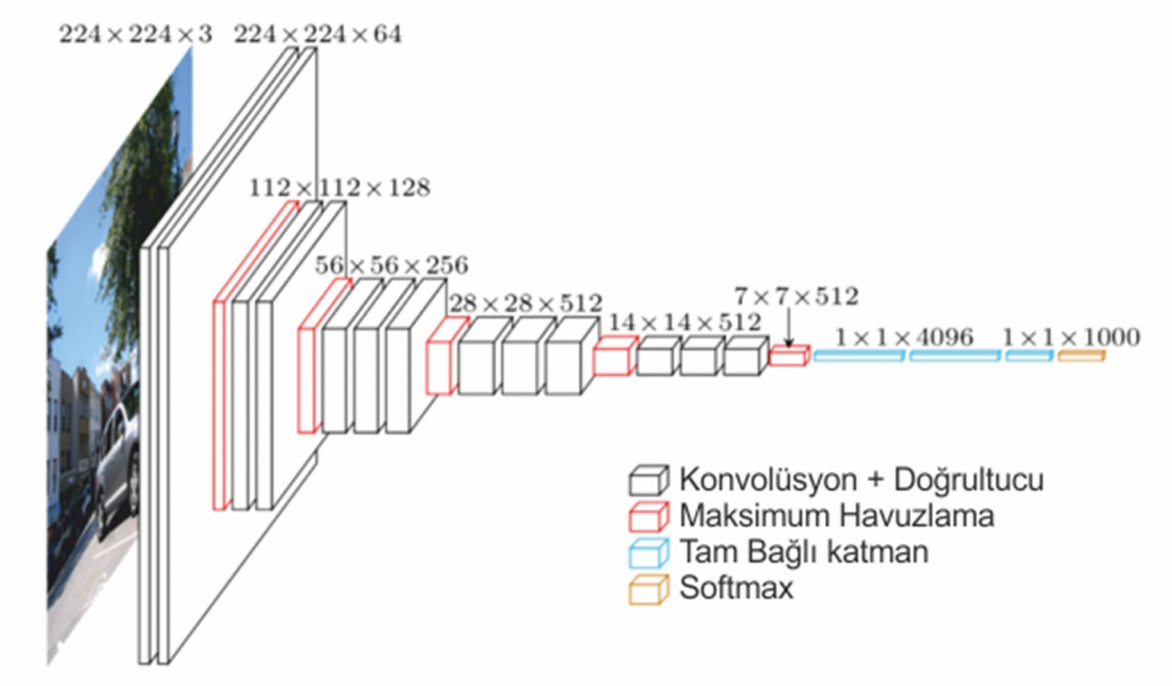
AlexNet, Alex Krizhevsky ve arkadaşları tarafından 2012 yılında ImageNet yarışmasında yarışmak için geliştirildi [61]. Bu model oldukça büyük olmasına rağmen, genel mimari LeNet-5'e oldukça benziyordu. Bu model 2012 ImageNet yarışmasında birinciliği aldı. Şekil 3.20 ile AlexNet Yapısı verilmiştir.



Şekil 3.20 AlexNet Yapısı [61]

- VGGNet

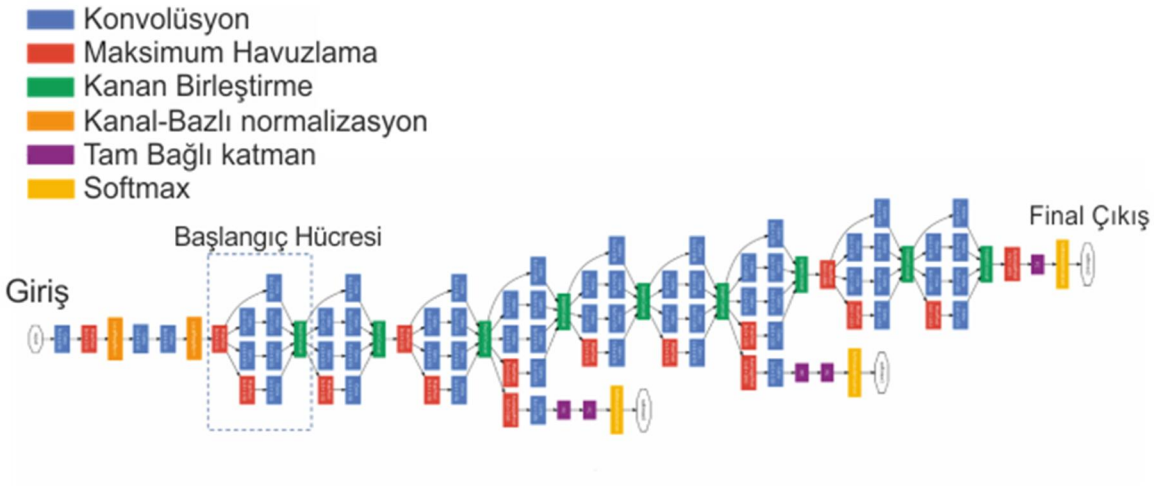
2014 yılında tanıtılan VGG ağı, yukarıda tartışılan evrişim yapılarının daha derin fakat daha basit bir türevi sunar. Şekil 3.21 ile VGGNet yapısı verilmiştir [62].



Şekil 3.21 VGGNet Yapısı [62]

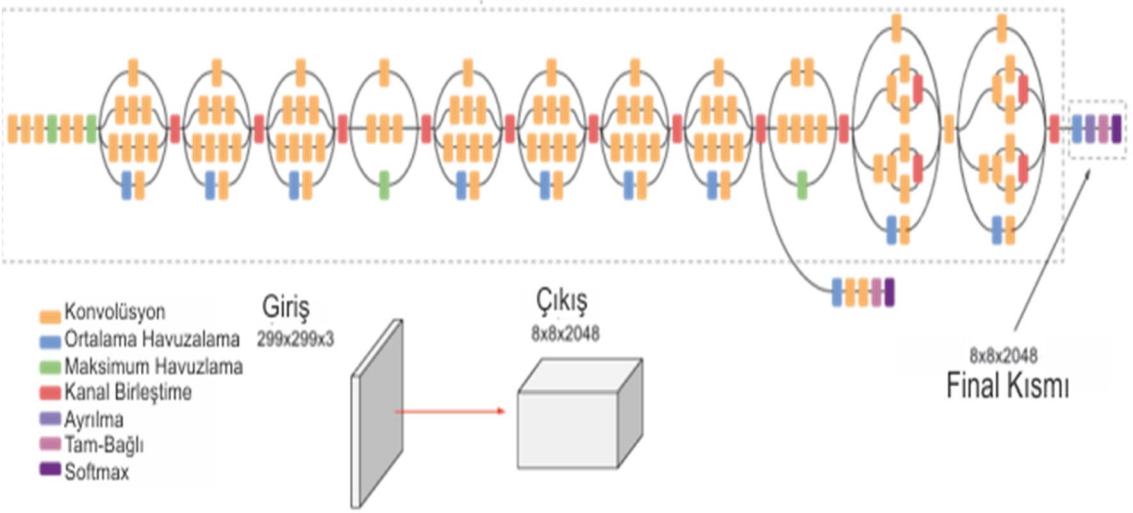
• Inception (GoogLeNet)

2014 yılında, Google'daki araştırmacılar, sınıflandırma ve tespit zorlukları için 2014 ImageNet yarışmasında birincilik kazanan Inception ağını tanıttı. Model, farklı ölçeklerde bir dizi konvolüsyon gerçekleştirdiğimiz ve ardından sonuçları topladığımız Başlangıç hücresi (Inception Cell) olarak adlandırılan temel bir birimden oluşur. Şekil 3.22 ile Inception V1 yapısı verilmiştir [63].



Şekil 3.22 Inception V1 yapısı [63]

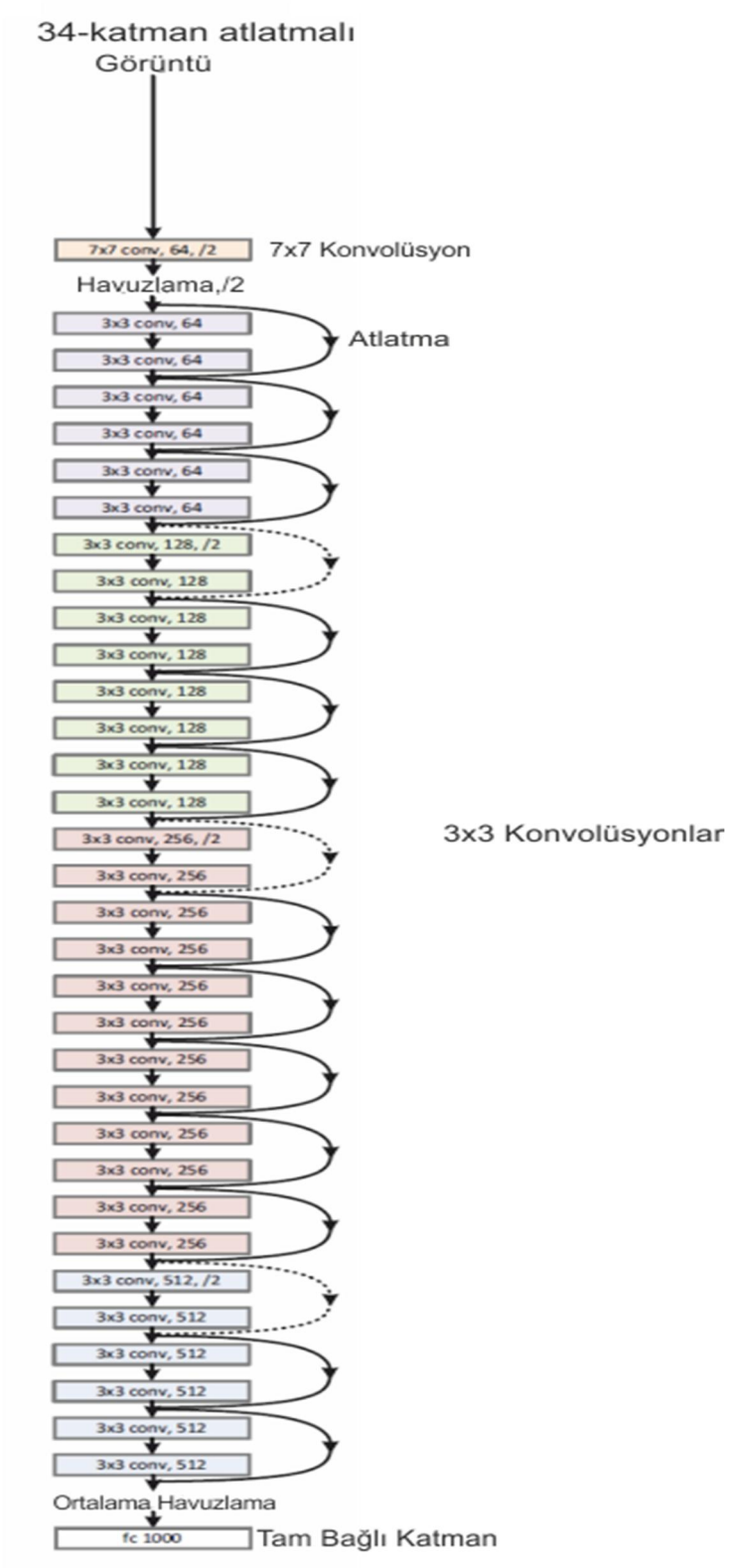
Inception ağının daha verimli Inception hücrelerinden yararlanan gözden geçirilmiş, daha derin bir versiyonu oluşturulmuştur. Şekil 3.23 ile Inception v3 yapısı verilmiştir [64].



Şekil 3.23 Inception V3 yapısı [64]

- ResNets

2016 yılında Inceptionv3 ile aynı anda geliştirilen ResNet yapısında, art arda iki konvolüsyon katmanın çıktısını beslerken, girdiyi bir sonraki katmana atlatır [65]. Şekil 3.24 ile ResNet yapısı verilmiştir.



Şekil 3.24 ResNet Yapısı [65]

4. TİD ALFABESİNİ GERÇEK ZAMANI TANIMLAYAN SİSTEM YAPISI

Bu tez çalışmasında Imagenet Inception-V3 önceden eğitilmiş konvolüsyonel yapay sinir ağı modeli kullanılmıştır. Sistemin yapısı tasarlanırken, sistemin eğitimi için gerekli bir takım donanımsal kaynakların kısıtlılığı ve sistemi eğitmek için gerekli zamandan tasarruf edilmesi göz önünde bulundurulduğunda daha önce pek çok çalışmada kullanılmış ve başarı elde etmiş öğrenmeyi transfer etme yöntemini [42,43,44,45,69,70] kullanarak önceden eğitilmiş bir model kullanmaya karar verilmiştir.

4.1. Öğrenmeyi Transfer Etme

Bir öğretmenin öğrettiği belirli konuda yıllar sonunda edindiği bir tecrübesi vardır. Tüm bu birikmiş bilgileri kullanarak öğretmen, öğrencilere konuyla ilgili kısa ve öz bir bakış niteliğindeki bilgiyi aktarır. Bu analojiyi sinir ağlarına uyarlamak mümkündür. Bir sinir ağı verilerle eğitilmiştir. Bu ağ, ağın ağırlıkları olarak derlenen bu verilerden bilgi kazanır. Bu ağırlıklar çıkarılabilir ve daha sonra herhangi bir sinir ağına aktarılabilir. Yani diğer sinir ağını sıfırdan eğitmek yerine, öğrenilen özellikleri aktarabiliriz.

Bir sinir ağı oluştururken hedefimiz ağ için doğru ağırlıkları belirlemektir. Daha önce büyük veri kümelerinde eğitilmiş modelleri kullanarak, elde edilen ağırlıkları ve mimariyi doğrudan kullanabilir ve öğrenmeyi kendi problemimiz üzerine uygulayabiliriz. Bu transfer öğrenme olarak bilinir. Başka bir deyişle, önceden eğitilmiş modelin öğrenmesini kendi özel durumumuza aktarırız.

4.2. Önceden Eğitilmiş Model

Önceden eğitilmiş bir model, benzer bir sorunu çözmek için başkası tarafından yaratılan bir modeldir. Benzer bir sorunu çözmek için sıfırdan bir model oluşturmak yerine, başka bir konuda eğitilmiş modeli başlangıç noktası olarak kullanabiliriz. Önceden eğitilmiş bir model uygulamanızda %100 doğru olmayabilir, ancak tüm mimariyi yeniden oluşturup eğitmek için gereken büyük çabaları azaltır. Ayrıca, önceden eğitilmiş bir modeli kullanmanın en büyük yararı, yoğun katmanı daha yüksek hassasiyetle eğitmek için neredeyse göz ardı edilebilecek zamandır.

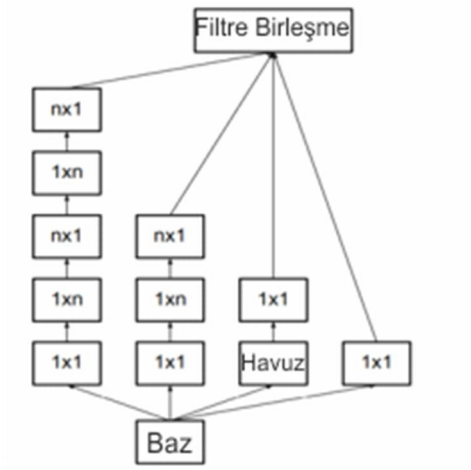
Önceden eğitilmiş bu ağlar, aktarma öğrenmesi yoluyla ImageNet veri kümesi dışındaki görüntülere genelleme yapma yeteneği gösterir. Önceden var olan modelde, modeli ince ayarlarla değiştiririz. Önceden eğitilmiş ağın oldukça iyi birşekilde eğitildiğini varsaydığımızdan, ağırlıkları çok fazla değiştirmek istemeyiz. Değişiklik yaparken, genellikle modeli başlangıçta eğitmek için kullanılandan daha küçük bir öğrenme oranı kullanırız.

TİD alfabesini sınıflandırmak için transfer öğrenmenin gerçekleştirileceği inception v3 modelini, bu modeli TİD alfabesi veri setiyle yeniden eğitmek ve ardından sisteme verilecek yeni görüntüleri sınıflandırabilmek için Tensorflow kütüphanesi [67] kullanılmıştır. Kullanacağımız Inception v3 modeli Şekil 3.22 ile verilmiştir.Modeldeki katmanlar Çizelge 4.1 ile verilmiştir.

Çizelge 4.1 Inception v3 modelindeki katmanlar [43]

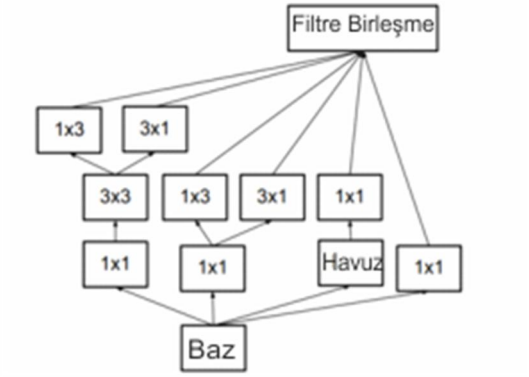
Tip	Boyutlar / adım veya açıklamalar	Giriş boyutu
Konvolüsyon	3×3/2	299×299×3
Konvolüsyon	3×3/1	149×149×32
Konvolüsyon	3×3/1	147×147×32
Havuzlama	3×3/2	147×147×64
Konvolüsyon	3×3/1	73×73×64
Konvolüsyon	3×3/2	71×71×80
Konvolüsyon	3×3/1	35×35×192
3×Inception	Şekil 4.1'deki gibidir.	35×35×288
5×Inception	Şekil 4.2'deki gibidir.	17×17×768
2×Inception	Şekil 4.3'deki gibidir.	8×8×1280
Havuzlama	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Şekil 4.1 Inception modeli-1 [43]



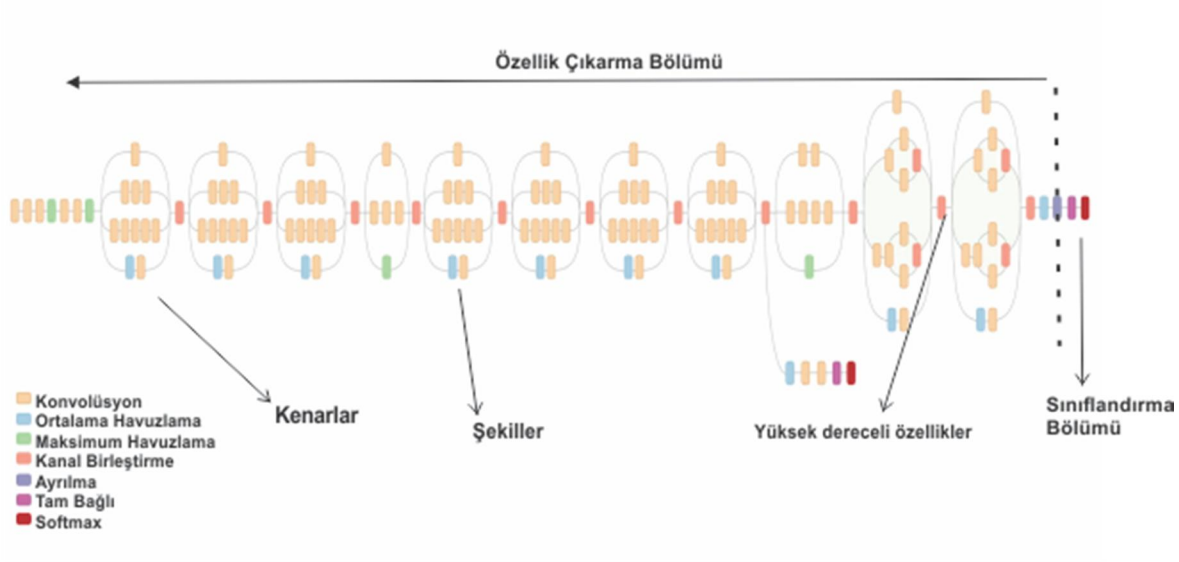
Şekil 4.2 Inception modeli-2 [43]

(17X17 için n=7 olarak verilmiştir.)



