SAKARYA
ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

Ad Soyad: Ahmad Saflo

Öğrenci Numarası: G211210558

2. Öğretim L Grubu

Ders: BSM 401 Bilgisayar Mühendisliği Tasarımı

Danışman Öğretim Üyesi:

Dr.Öğr.Üyesi Can YÜZKOLLAR

İçindekiler

1. Bölüm Giriş

1.1 Proje Adı:

Çocuklar İçin Yapay Zeka Destekli Çizim Tanıma Uygulaması

1.2 Proje Fikri

Bu projede çocuklara yönelik bir yapay zeka uygulaması geliştirme fikri, çocukların erken yaşta teknoloji ile etkileşime geçerek öğrenmelerini teşvik etme ihtiyacından doğmuştur. Çocukların gelişim aşamalarında yapay zekayı eğlenceli bir şekilde deneyimleyebilecekleri, onların yaşına uygun ve güvenli bir ortam sunulması hedeflenmektedir. Yapay zeka teknolojisinin bu projeye dahil etmemin sebebi çocukların istedikleri zaman ve yanlarında kimse olmadığı zamanda da çizim yapabilmeleri içindir. yaratıcılık, ve çizim yaparak hayal gücü becerilerine katkı sağlaması amaçlanmaktadır.

1.3 Projenin Amaçları

Bu proje ile başarmayı hedeflediğim temel amaçları belirli alt başlıklarda açıklayabilirim:

1. **Eğitici ve Eğlenceli Bir Deneyim Sağlama**:

Çocukların çeşitli kategorilerde nesne çizimleri yaparak, eğlenceli bir biçimde öğrenmelerini sağlamak için amaçlanmıştır. Bu

uygulama, çocukların yaparak öğrenme ilkesi ile hareket eder ve onları aktif bir katılımcı olmaya teşvik eder.

2. **Ebeveynler ve Eğitimciler İçin Yeni Bir Araç Sunma**: Uygulama, eğitimcilerin ve ebeveynlerin çocukların öğrenme süreçlerini desteklemesi için yeni bir dijital araç olarak tasarlanmıştır. Yapay zeka ile desteklenen bu uygulamanın, çocukların öğrenme ve gelişim süreçlerine olumlu katkılar sağlaması beklenmektedir.

3. **Kategorize Etme Yeteneğini Geliştirme:** Uygulamanın çocukların çizim yaparak el-göz koordinasyonu, gördüklerini çizimle ifade etmeleri ve dikkat geliştirme yeteneklerine katkı sağlaması amaçlanmaktadır. Aynı zamanda kategorize etme ve nesneleri tanıma becerilerini güçlendirecektir.

1.4 Çalışmanın Gerekliliği ve Kapattığı Eksiklikler

1. **Çocukların Yaratıcılığını Destekleyen Eğitim Araçlarının Azlığı:** Günümüzde pek çok dijital uygulama, çocukların eğitimi için tasarlanmış olsa da, yaratıcılık ve hayal gücünü geliştirmeye odaklanan araçlar sınırlıdır. Bu proje, çocukların kendi çizimlerini yapmalarına imkan tanıyarak, onların özgün düşünme becerilerini ve sanatsal yönlerini geliştirmelerine olanak tanır. Böylece eğitim

teknolojilerindeki yaratıcılık odaklı uygulama eksikliğini kapatmayı hedeflemektedir.

2. **Çizimle Öğrenme ve Kendini İfade Etme Fırsatlarının Yetersizliği:**

Çizim yaparak öğrenme, çocukların dünyayı keşfetme ve kendilerini ifade etme yollarından biridir. Ancak, çoğu dijital uygulama bu tür aktif katılım yerine bilgi aktarımını ön plana çıkarmaktadır. Bu proje, çocukların çizimleri zerinden öğrenmelerine imkan tanıyarak, onların hem kendilerini ifade etmelerini hem de öğrenmelerini destekleyen bir araç olarak gereklidir.

2. Bölüm Problem Tanımı

Günümüz çocukları, dijital dünyayla büyüyen bir nesil olarak teknolojiyle sıkı bir ilişki içerisindedir. Erken yaşlarda teknolojiyle tanışmaları, onların hem öğrenme süreçlerini hızlandırabilir hem de yaratıcılıklarını geliştirebilir. Ancak, mevcut teknolojik araçlar genellikle çocukların pasif bir şekilde bilgi tüketmesine odaklanmaktadır. Bu durum, çocukların üretkenlik, problem çözme ve yaratıcılık becerilerinin yeterince desteklenmemesi gibi önemli bir eksikliği ortaya koymaktadır.

2.1 Çocukların Teknoloji Kullanımında Karşılaşılan Zorluklar

Eğitim teknolojileri alanındaki hızlı gelişmelere rağmen, çocukların teknoloji kullanımında aşağıdaki sorunlar sıklıkla gözlemlenmektedir:

1. **Pasif Katılım:** Çoğu dijital uygulama, çocukları aktif bir katılımcı olmaktan ziyade pasif bir bilgi tüketicisi yapmaktadır. Bu durum, çocukların öğrenme sürecine olan ilgisini azaltabilir ve uzun vadede yaratıcılıklarını sınırlayabilir.
2. **Yaratıcı ve Eğitici Araç Eksikliği:** Mevcut eğitim araçlarının çoğu, bilgi aktarımı veya sınav pratiği gibi dar

kapsamlı hedeflere odaklanmaktadır. Yaratıcılığı teşvik eden uygulamalar ise oldukça sınırlıdır.

3. **Kendi Kendine Öğrenme İmkanlarının Azlığı:** Çocukların bağımsız öğrenme ve keşfetme becerilerini geliştirecek uygulamaların sayısı oldukça azdır. Çizim gibi etkinlikler, çocukların hem bireysel becerilerini geliştirebileceği hem de problem çözme yetilerini artırabileceği alanlar sunabilir.

2.2 Çizimle Öğrenmenin Önemi

Çizim, çocukların öğrenme süreçlerinde kritik bir rol oynar. Çocuklar, bir nesneyi çizerken hem bilişsel hem de motor becerilerini kullanır. Aynı zamanda, bir fikri veya nesneyi çizerken onun özelliklerini analiz etme ve organize etme yeteneklerini geliştirir. Çizim, aynı zamanda çocukların el-göz koordinasyonu, dikkat becerileri ve görsel algılarını güçlendiren bir aktivitedir. Ancak, çizimle öğrenmeyi dijital bir platforma taşıyan uygulamaların sayısı oldukça sınırlıdır.

2.3 Bu Projenin Hedeflediği Çözüm

Bu proje, yukarıdaki sorunlara çözüm sunmak amacıyla geliştirilmiştir. Çocukların bir web sitesi üzerinde yer alan çizim alanında özgürce çizim yapmasını sağlayan bu uygulama, çizilen

nesneleri yapay zeka algoritmaları ile tanımlayarak geri bildirim verir. Bu özellik, çocukların bir yandan yaratıcılıklarını kullanmalarına olanak tanırken, diğer yandan da öğrenme süreçlerini interaktif bir şekilde destekler.

- **Yaratıcılığı Teşvik Etme:** Çocuklar, bir nesneyi çizerken hayal güçlerini kullanır ve yapay zekanın sunduğu geri bildirimlerle daha fazla keşfetmeye yönelir.
- **Bağımsız Öğrenme Fırsatları:** Çocuklar, uygulamayı kullanarak herhangi bir rehberliğe ihtiyaç duymadan öğrenebilir ve çizim yapabilir. Bu da onların bireysel öğrenme becerilerini geliştirir.
- **Kategorize Etme ve Tanıma Becerilerini Geliştirme:** Çizim yaparak çocuklar, nesneleri kategorize etme ve tanıma becerilerini eğlenceli bir şekilde geliştirebilir.

