T.C.
KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
Bilgisayar Mühendisliği Bölümü

BİTİRME PROJESİ RAPORU

İŞARET DİLİ ÇEVİRİCİ

Derin Öğrenme Yöntemi ile Telefon Kamerası Üzerinden Örüntü Algılama

Adı Soyadı	Numarası	E-mail Adresi	Telefon No
M.E Berkay KOCAOĞLU	201513171070	meminberkaykocaoglu@gmail.com	+90 537 276 97 27
Batuhan GÜNEŞ	201513171055	batuhangunes001@gmail.com	+90 546 907 07 40
Hasan Hüseyin ÖZTUNÇ	201613171048	hasanhuseyin9e@gmail.com	+90 507 733 29 02

Danışman: Doç. Dr. Doğan Aydın

Proje Kaynak Kodları: https://github.com/BatuhanGunes/signLanguageConverter-Android

Rapor Tarihi: 22.06.2020

İÇİNDEKİLER

1. Bireysel Katkılar Dağılımı

1.1 Sorumluluk Matrisi

Görevlerin (Tasks/Activity) belirli roller veya kişilere göre bir tablo üzerinde sorumlularına haritalandırması işlevine RACI (Responsible, Accountable, Consultant, Informed) Matrisi denir. Matris 'deki atamalar İngilizce kelimelerin ilk harflerinden oluşmaktadır.

Sorumlu (Responsible): Görevi yapan/gerçekleştiren kişidir. Her bir görevde en az 1 kişi **R** olabilir. Bu rol için aşağıdaki sorulara cevap aranır.

- Bu işi kim yapacak?
- Bu işe kim atandı?

Yönetici (Accountable): Görevi durdurabilen, devam ettirebilen ve son kararı verebilen kişidir. Her göreve sadece bir **A** atanabilir. Proje yöneticisi, evde anne, baba gibi roller örnek gösterilebilir. Aşağıdaki sorulara cevap aranır bu rol için.

- Görev yanlış ilerlediğinde karar verici kim?
- Görevi yöneten kişi kim?

Danışman (Consultant): Görev yapılmadan hemen önce bilgisine başvurulması gereken kişidir. Bu kişi yapılacak görevle ilgili yeterli bilgiye sahip kişidir. Bir görevde çok fazla **C** olmamalıdır. Aksi durumda çok sesli ortamlarda karar çıkma olasılığı düşüktür. Aşağıdaki sorulara cevap aranır bu rol için.

- Görevin nasıl yapılacağı sürecinin nasıl işleyeceği konusunda en iyi pratiği kim söyleyebilir bana?
- Görev ile ilgili paydaş kim?

Bilgilendirilen (Informed): Görev yapıldıktan sonra, görevin bittiği konusunda bilgilendirilen kişidir. **I** sorumluluk alanında olan rol veya kişilere mail veya sms gibi yöntemlerle bilgi vermek en iyisidir. Aşağıdaki sorulara cevap aranır bu rol için.

- Çalışmaları bu göreve bağlı olan biri var mı?
- Görev ile ilgili herhangi bir güncellemede veya görev bitiminde kim haberdar olacak?

Yukarıdaki bilgiler esas alınarak proje için bir sorumluluk matrisi hazırlarmıştır. Bu sorumluluk matrisi tablo 1.1 sorumluluk matrisi tablosunda gösterilmiştir.

Aktivite \ Roller	Batuhan Güneş	M. E. Berkay Kocaoğlu	Hasan Hüseyin Öztunç
Rapor Hazırlama	A/R	R	R
Problem Tespit Etme	A/R	R	R
Çözüm Önerisi Sunma	A/R	R	R
Terimler Sözlüğü Oluşturma	R	R	A/R
Numaralandırılmış Fonksiyonel Gereksinimler	R	A/R	I
Numaralandırılmış Fonksiyonel Olmayan Gereksinimler	R	A/R	I
Aktörler ve Hedefleri Oluşturma	R	A/R	I
Kullanım Durumları (Sıradan Açıklama) Oluşturma	R	A/R	I
Kullanım Durumları (Tam Donanımlı) Oluşturma	R	A/R	I
Uygulama Arayüzü Tasarlama	A/R	I	I
Uygulama Arayüzünü Programlama	A/R	I	I
Mobil Uygulama Oluşturma	A/R	I	I
Mobil Uygulamaya TensorFlow Lite eklentisini ekleme	A/R	I	I
TensorFlow Lite ile alınan görüntüyü tanımlama	A/R	I	I
Tanımlanan görüntüyü ekrana yazdırarak metin oluşturma	A/R	I	I
Android kısmındaki önemli algoritmaların anlatımı	A/R	I	I
Kullanıcı Arayüzü Gereksinimleri Oluşturma	A/R	I	C
Sistem Sequence Diyagramlarını Oluşturma	R	A/R	R
Paydaşları Hazırlama	C	C	A/R
Dataset Oluşturma	R	A/I/R	R
Model Oluşturma	I	A/R	I
Model Test Etme	R	A/R	R
Model Dönüştürme	I	A/R	I

Tablo 1.1: Sorumluluk Matrisi Tablosu

1.2 Görev Dağılımı Çizelgesi

Aşağıda, tasarladığımız sistem için görev dağılımı çizelgesi bulunmaktadır.

Tablonun birinci bölümünde geliştiricilerin adı ve soyadı bilgileri bulunmakta. İkinci bölümde okul numaraları bulunmakta. Üçüncü bölümde ise hangi bölümde yer aldıkları ve hangi eğitim programında bulunduğu yazmaktadır. Son olarak dördüncü bölümde ise sistemimizde hangi görevleri yaptığı bulunmaktadır.

Adı ve Soyadı	Okul Numarası	Bölümü ve Eğitim Programı	Görevleri
Batuhan GÜNEŞ	201513171055	Kütahya Dumlupınar Üniversitesi, Mühendislik Fakültesi Bilgisayar Mühendisliği, Lisans Öğrencisi	• Kullanıcı problemlerinin belirlenmesi • Önerilen çözümü üretme • Kullanıcı arayüzü gereksinimleri oluşturma • Kullanıcı arayüzünü tasarlama • Android uygulamasını programlama • TensorFlow Lite frameworkü ile kameradan alınan görüntü ve modelden alınan veriler ile çıkarımlama da bulunma. • Uygulamanın ve modelin test edilmesi
Muhammed Emin Berkay KOCAOĞLU	201513171070	Kütahya Dumlupınar Üniversitesi, Mühendislik Fakültesi Bilgisayar Mühendisliği, Lisans Öğrencisi	• Bireysel katkılar dağılımı oluşturma • Numaralandırılmış fonksiyonel gereksinimler • İşlevsel gereksinimler (Aktörler ve hedefler, kullanım durumları (Use case) • Sistem sequence diyagramlarını oluşturma • Dataset Oluşturma • Model Oluşturma • Uygulamanın ve modelin test edilmesi
Hasan Hüseyin ÖZTUNÇ	201613171048	Kütahya Dumlupınar Üniversitesi, Mühendislik Fakültesi Bilgisayar Mühendisliği, Lisans Öğrencisi	• Terimler sözlüğünü hazırlama • Raporun kontrolü • Paydaşları hazırlama • Testlerin yönetimi ve gerçekleştirilmesi

Tablo 1.2: Görev Dağılımı Çizelgesi

2. Kullanıcı Gereksinimlerinin Belirlenmesi

2.1 Kullanıcı Problemlerinin Belirlenmesi

2.1.1 Problem

Günümüzde teknoloji hayatın birçok alanında çok sık kullanılmakta ve çok önemli bir yer teşkil etmektedir. Bu teknolojiye sahip olan ve sahip olup verimli bir şekilde kullanan kullanıcılar için günlük hayatta karşılaşılan bazı problemleri kolay, hızlı ve başarılı bir şekilde çözüm sunmakta. Şu an da insanlar tarafında sahip olma oranının çok yüksek olduğu bazı örneklere bakılacak olursa mesela akıllı telefonlar ve akıllı telefonlar ile birlikte kullanımı çok büyük bir şekilde artan sosyal medya uygulamaları insanların hayatlarında büyük yer edinmiş durumda.

Çok sık kullanılan bu teknolojiler kullanıcıların bazı temel ihtiyaçlarını karşılarken üreticiler tarafından kullanıcılar için kolaylık sağlayacak bazı özellikler ile desteklenmekte. Fakat geliştiriciler bu uygulamaları tasarlarken genelde bir hedef grubunu düşünerek onların ihtiyaçlarını en iyi şekilde karşılayacak, sorunlarına kolay çözümler bulabilecek tasarımlar yapmakta. Bu şekilde daha çok kullanıcıya ulaşabilir onlara iyi bir hizmet verebilirler. Fakat geliştiricilerin böyle bir yol izlemesi bazı kullanıcılara ulaşılamamasına sebep olmakta.

Geliştiriciler ve tasarımcıların hedef kitlelerinin dışında kalan topluluklardan birisi de işitme engelli insanlar. Her zaman söylenildiği gibi sosyal bir varlık olan insanın sosyal hayatının temelinde iletişim bulunmakta. Fakat işitme engelli kişiler için sosyal hayatın temeli olan iletişim kurmakta bazı zorluklar bulunmakta. Günlük hayatta ki işlerimizi yapabilmek için bazı kurumlara, devlet dairelerine ve en basitinden gittiğimiz bir makette bulamadığımız bir şeyi sorma ihtiyacı duymaktayız.

İşitme engelli insanlar için bu basit görünen işler çözülmesi çok zor durumlara dönüşebilir. İşitme engelli kişilerin iletişim için kullandıkları işaret dilini küçük bir işi halletmek için gittikleri yerdeki bir memur veya görevli bilmeyebilir ve bu iletişim kopukluğu sorunlara sebep olabilir. Günlük işler dışında arkadaşlık ilişkilerinde de zorluklar olabilir. İlerleyen ve gelişen teknoloji ile bu kişilere yardımcı olabilecek uygulamalar geliştirilebilir. İşitme engelli kişilerin sosyal hayattan soyutlanması veya zorlukların içinde boğulması problemine çözümler geliştirilebilir.

Teknolojinin gelişmesinin en büyük sebeplerinden biri olan insan hayatını kolaylaştırma fikri, buna büyük ölçüde ihtiyaç duyanlar için kullanılabilir.

2.1.2 Önerilen Çözüm

Bölüm 2.1.1 de anlatılan probleme çözüm olarak bir işaret dili çevirici kullanılabilir. Bu çevirici şu an kullanımı yaygınlaşmakta olan yapay zekâ uygulamaları kullanılarak yapılabilir. Tasarlanacak uygulamanın hedef kitlenin günlük hayatındaki işlerde kullanması amacıyla geliştirileceğinden mobil cep telefonlarında çalışması daha iyi bir seçenektir. Mobil cep telefonlarındaki kaliteli kameralar sayesinde kullanıcıların kendilerini ifade etmeleri için kullandıkları işaret dili kamera ile alınıp yorumlanabilir.

İşaret dilindeki kullanılan hareketlerin ne anlama geldikleri bir veri setinde toplanıp karşılaştırma verisi olarak kullanılabilir. Kameradan alınan kullanıcıların işaret dilindeki hareketleri oluşturulan veri seti ile karşılaştırılarak kullanıcının söyledikleri yazıya veya sese çevrilerek günlük hayattaki iletişim kopukluğuna bir çözüm sunulabilir. Oluşturulacak bu veri seti için her bir işaret teker kaydedilmeli. Bu verilerin karşılaştırma verisi olacak şekilde kullanılabilmesi için bir matematiksel forma getirilmeli.

Mobil cep telefonlarının kullanımının çok yaygın olması, taşınabilir olması, kullanımının kolay olması sebepleri ile bu ortamda çalışacak şekilde tasarlanabilir. Mobil telefonlarının kullanımının çok yaygın olmasından dolayı çok daha büyük bir kitleye ulaşılabilir. Şu anda kullanılan mobil telefonların kameralarının iyi kalitede fotoğraf çekmesi de bir artı olarak görülebilir.

Mobil telefonlarda kullanılmakta olan yapay zekâ uygulamaları algoritmaları ile birlikte bölüm 2.1.2 de problemin çözümü daha doğru çeviri yapan, daha hızlı çalışan bir uygulama geliştirilebilir. İşaret dilini yazı veya sese çevirme özelliklerinin yanında kullanıcıların işaret dilini öğrenmesine yardımcı olabilecek, işaret dili ile iletişim kuran insanlara işaret dili ile cevap vermelerini, konuşmalarını sağlayacak yazılan bir metni işaret diline çevirme özelliği eklenebilir. Bu şekilde hem işaret dili bilmeyip öğrenmek isteyen kullanıcılara bazı kolaylıklar getirilebilir.

2.2 Terimler Sözlüğü

Uygulamada kullanılan terimlerin tamamı tablo 2.2 terimler sözlüğü tablosu üzerinde gösterilerek açıklamaları yapılmıştır.

Sistem Gereksinimi	Sistem servislerinin tanımları ve gereksinimler mühendisliği sürecinde ortaya çıkan kısıtlamalardır.
Fonksiyonel Gereksinim	Sistemin sağlaması gereken hizmetleri ifade eder. Sistemin belirli girdilere nasıl tepki vermesi gerektiği ve sistemin belirli durumlarda nasıl davranması gerektiği tanımlanır.
Veri seti	Bir veri topluluğu.
Fonksiyonel Olmayan Gereksinim	Zamanlama kısıtlamaları, geliştirme sürecindeki kısıtlamalar, standartlar vb. gibi sistemin sunduğu hizmetler veya işlevler üzerindeki kısıtlamaları ifade eder.
Aktör	Sistemde görev alan kişi veya sistem parçaları.
Kullanıcı Durumu	Tek bir hedefi ya da görevi yerine getirecek özet bir senaryodur
Bilgisayar Görüsü	Gözümüzle yaptığımız algılama biçimini bilgisayara yaptırmaktır.
Akış Şeması	Yazılımların işlem basamaklarının geometrik şekiller ile gösterilmesi.
Derin Öğrenme	Makinelerin dünyayı algılamasında ve anlamasında kullanılan bir yaklaşım biçimi.
Örüntü Algılama	Derin öğrenme ile çıkarım yapma, karar verme ve kontrol
TextBox	Metin kutusu
ToolBox	Araç çubuğu
Gesture Recognizer	İnsan hareketlerini matematiksel algoritmalar yoluyla yorumlamayı amaçlayan bilgisayar bilimi ve dil teknolojisinde bir konudur.
Fragment	Bir Activity'de activity veya kullanıcı arabiriminin bir bölümünü temsil eder.
Kullanıcı arayüzü (UI – user interface)	Kullanıcı arayüzü bir hizmeti sunan sistem ile kullanan sistem arasında bir iletişim ve etkileşim köprüsü görevi gören kullanıcı arayüzlerinin tasarımı ile ilgili her şeydir.
Model	Derin öğrenimi ile oluşmuş, gönderilen veriler ile tahmin yapan bir üründür.
Image Classification	Resim verilerinin derin öğrenme ile sınıflandırılmasıdır.
CNN	Image classification yapılırken kullanılan derin öğrenme yöntemidir.
Transfer Learning	Bir model oluştururken, bir başka oluşturulmuş modeli ve özelliklerini kullanarak yeni bir model oluşturmaktır.

