ELLE ÇİZİLMİŞ TASLAK ÇİZİMLERDE KULLANICI ARABİRİMİ ÖGELERİNİN DERİN ÖRNEK SEGMENTASYONU

Cahit Berkay KAZANGİRLER

2021
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

Tez Danışmanı
Dr. Öğr. Üyesi Caner ÖZCAN

ELLE ÇİZİLMİŞ TASLAK ÇİZİMLERDE KULLANICI ARABİRİMİ ÖGELERİNİN DERİN ÖRNEK SEGMENTASYONU

Cahit Berkay KAZANGİRLER

T.C.
Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalında
Yüksek Lisans Tezi
Olarak Hazırlanmıştır

Tez Danışmanı
Dr. Öğr. Üyesi Caner ÖZCAN

KARABÜK
Eylül 2021

Cahit Berkay KAZANGİRLER tarafından hazırlanan “ELLE ÇİZİLMİŞ TASLAK ÇİZİMLERDE KULLANICI ARABİRİMİ ÖGELERİNİN DERİN ÖRNEK SEGMENTASYONU” başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Dr. Öğr. Üyesi Caner ÖZCAN .....
Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından Oy Birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 15/09/2021

Ünvanı, Adı SOYADI (Kurumu) İmzası

Başkan : Dr. Öğr. Üyesi Emrullah SONUÇ (KBÜ) .....

Üye : Dr. Öğr. Üyesi Caner ÖZCAN (KBÜ) .....

Üye : Dr. Öğr. Üyesi Rafet DURGUT (BANÜ) .....

KBÜ Lisansüstü Eğitim Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Prof. Dr. Hasan SOLMAZ .....
Lisansüstü Eğitim Enstitüsü Müdürü

“Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim.”

Cahit Berkay KAZANGİRLER

ÖZET

Yüksek Lisans Tezi

ELLE ÇİZİLMİŞ TASLAK ÇİZİMLERDE KULLANICI ARABİRİMİ ÖGELERİNİN DERİN ÖRNEK SEGMENTASYONU

Cahit Berkay KAZANGİRLER

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr. Üyesi Caner ÖZCAN
Eylül 2021, 73 sayfa

Kullanıcı arabirimi (UI), insanların bir makine, cihaz, bilgisayar programı ya da karmaşık aletlerle etkileşimini sağlayan yöntemlerin bileşkesine verilen addır. UI prototipleme, uygulama geliştirmenin ilk aşamalarında gerekli bir adımdır. Grafik kullanıcı arabiriminin taslak çizimlerini, kodlanmış bir UI uygulamasına dönüştürmek zaman alıcı bir görevdir. UI tasarımlarının basit bir şekilde uygulanması için insan çabası yerini alabilecek otomatik bir sistem, bu prosedürü büyük ölçüde hızlandıracaktır. Bu çalışmada, Mask Region-based Convolutional Neural Network (Mask R-CNN) ile elle çizilmiş UI ögelerinin otomatik tespiti ve segmentasyonu açıklanmaktadır. Mask R-CNN, bir görüntüdeki nesneleri verimli bir şekilde algılarken aynı zamanda her bir örnek için yüksek kaliteli bir segmentasyon maskesi oluşturur. Buna ek olarak, nesne olmayan bölgeler arka plan olarak nitelendirilmektedir. Veri kümesinde 370 eğitim görüntüsü ve eğitim kümesinde

bulunmayan 87 test görüntüsü olmak üzere toplam 457 adet veri yer almaktadır. Elde edilen taslak çizimlerde eğitim verileri için 3315 UI ögesi bulunurken test görüntülerinde ise 941 adet UI ögesi bulunmaktadır. Görüntüler, derin sinir ağına verilmeden önce UI ögelerinin tespit işleminin kolaylaşması için ön işleme aşamasından geçirilmiştir. Performans sonucunun doğruluğu için test görüntüleri üzerinde Microsoft Common Objects in Context (MS COCO) doğruluk metriklerinden hassasiyet (precision) ve ortalama genel hassasiyet olan Mean Average Precision (mAP) değeri ölçülmüştür. Kademeli olarak iterasyon sayısı artırımı ile 300 dönem sayısına kadar transfer öğrenme stratejisi kullanılarak eğitim yapılmıştır. Sonuç olarak en iyi çıktı için 300 iterasyon sayısı ile precision değeri %93,75'e ulaşırken mAP değeri ise %98,48'e ulaşmıştır.

Anahtar Sözcükler : Taslak çizim, kullanıcı arabirimi, görüntü segmentasyonu, yapay sinir ağları, evrişimli sinir ağları, nesne tespiti, görüntü işleme, derin öğrenme, derin örnek segmentasyon ve Mask R-CNN.

Bilim Kodu : 92414.

ABSTRACT

M. Sc. Thesis

DEEP INSTANCE SEGMENTATION OF USER INTERFACE ELEMENTS IN HAND-DRAWN WIREFRAMES

Cahit Berkay KAZANGİRLER

Karabük University
Institute of Graduate Programs
Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Caner OZCAN
September 2021, 73 pages

User interface (UI) is the combination of methods that enable people to interact with a machine, device, computer program, or complex tools. UI prototyping is a necessary step in the early stages of application development. Converting sketches of the graphical user interface into a coded UI application is a time-consuming task. An automated system that can replace human effort for simple implementation of UI designs will greatly speed up this procedure. This paper describes the automatic detection and segmentation of hand drawn UI elements with Mask Region-based Convolutional Neural Network (Mask R-CNN). Mask R-CNN efficiently detects objects in an image while also creating a high-quality segmentation mask for each sample. In addition, non-object regions qualify as background. The data set contains a total of 457 data, including 370 training images and 87 test images not included in the training set. While there are 3315 UI elements for training data in the draft drawings obtained, there are 941 UI elements in the test images. The images were pre-processed

by taking the facilitate the detection of UI elements before being delivered to the deep neural network. For the accuracy of the performance result, precision from Microsoft Common Objects in Context (MS COCO) accuracy metrics and Mean Average Precision (mAP) value, which is the general average precision, were measured on the test images. With the gradual increase in the number of iterations, training was carried out by using the transfer learning strategy up to 300 epochs. As a result, the precision value reached 93.75% with 300 epochs for the best output, while the mAP value reached 98.48%.

Key Words : Wireframe, user interface, image segmentation, artificial neural networks, convolutional neural networks, object detection, image processing, deep learning, deep instance segmentation and Mask R-CNN.

Science Code : 92414.

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren, bu satırlarla ifade edilemeyecek saygıyı hak eden sayın hocam, Dr. Öğr. Üyesi Caner ÖZCAN'a sonsuz teşekkürlerimi sunarım.

Çalışmalarım süresince benden desteğini esirgemeyen meslektaşım Buse Yaren TEKİN ve Muhammet DİLMAÇ'a teşekkürlerimi borç bilirim. Ek olarak, tez çalışmamızı "FYL-2020-2156" proje numarası ile desteklemeye layık gören Karabük Üniversitesi Bilimsel Araştırma Projeleri Birimi'ne teşekkürlerimi sunarım.

Sevgili aileme manevi hiçbir yardımı esirgemeden yanımda oldukları için tüm kalbimle teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

Sayfa

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

SİMGELER

dst : girilen veri değerine ait mesafe vektörü

$\mathcal{L}$: kayıp fonksiyon değeri

$\mathcal{L}_{cls}$: sınıf kayıp değeri

$\mathcal{L}_{box}$: sınıflandırmadan alınan sınırlayıcı kutucuk değeri

$\mathcal{L}_{mask}$: segmentasyondan alınan maske değeri

k : toplam sınıf sayısı

m : ilgi bölgesi boyutu

$\sum k$: k değeri için toplam sembolü

GTP : gerçek referans görseline ait toplam pozitif temel sayısı

KISALTMALAR

API : Application Programming Interface (Uygulama Programlama Arayüzü)

CNN : Convolutional Neural Network (Evrişimli Sinir Ağları)

COCO : Common Objects In Context (Bağlamdaki Ortak Nesneler)

CSS : Cascading Style Sheets (Basamaklanmış Stil Katmanları)

CUDA : Compute Unified Device Architecture (Birleşik Aygıt Mimarisini Hesaplama)

CUDNN : CUDA Deep Neural Network Library (CUDA Derin Sinir Ağları Kütüphanesi)

DevOps : Development ve Operations (Geliştirme ve Operasyon)

DOC : Document (Doküman)

DSL : Domain Specific Languages (Etki Alanına Özgü Dil)

FN : False Negative (Yanlış Negatif)

FP : False Positive (Yanlış Pozitif)

FPN : Feature Pyramid Network (Özellik Piramit Ağı)

GPU : Graphics Processing Unit (Grafik İşlem Birimi)

GUI : Graphical User Interface (Grafik Kullanıcı Arabirimi)

H1 : Heading 1 (Başlık 1)

H6 : Heading 6 (Başlık 6)

HI-FI : High Fidelity (Yüksek kaliteli)

HTML : Hypertext Markup Language (Hiper Metin İşaret Dili)

JPG : Joint Photographic Group (Ortak Fotoğraf Grubu)

JS : Javascript

JSON : Javascript Object Notation (Javascript Nesne Notasyonu)

MAP : Mean Average Precision (Ortalama Genel Hassasiyet)

MS : Microsoft

OpenCV : Open Source Computer Vision (Açık Kaynaklı Bilgisayarlı Görü)

PDF : Portable Document Format (Taşınabilir Doküman Formatı)

PHP : Hypertext Preprocessor (Hiper Metin Ön İşlemcisi)

PNG : Portable Network Graphics (Taşınabilir Ağ Grafiği)

R-CNN : Region Based Convolutional Neural Networks (Bölgesel Tabanlı Evrişimli Sinir Ağları)

RESNET : Residual Networks (Artık Ağlar)

ROI : Region Of Interest (İlgi Bölgesi)

RPN : Region Proposal Network (Bölge Teklif Ağı)

SASS : Syntactically Awesome Style Sheets (Yazımsal Olarak Muhteşem Biçim Sayfaları)

SVG : Scalable Vector Graphic (Ölçeklenebilir Vektör Grafikleri)

TD : Table Data (Tablo Verisi)

TH : Table Header (Tablo Başlığı)

TN : True Negative (Doğru Negatif)

TP : True Positive (Doğru Pozitif)

UI : User Interface (Kullanıcı Arabirimi)

UX	: User Experience (Kullanıcı Deneyimi)
VGG	: Visual Geometry Group (Görsel Geometri Grubu)
VIA	: VGG Image Annotator (VGG Görsel Etiketleyici)
YSA	: Yapay Sinir Ağları

BÖLÜM 1

GİRİŞ

Proje geliştirme, fikir aşamasından uygulama aşamasına kadar geçen süreçteki tüm çalışmaları ifade etmektedir. 1985'ten beri her iki yılda bir yazılım projeleri ile ilgili kapsamlı analiz raporları yayınlayan The Standish Group International'ın 2015 CHAOS raporuna göre, projelerin yalnızca %29'u başarılı olabilmiştir [1]. Başlangıçta belirledikleri özellikleri, kendileri için hedeflenen zamanda oluşturulan bütçeyi aşmadan teslim eden projeler başarılı sayılmaktadır. Projelerin %19'u başarısız olmuştur. Projeye başladıktan sonra tamamlanmadan iptal edilen veya tamamlansa bile hiç kullanılmayan projeler başarısız sayılmaktadır. Projelerin %52'si ise tamamlansa bile zorlanılmıştır. Bütçeyi aşan, geç teslim edilen, başlangıçta belirlenen özelliklerin daha azıyla teslim edilen projeler zorlanılmış sayılmaktadır. Yine CHAOS araştırmalarına göre 2020'de yazılım projelerinin sadece %31'i başarılı iken, %50'si geliştirirken zorlanmış ve %19'u başarısız olmuştur [2].

Çizelge 1.1. CHAOS raporunun yıllara göre sonuçları [1, 2].

	2011	**2012**	**2013**	**2014**	**2015**	**2020**
Başarılı	%29	%27	%31	%28	%29	%31
Zorlanılmış	%49	%56	%50	%55	%52	%50
Başarısız	%22	%17	%19	%17	%19	%19

Dünyada olduğu gibi ülkemizde de yazılım projeleri bütçe, zaman ve personel sorunlarından dolayı tamamlanamadan iptal edilmekte veya tamamlansa bile istenileni karşılamadığı için hiç kullanılmamaktadır.

Yazılım teknolojisi gelişmiş ülkelerde en hızlı büyüyen sektörlerden biridir [3]. Bu sebeple yazılım firmaları veya bünyesinde yazılım departmanı barındıran firmaların sayısı artmaya devam etmektedir. Bir proje geliştirme sürecinde tasarım ve yazılım

geliştirme ekipleri yer almaktadır. Projenin alanına göre ekipler içerisinde alt ekipler bulunabilir. Genel olarak şirketlerde kullanıcı deneyimi (UX), kullanıcı arabirimi (UI), web ön yüz (frontend), web arka yüz (backend) ekipleri ile ihtiyaca göre DevOps ve mobil geliştiricilerinde yer aldığı ekipler bulunabilir.

Bu çalışmada, yazılım projelerinin bütçe, zaman ve personel giderlerini azaltmak için görüntü işleme ve derin öğrenme kullanılarak nasıl azaltıldığı ele alınmıştır. Derin öğrenme, diğer alanlara, özellikle bilgisayarlı görme problemlerinde uygulandığında klasik tekniklere göre önemli bir başarı göstermiştir. Derin öğrenme yöntemlerinin bu projede yeni bir şekilde uygulanmasının, klasik bilgisayarlı görme tekniklerine göre performansı arttırabileceği varsayılmıştır.

Deneysel bulgular ve sonuçlar, son kısımda çizelgelerle belirtilmektedir. Analiz süreci boyunca geçen işlem süreleri ise ayrıca grafiklerle sunulmaktadır.

BÖLÜM 2

TASARIM SÜRECİ VE YAZILIM GELİŞTİRME

Bu bölümde baştan sona bir projenin nasıl oluşturulduğu, tasarım ve yazılım sürecinin nasıl çalıştığı ve taslak çizimleri koda çeviren bir uygulamanın neden yararlı olduğu açıklanmaktadır.

Bir web projesine başlamadan önce proje yönetimi için hangi aracın ve tekniğin kullanılacağı ile projede görev alacak ekipler belirlenir. Yalnızca teorik kapsamda ilerleyen projeler önemli oranda uygulamaya geçemeden sonlanmaktadır. Yazılım projelerinde; ekipman, uygulamalar, hizmetler ve bir organizasyon içinde operasyon, yönetim, analiz ve karar verme işlevlerini desteklemek için bilgi sağlayan temel teknolojiler geniş bir kapsamda uygulanmaktadır [4].

2.1. BİR PROJENİN OLUŞUM SÜRECİ

Projelerin gelişim sürecinde, proje üzerinde değişiklik ve düzenleme yapılması oldukça zordur. Bu sebeple, önce proje yönetiminin oluşturulması ve projede kullanılacak olan yazılım geliştirme modelinin belirlenmesi gerekmektedir. Proje yönetimi için pek çok uygulama mevcuttur. Proje bilgileri ve yönetimi önemli bir konu olduğu için bazı uygulamalar firmaların kendi sunucularına da kurulabilmektedir. Kurumlar tarafından en çok Trello, Jira, Asana, Monday, Teamwork gibi uygulamalar kullanılmaktadır.

Yazılım geliştirme modelleri ise iş sırası, tekrar edilebilmesi, projenin uzunluğu, bütçesi ve yazılımın geliştirileceği ortama (askeri, SaaS, mobil gibi) göre değişmektedir [5]. Yazılım geliştirme modellerine yer verilecek olunursa;

- Kodla & Düzelt (Code & Fix)

- Tekrara Dayalı
- Şelale (Waterfall)
- Tekrara Dayalı Şelale (Waterfall Iterative)
- V Modeli
- Artan (Incremental)
- Prototip
- Evrimsel (Evolutionary)
- Spiral
- Birleşik Süreç (Unified Proces)
- Çevik (Agile)

Bu modeller arasında sektörde en çok bilinen ve kullanılanlar şelale, artan, birleşik süreç ve çevik modeldir. Bir model belirlerken projenin özelliklerine ve süreçlerinize göre yaşam döngüsü fazlarının uzunluğu, önceliği ve tekrar durumuna bakılmaktadır [6]. Yaşam döngüsü üzerindeki fazlar planlama, analiz, tasarım, gerçekleştirme ve bakım olarak sınıflandırılabilir. Yaşam döngüsü üzerindeki her bir fazın kendi içerisinde birçok farklı disiplini bulunur [7].

Bir proje birbiriyle bağlantılı birçok modülden oluşmaktadır. Kullanıcılar, müşteriler, tasarım ve geliştirme ekibi, bakım ekibi ve yönetim kadrosunu eş zamanlı olarak koordine etmek projelerde proje yöneticisinin görevidir. Kullanıcılar çok fazla istekte bulunabilirler ve projenin her şeyi yapmasını isteyebilirler. Müşterilerde bütçeye önem verdikleri için düşük bütçeyle en iyi ürünü almak isterler. Yöneticilerde iddialı ve hedeflerine göre bir proje geliştirilmesini ve yüksek kar edilebilmesini istemektedir. Geliştirme ekibi ise ileride oluşabilecek teknik problemler ve zorluk aşamalarını da hesaba katarak projenin zaman sıkıntısının esnekleştirilmesini istemektedir. Tüm bu istekler bir arada düşünüldüğünde proje içinde çatışmalar meydana gelmektedir [8,9]. Belirlenecek olan yazılım modeli, proje yöneticisinin bu zorlukları aşmasına yardımcı olmalıdır. Modelden sonra bir uygulama oluşturmanın ilk adımı, UI yapısının belirlenmesini sağlayan taslak çizimlerin (wireframe) oluşturulmasıdır [10,11]. Geliştiriciler, oluşturulan taslak çizimleri ve UI tasarımlarını koda dönüştürürken, kod çıktısını tasarımlara göre oluştururlar. Bu iş geliştirici için zaman alıcıdır ve bu nedenle maliyetlidir [12].

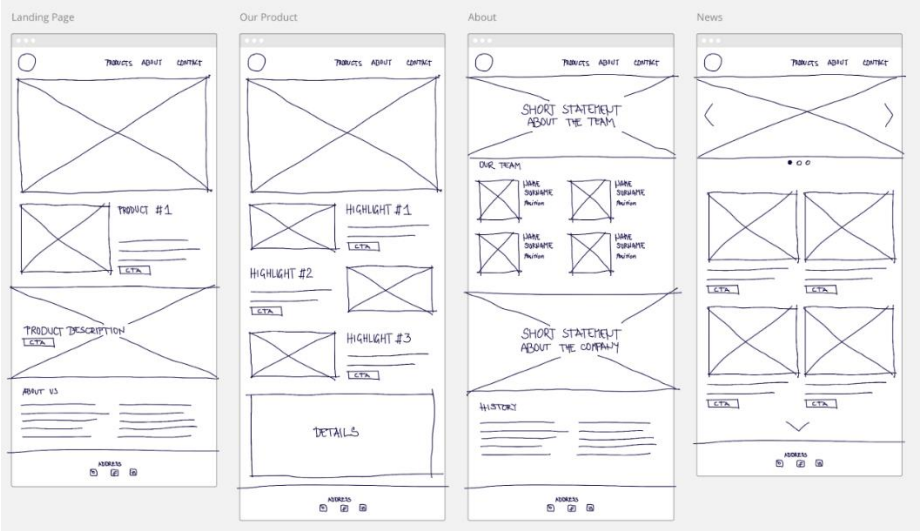
Bir tasarımı koda dönüştürme sırasında alınan sorunları daha önce ele alan çalışmalar olmuştur. SILK [13] uygulama aracılığıyla dijital ortamdaki çizimleri koda dönüştürür; DENIM [14] tasarım araçları ve kod çıktısı arasındaki uyumu yakalamak için çizimleri zenginleştirir; REMAUI [15], yüksek kaliteli ekran görüntülerini parçalayarak mobil uygulamalara dönüştürür; sketch2code [16], Faster R-CNNN kullanarak kâğıt üzerine çizilen UI ögeleri tespit etmektedir; Sketch2code [17], kâğıt üzerine çizilen taslak çizimlerden eğitilen 5 UI ögesini tespit ederek HTML koduna dönüştürür. Bu uygulamaların çoğu, tespit ve sınıflandırma gerçekleştirmek için klasik bilgisayarlı görme tekniklerine dayanır. Bu yöntemler dışında yapılan çalışmalar ise UI öge sayısı veya modelin tahmin etme mekanizması açısından verimli bulunmamaktadır.

2.2. TASARIM SÜRECİ

Tasarım süreci tasarımcıya göre değişmekle birlikte, birçok proje için genellikle dijital veya taslak çizimler ile başlar [10]. Bu bölümde bir uygulamanın sıfırdan tasarlanma aşamalarına yer verilmektedir.

2.2.1. Taslak Çizimler (Wireframe)

Taslak çizimler, uygulamanın temel yapısını özetleyen bir belgedir. Renkler gibi belirli ayrıntıları tanımlamadığı için aslına uygunluğu düşük bir tasarım belgesidir. Bir taslak çizim oluşturulduktan sonra gözden geçirilir ve daha fazla ayrıntı eklenir, yani daha yüksek kaliteli bir tasarım şablonu haline gelir [12]. Tasarım tamamlandıktan sonra bir geliştirici tarafından koda dönüştürülür. Genel olarak bir uygulama içerisinde çok fazla sayfa olduğu için bu süreç zaman alıcı bir süreçtir. Bir tasarımcı bir web sitesi oluşturmak istediğinde, fikirleri koda dönüştürülmeden önce tasarımda tüm ayrıntılar üzerinde çalışması gerekir. Ayrıca, bir tasarımı koda çevirmek zaman alıcıdır ve geliştiriciler maaşları yüksektir. Şekil 2.1'de örnek bir projedeki taslak çizim sayfaları gösterilmektedir.



Şekil 2.1. Taslak çizim örnekleri.

Taslak çizimler, Balsamiq [18], Marvel [19] ve Wirify [20] gibi çevrimiçi uygulamalar ile tasarlanabilir. Herhangi bir standart olmamasına rağmen yaygın olarak benzer bir dizi semboller kullanır. Bu tez çalışmasında, Bölüm 4.3'te detaylandırılan UI ögelerine odaklanılmaktadır.

Dijital taslak çizimi araçlarının varlığına rağmen birçok tasarımcı hala kâğıt üzerinde çizim yapmaya [10, 11] ve sayısallaştırmaya [21] devam edilmektedir. Myers'e göre bu, öncelikle tasarımcıların arka planının sanatta olması, dijital araçlarla kısıtlanmış hissetmesi veya herhangi bir yazılım öğrenmesine gerek kalmadan bunun en kolayı olmasıyla açıklanır [11]. Bu nedenle, tez çalışmasında dijital ortamda çizilen taslak çizimler ile kâğıda çizilen taslak çizimlerde yer alan UI ögelerinin tespit edilmesi amaçlanmıştır.

2.2.2. Kullanıcı Deneyimi (UX) ve Kullanıcı Arabirimi (UI)

Kullanıcı deneyimi ve arabirimi, genellikle birbirinin yerine kullanılan ancak aslında çok farklı şeyler ifade eden iki terimdir. Her iki terim bir ürün için çok önemlidir ve birlikte çalışır. Ancak rolleri oldukça farklıdır, ürün geliştirme sürecinin ve tasarım disiplininin çok farklı yönlerine atıfta bulunur. UX tasarımı, ürünleri tasarlamanın insan odaklı bir yoludur. Bilim adamı ve Nielsen Norman Group Design Consultancy'nin kurucu ortağı olan Don Norman, 1990'ların sonlarında kullanıcı

deneyimi terimini icat etmesiyle tanınmaktadır ve kullanıcı deneyimini şu şekilde tanımlar. "Kullanıcı deneyimi, son kullanıcının şirketin hizmetlerini ve ürünleri ile etkileşiminin tüm yönlerini kapsar [22]". UX tasarımcısının gerçekte ne yaptığı hakkında çok detay verilmese de UX tasarımın potansiyel veya aktif bir müşteri ile bir şirket arasındaki tüm etkileşimleri kapsadığı söylenmiştir. Bilimsel bir süreç olarak her şeye uygulanabilmektedir. Örneğin; sokak lambaları, arabalar, koltuklar vb.