2.4 Mevcut Çözümler ve Projenin Farklılığı

Piyasada eğitim teknolojisi alanında geliştirilen bazı uygulamalar, çocukların çizim yapmasını teşvik etse de çoğunluğu statik içeriklerle sınırlıdır. Örneğin, bazı mobil uygulamalar sadece nesnelerin nasıl çizileceğini öğretmekle kalır; ancak çocukların çizimlerinin tanınması ve geri bildirim verilmesi gibi interaktif özellikler sunmaz. Bu proje, çocukların çizim yaparak öğrenmesini sağlayan ve yapay zekayla aktif bir geri bildirim

döngüsü oluşturan bir çözüm olarak öne çıkar. Ayrıca, uygulamanın kullanımı oldukça basittir ve çocuklar için güvenli bir öğrenme ortamı sağlar.

2.5 Sonuç

Bu proje, teknolojinin çocukların eğitimine daha anlamlı bir şekilde entegre edilmesi gerektiği fikrinden yola çıkarak geliştirilmiştir. Çocukların yaratıcılıklarını, öğrenme süreçlerini ve bağımsız düşünme yeteneklerini destekleyen bu uygulama, eğitim teknolojilerinde önemli bir boşluğu doldurmayı hedeflemektedir. Çizim yapmayı dijital ortamda eğitici bir etkinlik haline getirerek, çocukların teknolojiyi bilinçli ve faydalı bir şekilde kullanmalarını sağlayacaktır.

3. Bölüm Kullanılan Algoritmalar, Yazılımlar ve Donanımlar

Bu bölümde, proje geliştirme sürecinde kullanılan teknolojiler, yazılımlar ve algoritmalar detaylı bir şekilde açıklanmıştır. Geliştirilen uygulamanın başarıyla çalışabilmesi için hem güçlü yazılım araçları hem de modern algoritmalar tercih edilmiştir.

3.1 Kullanılan Yazılımlar ve Programlama Dili

1. **Visual Studio Code:**
   Proje geliştirme ortamı olarak Visual Studio Code tercih edilmiştir. Bu yazılım, geniş eklenti desteği, kullanıcı dostu arayüzü ve kod düzenleme kolaylığı sayesinde projeye büyük bir katkı sağlamıştır. Visual Studio Code, HTML, CSS, JavaScript ve Python gibi farklı dillerle çalışmayı destekleyerek proje geliştirme sürecini hızlandırmıştır.
2. **Python Programlama Dili:**
   Projenin temel algoritmalarının yazılması için Python programlama dili kullanılmıştır. Python, geniş kütüphane desteği ve yapay zeka projelerine olan uygunluğu sayesinde tercih edilmiştir. Özellikle veri işleme, makine öğrenimi ve model geliştirme süreçlerinde Python önemli bir rol oynamıştır.

3. **TensorFlow ve Keras:**
    - **TensorFlow:** Projede yapay zeka modellerinin eğitimi ve kullanımı için TensorFlow kütüphanesi tercih edilmiştir. TensorFlow, derin öğrenme modellerini oluşturma, eğitme ve dağıtma süreçlerinde oldukça güçlü bir araçtır.
    - **Keras:** TensorFlow üzerine inşa edilmiş, kullanıcı dostu bir API olan Keras, model geliştirme sürecini kolaylaştırmıştır. Keras sayesinde, veri seti üzerinde eğitim ve test işlemleri rahatlıkla gerçekleştirilmiştir.
4. **Flask:**
   Projenin web tabanlı bir uygulama olarak sunulabilmesi için Flask framework'ü kullanılmıştır. Flask, hafif ve esnek bir yapıya sahip olmasıyla bilinir. Projenin backend kısmında Flask, model tahminlerini web arayüzüne aktarmak için kullanılmıştır. Flask, Python tabanlı olması sayesinde TensorFlow modeli ile kolayca entegre edilmiştir.
5. **HTML, CSS ve JavaScript:**
    - **HTML:** Web sayfasının yapısını oluşturmak için kullanılmıştır. Çizim alanı (canvas) ve diğer web bileşenlerinin tasarımı HTML ile yapılmıştır.

- **CSS:** Uygulamanın görsel tasarımı ve estetiği için CSS kullanılmıştır. Çocuklara hitap eden, renkli ve eğlenceli bir tasarım oluşturulmuştur.
- **JavaScript:** Web arayüzündeki etkileşimlerin yönetimi ve çizim alanı üzerindeki işlemlerin kontrolü için JavaScript kullanılmıştır. JavaScript, kullanıcıların çizimlerini gerçek zamanlı olarak görselleştirme ve işlem yapma süreçlerinde aktif bir rol oynamıştır.

3.2 Kullanılan Algoritmalar

Convolutional Neural Network (CNN):

Projede, **Convolutional Neural Network (CNN)** mimarisi kullanılmıştır. CNN, özellikle görüntü işleme ve sınıflandırma gibi görevlerde yaygın olarak kullanılan güçlü bir derin öğrenme modelidir. CNN, her bir katmanda görüntüden daha karmaşık özellikler çıkararak son katmanda bu bilgileri sınıflandırma amacıyla kullanır. CNN modeli, görüntülerdeki desenleri tanıma, nesneleri ayırt etme ve sınıflandırma gibi görevlerde son derece etkilidir.

CNN Mimarisi:

- **Convolutional Katmanlar (Evrişim Katmanları):** CNN'in temel bileşenlerinden biridir. Görüntüdeki belirli özellikleri,

filtreler (kernels) aracılığıyla tanır. Bu katmanlar, düşük seviyedeki özellikleri (örneğin kenarları, dokuları) öğrenir.

- **Pooling Katmanları:** Bu katmanlar, her evrişim katmanından çıkan veriyi daha küçük bir boyuta indirger. Genellikle **Max Pooling** kullanılır ve bu, modelin hesaplama verimliliğini artırır.
- **Flatten Katmanı:** Görüntüdeki yüksek seviyeli özellikleri tek bir vektöre dönüştürür. Bu vektör daha sonra sınıflandırma işlemi için kullanılır.
- **Fully Connected Katmanlar (Tam Bağlantılı Katmanlar):** CNN'in son katmanlarıdır ve görüntüdeki özellikleri, etiketli sınıflarla ilişkilendirir. Bu katmanlar, görüntüyü tanıyan ağın sonuca ulaşmasını sağlar.

Model Eğitim Süreci:

Proje için geliştirilen CNN modeli, **görüntü tanıma** amacıyla eğitilmiştir. Çocuklar tarafından çizilen nesneleri tanımak için model, çizim verisiyle eğitilmiş ve ardından test edilmiştir. Eğitim sırasında, modelin doğruluğunu artırmak için **veri artırma** teknikleri ve **loss function** (kayıp fonksiyonu) olarak **categorical crossentropy** kullanılmıştır.

CNN Modelinin Faydaları:

- **Görsel Veriyi Tanıma:** CNN, görsel verilerdeki desenleri tanımada oldukça başarılıdır. Bu, çocukların çizimlerinin doğru bir şekilde sınıflandırılmasını sağlar.
- **Yüksek Doğruluk Oranı:** CNN, derin öğrenme tabanlı bir model olduğu için, büyük veri setlerinde yüksek doğruluk oranları elde edilebilir. Bu da projenin amacına uygun bir çözüm sunar.

