*******Sakarya* *Üniversitesi*

*Bilgisayar ve Bilişim Bilimleri Fakültesi*

*Bilgisayar Mühendisliği bölümü*

**Derin Öğrenmede Evrişimsel Sinir Ağları ile Görüntü Sınıflandırma**

Abdullah Akçam - g140910076@sakarya.edu.tr

**Tasarım Özeti**

Bu çalışmada görüntü dosyalarından oluşan veri setleri

ile evrişimsel sinir ağları vasıtasıyla kahve görüntülerinin kategorize

edilmesini sağlayan derin öğrenme modelinin, TensorFlow framework’ü

ve Keras ile eğitilmesi, Flask kullanılarak oluşturulan Web Api’den

bu modelin erişimini sağlamak ve bu API’ın kullanımı için Xamarin

platformunda geliştirilmiş örnek bir mobil uygulamayla tasarlanacaktır

DANIŞMAN: Öğr.Gör.Dr. CAN YÜZKOLLAR

İÇİNDEKİLER

[1. GİRİŞ 3](#_Toc61533016)

[1.1 GÖRÜNTÜ SINIFLANDIRMA 3](#_Toc61533017)

[1.1.1 EĞİTİMSİZ SINIFLANDIRMA 4](#_Toc61533018)

[1.1.2 EĞİTİMLİ SINIFLANDIRMA 4](#_Toc61533019)

[1.1 VERİ SETİNİN OLUŞTURULMASI 4](#_Toc61533020)

[2. KULLANILACAK TEKNOLOJİLER 6](#_Toc61533021)

[2.1 TENSORFLOW 6](#_Toc61533022)

[2.1.2 KERAS 6](#_Toc61533023)

[2.2 FLASK 7](#_Toc61533024)

[2.2.1 NEDEN FLASK? 7](#_Toc61533025)

[2.2.2 WEB FRAMEWORK 7](#_Toc61533026)

[2.2.3 FLASK’IN BU PROJEDEKI KULLANIMI 8](#_Toc61533027)

[2.3 XAMARIN 8](#_Toc61533028)

[2.3.1 NEDEN XAMARIN? 8](#_Toc61533029)

[3. UYGULAMANIN GELİŞTİRİLMESİ 9](#_Toc61533030)

[3.2 MODELİN KULLANILMASI 14](#_Toc61533031)

[3.3 API UYGULAMASI 14](#_Toc61533032)

[3.3 MOBİL UYGULAMA 17](#_Toc61533033)

[3.4 UI GELİŞTİRMESİ 22](#_Toc61533034)

[3.5 URL DEN RESİM EKLEME 23](#_Toc61533035)

[4. SONUÇ 25](#_Toc61533036)

[5. KAYNAKLAR 26](#_Toc61533037)

# GİRİŞ

Kahve resimlerinin sınıflandırılması için görüntü sınıflandırma kullanacağım. Görüntü sınıflandırması, dijital görüntü işleme alanında en önemli rollerden birine sahiptir. Bu projede görüntü sınıflandırma yöntemiyle kahveleri ayırt edecek bir yapay zeka oluşturdum.

## GÖRÜNTÜ SINIFLANDIRMA

Görüntü sınıflandırmasını, projede kullanacağım resim dosyalarından önceden tanımladığım kategorilere göre görüntülerin yapay sinir ağları sayesinde sınıflandırılmasını sağlayacağım. Görüntü sınıflandırmasından elde edilen bu model, örneğin bir kahve resmi çektiğinizde kahvenin hangi tür kahve olduğunu ayırt edebilir. Sınıflandırmayı iki şekilde yapabilirim, eğitimli (supervised) ve eğitimsiz (unsupervised) olaraktan iki çeşit yöntem var.

### EĞİTİMSİZ SINIFLANDIRMA

Bu projede eğitimli sınıflandırma yöntemini kullanacağım, Eğer önceden sınıfları belirlememiş olsaydım bu eğitimsiz sınıflandırma olurdu. Eğitimsiz sınıflandırmada bilgisayar hangi görüntülerin birbirleriyle bağlantılı olabileceğini saptayıp isimsiz olaraktan sınıflandırmaları yapar. İstersek verilerin kaç farklı kategoriye ayrılması gerektiğini önceden tanımlayabiliriz.

