T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

# ÇİZİM TABANLI BİR İNSAN VE BİLGİSAYAR AYRIMI AMAÇLI TAM OTOMATİK GENEL TURING TESTİNİN TASARIMI VE ANALİZİ

Yüksek Lisans Tezi

Tezi Hazırlayan:
**Fırat OĞUZ**

İstanbul, 2018

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

# ÇİZİM TABANLI BİR İNSAN VE BİLGİSAYAR AYRIMI AMAÇLI TAM OTOMATİK GENEL TURING TESTİNİN TASARIMI VE ANALİZİ

Yüksek Lisans Tezi

Tezi Hazırlayan:
**Fırat OĞUZ**

Öğrenci No:
160820804

Danışman:
Dr. Öğr. Üyesi Cengiz ÖRENCİK

İstanbul, 2018

# YEMİN METNİ

Yüksek Lisans Tezi olarak sunduğum "ÇİZİM TABANLI BİR İNSAN VE BİLGİSAYAR AYRIMI AMAÇLI TAM OTOMATİK GENEL TURING TESTİNİN TASARIMI VE ANALİZİ" başlıklı bu çalışmanın, bilimsel ahlak ve geleneklere uygun şekilde tarafımdan yazıldığını, yararlandığım eserlerin tamamının kaynaklarda gösterildiğini ve çalışmamın içinde kullanıldıkları her yerde bunlara atıf yapıldığını belirtir ve bunu onurumla doğrularım. 29/11/2018

Fırat OĞUZ

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

# YÜKSEK LİSANS TEZ SAVUNMA SINAVI SONUÇ TUTANAĞI

**Beykent Üniversitesi**
**Fen Bilimleri Enstitüsü Müdürlüğü'ne,**

Aşağıda tez adı belirtilen yüksek lisans öğrencisi 160820804 no'lu FIRAT OĞUZ'un 15/11/2018 tarihinde yapılan tez savunma sınavı[1] sonucunda 100 dakika süreyle sunduğu ve savunduğu tezi hakkında[2] oybirliğiyle, kabul kararı verilmiştir.

Bilgilerinize saygılarımızla arz ederiz.

---

**Anabilim Dalı** : Bilgisayar Mühendisliği
**Programı** : Bilgisayar Mühendisliği
**Tez Başlığı**[3] : Çizim Tabanlı Bir İnsan ve Bilgisayar Ayrımı Amaçlı Tam Otomatik Genel Turing Testinin Tasarımı ve Analizi

---

| Tez Sınav Jürisi | Öğretim Üyesi | İmza |
|---|---|---|
| **Danışman** | : Dr. Öğr. Üyesi Cengiz ÖRENCİK | |
| **Üye** | : Dr. Öğr. Üyesi Ediz ŞAYKOL | |
| **Üye** | : Prof. Dr. Gökhan SİLAHTAROĞLU (İstanbul Medipol Üniversitesi) | |

[1] Jüri üyeleri, söz konusu tezin kendilerine teslim edildiği tarihten itibaren en geç bir ay içinde toplanarak öğrenciyi tez sınavına alır. Tez savunma sınav süresi en az 45, en çok 90 dakikadır. Jüri üyeleri, sınav öncesi yapılacak toplantıda, kendi aralarından danışman dışında bir üyeyi başkan seçer. Tez sınavı, tez çalışmasının sunulması ve bunu izleyen soru-cevap bölümünden oluşur. Tez sınavı, öğretim elemanları, lisansüstü öğrenciler ve alanın uzmanlarından oluşan dinleyicilerin katılımına açık ortamlarda gerçekleştirilir. Belirlenen günde yapılamayan jüri toplantısı, katılanların hazırladığı bir tutanakla enstitü yönetimine bildirilir. Bu durumda, jüri en geç on beş gün içinde toplanarak adayı tez savunma sınavına alır. (05 Ağustos 2017 tarihli 30145 sayılı Resmi Gazetede Yayınlanan Değişiklik-Madde 29-3)

[2] Tez sınavının tamamlanmasından sonra jüri, tez hakkında salt çoğunlukla "kabul", "düzeltme" veya "ret" kararı verir. Jüri başkanı, jüri üyelerince imzalanmış karar tutanağını, tez sınavını izleyen üç gün içinde ilgili enstitü yönetimine teslim eder. Tezi hakkında düzeltme kararı verilen öğrenci en geç üç ay içinde gerekli düzeltmeleri yaparak ve birinci fıkradaki usule göre tezini aynı jüri önünde yeniden savunur. Süresi içerisinde "düzeltme" savunmasına girmeyen öğrencinin enstitü ile ilişiği kesilir. ( Beykent Üniversitesi Lisansüstü Eğitim ve Öğretim Yönetmeliği-Madde 29-4)

[3] İleride doğabilecek aksaklıkların engellenmesi için tezin başlığının yazılması gerekmektedir.

# TEŞEKKÜR

Bu çalışmanın hipotez, teori ve uygulama aşamalarındaki öneri ve katkılarından dolayı danışman hocam sayın Dr. Öğr. Üyesi Cengiz ÖRENCİK'e, motivasyon ve destekleri için sevgili kardeşlerim Dicle ve Derya OĞUZ'a, son olarak tüm eğitim hayatımı borçlu olduğum sevgili annem Ayfer OĞUZ ve sevgili babam İsrafil OĞUZ'a sonsuz teşekkürlerimi sunarım.

Fırat OĞUZ

Kasım 2018

Adı ve Soyadı : Fırat OĞUZ
Danışmanı : Dr. Öğr. Üyesi Cengiz ÖRENCİK
Türü ve Tarihi : Yüksek Lisans, 2018
Alanı : Bilgisayar Mühendisliği
Anahtar Kelimeler : Yapay Zekâ, YZ, Görüntü İşleme

## ÖZ

## ÇİZİM TABANLI BİR İNSAN VE BİLGİSAYAR AYRIMI AMAÇLI TAM OTOMATİK GENEL TURING TESTİNİN TASARIMI VE ANALİZİ

Zamanında hayvanlardan aldığımız gibi bizim de bir gün bu dünyanın egemenliğini bizden sonra gelen üstün bir türe bırakmamız gerekebilir. Bu üstün türün kim ya da ne olacağını bilemeyiz ancak teknolojik gelişmeler dikkate alındığında şu an için en büyük adayın Yapay Zekâ olduğunu söyleyebiliriz.

Yapay zekâ ile insan zekâsı arasındaki farkın günden güne azalmasıyla otomasyona dayalı kötü niyetli girişimleri engelleyebilmek için makine ve insan arasında ayrım yapmak giderek zorlaşmaktadır.

Bu çalışma web ortamında insan makine ayrımı için kullanılan Captcha uygulamalarına görüntü analiz yöntemleri kullanarak çizim ile nesne sınıflandıran yenilikçi bir alternatif sunma amacı taşımaktadır.

Name and Surname : Fırat OĞUZ
Supervisor : Asst. Prof. Dr. Cengiz ÖRENCİK
Degree and Date : Master, 2018
Major : Computer Engineering
Key Words : Artificial Intellegience, AI, Image Processing

## ABSTRACT

## DESIGN AND ANALYSIS OF A DRAWING BASED CAPTCHA

As we took over from animals once upon in time, we may have to leave the dominion of this World, one day to a superior species that comes after us. We can not know who or what this superior species will be, but considering technological developments, we can say that the greatest candidate for now is Artificial Intelligence.

With the reduction day by day of the difference between artificial intelligence and human intelligence, it is increasingly difficult to distinguish between machine and human beings in order to prevent malicious initiatives based on automation.

The purpose of this work is to offer an innovative alternative that can classify objects by drawing using the image analysis methods to Captcha applications used for human machine separation in the web environment.

# İÇİNDEKİLER

**Sayfa No:**

## ŞEKİLLER LİSTESİ

**Sayfa No.**

# GÖRSELLER LİSTESİ

**Sayfa No.**

# 1.GİRİŞ

19.YY ortalarına doğru transistörün icadı ile sayısal elektronik çağının başlaması, emeklemekte olan zamanın teknolojisini günümüz şartlarına ulaştıran en büyük etken olmuştur. Gelişen teknolojinin insan odaklı olması zamanla yapay zekâlı sistemlerin ortaya çıkmasını sağlamıştır. Yapay zekâ algoritmalarının öğrenme temelli yeni nesil metotlar ile tasarlanması neticesinde de makine davranışlarında insana yakın sonuçlar alınabilmiştir.

Davranışsal olarak insana yakın sonuçlar veren sistemlerin tüm avantajına rağmen bu yapıların kötü niyetli kullanım ihtimalini de göz önünde bulundurmak gerekmektedir. Risk içeren bazı durumlarda makine ile insan arasında ayrım yapabilmek ihtiyaçtan ziyade bir zorunluluk haline gelmektedir. Genellikle web ortamında kullanılan Captcha sistemi insan ile makine ayrımına yönelik bir çözüm örneği olarak gösterilebilir.

Captcha sistemi makinelerin insanlar gibi düşünemediğini varsayarak bir insanın başarıp makinenin başaramayacağı öngörülen problemler tasarlar. Yapay zekânın gelişmesiyle davranışsal olarak insan ile makine arasındaki farkın azalması, mevcut Captcha sistemlerinde karmaşıklığı artırmak için revizyona gidilmesi veya yeni çözümler üretilmesi gerekliliğini ortaya çıkarmaktadır. Bu çalışma web ortamında insan makine ayrımı için kullanılan mevcut Captcha uygulamalarına görüntü analiz yöntemleri kullanarak çizim ile nesne sınıflandırabilen yenilikçi bir alternatif sunma amacı taşımaktadır.

Sekiz bölümden oluşan bu çalışma içerisinde sırasıyla değinilecek konular için gerekli temel kavramların tanımları yapılacak, halihazırda kullanılan bazı Captcha örnekleri incelenecek, sistemin mimari alt yapısını oluşturan görüntü işleme konusu ele alınacak, sistemin örnek uygulaması tasarlanacak, simülasyon üzerinden sistemin analizi gerçekleştirilecek, kullanıcı deneyimi araştırmasının sonuçları değerlendirilecek ve sistemin eleştirisi ile alternatif sistemlerin karşılaştırması yapılarak önerilen bu çözümün gerekliliği tartışılacaktır.

# 2.TEMEL BİLGİLER

## 2.1.ALGORİTMA

Algoritmanın tanımı, ilk defa Türk asıllı matematikçi Harezmî tarafından 9.YY'da yazılan "Hisab El-Cebir ve El-Mukabala" adlı kitap ile yapılmıştır (Parshall 1988). Algoritma bir görevin yapılması için tasarlanan koşullu adımlar bütünü şeklinde açıklanabilmektedir. Neticede bir komut seti olduğu için istenen sisteme entegre edilebilen algoritma, günümüz bilgisayar bilimlerinin de temelini oluşturmaktadır.

## 2.2.MAKİNE ÖĞRENİMİ

Algoritma tasarımı sürecinin öğrenme temelli bir yapay zekâ kullanımı ile gerçekleştirilmesi şeklinde tanımlanabilecek makine öğreniminde, eğitim kaynağı olarak kullanılacak referans veriler arasında gidişata, sonuca veya amaca yönelik korelasyon bulunmasıyla bir fonksiyon algoritması modeli oluşturulur. Yapay zekâ tarafından gerçekleştirilen nedensellik odaklı analiz neticesinde oluşan bu model farklı veriler üzerine uygulanabilir. Eğitim aşamasında amaca yönelik nitelikli ve doğru olan ne kadar fazla referans veri kullanılırsa, sistemin veriler aralarındaki bağlantıyı çözmesi veya alternatif bağlantılar bulması ile oluşturulan fonksiyon algoritmasının, uygulanacağı yeni veriler üzerinden tutarlı sonuçlar çıkarma olasılığı o kadar artacaktır.

Makine öğreniminde eğitim sonucu oluşturulan fonksiyon algoritmalarının içerikleri genellikle anlaşılmaz olurlar. Düşük seviyeli bir dil olarak da nitelendirilebilecek bu algoritmalarda çoğunlukla ağırlık veya katsayı gibi alanları temsil eden ondalık gösterimli rasyonel sayılar bulunmaktadır. Bu özellikleri sebebiyle sistemin içinde ne olduğu ve nasıl çalıştığı çoğu zaman bilinemez. Bu sistemin odak noktası alınan sonuç olduğu için işlem mantığı veya nasıl çalıştığı soruları kullanıcı için önemli değildir.

Makine öğrenimi referans verilere göre bir çıkarımda bulunarak davranışsal olarak bir kural oluşturmaya çalışır. Bu yapının doğru eğitimiyle günümüzde kullanılan görüntü/video/ses işleme, oyun, güvenlik, otomasyon gibi sistemlerde çok başarılı sonuçlar alınabilmektedir. Klasik algoritma tasarımına göre daha masrafsız, kararlı, güvenilir ve hızlı olduğu için makine öğrenimi ve türevleri gelecekte de yapay zekânın temel dinamiğini oluşturacaktır.

## 2.3.VERİ MADENCİLİĞİ

Veri, herhangi bir fonksiyon sonucunda ortaya çıkan ürün şeklinde tanımlanabilir. Neden sonuç ilişkisi içerisinde gözlemleyebildiğimiz her şeyin sonucunda bir veri oluşur. Bu veriye anlam katan istatistik türevi uygulamalar bütünü veri madenciliğinin temellerini oluşturmaktadır.

Veri madenciliği, veri yığını içerisindeki sonradan kullanılabilecek, bilinen ya da daha önce belirlenememiş korelasyonların, kuralların, sapmaların, örüntülerin veya benzeri nitelikli verilerin tespit edilmesi ve değerlendirilmesi şeklinde tanımlanabilir. Veri madenciliği sosyolojik araştırmalarda, pazarlama ve reklamcılık sektörlerinde, risk ve kişilik analizlerinde, bilişim destekli yönetim sistemlerinde, pazar araştırmaları ve finans sistemleri gibi birçok alanda kullanılmaktadır. Veri madenciliğinde kullanılan başlıca yöntemler ise sınıflandırma, kümeleme ve birliktelik kurallarıdır.

Sınıflandırma, değişkenlerin daha önceden belirlenmiş karakteristik özellikler doğrultusunda gruplandırılması veya halihazırda bulunan ve özelliklerine uygun ya da yakın olan bir gruba sonradan dahil edilmesidir. Linear-Nonlinear Support Vector Machine, Data Stream Mining, Naive Bayes Classifier, Decision Tree Learning, K-Nearest Neighbors veri madenciliğinde kullanılan başlıca sınıflandırma algoritmaları arasında yer almaktadır (Raval 2012).

Kümeleme, sınıflandırmadan farklı olarak önceden belirlenmiş herhangi bir grup kriteri olmadan değişkenlerin kendi aralarında ortak özellikleri doğrultusunda

dinamik olarak gruplandırılmasıdır. Bu özellikler sayısal nicelikler olabileceği gibi ortak nitelikler veya elemanlar olarak da belirlenebilir (Raval 2012). Örneğin K-Nearest Neighbors yani En Yakın Komşu algoritmasında veriler koordinat sisteminde gösterilir. Kümelenmesi istenen veri Öklid, Manhattan, Minkowski veya Gauss gibi bir uzaklık formülü kullanılarak koordinat sisteminde kendisine en yakın gelen küme merkezinin olduğu gruba dahil edilir (N. Suguna 2010).