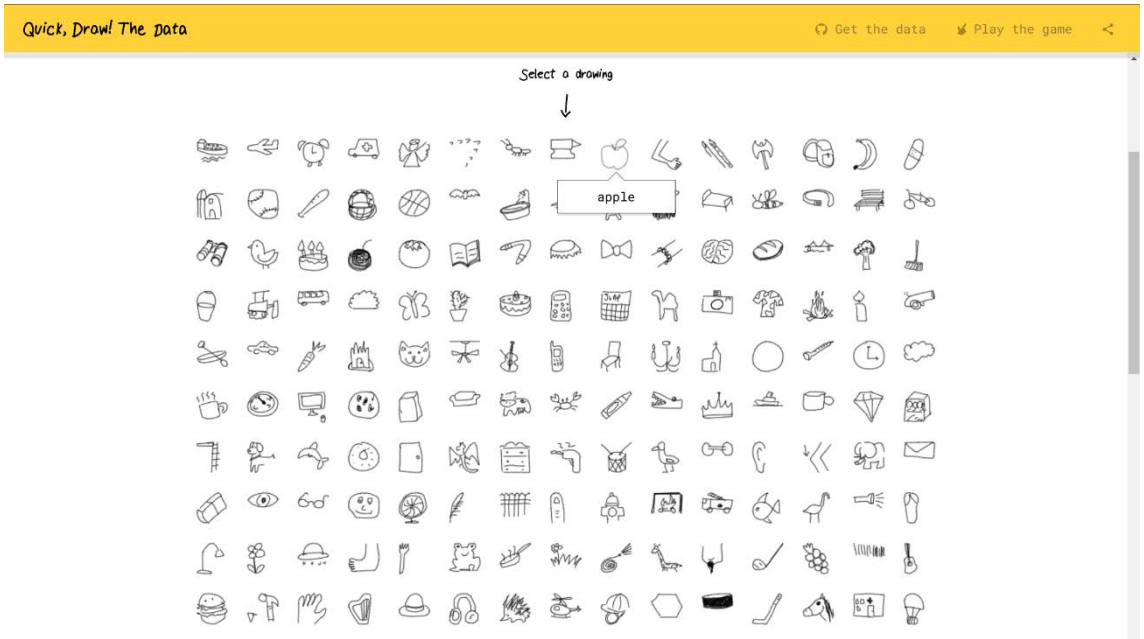
3.3 Kullanılan Donanım ve Ortam

1. **Bilgisayar Donanımı:**
    - İşlemci: Intel i7-8565U, model eğitimi ve test süreçlerinde yeterli performansı sağlamıştır.
    - Bellek: 8 GB RAM, büyük veri setlerini işlemek ve model eğitimi sırasında yeterli değildi bu biraz zorlamıştı eğitirken.
    - GPU: NVIDIA GeForce MX250 2GB, TensorFlow ile yapılan derin öğrenme işlemlerini hızlandırmak için kullanılmıştır.

4. Bölüm Veri Seti ve Algoritma

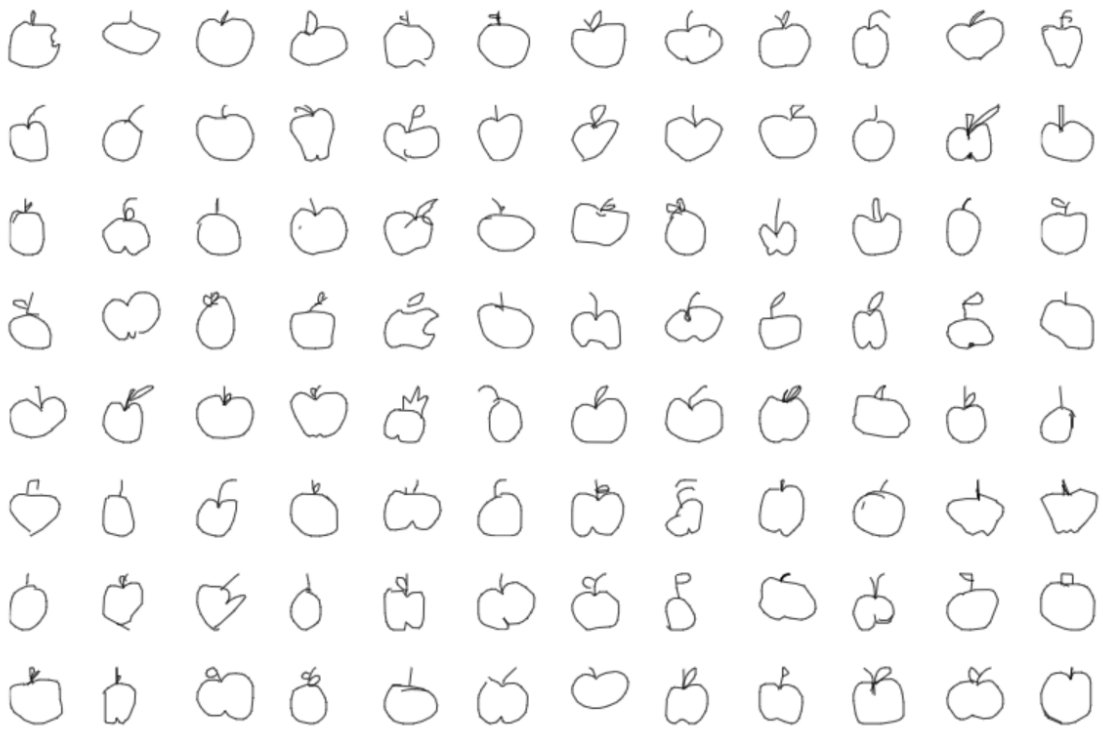
4.1 Veri Seti

Veri seti olarak, Google'ın **"Quick, Draw!"** (1) projesinden elde edilen çizim verileri kullanılmıştır. Quick, Draw! veri seti, dünya genelinde milyonlarca kullanıcı tarafından çizilen 345 farklı nesneyi içeren, büyük bir çizim veri setidir. Bu veri seti, her bir çizimin vektör formatında saklanmasını sağlar, bu da çizimlerin zamana dayalı bir şekilde temsil edilmesini mümkün kılar. Veri seti, eğitim amacıyla kullanılabilecek 50 milyondan fazla çizim



(resim 4.1.1) Dataseti Örnekleri

Projede kullandığım veri seti, her bir kategori için çeşitli çizimler içermektedir. Veri seti, kullanıcıların belirli nesneleri çizmeleri istendiğinde ortaya çıkan çizimleri içerir ve her çizim bir nesne kategorisiyle etiketlenir. Bu veriler, modelin farklı kategorilerdeki çizimleri öğrenmesi için kullanılır.



