TRAKYA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM411 PROJE I

PROJE RAPORU

PROJE ADI:

CLOTHES SCANNER

PROJE EKİBİ:

1221602807 – AHMET TOPUZ

1211602626 – TOUGKAI TASIM

PROJE DANIŞMANI:

Dr. Öğr. Üyesi REMBİYE KANDEMİR

Edirne - 2024

İÇİNDEKİLER

1. ÖZ VE GİRİŞ

Görme engelli bireylerin bağımsız bir şekilde kıyafet seçimi yapabilmelerini kolaylaştırmak amacıyla geliştirilen bu proje, kıyafetlerin desenlerini ve renklerini tanıyan, bu bilgileri sesli olarak kullanıcıya ileten bir mobil uygulamanın tasarım ve geliştirme sürecini kapsamaktadır. Uygulama, kullanıcıdan alınan bir kıyafet fotoğrafını işleyerek, desen türü (checkered, solid, dotted, striped, floral, zig-zag) ve renk bilgilerini (ortalama renk ve baskın renk) tespit eder. Sonuçlar, kullanıcıya sesli olarak aktarılır, böylece görme engelli bireylerin günlük hayatlarındaki bir ihtiyaç karşılanmış olur.

Geliştirme sürecinde, TensorFlow framework'ü kullanılarak bir Convolutional Neural Network (CNN) modeli eğitilmiştir. Modelin eğitimi için kullanılan veri kümesi, Alexander J. Medeiros ve ekibi tarafından geliştirilen, "Recognizing Clothing Colors and Visual Textures Using a Finger-Mounted Camera" [1] başlıklı çalışmaya dayanmaktadır. Bu dataset, kıyafet desenlerini tanımak için uygun görseller içermektedir. Bununla birlikte, modelin başarımını artırmak için verilerin %90'ı Google üzerinden elde edilen ek görsellerle genişletilmiştir. Toplam veri kümesi, görsellerin %10'u orijinal çalışmanın alt kümesinden, %90'ı ise farklı kaynaklardan toplanan ve proje kapsamında işlenen görsellerden oluşmaktadır. Veri kümesi, kıyafetlerin desen çeşitliliğini ve renk özelliklerini temsil etmek üzere düzenlenmiş, şeffaflık kaldırma, normalizasyon ve veri artırma işlemleriyle işlenmiştir.

Mobil uygulama iki ana bileşenden oluşmaktadır: React Native ile geliştirilmiş bir frontend ve Flask framework'ü üzerine kurulu bir backend. Frontend kısmında, expo-camera kütüphanesiyle kullanıcıdan fotoğraf alma, expo-media-library ile kaydedilen fotoğrafları yönetme ve expo-speech ile kullanıcıya sesli geri bildirim sağlama işlemleri gerçekleştirilmiştir. Kullanıcı deneyimini artırmak için react-native-animatable ile animasyonlar eklenmiş, uygulamanın genel erişilebilirliği göz önünde bulundurulmuştur. Backend tarafında, TensorFlow Lite (TFLite) modeli ile desen tahmini yapılmış, OpenCV kütüphanesi kullanılarak dominant ve ortalama renkler hesaplanmış, sonuçlar Flask API aracılığıyla frontend'e JSON formatında iletilmiştir.

Proje süresince, uygulamanın çeşitli aşamaları aşağıdaki yöntemlerle gerçekleştirilmiştir:

Veri İşleme: Veri setindeki görseller, şeffaflık içeriyorsa kaldırılmış ve RGBA formatından RGB formatına dönüştürülmüştür. Görseller, TensorFlow modeliyle uyumlu hale getirilmek için (128x128) çözünürlüğe yeniden boyutlandırılmıştır. Ayrıca, veri çeşitliliğini artırmak amacıyla döndürme, yakınlaştırma, çevirme, kesme ve ışık değişimi gibi veri artırma işlemleri uygulanmıştır.

Kullanılan kütüphaneler: Pillow (PIL), OpenCV, NumPy.

Makine Öğrenmesi Modeli: Projede bir Convolutional Neural Network (CNN) modeli kullanılmıştır. Model, desen tahmini için sırasıyla üç adet konvolüsyon katmanı, maksimum havuzlama katmanları ve yoğun (fully connected) katmanlardan oluşmuştur. Model, categorical_crossentropy kayıp fonksiyonu ve Adam optimizasyon algoritması ile eğitilmiş, erken durdurma (early stopping) ve öğrenme oranı azaltma (ReduceLROnPlateau) gibi tekniklerle optimize edilmiştir.

Kullanılan kütüphaneler: TensorFlow, TensorFlow Lite.

Dominant ve Ortalama Renk Hesaplama: Görseldeki baskın renk, K-Means kümeleme algoritması kullanılarak tespit edilmiştir. Ortalama renk ise görselin belirli bir bölgesindeki piksellerin ortalaması alınarak hesaplanmıştır. Bu bilgiler, ColorNamer kütüphanesi kullanılarak renk isimlerine dönüştürülmüştür.

Kullanılan kütüphaneler: OpenCV, NumPy, ColorNamer.

Mobil Geliştirme: React Native tabanlı mobil uygulama, kullanıcıdan gelen fotoğrafı backend'e göndererek işlem sonuçlarını sesli olarak kullanıcıya sunmuştur. Kamera, medya kütüphanesi ve mikrofon gibi cihaz özelliklerine erişim sağlamak için Expo platformu tercih edilmiştir. Backend ile iletişim için axios kütüphanesi kullanılmıştır.

Kullanılan kütüphaneler: React Native, Expo (expo-camera, expo-speech, expo-media-library), Axios.

Sonuçların Sesli Aktarımı: Kullanıcıya, elde edilen sonuçlar Türkçeye çevrilmiş ve expo-speech kütüphanesi ile sesli olarak iletilmiştir. Desenler ve renk isimleri, projede geliştirilen bir çeviri sözlüğü (tr_dict) yardımıyla Türkçe karşılıklarıyla ifade edilmiştir.

Bu projenin teorik altyapısı ve uygulama yöntemleri, aşağıdaki bilimsel kaynaklar ışığında geliştirilmiştir:

Clothing Style Recognition Method Based on Digital Image Processing: Bu çalışma, çok çözünürlüklü bir çerçeve kullanarak, her çözünürlük seviyesindeki görüntülerin kenarlarını tespit etmek için kenar algılama algoritmaları kullanır ve bunları dar kenar bantlarına genişletir. Elde edilen kenar bantları, nokta dağılımı modeliyle birleştirilerek çok çözünürlüklü görüntü arama işlemi gerçekleştirilir. Bu yöntem, görüntü işleme teknikleriyle kıyafet stillerinin tanınmasında etkili bir yaklaşım sunmaktadır [2].

Cloth Pattern Recognition Using Machine Learning and Neural Network: Bu makalede, K-en yakın komşu (KNN), Destek Vektör Makineleri (SVM) ve derin öğrenme ağları gibi makine öğrenimi algoritmaları kullanılarak kıyafet desenlerinin tanınması ele alınmıştır. Renk sınıflandırması için Hue Saturation Intensity (HSI) renk modeli kullanılmıştır. Kıyafet desenlerini tanımak için global ve lokal özellikler çıkarılmış ve bu özellikler, AlexNet, GoogleNet, VGG-16 ve VGG-19 gibi derin öğrenme ağlarıyla işlenmiştir. Bu çalışma, makine öğrenimi ve sinir ağlarıyla kıyafet desenlerinin tanınmasında kapsamlı bir yaklaşım sunmaktadır [3].

Convenient Clothing Pattern and Colour Recognition for People with Vision Impairment: Bu araştırma, görme engelli bireyler için kıyafet desenlerini dört kategoriye ayırarak (plaid, plain, irregular, striped) ve 12 rengi tanıyan bir prototip sistem tasarlamıştır. Sistem, entegre bir kamera, mikrofon, bilgisayar ve Bluetooth kulaklık kullanarak sesli açıklamalarla kullanıcıya geri bildirim sağlamaktadır. Bu çalışma, görme engelli bireylerin kıyafet seçiminde yardımcı olmak için sesli geri bildirim sağlayan bir sistem geliştirmektedir. [4]

Bu kaynaklar, projenin temelini oluşturan teorik ve uygulamalı çalışmaları temsil etmektedir.

2. GEREKSİNİM ANALİZİ

Proje, görme engelli ve renk körü bireylerin günlük hayatta karşılaştıkları bir sorunu çözmeyi amaçlamaktadır. Kıyafetlerin renklerini ve desenlerini tanımlamakta zorluk çeken bireyler için, bu uygulama önemli bir yardım aracıdır. Kullanıcılar, fotoğraf çekerek kıyafetlerinin desenini ve rengini öğrenebilir ve böylece daha bağımsız bir şekilde kıyafet seçimi yapabilirler. Mevcut örneklerden farklı olarak, uygulamanın temel avantajı, hızlı ve amaca yönelik çalışmasıdır. Görme engelliler için erişilebilirliği ön planda tutarak, basit ve anlaşılır bir kullanıcı arayüzü sunulmuştur. Bu, kullanıcının hızlı bir şekilde sonuç alabilmesini sağlar. Uygulamanın ayrıca, görsel algılama ve sesli geri bildirim sunma konusunda daha düşük gecikme süresiyle çalışması, mevcut çözümlerden daha verimli hale gelmesini sağlamaktadır.

Proje, görme engelli ve renk körü bireylerin kıyafet seçimlerinde bağımsızlıklarını artırmayı hedeflemektedir. Bu doğrultuda geliştirilen mobil uygulama, kullanıcıların görsel algılamalarını destekleyecek şekilde, kıyafetlerin desen ve renk bilgilerini analiz eder. Uygulama, Android işletim sistemi üzerinde çalışacak şekilde tasarlanmıştır ve iOS platformuna kolayca uyarlanabilir. Böylece, her iki platformda da kullanılabilirlik sağlanmış olur. Backend tarafında ise Flask framework'ü üzerinden sunucuyla iletişim sağlanarak, görsellerin işlenmesi ve analizi yapılmaktadır. Flask, bu proje için ideal bir seçim oldu, çünkü hızlı geliştirme, esneklik ve Python ile entegrasyon kolaylığı sunmaktadır.

Hedef kitlesi, görme engelliler ve renk körü bireylerden oluşmaktadır. Bu grup, genellikle kıyafetlerin desen ve renklerini tanımlamakta zorlanır ve bu uygulama onlara bu konuda yardımcı olacaktır. Uygulama, kullanımı kolay bir arayüzle tasarlandığı için her yaştan ve teknik bilgi seviyesinden kullanıcıya hitap etmektedir. Ayrıca, kullanıcı sayısının arttıkça uygulamanın farklı ihtiyaçlara göre uyarlanabilmesi de hedeflenmektedir. Kullanıcılar, uygulamayı kendi ihtiyaçlarına göre özelleştirebilirler.

Proje, Python programlama diliyle geliştirilmiştir ve TensorFlow ile makine öğrenmesi modeli eğitilmiştir. Python, güçlü kütüphaneleri ve esnek yapısı nedeniyle bu tür projeler için ideal bir dildir. TensorFlow, derin öğrenme ve makine öğrenmesi için güçlü bir araçtır ve bu projede kıyafet desenlerini doğru şekilde tanıyabilmek için kullanılmıştır [5]. Ayrıca, mobil uygulama geliştirme kısmında React Native kullanılmıştır. React Native [6], hem Android hem de iOS platformları için uygulama geliştirmeyi mümkün kılarken, hızlı prototipleme ve geliştirici verimliliği sağlar. Flask framework'ü, sunucu tarafında hızlı ve verimli bir çözüm sunar ve mobil uygulama ile uyumlu bir şekilde çalışır[7]. Kullanıcı arayüzü ve sesli geri bildirim sağlamak için Expo kütüphaneleri kullanılmıştır. Expo, mobil uygulama geliştirmeyi hızlandırır ve erişilebilirlik konusunda önemli avantajlar sunar [8].

Sunucu tarafında Render.com platformu kullanılarak, görsel işleme ve analiz işlemleri yapılmaktadır. Render.com, projenin hızlı bir şekilde çalışabilmesi için gerekli altyapıyı sağlar ve sunucu tarafında yüksek verimlilik sunar [9]. Projenin backend kısmında Flask ve TensorFlow Lite kullanılarak kıyafet desen ve renk tanımlama işlemleri gerçekleştirilir. Görsellerin işlenmesi sırasında, önceden toplanan veri seti kullanılarak, Flask API aracılığıyla mobil uygulamaya gönderilen sonuçlar, sesli geri bildirim olarak kullanıcılara iletilir.

Projenin geliştirilmesinde karşılaşılan kaynak kısıtları arasında, başlangıçta kullanılan veri kümesinin sınırlı olması yer almaktadır. Veri kümesinin %10'luk kısmı bir veri kümesinden alınmıştır. Bu veri kümesi kısıtlı görseller içermekte ve bu görseller de Şekil 2.1'de göründüğü gibi var olan görsellerin çeşitli açılarda yeniden fotoğraflandırılmasıyla oluşturulmuş görsellerden oluşup, geriye kalan %90'lık kısmı Google üzerinden görseller şeklinde toplanmıştır. Bu sınırlı veri kümesi, daha fazla görsel çeşitliliği sağlamak amacıyla ön işleme teknikleriyle çoğaltılmıştır. Verinin sınırlı olması, projenin başarısını etkilemiş olsa da, verinin çeşitlendirilmesi ve çoğaltılması, bu kısıtlamayı aşmak için etkili bir çözüm olmuştur. Ayrıca, sunucu tarafında kullanılan Render.com platformu, uygulamanın hızlı ve güvenilir bir şekilde çalışmasını sağlamak için tercih edilmiştir. Render.com, ölçeklenebilirlik ve düşük gecikme süresi ile verimli bir çözüm sunmaktadır. Bu altyapı sayesinde, kullanıcı taleplerine hızlı bir şekilde yanıt verilebilmektedir. Projeyi geliştirirken gerek model eğitiminde, gerek kod entegrasyonunda ve gerekse de çözülmeyi bekleyen birçok sorunda da ChatGPT adlı yapay zekanın engin bilgi havuzundan da faydalanılmıştır.

Şekil 2.1. Veri setindeki görüntülerin benzerliği

Bu metodoloji ve araçlar, projenin başarısını artırmak ve hedef kitlesine en iyi şekilde hizmet verebilmek için seçilmiştir.

3. TASARIM

Bu bölümde, projenin genel yapısı, işleyişi, kullanıcı arayüzü ve kullanılan algoritmaların süreçleri detaylandırılmıştır. Ayrıca uygulamanın ekranları ve mimarisi açıklanmış, görsellerin ve şemalarla detaylandırılmıştır.

3.1 Proje Mimarisi

Proje, iki ana bileşenden oluşmaktadır: frontend (React Native) ve backend (Flask). Mobil cihaz üzerinde çalışan frontend, kullanıcıdan fotoğraf alır ve sonuçları sesli şekilde iletir. Backend ise görselleri işleyip analiz ederek sonuçları frontend'e iletir (Şekil 3.1).

Şekil 3.1

Uygulama, şu adımlarla çalışır:

1. Kullanıcı, uygulamayı açar ve gerekli izinler alınır.
2. Fotoğraf çekme ekranı aktif hale gelir.
3. Kullanıcı kıyafet fotoğrafını çeker ve bu görsel backend'e gönderilir.
4. Backend, TensorFlow Lite modeli ile desen analizi ve OpenCV ile renk analizi yapar.
5. Analiz sonuçları JSON formatında frontend'e iletilir ve kullanıcıya sesli olarak aktarılır.

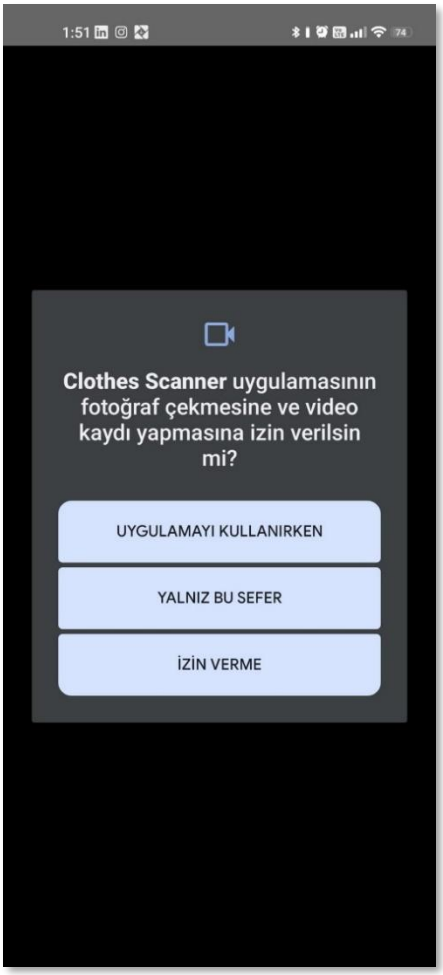
3.2 Kullanıcı Arayüzü

Uygulama, kullanıcı dostu bir arayüz ile geliştirilmiştir ve görme engelli bireylerin ihtiyaçlarını karşılamak için tasarlanmıştır.

3.2.1 Giriş Ekranı

Kullanıcı uygulamayı açtığında animasyonlu bir giriş ekranı (Şekil 3.2) ile karşılaşır. Bu ekranda kamera, mikrofon ve medya kütüphanesi izinleri (Şekil 3.3) istenir. İzinler alındıktan sonra fotoğraf çekme ekranına geçilir.