Tablo 2.2: Terimler Sözlüğü Tablosu

3. Sistem Gereksinimlerinin Belirlenmesi

3.1 Numaralandırılmış Fonksiyonel Gereksinimler

Aşağıda, Tablo 3.1: Numaralandırılmış Fonksiyonel Gereksinimler Tablosu üzerinde gösterilmekte olan, tasarladığımız sistem için her bir gereksinimin kapsamlı bir listesi bulunmaktadır.

Aşağıdaki tablo 3.1 üç bölümden oluşmaktadır. Birincisi numara (adlandırma) bölümüdür. Her gereksinime ait bir tanımlayıcı verir. İkincisi gereksinimin önemini belirten ağırlık derecesi (A.D.) bölümüdür. 1'den 5'e kadar verilen sayılar yardımıyla önemi temsil eder. Düşük olan ağırlık derecesi daha önceliklidir. Üçüncü bölüm ise her gereksinimin açıklamasını barındırır.

Numara	A.D.	Gereksinim
REQ-1	2	Kullanıcı, isteğe bağlı olarak kameradan yazıya çeviri, yazıdan görsele çeviri veya hakkımızda butonlarını seçebilmelidir.
REQ-2	1	Sistem, kameradan yazıya çeviri ekranına geçildiğinde kameranın doğrultulduğu kişinin el veya ellerini algılamalıdır.
REQ-3	1	Sistem, el veya ellerin hareketleri ile oluşturulan işaretlerin karşılığını veri seti içerisinde aramalıdır.
REQ-4	1	Kullanıcı, görüntüden yazıya çeviri ekranında algılattığı el veya ellerin hareketlerinin anlamlarını, ekranın alt kısmında metin şeklinde görebilmelidir.
REQ-5	2	Sistem, yazıdan görüntüye çeviri ekranına geçildiğinde tıklatılan butonun içeriğindeki kelime, harf veya rakam ifadesinin işaret dili karşılığını ekranda görüntüleyebilmelidir.
REQ-14	5	Kullanıcı, program geliştiricileri merak ettiğinde biz kimiz? ekranına girip öğrenebilmelidir.
REQ-15	3	Kullanıcı, görüntüden yazıya çeviri ekranında, çevrilen metin alanını sıfırlayabilmelidir.
REQ-16	3	Kullanıcı, görüntüden yazıya çeviri ekranında, hatalı bir çeviri durumda metnin son yazılan karakterini silebilmelidir.
REQ-17	4	Kullanıcı, girdiği bir ekranda başlık kısmının yanında bulunan geri tuşu ile ana sayfaya geri dönebilmelidir.
REQ-19	4	Kullanıcı, yazıdan görüntüye çeviri ekranında, alt tarafta bulunan tuş takımını sağ ve sol eksenli el hareketleri ile değiştirebilmelidir.
REQ-20	1	Sistem, model ile iletişime geçip kameradan alınan görüntü üzerinden bir çıkarımda bulunmalıdır.

Tablo 3.1: Numaralandırılmış Fonksiyonel Gereksinimler Tablosu

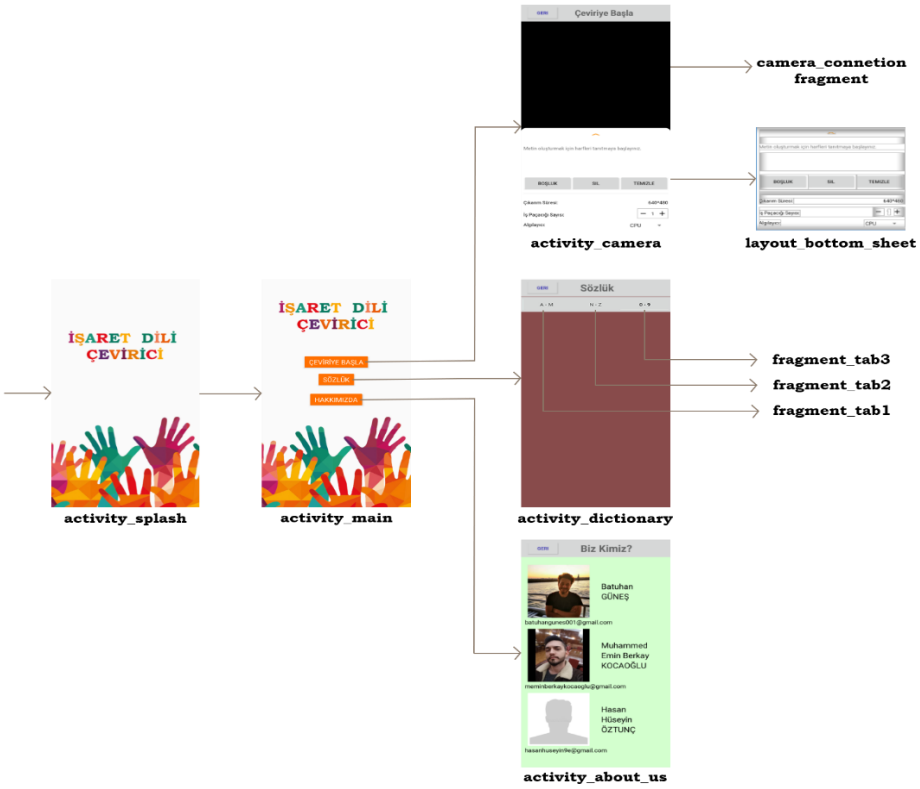
3.2 Numaralandırılmış Fonksiyonel Olmayan Gereksinimler

Numara	A.D.	Gereksinim
REQ-7	2	Sistem tüm kullanıcılar için açık olmalıdır. Üyelik gerektirmemelidir
REQ-8	1	Kamera el veya el hareketlerini algılamalıdır.
REQ-9	2	Sistem algıladığı el işaretlerini, algıladığı sıra ile yazdırmalıdır.
REQ-10	1	Veri seti içerisinde harf, rakam için veri bulunmalıdır.
REQ-11	3	Sistem dili Türkçe olmalıdır.
REQ-12	4	Biz kimiz? kısmı bulunmalı ve geliştirici bilgileri içerisinde yazmalıdır.

Tablo 3.2: Numaralandırılmış Fonksiyonel Olmayan Gereksinimler Tablosu

3.3 Kullanıcı Arayüzü Gereksinimleri

Bu başlık altındaki her alt bölüm, bu proje içerisindeki her bir arabirimin şemalarını içermektedir. Her bir arabirimin ne olduğunu ne için kullanılacağını ve ne zaman kullanıcıya gösterileceğini açıklayan kısa açıklamaları sunulmaktadır.



Şekil 3.3: Uygulama Akış Şeması

Bu uygulamanın tamamında ekranın dik bir şekilde kullanılması sağlanmıştır. Yatay kullanım sınırlandırılmıştır. Uygulama başlangıcında kullanıcıya uygulamanın açıldığına dair bir başlangıç ekranı gösterilmektedir. Bu başlangıç ekranı gösterimi 1 saniye sürdükten sonra kullanıcı Ana Sayfa Ekranını görmektedir. Bu başlangıç sayfasının detayları 3.3.1 numaralı başlık altında anlatılmaktadır.

"Ana Sayfa Ekranı" üzerinde bulunan butonlar ile diğer sayfalara geçiş yapılabilmektedir. Bu geçişlerin anlatıldığı yukarıda gösterilmekte olan Uygulama Akış Şemasını şekil 3.3 üzerinde görebilirsiniz.

3.3.1 Başlangıç Ekranı

Şekil 3.3.1: Başlangıç Ekranı

Uygulama başlangıcında kullanıcıya uygulamanın açıldığına dair bir başlangıç ekranı gösterilmektedir. Kullanıcıya sunulan ilk ekrandır. Bu başlangıç ekranı gösterimi 1 saniye sürdükten sonra kullanıcı Ana Sayfa Ekranını görmektedir. Diğer bir adı Splash screen olarak da bilinmektedir. Telefon uygulamalarında sıkça kullanılan tasarımsal bir yöntemdir. Bu başlangıç sayfasının detayları 3.3.1 numaralı şekilde görebilirsiniz.

Bu ekranın amacı sadece kullanıcıya uygulamanın başlatıldığını göstermektir. Kullanıcı bu ekran üzerinde herhangi bir işlem gerçekleştiremez. Kullanıcı için görsel bir geçiş sağlanmıştır. Bu görsel kısa süreli kullanıcıya gösterileceğinden dolayı sade ve az içerikli olması ön planda tutulmuştur.

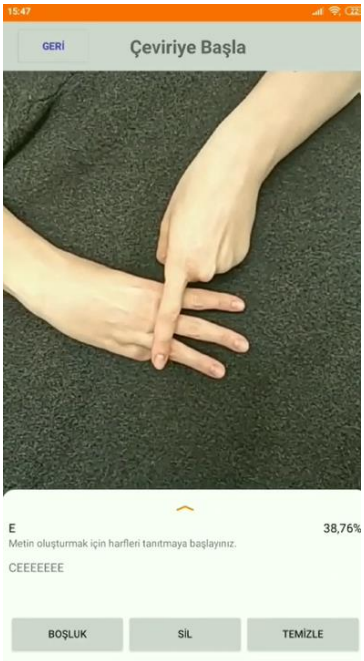
3.3.2 Ana Sayfa Ekranı

Şekil 3.3.2: Ana Sayfa Ekranı

Başlangıç ekranının kullanıcıya bir saniyelik gösteriminden hemen sonra bu sayfa kullanıcıya sunulmaktadır. Bu sayfa uygulama açıldıktan sonra kullanıcının etkileşime geçebileceği ilk sayfadır. Bu sayfa üzerinde 3 adet buton bulunmakta. Bu butonlara tıklama işlemi gerçekleştirerek farklı işlemler için farklı ekranlara geçiş sağlanmaktadır.

Geçiş sağlandıktan sonra tekrar farklı bir ekrana geçiş yapmak için kullanıcılar yine bu sayfaya dönmek zorundadırlar. Yani bu sayfayı uygulamanın merkezi olarak da adlandırabiliriz. Bu sayfanın tasarımında yine kullanıcının gözünü rahatsız etmemek için en az bileşen ile sade bir görüntü elde etmeye ve aynı zamanda renkli ifadeler ile dikkat çekici olmasına özen gösterdik.

3.3.3 Görüntüden Yazıya Çeviri Ekranı



Şekil 3.3.3: Görüntüden Yazıya Çeviri Ekranı

Kullanıcının Ana sayfa ekranı üzerinden “Görüntüden Yazıya Çeviri” butonuna tıklama aksiyonu ile bu ekranın gösterimini gerçekleştirdik. Bu sayfanın görselini 3.3.3 numaralı şekilde görebilirsiniz.

Bu ekranda cihaz kamerasının aktif hale getirilmesi ile birlikte kameradan alınan görüntülerin gösteriminin gerçekleştirileceği bir görüntü alanı oluşturduk. Bu görüntü alanından alınan fotoğraflar üzerinde “bilgisayar görüsü ve derin öğrenme ile örüntü algılama” teknolojisi kullanılarak görüntü üzerinden önceden tanımlanmış olan örüntüleri yani el ile yapılan işaret dili işaretlerini tanımlamış olduk.

Bu tanımladığımız el işaretinin Türkçe alfabesi içindeki karşılığını kullanıcıya göstermek amacı ile sayfanın alt tarafında bulunan text box'a yazılacak. Her tanımlanan yeni harf eklenerek sonunda bir metin oluşturulması sağlanacaktır. Bu işlemin sonunda işaret dili ile kodlanan bir metnin yazı karşılığını elde etmiş olacağız. Böylelikle işaret dili ile iletişim kuran biri işaret dili bilmeyen bir kişi ile iletişim kuramazken uygulamamızla kurabilecektir.

Metin bloğunun altında metin ile ilgili işlemlerin yapılabilmesi için bir tool box yerleştirilmiştir. Bu tool box içerisinde 3 adet farklı fonksiyon bulunmaktadır.

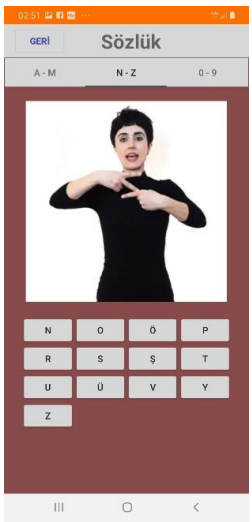
Bu fonksiyonlardan sonuncusu olan "Temizle" butonu metin alanındaki tüm yazıyı silerek yeni yazının eklenmesine olanak sağlamaktadır.

İkinci fonksiyon ise "Sil" fonksiyonudur. Bu fonksiyon ile yanlış tanımlanmış bir karakterin metin alanından kaldırılmasını sağlamaktadır. Kaldırılacak bu karakter en son tanımlanmış olan karakter olacaktır. Bu fonksiyon aynı zamanda çift taraflı olarak gerçekleştirilebilmektedir. Yani Telefon kullanıcısı bu fonksiyonu bu buton yardımıyla yapabilirken, işaret dilini kullanan kullanıcı uygulama için geliştirdiğimiz sil işaretini kullanarak da bu fonksiyonu aktif hale getirebilmektedir. Kaldırma işleminden sonra kameradan yeni harf tanımlama işlemine devam edilebilecektir.

İlk fonksiyon olan "Boşluk" fonksiyonu ise bir kelimenin oluşumu bittikten sonra yeni bir kelimeye başlamak için gerekli olan boşluk karakterinin metin alanına yazdırılmasını sağlamaktadır. Bu fonksiyonda sil fonksiyonu gibi çift taraflı çalışabilmektedir. Boşluk için türetilmiş el hareketini kullanarak veya bu butona basarak bu fonksiyon aktif hale gelebilmektedir.

Aynı zamanda en üstte bulunan başlık kısmı ile hangi sayfada bulunduğumuz kullanıcıya aktarılmıştır. Bu başlık kısmının solunda bulunan geri tuşu ile kullanıcı bir önceki sayfaya dönebilmekte ve farklı sayfalara oradan geçiş yapılabilmektedir.

3.3.4 Yazıdan Görüntüye Çeviri Ekranı



Şekil 3.3.3: Yazıdan Görüntüye Çeviri Ekranı

Bu ekranda ise işaret dili ile yapılan şekillerin gösterilmesi için bir gif alanı tanımlandı. Bu tanımlanan alanın içerisindeki gifleri değiştirmek adına ekranın alt tarafına rakamları ve harfleri belirten butonlar koyuldu. Bu butonlara tıklanma aksiyonlarına göre gif alanındaki gifler değiştirilmektedir. Böylece tıklanan butonun karşılığı hareketli resimler ile kullanıcıya gösterilmektedir.

Butonların fazla olmasından dolayı görsel bir karmaşıklığa yol açmaması için tuş takımı üç kısma ayrıldı.

Bu kısımlar kullanıcı tarafından yapılan sağ ve sola doğru akan el hareketleri ile yani slide gesture recognizer ile kontrol edilmektedir. Aynı zamanda kaçıncı alanda olduğunu kullanıcıya göstermek amacı ile tool bar alanı kullanıldı. Bu ikon ile aktif olan tuş takım sayfası kullanıcıya gösterildi. Her tuş takımı kendine özel bir fragment sayfası içerisinde oluşturuldu. Böylece görsel bir kalabalık önlenmiş oldu aynı zamanda bu alanın kullanımı da gelişmiş oldu.