Şekil 4.3 Inception modeli-3 [43]

Inception v3 yapısı görüldüğü gibi birden fazla konvolüsyon ve havuzlama işleminden oluşmaktadır. Bu işlemler özellik çıkarma bölümünde yapılırken, Şekil 4.4 ‘de görüleceği gibi Inception v3 modelinde son katmanlara kadar herhangi bir sınıflandırma işlemi uygulanmamaktadır.

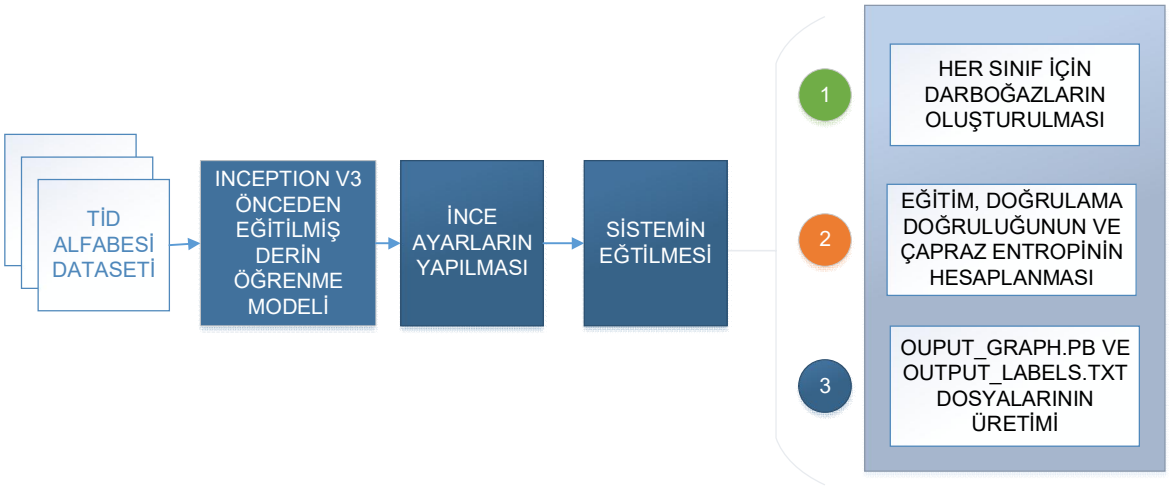


Şekil 4.4 Inception v3 modelinin özellik çıkarma ve sınıflandırma bölümleri [43]

Inception v3 modeli ImageNet veri tabanı kullanılarak eğitilmiştir. ImageNet, Stanford Vision Lab tarafından oluşturulmuş 1000 sınıftan oluşan bir görüntü veritabanıdır [68]. ImageNet, aslen bu TİD alfabesi türlerinin hiçbirinde eğitilmemiştir. Ancak, ImageNet'in 1000 sınıf arasında ayrım yapmasını mümkün kılan bilgi türleri, diğer nesneleri ayırt etmek için de faydalıdır. Önceden eğitilmiş bu ağı kullanarak, bu bilgi TİD alfabesi sınıflarımızı ayırt eden son sınıflandırma katmanına girdi olarak kullanılmıştır.

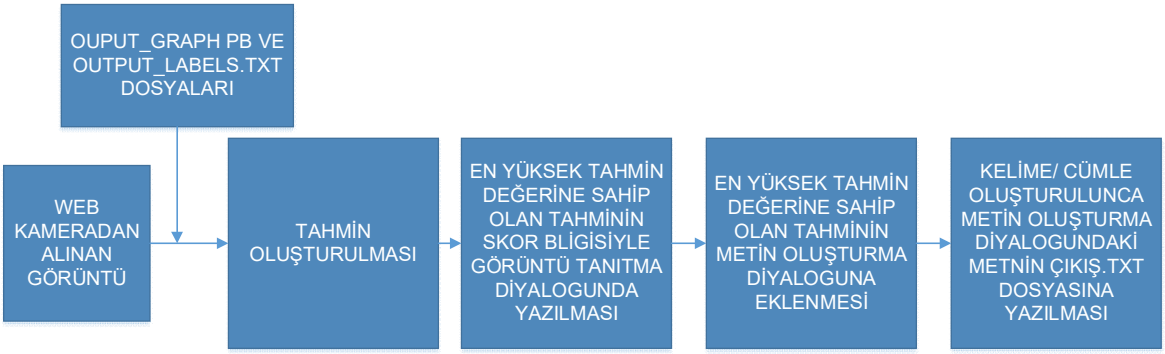
4.3. Sistem Blok Diyagramı

Önceden eğitilmiş Inception V3 konvolüsyonel sinir ağında, öğrenme transferi methodunu kullarak yapılan eğitim için izlenen adımların bir özeti olan sistemin eğitilme aşamaları Şekil 4.5 ile blok şema halinde verilmiştir.



Şekil 4.5 Sistemin eğitim aşamaları

Sistemin eğitimi tamamlandıktan sonra, TİD Alfabesinin gerçek zamanlı tanınması için izlenen adımlar Şekil 4.6 ile gösterilmiştir.



Şekil 4.6 Eğitim sonrası TİD alfabe karakterinin gerçek zamanlı tanınması

4.4. TİD Alfabesi Veri Seti

4.4.1. Genel bilgiler

Veri seti araştırması yapılmış, sonucunda türkçe işaret dili için kaydedilmiş bir veri seti [66] bulunmuştur. Dataset içinde her harf için 105 adet .png uzantılı ve boyutları birbirinden farklı görüntülerin olduğu görülmüş, veri seti kaydedilirken arka planın devamlı değiştiği görülmüştür. Bu veri setinin tasarlanan sisteme uygun olmadığı değerlendirilmiştir.

Bu değerlendirme yapılırken tensorflow kütüphanesinde veri seti özellikleri için dikkat edilmesi gerekn hususlar esas alınmıştır. Dosyaların formatının öğrenimi aktarmak için belirli bir formatta olması gerekmektedir [42, 43, 67]. Proje dosyası altında öncelikle “dataset” adında bir klasör oluşturulmalıdır. Bu klasör, sınıflandırmanın gerçekleştirileceği tüm sınıflar için görüntü veri setlerini içerecektir. Veri kümesi klasörünü oluşturulduktan sonra, tüm veri kümeleri için görüntüleri Şekil 4.7 deki gibi isimlendirilmelidir.

```
|
---- /dataset
|    |
|    |
|    ---- /A
|    |    A1.jpg
|    |    A2.jpg
|    |    ...
|    |
|    |
|    ---- /B
|         B1.jpg
|         B2.jpg
|         ...
|
```

Şekil 4.7 Veri seti klasörünün yapısı ve dosya isimlendirmesi

Böylece, resimlerin A ve B veri kümeleri arasında sınıflandırılmasını sağlanmış olacaktır. Hazır veri setinin istenilen şekilde kaydedilmemiş olduğu ve yeterli sayıda veriye sahip olmadığı değerlendirilmiştir. Bunun üzerine Türkçe İşaret Dili Alfabesi veri setinin istenilen formatta sıfırdan kaydedilmesine karar verilmiştir.

Veri Setini kaydetmek için JetBrains PyCharm IDE kullanımıştır. Bunun için Python dilinde dataset oluşturan bir kod yazılmıştır. Önce oluşturulmak istenen dosya ismini soran, projenin /dataset dizini altına bu isimle bir klasör oluşturup, bu klasör içine dosya ismini her seferinde bir arttırarak web kamera üzerinden aldığı görüntüleri kaydeden bir program ile veri seti oluşturulmuştur. Oluşturulan veri setinin özellikleri Çizelge 4.2 ile verilmiştir.

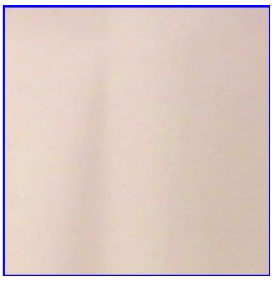
Çizelge 4.2 Veri setinin özellikleri

TİD Alfabe karakter sayısı	29
Metin oluşturma için özel tanımlı sınıf sayısı	3 (Arka plan, silme ve boşluk bırakmak için tanımlanmıştır)
Toplan sınıf sayısı	32
Her sınıf için kaydedilen görüntü sayısı	1500
Veri setindeki toplam görüntü sayısı	48000
Değişkenler	Işık, Yön, Uzaklık
Sınıfların ortalama dosya büyüklüğü	16 MB- 22 MB
Veri seti dosya büyüklüğü	651 MB
Bir sınıf için veri seti kaydetme süresi	2.5-3 dakika
Toplam veri seti oluşturma süresi	≈1 saat 36 dk

Her bir veri seti kümesini 1500'er adet fotoğraf dosyasıyla kayıt etmek, yaklaşık her harf için 2,5 ile 3 dk arası sürmektedir. Veri Setinde Türkçe İşaret Dili alfabesi için 29 adet, özel olarak tanımlanan boş ekran, boşluk bırakma ve silme işlemleriyle beraber toplamda 32 adet klasör ve her klasörde 1500 adet görüntü bulunmaktadır. Veri seti toplamda 48000 adet görüntü dosyasından oluşan, 651 MB boyutunda bir veri setidir. Görüntüler .jpg formatında ve herbiri aynı boyuttadır.Her karakter için dosya boyutları 16 MB- 22 MB rasında değişmektedir.

Bir metin oluşturabilmek için tanımlanan "ARA" ,"YOK", "YOKET" işlemleri için tanımlanmış işaretler Şekil 4.8, Şekil 4.9 ve Şekil 4.10 ile verilmiştir.

Şekil 4.8 ARA işareti



Şekil 4.9 YOK durumu

Şekil 4.10 YOKET işareti

YOK herhangi bir işaret gösterilmediği durumu ifade eder. Bu durumda metin dosyasına herhangi birşey yazılmaz. YOKET işareti yanlış yazılan harfleri silmek için kullanılır.

ARA işareti ise, bir cümle yazılamak istendiğinde, iki kelime arasına boşluk bırakabilmek için kullanılır. TİD alfabesi veri seti örneklerinden bazıları Şekil 4.11, 4.12, 4.13,4.14, 4.15 ile verilmiştir.

Şekil 4.11 A işareti

Şekil 4.12 C işareti

Şekil 4.13 E işareti

Şekil 4.14 G işareti

Şekil 4.15 R işareti

Şekil 4.16 Ü işareti

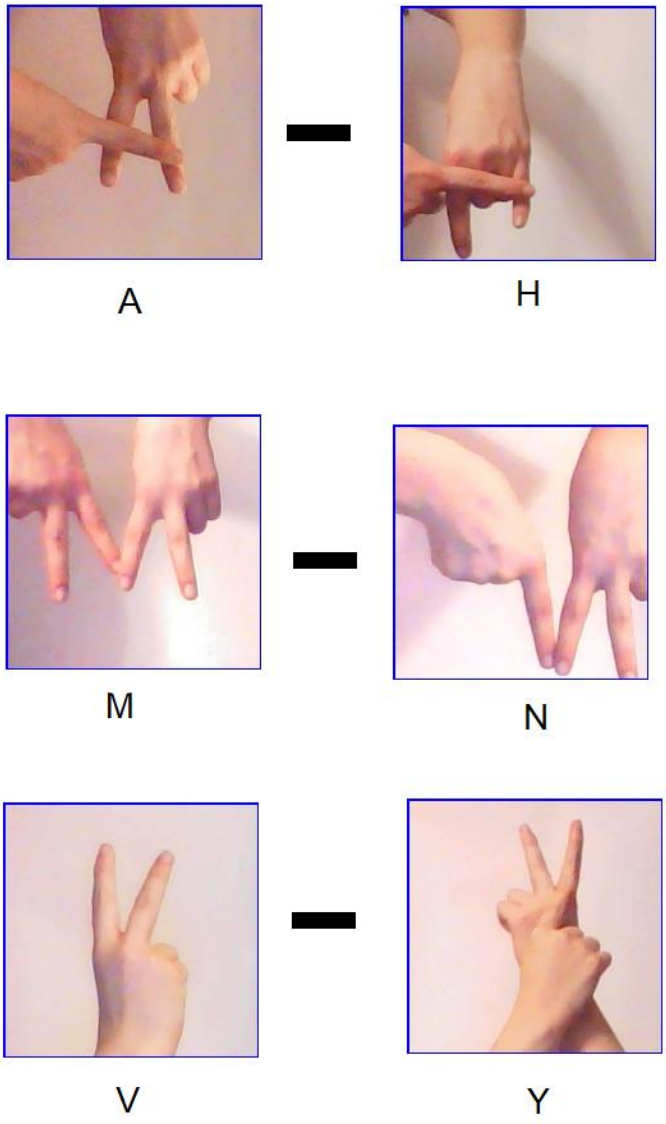
Kaydedilen görüntülerin özellikleri Çizelge 4.3'deki gibidir.

Çizelge 4.3 TİD veri seti görüntü özellikleri

Boyutlar	200 x 200
Genişlik	200 piksel
Yükseklik	200 piksel
Yatay çözünürlük	96 dpi
Dikey Çözünürlük	96 dpi
Bit derinliği	24

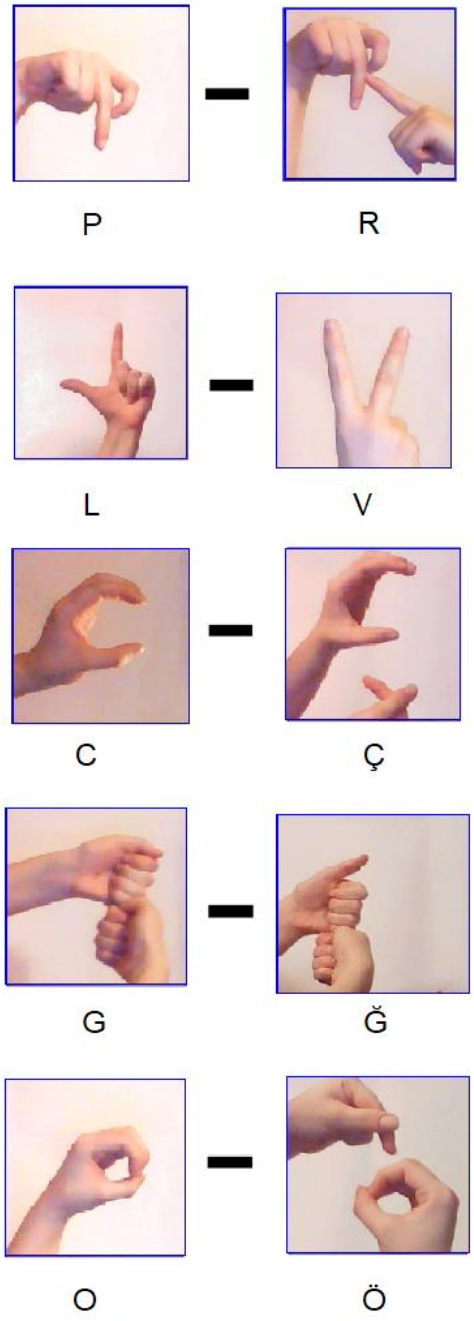
Tanımlı sınıflar altındaki her bir görüntü birbiriyle aynı boyutta olacak şekilde veri seti oluşturulmuştur. Bu görüntüler ışık, uzaklık-yakınlık, pozisyon değişiklikleri yapılarak kaydedilmiştir.

Alfabede bazı karakterler birbirleriyle gösterdikleri simetrik benzerlik sebebiyle tanımada karışıklıklara sebep olmaktadır. TİD Alfabesi için tanımlamada simetrik karışıklık yaşanan harfler Şekil 4.17 ile verilmiştir.



Şekil 4.17 Simetrik karışıklık yaşanan harfler

Alfabede simetrik bir benzerlik olmamasına karşın, bazı harflerde tanımada karışıklık yaşandığı görülmüştür. TİD Alfabesi için tanımlamada asimetrik karışıklık yaşanan harfler Şekil 4.18 ile verilmiştir.



Şekil 4.18 Asimetrik karışıklık yaşanan harfler

Sınıflar için kaydedilen görüntülerin birbirine benzerliğini gösteren sınıfların korelasyon matrisi Çizelge 4.4 ile verilmiştir.