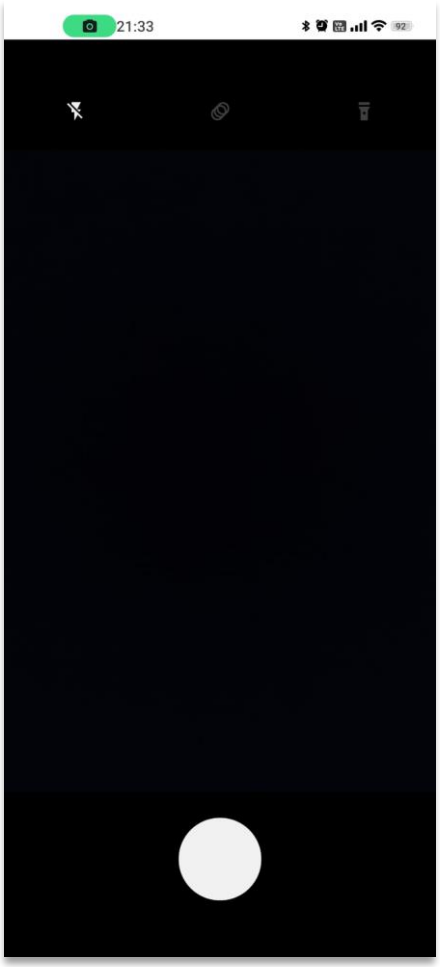
Şekil 3.2. Açılış animasyonu



Şekil 3.3. Kullanıcıdan alınan izinler

3.2.2 Fotoğraf Çekme Ekranı

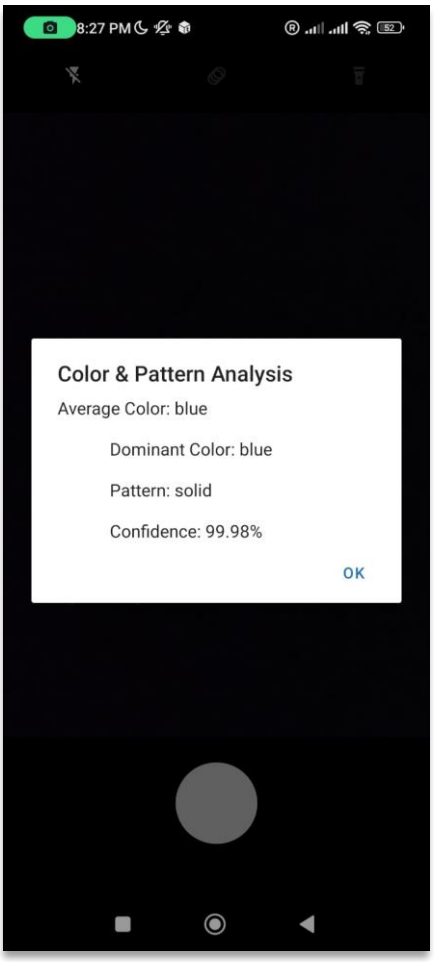
Kullanıcı, cihaz kamerasını kullanarak kıyafet fotoğrafını çeker. Kamera arayüzünde flaş kontrolü, zoom, ön/arka kamera geçişi gibi özellikler bulunmaktadır (Şekil 3.4).



Şekil 3.4. Kamera arayüzü

3.2.3 Sonuç Gösterim Ekranı

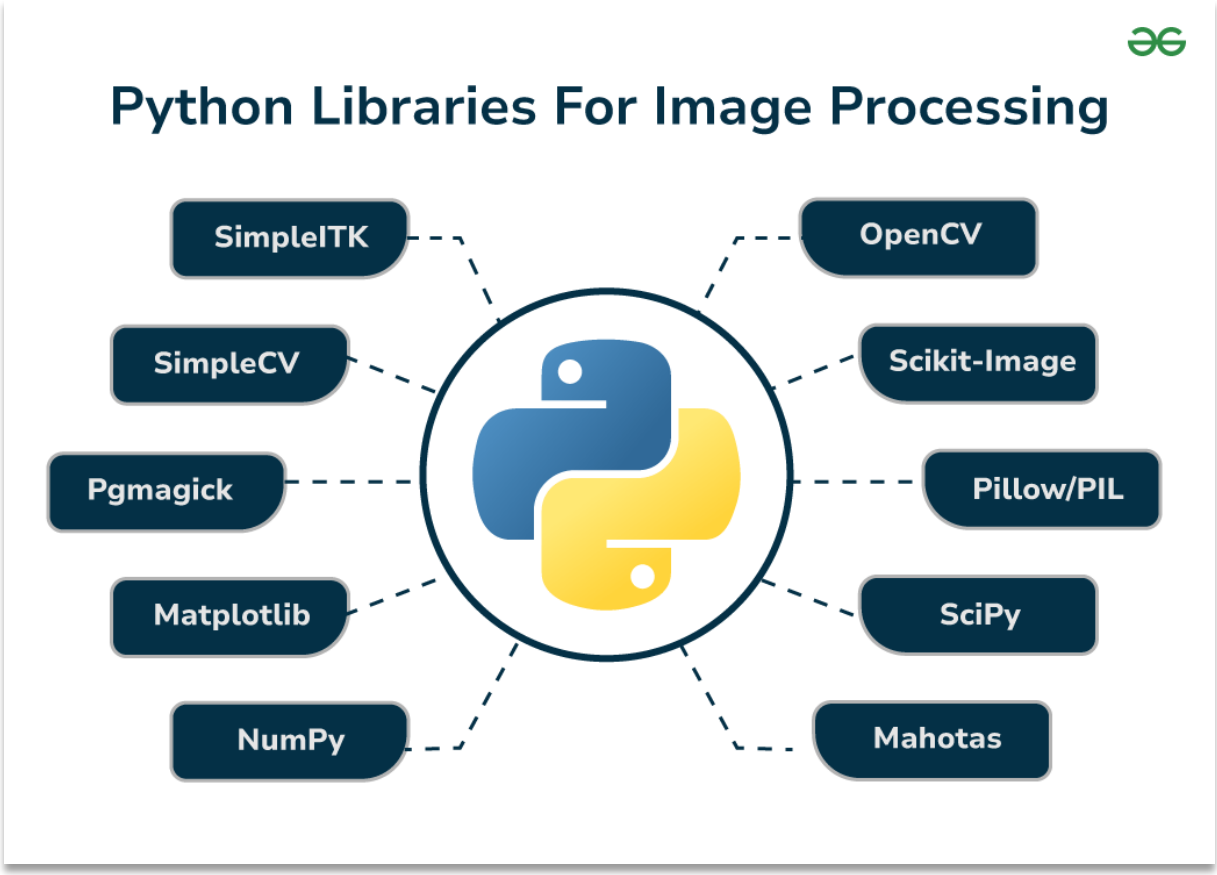
Fotoğraf çekildikten sonra backend'den alınan analiz sonuçları ekranda gösterilir. Sonuçlar arasında kıyafetin deseni, baskın renkleri ve ortalama rengi bulunur. Bu bilgiler ayrıca expo-speech kütüphanesi [10] ile sesli olarak iletilir.



Şekil 3.5. Sonuç gösterim ekranı

3.3 Algoritmalar ve Veri İşleme Süreci

Bu bölümde, projede kullanılan algoritmalar, veri işleme teknikleri ve yazılım kütüphaneleri detaylı bir şekilde açıklanmıştır. Görüntü işleme projelerinde sıkça kullanılan kütüphanelerden (Şekil 3.6) de bahsedilmiştir. Görsellerin işlenmesi, desenlerin tanımlanması, renk analizlerinin yapılması ve kullanılan teknolojilerin neden tercih edildiği üzerinde durulmuştur. Ayrıca, ilgili kavramlar ve kullanılan araçlar hakkında teorik bilgiler sunulmuştur.



Şekil 3.6. Görüntü işleme projelerinde sıkça kullanılan Python kütüphaneleri [11]

3.3.1 Veri İşleme

Makine öğrenmesi modelleri, girdilerin belirli bir formatta olmasını gerektirir. Bu nedenle, ham görseller modelin gereksinimlerine uygun hale getirilmek için çeşitli veri işleme adımlarından geçirilmiştir. Projede bu işlemler sırasında Pillow (PIL), NumPy ve TensorFlow gibi güçlü kütüphaneler kullanılmıştır.

1. Şeffaflık Kaldırma

Bazı görseller, özellikle PNG formatında, şeffaflık (transparency) içermektedir. Şeffaflık, modelin görselleri doğru bir şekilde işlemesini zorlaştırabilir. Modelin yalnızca RGB (kırmızı, yeşil, mavi) değerleriyle çalışabilmesi için bu tür görsellerin şeffaflık içeren kanalları kaldırılmıştır. Şeffaflık kaldırma işlemi sırasında arka plan beyaz renkle doldurulmuştur. Bu işlem modelin hata oranını azaltma amacı gütmektedir [12].

Kullanılan Araç: Pillow (PIL) kütüphanesi.

2. Yeniden Boyutlandırma

Makine öğrenmesi modelleri, sabit boyutlarda giriş verisi gerektirir. Bu projede, model giriş katmanının gereksinimlerini karşılamak için tüm görseller (128x128) boyutuna yeniden boyutlandırılmıştır [13].

Kullanılan Araçlar:

- Pillow (PIL): Görsellerin yeniden boyutlandırılması için kullanılmıştır.
- OpenCV: Alternatif olarak görsellerin boyutlarını değiştirmek için kullanılabilir.

3. Normalizasyon

Normalizasyon, görseldeki piksel değerlerini [0, 1] aralığına indirger. Piksel değerleri varsayılan olarak [0, 255] aralığında olduğundan, modelin daha hızlı ve stabil öğrenebilmesi için bu aralık küçültülmüştür [14].

Kullanılan Araçlar:

- NumPy: Görsellerin piksellerini kolayca işlemede kullanılmıştır.
- TensorFlow: Verilerin model için normalize edilmesini sağlayan yerleşik fonksiyonlar sunar.

4. Veri Artırma (Data Augmentation)

Veri artırma, modelin eğitim veri setinin çeşitliliğini artırmak ve genelleme kapasitesini geliştirmek amacıyla uygulanmıştır. Eğitim sırasında rastgele dönüşümler yapılarak modelin daha dayanıklı hale gelmesi sağlanmıştır.

Uygulanan Teknikler:

Döndürme (Rotation): Görseller rastgele döndürülerek farklı açılardan alınan veriler oluşturulmuştur.

Yakınlaştırma (Zoom): Görsellerin bazı bölümleri yakınlaştırılarak odaklanılan alanlar değiştirilmiştir.

Yatay Çevirme (Flip): Görseller yatay olarak çevrilmiştir.

Kaydırma (Shift): Görsellerin belirli eksenlerde yer değiştirmesi sağlanmıştır.

Parlaklık ve Kontrast Ayarı: Görsellerde rastgele parlaklık ve kontrast değişiklikleri uygulanmıştır.

Kullanılan Araçlar:

- TensorFlow (ImageDataGenerator): Veri artırma işlemlerini kolayca uygulamak için kullanılmıştır [15].

Fonksiyonlar:
rotation_range: Görselleri belirli bir açı aralığında döndürür.
zoom_range: Yakınlaştırma işlemini uygular.
width_shift_range ve height_shift_range: Görselleri eksenlerde kaydırır.
horizontal_flip: Görselleri yatay olarak çevirir.

- OpenCV: Görsellerin döndürülmesi, parlaklık ve kontrast artırma gibi özelleştirilmiş veri artırma işlemlerinde kullanılmıştır [16].

Fonksiyonlar:
cv2.getRotationMatrix2D: Görselleri belirli bir açıyla döndürmek için bir dönüşüm matrisi oluşturur.
cv2.warpAffine: Görselleri döndürmek veya kaydırmak için kullanılır.
cv2.addWeighted: Görselin parlaklığını ve kontrastını ayarlamak için kullanılır.

Kullanılan Araçların Genel Değerlendirmesi

- Pillow (PIL): Görsellerin açılması, formatlarının dönüştürülmesi ve yeniden boyutlandırılması gibi işlemlerde kullanılmıştır.

Önemli Fonksiyonlar:
Image.open: Görsel dosyasını açar.
Image.resize: Görselleri yeniden boyutlandırır.
Image.save: Düzenlenmiş görselleri kaydeder.

- NumPy: Piksel değerlerini normalize etmek ve görselleri işlemek için kullanılmıştır.

Önemli Fonksiyonlar:
np.array: Verileri NumPy dizisine dönüştürür.
np.mean: Belirli bir alandaki piksellerin ortalamasını alır.

- TensorFlow (ImageDataGenerator): Veri artırma işlemleri ve normalizasyon için kullanılmıştır.

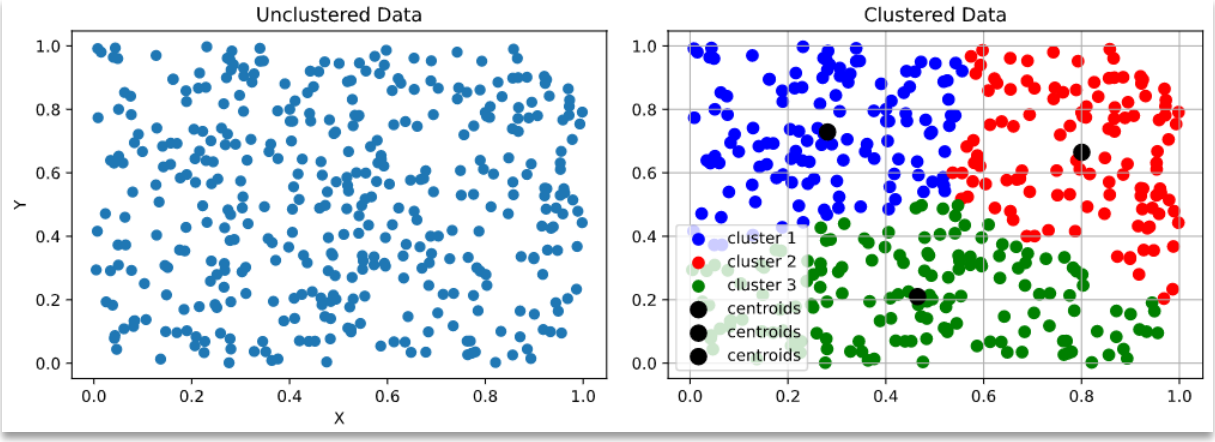
Önemli Fonksiyonlar:
ImageDataGenerator: Veri artırma işlemlerini otomatikleştirir.
flow_from_directory: Görselleri belirli bir dizinden yükler ve dönüşümleri uygular.

3.3.2 Renk Analizi

Projede renk analizi, görsellerdeki baskın renklerin ve ortalama rengin tespit edilmesi için gerçekleştirilmiştir. Bu süreçte, görsellerden anlamlı bilgiler çıkarmak ve kullanıcıya dost bir deneyim sunmak amacıyla **K-Means kümeleme algoritması**, **ortalama renk hesaplama** ve **ColorNamer** kütüphanesi kullanılmıştır. Kod incelendiğinde, bu işlemlerin etkin bir şekilde uygulandığı ve gereksinimlere uygun sonuçlar ürettiği görülmüştür.

1. K-Means Kümeleme Algoritması

K-Means kümeleme algoritması, pikselleri renk değerlerine (R, G, B) göre gruplandırarak (Şekil 3.7) görseldeki baskın renkleri belirlemek için kullanılmıştır [17]. Projede, görseldeki farklı renklerin gruplandırılması için 5 küme merkezi (k = 5) belirlenmiştir.



Şekil 3.7. K-Means kümeleme örneği

K-Means Algoritmasının İşleyişi

1. **Veri Hazırlama:**

- Görseller cv2.resize kullanılarak (480x640) boyutuna yeniden boyutlandırılmış ve her piksel (R, G, B) değerlerinden oluşan bir diziye dönüştürülmüştür.

2. **Başlangıç Küme Merkezlerinin Seçilmesi:**

- Küme merkezleri rastgele olarak seçilir. Bu merkezler başlangıç noktasıdır ve her iterasyonda güncellenir.

3. **Veri Noktalarının Kümelere Atanması:**

- Her veri noktası (örneğin, bir piksel), en yakın küme merkezine atanır.
- Yakınlık ölçütü olarak genellikle **Öklid Mesafesi** kullanılır:

$$d = \sqrt{\sum_{i=1}^{n}(x_i - \mu_i)^2}$$

4. **Küme Merkezlerinin Güncellenmesi:**

- Her kümenin merkez noktası, kümeye atanmış veri noktalarının ortalaması olarak yeniden hesaplanır.

5. **Tekrarlama:**
- Veri noktalarının kümelere atanması ve küme merkezlerinin güncellenmesi işlemleri, küme merkezleri sabitlenene kadar tekrarlanır.

6. **Baskın Renklerin Belirlenmesi:**

- Küme merkezlerinden en sık görülen renk, görselin baskın rengi olarak seçilmiştir.

Avantajları:

- **Hız ve Basitlik:** Hesaplama açısından hızlıdır ve kolayca uygulanabilir.
- **Uyarlanabilirlik:** Küme sayısı (k) projeye göre ayarlanabilir.
- **Görselleştirme:** Kümeleme sonuçları görselleştirilerek analiz edilebilir.

Dezavantajları:

- Küme sayısının (k) önceden belirlenmesi gerekir.
- Küme merkezlerinin başlangıç noktaları sonuçları etkileyebilir (yerel minimuma takılma riski).
- Küre biçiminde olmayan kümeler için uygun değildir.

Kullanım Alanları:

- **Görüntü İşleme:** Görsellerdeki renk kümelerinin tespit edilmesi.
- **Veri Madenciliği:** Büyük veri setlerinin gruplandırılması.
- **Müşteri Segmentasyonu:** Pazarlama verilerinin kümelenmesi.
- **Biyoinformatik:** Genetik verilerin analizi.

