Bilgisayar Bilimleri ve Mühendisliği Dergisi

Cilt-16
Sayı-2
2023

Görüntülerden veya Çizimlerden Otomatik Kot Oluşturma Teknikleri: Bir Derleme Çalışması

Automatic Code Generation Techniques from Images or Sketches: A Review Study

Musa Selman KUNDURACI
Bursa Teknik Üniversitesi
Bilgisayar Mühendisliği
Bursa, Türkiye
mskunduraci@gmail.com
ORCID: 0000-0001-9823-3387

Turgay Tugay BİLGİN
Bursa Teknik Üniversitesi
Bilgisayar Mühendisliği
Bursa, Türkiye
turgay.bilgin@btu.edu.tr
ORCID: 0000-0002-9245-5728

Öz

Bir yazılımın geliştirilmesi sürecinde, tasarım ve öncül üretim en önemli ve zaman alıcı aşamalardır. Kullanıcılar yazılımların görsel arayüzlerine ve tasarımlarına oldukça önem vermektedir. İyi bir görsel arayüz tasarımına sahip bir yazılım daha iyi işleve sahip olup fakat arayüzü kullanışsız olan benzerinden daha fazla tercih edilmektedir. Görsel arayüz tasarımı sürecinde geliştiriciler öncelikle kâğıt üzerinde tasarım gerçekleştirip ardından görsel arayüz tasarım programları ile dijital tasarıma dönüştürürler. Sonraki aşamada, tasarımın çeşitli biçimlendirme dilleriyle (xml, html, css vb.) veya doğrudan programlama dilleriyle kodlanması gerekmektedir. Otomatik kot üretme yaklaşımlarında amaç minimum yazılım geliştirici maliyeti ile kısa zamanda verimli ve hızlı uygulamalar geliştirmektir. Bu çalışmada, çeşitli yöntemleri kullanarak otomatik kot üretimi gerçekleştiren çalışmaları içeren geniş bir yayın taraması oluşturulmuştur. İncelenen makalelerde çoğunlukla derin öğrenme, görüntü işleme, yapay sinir ağları veya makine öğrenmesi yöntemleri kullanılmıştır. Bu derleme çalışması ile bu alanda çalışma yapacak araştırmacılara rehber olunması amaçlanmıştır.

Anahtar sözcükler: Otomatik Kot Üretimi, Derin Öğrenme, Makine Öğrenmesi

Abstract

In the process of developing a software, design and prototyping are the most important and time-consuming stages. Users attach great importance to the visual interfaces and designs of the software. A software with a good visual interface design is preferred more than a similar one with better functionality but an unusable interface. In the process of visual interface design, developers first design on paper and then turn it into digital design with visual interface design progra

ms. In the next step, the design needs to be coded with various markup languages (xml, html, css etc.) or directly with programming languages. The aim of automatic code generation approaches is to develop efficient and fast applications in a short time with minimum software developer cost. In this study, a large literature review was created that includes studies that perform automatic code generation using various methods. In the reviewed articles, mostly deep learning, image processing, artificial neural networks or machine learning methods were used. With this review study, it is aimed to guide researchers who will work in this field.

Keywords*:* Automatic Code generation, Deep Learning, Machine Learning

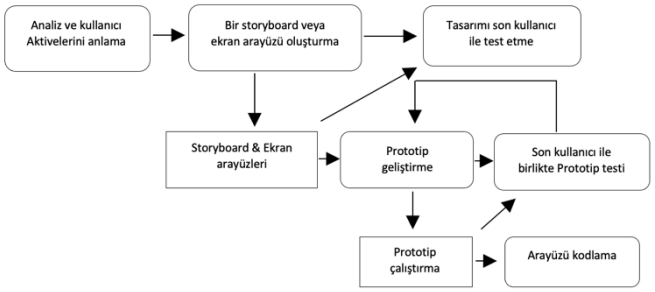
Gönderme, düzeltme ve kabul tarihi: 16.10.2022 - 14.03.2023 – 07.05.2023
Makale türü: Derleme

1. Giriş

Kullanıcı arayüzü diğer bir adıyla kullanıcı arabirimi, kullanıcıların bir makine veya cihazı kullanabilmelerine olanak sağlayan yöntemlerin bileşkesi olarak tanılanabilir. Kullanıcı arayüzü sayesinde kullanıcılar bir sistemin işleyişini değiştirebilir veya sistem, kullanıcının yaptığı değişikliklerin sonuçlarını üretebilir. Görsel arayüz tasarımı bilgisayarlardan akıllı saatlere, arabalardan uçaklara kadar birçok üründe kullanılmaktadır. Görsel arayüz tasarımlarının karmaşıklığı arttıkça kullanıcıların onu kullanabilmeleri için gerekli bilgi ve beceri ihtiyacı da artmaktadır. Bu nesnelere göre Android Tasarlanan arayüzü kullanmak için pilotluk eğitimi gerekirken, akıllı saati kullanmak için kullanım kılavuzunu okumak yeterli olacaktır.

Kullanıcı arayüz tasarım süreci Şekil-1'de gösterilmiştir. Kullanıcı arayüz tasarımında ilk aşama Analiz ve Kullanıcı Aktivelerini anlamadır. Kullanıcıların arayüz kullanım alışkanlıkları, benzer uygulamaların arayüzlerinin analizi yapılarak kullanıcıların uygulamayı en iyi şekilde nasıl

kullanabileceği anlaşılmaya çalışılır. Tasarlanan arayüzlerin nasıl bir sıra veya olay ile görüntüleneceği bir sonraki aşamada gerçekleştirilir. Bu aşamadan sonra son kullanıcı tarafından test edilir ve ekran arayüzleri için aynı işlemler tekrarlanır. Ekran arayüz sayısı birden fazla ise bir prototip geliştirilerek test edilir. Eğer prototipin çalışmasında bir problem yoksa arayüz kodlanabilir.



Şekil-1: Kullanıcı Arayüzü Tasarım süreci [1]

İyi bir kullanıcı arayüzü kullanıcıların yapacağı işleri kolaylaştırılmalıdır. İyi bir arayüz tasarımı için de uzun bir arayüz tasarım süreci gerekebilir. Arayüz tasarım süreçlerinin uzaması yazılım şirketleri için maliyetlerin artması anlamına gelmektedir [1].GUI çizimleri veya eskiz çalışmalarının koda dönüştürülmesi programcılar tarafından yaygın olarak gerçekleştirilen bir görevdir. Bu görevin gerektirdiği zaman nedeniyle uygulamanın arka plan kodlarına daha az zaman kalmaktadır. Bu durum, programın istenilen kalitede olamamasına hatta istenilen sürede tamamlanamamasına yol açabilmektedir.

Serbest el kullanıcı arabirimi (UI) çizimini sıkı bir şekilde entegre etmek, günümüzün yazılım geliştirme sürecinde önemli bir boşluğu dolduracaktır. Özellikle, kağıt tabanlı prototipler üzerinde yineleme yaptıktan sonra, bugün UI tasarımcıları prototipleri Photoshop veya bir IDE gibi zahmetli ve maliyetli (GUI oluşturucularda bile) "yüksek doğrulukta" araçlarda manuel olarak yeniden oluşturmak zorundadır. Bu boşluk aynı zamanda UI tasarımcılarını ekiplerinin sonunda ürettiği koddan ayırır ve bu da genellikle programlama yapılarında uygulanması zor olan UI tasarımlarına yol açar[2].

Bunun önüne geçmek için bu çizimlerin makine öğrenmesi, yapay zeka veya görüntü işleme teknikleri ile arayüz koduna otomatik olarak dönüştürülmesi, bu aşamada harcanacak sürenin minimuma inmesini sağlayacaktır.

Son yıllarda bu alanda yapılan çalışmaların artması bu süreci daha da hızlandıracak ve doğruluk oranını artıracaktır. Farklı yöntemlerin denenerek diğer yöntemlere göre avantaj ve dezavantajların belirlenmesi, bu konuda çalışma yapacak araştırmacılar yol gösterecektir. Bu konuda yapılacak çalışmaların artması karmaşık arayüz tasarımlarının bile rahatlıkla arayüz koduna dönüştürülmesi sağlayarak son kullanıcı açısından daha kullanışlı programların ortaya çıkmasıyla neticelenebilir.

2. Otomatik Kot Oluşturma için Kullanılan Yöntem ve Teknikler

2.1 Makine Öğrenmesi

Makine öğrenmesinin tanımına bakıldığında; bir bilgisayarın doğrudan yönergeler olmadan öğrenmesine yardımcı olmak için matematiksel modelleri kullanma biçimi olduğu ifade edilmektedir. Makine öğrenmesi esasında yapay zekanın bir alt kümesi olarak değerlendirilir [3].Yapay zeka insan zekasını taklit eden sistemler veya makineler olarak tanımlanabilir. Makine öğrenimi ve yapay zeka genellikle bir arada değerlendirilir. Kimi durumlarda birbirinin yerine kullanılır ancak aynı anlama gelmezler. Tüm makine öğrenimi çözümleri yapay zekâ iken tüm yapay zeka çözümlerinin makine öğrenimi olmaması önemli bir ayrımdır[4].