Veriler arasındaki ilişkilerin analizi esasına dayanan birliktelik kuralları genellikle pazar sepeti problemi üzerinden örneklendirilir (Leskovec, Rajaraman ve Ullman 2014). Sepeti oluşturan elemanlar arasındaki bağıntın analizi ikame veya tamamlayıcı özelliklerin tespitiyle yöneylem araştırmalarına olanak sağlar.

Veri madenciliğinin uygulanması aşamasında istatistiki yöntemler dışında yapay zekâ türevi olan makine öğrenimi algoritmaları da kullanılabilmektedir. Algoritma eğitimleri amaç ve mevcut verinin niteliği doğrultusunda gözetimli veya gözetimsiz olarak yapılabilir.

Gözetimli öğrenme metodunu kullanabilmek için yeterli sayıda nitelikli örnek veriye sahip olmak gerekir. Öğrenme algoritması basitçe faklı veri kümeleri arasında aynı amaca, gidişata, benzerliklere veya sonuca sahip ya da bunlara ulaşılabilecek verilerin korelasyonlarını tespit ederek hepsinde geçerli ortak bir fonksiyon tanımlamaya çalışır. Başarılı bir eğitim sonrasında oluşturulan fonksiyonun yeni veri kümesi üzerinde kullanılmasıyla tutarlı sonuçlar alması beklenir (Donalek 2011).

Gözetimsiz öğrenme ise kullanılan algoritma doğrultusunda veri kümesinde yer alan değişkenlerin kendi aralarındaki ortak özelliklere göre gruplandırılmasıdır. En bilinen örneklerinden K-Ortalama algoritması sırasıyla koordinat sisteminde gösterilen elemanlar arasında küme merkezlerinin belirlenmesi ve yapı kararlı duruma gelinceye kadar döngü içerisinde merkeze yakınlığa göre elemanların sınıflandırılması ve yeni merkezin belirlenmesi şeklinde çalışır (Donalek 2011).

## 3.İNSAN BİLGİSAYAR AYRIMI TESTİ (CAPTCHA)

Teknolojinin gelişmesiyle küreselleşen dünyada bilgiye erişimin giderek kolaylaşması bilgi güvenliğini de bir o kadar zorlaştırmaktadır. Otomasyon temelli çalışan sistemlerde bilgi güvenliğine karşı girişimleri önleyebilmek için insan ile yapay zekânın ayrımının yapılması risk içeren bazı durumlarda bir zorunluluk haline gelebilmektedir.

Gelecek bilimciler tarafından bazı platformlarda makinelerin insan zekâsına 2050 yılından önce ulaşacağı ve kısa bir süre sonrasında insanı zekâ olarak geçeceği öngörülerinde bulunulmaktadır. (SXSW 2017). Bu öngörülerin temelinde son yüzyılda, o zamana kadarki tüm insanlık tarihinden daha fazla teknolojik gelişim gösterilmesinin payı büyüktür.

Yapay zekânın bu gelişimi sebebiyle makine ile insanı ayırt edebilmek için giderek daha karmaşık testlere ihtiyaç duyulmaktadır. İlk olarak Bilgisayar bilimlerinin kurucu olarak kabul edilen Alan Turing tarafından 1947 yılında yapılan ve kendi adı ile bilinen test ile makine ve insan ayırt edilmeye çalışılmıştır (Turing 1950, 433). Testin amacı düşünebilen makinelerin yeterliliğinin tescillenmesi olsa da testin karmaşıklaştırılıp hata toleransının düşürülmesiyle insan ile makine teoride ayırt edilebilmektedir. Ancak bu mimarinin temelindeki makinelerin insanlar gibi düşünemeyeceği ve insandan farklı davranışlar sergileyeceği görüşü ilerleyen zamanda sentetik zekânın insan zekâsına yaklaşmasıyla geçerliliğini yitirecek ve benzeri davranış tabanlı testlerden çok daha farklı çözümlerin üretilmesi kaçınılmaz bir zorunluluk olacaktır.

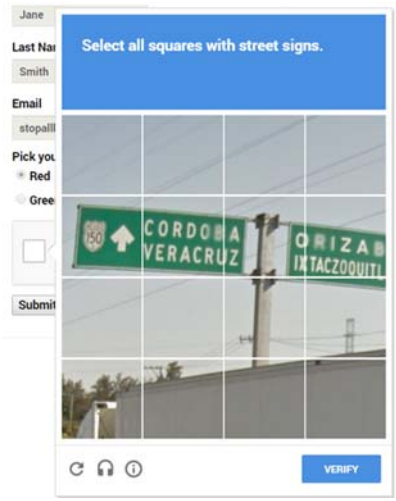
Düşünebilen makineler halen bilgisayar bilimlerinde bir araştırma konusu olsa da insan ve makine ayrımına yönelik davranışsal farklılıkları esas alan pratikteki bazı örnekleri günlük hayatta on yıldan uzun bir süredir kullanılmaktadır (M. B. Luis von Ahn 2003). Genellikle web ortamında görmeye alışık olduğumuz, bot adı verilen otomasyon temelli uygulamalar ile insanı ayırt etme amacı taşıyan Captcha bu konuda en çok bilinen sistemlerin başında gelmektedir. Completely Automated Public Turing

test to tell Computers and Humans Apart baş harflerinden oluşan bu sistem Carnegie Mellon Üniversitesi tarafından 1997 yılında tasarlanmıştır. Tasarımcıları, Mark D. Lillibridge, Martin Abadi, Krishna Bharat, Andrei Z. Broder, Eran Reshef, Gili Raanan ve Eilon Solan'dır (M. B. Luis von Ahn 2003).

Captcha sisteminin ilk örnekleri font ve karakteri deforme olmuş harf veya rakamların olduğu görseller şeklindedir. Görselde bulunan harf ve rakamların klavye karşılıklarının istenen yere yazmasıyla test geçilmektedir. Tasarımda bir insanın bunu yapabileceği ancak zamanın teknolojisine sahip bir makinenin bunu yapamayacağı öngörülmüştür. Bu sebeple görüntü işleme yöntemleri ve yapay zekânın gelişmesine karşı zamanla daha kompleks versiyonları ortaya çıkmıştır.

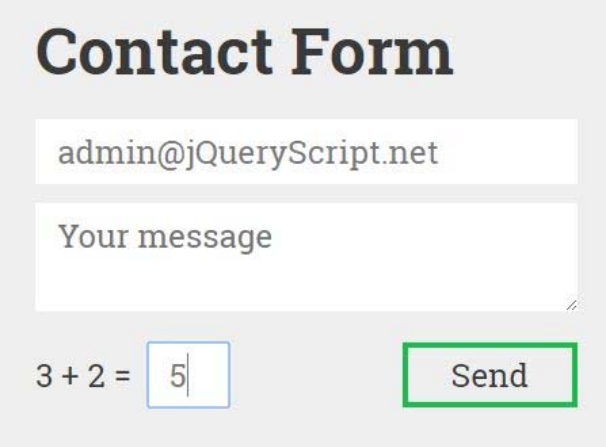
Günümüzde kullanılan bazı Captcha örnekleri:

1. Google ReCaptcha: Bu sistemin yapısında derin öğrenme çatısı altındaki çeşitli makine öğrenimi algoritmaları kullanılmıştır. Kullanıcı tarafından verilen cevaplar işlenerek eğitim serisine eklenmekte ve geri beslemeli bir yapı içerisinde algoritmanın optimizasyonu sağlanmaktadır. Örnek görselde yol işareti içeren karelerin kullanıcı tarafından seçilmesi istenmektedir (B. M. Luis von Ahn 2008) (Google Inc. 2018).



Görsel 1. Google ReCaptcha

2. Math Captcha: Kompleks işlemler talep eden farklı örnekleri de bulunan bu Captcha basit bir matematik işleminin sonucunu istemektedir (Piskin 2017).

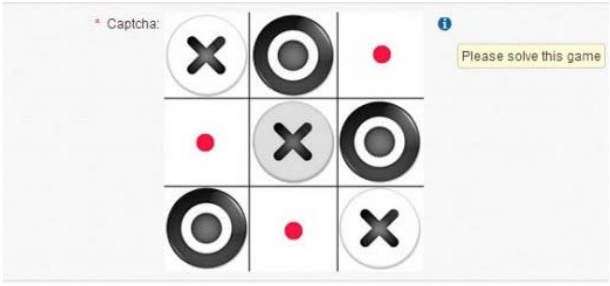



Görsel 2. Math Captcha

3. Fancy Captcha: Cevap olarak istenilen simgenin, toplu halde verilen simgeler arasından seçilerek yuvarlak alana sürüklenip bırakılması ile test geçilmektedir (TheGamesDB 2013).


Görsel 3. Fancy Captcha

4. XOX Captcha: Bu Captcha kullanıcıdan Tic-Tac-Toe olarak da bilinen XOX oyununu kazanmasını istemektedir. Doğru hamleler ile test geçilebilmektedir (AndrewP 2017).

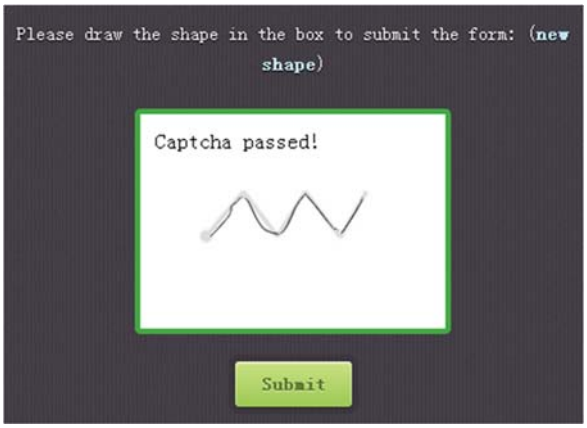


Görsel 4. XOX Captcha

5. Drag Sort Captcha: Kullanıcıdan karışık olarak verilen rakamların ardışık olarak sıralanması istenmektedir. Doğru sıralamayla test geçilebilmektedir. (Cody 2009).

Görsel 5. Drag Sort Captcha

6. Motion Captcha: Çizime yönelik bu Captcha uygulaması verilen referans çizgilerinin üzerinden geçilmesini istemektedir. Yapılan çizimin yeterli doğrulukta olması halinde test geçilebilmektedir (Crowcroft 2011).



Görsel 6. Motion Captcha

# 4.GÖRÜNTÜ İŞLEME

Teknolojik gelişmeler için genelde doğadan ilham alınır. Fotoğraf makinesi için göz yapısının, uçak için kuş kanatlarının veya radar sonar gibi cihazlar için yarasa ve yunusların referans alınması bunlara örnek olarak gösterilebilir. Ancak makine arayüzü için günümüzde kullanılan ekranlar teknoloji olarak bu genellemenin dışında yer almaktadır. Sebebi ise mevcut ekran teknolojisinin insan gözünün eksik ve kusurlu yönlerini manipüle edecek şekilde tasarlanmış olmasıdır. Bu manipülasyon sayesinde insan beyni piksellere anlam katarak bir bütün olarak algılar (Watson 1993).

Piksellerin sadece insan beyni için anlamlı olması görsel tabanlı yazılım uygulamalarının tasarımını zorlaştırmaktadır. Bu tür uygulamaların hayata geçirilebilmesi için insan beyninin piksel yığınına sonradan kattığı derinlik ve perspektif gibi boyutların da algısal olarak taklit edilebilmesi gerekmektedir.

Görüntülerin analizi için genellikle sayısal sinyal işlemenin alanına giren görüntü işleme yöntemlerine başvurulur. Arasında kendisi kadar boşluk bulunan iki aynı çaptaki yuvarlak karartı ve aradaki boşluğun iki karartı çapı altında yer alan karartı çapı genişliğindeki paralel bir çizgi, basit bir yüz tanıma algoritması için çekirdek oluşturabilir. Görüntü işleme yöntemleri üzerinden algoritmanın uygulanabilmesi için kabaca bu karakteristik özelliklerin bir sınıf haline getirilmesi ve sınıf kriterlerine uygun olan piksellerin bir arada bulunup bulunmadığının tespit edilmesi gerekmektedir. Bir piksel yığını içerisinde bunu gerçekleştirerek insan beyninin yaptığı gibi nesneler oluşturmaya, anlam katmaya ve sınıflandırmaya çalışan algoritmalar yirmi yılı aşkın süredir kullanılmaktadır (Dana Harry Ballard 1982).

Görüntü işleme için kullanılan bazı yazılımlar:

1. Microsoft Computer Vision API: Microsoft'un Azure çatısı altında bulunan görüntü işleme ve analizi üzerine yapılabilecek hemen her işlemin arayüz ile sunulduğu bu uygulama nesne tanıma, karakter tanıma (OCR), ürün ve önemli

yerleri tanıma gibi gelişmiş işlevleri barındırmakla birlikte gerçek zamanlı video işlemeyi de desteklemektedir. (Microsoft Co. 2018).

2. Google CloudVision API: Google firmasının sunduğu bulut tabanlı bu uygulama karakter tanıma (OCR), yüz tanıma, logo tanıma, nesne ve hayvan tanıma gibi işlevleri sunmakla birlikte görüntü içeriğiyle ilgili yorum yaparak filtreleme de gerçekleştirebilmektedir (Google Inc. 2018).
3. Amazon Rekognition: Amazon tarafından sağlanan bu hizmet görüntü ve gerçek zamanlı video işleme konusunda çözümler sunmaktadır. Bünyesinde yüz, karakter (OCR), nesne tanıma, sahne veya olay tanımlama ayrıca uygunsuz içerikleri tespit etme gibi gelişmiş özellikleri barındırmaktadır (Amazon Inc. 2018).
4. OpenCV: Açık kaynaklı bir görüntü işleme kütüphanesi olan OpenCV genellikle C++, Java ve Python programlama dilleri ile birlikte kullanılmaktadır. Kütüphane olması herhangi bir platforma bağlı olma zorunluluğunu da ortadan kaldırmaktadır. Bünyesinde görüntü-video işleme ve makine öğrenimiyle ilgili çok sayıda algoritma barındıran OpenCV tamamen yönetilebilir olması sebebiyle diğer uygulamalara kıyasla biraz daha uzmanlık gerektirebilmektedir. (OpenCV Foundation 2018).
5. SimpleCV: Python scriptleri ile programlanabilen açık kaynaklı bu uygulama görüntü-video işleme ve analizi için kullanılmaktadır. İçerisinde kullanıma hazır birçok araç barındıran SimpleCV kullanım kolaylığı ile ön plana çıkmaktadır (SimpleCV 2018).

# 5.ÇİZİM TABANLI İNSAN BİLGİSAYAR AYRIMI TESTİ

Teze konu olan yapı görüntü analizi ile Captcha sistemini birleştirmeyi amaçlamaktadır. Çalışma prensibi yapılan çizim üzerinden nesne sınıflandırma esasına dayanmaktadır.