Bu ekrandaki işlemler sayesinde kullanıcının türkçe alfabesindeki herhangi bir harfin veya rakamın işaret dili karşılığını görsel olarak görebilmesi sağlandı. Aynı zamanda ekranda gösterilen fotoğrafın üstüne çift tıklama yani double tab gasture recognizer aksiyonu ile 2x yakınlaştırma işlemi kullanıcı tarafından yapılabilmektedir. Bu yapılan yakınlaştırma işlemi için gerçekleştirilen çift tıklama aksiyonu tekrar yapıldığı halde fotoğraf bu sefer başlangıç durumuna dönmektedir.

Başka bir farklı yakınlaştırma işlemi olarak ise iki parmak hareketi ile yapılan büyültme küçültme yani pinch gesture recognizer işlemi yapılabilmektedir. Bu işlem ile kullanıcı manuel olarak büyültme küçültme işlemi yapabilmekte. Bu işlem sonrası fotoğrafın başlangıç konumuna dönmesi için yine çift tıklama aksiyonu gerçekleştirilmelidir. Bu yakınlaştırma işlemleriyle yapılan işaretlerin kullanıcı tarafından detaylı bir şekilde incelenmesine olanak sağlanmak istenilmiştir.

Görüntüden yazıya çeviri ekranında da olduğu gibi en üstte bulunan başlık kısmı ile hangi sayfada bulunduğumuz kullanıcıya aktarılmıştır. Bu başlık kısmının solunda bulunan geri tuşu ile kullanıcı bir önceki sayfaya dönebilmekte ve farklı sayfalara oradan geçiş yapılabilmektedir.

3.3.5. Biz Kimiz? Ekranı

Şekil 3.3.5: Biz Kimiz? Ekranı

Bu ekran üzerinde ise bu projede çalışan kişilerin isimleri fotoğrafları ve iletişim amacı ile koyulan e-mail adresleri bulunmaktadır. Bu bilgilerin detaylı halini bu belgenin başındaki sayfada bulabilirsiniz. Kullanıcı bu sayfa üzerinde tek bir işlem yapabilmektedir. Bu işlem ise bir önceki sayfaya dönebilmektir.

Bu sayfada da diğer geçiş sayfalarında olduğu gibi en üstte bulunan başlık kısmı ile hangi sayfada bulunduğumuz kullanıcıya aktarılmıştır. Bu başlık kısmının solunda bulunan geri tuşu ile kullanıcı bir önceki sayfaya dönebilmekte ve farklı sayfalara oradan geçiş yapılabilmektedir.

4. İşlevsel Gereksinimler

4.1 Paydaşlar

Paydaş	Açıklama
Proje Takımı	• Sign Language Converter projesini geliştir. • Projenin dışarından gelen saldırılara karşı savunmasından sorumludur. • Projenin istenildiği ve gerektiği şekilde çalışıp çalışmadığını kontrol eder. • Sonradan ortaya çıkan uygulama ile ilgili hataları düzeltir.
Kullanıcı	• Sign Language Converter uygulamasını kullanılır. • Uygulama ile işaret dilini yazıya çevirebilir. • Uygulamayı kullanarak yazıları işaret diline çevirir.
Proje Yöneticisi	• Proje işleyişini proje takımından aldığı raporlar ile denetler. • Aldığı raporlarda projede hatalı olan işler görürse proje takımını uyarır.
Test Grubu	• Ortaya çıkan projeyi test eder. • Uygulamanın daha kullanışlı olması için kendi fikirlerini dile getirir. • Uygulamada bir hatayla karşılaşırsa proje takımına bildirir.
Veri analisti	• Uygulamada kullanılacak işaret dili veri setini oluşturur. • Veri setini projede kullanılabilecek şekilde düzenler.
İşaret dili Çevirmeni	• Veri seti oluşturan veri analistine veri setini oluşturmak için gerekli işaret dilinin hareketlerini gösterir. • Veri setinde kullanılacak işaretlerin fotoğraflarının çekilmesine yardımcı olur.

Tablo 3.6.1

4.2 Aktörler ve Hedefler

Aşağıda tasarladığımız sistem için her bir aktörlerin ismi, bu aktörlerin hedefleri ve kullanım durumları (Use Case) açıklayıcı bir listesi bulunmaktadır.

Tablo 3 bölümden oluşmaktadır. Birinci bölüm Aktörlerdir. Sistemde görev alan aktörlerin isimleri bulunmaktadır. İkinci bölümde bu aktörlerin yapacağı işler Hedefler adı altında toplanmaktadır. Her bir aktörün birden fazla hedefi olacağı için, tek bir satırda toplamak yerine tüm hedefler ayrı satırlarda toplanmıştır. Üçüncü bölüm ise Use Caseler'dir, yani kullanım durumlarıdır. Başlık 3.4.2 Kullanım Durumları (Use Case)'in altında bulunan 3.4.2.1 Sıradan Açıklama kısmındaki Kullanım Durumları Sıradan Açıklama Tablosu'nda bulunan Use Case'ler kısaca isimleri, açıklamaları ve gereksinimleri belirtilmiştir. Bu Use Caseler Aktörler ve Hedefler Tablosuna da yazılarak bir aktörün bir hedefi gerçekleştirirken kullanacağı Use Caseleri neler yapacağını belirtir.

Aktörler	Hedefler	Use Caseler
Kullanıcı	Programı açar ve görüntüden yazıya çeviri, yazıdan görüntüye çeviri veya biz kimiz? ekranını açar.	UC-1, UC-2, UC-5
Kullanıcı	İşaret dili kullanan birisine telefon kamerasını tutarak, metin karşılığını görür.	UC-1, UC-3
Kullanıcı	Merak ettiği harf veya rakama tıklayarak bu harf veya rakamın işaret dili karşılığını görür.	UC-2, UC-4
Kullanıcı	Biz Kimiz? ekranında geliştirici bilgilerini görür.	UC-5
Kullanıcı	Çevrilen metini sıfırlar.	UC-7
Kullanıcı	Çevrilen metnin son karakterini siler.	UC-8
Kullanıcı	Herhangi bir ekrandan ana sayfaya döner.	UC-9
Kullanıcı	Yazıdan görüntüye çeviri ekranında tuş takımını kaydırarak değiştirir.	UC-2, UC-10
Kamera	El ve eller ile yapılan şekilleri algılar.	UC-1, UC-3
Veri seti	Türkçe işaret dilinde bulunan harfleri ve rakamları bünyesinde barındırarak, kullanıcı kullanacağı zaman sistem ile paylaşır. Kamera üzerinden alınan görüntü ile karşılaştırma işlemi için kaynak veri olarak kullanılır.	UC-1, UC-3
Ekran	Görüntüden yazıya çeviri ekranında algılanan işaretlerin yazı olarak karşılığını ekranda bulunan metin alanında gösterir.	UC-1, UC-3
Ekran	Yazıdan görüntüye çeviri ekranında tıklanan bir buton içeriğinin karşılığı olan işaretleri üst kısımda gösterir.	UC-2, UC-4
Ekran	Hakkımızda ekranında geliştirici bilgilerini gösterir.	UC-5

Tablo 3.3.1: Aktörler ve Hedefler Tablosu

4.3 Kullanım Durumları (Use Case)

4.3.1 Sıradan Açıklama

Aşağıda tasarladığım sistem için kullanım durumları (Use Case) ve bu kullanım durumlarının ismi, açıklaması ve gereksinimleri olan açıklayıcı bir liste bulunmaktadır.

Tablo 4 bölümden oluşmaktadır. Birinci bölüm Use Casedir. Her bir isme ait bir tanımlayıcı verir. İkinci bölüm isimlerdir. Use Casenin yapacağı işin birkaç kelime ile temsil edilen bölümdür. Üçüncü bölüm açıklama bölümüdür. Use Caseyi kısaca açıklar. Dördüncü ve son bölüm ise gereksinimler bölümüdür. 3.1 altında bulunan Numaralandırılmış Fonksiyonel Gereksinimler Tablosundaki ile 3.2 altında bulunan Numaralandırılmış Fonksiyonel Olmayan Gereksinimler Tablosundaki gereksinim numaraları bulunmaktadır.

Use Case	İsim	Açıklama	Gereksinimler
UC-1	goruntudenYaziya CeviriEkraniAcma	Görüntüden yazıya çeviri ekranı açılır.	REQ-1, REQ-7
UC-2	yazidanGoruntuye CeviriEkraniAcma	Yazıdan görüntüye çeviri ekranı açılır.	REQ-1
UC-3	canliCeviri	Kameradan alınan fotoğraflar üzerindeki el hareketlerini algılayarak yazıya çevirir.	REQ-1, REQ-2, REQ-3, REQ4, REQ-8, REQ-9, REQ-11
UC-4	karakterCevirisi	Tıklanan karakterin işaret dili karşılığındaki görüntüsünün gösterilmesi.	REQ-1, REQ-5, REQ-11, REQ-20
UC-5	bizKimizEkraniAcma	Biz Kimiz? ekranını açar.	REQ-1, REQ-12, REQ-14
UC-6	metinSifirlama	Çevrilmiş metin bloğunu sıfırlanır.	REQ-1, REQ-2, REQ-3, REQ-4, REQ-16
UC-7	karakterSilme	Çevrilmiş metinden son karakteri siler.	REQ-1, REQ-2, REQ-3, REQ-4, REQ-17
UC-8	anaSayfayaDon	Geçiş sayfalarından bulunan başlık kısmındaki geri tuşu ile ana sayfaya geri dönülür.	REQ-1
UC-9	tusTakiminiDegistirme	Tuş takımı sağ ve sol el hareketleri ile değiştirilir.	REQ-1

Tablo 3.3.2.1: Kullanım Durumları Sıradan Açıklama Tablosu

4.3.2 Tam Donanımlı Açıklama

Aşağıda tasarladığımız sistem için tam donanımlı açıklama tabloları bulunmaktadır.

Her bir tablo 7 bölümden oluşmaktadır. Birinci bölümde Use Case adı vardır. 3.3.2.1 altında Kullanım Durumları Sıradan Açıklama tablosundaki Use Caseler buradan burada detaylı olarak açıklanmak üzere isimleri yazılır. İkinci bölümde Use Casenin gereksinimi yani 3.1 altında bulunan Numaralandırılmış Fonksiyonel Gereksinimler Tablosundaki ile 3.2 altında bulunan Numaralandırılmış Fonksiyonel Olmayan Gereksinimler Tablosundaki gereksinim numaraları bulunmaktadır.

Üçüncü bölüm aktör bölümüdür. O Use Case'i kullanan aktörün ismi mevcuttur. Dördüncü bölümde aktörün hedefi yani yapmak istediği bulunmaktadır. Tablonun beşinci bölümünde mevcut hedefi gerçekleştirmek için gereken ön koşullar yer almaktadır. Altıncı bölümde ise gerçekleşmesini istediğimiz hedefin, hedef şartlar kısmı bulunmaktadır. Son bölüm olan yedinci bölümde ise olayların akışı bulunmaktadır. Her bir olayın adım adım ilerleyişini anlatmaktadır.

Use Case UC-1:	görüntüdenYaziyaCeviriEkraniAcma
Gereksinimler:	REQ-1
Aktör:	Kullanıcı, Ekran
Aktörün Hedefi:	Ana sayfadan hedef sayfaya geçiş sağlamak.
Ön Koşullar:	Uygulama "Ana Sayfa" ekranını gösteriyor olmalı ve kullanıcının kamera izinlerini onaylamış olmalı.
Hedef Şartlar:	Görüntüden yazıya çeviri ekranı açılmalı.
Olayların Akışı:	
→ Kullanıcı uygulamaya tıklayarak açar. ← Uygulama ekranda açılır. → Kullanıcı uygulamada "Görüntüden Yazıya Çeviri" butonuna tıklar. ← Görüntüden yazıya çeviri ekranı açılır.	

Use Case UC-2:	yazıdanGörüntüyeCeviriEkraniAcma
Gereksinimler:	REQ-1
Aktör:	Kullanıcı, Ekran
Aktörün Hedefi:	Ana sayfadan hedef sayfaya geçiş sağlamak.
Ön Koşullar:	Uygulama "Ana Sayfa" ekranını gösteriyor olmalı.
Hedef Şartlar:	Yazıdan görüntüye çeviri ekranı açılmalı.
Olayların Akışı:	
→ Kullanıcı uygulamaya tıklayarak açar. ← Uygulama ekranda açılır. → Kullanıcı uygulamada "Yazıdan Görüntüye Çeviri" butonuna tıklar. ← Yazıdan görüntüye çeviri ekranı açılır.	

Use Case UC-3:	canliCeviri
Gereksinimler:	REQ-1, REQ-2, REQ-3, REQ4, REQ-8, REQ-9, REQ-11
Aktör:	Kamera, Veri seti, Ekran
Aktörün Hedefi:	Kamera karşısındaki işitme dili kullanan bireyin el hareketlerini algılayarak, işaret dili bilmeyen bireyin anlayabilmesi için metin kısmına yazdırılması.
Ön Koşullar:	Kameradan yazıya çeviri ekranı ve kamera açık olmalı. Kameranın karşısında bulunan bireylerin elleri kameraya dönük olmalı. Ortam ellerin kamera tarafından görüntülenebileceği bir aydınlıkta olmalı.
Hedef Şartlar:	İşaret dili yazıya çevrilmeli.
Olayların Akışı:	
→ Kullanıcı kameradan yazıya çeviri ekranına girer. (UC-1 dahil edildi.) ← Kamerayı karşısındaki işitme dili kullanan bireye doğrultur. → Kamera bireyin ellerini algılar. ← Eller veri setindeki veriler ile karşılaştırılır. → Karşılaştırmanın sonucu uygulama tarafından hesaplanır. ← Karşılaştırmanın sonucunda çıkan karakter ekranın alt köşesinde bulunan alana yazdırılır.	

Use Case UC-4:	karakterCevirisi
Gereksinimler:	REQ-1, REQ-5, REQ-11, REQ-20
Aktör:	Ekran
Aktörün Hedefi:	Kullanıcının işaret dili karşılığını merak ettiği harfin veya rakamın bulunduğu butona tıklayarak görmek.
Ön Koşullar:	Yazıdan görüntüye çeviri ekranı açık olmalı.
Hedef Şartlar:	Herhangi bir karakter işaret dili karşılığı gösterilmeli.
Olayların Akışı:	
→ Kullanıcı yazıdan görüntüye çeviri ekranına girer. (UC-2) ← Kullanıcı çevirmek istediği karaktere tıklar. → Kullanıcının tıklamış olduğu karakterin karşılığı olan fotoğraf sistem tarafından bulunur. ← Bulunan fotoğraf, fotoğraf alanında işaret dili karşılığı olarak kullanıcıya gösterilir.	

Use Case UC-5:	bizKimizEkraniAcma
Gereksinimler:	REQ-1, REQ-12, REQ-14
Aktör:	Ekran
Aktörün Hedefi:	Ana sayfadan hedef sayfaya geçiş sağlamak.
Ön Koşullar:	Uygulama "Ana Sayfa" ekranını gösteriyor olmalı.
Hedef Şartlar:	Sayfa açıldıktan sonra geliştirici bilgileri mevcut olup, gösterilmeli.
Olayların Akışı:	
→ Kullanıcı uygulamaya tıklayarak açar. ← Uygulama ekranda açılır. → Kullanıcı uygulamada "Biz Kimiz?" butonuna tıklar. ← Biz Kimiz? ekranı açılır. → Geliştirici bilgileri ekranda gösterilir. ← Kullanıcı merak ettiği geliştirici bilgilerini öğrenir.	