Denetimli Makine Öğrenimi ve Denetimsiz Makine Öğretimi olmak üzere iki tür makine öğretimi vardır. Denetimli makine öğrenimine lojistik regrasyon, destek vektör makineleri (SVM) ve çoklu sınıf sınıflandırma gibi algoritmalar örnek verilebilir[4].

Denetimsiz Makine Öğreniminde ise belirli bir etikete sahip olmayan verilerin sınıflandırılması veya kümelenmesi durumu vardır. K-NN gibi algoritmalara buna örnek verilebilir[4].

2.2 Derin Öğrenme

Derin öğrenme bir veya daha fazla gizli katman içeren yapay sinir ağları ve benzeri makine öğrenme algoritmalarını kapsayan çalışma alanıdır. Derin öğrenme, özünde, makinelere insan zekasını taklit etmeyi öğretmek için yinelemeli yöntemlere dayanır. Yapay bir sinir ağı, bu yinelemeli yöntemi birkaç hiyerarşik düzey aracılığıyla gerçekleştirir. İlk seviyeler, nöronların basit bilgileri öğrenmesine yardımcı olur ve seviyeler arttıkça bilgi birikmeye devam eder. Her yeni seviye ile nöronlar daha fazla bilgi toplar ve bunları son seviyede öğrendikleriyle birleştirir. Sürecin sonunda, elde edilen bilgi bir nevi mantıksal düşünce örneği gibidir. Derin öğrenme yöntemleri kullanılarak tıbbi cihazlardan elde edilen görüntülerden hastalık tespiti, araçlarda plaka okumanın yanında marka model tespiti, hayvan, böcek, bitki, kuş ya da mikro düzeyde canlıların sınıflandırılması gerçekleştirilebilir [5][6].

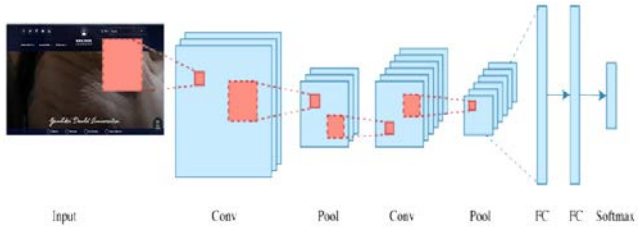
2.2.1 Evrişimsel Sinir ağları

Evrişimsel sinir ağları (Convolutional neural network - CNN), derin öğrenmenin bir alt dalıdır ve genellikle görsel bilginin analiz edilmesinde kullanılır. CNN, convolution ve pooling operatörlerini kullanır.

Bir CNN üç temel katman türüne sahiptir:

- Convolutional katmanı
- Pooling katmanı
- Fully-connected katmanı

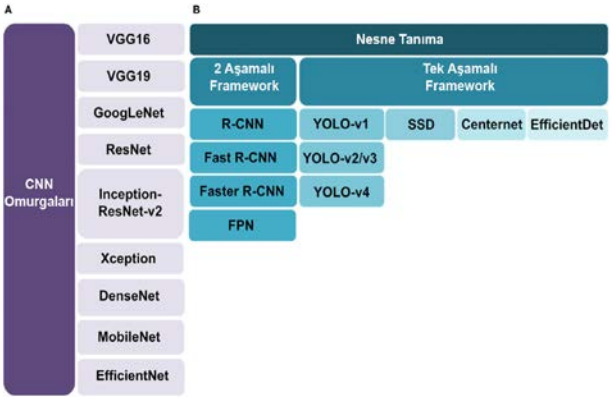
Art arda çok sayıda convolution+pooling katmanı olabilir. Bunu takiben birkaç tane fully connected katman bulunur. Çok etiketli sınıflandırma problemlerinde, en sonda softmax katmanı bulunur.



Şekil-2: Evrişimsel sinir ağları yapısı [7]

Şekil-2'de görüldüğü gibi pooling katmanının görevi, kayma boyutunu ve ağ içindeki parametreleri ve hesaplama sayısını azaltmaktır. Bu sayede ağdaki uyumsuzluk kontrol edilmiş olur. Fully-connected katmanı üç boyutlu girişi tek boyuta indirgeyerek alır ve bir sınıf etiketi elde eder. Softmax katmanı çıkış sınıflarının olasılık dağılımını hesaplar.

CNN Omurgaları ve Nesne Tanıma Algoritmaları Şekil-3'de gösterilmiştir. A kısmında yer alanlar CNN omurgalarıdır. Bunlar Nesne tanımada özellik çıkarımı elde etmek için kullanılabilirler. B kısmında yer alanlar ise Nesne Tanıma Algoritmalarıdır. Bunlar iki şekilde ele alınabilir. İki aşamalı çözümlere bakıldığında R-CNN, Fast R-CNN, Faster R-CNN ve FPN olduğu görülmektedir. Bu framework'ler hız açısından tek aşamalı çözümlere göre daha yavaş olsa da doğruluk açısından daha başarılı olabilmektedir [8]. Tek aşamalı çözümler ise YOLO versiyonlarını [9], SSD [10], CenterNet ve EfficientDet vardır [11]. Bunlar hızlı çalışırlar fakat doğruluk oranları daha düşük olabilmektedir.

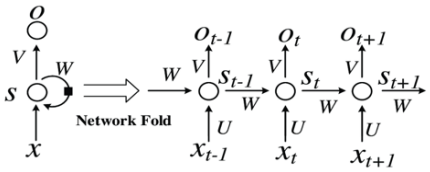


Şekil-3: Evrişimsel sinir ağları omurgaları [12]

Makaleler incelendiğinde CNN omurga ve algoritmalarının kullanıcı arayüz tasarım ya da çizimlerinden nesne tespiti yapmak ya da sınıflandırma amacıyla kullanıldığı görülmektedir.

2.2.2 Yinelemeli Sinir ağları (RNN)

Yinelemeli Sinir Ağları (Recurrent Neural Network=RNN) bir sonraki adımı tespit etmek için kullanılan Derin Öğrenme yapılarıdır. Diğer derin öğrenme yapılarından farkı "hatırlama" yapısıdır. Yani diğer sinir ağlarında her girdi birbirinden bağımsız iken RNN'lerde her girdi birbiri ile ilişkilidir. Bir sonraki adımı tahmin etmek için girdiler arasındaki ilişkiyi kullanır[13].

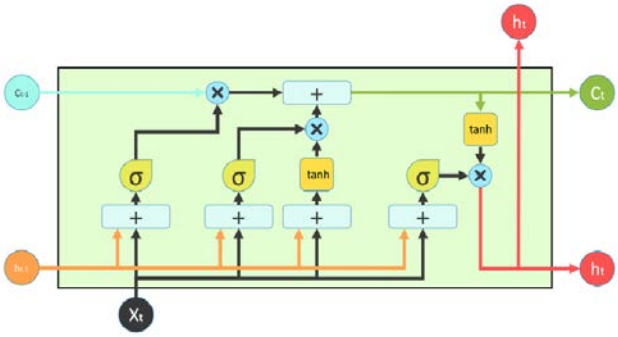


Şekil-4: Temel bir Yinelemeli sinir ağları yapısı[14]

RNN yapıları kullanıcıların el yazılarını, görüntü, konuşma tanıma gibi birçok alanda kullanılmaktadır.

2.2.3 Uzun Kısa Süreli Bellek (LSTM)

Yinelemeli Sinir Ağlarının kısa süreli olan bellekleri bazı problemlerde iyi sonuç vermemektedir. Bu sorunu oradan kaldırmak için araştırmacılar Sepp Hochreiter ve Jürgen Schmidhuber LSTM modelini önermişlerdir. Bir RNN yapısında tek bir katman yer alırken LSTM'nin yapısında birbiriyle iletişim halinde olan 4 katman (Unutma Kapısı-Giriş Kapısı-Çıkış Kapısı-Hücre Durumu) bulunmaktadır[15].



Şekil-5: Uzun Kısa Süreli Bellek Sinir Ağı'nın (LSTM) yapısı[16].

Unutma yapısı hangi bilgilerin unutulacağı veya tutulacağına karar veren kapıdır. Giriş kapısı Hücre Durumu güncellemesi yapar. Çıkış kapısı ise sonraki girişteki veriyi belirler. Bu aşamadan sonra hücre durumuna karar verilir.