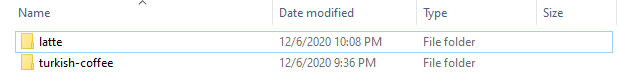
### EĞİTİMLİ SINIFLANDIRMA

Eğitimli sınıflandırma, modeli eğitecek kişi tarafından önceden belirlenmiş kategorilerdeki girdilerin kullanılarak modelin eğitilmesini sağlar. Bu projede, sınıflar önceden tanımlı olduğu için ve eğitimli araştırmanın tamamen bilgisayarın kendi kendine oluşturduğu resimler arası karmaşık bağlar olduğundan dolayı neye göre kategorilendireceği kesin olmamakla birlikte istediğim sonuçtan çok daha farklı alanlarda sınıflandırma yapabilirdi.

## VERİ SETİNİN OLUŞTURULMASI

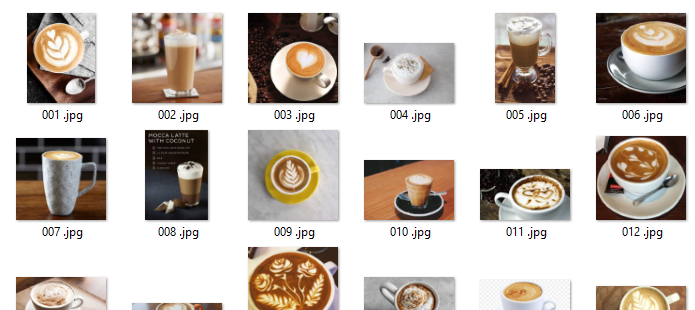
Bu çalışmada veri seti olarak kahve resimleri kullanıyorum. Amacım modele sunacağım kahvenin türk kahvesi mi yoksa latte kahve mi olduğunu ayırt edecek bir model oluşturmak.

Veri seti olarak kendim Google resimlerden aratarak derlediğim 75 adet türk kahvesi ve 75 adet latte olaraktan toplamda 150 adet resmi aşağıdaki gibi klasörlere kaydettim. Modelin düzgün çalışması için örneklerin dengeli sayıda olmasınin önemli olduğunu öğrendim.



Resim dosyalarının isimlerini düzenlemek için aşağıdaki kod parçasını yazdım:

1. **import** os
3. os.chdir('C://Users//Abdullah//Desktop//2020-2021//Bilg.tasarim//VSCode//dataset//latte')
4. **print**(os.getcwd())
5. COUNT = 1
7. # Dosyaların sayılara göre isim almasını sağlayan fonksiyon
8. **def** increment():
9. **global** COUNT
10. COUNT = COUNT + 1
12. **for** f **in** os.listdir():
13. f\_name, f\_ext = os.path.splitext(f)
14. f\_name = str(COUNT).zfill(3)
15. increment()
16. new\_name = '{} {}'.format(f\_name, f\_ext)
17. os.rename(f, new\_name)
19. os.chdir('C://Users//Abdullah//Desktop//2020-2021//Bilg.tasarim//VSCode//dataset//turkish-coffee')
20. COUNT = 0
22. **for** f **in** os.listdir():
23. f\_name, f\_ext = os.path.splitext(f)
24. f\_name = str(COUNT).zfill(3)
25. increment()
26. new\_name = '{} {}'.format(f\_name, f\_ext)
27. os.rename(f, new\_name)



*“latte” klasörü*

# KULLANILACAK TEKNOLOJİLER

## TENSORFLOW

Google tarafından geliştirilen derin öğrenme desteği sunan TensorFlow yapay zeka frameworkünü kullanacağım. Doğrudan TensorFlow’u kullanmak yerine, TensorFlow’un üzerine inşa edilmiş bir kütüphane olan Keras’ı kullanacağım.

### 2.1.2 KERAS

Keras, yüksek seviyeli bir kütüphane olarak

TensorFlow’un üzerine inşa edilmiştir.

Keras sayesinde derin öğrenme ve yapay zeka’nın çok derinliklerine inmeden, anlaşılır şekilde ve aynı zamanda TensorFlow’un bütün kabiliyetlerini kullanabiliyorum. Keras bu açıdan projede kolaylık sağlıyor.

## Flask Framework Nedir ve Kurulumu – Cahit İŞLEYENFLASK

Oluşturacağım yapay sinir ağını ulaşılabilir kılmak

İçin bir web uygulamasına entegre edeceğim.

Bunun için Flask kullanacağım. Eğer Microsoft

ASP.NET Core tabanlı bir web uygulaması kullanılmak istenirse şayet modeli yüklemek için önce ONNX tipine çevrilmesi gerekiyor.