Use Case UC-6:	metinSifirlama
Gereksinimler:	REQ-1, REQ-2, REQ-3, REQ-4, REQ-16
Aktör:	Kamera, Ekran
Aktörün Hedefi:	Çevrilmiş metni sıfırlamak.
Ön Koşullar:	Görüntüden yazıya çeviri ekranı açık olmalı ve sıfırla tuşuna tıklanabilmeli.
Hedef Şartlar:	Çevrilmiş metin alanı sıfırlanabilmeli.
Olayların Akışı:	
→ Kullanıcı, kameradan yazıya çeviri ekranına girer. (UC-1 dahil edildi.) ← Kullanıcı, sıfırlama tuşuna tıklar. → Çevrilmiş metin alanı sıfırlanır.	

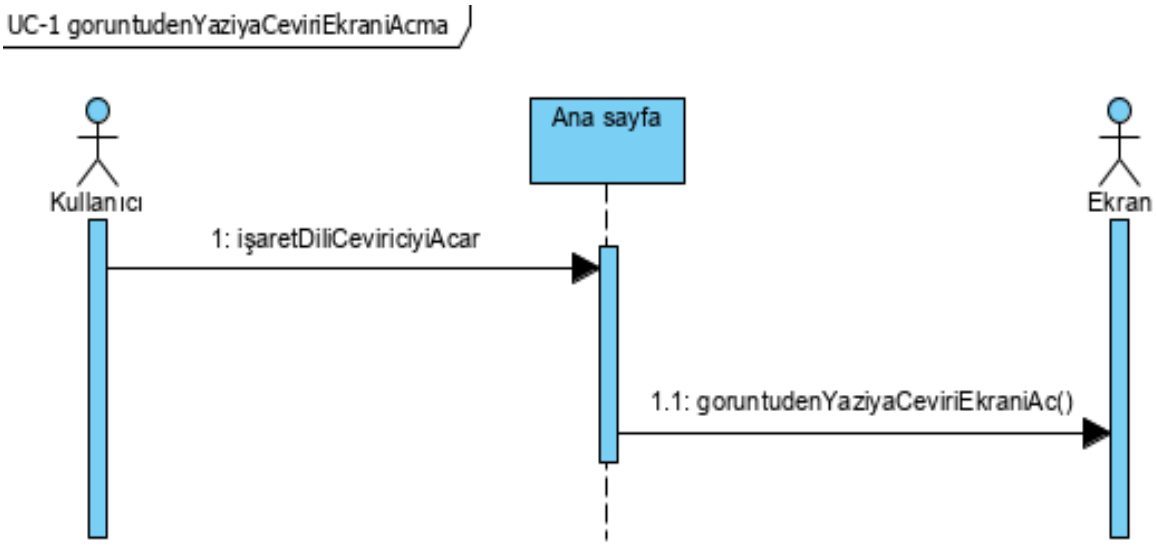
Use Case UC-7:	karakterSilme
Gereksinimler:	REQ-1, REQ-2, REQ-3, REQ-4, REQ-17
Aktör:	Kamera, Ekran
Aktörün Hedefi:	Kullanıcı hedefi çevrilmiş metinden karakter silmek.
Ön Koşullar:	Kameradan yazıya çeviri ekranı açık olmalı ve silme tuşuna tıklanabilmeli.
Hedef Şartlar:	Çevrilmiş metin alanından son karakter silinmeli.
Olayların Akışı:	
→ Kullanıcı, kameradan yazıya çeviri ekranına girer. (UC-1 dahil edildi.) ← Kullanıcı, silme tuşuna kaç karakter silmek istiyorsa o kadar tıklar. → Çevrilmiş metinden tıkanıldığı kadar karakter silinir.	

Use Case UC-8:	anaSayfayaDon
Gereksinimler:	REQ-1
Aktör:	Kullanıcı, Ekran
Aktörün Hedefi:	Geçiş yapılan sayfa üzerinden ana sayfaya geri dönmek.
Ön Koşullar:	Önceden ana sayfa üzerinden herhangi bir sayfaya geçiş sağlanmalı.
Hedef Şartlar:	Geri tuşuna basıldığında ana sayfaya dönülmeli.
Olayların Akışı:	
→ Kullanıcı, uygulamayı açar. ← Kullanıcı, ana sayfa üzerinden bir butona tıklayarak farklı bir sayfaya geçiş sağlar. → Kullanıcı, geçiş yaptığı sayfa üzerindeki başlık kısmında bulunan geri tuşuna basar. ← Uygulama, aktif olan sayfayı sonlandırarak ana sayfaya geri döner.	

Use Case UC-9:	tusTakiminiDegistirme
Gereksinimler:	REQ-1, REQ-20
Aktör:	Kullanıcı, Ekran
Aktörün Hedefi:	Yatayda yapılan bir el hareketi ile tuş takımını değiştirmek
Ön Koşullar:	Yazıdan görüntüye çeviri ekranı açık olmalı.
Hedef Şartlar:	Tuş takımı değişmeli
Olayların Akışı:	
→ Kullanıcı, yazıdan görüntüye çeviri ekranına girer. (UC-2) ← Kullanıcı, tuş takımı üzerinde yatay eksende sağa veya sola doğru bir el hamlesi yapar. → El hamlesi doğrultusuna göre tuş takımı o yönde değişir. ← Değişim page control üzerinde kullanıcıya gösterilir.	

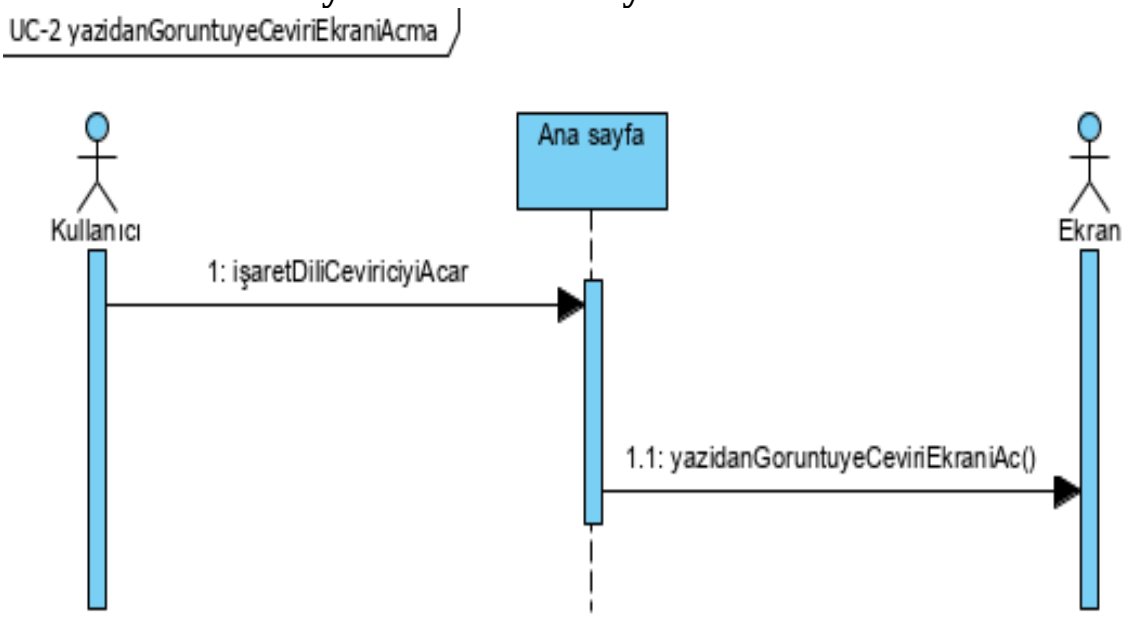
4.4 Sistem Sequence Diyagramları

4.4.1 Use Case 1: goruntudenYaziyaCeviriEkraniAcma



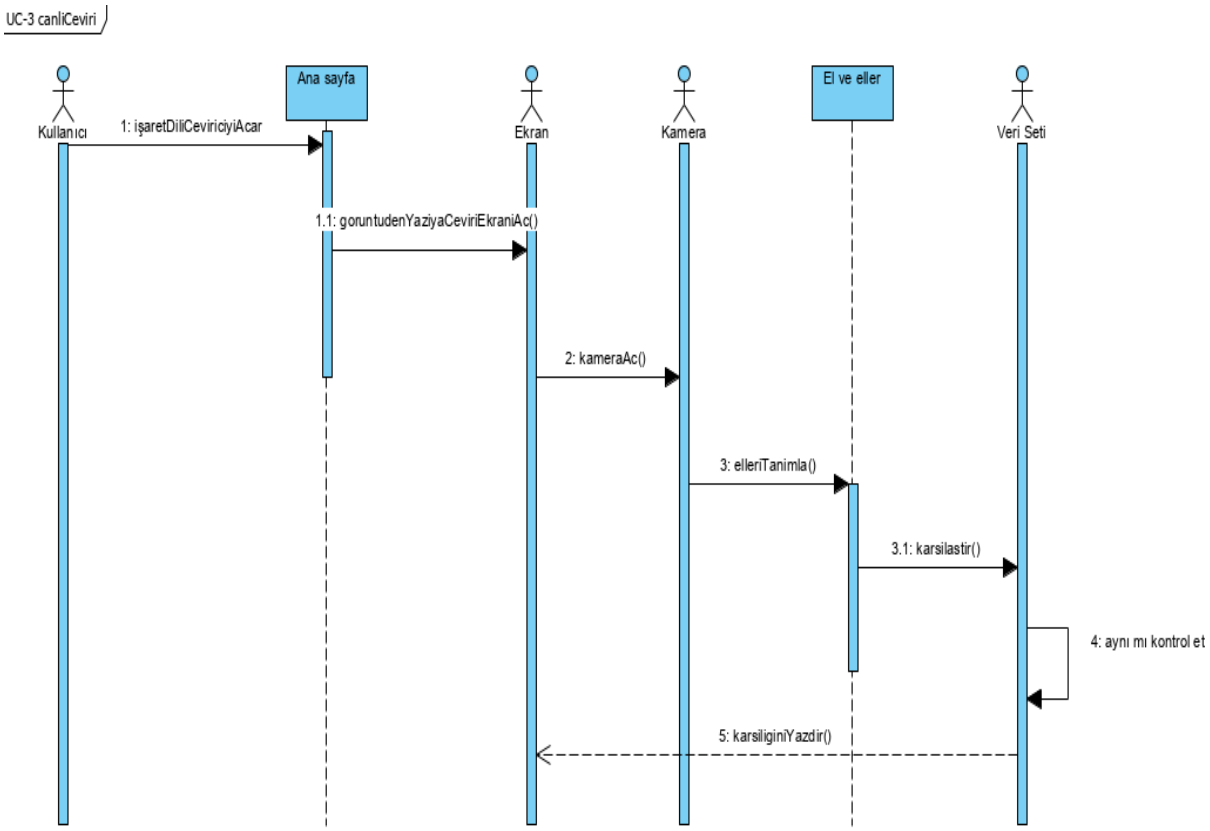
Şekil 4.4.1: goruntudenYaziyaCeviriEkraniAcma Sistem Sequence Diyagramı

4.4.2 Use Case 2: yazidanGoruntuyeCeviriEkraniAcma



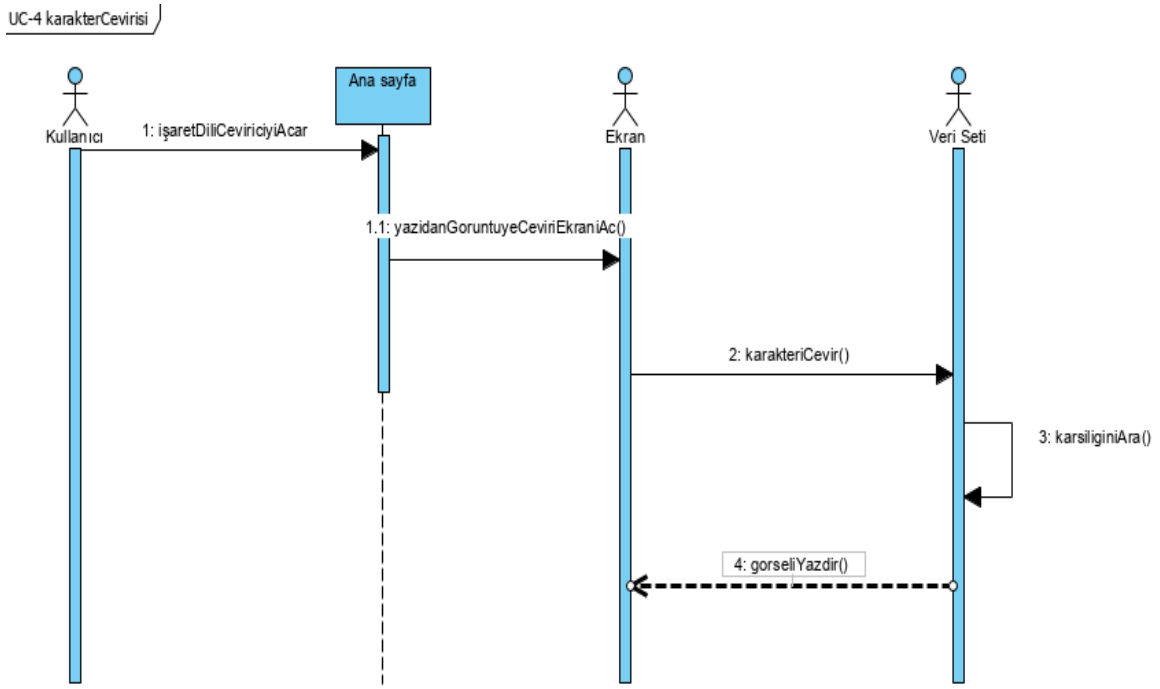
Şekil 4.4.2: yazidanGoruntuyeCeviriEkraniAcma Sistem Sequence Diyagramı