3. Araştırma Yöntemi

Bu derleme makalesinde otomatik kot oluşturma teknikleri ile ilgili bilimsel çalışmalar geniş bir yayın taraması ile tespit edilmiş ve bu çalışmaların analizi yapılmıştır. Makale ve bildiriler, bilimsel makale arama motorlarında ve veri tabanlarında önceden belirlenen anahtar sözcüklere göre araştırılmıştır. Bu anahtar sözcükler, eş anlamları ile birlikte Çizelge 1'de verilmiştir.

Çizelge-1: Anahtar Sözcükler

Anahtar Sözcükler	Eş anlamlıları
Sketch	Çizim, mockup, screenshot
Deep learning	CNN, derin öğrenme
User interface	UI, GUI
Machine learning	Makine öğrenmesi
Software	App,mobile,web,css,html,android, ios
Code generation	Automatic code generation, Kot üretimi, otomatik kot üretimi

Çalışma kapsamında taranan veri tabanları; Google Scholar, ACM Digital Library, Scopus, IEEE Xplore Digital Library ve arXiv.org olarak belirlenmiştir. Veri tabanlarında yukarıdaki

çizelgede belirtilen anahtar sözcükleri barındıran çalışmalardan, 2017 ve sonrasında yayınlananlar ayıklanmıştır.

3.1 Derleme Çalışmasının Amacı

Bu derleme çalışması ile aşağıdaki araştırma sorularının cevabı aranmaktadır;

Q1. Görüntüden otomatik kot oluşturan çalışmalarda hangi yaklaşımlar mevcuttur?

Q2. Kullanılan yaklaşımlar ne tür bir platform ve girdi/çıktı verileri içindir?

Q3. Hangi veri setleri kullanılmıştır?

Q4. Yaklaşımlarda kullanılan yöntemlerin karakteristikleri nelerdir?

Q5. Çalışmaların başarımları nasıldır?

Bu araştırma sorularına göre çalışmalar incelenmiş ve analizi yapılmıştır.

4. Bulgular ve Tartışma

Bu bölümde incelenen makalelerin özeti, giriş-çıkış karakteristikleri, teknik analizleri, veri kümesi analizleri, başarım ölçütlerine göre analizleri yapılmıştır.

4.1 İncelenen Çalışmalar

Otomatik kot üretimi ile ilgili son 5 yıldaki çalışmalar incelenmiştir. Çalışmalarda farklı veri setleri, yöntemler ve ölçünler kullanılmıştır. Bu bölümde bu çalışmalardaki farklılıklar ortaya çıkarılmaya çalışılmıştır. Ayrıca incelenen çalışmaların benzer, ortak noktaları belirlenerek çalışmaların yönelimi gösterilmiştir.

İncelenen çalışmalar giriş yöntemine göre gruplandırılarak incelenmiştir. Giriş yöntemi olarak; Eskiz, Çizim, GUI Tasarımı, Ekran Görüntüsü veya Mockup görüntüsü kullananlar olarak gruplandırılmıştır.

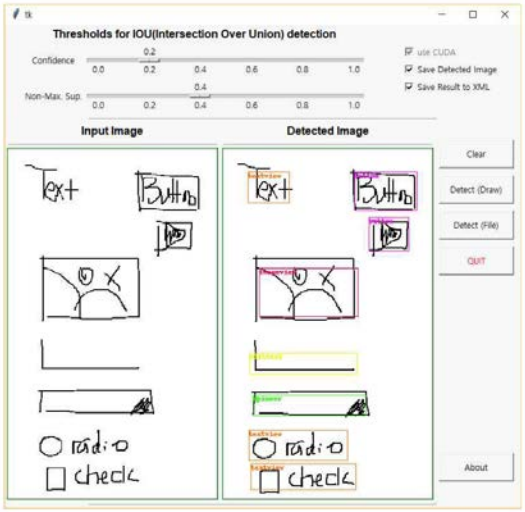
4.1.1 Eskiz veya Çizim Kullanan Çalışmalar

Bir yazılım arayüzünün oluşturulmasındaki ilk aşama kağıt çizimi ya da eskiz olarak nitelendirilebilir. Çünkü arayüz tasarımı oluşturulurken fikir değişiklikleri çok sık olmaktadır. Ayrıca kağıt ve kalem her an her yerde bulunabilecek materyaller olduğundan kolaylıkla herkes istenilen bir zamanda, herhangi bir program kullanma bilgisi olmasa bile oluşturabilir[17]. Yapılan çalışmaların amaçları incelendiğinde birçok çalışmada amacın eskiz tasarımından kot kısmına olan süreci hızlandırmaya yönelik olduğu görülmektedir. Bazı çalışmalarda ise eğitimde arayüz tasarımını öğretilmesine yönelik olduğu görülmektedir [18]. Eskiz veya çizim kullanan çalışmalara bakıldığında; bu çalışmalar ürettikleri çıktı kodlarına göre 2 alt grupta inceleyebildiği görülmüştür. Bu alt gruplar Web kodu üretenler ve Mobil ya da Android kodu üretenler olarak tespit edilmiştir.

Han ve ark. 2018 yılındaki CSSSketch2Code adını verdikleri çalışmalarında çizimden otomatik HTML kodu üreten bir çalışma gerçekleştirmişlerdir. Bu çalışmada diğerlerinden farklı olarak CSS kodu da üretilmiştir [19]. Şekil-6'da CSSSketch2Code çalışmasının girdi resmi olan el çizimi ve çıkış olarak koda dönüştürdüğü hali görülmektedir. Aşıroğlu ve ark. 2019 yılındaki çalışmalarında makine öğrenmesi yöntemlerini kullanarak Çizimden otomatik Web kodu üreten bir çalışma gerçekleştirmişlerdir. Çalışmada elle çizilen tasarımdan Web öğeleri tespit edildikten sonra HTML kodu üretilip konumuna göre yerleştirme işlemi gerçekleştirilmektedir[20]. 2022 yılındaki bir başka çalışmada kullanıcıların çizdiği web sitesi şablonunu Derin öğrenme çerçevelerinden VGG kullanarak HTML ve CSS koduna çeviren bir yaklaşım uygulanmıştır. Bu çalışmada özellikle web sitelerinin navbar kısımlarının koda dönüştürülmesi üzerinde durulmuştur [21]. Vitkare ve ark. elle çizilen bir arayüz tasarımını otomatik HTML koduna çeviren bir çalışma yapmışlardır. Bu çalışmada görüntü işleme teknikleri ve CNN modeli kullanmışlardır. Veri kümesi olarak Pix2Code veri kümesini kullanmışlardır[22].

Şekil-6: Han ve ark. CSSSketch2Code adını verdikleri çalışmada çizimden koda dönüştürme [19].

Adefris 2020 yılında gerçekleştirdiği çalışmasında derin öğrenme yöntemleri kullanarak düşük doğruluk oranına sahip grafik arayüz çizimlerinden otomatik kot oluşturan bir yaklaşım sunmuştur [23]. Yun ve ark. 2018 yılındaki çalışmalarında derin ağ yaklaşımlarını kullanarak çizimleri otomatik olarak Android uygulamaların arayüz tasarım dili olan XML'e çevirmişlerdir [24]. Yun ve ark. 2019 yılında bir başka çalışmada önceki çalışmalarını geliştirmişlerdir [25]. Mohian ve Csaallner 2020 yılındaki çalışmalarında da Çizimden otomatik Android kodu üretmişlerdir. Doodle2App adını verdikleri bu yazılımda derin öğrenme yöntemleri kullanmışlardır [2]. Jisu Park ve ark. da 2020 yılında diğer çalışmalar gibi Çizimden otomatik Android kodu üreten bir çalışma gerçekleştirmişlerdir [26]. 2020 yılındaki diğer bir çalışma olan Rahmadi ve Sudaryanto'nun çalışmalarında Derin Öğrenme Tekniği Kullanılarak Grafik Kullanıcı Arayüzü Bileşenlerinin Görsel Olarak Tanınması gerçekleştirilmiştir [27]. Jain ve ark. 2019 yılındaki çalışmalarında kendi oluşturdukları veri kümesi ile derin öğrenme yöntemlerinden ResNet50 kullanarak çizimden mobil uyumlu web koda dönüştüren bir yazılım geliştirmişlerdir [28] .