Bkz.: https://stackoverflow.com/questions/55923311/load-model-with-ml-net-saved-with-keras

### NEDEN FLASK?

Flask bir python frameworkü olduğu için, uygulamayı da Python ile yazdığım için daha native bir çözüm gibi geliyor bana. Diğer framework’ler de kullanılabilir fakat bunlar zahmetli ve sıkıntılı olurdu.

### WEB FRAMEWORK

Oluşturacağım uygulamaya dışarıdan ulaşılması ve kullanımı sağlamak için bir web framework kullanmam gerekiyordu. Flask ile geliştirdiğim web uygulaması sayesinde içerine yüklediğim model ile dışarıdan bu modele ulaşılabilir olacak ve bu modeli isteyenler içtiği kahvenin ne olduğunu öğrenmek için kullanabilecekler. 😊

Bu proje için Flask’ın artıları

· Son derece esnek

· Öğrenilmesi ve kullanması kolay

· Yönlendirme URL’leri kolay

Ve tabi her şeyden önce Python’nun yorumlanabilir bir dil olması yani derlenmeye ihtiyaç duymaması işlemlerin çok daha kısa sürmesini sağlamakta.

### FLASK’IN BU PROJEDEKI KULLANIMI

Flask’ı oluşturacağım Web uygulamasının REST API’sini geliştirmek için kullanacağım.

## Xamarin - Vikipedi XAMARIN

Oluşturulan modelin, tasarladığım

API vasıtasıyla kullanılmasını birçok yöntemle

sağlanabilir. Bu projede mobil cross-platform uygulamalar geliştirebileceğimiz Xamarin’i kullanacağım.

## NEDEN XAMARIN?

Xamar kullanmamın iki sebebi var:

* Cross-platform olması
* En iyi bildiğim dilin C#, ve platformun .NET olması.

# UYGULAMANIN GELİŞTİRİLMESİ

* 1. VERİSETİNİN DERLENMESİ

Adımların teker teker izlenebilmesi için editör olarak numpy kullandım. Veriseti için ilk resmin ve kütüphanelerin eklenmesi için kod bloğu:

**from tensorflow.keras.preprocessing.image import ImageDataGenerator**

**from tensorflow.keras.preprocessing import image**

**from tensorflow.keras.optimizers import RMSprop**

**import matplotlib.pyplot as plt**

**import tensorflow as tf**

**import numpy as np**

**import cv2**

**import os**

**img = image.load\_img('dataset/train/latte/001 .jpg')**

**plt.imshow(img)**

şimdilik ilk resmin yüklenmesi kodun çalışıp çalışmadığını görmek için yeterli

<matplotlib.image.AxesImage at 0x22e15202470>



Resim dosyasını gray moda alarak iki boyutlu matrislerden oluşmasını sağladım. Eğer renki resimler kullansaydım 3 boyutlu matrisler olacaktı ve bu da modelin eğitilmesi açısından zorluk çıkartacaktır. Bunun dışında bu proje için renklerden ziyade renklerin kontrastı daha önemli olduğu için renklerin kullanılmasına gerek yoktur.

1. print(img\_array)
2. [[233 232 232 ... 220 220 220]  
    [233 232 232 ... 220 220 220]  
    [233 232 232 ... 220 220 220]  
    ...  
    [214 214 215 ... 172 172 171]  
    [215 215 216 ... 205 204 204]  
    [216 216 217 ... 206 205 205]]

Resmin boyutları:

1. cv2.imread('dataset/train/latte/001 .jpg').shape
2. (375, 500)

Görüldüğü üzere resim boyutları standart değil, bu modelin eğitilmesi açısından uygun değildi, resimleri uygun ve standart boyutlara getiriyorum:

train = ImageDataGenerator(rescale=1/255)

validation = ImageDataGenerator(rescale=1/255)

Dataseti yüklüyorum:

train\_dataset = train.flow\_from\_directory('dataset/train', target\_size=(200, 200), batch\_size=3, class\_mode='binary')

validation\_dataset = validation.flow\_from\_directory('dataset/validation', target\_size=(200, 200), batch\_size=3, class\_mode='binary')

Found 150 images belonging to 2 classes.

Found 66 images belonging to 2 classes.