4.4.3 Use Case 3: canliCeviri



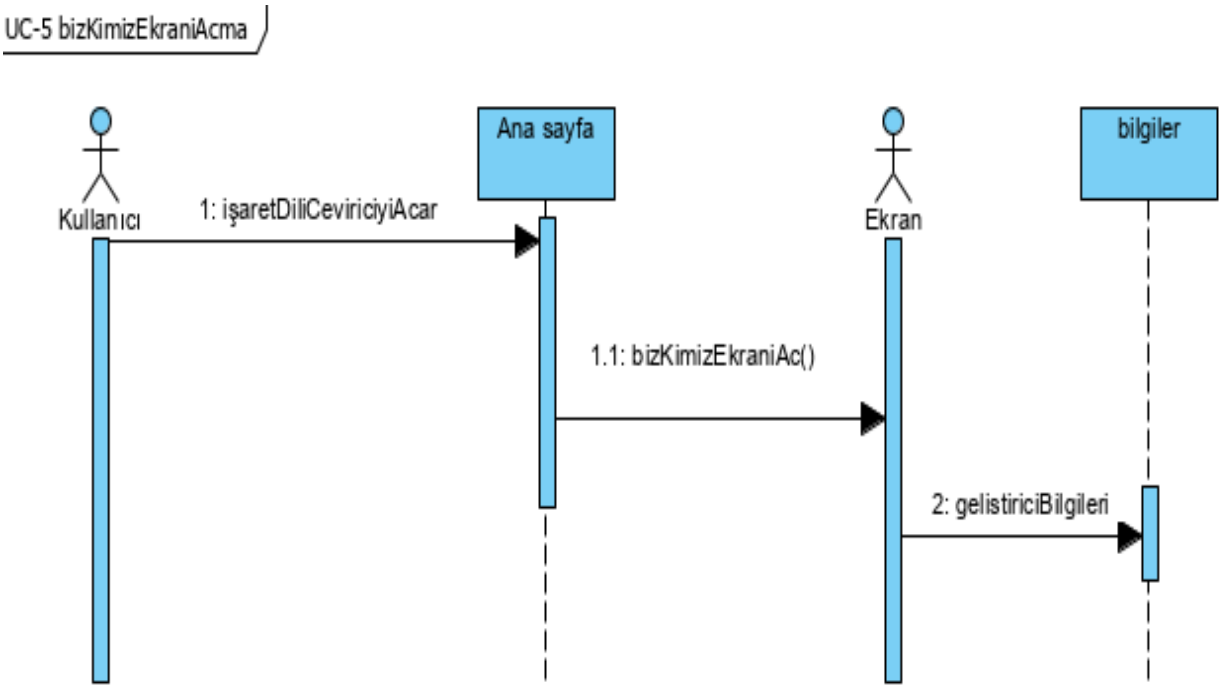
Şekil 4.4.3: canliCeviri Sistem Sequence Diyagramı

4.4.4 Use Case 4: karakterCevirisi



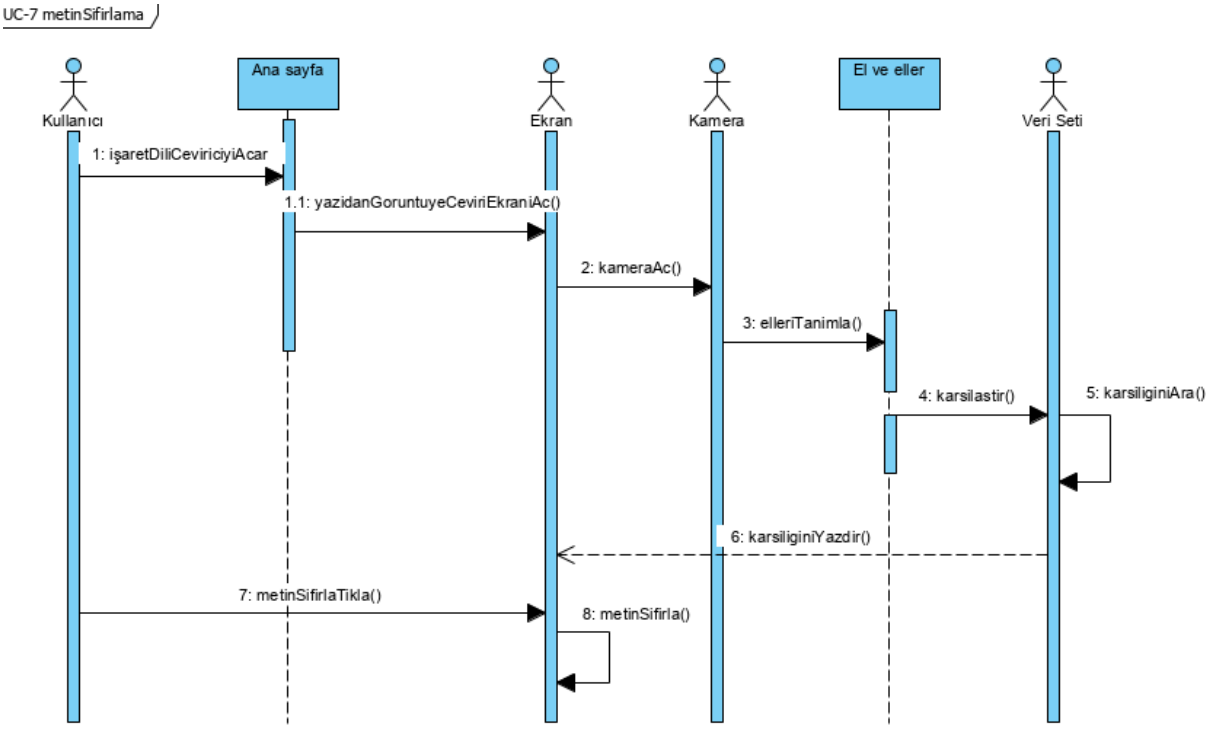
Şekil 4.4.4: karakterCevirisi Sistem Sequence Diyagramı

4.4.5 Use Case 5: bizKimizEkraniAcma



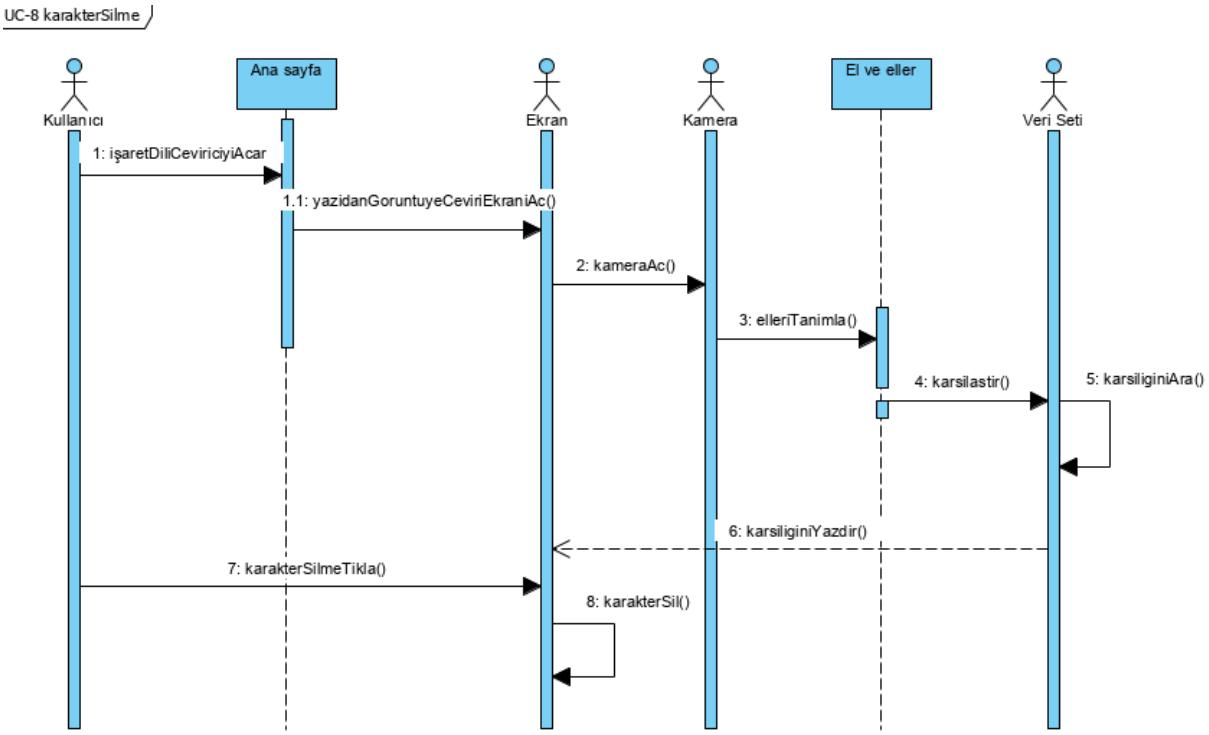
Şekil 4.4.5: bizKimizEkraniAcma Sistem Sequence Diyagramı

4.4.6 Use Case 6: metinSifirlama



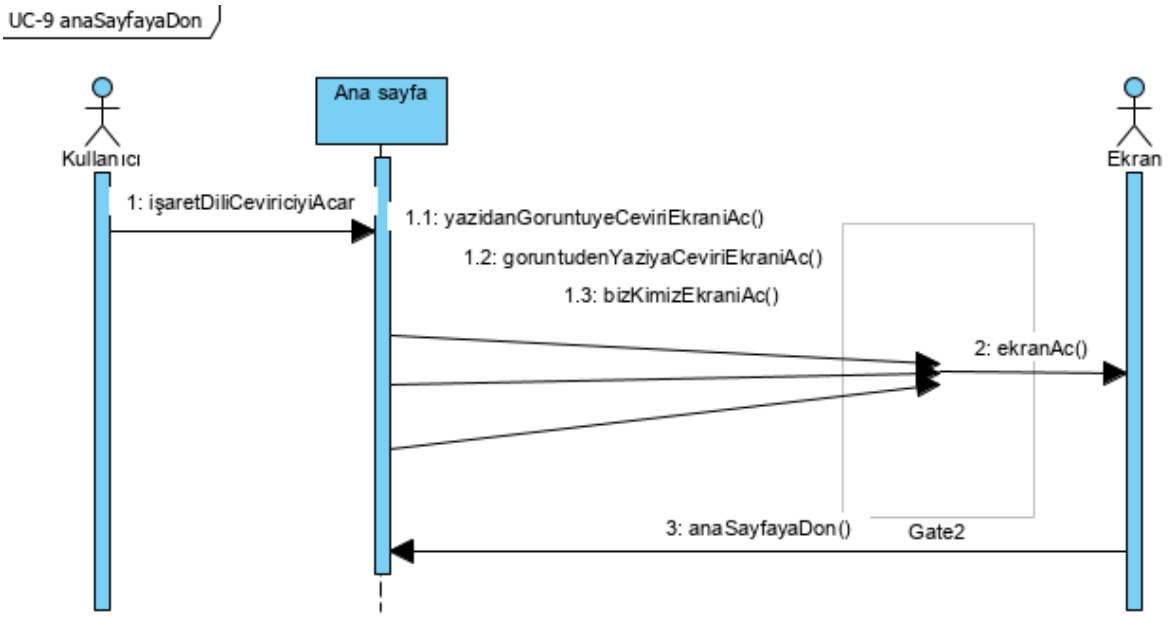
Şekil 4.4.6: metinSifirlama Sistem Sequence Diyagramı

4.4.7 Use Case 7: karakterSilme



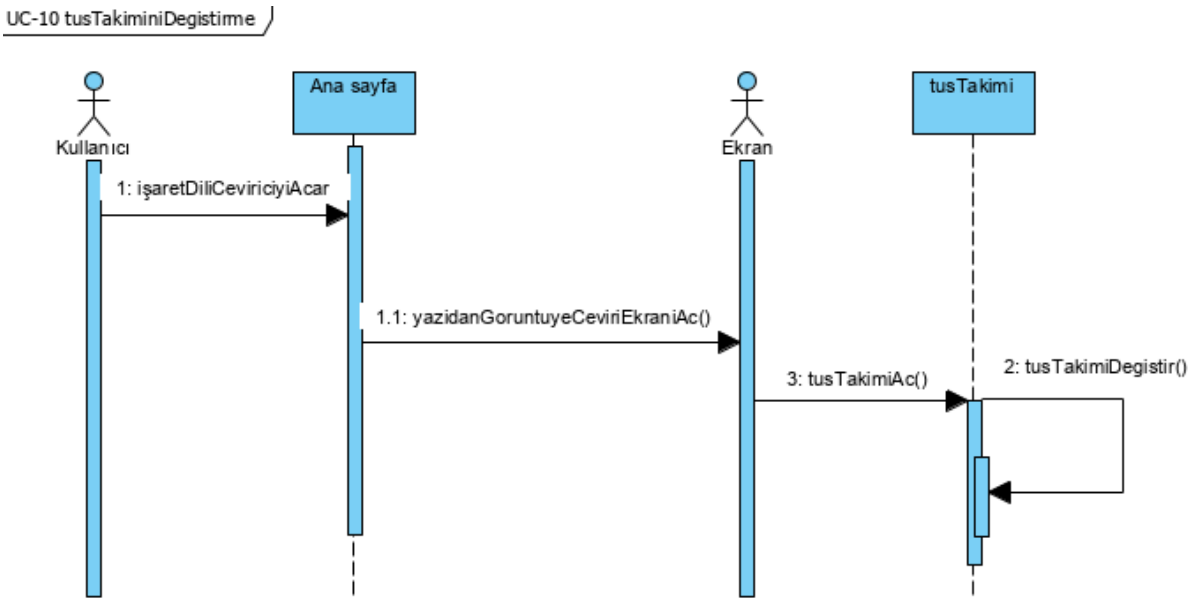
Şekil 4.4.7: karakterSilme Sistem Sequence Diyagramı

4.4.8 Use Case 8: anaSayfayaDon



Şekil 4.4.8: anaSayfayaDon Sistem Sequence Diyagramı

4.4.9 Use Case 9: tusTakiminiDegistirme



Şekil 4.4.9: tusTakiminiDegistirme Sistem Sequence Diyagramı

5. Model

5.1 Model Tanımı

Modelimiz Türkçe alfabenin işaret dilinin data setini içerdiği ve MobileNet modelini base alarak, yaklaşık 15 saat boyunca eğitildiği CNN (Konvolüsyonel Sinir Ağları) yapısında bir image classification (Resim Sınıflandırma) işlemi yapan deep learning(Derin öğrenme) gerçekleştirmiş bir transfer learning modelidir.

5.2 Dataset ve Model Olusturma

1-Dataset Oluşturma : Data seti oluşturma aşamaları
2-Data Augmentation : Data seti oluştururken resimleri çoğaltma aşaması
3-Model Oluşturma : Modelin özeliklerini ve eğitimini gerçekleştirme aşaması
4-Model Test Etme : Oluşan modelin çalıştırıp test etme aşaması
5-Model Dönüştürme : Modeli android için tflite modele dönüştürme aşaması

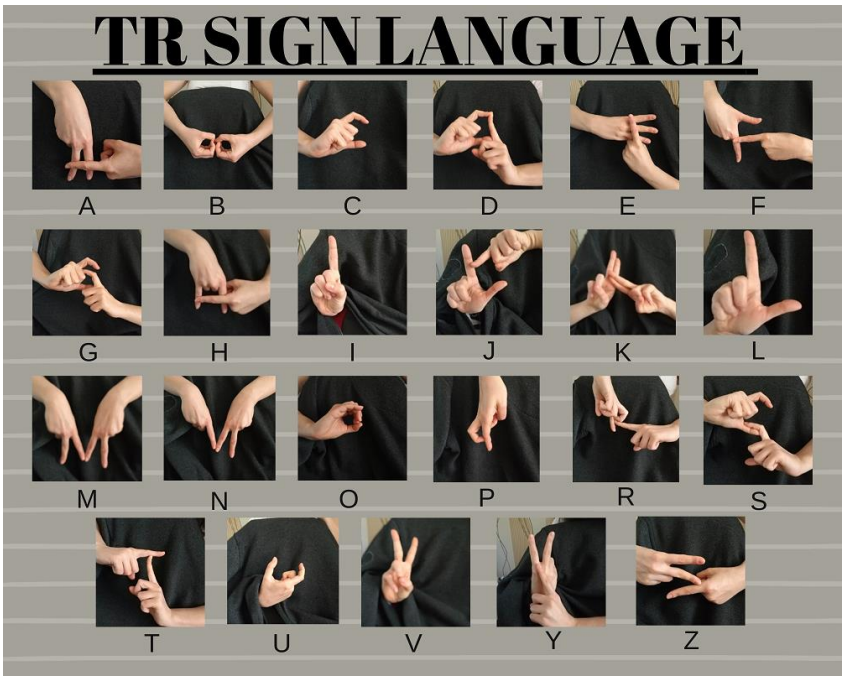
5.2.1 Dataset Oluşturma

Bir model eğitilmek için verilere ihtiyaç duyar. Bunlar eğitim ve test verileridir. Eğitim verileri ile belirlenen sayıdaki gruplar ile verileri parça parça yapısına alarak içerisinde sınıflandırma işlemi gerçekleştirir. Bu verilerin bulunduğu topluluğa data set denilir.