Ancak bilimsel bir terim olmasına rağmen, başlangıcından bu yana kullanımı neredeyse tamamen dijital alanlarda olmuştur. Bunun en büyük nedeni, UX teriminin icadı sırasında teknoloji endüstrisinin çok hızlı büyümeye başlamasıdır. Esasen UX, kullanıcı ile bir ürün veya hizmet arasındaki etkileşimi ifade eder. UX tasarımı, bu deneyimi şekillendiren tüm farklı unsurları dikkate alır. Bir UX tasarımcısı, deneyimin kullanıcıyı nasıl hissettirdiğini ve kullanıcının istenen görevleri gerçekleştirmesinin ne kadar kolay olduğunu düşünür ve bu duruma göre tasarıma katkı sağlar. Kullanıcı deneyimi, bir ürünün etkili ve keyifli kullanım için optimizasyonuna odaklanan bir görevler topluluğu iken, UI tasarımı onun tamamlayıcısıdır.

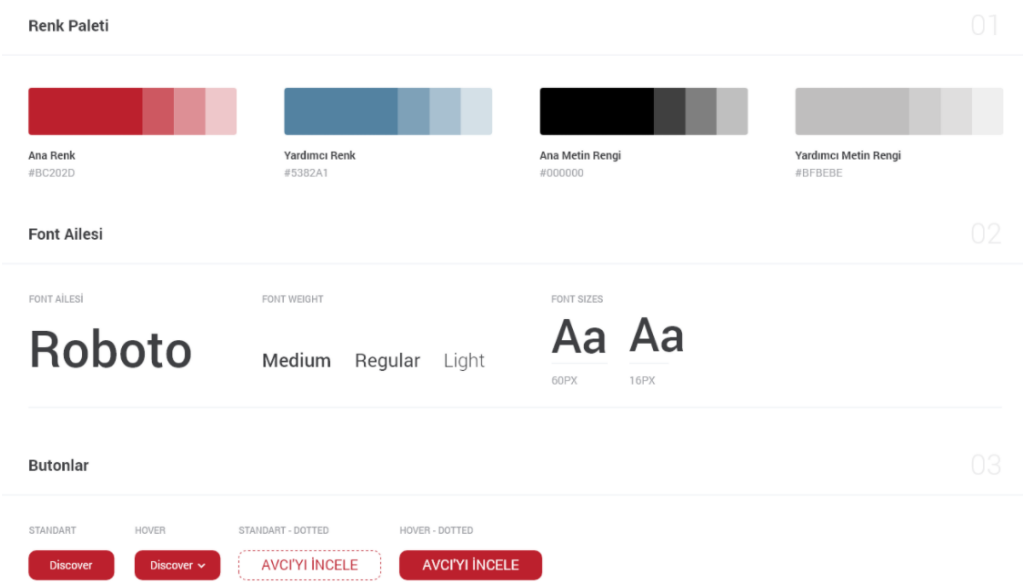
UI tasarımcıları için oluşturulan iş tanımlarına bakacak olunursa, mesleğin çoğunlukla grafik tasarıma benzeyen, bazen marka tasarımına ve hatta ön yüz geliştirmeye kadar uzanan tanımlara ulaşılacaktır. Uzmanlar tarafından yapılan tanımlara bakılacak olunursa UI tasarımı için çoğunlukla UX tasarımı ile kısmen aynı olduğu ve hatta aynı yapısal tekniklere atıfta bulunan açıklamalara ulaşılacaktır. Kullanıcı deneyiminden farklı olarak, UI tasarımı tamamen dijital bir terimdir. UI, akıllı telefonlardaki dokunmatik ekran veya kahve makinesinden ne tür kahve istendiğini seçmek için kullanılan dokunmatik yüzey gibi, kullanıcı ile dijital cihaz veya ürün arasındaki etkileşim noktasıdır. Web siteleri ve uygulamalarla ilgili olarak, UI tasarımı ürünün görünümünü, verdiği hissi ve etkileşimi dikkate alır. Her şey, bir ürünün kullanıcı arabiriminin mümkün olduğunca sezgisel olduğundan emin olmakla ilgilidir ve bu, kullanıcının karşılaşabileceği her görsel, etkileşimli ögeyi dikkatlice düşünmek anlamına gelir [23].

Bir UI tasarımcısı, ikonlar, butonlar, tipografi, renk seçenekleri, boşluklar, görseller ve duyarlı tasarım (responsive) hakkında çalışmalar yapmaktadır. UX tasarımı gibi, UI

tasarımı da çok yönlü ve zorlu bir roldür. Bir ürünün geliştirme, araştırma, içerik ve düzeninin kullanıcılar için çekici, yol gösterici ve duyarlı bir deneyime aktarılmasından sorumludur. UI tasarımının amacı, kullanıcıya bir ürünün arabirimi boyunca görsel olarak rehberlik etmektir. Her şey, kullanıcının çok fazla düşünmesini gerektirmeyen sezgisel bir deneyim yaratmakla ilgilidir. UI tasarımı, markanın güçlü yönlerini ve görsel varlıklarını bir ürünün arabirimine aktararak tasarımın tutarlı ve estetik açıdan hoş olmasını sağlar [24].

2.2.3. Stil Kılavuzları

Bir stil kılavuzu, tutarlı bir marka bilinci oluşturma için gerekli tüm ayrıntıları tek bir merkezi belgede toplamayı sağlayan kılavuzdur. Çalışmaların stil kılavuzlarında tipografi standartları, renk kombinasyonları, logo varyasyonları, metin stilleri, UI ögelerin görünümleri ve stilleri, görsel şablonları vb. ögeler bulunabilir. Şekil 2.2'de görüldüğü üzere, örnek olarak bir stil kılavuzuna yer verilmiştir.



Şekil 2.2. Örnek bir stil kılavuzu.

Eğer bir logonun versiyonları ile ilgili net kurallar var ise bunlar stil kılavuzunda belirtilir. Logo etrafında ne kadar boşluk bırakılmalı, renk seçenekleri, alternatif yatay ve dikey logolar, logonun boyutsal orantılı standartları belirtilir [25].

2.3. YAZILIM GELİŞTİRME

Yazılım geliştirme, bir uygulamanın birbirini takip eden işlemlerin düzenli bir şekilde tamamlanması işlemidir. Bu işlemler, uygulamada kullanılacak kodların yazıldığını değil, kodlanacak uygulamanın dokümantasyonunu, mimarisini, tasarımını, hedeflerini, gereksinimlerini ve tamamlandığında amaçların sağlanıp sağlanmadığının doğrulanmasını içerir. Bu işlemler Bölüm 2.1'de anlatılan yazılım geliştirme modellerine göre değişebilir.

2.3.1. Web Ön Yüz (Frontend) ve Web Arka Yüz (Backend)

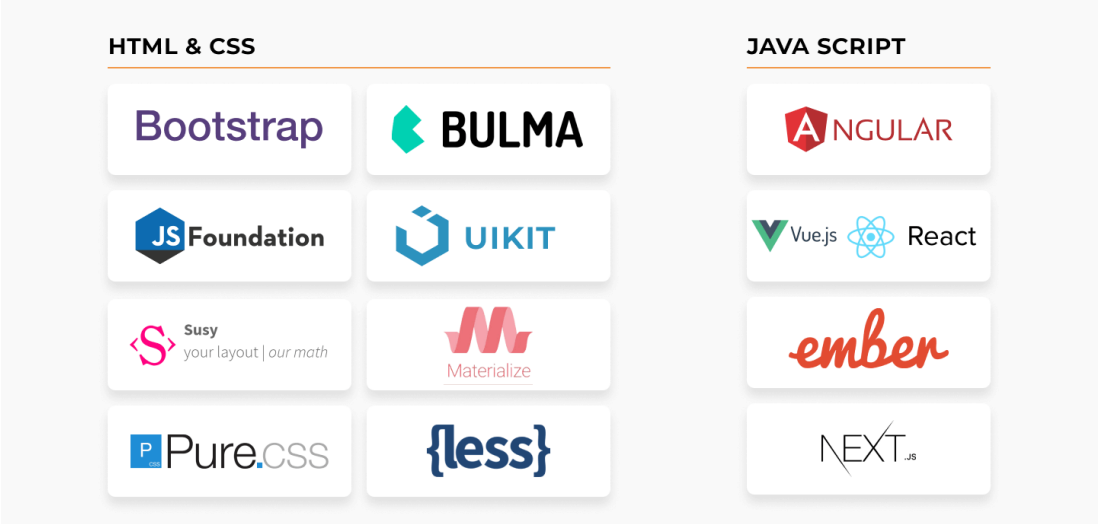
Tüm web siteleri aynı dilde, hiper metin işaretleme dilinde (HTML) oluşturulurken, mobil ve masaüstü uygulamaları birkaç dilde oluşturulabilir. Frontend ve backend, web geliştirmede kullanılan en popüler iki terimdir. Bu terimler web geliştirme için çok önemlidir ancak birbirinden oldukça farklıdır. Web sitesinin işlevselliğini geliştirmek için her iki tarafın da diğeriyle tek bir birim olarak etkin bir şekilde iletişim kurması ve çalışması gerekir.

Kullanıcının doğrudan etkileşimde bulunduğu bir web sitesinin parçası frontend olarak adlandırılır. Ayrıca uygulamanın "istemci tarafı" olarak da anılır. Kullanıcıların doğrudan deneyimlediği her şeyi içerir: metin renkleri ve stilleri, görseller, grafikler ve tablolar, butonlar, renkler ve menü. HTML, CSS ve JavaScript, frontend geliştirme için kullanılan dillerdir. Web siteleri, web uygulamaları veya mobil uygulamalar açıldığında tarayıcı ekranlarında görülen her şeyin yapısı, tasarımı, davranışı ve içeriği frontend geliştiriciler tarafından uygulanır. Duyarlılık ve performans, ön yüz geliştirmenin iki ana hedefidir. Geliştirici, sitenin duyarlı olduğundan, yani her boyuttaki cihazlarda doğru göründüğünden emin olmalıdır, ekranın boyutundan bağımsız olarak web sitesinin hiçbir parçası anormal şekilde davranmamalıdır [26].

Bir işaretleme dili kullanarak web sayfalarının ön yüz bölümünü tasarlamak için kullanılır. HTML, hiper metin ve işaretleme dilinin birleşimidir. Hiper metni, web sayfaları arasındaki bağlantıyı tanımlar. İşaretleme dili, web sayfalarının yapısını tanımlayan etiket içindeki metin belgelerini tanımlamak için kullanılır. CSS

(Basamaklı Stil Şablonları), web sayfalarını sunulabilir hale getirme sürecini basitleştirmeyi amaçlayan basit bir şekilde tasarlanmış bir dildir. CSS, web sayfalarına stiller uygulamanıza olanak tanır. Daha da önemlisi, CSS bunu her web sayfasını oluşturan HTML'den bağımsız olarak yapmanızı sağlar. JavaScript, siteyi kullanıcı için etkileşimli hale getirmek için kullanılan bir betik dildir.

Ön yüz geliştirme kendi içinde web çerçevelerine ve kütüphanelerine ayrılmaktadır. Bunlar; AngularJS, jQuery, React JS, Vue JS, Bootstrap, SASS, Semantic UI vb.'dir. Şekil 2.3'te bu web çerçeveleri ve kütüphanelerinden birkaçına yer verilmiştir.



Şekil 2.3. Frontend geliştirme web çerçeveleri ve kütüphaneleri.

Arka yüz, web sitesinin sunucu tarafıdır. Verileri depolar, düzenler ve ayrıca web sitesinin istemci tarafındaki her şeyin düzgün çalışmasını sağlar. Web sitesinin göremediğiniz ve etkileşimde bulunamayacağınız kısmıdır. Yazılımın kullanıcılarla doğrudan temas etmeyen kısmıdır. Arka yüz geliştiricileri tarafından geliştirilen parçalara ve özelliklere, kullanıcılar tarafından bir ön yüz uygulaması aracılığıyla dolaylı olarak erişilir. API yazma, veri tablolarını oluşturma, kullanıcı arabirimleri ve hatta bilimsel programlama sistemleri olmadan sistem bileşenleriyle çalışma gibi etkinlikler de arka yüzde yer alır [27].

Şekil 2.4. En çok kullanılan backend dilleri.

Arka yüz geliştirme kendi içinde dilleri ve dillere bağlı olarak çerçeve ve kütüphanelere ayrılmaktadır. Şekil 2.4'te gösterildiği gibi PHP, Ruby, Python, Java, C++, JavaScript, Go vb. diller arka yüz geliştirme dillerine, Express, Rails, Django, Laravel, CodeIgniter vb. çerçevelere örnek olarak gösterilebilir. Ön yüz, kullanıcıların gördüğü ve etkileşime girdiği alan anlamına gelirken arka yüz, nasıl çalıştığının planlandığı alandır.

BÖLÜM 3

LİTERATÜR TARAMASI

Bu bölümde, tasarımdan kodlama uygulamaları ve UI ögeleri ile yapılan çalışmalar incelenerek tez çalışması için bir hazırlık yapılması ön görülmüştür. Tez çalışmasında kullanılacak olan taslak çizimlerin sektördeki önemi ve kullanım alanları anlatılmaktadır.

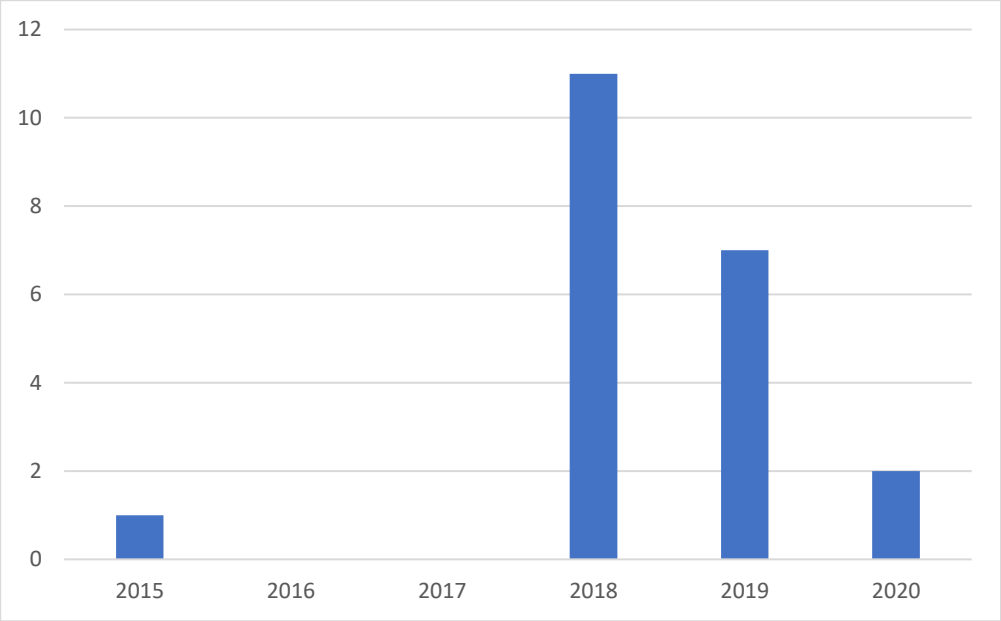
Otomatik fonksiyonel programlama alanında Bunel ve ark. [28] verimli program öğrenimi sorununu ele almak için uyarlanabilir bir sinirsel derleme çerçevesi önermiştir. Riedel ve ark. [29], çalışmasında, programcıların program giriş-çıkış verilerinden eğitilmiş davranışlarla doldurulabilen alanlara sahip program taslakları yazmasına olanak tanıyan Forth programlama dili için uçtan uca türevlenebilir bir yorumlayıcı üzerinde çalışmıştır. Ling ve ark. [30], yeni bir sinir ağı mimarisi sunarak, karışık bir doğal dil ve yapılandırılmış sınıflar ile programlama kodu üretme sorunu ele almıştır. Gaunt ve ark. [31], girdi-çıktı örnekleri verildiğinde, girdileri karşılık gelen çıktılara eşleyen kaynak kodunu sentezlemeye çalışmıştır. Balog ve ark. [32], derin öğrenme kullanarak bilgisayar programları oluşturmak için istatistiksel tahminlerden yararlanan DeepCoder'ı önermiştir.

Bu çalışmalara kadar görsel girdilerle kod oluşturma, Beltramelli [33] pix2code'u önerene kadar hala keşfedilmemiş bir araştırma alanıdır. Pix2code mimarisi, diğer alanlara uygulanan bazı modellere benzemektedir [34-37]. Pix2code'un kod üretimi dikkat mekanizmasından yoksun olduğundan [33], pix2code'un sonucu beklentiden uzaktır. Huang ve ark. [38], pix2code'daki veri setini kullanarak bu sorunu ele almışlardır. Bu çalışmaların çoğu etki alanına özgü dillere (DSL) dayanmaktadır. DSL'ler özel bir etki alanı için tasarlanmış programlama dilleridir. Tam özellikli programlama dilleriyle karşılaştırıldığında DSL'ler daha kısıtlayıcıdır. DSL'ler,

programlama dilinin karmaşıklığını sınırlar, bu da otomatik programlamayı kolaylaştırır ve özel amaçlı arama algoritmasını verimli hale getirir [39].

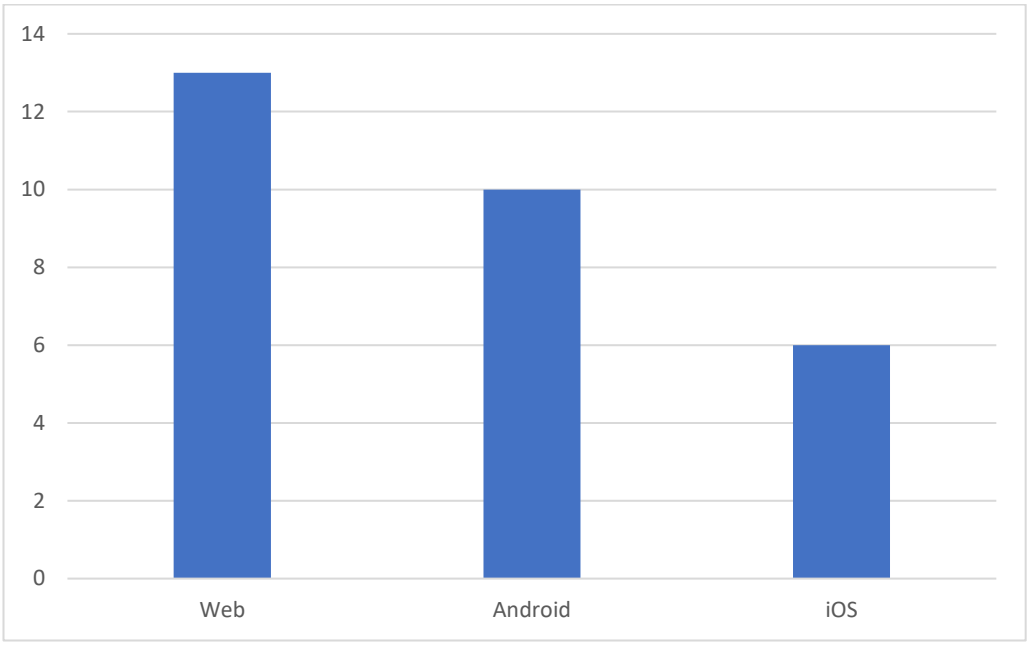
Robinson [17], çalışmasında derin öğrenme yaklaşımının klasik bilgisayarlı görme yaklaşımlarından daha iyi performans gösterdiğini, kendi veri setini yayınlayarak açıklamıştır. Abdelhamid ve ark. [40], YOLOv5 kullanarak android tabanlı GUI prototipine dönüştürmeyi araştırmıştır. Pan ve ark. [41], UI tasarım ile web ön yüz kodlaması arasındaki zamanı hızlandırmak için Visual Compiler adında program önermişlerdir. Xu ve ark. [42], UI ögelerini algılama ve web ön yüze çevirmek için image2emmet adlı bir yaklaşım önermiştir. Baule ve ark. [43], taslak çizimleri oluşturabilen, yapılandırabilen ve koda dönüştüren WebWire adlı bir aracın geliştirilmesini önermiştir. Aşıroğlu ve ark. [44], derin öğrenme yöntemleri kullanarak elle çizilmiş taslak çizimlerinden kod oluşturma sürecinin otomasyonu için yaklaşım önermişlerdir. İlgili çalışmaların çoğu son 3 yılda yayınlanmıştır. Çalışmaların yıllarda göre dağılımı Çizelge 3.1'de gösterilmiştir. Bu da çalışma alanımıza ilginin son yıllarda arttığını göstermektedir.

Çizelge 3.1. Çalışma alanında yıllık yapılan çalışma sayısı [45].



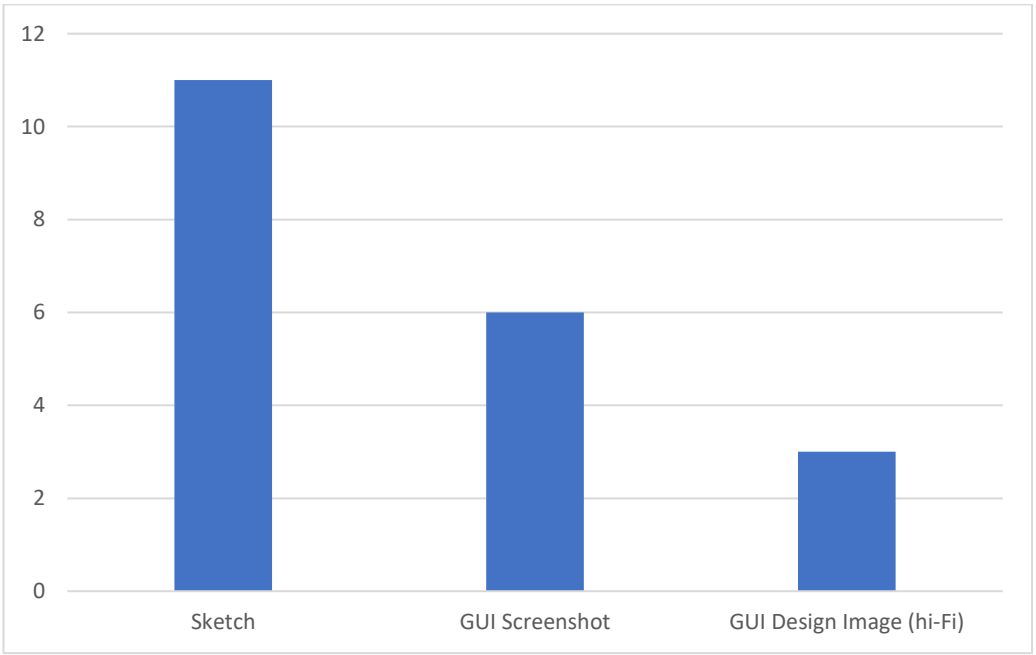
Çalışmalar web ve mobil alanı hedeflenmektedir. Çoğu çalışma web GUI'leri ile yapılmıştır. Platforma göre dağılım Çizelge 3.2'de gösterilmiştir.

Çizelge 3.2. Platforma göre çalışma sayıları [45].



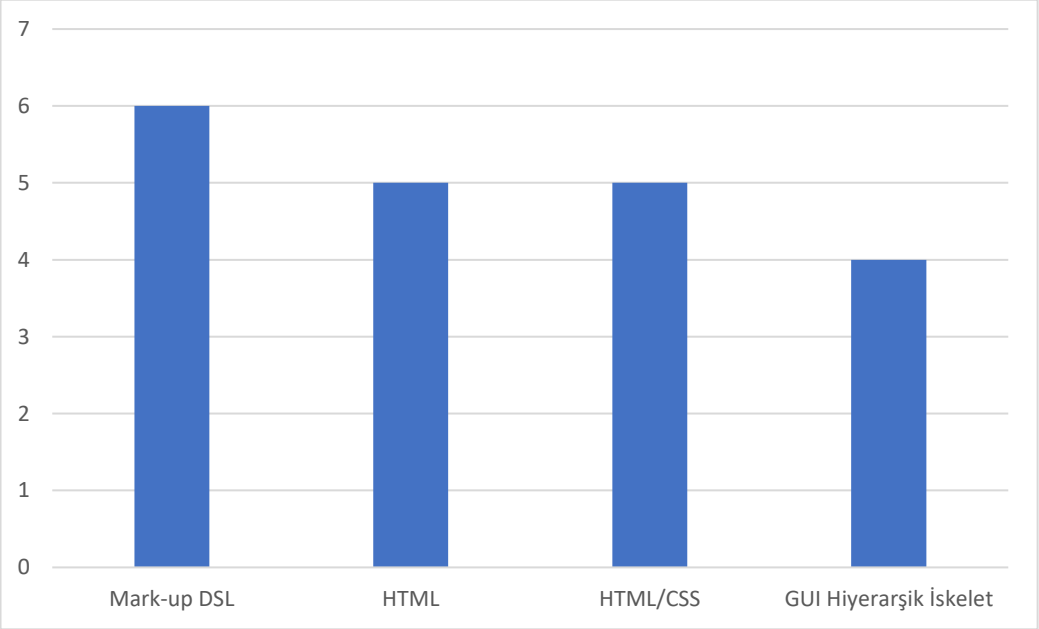
Çizelge 3.3'te veri seti olarak kullanılan taslakların çalışma sayıları gösterilmektedir. Çizelge 3.3 incelendiğinde, çoğunlukla taslak çizimler üzerinde wireframe üzerinde çalışmalar yapıldığı gözlemlenmiştir.