Şekil-7: Yun ve ark. 2018 yılındaki çalışmalarından bir görüntü [24]

Birçok çalışmada derin öğrenme yöntemleri kullanılırken görüntü işleme ve OCR kullanan çalışmalar da bulunmaktadır. Bu çalışmalardan S. Kim ve arkadaşlarının çalışmasında çizimler üzerinde kenar belirleme ve OCR gibi teknikler kullanılarak nesne tespiti yapılmıştır ve bu nesnelere göre Android Arayüz XML kodu üretilmiştir [29]. Ge 2019 yılında Elle çizilen eskizleri derin öğrenme yöntemleri kullanarak eskize en yakın GUI görüntüsünün kodunu oluşturan bir çalışma gerçekleştirmiştir. Çalışmada Android arayüz tasarımlarında kullanılan 7 widget, geliştirilen yöntem ile tespit edilip kodu üretilmektedir [30]. Baule ve ark. eskiz görüntülerinden otomatik Android uygulama tasarım koduna çeviren bir çalışma yapmışlardır. Sketch2aia adını verdikleri yazılım çizimleri AppInventor platformunda bir kullanıcı arayüzüne dönüştürmektedir. Yaptıkları çalışmanın amacını K-12 düzeyindeki öğrencilerin mobil uygulama öğrenmesini kolaylaştırmak olarak bildirmişlerdir. Yöntem olarak derin öğrenme modellerinden Yolo-Darknet kullanmışlardır. Yaptıkları testlerde yaklaşık %87 sınıflandırma başarısı elde etmişlerdir[18]. Jadhav, Gaikwad ve Gawande elle çizilen arayüz çizimlerinden DSL kodu üreten bir çalışma gerçekleştirmişlerdir.

4.1.2 GUI Tasarımı ya da Ekran Görüntüsü Kullanan Çalışmalar

Graphical User Interface (GUI) bir yazılımın ekran görüntüsü ya da bir tasarımcının bir yazılım için oluşturduğu arayüz tasarımı olarak adlandırılabilir. GUI kullanıcılar ile etkileşim açısından büyük öneme sahiptir[31]. Tasarımcılar da bunu dikkate alarak kullanıcıların kolay anlayabileceği ve kullanışlı arayüzler oluşturmaya çalışırlar[32]. Bununla birlikte kullanıcıların belirli arayüz alışkanlıkları da olabilmektedir. Yazılım firmaları da bu alışkanlıkları da göz önünde bulundurarak özgün bir tasarım geliştirmeye çalışırlar[33]. Bu sürecin daha hızlı ilerleyebilmesi için mevcut uygulamaların GUI tasarımı veya ekran görüntülerinden otomatik bir tasarım oluşturma yaklaşımı ortaya çıkmıştır. Bu alanda çalışma yapanlardan; Pang ve ark. 2020 yılındaki çalışmalarında giriş olarak GUI tasarımı kullanarak dikkate dayalı derin sinir ağı ile yeni bir söz dizimine duyarlı otomatik grafik kodu oluşturan SGui2Code ve HGui2Code adında iki model sunmuşlardır. Bu modellerde LSTM ve CNN kullanmışlardır. Görüntü tespiti için CNN kullanılırken, DSL kodunu çözmek için LSTM modelini kullanmışladır [34]. DSL kodu üreten diğer bir çalışma Yang Liu vd.'nin 2019 yılındaki çalışmasıdır. Bu çalışmada Mobil uygulamalar için otomatik platform bağımsız GUI kodu üreten bir yazılım geliştirmişlerdir [35]. C. Chen ve ark. 2018 yılında benzer şekilde GUI tasarım resminden GUI DSL iskeletine çeviren ve sinir ağlarını kullanan bir uygulama gerçekleştirmişlerdir [36]. 2019 yılında ise bu çalışmalarını geliştirerek Gallery D.C. adını verdikleri bir yazılım ile tasarımı algılayıp otomatik arama gerçekleştiren bir çalışma gerçekleştirmişlerdir [37]. Xiao ve ark. IconIntent adını verdikleri, Android Uygulamaları için Simge Sınıflandırmasına dayalı olarak Hassas UI Widget'larının Otomatik Tanımlanması çalışmasını gerçekleştirmişlerdir [38]

Sethi, Kumar ve Swami 2019 yılında Mix-NLP kullanarak tema algılama ve otomatik web kodu oluşturan bir çalışma gerçekleştirmişlerdir [39]. Benzer bir çalışma da Kolthoff, müşterinin istediği tasarımı NLP teknikleriyle analiz edip veri kümesindeki GUI tasarımlarında derin öğrenme teknikleri ile yeni bir GUI öncül tasarım oluşturan bir sistem geliştirmiştir [40]. Nguyen ve arkadaşları 2018 yılında Mobil uygulamaların derin öğrenme kullanıcı arayüzü tasarım kalıpları sunan bir çalışma gerçekleştirmişlerdir [41]. Mobil uygulama arayüz kodu dönüşümü gerçekleştiren diğer bir çalışmada Yun ve ark. CNN kullanarak GUI öğelerinin tespitini yapmışlardır [25]. Bir başka çalışmada J.Chen ve ark. Derin Öğrenme ve Bilgisayarlı görü teknikleri karşılaştırılarak kendi yöntemlerini sunmuşlardır [42]. Xie ve ark. 2020 yılındaki çalışmalarında CNN nesne tespiti ile GUI ekran görüntüsündeki nesneleri algılayıp bunları Android XML koduna çeviren bir çalışma ortaya koymuşlardır [43]. Mohian ve Csallner 2022 yılındaki çalışmalarında benzer olarak derin öğrenme yöntemleri kullanarak GUI ekran görüntüsünden Android arayüz kodu elde eden bir yaklaşım sunmuşlardır [44]. Chen ve ark. ise dikkate dayalı kodlayıcı-kod çözücü modeli aracılığıyla bir grafik kullanıcı arabiriminden kod oluşturma yaklaşımı sunmuşlardır. Giriş verisi olarak verilen mobil uygulamanın arayüz tasarımı ya da bir web sitesinin tasarım görüntüsü kullanılmaktadır. Bu giriş görüntüsü VGG-16 derin öğrenme modeli kullanılarak DSL kodu oluşturulmaktadır. Yapılan testler sonucunda hata oranları karşılaştırılarak benzer çalışmalardan web, Android ve iOS için de daha başarılı olduklarını göstermişlerdir [45]. Saravanan ve ark. web görüntüleri üzerinden HTML kodu üreten bir çalışma gerçekleştirmişlerdir. Bu çalışmada LSTM, Canny kenar tespiti, CNN VGG-19 yöntemlerini birleştiren hibrit bir model ortaya koymuşlardır[46]. Zhao ve ark. GUIGAN adını verdikleri çalışmada genetik algoritmalarını kullanarak otomatik GUI tasarımı oluşturmuşlardır. Fakat bu tasarımı koda dönüştüren bir yapı kurmamışlardır [47]. Xu ve ark. web arayüz resimlerinden otomatik HTML, CSS ve emmet kodu üreten bir çalışma yapmışlardır. Bu çalışmada derin öğrenme obje tespiti ile web arayüzündeki bileşenleri tespit etmişlerdir. Derin öğrenmede; CNN modeli olarak Faster R-CNN, RNN modeli olarak LSTM kullanmışlardır. Çalışmada image2emmet ile resimden emmet koduna dönüşüm işlemi yapılırken, image2html ile ise resimden HTML ve CSS koduna

dönüştürme yapılmaktadır. Diğer çalışmada farklı olarak kod oluşturucunun doğruluğunu sınıflandırmak için NLP yöntemleri kullanmışlardır. Oluşturulan kot ile olması gereken kot arasındaki benzerlik oranını ölçmek için Terim frekansı-Tersine Doküman Frekansı (TF-IDF) hesaplamışlardır [48]. Wu ve ark. tersine mühendislik ile UI görüntülerini ayrıştırma çalışması yapmışlardır. Bu çalışmada UI benzerlik araması, erişilebilirlik geliştirmesi ve UI'den kot oluşturma olmak üzere 3 uygulama geliştirmişlerdir. UI bileşenlerinin tespiti için Faster R-CNN modelini ResNet-50 algoritmasıyla birlikte kullanmışlardır. Çalışmada iOS uygulamaların ekran görüntülerini barındıran AMP veri kümesini ve Android uygulamaların ekran görüntülerini barındıran RICO veri kümesini kullanmışlardır [49].

4.1.3 Mockup Kullanan Çalışmalar

Mockup, bir servisin, bir ürünün veya bir arayüzün duyulara hitap edecek şekilde modellenip, sınanabildiği ya da sunulabildiği bir modelleme yöntemidir. Bir başka tanımla mockup, yapılan tasarımın görsel olarak zenginleştirildiği ve statik bir şekilde sunulduğu bir tasarım modelleme yöntemidir. Mockup görüntüden otomatik kod oluşturan çalışmalara bakıldığında Moran, Li ve ark. 2018 yılındaki çalışmalarında uygulama marketlerindeki mockup görüntülerinden Android'te uygulama tasarımındaki kullanılan öğeleri tespit ederek hiyerarşik GUI iskeletine çeviren bir çalışma gerçekleştirmişlerdir [50]. Moran ve arkadaşları 2018 yılındaki başka bir çalışmalarında önceki çalışmalarında kullandıkları yapıyı tasarım ihlallerini tespit etmek ve raporlamak için kullanmışlardır [51]. Çıkış olarak Android kodu üreten bir başka çalışmada Abdelhamid ve arkadaşları derin öğrenme çerçevelerinden YoloV5 kullanarak nesne tespiti gerçekleştirmişler ve Android GUI kodu üretmişlerdir [52].