(resim 4.1.2) Bir Kategoride datasetin içerik örnekleri elma kategorisi

Çizimler, belirli kategoriler altında sınıflandırılmıştır. Bu kategoriler, örneğin "apple", "banana", "bus", "carrot", "cloud" gibi çeşitli günlük hayatta karşılaşılan nesneleri içerir. Veri setindeki her kategori, birden fazla çizim örneği ile resimde gösterildiği gibi temsil edilmiştir.

```python
import numpy as np
import cv2
import os
import ndjson

# resme cevirip kaydetmek icindir
def vektordenResme(cizimler,boyut=256):
    canvas=np.zeros((boyut,boyut))
    for stroke in cizimler:
        x,y=stroke
        for i in range(len(x)-1):
            canvas =  cv2.line(canvas,(x[i],y[i]),(x[i+1],y[i+1]),255,3)
    return 255 -canvas

kayitYolu="C:/Users/ahmad/Desktop/25kategori1500Resim"
dosyaAdresi = 'C:/Users/ahmad/Desktop/yenidatandjson'
kategorilerdizi = ["apple","banana","basketball","broccoli","broom", "bus", "camera",
                   "carrot", "chair","circle","cloud","door","eye","flower","hand",
                   "ladder","parachute", "pear","pineapple","rainbow","square",
                   "star","strawberry","sun","triangle"]
for j in kategorilerdizi:
    acma_yolu = os.path.join(dosyaAdresi,f'full_simplified_{j}.ndjson')
    def cizimleri_yukle(dosyaAdresi,limit=1001):
        with open(dosyaAdresi) as f:
            data=ndjson.load(f)
            return data[:limit]

    data=cizimleri_yukle(acma_yolu)
    print(f"{len(data)} cizim yuklendi {j} datalari. ")
    os.makedirs(f"C:/Users/ahmad/Desktop/25kategori1500Resim/{j}")
    print(f"{j} olusturuldu. ")

    for i in range(len(data)-1):
        cizimler = data[i]['drawing']
        resim = vektordenResme(cizimler)
        kayit_yolu = os.path.join(kayitYolu,f'{j}', f'{j}{i}.png')
        cv2.imwrite(kayit_yolu, resim)

```

(resim 4.1.3) Resimlere dönüştürüp kategorilere ayırarak kaydetmek için kullanılan kod

Veri setini, daha verimli bir şekilde işleyebilmek için çizimlerden görseller oluşturulmuştur. Bu işlem, çizimlerin vektör formatından, daha klasik görsel formatlarına dönüştürülmesini içerir. Bu sayede, modelin eğitimine uygun formatta veriler elde edilmiştir.

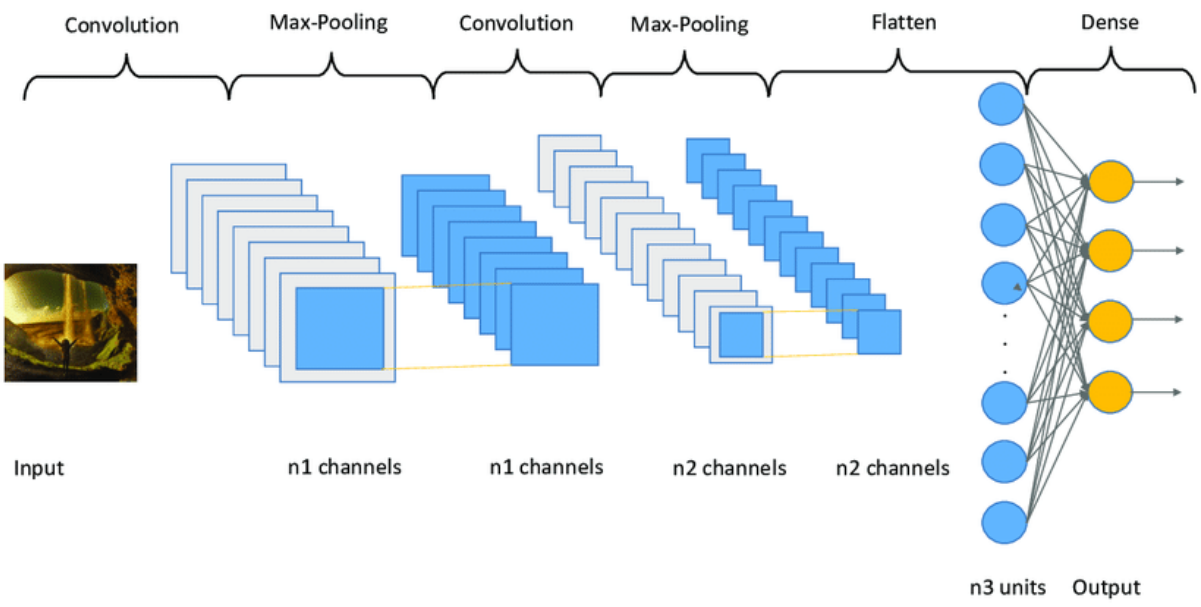
4.2 Kullanılan Algoritmalar ve Çözüm

Projemde kullandığım temel algoritma, **Convolutional Neural Network (CNN)** (2) (3) mimarisidir. CNN, görüntü verilerini analiz etmek ve sınıflandırmak için yaygın olarak kullanılan derin öğrenme algoritmalarından biridir. Çizim tanıma gibi görsel veri sınıflandırma görevlerinde yüksek başarı elde etmemi sağlayan bu model, çeşitli katmanlardan oluşur:

1. **Konvolüsyonel Katmanlar:** Görüntüdeki önemli özellikleri çıkartır.
2. **Havuzlama Katmanları:** Özelliklerin boyutlarını küçültür ve modelin daha hızlı öğrenmesini sağlar.
3. **Tam Bağlantılı Katmanlar:** Çıkartılan özellikleri kullanarak sınıflandırma işlemi yapar.

Bu mimari, modelin çizimlerin temel özelliklerini öğrenmesini ve doğru bir şekilde sınıflandırma yapmasını sağlar. Proje kapsamında, TensorFlow ve Keras kütüphanelerini kullanarak bu CNN modelini inşa ettim.

Modelin eğitimi için, veri setindeki çizimler resim formatına dönüştürülüp etiketlerle birlikte kullanılmıştır. Bu sayede, model her çizimi bir kategoriye ait olarak öğrenmiştir.



(resim 4.2.1) Örnek bir CNN modeli ve katmanları (5)

4.3 Model Mimarisi, Katmanları ve Eğitim Kodları

4.3.1 Model Mimarisi ve Katmanlar

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 256, 256, 3)	0
rescaling (Rescaling)	(None, 256, 256, 3)	0
conv2d (Conv2D)	(None, 256, 256, 16)	448
max_pooling2d (MaxPooling2D)	(None, 128, 128, 16)	0
conv2d_1 (Conv2D)	(None, 128, 128, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_2 (Conv2D)	(None, 64, 64, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 64)	0
dropout (Dropout)	(None, 32, 32, 64)	0
flatten (Flatten)	(None, 65536)	0
dense (Dense)	(None, 128)	8,388,736
dense_1 (Dense)	(None, 22)	2,838

(resm 4.3.1) Model Mimarisi ve Katmanları (VSCode uygulamasından)

1. **Rescaling Katmanı**:
    - **Rescaling**: Giriş verisi olarak alınan piksel değerlerini ölçekler. Genellikle 0-255 aralığındaki piksel değerlerini [0,1] aralığına normalize eder.
    - **Çıkış Boyutu**: (256, 256, 3)
    - **Parametre Sayısı**: 0 (Bu katmanda öğrenilecek parametre yoktur).
2. **Birinci Konvolüsyon ve Havuzlama Bloğu**:
    - **Conv2D (16 filtre)**: 3x3 boyutundaki filtrelerle özellik çıkarımı yapılır.

- **MaxPooling2D**: 2x2 havuzlama işlemiyle özellik haritası boyutu yarıya indirilir.
- **Çıkış Boyutu**: (128, 128, 16)
- **Parametre Sayısı**: 448 (16 filtre * 3x3 * 3 kanal + bias).