Kendi çektiğimiz ve internet üzerinden bulduğumuz Türkçe işaret dili alfabesi resimlerini öncelikle bilgisayar ortamında kayıt ettik. Ardından her bir resmi içerdiği harflere göre ayrı klasörlere koyduk. Klasörleme işleminde bittikten sonra PhotoScape programı yardımı ile resimlerin isimlerini içerdiği harfler olarak değiştirdik. İsimlendirme işlemi bittikten sonra PhotoScape programı ile resimleri 224x224 piksel boyutuna dönüştürdük. Her bir harf klasöründe 80 adet farklı şekilde çekilmiş ve isimlendirme ile boyutlandırma işlemlerinin gerçekleştiği resimlerimiz veri çoğaltma yani Data Augmentation işlemini yapmaya hazır hale geldi. Data augmentation ile her resimde yakınlaştırma işlemi, döndürme işlemi, parlaklık değiştirme işlemi gibi işlemler gerçekleştirerek 80 ana resimden 4800 farklı resim verisi elde ettik. Bu sayede toplamda 119.400 adet eğitim verimizin ve 47.796 adet test verimizin bulunduğu veri setimiz hazır hale geldi.

Hazırladığımız datasetimizi yapay zekâ, makine öğrenimi gibi alanlarda kullanılan diğer datasetlerin bulunduğu kaggle adlı siteye yükleyerek ihtiyacı olan insanların erişebilmesi için paylaştık.

Data setimize ulaşmak için: https://www.kaggle.com/berkaykocaoglu/tr-sign-language



Şekil 5.2.1: TR İşaret dili veri seti karaketerleri

5.2.2 Data Augmentation

Bir önceki başlıkta (Dataset Oluşturma) biraz bahsetmiştik. 80 ana resimden 4800 farklı resim verisi elde etmek için kullandığımızı belirtmiştik. Bu yöntem elde bulunan datasetteki verilerin az olduğundan ve bunun model eğitimi için bir sıkıntı olacağından verileri çoğaltmak için kullanılır. Bizim projemizde de resimleri tek bir şekilde tanıtmak yerine resimler üzerinde oynayarak onların farklı açılarda, mesafede, yönde çoğaltıp daha iyi bir veri seti ile daha iyi bir eğitim sağlamak için kullandık.

Bu işlemde ImageDataGenerator ile resimlerin hangi oranlarla dönüştürüleceğini belirledik. Oranlar sırasıyla;

-En fazla 40 derecelik açıda herhangi bir yöne döndürme
-Genişliği 1/5 oranla kaydırma
-Yüksekliği 1/5 oranla kaydırma
-Parlaklığı 0.2 ve 1.0 oranları arasında değiştirme
-Resim yakınlığını 0.5 ve 1.0 oranları arasında değiştirme
-Bir kenardan 1/5 oranla resmi kırpma
-Bu işlemler oluşa

n boşlukları en yakın olacak şekilde kapatma olarak gerçekleştirdik.

Bu işlemler her resim için uygulandı ama bir resim için tüm bu özelliklerinin hepsi uygulamadık. 1000 adet resimde yakınlaştırma işlemi varken parlaklık değiştirme işlemi varken diğerlerinde kullanılmayarak, verilerin farklı şekilde çoğalmalarını sağladık.

```
imgProperties = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        brightness_range=[0.2,1.0],
        zoom_range=[0.5,1.0],
        shear_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

Tablo 5.2.2: Data Augmentation İşlemleri

Aşağıdaki işlemde ise öncelikle çoğaltılacak resimlerinin klasör isimlerini bir listeye ekledik. Ardından bu liste ile döngüyü başlatıp, her bir resmin ImageDataGenerator metoduna girmesini sağladık. Metoda giren resimler işlem uygulandıktan sonra belirtilen yollara kaydettik.

```
def dataAugRun():
    letters = ['A','B','C','D','E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'R', 'S', 'T', 'U', 'V', 'Y', 'Z']
    for l in letters:
        for nmr in range(1,31):
            fileDir = 'tr_signLanguage_dataset/dataAugImg'
            lttr = l
            mainFileName = '{}.{}'.format(lttr,nmr)
            img = load_img('{}/{}/{}.jpg'.format(fileDir,lttr,mainFileName))
            x = img_to_array(img)
            x = x.reshape((1,) + x.shape)

            i = 0
            for batch in imgProperties.flow(x, batch_size=1,
                                    save_to_dir='tr_signLanguage_dataset/train/{}/'.format(lttr),
save_prefix='{}'.format(lttr), save_format='jpg'):
                i += 1
                if i > 100:
                    break  # otherwise the generator would loop indefinitely

            i = 0
            for batch in imgProperties.flow(x, batch_size=1,
                                    save_to_dir='tr_signLanguage_dataset/validation/{}/'.format(lttr),
save_prefix='{}'.format(lttr), save_format='jpg'):
                i += 1
                if i > 30:
                    break
```

Tablo 5.2.1.1.2: Data Augmentation Uygulama

5.2.3 Model Oluşturma

5.2.3.1 Dataseti Dahil Etmek

```
train_dir = os.path.join('tr_signLanguage_dataset/train')
validation_dir = os.path.join('tr_signLanguage_dataset/validation')
train_batches = ImageDataGenerator().flow_from_directory(train_dir,target_size=(img_width,img_height),
classes=['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','R','S','T','U','V','Y','Z','del','nothing','space'],batch_size=t
rainBatchSize)
valid_batches = ImageDataGenerator().flow_from_directory(validation_dir,target_size=(img_width,img_height),
classes=['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','R','S','T','U','V','Y','Z','del','nothing','space'],batch_size=
validBatchSize)
```

Tablo 5.2.3.1: Dataset Dahil Etme

Öncelikle daha önceden hazırladığımız datasetimizi sisteme dahil ettik. Train yani eğitim verilerimizi train_dir'e atadık ve test yani validation verilerimizi de validation_dir'e atadık. Ardından ImageDataGenerator resim seçme bunların ayarlarını yapma metodunda bulunan flow_from_directory ile de verilerimizi hangi sınıflardan(classes=) alacağını ve her seferde kaç adet alacağını(batch_size) ve boyutlarının(target_size) belirledik. Bunlardaki amaçlar ise;

-Bir model eğitilirken dosya boyutu 100,100- 128,128- 224,224 gibi farklı değerler alınabilir ama mobilenetin de kullandığı ve resimlerin daha iyi eğitilmesini sağlayan tavsiye edilen boyut 224,224 olduğu için biz de bu şekilde ayarladık.

-Bir modelde batch size her eğitim aşamasında bir arada alıp eğitilen veri sayısıdır. Batch size düşük olursa, eğitim hızlanır fakat yüksek hata oranları alınır. Eğer batch size yüksek olursa daha fazla veri bir arada alınacağı için daha iyi bir eğitim gerçekleşir, hata oranı azalır ama eğitim daha uzun sürede tamamlanır. Biz de bu yüzden batch size'ı 128 yapıp iyiye yakın bir eğitim gerçekleştirmeyi düşündük.

5.2.3.2MobileNet'i Dahil Etmek

```
base_model = MobileNet(
    weights='imagenet',
    include_top=False,
    input_shape=IMG_SHAPE,
    pooling='avg',
)
base_model.trainable = True
```

Tablo 5.2.3.2: MobileNet'i Dahil Etmek

Mobilenet Google tarafından geliştirilmiş, minimum bellek ve kaynak kullanmasına rağmen yüksek isabet oranına sahip bir sinir ağı mimarisi yani bir modeldir. Daha çok mobil ve gömülü sistemler için uygundur. Biz bu MobileNet'i projemize dahil ederek hem oluşturduğumuz modelin dosya boyutunun düşmesini hem android platformda çalışmasını hem de eğitim işleminin hızlanmasını sağladık.

5.2.3.3 Model Eğitimini Ayarlamak

```
model = Sequential()
model.add(base_model)
model.add(Dropout(0.25))
model.add(Dense(26, activation='softmax'))
model.summary()

stepsPerEpoch = numpy.ceil(train_batches.n/4000)
validationSteps= numpy.ceil(valid_batches.n/2000)
model.compile(Adam(lr=.0001), loss='categorical_crossentropy',metrics = ['accuracy'])
```

Tablo 5.2.3.3: Model Eğitimini Ayarlamak

Veri setimizi dahil edip ayarladıktan sonra ve MobileNet'i dahil ettikten sonra model eğitimine başlama aşamasına geldik. Aşamalar ise:

-Önce Sequential metodu ile modelimizin eğitilecek bir yapı olduğunu tanımladık.
-.add(base_model) ile MobileNet modelimizi kullanabilmek için kendi yapacağımız modelin içerisine ekledik.

-.add(Dropout(0.25) ile de her eğitim adımında 4 nörondan 1 nöronu kapatıp işlemi bu şekilde gerçekleştirmesini istedik. Dropout işlemi overfittingi yani modelin sürekli aynı yollarla eğitilmesinden kaynaklı hataların engellemesini sağlar. Optimum sonuçlar almak için kullanılan bir iyileştirme metotudur.
-.add(Dense(26, activation='softmax')) ile de önce 26 adet sınıfımız olduğu için 26 adet nöron oluşturduk. Daha sonra da activation fonksiyonu olarak da softmax'i tanımladık. Yapay sinir ağı modellerinin çıktı olarak verdiği skor değerler normalize edilmemiş değerlerdir. Aktivasyon fonksiyonu olan softmax bu değerleri normalize ederek olasılık değerlerine dönüştürmektedir. -.summary() ile yapımızı özetledik.
-stepsPerEpoch her eğitim adımında kaç adım olacağını belirledik ve validationSteps ise test atımlarını belirledik.
-son olarak da compile hata oranlarını azaltılmasının saplaması için optimizasyon fonksiyonu olan Adam fonksiyonunu, hatalı verilerin zamanla 0'a yaklaşması için cross entropy fonksiyonunu ekledik. metrics ile de oranları ekrana yazdırmayı ayarladık.

5.2.3.4 Checkpoint Noktaları Oluşturmak

```
checkDir = 'checkpoints'
checkpoint = ModelCheckpoint(filepath= os.path.join(checkDir,'model-{epoch:02d}.h5'))
```

Tablo 5.2.3.4: Checkpoint Ayarlamak

Model eğitim süresi veri sayısı yükseldikçe, batch size arttıkça, nöron sayısı arttıkça artmaktadır. Eğitim süresi günlerce sürebilmektedir. Biz de eğitim sırasında elektrik kesintisi, bilgisayar hatası gibi hatalardan dolayı eğitimimizin boşa gitmemesi için her adımda bir kayıt dosyası almak için Modelcheckpoint metodunu kullandık.

5.2.3.5 Eğitimi Başlatmak

```
history = model.fit_generator(train_batches, steps_per_epoch=10,
                validation_data= valid_batches, validation_steps=10, epochs=100,verbose=1,callbacks=[checkpoint])
```

Tablo 5.232.5: Eğitimi Başlatmak

Modelimizin içerisinde fit_generator ile eğitim verilerini test verileri atadık. Kaçar adım olacağını her adımda kaçar eder işlem yapacağını ve kayıt dosyası olarak hangi metodu kullanacağımızı belirledik ve eğitimi başlattık.

5.2.3.6 Eğitim Süreci

Model eğitimi yaklaşık 12 saatlik bir süreç sonucunda tamamlandı.
Loss, hata oranıdır. Accuracy ise başarı oranıdır. val_loss, test verilerinin hata oranıdır. val_accuracy ise test verilerinin başarı oranıdır.
1.adımda loss : 3.6690 - accuracy : 0.0953 - val_loss : 4.5646 - val_accuracy : 0.0391
100.adımda loss : 0.0082 - accuracy : 0.9992 - val_loss : 0.0076 - val_accuracy : 1.0000 olduğunu gözlemledik.

5.2.3.7 Model ve Labels Dosyalarını Kaydetmek

```
def createLabelsTXT():
    train_datagen = tf.keras.preprocessing.image.ImageDataGenerator()

    train_generator = train_datagen.flow_from_directory(directory=train_dir, target_size=(img_width, img_height),
batch_size=32)

    labels = '\n'.join(sorted(train_generator.class_indices.keys()))
    with open('completed_results/labels.txt', 'w') as f:
        f.write(labels)
```

Tablo 5.2.3.7: Labels.txt Kaydetmek

Labels.txt dosyası model ile birlikte kullanılan içerisinde modeldeki sınıfların isimlerinin bulunduğu bir metin belgesidir. Eğitim sonunda oluşturduk.