Çizelge 3.3. Veri seti olarak kullanılan tipe göre çalışma sayıları [45].



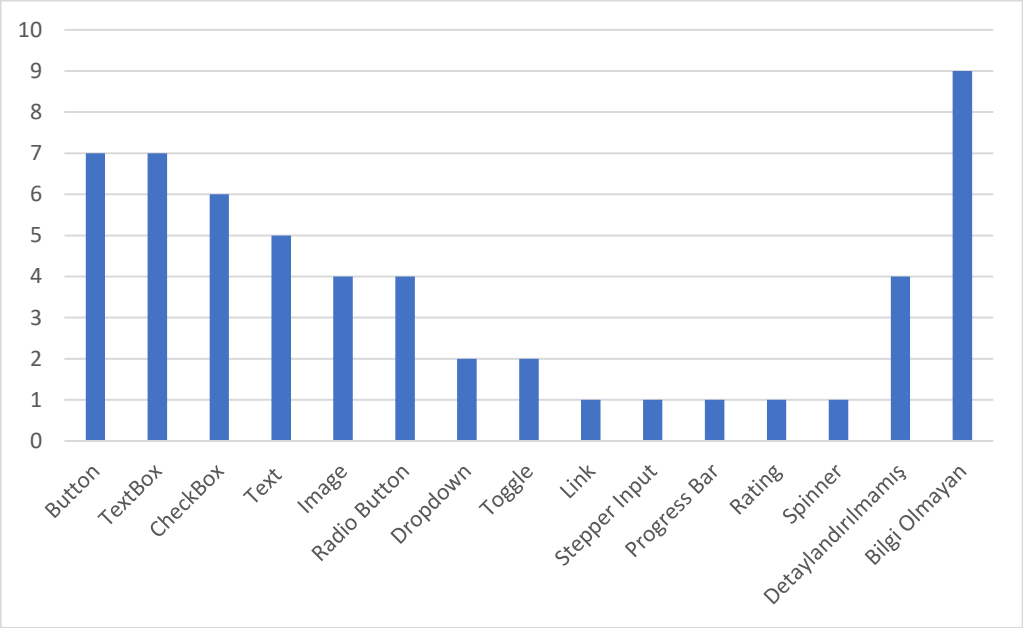
Çizelge 3.4'te ise kod çıktısı olarak hedeflenen çalışma sayıları gösterilmektedir. Bununla birlikte Çizelge 3.4 göz önünde bulundurulduğunda ise çıktı olarak DSL ve HTML çıktıların tercih edildiği gözlemlenmiştir [44].

Çizelge 3.4. Kod çıktısı olarak hedeflenen çalışma sayıları [45].



Birçok çalışmada, hangi tip UI ögelerinin tespit edildiğine dair detaylı bir bilgi yoktur. Genel olarak algılamayı basitleştirmek için bazı çalışmalar algılanacak ögeleri yalnızca küçük bir kümeyle sınırlamıştır. Örneğin, Aşıroğlu ve ark. [44], yalnızca metin, dropdown, buton ve checkbox ögeleri dikkate almıştır. Robinson [17] dört ögeyi, Ge [46] yedi ögeyi dikkate alır. Jain ve ark. [47] en sık kullanılan on ögeye odaklanırken, Moran ve ark. [48] 15 ögeye odaklanmıştır. Suleri ve ark. [49] Material Design kapsamında 19 ögesine odaklanırken, Pandian ve ark. [50], en fazla UI ögesi ile çalışma yaparak 25 ögeye odaklanmıştır. En çok karşılaşılan buton ve metin daha çok odaklanıldığı gözlemlenmiştir. Detaylı istatistik Çizelge 3.5'te gösterilmiştir. Çoğu çalışma metni ayırmadan algılama yapmıştır. Yalnızca Jain ve ark. [47] ile Robinson [17], metni paragraf ve başlık olarak ayrı algılamıştır.

Çizelge 3.5. Yapılan çalışmalarda en çok odaklanılan UI ögeleri [45].



Çizelge 3.6'de literatürde yer alan çalışmalara yer verilmiştir. Buna göre, çalışmalarda kullanılan en fazla UI ögesine sahip olan çalışma Pandian ve ark. [50] tarafından gerçekleştirilmiştir.

Çizelge 3.6. Yapılan çalışmalarda platform, girdi ve çıktı özellikleri.

İlgili çalışma	**Platform**	**Girdi**	**Çıktı**	**Öge sayısı**
Aşıroğlu ve ark. [44]	Web	Taslak çizim	HTML	5
Bajammal ve ark. [51]	Web	UI görsel (hi-fi)	HTML	-
Beltramelli [52]	Android, iOS, Web	Taslak çizim	HTML/CSS	-
Beltramelli [33]	Android, iOS, Web	UI ekran görüntüsü	DSL	-
Chen [53]	Android	UI görsel (hi-fi)	DSL	-
Chen [54]	Android, iOS	UI ekran görüntüsü	DSL, Android ve iOS	-
Ge [46]	Android	Taslak çizim	JSON, Android	7
Halbe ve ark. [55]	Web	Taslak çizim	HTML	-
Han ve Joshi [56]	Web	Taslak çizim	HTML	-

Huang ve ark. [38]	Web	UI ekran görüntüsü	HTML/CSS	8
Jain ve ark. [47]	Android, iOS, Web	Taslak çizim	HTML	10
Kim ve ark. [57]	Web	Taslak çizim	HTML/CSS	5
Kumar [58]	Web	Taslak çizim	HTML/CSS	16
Liu ve ark. [59]	Android, iOS, Web	UI ekran görüntüsü	DSL	-
Robinson [17]	Web	Taslak çizim	Hiyerarşik UI iskelet, HTML	5
Pandian ve ark. [60]	Android	UI ekran görüntüsü	Hiyerarşik UI iskelet	25
Moran ve ark. [48]	Android, iOS	UI ekran görüntüsü	Hiyerarşik UI iskelet	15
Suleri ve ark. [56], Pandian ve ark. [61]	Android	Taslak çizim	DSL, XML	19
Wallner [62]	Web	UI ekran görüntüsü	HTML/CSS	17
Yun ve ark. [63]	-	Taslak çizim	DSL, XML	-

BÖLÜM 4

VERİ SETİ

Taslak çizimler genel olarak kâğıt veya beyaz tahta üzerinde tasarlanmaktadır. Bu sebeple veri seti, bilgisayar ortamında beyaz arka plan üzerine koyu renkli işaretleyiciler ile çizilen taslak çizimler üzerine çalışmakla sınırlandırılmıştır.

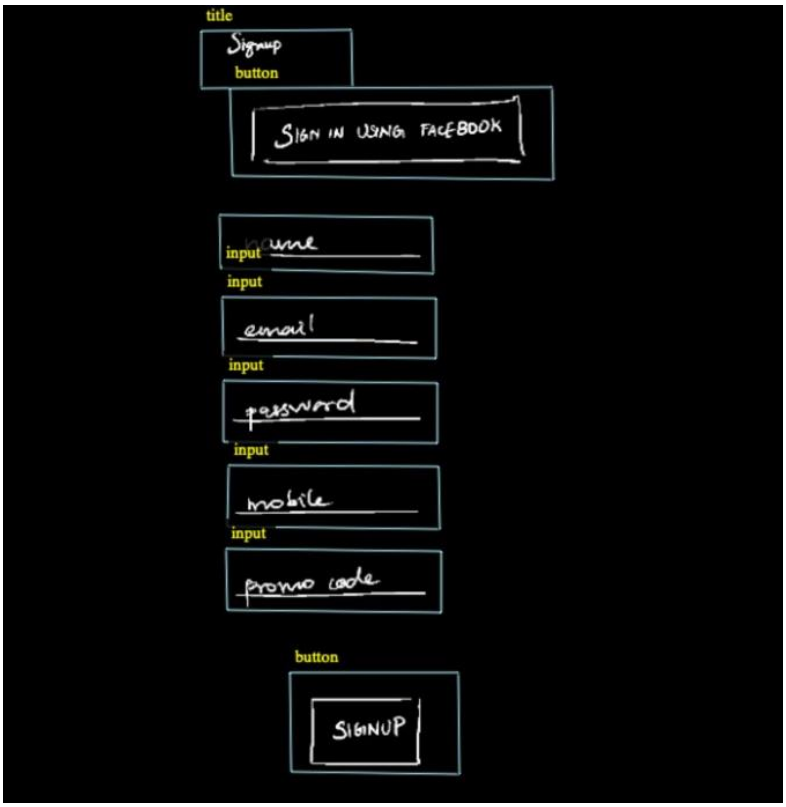
4.1. VERİYİ HAZIRLAMA

Çalışmada kullanılacak olan veri kümesi, Wacom Bamboo Slate Large CDS-810S model akıllı ajanda yardımıyla çizilmiştir. Akıllı ajanda yardımıyla ajanda üzerinde bulunan kâğıda çizilen UI ögeleri Wacom Inkspace uygulaması yardımıyla canlı olarak telefon, tablet ve bilgisayar ortamı kullanılarak bulut sunucu üzerine aktarılabilmektedir. Akıllı ajanda ile kullanılan Wacom Bamboo Slate Wired model kalem yardımıyla basınç hassasiyeti de dikkate alınmıştır. Bulut sunucu üzerinde bulunan çizimler DOC, JPG, PNG, PDF ve SVG gibi formatlarda kaydedilmektedir. Toplamda 457 adet taslak çizim içinde 4256 UI ögesi kullanılmıştır. Wired kaleminin basınç hassasiyeti dikkate alınarak silik, kalın gibi şekillerde veriler oluşturulmuştur.

4.2. VERİ ETİKETLEME

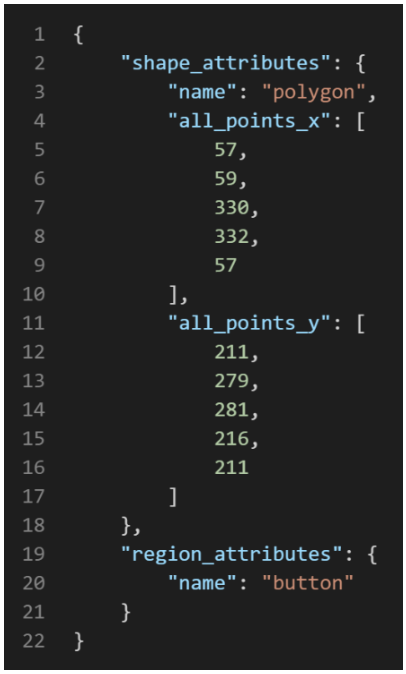
Etiketleme aracı olarak, VGG Image Annotator 1.0 versiyonu kullanılmıştır. VGG Image Annotator (VIA), görüntü, ses ve video için basit ve bağımsız bir manuel açıklama yazılımıdır. VIA bir web tarayıcısında çalışır ve herhangi bir kurulum veya kurulum gerektirmez. Eksiksiz VIA yazılımı, çoğu modern web tarayıcısında çevrimdışı bir uygulama olarak çalışan, 400 kilobayttan daha küçük, tek bir bağımsız HTML sayfasına sığar [64].

Şekil 4.1'de verilerin etiketlenme arabirimine yer verilmektedir. Şekil 4.1'de görüldüğü gibi veriler etiketlenerek gerçek referans (ground-truth) etiketleri oluşturulmaktadır. VIA yazılımı kullanıcılara etiketlemede kullanılması için kare, dikdörtgen, daire ve poligon gibi araçlar sunmaktadır. UI ögeleri çok yakın mesafelerde, iç içe ve düzensiz şekillerde bulunabileceğinden kare, dikdörtgen, daire gibi sabit şekiller yerine kullanıcının belirleyebileceği poligon aracı tercih edilmiştir. Şekil 4.1'de VIA aracı ile örnek bir taslak çizim örneği gösterilmektedir. Bu görsele göre nesne olan bölgeler poligon nesnesi olarak etiketlenmiştir.



Şekil 4.1. VIA aracı ile etiketlenen taslak çizim örneği.

Etiketleme ve ek açıklamalar aşamasında kullanılan VIA aracı ile köşe koordinatlar her bir görüntü için Şekil 4.2'deki gibi JSON dosyasına kaydedilmektedir. Kategoriler değerlendirilirken nesne olan UI ögeleri 1 olarak etiketlenir. Arka plan olarak işlem gören bölgeler ise 0 olarak nitelendirilmektedir.



Şekil 4.2. Bir UI ögesinin etiketleme sonrası elde edilen JSON kodu.

4.3. UI ÖGELERİ

UI ögeleri, tüm uygulamalar için temel yapı taşlarıdır. Mobil, web, masaüstü veya sanal gerçeklik uygulaması fark etmeksizin bir yazılım uygulamasının en ayrılmaz parçasıdır. UI ögeleri, kullanıcı ve uygulama arasındaki etkileşimlerden sorumludur. Bir UI ögesi kullanıcının alışkanlıklarına göre farklı şekilde tasvir edilebilmektedir. Bu sebeple kullanım sıklığı da dikkate alınarak farklı şekillerde UI ögeleri çizilmiştir.

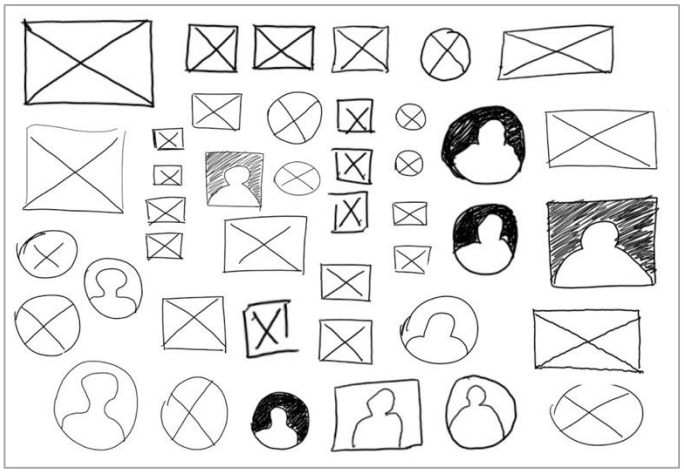
UI ögeleri 3 ana kategoride gruplanmaktadır.

- Giriş Ögeleri: Farklı kullanıcı girişlerinin işlenmesinden sorumludur. Bazen de giriş doğrulama sürecinin bir parçası olurlar. En çok kullanılan girdi ögelerinden bazıları şunlardır: dropdown, button, combo box, toggle, input, datepicker, radio button, checkbox
- Çıkış Ögeleri: Çeşitli kullanıcı girdilerine karşı sonuçların gösterilmesinden sorumlu olan ögelerdir. Ayrıca kullanıcılara bilgilendirme, uyarı, başarılı ve hatalı mesajları gösterirler. Çıktı ögeleri doğası gereği nötr değildir. Toast veya popup olarak örnek olarak gösterilebilir.

- Yardımcı Ögeler: Diğer tüm unsurlar yardımcı ögeler kategorisine girmektedir. Yaygın olarak kullanılanlar: bildirimler, breadcrumblar, ikonlar, progress barlar ve tooltipler. Yardımcı ögelerde kendi içerisinde 3 kategoriye ayrılmaktadır.
    - Navigasyonlar: UI navigasyonundan sorumludur. Gezinmeye yardım eden ögeler, gezinme menüleri, bağlantı listesi ve breadcrumblar bu kategoriye girmektedir.
    - Bilgilendirici: Bilgiyi temsil etmekten sorumludur. Tooltipler, ikonlar ve progress barlar bu kategoriye girmektedir.
    - Kutular/Gruplar: Çeşitli ögeleri bir arada tutmaktan sorumludurlar. Widgetlar, sidebar ve menüler bu kategoriye girmektedir.

4.3.1. Resim (Image)

Resimler, bir web sayfasının tasarımını ve görüntüsünü iyileştirmek için kullanılmaktadır. Kullanıcıların alışkanlıklarına göre insan silueti veya geometrik şekil içerisine çarpı işareti ile tasvir edilebilir.

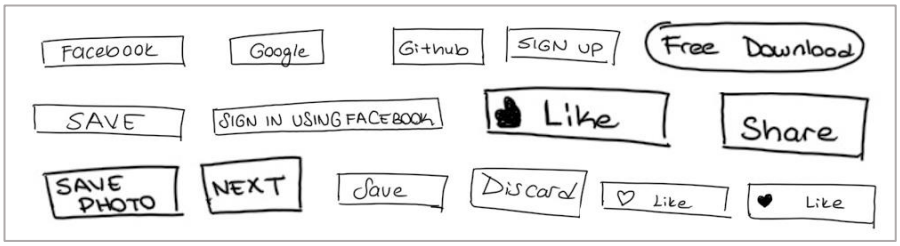


Şekil 4.3. Çalışmada oluşturulan birden fazla resim görseli.

Şekil 4.3'te çalışmada üretilen birden fazla resim görseli bulunmaktadır. Çeşitlilik artırılarak modelin test işlemi için uygulayacağı tahmin kapasitesinin yüksek performansta işlem yapması sağlanmaktadır.

4.3.2. Buton (Button)

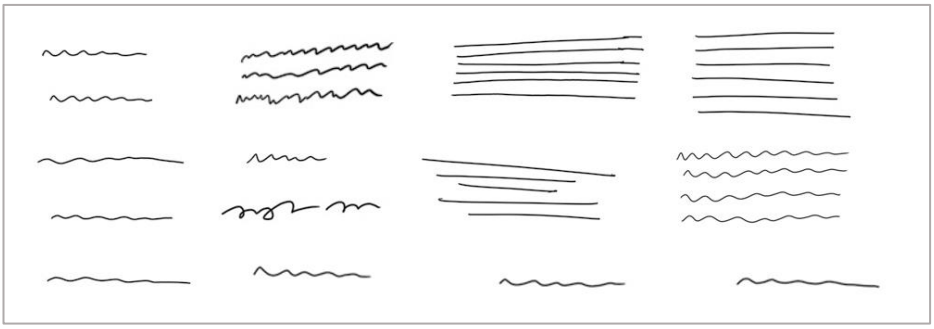
Butonlar, kullanıcıların dokunma veya tıklama ile bir işlem gerçekleştirmesini sağlar. Genellikle metin, ikonlar veya her ikisi ile etiketlenir. Butonlar, bir kullanıcı arabiriminin en önemli parçalarından biridir. Bu nedenle, kullanıcının gerçekten tıklayacağı bir düğme tasarlamak önemlidir. İçinde metin veya herhangi bir şekil çizilen kare veya dikdörtgen kutular ile tasvir edilmektedir. Şekil 4.4'te ise oluşturulan veri kümesi içinde farklı buton nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.4. Buton nesnelerinin farklı çizimlendirilmesi.

4.3.3. Paragraf (Paragraph)

Bir metin bloğunun tasvir edilme şekli olarak paragraflar kullanılmaktadır. Paragraflar yeni bir satırda başlarlar ve kendisinden önce ve sonra boşluk eklenir. Sadece bir adet karalama tarzı çizgi, çok uzun metin veya çok satırlı karalama çizgileri ile tasvir edilebilir. Şekil 4.5'te bu gözlemler göz önünde bulundurularak farklı paragraf çizimlerine yer verilmiştir.



Şekil 4.5. Paragraf nesnelerinin farklı çizimlendirilmesi.

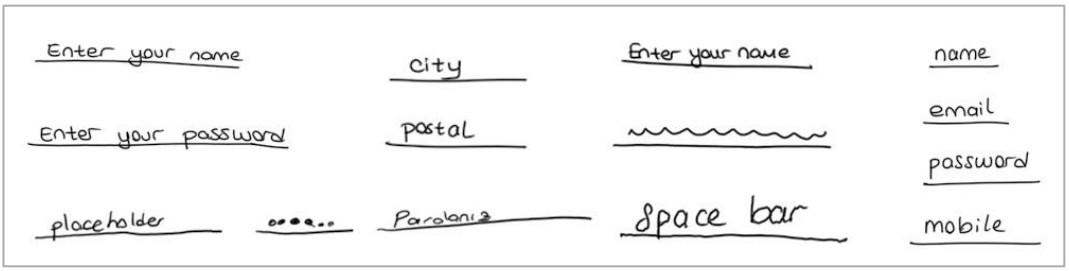
4.3.4. Başlık (Header)

HTML'nin altı farklı başlık etiketi vardır: H1'den H6'ya kadar numaralandırılarak kullanılır. H1 en önemli etiket olarak kabul edilir ve H6 en az önemli etikettir. Etiketler genellikle büyükten (veya en önemliden) en küçüğe (veya en az önemli) doğru biçimlendirilir. Yazı fontu olarak da H1 en büyük yazı karakteri büyüklüğüne sahip iken H6 en küçük yazı karakteri büyüklüğüne sahiptir. Şekil 4.6'da oluşturulan veri kümesi içinde farklı başlık nesnelerinin çizimlerine yer verilmiştir.

Şekil 4.6. Başlık nesnelerinin farklı çizimlendirilmesi.

4.3.5. Girdi (Input)

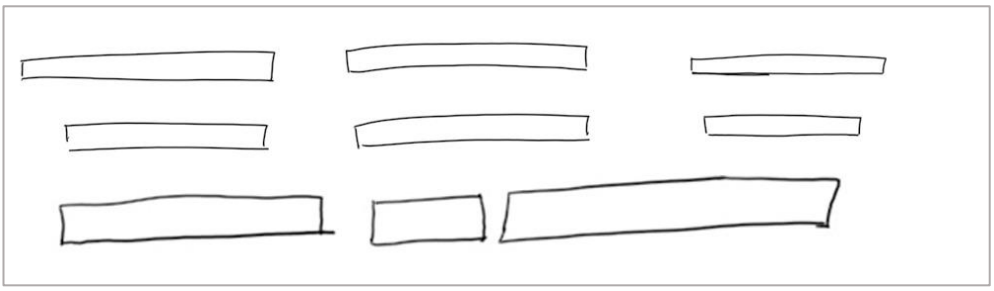
Giriş alanları, oldukça basit bir şekilde, kullanıcıların sisteme içerik girebildikleri alanlardır. Metin, parola, e-posta ve numara girileceği belirtilerek girdiler kısıtlanabilir. Kısa ve tek satır bilgiler için kullanılmaktadır. Şekil 4.7'de oluşturulan veri kümesi içinde farklı girdi nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.7. Girdi nesnelerinin farklı çizimlendirilmesi.

4.3.6. Metin Alanı (Textarea)

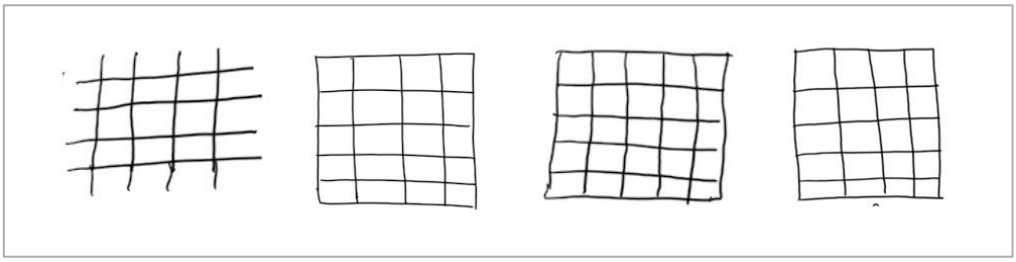
Metin alanı, girdi ile karıştırılabilir ancak girdiden farklı olarak çok satırlı metin giriş alanları olarak kabul edilmektedir. Metin alanlarında kullanıcının alt satıra geçebilmesine izin vermektedir. İçi boş dikdörtgen veya kare kutular ile tasvir edilmektedir. Şekil 4.8'de oluşturulan veri kümesi içinde farklı metin alanı nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.8. Metin alanı nesnelerinin farklı çizimlendirilmesi.