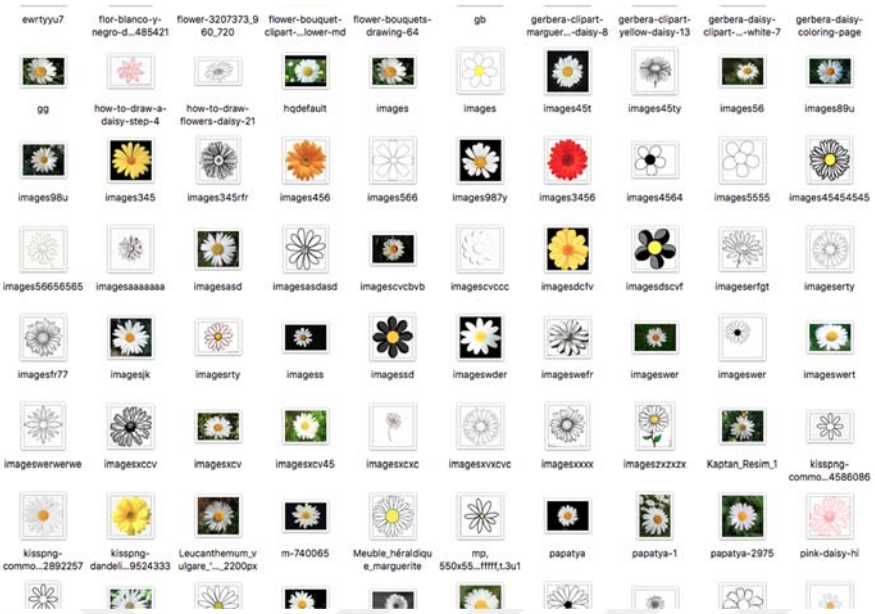
Web arayüzü ile çalışacak olan bu sistemde sayfa üzerine yapılacak nesne çizimi görüntü işleme yöntemleri kullanılarak daha önce sınıflandırması yapılmış nesne şablonu ile karşılaştırılacaktır. Çizim ve şablonun karakteristik olarak uyuşması halinde Captcha onaylanacaktır.

Sistemin tasarımında lisansı itibariyle bütün projelerde kullanılabilmesi, platformlar arası bir yapıya uyum sağlayabilmesi ve tüm sisteme müdahale edebilme serbestisi sunması nedeniyle OpenCV tercih edilmiştir (OpenCV Foundation 2018).

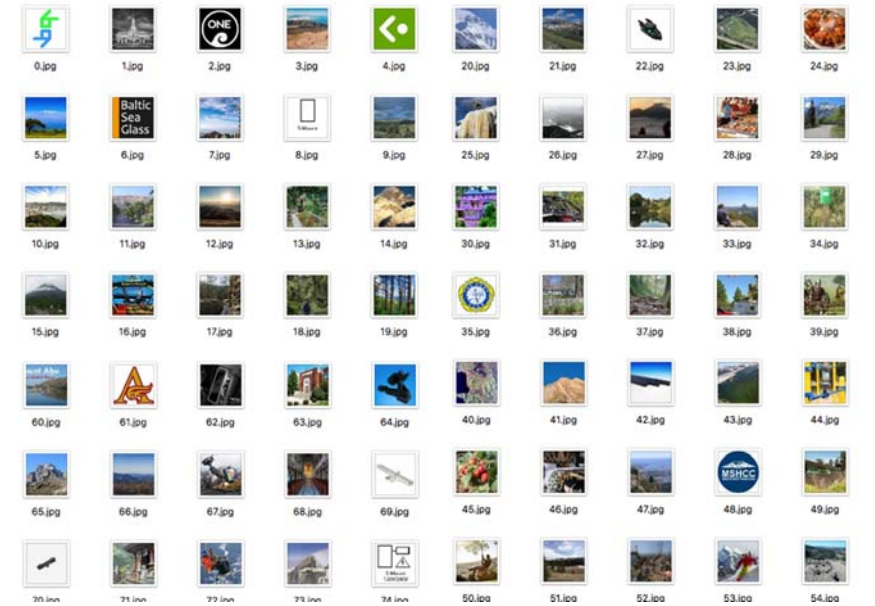
OpenCV, görüntü analizi alt yapısında kullanılabilecek ve arasında makine öğrenimi metodu uygulanabilen nesne sınıflandırıcılarının da bulunduğu geniş bir kütüphane sunmaktadır (OpenCV Foundation 2018). Daha hızlı, daha tutarlı ve daha güvenilir olması sebebiyle sınıflandırıcı uygulama örneğinde makine öğrenimi metodu kullanılmıştır.

## 5.1.SINIFLANDIRICI EĞİTİMİ HAZIRLIĞI

Nesne tanımayı sağlayacak sınıflandırıcının makine öğrenimi yöntemleri ile gerçekleştirilen eğitim aşamasında referans veri olarak, içerisinde tespit edilmesi istenen nesnelerin bulunduğu ve bulunmadığı birbirlerinden farklı görseller kullanılmaktadır. Bu görseller pozitif ve negatif resimler olarak adlandırılır.




Görsel 7. Pozitif Resimler



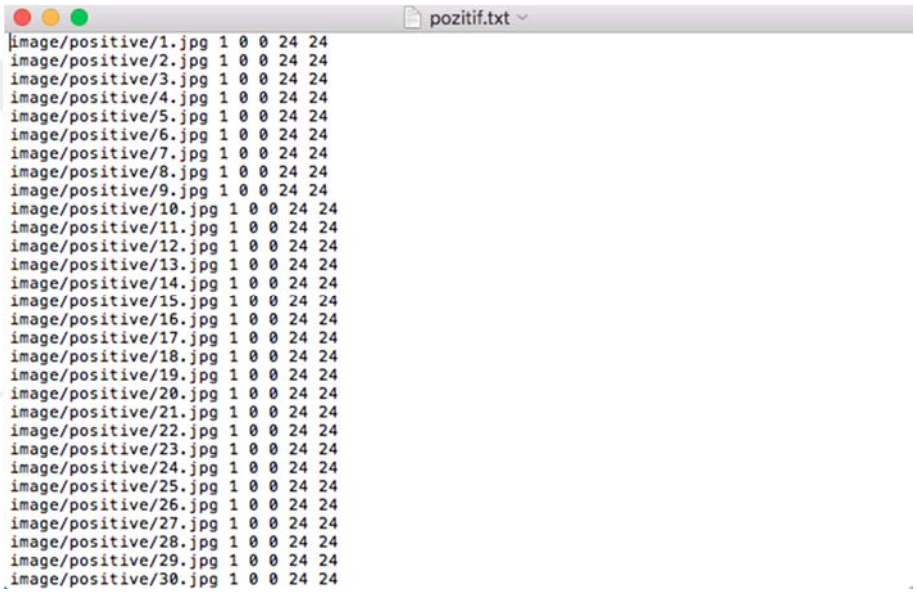

Görsel 8. Negatif Resimler

OpenCV, eğitim aşamasında basitçe pozitif resimlerde olup negatif resimlerde olmayan nesneleri tespit ederek sınıflandırmaya çalışmaktadır.

Başarılı bir sınıflandırıcı eğitimi için birkaç bin ile ifade edilebilecek sayılarda pozitif ve negatif resme ihtiyaç duyulabilmektedir. Diğer gözetimli makine öğrenimi sistemlerinde olduğu gibi burada da en önemli olan nokta eğitim aşamasında amaca uygun ve nitelikli verilerin doğru parametreler ile kullanılmasıdır.

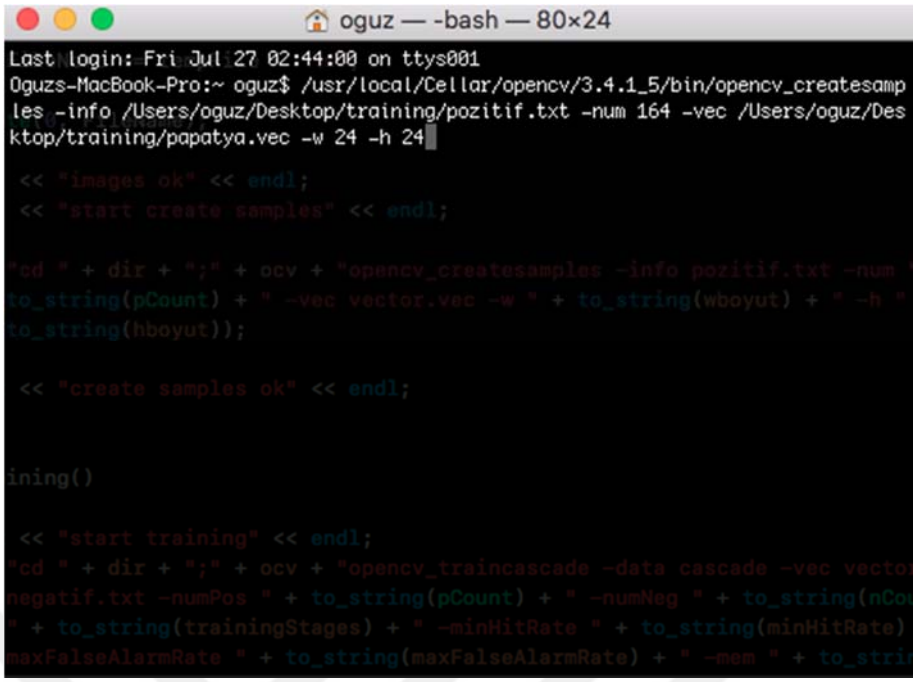
Eğitim verilerinin hazırlanması aşamasında öncelikle pozitif resim örnekleri vektör dosyası haline getirilir. OpenCV kütüphanesinde bu işlem için createsamples adında çalıştırılabilir bir dosya bulunmaktadır.

Kullanımı için info, num ve vec olan en az üç parametre gerekmektedir. Info, sırasıyla pozitif resimlerin bulunduğu adresleri, resimlerde yer alan nesne adetlerini, resimlerin başlangıç-bitiş koordinatlarını ve resimlerin genişlik-yükseklik değerlerini tutan dosya yoludur. Num, pozitif resimlerin sayısını, vec ise oluşturulacak vektör dosyasının adını temsil etmektedir.

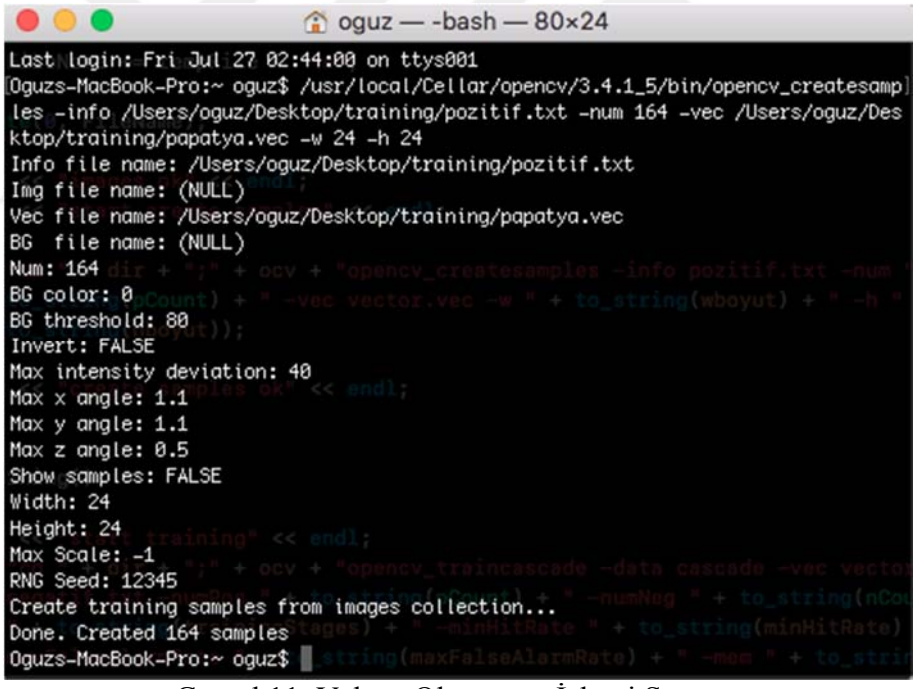


Görsel 9. Pozitif Resim Adresleri

Uygulama örneğinde parametreler “opencv_createsamples -info pozitif.txt -num 164 -vec papatya.vec -w 24 -h 24” olarak kullanılmıştır. Anlaşılır olabilmesi için örneğe eklenen -w ve -h, genişlik ve yükseklik değerlerini temsil etmekte olup varsayılanları 24 ve opsiyoneldir.




Görsel 10. Vektör Oluşturma Parametreleri

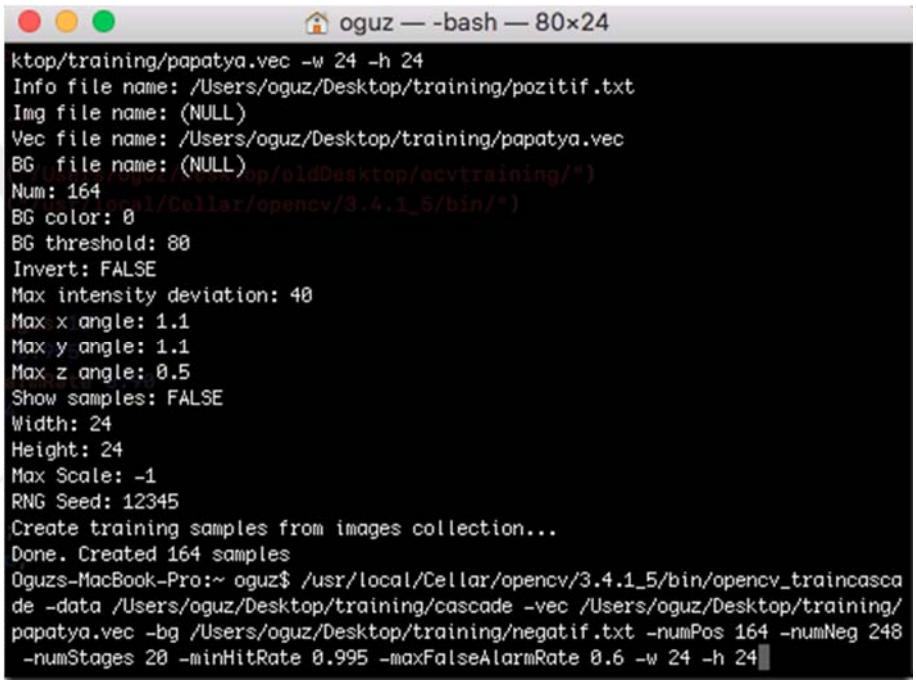



Görsel 11. Vektör Oluşturma İşlemi Sonucu


Görsel 12. Vektör Dosyası

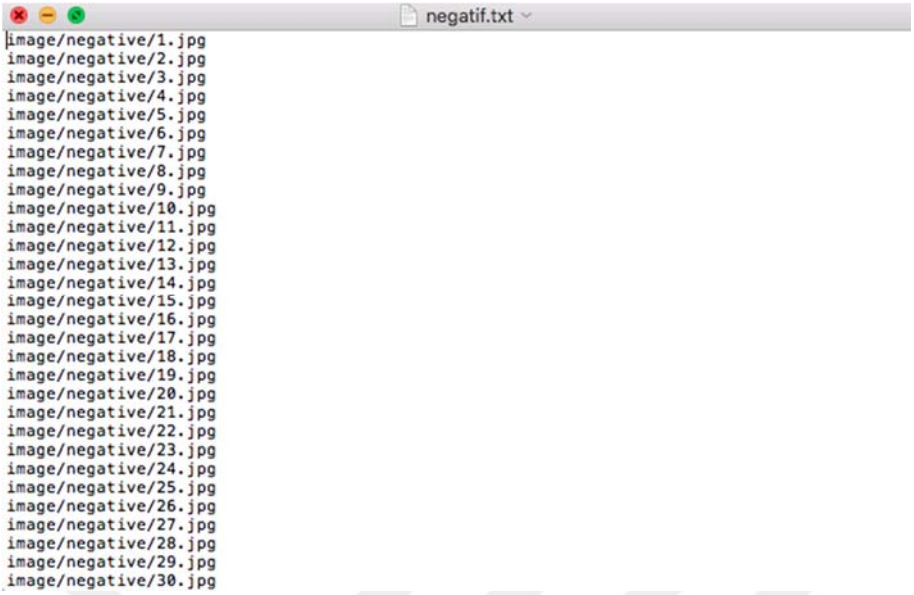
## 5.2.SINIFLANDIRICI EĞİTİMİ

Vektör dosyası oluşturulduktan sonra ikinci aşama olan eğitim aşamasına geçilebilir. OpenCV kütüphanesinde bu işlem için traincascade adında çalıştırılabilir bir dosya bulunmaktadır. Eğitim için uygulama örneğinde "opencv_traincascade -data cascade -bg negatif.txt -vec papatya.vec -numPos 164 -numNeg 248 -numStages 20 -minHitRate 0.995 -maxFalseAlarmRate 0.6 -w 24 -h 24" parametreleri kullanılmıştır.