Giriş görüntüsü olarak Mockup kullanan çalışmalardan Bouças ve Esteves'in 2020 yılındaki çalışmalarında Derin Öğrenme ile bu mockup web görüntülerini HTML vs CSS koduna dönüştüren bir çalışma gerçekleştirmişlerdir. Şekil 8'de çalışmada kullanılan giriş mockup görüntüsü (a) ve çıkış görüntüsü (b) gösterilmiştir [54].

4.2 Giriş-Çıkış Karakteristikleri

Bu başlıkta, araştırma sorularından Q2'nin cevabı verilmektedir. İncelenen çalışmalar Web veya Mobil platformlara yöneliktir. İncelenen çalışmaların 13 tanesi sadece Web, 8 tanesi sadece Android, 14 tanesi hem iOS hem Android, 3 tanesi hem Web hem de Mobil platforma yöneliktir.

İncelenen çalışmalar giriş görüntüsü olarak Çizim, UI tasarımı, Ekran görüntüsü ya da Mockup tasarımı kullanmaktadırlar. Çıkış olarak ise çalışma hangi platforma yönelik bir çalışma ise ona uygun bir çıktı kodu üretilmektedir. Web platformuna

(a) (b)

Şekil-8: Boçucas ve Esteves'in çalışmasından bir görüntü[53]

yönelik ise HTML, Mobil uygulama yönelik ise XML arayüz kodu vermektedir. Bazı çalışmalar ise hem Web hem de Mobil platforma yönelik olduğundan ona uygun bir çıkış sağlanmaktadır [27]. Çizelge 2'de incelenen çalışmaların girdi-çıktı karakteristikleri özetlenmiştir.

Çizelge-2: Çalışmaların Girdi- Çıktı Karakteristikleri

Referans	Platform	Girdi	Çıktı	Algılanan GUI öğeleri
[34]	Mobil	UI tasarımı	DSL kodu	-
[30]	Android	Çizim	Hiyerarşik GUI iskeleti	Textview, EditText, ImageView, Button, RadioButton, Switch ve Chechbox
[35]	Android, iOS ve Web	GUI ekran görüntüsü	İşaretleme benzeri DSL	-
[50]	Mobil	Mockup	Hiyerarşik GUI iskeleti	Textview, EditText, ImageView, Button, RadioButton, Switch ve Chechbox, RatingBar, Spinner
[39]	Web	Ekran Görüntüsü	HTML	Div, header, span
[23]	Web	Çizim	HTML	Image, checkbox, input form elemanlar
[40]	Mobil	NLP	Hiyerarşik GUI iskeleti	
[20]	Web	Çizim	HTML	textBox, dropdown, button ve checkbox
[24]	Android	Çizim	XML	-
[54]	Web	Mockup Resim	HTML, CSS	
[19]	Web	Çizim	HTML	-
[41]	Mobil	UI tasarımı	Android GUI kodu	
[52]	Mobil	Mockup çizimi	Android GUI kodu	

[25]	Mobil	GUI çizimi	Android GUI kodu	
[2]	Mobil	Çizim	Android Kodu	
[36]	Android	GUI tasarım resmi	DSL GUI framework	-
[37]	Android, iOS	GUI ekran görüntüsü	Android veya iOS kodu	-
[38]	Android	GUI ekran görüntüsü	Android Kodu	
[29]	Android	Çizim	Android Kodu	EditText, ImageView
[51]	Mobil	Mockup	GUI hiyerarşik iskeleti	
[42]	Mobil	GUI Ekran Görüntüsü	Android Kodu	
[44]	Mobil	GUI Ekran Görüntüsü	Android Kodu	
[28]	Android, iOS ve Web	Çizim	HTML	HTML etiketleri
[26]	Android	Çizim	Android Kodu	
[43]	Mobil	GUI Ekran Görüntüsü	Android Kodu	Edittext,
[27]	Mobil	Çizim	Android Kodu	
[21]	Web	Çizim	HTML vs CSS	Navbar
[45]	Android, iOS ve Web	GUI Ekran Görüntüsü	XML, HTML	
[46]	Web	GUI Ekran Görüntüsü	HTML	
[22]	Web	Çizim	HTML	
[18]	Android	Çizim	XML	
[47]	Android	GUI Ekran Görüntüsü	XML	
[48]	Web	GUI Ekran Görüntüsü	HTML, CSS, emmet	
[49]	iOS	GUI Ekran Görüntüsü	SwiftUI kodu	
[55]	Web	Çizim	DSL kodu, HTML	

4.3 Kullanılan Yöntem ve Teknikleri Analizi

Bu bölümde çalışmalarda kullanılan yaklaşım ve teknikler analiz edilmiştir. Ayrıca, derin öğrenme kullanan yaklaşımların varsa CNN modelleri ve kütüphaneleri belirlenmiştir.

Çizelge-3: Çalışmalarda Kullanılan Yöntem ve Teknikler

Referans	Yaklaşım	Kullanılan Teknikler	CNN model(ler)i ve Kütüphane	Programlama
[34]	Derin Öğrenme	CNN, RNN	LSTM	-
[30]	Derin Öğrenme	CNN	-	-
[35]	Derin Öğrenme	CNN, RNN	-	-
[50]	Hibrit	Bilgisayarlı Görü Teknikleri, CNN, kNN	kNN VE CNN'in beraber kullanıldığı kendi	MATLAB
[39]	Derin Öğrenme	CNN, NLP	-	-
[23]	Derin Öğrenme	CNN obje tespiti	Faster R-CNN, Tensorflow	Python
[40]	Derin Öğrenme	NLP, CNN	-	-
[20]	Hibrit	Filtreleme, CNN ve BiLSTM	Kendi CNN yöntemleri	
[24]	Derin Öğrenme	DNN	-	-
[54]	Hibrit	Makine Öğrenmesi ve CNN	YOLOv3, Darknet	Python
[19]	Derin Öğrenme	CNN obje tespiti, Bi-LSTM	CNN, Bi-LSTM, Mask R-CNN	-
[41]	Derin Öğrenme	RNN, GAN	-	-
[52]	Derin Öğrenme	CNN obje tespiti	YOLOv5	
[25]	Derin Öğrenme	CNN obje tespiti	YOLO	
[2]	Derin Öğrenme	RNN ile obje tespiti	RNN	
[36]	Derin Öğrenme	CNN, RNN	-	-
[37]	Hibrit	CNN, kenar belirleme	-	-
[38]	Görüntü İşleme, Bilgisayarlı	Resim Filtreleme, OCR	-	-
[29]	Görüntü İşleme, Bilgisayarlı	OCR, Kenar Belirleme	-	-
[51]	Hibrit	CNN, Bilgisayarlı görü teknikleri	-	-
[42]	Bilgisayarlı Görü ve Derin	OCR, CNN obje tespiti	Yolo, Centernet,	

[44]	Derin Öğrenme	CNN	Tensorflow	Python
[56]	Diğer	Tasarım Mining	-	-
[28]	Derin Öğrenme	CNN Obje Tespiti	SSD	C++
[26]	Derin Öğrenme	CNN Obje tespiti	Yolo, Darknet	
[43]	Hibrit	CNN, Görüntü İşleme teknikleri	Kendi Modelleri	
[27]	Derin Öğrenme	CNN Obje Tespiti	MobileNet	
[21]	Derin Öğrenme	CNN Obje Tespiti	VGG	
[45]	Derin Öğrenme	CNN Obje Tespiti	VGG-16	
[46]	Görüntü İşleme, Derin Ö	CNN, LSTM, Canny Kenar Belirleme	VGG-19	
[22]	Görüntü İşleme, Derin Ö	CNN, Kenar belirleme teknikleri		
[18]	Derin Öğrenme	CNN	Yolo, Darknet	
[47]	Derin Öğrenmesi	GAN, SeqGAN		
[48]	Derin Öğrenme	CNN, RNN - LSTM	Faster R-CNN	
[49]	Derin Öğrenme	CNN	Faster R-CNN, ResNet-50	
[55]	Derin Öğrenme	CNN		

Çalışmaların hangi platformlara yönelik olduğu ve hangi yöntemlere göre olduğu Çizelge 2 ve Çizelge 3'de gösterilmiştir. Çizelge 3'te ise yöntemlerin platformlara göre çalışma sayılarının bilgisi verilmektedir. Çizelge görüldüğü gibi Derin Öğrenme yöntemini kullanarak Mobil uygulama kodu üreten çalışmaların sayısı fazladır. Derin Öğrenme kullanan toplam 28 çalışma bulunmaktadır. Sadece Görüntü İşleme (IP) veya Bilgisayarlı Görü(CV) yöntemlerini kullanan 2 çalışma vardır. Bu iki yöntemi ya da Makine Öğrenmesi yöntemlerini de birleştirerek melez bir yöntem kullanan çalışma sayısı ise 8'dir.