4.3.7. Tablo (Table)

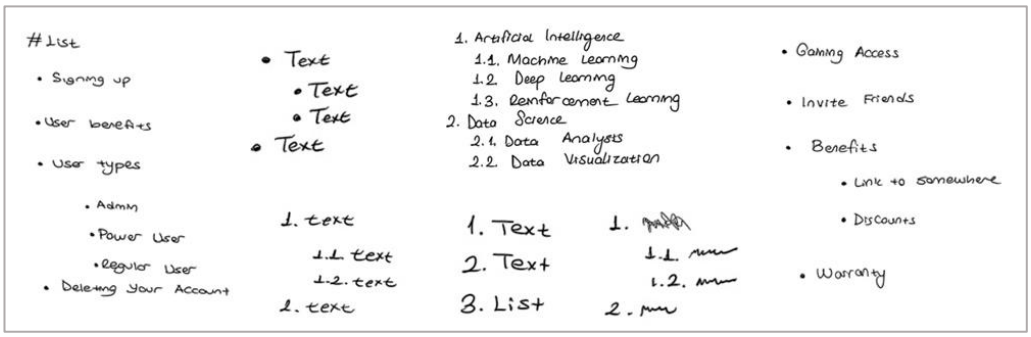
Kullanıcıya karmaşık bir veri seti sunulduğunda, veri yapısında birçok girişten oluşan veriler olduğunda, veriler arasında karşılaştırma ve sınıflandırma yapıldığında tablolar kullanılmaktadır. Tablolar baş ve gövde olarak ayrılmaktadır. Baş (table head – th), sütundaki içeriği açıklamaktadır. Sütun başlıkları, sütundaki metin ile aynı hizada bulunur. Gövde (table body – td) ise tablo içeriğini içerir. Şekil 4.9'da oluşturulan veri kümesi içinde farklı tablo nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.9. Tablo nesnelerinin farklı çizimlendirilmesi.

4.3.8. Liste (List)

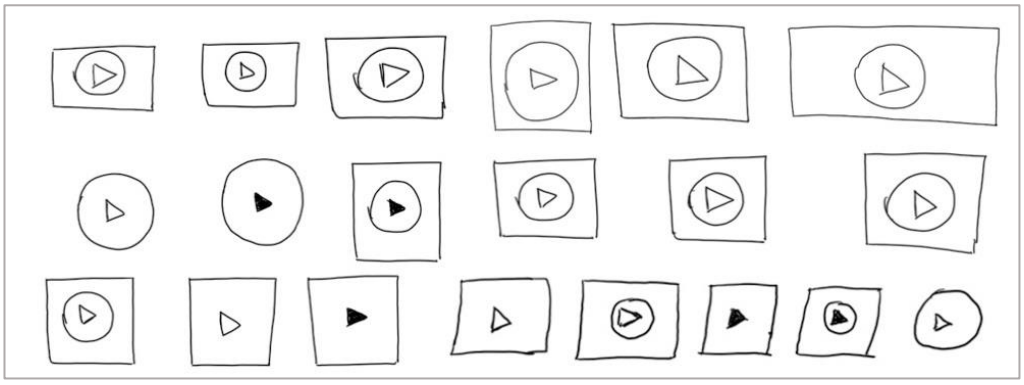
İç içe bilgi veya başlık içeren UI ögesidir. Kullanılan alanına göre numara veya sembol ile ifade edilebilir. Bu ifadeler kendi içinde de özelleştirilerek tasarıma uygun hale getirilebilir. Şekil 4.10'da oluşturulan veri kümesi içinde farklı liste nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.10. Liste nesnelerinin farklı çizimlendirilmesi.

4.3.9. Video

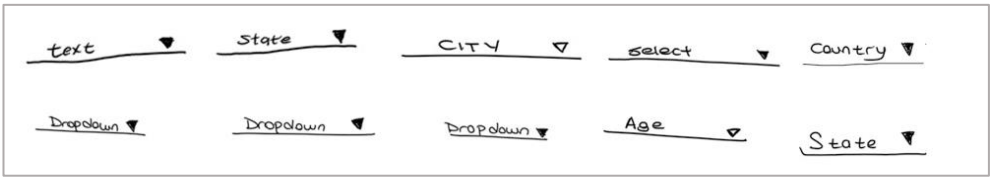
İçerisinde gömülü olarak video oynatılmasına izin veren UI ögesidir. Genellikle oynatma simgesiyle veya Youtube logosuna benzeyen bir çizim ile tasvir edilmektedir. Şekil 4.11'de oluşturulan veri kümesi içinde farklı video nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.11. Video nesnelerinin farklı çizimlendirilmesi.

4.3.10. Açılır Menü (Dropdown)

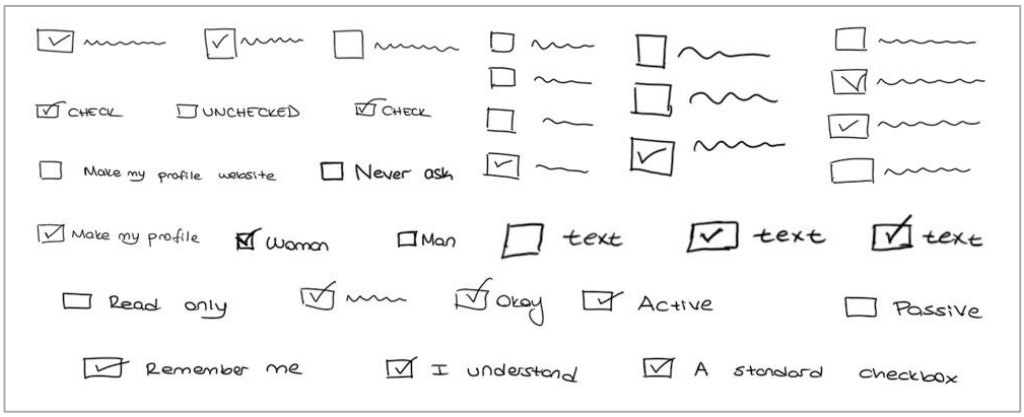
Kullanıcıların bir kez tıkladıklarında açılan bir listeden bir öge seçmesine olanak tanır. Kullanılan kütüphaneye göre açılır menülere çoklu öge seçme, listede arama yapabilme, listede seçilenlerin eksiltilmesi gibi farklı özellikler sağlanabilmektedir. Radyo butonlarından daha kompakttırlar. Ayrıca alandan tasarruf edilebilmeyi sağlarlar. Şekil 4.12'de oluşturulan veri kümesi içinde farklı açılır menü nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.12. Açılır menü nesnelerinin farklı çizimlendirilmesi.

4.3.11. Onay Kutuları (Checkbox)

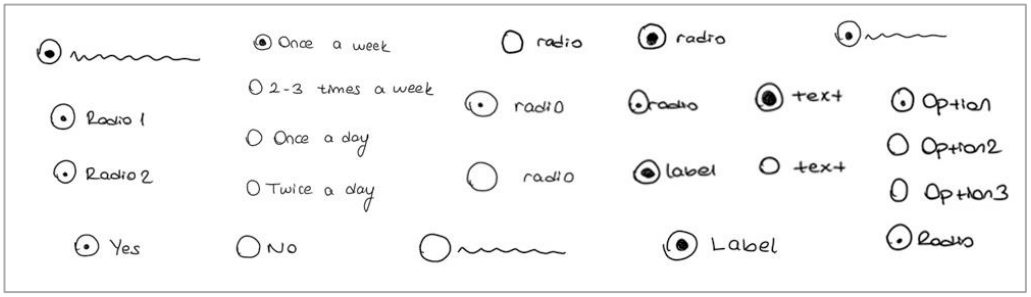
Onay kutuları, kullanıcının bir seçenek kümesinden bir veya daha fazla seçenek seçmesine olanak tanır ve her bir onay kutusu ayrı ayrı çalışır. Onay kutusunun işaretlenmesi, küçük bir onay işaretiyle (✓) belirtilmektedir. Şekil 4.13'te oluşturulan veri kümesi içinde farklı onay kutuları nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.13. Onay kutuları nesnelerinin farklı çizimlendirilmesi.

4.3.12. Radyo Butonlar (Radio Button)

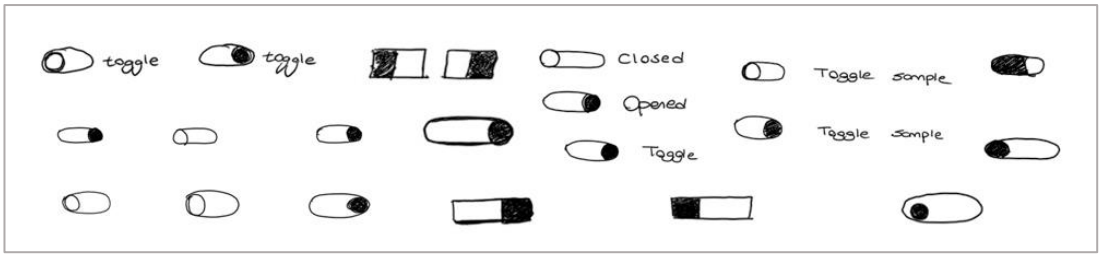
Genellikle onay kutularıyla karıştırılan radyo butonlar, kullanıcıların bir liste arasından bir seçenek seçmesine olanak tanıyan küçük dairesel ögelerdir. Buradaki anahtar, kullanıcıların onay kutularında olduğu gibi birden fazla seçeneği değil, yalnızca bir seçeneği seçebileceklerini belirtmektir. Şekil 4.14'te oluşturulan veri kümesi içinde farklı radyo buton nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.14. Radyo buton nesnelerinin farklı çizimlendirilmesi.

4.3.13. Geçişler (Toggle/Switch)

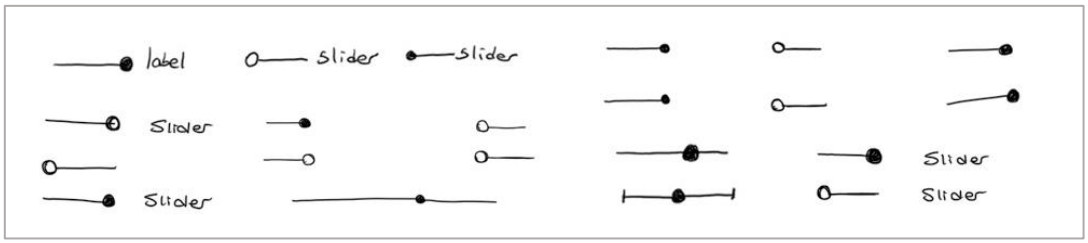
Geçişler, kullanıcının iki durum arasında bir görünüm/değer/ayar değiştirmesine izin verir. Açık ve kapalı durum arasında geçiş yapmak veya liste görünümü ile ızgara görünümü arasında geçiş yapmak için kullanışlıdır. Şekil 4.15'te oluşturulan veri kümesi içinde farklı geçiş nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.15. Geçiş nesnelerinin farklı çizimlendirilmesi.

4.3.14. Kaydırıcı (Slider)

Kaydırıcılar, bir değer veya bir değer aralığı seçmek için kullanılan yaygın bir UI ögesidir. Kullanıcı, kaydırıcıyı parmağıyla veya faresiyle sürükleyerek, alışveriş sırasında belirlenen aralığı istediği değere kademeli ve hassas bir şekilde ayarlayabilir.

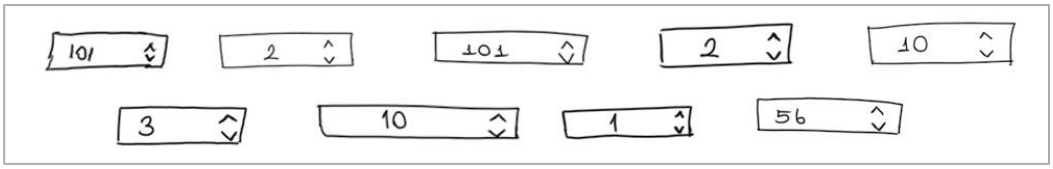


Şekil 4.16. Kaydırıcı nesnelerinin farklı çizimlendirilmesi.

Şekil 4.16'da oluşturulan veri kümesi içinde farklı kaydırıcı nesnelerinin çizimlerine yer verilmiştir.

4.3.15. Kademeler (Stepper Input)

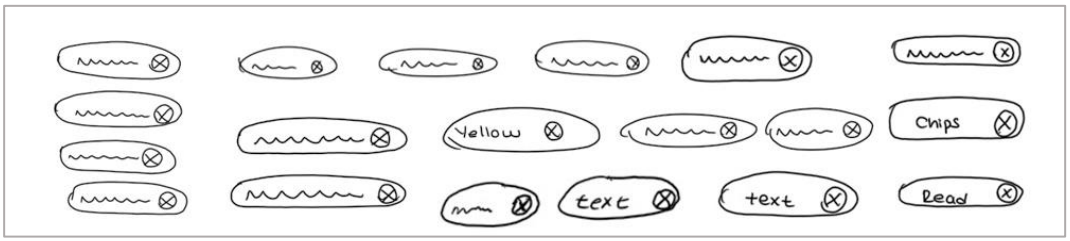
Kademeler, kullanıcıların bir değeri ayarlamasına da izin veren iki segmentli kontrollerdir. Ancak, kaydırıcılardan farklı olarak, kullanıcıların değeri yalnızca önceden tanımlanmış artışlarla değiştirmesine izin verir. Şekil 4.17'de oluşturulan veri kümesi içinde farklı kademe nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.17. Kademeler nesnelerinin farklı çizimlendirilmesi.

4.3.16. Çip (Chip)

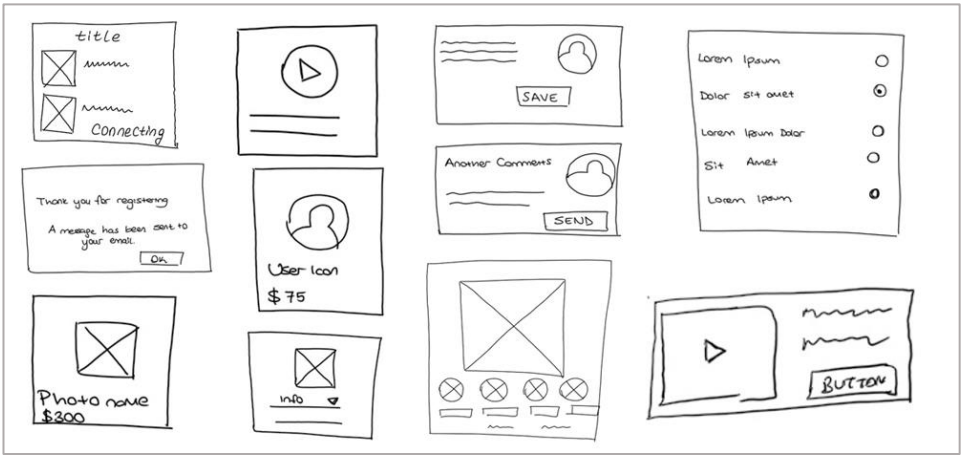
Küçük bilgi bloklarını temsil etmek için kullanılırlar. Yaygın olarak kişi ve etiketler için kullanılırlar. Şekil 4.18'de oluşturulan veri kümesi içinde farklı çip nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.18. Çip nesnelerinin farklı çizimlendirilmesi.

4.3.17. Kutu/Kart (Box/Card)

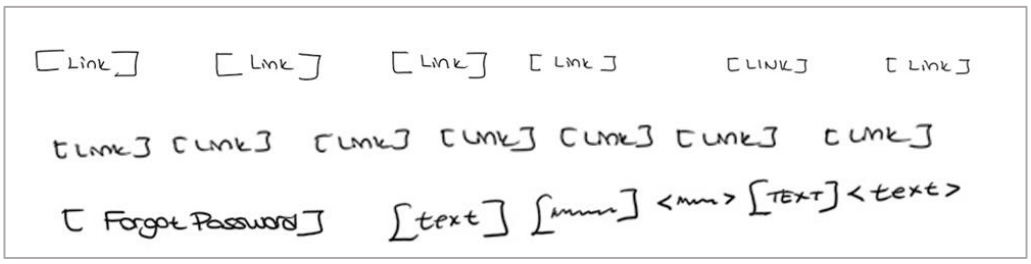
Son günlerde çok popüler olan kartlar, düğmeler, metinler, zengin medya vb. şeklinde farklı türde bilgiler içeren küçük dikdörtgen veya kare modüllerdir. Kartlar, kullanıcı için bir giriş noktası görevi görerek, kullanıcının daha sonra tıklayabileceği farklı içerik türlerini yan yana görüntüler. Kartlar, mevcut alanı akıllıca kullanmak ve kullanıcıya geleneksel bir listede kaydırma yapmadan birden fazla içerik seçenek sunulmak isteniyor ise doğru bir UI tasarım seçimidir. Şekil 4.19'da oluşturulan veri kümesi içinde farklı kutu nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.19. Kutu nesnelerinin farklı çizimlendirilmesi.

4.3.18. Bağlantı (Link)

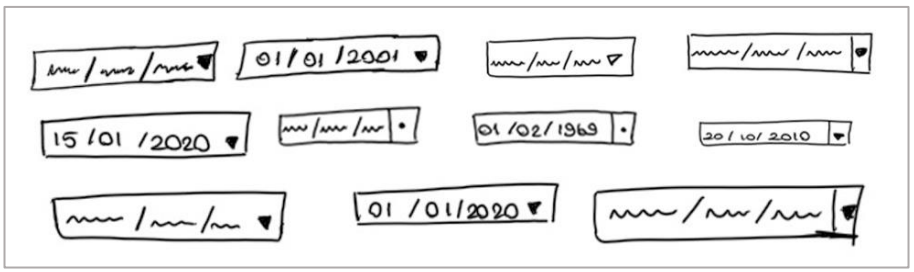
Bağlantılar, kullanıcıların bir sayfadan başka bir sayfaya veya uygulamaya geçmelerine olanak sağlayan UI ögeleridir. Taslak çizimlerinde ise yalnızca metinlerde gösterilebilmektedir. Ancak bağlantılar, resim ve buton gibi girdi olmayan UI ögelerinde de bulunabilir. Varsayılan olarak bir bağlantının üzerine gelindiğinde fare oku, küçük bir el simgesine dönüşmektedir. Şekil 4.20'de oluşturulan veri kümesi içinde farklı bağlantı nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.20. Bağlantı nesnelerinin farklı çizimlendirilmesi.

4.3.19. Tarih ve Saat Seçiciler (Date picker)

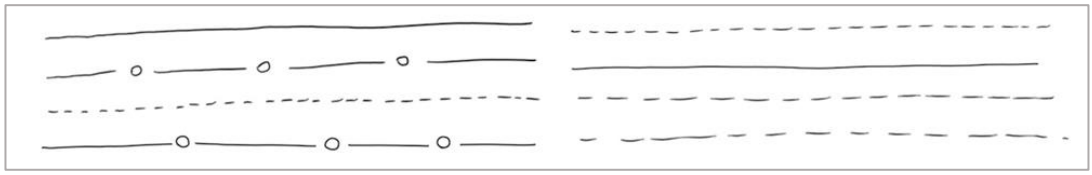
Tarih ve saat seçiciler, kullanıcıların tarih ve saatleri seçmesine olanak tanır. Seçicileri giriş alanları üzerinden kullanmanın avantajı, kullanıcıların girdiği tüm verileri düzenli bir şekilde veri tabanında tutmayı sağlar, bu da bilgileri yönetilebilir ve erişimi kolay hale getirmektedir. Şekil 4.21'de oluşturulan veri kümesi içinde farklı tarih ve saat seçiciler nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.21. Tarih ve saat seçici nesnelerinin farklı çizimlendirilmesi.

4.3.20. Satır Sonları (Line Break)

UI ögelerini veya bölümleri ayırmak için kullanılan ayraçlardır. Genellikle bölümün en solundan, sağına kadar çizgi ile ifade edilmektedir. Şekil 4.22'de oluşturulan veri kümesi içinde farklı satır sonu nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.22. Satır sonu nesnelerinin farklı çizimlendirilmesi.

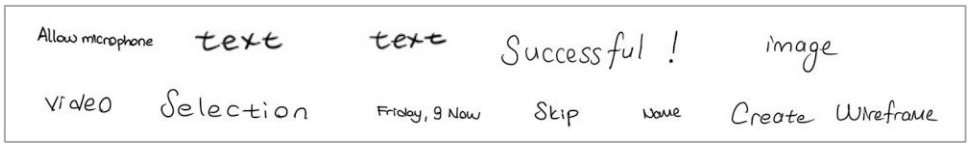
4.3.21. Değerlendirme/Derecelendirme (Rating)

Çoğunlukla yıldız ve kalp simgesiyle tasvir edilen derecelendirme ögeleridir. Derecelendirme işlemleri de bir girdi olarak sayılmaktadır. Kullanım durumuna göre 5 ila 10 adetten oluşabilir. Kullanıcıların değerlendirme sorularına, radyo buton veya onay kutusu yerine farklı şekillerle temsil edilen bir ölçekte sıralamalarına izin veren bir derecelendirme sorusu türüdür. Örneğin; kullanıcıların içerik ve hizmetlerden memnuniyet derecelerini yansıtan derecelendirmeleri 5 adet yıldız veya kalp ile yapılması farklı tip örnekler olarak verilebilir. Şekil 4.23'te oluşturulan veri kümesi içinde farklı değerlendirme nesnelerinin çizimlerine yer verilmiştir.

Şekil 4.23. Değerlendirme nesnelerinin farklı çizimlendirilmesi.

4.3.22. Metin (Text)

Yalnızca az sayıda metnin yer aldığı alanlarda kullanılmaktadır. Başlıklar metin olarak geçmemektedir. Uzun metinler için paragraflar tercih edilmektedir. Şekil 4.24'te oluşturulan veri kümesi içinde farklı metin nesnelerinin çizimlerine yer verilmiştir.



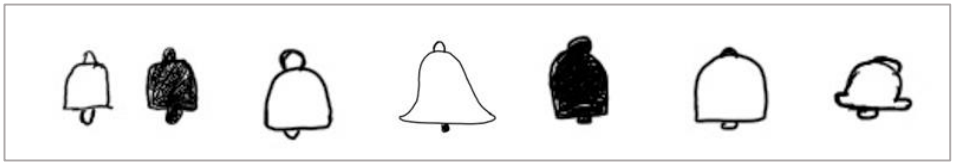
Şekil 4.24. Metin nesnelerinin farklı çizimlendirilmesi.

4.3.23. Simgeler/İkonlar (Icon)

Simgeler, kullanıcılara çeşitli şeyleri iletmek için kullanılan resimlerdir. İçeriği daha iyi iletmeye yardımcı olabilirler veya belirli bir eylemi iletebilir ve tetikleyebilirler. Simgeler genellikle <i> etiketinin ile gösterilmektedir. Kullanılan kütüphaneye göre çağırılmaları farklılık gösterebilir. Simgeler bildirim ve telefon, e-posta olarak gruplandırılmıştır.

4.3.23.1. Bildirim

Kullanıcılar bildirim ikonunu genellikle çana benzeyen bir çizim ile tasvir etmektedirler. Bu sebeple içi boş ve dolu bir çan çizimi üzerinden çalışılmıştır. Şekil 4.25'te ve Şekil 4.26'da oluşturulan veri kümesi içinde farklı simge nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.25. Bildirim simgelerinin farklı çizimlendirilmesi.

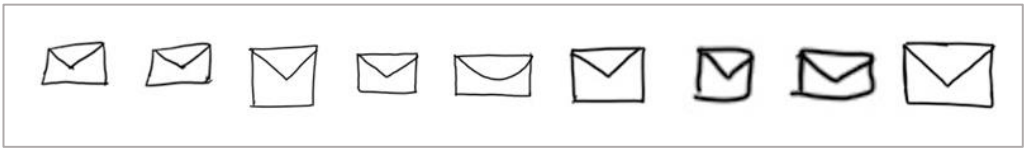
4.3.23.2. Telefon

Telefon simgesi, akıllı cihazlardaki tuş sırasıyla, cep telefonu veya ankesörlü telefon ile tasvir edilebilmektedir. Çalışmamızda içi boş ve dolu şekilde ankesörlü telefona benzeyen bir çizim üzerinde tespit yapılmıştır.

Şekil 4.26. Telefon simgelerinin farklı çizimlendirilmesi.

4.3.23.3. E-posta

E-posta simgesi, kapalı veya açık zarf ve posta kutusu ile tasvir edilebilmektedir. Çalışmamızda kapalı zarf üzerinde çizimler kullanılarak veriler hazırlanmış ve tespit edilmiştir. Şekil 4.27'de oluşturulan veri kümesi içinde farklı e-posta nesnelerinin çizimlerine yer verilmiştir.



Şekil 4.27. E-posta simgelerinin farklı çizimlendirilmesi.