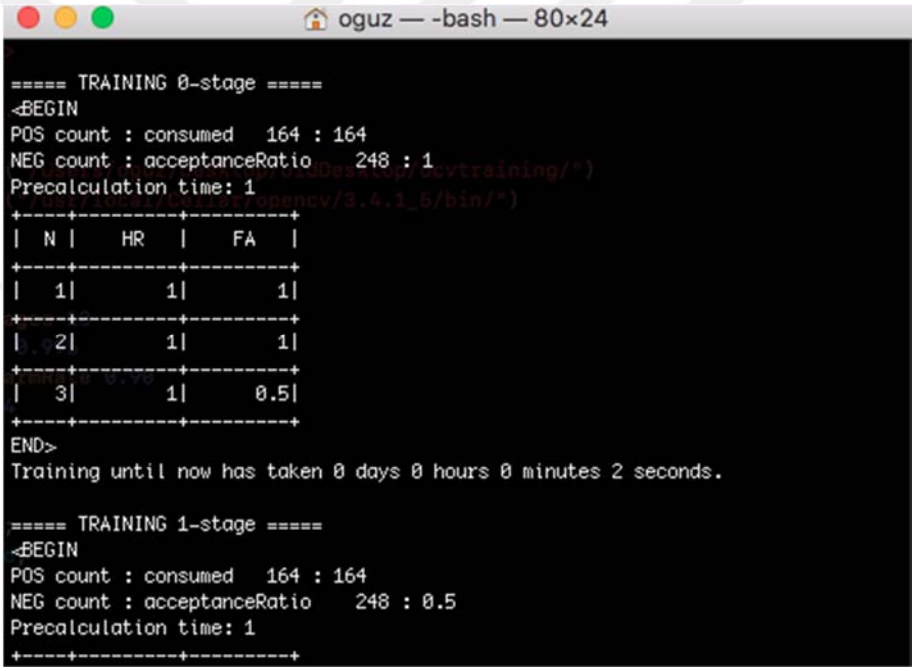



Görsel 13. Cascade Eğitimi Parametreleri

Data parametresi XML çıktı dosyalarının oluşturulması istenen dizinin adresini, vec parametresi vektör dosyasının adresini, bg parametresi negatif resimlerin listesinin tutulduğu dosyanın adresini, numPos parametresi pozitif resim sayısını, numNeg parametresi negatif resim sayısını, numStages parametresi işlem adım sayısını, minHitRate parametresi isabet oranını ve maxFalseAlarmRate parametresi kabul edilebilir en fazla hata oranını temsil etmektedir.

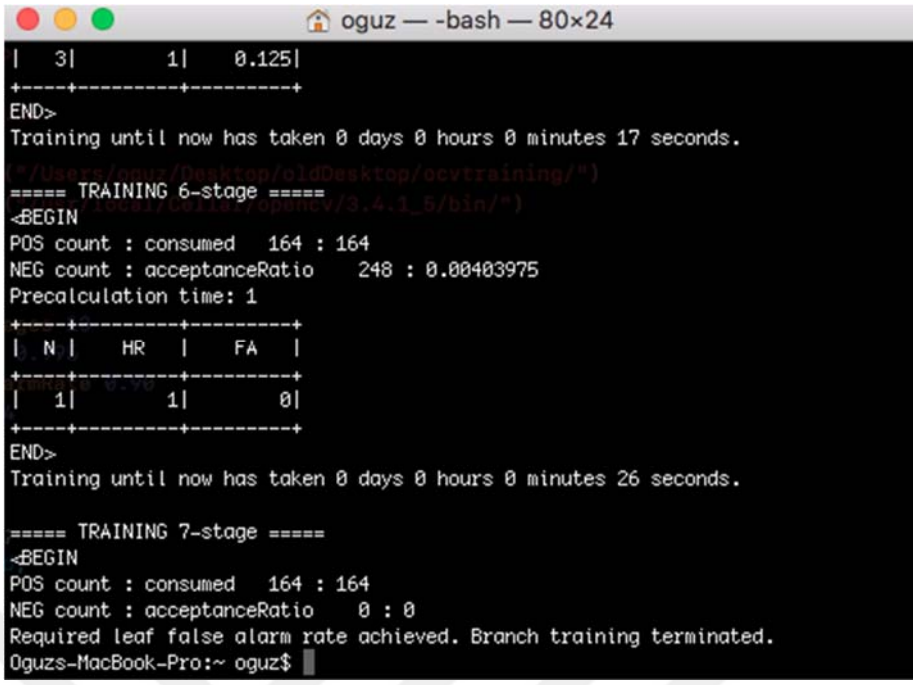



Görsel 14. Negatif Resim Adresleri



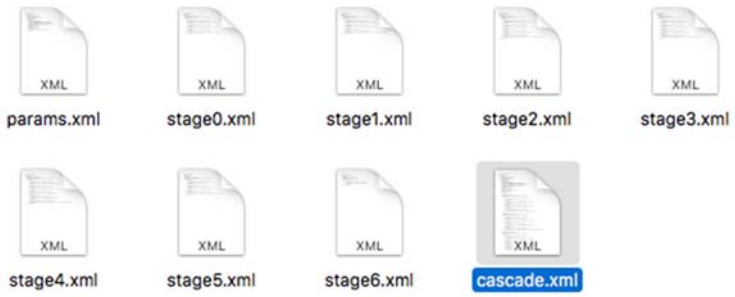

Görsel 15. Cascade Eğitimi Aşaması

Cascade eğitimi aşaması isabet oranı, hata oranı, genişlik ve yükseklik, işlem adımı sayısı gibi parametrelere ve eğitim verisi için kullanılan resim sayısına bağlı olarak birkaç saniye ile birkaç saat arasında zaman alabilmektedir. İşlenen resim, işlem adımı, geçen süre, başlangıç-bitiş ve hata-isabet oranıyla ilgili bilgiler eğitim aşaması esnasında kullanıcıya sunulmaktadır.

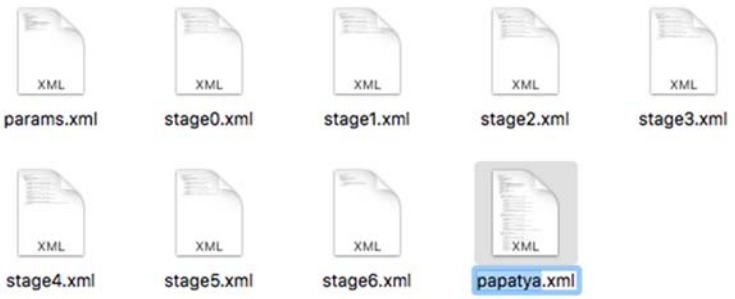



Görsel 16. Cascade Eğitimi Sonucu

NumStages olarak belirtilen işlem adım sayısı henüz tamamlanmamış olsa da parametre olarak verilen maksimum hata oranına ulaşıldığı için eğitim sonlandırılmıştır.




Görsel 17. Cascade Dosya Çıktısı




Görsel 18. Papatya.xml

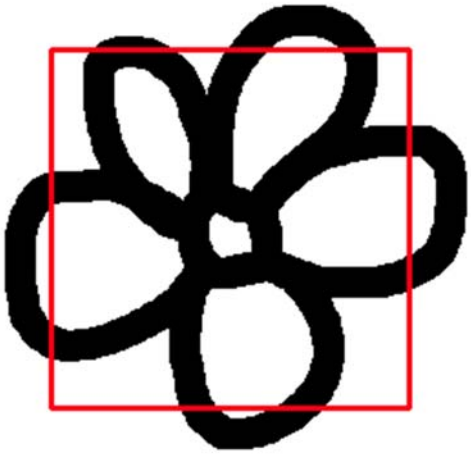
```
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;
using namespace std;

int main(int argc, char** argv )
{
    std::vector<Rect> daisy;
    String filePath = "/Users/oguz/Desktop/training/cascade/papatya.xml";

    CascadeClassifier c = *new CascadeClassifier(filePath);
    Mat image = imread("/Users/oguz/Desktop/daisy.jpg", CV_LOAD_IMAGE_COLOR);

    c.detectMultiScale(image, daisy);
    for( size_t i = 0; i < daisy.size(); i++ )
    {
        Point center( daisy[i].x + daisy[i].width/2, daisy[i].y + daisy[i].height/2 );
        rectangle(image,Point(daisy[i].x,daisy[i].y),Point(daisy[i].x + daisy[i].width, daisy[i].y +
daisy[i].height),Scalar(0,0,255), 2);
    }
    imshow( "Papatya", image );
    waitKey(0);
    return 0;
}
```

Eğitim sonrası oluşan ve sınıflandırıcı verisini tutan XML dosyası yukarıda yer alan kod yığını ile test edilebilecektir (C++).



Görsel 19. Nesne Tespit Çıktısı

Sınıflandırıcı eğitimi tamamlanmış olup XML dosyası kullanıma hazırdır.

## 5.3.UYGULAMA

Proje örneği Microsoft .NET Framework üzerinde MVC (Model-View-Controller) mimarisine sahip bir ASP (Active Server Pages) uygulaması olarak tasarlanmıştır (Microsoft Co. 2018).

Uygulamanın web arayüzü için 300x300 boyutlarında bir Canvas elementi ve onay, yenileme, temizleme görevi gören butonlar kullanılmıştır. (HTML)

```
...
<div id='myModal' class='modal'>
  <div class='modal-content'>
    <span class='close'>&times;</span>
    <div class='captcha'>
      <h4>Aşağıdaki kutuya basit bir <span id='captchatext'></span> çizin</h4>
      <div id='paint'>
        <canvas id='myCanvas' width="300" height="300" class='draw'></canvas>
      </div>
      <button id='verify' onclick='cverify()'>Doğrula</button>
      <button onclick='crecast()'>Değiştir</button>
      <button onclick='cclear()'>Temizle</button>
    </div>
  </div>
</div>
...
```

Yukarıdaki HTML kodunun CSS destekli hali aşağıdaki gibidir. (WEB)


Görsel 20. Captcha Canlı

Bu uygulama CSHTML sayfasına MVC Helper olarak eklenecektir. Bu sebeple uygulamanın, bulunduğu sayfada yer alan tüm formların gönderimini kendisine bağlaması gerekmektedir. (JS)

```
…
var modal = document.getElementById('myModal');
var forms = document.getElementsByTagName('form');
for (var i = 0; i < forms.length; i++) {
        forms[i].onsubmit = function() { currentform = this; modal.style.display = 'block'; return false };
}
…
```

Javascript alanına yazılan bir döngü ile sayfadaki tüm formların post edilmesi engellenip işlem esnasında uygulamanın görünür hale gelmesi sağlanmaktadır.

Canvas elementi üzerinde çizim yapabilmek için aşağıda yer alan kod yığını kullanılmıştır. (JS)

```
…
var canvas = document.getElementById('myCanvas');
var color = canvas.getContext('2d');
var span = document.getElementsByClassName('close')[0];
span.onclick = function() {
        modal.style.display = 'none';
}
window.onclick = function(event) {
        if (event.target == modal) {
            modal.style.display = 'none';
        }
}
var canvas = document.getElementById('myCanvas');
var ctx = canvas.getContext('2d');
var painting = document.getElementById('paint');
var paint_style = getComputedStyle(painting);
canvas.width = 300;
canvas.height = 300;
var mouse = { x: 0, y: 0 };
canvas.addEventListener('mousemove', function(e) {
        mouse.x = e.pageX - this.offsetLeft;
        mouse.y = e.pageY - this.offsetTop;
}, false);
ctx.lineWidth = 10;
ctx.lineJoin = 'round';
ctx.lineCap = 'round';
ctx.strokeStyle = '#000000';
canvas.addEventListener('mousedown', function(e) {
```

```
        ctx.beginPath();
        ctx.moveTo(mouse.x, mouse.y);
        canvas.addEventListener('mousemove', onPaint, false);
}, false);
canvas.addEventListener('mouseup', function() {
        canvas.removeEventListener('mousemove', onPaint, false);
}, false);
var onPaint = function() {
        ctx.lineTo(mouse.x, mouse.y);
        ctx.stroke();
};
…
```

Yukarıda yer alan kodlar basitçe Mouse hareketi ile imlecin konumunda bulunan piksellerin renklerini değiştirerek çizim yapıldığı izlenimini vermektedir. (JS)

Kontrol edilmesi için yapılan çizimin sunucu üzerinde bulunan OpenCV kütüphanesine gönderilmesi gerekmektedir. Sunucu üzerinden yapılacak bu ve benzeri işlemler için bir AJAX (Asynchronous Javascript and XML) köprüsü tasarlanmıştır. (JS)

```
…
function cajax(controller, action, parameter, value) {
        var returnstring;
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                returnstring = xhttp.responseText;
            }
        };
        xhttp.open('POST', '/' + controller + '/' + action, false);
        var formData = new FormData();
        formData.append(parameter, value);
        xhttp.send(formData);
        return returnstring;
}
…
```

Canvas üzerine yapılan çizimin alfa piksel değerleri bir matris dizisine dönüştürülerek köprü fonksiyonu aracılığıyla sunucuya gönderilir. (JS)

```
…
function cverify() {
        var rgb = [];
        for (var i = 0; i < canvas.height; i++) {
            for (var j = 0; j < canvas.width; j++) {
                var img = color.getImageData(j, i, 1, 1);
                img.data[3] == 255 ? rgb.push(0) : rgb.push(255);
            }
        }
        cajax('Default', 'verify', 'matrix', rgb).toLowerCase() == 'true' ? currentform.submit() : chata();
}
…
```

Piksel değerlerinin bulunduğu matris dizisini içeren String veri sunucu tarafında Byte dizisine dönüştürülür. (C#)

```
…
public class DefaultController : Controller
{
…
            public bool verify(string[] matrix)
            {
          byte[] bytes = matrix[0].Split(',').Select(x => byte.Parse(x,
System.Globalization.NumberStyles.Integer)).ToArray();
          return MatrixVerify(bytes, cascadeXml);
      }
}
```

Byte dizisi Opencv dinamik kütüphanesine aşağıdaki kodlar kullanılarak gönderilir. (C#)

```
…
[DllImport("opencv.dll", CallingConvention = CallingConvention.Cdecl)]
        public static extern bool MatrixVerify(byte[] i, string cascadeXml);
…
```