2. Ortalama Renk Hesaplama

Ortalama renk hesaplama, görsellerin genel renk tonunu belirlemek için basit ancak etkili bir yöntemdir. Bu yöntem, görseldeki tüm piksellerin RGB değerlerinin aritmetik ortalamasını alır [18].

1. Görselin tüm piksel değerleri bir diziye dönüştürülür.
2. Her bir renk kanalı için ortalama değer hesaplanır:

$$R_{avg} = \frac{\sum_{i=1}^{n} R_i}{n}, \quad G_{avg} = \frac{\sum_{i=1}^{n} G_i}{n}, \quad B_{avg} = \frac{\sum_{i=1}^{n} B_i}{n}$$

3. Ortaya çıkan değerler, ortalama rengin RGB temsilcisini oluşturur.

3. Renklerin Anlamlandırılması (ColorNamer Kullanımı)

Renk analizi sürecinde tespit edilen baskın ve ortalama renkler, ColorNamer kütüphanesi kullanılarak anlamlı isimlere dönüştürülmüştür [19].

ColorNamer İşlevleri:

- Renk Ailesi Belirleme: RGB renk değerlerinden anlamlı bir isim çıkarmak için renk ailesi tahmini yapılmıştır.
- Base Color Mapping: Renk isimleri, proje gereksinimlerine uygun şekilde yeniden düzenlenmiştir.

4. Alternatif Algoritmalar ve Yöntemler

Projede kullanılmayan diğer alternatif modeller de aşağıdaki gibidir:

1. Gaussian Mixture Models (GMM)

Gaussian Mixture Models, veri noktalarını birden fazla Gauss dağılımına dayalı olarak kümeler. K-Means algoritmasına göre daha esnek bir yöntemdir [20].

Avantajları:

- Her veri noktası, birden fazla kümeye belirli bir olasılıkla atanır.
- Daha karmaşık veri dağılımlarında daha iyi sonuç verir.

Dezavantajları:

- Daha yavaş ve hesaplama açısından daha yoğundur.

Kullanım Alanları:

- Görsellerdeki renk gruplarını anlamlandırmak.
- Ses işleme ve konuşma tanıma.

2. Mean-Shift Clustering

Bu algoritma, veri yoğunluklarını analiz ederek doğal küme merkezlerini belirler. Küme sayısını önceden belirtmek gerekmez [21].

Avantajları:

- Küme sayısı otomatik olarak belirlenir.
- Renk yoğunluğu temelli analizler için uygundur.

Dezavantajları:

- Büyük veri setlerinde yavaş çalışır.
- Belirli bir bant genişliği (h) seçimi sonuçları etkileyebilir.

Kullanım Alanları:

- Görsellerdeki nesne tespiti.

- Video analizinde hareketli nesne takibi.

3. Hierarchical Clustering

Hierarchical Clustering, veri noktalarını iteratif olarak birleştirerek veya ayırarak kümeler [22].

Avantajları:

- Küme sayısını belirtmeden kümeleme yapılabilir.
- Kümeler arası hiyerarşi görselleştirilebilir (dendrogramlar ile).

Dezavantajları:

- Büyük veri setlerinde hesaplama maliyeti yüksektir.
- Küme sayısının doğru belirlenmesi zordur.

Kullanım Alanları:

- Biyoinformatik: Genetik veri analizi.
- Belge Kümeleme: Metinlerin sınıflandırılması.

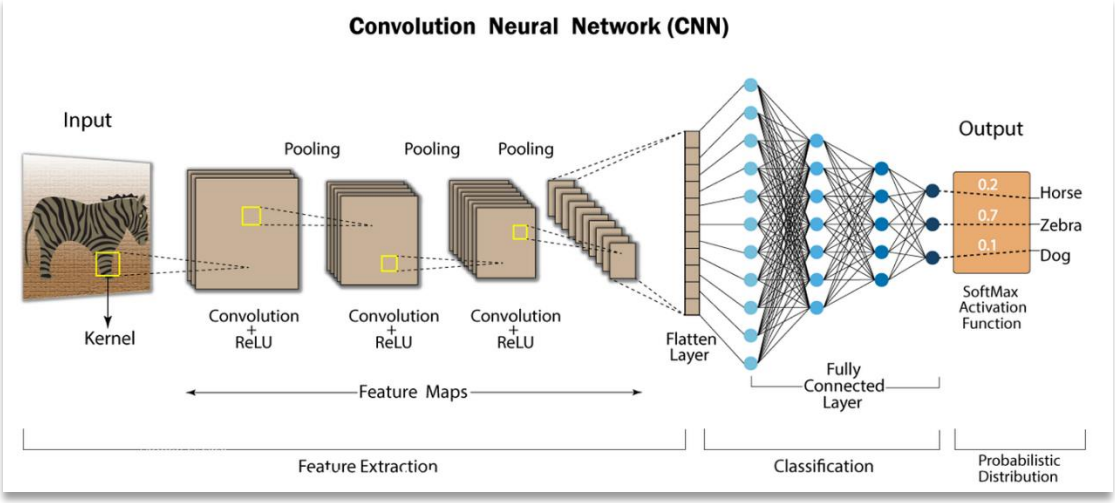
Sonuç ve Katkılar

Bu süreçte K-Means algoritması ve ortalama renk hesaplama yöntemleri bir arada kullanılarak, görsellerden baskın ve genel renk bilgileri başarıyla çıkarılmıştır. ColorNamer ile renklerin anlamlandırılması, kullanıcı deneyimini artırmıştır. K-Means algoritmasının hızı ve doğruluğu, bu projede renk analizi için en uygun yöntemlerden biri olduğunu kanıtlamıştır.

3.3.3 Desen Analizi ve Convolutional Neural Network (CNN)

CNN Nedir?

Convolutional Neural Network (CNN), görsel verilerden özellik çıkarma ve sınıflandırma için kullanılan bir derin öğrenme modelidir. CNN'ler, konvolüsyon katmanları aracılığıyla görseldeki çizgiler, kenarlar ve dokular gibi yerel özellikleri analiz eder [23].



Şekil 3.8. CNN model çalışma mantığı

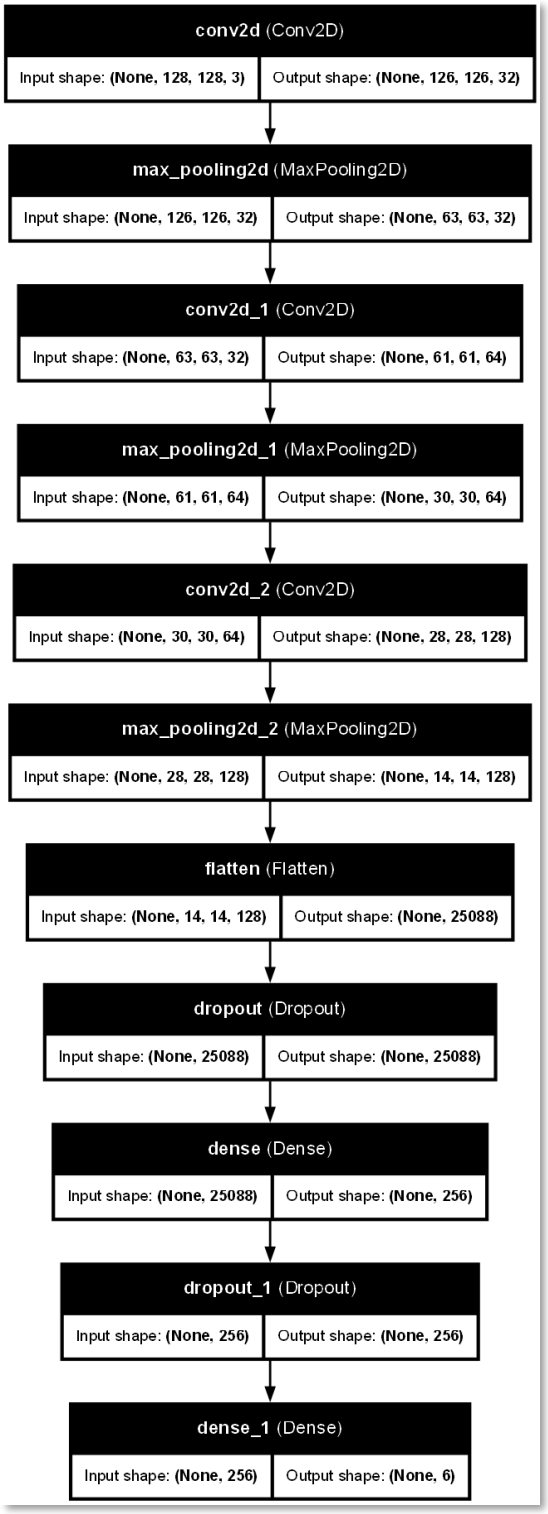
Projede Kullanılan CNN Modeli

Bu projede kullanılan CNN modeli (Şekil 3.9), üç adet konvolüsyon katmanı, havuzlama katmanları ve yoğun (fully connected) katmanlardan oluşmaktadır.

Model Yapısı:

1. **Girdi Katmanı:** (128x128x3) boyutunda RGB görsel alır.
2. **Konvolüsyon Katmanları:** Görselin yerel özelliklerini çıkarır.
   - Kullanılan Conv2D fonksiyonu, çekirdek boyutu (3x3) olarak ayarlanmıştır.
3. **Havuzlama Katmanları:** (2x2) havuzlama işlemiyle boyut küçültülür.
4. **Dropout:** Rastgele nöronları devre dışı bırakarak aşırı öğrenmeyi (overfitting) önler.
5. **Softmax Çıkış Katmanı:** Görseli belirlenen 6 sınıftan birine atar.

Kullanılan Araç: TensorFlow.



Şekil 3.8. Projede kullanılan CNN modelinin akış şeması

CNN ile Yapılabilecek Projeler

- **Görüntü Sınıflandırma:** Örneğin, kıyafet türlerini veya bitki yapraklarını tanıma.
- **Nesne Algılama:** Trafik işaretlerini tespit etme.

- **Tıbbi Görüntü Analizi:** Röntgenlerde tümör tespiti.
- **Sanat Analizi:** Tablo türlerini veya ressamları tanımlama.

3.3.4 Kullanılan Kütüphaneler ve Fonksiyonlar

Bu bölümde, projede kullanılan tüm kütüphaneler detaylı olarak açıklanmıştır. Hem frontend hem de backend tarafında kullanılan bu kütüphanelerin temel fonksiyonları ve projemize sağladıkları katkılar detaylandırılmıştır. Ayrıca, kullanılan kütüphanelerin sunduğu ek fonksiyonlar ve alternatif yöntemler de ele alınmıştır.

Frontend (React Native) Kütüphaneleri

1. **expo-camera**

İşlev: Kamera erişimini sağlar ve fotoğraf çekme işlemleri için kullanılır [24].

Temel Fonksiyonlar:

- CameraView: Kamera bileşenini oluşturur.
- takePictureAsync(): Fotoğraf çekme işlemini gerçekleştirir.

Projedeki Kullanımı:

- Kullanıcının kamera ile fotoğraf çekmesine olanak tanınmıştır.

2. **expo-media-library**

İşlev: Çekilen fotoğrafların cihaza kaydedilmesi ve yönetilmesini sağlar [25].

Temel Fonksiyonlar:

- createAssetAsync(): Yeni medya dosyası oluşturur.
- getAssetsAsync(): Mevcut medya dosyalarını listeler.

Projedeki Kullanımı:

- Fotoğrafların cihaz hafızasında saklanması için kullanılmıştır.

3. **Axios**

İşlev: HTTP istekleri yaparak frontend ve backend arasında veri alışverişi sağlar [26].

Temel Fonksiyonlar:

- axios.post(): Backend'e veri göndermek için kullanılır.
- axios.get(): Backend'den veri almak için kullanılır.

Projedeki Kullanımı:

- Fotoğrafların backend'e gönderilmesi ve sonuçların alınması için tercih edilmiştir.

4. **expo-speech**

İşlev: Metni sese dönüştürür [10].

Temel Fonksiyonlar:

- Speech.speak(): Belirtilen metni sesli olarak okur.

Projedeki Kullanımı:

- Desen ve renk analiz sonuçlarının kullanıcıya sesli olarak iletilmesi sağlanmıştır.

5. Translate

İşlev: Metin çevirileri yapmak için kullanılır.

Temel Fonksiyonlar:

- translate(): Belirtilen metni hedef dile çevirir.

Projedeki Kullanımı:

- Çıktıların Türkçe'ye çevrilmesinde kullanılmıştır.

Backend (Flask ve Python) Kütüphaneleri

1. Flask

İşlev: Web sunucusu ve API işlemleri için kullanılır [7].

Temel Fonksiyonlar:

- @app.route(): URL son noktalarını tanımlar.
- request.files: Kullanıcıdan gelen dosyaları işler.

Projedeki Kullanımı:

- Backend servislerini çalıştırmak ve API işlemlerini gerçekleştirmek için kullanılmıştır.

2. OpenCV (cv2)

İşlev: Görüntü işleme algoritmaları sağlar [6].

Temel Fonksiyonlar:

- cv2.resize(): Görselleri yeniden boyutlandırır.
- cv2.kmeans(): Görsel renklerini analiz etmek için K-Means algoritmasını uygular.

Projedeki Kullanımı:

- Görseldeki baskın renklerin tespiti ve veri artırma işlemleri için kullanılmıştır

3. TensorFlow ve TensorFlow Lite

İşlev: Makine öğrenmesi modeli eğitimi ve çalıştırılması için kullanılır [27].

Temel Fonksiyonlar:

- tf.keras.Sequential(): Modelin katmanlarını tanımlar.
- tf.lite.Interpreter(): TensorFlow Lite modellerini çalıştırır.

Projedeki Kullanımı:

- Desen tanıma modeli oluşturulmuş ve düşük maliyetli çalıştırma için TFLite kullanılmıştır.

4. NumPy

- **İşlev:** Sayısal veri işlemleri sağlar [28].
- **Temel Fonksiyonlar:**
- np.array(): Veriyi NumPy dizisine dönüştürür.
- np.mean(): Ortalama değer hesaplar.

Projedeki Kullanımı:

- Piksel değerleri ve renk analizlerinde hesaplamalar için kullanılmıştır.

5. Pillow (PIL)

İşlev: Görsellerin işlenmesi ve format dönüştürme işlemleri için kullanılır [13].

Temel Fonksiyonlar:

- Image.open(): Görsel dosyasını açar.
- Image.alpha_composite(): Şeffaflık içeren görsellerin arka planını doldurur.

Projedeki Kullanımı:

- Görsellerin yeniden boyutlandırılması ve şeffaflığın kaldırılması işlemlerinde kullanılmıştır.

6. ColorNamer

İşlev: RGB renk değerlerini anlamlı isimlere dönüştürür [29].

Temel Fonksiyonlar:

- get_color_from_rgb(): Belirtilen renk değerini adlandırır.

Projedeki Kullanımı:

- Görseldeki baskın ve ortalama renklerin kullanıcı dostu şekilde adlandırılması sağlanmıştır

Sonuç

Projede kullanılan kütüphaneler, farklı ihtiyaçlara yönelik çözüm sağlayarak her aşamanın başarıyla tamamlanmasını mümkün kılmıştır. Özellikle TensorFlow, OpenCV ve Flask gibi güçlü araçlar, hem performans hem de kullanım kolaylığı açısından projeye katkı sağlamıştır. Her bir kütüphanenin yetenekleri detaylı olarak değerlendirilmiş ve projeye uygun olanlar seçilmiştir.

4. GELİŞTİRME

Uygulamanın genel işleyişine (Şekil 4.1) göre adım adım uygulamanın geliştirilmesi sağlanmıştır. İlk adım veri setinin optimizasyonu daha sonraki adımlar sırasıyla modelin eğitilmesi, backend ve frontend tasarımlarıdır.

Şekil 4.1. Uygulamanın kullanıcı tarafında genel işleyişi

4.1 Veri Seti Araştırması ve Yapısı

Bu çalışma için kullanılan veri seti, altı farklı sınıftan oluşan geniş bir "Clothing Pattern Dataset" üzerinedir. Bu sınıflar, düz (solid), çizgili (striped), noktalı (dotted), kareli (checkered), zikzak (zigzag) ve çiçekli (floral) desenlerden oluşmaktadır. Veri seti, bu sınıfları temsil eden 6000 görüntüyü içermektedir.

Veri setinin büyük bir kısmı (%90) çevrimiçi kaynaklardan ve Google üzerinden toplanmıştır. Bu veri setinin orijinal versiyonuna (%10) şu GitHub bağlantısından erişebilirsiniz: Clothing Pattern Dataset [30]. Ancak veri setinin doğrudan bir kopyası sağlanmamakta, bunun yerine görüntü URL'leri ve görüntülerin kırpılması ve yeniden boyutlandırılması için gerekli bilgiler sunulmaktadır.