Çizelge 4.4 Sınıfların Korelasyon Matrisi

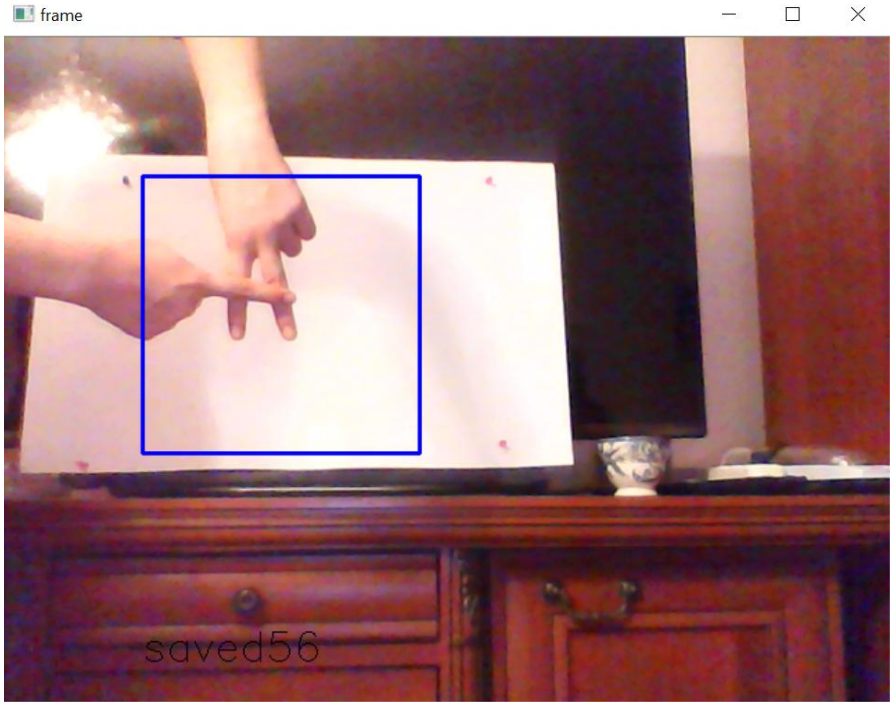
	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	YOKET	YOK	ARA
A	1	0,1	0,18	0	0,097	0,1	0,14	0,1	0,05	0,1	0,03	0,1	-0	0,1	0,04	0,1	0,1	0,1	0,11	0,07	0,1	0,06	0,06	0,11	0	0,1	0,06	0,07	-0	0,0952	0,07	0,07
B	0,07	1	0,14	0,1	0,144	0,2	0,1	0,1	0,11	0,12	0,08	0	0,13	0,1	0,12	0,1	0,1	0,1	0,04	0,18	0,1	0,07	0,08	0,03	0,2	0,07	0,1	0,07	0,19	0,1193	0,09	0,06
C	0,18	0,1	1	0,1	0,248	0,1	0,13	0,1	0,18	0,16	0,08	0,1	0,09	0,1	0,06	0,1	0,1	0	0,08	0,1	0,12	0,03	0,13	0,09	0,1	0,06	0,08	0,05	0,11	0,1407	0,11	0,07
Ç	0	0,1	0,09	1	0,155	0,1	0,02	0,1	0,2	0,11	0,08	0	0,17	-0	0,15	0	-0	0	0,02	0,01	0,01	0,04	0,14	0,06	0,2	0,11	0,09	0,13	0,03	0,0214	0,08	0,12
D	0,1	0,1	0,25	0,2	1	0,2	0,09	0,1	0,19	0,12	0,09	0,1	0,22	0,1	0,21	0	0,1	0,2	0,12	0,13	0,08	-0	0,12	0,08	0,2	0,09	0,15	0,2	0,12	0,1274	0,16	0,22
E	0,09	0,2	0,15	0,1	0,21	1	0,06	0,1	0,24	0,15	0,11	0,1	0,22	0,1	0,18	0,1	0,1	0,2	0,1	0,21	0,12	0,05	0,14	0,11	0,2	0,07	0,17	0,16	0,19	0,17	0,16	0,16
F	0,14	0,1	0,13	0	0,093	0,1	1	0,1	0,06	0,1	-0,1	0	-0	-0	0,05	0,1	0,1	0	0,06	0,02	0,09	0	0,05	0,09	0,1	0,07	0,04	0,02	0,07	0,082	0,1	-0
G	0,09	0,1	0,06	0,1	0,147	0,1	0,08	1	0,03	0	-0	0	0,11	0	0,17	0	0,1	0,1	0,13	0,09	0,05	-0	0,01	0,08	0,1	0,05	0,07	0,16	0,01	0,0999	0,11	0,1
Ğ	0,05	0,1	0,18	0,2	0,187	0,2	0,06	0	1	0,21	0,09	0,1	0,21	0	0,1	0,1	0	0	0,02	0,07	0,08	0,06	0,18	0,12	0,1	0,08	0,14	0,08	0,13	0,0951	0,14	0,09
H	0,1	0,1	0,16	0,1	0,119	0,2	0,1	0	0,21	1	0,02	0	0,14	-0	0,02	0,1	0	-0	-0	0,01	0,07	0,03	0,16	0,09	0,1	0,07	0,03	0,02	0,08	0,0226	0,09	0,03
I	0,03	0,1	0,08	0,1	0,088	0,1	-0,1	-0	0,09	0,02	1	0,2	0,1	0,2	0,07	-0	0	0,1	0,06	0,16	0,03	0,03	0,1	0,04	0,1	0,06	0,12	0,12	0,09	0,0715	0,11	0,18
İ	0,07	0	0,12	0	0,087	0,1	0,01	0	0,13	0,04	0,17	1	0,03	0,2	0,01	0,1	0	0,1	0,07	0,08	0,12	0,1	0,05	0,11	0,1	0,03	0,15	0,06	0,09	0,1395	0,15	0,17
J	-0	0,1	0,09	0,2	0,222	0,2	-0	0,1	0,21	0,14	0,1	0	1	-0	0,22	0	0,1	0,1	0,05	0,07	0	-0	0,14	0,02	0,2	0,14	0,17	0,18	0,07	0,0192	0,17	0,14
K	0,12	0,1	0,08	-0	0,072	0,1	-0	0	0,01	-0	0,17	0,2	-0	1	-0,1	0	0,1	0,1	0,07	0,21	0,14	0,12	-0	0,09	-0	-0,1	0,09	0,04	0,15	0,2289	0,11	0,11
L	0,04	0,1	0,06	0,1	0,212	0,2	0,05	0,2	0,1	0,02	0,07	0	0,22	-0,1	1	0	0,1	0,2	0,09	0,12	0,02	-0	0,07	0,01	0,2	0,19	0,17	0,25	0,03	0,0369	0,14	0,16
M	0,1	0,1	0,1	0	0,032	0,1	0,14	0	0,11	0,12	-0	0,1	0,01	0	0,01	1	0,1	-0	0,05	0,05	0,11	0,05	0,07	0,08	0	0,01	0,07	-0	0,08	0,0994	0,08	0,01
N	0,12	0,1	0,07	-0	0,086	0,1	0,13	0,1	0,02	0,01	0,01	0	0,07	0,1	0,1	0,1	1	0,2	0,11	0,18	0,09	0,05	0,01	0,07	0	0,04	0,11	0,1	0,06	0,1665	0,11	0,09
O	0,07	0,1	0,04	0	0,181	0,2	0,04	0,1	0,04	-0	0,11	0,1	0,12	0,1	0,23	-0	0,2	1	0,07	0,24	0,09	-0	0,04	0,09	0,2	0,1	0,18	0,22	0,07	0,1158	0,06	0,24
Ö	0,11	0	0,08	0	0,117	0,1	0,06	0,1	0,02	-0	0,06	0,1	0,05	0,1	0,09	0	0,1	0,1	1	0,07	0,02	-0	0,02	0,08	0	0,06	0,07	0,09	0,02	0,0957	0,14	0,07
P	0,07	0,2	0,1	0	0,132	0,2	0,02	0,1	0,07	0,01	0,16	0,1	0,07	0,2	0,12	0,1	0,2	0,2	0,07	1	0,11	0,07	0,04	0,11	0,1	0,01	0,13	0,12	0,18	0,241	0,09	0,16
R	0,1	0,1	0,12	0	0,083	0,1	0,09	0,1	0,08	0,07	0,03	0,1	0	0,1	0,02	0,1	0,1	0,1	0,02	0,11	1	0,15	0,01	0,09	0,1	0,02	0,07	0,06	0,11	0,136	0,08	0,08
S	0,06	0,1	0,03	0	-0	0,1	0	-0	0,06	0,03	0,03	0,1	-0	0,1	-0	0	0,1	-0	-0	0,07	0,15	1	0,01	0,03	0	0,02	0,07	0,01	0,1	0,0884	0,07	0,02
Ş	0,06	0,1	0,13	0,1	0,118	0,1	0,05	0	0,18	0,16	0,1	0,1	0,14	-0	0,07	0,1	0	0	0,02	0,04	0,01	0,01	1	0,06	0,1	0,16	0,09	0,09	0,06	-3E-04	0,12	0,12
T	0,11	0	0,09	0,1	0,077	0,1	0,09	0,1	0,12	0,09	0,04	0,1	0,02	0,1	0,01	0,1	0,1	0,1	0,08	0,11	0,09	0,03	0,06	1	0	0,02	0,07	0,04	-0	0,1533	0,1	0,09
U	0	0,2	0,08	0,2	0,17	0,2	0,07	0,1	0,14	0,1	0,12	0,1	0,18	-0	0,19	0	0	0,2	0,03	0,15	0,05	0,02	0,13	0,05	1	0,15	0,09	0,17	0,08	0,0428	0,09	0,11
Ü	0,1	0,1	0,06	0,1	0,088	0,1	0,07	0,1	0,08	0,07	0,06	0	0,14	-0,1	0,19	0	0	0,1	0,06	0,01	0,02	0,02	0,16	0,02	0,1	1	0,06	0,17	-0	-0,061	0,12	0,1
V	0,06	0,1	0,08	0,1	0,145	0,2	0,04	0,1	0,14	0,03	0,12	0,1	0,17	0,1	0,17	0,1	0,1	0,2	0,07	0,13	0,07	0,07	0,09	0,07	0,1	0,06	1	0,13	0,07	0,1099	0,16	0,19
Y	0,07	0,1	0,05	0,1	0,205	0,2	0,02	0,2	0,08	0,02	0,12	0,1	0,18	0	0,25	-0	0,1	0,2	0,09	0,12	0,06	0,01	0,09	0,04	0,2	0,17	0,13	1	0,06	0,0759	0,16	0,19
Z	-0	0,2	0,11	0	0,119	0,2	0,07	0	0,13	0,08	0,09	0,1	0,07	0,1	0,03	0,1	0,1	0,1	0,02	0,18	0,11	0,1	0,06	-0	0,1	-0	0,07	0,06	1	0,1436	0,14	0,06
YOKET	0,1	0,1	0,14	0	0,127	0,2	0,08	0,1	0,1	0,02	0,07	0,1	0,02	0,2	0,04	0,1	0,2	0,1	0,1	0,24	0,14	0,09	-0	0,15	0	-0,1	0,11	0,08	0,14	1	0,14	0,1
YOK	0,07	0,1	0,11	0,1	0,163	0,2	0,1	0,1	0,14	0,09	0,11	0,2	0,17	0,1	0,14	0,1	0,1	0,1	0,14	0,09	0,08	0,07	0,12	0,1	0,1	0,12	0,16	0,16	0,14	0,142	1	0,15
ARA	0,07	0,1	0,07	0,1	0,217	0,2	-0	0,1	0,09	0,03	0,18	0,2	0,14	0,1	0,16	0	0,1	0,2	0,07	0,16	0,08	0,02	0,12	0,09	0,1	0,1	0,19	0,19	0,06	0,098	0,15	1

4.4.2. Veri setinin oluşturulması

Bir veri kümesi için belirtilen formatta veri kaydetmek için, “python create_webcam_datset.py” komutu çalıştırmak veya PyCharm IDE üzerinden create_webcam_datset.py python betiğini koşmak gerekir.

Bununla beraber “Lutfen dosya ismini giriniz :” kısmına oluşturmak istediğimiz veri seti kümesinin ismi girilir. Örneğin “A” harfi için veri seti kaydetmek istiyor isek, bu kısma A yazıp enter tuşuna basmamız gerekir.

Bununla beraber Şekil 4.19’da gösterildiği gibi web kamera ekranı açılır. Kaydetmek istediğimiz işareti mavi ile seçili alanda göstermemiz yeterlidir. Mavi çerçeve dışında kalan alanlar hiçbir şekilde kaydedilmeyecekitr. Kaydetme işlemi sürerken, kaçıncı dosyanın kayıt edildiği bilgisi alt kısımda yazar.



Şekil 4.19 Veri setinin kaydedilmesi

4.5. Önceden Eğitilmiş Modele İnce Ayar Yapmanın Yolları

Transfer öğrenme metodolojileri farklı unsurlara dayanır. Ancak en temel iki tanesi yeni veri setinin büyüklüğü ve orjinal veri setine benzerliğidir. Derin Konvolüsyonel Sinir Ağlarında özelliklerin ilk katmanlarda daha genel iken sonraki katmanlarda veriye özgü olduğu göz önünde bulundurularak 4 farklı senaryo mevcuttur [42,43].

Senaryo 1 - Veri benzerliği çok yüksekken, veri setinin boyutu küçük - Bu durumda, veri benzerliği çok yüksek olduğu için modeli yeniden test etmemize gerek yoktur. Tek yapmamız gereken, çıktı katmanlarını problem tanımımıza göre özelleştirmek ve değiştirmek. Önceden ölçülmüş modeli bir özellik çıkarıcı olarak kullanmaktır.

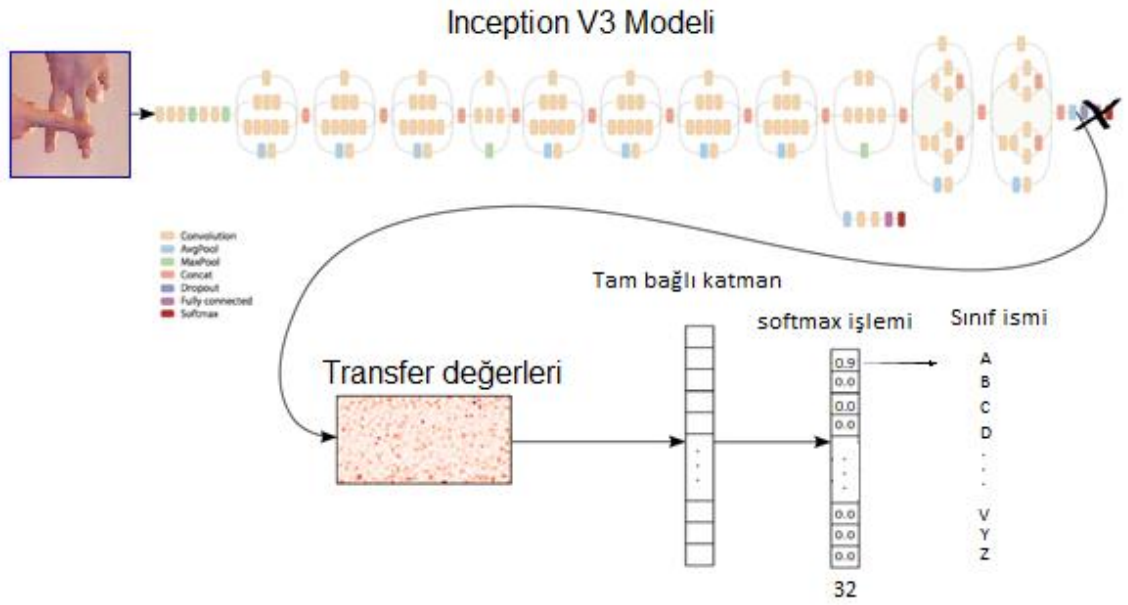
Senaryo 2 - Veri boyutu küçük ve veri benzerliği çok düşük - Bu durumda, önceden eğitilmiş modelin ilk katmanlarını dondurabilir ve geriye kalan katmanlarını yeniden eğitebilir. Üst katmanlar daha sonra yeni veri kümesine göre uyarlanır. Veri setinin küçük boyutu, başlangıç katmanlarının önceden ölçülmüş halde tutulması (daha önce büyük bir veri setinde eğitilmiş) ve bu katmanların ağırlıklarının donması ile telafi edilir.

Senaryo 3 - Veri setinin boyutu büyük ancak veri benzerliği çok düşük - Bu durumda, büyük bir veri setine sahip olduğumuz için sinir ağı eğitimimiz etkili olacaktır. Ancak, sahip olduğumuz veriler önceden belirlenmiş modellerimizin eğitimi için kullanılan verilerle karşılaştırıldığında çok farklı olduğundan, önceden belirlenmiş modeller kullanılarak yapılan tahminlerde etkili olmayacaktır. Bu nedenle, sinir ağını verilerinize göre yeniden eğitmek en iyisidir.

Senaryo 4 - Verilerin büyüklüğü ve veri benzerliği yüksek - Bu ideal durumdur. Bu durumda, önceden belirlenmiş model en etkili olmalıdır. Modeli kullanmanın en iyi yolu, modelin yapısını ve modelin ilk ağırlıklarını korumaktır. Ardından, önceden eğitilmiş modelde ilklendirilen ağırlıkları kullanarak bu modeli yeniden eğitebiliriz.

TİD alfabesi veri seti sınıflarından hiçbiri, tüm ağın eğitildiği orijinal ImageNet sınıflarında değildir. TİD alfabesi veri seti için, veri setinin büyüklüğü ve orjinal veri setine benzerliği göz önünde bulundurulduğunda, inception veri setiyle karşılaştırıldığında, TİD alfabesinin tanımlanması problemi için yapılması gereken

ince ayar yönteminde senaryo-2 ile eşleşmektedir. Senaryo 2 ile inception v3 modelinde yapacağımız düzenleme ve ince ayarlar sonrasında modele kendi veri setimizden bir görüntüyü verdiğimizde geçeceği aşamalar Şekil 4.20 ile verilmiştir.



Şekil 4.20 Inception v3 ile öğrenmeyi transfer ederek sınıflandırma

4.6. Darboğazlar

Inception V3 modeli üst üste istiflenmiş birçok katmandan meydana gelir. Bu katmanlar önceden eğitilmiştir ve çoğu görüntüyü sınıflandırmaya yardımcı olacak bilgileri bulma ve özetleme konusunda çok değerlidir. Önceki katmanlar zaten eğitilmiş durumlarını koruduğundan sadece son katman eğitmek hedeflenmiştir. İlk aşamada "dataset" klasörü altındaki sınıfları çıkartır ve bunların içindeki her bir görüntü için darboğaz değerlerini hesaplar. Sınıflandırmayı gerçekten yapan son çıkış katmanından hemen önceki katman gayri resmi olarak darboğaz olarak adlandırılır. Bu sondan görünüşlü katman, sınıflandırıcının tanımlanması istenen tüm sınıfları ayırt etmek için kullanabileceği kadar iyi bir değer kümesi üretmek için eğitilmiştir. Son katman eğitimimizin yeni sınıflar üzerinde çalışabilmesinin nedeni, ImageNet'teki tüm 1000 sınıfı ayırmak için gerekli bilginin çıkarılması ve bu bilginin ayrıca diğer nesnelerin ayırt edilebilmesinde de oldukça yararlı olmasıdır.