Opencv dinamik kütüphane dosyasının içeriği aşağıdaki gibidir. (C++)

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace std;
using namespace cv;

extern "C"
{
```

```
__declspec(dllexport) bool MatrixVerify(unsigned char *u, char *cascadeXml)
{
    FileStorage cascade(cascadeXml, FileStorage::READ | FileStorage::MEMORY);
    Mat image = Mat(300, 300, CV_8UC1, u);
    std::vector<Rect> Object;
    CascadeClassifier c;
    c.read(cascade.getFirstTopLevelNode());
    c.detectMultiScale(image, Object);
    return Object.size();
}
}
```

Opencv kütüphanesine gönderilen Byte dizisi 300x300 boyutunda ve 8 bit piksel derinliğinde bir OpenCV Mat nesnesine dönüştürülür. (C++)

Çizime ait görsel verisini barındıran Mat objesinde nesne tespiti için CascadeClassifier sınıfı kullanılır. CascadeClassifier::read metodu ile sınıflandırma eğitimi aşamasında oluşturulan XML verisi sınıflandırıcıya tanıtılır. (C++)

CascadeClassifier::detectMultiScale metodu, ilk parametre olarak gönderilen görselin içerisinde sınıflandırıcı tarafından tespit edilen nesnelerin koordinatlarını ikinci parametre olarak gönderilen listeye ekler. (C++)

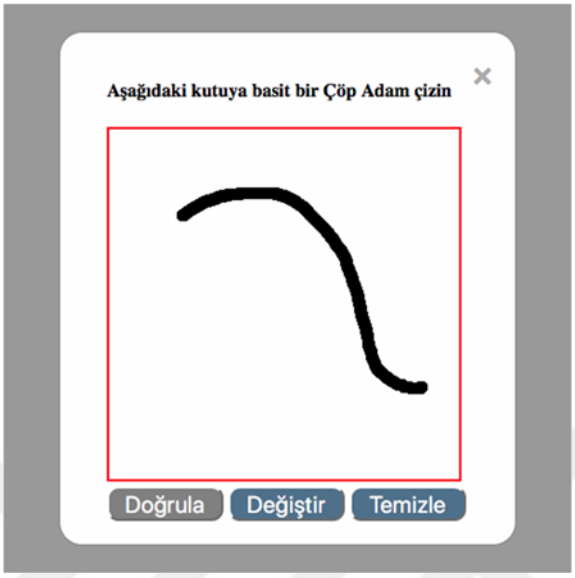
Koordinatların tutulduğu listenin boyutu kütüphanenin dönüş değerini verir. Görsel içerisinde nesne bulunamazsa bu liste 0 yani FALSE değeri döndürecektir. Bir ya da daha fazla nesne bulması halinde de pozitif bir sayı yani TRUE değerini döndürecektir. (C++)

```
...
cajax('Default', 'verify', 'matrix', rgb).toLowerCase() == 'true' ? currentform.submit() : chata();
...
```

Opencv Dinamik Kütüphanesinin dönüş değeri AJAX köprüsü tarafından istemciye aktarılır. Dönüş değerinin FALSE olması durumunda aşağıdaki hata fonksiyonu çağrılır. (JS)