Her bir görüntü aşağıdaki adımlarla işlenmiştir:

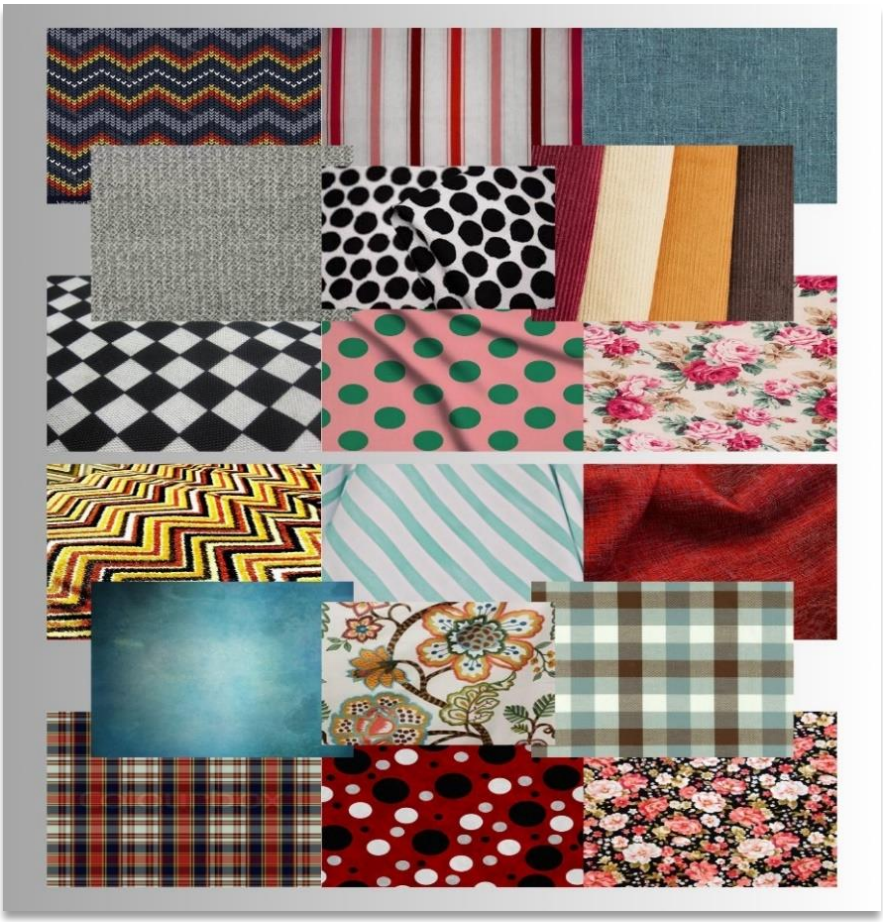
- Sağlanan URL'lerden indirilen görüntüler, verilen dikdörtgen koordinatlarla kırpılmıştır.

- Kırpılan görüntüler, 224x224 piksel çözünürlüğünde olacak şekilde ölçeklendirilmişti. Projede görüntülerin büyük çoğunluğu 640x640 piksel boyutundadır. Ancak farklı çözünürlüklerdeki görüntüler de yeniden boyutlandırılmıştır.
- Her ölçek için, görüntünün 30 derece artışlarla döndürülerek 12 farklı varyasyonu oluşturulmuştur. Bunun dışında kontrast, parlaklık ve renk değişimleri yapılarak veri seti çoğaltma işlemi yapılmıştır.

Bu çalışmada, 6000 görüntüden oluşan veri seti, %80 eğitim (training) ve %20 doğrulama (validation) olmak üzere iki gruba ayrılmıştır (Şekil 4.2). Eğitim veri seti, modeli öğrenmek için kullanılırken, doğrulama veri seti modelin performansını değerlendirmek için kullanılmaktadır.

Veri setinin çeşitliliği, kumaş desenlerinin farklı boyutlar ve perspektiflerden temsil edilmesiyle artırılmıştır. Bu, modelin genel performansını artırmayı hedeflemiştir.

Bu bölüm, veri seti araştırmasının genel yapısını ve işleme adımlarını kapsamaktadır. Aşağıdaki bölümlerde; model eğitimi ve sonuçları, kod yapısını incelenerek detaylı olarak ele alınacaktır.



Şekil 4.2. Veri setindeki eğitim ve doğrulama görselleri

4.2 Model Eğitimi

4.2.1 Amaç ve Model Seçimi

Bu projede, kıyafet desenlerini tanıma ve sınıflandırma işlemleri için bir **Convolutional Neural Network (CNN)** modeli geliştirilmiştir. CNN, görsellerin uzaysal ilişkilerini ve desenlerini öğrenmekte oldukça başarılı bir yapay sinir ağı türüdür [23]. Bu tür bir model, özellikle görsel veri üzerinde çalışıldığında yüksek doğruluk oranları sağladığı için tercih edilmiştir. Modelin temel amacı, görselleri altı sınıfa doğru bir şekilde sınıflandırmaktır: **checkered**, **dotted**, **floral**, **solid**, **striped** ve **zigzag**.

Eğitim süreci TensorFlow kullanılarak gerçekleştirilmiş, model daha düşük maliyetli ve verimli bir şekilde çalıştırılabilmesi için **TensorFlow Lite (TFLite)** formatına dönüştürülmüştür. CNN mimarisi, aşağıdaki katmanlardan oluşmaktadır ve her bir katman farklı bir işlevi yerine getirir:

CNN Mimarisi ve Katmanların İşlevleri

1. **Convolutional (Konvolüsyon) Katman:**

- Görselden özellik çıkarma işlemini gerçekleştirir.
- Her filtre, görseldeki kenarlar, köşeler veya belirli desenler gibi özellikleri öğrenir.
- Aktivasyon fonksiyonu olarak **ReLU (Rectified Linear Unit)** kullanılmıştır. ReLU, negatif değerleri sıfıra çekerek modelin doğrusal olmayan ilişkileri öğrenmesini sağlar.

2. **MaxPooling Katmanı:**

- Görselden çıkarılan özellikleri özetler ve boyutlarını küçültür.
- Hesaplama yükünü azaltır ve modelin aşırı öğrenme riskini düşürür.

3. **Flatten (Düzleştirme) Katmanı:**

- İki boyutlu özellik haritalarını, yoğun bağlantı (dense) katmanlarına giriş olarak kullanılacak tek boyutlu bir vektöre dönüştürür.

4. **Dense (Yoğun Bağlantı) Katmanı:**

- Görseldeki özelliklerden anlam çıkarır ve sınıflandırma işlemini gerçekleştirir.
- İlk dense katman, daha fazla öğrenme kapasitesi için 256 nöron kullanır.

5. **Dropout Katmanı:**

- Aşırı öğrenmeyi önlemek için bazı nöronların rastgele devre dışı bırakılmasını sağlar.
- Bu modelde, %50 oranında dropout uygulanmıştır.

6. **Softmax Çıkış Katmanı:**

- Modelin her sınıfa ait olasılık skorunu üretir. En yüksek olasılık değeri, görselin ait olduğu sınıfı belirler.

Kütüphanelerin Yüklenmesi

TensorFlow gibi güçlü bir kütüphanenin kullanımı, model eğitiminin her aşamasını kolaylaştırmıştır. Gerekli kütüphaneler şu komutlarla yüklenebilir:

```
pip install tensorflow
pip install numpy
pip install matplotlib
```

4.2.2 Kod İncelemesi ve Yorumlaması

Kodun önemli bölümleri aşağıda açıklamalı olarak verilmiştir:

Veri Ön İşleme

Veri setindeki görseller, şeffaflık içerebileceği için şeffaflık kaldırma işlemi gerçekleştirilmiştir. Şeffaflık kaldırma, bir görüntüyü RGBA formatından RGB formatına dönüştürerek beyaz bir arka plan eklenmesiyle sağlanmıştır. Bu işlem, hem eğitim hem de doğrulama verileri için uygulanmıştır [12].

```python
# Şeffaflık kaldırma fonksiyonu
def remove_transparency(image_path, output_path):
    """Bir görüntünün şeffaflığını kaldırır ve çıktıyı kaydeder."""
    img = Image.open(image_path).convert("RGBA")
    background = Image.new("RGBA", img.size, (255, 255, 255))
    img = Image.alpha_composite(background, img).convert("RGB")
    img.save(output_path)

# Veri setindeki tüm görselleri işlemek için fonksiyon
def preprocess_images(directory):
    for root, _, files in os.walk(directory):
        for file in files:
            if file.lower().endswith(('png', 'jpg', 'jpeg')):
                file_path = os.path.join(root, file)
                try:
                    temp_path = file_path
                    remove_transparency(file_path, temp_path)
                    os.replace(temp_path, file_path)
                except Exception as e:
                    print(f"Error processing {file_path}: {e}")

# Eğitim ve doğrulama görsellerini ön işleme
preprocess_images('backend\\datasets\\train')
preprocess_images('backend\\datasets\\validate')
```

Veri Artırma

Eğitim setindeki görseller üzerinde veri artırma işlemleri gerçekleştirilmiştir. Bu işlemler arasında dönme, yatay kaydırma, dikey kaydırma, kesme, yakınlaştırma ve yatay çevirme yer almaktadır. Doğrulama seti için yalnızca yeniden ölçeklendirme yapılmıştır [15].

```
# Eğitim ve doğrulama veri jeneratörleri
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

validation_datagen = ImageDataGenerator(rescale=1.0 / 255)

train_generator = train_datagen.flow_from_directory(
    'backend\\datasets\\train',
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical',
    shuffle=True
)

validation_generator = validation_datagen.flow_from_directory(
    'backend\\datasets\\validate',
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)
```

Model Mimarisi

Model, üç adet evrişim (Conv2D) ve havuzlama (MaxPooling2D) katmanına sahiptir. Ayrıca, tamamen bağlı (Dense) katmanlar ve aşırı öğrenmeyi önlemek için Dropout katmanları kullanılmıştır. Çıkış katmanı, sınıfların olasılık dağılımını tahmin etmek için Softmax aktivasyon fonksiyonuna sahiptir [23].

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(len(train_generator.class_indices), activation='softmax')
])
```

Model Eğitimi

Model, categorical_crossentropy kayıp fonksiyonu ve Adam optimizasyon algoritması kullanılarak eğitilmiştir [31]. Modelin performansını izlemek ve iyileştirmek için erken durdurma (EarlyStopping) ve öğrenme oranını azaltma (ReduceLROnPlateau [32]) gibi callback'ler eklenmiştir.

```
model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    metrics=['accuracy']
)

callbacks = [
    tf.keras.callbacks.ReduceLROnPlateau(
        monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6, verbose=1
    ),
    tf.keras.callbacks.EarlyStopping(
        monitor='val_loss', patience=5, restore_best_weights=True, verbose=1
    ),
    tf.keras.callbacks.ModelCheckpoint(
        'backend/pattern_recognition_best_model.keras', save_best_only=True, verbose=1
    )
]

history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    epochs=50,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // validation_generator.batch_size,
    callbacks=callbacks
)
```

Modelin Kaydedilmesi

Eğitim tamamlandıktan sonra, model dosya sistemine kaydedilmiştir:

```
model.save('backend/pattern_recognition_model.keras')
```

4.3 Eğitilen Modelin Test Edilmesi

4.3.1 Amaç ve Model Testi

Bu bölümde, eğitilen modelin test edilmesi için bir dizi görsel kullanılmıştır. Amaç, modelin doğruluğunu ve güvenilirliğini test etmek ve her bir görüntü için sınıflandırma sonuçlarını elde etmektir. Model, daha önce eğitim verileri üzerinde eğitilmiş ve ardından test için kullanılacak yeni görsellerle test edilmiştir. Bu test işlemi, modelin görsel verileri doğru sınıflara atayıp atamadığını ölçmek için önemlidir.

Model, sınıflandırma yapabilmesi için görselleri alır, bunları uygun şekilde işler ve her bir görselin hangi sınıfa ait olduğunu belirler. Test süreci sırasında her görsel için sınıflandırma sonuçları ve modelin güven seviyeleri de alınarak, modelin ne kadar doğru tahminler yaptığına dair bilgi sağlanır.

4.3.2 Kod İncelemesi ve Yorumlaması

Kodun önemli bölümleri aşağıda açıklamalı olarak verilmiştir:

Modelin Yüklenmesi

Bu bölümde, tensorflow ve numpy kütüphaneleri dahil edilmiştir. Modelin dosya yolu model_path değişkenine atanmıştır. Daha sonra, model dosyasının mevcut olup olmadığı kontrol edilmekte ve dosya mevcutsa model tf.keras.models.load_model() fonksiyonu ile yüklenmektedir. Eğer model dosyasında bir hata varsa, FileNotFoundError ya da ValueError hatası verilmektedir.

```
import tensorflow as tf
import numpy as np
import os
from tensorflow.keras.preprocessing.image import load_img, img_to_array  # type: ignore

# Path to the model
model_path = "backend/pattern_recognition_best_model.keras"

# Check if the model file exists
if not os.path.exists(model_path):
    raise FileNotFoundError(f"Model file not found: {model_path}. Please ensure the file i

# Load the trained model
try:
    model = tf.keras.models.load_model(model_path)
except ValueError as e:
    raise ValueError(f"Error loading model: {e}. Ensure the file is a valid .keras model."
```

Sınıf Etiketleri

Burada, sınıflar (etiketler) belirlenmiştir. Modelin tahmin edeceği sınıflar şu şekilde sıralanmıştır: 'checkered', 'dotted', 'floral', 'solid', 'striped', 'zigzag'.

```
# Class labels
classes = ['checkered', 'dotted', 'floral', 'solid', 'striped', 'zigzag']
```

Test Edilecek Resmin Hazırlanması ve Model ile Tahmin Yapılması

Bu fonksiyon, test edilmek istenen görselin yolunu alır. İlk olarak, belirtilen dosyanın varlığı kontrol edilir. Eğer dosya bulunamazsa, FileNotFoundError hatası alınır. Dosya mevcutsa, load_img fonksiyonu ile resim yüklenir ve img_to_array fonksiyonu ile numpy dizisine dönüştürülür. Görselin boyutu 128x128 olarak ayarlanır ve normalize edilir (0-1 aralığına çekilir). Son olarak, model tahminini yapmak için resim bir batch (mini-batch) haline getirilir.

Tahmin işlemi model.predict() ile yapılır. Modelin tahmini doğrultusunda, np.argmax() fonksiyonu ile en yüksek olasılığı taşıyan sınıf belirlenir ve bu sınıfın adı döndürülür. Ayrıca, tahminin doğruluğu (güven düzeyi) yüzdelik olarak hesaplanır.

```
def predict_pattern(image_path):
    # Check if the image file exists
    if not os.path.exists(image_path):
        raise FileNotFoundError(f"Image file not found: {image_path}. Please provide a vali

    # Load and preprocess the image
    try:
        img = load_img(image_path, target_size=(128, 128))
        img_array = img_to_array(img) / 255.0
        img_array = np.expand_dims(img_array, axis=0)
    except Exception as e:
        raise ValueError(f"Error processing image: {e}")

    # Make prediction
    try:
        predictions = model.predict(img_array)
        predicted_class = classes[np.argmax(predictions)]
        confidence = np.max(predictions) * 100
    except Exception as e:
        raise RuntimeError(f"Error during prediction: {e}")

    return predicted_class, confidence
```

Modelin Test Edilmesi

Bu bölümde, predict_pattern fonksiyonu test edilen resimler üzerinde çalıştırılmaktadır. Her bir resim için tahmin edilen desen ve güven düzeyi yazdırılmaktadır. Eğer herhangi bir hata oluşursa, hata mesajı yakalanarak kullanıcıya bildirilir.

```python
# Test the model
image_path = r"test_image1.jpg"
image_path2 = r"test_image2.jpg"
# ... (diğer resim yolları)

try:
    pattern, confidence = predict_pattern(image_path)
    print(f"so-Pattern: {pattern}, Confidence: {confidence:.2f}%")
    pattern, confidence = predict_pattern(image_path2)
    print(f"do-Pattern: {pattern}, Confidence: {confidence:.2f}%")
    # ... (diğer test görüntüleri)
except Exception as e:
    print(f"An error occurred: {e}")
```

Sonuçların Değerlendirilmesi

Eğitilen modelin tahmin sonuçları, her bir görsel için tahmin edilen desenin ve modelin bu tahminde ne kadar güvenli olduğunu (güven düzeyi) göstermektedir. Örneğin: Aşağıdaki çıktı, modelin test_image1.jpg görseli için tahmin ettiği desenin "striped" olduğunu ve modelin bu tahmine %92.45 güven duyduğunu gösterir. Her resim için benzer şekilde tahminler yapılır ve her bir tahminin güven düzeyine göre modelin başarısı değerlendirilir.