BÖLÜM 5

GÖRÜNTÜ İŞLEME

Dijital görüntü işleme, temel olarak görüntüsel veriler üzerinde çalışmaktadır. Görüntü işleme, gelişmiş bir görüntü elde etmek veya ondan bazı yararlı bilgiler çıkarmak için bir görüntü üzerinde bazı işlemleri gerçekleştirme yöntemidir. Girişin bir görüntü olduğu ve çıkışın görüntü veya o görüntüyle ilişkili karakteristikler/özellikler olabileceği bir sinyal işleme türüdür. Günümüzde görüntü işleme hızla gelişen teknolojiler arasında yer almaktadır. Mühendislik ve bilgisayar bilimleri disiplinlerinde de temel araştırma alanını oluşturur. Özetlemek gerekirse, görüntü işleme sistemleri daha önceden belirlenmiş sinyal işleme yöntemlerini uygulayarak görüntüleri iki boyutlu sinyaller olarak ele almaktadır [65]. Bu şekilde mevcut görseller üzerinde birtakım değişiklikler yapılarak verinin analizi sağlanmaktadır. Görüntünün belirli filtrelerden geçirilerek matematiksel işlemler uygulanması sonucunda çıktıda kimi zaman görüntünün değiştirilmiş hali ile karşılaşılır. Düşük seviye ile çalışılan görüntü işleme sistemlerinde görüntünün iyileştirilmesi, görüntünün yumuşatılması ve görüntünün keskinleştirilmesi yapılırken yüksek seviyeli görüntü işleme sistemlerinde ise orta seviyede çıkarılan özellik haritalarındaki özellikler sisteme verilerek görüntünün analizi yapılmaktadır. Bu bölümde, tez çalışması sırasında kullanılacak verilerin daha iyi tespit edilmesi için araştırılan belirli görüntü işleme teknikleri açıklanmaktadır. Ek olarak, çalışmanın görüntü işleme adımları için Python programlama dili ve OpenCV [66] kütüphanesi tercih edilmiştir.

Görüntü iyileştirme, sonuçların görüntüleme veya daha fazla görüntü analizi için daha uygun olması için dijital görüntüleri ayarlama işlemidir. Örneğin, temel özellikleri tanımlamayı kolaylaştırmak için paraziti kaldırabilir, bir görüntü keskinleştirebilir veya parlaklaştırılabilir. Bu çalışmada kullanılan taslak çizim görüntüleri, dijital ortamda oluşturulduğu için kâğıt üzerinde oluşturulan çizimlere göre daha az dijital gürültü içermektedir. Şekil 5.1'de görüldüğü üzere dijital ortamda oluşturulmuş bir taslak çizim yer almaktadır. Bu taslak çizimde oluşturulan resim ve buton nesnelerinde insan çizimi faktörü büyük rol oynadığı için dalgalanmalar mevcuttur. Bu gibi durumlarda görüntü üzerinde adaptif eşik filtrelemesi uygulanarak tüm pikseller için

basit eşik filtresi kullanılmaktadır. Ardından OpenCV kütüphanesinde var olan **THRESH_BINARY_INV** modülü ile görüntünün tersi alınmıştır.

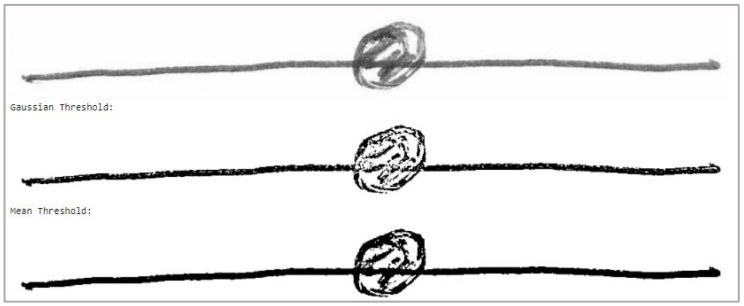
Şekil 5.1. Veri kümesinden alınan örnek bir taslak çizim görseli.

Adaptif eşik filtrelemesinde, farklı eşik değerleri kullanılarak daha küçük bölgeler için eşik değerleri hesaplatılmaktadır [67]. Adaptif eşik filtreleme, bir görüntü üzerinde eşikleme yaparken **adaptiveMethod** adlı parametresi ile Gauss veya Ortalama (Mean) filtreleme kullanılacağını belirtir. Adaptif eşiklemede öncelikle taslak çizimlerde yer alan UI ögelerinin kırpılması sağlanmıştır. Buna göre öncelikle bazı UI ögelerinin yer aldığı küçük görsellerde görüntünün ön işleme adımları sağlanmıştır. Şekil 5.2'de sırasıyla adaptif gauss eşikleme ve adaptif ortalama eşik filtresi uygulama kodları gösterilmektedir. Burada yer alan 255 parametresi görüntünün arka plan rengini değiştirmektedir. 255 seçildiği için görüntünün arka planı beyaz olarak kaydedilmektedir. Eğer 128 gibi bir değer seçilseydi arka plan gri olacaktı ve UI ögelerinin tespit edilmesi zorlaşacaktı. Bu sebeple kodda 255 olarak parametre ayarlanmıştır.

```
#Grayimage = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gaussianTH = cv2.adaptiveThreshold(image,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
          cv2.THRESH_BINARY,25,15)
meanTH = cv2.adaptiveThreshold(image,255,cv2.ADAPTIVE_THRESH_MEAN_C,\
         cv2.THRESH_BINARY,25,15)
print('Gaussian Threshold:')
cv2_imshow(gaussianTH)
print('Mean Threshold:')
cv2_imshow(meanTH)
```

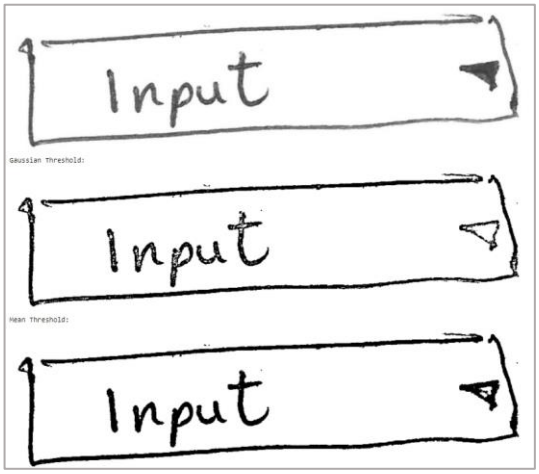
Şekil 5.2. Adaptif gauss ve adaptif ortalama eşik filtreleme Python kodları.

Şekil 5.3'te sırasıyla orijinal taslak çizim görseli, gauss eşikleme sonucu ve ortalama eşik sonuçları gösterilmektedir.



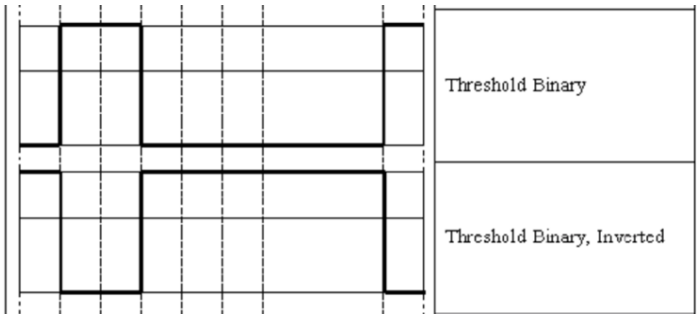
Şekil 5.3. Slider ögesinin görüntü işleme aşamalarından geçirilmiş hali.

Şekil 5.4'te verilen görselde ise taslak çizim örneğinden alınmış girdi ögesi yer almaktadır. Bu görsel sırasıyla incelendiğinde, orijinal görüntü, gauss eşiklenmiş görüntü ve ortalama filtre uygulanmış görüntü yer almaktadır. Buna göre belirli eşik değerinde gauss uygulanmış bir taslak çizim ögesinin üzerine ortalama filtreleme uygulanırsa daha verimli sonuçların elde edileceği sonucu çıkarılmıştır.



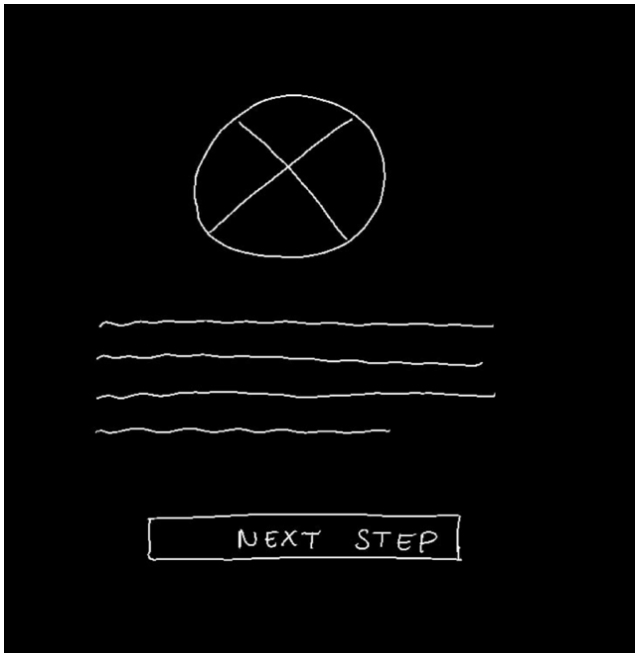
Şekil 5.4. Girdi (Input) ögesinin görüntü işleme aşamalarından geçirilmiş hali.

Veri kümesinde yer alan taslak çizim görüntüleri, yukarıda bahsi geçen ön işleme adımları tamamlandıktan sonra son adım olan görüntünün tersini alma işlemine geçilmiştir. Bu adımda orijinal görüntülerde beyaz olan arka plan ve UI ögelerinin çizimi sonucu oluşan siyah çizgiler tam tersi renk tonuna bürünmüştür. Bu adımın daha açıklayıcı olması için Şekil 5.5'te eşiklenmiş ikili görüntü ve tersi alınmış görüntü sinyallerinin çizimi gösterilmektedir.



Şekil 5.5. İkili eşik ve tersine çevrilmiş ikili eşik işlem tipleri [68].

İkili olarak düşünüldüğünde 0 olan pikseller 1, 1 olan piksel değerleri ise 0 olarak tersine çevrilmiştir. Şekil 5.6'da ise önceki görüntü işleme adımlarından geçirilmiş taslak çizimin tersini alma işlemi yapıldıktan sonraki haline yer verilmiştir.



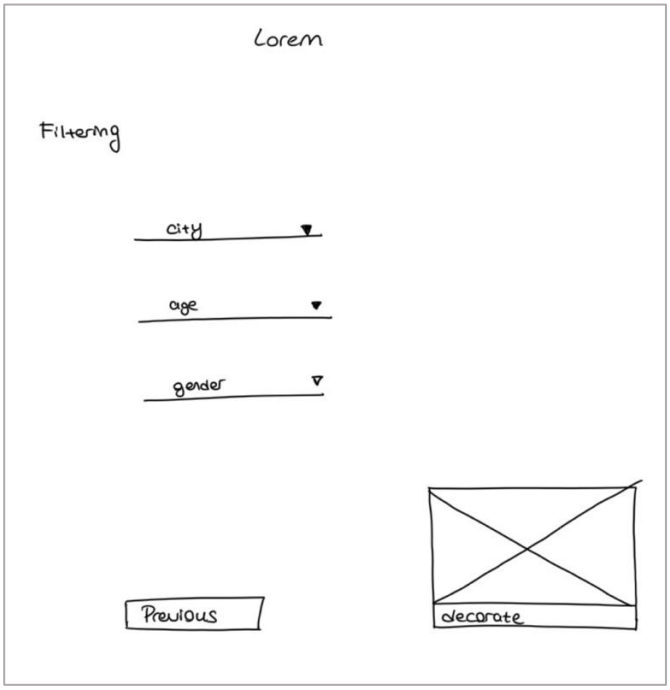
Şekil 5.6. Örnek taslak çizimin tersini alma işlemi ile elden edilen görsel sonucu.

Şekil 5.6'da yer alan görüntünün bu adıma gelmesi için OpenCV'nin **THRESH_BINARY_INV** fonksiyonu kullanılmıştır. Bu fonksiyonun hesaplatılmasında kullanılan formüle Eşitlik 5.1'de yer verilmiştir.

$$dst(x,y) = \begin{cases} 0 & if\ src(x,y) > thresh \\ \text{maxval} & otherwise \end{cases} \tag{5.1}$$

Eşitlik 5.1'de verildiği üzere *dst(x,y)* değişkeni girdi görüntüsünün formül uygulandıktan sonraki mesafe çıktısına eşittir. Eğer piksel kaynağı verilen eşik değerinden büyük ise çıktı 0 olarak alınır, değil ise diğer durumlarda *maxvalue* olarak nitelendirilen en büyük değeri almaktadır.

Şekil 5.7'de yer alan UI ögeleri son görüntü işleme adımından geçirilmesi ile elde edilen daha belirgin çizgiler sayesinde istenilen sonuca ulaşılmıştır. Veri kümesinden alınan taslak çizim görüntüleri bu adımla birlikte sinir ağı modeline verilecek duruma getirilmiştir. Böylelikle görüntünün iyileştirilmesi sağlanarak yapay sinir ağının daha yüksek performans ile nesne tespiti yapabilmesi sağlanmıştır. Şekil 5.7 ve Şekil 5.8'de görüntülerin ön işleme adımından geçirilmesinin neden önemli olduğuna dair görseller yer almaktadır. Şekil 5.7'de yer alan taslak çizim incelendiğince içerisinde yer alan metin alanlarındaki yazılar ve *dropdown* ögeleri Şekil 5.8'deki sonuç görüntüsüne göre daha yumuşak kenarlı kalmaktadır. Şekil 5.8'de siyah çizgilerin beyaza, beyaz çizgilerin ve arka planın ise siyaha dönüştürülmesiyle nesneler ön plana çıkarılmıştır.

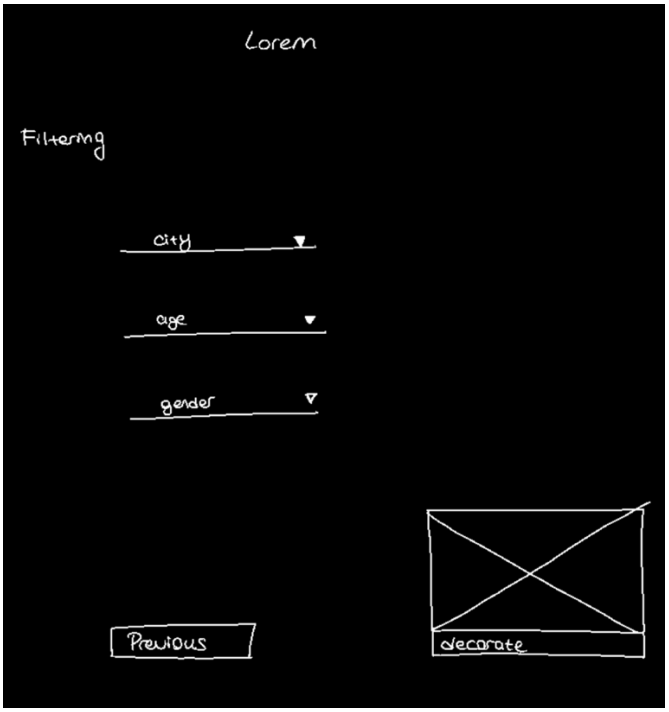


Şekil 5.7. Tersine çevrilmemiş bir taslak çizim örneği.

Şekil 5.7 incelendiğinde, taslak çizimin içerisinde farklı çeşitlerde UI ögeleri barınmaktadır. Örneğin içerisinde *city, age* ve *gender* yazılı UI ögelerinin birer dropdown nesnesi olduğu bilinmektedir. Bununla birlikte bir adet previous yazılı *buton, box* adı verilen UI ögeleri ve çeşitli metin alanı nesneleri mevcuttur. Şekil 5.8'de ise bu nesnelerin Eşitlik 5.1'deki formülün kullanımı ile ters ikili eşikleme yapılmıştır. Bu işlem **BINARY_INV** adı verilen ikili eşiklemenin tam tersi şekilde çalışmaktadır. Bu işlem için oluşturulan Python kodunun genel yapısı şu şekildedir.

th, dst = cv2.threshold(src,thresh, maxValue, cv2.THRESH_BINARY_INV)

Şekil 5.8'de ise bu görüntünün ön işleme adımlarının son adımı tamamlanmıştır. Böylelikle bir sonraki bölüm olan derin öğrenme adımına geçilirken taslak çizim görüntüleri gerektiği gibi temiz ve hazırlıklı verilecektir. Bu bölümden itibaren taslak çizim görüntü verileri modele verilmek üzere sinir ağı modeline iletilmiştir.



Şekil 5.8. Verilen örnekteki taslak çizimin tersi alınmış son durumu.

BÖLÜM 6

DERİN ÖĞRENME

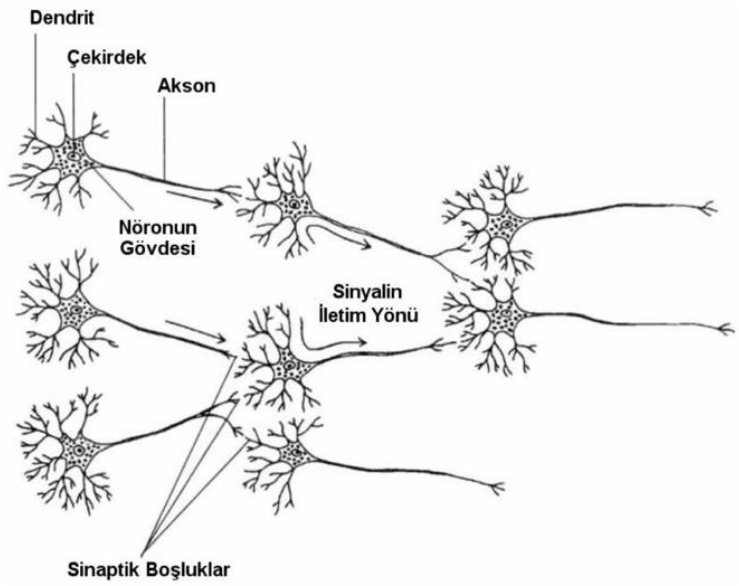
Yapay zekâ, 1950'li yıllardan itibaren gelişmeye başlayan ve bilgisayarların düşünebilme kabiliyeti kazanabilmesi için çaba gösterilen bir bilişim alanıdır. Doğal bir insan zekâsı ve sinir hücreleri baz alınarak ortaya çıkarılmıştır [69]. Makine öğrenimi, genel olarak, performansı artırmak veya doğru tahminler yapmak için deneyimi kullanan hesaplama yöntemleri olarak tanımlanmaktadır. Burada deneyim, tipik olarak toplanan ve analiz için hazır hale getirilen elektronik veri şeklini alan, öğrenen için mevcut olan geçmiş bilgileri ifade eder. Bu veriler, dijitalleştirilmiş insan etiketli eğitim setleri veya çevre ile etkileşim yoluyla elde edilen diğer bilgi türleri şeklinde olabilir. Bir öğrenme problemine örnek olarak, görünmeyen belgelerin konusunu doğru bir şekilde tahmin etmek için her biri bir konu ile etiketlenmiş, rastgele seçilmiş belgelerin sonlu bir örneğinin nasıl kullanılacağı gösterilebilir. Açıkçası, örnek ne kadar büyükse, görev o kadar kolay olur. Ancak görevin zorluğu, göreve atanan etiketlerin kalitesine de bağlıdır [70].

Derin öğrenme, içeriğinde birçok yapay sinir ağları içermektedir. Derin öğrenme projelerinde, var olan her katmanda yapay sinir ağlarının aktarımı gösterilmektedir. Birbiriyle yakından bağlantılı ağda bir araya getirildiğinde bir dizi işlem birimi, biyolojik sinir ağının bazı özelliklerini sergileyen şaşırtıcı derecede zengin bir yapı sunar. Bahsedilen bu zengin yapıya "**Yapay Sinir Ağları (YSA)"** denir [71].

6.1. YAPAY SİNİR AĞLARI

Sinir ağları bilgi işlemede yavaştır. En gelişmiş bilgisayarlar için yürütmeye karşılık gelen döngü süresi, merkezi işlem birimindeki bir programın bir adımı aralığındadır. Bir sinir olayına karşılık gelen döngü süresi milisaniye aralığında bir dış uyaran tarafından harekete geçirilir. Böylece bilgisayar YSA sayesinde bilgiyi neredeyse bir

milyon kat daha hızlı işler. Sinir ağları büyük ölçüde paralel işlemler gerçekleştirebilir. Çoğu programın çok sayıda talimatı vardır ve bunlar geleneksel bir bilgisayarda birbiri ardına sıralı bir modda çalışırlar. Öte yandan, beyin, her biri nispeten daha büyük olan, büyük ölçüde paralel işlemlerle çalışır. Bu durum, bilginin bilgisayar tarafından işlenmesine kıyasla birkaç büyüklük sırası daha yavaş olmasına rağmen, belirli görevler için insan bilgi işlemenin üstün performansını açıklar [71].



Şekil 6.1. Tipik bir biyolojik sinir hücresinin yapısı [72].

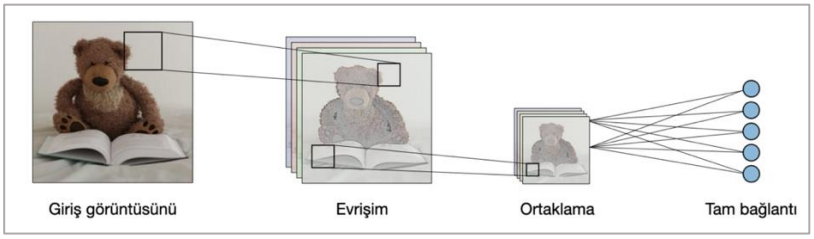
Sinir ağları çok sayıda hesaplama elemanına sahiptir ve hesaplama nöronlarla sınırlı değildir. Şekil 6.1'de görüldüğü üzere tipik bir sinir hücresinin soma, akson ve dendrit bölgesi gibi farklı bölümleri mevcuttur. Bir beyindeki nöron (sinir hücresi) sayısının yaklaşık 1011 ve toplam ara bağlantı sayısının 1015 olduğu tahmin edilmektedir. Beyne bizim yapamadığımız karmaşık örüntü tanıma görevlerini gerçekleştirme gücünü veren, bağlantıların bu boyutu ve karmaşıklığıdır. Beynin karmaşıklığı, hesaplamanın yalnızca hücre gövdesi veya soma içinde değil, aynı zamanda dendrit ve sinapslarda da gerçekleşmesi gerçeğiyle daha da karmaşıklaştırmaktadır [71]. Bir beyin hücresinde, elektrik akımını alan dendrit hücresi bu elektrik akımını hücre gövdesine gönderir. Böylelikle bir sinir hücresinin yaşam döngüsü tamamlanmış olur.

6.2. EVRİŞİMLİ SİNİR AĞLARI (CNN)

Evrişimli sinir ağları, görüntüler üzerinde çalışan yapay sinir ağları türündendir. Evrişimli sinir ağları, yapay sinir ağlarından esinlenerek oluşturulmuş ve toplanmış bilgileri uçtan uca öğrenebilen bir mimaridir. Evrişimli sinir ağları, yeterli kapasitesi ve akıllı model yapısı sayesinde büyük ölçekli verilerin de üstesinden gelebilmektedir. Evrişimli sinir ağlarının dezavantajları ise eğitim sürecinin uzun olması ve eğitim sürecinde yerel bir çözüme takılma ihtimalidir [73]. Evrişimli sinir ağları, yapay sinir ağlarının ileriye dönük işlem yapan ve sinir ağlarından farklı olarak özellik çıkarıcı bir katmanın bulunduğu derin öğrenme yaklaşımıdır. Evrişimli sinir ağlarında iki temel katman bulunmaktadır. Bunlardan biri evrişim diğeri ise havuzlama katmanıdır. Evrişimli sinir ağları, bu iki katmanda temel bazı işlemler ile görüntünün önemli olan özelliklerini çıkarmayı amaçlamaktadır [74].

6.2.1. Evrişim Katmanı

Evrişim katmanında, girdi olarak alınan görüntülerin üzerinde filtresel olarak evrişim işlemi uygulanır. Evrişim işlemi, 3 boyutlu tensörler üzerinde işlemini gerçekleştirir.

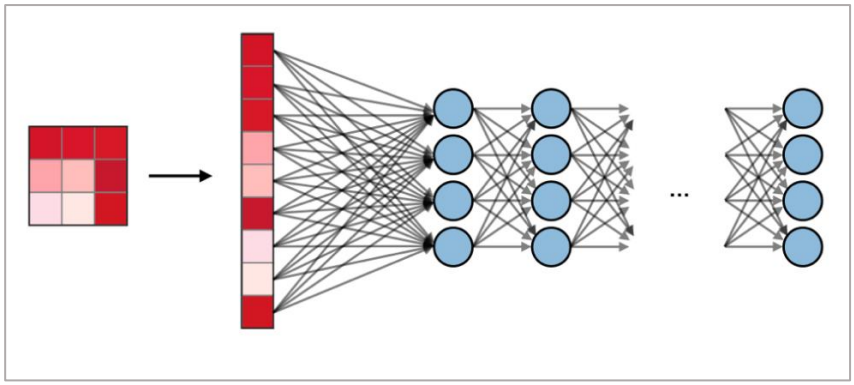


Şekil 6.2. Evrişimli sinir ağlarının örnek bir giriş görüntüsünde çalıştırılması [75].