Her görüntünün birden çok kez kullanılması ve her görüntü için darboğaz değerlerinin hesaplanması uzun sürdüğü için, bu darboğaz değerlerini diskte önbelleğe almak faydalı olabilir. Böylece her eğitimde bu değerlerin tekrar tekrar hesaplanması gerekmez.

4.7. Sistemi Eğitirken Eklenen Parametere Opsiyonları

Sistemi eğitiken Inception-v3 dışında bir model kullanılmıyorsa, değiştirilmemesi gereken parametreler Çizelge 4.5 ile verilmiştir. Bu parametrelerin inception v3 için değiştirilmemesinin sebebi değerlerin inception-v3 model mimarisine bağlı parametreler olmasıdır. Eğer farklı bir model kullanılmak isteniyorsa, bu parametrelerin o modelin mimarisine uygun olacak şekilde değiştirilmesi gerekmektedir.

Çizelge 4.5 Inception V3 modeli için girilen değiştirilmeyen parametreler

DATA_URL	'http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz'	Inception v3 modelinin indirileceği lokasyondur.
BOTTLENECK_TENSOR_NAME	'pool_3/_reshape:0'	Inception modelinde bottleneck katmanı bu şekilde adlandırılmıştır.
BOTTLENECK_TENSOR_SIZE	2048	Darboğaz sensöründe kaç giriş olacağını gösterir.
MODEL_INPUT_WIDTH	299	Model giriş yüksekliği
MODEL_INPUT_HEIGHT	299	Model giriş genişliği
MODEL_INPUT_DEPTH	3	Model giriş derinliği
JPEG_DATA_TENSOR_NAME	'DecodeJpeg/contents:0'	Inception modelinde girişlerin tutulduğu yerdir.
MAX_NUM_IMAGES_PER_CLASS	1500	Bir sınıftaki maksimum görüntü sayısı

Bunun yanında bir takım parametreler eğitim yapılırken kullanılan betiğe arguman olarak verilebilir. Bu argumanlar kullanılan verinin yapısı, boyutu ve özelliklerine göre verilebilir. Bu parametreler Çizelge 4.6 ile verilmiştir.

Çizelge 4.6 Sistem eğitilirken değiştirilebilecek parametreler

Arguman İsmi	İşlevi	Varsayılan değeri
--image_dir	Görüntü dosyalarının olduğu dizindir.	‘’
--output_graph	Eğitilmiş grafiğin kaydedileceği dizindir.	'logs/output_graph.pb'
--output_labels	Eğitilmiş graikteki sınıfların kaydedileceği dizindir.	logs/output_labels.txt
--summaries_dir	Tensorboard için özet logların kaydedileceği dizindir.	'logs/retrain_logs'
how_many_training_steps	Bitirmeden önce kaç eğitim adımı olacağıdır.	4000
--learning_rate	Eğitim sırasında ne kadar büyük bir öğrenme oranı kullanıldığıdır.	0.01
--testing_percentage	Test seti olarak kullanılacak görüntülerin yüzdesidir.	10
--validation_percentage	Doğrulama seti olarak kullanılacak görüntülerin yüzdesidir.	10
--eval_step_interval	Eğitim sonuçlarının ne sıklıkla değerlendirileceğidir.	100
--train_batch_size	Bir seferde kaç görüntü üzerinde eğitim yapılacağıdır.	100
--test_batch_size	Test edilecek kaç resim olduğunu belirtir.	-1
--validation_batch_size'	Bir doğrulama setinde kaç resim kullanılacağını belirtir.	100
--print_misclassified_test_images	Yanlış sınıflandırılmış tüm test resimlerinin bir listesini yazdırılma seçeneği	False
--model_dir	Inception v3 modelinin diznidir.	'logs/imagenet'
--bottleneck_dir	Darboğaz katmanı değerlerinin dosya olarak önbelleğe alınacağı dizindir.	/tmp/bottleneck
--final_tensor_name	Yeniden eğitilmiş grafikte çıkış sınıflandırma katmanı	final_result

Çizelge 4.6 devam ediyor.

--flip_left_right	Antrenman görüntülerinin yarısının yatay olarak rasgele çevrilmesi seçeneğidir.	False
--random_crop	Marjın ne kadarının rasgele eğitim görüntülerini keseceğini belirleyen bir yüzdedir.	0
--random_scale	Eğitim görüntülerinin boyutunu rastgele ölçeklendirmenin ne kadar olacağını belirleyen bir yüzdedir.	0
--random_brightness'	Eğitim görüntüsü giriş piksellerini yukarı veya aşağıya rasgele çarpmanın ne kadar olacağını belirleyen bir yüzde	0

Bu parametrelerden değiştirilirken dikkat edilmesi gereken bazı hususlar ilgili başlıklar altında aşağıda verilmiştir.

- Eğitim Adımı

Eğtim adımı “--how_many_training_steps” argümanı ile verilir. Bu varsayılan olarak 4.000’dir, ancak 8.000’e çıkarırsak iki kat daha uzun süre çalışacaktır. Doğruluktaki iyileşme oranı ne kadar uzun süre antrenman yapıyorsak yavaşlar ve bir noktada tamamen durur [67].

- Öğrenme Oranı

Eğitim sırasında son katmandaki güncellemelerin büyüklüğünü kontrol eder. Sezgisel olarak bu küçük ise öğrenme daha uzun sürer ama genel kesinlik için yardımcı olabilir.

- Eğitim Verisini Çarpıtma

Görüntü eğitiminin sonuçlarını iyileştirmenin yaygın bir yolu, eğitim girişlerini rastgele şekillerde deforme etmek, kırpmak veya parlatmaktır. Bu çarpıklıkları senaryomuzda etkinleştirmenin en büyük dezavantajı, darboğaz önbelleğe almanın artık kullanışlı olmamasıdır, çünkü girdi görüntüleri asla tekrar

kullanılmaz. Bu, eğitim sürecinin çok daha uzun sürdüğü (saatlerce sürdüğü) anlamına gelir. Bu çarpıklıklar, “--random_crop”, “--random_scale”, “--random_brightness” komutlarına geçerek etkinleştirilebilir. ”--flip_left_right” uygulamalarınızda gerçekleşmesi muhtemel olduğu sürece, görüntülerin yarısını yatay olarak rasgele yansıtacaktır. Fakat, harfleri tanımaya çalışıyorsak bunu kullanmak pek önerilmez [67].

- Test Setinin Boyutu

Test edilecek kaç resim olduğu “--test_batch_size” argümanıyla belirtilir. Bu test seti, eğitim tamamlandıktan sonra modelin kesin doğruluğunu değerlendirmek için yalnızca bir kez kullanılır. -1 değeri, testin tamamının kullanılmasına neden olur ve bu, çalıştırmalarda daha kararlı sonuçlara yol açar.

- Doğrulama Setinin Boyutu

Bir değerlendirme partisinde kaç tane resim kullanılacağı “--validation batch size“ argümanı ile verilir. Bu doğrulama seti test setinden çok daha sık kullanılır ve modelin eğitim sırasında ne kadar doğru olduğunun erken bir göstergesidir. -1 değeri tüm doğrulama setinin kullanılmasına neden olur ve bu da eğitim yinelemelerinde daha istikrarlı sonuçlar sağlar, ancak büyük eğitim setlerinde eğitimin daha yavaşlamasına sebep olabilir.

- Eğitim ve Test Veri Seti Yüzdesi

Verilerimizin bir kısmını eğitim sürecinden uzak tutuyoruz, böylece model onları ezberleyemiyor. Olağan bölünme, görüntülerin %80'ini ana eğitim setine koymak,% 10'unu eğitim sırasında sık sık doğrulama yapmak için bir kenara ayırmak ve daha sonra da gerçekleri tahmin etmek için bir test seti olarak daha az kullanılan son% 10'una sahip olmaktır. Bu oranlar “--testing_per%” ve “--validation_per%” parametrelerini kullanarak kontrol edilebilir. Genel olarak, bu değerleri varsayılan değerlerinde bırakmak iyidir, çünkü bunları ayarlamanın eğitim için genellikle bir avantajı bulamaz [67].

- Yanlış Sınıflandırılmış Görüntülerin İncelenmesi

Eğitim tamamlandığında, yanlış sınıflandırılmış görüntüleri test setinde incelemeyi için “--print_misclassified_test_images” parametresi “True” olarak eklenmedir. Bu, model için hangi görüntü türlerinin en çok kafa karıştırıcı olduğu ve hangi kategorileri ayırt etmenin en zor olduğu konusunda bir fikir edinmekte yardımcı olur. Çoğu zaman, yanlış sınıflandırılmış görüntüleri incelemek de yanlış etiketlenmiş, düşük kaliteli veya belirsiz görüntüler gibi girdi veri setindeki hataları işaret edebilir [67].

4.8. Eğitim

Önceden eğitilmiş model için ince ayarları yaparak sistemi kendi veri setimizle tekrar eğitmeye hazır hale geliriz. Proje genelinde betiklerde Çizelge 4.7 ile verilen Python kütüphaneleri kullanılmıştır.

Çizelge 4.7 Proje genelinde kullanılan kütüphaneler

Kütüphane	Kullanım Amacı	Versiyon
tensorflow	Derin Öğrenme kütüphanesidir.	1.12.0
matplotlib	Eğitim verilerinin grafikleri çizdirilmek için kullanılmıştır.	3.0.2
numpy	Sayısal işlemlerde kullanılmıştır.	1.15.4
Open-cv-python	Gerçek zamanlı bilgisayarlı görü için kullanılmıştır.	4.0.0.21
Scikit-learn	Makine öğrenmesi veri setinin işlenmesi için kullanılmıştır.	0.20.2

Sistemi eğitmek için “train.py” python betiği aşağıdaki gibi gerekli argümanları eklenerek çalıştırmalıdır.

“python train.py --bottleneck_dir=logs/bottlenecks\
--model_dir=inception\
--summaries_dir=logs/training_summaries/basic\
--output_graph=logs/output_graph.pb\
--output_labels=logs/output_labels.txt\
--image_dir=./dataset
--print_misclassified_test_images=True”

Bunun yerine PyCharm IDE ide “train.py” betiği için konfigürasyonları düzenle menüsünden gerekli parametreler ilgili alanlara eklenip betik çalıştırılabilir.

Darboğazların hesaplanması tamamlandığında, ağın en üst katmanının eğitimi başlar. Aşağıdaki gibi her biri eğitim doğruluğu, çapraz entopi ve doğrulama doğruluğunu gösteren bir dizi adım çıktısı görülür.

“Step: 0, Train accuracy: 19.0000%, Cross entropy: 3.418185, Validation accuracy: 7.0% (N=100)”

Burada eğitim doğruluğu, mevcut eğitim partisinde kullanılan görüntülerin yüzde kaçının doğru sınıfla etiketlendiğini gösterir. Doğrulama doğruluğu ise, farklı bir kümeden rastgele seçilen bir görüntü grubunun kesinliğidir.

Bu ikisi arasındaki temel fark eğitim doğruluğu ağdan öğrenebileceği eğitim için kullanılan görüntülerden hesaplama yapar. Doğrulama doğruluğunda ise amaç ağın performansının gerçek ölçüsü eğitim setinde bulunmayan veri setindeki performansını ölçmektir. Dolayısıyla doğrulama doğruluğu hesaplanırken ağın eğitimde hiç kullanmadığı verilerin kullanılır.

Çapraz entropi değeri ise, öğrenme sürecinin nekadar iyi ilerlediğine dair fikir veren bir kayıp fonksiyonudur. Eğitimin amacı, kaybı olabildiğince küçültmektir, bu nedenle çapraz entopinin aşağı yönlü bir eğilim göstermesi gerekir. Böylece eğitimin işe yarayıp yaramadığına dair bir fikir edinebiliriz. Her adım, eğitim setinden rastgele 100 görüntü seçer, önbellekteki darboğazları bulur ve tahminleri almak için onları son katmana besler. Bu tahminler daha sonra geri yayılma işlemi ile son katmanın ağırlıklarını güncellemek için gerçek etiketlerle karşılaştırılır. Süreç devam ederken rapor edilen doğruluğun arttığı görülmelidir.

Tüm adımlar tamamlandıktan sonra eğitim ve doğrulama görüntülerinden ayrı tutulan bir dizi görüntü üzerinde son bir test doğruluğu değerlendirmesi yapılır.

Bu test değerlendirmesi sonucu adımların tamamlanmasının hemen ardından aşağıdaki gibi yazılır.

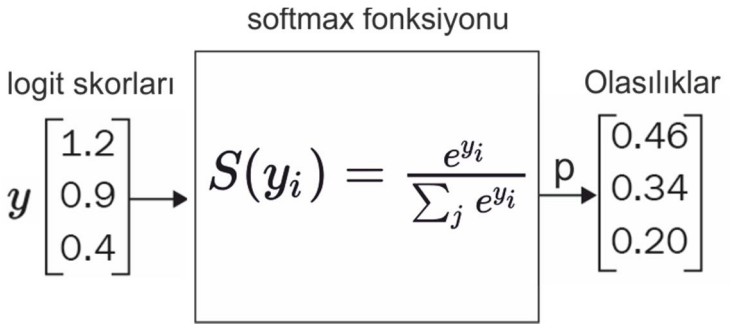
“Final test accuracy = 99.9% (N=4819)”

Bu test değerlendirmesi, eğitimli modelin sınıflandırma görevinde nasıl bir performans göstereceğine dair en iyi tahmindir. Eğitim sürecinde rastlantısallık olduğu için bu değer koşudan koşuya değişecek olsada %90 ile %95 arasında bir doğruluk değeri sınıflandırma performansının beklenilen şekilde yapıldığını gösterir. Bu hesaplama eğitim tamamlandıktan sonra test setinde doğru etiketlenen görüntülerin yüzdesi olarak hesaplanmaktadır.

4.9. Yeniden Eğitim Modelini TİD Alfabesini Sınıflandırmada Kullanma

Eğitim betiği kategorize edilen Inception V3 ağının bir versiyonu olan yeni modeli logs/output_graph.pb dosyasına, eğitilen etiketleri ise logs/output_labels.txt metin dosyasına yazacaktır. Yeni model hem Tensorflow merkez modülünü hem de yeni sınıflandırma katmanını içerir. Her iki dosya da C ++ ve Python görüntü sınıflandırma örneklerinin okuyabileceği biçimdedir, böylece yeni modeli TİD alfabesi veri setini sınıflandırma için kullanabiliriz.

Tensorflow kütüphanesini kullanarak yeniden çizilen grafiği ve metin etiketlerini yüklemek için bir “classify.py” python betiği kullandık. Daha sonra ilk tahminde bulunmak için yeni resmi grafiğe giriş olarak besledik. Son olarak girdi katmanını beklenen çıkış olasılıkları ile eşlemek için son katmandaki softmax işlevini kullandık. Softmax fonksiyonu ve olasılık değerlerinin çıkarılması Şekil 4.21 ile gösterilmiştir.



Şekil 4.21 Softmax fonksiyonu ve kullanım

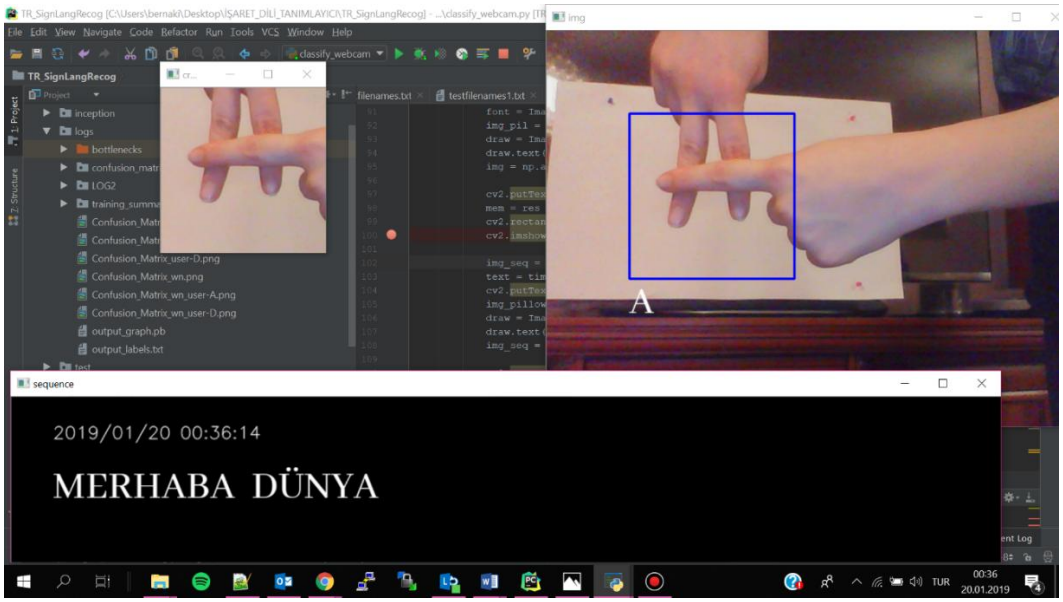
Böylece giriş olarak verilen görüntü için sınıflandırmada en yüksek olasılık değerine sahip olan sınıf en üstte olacak şekilde tüm sınıflar ve olasılık değerleri ekrana yazdırılır.

4.10. Sistemin Gerçek Zamanlı Kullanımı

Sistem Şekil 4.22 ile gösterildiği gibi web kamera aracılığıyla lacivert seçili alana yerleştirilen işaret dili alfabesi karakterini keser, eğitilmiş sisteme verir ve en yüksek tahmin değerine sahip olan karakteri ve skor değerini yazdırır. Bununla beraber tahmin edilen karakteri bir kelime veya cümle oluşturmak için ayırılan alana yerleştirir. Sisteme TİD alfabesi karakterleri gösterildikçe bu alana en yüksek tahmin değerine sahip karakter eklenmeye devam eder. Yazılmak istenen kelime veya cümle içerdiği tim karakterler gösterilip yazdırıldıktan sonra “Esc” tuşuna basılırak işlem sonlandırılır. Bunun sonunda oluşturulan cümle bir metin dosyası olan “cikis.txt” dosyasına yazılır. Herhangi bir TİD alfabe karakteri gösterilmiyorsa ekranda “YOK” yazar ve herhangi bir harf kelime oluşturmak üzere eklenmez.