3. **İkinci Konvolüsyon ve Havuzlama Bloğu**:
   - **Conv2D (32 filtre)**: Daha fazla filtre ile daha karmaşık özellikler öğrenilir.
   - **MaxPooling2D**: Boyut tekrar yarıya indirilir.
   - **Çıkış Boyutu**: (64, 64, 32)
   - **Parametre Sayısı**: 4,640.
4. **Üçüncü Konvolüsyon ve Havuzlama Bloğu**:
   - **Conv2D (64 filtre)**: Daha karmaşık ve detaylı özellikler çıkarılır.
   - **MaxPooling2D**: Boyut tekrar küçültülür.
   - **Çıkış Boyutu**: (32, 32, 64)
   - **Parametre Sayısı**: 18,496.
5. **Dropout Katmanı**:
   - **Dropout**: Aşırı öğrenmeyi (overfitting) önlemek için bazı nöronlar rastgele devre dışı bırakılır.
   - **Parametre Sayısı**: 0.
6. **Düzleştirme Katmanı**:
   - **Flatten**: Konvolüsyon katmanlarından gelen çok boyutlu veriyi tek boyutlu bir vektöre dönüştürür.
   - **Çıkış Boyutu**: (65536)
   - **Parametre Sayısı**: 0.

7. **Tam Bağlantılı Katmanlar (Dense Layers)**:
    - **Dense (128 nöron)**: Çıkış boyutu 128 olan tam bağlantılı bir katmandır.
    - **Parametre Sayısı**: 8,388,736 (65536 giriş * 128 nöron + bias).
    - **Dense (22 nöron)**: Çıkış boyutu 22 olan sınıflandırma katmanıdır.
    - **Parametre Sayısı**: 2,838 (128 giriş * 22 nöron + bias).

4.3.2 Model Eğitim Kodları

```
import os
import numpy as np

import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import load_img, ImageDataGenerator
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
```

(resim 4.3.2.1) Gerekli kütüphanelerin eklenmesi

```
count = 0
dirs = os.listdir('25kategori1500Resim/')
for dir in dirs:
    files = list(os.listdir('25kategori1500Resim/'+dir))
    print( dir +' kategorisinde '+ str(len(files)) + ' resim var')
    count = count + len(files)
print( 'toplam '+ str(count) + ' resim var')
```

(resim 4.3.2.2) Resimleri alma ve sayma kodu

(resim 4.3.2.2)'deki kod, belirli bir klasördeki alt klasörlerin her birinde bulunan resimlerin sayısını hesaplamak için kullanılır. İlk olarak, os.listdir() fonksiyonu ile '25kategori1500Resim/' klasöründeki tüm alt klasörler (dirs) liste olarak alınır. Ardından

her bir alt klasör için, yine os.listdir() kullanılarak o klasördeki dosyalar (files) listelenir ve her bir klasördeki dosya sayısı ekrana yazdırılır. Son olarak, tüm alt klasörlerdeki toplam resim sayısı (count) hesaplanır ve bu toplam da ekrana yazdırılır.

```
apple kategorisinde 1000 resim var
banana kategorisinde 1000 resim var
basketball kategorisinde 1000 resim var
broccoli kategorisinde 1000 resim var
broom kategorisinde 1000 resim var
bus kategorisinde 1000 resim var
camera kategorisinde 1000 resim var
carrot kategorisinde 1000 resim var
chair kategorisinde 1000 resim var
circle kategorisinde 1000 resim var
cloud kategorisinde 1000 resim var
door kategorisinde 1000 resim var
flower kategorisinde 1000 resim var
hand kategorisinde 1000 resim var
ladder kategorisinde 1000 resim var
pear kategorisinde 1000 resim var
pineapple kategorisinde 1000 resim var
rainbow kategorisinde 1000 resim var
square kategorisinde 1000 resim var
star kategorisinde 1000 resim var
sun kategorisinde 1000 resim var
triangle kategorisinde 1000 resim var
toplam 22000 resim var
```

(resim 4.3.2.3) bir önceki kodun ekran çıktısı

```python
base_dir = '25kategori1500Resim/'
img_size = 256
batch = 16

train_ds = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed = 123,
    validation_split=0.2,
    subset = 'training',
    batch_size=batch,
    image_size=(img_size,img_size))

val_ds = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed = 123,
    validation_split=0.2,
    subset = 'validation',
    batch_size=batch,
    image_size=(img_size,img_size))
```

(resim 4.3.2.4) Veri Kümesi Oluşturma

Bu kod, TensorFlow ile eğitim ve doğrulama veri kümeleri oluşturur. base_dir ile görüntülerin bulunduğu klasör belirlenir, img_size ile görüntü boyutu 256x256, batch ile batch boyutu 16 yapılır. train_ds, %20'lik doğrulama ayrılarak eğitim verisi

oluştururken, val_ds doğrulama verisini oluşturur. Her iki veri kümesi de 16 görüntü içerir ve 256x256 piksele yeniden boyutlandırılır. Bu şekilde modelin eğitim ve doğrulama verileri hazırlanır.

```
Found 22000 files belonging to 22 classes.
Using 17600 files for training.
Found 22000 files belonging to 22 classes.
Using 4400 files for validation.
```

(resim 4.3.2.5) Bir önceki kodun çıktısı

```
drawing_names = train_ds.class_names
drawing_names
```

```
['apple',
 'banana',
 'basketball',
 'broccoli',
 'broom',
 'bus',
 'camera',
 'carrot',
 'chair',
 'circle',
 'cloud',
 'door',
 'flower',
 'hand',
 'ladder',
 'pear',
 'pineapple',
 'rainbow',
 'square',
 'star',
 'sun',
 'triangle']
```