```
so-Pattern: striped, Confidence: 92.45%
```

4.4 Backend Geliştirme

4.4.1 Amaç ve Genel Yapı

Backend tarafı, kullanıcıdan alınan görsellerin işlenmesi, desen ve renk analizi yapılması, ardından sonuçların frontend'e iletilmesi gibi temel görevleri yerine getiren bir API yapısıdır. Bu yapı, Python ve Flask framework'ü kullanılarak geliştirilmiştir. Flask, hızlı ve esnek bir web framework'ü olması nedeniyle tercih edilmiştir. Görsel işleme ve analiz işlemleri için **OpenCV**, **NumPy** ve **ColorNamer** gibi kütüphaneler kullanılmıştır.

Backend'in ana görevleri:

1. Kullanıcıdan alınan görsellerin işlenmesi (şeffaflık kaldırma, yeniden boyutlandırma).
2. Renk analizi (baskın ve ortalama renklerin belirlenmesi).
3. Makine öğrenmesi modeli kullanılarak desen sınıflandırması yapılması.
4. İşlenmiş sonuçların frontend'e JSON formatında iletilmesi.

Kütüphanelerin Yüklenmesi

```
pip install flask tensorflow opencv-python psutil pillow colornamer memory-profiler
```

Bu komutları terminal veya komut satırında çalıştırarak, aşağıdaki kütüphaneleri yükleyebilirsiniz:

- **Flask**: Web uygulaması geliştirmek için [7].
- **TensorFlow**: Derin öğrenme modelleri için [27].
- **OpenCV**: Görüntü işleme için [6].
- **psutil**: Sistem kaynaklarını izlemek için [33].
- **Pillow (PIL)**: Görüntü işleme ve dönüşümleri için [13].
- **colornamer**: RGB renklerinden renk adı belirlemek için [29].
- **memory-profiler**: Bellek kullanımı izlemek için [34].

4.4.2 Kod İncelemesi ve Yorumlaması

Aşağıda backend geliştirme sürecinde kullanılan kodlar ve açıklamaları verilmiştir:

Flask Uygulamasının Başlatılması

```
app = Flask(__name__)
```

Açıklama:
Bu satır, Flask uygulamasını başlatarak API'nin temellerini oluşturur.

Modelin Yüklenmesi ve TensorFlow Lite Entegrasyonu

```
PROCESSED_IMAGE_DIR = 'images'
os.makedirs(PROCESSED_IMAGE_DIR, exist_ok=True)

TFLITE_MODEL_PATH = "pattern_recognition_model.tflite"
if not os.path.exists(TFLITE_MODEL_PATH):
    raise FileNotFoundError(f"TFLite model file not found: {TFLITE_MODEL_PATH}")
```

Açıklama:

- PROCESSED_IMAGE_DIR ile işlenmiş resimler için bir klasör oluşturuluyor.

- TensorFlow Lite model dosyasının yolu belirleniyor ve modelin var olup olmadığı kontrol ediliyor. Eğer model bulunamazsa, hata mesajı veriliyor.

TFLite Modelinin Başlatılması

```
interpreter = tf.lite.Interpreter(model_path=TFLITE_MODEL_PATH)
interpreter.allocate_tensors()

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

classes = ['checkered', 'dotted', 'floral', 'solid', 'striped', 'zigzag']
```

Açıklama:

- TensorFlow Lite modelini yüklemek için bir Interpreter nesnesi oluşturuluyor ve model için gerekli tensörler tahsis ediliyor [27].
- Modelin giriş ve çıkış detayları (input_details ve output_details) alınarak modelin nasıl veri aldığını ve çıktıyı nasıl verdiği belirleniyor.
- Sınıf etiketleri (classes) modeli test ederken kullanılacak.

Renk İsmi Elde Etme

```
def get_closest_color_name(rgb_color):
    if all(value <= 60 for value in rgb_color):
        return "black"
    if all(value >= 240 for value in rgb_color):
        return "white"
    color_info = get_color_from_rgb(rgb_color)
    color_family = color_info['color_family'].lower()
    return base_color_mapping.get(color_family, color_family)
```

Açıklama:

Verilen RGB değerlerine en yakın renk ismini belirlemek için bir fonksiyon tanımlanıyor. Önce siyah veya beyaz olup olmadığı kontrol ediliyor. Ardından, colornamer kütüphanesi kullanılarak renk ailesi belirleniyor ve bu aileye en uygun renk ismi base_color_mapping üzerinden alınıyor.

Desen Tahmin Fonksiyonu

```python
def predict_pattern(pil_image):
    img_array = np.array(pil_image.resize((128, 128))) / 255.0
    img_array = np.expand_dims(img_array, axis=0).astype(np.float32)

    interpreter.set_tensor(input_details[0]['index'], img_array)
    interpreter.invoke()

    predictions = interpreter.get_tensor(output_details[0]['index'])
    predicted_class = classes[np.argmax(predictions)]  # Predicted class
    confidence = np.max(predictions) * 100  # Confidence score
    return predicted_class, confidence
```

Açıklama:

Bu fonksiyon, PIL formatındaki bir görüntüyü alır, boyutunu 128x128'e küçültür, normalize eder ve TensorFlow Lite modeline uygun hale getirir. Model üzerinden tahmin yaparak en yüksek olasılıkla sınıfı döndürür. Ayrıca, tahminin güven skorunu da hesaplar.

Görüntü İşleme ve Desen Tanıma API'si

```python
@app.route('/process-image', methods=['POST'])
def process_image_route():
    if 'image' not in request.files:
        return jsonify({'error': 'No image file provided'}), 400

    image_file = request.files['image']
    image_bytes = BytesIO(image_file.read())
    pil_image = Image.open(image_bytes).convert("RGB")
    img_rgb = np.array(pil_image)

    img_resized = cv2.resize(img_rgb, (480, 640))
    pixels = img_resized.reshape(-1, 3)
    k = 5  # Number of clusters
    _, labels, centers = cv2.kmeans(pixels.astype(np.float32), k, None,
        (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0),
        10, cv2.KMEANS_RANDOM_CENTERS
    )
```

Açıklama:

Bu route, bir POST isteği alır ve bir resim üzerinde işlem yapar. Resim önce belleğe yüklenir, sonra OpenCV kullanılarak yeniden boyutlandırılır [13]ve K-Means kümeleme algoritması [17] ile en baskın renkleri bulur. Sonrasında, baskın renkler ve renk isimleri döndürülür.

4.4.3 Hostlama Süreci

Backend uygulaması, Flask framework'ü üzerine inşa edilmiştir ve bir web sunucusu olarak çalışmaktadır. Hostlama sürecinde, bu uygulamanın yerel olarak veya bir bulut platformunda çalıştırılması sağlanmıştır. Projenin geliştirildiği sürece uygun olarak Render.com platformu tercih edilmiştir. Aşağıda, hostlama işlemleri ve kod incelemesi yer almaktadır.

Hostlama için Gerekli Ayarlar

Platform Seçimi:

- **Render.com:** Ücretsiz katmanı ve Flask tabanlı uygulamalara uyumluluğu nedeniyle tercih edilmiştir [9].
- Alternatifler: **Heroku**, **AWS Lambda** veya **Google Cloud Functions** gibi platformlar [35].

Kod İncelemesi: Hostlama ve Çalıştırma

a. Flask Uygulamasını Çalıştırma:

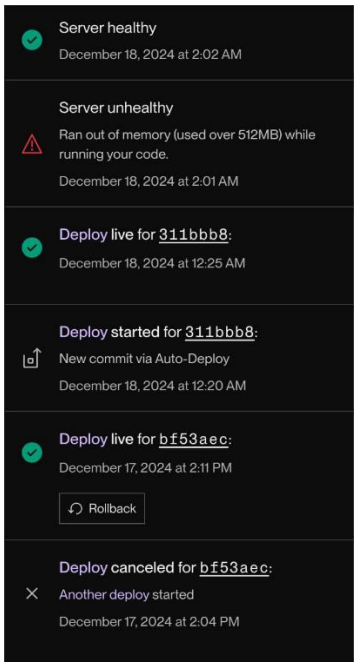
```
if __name__ == '__main__':
    threading.Thread(target=print_memory_usage, daemon=True).start()
    print("Starting Flask server...")
    app.run(host='0.0.0.0', port=5000, debug=False)
```

Açıklama:

- app.run: Flask uygulamasını belirtilen IP adresi ve port üzerinde başlatır.
    - host='0.0.0.0': Uygulamanın, bağlı tüm IP adreslerinden erişilebilir olmasını sağlar.
    - port=5000: Flask uygulaması, varsayılan olarak bu portta çalışır.
- debug=False: Debug modu kapatılmıştır. Bu, üretim ortamında güvenlik için gereklidir.
- Satır 2'deki kod uygulama çalışırken bellek kullanımını izler ve geliştiricilere optimizasyon fırsatları sunar.

b. Bulutta Çalıştırma Süreci:

1. Render.com üzerinde yeni bir web servisi oluşturulur.
2. Proje deposu GitHub veya manuel olarak yüklenir.
3. Port 5000 olarak ayarlanır ve başlangıç komutu belirtilir.
4. Uygulama dağıtıldıktan sonra, URL üzerinden API uç noktalarına erişim sağlanır.



Şekil 4.3. Render.com üzerinde sunucunun konsol geçmişi

4.4.4 Sonuç ve Öneriler

Bu backend yapısı, görsel işleme ve analiz işlemleri için optimize edilmiştir. Flask'ın esnekliği ve TensorFlow Lite'ın düşük maliyetli çözümü sayesinde, desen ve renk analizi başarılı bir şekilde gerçekleştirilmiştir. Gelecekte, paralel işlem teknikleri ve daha büyük veri kümeleriyle performans artırılabilir.

4.5 Frontend Tasarımı

4.5.1 Amaç ve Genel Yapı

Bu bölümde, kullanıcıların mobil uygulama üzerinden kamera ile giysi tarama, renk ve desen bilgilerini alma işlevlerini gerçekleştirebileceği frontend tasarımını detaylı olarak ele alıyoruz. Kullanılan React Native teknolojisi sayesinde uygulama hem Android hem de iOS platformlarında çalışabilir hale gelecektir.

Kullanılan Kütüphaneler

Frontend geliştirmede, aşağıdaki kütüphaneler kullanılmıştır:

- react-native: Temel React Native bileşenleri ve yapısı [6].
- expo-camera: Kamera işlemleri ve yetkilendirme [24].
- expo-media-library: Fotoğrafları kaydetmek ve albümlerden erişim sağlamak [25].
- expo-speech: Metin okuma (text-to-speech) desteği [10].
- react-native-animatable: Görsel animasyonlar [36].
- axios: API ile iletişim [26].

- translate: Metin çeviri işlemleri .

Kodun doğru çalışması için aşağıdaki komutlarla kütüphaneleri yükleyin:

```
npm install react-native
npm install react-native-animatable
npm install axios
npm install expo-camera
npm install expo-media-library
npm install expo-speech
npm install translate
```

Uygulama Akışı

1. **Giriş Animasyonu ve Yetkilendirme:**
    - Uygulama ilk açıldığında, 3 saniyelik bir logo animasyonu oynatılır.
    - Kullanıcıdan kamera, mikrofon ve medya kütüphanesi izinleri talep edilir.
2. **Kamera Arayüzü:**
    - Kullanıcı kamerayı kullanarak fotoğraf çekebilir.
    - Kamera ayarları (flaş, zoom, şıkırıtı animasyonu vb.) değiştirilebilir.
3. **Görüntüler Arası Geçiş:**
    - Fotoğraf çekildikten sonra kaydedilir ve bir önizleme ekranında gösterilir.
    - İşlenmiş veriler (renk ve desen bilgileri) görüntülenir ve sesli olarak aktarılır.
4. **API Entegrasyonu:**
    - Fotoğraflar Flask backend'e gönderilir.
    - Gelen renk ve desen bilgileri çevirilerek ekrana yazılır ve sesli olarak ifade edilir.

4.5.2 Kod İncelemesi ve Yorumlaması

Komponentlerin Tanımlanması:

```
import React, { useState, useRef, useEffect } from 'react';
import { StyleSheet, Text, TouchableOpacity, View, Image, Alert } from 'react-native';
import * as Animatable from 'react-native-animatable';

import axios from 'axios';

import * as MediaLibrary from 'expo-media-library';
import { CameraView, useCameraPermissions, useMicrophonePermissions } from 'expo-camera
import * as Speech from 'expo-speech';

import translate from 'translate';
import tr_dict from './frontend/turkish_localization';
translate.engine = 'google';
```

Bu kısım, uygulamada kullanılan React ve React Native modüllerini, özellikle Expo'nun kamera ve medya kütüphanelerini yükler.

State ve Referans Tanımlamaları:

```
const [cameraPermission, requestCameraPermission] = useCameraPermissions();
const [microphonePermission, requestMicrophonePermission] = useMicrophonePermissions()
const [mediaLibraryPermision, requestMediaLibraryPermission] = MediaLibrary.usePermiss
const [cameraProps, setCameraProps] = useState({
  zoom: 0,
  facing: 'back',
  flash: 'off',
  animateShutter: false,
  enableTorch: false
});
const [image, setImage] = useState(null);
const [processedImage, setProcessedImage] = useState(null);
const [previousImage, setPreviousImage] = useState(null);

const cameraRef = useRef(null);

const [colorData, setColorData] = useState(null);
const [showIntro, setShowIntro] = useState(true);
```

- **cameraPermission, microphonePermission, mediaLibraryPermission:** Kullanıcıdan gerekli izinlerin alınmasını kontrol eder.
- **cameraProps:** Kamera ayarlarını (zoom, flaş, vb.) takip eder.
- **image, processedImage:** Kamera ile çekilen fotoğrafları ve işlenmiş sonuçları saklar.
- **showIntro:** Animasyonlu giriş ekranını kontrol eder.

Giriş Animasyonu ve Yetkilendirme:

```
useEffect(() => {
  const handleIntroAndPermissions = async () => {
    // Simulate animation duration
    await new Promise(resolve => setTimeout(resolve, 3000)); // 3-second intro

    // Request permissions
    await requestCameraPermission();
    await requestMediaLibraryPermission();
    await requestMicrophonePermission();

    // Update state
    setShowIntro(false);
  };

  handleIntroAndPermissions();
}, []);
```

Bu kısım, uygulama açılışında animasyon oynatılmasını ve kamera, mikrofon ile medya kütüphanesi izinlerinin talep edilmesini sağlar.

Fotoğraf Çekme Fonksiyonu:

```
const takePicture = async () => {
  if (cameraRef.current) {
    try {
      const picture = await cameraRef.current.takePictureAsync();
      console.log("Picture URI:", picture.uri); // Debugging için URI'yi yazdır
      setImage(picture.uri);

      // Görüntüyü işleme gönder
      processImage(picture.uri); // Fonksiyona URI gönderiyoruz
    } catch (err) {
      console.log("Error while taking picture!", err);
      Speech.speak('Fotoğraf çekerken bir hata oluştu', {
        language: "tr-TR",
      });
    }
  }
};
```

Bu fonksiyon, kullanıcının kamera aracılığıyla fotoğraf çekmesini sağlar ve resmi backend'e gönderir.

Fotoğraf İşleme ve Seslendirme:

```
const processImage = async (imageUri) => {
  if (imageUri) {
    try {
      const formData = new FormData();
      formData.append("image", {
        uri: imageUri,
        type: "image/jpeg",
        name: "photo.jpg",
      });

      const response = await axios.post(
        "https://backend-endpoint.com/process-image",
        formData,
        {
          headers: {
            "Content-Type": "multipart/form-data",
          },
        }
      );

      if (response.status === 200) {
        const { color, pattern } = response.data;
```

```
        // Gelen bilgileri seslendirme ve ekrana yazdırma
        const message = `Giysinin rengi ${color}, deseni ise ${pattern}.`;
        Speech.speak(message, {
          language: "tr-TR",
        });

        setColorData({ color, pattern });

        Alert.alert("Görüntü İşleme Başarılı", message, [{ text: "Tamam" }]);
      } else {
        throw new Error("İşleme sırasında beklenmedik bir hata oluştu.");
      }

      // İşlenmiş resmi ve verileri kaydet
      MediaLibrary.saveToLibraryAsync(imageUri).then(() => {
        console.log("Fotoğraf kütüphaneye kaydedildi");
      }).catch((err) => {
        console.error("Fotoğraf kaydedilemedi", err);
        Speech.speak("Fotoğraf kaydedilemedi", { language: "tr-TR" });
      });
    } catch (err) {
      console.error("Error while processing image!", err);
      Speech.speak("Görüntü işleme sırasında bir hata oluştu", {
        language: "tr-TR",
      });
    }
  }
};
```

- **FormData Kullanımı**: Fotoğraf FormData formatında hazırlanarak backend'e gönderiliyor.

- **axios.post**: Flask tabanlı backend ile iletişim sağlanıyor. Backend'den gelen renk ve desen bilgileri işleniyor.
- **Seslendirme ve Geri Bildirim**: Gelen bilgiler sesli olarak aktarılıyor ve aynı zamanda ekranda bir uyarı olarak gösteriliyor.
- **Medya Kütüphanesi Kaydı**: Fotoğraf başarıyla işlendikten sonra cihazın medya kütüphanesine kaydediliyor.

Uygulama Stilleri

Bu stiller, giriş animasyonu, kamera ekranı, kontrol butonları ve diğer arayüz elemanlarının düzenini sağlar.

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#ffffff",
    justifyContent: "center",
    alignItems: "center",
  },
  logoText: {
    fontSize: 32,
    fontWeight: "bold",
    color: "#333333",
  },
  camera: {
    flex: 1,
    width: "100%",
    justifyContent: "flex-end",
    alignItems: "center",
  },
  controls: {
    position: "absolute",
    bottom: 20,
    flexDirection: "row",
    justifyContent: "space-evenly",
    width: "100%",
  },
  shutterButton: {
    width: 70,
    height: 70,
    borderRadius: 35,
    backgroundColor: "#ff5722",
    justifyContent: "center",
    alignItems: "center",
  },
});
```

4.5.3 Sonuç

Frontend tasarımı, kullanıcı dostu bir deneyim sağlamak için React Native ve Expo kütüphanelerini [8]kullanarak modern bir mobil uygulama geliştirme yaklaşımlarını benimser. Kullanıcı, giysinin rengini ve desenini kolayca öğrenebilir ve bu bilgileri sesli geri bildirim olarak alabilir. Hata durumları için sesli uyarılar ve görsel bildirimler entegre edilmiştir.

4.6 Android Kurulumu

4.6.1 Expo Projesi Hazırlama

Expo projesinin .aab dosyası oluşturabilmesi için aşağıdaki adımlar izlenmelidir:

Gerekli Yüklemeler

EAS CLI, Expo projelerini build etmek (örneğin .apk veya .aab oluşturmak) ve Expo'nun gelişmiş özelliklerini kullanmak için gerekli bir aracıdır. Yükleme komutu:

```
npm install -g eas-cli
```

Expo Projesini Yapılandırma

EAS CLI kullanılabilmesi için proje ayarlarının yapılandırılması gerekir. Bunun için aşağıdaki komut çalıştırılır:

```
eas build:configure
```

Bu komut, projenin gerekli eas.json yapılandırma dosyasını oluşturur ve Expo projesini build süreçlerine hazır hale getirir.