```
...
```

```
function chata() {
	document.getElementById('verify').disabled = true;
	document.getElementById('myCanvas').style.border = '2px solid red';
}
```




Görsel 21. Captcha Hata

Dönüş değerinin TRUE olması halinde ise seçili form post edilir. (WEB)

## 6.BENZETİM

Çizim Tabanlı Captcha sisteminin yeterliliği web ortamı simüle edilerek test edilmiştir. Test aşamasında Captcha uygulamasından yüz, papatya ve a harfi olmak üzere üç örneğe ait muhtemel çizimleri doğru olarak sınıflandırması istenmiştir.

Sınıflandırıcı eğitimlerinde papatya için web ortamından elde edilen 164 adet pozitif ve 248 adet negatif, a harfi için ise web ortamında bulunan tek bir görselin farklı manipülasyonlarıyla oluşturulan 1000 adet pozitif ve mevcut 248 adet negatif resim kullanılmıştır. Yüz tespiti için halihazırda OpenCV kütüphanesinde yer alan yüz tanıma sınıflandırıcısı kullanılmıştır (OpenCV Foundation 2013).

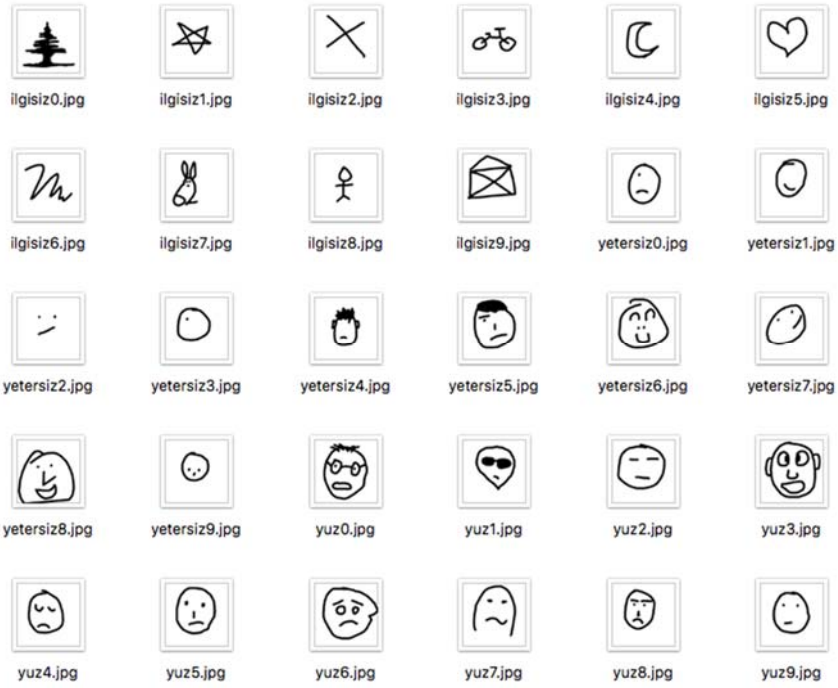
### 6.1.YÜZ ÇİZİMİ

Yüz çizimini tanıması istenen Captcha uygulamasının yeterlilik testi için 300x300 boyutlarındaki alana 10px kalınlığında 10 adet doğru olması beklenen 10 adet yetersiz ve 10 adet de ilgisiz olduğu düşünülen toplamda 30 adet çizim yapılmıştır.

Görsel 22. Doğru Yüz Çizim Örneği

Görsel 23. Yetersiz Yüz Çizim Örneği

Görsel 24. İlgisiz Yüz Çizim Örneği

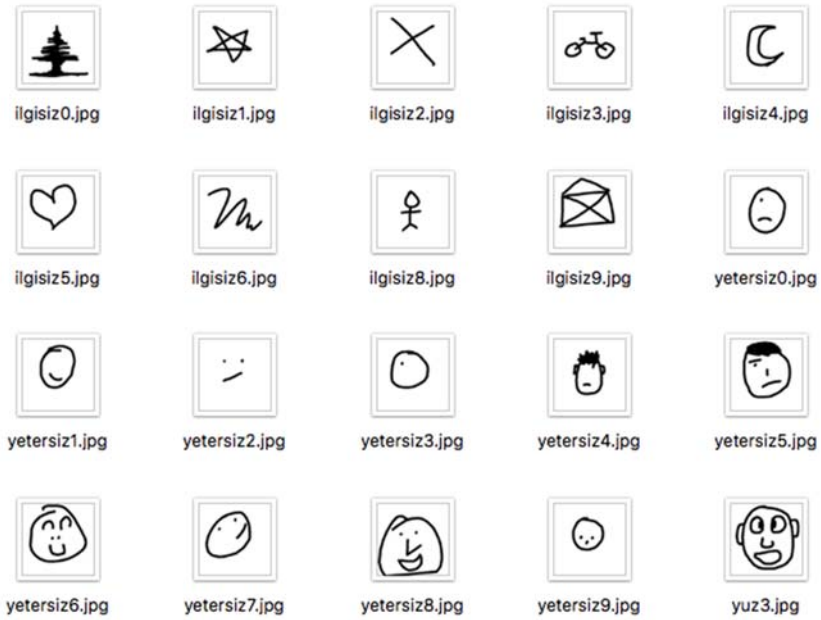



Görsel 25. Yüz Çizimi Test Dokümanları

Çizim dokümanları “yuz”, “yetersiz” ve “ilgisiz” olarak isimlendirilmiş, sonrasında da nesne sınıflandırıcısıyla işleme alınmıştır.


Görsel 26. Yüz Geçerli Çizimler

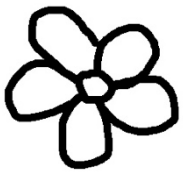
Test sonucunda geçerli kabul edilen çizimlerin 9 adedi doğru olması beklenenler arasından 1 adedi de ilgisiz görünen arasından çıkmıştır. Ayrıca doğru olması beklenen 1 adet çizim testi geçememiştir.




Görsel 27. Yüz Geçersiz Çizimler

## 6.2.PAPATYA ÇİZİMİ

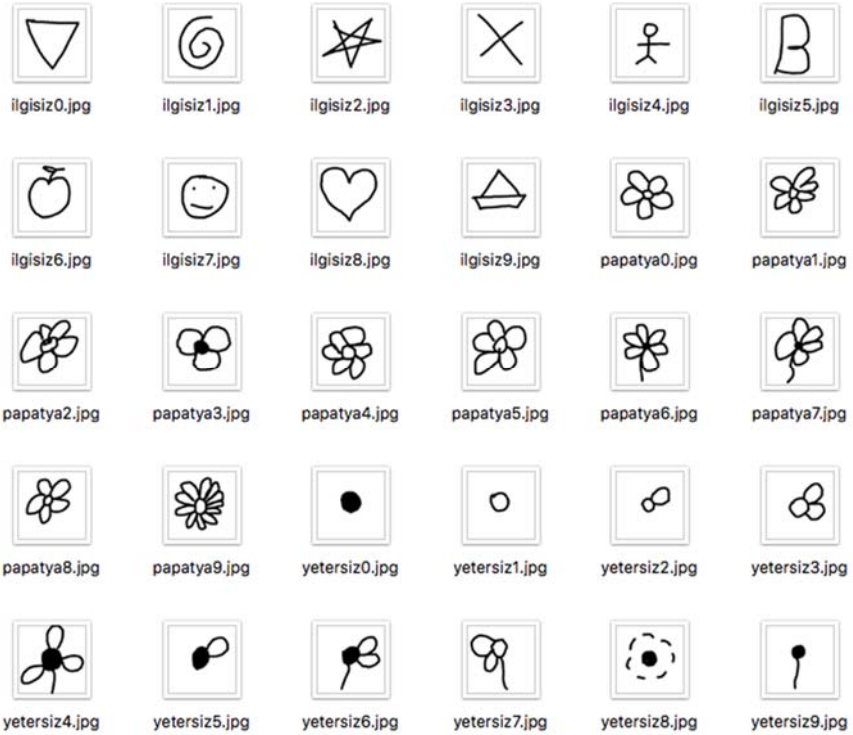
Papatya çizimini tanıması istenen Captcha uygulamasının yeterlilik testi için 300x300 boyutlarındaki alana 10px kalınlığında 10 adet doğru olması beklenen 10 adet yetersiz ve 10 adet de ilgisiz olduğu düşünülen toplamda 30 adet çizim yapılmıştır.



Görsel 28. Doğru Papatya Çizim Örneği

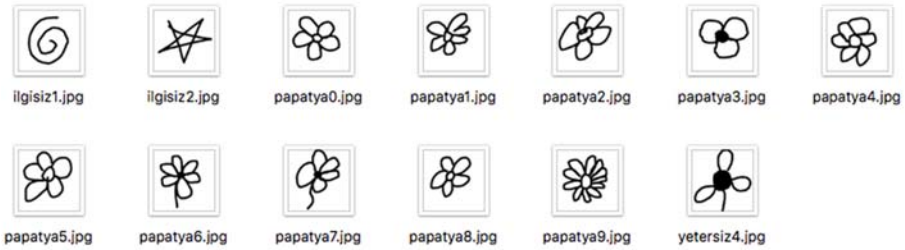
Görsel 29. Yetersiz Papatya Çizim Örneği

Görsel 30. İlgisiz Papatya Çizim Örneği



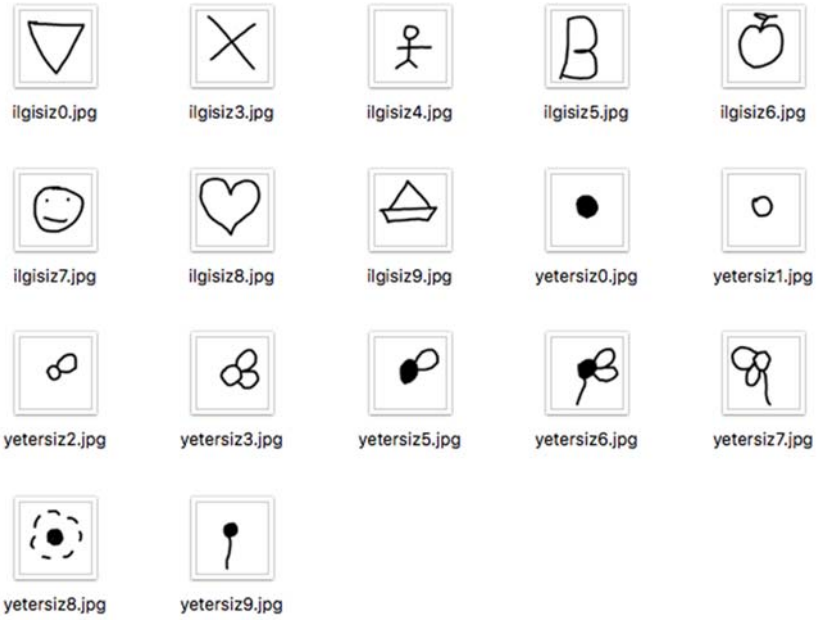

Görsel 31. Papatya Çizimi Test Dokümanları

Çizim dokümanları “papatya”, “yetersiz” ve “ilgisiz” olarak isimlendirilmiş, sonrasında da nesne sınıflandırıcısıyla işleme alınmıştır.




Görsel 32. Papatya Geçerli Çizimler

Test sonucunda doğru olması beklenen çizimlerin tamamı testi geçmiştir. Ayrıca yetersiz olduğu düşünülen 1 adet ve ilgisiz olduğu düşünülen 2 adet çizim de test sonucunda geçerli kabul edilmiştir.



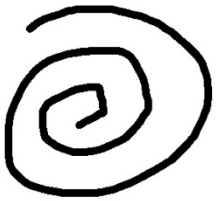

Görsel 33. Papatya Geçersiz Çizimler

### 6.3.A ÇİZİMİ

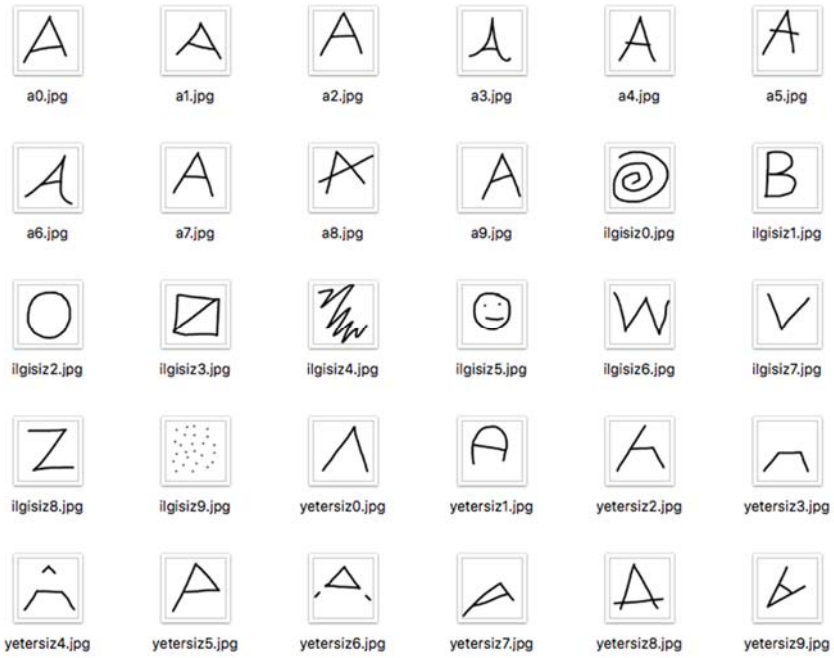
A harfi çizimini tanıması istenen Captcha uygulamasının yeterlilik testi için 300x300 boyutlarındaki alana 10px kalınlığında 10 adet doğru olması beklenen 10 adet yetersiz ve 10 adet de ilgisiz olduğu düşünülen toplamda 30 adet çizim yapılmıştır.

Görsel 34. Doğru A Çizimi Örneği

Görsel 35. Yetersiz A Çizimi Örneği

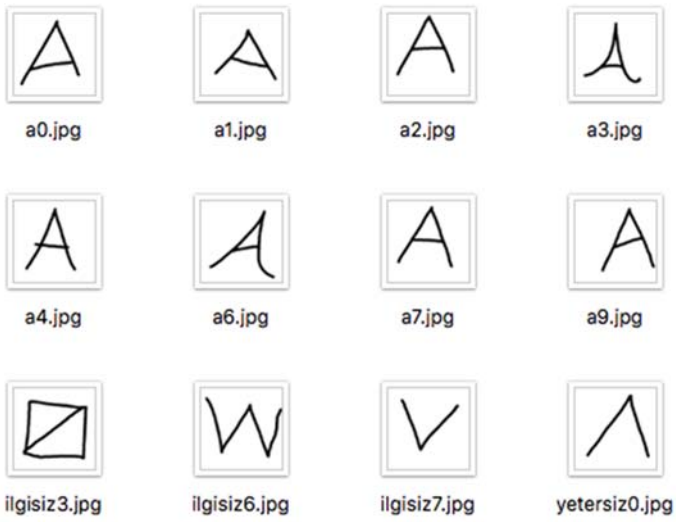


Görsel 36. İlgisiz A Çizimi Örneği



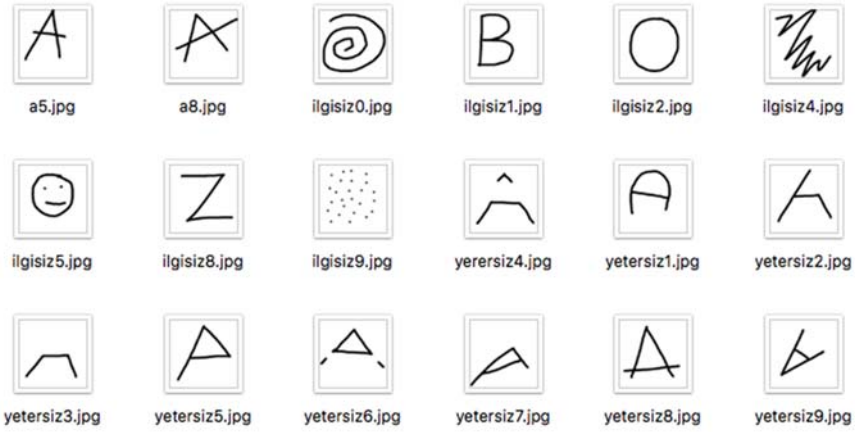

Görsel 37. A Çizimi Test Dokümanları

Çizim dokümanları “a”, “yetersiz” ve “ilgisiz” olarak isimlendirilmiş, sonrasında da nesne sınıflandırıcısıyla işleme alınmıştır.




Görsel 38. A Geçerli Çizimler

Test sonucunda doğru olması beklenen çizimlerin 2 adedi testi geçememiştir. Ayrıca ilgisiz olduğu düşünülen 3 adet ve yetersiz görünen 1 adet çizim geçerli kabul edilmiştir.




Görsel 39. A Geçersiz Çizimler

## 6.4.ANALİZ

İlk test olan yüz çizimi sonucunda yuz3 adındaki doğru olduğu düşünülen çizim geçersiz kabul edilmiştir. Eğitim aşamasında kullanılan görseller gerçek kişilere ait olduğu için fazla karikatürize edilmiş bir çizimin testi geçememiş olması olağan görünmektedir.

İkinci test olan papatya çiziminde testi geçmesi beklenen tüm çizimler istenilen sonucu vermiştir. Ancak bunların dışında 1 adet yetersiz ve 2 adet ilgisiz çizimin de testi geçmiş olması dikkat çekmektedir. Sınıflandırma için kullanılan algoritmanın eğitimi için yeterli ve nitelikli örnek kullanılamamış olması ihtimali, eksik ve yetersiz çizimlerin testi geçmesinde temel sebep olarak düşünülmektedir.

Üçüncü test olan a harfi çiziminde doğru olduğu düşünülen 2 adet çizim testi geçememiştir. Bu durumla birlikte testi geçememesi beklenen, ilgisiz ve yetersiz olduğu düşünülen toplamda 4 adet çizimin testi geçmesine temel sebep olarak, ikinci testte olduğu gibi sınıflandırma için kullanılan algoritmanın eğitiminde yeterli ve nitelikli örnek kullanılamamış olması gösterilebilir.

| **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** |
|---|---|---|---|---|---|---|---|
| Yüz | 30 | 10 | 20 | 9 | 1 | 1 | 19 |
| Papatya | 30 | 13 | 17 | 10 | 0 | 3 | 17 |
| A | 30 | 12 | 18 | 8 | 2 | 4 | 16 |

A: Captcha
B: Toplam çizim sayısı
C: Pozitif (Geçerli) çizim sayısı
D: Negatif (Geçersiz) çizim sayısı
E: Doğru pozitif (True positive - TP)
F: Hatalı negatif (False negative - FN)
G: Hatalı pozitif (False positive - FP)
H: Doğru negatif (True negative - TN)

Captcha uygulamasının temel amacı insan makine ayrımını yüksek bir doğrulukta yapabilmek olsa da kabul toleransı düşürülen testin gerçek kişi için masraflı olması kullanılabilirliği azaltmaktadır. Toleransın yükseltilmesi sonucu kullanılabilirliğin artışı beraberinde güvenlik zafiyeti de getirebileceği için kullanılabilirlik ve güvenlik arasında optimum bir oran oturtulmalıdır.

| A | I | J | K | L | M |
|---|---|---|---|---|---|
| Yüz | 0,9 | 0,9 | 0,9 | 0,93 | 0,05 |
| Papatya | 0,77 | 1 | 0,87 | 0,9 | 0,15 |
| A | 0,66 | 0,8 | 0,72 | 0,8 | 0,2 |

I: Hassasiyet (Precision) - E / (E + G)
J: Duyarlılık (Recall) - E / (E + F)
K: F-ölçümü (F-measure) -√(I . J)
L: Doğruluk (Accuracy) - (E + H) / B
M: Hatalı pozitif oranı (False positive rate) - G / (G + H)

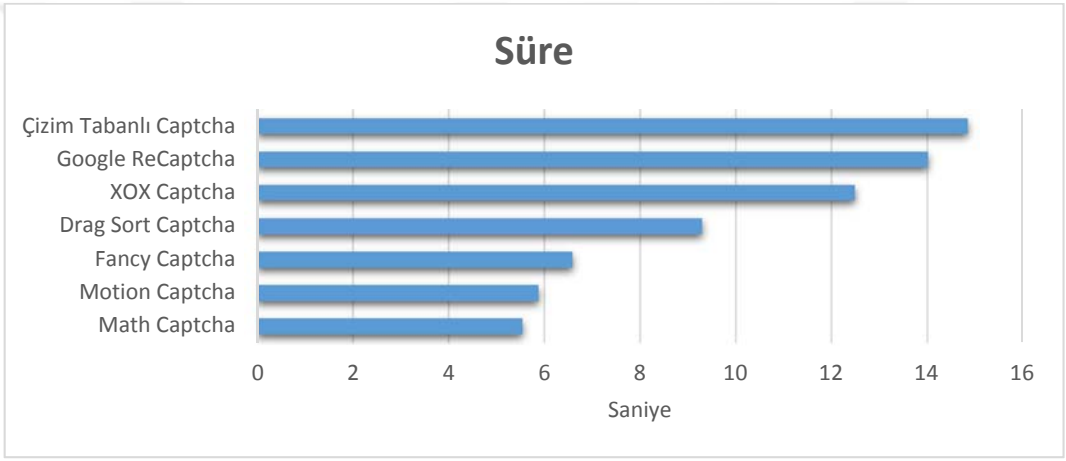
Captcha uygulamasında güvenlik kriterlerinin karşılanması kullanılabilirlikten daha önemlidir. Sonuçlar bu açıdan incelendiğinde isabet oranı ile birlikte, doğru olması beklenen ancak doğru kabul edilmeyen çizimler yerine yetersiz veya hatalı olduğu düşünülen ancak testi geçen çizimlere odaklanılması gerekliliği ortaya çıkmaktadır. Bu sebeple Papatya testinin J sütununda %100 olarak gösterilen, doğru olması beklenen çizimlerin kabul edilme oranından daha çok M sütununda %15 olarak gösterilen yetersiz veya ilgisiz çizimlerin kabul edilme oranı dikkate alınmalıdır. Neticede gerçek kişi için birkaç defa çizim yapmak kullanılabilirliği azaltsa da uygulama güvenliği kullanıcı deneyiminden daha önce gelmektedir.

Test sonuçları bir bütün olarak analiz edildiğinde beklentilerin büyük çoğunluğunun karşılanabildiği görülmektedir. Belirtilen başarı oranlarının sınıflandırıcı eğitimi ile tam korelasyonlu olduğu unutulmamalıdır. Bu durum Çizim Tabanlı Captcha sisteminde yeterli ve nitelikli eğitim ile teorik olarak %100 başarı sağlanabileceğini göstermektedir.

## 7.KULLANICI DENEYİMİ

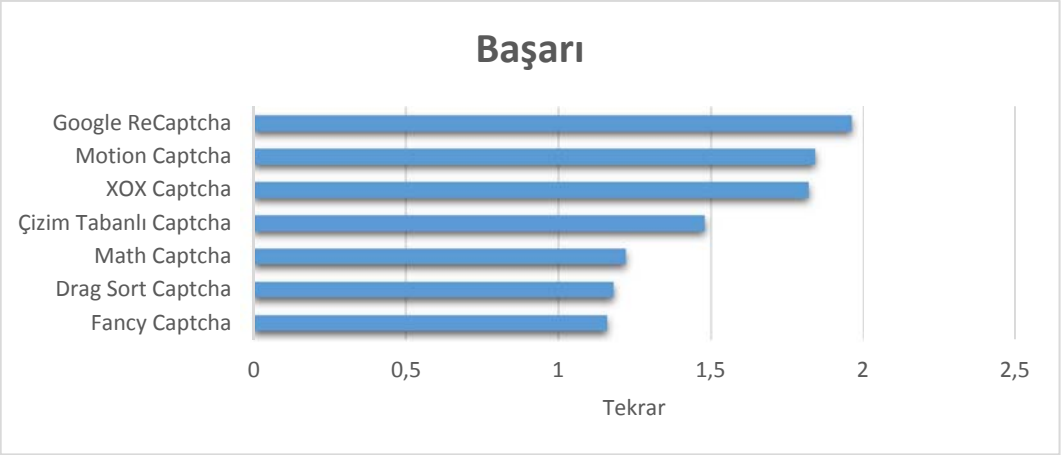
Çalışmaya konu olan Captcha mimarisinin geçerliliğinin test edilebilmesi için 17-40 yaş aralığındaki 50 gerçek kullanıcının katılımıyla bir araştırma yapılmıştır. Bu kullanıcı deneyimi araştırmasında katılımcılar Çizim Tabanlı Captcha ve buna alternatif olan diğer Captcha sistemlerini deneyimlemiştir.

Araştırma esnasında katılımcıların Captcha uygulamaları üzerinde harcadıkları zaman ve başarılı olabilmek için yaptıkları tekrar sayısı kayıt altına alınmıştır.




Şekil 1. Süre

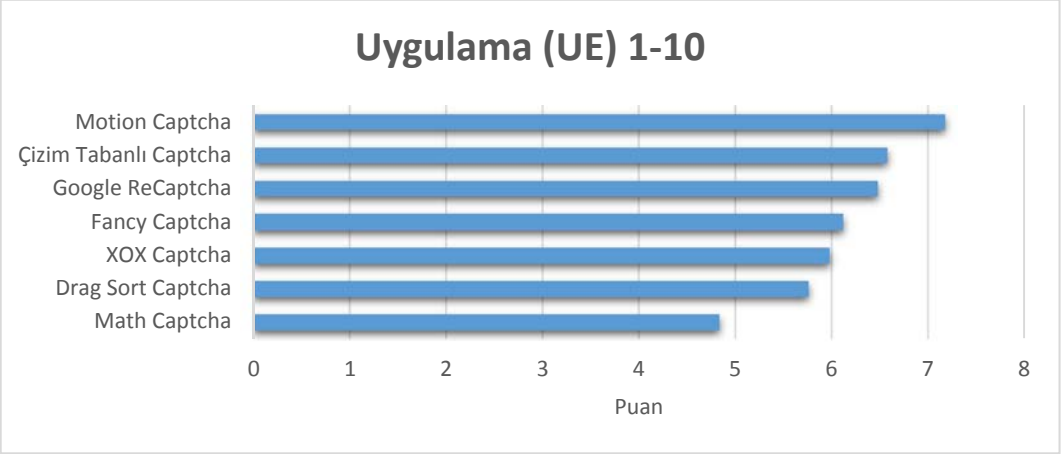
Araştırma sonuçlarının ortalama süre açısından değerlendirilmesiyle en çok zaman harcanan Captcha uygulaması Çizim Tabanlı Captcha olarak belirlenmiştir. Katılımcılar en az zamanı ise Math Captcha uygulamasında harcamıştır.




Şekil 2. Başarı

Karşılaştırma sonuçlarının ortalamalarına göre katılımcılar başarılı olabilmek için en fazla tekrarı Google ReCaptcha uygulamasında, en az tekrarı ise Fancy Captcha uygulamasında yapmıştır.

Kullanıcı deneyiminin beğeni açısından belirlenebilmesi için katılımcılardan en düşük 1 en yüksek 10 olmak üzere test ettikleri Captcha uygulamalarını puanlamaları istenmiştir.

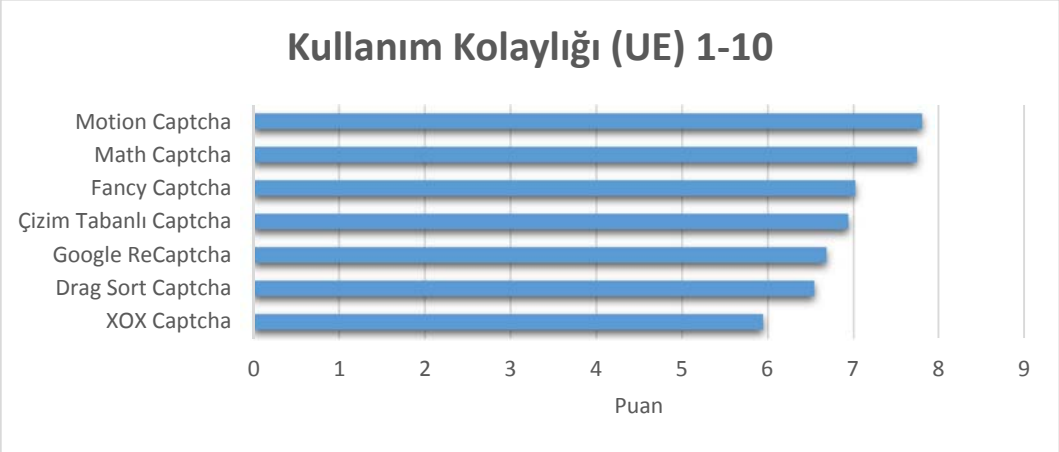



Şekil 3. Uygulama Puanı

Katılımcıların puan ortalamaları doğrultusunda en fazla beğenilen Captcha uygulaması Motion Captcha, en az beğenilen Captcha uygulaması ise Math Captcha olmuştur.

Katılımcılardan deneyimledikleri Captcha uygulamalarının teknik yeterlilikleriyle ilgili görüşlerini kullanılabilirlik ve güvenilirlik açısından belirtmeleri istenmiştir.

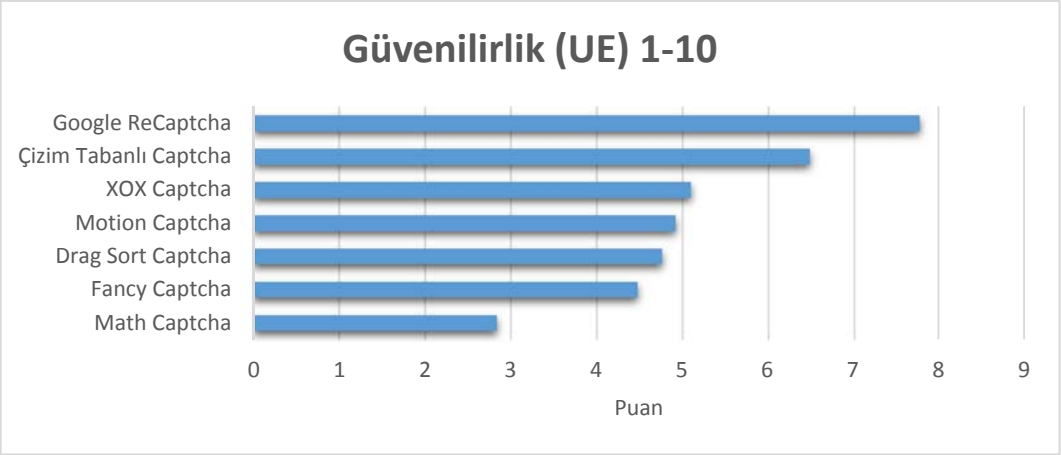
Kullanılabilirliğin belirlenebilmesi için katılımcılardan kullanım kolaylığına göre deneyimledikleri Captcha uygulamalarına en düşük 1 en yüksek 10 olmak üzere puan vermeleri istenmiştir.




Şekil 4. Kullanılabilirlik Puanı

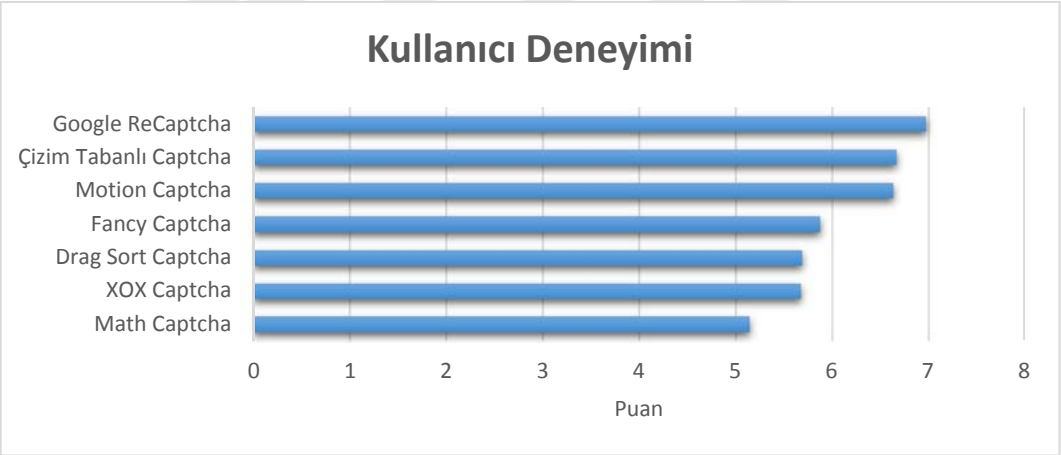
Katılımcı puanlarının ortalamaları doğrultusunda kullanılabilirliği en yüksek uygulama Motion Captcha, en düşük uygulama ise XOX Captcha olarak belirlenmiştir.

Uygulama güvenilirliğinin belirlenebilmesi için katılımcılardan deneyimledikleri Captcha sistemlerinin bot saldırılarına karşı sağlayabileceklerini düşündüğü güvenlik düzeyini en düşük 1 en yüksek 10 olmak üzere puanlayarak belirtmeleri istenmiştir.




Şekil 5. Güvenilirlik Puanı

Uygulama güvenilirliği açısından katılımcıların puan ortalamalarına göre ilk sırayı Google ReCaptcha almış olup katılımcılara göre en az güvenilirlik sağlayan uygulama Math Captcha olarak belirlenmiştir.




Şekil 6. Kullanıcı Deneyimi

Kullanıcı deneyimi araştırmasının sonuçları değerlendirildiğinde en çok zaman harcanan Captcha uygulaması olan Çizim Tabanlı Captcha, testi geçebilmek için gerçekleştirilen tekrar sayısının genel ortalamanın altında olması ve yenilikçi yaklaşımı sonucunda katılımcılardan aldığı beğeni, kullanım kolaylığı ve uygulama güvenilirliği puan ortalamalarıyla rakiplerini geride bırakarak Google ReCaptcha uygulamasından sonra ikinci sıraya yerleşmiştir.

## 8.SONUÇ

Çizim Tabanlı Captcha mimarisi yapılan çizim üzerinden nesne sınıflandırma esasına dayanmaktadır. Sistemin hata oranı tamamı ile sınıflandırıcıya verilen eğitimin kalitesine bağlıdır. Nitelikli ve iyi bir eğitim ile istenen nesnenin olası tüm çizimlerinin kesin doğrulukta tespit edilmesi teorik olarak mümkündür.

Sistemin çizime yönelik diğer Captcha uygulamalarıyla arasındaki en büyük fark istenen nesnenin çizimi için net bir kriterin olmamasıdır. Farklı şekillerde, yapılarda veya boyutlarda yapılan bir çizim nesne sınıflandırıcısı tarafından tespit edilebilmektedir. Alternatif yapılarda bu yalnızca referans verilen bir çizimin yüksek doğrulukta taklit edilmesi şeklinde gerçekleştirilmektedir. Bu durum Çizim Tabanlı Captcha sistemini benzeri Captcha uygulamalarının önüne geçirmektedir.

İşlenmesi amacıyla sınıflandırıcıya gönderilecek çizim mimariye göre değişebilmekle birlikte genellikle matrise ya da resim formatına dönüştürülmektedir. Dolayısıyla sınıflandırıcıya halihazırda istenen çizim yapılmış bir resmin veya çizime ait matris değerlerinin gönderilebilmesi halinde de geçerli kabul edilmesi mümkün olacaktır. Güvenlik kriterleri açısından incelendiğinde pratikte benzeri enjeksiyon türevi ataklardan korunabilmek amcacıyla Javascript veya SQL sistemleri için kullanılan önlemlere benzer tedbirler almak gerekebilir. Öncelikle sınıflandırıcının istemci üzerine kurulması Captcha sistemini girişime açık hale getirecektir. Bu sebeple uygulama örneğinde sınıflandırıcı işlemleri sunucu tarafında yapılmıştır. Ancak buna rağmen sunucuya gönderilen değerlerin yapılan çizime ait olup olmadığının tespiti kesin doğrulukta sağlanamayabilir. Alternatif bir yaklaşımla çizimin geri bildirimli olarak sunucu üzerinde gerçekleştirilmesi enjeksiyon bazlı saldırılara karşı güvenliği artıracaktır. Saldırılardan korunmak için bu ve benzeri çözüm önerileri uygulanacak mimari esas alınarak çeşitlendirilebilse de hiçbir sistemin geçilemez olmadığı gerçeğinin unutulmaması gerekmektedir.

Turing testi neticede insan ile makine arasındaki davranış farklılıkları üzerinden çıkarım yapmaya çalışan bir algoritmalar bütünüdür. Bununla birlikte

gelişen teknoloji karşısında insan ile makine davranışı arasındaki fark giderek azalmaktadır. Bu sebeple Turing temelli diğer Captcha uygulamalarında olduğu gibi Çizim Tabanlı Captcha uygulaması da teoride geçilemez değildir. Ancak pratikte günümüz yapay zekâsı ile tasarlanan bot uygulamaları ele alınınca diğer Captcha sistemlerine göre daha fazla güvenilirlik sağladığı açıktır.

Sonuç olarak Çizim Tabanlı Captcha sistemi halihazırda kullanılmakta olan Captcha sistemlerine göre yenilikçi bir bakış açısı sunmaktadır. Kullanılan mimari sebebiyle yapının farklı platformlara kolaylıkla port edilmesi mümkündür. Ayrıca geliştiriciye mimari tasarıma müdahale edebilme, yeni sınıflandırıcılar ekleyebilme ve ekstra parametreler ile özelleştirebilme imkânı sunulmaktadır.

Kullanıcı deneyimi araştırma sonuçları da dikkate alındığında belirtilen tüm bu özellikler Çizim Tabanlı Captcha sistemini mevcut Captcha uygulamalarına güçlü bir alternatif yapmaktadır.

## KAYNAKÇA


Amazon Inc. 2018. *Amazon Rekognition.* Erişildi: August 13, 2018. https://aws.amazon.com/rekognition/.

AndrewP. 2017. *Boonex.* Erişildi: August 15, 2018. https://www.boonex.com/m/tic-tac-toe-captcha.

Cody. 2009. *jQuery Fancy Draggable Captcha.* 8 September. Erişildi: November 21, 2018. https://tympanus.net/codrops/2009/09/08/jquery-fancy-draggable-captcha/.

Crowcroft, William J. 2011. *Github.* Erişildi: August 15, 2018. https://github.com/wjcrowcroft/MotionCAPTCHA.

Dana Harry Ballard, Christopher M. Brown. 1982. *Computer Vision.* New Jersey: Prentice Hall.

Donalek, Ciro. 2011. «Supervised and Unsupervised Learning.» (California Institute of Technology). http://www.astro.caltech.edu/~george/aybi199/Donalek_Classif.pdf.

Google Inc. 2018. *AI & Machine Learning Products.* Erişildi: August 13, 2018. https://cloud.google.com/vision/.

—. 2018. *reCAPTCHA.* Erişildi: August 13, 2018. https://www.google.com/recaptcha/intro/v3beta.html.

Leskovec, Jure, Anand Rajaraman, ve Jeffrey David Ullman. 2014. *Mining of Massive Datasets.* Cilt II. Cambridge University Press.

Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, Manuel Blum. 2008. «reCAPTCHA: Human-Based Character Recognition via Web Security Measures.» *Sciencemag* 321: 1465-1467.

Luis von Ahn, Manuel Blum, Nicholas J. Hopper, John Langford. 2003. «CAPTCHA: Using Hard AI Problems For Security.» Pittsburgh - Yorktown Heights, USA: Computer Science Dept.Carnegie Mellon University, IBM T.J. Watson Research Center.

Microsoft Co. 2018. *ASP.NET.* Erişildi: August 17, 2018. https://www.asp.net/mvc.

—. 2018. *Microsoft Azure.* Erişildi: August 13, 2018. https://azure.microsoft.com/en-ca/services/cognitive-services/computer-vision/.

N. Suguna, Dr. K. Thanushkodi. 2010. «An Improved k-Nearest Neighbor Classification Using Genetic Algorithm.» *International Journal of Computer Science Issues* (Akshaya College of Engineering and Technology) 7 (4): 18-21.

OpenCV Foundation. 2013. *Github.* 19 December. Erişildi: August 15, 2018. https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml.