(resim 4.3.2.6) Sınıfların isimlerini kaydedip ekrana yazdıran kod ve çıktısı

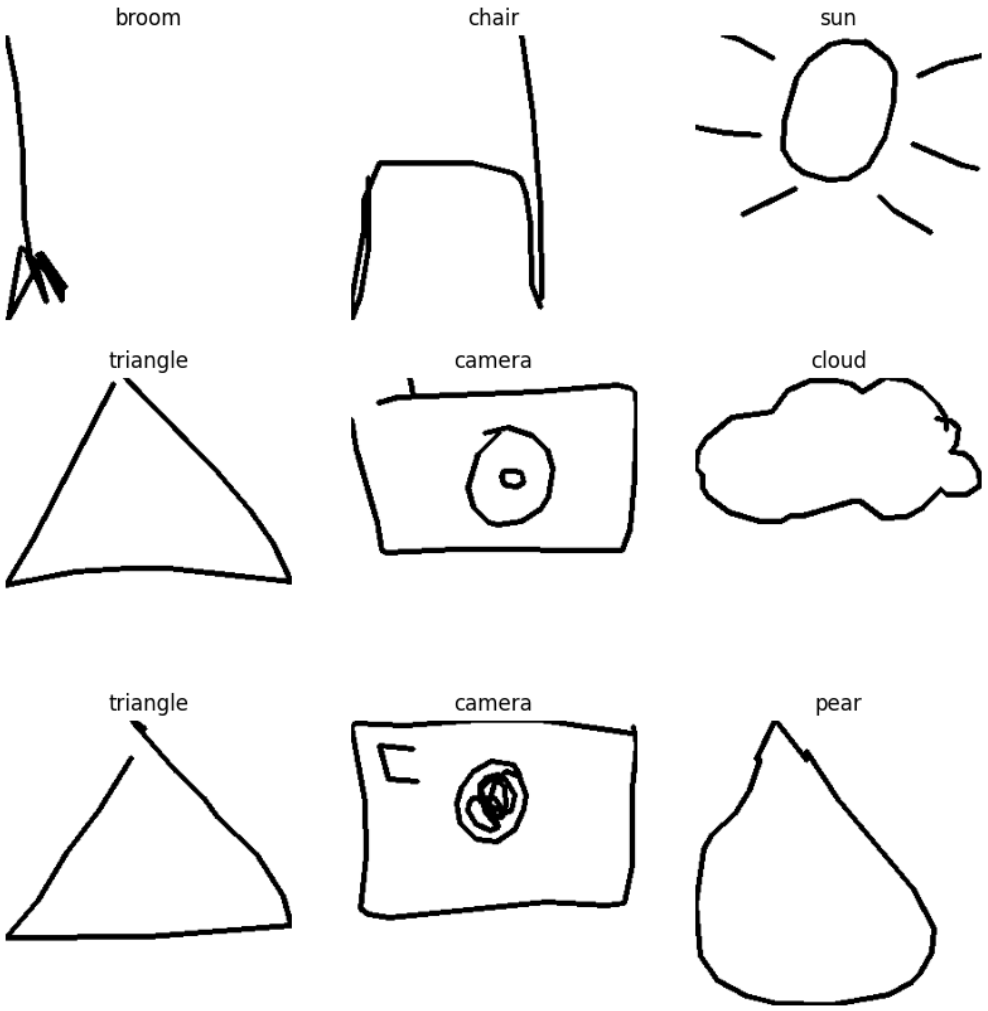
```
import matplotlib.pyplot as plt

i = 0
plt.figure(figsize=(10,10))

for images, labels in train_ds.take(1):
    for i in range(9):
        plt.subplot(3,3, i+1)
        plt.imshow(images[i].numpy().astype('uint8'))
        plt.title(drawing_names[labels[i]])
        plt.axis('off')
```

(resim 4.3.2.7) Veri kümesinden rastgele seçilen 9 görüntü ve etiketleri

Bu kod, TensorFlow ile yüklenen eğitim verisinden rastgele 9 görüntüyü görselleştirir. plt.figure(figsize=(10,10)) ile grafik penceresinin boyutu ayarlanır. train_ds.take(1) ile veri kümesinden bir batch alınır ve bu batch'teki ilk 9 görüntü, plt.subplot(3,3, i+1) ile 3x3'lük bir ızgarada sırasıyla yerleştirilir. Her bir görüntü plt.imshow(images[i].numpy().astype('uint8')) ile görselleştirilir, başlık olarak etiketin karşılık geldiği kategori adı plt.title(drawing_names[labels[i]]) ile eklenir, ve eksenler plt.axis('off') ile gizlenir. Sonuçta, 9 görüntü başlıklarıyla birlikte görselleştirilir aşağıdaki resimde gösterildiği gibi.



(resim 4.3.2.8) Bir önceki resmin ekran çıktısı

```
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size = AUTOTUNE)

#veri arttirmak
data_augmentation = Sequential([
    layers.RandomFlip("horizontal", input_shape = (img_size,img_size,3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])
```

(resim 4.3.2.9) Veri ön işleme ve artırma işlemleri kodu

İlk olarak, train_ds veri kümesi cache edilir, rastgele karıştırılır ve verilerin yüklenmesi sırasında önceden yüklenmiş veriler kullanılarak hızlandırılır (prefetch). val_ds veri kümesi de cache edilir ancak sadece ön yükleme (prefetch) yapılır, çünkü doğrulama veri kümesinde genellikle veri artırma yapılmaz.

Sonra, veri artırma (data augmentation) işlemi için bir dizi dönüşüm tanımlanır. Bu dönüşümler, eğitim verileri üzerinde rastgele yatay çevirme, dönüşüm ve yakınlaştırma işlemleri yapar. Bu, modelin genelleme yeteneğini artırmak ve aşırı öğrenmeyi önlemek amacıyla kullanılır.

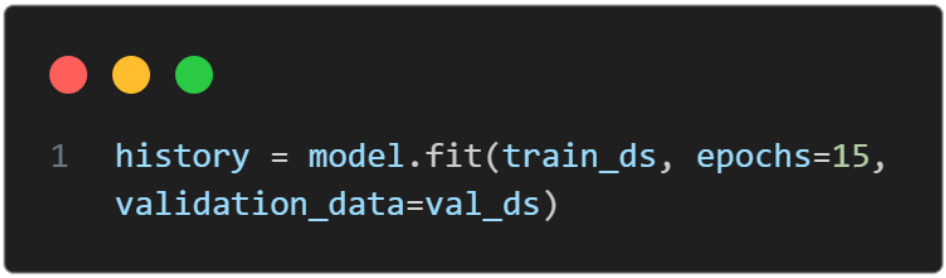
```python
#model olusturuyorum
model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255),
    Conv2D(16, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Dropout(0.2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(22)
])
```

(resim 4.3.2.10) Model oluşturma kodu

```python
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

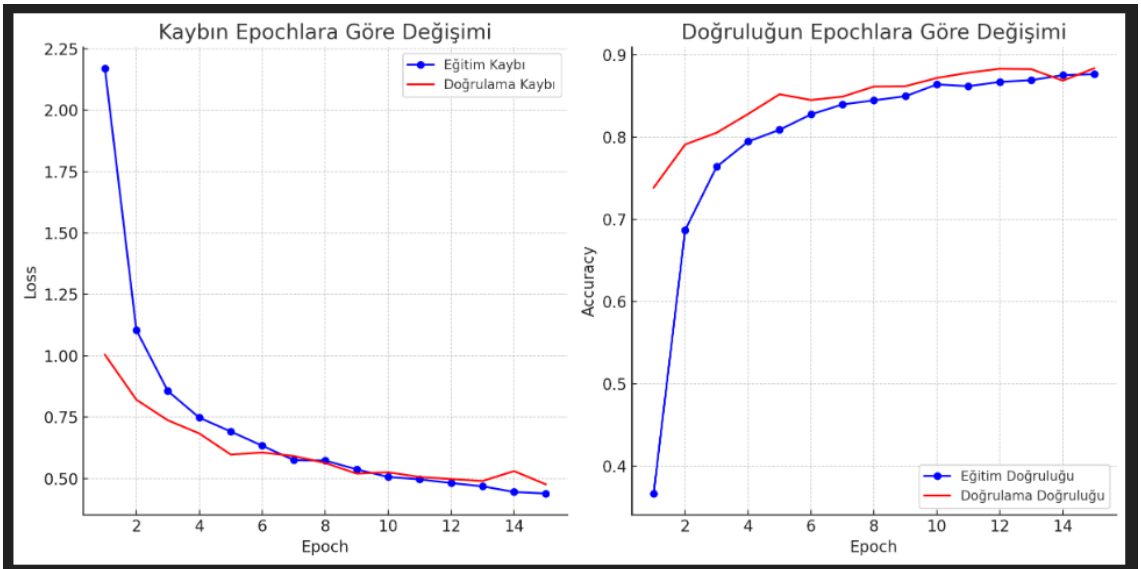
(resim 4.3.2.11) Model eğitim sürecinin yapılandırılması

Bu kod, modelin eğitim sürecini yapılandırır. Adam optimizasyon algoritması, modelin parametrelerini günceller. SparseCategoricalCrossentropy kayıp fonksiyonu, çok sınıflı sınıflandırma problemleri için kullanılır ve etiketlerin tam sayı formatında (sınıf indeksleri) olduğunu varsayar. from_logits=True parametresi, modelin son katmanının doğrusal çıktılar verdiğini belirtir. accuracy metriği ise modelin doğruluğunu izler, yani doğru sınıf tahminlerinin oranını hesaplar.



(resim 4.3.2.12) Model eğitimi başlatma kodu

Bu kod, bir makine öğrenimi modelini eğitmek için kullanılan fit() fonksiyonunun bir örneğidir. Kodda, modelin eğitim verisi (train_ds) üzerinde 15 dönem (epoch) boyunca eğitim yapması ve her bir dönem sonunda doğrulama verisi (val_ds) ile modelin doğruluğunun test edilmesi sağlanmıştır. Eğitimin her aşamasında model, belirtilen eğitim verisi üzerinde parametrelerini optimize ederken, doğrulama verisi ile de genel performansını kontrol eder. Bu işlem, modelin öğrenme sürecini takip etmek ve aşamalı olarak daha iyi bir performans elde etmek amacıyla yapılır.