Örneğin, “MERHABA” kelimesini oluşturmak için ayrı ayrı “M”, “E”, “R”,H”, “A”, “B”, “A” harflerinin gösterilmesi gerekmektedir. Yanlış yazılan harfleri silmek için özel olarak tanımlanmış “YOKET” işareti gösterilirse son yazlılan karakter silinir. Buna ek olarak eğer bir cümle yazmak istiyorsak; iki kelime arasında bırakılacak boşluğu özel olarak tanımlanmış “ARA” karakterini göstererek ekleyebiliriz.

Sistem çalışırken alınan “MERHABA DÜNYA” yazarken alınan ekran görüntüsü Şekil 4.22 ile verilmiştir.



Şekil 4.22 Sistemin ekran görüntüsü

4.11. Sözde Kod (Gerçek Zamanlı Tanıma Sistemi)

1 : i = 0, mem = “”, consecutive=”” sequence=”” başlangıç değerini ver

2 : Web kameradan video yakalama ile tüm görüntüyü al

3 : Lacivert çerçeve ile seçilmiş alandaki görüntüyü kesip al

4 : Esc ye basıldı ise

5: Sequence içinde yazan metni “cikis.txt” dosyasına kaydet

6: Çık

7: i != 4 ise

8: i değerini 1 arttır

9: i değerini sıfırla

10: Softmax fonksiyonunu adım 2 de kesip alınan görüntü ile besle ve tahmin değerlerini al

11: max_score = 0.0 ve human_string =”” başlangıç değerlerini ver

12: Alfabenin ilk harfi için max_score ve human_string değerlerini al

13: Harfin score değeri max_score değerinden büyük ise

14: max_scrore = harfin skor değeri, human_string = harfin değeri

15: Alfabedeki tüm harflere bakılmadı ise

16: alfabenin bir sonraki harfini al ve 13. adıma git

17: En yüksek skorlu human_string ve max_scrore değerini döndür

18: human_string = mem ise

19: consecutive değerini 1 arttır

20: consecutive değerini 0’a eşitle

21: consecutive değeri = 2 ve human_string != “YOK” ise

22: human_string = “YOKET” ise

23: sequence de en son yazılan kararkteri sil

24 consecutive=0 a eşitle, i’yi 1 arttır, adım 2 e git

25: human_string = “ARA” ise

26: sequence içine bir karakter boşluk ekle

27: consecutive=0 a eşitle, i’yi 1 arttır, adım 2 e git

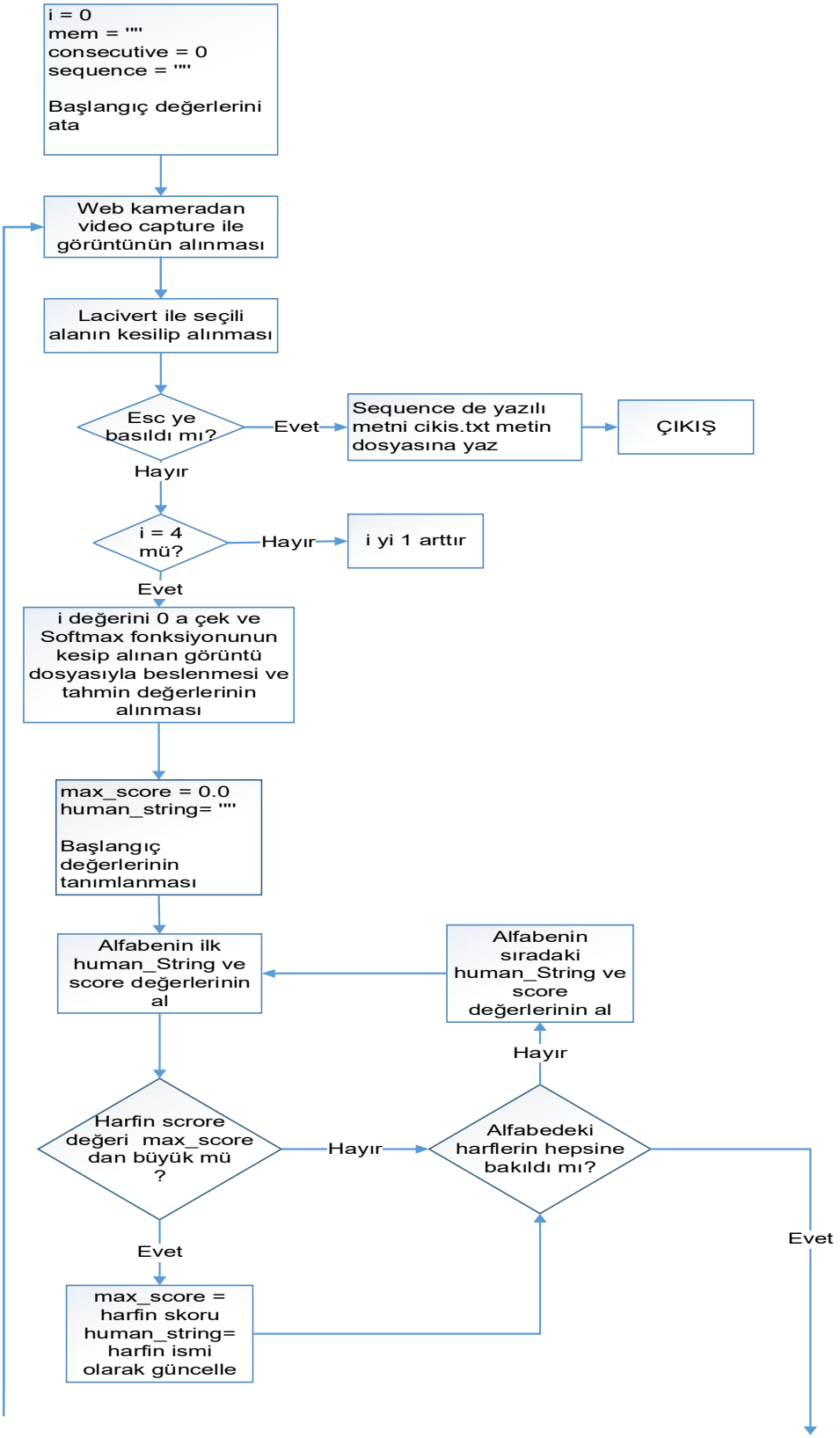
28: human_string != “ARA” ve human_string != “YOKET” ise

29: sequence içine alfabe karakterini ekle

30: consecutive=0 a eşitle, i’yi 1 arttır, adım 2 e git

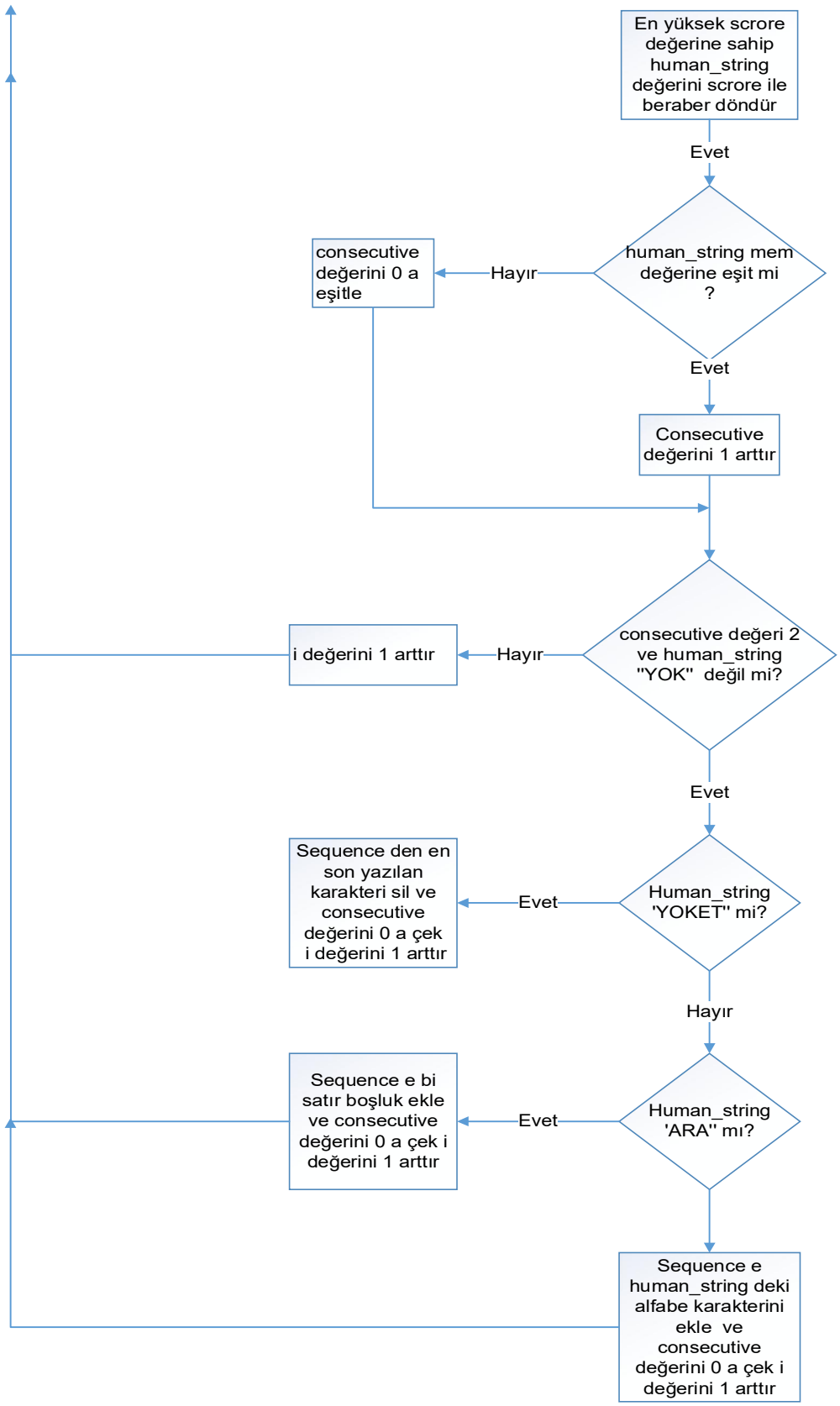
31: i’yi 1 arttır, adım 2 e git

4.12. Akış diyagram(Gerçek Zamanlı Tanıma Sistemimi)



Şekil 4.23 Akış diyagramı (Gerçek Zamanlı Tanıma Sistemi)

Şekil 4.23 devam ediyor.



5. TESTLER VE SONUÇLAR

Sistemin ilk olarak eğitilmesi yaklaşık olarak 4.5 saat sürmüştür. Bu süre içinde inception modeli indirilmiş, her klasör içindeki görüntü dosyaları için darboğazlar hesaplanmış ve sistem varsayılan eğitim adımı olan 4000 adımda son katmanın eğitimi gerçekleşmiştir. Eğitim sırasında her eğitim adımında eğitim doğruluğu, çapraz entropi ve doğrulama doğruluğu hesaplanmıştır. Son olarak eğitim adımları bitince final test doğruluğu hesaplanmış ve sınıflandırılamayan görüntü dosyalarının listesi çıkarılmıştır. Eğitim yapılırken veri setinin %80'i eğitim için, %10'u doğrulama için, %10'luk bir kısmı ise test için kullanılmıştır. Doğrulama ve test için ayırılan veri seti eğitimde kullanılmamaktadır.

Burada eğitim dopruluğu, mevcut eğitim setinde kullanılan görüntülerin yüzde kaçının doğru etiketle sınıflandırıldığını göstermektedir. Doğrulama doğruluğu ise, farklı bir kümeden rastgele seçilen bir görüntü gubunun kesinliğiidir. Son olarak çapraz entropi değeri, öğrenmesürecinin ne kadar iyi ilerlediğine dair fikir veren bir kayıp fonksiyonudur. Eğitimin amacı bu kaybı olabildiğince küçültmek, bu sırada eğitim ve doğrulama doğruluğunuda olabildiğinde yükseltmektir.

Bununla beraber test için kullanılan, eğitimde hiç kullanılmamış görüntü dosyalarının ismi daha sonra yapılan test ile alakalı karışıklık matrisnin çizdirilmesinde kullanılmak üzere "testfilenames.txt" metin dosyasına yazıldıktan sonra eğitim doğruluğu, çapraz entropi ve doğrulama doğruluğu grafikleri çizdirilmiştir.

Tensorflow üzerinden eğitim sonrası alınmış eğitim doğruluğu, çapraz entropi, doğrulama doğruluğu ve final testi doğruluğu sonuçları ilk eğitim için Çizelge 5.1 ile verilmiştir.

Çizelge 5.1 İlk eğitim için alınan sonuçlar

ADIMLAR	EĞİTİM DOĞRULUĞU	ÇAPRAZ ENTROPİ	DOĞRULAMA DOĞRULUĞU
ADIM 0	16.0000%	3.411604	9.0%
ADIM 100	85.0000%	2.310564	78.0%
ADIM 200	97.0000%	1.465969	91.0%
ADIM 300	97.0000%	1.038855	97.0%
ADIM 400	95.0000%	0.853455	97.0%
ADIM 500	98.0000%	0.700240	95.0%
ADIM 600	99.0000%	0.537264	100.0%
ADIM 700	98.0000%	0.524185	98.0%
ADIM 800	99.0000%	0.438457	100.0%
ADIM 900	100.0000%	0.352995	100.0%
ADIM 1000	100.0000%	0.317759	100.0%
ADIM 1100	100.0000%	0.310324	100.0%
ADIM 1200	100.0000%	0.255989	99.0%
ADIM 1300	100.0000%	0.227228	99.0%
ADIM 1400	99.0000%	0.238392	100.0%
ADIM 1500	99.0000%	0.202928	99.0%
ADIM 1600	100.0000%	0.179013	100.0%
ADIM 1700	98.0000%	0.238220	100.0%
ADIM 1800	100.0000%	0.173303	100.0%
ADIM 1900	100.0000%	0.209276	100.0%
ADIM 2000	100.0000%	0.148093	100.0%
ADIM 2100	100.0000%	0.189061	100.0%
ADIM 2200	100.0000%	0.156464	100.0%
ADIM 2300	100.0000%	0.148904	100.0%
ADIM 2400	100.0000%	0.137956	99.0%
ADIM 2500	100.0000%	0.146562	100.0%
ADIM 2600	100.0000%	0.118287	100.0%
ADIM 2700	100.0000%	0.117218	100.0%
ADIM 2800	100.0000%	0.121428	100.0%
ADIM 2900	100.0000%	0.104163	100.0%
ADIM 3000	100.0000%	0.110897	100.0%
ADIM 3100	100.0000%	0.108938	100.0%
ADIM 3200	100.0000%	0.101818	100.0%
ADIM 3300	100.0000%	0.096521	100.0%
ADIM 3400	100.0000%	0.089899	99.0%
ADIM 3500	100.0000%	0.088558	100.0%
ADIM 3600	100.0000%	0.079973	99.0%
ADIM 3700	100.0000%	0.086131	100.0%
ADIM 3800	100.0000%	0.097369	100.0%
ADIM 3900	100.0000%	0.080011	100.0%
ADIM 4000	100.0000%	0.081351	100.0%

Bunun sonucunda eğitim adımları arttıkça başlangıçta 16.0000% olan eğitim doğruluğu yüzdesinin beklendiği gibi adımlar arttıkça yükseldiği ve test sonunda 100.0000% olduğu görülmüştür. Çapraz entropi yani kayıbın başlangıçta 3.411604 değerindeyken beklendiği gibi adımlar atıldıkça kaybın azaldığı görülmüş ve test sonunda bu kaybın 0.081351 değerinde olduğu görülmüştür. Doğrulama yüzdesinin ise, başlangıçta 9.0% değerlerindeyken adımlar atıldıkça beklendiği gibi artış gösterdiği ve test sonucunda 100.0000% değerine geldiği görülmüştür.

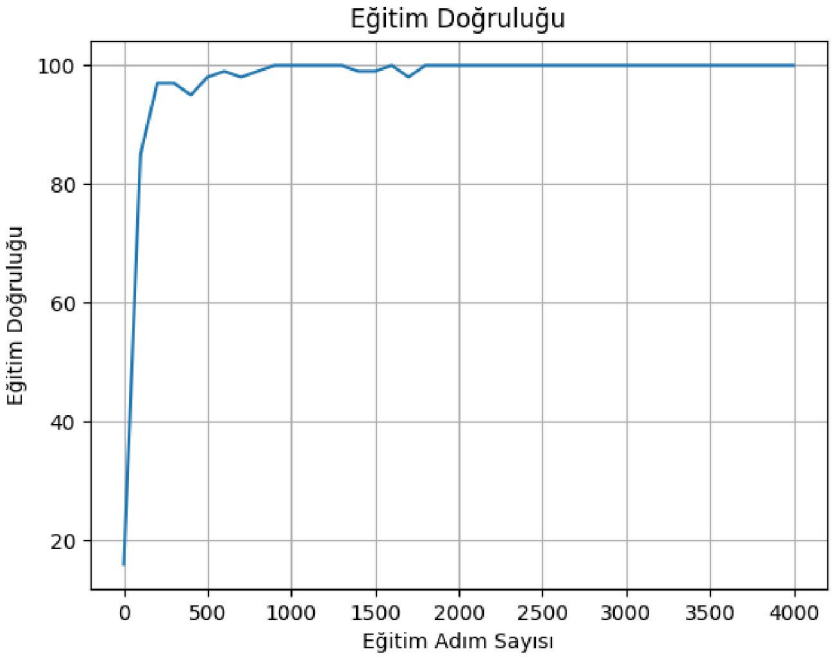
Eğitimde hiç kullanılmamış veri setinin %10 luk kısmıyla yapılmış final test doğruluğu 99.9% olarak ölçülmüştür.

Yanlış sınıflandırılan görüntülerin listesi Çizelge 5.2 ‘de verilmiştir.