Şekil 6.2'de görüldüğü üzere örnek bir giriş görüntüsü evrişimli sinir ağlarına girdi olarak verilir. Ardından görüntü üzerinde evrişim işlemi gerçekleştirilerek nesneye ait özellikler çıkartılır. Bu özellikler, nesne için ayrılan küçük bilgiler ile görüntü için önemli detaylar sağlamaktadır. Daha sonra ise özellik haritaları ve bilgiler tam bağlı katmana iletilir ve diğer işlemler gerçekleştirilir.

6.2.2. Tam Bağlı Katman

Tam bağlı katman, her girişin tüm sinir hücrelerine bağlı olduğu bir giriş üzerinde çalışmaktadır. Eğer tam bağlı katman var ise bu katmanlar genellikle CNN mimarisinin sonuna doğru bulunur ve sınıf skorları gibi hedefleri optimize etmek için kullanılabilmektedir [75].



Şekil 6.3. Evrişim işlemi sonrası tam bağlı katman gösterimi [74].

Şekil 6.3'te evrişim işleminden geçirilen verinin düzleştirilmesi sonrası tam bağlı katmana gönderildiği gösterilmektedir. Tam bağlı katmanda, tüm sinir hücreleri Şekil 6.3'teki gibi birbirine açıkta kalmayacak şekilde bağlanmaktadır.

6.3. GÖRÜNTÜ SEGMENTASYONU

Görüntü segmentasyonu, bir görüntüyü birden çok segmente ayırma işlemidir. Görüntü segmentasyonu tipik olarak görüntülerdeki nesneleri ve sınırları bulmak için kullanılır. Bilgisayarlı görü için oldukça önemli olan görüntü segmentasyonu, bir görüntünün bölgelerinin anlamlı ve ayrık olduğu bazı kriterlere göre bölgelere ayrılması olarak tanıtılmaktadır. Görüntü segmentasyonu genellikle bazı örüntü tanıma uygulamalarının bir ara adımı olarak kabul edilir. Kenar tabanlı görüntü segmentasyonu teknikleri, bir giriş görüntüsündeki kenarları algılamayı amaçlar. Böylece giriş görüntüsünde bölge sınırlarının belirlenmesi ile segmentasyon yapılır. Bölge tabanlı görüntü segmentasyon teknikleri ise başlangıçta giriş görüntüsünde bazı

tohum noktaları arar ve nesnelerin sınırlarına ulaşmak için uygun bölge büyütme yaklaşımları kullanılır [76].

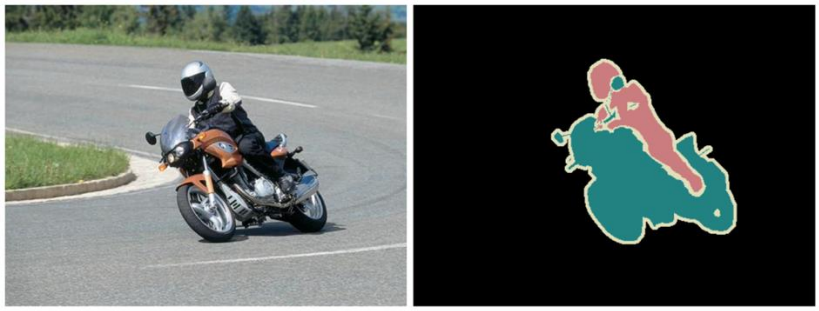
Şekil 6.4. Görüntü segmentasyonu gerçekleştirilmiş örnek bir görüntü [77].

Taslak çizim ögelerini sınıflandırmak için öncelikle tespit edilmeleri gerekir. Bir taslak çizimi birçok eleman içerecektir ve bu nedenle eleman sınırlarını tespit etmek için bir yöntem gereklidir [17]. Ögelerin sınırlarını belirlerken farklı tespit yöntemleri mevcuttur. Şekil 6.4'te örnek bu yöntemlerden birisi olan ve bir görüntüde yer alan nesnelerin görüntü segmentasyonu sonucuna yer verilmiştir. Bu tez çalışmasında, nesne tespiti yöntemleri için görüntü segmentasyonu tercih edilmiştir.

6.3.1. Semantik Segmentasyon

Semantik segmentasyon, arka plan dahil olmak üzere farklı semantik sınıflardaki nesneler arasında ayrım yapmada modele yardımcı olurken her pikselin karşılık gelen merkezine doğru yönünü tahmin ederek, aynı semantik sınıfın örneklerini ayırmaya izin vermektedir [78]. Arka plan sınıfı da dahil olmak üzere piksel bazında sınıflandırma bilgisini kodlayan semantik segmentasyon tahmini, farklı semantik sınıflardaki nesneleri (örneğin, kişi ve arka plan) ayırt etmek için benimsenir ve

böylece yinelenen arka plan kodlamasını kaldırır. Ek olarak, aynı anlamsal etiketin nesne örneklerini ayırmak için yön tahmini kullanılır [79].

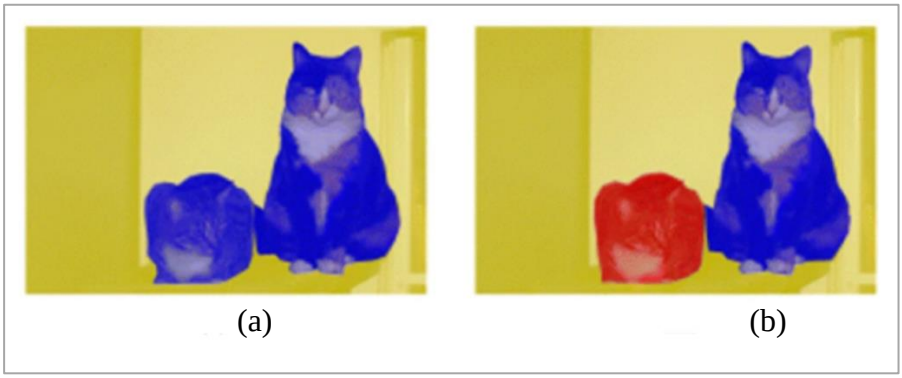


Şekil 6.5. Örnek bir görüntünün semantik segmentasyondan geçirilmiş sonucu [80].

Şekil 6.5'te örnek bir semantik segmentasyonu sonucuna yer verilmektedir. Şekil 6.5 incelendiğinde motosiklet ve insan nesnesi sadece nesneye ait olduğu maske rengine boyanmıştır. Bunun sebebi, semantik segmentasyonun anlamsal olarak nesneleri aynı renge boyamasından kaynaklanmaktadır.

6.3.2. Örnek Segmentasyon ve Mask R-CNN

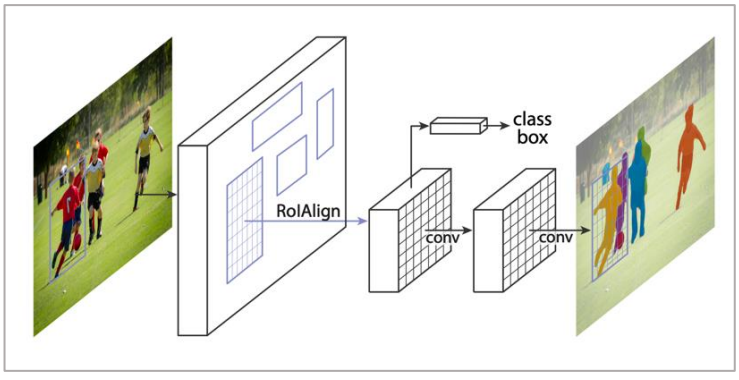
Örnek segmentasyon, segmentlere ayrılmış nesnelerin genellikle maskelerle temsil edildiği piksel düzeyinde bir görüntüdeki ilgilenilen nesneleri lokalize etme görevidir. Bu görev hem nesne tespiti hem de anlamsal bölümleme ile yakından ilgilidir [81].



Şekil 6.6. Segmentasyon türleri. a) Semantik segmentasyon. (b) Örnek segmentasyon [82].

Şekil 6.6'da bir görüntünün semantik segmentasyon ve örnek segmentasyon sonuçlarına yer verilmektedir. Buna göre semantik segmentasyon örneği için kedi nesnesi mavi ve arka plan sarı olarak boyanırken örnek segmentasyonda her bir kedi nesnesi farklı renge arka plan ise sarıya boyanmaktadır. Taslak çizimlerde oluşturulan her bir UI ögesi tespit açısından farklı maske renklerinde çıktıya sahip olmalıdır. Bununla birlikte, evrişimli sinir ağları türlerinden olan Mask R-CNN, görüntüsel verilerde örnek segmentasyon gerçekleştirmektedir. Bu sebeple çalışma için öngörülen segmentasyon türü örnek segmentasyon ve sinir ağı ise Mask R-CNN olarak seçilmişir.

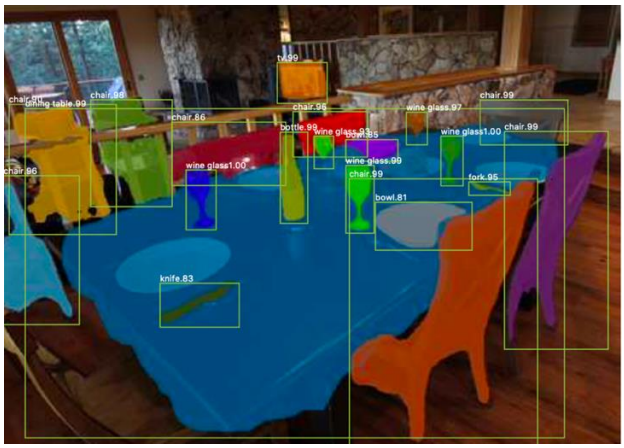
Mask-RCNN, örnek segmentasyon problemini doğrudan ele almak için tasarlanan ve daha sonra ağın sayısız hiper parametresini düzeltmeye yönelik çabalar hedefleyen evrişimli sinir ağı türüdür [83]. Mask-RCNN, nesne algılama, nesne lokalizasyonu ve doğal görüntülerin nesne örnek segmentasyonu için yakın zamanda önerilen son teknoloji bir algoritmadır. Prensipte Mask R-CNN, Faster R-CNN'nin [83] sezgisel bir uzantısıdır, ancak iyi sonuçlar için maske dalını doğru şekilde oluşturmak çok önemlidir. En önemlisi, Faster R-CNN, ağ girişleri ve çıkışları arasında pikselden piksele hizalama için tasarlanmamıştır. Bu en çok, örneklerle ilgilenmek için fiili çekirdek işlemi olan ilgi bölgesi havuzlamanın özellik çıkarımı için kaba uzamsal nicelemeyi nasıl yaptığında belirgindir [84].



Şekil 6.7. Mask R-CNN modeli [84].

Mask R-CNN sonucunda örnek segmentasyon gerçekleştirilerek Faster R-CNN ağına ek olarak kaliteli maskeler oluşturulmaktadır. Şekil 6.7'de Mask R-CNN'nin orijinal makalesinden alınan model mimarisine yer verilmiştir. Bu şekille, verilen girdi

görüntüsünün Mask R-CNN'den geçirilerek çıktı olarak maske ve sınıf skorları çıkarıldığı gösterilmektedir. Mimari içerisinde 2 evrişim işlemi uygulanmaktadır. Mask R-CNN, aynı ilk aşamaya (RPN) sahip aynı iki aşamalı prosedürü benimser. İkinci aşamada, sınıf ve kutu uzaklığını tahmin etmeye paralel olarak Mask R-CNN ayrıca her bir ilgi bölgesi (ROI) için bir ikili maske çıktısı verir. Bu, sınıflandırmanın maske tahminlerine bağlı olduğu en yeni sistemlerin aksinedir. Mask R-CNN yaklaşımı ile paralel olarak sınırlayıcı kutu sınıflandırması ve regresyon uygulayan Faster R-CNN takip edilmektedir [85].



Şekil 6.8. COCO test setinden alınan Mask R-CNN sonuçları [84].

Resmi olarak, eğitim sırasında örneklenen her bir ilgi bölgesinde çoklu görev kaybının hesaplanmasına Eşitlik 6.1'de yer verilmiştir.

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{bbox} + \mathcal{L}_{mask} \tag{6.1}$$

Eşitlik 6.1'de yer alan değişkenler incelendiğinde $\mathcal{L}$ değişkeninin herhangi bir veriye ait kayıp değeri olduğu çıkarılmaktadır. Ek olarak sınıflandırma kaybı olan $\mathcal{L}_{cls}$ ve sınırlayıcı kutu kaybı olan $\mathcal{L}_{bbox}$ Faster R-CNN makalesindeki hesaplamalar ile aynıdır. Maske dalı, her K sınıfı için bir tane olmak üzere, $m \times m$ çözünürlüklü K ikili maskelerini kodlayan her bir ilgi bölgesi için K $\times$ m^2 boyutlu bir çıktıya sahiptir. Buna piksel başına bir sigmoid uygulanmaktadır ve $\mathcal{L}_{mask}$ ortalama ikili çapraz entropi kaybı olarak tanımlanır. Gerçek referans sınıfı k ile ilişkili bir ilgi bölgesi için, $\mathcal{L}_{mask}$ yalnızca k'ninci maskede tanımlanır (diğer maske çıktıları kayba katkıda

bulunmaz) [84]. Mask R-CNN terminolojisinde, ağ derinliği özellikleri kullanılarak omurga mimarisi belirlenmiştir. Artık ağ anlamına gelen ResNet (Residual Networks) [85] ağ, Mask R-CNN'nin makalesinde 50 veya 101 katmanlı olarak bilinmektedir. Çizelge 6.1'de verilen tablo ve Mask R-CNN makalesinde gerçekleştirilen deneysel sonuçlar incelendiğinde 101 katmanlı omurga ağı olan ResNet-101'in daha yüksek performanslı sonuçlar verdiği sonucuna ulaşılmıştır. Mask R-CNN için önceden eğitilmiş **MS COCO** [86] ağırlıkları kullanılmıştır. Microsoft'un ortaya koyduğu COCO (Bağlamdaki Ortak Nesneler, Common Objects in Context), büyük ölçekli bir nesne algılama, segmentasyon ve resim yazısı veri kümesidir [87].

Çizelge 6.1. Artık ağların ortalama hassasiyet karşılaştırma tablosu [86].

Ağ derinlik özellikleri	**AP**	**AP_{50}**	**AP_{75}**
ResNet-50-C4	30,3	51,2	31,5
ResNet-101-C4	32,7	54,2	34,3
ResNet-50-FPN	33,6	55,2	35,3
ResNet-101-FPN	35,4	57,3	37,5
ResNeXt-101-FPN	36,7	59,5	38,9

6.4. DERİN ÖĞRENME ORTAMININ HAZIRLANMASI

Bu adımda, kullanılacak YSA modeli ve omurga ağı belirlenmiştir. Bir sonraki aşama ise bu sinir ağının kullanılacak ortamının hazırlanma aşamasından oluşmaktadır. Bu adımda tez çalışmasında kullanılacak bilgisayara gerekli derin öğrenme ortamının kurulması gerekmektedir. Çalışma için seçilen programlama dili Python olarak belirlenmiştir. Şekil 6.9'da görüldüğü üzere derin öğrenme kütüphaneleri olarak Keras ve TensorFlow kullanılmıştır. Kütüphanelerin ve modelin yükleneceği ortam için Jupyter Notebook yüklenmiştir. Verilerin görselleştirilmesi için ise TensorBoard [88] kullanılmasına karar verilmiştir. Tüm bu derin öğrenme ortamı için Windows 10 Pro bilgisayarda NVIDIA RTX 5000 grafik kartına sahip bilgisayara Cuda ve cuDNN [89] kurulumları gerçekleştirilmiştir.

Şekil 6.9. Proje çalışması kapsamında kullanılan kütüphaneler ve araçlar.

6.4.1. Keras ve TensorFlow Kütüphaneleri

Keras, Python ile yazılmış, makine öğrenimi platformu TensorFlow üzerinde çalışan bir derin öğrenme API'sidir. Hızlı denemeyi mümkün kılmaya odaklanarak geliştirilmiştir. Bir fikirden sonuca olabildiğince hızlı gidebilmek, iyi araştırma yapmanın anahtarıdır [90]. TensorFlow, makine öğrenimi algoritmalarını ifade etmek için bir arabirimi ve bu tür algoritmaları yürütmek için bir uygulamadır. TensorFlow kullanılarak ifade edilen bir hesaplama, telefonlar ve tabletler gibi mobil cihazlardan yüzlerce makineden oluşan büyük ölçekli dağıtılmış sistemlere ve aşağıdakiler gibi binlerce hesaplama cihazına kadar çok çeşitli heterojen sistemlerde çok az değişiklikle veya hiç değişiklik olmadan yürütülebilir [91]. GPU kullanımında sistem esnektir ve derin sinir ağı modelleri için eğitim ve çıkarım algoritmaları dahil olmak üzere çok çeşitli algoritmaları ifade etmek için kullanılabilir. Bir düzineden fazla ülkede araştırma yapmak ve makine öğrenimi sistemlerini üretime yerleştirmek için kullanılmıştır. Konuşma tanıma, bilgisayarlı görü, robotik, bilgi alma, doğal dil işleme, coğrafi bilgi çıkarma ve hesaplamalı ilaç keşfi dahil olmak üzere bilgisayar bilimi ve diğer alanlarda aktif olarak kullanılmaktadır [92].

Bu çalışmada kullanılacak Mask R-CNN ağı için ön işleme adımından geçirilmiş taslak çizim görüntüleri ilgili klasöre yüklenmiştir. Ardından Python programlama

dilinin kodlanacağı Jupyter Notebook ise Anaconda [93] programı içerisinden etkinleştirilmiştir. Keras ve TensorFlow kütüphaneleri Mask R-CNN için uyumlu olacak şekilde yüklenmiştir. Keras ve TensorFlow sürüm dağıtımları aktif olarak güncelleme aldığı için hata almamak adına aşağıdaki versiyonların kurulumu yapılmıştır.

Keras = 2.2.5
TensorFlow-gpu = 1.15.0

6.4.2. Deneysel Çalışmalar

Tez çalışması boyunca NVIDIA kart destekli bir GPU kullanılacağı için TensorFlow yüklenirken GPU destekli olmasına özen gösterilmiştir. Şekil 6.10'da gösterildiği gibi cihazda var olan grafik kartın eğitim sırasında kullanılabilirliğini kanıtlamak adına yerelde bulunan tüm cihazlar aşağıdaki Python kodu ile listelenmiştir.

from tensorflow.python.client import device_lib
device_lib.list_local_devices( )

```
from tensorflow.python.client import device_lib
device_lib.list_local_devices()
```

```
[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 598492454864949918,
 name: "/device:GPU:0"
 device_type: "GPU"
 memory_limit: 15134713447
 locality {
   bus_id: 1
   links {
   }
 }
 incarnation: 13912145778032135420
 physical_device_desc: "device: 0, name: Quadro RTX 5000, pci bus id: 0000:17:00.0, compute capability: 7.5",
 name: "/device:GPU:1"
 device_type: "GPU"
 memory_limit: 15134713447
 locality {
   bus_id: 1
   links {
   }
 }
 incarnation: 7185622072302165551
 physical_device_desc: "device: 1, name: Quadro RTX 5000, pci bus id: 0000:65:00.0, compute capability: 7.5"]
```

Şekil 6.10. TensorFlow ile yerelde bulunan cihazların kontrolü.

Ardından sinir ağının eğitimi tamamlandığında oluşturulan günlük dosyalarının kaydedileceği bir klasör gereklidir. Bununla birlikte Mask R-CNN COCO

ağırlıklarının dosyada yok ise indirilmesi gerekmektedir. Şekil 6.11'de bu işlemler anlatılmaktadır.

```
# Günlükleri ve eğitilmiş modeli kaydetmek için dizin seçin.
MODEL_DIR = os.path.join(ROOT_DIR, "logs")

# Eğitilmiş ağırlıklar dosyasına yerel yol ayarlayın.
COCO_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")
# Gerekirse COCO eğitimli ağırlıkları sürümlerden indirin
if not os.path.exists(COCO_MODEL_PATH):
    utils.download_trained_weights(COCO_MODEL_PATH)
```

```
Using TensorFlow backend.
```

Şekil 6.11. Günlük dosyaların ve ağırlıkların yüklenmesi.

Bir sonraki adımda ise modelin konfigürasyonları yapılmaktadır. Şekil 6.12'de görüldüğü üzere ögelerin 25 adet ve arka plan olarak sınıf sayısına atandığı ve her bir iterasyon (epoch) sayısının 100 adımdan oluştuğu çıkarımı yapılmıştır.

```
class uibeeConfig(Config):
    """Taslak çizim veri kümesinde eğitim için yapılandırma.
       Temel Config sınıfından türetilir ve bazı değerleri geçersiz kılar.
    """
    # Yapılandırmaya tanınabilir bir ad verin
    NAME = "elements"

    GPU_COUNT = 1

    # İki görüntüyü sığdırabilen 16 GB belleğe sahip bir GPU kullanıyoruz.
    # Daha küçük bir GPU kullanıyorsanız aşağı ayarlayın.
    IMAGES_PER_GPU = 1

    # Sınıf sayısı (arka plan dahil)
    NUM_CLASSES = 1 + 25  # Arka plan + düğme + resim + onay kutusu

    # Dönem başına eğitim adımı sayısı
    STEPS_PER_EPOCH = 100

     # < %90 güvenle algılamaları atla
     # Bu eşiğin altındaki ROI'ler atlandı
    DETECTION_MIN_CONFIDENCE = 0.9
    DETECTION_NMS_THRESHOLD= 0.3
config = uibeeConfig()
config.display()
```

Şekil 6.12. Sinir ağı modelinin konfigürasyonları.

Bu sınıf konfigürasyonu çalıştırıldıktan sonra Şekil 6.13'teki gibi çıktı alınmaktadır. Bu şekil sonuçlarına göre, öğrenme oranı 0,001 olarak belirlenirken Mask R-CNN boyut indirgemesi ile taslak çizim görüntüleri 1024x1024 boyutlarına getirilmiştir. Maske boyutu ise 28x28 seçilirken toplam sınıf sayısının 26 olduğu Şekil 6.14'te gözlemlenmiştir.

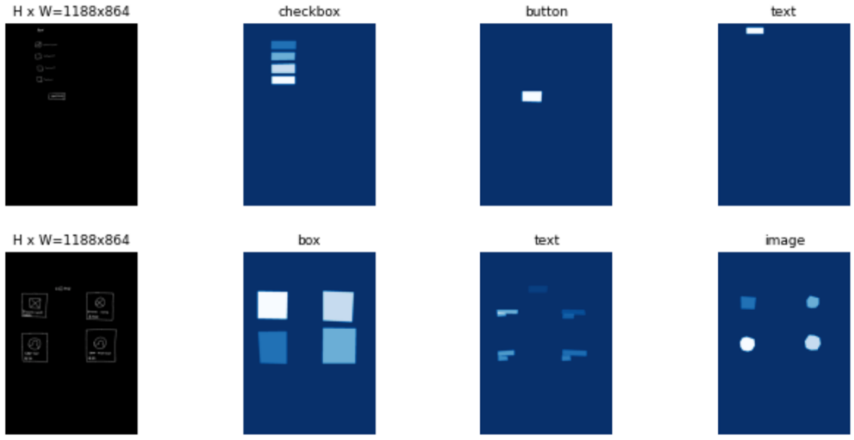
```
IMAGE_SHAPE                    [1024 1024    3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.001
LOSS_WEIGHTS                   {'rpn_class_loss': 1.0,
'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               100
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           elements
NUM_CLASSES                    26
```

Şekil 6.13. Model konfigürasyon çıktıları.

Model çalıştırılmadan önce taslak çizimlerde bulunan verilere ait doğru etiket dosyalarının çekildiğini gözlemlemek adına aşağıda bulunan Python kodu yazılmıştır.

```
image_ids = np.random.choice(dataset_train.image_ids, 2)
for image_id in image_ids:
    image = dataset_train.load_image(image_id)
    mask, class_ids = dataset_train.load_mask(image_id)
    visualize.display_top_masks(image, mask, class_ids, dataset_train.class_names)
```

Yazılan bu kod parçacığı çalıştırıldığında rastgele 2 adet taslak çizim verisi içinde bulunan UI ögeleri ile Şekil 6.14'te görüldüğü gibi görselleştirilecektir.



Şekil 6.14. Taslak çizimlerin görselleştirilmesi.