```
Epoch 1/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 346s 284ms/step - accuracy: 0.3664 - loss: 2.1704 - val_accuracy: 0.7384 - val_loss: 1.0039
Epoch 2/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 308s 280ms/step - accuracy: 0.6868 - loss: 1.1039 - val_accuracy: 0.7909 - val_loss: 0.8207
Epoch 3/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 303s 276ms/step - accuracy: 0.7638 - loss: 0.8560 - val_accuracy: 0.8052 - val_loss: 0.7370
Epoch 4/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 305s 277ms/step - accuracy: 0.7945 - loss: 0.7475 - val_accuracy: 0.8280 - val_loss: 0.6832
Epoch 5/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 301s 274ms/step - accuracy: 0.8088 - loss: 0.6912 - val_accuracy: 0.8520 - val_loss: 0.5974
Epoch 6/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 311s 282ms/step - accuracy: 0.8275 - loss: 0.6333 - val_accuracy: 0.8450 - val_loss: 0.6058
Epoch 7/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 308s 280ms/step - accuracy: 0.8396 - loss: 0.5749 - val_accuracy: 0.8491 - val_loss: 0.5913
Epoch 8/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 307s 279ms/step - accuracy: 0.8446 - loss: 0.5730 - val_accuracy: 0.8614 - val_loss: 0.5633
Epoch 9/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 305s 278ms/step - accuracy: 0.8496 - loss: 0.5371 - val_accuracy: 0.8618 - val_loss: 0.5200
Epoch 10/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 305s 277ms/step - accuracy: 0.8640 - loss: 0.5063 - val_accuracy: 0.8718 - val_loss: 0.5253
Epoch 11/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 307s 279ms/step - accuracy: 0.8618 - loss: 0.4967 - val_accuracy: 0.8782 - val_loss: 0.5061
Epoch 12/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 308s 280ms/step - accuracy: 0.8670 - loss: 0.4815 - val_accuracy: 0.8830 - val_loss: 0.4978
Epoch 13/15
...
Epoch 14/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 301s 273ms/step - accuracy: 0.8752 - loss: 0.4451 - val_accuracy: 0.8686 - val_loss: 0.5303
Epoch 15/15
1100/1100 ━━━━━━━━━━━━━━━━━━━━ 304s 277ms/step - accuracy: 0.8765 - loss: 0.4389 - val_accuracy: 0.8836 - val_loss: 0.4766
```

(resim 4.3.2.13) Model öğrenme süreci ve performans takip grafikleri

4.3.3 Kullanım Arayüzü

4.3.4 Flask API Kullanımı

```python
from flask import Flask, request, jsonify,render_template
import base64
from io import BytesIO
from PIL import Image
import os
from tensorflow.keras.models import load_model
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array

import tensorflow as tf
import numpy as np

app = Flask(__name__)
model = load_model('Drawing_Recog_125.h5')

# Resimlerin kaydedileceği dizin
UPLOAD_FOLDER = './kaydedilen_resimler'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

@app.route('/')
def index():
    return render_template('cizim.html')

@app.route('/kaydet', methods=['POST'])
def kaydet():
    try:
        # JSON verisini al
        data = request.get_json()
        image_data = data['image']

        # Base64 kodunu çöz
        image_data = image_data.split(',')[1]  # "" kısmını kaldır
        image_bytes = base64.b64decode(image_data)

        # Resmi kaydet
        image = Image.open(BytesIO(image_bytes))
        image.save(os.path.join(UPLOAD_FOLDER, 'cizim.png'))

        image_path= "./kaydedilen_resimler/cizim.png"
        image = load_img(image_path, target_size=(256, 256))
        image_array = img_to_array(image)
        exp_dim = tf.expand_dims(image_array,0)
        predictions = model.predict(exp_dim)
        result = tf.nn.softmax(predictions[0])
        drawing_names = ['Elma', 'Muz', 'Basketbol', 'Brokoli', 'Süpürge', 'Otobüs', 'Kamera',
                    'Havuç', 'Sandalye', 'Daire', 'Bulut', 'Kapı', 'Çiçek', 'El', 'Merdiven',
                    'Armut', 'Ananas', 'Gökkuşağı', 'Kare', 'Yıldız', 'Güneş', 'Üçgen']

        outcome = str(np.max(result)*100) + "% oranla "+ drawing_names[np.argmax(result)] + " olduğunu düşünüyorum..."

        return jsonify({'message': outcome})
    except Exception as e:
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

(resim 4.3.4) Flask kullanımı ve python kodu arayüzü bağlayan kod

(resim 4.3.4) 'deki bu kod, bir Flask uygulaması (4) oluşturarak bir AI destekli çizim tanıma uygulaması sunar. İşlevsellikleri şu şekildedir:

1. **Flask Uygulaması Oluşturma (app = Flask(__name__))**:
    - Flask framework'ünü kullanarak bir web uygulaması başlatır.
2. **Model Yükleme (model = load_model('Drawing_Recog_125.h5'))**:
    - 'Drawing_Recog_125.h5' adlı eğitilmiş bir TensorFlow modelini yükler. Bu model, çocukların çizimlerini tanımak için kullanılır.
3. **Resim Yükleme Dizini Tanımlama (UPLOAD_FOLDER = './kaydedilen_resimler')**:
    - Çizimlerin kaydedileceği dizini tanımlar ve varsa oluşturur (os.makedirs(UPLOAD_FOLDER, exist_ok=True)).
4. **Ana Sayfa (@app.route('/'))**:
    - Ana sayfayı temsil eden bir route tanımlar (index() fonksiyonu ile render_template kullanarak 'cizim.html' şablonunu döndürür).
5. **Resim Kaydetme (@app.route('/kaydet', methods=['POST']))**:
    - /kaydet endpoint'ine POST isteği alır.
    - Gelen JSON verisinden çizim verisini alır (data['image']).
    - Base64 kodunu çözerek resmi oluşturur ve 'cizim.png' olarak kaydeder.
    - Kaydedilen resmi yükler, TensorFlow ile modelde tahmin yapar ve sonucu döndürür.