—. 2018. *License.* Erişildi: August 15, 2018. https://opencv.org/license.html.

—. 2018. *OpenCV.* Erişildi: August 13, 2018. https://opencv.org.

Parshall, Karen Hunger. 1988. *The Art of Algebra from Al-Khwarizmi to Viète: A Study in the Natural Selection of Ideas.* Champaign, Illinois: Science and Engineering Library, University of Virginia.

Piskin, Emre. 2017. *Github.* Erişildi: August 15, 2018. https://github.com/pemre/jquery-captcha-basic.

Raval, Kalyani M. 2012. «Data Mining Techniques.» *International Journal of Advanced Research in Computer Science and Software Engineering* (Maharaja Krishnakumarsinhji Bhavnagar University) 2 (10): 439-442.

SimpleCV. 2018. *SimpleCV.* Erişildi: August 13, 2018. http://simplecv.org.

SXSW, röportaj yapan: Dom Galeon ve Christianna Reedy. 2017. *Kurzweil Claims That the Singularity Will Happen by 2045* (5 October).

TheGamesDB. 2013. *Github.* Erişildi: August 15, 2018. https://github.com/TheGamesDB/TheGamesDB/tree/master/js/ajax-fancy-captcha-php.

Turing, Alan Mathison. 1950. *Computing Machinery And Intelligence.* Cilt 59. 236 cilt. Oxford University Press on behalf of the Mind Association.

Watson, Andrew B., dü. 1993. *Digital Images and Human Vision.* London, England: The MIT Press.

## Ek-1: Çizim Tabanlı Captcha Uygulaması

*CaptchaClass.cs*

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;
using System.Data.SqlClient;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Web.Mvc;
using System.Xml.Linq;

namespace CaptchaLibrary
{
    public static class CaptchaClass
    {
        public enum captchaSource
        {
            local,
            database
        }

        public static dynamic properties
        {
            get
            {
                return new { type = (captchaSource)System.Web.HttpContext.Current.Session["source"], select =
(dynamic)System.Web.HttpContext.Current.Session["select"], defaults =
(dynamic)System.Web.HttpContext.Current.Session["defaults"], parameters =
(dynamic)System.Web.HttpContext.Current.Session["parameters"] };
            }

            set
            {
                System.Web.HttpContext.Current.Session["defaults"] = value.defaults;
                System.Web.HttpContext.Current.Session["source"] = value.source;
                System.Web.HttpContext.Current.Session["select"] = value.select;
                System.Web.HttpContext.Current.Session["parameters"] = value.parameters;
                System.Web.HttpContext.Current.Session["error"] = !new CaptchaController().captchaInit(value.source);
            }
        }

        public static CaptchaModel current
        {
```

```
        get
        {
            return (CaptchaModel)System.Web.HttpContext.Current.Session["current"];
        }

        set
        {
            System.Web.HttpContext.Current.Session["current"] = value;
        }
    }

    /// <summary>
    /// local veya veritabani
    /// </summary>
    /// <param name="source">kaynak secimi</param>
    /// <param name="select">manuel captcha secimi</param>
    /// <param name="defaults">varsayilan parametreler new { height = 300, width = 300, minsize = 1, maxsize = 300,
flags = 0, scalefactor = 1.1, neighbors = 3 }</param>
    /// <param name="parameters">manuel parametreler (override) new { height = int, width = int, minsize = int,
maxsize = int, flags = int, scalefactor = double, neighbors = int }</param>
    /// <returns></returns>
    public static MvcHtmlString Ekle(captchaSource source, dynamic select = null, dynamic defaults = null, dynamic
parameters = null)
    {
        properties = new { source = source, select = select, defaults = defaults ?? new { height = 300, width = 300,
minsize = 1, maxsize = 300, flags = 0, scalefactor = 1.1, neighbors = 3 }, parameters = parameters };
        if ((bool)System.Web.HttpContext.Current.Session["error"])
        return MvcHtmlString.Create("<script>(function (){alert('Captcha not found');})();</script>");

        TagBuilder style = new TagBuilder("style");
        style.InnerHtml = ".captcha {  width: 310px;  margin: auto;  }  .captcha button {  height:
30px;  width: 100px;  border-radius: 10px;  color: #FFFFFF;  font-size: 20px;
background-color: #516f8d;  }  .captcha button:hover {  background-color: #2a6099;  }
.captcha button:disabled{  background-color: #808080;  }  .draw {  border: 2px solid gray;  }
.modal {  display: none;  position: fixed;  z-index: 1;  padding-top: 100px;  left: 0;  top: 0;
width: 100%;  height: 100%;  overflow: auto;  background-color: rgb(0,0,0);  background-color:
rgba(0,0,0,0.4);  }  .modal-content {  background-color: #fefefe;  margin: auto;  padding: 20px;
border: 1px solid #888;  width: 350px;  border-radius: 20px;  }  .close {  color: #aaaaaa;  float:
right;  font-size: 28px;  font-weight: bold;  }  .close:hover,  .close:focus {  color: #000;
text-decoration: none;  cursor: pointer;  }";

        TagBuilder mymodaldiv = new TagBuilder("div");
        mymodaldiv.Attributes["id"] = "myModal";
        mymodaldiv.AddCssClass("modal");

        TagBuilder modalcontentdiv = new TagBuilder("div");
        modalcontentdiv.AddCssClass("modal-content");

        TagBuilder spanclose = new TagBuilder("span");
```

```
        spanclose.AddCssClass("close");
        spanclose.InnerHtml = "&times";

        TagBuilder captchadiv = new TagBuilder("div");
        captchadiv.AddCssClass("captcha");

        TagBuilder header = new TagBuilder("h4");
        header.InnerHtml = "Asagidaki kutuya basit bir <span id='captchatext'></span> cizin";

        TagBuilder paintdiv = new TagBuilder("div");
        paintdiv.Attributes["id"] = "paint";

        TagBuilder canvas = new TagBuilder("canvas");
        canvas.AddCssClass("draw");
        canvas.Attributes["id"] = "myCanvas";

        TagBuilder verifybutton = new TagBuilder("button");
        verifybutton.Attributes["id"] = "verify";
        verifybutton.Attributes["onclick"] = "cverify()";
        verifybutton.InnerHtml = "Dogrula";

        TagBuilder crecastbutton = new TagBuilder("button");
        crecastbutton.Attributes["onclick"] = "crecast()";
        crecastbutton.InnerHtml = "Degistir";

        TagBuilder clearbutton = new TagBuilder("button");
        clearbutton.Attributes["onclick"] = "cclear()";
        clearbutton.InnerHtml = "Temizle";

        TagBuilder script = new TagBuilder("script");
        script.InnerHtml = "var currentform; var canvas = document.getElementById('myCanvas'); var color =
canvas.getContext('2d'); var modal = document.getElementById('myModal'); var span =
document.getElementsByClassName('close')[0];  var forms = document.getElementsByTagName('form');  for (var i = 0; i
< forms.length; i++) { forms[i].onsubmit = function() { crecast(); currentform = this; modal.style.display = 'block'; return
false }; }  window.onclick = function(event) { if (event.target == modal) { modal.style.display = 'none'; } };  span.onclick =
function() { modal.style.display = 'none'; };   var canvas = document.getElementById('myCanvas'); var ctx =
canvas.getContext('2d');  var painting = document.getElementById('paint'); var paint_style =
getComputedStyle(painting); canvas.width = " + CaptchaClass.properties.defaults.width + ";canvas.height = " +
CaptchaClass.properties.defaults.height + ";  var mouse = { x: 0, y: 0 };  canvas.addEventListener('mousemove',
function(e) {         mouse.x = e.pageX - this.offsetLeft;         mouse.y = e.pageY - this.offsetTop; }, false);  ctx.lineWidth =
10; ctx.lineJoin = 'round'; ctx.lineCap = 'round'; ctx.strokeStyle = '#000000';  canvas.addEventListener('mousedown',
function(e) {         ctx.beginPath();         ctx.moveTo(mouse.x, mouse.y);         canvas.addEventListener('mousemove',
onPaint, false); }, false);  canvas.addEventListener('mouseup', function() {
canvas.removeEventListener('mousemove', onPaint, false); }, false);  var onPaint = function() {         ctx.lineTo(mouse.x,
mouse.y);         ctx.stroke(); };  function chata() {         document.getElementById('verify').disabled = true;
document.getElementById('myCanvas').style.border = '2px solid red'; }   function cclear() {
document.getElementById('verify').disabled = false;         canvas.style.border = '2px solid gray';         color.clearRect(0,
0, canvas.width, canvas.height); }  function crecast() {         cclear();
document.getElementById('captchatext').innerHTML = cajax('Captcha', 'recast'); }  function cajax(controller, action,
```

```
parameter, value) {         var returnstring;         var xhttp = new XMLHttpRequest();         xhttp.onreadystatechange =
function() {             if (this.readyState == 4 && this.status == 200) {                 returnstring = xhttp.responseText;             }
};         xhttp.open('POST', '/' + controller + '/' + action, false);         var formData = new FormData();
formData.append(parameter, value); xhttp.send(formData); return returnstring; } function cverify() { var rgb = []; for (var i
= 0; i < canvas.height; i++) { for (var j = 0; j < canvas.width; j++) { var img = color.getImageData(j, i, 1, 1); img.data[3] ==
255 ? rgb.push(0) : rgb.push(255);}} cajax('Captcha', 'verify', 'matrix', rgb).toLowerCase() == 'true' ? currentform.submit()
: chata();  } ";

        StringBuilder htmlBuilder = new StringBuilder();

        htmlBuilder.Append(style.ToString(TagRenderMode.Normal));
        htmlBuilder.Append(mymodaldiv.ToString(TagRenderMode.StartTag));
        htmlBuilder.Append(modalcontentdiv.ToString(TagRenderMode.StartTag));
        htmlBuilder.Append(spanclose.ToString(TagRenderMode.Normal));
        htmlBuilder.Append(captchadiv.ToString(TagRenderMode.StartTag));
        htmlBuilder.Append(header.ToString(TagRenderMode.Normal));
        htmlBuilder.Append(paintdiv.ToString(TagRenderMode.StartTag));
        htmlBuilder.Append(canvas.ToString(TagRenderMode.Normal));
        htmlBuilder.Append(paintdiv.ToString(TagRenderMode.EndTag));
        htmlBuilder.Append(verifybutton.ToString(TagRenderMode.Normal));
        htmlBuilder.Append(crecastbutton.ToString(TagRenderMode.Normal));
        htmlBuilder.Append(clearbutton.ToString(TagRenderMode.Normal));
        htmlBuilder.Append(captchadiv.ToString(TagRenderMode.EndTag));
        htmlBuilder.Append(modalcontentdiv.ToString(TagRenderMode.EndTag));
        htmlBuilder.Append(mymodaldiv.ToString(TagRenderMode.EndTag));
        htmlBuilder.Append(script.ToString(TagRenderMode.Normal));

        return MvcHtmlString.Create(htmlBuilder.ToString());
    }

}

[Table("Captcha")]
public class CaptchaModel
{
    [Key]
    public int id { get; set; }
    public string name { get; set; }
    public string cascade { get; set; }
    public int minsize { get; set; } = CaptchaClass.properties.defaults.minsize;
    public int maxsize { get; set; } = CaptchaClass.properties.defaults.maxsize;
    public int flags { get; set; } = CaptchaClass.properties.defaults.flags;
    public double scalefactor { get; set; } = CaptchaClass.properties.defaults.scalefactor;
    public int neighbors { get; set; } = CaptchaClass.properties.defaults.neighbors;
}

public class CaptchaContext : DbContext
{
    public CaptchaContext(string sql) : base(sql) { }
```

```
    public DbSet<CaptchaModel> CaptchaData { get; set; }
}

public class CaptchaController : Controller
{
    [DllImport("kernel32.dll", CharSet = CharSet.Unicode, SetLastError = true)]
    [return: MarshalAs(UnmanagedType.Bool)]
    static extern bool SetDllDirectory(string lpPathName);

    [DllImport("opencv.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern bool MatrixVerify(byte[] i, string cascade, int canvasheight, int canvaswidth, int minsize, int
maxsize, int flags, double scaleFactor, int minneighbors, bool test = false);


    public bool captchaInit(CaptchaClass.captchaSource source)
    {
        switch (source)
        {
            case CaptchaClass.captchaSource.local:
                string[] cascade =
Directory.GetFiles(System.Web.HttpContext.Current.Server.MapPath("~/opencv/cascade/"), "*" +
CaptchaClass.properties.select +  ".xml", SearchOption.AllDirectories);
                if (cascade.Length == 0) return false;
                string current = cascade[new Random().Next(cascade.Length)];
                CaptchaModel model = new CaptchaModel();
                model.cascade = System.IO.File.ReadAllText(current);
                model.name = Path.GetFileName(current).Split('.')[0];
                string[] parametreFiles =
Directory.GetFiles(System.Web.HttpContext.Current.Server.MapPath("~/opencv/cascade/"), "*param");

                if (parametreFiles.Length != 0)
                {
                    XDocument parametre = XDocument.Load(parametreFiles[0]);
                    XElement[] element = parametre.Root.Elements().Where(x => x.Element("name").Value + ".xml" ==
Path.GetFileName(current)).ToArray();
                    if (element.Count() != 0)
                    {

                        model.name = element.Select(x => x.Element("name").Value).First();
                        model.minsize = Convert.ToInt32(element.Select(x => x.Element("minsize").Value).First());
                        model.maxsize = Convert.ToInt32(element.Select(x => x.Element("maxsize").Value).First());
                        model.flags = Convert.ToInt32(element.Select(x => x.Element("flags").Value).First());
                        model.scalefactor = Convert.ToDouble(element.Select(x => x.Element("scalefactor").Value).First());
                        model.neighbors = Convert.ToInt32(element.Select(x => x.Element("neighbors").Value).First());
                    }
                }
                CaptchaClass.current = model;
                break;
            case CaptchaClass.captchaSource.database:
```

```
                if (System.Web.HttpContext.Current.Session["captcha"] == null)
                    System.Web.HttpContext.Current.Session["captcha"] = new CaptchaContext(@"Data
Source=.\SQLEXPRESS;Initial Catalog=CaptchaDb;Integrated
Security=True;MultipleActiveResultSets=True").CaptchaData.ToList();
                List<CaptchaModel> captcha =
((List<CaptchaModel>)System.Web.HttpContext.Current.Session["captcha"]);
                if(CaptchaClass.properties.select != null && captcha.Where(x => x.name ==
CaptchaClass.properties.select).FirstOrDefault() == null) return false;
                CaptchaClass.current = captcha.Where(x => x.name == CaptchaClass.properties.select).FirstOrDefault()
?? captcha[new Random().Next(captcha.Count)];
                break;
            }
            return true;
        }

        public string recast()
        {
            captchaInit(CaptchaClass.properties.type);
            return CaptchaClass.current.name;
        }

        public bool verify(string[] matrix)
        {
            SetDllDirectory(Server.MapPath("~/opencv/x64/Debug"));
            byte[] bytes = matrix[0].Split(',').Select(x => byte.Parse(x,
System.Globalization.NumberStyles.Integer)).ToArray();
            dynamic parametre = CaptchaClass.properties.parameters ?? CaptchaClass.current;
            return MatrixVerify(bytes, CaptchaClass.current.cascade, (CaptchaClass.properties.parameters ??
CaptchaClass.properties.defaults).height, (CaptchaClass.properties.parameters ??
CaptchaClass.properties.defaults).width, parametre.minsize, parametre.maxsize, parametre.flags,
parametre.scalefactor, parametre.neighbors);
        }
    }
}
```