Çizelge-4: Platform ve Yöntemlere göre Çalışma Sayıları

		Platform					Toplam
		Web	Mobil	Andorid	Mobil, Web	iOS	
Yöntem	Derin Öğrenme	7	8	7	3	1	**27**
	melez	3	5	0	0	0	**8**
	IP-CV	0	0	2	0	0	**2**
Toplam		**11**	**13**	**9**	**3**	**1**	**38**

Çizelge 4'de kolayca anlaşılacağı üzere çalışmaların büyük bir bölümünde derin öğrenme yöntemleri kullanılmıştır. Son yıllarda artan derin öğrenme yöntemi kullanımının bu durumun oluşmasında payı büyüktür.

4.4 Veri Seti Analizleri

Bu bölümde, makalelerde girdi olarak kullanılan GUI görüntü veri setleri ile ilgili çalışmalar ele alınmış ve bu çalışmalardaki veri setlerinin büyüklükleri karşılaştırılmıştır. Çizelge-5'de görüntü birleştirme problemi ile ilgili incelenen çalışmalara yer verilmiştir.

Deka ve ark. bu alanda çalışma yapan araştırmacılara yönelik oluşturduğu Rico veri kümesi bu konudaki araştırmalar için önemli bir materyal olmuştur [56]. Rico veri kümesinde 9.700 uygulamadan elde edilen 72.200 GUI ekran görüntüsü vardır. Daha önce benzer amaçlarla oluşturulmuş ERICA veri kümesinde 18,600 GUI ekran görüntüsü, Shirazi ve ark. çalışmalarında 29,000 ekran görüntüsü vardır [57], [58]. Zhang ve ark. 4068 iPhone uygulamasından 77,637 adet ekran görüntüsü toplamış ve etiketlemişlerdir. AMP adını verdikleri bu veri kümesini obje tanımlamak için kullanmışlardır [59].

Çizelge-5: Çalışmaların Veri Kümesi Analizi

Referans	Veri kümesi Adı	Veri kümesi Büyüklüğü
[34]	Pix2code	1750 ekran görüntüsü
[30]	Rico dataset	66.261 GUI görüntüsü
[35]	Google Play, AppStore Uygulama ekran görüntüleri	1.842.580 ekran görüntüsü
[50]	REDRAW	14.382 GUI görüntüsü
[39]	Bilgi Yok	-
[23]	Kendi veri kümeleri	562 UI resmi 11.152 UI bileşeni
[40]	Bilgi yok	-
[20]	Bilgi yok	-
[24]	Bilgi yok	600 GUI görüntüsü
[54]	Balsamiq mockups	1100 resim 1000 eğitim,100 test için
[19]	Kendi veri kümeleri	1500 web script
[41]	Bilgi yok	-
[52]	Kendi veri kümeleri	490 resim 390 eğitim, 100 sınama için
[25]	Kendi veri kümeleri	600 GUI bileşeni içeren 50 resim
[2]	Kendi veri kümeleri	11.500 resim
[36]	Andorid Uygulama görüntüleri	185.277 GUI görüntüsü
[37]	Google Play, AppStore Uygulama ekran görüntüleri	1.842.580 ekran görüntüsü
[38]	Kendi veri kümeleri	1576 eğitim için
[29]	Kendi veri kümeleri	50 ekran görüntüsü
[51]	REDRAW	14.382 GUI görüntüsü
[42]	Rico dataset	66.261 GUI görüntüsü

[44]	Rico dataset	66.261 GUI görüntüsü
[28]	Kendi veri kümeleri	2001 örnek
[26]	Bilgi Yok	-
[43]	Rico dataset	66.261 GUI görüntüsü
[27]	GUI CORE	330 Bileşen resmi
[21]	Kendi veri kümeleri	3000 navbar
[46]	Pix2code	1750 ekran görüntüsü
[18]	Kendi veri kümeleri	279 resim
[48]	Kendi veri kümeleri	9000 UI Bileşen görüntüsü
[49]	AMP ve RICO veri kümesi	130000 iOS ekranı 80000 Android ekranı
[55]	Bilgi yok	-

4.4 Başarım Ölçütleri Analizleri

Başarım ölçünleri akademik bir çalışmada bir yaklaşımın başarımını ölçmek için kullanılır [60]. Çalışmada kullanılan yönteme göre kullanılan ölçün değişkenlik gösterir. Derin Öğrenme kullanan çalışmalar giriş resimlerinden nesne sınıflandırmak için konfüsyon matrisi oluşturarak Doğruluk oranı (1), Precision (Tutturma)(2), Recall (Bulma)(3) ve F değeri(4) kullanmışlarıdır. Bu ölçünlerin hesaplanması için konfüsyon matrislerinden TP (True Positive), TN (True Negative), FP (False Negative), ve FP (False Positive) değerleri bulunmalıdır. Bu değerler bulunduktan sonra aşağıdaki formüllere göre bu ölçünler hesaplanır. Örneğin Liu ve ark. çalışmalarında Derin Öğrenme yöntemi kullandıklarından ölçün olarak doğruluk, F-Değeri, Bulma gibi ölçünler kullanmışlardır [35].

$$Accuracy(Doğruluk) = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision\ (Tutturma) = \frac{TP}{TP + FP} \quad (2)$$

$$Recall(Bulma) = \frac{TP}{TP + TN} \quad (3)$$

$$F\ Ölçüsü = 2\frac{Precision * Recall}{Precision + Recall} \quad (4)$$

BLEU (BiLingual Evaluation Understudy), makine tarafından çevrilmiş metni otomatik olarak değerlendirmek için bir ölçümdür. BLEU puanı, makine tarafından çevrilmiş metnin bir dizi yüksek kaliteli referans çevirisine benzerliğini ölçen sıfır ile bir arasında bir sayıdır[61]. BLEU otomatik kot üreten çalışmalarda kodun doğru üretilip üretilmediğini kontrol etmek için kullanılan bir başarım ölçünüdür. İncelenen çalışmaların üç tanesinde BLEU skoru başarım ölçünü olarak kullanılmıştır.

Makine Öğrenmesi tekniklerinin başarımını ölçmek için kullanılan diğer ölçünler ise METEOR, ROGUE-L ve SUM'dur. METEOR (Metric for Evaluation of Translation with Explicit Ordering), makine çevirisi çıktısının değerlendirilmesi için bir ölçüdür. Bu ölçün, unigram tutturma bulma değerlerinin harmonik ortalamasına dayalıdır ve bulma değeri, tutturma değerinden daha ağırlıklıdır[62]. ROUGE (Recall-Oriented Understudy for Gisting Evaluation), doğal dil işlemede otomatik özetleme ve makine çevirisi yazılımını değerlendirmek için kullanılan bir ölçündür[63]. SUM (Single Usability Metric) sistemin kullanılabilirliğini ölçmek için kullanılır[64].

Modellerin tahmin başarımını ölçmek için kullanılan diğer ölçünler ise hatanın kareler ortalaması (Mean Squared Error=MSE) ve hatanın mutlak ortalamasıdır (Mean Absoulte Error=MAE). Chen ve ark. 2019 yılındaki çalışmalarında MAE, MSE gibi ölçünleri kullanmışlardır [37]. Bazı çalışmalarda ise BLEU değeri ölçün olarak kullanılmıştır.

Nesne tespiti ve sınıflandırılmasında tahmin başarımını ölçmek için kullanılan başarım ölçünlerinden bir tanesi de genel ortalama tahmindir (Mean Average Precision=mAP). mAP değeri nesne tespiti ve sınıflandırılması için geliştirilen CNN modellerinde de başarım ölçünü olarak kullanılmıştır[8]. İncelenen çalışmaların dördünde başarım ölçütü olarak mAP kullanılmıştır.

Wu ve ark. çalışmasında graf ve ağaç yapısı kullanılmıştır. Bu yüzden çalışmada ölçün olarak graf ve ağaçlarına uygun ölçünler kullanmışlardır. Ağaç yapılarında yaprak düğümlü kenarların F1 skoru olarak F1 Leaves kullanılmıştır. GED (Graph Edit Distance), bir grafiği diğerine izomorf yapmak için gereken minimum "düzenleme" sayısını dikkate alan bir grafik benzerliği ölçümüdür[49].

Çizelge 6'da incelenen çalışmaların başarım ölçünleri ve sonuçları verilmiştir.