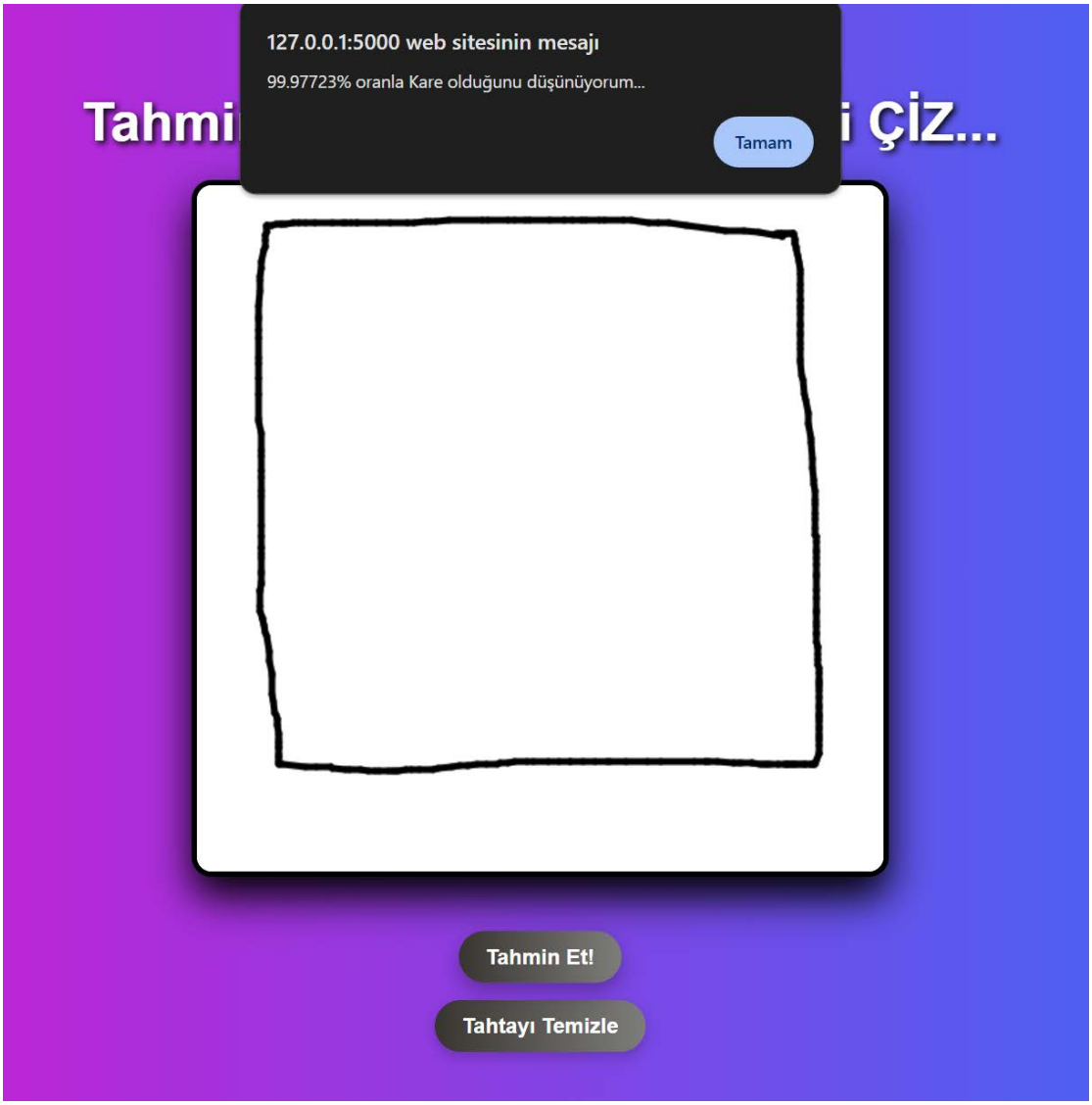
- Tahmin sonucunu, en yüksek olasılıklı nesne adı ve olasılık yüzdesiyle birlikte JSON formatında döndürür (jsonify({'message': outcome})).

6. **Hata Yönetimi**:
   - Herhangi bir istisna durumunda (Exception) hata mesajını JSON formatında döndürür (jsonify({'error': str(e)})).
7. **Uygulamayı Başlatma (if __name__ == '__main__':)**:
   - Uygulamayı debug modunda çalıştırır (app.run(debug=True)).

Bu kod, web üzerinden gönderilen çizim verilerini alır, bunları eğitilmiş bir model üzerinden tahminlerde bulunmak için kullanır ve sonucu kullanıcıya geri döner. Flask kullanarak basit bir API oluşturarak, çocukların çizimlerini tanımlayan interaktif bir uygulama sunar.

5. Bölüm Test ve Sonuç

(resim 5.1) Test 1



(resim 5.2) Test 2

(resim 5.3) Test 3

(resim 5.4) Test 4

6. Proje Planı ve Maliyet Analizi

6.1 Proje Planı

Proje, başlangıçtan itibaren aşağıdaki adımlarla planlanmıştır:

1. **Araştırma ve Ön Hazırlık**: Proje fikrinin geliştirilmesi ve gerekli kaynakların belirlenmesi.
2. **Veri Seti Hazırlığı**: Çizim verilerinin toplanması ve uygun bir veri setinin seçimi.
3. **Model Geliştirme**: TensorFlow kullanılarak VGG16 modeli üzerinde eğitim, test ve doğrulama çalışmaları.
4. **Uygulama Geliştirme**: Çocuklar için kullanıcı dostu bir ara yüz tasarımı ve uygulamanın entegrasyonu.
5. **Test ve Değerlendirme**: Prototipin test edilmesi ve geri bildirimlere göre iyileştirilmesi.
6. **Proje Raporlama**: Proje sonuçlarının belgelenmesi ve final sunumu.

Bu plan çerçevesinde proje zamanında tamamlanmış ve hedeflenen işlevsellik sağlanmıştır.

6.2 Maliyet Analizi

Bu proje, tamamen kişisel bilgi ve yetenekler ile gerçekleştirilmiş olup herhangi bir maddi maliyet gerektirmemiştir. Bunun başlıca sebepleri şunlardır:

- TensorFlow gibi açık kaynaklı yazılımların kullanılması.
- Veri setinin ücretsiz olarak erişilebilir olması.
- Geliştirme sürecinde herhangi bir üçüncü taraf ekipman veya hizmet gereksiniminin olmaması.

- Projenin kişisel bilgisayar ve mevcut altyapılar kullanılarak geliştirilmesi.

Sonuç olarak, proje tamamen maliyetsiz bir şekilde tamamlanmıştır, ancak yüksek seviyede zaman, emek ve öğrenme süreci gerektirmiştir.

Kaynakça

1. **Kullanılan Dataset.** ***Quick, Draw!*** **[Online] https://quickdraw.withgoogle.com/data.**

2. CNN modeli anlatımı. ***Fırat Bulak.*** **[Çevrimiçi] YouTube Video, 2021. https://www.youtube.com/watch?v=SVeGBgI7MYg.**

3. Convolutional Neural Network (CNN): A Complete Guide. ***LearnOpenCV.*** **[Çevrimiçi] january 2023. https://learnopencv.com/understanding-convolutional-neural-networks-cnn/.**

4. Flask User Guide. ***Flask's documentation.*** **[Çevrimiçi] https://flask.palletsprojects.com/en/stable/.**

5. Evrişimsel Sinir Ağları (CNN — Convolutional Neural Networks) Katmanları. ***Medium.*** **[Çevrimiçi] 21 June 2023. https://medium.com/yaz%C4%B1l%C4%B1m-ve-bili%C5%9Fim-kul%C3%BCb%C3%BC/evri%C5%9Fimsel-sinir-a%C4%9Flar%C4%B1-cnn-convolutional-neural-networks-katmanlar%C4%B1-d7d42bf8e712.**