Çizelge 5.2 Yanlış sınıflandırılan görüntüler

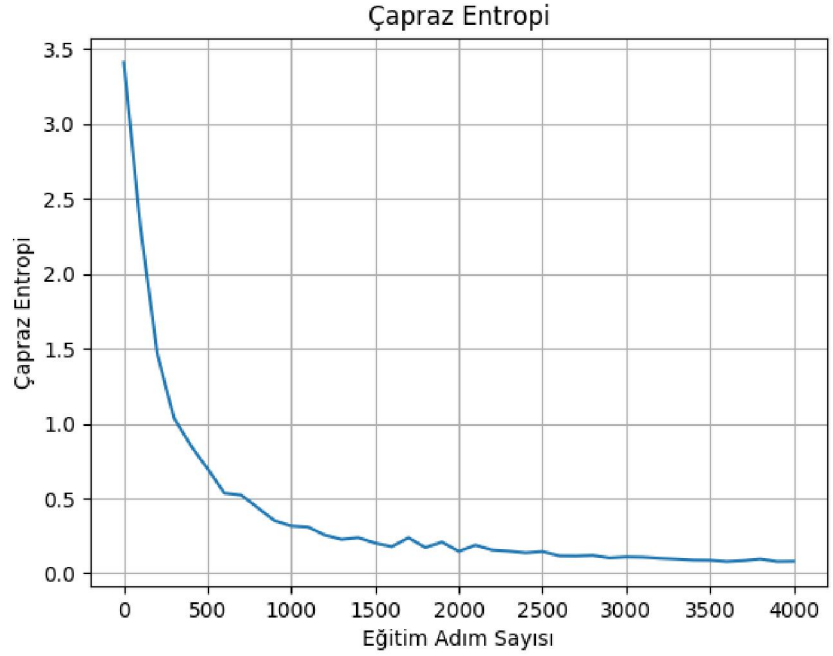
Dosya lokasyonu	Gerçek Değer	Tahmin Edilen Değer
./dataset\M\M695.jpg	M	A
./dataset\N\N563.jpg	N	C
./dataset\O\O1101.jpg	O	ARA
./dataset\P\P508.jpg	P	R
./dataset\P\P509.jpg	P	R
./dataset\P\P570.jpg	P	R
./dataset\P\P904.jpg	P	R
./dataset\R\R80.jpg	R	P
./dataset\T\T898.jpg	T	L
./dataset\Ö\Ö33.jpg	Ö	Ş

Şekil 5.1 ile eğitim doğruluğu, Şekil 5.2 ile çapraz entropi ve Şekil 5.3 ile doğrulama doğruluğu grafikleri çizdirilmiştir.



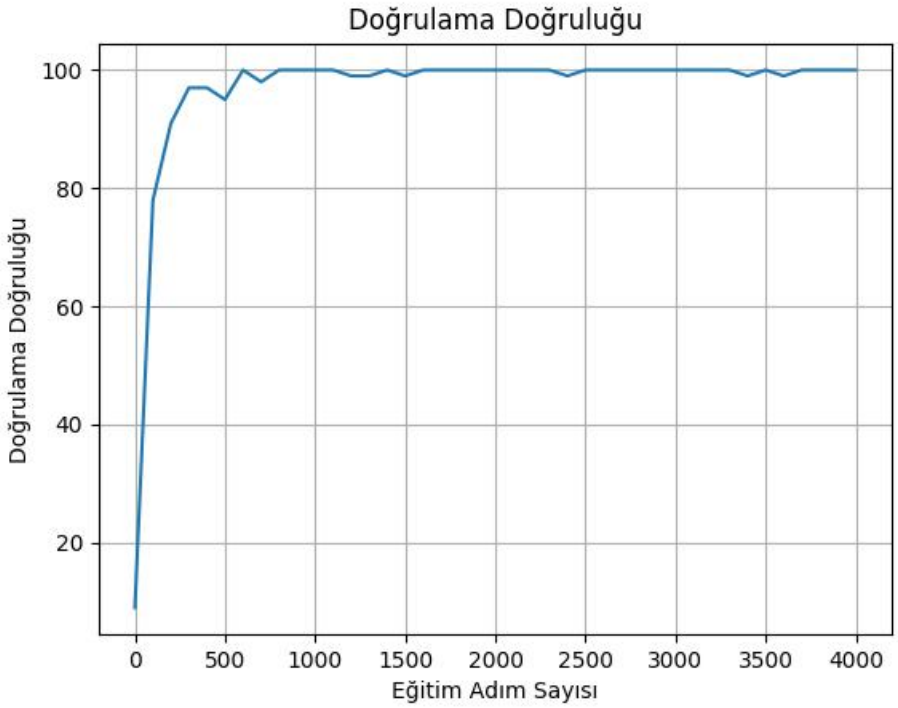
Şekil 5.1 İlk eğitimde eğitim doğruluğu grafiği (Veri setinin %80'lik kısmı)

Grafiktende görülebileceği gibi, eğitim doğruluğunda beklenen artış gözlemlenmiştir.



Şekil 5.2 İlk eğitimde çapraz entropi grafiği

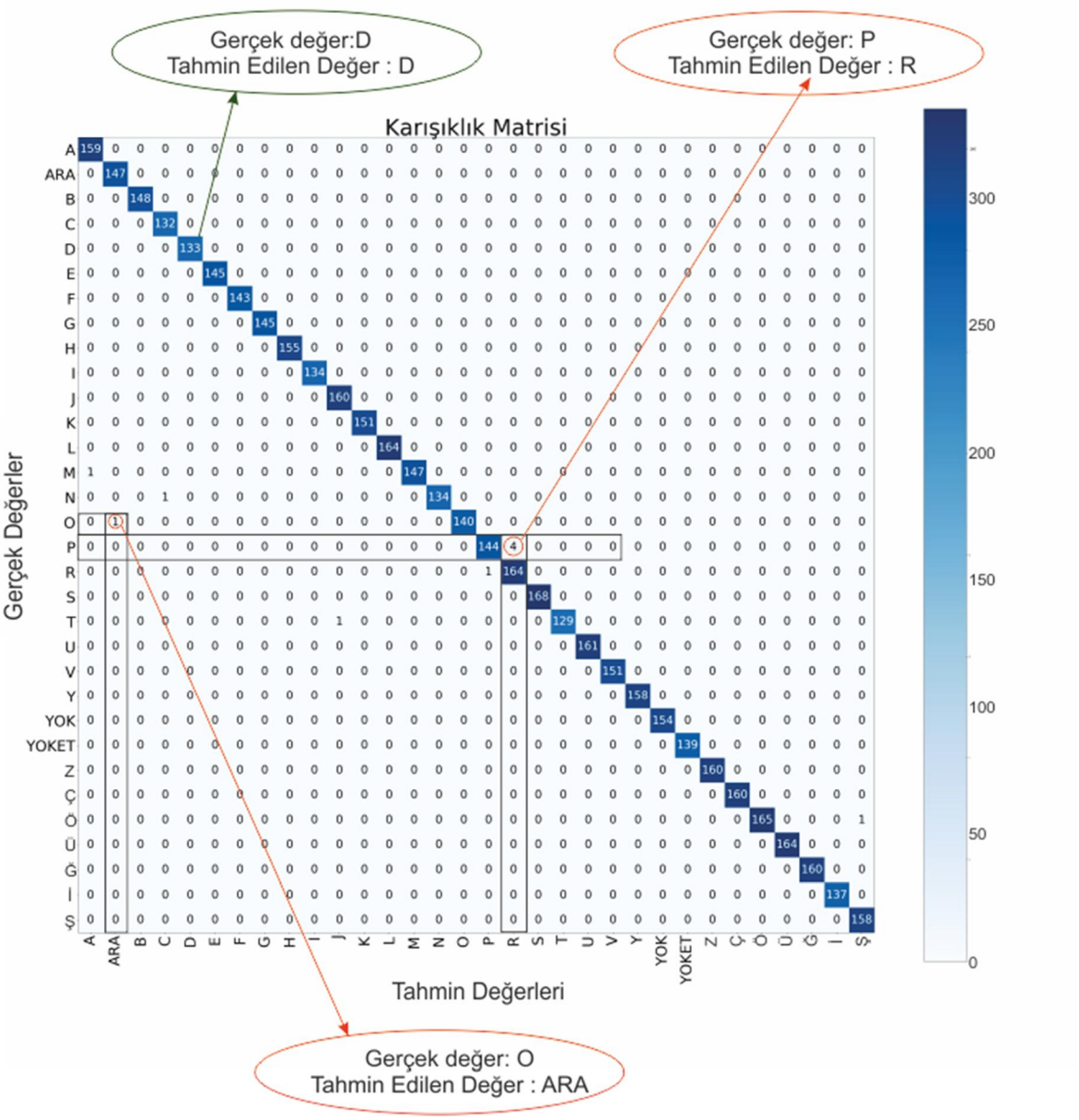
Çapraz entropi değerinin eğitim adımları atıldıkça beklendiği gibi düştüğü grafikte gözlemlenmiştir.



Şekil 5.3 İlk Eğitim için doğrulma doğruluğu grafiği (Veri setinin %10'lik kısmı)

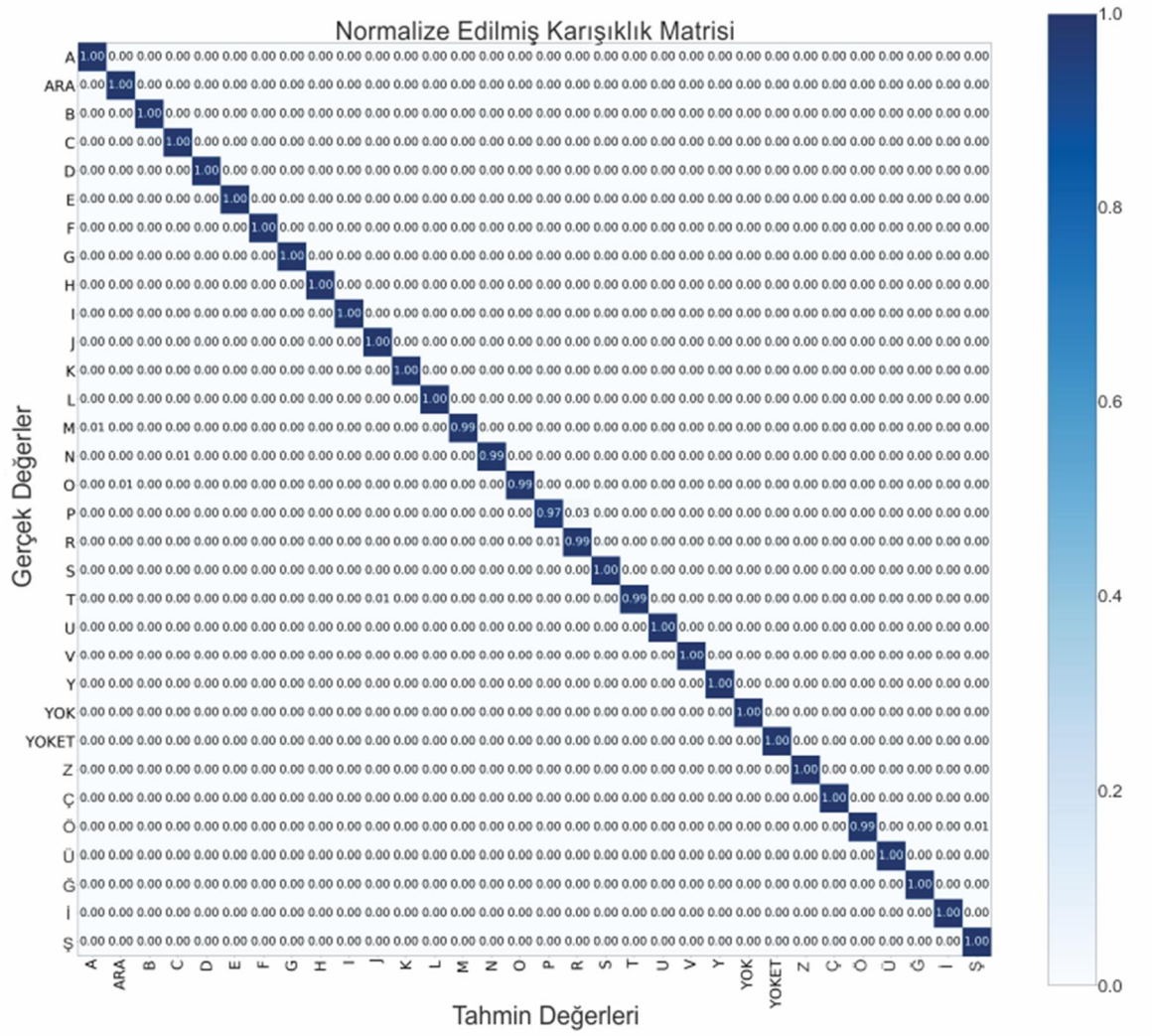
Doğrulama doğruluğu değerlerinin beklendiği gibi eğitim adımları atıldıkça arttığı grafikten gözlemlenmiştir.

Test için ayırılmış, eğitimde hiç kullanılmamış ve rastgele seçilmiş test dosyaları testfilenames.txt dosyasına kaydedilmiştir. Bu dosyalar kullanılarak oluşturulmuş karışıklık(hata) matrisi Şekil 5.4 ile verilmiştir. Karışıklık matrisi, gerçek değerlerin bilinen bir dizi test verisi üzerinde, sınıflandırma modelinin performansını tanımlamak için sıklıkla kullanılan bir tablodur. karışıklık matrisİ ve normalize edilmiş karışıklık matrisİ Şekil 5.4 ve Şekil 5.5 ile verilmiştir.



Şekil 5.4 Karışıklık matrisi

Karışıklık matrisinde harflerin kendileriyle kesiştikleri köşegenlerdeki değerler doğru tahminleri dışında kalan alanlar ise yanlış tahminleri gösterir. Bu alanların %100 başarılı bir sınıflandırma için 0 değerinde olması beklenir. Bu değerler 0 değil ise sınıflandırmada bir karışıklık olduğunu ifade eder. Karışıklığın olduğu değerin satır sütun kesişimlerine bakarak gerçek değerin ve yanlış tahmin edilen değerin ne olduğunu görebiliriz.



Şekil 5.5 Normalize edilmiş karışıklık matrisi

Bu tabloda ise Şekil 5.4 ile verilen karışıklık matrisinin normalize edilmiş halini görebiliriz. Köşegen değerleri 1 olacak şekilde matris normalize edilmiştir.

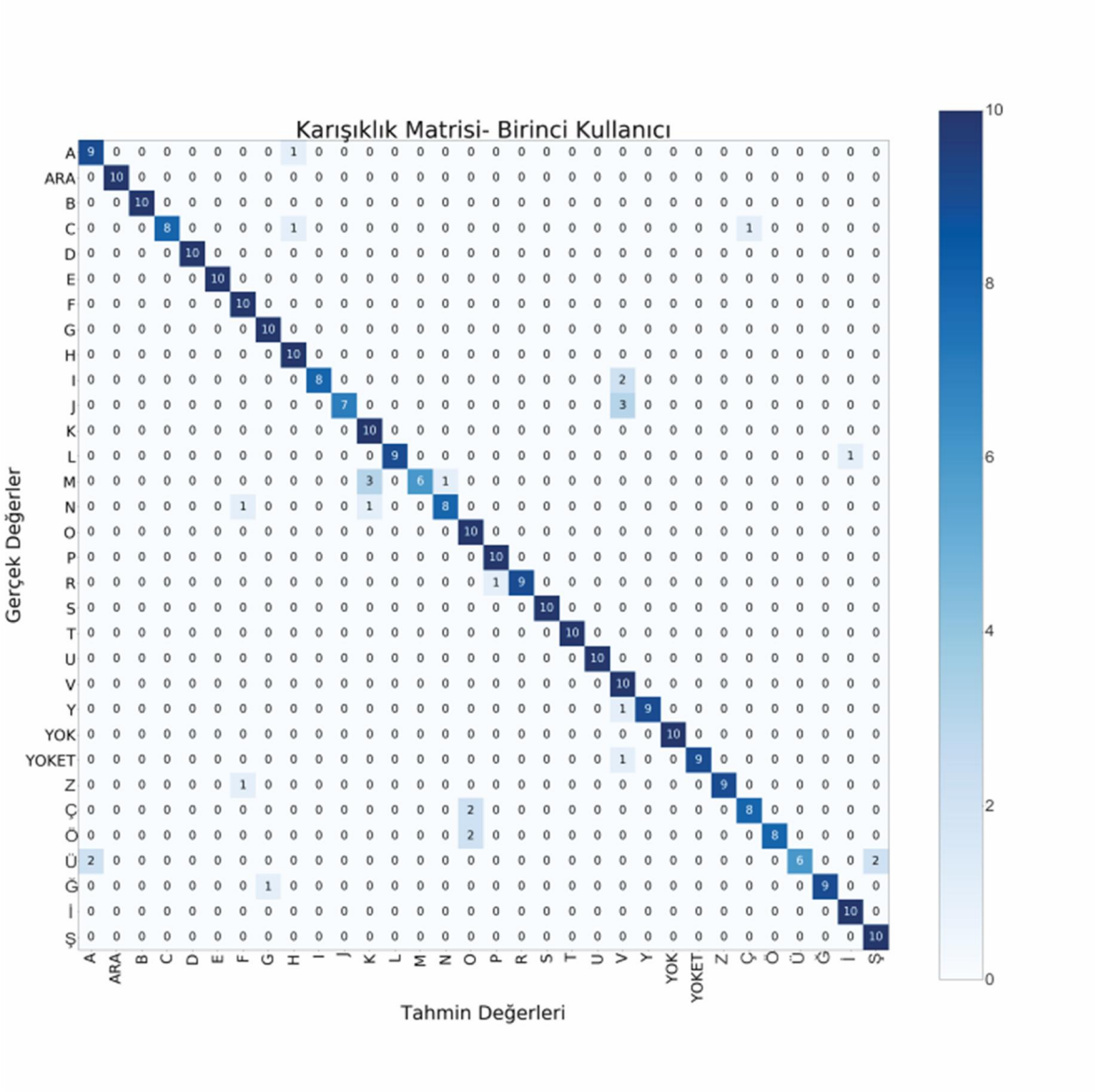
5.1. Farklı Kullanıcılarla Sistemin Test Edilmesi

Eğitilmiş sistem eğitimde ikisi kadın, ikisi erkek olmak üzere 4 farklı kullanıcıdan alınan test veri setiyle sınıflandırılmış ve karışıklık matrisleri çizdirilmiştir. Her kullanıcıdan her harf için 10 tane örnek alınmıştır. Her kullanıcı için 320 adet test verisi üzerinde eğiilmiş sistem başarısı test edilmiştir. Bu kulanıcılar daha önce TİD alfabesini bilmeyen ve kullanmayan kişilerdir. Kullanıcılardan alınarak kaydedilen bu veri setinin özellikleri Çizelge 5.3 ile verilmiştir.