Şekil 6.14'te girdi olarak verilen taslak çizimler incelendiğinde yükseklik ve genişlik olarak 1024 boyutundan farklı olduğu görülmektedir. Bunun sebebi modelin henüz

eğitime hazırlanmadığını göstermektedir. Kısacası, verilerin henüz modele verilmediği gösterilmektedir. Görselleştirme işlemi tamamlandıktan sonra modelin eğitime hazırlanması gerekmektedir. Şekil 6.15'te sinir ağı modeli **"training"** modunda seçilerek eğitim moduna geçirilmiş ve ağırlıklar yüklenmiştir.

```
# Eğitim modunda model oluşturun
model = modellib.MaskRCNN(mode="training", config=config,
                          model_dir=MODEL_DIR)
```

```
# Başlangıç için kullanılacak ağırlık seçin.
init_with = "coco"  # imagenet, coco, veya bir başka model

if init_with == "imagenet":
    model.load_weights(model.get_imagenet_weights(), by_name=True)
elif init_with == "coco":
     # MS COCO'da eğitilmiş ağırlıkları yükleyin, ancak
     # farklı sınıf sayısı nedeniyle farklıdır
     # COCO ağırlıklarını indirme talimatları için README'ye bakın
    model.load_weights(COCO_MODEL_PATH, by_name=True,
                       exclude=["mrcnn_class_logits", "mrcnn_bbox_fc",
                                "mrcnn_bbox", "mrcnn_mask"])
elif init_with == "last":
    # Eğittiğiniz son modeli yükleyin ve eğitime devam edin
    model.load_weights(model.find_last(), by_name=True)
```

Şekil 6.15. Sinir ağı modelinin eğitim modunda seçilmesi ve ağırlıkların yüklenmesi.

Eğitim başlatıldığında kademeli olarak artacak şekilde 10, 50, 100, 150, 250, 300 epoch sayıları için **"epochs"** parametresi doldurulmuştur. Şekil 6.16'da örnek bir 300 iterasyonluk eğitim başlatılması gösterilmektedir.

```
# Baş dalları eğitin
# Layer="heads" geçişi, head dışındaki tüm katmanları dondurur
# katmanlar. Ayrıca seçmek için normal bir ifade iletebilirsiniz.
# ad kalıbına göre hangi katmanların eğitileceği.
model.train(dataset_train, dataset_val,
            learning_rate=config.LEARNING_RATE,
            epochs=300,
            layers='heads')
```

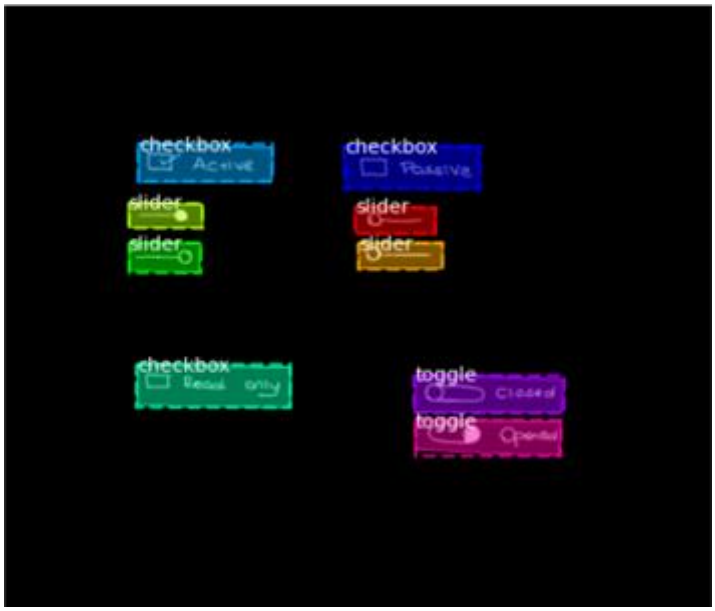
Şekil 6.16. Örnek 300 iterasyonluk eğitim başlatılması.

Seçilen konfigürasyonlardan dolayı eğitim adımları 1'den başlayarak 300'e kadar 100'er basamak doldurulacak şekilde ilerletilmiştir. Şekil 6.17'de bu adımlardan yalnızca 2 tanesine yer verilmiştir.

```
Epoch 1/300
100/100 [==============================] - 108s 1s/step -
n_class_loss: 1.0087 - mrcnn_bbox_loss: 0.6173 - mrcnn_ma
_rpn_bbox_loss: 0.6052 - val_mrcnn_class_loss: 0.9770 - v
Epoch 2/300
100/100 [==============================] - 99s 988ms/step
cnn_class_loss: 0.6365 - mrcnn_bbox_loss: 0.4106 - mrcnn_
al_rpn_bbox_loss: 0.4585 - val_mrcnn_class_loss: 0.7659 -
```

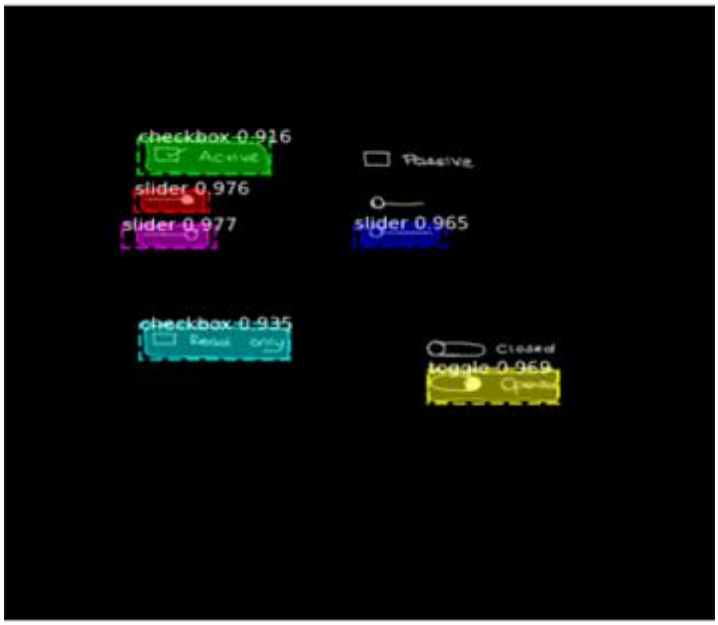
Şekil 6.17. 300 iterasyonluk eğitimin ilk 2 adımının gösterilmesi.

Sinir ağının eğitimleri tamamlandıktan sonra ilk olarak 50 epoch sayısına ait verilere Şekil 6.18'de ve Şekil 6.19'da yer verilmektedir.



Şekil 6.18. 50 epoch sayısına ait gerçek referans görseli.

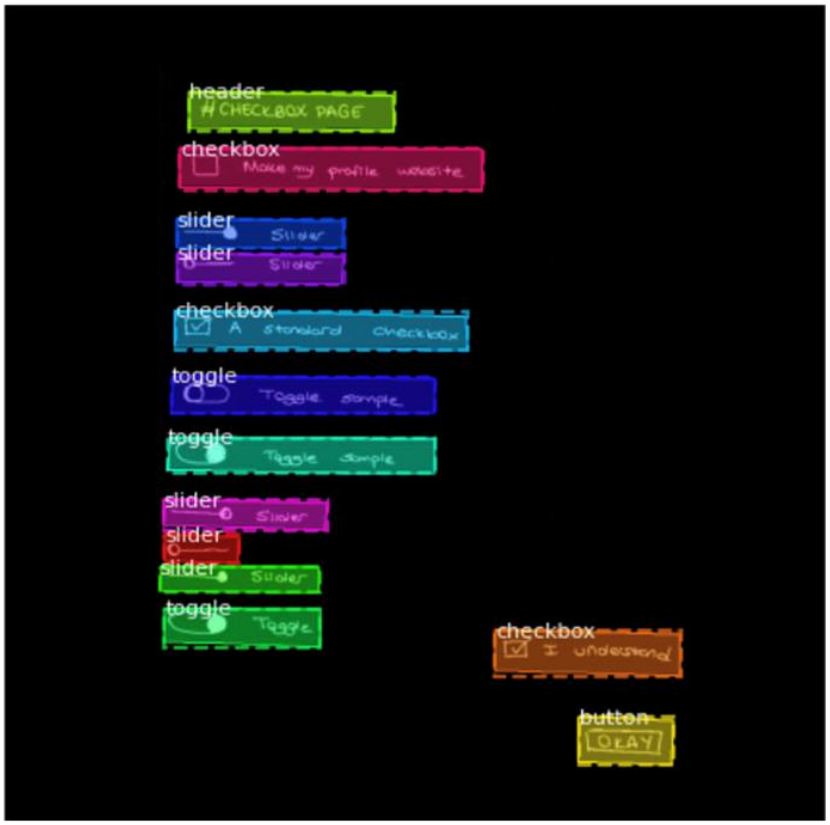
Şekil 6.18'de verilen görsel, 50 epoch için gösterilmiş gerçek referans görseline aittir. Gerçek referans görseli, etiket verilerinin modele gösterildiği ve sinir ağının öğrenmek için baz aldığı koordinat noktalarından oluşmaktadır. Şekil 6.19'da ise 50 epoch için çalıştırılmış sinir ağının tahmin sonucu gösterilmektedir.



Şekil 6.19. 50 epoch sayısına ait tahmin sonuç görseli.

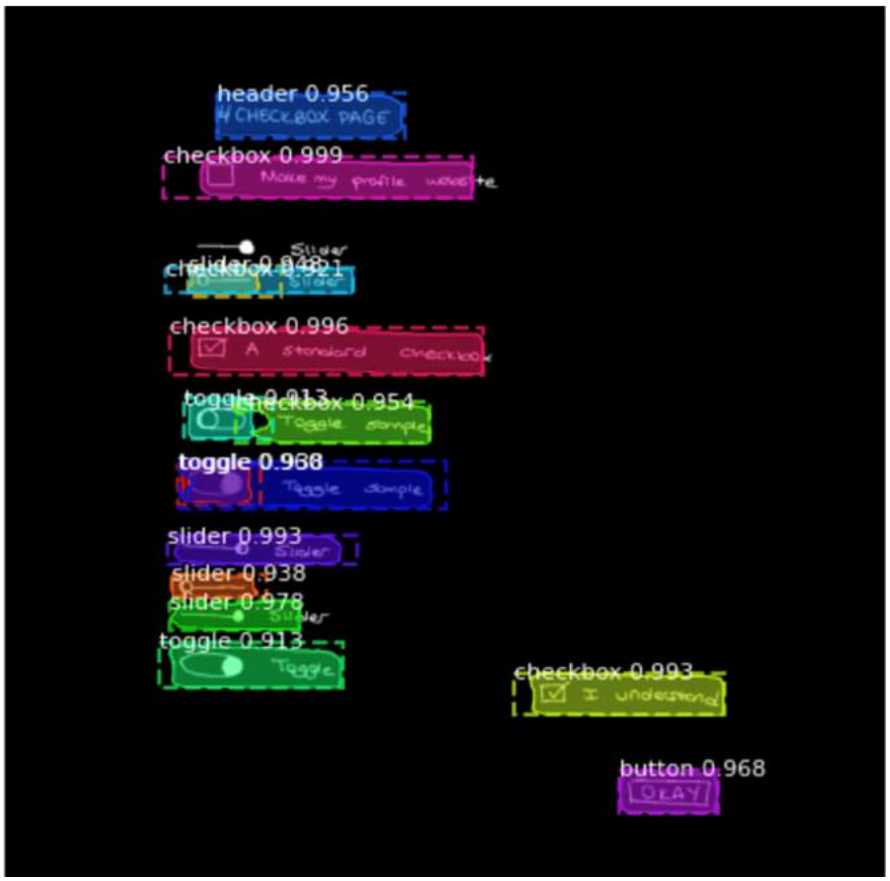
Şekil 6.19'daki tahmin sonuçları incelendiği 50 epoch sayısının yeterli olmadığına ve eksik tahminlerin var olduğu sonucuna varılmıştır. Bu sebeple, epoch sayısı artırılarak eğitim doğruluğunun daha yüksek olması hedeflenmiştir. Ardından ilk hedef olarak 100 epoch için eğitim gerçekleştirilmiştir.

Şekil 6.20'de 100 epoch sayısına ait gerçek referans görseli, Şekil 6.21'de ise bu epoch sayısının tahmin sonucuna yer verilmiştir. Şekil 6.21'deki yer alan UI ögelerinin tahmin sonuçlarını incelendiğinde, yanlış tahminler ve çakışan kutucuk problemlerinden kaynaklı üst üste hataların var olduğu sonucuna ulaşılmıştır. Hedeflenen yüksek doğruluğa ait sınıf skorlarının bulunduğu tahmin kutucukları, 100 epoch sayısı ile eşleşmemektedir. Şekil 6.20'de yer alan taslak çizimde toplam 13 adet UI ögesi bulunmaktadır. Bu nesneler, sinir ağı modeli olan Mask R-CNN ve örnek segmentasyon kullanıldığı için aynı nesne türüne ait olmasına rağmen farklı maske renklerinde boyanmıştır. Bu durum, örnek segmentasyonun kendine has bir özelliğidir. Bu görsele göre header, toggle, slider, checkbox ve button nesneleri toplam 5 türden oluşmaktadır. Etiketleme sırasında yapılan poligon koordinatlarından kaynaklı maske görselleri gösterilmektedir.



Şekil 6.20. 100 epoch sayısına ait gerçek referans görseli.

Şekil 6.21'de ise ilk tahmin sonucu olan header nesnesi %95,6 skorunda tahmin edilmiştir. Ancak dikkat edilmelidir ki ikinci tahmin olan checkbox nesnesinin altında yer alan slider nesnesi maalesef modelin gözünden kaçmış ve tahmini yapılmamıştır. Bu durum istenen bir durum değildir. Bu sebeple 100 epoch sayısı da artırılarak kademeli olarak eğitime devam edilmiştir.



Şekil 6.21. 100 epoch sayısına ait tahmin sonuç görseli.

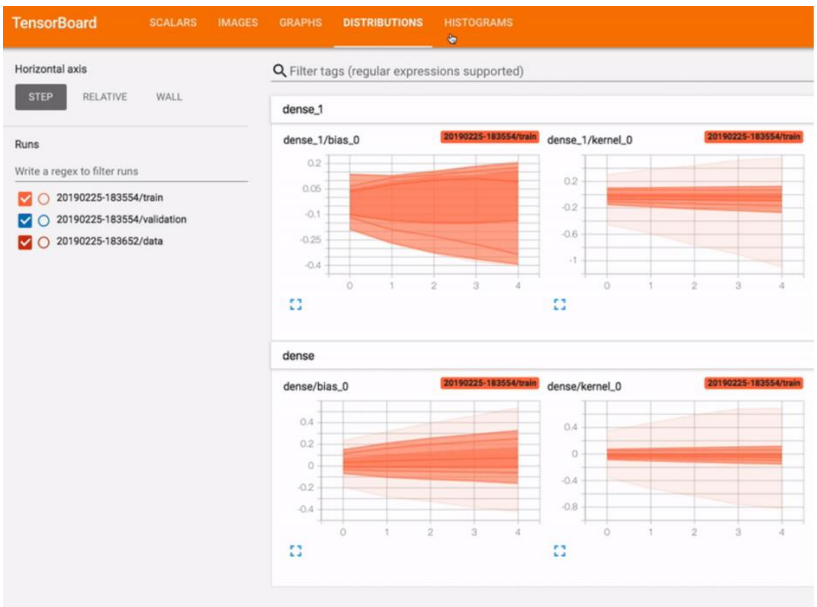
Eğitim sırasında sinir ağı modelinin başarım oranını hesaplamak ve ek olarak ağın aşırı uydurma yaptığının kontrolünü gözlemlemek için bazı hesaplama metrikleri mevcuttur. Model başarısının değerlendirilmesinde kullanılan en basit ve yaygın yöntem, doğruluk oranına bakmaktır. Doğru sınıflandırılmış örnek sayısının (True Positive + True Negative) toplam örnek sayısına (True Positive + True Negative + False Positive + False Negative) bölünmesiyle çıkan sonuçtur. Hata oranı ise yanlış sınıflandırılmış örnek sayısının toplam örnek sayısına bölünmesi ile bulunur. Diğer bir ifade ile hata oranı doğruluk oranını 1'e tamamlayan değerdir [94].

Hassasiyet (Precision), doğru sınıflandırılmış pozitif örneklem (TP) sayısının, sınıfı pozitif olarak tahmin edilmiş toplam örneklem sayısına bölünmesi ile bulunur. Hassasiyet değerinin hesaplanmasına Eşitlik 6.2'de yer verilmektedir. Sonuç [0,1] aralığında çıkar [95].

$$Hassasiyet\ (Precision) = \frac{TP}{TP+FP} \tag{6.2}$$

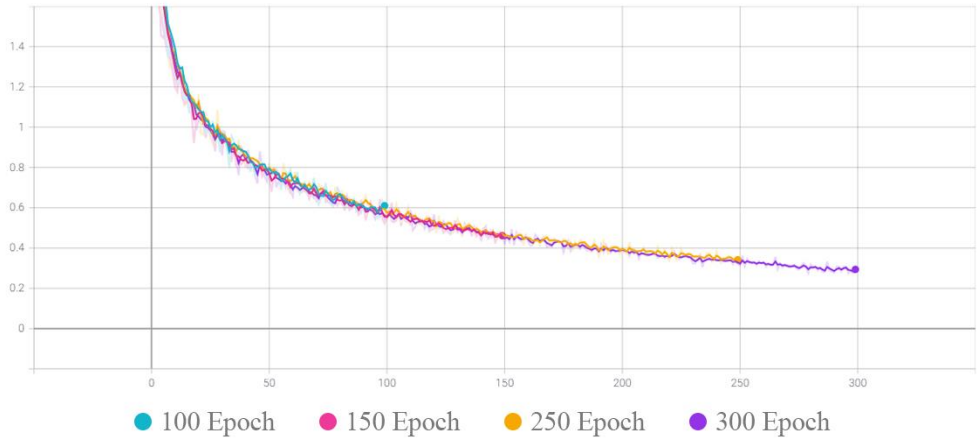
$$AP\ (Ortalama\ Hassasiyet) = \frac{1}{GRP} \sum_{k}^{n} P@k\ x\ rel@k \tag{6.3}$$

Ortalama Genel Hassasiyet (mAP), bir dizi sorgu için genel ortalama hassasiyet, her sorgu için ortalama hassasiyet puanlarının ortalamasıdır. Sonuç olarak tüm AP puanlarının ortalaması alınarak mAP değeri elde edilir ve hesaplanmasına Eşitlik 6.3'te yer verilmiştir. Mask R-CNN ağı için olan hesaplama metriği olarak ortalama genel hassasiyet (mAP – mean average precision) kullanılmaktadır. Eşitlik 6.3'te verilen başarım metriği bu tez çalışmasında baz alınarak doğruluk hesaplatılmıştır. Deneysel çalışmaların son aşamasında sinir ağı modelinin başarısı ve kayıp grafiklerinin gösterilmesi için TensorBoard kullanılmıştır. TensorBoard, makine öğrenimi temelli projelerde veri görselleştirilmesine imkân tanıyan bir kütüphanedir. TensorFlow kütüphanesinin son sürümleri ile otomatik olarak yüklenmektedir. TensorBoard sayesinde makine öğrenimi modellerinin görselleştirilmesi sağlanmaktadır. Şekil 6.22'de TensorBoard örnek paneli gösterilmektedir.



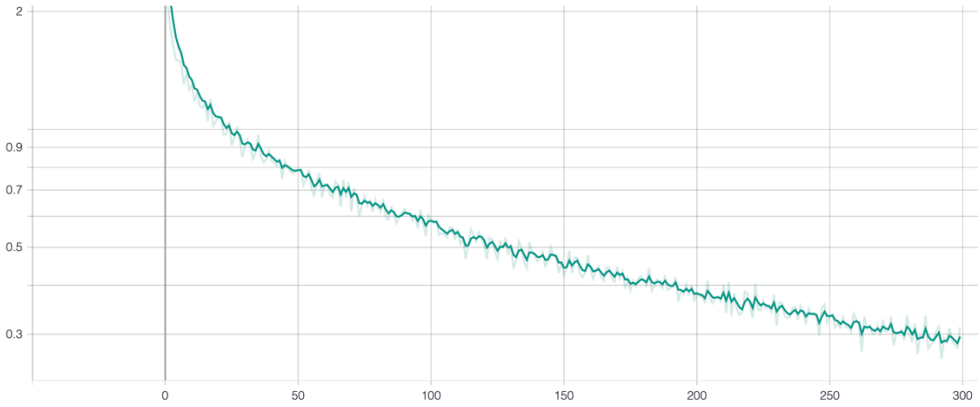
Şekil 6.22. TensorBoard görselleştirme paneli [88].

TensorBoard sayesinde kayıp değerleri görselleştirilerek Şekil 6.23'te gösterilmektedir. Şekil 6.23'te 300 epoch sayısına kadar olan günlükler görselleştirilmiştir.



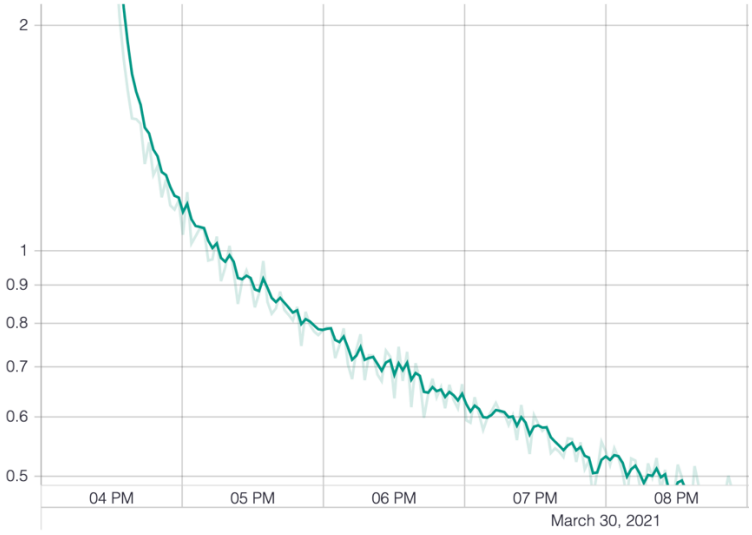
Şekil 6.23. 300 epoch sayısına kadar olan kayıp grafikleri.

Şekil 6.23'te yer alan grafikler her bir günlük için farklı renklerde oluşturulmuştur. En son oluşturulan yeşil grafik ise 300 epoch sayısına kadar ulaşmış olup kayıp değeri 0,3 değerinin altına kadar düşmüştür. Bu durum istenilen başarım oranına ulaşıldığını göstermektedir. Şekil 6.24'te ise daha iyi incelemek adına, 300 epoch sayısına ait kayıp grafiğine yer verilmektedir.



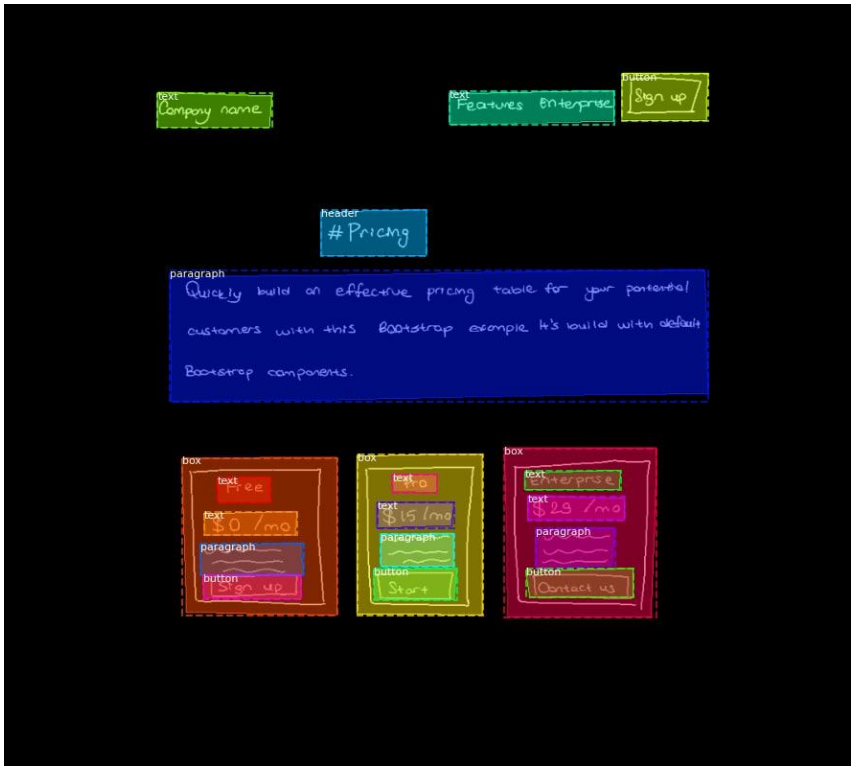
Şekil 6.24. Yalnızca 300 epoch sayısına ait kayıp grafiği.

Deneysel çalışmalar sırasında, TensorBoard içerisinde Mask R-CNN ağına ait olan ***mrcnn_loss_wall*** grafiği çizdirildiğinde ise Şekil 6.25'teki gibi bir grafik elde edilmiştir. Mrcnn değeri ise Mask R-CNN ağının klasörüne aittir.