4.6.2 Android App Bundle (.aab) Dosyası Oluşturma

Android AAB dosyasını oluşturmak için aşağıdaki komut kullanılır:

```
eas build --platform android
```

Bu komut, Android için bir build işlemi başlatır ve .aab dosyası oluşturur. Build işlemi tamamlandığında, indirilebilir bir URL oluşur:

```
Android app: https://expo.dev/artifacts/eas/"...".aab
```

Build Modları

- Production Mode (Varsayılan): Optimize edilmiş bir .aab dosyası oluşturur ve Play Store'a yüklenebilir.

- Development Mode: Test amacıyla kullanılabilir.

4.6.3 AAB Dosyasını APK Dosyasına Dönüştürme

Bundletool, Android uygulamalarını .aab formatından .apks formatına dönüştürmek için kullanılır. Bu aracı kullanarak farklı cihaz yapılandırmalarına uygun APK'lar oluşturulabilir.

Gereksinimler

1. Java: Java yüklü ve PATH'e eklenmiş olmalıdır.
2. Bundletool: Bundletool'un en son sürümü resmi web sitesinden indirilmelidir.
3. .aab Dosyası: Expo veya Android Studio kullanılarak .aab dosyası oluşturulmuş olmalıdır.

APK Seti (APKS) Oluşturma

Aşağıdaki komut ile bir .aab dosyası .apks formatına dönüştürülür:

```
java -jar bundletool.jar build-apks --bundle='dosya_ismi'.aab --output='cikti_dosyasi'.apks --mode=universal
```

- --bundle='dosya_ismi'.aab: APK oluşturulacak Android App Bundle dosyası.
- --output='cikti_dosyasi'.apks: Çıkış dosyasının adı.
- --mode=universal: Her cihazda çalışabilir tek bir APK oluşturur.

APK'ları Çıkarma

Oluşturulan .apks dosyasından APK'ları çıkarmak için aşağıdaki komut kullanılır:

```
unzip your_app.apks -d output_folder
```

Bu komut, tüm APK'ları belirtilen output_folder dizinine çıkarır.

Cihaza Yükleme

```
java -jar bundletool.jar install-apks --apks=your_app.apks
```

- --apks=your_app.apks: Yüklenecek APK Seti dosyası.

5. TEST VE DOĞRULAMA

5.1 Model Eğitimi Performans Değerlendirme

Bu bölümde, modelin performansını değerlendireceğiz. İlk olarak, sağlanan *classification report* (Şekil 5.1) verilerine dayanarak birkaç önemli noktaya değinebiliriz [37]:

Classification Report

	Precision	Recall	F1-Score	Support
Checkered	0.88	0.86	0.87	200
Dotted	0.87	0.71	0.79	200
Floral	0.74	0.89	0.81	200
Solid	0.94	0.92	0.93	200
Striped	0.86	0.89	0.87	200
Zigzag	0.74	0.74	0.74	200
Accuracy			0.83	1200
Macro avg	0.84	0.83	0.83	1200
Weighted avg	0.84	0.83	0.83	1200

Şekil 5.1 Eğitilmiş modelin test sonuçları

1. **Precision (Kesinlik)**: Modelin doğru tahmin ettiği pozitif sınıfların oranını gösterir. Örneğin, "checkered" sınıfı için precision değeri 0.88, bu da modelin tahmin ettiği "checkered" sınıflarının %88'inin gerçekten doğru olduğunu ifade eder. Genellikle yüksek precision, modelin yanlış pozitifleri az yaptığı anlamına gelir.
2. **Recall (Duyarlılık)**: Modelin doğru bir şekilde tüm gerçek pozitif örnekleri sınıflandırabilme yeteneğini gösterir. "Floral" sınıfı için recall değeri 0.89, bu da modelin gerçekten "floral" olan örneklerin %89'unu doğru şekilde sınıflandırabildiği anlamına gelir. Yüksek recall, modelin yanlış negatifleri az yaptığı anlamına gelir.
3. **F1-Score**: Precision ve recall değerlerinin harmonik ortalamasıdır ve modelin genel başarısını gösterir. Yüksek F1-score, genellikle modelin her iki metriği de iyi dengelediğini ifade eder. Örneğin, "solid" sınıfı için F1-score değeri 0.93, modelin bu sınıfı oldukça iyi sınıflandırdığı anlamına gelir.
4. **Accuracy (Doğruluk)**: Modelin genel doğruluğunu gösterir, burada %83, modelin tüm test örneklerinin %83'ünü doğru sınıflandırdığını belirtir.

5. **Macro Avg ve Weighted Avg**:

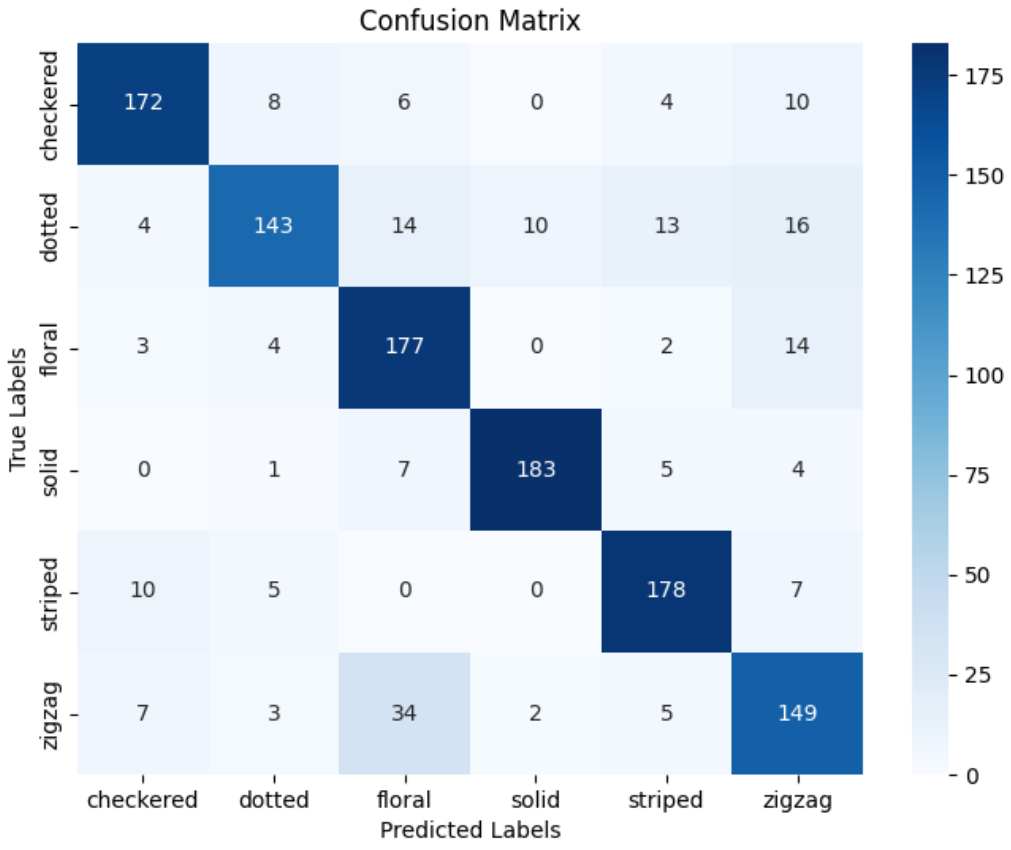
- **Macro average**: Her sınıfın performansını eşit bir şekilde değerlendirir, burada precision, recall ve F1-score değerlerinin ortalamalarını verir.
- **Weighted average**: Her sınıfın örnek sayısına göre ağırlıklı ortalamalarını hesaplar.

Öne Çıkan Değerler:

- **"Solid" sınıfı**: Yüksek precision (0.94), recall (0.92) ve F1-score (0.93) değerleriyle modelin bu sınıfı oldukça doğru tahmin ettiğini gösteriyor.
- **"Dotted" ve "Zigzag" sınıfları**: Bu sınıflar daha düşük recall ve F1-score değerlerine sahip, bu da modelin bu sınıflarda daha fazla hata yaptığı anlamına gelebilir. Özellikle "dotted" sınıfı için recall değeri 0.71, bu sınıfın bazı örneklerini doğru tanımadığını gösteriyor.

Test ve Doğrulama Kısmı:

Raporun bu kısmında **Confusion Matrix** (Şekil 5.2) ve **ROC Curve** (Şekil 5.3) görselleri yer alacak, bu da modelin daha detaylı bir şekilde performansını anlamamıza yardımcı olacak.



Şekil 5.2. Confusion Matrix tablosu

5.1.1 Confusion Matrix Yorumlanması

Modelin performansı, genellikle doğruluğunun yanı sıra, sınıflar arasındaki karışıklıkları da içerir. Aşağıda, confusion matrix'teki bazı anormal ve dikkat çekici durumlar açıklanmıştır [38]:

Zigzag (True) ve Floral (Predicted) Çakışması: Bu durumda, 34 örnek doğru label'ı *Zigzag* olan ancak *Floral* olarak tahmin edilen sınıflarla çakışmaktadır. Bu, modelin *Zigzag* sınıfını *Floral* sınıfına doğru şekilde ayırt edemediğini gösteriyor ve *Zigzag* sınıfındaki örneklerin bazıları *Floral* sınıfına benzer özelliklere sahip olabilir.

Dotted (True) ve Diğer Sınıfların Çakışması: *Dotted* sınıfında, modelin tahminlerinde çeşitli hatalar gözlemlenmektedir:

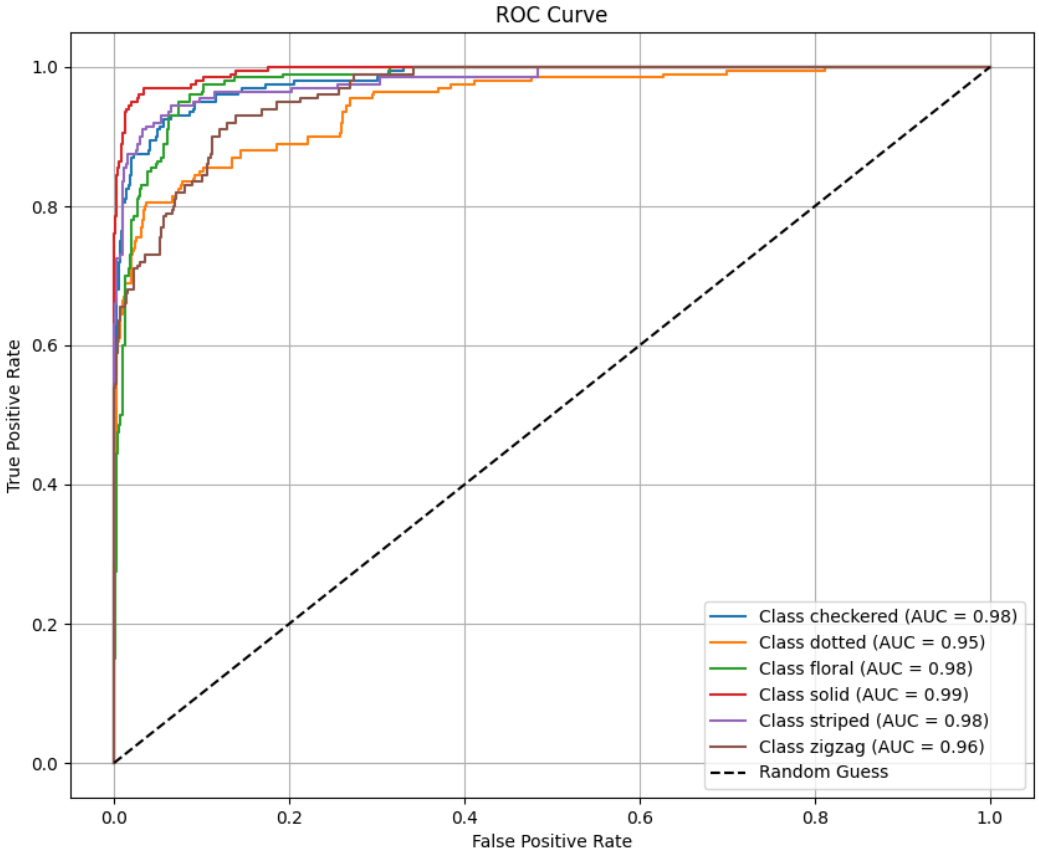
- *Floral* ile 14 örnek çakışıyor.
- *Solid* ile 10 örnek çakışıyor.
- *Striped* ile 13 örnek çakışıyor.
- *Zigzag* ile 16 örnek çakışıyor.

Bu hatalar, *Dotted* sınıfındaki örneklerin *Floral*, *Solid*, *Striped* ve *Zigzag* sınıflarına benzer özellikler taşıdığı anlamına gelebilir. Modelin bu sınıfları ayırt etmekte zorlandığı ve daha fazla örnek üzerinde bu karışıklıkları çözmek için ek eğitim verilmesi gerektiği söylenebilir.

Zigzag (Predicted) ve Diğer Sınıfların Çakışması: *Predicted Zigzag* etiketli bazı örnekler, doğru sınıflar olan *Floral*, *Dotted* ve *Checkered* sınıflarıyla çakışmaktadır:

- *Floral* ile 14 örnek,
- *Dotted* ile 16 örnek,
- *Checkered* ile 10 örnek çakışıyor.

Bu hatalar, modelin *Zigzag* sınıfını doğru şekilde ayırt edemediğini ve diğer sınıfların benzer özelliklerini taşıyan örnekleri *Zigzag* olarak tahmin ettiğini gösteriyor. Özellikle *Floral* ve *Dotted* sınıflarıyla yaşanan hatalar, bu sınıfların belirli özelliklerinin *Zigzag* sınıfına benzer olduğunu ve modelin bu benzerlikleri doğru bir şekilde analiz edemediğini işaret eder.



Şekil 5.3. ROC Eğrisi grafiği

5.1.2 ROC Eğrisinin Yorumlanması

ROC (Receiver Operating Characteristic) eğrisini yorumlarken, **True Positive Rate (TPR)** ve **False Positive Rate (FPR)** değerleri arasındaki ilişkiyi anlamak önemlidir. TPR, modelin doğru sınıflandırdığı pozitif örneklerin oranını gösterirken (duyarlılık), FPR, yanlış sınıflandırılmış negatif örneklerin oranını gösterir [39].

Verdiğiniz değerlere göre her bir sınıf için TPR (True Positive Rate) sırasıyla:

1. **Checkered**: 0.98
2. **Dotted**: 0.95
3. **Floral**: 0.98
4. **Solid**: 0.99
5. **Striped**: 0.98
6. **Zigzag**: 0.96

Grafiğe göre:

- **Checkered, Floral, Solid, Striped**: Bu sınıfların TPR değerleri oldukça yüksek (0.98, 0.98, 0.99, 0.98). Yüksek bir TPR değeri, modelin bu sınıfları doğru şekilde ayırt ettiğini ve doğru tahminler yaptığı anlamına gelir. Bu sınıflar için modelin performansı oldukça iyi.
- **Dotted ve Zigzag**: Bu sınıfların TPR değerleri sırasıyla 0.95 ve 0.96. Bu da modelin, özellikle *Dotted* ve *Zigzag* sınıflarında biraz daha düşük performans sergilediğini gösteriyor. Bunun anlamı, modelin bu sınıfları doğru şekilde tanımlamakta diğer sınıflara göre biraz daha zorlandığıdır.

5.1.3 Genel Yorum

Grafikte yüksek TPR değerleri, modelin çoğu sınıfı doğru tahmin ettiğini ve doğru sınıflama oranının yüksek olduğunu gösteriyor. *Solid* sınıfının 0.99'luk TPR değeri, bu sınıf için modelin mükemmel derecede iyi bir performans sergilediğini işaret eder. *Dotted* ve *Zigzag* sınıfları için ise TPR değerlerinin diğer sınıflara göre biraz daha düşük olması, bu sınıfların model tarafından daha zor tahmin edilen sınıflar olduğunu gösteriyor. Ancak, yine de bu değerlerin yüksek olması, modelin bu sınıflar için de yeterince iyi performans sergilediğini gösteriyor.

5.2 Model Eğitiminin Test Sonuçları

Bu kısımda eğitim sonuçlarından ve grafiklerinden bağımsız olarak modelin gerçek hayatta doğruluğunu test etmek amacıyla 30 fotoğraf üzerinde maksimum başarıyı sağlamak için tekrar tekrar eğitip mükemmel sonucu yakalamaya çalıştığımız kısımdayız. Desen fotoğrafları kolajda (Şekil 5.4) soldan sağa doğru numaralandırılmış ve ***4.3 Eğitilen Modelin Test Edilmesi*** kısmında verdiğimiz kod ile sırasıyla sonuçları alınmıştır.