*DefaultController.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace DrawCaptcha.Controllers
{
    public class DefaultController : Controller
    {
        public ActionResult Index()
        {
```

```
            return View();
        }
    }
}
```

*Index.cshtml*

```
@{ Layout = null;}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width"/><title>Index</title>
</head>
<body>
    <div>
        <form>
            <h1>Test Form</h1>
            <input type="submit"/>
        </form>
        @CaptchaLibrary.CaptchaClass.Ekle(CaptchaLibrary.CaptchaClass.captchaSource.local)
        @*@CaptchaLibrary.CaptchaClass.Ekle(CaptchaLibrary.CaptchaClass.captchaSource.local, "papatya")
        @CaptchaLibrary.CaptchaClass.Ekle(CaptchaLibrary.CaptchaClass.captchaSource.database)*@
    </div>
</body>
</html>
```

*Opencv.cpp*

```
#include "stdafx.h"
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;
extern "C"
{
    __declspec(dllexport) bool MatrixVerify(unsigned char *u, char *cascade, int height, int width, int minsize, int maxsize,
int flags, double scaleFactor, int neighbors)
    {
        FileStorage cas(cascade, FileStorage::READ | FileStorage::MEMORY);
        Mat image = Mat(height, width, CV_8UC1, u);
        std::vector<Rect> object;
        CascadeClassifier c;
        c.read(cas.getFirstTopLevelNode());
        c.detectMultiScale(image, object, scaleFactor, neighbors, flags, Size(minsize, minsize), Size(maxsize, maxsize));
        return object.size();
    }
}
```

# ÖZGEÇMİŞ

7 Aralık 1988 tarihi Çankaya doğumluyum. İlköğretim eğitimimi Çankaya Hürriyet İlköğretim Okulunda, Lise eğitimimi Çankaya Anıttepe Lisesinde tamamladım. Anadolu Üniversitesi İktisat bölümünden 2015 yılında mezun oldum. Aynı yıl Beykent Üniversitesi Bilgisayar Mühendisliği Anabilim Dalında yüksek lisans eğitimime başladım.

**Fırat OĞUZ**