Verisetinin kullandığı dosya isimlerine göre sınıflara ayrıldığını görebiliyorum:

train\_dataset.class\_indices

{'latte': 0, 'turkish-coffee': 1}

Modelin konfigrasyonunu yapyorum, aktivasyon metodunu ve pool çeşidini belirtiyorum:

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input\_shape=(200, 200, 3)),

tf.keras.layers.MaxPool2D(2, 2),

tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),

tf.keras.layers.MaxPool2D(2, 2),

tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),

tf.keras.layers.MaxPool2D(2, 2),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(512, activation='relu'),

tf.keras.layers.Dense(1, activation='sigmoid')

])

Veri setini 30 epoch ile oluşturuyorum:

100%|██████████| 75/75 [00:00<00:00, 108.63it/s]  
100%|██████████| 75/75 [00:01<00:00, 45.79it/s]   
  
150

Modeli derliyorum:

model.compile(loss='binary\_crossentropy', optimizer=RMSprop(lr=0.001), metrics=['accuracy'])

Modeli oluşturuyorum:

*model\_fit = model.fit(train\_dataset, steps\_per\_epoch=3, epochs=30, validation\_data=validation\_dataset)*

Epoch 1/30

3/3 [==============================] - 3s 1s/step - loss: 12.6837 - accuracy: 0.6667 - val\_loss: 2.4246 - val\_accuracy: 0.4545

Epoch 2/30

3/3 [==============================] - 2s 805ms/step - loss: 1.6979 - accuracy: 0.4444 - val\_loss: 1.1268 - val\_accuracy: 0.4545

Epoch 3/30

3/3 [==============================] - 2s 745ms/step - loss: 0.7960 - accuracy: 0.5556 - val\_loss: 1.2738 - val\_accuracy: 0.4545

Epoch 4/30

3/3 [==============================] - 3s 871ms/step - loss: 0.7901 - accuracy: 0.5556 - val\_loss: 0.7081 - val\_accuracy: 0.5455

Epoch 5/30

3/3 [==============================] - 3s 842ms/step - loss: 1.3937 - accuracy: 0.6667 - val\_loss: 0.6515 - val\_accuracy: 0.5455

Epoch 6/30

3/3 [==============================] - 2s 778ms/step - loss: 0.6095 - accuracy: 0.6667 - val\_loss: 0.6492 - val\_accuracy: 0.5758

Epoch 7/30

3/3 [==============================] - 2s 716ms/step - loss: 0.7239 - accuracy: 0.6667 - val\_loss: 0.5901 - val\_accuracy: 0.7424

Epoch 8/30

3/3 [==============================] - 2s 707ms/step - loss: 0.6903 - accuracy: 0.5556 - val\_loss: 0.5814 - val\_accuracy: 0.6970

Epoch 9/30

3/3 [==============================] - 2s 736ms/step - loss: 0.7290 - accuracy: 0.4444 - val\_loss: 0.6478 - val\_accuracy: 0.5455

Epoch 10/30

3/3 [==============================] - 2s 730ms/step - loss: 0.5562 - accuracy: 0.8889 - val\_loss: 0.5325 - val\_accuracy: 0.8030

Epoch 11/30

3/3 [==============================] - 2s 747ms/step - loss: 1.6003 - accuracy: 0.5556 - val\_loss: 0.5725 - val\_accuracy: 0.7727

Epoch 12/30

3/3 [==============================] - 2s 691ms/step - loss: 0.8722 - accuracy: 0.5556 - val\_loss: 0.6579 - val\_accuracy: 0.7576

Epoch 13/30

3/3 [==============================] - 2s 755ms/step - loss: 0.6279 - accuracy: 0.6667 - val\_loss: 0.6113 - val\_accuracy: 0.7879

Epoch 14/30

3/3 [==============================] - 2s 653ms/step - loss: 1.2230 - accuracy: 0.8889 - val\_loss: 0.5892 - val\_accuracy: 0.7273

Epoch 15/30

3/3 [==============================] - 2s 746ms/step - loss: 1.6352 - accuracy: 0.4444 - val\_loss: 0.6043 - val\_accuracy: 0.8485

Epoch 16/30

3/3 [==============================] - 2s 813ms/step - loss: 0.5991 - accuracy: 0.6667 - val\_loss: 0.6026 - val\_accuracy: 0.5758

Epoch 17/30

3/3 [==============================] - 3s 987ms/step - loss: 0.7821 - accuracy: 0.5556 - val\_loss: 0.5902 - val\_accuracy: 0.8636

Epoch 18/30

3/3 [==============================] - 3s 917ms/step - loss: 0.5634 - accuracy: 0.5556 - val\_loss: 0.5370 - val\_accuracy: 0.5909