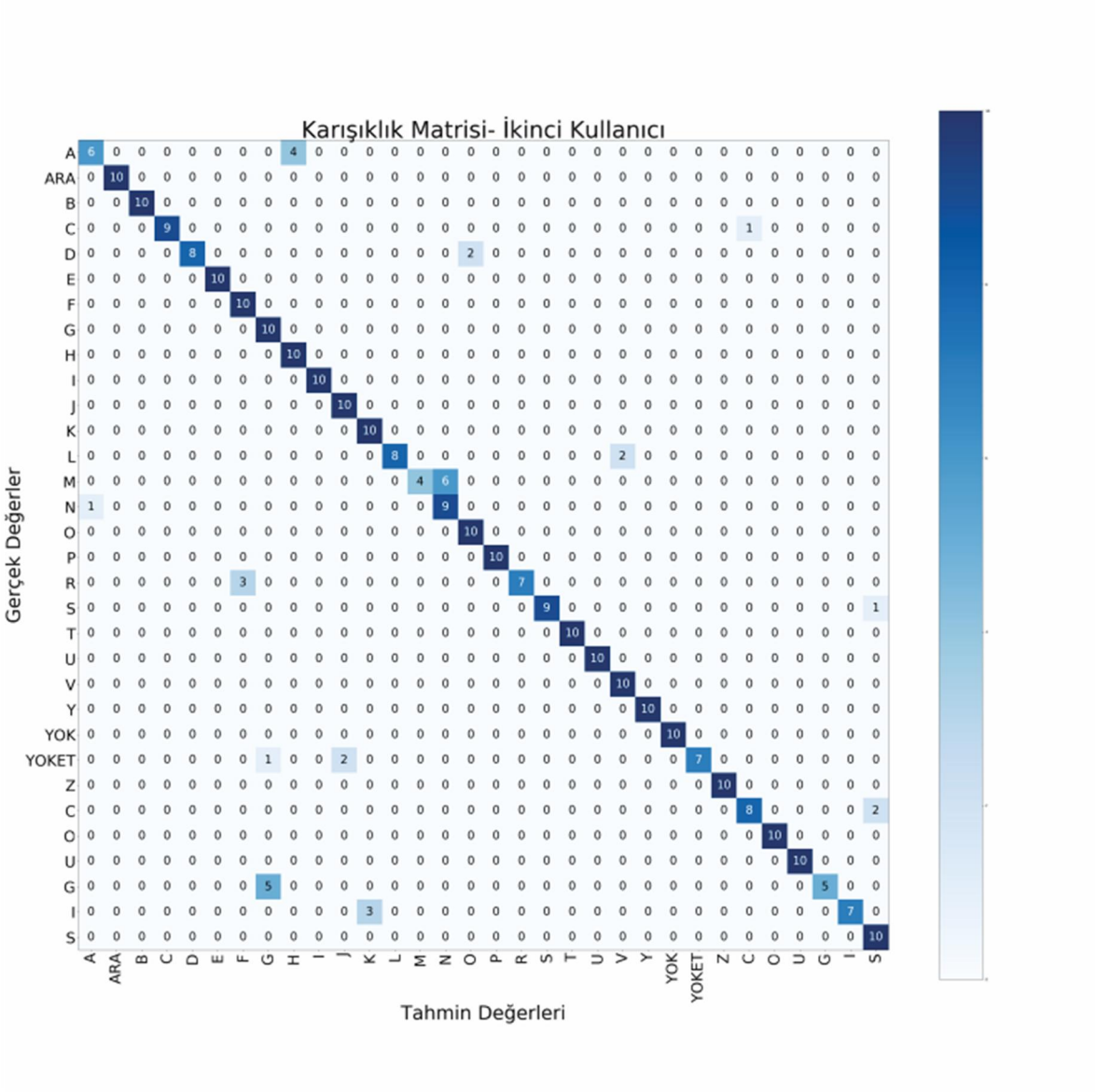
Çizelge 5.3 Kullanıcı test verisi özellikleri

Kullanıcı sayısı	4
Kullanıcı cinsiyet dağılımı	2 Kadın, 2 Erkek kullanıcı
Her sınıf için alınan görüntü sayısı	10
Bir kullanıcı için alınan toplam görüntü sayısı	320
Her işaret için tüm kullanıcılardan alınmış test verisi sayısı	40
Toplam test verisi sayısı	1280
Her kullanıcı için dosya büyüklüğü	3.55 - 3.78 MB
Toplam dosya büyüklüğü	14.6 MB

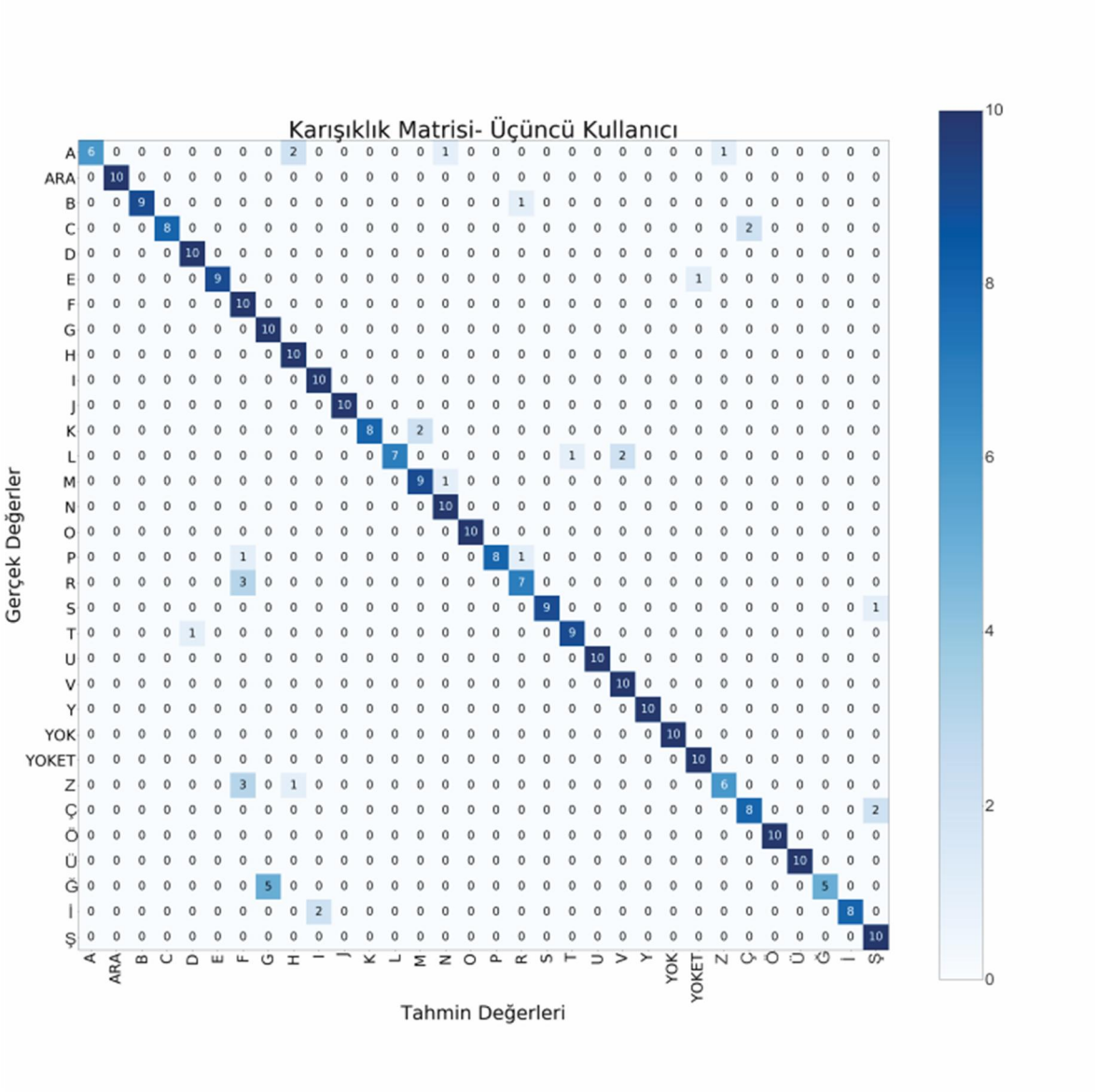
Birinci kullanıcı (Kadın), ikinci kullanıcı (Erkek), üçüncü kullanıcı (Erkek) ve dördüncü kullanıcı (Kadın) için test sonucunda çıkan karışıklık matrisleri Şekil 5.6, Şekil 5.7, Şekil 5.8 ve Şekil 5.9 ile verilmiştir.



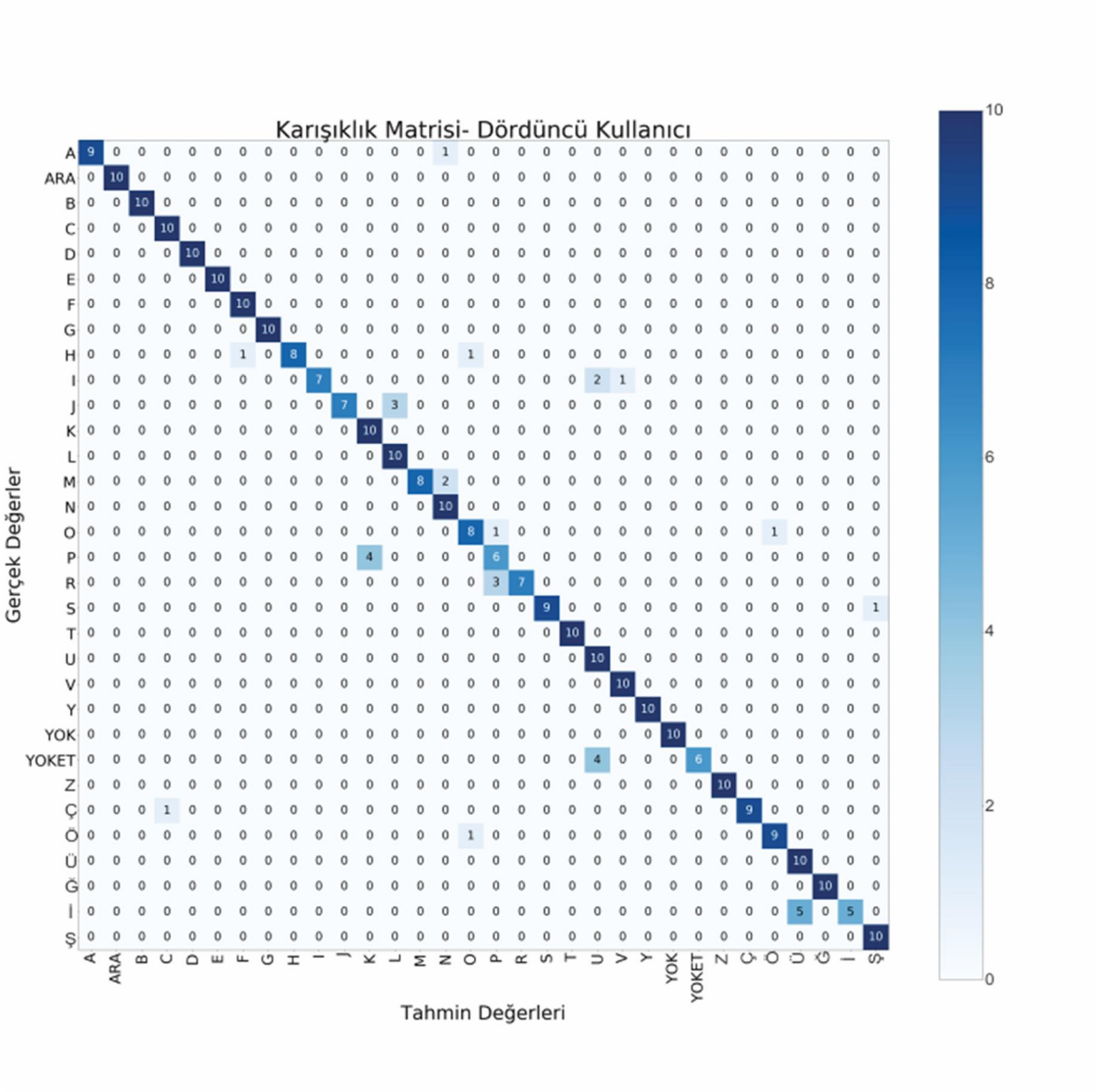
Şekil 5.6 Birinci kullanıcı karışıklık matrisi (%91 başarı ile tanıma)



Şekil 5.7 İkinci kullanıcı karışıklık matrisi (%89,6875 başarı ile tanıma)



Şekil 5.8 Üçüncü kullanıcı karışıklık matrisi (%86.375 başarı ile tanıma)



Şekil 5.9 Dördüncü kullanıcı karışıklık matrisi (%90 başarı ile tanıma)

Her kullanıcı için bu karışıklık matrisleri üzerinden harf bazında ve alfabe bazında doğruluk değerleri hesaplanmıştır. Çizelge 5.4 ile kullanıcı ve genel bazlı doğruluk tablosu verilmiştir.

Çizelge 5.4 Kullanıcılar için harf ve alfabe bazlı doğruluk tablosu

	1.Kullanıcı	2.Kullanıcı	3.Kullanıcı	4.Kullanıcı	Toplam
A	%90	%60	%60	%90	%75
B	%100	%100	%90	%100	%97,5
C	%80	%90	%80	%100	%87,5
Ç	%80	%80	%80	%90	%82,5
D	%100	%80	%100	%100	%95
E	%100	%100	%90	%100	%97,5
F	%100	%100	%100	%100	%100
G	%100	%100	%100	%100	%100
Ğ	%90	%50	%50	%100	%72,5
H	%100	%100	%100	%80	%95
I	%80	%100	%100	%70	%87,5
İ	%100	%70	%80	%50	%75
J	%70	%100	%100	%70	%85
K	%100	%100	%80	%100	%95
L	%90	%80	%70	%100	%85
M	%60	%40	%90	%80	%67,5
N	%80	%90	%100	%100	%92,5
O	%100	%100	%100	%80	%95
Ö	%80	%100	%100	%90	%92,5
P	%100	%100	%80	%60	%85
R	%90	%70	%70	%70	%75
S	%100	%90	%90	%90	%92,5
Ş	%100	%100	%100	%100	%100
T	%100	%100	%90	%100	%97,5
U	%100	%100	%100	%100	%100
Ü	%60	%100	%100	%100	%90
V	%100	%100	%100	%100	%100
Y	%90	%100	%100	%100	%97,5
Z	%90	%100	%60	%100	%87,5
ARA	%100	%100	%100	%100	%100
YOKET	%90	%70	%100	%60	%80
YOK	%100	%100	%100	%100	%100
TOPLAM	%91	%89,6875	%89,375	%90	%90

Tablodan da görülebiceği gibi kullanıcı genelinde en çok karışan harfler A-H, M-N, V-Y, P-R, G-Ğ, L-V harfleri olmuştur. Birinci kullanıcı tarafından gösterilen işaretler alfabe genelinde %91 doğrulukla, ikinci kullanıcı tarafından gösterilen işaretler alfabe genelinde %89,6875 doğrulukla, üçüncü kullanıcı tarafından gösterilen

işaretler alfabe genelinde %89,375 doğrulukla ve son olarak dördüncü kullanıcı tarafından gösterilen işaretler alfabe genelinde %90 doğrulukla sınıflandırılabilmiştir. Tüm kullanıcılar ve tüm alfabe için %90 doğrulukla sınıflandırma yapılabilmiştir.

En düşük tanıma değerlerinin sırasıyla, Ğ, A, M, İ ve R harflerinde olduğu görülmüştür. A harfi simetrik olarak H harfi ile, M harfi ise simetrik olarak N harfi ile karışmaktadır. Bu karışıklığın azaltılması için veri seti ve çeşitliliği arttırılabilir. Asimetrik olmasına karşın karışıklık yaşanıp en düşük tanıma değerlerine sahip olan Ğ, R ve İ işaretleri için ise yine veri seti arttırılabilir fakat buna ek olarak bu hareketler yapılırken parmaklar hareket içerdiği için bu haraketlerde video çekilip, kareler olarak ayırlıp tanıma yapılırsa daha iyi sonuçlar alınabilir.

6. SONUÇ ve ÖNERİLER

TİD Alfabe karakterleri ve özel tanımlanmış karakterler kullanılarak web kamera kullanılarak gerçek zamanlı TİD alfabesinin tanınması ve karakterlerle bir kelime ya da bir cümle oluşturmaya yönelik yazılım tabanlı çalışma gerçekleştirildi.

Çalışmanın ilk aşamasında PyCharm ortamında Python dili kullanılarak TİD alfabe veri setinin kaydedilmesi için yazılım geliştirildi. Her sınıf için 1500 adet görüntü kaydedildi. Metin oluşturabilmek için özel karakterle beraber toplamda 32 sınıftan ve 48000 adet görüntüden oluşan veri seti oluşturuldu. İkinci aşamada sistemin eğitimi için kullanılacak önceden eğitilmiş sistemin ince ayarları yapıldı. Üçüncü aşamada oluşturulan veri set ile PyCharm ortamında Python dili kullanılarak ve TensorFlow derin öğrenme kütüphanesi, eğitim sonucunda çıkan grafikleri çizdirmek için matplotlib kütüphanesi kullanılarak önceden eğitilmiş sistem 4000 adımda tekrar eğitildi eğitim sonuçları grafiksel olarak çıkarıldı. Son olarak dördüncü aşamada PyCharm ortamında Python dili kullanılarak gerçek zamanlı bilgisyarlı görüyü sağlamak için open-cv Python kütüphanesi ve tensorFlow derin öğrenme kütüphanesi kullanılarak eğitilmiş sistemin web kameradan alınan görüntülerle gerçek zamanlı TİD alfabesi ve özel tanımlı karakterleri tanıması ve tanımlanan karakterler kullanılarak bir kelime ya da bir cümle yazılması sağlandı.

Kaydedilen veri setinin %80'lik kısmı eğitim için kullanılırken, eğitim için hiç kullanılmayan %10'luk kısmıyla doğrulama kalan %10'luk kısmı ile test yapıldı. Yapılan çalışma performans kriter grafikleri ve karışıklık matrisleri üzerinden değerlendirildi. Sistemin veriseti üzerinde %100 doğrulama, %99.9 test başarısı elde ettiği görüldü.