Şekil 5.4. Test edilen desen fotoğrafları

Sonuçlar:

1. so-Pattern: solid, Confidence: 99.98%
2. do-Pattern: dotted, Confidence: 100.00%
3. str-Pattern: striped, Confidence: 98.65%
4. zig-Pattern: zigzag, Confidence: 99.99%
5. che-Pattern: checkered, Confidence: 98.26%
6. flo-Pattern: floral, Confidence: 99.61%
7. flo-Pattern: floral, Confidence: 65.68%
8. so-Pattern: dotted, Confidence: 64.95%
9. so-Pattern: solid, Confidence: 74.79%
10. so-Pattern: floral, Confidence: 53.02%
11. so-Pattern: solid, Confidence: 97.43%
12. che-Pattern: checkered, Confidence: 99.95%
13. che-Pattern: checkered, Confidence: 100.00%
14. che-Pattern: checkered, Confidence: 99.96%
15. flo-Pattern: floral, Confidence: 99.90%
16. flo-Pattern: floral, Confidence: 98.21%
17. flo-Pattern: floral, Confidence: 93.28%
18. flo-Pattern: floral, Confidence: 52.79%
19. do-Pattern: dotted, Confidence: 100.00%
20. do-Pattern: dotted, Confidence: 99.97%
21. do-Pattern: dotted, Confidence: 78.83%
22. do-Pattern: dotted, Confidence: 100.00%
23. str-Pattern: striped, Confidence: 100.00%
24. str-Pattern: striped, Confidence: 96.23%
25. str-Pattern: striped, Confidence: 97.00%
26. zig-Pattern: zigzag, Confidence: 99.97%
27. zig-Pattern: zigzag, Confidence: 79.60%
28. zig-Pattern: zigzag, Confidence: 72.63%
29. zig-Pattern: zigzag, Confidence: 99.87%
30. zig-Pattern: zigzag, Confidence: 98.02%

Özet olarak:

Hatalı sadece 2 tahmin vardı:

- 8. so-Pattern: dotted, Confidence: 64.95%
- 10. so-Pattern: floral, Confidence: 53.02%

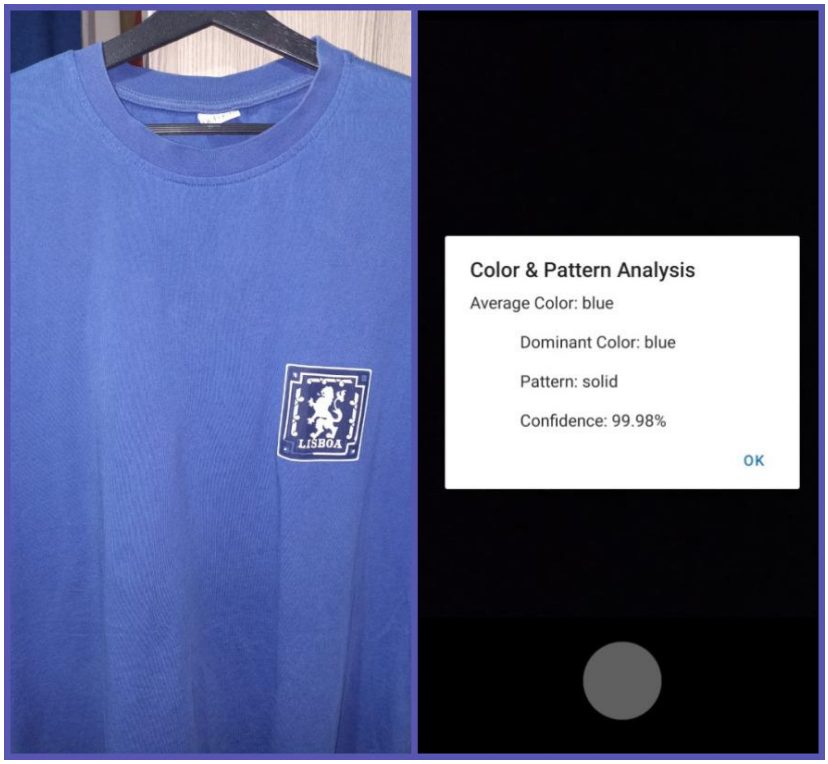
İkisi de *solid* sınıfına ait görsellerdi ve sırasıyla *dotted* ve *floral* olarak tahmin edildiler. Modelin eğitimine başlandığı ilk zamandan bu yana test edilen bu görsellerde *solid* sınıfı her zaman ve bütün örneklerde yanlış tahmin ediliyordu. Bunu önlemek amacıyla çok katmanlı bir yapı daha kurmak yerine bu sınıfın verilerini, veri setinde optimize ettik ve diğer sınıflara göre daha fazla eğitilmesini sağladık. Bu da ***5.1 Model Eğitimi Performans Değerlendirme*** kısmında *solid* sınıfının eğitim sonuçlarının diğerlerine göre yüksek olduğunu açıklıyordu. Bunca şeye rağmen ***5.1*** kısmında Confusion Matrix tablosunda göründüğü üzere *solid* sınıfı *floral* ile çok sayıda çakışma yaşıyor, *dotted* sınıfı ile de kısmi de olsa çakışmalar yaşanıyordu. Bu sonuçlar da bunun göstergesi oldu. Bunu önlemek amacıyla CNN modelimizde çok

katmanlı bir yapı oluşturabilir, *solid* sınıfını diğer sınıflardan ayıran bir yapı kurabilirdik, ama uygulama kısmında çok fazla hata almadığımız için bu yola gitmedik.

5.3 Uygulama Tarafı Test Sonuçları

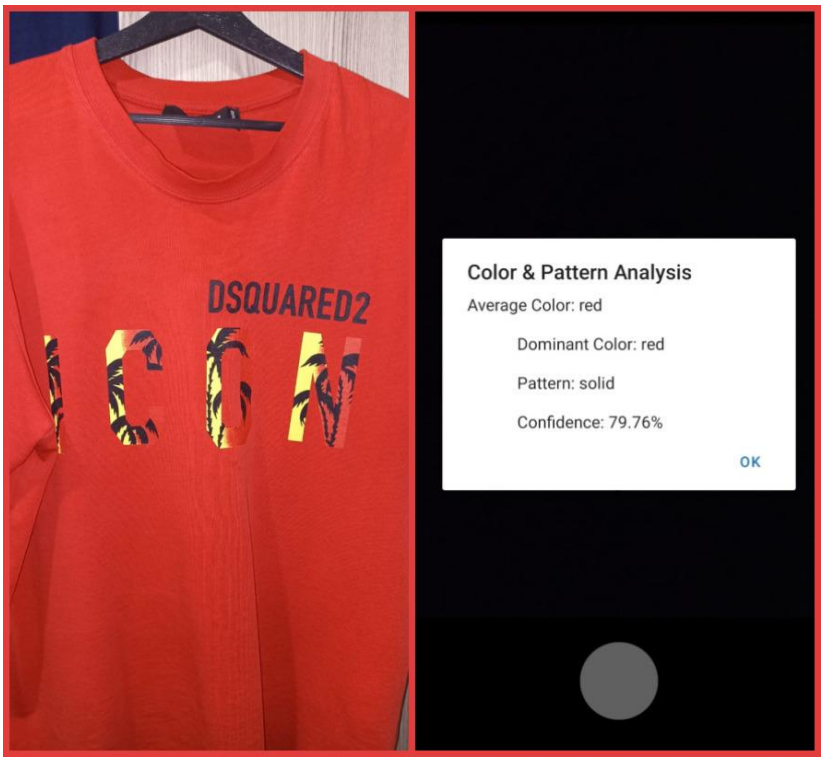
Bu kısımda **CNN (Convolutional Neural Network)** ile kurmuş olduğumuz desen analizi modeli ile ***3.3.2 Desen Analizi*** kısmında anlattığımız, ***4.4.2*** kısmında kod işleyişini gösterdiğimiz **K-Means kümeleme algoritması**, **ortalama renk hesaplama** ve **ColorNamer** algoritmaları ve kütüphaneleriyle kurulmuş uygulamamızın çalışma testlerini ve sonuçlarını paylaşacağız.

Uygulamamız görme engelliler için tasarlandığından sadece sesli komutlarla sonuçları bize vermektedir. Ancak eski bir sürümünde yazılı olarak verildiğinden, bu eski sürüm ile sonuç görsellerini sırasıyla; çekilen fotoğrafı ve tahmin edilen desen, ortalama ve dominant renk sonuçlarını cep telefonundan ekran görüntüsü olarak paylaşacağız.



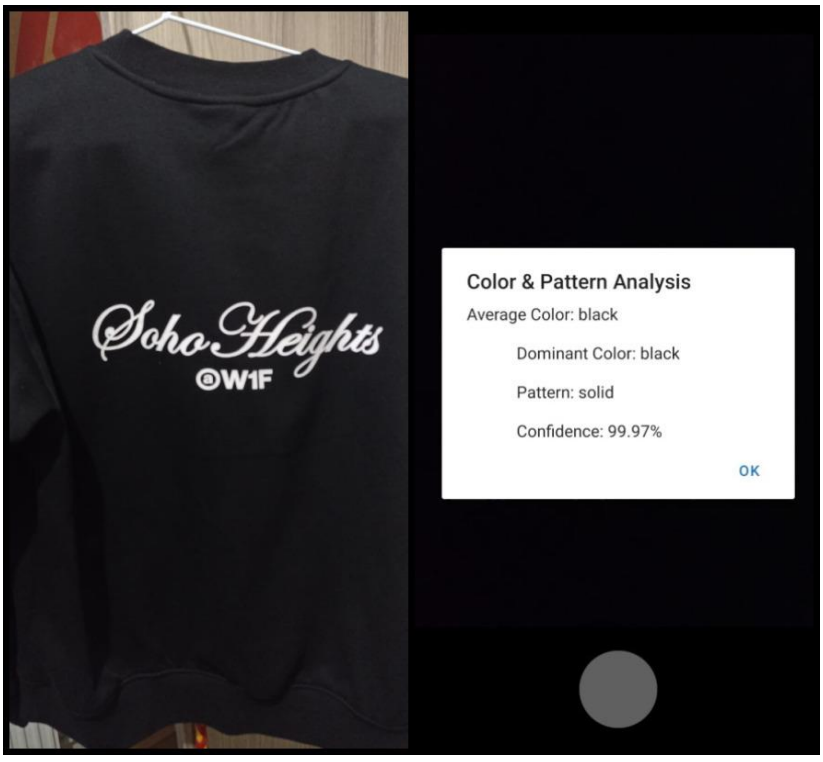
Şekil 5.5. Mavi düz desenli tişört ve test sonuçları

- Şekil 5.5'te mavi renkli bir tişörtün fotoğrafı çekilmiş ve uygulama üzerinde ortalama olarak mavi renge, baskın olarak da mavi renge sahip olup %99.98 oranla *solid* (düz) desen sınıfına ait olduğu bildirilmiş. Bu da büyük oranla doğru bir sonuç verdiğini bize söylüyor.



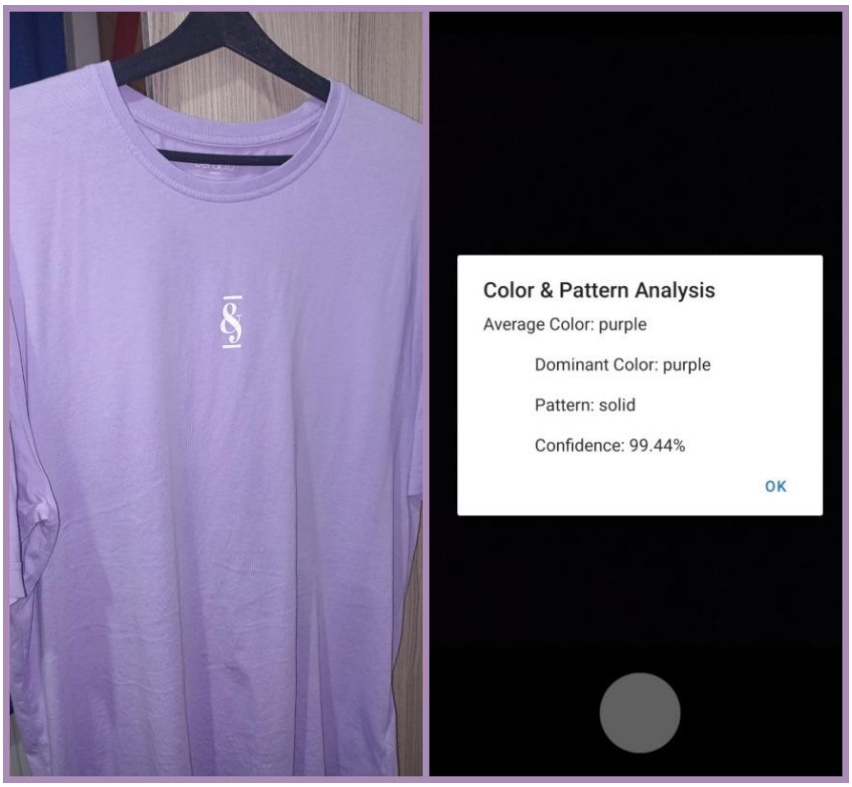
Şekil 5.6. Kırmızı düz desenli tişört ve test sonuçları

- Şekil 5.6'da kırmızı renkli bir tişörtün fotoğrafı çekilmiş ve uygulama üzerinde ortalama olarak kırmızı renge, baskın olarak da kırmızı renge sahip olup %79.76 oranla *solid* (düz) desen sınıfına ait olduğu bildirimiş. Bu da tişörtün üzerinde bulunan baskı dolayısıyla biraz yanılsa da doğru bir sonuç verdiğini bize söylüyor.



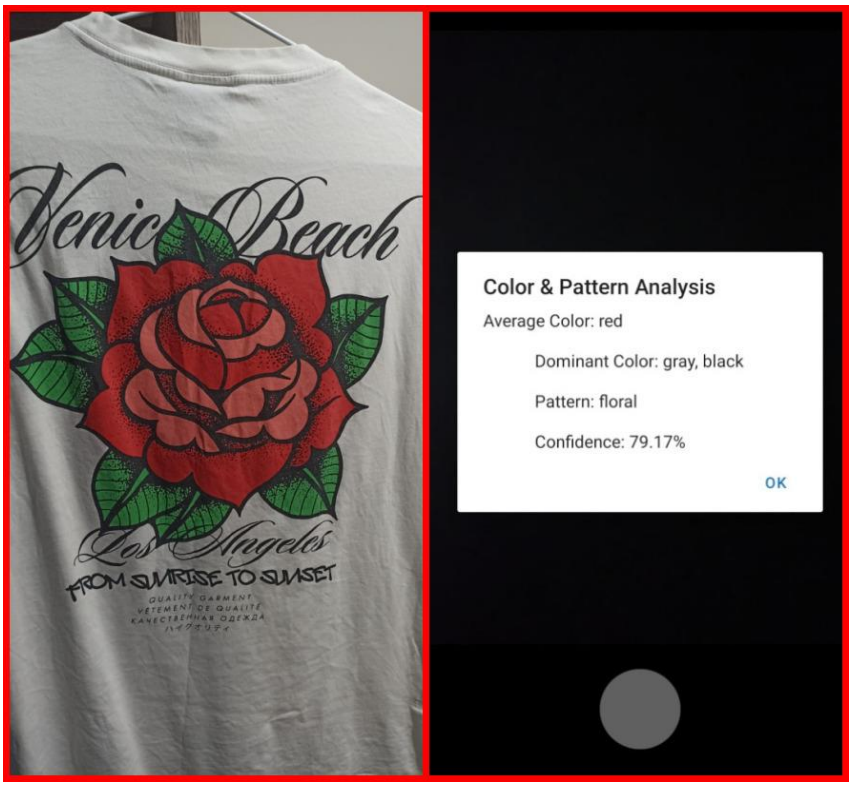
Şekil 5.7. Siyah düz desenli kazak ve test sonuçları

- Şekil 5.7’de siyah desensiz bir tişörtün fotoğrafı çekilmiş ve uygulama üzerinde ortalama olarak siyah renge, baskın olarak da siyah renge sahip olup %99.97 oranla *solid* (düz) desen sınıfına ait olduğu bildirilmiş. Bu da çok büyük bir oranda doğru sonuç verildiğini bize gösteriyor.



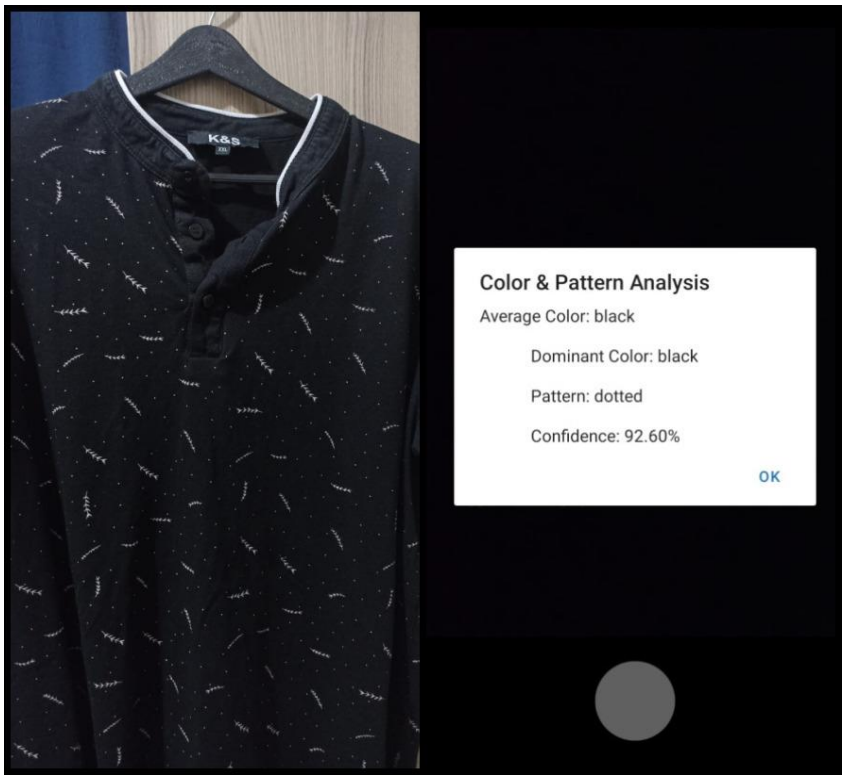
Şekil 5.8. Mor düz desenli tişört ve test sonuçları

- Şekil 5.8’de mor desensiz bir tişörtün fotoğrafı çekilmiş ve uygulama üzerinde ortalama olarak mor renge, baskın olarak da mor renge sahip olup %99.44 oranla *solid* (düz) desen sınıfına ait olduğu bildirilmiş. Bu da uygulamanın çok büyük bir oranda doğru sonuç verdiğini bize gösteriyor.