```
def modelSaver():
    keras.models.save_model(model,'completed_results/isaretDili.h5')
    #model.save('isaretDili.h5')
    print("Model is successfully stored!")
```

Tablo 5.2.3.7.2: Modeli Kaydetmek

Model eğitimi sonunda modelimizi keras modeli olarak .h5 uzantılı şekilde kayıt ettik.

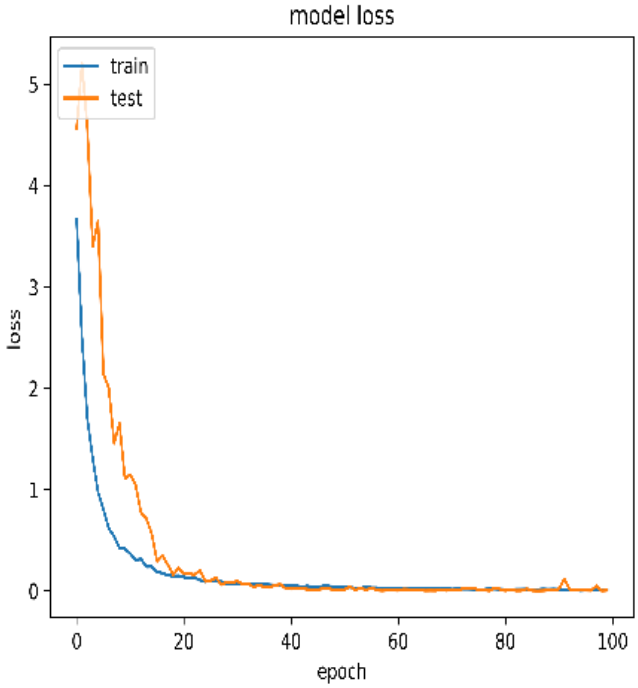
5.2.3.8 Eğitimdeki Hata ve Başarım Oranlarını Gözlemlemek

```
def createGraphic():
    print(history.history.keys())
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.savefig("completed_results/accuracy.png",dpi=720)
    plt.close()
    #plt.show()
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.savefig("completed_results/loss.png",dpi=720)
    #plt.show()
```

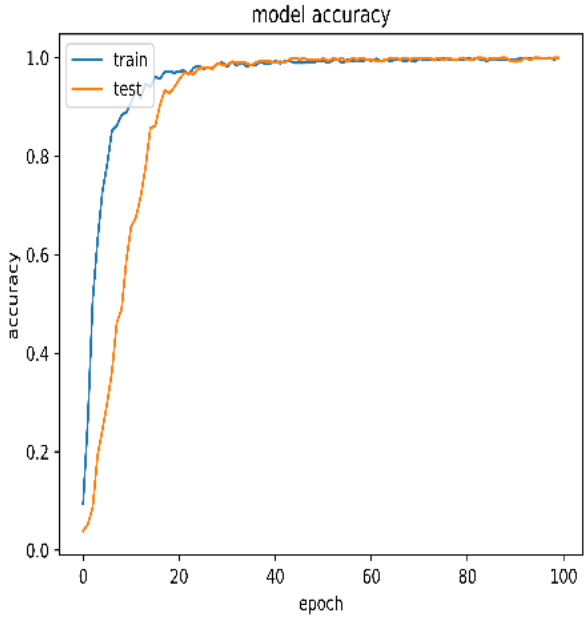
Tablo 5.2.3.8: Oran Gözlemlemek

Loss Grafiğinde modelin adım adım hata oranının azalışını görmekteyiz. Mavi olan kısımlar eğitimdeki hata oranı, turuncu kısımlar testteki hata oranlarıdır. Belirli bir süreden sonra hata oranının çok yavaş düştüğünü ve gittikçe 0'a yaklaştığını görmekteyiz.

Accuracy Grafiğinde modelin adım adım başarı oranının artışını görmekteyiz. Mavi olan kısımlar eğitimdeki başarı oranı, turuncu kısımlar testteki başarı oranlarıdır. Belirli bir süreden sonra başarı oranının çok yavaş arttığını ve gittikçe 1'a yaklaştığını görmekteyiz.



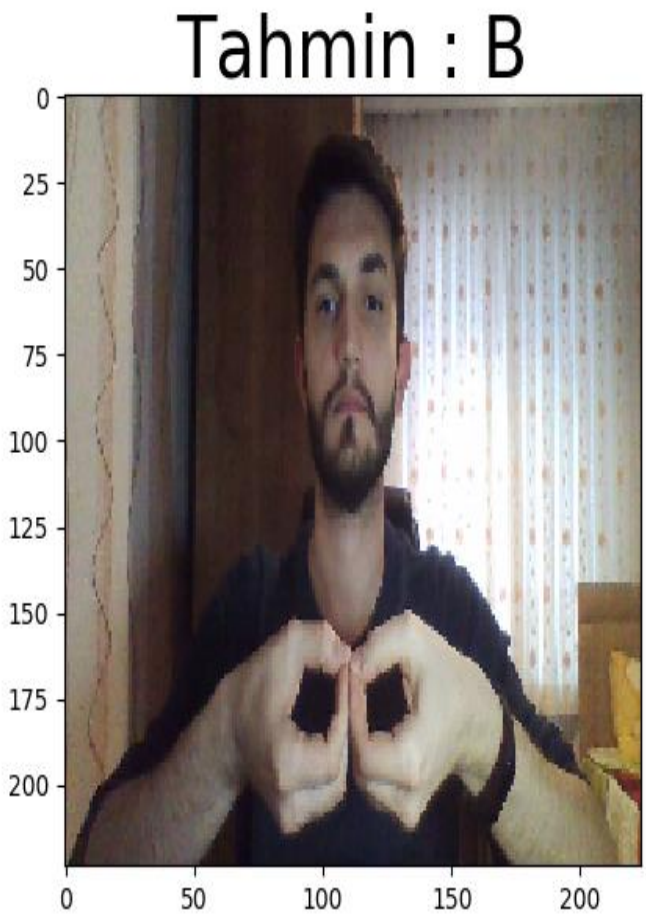
Şekil 5.2.3.8.1: Accuracy Tablosu



Şekil 5.2.3.8.2: Loss Tablosu

5.2.4 Model Test Etme

Eğittimiz modeli android ortamında çalıştırmadan yani model dönüştürme işlemi yapmadan önce PyCharm üzerinden test ettik. Seçtiğimiz bir resimleri sisteme dahil edip test edip sonuçları ise aşağıdaki gibi gözlemledik.



Şekil 5.2.4.1: Test Tablosu

Öncellikle load_model ile hazırladığımız modeli sisteme dahil ettik. Ardından compile ile bunu çalıştırdık. Test metodumuz ile seçtiğimiz resmi model.predict_classes ile tahmin etme işlemine soktuk. Çıkan tahmin sonucunun bulunduğu sınıfın yani harfin ismini alıp bunu if döngüsüne sokup matploblib kütüphanesi ile ettiği tahmini ve kullandığımız resmi ekranda gösterdik. Bu işlemin gerçekleştirildiği kodlar ise aşağıda bulunmaktadır.

```
model = load_model('completed_results/isaretDili.h5')
model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
def test(filePath):
    img = image.load_img(filePath, target_size=(img_width, img_height))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    images = np.vstack([x])
    classes = model.predict_classes(images, batch_size=128)
    if classes == 0:
        plt.title('Tahmin : A',fontsize=30)
    elif classes == 1:
        plt.title('Tahmin : B',fontsize=30)
    elif classes == 2:
        plt.title('Tahmin : C',fontsize=30)
    elif classes == 3:
        plt.title('Tahmin : D',fontsize=30)
.
.
.
    elif classes == 20:
        plt.title('Tahmin : V',fontsize=30)
    elif classes == 21:
        plt.title('Tahmin : Y',fontsize=30)
    elif classes == 22:
        plt.title('Tahmin : Z',fontsize=30)

    plt.imshow(img)
    plt.show()
```

Tablo 5.2.4.2: Test Kodları

5.2.5 Model Dönüştürme

Tensorflow ve keras yardımıyla eğittimiz modelimizi, TFLiteConverter.from_keras_model_file metodu ile programa dahil edip bunu bir değişkene atadık. Ardından bu değişken ile convert metodunu çağırarak modelimizi tflite modele çevirdik. Ardından bunu da dosya yazma işlemlerinde kullanılan metot ile kaydettik.

```
converter = tf.lite.TFLiteConverter.from_keras_model_file('completed_results/isaretDili.h5')
tflite_model = converter.convert()
file = open("completed_results/isaretDili.tflite", "wb")
file.write(tflite_model)
print("TfLite model is succesfully stored!")
```

Tablo 5.2.5: Model Dönüştürme

6. Android Algoritması

Bu bölümde android kısmının önemli kısımlarını inceleyeceğiz.

6.1 Kameradan Görüntü Alınması

Bu mobil uygulama kamera girişini CameraActivity.java dosyasında tanımlanan fonksiyonları kullanarak alır. Bu dosya kamera izinlerini ayarlamak için AndroidManifest.xml dosyasına bağlıdır.

CameraActivity ayrıca kullanıcı arayüzünden kullanıcı tercihlerini yakalayarak diğer sınıfların kullanımını sağlar.

```
device = Device.valueOf(deviceSpinner.getSelectedItem().toString());
numThreads = Integer.parseInt(threadsTextView.getText().toString().trim());
```

Tablo 6.1: Kameradan görüntü alınırken ki yapılan ayarlamalar

6.2 Sınıflandırıcı

Classifier.java dosyası, kamera girişini işlemek ve çıkarım yapmak için karmaşık mantığın çoğunu içerir.

Classifier.java sınıfının 'Quantized' modellerin kullanılmasını sağlayan ClassifierQuantizedMobileNet.java adında bir alt sınıfı mevcuttur.

Classifier sınıfı, statik bir yöntem olan "create" metodunu çalıştırarak Quantized türündeki modeli çağırır.

6.2.1 Model ve Yorumlayıcı

Çıkarım yapmak için bir model dosyası yüklememiz ve bir yorumlayıcı başlatmamız gerekir. Bu olay, Classifier sınıfının 'Constructor' metodunda gerçekleştirilir. Cihaz tipi ve iş parçacığı sayısı hakkındaki bilgiler, constructor metoduna aktarılan Interpreter.Options nesnesi aracılığıyla yorumlayıcıyı yapılandırmak için kullanılır.

Bir DSP (Dijital Sinyal İşlemcisi) veya NPU (Sinir İşleme Ünitesi) varsa, bu donanımdan tam olarak yararlanmak için bir Delegate kullanılabilir. Bu seçenekler daha fazla performans sağlayabilirken, aynı zamanda cihazın güç tüketimini attırmaktadır.

```
protected Classifier(Activity activity, Device device, int numThreads) throws
    IOException {
  tfliteModel = FileUtil.loadMappedFile(activity, getModelPath());
  switch (device) {
    case NNAPI:
      nnApiDelegate = new NnApiDelegate();
      tfliteOptions.addDelegate(nnApiDelegate);
      break;
    case CPU:
      break;
  }
  tfliteOptions.setNumThreads(numThreads);
  tflite = new Interpreter(tfliteModel, tfliteOptions);
  labels = FileUtil.loadLabels(activity, getLabelPath());
```

Tablo 6.2.1: Model ve Yorumlayıcı

Daha hızlı yükleme süreleri sunmak ve bellekteki kirli sayfaları azaltmak için model dosyasını önceden yüklemeyi ve bellek eşlemeyi gerçekleştirmek için FileUtil.loadMappedFile metotdu kullanılır. Bu yöntem modeli içeren bir "MappedByteBuffer" döndürür.

"MappedByteBuffer", bir "Interpreter.Options" nesnesi ile birlikte "Interpreter" yapıcısına iletilir. Bu nesne, yorumlayıcıyı yapılandırmak için kullanılabilir, örneğin iş parçacığı sayısını ayarlayarak (. setNumThreads (1)) veya [NNAPI] (.addDelegate (nnApiDelegate)) etkinleştirerek.

6.2.2 İşlem Öncesi Bitmap Görüntüsü

Classifier constructor ile giriş kamerasından bitmap görüntüsünü alıyoruz, verimli işleme için bir TensorImage formatına dönüştürüyor ve ön işlemler yapıyoruz. Adımlar 'private loadImage' metoodunda gösterilir:

```
/** Loads input image, and applys preprocessing. */
private TensorImage loadImage(final Bitmap bitmap, int sensorOrientation) {
  // Loads bitmap into a TensorImage.
  image.load(bitmap);

  // Creates processor for the TensorImage.
  int cropSize = Math.min(bitmap.getWidth(), bitmap.getHeight());
  int numRoration = sensorOrientation / 90;
  ImageProcessor imageProcessor =
      new ImageProcessor.Builder()
          .add(new ResizeWithCropOrPadOp(cropSize, cropSize))
          .add(new ResizeOp(imageSizeX, imageSizeY, ResizeMethod.BILINEAR))
          .add(new Rot90Op(numRoration))
          .add(getPreprocessNormalizeOp())
          .build();
  return imageProcessor.process(inputImageBuffer);
}
```

Tablo 6.2.2: Bitmap Görüntüsü Alma

6.2.3 Çıktı Nesnesi

Modelin çıkışı için TensorBufferçıkışının başlatılması gerekmektedir. Bu işlem:

```
/** Output probability TensorBuffer. */
private final TensorBuffer outputProbabilityBuffer;

//...
// Get the array size for the output buffer from the TensorFlow Lite model file
int probabilityTensorIndex = 0;
int[] probabilityShape =
    tflite.getOutputTensor(probabilityTensorIndex).shape(); // {1, 1001}
DataType probabilityDataType =
    tflite.getOutputTensor(probabilityTensorIndex).dataType();

// Creates the output tensor and its processor.
outputProbabilityBuffer =
    TensorBuffer.createFixedSize(probabilityShape, probabilityDataType);

// Creates the post processor for the output probability.
probabilityProcessor =
    new TensorProcessor.Builder().add(getPostprocessNormalizeOp()).build();
```

Tablo 6.2.3: Çıktı Nesnesi Oluşturma

Nicemlenmiş modeller için, tahminleri NormalizeOp ile nicelleştirmemiz gerekir. Daha spesifik olmak gerekirse,

ClassifierQuantizedMobileNet, içinde normalleştirilmiş parametreler şu şekilde tanımlanır:

```
private static final float PROBABILITY_MEAN = 0.0f;
private static final float PROBABILITY_STD = 255.0f;
```

Tablo 6.5.2: Çıktı nesnesinin ayarları

6.2.4 Sonuç Çıkarımlama

Çıkarım, Classifier sınıfında aşağıdakiler kullanılarak gerçekleştirilir:

```
tflite.run(inputImageBuffer.getBuffer(),
    outputProbabilityBuffer.getBuffer().rewind());
```

Tablo 6.2.4: Sonuç çıkarımlama

6.2.5 Görüntü tanımlama

Doğrudan run çağırmak yerine, recognizeImage metotu kullanılır. Bir bitmap ve sensör yönelimini kabul eder, çıkarım yapar ve her biri bir etikete karşılık gelen sıralı Recognition örnekleri listesini döndürür. Yöntem, MAX_RESULTS ile sınırlı bir dizi sonuç döndürür, varsayılan olarak 1 değerindedir.

Recognition, belirli bir tanıma sonucu hakkında bilgi içeren basit bir sınıftır, buna title and confidence bilgileride dahildir. Tanımlama işlemi sonrasında değerler 0 ila 1 arasında bir değer alır. Daha sonra bu değerlere sabit olan nesneler sıralanır.

```
/** Gets the label to probability map. */
Map<String, Float> labeledProbability =
    new TensorLabel(labels,
        probabilityProcessor.process(outputProbabilityBuffer))
        .getMapWithFloatValue();
```

Tablo 6.2.5.1: Görüntü tanımlama

Sıralama için bir PriorityQueue metotu kullanılır.

```
/** Gets the top-k results. */
private static List<Recognition> getTopKProbability(
    Map<String, Float> labelProb) {
  // Find the best classifications.
  PriorityQueue<Recognition> pq =
      new PriorityQueue<>(
          MAX_RESULTS,
          new Comparator<Recognition>() {
            @Override
            public int compare(Recognition lhs, Recognition rhs) {
              // Intentionally reversed to put high confidence at the head of
              // the queue.
              return Float.compare(rhs.getConfidence(), lhs.getConfidence());
            }
          });

  for (Map.Entry<String, Float> entry : labelProb.entrySet()) {
    pq.add(new Recognition("" + entry.getKey(), entry.getKey(),
             entry.getValue(), null));
  }

  final ArrayList<Recognition> recognitions = new ArrayList<>();
  int recognitionsSize = Math.min(pq.size(), MAX_RESULTS);
  for (int i = 0; i < recognitionsSize; ++i) {
    recognitions.add(pq.poll());
  }
  return recognitions;
}
```

Tablo 6.2.5.2: Tanımlanan görüntülerin sıralanması

6.3 Sonuçların Görüntülenmesi

Classifier sınıfı çağrılır ve sonuçlar çıkartılır. Çıkarılan sonuçlar ClassifierActivity.java içindeki processImage() fonksiyonu tarafından görüntülenir.

ClassifierActivity, kamera görüntüsünü oluşturan, sınıflandırmayı çalıştıran ve sonuçları görüntüleyen yöntem uygulamalarını içeren CameraActivity'nin bir alt sınıfıdır. ProcessImage() fonksiyonu, iş parçacığında yapılan sınıflandırmayı olabildiğince hızlı çalıştırır, çıkarımın engellenmesini ve gecikmenin oluşmasını önlemek için UI iş parçacığında bilgi oluşturur.