Çizelge 6. Çalışmaların Başarım Ölçün Analizi

Referans	Ölçün Adı	Sonuçlar
[34]	Hata oranı, F ve C değerleri	%3,4 hata oranı
[30]	Görsel olarak benzer uygulamaları bulma yaklaşımının potansiyeli	-
[35]	Doğruluk Tutturma Bulma	%85 Doğruluk
[50]	Doğruluk	%85 doğruluk oranı
[39]	Ölçüm yok	-
[23]	mAP	%95,5 mAP skoru
[40]	Ölçüm yok	-
[20]	Metot doğruluk ve validation accuracy	
[24]	mAP	%87 mAP değeri
[54]	BLEU Değeri Doğruluk	%83 doğruluk
[19]	Farklı diğer yaklaşımlarla karşılaştırma: -METEOR -ROUGE-L -BLEU -SUM	BLEU 0,69 METEOR 0,51 ROUGE 0,78 SUM 0,65
[41]	Ölçüm yok	-
[52]	Tutturma – Doğruluk Bulma F-Değeri	13 sınıf için ayrı ayrı ölçülmüştür.
[25]	Ölçüm yok	-

[2]	Doğruluk Zaman ölçümü	%93,9 doğruluk Sınıflandırma süresi: 526 ms yerelde (Android)
[36]	Doğruluk Zaman, Benzerlik durumları	%60,28 doğruluk oranı
[37]	Diğer Makine öğrenmesi yöntemleriyle doğruluk oranı karşılaştırması MAE (Mean Ablosute error) MSE (Mean squared error)	%85 doğruluk oranı
[38]	Tutturma Bulma F-Değeri	%88,2 tutturma değeri %87,3 bulma değeri %87,7 F-Skoru
[29]	Başarım sınaması yok	-
[51]	Detection Precision Classficaiton Precision Recall	
[42]	Tutturma Bulma F-Değeri Doğruluk	0,490 0,557 0,524 0,91
[44]	Doğruluk	%88
[28]	Öğe tespiti çıkarım zamanı	Çıkarım süresi 0,2 ila 1,7 saniye arasında sürmektedir.
[26]	mAP	Her bileşen için ayrı ayrı hesaplanmıştır.
[43]	F1 Değeri	0,524 F1 değeri
[27]	Doğruluk	%95 doğruluk oranı
[44]	Doğruluk	Sentetik Çizimlerde 0,91, Gerçek Çizimlerde 0.69 doğruluk
[46]	BLEU Değeri	Mean 0,89 Std 0,38
[18]	Doğruluk F1 Değer, mAP	%87 doğruluk 0,88 %87,72
[48]	Ortalama Doğruluk (AP), Recall, Precision	Her bileşen için ayrı hesaplanmış
[49]	F1 Değeri, GED, F1 Leaves	F1 0,60 F1 Leaves 0,67 GED 20,2
[55]	Doğruluk Oranı	%87

3. Sonuç ve Öneriler

Bu çalışmada bir çizimden veya ekran görüntüsünden otomatik kot oluşturan çalışmaların analizi yapılmıştır. Bu analizler 4 kategoride incelenmiştir. Çalışmaların giriş çıkış karakteristikleri, çalışmalarda kullanılan veri kümeleri, yöntemler ve kullanılan başarım ölçünlerine göre analizler gerçekleştirilmiştir.

İncelenen çalışmalarda kullanılan veri kümeleri genellikle daha önceki çalışmalarda oluşturulan veri kümeleridir. Bu alanda çalışma yapacak araştırmacılar da bu veri kümeleri kullanabilir ve geliştirebilir. Bilimin doğasına uygun olarak her bir araştırmacının sağladığı verilerle gelişen veri kümeleri sayesinde araştırmalardaki başarı artacaktır.

Metot açısından incelendiğinde otomatik kot üretimi tekniklerinin eski yıllarda görüntü işleme, bilgisayarlı görü yöntemlerini kullandıkları görülmüştür. Son yıllarda ise derin öğrenme ve makine öğrenimi yöntemlerine doğru bir eğilim gerçekleşmiştir. Derin öğrenmede ise CNN ile nesne tespiti kullanılmıştır. Farklı CNN çerçeveleri (framework) farklı sonuçlar vermektedir. Bazı çalışmalarda melez yöntemlerin daha etkili olduğu görülmüştür. Bu alanda çalışma yapacak araştırmacıların derin öğrenme ve bilgisayarlı görü tekniklerini ileri düzeyde bilmesi gerekmektedir.

Başarım metriği olarak derin öğrenme kullanan çalışmalar ağırlıkta olduğundan genellikle Tutturma, Bulma ve F-Değeri kullanılmıştır. Bu değerler her sınıf için ayrı hesaplandığı gibi ortalama olarak da hesaplanabilmektedir. Bu alanda çalışma yapan araştırmacılar bu ölçünleri öncelikli olarak tercih etmeleri tavsiye edilmektedir.

Sonuç olarak Çizimden, ekran görüntüsünden ya da Mockup görüntülerinden otomatik kot geliştirme çalışmaları gelişen bir çalışma alanıdır. Bu alanda çalışma yapacak araştırmacılar yayın taramasını özenli yaptığı takdirde nitelikli bir veri kümesi ve uygun bir yöntem ile başarılı sonuçlar elde edebilirler.

Kaynakça

[1] D. Stone, C. Jarrett, M. Woodroffe, and S. Minocha, *User interface design and evaluation*. Elsevier, 2005.

[2] S. Mohian and C. Csallner, "Doodle2App: Native app code by freehand UI sketching," in *Proceedings - 2020 IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems, MOBILESoft 2020*, Jul. 2020, pp. 81–84. doi: 10.1145/3387905.3388607.

[3] T. M. Mitchell and T. M. Mitchell, *Machine learning*, vol. 1, no. 9. McGraw-hill New York, 1997.

[4] T. M. Mitchell and T. M. Mitchell, *Machine learning*, vol. 1, no. 9. McGraw-hill New York, 1997.

[5] D. Ozdemir and M. S. Kunduraci, "Comparison of Deep Learning Techniques for Classification of the Insects in Order Level With Mobile Software Application," *IEEE Access*, vol. 10, pp. 35675–35684, 2022, doi: 10.1109/ACCESS.2022.3163380.

[6] M. F. Kunduraci and H. K. Örnek, "Vehicle Brand Detection Using Deep Learning Algorithms," *International Journal of Applied Mathematics Electronics and Computers*, pp. 0–3, 2019.

[7] M. Mandal, "Introduction to Convolutional Neural Networks (CNN)," *analyticsvidhya.com*, May 01, 2021.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks," pp. 1–14, 2016.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[10] W. Liu, "SSD : Single Shot MultiBox Detector SSD : Single Shot MultiBox Detector," no. December, 2015.

[11] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, 2019, pp. 6105–6114.

[12] R. Yang and Y. Yu, “Artificial Convolutional Neural Network in Object Detection and Semantic Segmentation for Medical Imaging Analysis,” *Frontiers in Oncology*, vol. 11. Frontiers Media S.A., Mar. 09, 2021. doi: 10.3389/fonc.2021.638182.

[13] L. R. Medsker and L. C. Jain, “Recurrent neural networks,” *Design and Applications*, vol. 5, pp. 64–67, 2001.

[14] M. Gao, G. Shi, and S. Li, “Online prediction of ship behavior with automatic identification system sensor data using bidirectional long short-term memory recurrent neural network,” *Sensors (Switzerland)*, vol. 18, no. 12, Dec. 2018, doi: 10.3390/s18124211.

[15] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] Y. Guo, X. Cao, B. Liu, and K. Peng, “El Nino index prediction using deep learning with ensemble empirical mode decomposition,” *Symmetry (Basel)*, vol. 12, no. 6, Jun. 2020, doi: 10.3390/SYM12060893.

[17] J. A. Landay and B. A. Myers, “Interactive sketching for the early stages of user interface design,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1995, pp. 43–50.

[18] D. Baulé, C. G. von Wangenheim, A. von Wangenheim, J. C. R. Hauck, and E. C. V. Júnior, “Automatic code generation from sketches of mobile applications in end-user development using Deep Learning,” *arXiv preprint arXiv:2103.05704*, 2021.

[19] Y. Han, J. He, and Q. Dong, “CSSSketch2Code: An automatic method to generate web pages with CSS style,” in *ACM International Conference Proceeding Series*, Oct. 2018, pp. 29–35. doi: 10.1145/3292448.3292455.

[20] B. Asiroglu *et al.*, “Automatic HTML Code Generation from Mock-upImages Using Machine Learning Techniques,” *IEEE*, 2019.

[21] T. Calò and L. de Russis, “Style-Aware Sketch-to-Code Conversion for the Web,” in *EICS 2022 - Companion of the 2022 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, Jun. 2022, pp. 44–47. doi: 10.1145/3531706.3536462.

[22] G. Vitkare, R. Jejurkar, S. Kamble, Y. Thakare, and A. P. Lahare, “AUTOMATED HTML CODE GENERATION FROM HAND DRAWN IMAGES USING MACHINE LEARNING METHODS”.