Epoch 19/30

3/3 [==============================] - 2s 687ms/step - loss: 0.6239 - accuracy: 0.7778 - val\_loss: 1.7915 - val\_accuracy: 0.4545

Epoch 20/30

3/3 [==============================] - 2s 753ms/step - loss: 1.1902 - accuracy: 0.4444 - val\_loss: 0.6014 - val\_accuracy: 0.8333

Epoch 21/30

3/3 [==============================] - 2s 711ms/step - loss: 0.7898 - accuracy: 0.4444 - val\_loss: 0.5919 - val\_accuracy: 0.8788

Epoch 22/30

3/3 [==============================] - 2s 726ms/step - loss: 2.0487 - accuracy: 0.7778 - val\_loss: 0.5859 - val\_accuracy: 0.9242

Epoch 23/30

3/3 [==============================] - 2s 662ms/step - loss: 0.6308 - accuracy: 0.5556 - val\_loss: 0.5437 - val\_accuracy: 0.6515

Epoch 24/30

3/3 [==============================] - 2s 699ms/step - loss: 1.5346 - accuracy: 0.6667 - val\_loss: 0.6040 - val\_accuracy: 0.8788

Epoch 25/30

3/3 [==============================] - 2s 708ms/step - loss: 0.5507 - accuracy: 0.8889 - val\_loss: 0.5711 - val\_accuracy: 0.6364

Epoch 26/30

3/3 [==============================] - 2s 814ms/step - loss: 0.5957 - accuracy: 0.7778 - val\_loss: 0.5247 - val\_accuracy: 0.7727

Epoch 27/30

3/3 [==============================] - 3s 1s/step - loss: 0.8611 - accuracy: 0.4444 - val\_loss: 0.6161 - val\_accuracy: 0.5758

Epoch 28/30

3/3 [==============================] - 3s 1s/step - loss: 0.6237 - accuracy: 0.5556 - val\_loss: 0.5965 - val\_accuracy: 0.6212

Epoch 29/30

Ve son olarakta modeli 64x3 olarak kaydediyorum:

model.save('64x3-CNN-01.h5', save\_format='h5')

## 3.2 MODELİN KULLANILMASI

Modeli Python ortamında doğrudan kullanabiliriz veya bir sunucuya yükleyerek oradan modele ulaşıp kullanabiliriz. Bu projede modeli Flask’da oluşturacağım sunucuda kullanacağımdan bahsetmiştim.

Flask uuygulamasında modeli yüklüyorum:

def get\_model():

    global model

    model = load\_model('64x3-CNN-01.h5', compile=False)

    print(" \* Model loaded!")

## 3.3 API UYGULAMASI

API için Flask’da oluşturduğum web servere modeli yükledim.

import base64

import numpy as np

import tensorflow as tf

import io

from PIL import Image

from tensorflow import keras

from keras import backend as K

from keras.models import Sequential

from keras.models import load\_model

from keras.preprocessing.image import ImageDataGenerator

from keras.preprocessing.image import img\_to\_array

from flask import request

from flask import jsonify

from flask import Flask

app = Flask(\_\_name\_\_)

CATEGORIES = ["Latte", "Türk Kahvesi"]

def get\_model():

    global model

    model = load\_model('64x3-CNN-01.h5', compile=False)

    print(" \* Model loaded!")

def preprocess\_image(image, target\_size):

    if image.mode != "RGB":

        image = image.convert("RGB")

    image = image.resize(target\_size)

    image = img\_to\_array(image)

    image = np.expand\_dims(image, axis=0)

    return image

print(" \* Loading Keras model...")

# modelin yuklenmesi icin gerekli fonksiyonu cagiriyor

get\_model()

# CONNECTION REFUSED hatasına çözüm olarak headerlere eklediğim Access-Control başlıkları

@app.after\_request

def after\_request(response):

  response.headers.set('Access-Control-Allow-Origin', '\*')

  response.headers.set('Access-Control-Allow-Headers', 'Content-Type,Authorization')

  response.headers.set('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE,OPTIONS')

  return response

# POST olarak gonderilen resmin işlenip sunucu tarafından alınması ve burada ki değerlendirme süreci

@app.route("/predict", methods=["POST"])

def predict():

    response = {"success": False}

    if request.method == "POST":

        if request.files.get("image"):

            try:

                image = request.files["image"].read()

                image = Image.open(io.BytesIO(image))

                processed\_image = preprocess\_image(image, target\_size=(200, 200))

                prediction = model.predict(processed\_image).tolist()

            except Exception as ex:

                # hataları burada değerlendirebiliriz

                print("===== ERROR =====")

                print(ex.with\_traceback)

                response = {

                    'error': 'true'

                }

                return jsonify(response)

            response = {

                'prediction': CATEGORIES[int(prediction[0][0])]

            }

            print('predicted: ' + CATEGORIES[int(prediction[0][0])])

        else:

            print('Wrong file')

    else:

        print('Wrong method')

    return jsonify(response)

                image = Image.open(io.BytesIO(image))

                processed\_image = preprocess\_image(image, target\_size=(200, 200))

                prediction = model.predict(processed\_image).tolist()

            except Exception as ex:

                # handle exception here

                print("===== ERROR =====")

                print(ex.with\_traceback)

                response = {

                    'error': 'true'

                }

                return jsonify(response)

            response = {

                'prediction': CATEGORIES[int(prediction[0][0])]

            }

            print('predicted: ' + CATEGORIES[int(prediction[0][0])])

        else:

            print('Wrong file')

    else:

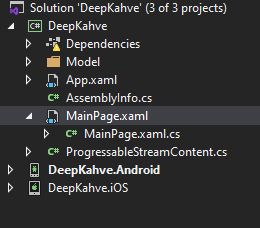
        print('Wrong method')

    return jsonify(response)

## 3.3 MOBİL UYGULAMA

Mobil uygulamın adı Deep Kahve. Deep Kahve ile galeriden seçeceğiniz resimleri veya kameradan çekeceğiniz resimeri Flask uygulamasındaki DNN sunucusuna gönderip orada işlenip resmin ne olduğunu öğrendikten sonra uygulamanın alt tarafında sonucu yazan bir metin çıkacaktır.

Projenin yapısı:



Elimde iOS telefon olmadığı için uygulamanın iOS versiyonunu test edemedim.

Aşağıdaki dosya uygulamanın layoutunu göstermektedir.

MainPage.xaml:

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="DeepKahve.MainPage">

<ContentPage.Resources>

<ResourceDictionary>

<Style x:Key="labelStyle" TargetType="Label">

<Setter Property="HorizontalOptions"

Value="Center" />

<Setter Property="VerticalOptions"

Value="CenterAndExpand" />

<Setter Property="FontSize" Value="Large" />

<Setter Property="TextColor" Value="Black" />

</Style>

</ResourceDictionary>

</ContentPage.Resources>

<StackLayout>

<Frame BackgroundColor="Brown" Padding="14" CornerRadius="0">

<Label Text="Deep Kahve!" HorizontalTextAlignment="Center" TextColor="White" FontSize="36"/>

</Frame>

<Label Text="Kahveniz Latte mi, yoksa Türk kahvesi mi?" FontSize="18" Padding="30,10,30,10"/>

<Label Text="Öğrenmek için kameranızdan bir resim çekin veya var olan bir kahve resmini yükleyin!" FontSize="16" Padding="30,0,30,0"/>

<Image

x:Name="imageFile"

Aspect="Fill"

Source="hotcup.png"

HeightRequest="{OnPlatform iOS=300, Android=250}"

WidthRequest="{OnPlatform iOS=300, Android=250}"

HorizontalOptions="Center" />

<StackLayout

Orientation="Horizontal"

HorizontalOptions="Center">

<Button Text="Galeriden Resim Seç"

VerticalOptions="CenterAndExpand"

HorizontalOptions="Center"

Clicked="OnButtonGalleryClicked" />

<Button Text="Kameradan Resim Çek"

VerticalOptions="CenterAndExpand"

HorizontalOptions="Center"

Clicked="OnButtonCameraClicked" />

</StackLayout>

<Button Text="Hesapla"

VerticalOptions="CenterAndExpand"

HorizontalOptions="Center"

Clicked="OnButtonSubmitClicked"

Style="{StaticResource buttonStyle}"/>

<Label FontSize="22" Padding="15,10,30,30"

HorizontalTextAlignment="Center"

x:Name="sonucLabel"

Style="{StaticResource labelStyle}">

<Label.FormattedText>

<FormattedString>

<FormattedString.Spans>

<Span Text="SONUÇ" FontAttributes="Bold"/>

</FormattedString.Spans>

</FormattedString>

</Label.FormattedText>

</Label>

</StackLayout>

</ContentPage>

MainPage.xaml.cs dosyası ise aşağıdaki gibidir. Burada uygulamadan resmi seçip veya kameradan çekilen resmin sunucuya gönderilmesini yarayan kodlar var:

using Android;

using Android.App;

using Android.Graphics;

using Android.Widget;

using DeepKahve.Model;

using Newtonsoft.Json;

using Plugin.Media;

using Plugin.Media.Abstractions;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

namespace DeepKahve

{

public partial class MainPage : ContentPage

{

MediaFile file;

public MainPage()