Eğitilen sistem dört farklı kullanıcı üzerinde test edildi. Bu kullanıcılar daha önce TİD alfabesini bilmeyen ve kullanmayan kişilerdir. Bu testler için her kullanıcıdan her sınıf için 10'ar görüntüden oluşan toplamda 32 sınıf için 1280 adet görüntüden oluşan test veri seti kaydedildi. Test sonuçlarını değerlendirmek için karışıklık matrisleri çıkarıldı. Hangi harflerin birbiriyle karışıyor olduğu karışıklık matrisleri üzerinden tespit edildi. Alınan sonuçlara göre %100 başarı elde edilen sınıfların F, G, Ş, U, ARA ve YOK sınıfları olduğu görüldü. Bunun yanında en düşük başarının sırasıyla, Ğ, A, M, R ve İ sınıflarında alındığı görüldü.

Kullanıcılardan alınan test verisi üzerinde eğitilmiş sistemin tüm alfabede kullanıcılar için ayrı ayrı %91, %89,6875, %89.375 ve %90 başarı elde ettiği görüldü.

Test sonuçlarına sistem veri seti üzerinde %99.9, farklı kullanıcılarla yapılan testlerde %90 başarı elde ettiği görüldü. Bu değerlere bakılarak tasarlanan sistemin tatmin edici bir başarıya sahip olduğu gözlemlendi.

Testlerde birbiriyle karışan harfler daha çok veri seti çekilerek sistemin işaretler arasındaki ayrımı yapabilmesi sağlanabilir. Buna ek olarak asimetrik karışıklık yaşanan harflerin yapılırken hareket içeren harfler olduğu gözlemlendi. Bunun için video kaydedilip kareler ayırılarak yapılacak bir tanımanın daha başarılı olabileceği düşünülmektedir.

Mevcut sistemde veri seti, metin yazarken hiç işaret gösterilmediği durumun ayırt edilebilmesi için beyaz arka planda çekilmiştir. Farklı kullanıcılardan alınan test verilerinde de bu arka plan kullanılmıştır. Farklı durumlarında bu sınıf içine eklenmesi ve veri setinin bu arka planlara göre çeşitlendirilmesiyle arka plan bağımlılığı ortadan kaldırılabilir.

Buna ek olarak metin yazabilmeye olanak sağlayan karakterlere noktalama işaretleri ve sayılar eklenerek veri seti zenginleştirilebilir.

Çalışma mevcut haliyle TİD alfabesini metine dönüştürecek şekilde tasarlanmıştır. İşitme engellilerin TİD alfabesi dışında anlaşmak için kullandıkları kelime bazlı sözlük için video bazlı bir tanıma sistemi yapılabilir. Ek bir çalışma ile ses tanıma ve sesten işaret diline çevirim yapma özelliği eklenerek, sistemin duyabilen ve duyamayan insanlar arasında tam bir iletişim aracı olması sağlanabilir.

Bu çalışmada sisteme derin öğrenme prensibine uygun olacak şekilde yeni bir veri seti eklemek oldukça kolaydır. Kullanıcılar veri seti kaydetmek için yazılım Python betiğini kullanarak sisteme yeni sınıflar ekleyebilir, sistemi bu veri seti ile tekrar eğiterek bu sınıfların da tanınmasını sağlayabilir.

Mevcut çalışmada Inception-V3 modeli kullanılmıştır. Ek bir çalışma ile farklı önceden eğitilmiş modellere ince ayarlar yapılarak, sitemin başarısı karşılaştırılabilir.

Bu tez çalışmasında tasarlanan sistem geliştirilerek hastane, havaalanı, otobüs ve otobüs durakları gibi kamuya açık yerlerde işitme engellilerle iletişim kurmak amacıyla, okullarda ve eğitim kuruluşlarında TİD eğitiminde kullanılabilir.

Sonuç olarak, bu tez çalışmasında tasarlanan sistem ve gerçek zamanlı tanıma algoritması yazılım ortamına aktarılmış, TİD alfabesini gerçek zamanlı tanıma ve yazılı metine dönüştürmede tatmin edici bir başarı elde edildiği görülmüştür.

KAYNAKLAR LİSTESİ

[1] OZ C., LEU M.C., American Sign Language Word Recognition With A Sensory Glove Using Artificial Neural Networks, Engineering Applications of Artificial Intelligence, vol.24, s.1204–1213, 2011.

[2] BOWDENA R. and SARHADIB M., A Non-Linear Model Of Shape And Motion For Tracking Finger Spelt American Sign Language, Image and Vision Computing, vol.20, no.9-10, s.597-607, 2002.

[3] HU Y., Finger Spelling Recognition Using Depth İnformation And Support Vector Machine, Multimedia Tools and Applications, vol.77, no.21, s.29043–29057, 2018.

[4] QUESADA L., LOPEZ G., GUERRERO L., Automatic Recognition Of The American Sign Language Fingerspelling Alphabet To Assist People Living With Speech Or Hearing Impairments, Journal of Ambient Intelligence and Humanized Computing, vol.8, no.4, s.625-635, 2017.

[5] VIET V.H., SON T.T., NGOC L.Q., Hybrid Deep Network and Polar Transformation Features for Static Hand Gesture Recognition in Depth Data, International Journal of Advanced Computer Science And Applications, vol.7, no.5, s.255-263, 2016.

[6] SALEM, A. and SUNIL V., A Convolutional Neural Network To Classify American Sign Language Fingerspelling From Depth and Colour Images, Expert Systems, vol.34, no.3, 2017.

[7] IBRAHIM N.B., SELIM M.M., ZAYED H.H., An Automatic Arabic Sign Language Recognition System (ArSLRS), Journal of King Saud University - Computer and Information Sciences, vol.30, no.4, s.470-477, 2018.

[8] ELPELTAGY M., ABDELWAHAB M., HUSSEIN M.E., SHOUKRY A., SHOALA A., GALAL M., Multi-Modality-Based Arabic Sign Language Recognition, IET Computer Vision, vol.12, no.7, s.1031-1039, 2018.

[9] SIDIG A.A.I. and MAHMOUD S.A., Trajectory based Arabic Sign Language Recognition, International Journal of Advanced Computer Science and Applications, vol.9, no.4, s.283-291, 2018.

[10] AL-JARRAH O., HALAWANI A., Recognition of gestures in Arabic sign language using neuro-fuzzy systems, Artificial Intelligence, vol.133, no.1-2, s.117-138, 2001.

[11] LALİT K. and PRİTEE K., A Framework For Live And Cross Platform Fingerspelling Recognition Using Modified Shape Matrix Variants On Depth Silhouettes, Computer Vision and Image Understanding, vol.141, s.138-151, 2015.

[12] KUMAR E.K., KISHORE P.V.V., SASTRY A.S.C.S., KUMAR T.K., KUMAR D.A, Training CNNs for 3-D Sign Language Recognition With Color Texture Coded Joint Angular Displacement Maps, IEEE Sıgnal Processıng Letters, vol.25, no.5, 2018.

[13] YANGHO J., SUNMOK K., YOUNG J., KI-BAEK L., Human-like Sign-language Learning Method Using Deep Learning, Special Issue on Smart

Interactions in Cyber-physical Systems: Humans, Agents, Robots, Machines and Sensors, vol.40, no.4, s.435-445, 2018.

[14] VASSILIA P.N., KONSTANTINOS M.G., Multimodal Continuous Recognition System For Greek Sign Language Using Various Grammars, Advances In Artificial Intelligence Proceedings, vol.3955, s.584-587, 2006.

[15] DREUW P., KEYSERS D., DESELAERS T., NEY H., Gesture Recognition Using Image Comparison Methods, Gesture in Human-Computer Interaction and Simulation , vol.3881, s.124-128, 2006

[16] MALIK M.S.A, KOUSAR N., ABDULLAH T., AHMED M., RASHEED F., AWAIS M., Pakistan Sign Language Detection using PCA and KNN, International Journal of Advanced Computer Science and Applications, vol.9, no.54, s.78-81, 2018.

[17] SHAH S.M.S., ABBAS H., KHAN J.I., RAMZAN M., ZULQARNAIN, KHANA H.U., Shape Based Pakistan Sign Language Categorization Using Statistical Features and Support Vector Machines, IEEE Access, vol.6, s.59242 - 59252, 2018.

[18] GONZALEZ G.S., SANCHEZ J.C., DIAZ M.M.B., PEREZ A.A., Recognition and Classification of Sign Language for Spanish, COMPUTACION Y SISTEMAS, vol.22, no.1, s.271-277, 2018.

[19] JIMENEZ J., MARTIN A., UC V., ESPINOSA A., Mexican Sign Language Alphanumerical Gestures Recognition using 3D Haar-like Features, IEEE Latin America Transactions, vol.15, no.10, 2017.

[20] VO D.H., HUYNH H.H., DOAN P.M., MEUNIER J., Dynamic Gesture Classification for Vietnamese Sign Language Recognition, International Journal of Advanced Computer Science and Applicatıons, vol.8, no.3, s.415-420, 2017.

[21] BUI T. D. and NGUYEN L.T. , Recognizing Postures in Vietnamese Sign Language With MEMS Accelerometers , IEEE Sensors Journal, vol.7, no.5, 2017.

[22] RONCHETTI F., QUİROGA F., ESTREBOU C., LANZARİNİ L., Handshape recognition for Argentinian Sign Language using Problem, Journal of Computer Science & Technology, vol.16, no.1, s.1-5, 2016.

[23] GANI E. and KIKA A., Albanian Sign Language (AlbSL) Number Recognition from Both Hand's Gestures Acquired by Kinect Sensors, International Journal of Advanced Computer Science and Applicatıons, vol.7, no.7, s.216-220, 2016.

[24] FAGIANI M., PRINCIPI E., SQUARTINI S., PIAZZA, F., Signer İndependent İsolated Italian Sign Recognition Based On Hidden Markov Models, Pattern Analysis and Applications, vol.18, no.2, s.385-402, 2015.

[25] LEE G.C., YEH F.H., HSIAO, Y.H., Kinect-Based Taiwanese Sign-Language Recognition System, Multimedia Tools and Applications, vol.75, no.1, s.261-279, 2016.

[26] ZADGHORBAN M. and NAHVI M., An Algorithm On Sign Words Extraction And Recognition Of Continuous Persian Sign Language Based On Motion

And Shape Features of Hands, Pattern Analysis and Applications, vol.21, no:2, s.323-335, 2018.

[27] HOLDEN E. J., LEE G., OWENS R., Australian Sign Language Recognition, Machine Vision And Applications, vol.16, no.5, s.312-320, 2005.

[28] KIM J. B., BIEN Z. N., Recognition Of Continuous Korean Sign Language Using Gesture Tension Model And Soft Computing Technique, IEICE Transactions on Information and Systems, vol.E87D, no.5, s.1265-1270, 2004.

[29] KIM J. S., JANG W., BIEN Z., A Dynamic Gesture Recognition System For The Korean Sign Language (KSL), IEEE Transactions on Systems, Man, and Cybernetics, Part B, vol.26, no.1, 1996.

[30] ZARE A. A., ZAHIRI S. H., Recognition of a real-time signer-independent static Farsi sign language based on fourier coefficients amplitude, International Journal of Machine Learning and Cybernetics, vol.9, no.5 ,s. 727–741, 2018.

[31] ALMEIDA S. G. M., GUIMARÃES F. G., RAMÍREZ, J. A., Feature extraction in Brazilian Sign Language Recognition based on phonological structure and using RGB-D sensors, Expert Systems with Applications vol.41, s.7259–7271, 2014.

[32] HRÚZ M., CAMPR P , DİKİCİ E., KINDIROĞLU A.A., KRŇOUL Z., RONZHİN A., SAK H., SCHORNO D., YALÇINH., AKARUN L., ARAN O., KARPOV A., SARAÇLAR M., ŽELEZNÝ M., Automatic Fingersign-To-Speech Translation System, Journal on Multimodal User Interfaces, vol. 4, no.2, s.61–79, 2011.

[33] ALTUN O. and ALBAYRAK S., Turkish Fingerspelling Recognition System Using Generalized Hough Transform Interest Regions And Local Descriptors, Pattern Recognition Letters , vol.32, s.1626–1632, 2011.

[34] HABERDAR H. and ALBAYRAK S., A Two-Stage Visual Turkish Sign Language Recognition System Based On Global And Local Features, Foundations of Intelligent Systems, vol.4203, s.19-37, 2006.

[35] Lee D., You W., Recognition Of Complex Static Hand Gestures By Using The Wristband-Based Contour Features, IET Journals Image Processing, vol.12, no.1 s.80-87, 2017.

[36] MEMİS A., ALBAYRAK S., Turkish Sign Language Recognition Using Spatio-temporal Features on Kinect RGB Video Sequences and Depth Maps, 21st Sıgnal Processıng And Communıcatıons Applicatıons Conference (SIU), 2013.

[37] CEBER Y. E., KARACAOGLAN E., UYSAL F., TOKMAKCI M., The Design of Glove That Can Translate Sign Language to Turkish Language, Medical Technologies National Congress (TIPTEKNO), 2017.

[38] SELDA B., VASİF N.V., Recognition of vowels letters of Turkish sign language by artificial neural networks, IEEE 14th Signal Processing and Communications Applications, 2006.

[39] LIANG Z., LIAO S., HU B., 3D Convolutional Neural Networks for Dynamic Sign Language Recognition, The Computer Journal, vol.61, no.11, s.1724–1736, 2018.

[40] HU Y., ZHAO H.F., WANG, Z.G., Sign Language Fingerspelling Recognition Using Depth Information and Deep Belief Networks, International Journal of Pattern Recognition and Artificial Inteligence, vol.32, no.6, 2018.

[41] AVOLA D., BERNARDI M. ,CINQUE L. , FORESTİ G.L., MASSARONI C., Exploiting Recurrent Neural Networks and Leap Motion Controller for the Recognition of Sign Language and Semaphoric Hand Gestures, IEEE Transactions on Multimedia, vol.21, no.1, 2019.

[42] DEVİKAR P., Transfer Learning For Image Classification Of Various Dog Breeds, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), vol.5, no.12, 2016.

[43] TAPAS, A., Transfer Learning for Image Classification and Plant

Phenotyping, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), vol. 5, no.11, 2016.

[44] ABADI, M., TensorFlow: Learning Functions at Scale, 21st ACM SIGPLAN International Conference on Functional Programming – ICFP, 2016.

[45] DONAHUE J., JIA Y., VINYALS O., HOFFMAN J., ZHANG N., TZENG E., DARRELL T., DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, International Conference on Machine Learning (ICML), 2014.

[46] ZESHAN Ulrike, Aspects of Türk İşaret Dili (Turkısh Sign Language), Sign Language & Linguistics, vol.6, no.1, s.43-75., 2003.

[47] DİKYUVA H. ve ZESHAN U., Türk İşaret Dili – Birinci Seviye, Ishara Press, Nijmegen, 2008.

[48] Türk İşaret Dili Sözlüğü, https://orgm.meb.gov.tr

[49] TÜİK, Türkiye Engelliler Araştırması, 2002, http://www.tuik.gov.tr

[50] MILES M., Signing in the Seraglio: Mutes, dwarfs and gestures at the

Ottoman Court 1500-1700, Disability & Society, vol. 15, no. 1, 115-134, 2000.

[51] Türk İşaret Dili Sözlüğü, Boğaziçi Üniversitesi Bilgisayar Mühendisliği.

[52] ULUDAĞ B., ULUDAĞ S., Online Türkçe İşaret Dili Sözlüğü. http://isaretce.com

[53] ÖZTEMEL, Ercan, Yapay Sinir Ağları, 3. Basım, Papatya Yayıncılık, 2012.

[54] LARSEN, Jan, Introduction to Artificial Neural Networks, Technical University Of Denmark, 1st Edition, 1999.

[55] KRIESEL, David, A Brief Introduction to Neural Networks, University of Bonn in Germany, 2005.

[56] HINTON G. E., OSINDERO S., TEH Y. W., A Fast Learning Algorithm For Deep Belief Nets, Neural computation, vol.18, no.7, s.1527-1554, 2006

[57] HAYKIN, Simon, Neural Networks A Comprehensive Foundation, Second Edition, Pearson Education, 1999.

[58] KAYAALP, Kıyas ve SÜZEN, Ahmet Ali, Derin Öğrenme ve Türkiye'deki Uygulamaları, IKSAD Yayınevi, ISBN 978-605-7510-53-2, 2018.

[59] HU W., HUANG Y., WEİ L., ZHANG F., Lİ H., Deep Convolutional Neural Networks For Hyperspectral Image Classification, Journal of Sensors, vol.2015, s.12, 2015.

[60] LECUN Y., BOTTOU L., BENGIO Y, HAFFNER P., Gradient-Based Learning Applied To Document Recognition, IEEE, vol. 86, no.11, s.2278 - 2324 ,1998.

[61] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E., ImageNet Classification with Deep Convolutional Neural Networks, Advances in neural information processing systems, vol.25, no.2, 2012.

[62] SIMONYAN, Karen and ZISSERMAN A., Very Deep Convolutional Networks for Large-Scale Image Recognition, Conference on Machine Learning, 2015.

[63] SZEGEDY C., LIU W., JIA Y.; SERMANET P., REED S., ANGUELOV D.; ERHAN D., VANHOUCKE V., RABİNOVICH A., Going Deeper With Convolutions, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[64] SZEGEDY C., VANHOUCKE V., IOFFE S., SHLENS J., WOJNA Z., Rethinking the Inception Architecture, Computer Vision and Pattern Recognition Conference, 2016.

[65] HE K., ZHANG X., REN S., SUN J., Deep Residual Learning for Image Recognition, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[66] TSL (Turkish Sign Language) Data Seti https://www.kaggle.com/feronial/turkish-sign-languagefinger-spelling

[67] Tensorflow - How to Retrain an Image Classifier for New Categories https://www.tensorflow.org/hub/tutorials/image_retraining

[68] ImageNet Stanford Vision Lab http://image-net.org/index

[69] RATHI D., Optimization of Transfer Learning Sign Language Recognition Targeting Mobile Platform, Internatonal Journal Recent Innovation Trends In Computing and Communication (IJRITCC), vol.6, no.4, 2018.

[70] DINESH S., SIVARPRAKASH S., KESHAV M., RAMYA K., Real-Time Amerikan Sign Language Recognition with Faster Regional Convolutional Neuralnetworks, Internatonal Journal Innovative Research in Science Engineering and Technology (IJIRITSET), vol.7, special Issue.2, 2018.

[71] Dzone-articles Comparison Between Deep Learning and Machine Learning https://dzone.com/articles/comparison-between deep-learning-vs-machine-learning