```
@Override
protected void processImage() {
 rgbFrameBitmap.setPixels(getRgbBytes(), 0, previewWidth, 0, 0, previewWidth,
    previewHeight);
 final int imageSizeX = classifier.getImageSizeX();
 final int imageSizeY = classifier.getImageSizeY();

 runInBackground(
    new Runnable() {
      @Override
      public void run() {
        if (classifier != null) {
          final long startTime = SystemClock.uptimeMillis();
          final List<Classifier.Recognition> results =
              classifier.recognizeImage(rgbFrameBitmap, sensorOrientation);
          lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;
          LOGGER.v("Detect: %s", results);

          runOnUiThread(
              new Runnable() {
                @Override
                public void run() {
                  showResultsInBottomSheet(results);
                  showInference(lastProcessingTimeMs + "ms");
                }
              });
        }
        readyForNextImage();
      }
    });
}
```

Tablo 6.3.1: Sonuçların Görüntülenmesi

ClassifierActivity nin bir diğer önemli rolü, kullanıcı tercihlerini belirlemek (CameraActivity yi sorgulayarak) ve uygun şekilde yapılandırılmış Classifier alt sınıfını başlatmaktır. Bu, video feed'i başladığında (onPreviewSizeChosen() aracılığıyla) ve kullanıcı arayüzünde seçenekler değiştirildiğinde (onInferenceConfigurationChanged() aracılığıyla) gerçekleşir.

```
private void recreateClassifier(Model model, Device device, int numThreads) {
 if (classifier != null) {
  LOGGER.d("Closing classifier.");
  classifier.close();
  classifier = null;
 }
 try {
  LOGGER.d(
      "Creating classifier (model=%s, device=%s, numThreads=%d)", model,
      device, numThreads);
  classifier = Classifier.create(this, model, device, numThreads);
 } catch (IOException e) {
  LOGGER.e(e, "Failed to create classifier.");
 }
}
```

Tablo 6.3.2: Sonuçların Kullanıcının tercihlerine göre listelenmesi

6.4 Sonuçlar ile metin oluşturulması

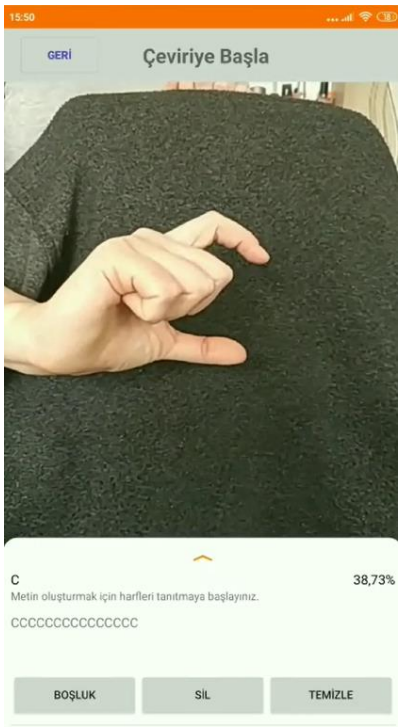
CameraActivity sınıfı içerisinde, bulunan showResultsInBottomSheetmetotdu içerisinde sonuçların kaydedildiği listenin verileri okunularak, gelen verinin türüne göre işlem yapılabilmektedir.

Gelen veriler metnin oluşturulacağı texview alanına yazılmaktadır. Burada oluşabilicek küçük algılama hatalarının giderilmesi ve sistemin daha kararlı çalışcabilmesi için gelen verilerin doğrulanmasını sağladık.

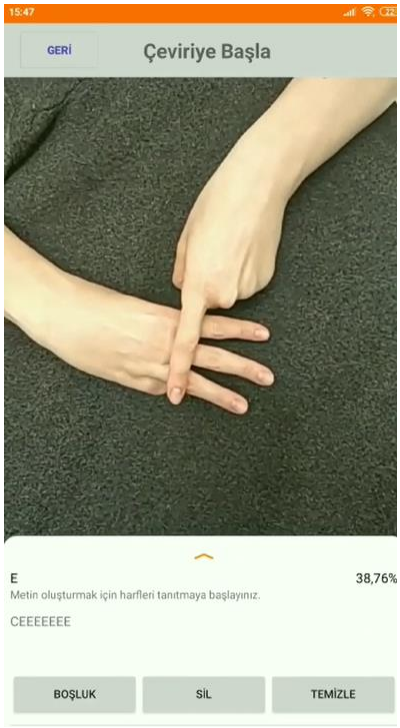
```
protected void showResultsInBottomSheet(List<Recognition> results) {
  if (results != null && results.size() >= 1) {
    Recognition recognition = results.get(0);
    if (recognition != null) {
      if (recognition.getTitle() != null){
        recognitionTextView.setText(recognition.getTitle());

        if (recognition.getTitle().equals("space")){
          recognitionTextView.setText("Boşluk Bırakıldı.");
          composed_text.setText(composed_text.getText() + " ");
        }
        else if (recognition.getTitle().equals("del")){
          recognitionTextView.setText("Harf Silindi.");
          button_delete.callOnClick();
        }
        else if (recognition.getTitle().equals("nothing")){
          recognitionTextView.setText("Harf Bulunamadı.");
          // do nothing
        }
        else {
          if (recognition.getTitle() == active_latter) {
            active_latter_number += 1;
            if (active_latter_number >= 5) {
              composed_text.setText(composed_text.getText() + active_latter);
            } else {
              //do nothing
            }
          } else {
            active_latter = recognition.getTitle();
            active_latter_number = 1;
          }
        }
      }
      if (recognition.getConfidence() != null)
        recognitionValueTextView.setText(String.format("%.2f", (10000 * recognition.getConfidence()))
+ "%");
    }
  }
 }
```

Tablo 6.4: Sonuçlar ile bir metin oluşturulması

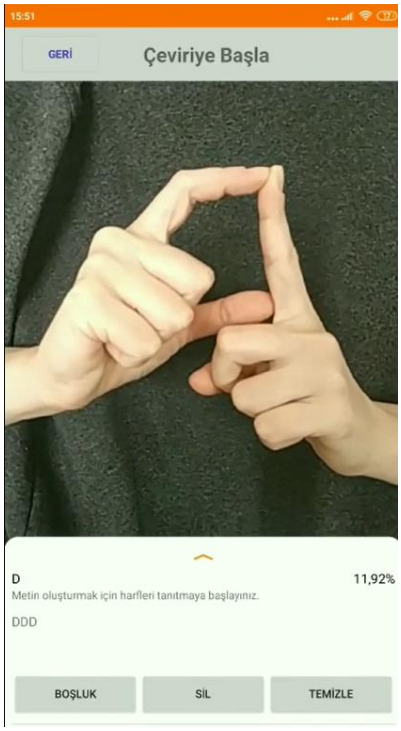
6.5 Uygulamanın Canlıdaki Ekran Görüntüleri



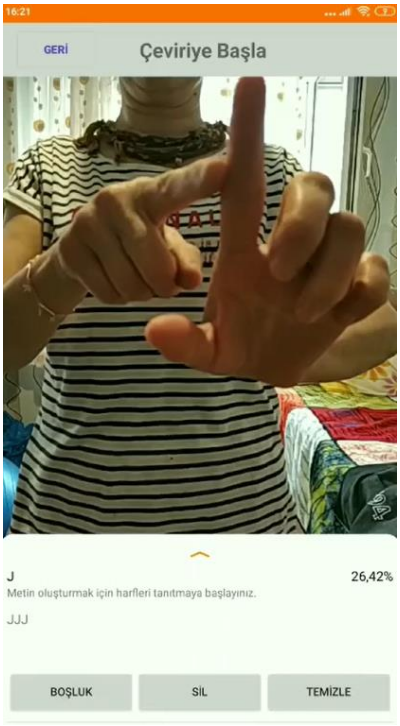
Şekil 6.5.1: C



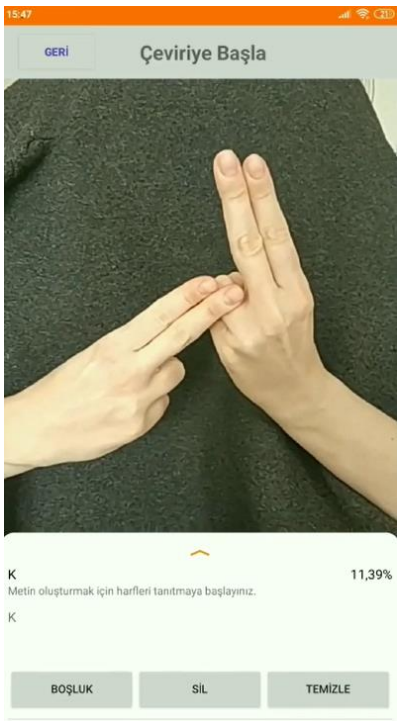
Şekil 6.5.2: E



Şekil 6.5.3: D



Şekil 6.5.4: J



Şekil 6.5.5: K

Şekil 6.5.6: A

7. Raporları Karşılaştırma

Gereksinimler	Gerçekleştirdik	Gerçekleştirmedik
Kullanıcı, isteğe bağlı olarak kameradan yazıya çeviri, yazıdan görsele çeviri veya hakkımızda butonlarını seçebilmelidir.	+	
Sistem, kameradan yazıya çeviri ekranına geçildiğinde kameranın doğrultulduğu kişinin el veya ellerini algılamalıdır.	+	
Sistem, el veya ellerin hareketleri ile oluşturulan işaretlerin karşılığını veri seti içerisinde aramalıdır.	+	
Kullanıcı, görüntüden yazıya çeviri ekranında algılattığı el veya ellerin hareketlerinin anlamlarını, ekranın alt kısmında metin şeklinde görebilmelidir.	+	
Sistem, yazıdan görüntüye çeviri ekranına geçildiğinde tıklatılan butonun içeriğindeki kelime, harf veya rakam ifadesinin işaret dili karşılığını ekranda görüntüleyebilmelidir.	Kelime, harf veya rakam ifadelerinin işaret dili karşılığı görüntülenebilmektedir.	Ekranın adını sözlük olarak değiştirdik.
Kullanıcı, program geliştiricileri merak ettiğinde biz kimiz? ekranına girip öğrenebilmelidir.	+	
Kullanıcı, görüntüden yazıya çeviri ekranında, çevrilen metin alanını sıfırlayabilmelidir.	+	
Kullanıcı, görüntüden yazıya çeviri ekranında, hatalı bir çeviri durumda metnin son yazılan karakterini silebilmelidir.	+	
Kullanıcı, girdiği bir ekranda başlık kısmının yanında bulunan geri tuşu ile ana sayfaya geri dönebilmelidir.	+	
Kullanıcı, yazıdan görüntüye çeviri ekranında, alt tarafta bulunan tuş takımını sağ ve sol eksenli el hareketleri ile değiştirebilmelidir.	Tuş takımını sağ ve sol eksenli el hareketleri ile değiştirebilmelidir.	Ekranın adını sözlük olarak değiştirdik.

Sistem, model ile iletişime geçip kameradan alınan görüntü üzerinden bir çıkarımda bulunmalıdır.	+	
Sistem tüm kullanıcılar için açık olmalıdır. Üyelik gerektirmemelidir	+	
Kamera el veya el hareketlerini algılamalıdır.	+	
Sistem algıladığı el işaretlerini, algıladığı sıra ile yazdırmalıdır.	+	
Veri seti içerisinde harf, rakam için veri bulunmalıdır.	+	
Sistem dili Türkçe olmalıdır.	+	
Biz kimiz? kısmı bulunmalı ve geliştirici bilgileri içerisinde yazmalıdır.	+	
Kullanıcı, yazıdan görüntüye çeviri ekranında görüntülediği işaretleri iki parmak hareketi ile yakınlaştırarak veya uzaklaştırarak detaylı görebilmelidir.	Bu özellik yerine Gif dosyaları ekleyerek işaretlerin daha anlaşılabilir olmasını sağladık.	Ekranın adını sözlük olarak değiştirdik. Her harf ve rakam için gif eklediğimizden yakınlaştırma işlemini eklemedik.
Kullanıcı, görüntüden yazıya çeviri ekranında çevrilen metni isteğe bağlı olarak dinleyebilmelidir.	Bu özellik yerine Gif dosyaları ekleyerek işaretlerin daha anlaşılabilir olmasını sağladık.	Ekranın adını sözlük olarak değiştirdik. Dinleme özelliğini eklemedik.
Kullanıcı, yazıdan görüntüye çeviri ekranında, görüntüye çift tıklayarak yakınlaştırma işlemi gerçekleştirdikten sonra yine çift tıklayarak uzaklaştırma işlemi gerçekleştirebilmelidir.	Bu özellik yerine Gif dosyaları ekleyerek işaretlerin daha anlaşılabilir olmasını sağladık.	Ekranın adını sözlük olarak değiştirdik. Her harf ve rakam için gif eklediğimizden yakınlaştırma işlemini eklemedik.

8. Kaynakça

Rapor Düzeni ve Bilgisi;

- https://www.ece.rutgers.edu/~marsic/Teaching/SE1/syllabus.html
- https://www.ece.rutgers.edu/~marsic/books/SE/projects/ViBE/2018-g10-report3.pdf

Sorumluluk Matrisi;

- https://medium.com/@hakirac/raci-matriksi-sorumluluk-matriksi-45cb19048c08
- https://www.endustrimuhendisligim.com/raci-matrisi/

Kullanıcı Arayüzü Gereksinimleri;

- https://medium.com/turkce/muhtesem-bir-kullan-c-arayuzu-tasarlaman-n-7-kural-bolum-1-13d20e5edb6
- https://journo.com.tr/6-adimda-mobil-uygulama-arayuzu-tasarimi
- https://www.hokkaweb.com/blog/6-adimda-mobil-uygulama-arayuzu-tasarimi/
- https://forum.java.com.tr/yazilim-arayuz-tasarimi/
- https://www.mobil13.com/mobil-kullanici-mobil-ui-arayuzu-nedir-19746.html

Android TensorFlow Lite;

- https://www.tensorflow.org/lite
- https://www.tensorflow.org/lite/guide
- https://www.tensorflow.org/lite/models
- https://www.tensorflow.org/lite/examples
- https://github.com/tensorflow/examples

Sistem Sequence Diyagramları;

- http://www.csharpnedir.com/articles/read/?id=402
- https://www.slideshare.net/aselmanb/uml-ile-modelleme

Dataset İşlemleri;

- https://www.kaggle.com/general/51785
- http://www.photoscape.org/ps/main/index.php

Model İşlemleri;

- https://www.tensorflow.org/tutorials/images/classification
- https://blog.francium.tech/build-your-own-image-classifier-with-tensorflow-and-keras-dc147a15e38e
- https://www.tensorflow.org/lite/convert/cmdline
- https://www.tensorflow.org/lite/convert
- https://www.tensorflow.org/api_docs/python/tf/lite/TFLiteConverter?version=nightly
- https://medium.com/@konstantin.poklonskiy/image-recognition-with-mobilenet-and-ml-net-e6d8e8323127
- https://blog.tensorflow.org/2018/03/using-tensorflow-lite-on-android.html
- https://androidkt.com/image-classify-tensorflow-lite/