{

InitializeComponent();

}

private void OnButtonGalleryClicked(object sender, EventArgs args)

{

UploadPhoto();

}

private void OnButtonCameraClicked(object sender, EventArgs args)

{

TakePhoto();

}

private void OnButtonSubmitClicked(object sender, EventArgs args)

{

if (file == null)

{

sonucLabel.TextColor = Xamarin.Forms.Color.Red;

sonucLabel.Text = "RESİM SEÇİN";

sonucLabel.TextColor = Xamarin.Forms.Color.Black;

return;

}

Upload(file.Path);

}

async void TakePhoto()

{

await CrossMedia.Current.Initialize();

file = await CrossMedia.Current.TakePhotoAsync(new Plugin.Media.Abstractions.StoreCameraMediaOptions

{

PhotoSize = Plugin.Media.Abstractions.PhotoSize.Medium,

CompressionQuality = 40,

Name = "001.jpg",

Directory = "sample"

});

if (file == null) return;

//byte[] imageArray = File.ReadAllBytes(file.Path);

//var bitmap = BitmapFactory.DecodeByteArray(imageArray, 0, imageArray.Length);

imageFile.Source = ImageSource.FromFile(file.Path);

}

async void UploadPhoto()

{

await CrossMedia.Current.Initialize();

if (!CrossMedia.Current.IsPickPhotoSupported)

{

return;

}

file = await CrossMedia.Current.PickPhotoAsync(new PickMediaOptions

{

PhotoSize = PhotoSize.Full,

CompressionQuality = 40

});

imageFile.Source = ImageSource.FromFile(file.Path);

}

async void Upload(string path)

{

var image = File.ReadAllBytes(path);

var fileName = System.IO.Path.GetFileName(path);

var stream = new MemoryStream(image);

var streamContent = new StreamContent(stream);

using var client = new HttpClient();

using var multiForm = new MultipartFormDataContent();

client.Timeout = TimeSpan.FromSeconds(30);

//var file = new ProgressableStreamContent(streamContent, (sent, total) =>

//{

// sonucLabel.Text = "\bYükleniyor: " + ((float)sent / total) \* 10f;

//});

var file = new StreamContent(stream);

multiForm.Add(file, "image", fileName);

// 169.254.80.80 Microsoft emulator icin localhost gateway IP si

// 10.0.2.2 Android emulator icin IP

var response = await client.PostAsync("http://10.0.2.2:5000/predict", multiForm);

if (response.StatusCode == HttpStatusCode.OK)

{

string jsonRes = await response.Content.ReadAsStringAsync();

var settings = new JsonSerializerSettings()

{

Culture = new System.Globalization.CultureInfo("tr-TR")

};

var res = JsonConvert.DeserializeObject<Prediction>(jsonRes, settings);

sonucLabel.Text = res.Text;

}

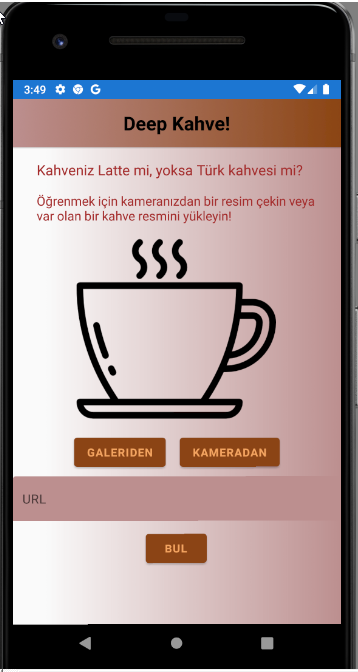
}

}

}

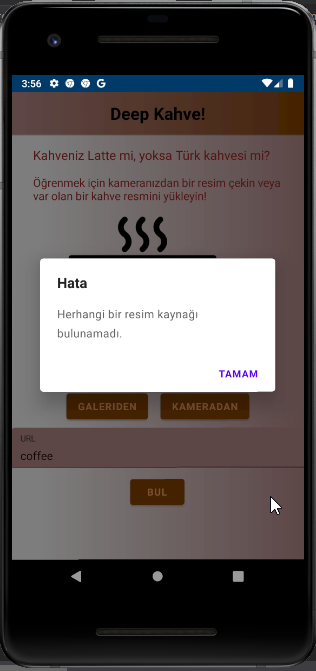
## 3.4 UI GELİŞTİRMESİ

UI için XF. Material kütüphanesini kullandım.