[23] B. B. Adefris, “Automatic Code Generation From Low Fidelity Graphical User Interface Sketches Using Deep Learning,” 2020.

[24] Y. S. Yun, J. Park, J. Jung, S. Eun, S. Cha, and S. S. So, “Automatic Mobile Screen Translation Using Object Detection Approach Based on Deep Neural Networks,” *Journal of Korea Multimedia Society*, vol. 21, no. 11, pp. 1305–1316, 2018, doi: 10.9717/kmms.2018.21.11.1305.

[25] Y. S. Yun, J. Jung, S. Eun, S. S. So, and J. Heo, “Detection of GUI elements on sketch images using object detector based on deep neural networks,” in *Lecture Notes in Electrical Engineering*, 2019, vol. 502, pp. 86–90. doi: 10.1007/978-981-13-0311-1_16.

[26] Jisu Park, Jinman Jung, Seungbae Eun, and Young-Sun Yun, “UI Elements Identification for Mobile Applications based on Deep Learning using Symbol Marker,” *The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)*, vol. 20, no. 3, pp. 89–95, Mar. 2020, doi: https://doi.org/10.7236/JIIBC.2020.20.3.89.

[27] A. A. Rahmadi and A. Sudaryanto, “Visual Recognition Of Graphical User Interface Components Using Deep Learning Technique,” Surabaya, Jan. 2020.

[28] V. Jain, P. Agrawal, S. Banga, R. Kapoor, and S. Gulyani, “Sketch2Code: Transformation of Sketches to UI in Real-time Using Deep Neural Network,” Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.08930

[29] S. Kim *et al.*, “Identifying UI Widgets of Mobile Applications from Sketch Images,” 2018.

[30] X. Ge, “Android GUI Search Using Hand-drawn Sketches.”

[31] W. O. Galitz, *The essential guide to user interface design: an introduction to GUI design principles and techniques*. John Wiley & Sons, 2007.

[32] R. Lal, *Digital design essentials: 100 ways to design better desktop, web, and mobile interfaces*. Rockport Pub, 2013.

[33] D. Gavalas and D. Economou, “Development platforms for mobile applications: Status and trends,” *IEEE Softw*, vol. 28, no. 1, pp. 77–86, 2010.

[34] X. Pang, Y. Zhou, P. Li, W. Lin, W. Wu, and J. Z. Wang, “A novel syntax-aware automatic graphics code generation with attention-based deep neural network,” *Journal of Network and Computer Applications*, vol. 161, Jul. 2020, doi: 10.1016/j.jnca.2020.102636.

[35] Y. Liu, S. Chen, L. Fan, L. Ma, T. Su, and L. Xu, “Automated Cross-Platform GUI Code Generationfor Mobile Apps,” 2019.

[36] C. Chen, T. Su, G. Meng, Z. Xing, and Y. Liu, “From UI design image to GUI skeleton: A neural machine translator to bootstrap mobile GUI implementation,” in *Proceedings - International Conference on Software Engineering*, May 2018, pp. 665–676. doi: 10.1145/3180155.3180240.

[37] C. Chen, S. Feng, Z. Xing, L. Liu, S. Zhao, and J. Wang, “Gallery D.C.: Design search and knowledge discovery through auto-created GUI component gallery,” *Proc ACM Hum Comput Interact*, vol. 3, no. CSCW, Nov. 2019, doi: 10.1145/3359282.

[38] X. Xiao, X. Wang, Z. Cao, H. Wang, and P. Gao, “IconIntent: Automatic Identification of Sensitive UI Widgets based on Icon Classification for Android Apps.”

[39] N. Sethi, A. Kumar, and R. Swami, “Automated web development: Theme detection and code generation using Mix-NLP,” in *ACM International Conference Proceeding Series*, Jun. 2019. doi: 10.1145/3339311.3339356.

[40] K. Kolthoff, “Automatic generation of graphical user interface prototypes from unrestricted natural language requirements,” in *Proceedings - 2019 34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019*, Nov. 2019, pp. 1234–1237. doi: 10.1109/ASE.2019.00148.

[41] T. T. Nguyen, P. M. Vu, H. V. Pham, and T. T. Nguyen, “Deep learning UI design patterns of mobile apps,” in *Proceedings - International Conference on Software Engineering*, May 2018, pp. 65–68. doi: 10.1145/3183399.3183422.

[42] J. Chen *et al.*, “Object detection for graphical user interface: Old fashioned or deep learning or a combination?,” in *ESEC/FSE 2020 - Proceedings of the 28th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Nov. 2020, pp. 1202–1214. doi: 10.1145/3368089.3409691.

[43] M. Xie, S. Feng, Z. Xing, J. Chen, and C. Chen, "UIED: A hybrid tool for GUI element detection," in *ESEC/FSE 2020 - Proceedings of the 28th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Nov. 2020, pp. 1655–1659. doi: 10.1145/3368089.3417940.

[44] S. Mohian and C. Csallner, "PSDoodle: Searching for App Screens via Interactive Sketching," Apr. 2022, doi: 10.1145/3524613.3527807.

[45] W. Y. Chen, P. Podstreleny, W. H. Cheng, Y. Y. Chen, and K. L. Hua, "Code generation from a graphical user interface via attention-based encoder–decoder model," *Multimed Syst*, vol. 28, no. 1, pp. 121–130, Feb. 2022, doi: 10.1007/s00530-021-00804-7.

[46] V. Saravanan, "Automated Web Design And Code Generation Using Deep Learning," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 6, pp. 364–373, 2021.

[47] T. Zhao, C. Chen, Y. Liu, and X. Zhu, "Guigan: Learning to generate gui designs using generative adversarial networks," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 748–760.

[48] Y. Xu, L. Bo, X. Sun, B. Li, J. Jiang, and W. Zhou, "image2emmet: Automatic code generation from web user interface image," *Journal of Software: Evolution and Process*, vol. 33, no. 8, p. e2369, 2021.

[49] J. Wu, X. Zhang, J. Nichols, and J. P. Bigham, "Screen Parsing: Towards Reverse Engineering of UI Models from Screenshots," in *The 34th Annual ACM Symposium on User Interface Software and Technology*, 2021, pp. 470–483.

[50] K. Moran, B. Li, C. Bernal-Cárdenas, D. Jelf, and D. Poshyvanyk, "Automated reporting of GUI design violations for mobile apps," May 2018, pp. 165–175. doi: 10.1145/3180155.3180246.

[51] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, and D. Poshyvanyk, "Machine learning-based prototyping of graphical user interfaces for mobile apps," *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 196–221, 2018.

[52] A. A. Abdelhamid, S. R. Alotaibi, and A. Mousa, "Deep learning-based prototyping of android gui from hand-drawn mockups," *IET Software*, vol. 14, no. 7, pp. 816–824, Dec. 2020, doi: 10.1049/iet-sen.2019.0378.

[53] T. Bouças and A. Esteves, "Converting web pages mockups to HTML using machine learning," 2020.

[54] T. Bouças and A. Esteves, "Converting web pages mockups to HTML using machine learning," in *WEBIST 2020 - Proceedings of the 16th International Conference on Web Information Systems and Technologies*, 2020, pp. 217–224. doi: 10.5220/0010116302170224.

[55] G. Jadhav, H. Gaikwad, and M. Gawande, "Generation Source Code from Hand Draw Image–A Machine Learning Approach," *Generation Source Code from Hand Draw Image–A Machine Learning Approach (February 25, 2022)*, 2022.

[56] B. Deka *et al.*, "Rico: A mobile app dataset for building data-driven design applications," in *UIST 2017 - Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, Oct. 2017, pp. 845–854. doi: 10.1145/3126594.3126651.

[57] B. Deka, Z. Huang, and R. Kumar, "ERICA: Interaction Mining Mobile Apps," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, Oct. 2016, pp. 767–776. doi: 10.1145/2984511.2984581.

[58] A. S. Shirazi, N. Henze, A. Schmidt, R. Goldberg, B. Schmidt, and H. Schmauder, *Insights into Layout Patterns of Mobile User Interfacesby an Automatic Analysis of Android Apps*. 2013.

[59] X. Zhang, L. de Greef, and S. White, "Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels," in *Conference on Human Factors in Computing Systems - Proceedings*, May 2021. doi: 10.1145/3411764.3445186.

[60] Y. Liu, Y. Zhou, S. Wen, and C. Tang, "A strategy on selecting performance metrics for classifier evaluation," *International Journal of Mobile Computing and Multimedia Communications (IJMCMC)*, vol. 6, no. 4, pp. 20–35, 2014.

[61] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[62] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 2005, pp. 65–72.

[63] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[64] J. Sauro and E. Kindlund, "A method to standardize usability metrics into a single score," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2005, pp. 401–409.