Şekil 5.9. Kırmızı çiçek desenli beyaz tişört ve test sonuçları

- Şekil 5.9'da kırmızı çiçek desenli ama özünde beyaz bir tişörtün fotoğrafı çekilmiş ve uygulama üzerinde ortalama olarak kırmızı renge, baskın olarak da siyah ve gri renklerine sahip olup %79.17 oranla *floral* (çiçekli) desen sınıfına ait olduğu bildirilmiş. Bu da tişörtün baskısında bulunan resmin desen olarak alındığı, kalanı desensiz olduğu için de az oran verdiğini bize gösteriyor. Renk kısmında da desenin rengini ortalama renk olarak almış, geri kalan tişört üzerindeki yazıları ve tişörtün kendi rengini gri ve siyah olarak söylemiş. Gri kısmı hatalı gibi de dursa özellikle fotoğrafa baktığımızda biz beyaz olduğunu ne kadar bilsek de alt kısmının aşırı gölgeden gri gibi gözüktüğünü ve siyaha yaklaştığını görmekteyiz. Bu da özellikle gölgeli, farklı renk ışıklı mekanlarda ve kamera yapay zekasının renk manipülasyonuyla çekilmiş fotoğraflarda modelin yanlış sonuçlar da elde edebileceğini göstermektedir.



Şekil 5.10. Siyah buğday desenli polo tişört ve test sonuçları

- Şekil 5.10'da beyaz buğday desenli ama özünde siyah bir polo tişörtün fotoğrafı çekilmiş ve uygulama üzerinde ortalama olarak siyah renge, baskın olarak da siyah rengine sahip olup %92.60 oranla *dotted* (benekli) desen sınıfına ait olduğu bildirilmiş. Bu da tişörtün üzerinde bulunan buğday desenlerini benekli olarak algıladığını renk kısmında da hatasız davrandığını bize göstermiş. Buğday deseni sınıflandırılırken en yakın olarak benekli desenine sınıflandırılması modelin hatalı olduğunu göstermez. Beklediğimiz sınıflandırma zaten budur.

6. SONUÇ

Bu proje, görme engelli ve renk körü bireylerin bağımsız bir şekilde kıyafet seçimi yapmalarına yardımcı olmak amacıyla tasarlanmış, yenilikçi bir mobil uygulamayı başarıyla geliştirmiştir. Uygulama, kıyafetlerin desen ve renk özelliklerini analiz ederek bu bilgileri sesli geri bildirim yoluyla kullanıcıya sunmaktadır. Böylece, görme engelli bireyler için erişilebilirlik ve günlük yaşamda kolaylık sağlama noktasında önemli bir çözüm ortaya koyulmuştur.

Model testleri ve uygulama doğrulama süreçleri, sistemin yüksek doğruluk oranıyla çalıştığını ve hedef kullanıcı kitlesine fayda sağlayacak şekilde optimize edildiğini göstermektedir. Özellikle "*solid*" sınıfında elde edilen yüksek performans, veri artırma ve model

optimizasyonu tekniklerinin başarıyla uygulandığını kanıtlamaktadır. Bununla birlikte, "*dotted*" ve "*zigzag*" sınıflarında gözlemlenen hafif performans düşüşleri, bu alanlarda daha fazla veri ve model iyileştirmesine ihtiyaç duyulduğunu işaret etmektedir. Ancak genel olarak, uygulamanın farklı desen ve renk sınıflarını ayırt etme yeteneği, pratik kullanımda yeterli bir doğruluk seviyesindedir.

Gelecekteki Geliştirme Olanakları

Projenin gelecekte daha geniş bir kullanıcı kitlesine hitap etmesi ve daha kapsamlı işlevler sunması için aşağıdaki geliştirme olanakları önerilmektedir:

1. **Veri Çeşitliliğinin Artırılması ve Özgün Veri Setleri Oluşturulması:**

- Modelin performansını daha da geliştirmek için farklı coğrafi bölgelerden, çeşitli kültürel kıyafet desenlerini kapsayan daha geniş bir veri seti ile çalışılabilir.
- Özgün desen setleri ile veri seti, kullanıcıların gerçek yaşam koşullarında çektikleri fotoğraflarla zenginleştirilerek modelin genelleme yeteneği artırılabilir. (Pinterest gibi uygulamalardan elde edilebilir.)

2. **Kullanıcı Deneyimini İyileştiren Özellikler Eklenmesi:**

- Görme engelli bireyler için sesli komut desteği geliştirilebilir; kullanıcılar uygulamayı yalnızca sesle kontrol ederek fotoğraf çekme ve sonuç alma işlemlerini gerçekleştirebilir.
- Renk körü bireyler için özel ayarlar eklenerek, renklerin algılanmasını destekleyecek şekilde özelleştirilmiş açıklamalar sağlanabilir (örneğin, “Kırmızı renk tonu, maviye yakın” gibi detaylar).

3. **Ek Görsel Özelliklerin Tanımlanması:**

- Kıyafetlerin kumaş türleri, dokuları (örneğin, pamuk, polyester) ve kalınlıkları gibi özellikler de analiz edilebilir. Bu sayede kullanıcılar sadece desen ve renk bilgisine değil, dokunarak tanımlayamayacağı kumaş türlerine de erişebilir.

4. **İleri Görüntü İşleme Tekniklerinin Entegrasyonu:**

- Daha gelişmiş görüntü işleme algoritmaları, kullanıcıların düşük ışık koşullarında çektiği fotoğrafların kalitesini artırabilir.
- Derin öğrenme modellerine daha fazla odaklanarak, gürültülü ve karmaşık arka planlara sahip fotoğraflarda desen ve renk tespitinin doğruluğu yükseltilebilir.

5. **Mobil ve Giyilebilir Teknoloji Entegrasyonu:**

- Akıllı saatler, akıllı lens ve gözlükler veya taşınabilir kameralar gibi cihazlarla entegrasyon sağlanarak daha pratik bir kullanım imkanı sunulabilir.
- Giyilebilir cihazlardan alınan görüntüler, uygulamanın backend’i ile işlenerek hızlı sonuç döndürme özelliği eklenebilir.

6. **Çoklu Platform Desteği:**

- Uygulama, sadece Android ile sınırlı kalmayıp iOS, HarmonyOS gibi diğer platformlar için de optimize edilebilir.

- Web tabanlı bir sürüm geliştirilerek, masaüstü cihazlardan da erişim imkanı sunulabilir (Trendyol, Hepsiburada gibi kıyafet satan uygulamalar için).

7. Bulut Bilişim ve Gerçek Zamanlı İşleme:

- Render.com gibi platformlar üzerinden daha büyük veri kümeleri işlenerek, hem doğruluk oranı artırılabilir hem de mobil cihazların işlem yükü hafifletilebilir.
- Gerçek zamanlı analizler sayesinde, kullanıcı bir kıyafeti kameranın önünde hareket ettirirken bile sonuçları anında görebilir.

8. Kişiselleştirme ve Yapay Zeka Destekli Öneri Sistemleri:

- Kullanıcıların sıklıkla tercih ettiği desen ve renk türlerine göre, uygulama kişiselleştirilmiş öneriler sunabilir. Örneğin, "Daha önce beğendiğiniz desenlere benzer bir desen" şeklinde öneriler yapılabilir.
- Yapay zeka destekli bir rehberlik sistemi eklenerek, kullanıcılar için giyim kombin önerileri oluşturulabilir.

9. Topluluk Geri Bildirimi ve Veri Paylaşımı:

- Kullanıcılar, uygulamanın analiz sonuçları hakkında geri bildirim vererek modelin daha fazla öğrenmesini sağlayabilir.
- Topluluk tarafından paylaşılan verilerle uygulama daha geniş bir bilgi ağına erişim sağlayabilir.

10. Erişilebilirlik ve Çok Dilli Destek:

- Uygulamanın farklı dillerde sesli geri bildirim verebilmesi için çok dilli destek geliştirilebilir.
- Erişilebilirlik standartlarına uygun olarak, uygulama işitme engelli bireyler için yazılı açıklamalarla desteklenebilir.

11. Daha Kapsamlı Eğitim ve Bilgilendirme Özellikleri:

- Kullanıcılara uygulama içi rehberlik sunularak, kıyafet seçimlerinde dikkat etmeleri gereken ipuçları sağlanabilir.
- Görme engelli bireylerin uygulamanın tüm özelliklerinden kolayca faydalanabilmesi için sesli öğretici bir mod geliştirilebilir.

12. Sosyal Sorumluluk ve Daha Geniş Kullanım Alanları:

- Uygulama, özel ihtiyaçları olan bireyler için geliştirilmiş farklı modlarla genişletilebilir. Örneğin, çocuklar için eğitici bir oyun modu veya yaşlı bireyler için basitleştirilmiş bir arayüz sunulabilir.
- Teknolojinin sosyal sorumluluk projelerinde kullanımını artırmak için bu uygulama farklı görsel tanıma projelerine entegre edilebilir.

Bu geliştirme olanakları, uygulamanın hem teknik açıdan daha güçlü hale gelmesini hem de kullanıcı kitlesini genişletmesini sağlayacaktır. Projenin sunduğu temel değerler korunurken, önerilen yeniliklerle birlikte toplumun daha geniş kesimlerine fayda sağlayacak bir teknoloji geliştirilmiş olacaktır.

KAYNAKLAR

[1] A. Medeiros, L. Stearns, L. Findlater, C. Chen, and J. Froehlich, “Recognizing Clothing Colors and Visual Textures Using a Finger-Mounted Camera: An Initial Investigation,” Dec. 2017, pp. 393–394. doi: 10.1145/3132525.3134805.

[2] “Clothing Style Recognition Method Based on Digital Image Processing,” *International Journal of Frontiers in Sociology*, vol. 3, no. 18, 2021, doi: 10.25236/ijfs.2021.031804.

[3] R. Sreemathy, M. P. Turuk, and S. Khurana, “Cloth Pattern Recognition Using Machine Learning and Neural Network,” 2022. [Online]. Available: https://mjsat.com.my/

[4] V. kumarR and Y. kumarR, “Convenient Clothing Pattern and Colour Recognition for People with Vision Impairment.”

[5] Kimberly. Keeton, *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*. USENIX Association, 2016.

[6] “Implementing OpenCV Image Processing in React Native: A Step-by-Step Guide | by Hasitha Sai Kutala | Nov, 2024 | Medium.” Accessed: Dec. 25, 2024. [Online]. Available: https://medium.com/@hasitha.kutala/implementing-opencv-image-processing-in-react-native-a-step-by-step-guide-124481f8416a

[7] “Building an Image Processing Flask App: A Step-by-Step Guide | by Balazs Kocsis | Medium.” Accessed: Dec. 25, 2024. [Online]. Available: https://medium.com/@balazskocsis/background-removal-and-image-captioning-write-a-flask-app-and-host-it-5c8ca4194542

[8] B. Moedano, “Expo Go vs Development Builds: Which should you use?” Accessed: Dec. 25, 2024. [Online]. Available: https://expo.dev/blog/expo-go-vs-development-builds

[9] “Deploy a Flask App on Render – Render Docs.” Accessed: Dec. 25, 2024. [Online]. Available: https://render.com/docs/deploy-flask

[10] “javascript - Follow words said in text to speech from expo-speech - Stack Overflow.” Accessed: Dec. 25, 2024. [Online]. Available: https://stackoverflow.com/questions/73037403/follow-words-said-in-text-to-speech-from-expo-speech

[11] “Top 5 Tools for Image Processing in Python | by Meng Li | Top Python Libraries | Nov, 2024 | Medium.” Accessed: Dec. 25, 2024. [Online]. Available: https://medium.com/top-python-libraries/top-5-tools-for-image-processing-in-python-407a894a7230

[12] “python - Remove transparency/alpha from any image using PIL - Stack Overflow.” Accessed: Dec. 25, 2024. [Online]. Available: https://stackoverflow.com/questions/35859140/remove-transparency-alpha-from-any-image-using-pil

[13] “Python Image Resize With Pillow and OpenCV.” Accessed: Dec. 25, 2024. [Online]. Available: https://cloudinary.com/guides/bulk-image-resize/python-image-resize-with-pillow-and-opencv

[14] “Image Preprocessing for Computer Vision: Normalization | by Elmar H. | Medium.” Accessed: Dec. 25, 2024. [Online]. Available: https://elmarhuseynv.medium.com/image-preprocessing-for-computer-vision-normalization-6c3b0138e3a9

[15] “Image Data Augmentation using TensorFlow | by Harisudhan.S | Medium.” Accessed: Dec. 25, 2024. [Online]. Available: https://medium.com/@speaktoharisudhan/image-data-augmentation-using-tensorflow-46d884f420f6

[16] “Image augmentation using openCV.” Accessed: Dec. 25, 2024. [Online]. Available: https://www.kaggle.com/code/ahmedabdelfattah20/image-augmentation-using-opencv

[17] “K-Means Algoritması. Hiç kitaplığınızı, aynı konulara ait… | by Şevket Ay | Deep Learning Türkiye | Medium.” Accessed: Dec. 25, 2024. [Online]. Available: https://medium.com/deep-learning-turkiye/k-means-algoritmas%C4%B1-b460620dd02a

[18] “How to find the average colour of an image in Python with OpenCV? - Stack Overflow.” Accessed: Dec. 25, 2024. [Online]. Available: https://stackoverflow.com/questions/43111029/how-to-find-the-average-colour-of-an-image-in-python-with-opencv

[19] “What Color Is This? | Stitch Fix Technology – Multithreaded.” Accessed: Dec. 25, 2024. [Online]. Available: https://multithreaded.stitchfix.com/blog/2020/09/02/what-color-is-this/

[20] D. K. Kim, “Color detection using Gaussian mixture model,” *J Theor Appl Inf Technol*, vol. 95, pp. 4313–4320, Sep. 2017.

[21] “Mean Shift Clustering: A Comprehensive Guide | DataCamp.” Accessed: Dec. 25, 2024. [Online]. Available: https://www.datacamp.com/tutorial/mean-shift-clustering

[22] “Hierarchical Clustering — Orange Visual Programming 3 documentation.” Accessed: Dec. 25, 2024. [Online]. Available: https://orange3.readthedocs.io/projects/orange-visual-programming/en/master/widgets/unsupervised/hierarchicalclustering.html

[23] “Image Processing Using CNN: A beginner’s guide | by Hrushabhjadhav | Medium.” Accessed: Dec. 25, 2024. [Online]. Available: https://medium.com/@hrushabhjadhav007/image-processing-using-cnn-a-beginners-guide-6b4c67a15b5f

[24] “Camera - Expo Documentation.” Accessed: Dec. 25, 2024. [Online]. Available: https://docs.expo.dev/versions/latest/sdk/camera/

[25] “MediaLibrary - Expo Documentation.” Accessed: Dec. 25, 2024. [Online]. Available: https://docs.expo.dev/versions/latest/sdk/media-library/

[26] “Getting Started | Axios Docs.” Accessed: Dec. 25, 2024. [Online]. Available: https://axios-http.com/docs/intro

[27] “On-device training in TensorFlow Lite — The TensorFlow Blog.” Accessed: Dec. 25, 2024. [Online]. Available: https://blog.tensorflow.org/2021/11/on-device-training-in-tensorflow-lite.html

[28] “Introduction To Numpy. In this Blog, I will be writing about… | by Shubhang Agrawal | Analytics Vidhya | Medium.” Accessed: Dec. 25, 2024. [Online]. Available: https://medium.com/analytics-vidhya/introduction-to-numpy-82321478e788

[29] “GitHub - stitchfix/colornamer: Given a color, return a hierarchy of names.” Accessed: Dec. 25, 2024. [Online]. Available: https://github.com/stitchfix/colornamer/?tab=readme-ov-file

[30] “GitHub - lstearns86/clothing-pattern-dataset: A large dataset containing images of clothing patterns.” Accessed: Dec. 25, 2024. [Online]. Available: https://github.com/lstearns86/clothing-pattern-dataset

[31] “Derin Öğrenmede Kayıp Fonksiyonları ve Optimizasyon Algoritmaları | by Ahmet Okan YILMAZ | Medium.” Accessed: Dec. 25, 2024. [Online]. Available: https://aoyilmaz.medium.com/derin-%C3%B6%C4%9Frenmede-kay%C4%B1p-fonksiyonlar%C4%B1-ve-optimizasyon-algoritmalar%C4%B1-8fc754e7a639

[32] “ReduceLROnPlateau — PyTorch 2.5 documentation.” Accessed: Dec. 25, 2024. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html

[33] “Python psutil kütüphanesi-1.” Accessed: Dec. 25, 2024. [Online]. Available: https://coderistan.blogspot.com/2017/07/python-psutil-kutuphanesi-1.html

[34] “bellek-profilleyici · PyPI.” Accessed: Dec. 25, 2024. [Online]. Available: https://pypi.org/project/memory-profiler/

[35] “Top 10 Render Alternatives & Competitors in 2024 | G2.” Accessed: Dec. 25, 2024. [Online]. Available: https://www.g2.com/products/render-render/competitors/alternatives

[36] “react-native-animatable - npm.” Accessed: Dec. 25, 2024. [Online]. Available: https://www.npmjs.com/package/react-native-animatable?activeTab=readme

[37] “Görüntü sınıflandırması | TensorFlow Core.” Accessed: Dec. 25, 2024. [Online]. Available: https://www.tensorflow.org/tutorials/images/classification?hl=tr

[38] “Confusion Matrix: How To Use It & Interpret Results [Examples].” Accessed: Dec. 25, 2024. [Online]. Available: https://www.v7labs.com/blog/confusion-matrix-guide

[39] D. K. McClish, “Analyzing a Portion of the ROC Curve,” *Medical Decision Making*, vol. 9, no. 3, pp. 190–195, 1989, doi: 10.1177/0272989X8900900307.