## 3.5 URL DEN RESİM EKLEME

Kullanıcıların URL den resim eklemesi için URL girilebilecek bir textbox oluşturdum. Bu textbox’da eğer herhangi bir yazı varsa öncelik olarak oradan resmi yüklemeye çalışır. Eğer URL de düzgün formatlanmış bir URL yoksa hata mesajı verir.



# 4. SONUÇ

Serveri çalıştırmak için Flask App klasöründe

set FLASK\_APP=predict\_app.py

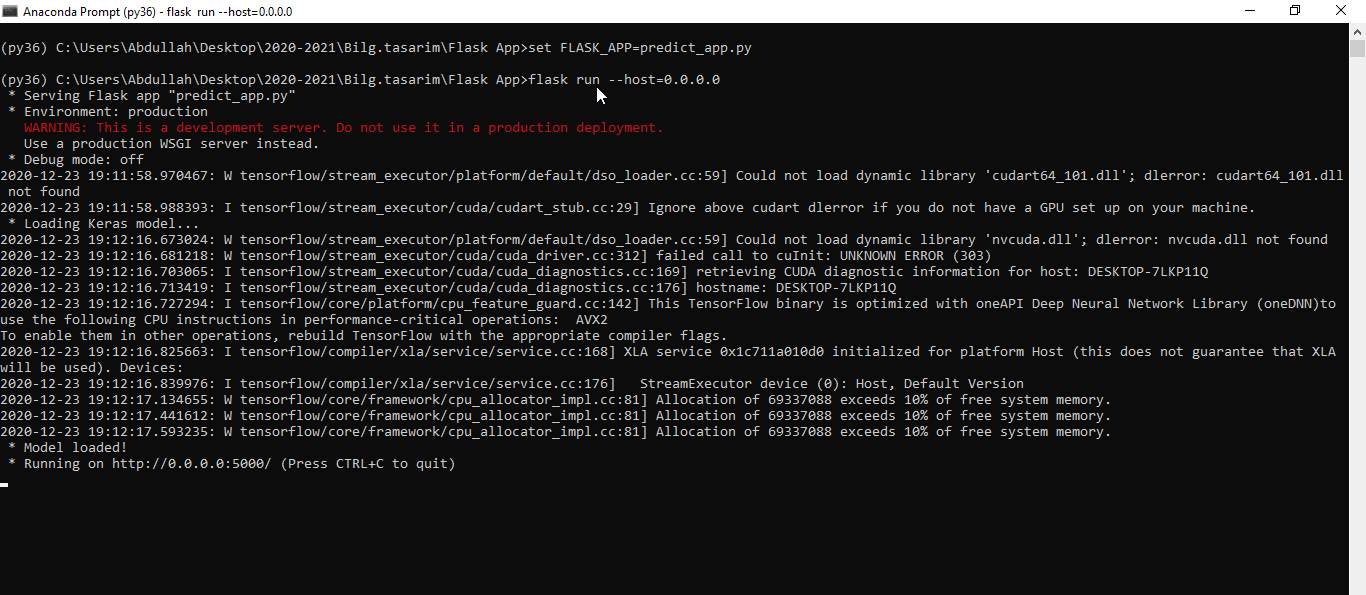
set FLASK\_ENV=development

flask run --host=0.0.0.0

satırları çalıştırılır.

FLASK\_ENV development modda çalıştırır, .py dosyasında değişiklik olduğu zaman otomatik olarak sunucuyu baştan başlatır.

host belirtilmesinin sebebi local wifi ağında bütün cihazlara açık olarak sunucunun erişilmesini sağlar. (Ethernet üzerinden çalışmıyor)



Galeriden bir kahve resmi seçtim ve BUL butonuna bastım. Sonucu Türk Kahvesi olarak doğru bulmuş oldu.



# 5. KAYNAKLAR

* <https://keras.io/api/>
* <https://www.investopedia.com/terms/n/neuralnetwork.asp#:~:text=A%20neural%20network%20is%20a,organic%20or%20artificial%20in%20nature>.
* <https://www.tensorflow.org/tutorials>
* <https://github.com/Baseflow/XF-Material-Library>
* <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/>