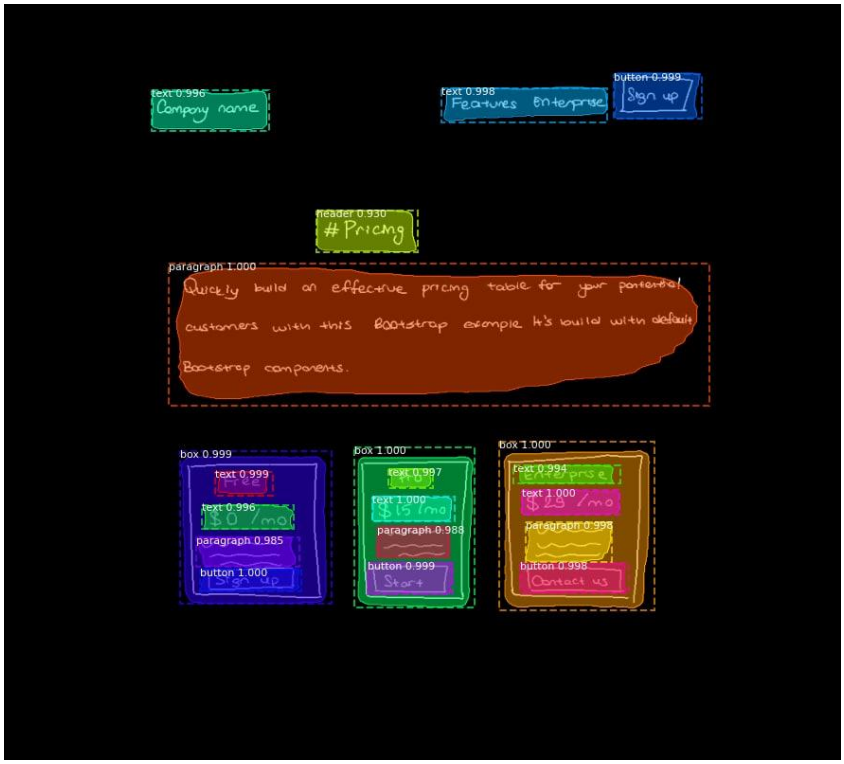


Şekil 6.25. Mrcnn 300 epoch sayısına ait farklı kayıp görselleştirme sonucu.

Şekil 6.26'da örnek bir wireframe verisine ait gerçek referans görseli yer alırken Şekil 6.27'de ise tahmin sonucu yer almaktadır.



Şekil 6.26. 300 epoch sayısına ait gerçek referans görseli.



Şekil 6.27. 300 epoch sayısına ait tahmin sonuç görseli.

Tüm bu eğitimler sonucunda elde edilen başarım sonuçlarına ise Çizelge 6.2'de yer verilmiştir.

Çizelge 6.2. Test verileri için farklı epoch sayılarına ait sonuçlar.

Epoch Sayısı	**Toplam Eğitim Süresi**	**Ortalama Genel Hassasiyet (%)**	**Ortalama Genel Duyarlılık (%)**	**F1-Skoru (%)**
10	15d 5s	48,21	21,01	28,98
50	1s 15d 10s	73,60	53,74	63,04
100	2s 53d 1s	90,86	55,68	68,91
150	4s 7d 9s	96,36	52,95	69,22
300	8s 38d 2s	98,48	59,05	73,27

Çizelge 6.2, üç ayrı özellik sütunundan oluşmaktadır. Buna göre epoch sayısı, sinir ağı modelinin kaç adımda eğitildiğini göstermektedir. Süre sütununda ise ilk s harfi saati, d harfi dakikayı ve bir sonraki s harfi ise saniyeyi temsil etmektedir. Ek olarak mAP değeri ise modelin başarım performansını ölçerken kullanılan metriktir. Çizelge

6.2’deki değerler incelendiğinde 150 epoch sonrası eğitim başarısının artışının zamanla azaldığı gözlemlenmiştir. Bu sebeple 300 epoch sonucunda istenen hedefe ulaşıldığı kararı alınarak yeniden bir eğitime ihtiyaç duyulmamıştır. Özetleyecek olursak görselleştirilen grafikler ve çizelge incelendiğinde, eğitim kaybı her iterasyonda düşerken eğitim başarımı her iterasyonda artmaktadır. Ek olarak, en yüksek mAP değerinin ise 300 epoch sayısına ait %98,48 skoru olduğu belirlenmiştir.

BÖLÜM 7

SONUÇLAR VE ÖNERİLER

Bu çalışmada, elle oluşturulmuş taslak çizim verilerinde yer alan UI ögelerinin tespit edilmesi için evrişimli sinir ağı türlerinden olan Mask R-CNN ağı kullanılmıştır. Sinir ağında yer alan girdiler için aktivasyon fonksiyonunda kullanılan ağırlıklar önceden eğitilmiş MS COCO ağırlıklarıdır. Sinir ağının omurgasında kullanılan yapı ise artık ağlardan olan ResNet-101 ağıdır. Özellik çıkarımı için Mask R-CNN ağının iç yapısında var olan RPN ve FPN mimarileri kullanılmıştır.

Sonuçlar ve çalışmanın süreci boyunca Windows işletim sistemine ait olan Windows 10 Pro yüklü bir sistem kullanılmıştır. Bu sistem, i9 10980XE işlemcili ve NVIDIA Quadro RTX 5000 ekran kartına sahip bileşenlerden oluşmaktadır. Sinir ağının eğitimi sonucunda, veri setinde yer alan ancak eğitim kümesinde mevcut olmayan toplam 87 test görüntüsü test için ağa verilmiştir. Eğitim için sırasıyla 300 epoch sayısına kadar sinir ağı eğitildikten sonra, 87 adet test verisi test edilerek başarım metriği olarak kullanılan mAP değeri için en yüksek değerin %98,48 skoru olduğu belirlenmiştir.

Gelecek çalışmalar için, literatürde sıkça yer alan ancak oluşturulan veri setinde yer almayan UI ögelerinin bulunduğu taslak çizimler çalışmaya dahil edilerek daha çok UI ögesinin tespit edilmesi sağlanabilir. Bununla birlikte, insan çiziminden kaynaklanan olumsuz sebeplerden ötürü çizilen çok farklı türde UI ögesi teste verildiğinde beklenilen yüksek performans sağlanmayabilir. Bu sebeple, gelecek çalışmalar için veri seti olabildiğince çoğaltılarak ağın aykırı verilerde de performanslı sonuçlar vermesi beklenebilir. Ek olarak, yazılım ve tasarım sektöründe projenin oluşum sürecinde var olan ancak çok zaman alan UI/UX aşamaları tespit edilen UI ögeleri ile ilgili yazılım kodlamalarına dönüştürülerek sektörde kullanılabilecek bir projeye dönüştürülebilir.

KAYNAKLAR

1. İnternet: The Standish Group, "CHAOS Report 2015", **https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf** (2015).

2. İnternet: The Standish Group, "CHAOS 2020: Beyond Infinity", **https://hennyportman.wordpress.com/2021/01/06/review-standish-group-chaos-2020-beyond-infinity** (2020).

3. LLC Digital Publications, "Software development lifecycle phases" (2005).

4. Erdem, A., Younis, E., "Yazılım Projelerinin Geliştirme Sürecinde Yönetim", ***Bilişim Teknolojileri Dergisi***, 7 (2014).

5. İnternet: Dayıbaşı, O., "Yazılım Geliştirme Modelleri", **https://medium.com/architectural-patterns/yazılım-geliştirme-modelleri-62915545c51e** (2020).

6. İnternet: Dayıbaşı, O., "Neden Farklı Yazılım Geliştirme Modelleri Var?", **https://medium.com/architectural-patterns/yazılım-geliştirme-yaşam-döngüsü-ad8a5c5b9684** (2020).

7. İnternet: Dayıbaşı, O., "Yazılım Geliştirme Fazları", **https://medium.com/architectural-patterns/yazılım-geliştirme-fazları-1f016ce1d412** (2020).

8. Boehm, B. W., and Ross, R, "Theory-W software project management principles and examples", ***IEEE Transactions on Software Engineering***, 15(7), 902-916 (1989).

9. Gökgöz, B., Keskinkılıç, M., "Yazılım Proje Geliştirme Sürecinde Proje Yönetim Aşamaları ve Risk Analizi İncelemesi", ***SETSCI Conference Indexing System***, 3, 1040-1045 (2018).

10. Campos, P., and Nunes, N. J., "Practitioner tools and workstyles for user-interface design", ***IEEE software***, 24(1), 73-80 (2007).

11. Landay, J. A., and Myers, B. A, "Interactive sketching for the early stages of user interface design", ***In Proceedings of the SIGCHI conference on Human factors in computing systems*** (pp. 43-50) (1995).

12. Da Silva, T. S., Martin, A., Maurer, F., and Silveira, M., "User-centered design and agile methods: a systematic review", ***In 2011 AGILE conference*** (pp. 77-86), IEEE (2011).

13. Landay, J. A., and Myers, B. A., "Sketching interfaces: Toward more human interface design", ***Computer***, *34*(3), 56-64 (2001).

14. Lin, J., Newman, M. W., Hong, J. I., and Landay, J. A., "DENIM: Finding a tighter fit between tools and practice for web site design", ***In Proceedings of the SIGCHI conference on Human factors in computing systems*** (pp. 510-517) (2001).

15. Nguyen, T. A., & Csallner, C., "Reverse engineering mobile application user interfaces with remaui (t)", ***In 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*** (pp. 248-259), IEEE (2015).

16. Zita, A., Picek, L., and Říha, A., "Sketch2Code: Automatic hand-drawn UI Elements Detection with Faster-RCNN", ***In CLEF (Working Notes)*** (2020).

17. Robinson, A., "Sketch2code: Generating a website from a paper mockup" (2019).

18. İnternet: Balsamiq, "Balsamiq", **https://balsamiq.com** (2021).

19. İnternet: Marvel, "Marvel App", **https://marvelapp.com** (2021).

20. İnternet: Wirify, "Wirify", **https://www.wirify.com** (2021).

21. Wong, Y. Y., "Rough and ready prototypes: Lessons from graphic design", ***In Posters and short talks of the 1992 SIGCHI conference on Human factors in computing systems*** (pp. 83-84) (1992).

22. Dong, C., Loy, C. C., He, K., and Tang, X, "Image super-resolution using deep convolutional networks", ***IEEE transactions on pattern analysis and machine intelligence***, *38*(2), 295-307 (2015).

23. Varadarajan, B., Toderici, G., Vijayanarasimhan, S., and Natsev, A., "Efficient large scale video classification", ***arXiv preprint*** *arXiv:1505.06250* (2015).

24. Vinyals, O., Toshev, A., Bengio, S., and Erhan, D., "Show and tell: A neural image caption generator", ***In Proceedings of the IEEE conference on computer vision and pattern recognition*** (pp. 3156-3164) (2015).

25. Karpathy, A., and Fei-Fei, L., "Deep visual-semantic alignments for generating image descriptions", ***In Proceedings of the IEEE conference on computer vision and pattern recognition*** (pp. 3128-3137) (2015).

26. Gatys, L. A., Ecker, A. S., and Bethge, M., "A neural algorithm of artistic style", ***arXiv preprint*** *arXiv:1508.06576* (2015).

27. Vacek, R., "President's message: UX thinking and the LITA member experience", ***Information Technology and Libraries***, 33(3), 1-4 (2014).

28. Bunel, R. R., Desmaison, A., Mudigonda, P. K., Kohli, P., and Torr, P., "Adaptive neural compilation", ***Advances in Neural Information Processing Systems***, *29*, 1444-1452 (2016).

29. Bošnjak, M., Rocktäschel, T., Naradowsky, J., and Riedel, S., "Programming with a differentiable forth interpreter", ***In International conference on machine learning*** (pp. 547-556). PMLR (2017).

30. Ling, W., Grefenstette, E., Hermann, K. M., Kočiský, T., Senior, A., Wang, F., and Blunsom, P., "Latent predictor networks for code generation", ***arXiv preprint*** *arXiv:1603.06744* (2016).

31. Gaunt, A. L., Brockschmidt, M., Singh, R., Kushman, N., Kohli, P., Taylor, J., and Tarlow, D., "Terpret: A probabilistic programming language for program induction", ***arXiv preprint*** *arXiv:1608.04428* (2016).

32. Balog, M., Gaunt, A. L., Brockschmidt, M., Nowozin, S., and Tarlow, D., "Deepcoder: Learning to write programs", ***arXiv preprint*** *arXiv:1611.01989* (2016).

33. Beltramelli, T., "pix2code: Generating code from a graphical user interface screenshot", ***In Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*** (pp. 1-6) (2018).

34. Karpathy, A., and Fei-Fei, L., "Deep visual-semantic alignments for generating image descriptions", ***In Proceedings of the IEEE conference on computer vision and pattern recognition*** (pp. 3128-3137) (2015).

35. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., and Bengio, Y., "Show, attend and tell: Neural image caption generation with visual attention", ***In International conference on machine learning*** (pp. 2048-2057). PMLR (2015).

36. Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T., "Long-term recurrent convolutional networks for visual recognition and description", ***In***

Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2625-2634) (2015).

37. Luong, M. T., Pham, H., & Manning, C. D., "Effective approaches to attention-based neural machine translation", ***arXiv preprint*** *arXiv:1508.04025* (2015).

38. Huang, X., and Liao, F., "Automatically Generating Codes from Graphical Screenshots Based on Deep Autocoder", ***arXiv preprint*** *arXiv:2007.02272* (2020).

39. Polozov, O., & Gulwani, S., "Flashmeta: A framework for inductive program synthesis", ***In Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*** (pp. 107-126) (2015).

40. Abdelhamid, A. A., Alotaibi, S. R., and Mousa, A., "Deep learning-based prototyping of android GUI from hand-drawn mockups", ***IET Software***, 14(7), 816-824 (2021).

41. Pan, J., Chen, X., Chen, T., Tang, B., Yang, J., Chen, Y., and Meng, J., "Visual Compiler: Towards Translating Digital UI Design Draft to Front-End Code Automatically", ***In International Conference on Human-Computer Interaction*** (pp. 385-394) (2020).

42. Xu, Y., Bo, L., Sun, X., Li, B., Jiang, J., and Zhou, W., "image2emmet: Automatic code generation from web user interface image", ***Journal of Software: Evolution and Process***, 33(8), e2369 (2021).

43. De Carvalho Ferreira, B. S. L., "Automatic Generation of Synthetic Website Wireframe Datasets from Source Code" (2020).

44. Aşıroğlu, B., Mete, B. R., Yıldız, E., Nalçakan, Y., Sezen, A., Dağtekin, M., and Ensari, T., "Automatic HTML code generation from mock-up images using machine learning techniques", ***In 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*** (pp. 1-4) (2019).

45. De Souza Baulé, D., von Wangenheim, C. G., von Wangenheim, A., and Hauck, J. C., "Recent Progress in Automated Code Generation from GUI Images Using Machine Learning Techniques", ***J. Univers. Comput. Sci***., 26(9), 1095-1127 (2020).

46. Ge, X., "Android GUI search using hand-drawn sketches", ***In 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*** (pp. 141-143) (2019).

47. Jain, V., Agrawal, P., Banga, S., Kapoor, R., and Gulyani, S., “Sketch2code: transformation of sketches to UI in real-time using deep neural network”, arXiv preprint arXiv:1910.08930 (2019).

48. Moran, K., Bernal-Cárdenas, C., Curcio, M., Bonett, R., and Poshyvanyk, D., “Machine learning-based prototyping of graphical user interfaces for mobile apps”, ***IEEE Transactions on Software Engineering***, 46(2), 196-221 (2018).

49. Suleri, S., Sermuga Pandian, V. P., Shishkovets, S., and Jarke, M., “Eve: A sketch-based software prototyping workbench”, ***In Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*** (pp. 1-6) (2019).

50. Pandian, V. P. S., Suleri, S., and Jarke, M., “Blu: What GUIs are made of”, ***In Proceedings of the 25th International Conference on Intelligent User Interfaces Companion*** (pp. 81-82) (2020).

51. Bajammal, M., Mazinanian, D., and Mesbah, A., “Generating reusable web components from mockups”, ***In 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*** (pp. 601-611) (2018).

52. Beltramelli, T., “HacN your design sprint: wireframes to prototype in under 5 minutes” (2019).

53. Chen, C., Su, T., Meng, G., Xing, Z., and Liu, Y., “From ui design image to gui skeleton: a neural machine translator to bootstrap mobile gui implementation”, ***In Proceedings of the 40th International Conference on Software Engineering*** (pp. 665-676) (2018).

54. Chen, S., Fan, L., Su, T., Ma, L., Liu, Y., and Xu, L., “Automated cross-platform GUI code generation for mobile apps”, ***In 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile)*** (pp. 13-16) (2019).

55. Halbe, A., and Joshi, A. R., “A novel approach to HTML page creation using neural network”, ***Procedia Computer Science***, 45, 197-204 (2015).

56. Han, Y., He, J., and Dong, Q., “CSSSketch2Code: An Automatic Method to Generate Web Pages with CSS Style”, ***In Proceedings of the 2nd International Conference on Advances in Artificial Intelligence*** (pp. 29-35) (2018).

57. Kim, B., Park, S., Won, T., Heo, J., and Kim, B., "Deep-learning based web UI automatic programming", ***In Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems*** (pp. 64-65) (2018).

58. Kumar, A., "Automated front-end development using deep learning", ***Medium Insight*** **(**2018).

59. Liu, Y., Hu, Q., and Shu, K., "Improving pix2code based Bi-directional LSTM", ***In 2018 IEEE International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*** (pp. 220-223) (2018).

60. Pandian, V. P. S., Suleri, S., and Jarke, M., "Blu: What GUIs are made of", ***In Proceedings of the 25th International Conference on Intelligent User Interfaces Companion*** (pp. 81-82) (2020).

61. Pandian, V. P. S., and Suleri, S., "BlackBox Toolkit: Intelligent Assistance to UI Design", ***arXiv preprint*** arXiv:2004.01949 (2020).

62. Wallner, E., "Turning Design MocNups into Code with Deep Learning", ***Retrieved January***, 13 (2018).

63. Yun, Y. S., Jung, J., Eun, S., So, S. S., and Heo, J., "Detection of gui elements on sketch images using object detector based on deep neural networks", ***In International Conference on Green and Human Information Technology*** (pp. 86-90) (2018).

64. Dutta, A., and Zisserman, A., "The VIA annotation software for images, audio and video", ***In Proceedings of the 27th ACM international conference on multimedia*** (pp. 2276-2279) (2019).

65. Bradski, G., and Kaehler, A., "OpenCV", ***Dr. Dobb's journal of software tools 3*** **(**2000).

66. Blackledge, M. J., ***Digital Image Processing***, (2005).

67. İnternet: OpenCV Orijinal Dokümantasyon, "Image Thresholding", **https://docs.opencv.org/4.5.1/d7/d4d/tutorial_py_thresholding.html** (2021).

68. Chollet, F., "*Deep learning with Python*", Simon and Schuster, ***Manning Publications***, ISBN 9781617296864 (2017).

69. Mohri, M., Rostamizadeh, A., and Talwalkar, A., "*Foundations of machine learning*", ***MIT press*** (2018).

70. Yegnanarayana, B., "Artificial neural networks", ***PHI Learning Pvt. Ltd.*** (2009).

71. Köprü, E. Y., "Yapay sinir ağları ile sıvı ham demir tahmini ve 5. yüksek fırın uygulaması", Yüksek Lisans Tezi, ***Karabük Üniversitesi Lisansüstü Eğitim Enstitüsü***, Karabük (2020).

72. Saraç, T., "Yapay sinir ağları", Seminer Projesi, ***Gazi Üniversitesi Endüstri Mühendisliği Anabilim Dalı***, Ankara (2004).

73. Amidi, A., ve Amidi, S., "Evrişimli Sinir Ağları El Kitabı VIP", ***Stanford Üniversitesi*** (2019).

74. Zhang, H., Lee, K., Chen, Z., Kashyap, S., and Sonka, M., "LOGISMOS-JEI: Segmentation using optimal graph search and just-enough interaction", ***In Handbook of Medical Image Computing and Computer Assisted Intervention,*** pp. 249-272 (2020).

75. Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J., "Path aggregation network for instance segmentation", ***In Proceedings of the IEEE conference on computer vision and pattern recognition***, pp. 8759-8768 (2018).

76. Chen, L. C., Hermans, A., Papandreou, G., Schroff, F., Wang, P., & Adam, H., "Masklab: Instance segmentation by refining object detection with semantic and direction features", ***In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,*** pp. 4013-4022 (2018).

77. Li, Y., Qi, H., Dai, J., Ji, X., and Wei, Y., "Fully convolutional instance-aware semantic segmentation", ***In Proceedings of the IEEE conference on computer vision and pattern recognition***, pp. 2359-2367 (2017).

78. Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A., "The pascal visual object classes (voc) challenge", ***International journal of computer vision***, *88*(2), 303-338 (2010).

79. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., and Lin, D., "Hybrid task cascade for instance segmentation", ***In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition***, pp. 4974-498s (2019).

80. Casado-García, Á., Domínguez, C., García-Domínguez, M., Heras, J., Inés, A., Mata, E., and Pascual, V., "CLoDSA: a tool for augmentation in classification, localization, detection, semantic segmentation and instance segmentation tasks", ***BMC bioinformatics***, *20*(1), 1-14 (2019).

81. Vuola, A. O., Akram, S. U., and Kannala, J., "Mask-RCNN and U-net ensembled for nuclei segmentation", ***In 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)***, pp. 208-212 (2019).

82. Johnson, J. W., "Adapting mask-rcnn for automatic nucleus segmentation", ***arXiv preprint arXiv:1805.00500*** (2018).

83. Ren, S., He, K., Girshick, R., and Sun, J., "Faster r-cnn: Towards real-time object detection with region proposal networks", ***Advances in Neural Information Processing Systems***, *28*, 91-99 (2015).

84. He, K., Gkioxari, G., Dollár, P., and Girshick, R., "Mask r-cnn", ***In Proceedings of the IEEE international conference on computer vision***, pp. 2961-2969 (2017).

85. He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition", ***In Proceedings of the IEEE conference on computer vision and pattern recognition***, pp. 770-778 (2016).

86. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., and Zitnick, C. L., "Microsoft coco: Common objects in context", ***In European conference on computer vision***, pp. 740-755 (2014).

87. İnternet: COCO Dataset, "Microsoft COCO Dataset - Common Objects in Context", **https://cocodataset.org/** (2021).

88. İnternet: TensorBoard: TensorFlow'un görselleştirme araç takımı, "TensorBoard", https://www.tensorflow.org/tensorboard (2021).

89. Sanders, J., and Kandrot, E., "CUDA by example: an introduction to general-purpose GPU programming", Addison-Wesley Professional (2010).

90. İnternet: Keras Applications, "Keras Applications VGG16 Network Function", **https://keras.io/api/applications/** (2021).

91. İnternet: TensorFlow Web Sitesi, "TensorFlow", **https://www.tensorflow.org** (2021).

92. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., and Zheng, X., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems", ***arXiv preprint arXiv:1603.04467*** (2016).

93. İnternet: Anaconda Orijinal Web Sitesi, "Anaconda Data Science Technology", **https://www.anaconda.com** (2021).

94. BİLGİN, M., "Gerçek veri setlerinde klasik makine öğrenmesi yöntemlerinin performans analizi", ***Breast***, 2(9), 683, 2017.

95. İnternet: Wikipedia, The free encyclopedia, "Evaluation measures (information retrieval)", (2021).

ÖZGEÇMİŞ

Cahit Berkay KAZANGİRLER ilk, orta ve lise öğrenimini aynı şehirde tamamladı. Eskişehir'de yer alan H. Süleyman Çakır Lisesi Sayısal bölümünden mezun oldu. 2012 yılında Karabük Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nde öğrenime başlayıp 2017 yılında mezun oldu. 2018 yılında Karabük Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda başlamış olduğu yüksek lisans programını, Karabük Üniversitesi Lisansüstü Eğitim Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı altında sürdürmektedir. 2018 yılında AKCHE Soft'ta Yazılım Geliştirici ünvanı ile 4 ay çalıştıktan sonra ADEO Bilişim Hizmetleri Şirketi'nde 10 ay Yazılım Geliştiriciliği ve Arayüz Tasarımcılığı yaptı. 2019 yılında Kodia Yazılım ve Teknoloji Şirketi'nde 10 ay Arayüz Tasarımcısı olarak çalıştı. 2020 yılında Arayüz Tasarımcısı ve Web Ön Yüz Geliştiricisi olarak girdiği Privia Security Şirketi'nde son 8 ay Proje Yöneticiliği olmak üzere toplam 1 yıl 7 ay çalıştı. 2021 yılından bu yana ise aktif olarak Ara yüz Tasarımcısı ve Yazılım Geliştiricisi olarak çalışmaktadır.