T.C.
DÜZCE ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

TENSORFLOW LITE İLE KÖPEK CİNSİ SINIFLANDIRMASI İÇİN ANDROID UYGULAMASI

MAHMUT CAN KURT

BİLGİSAYAR MÜHENDİSLİĞİ PROJE TASARIMI DERSİ

PROJE RAPORU

DANIŞMAN

DOÇ. DR. YUSUF ALTUN

DÜZCE, 2021

İÇİNDEKİLER

ŞEKİL LİSTESİ

BEYAN

Bu proje çalışmasının kendi çalışmam olduğunu, projenin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu projedeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu proje çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu projenin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını beyan ederim.

24 Ocak 2021

Mahmut Can KURT

TEŞEKKÜR

Lisans öğrenimimde ve bu projenin hazırlanmasında gösterdiği her türlü destek ve yardımından dolayı çok değerli hocam Doç. Dr. Yusuf ALTUN'a en içten dileklerimle teşekkür ederim.

Proje çalışmam boyunca değerli katkılarını esirgemeyen eş danışmanım Dr. Öğr. Üyesi Sinan TOKLU'ya da şükranlarımı sunarım.

Bu çalışma boyunca yardımlarını ve desteklerini esirgemeyen sevgili aileme ve çalışma arkadaşlarıma sonsuz teşekkürlerimi sunarım.

24 Ocak 2021 **Mahmut Can KURT**

ÖZET

TENSORFLOW LITE İLE KÖPEK CİNSİ SINIFLANDIRMASI İÇİN ANDROID UYGULAMASI

Mahmut Can KURT

Düzce Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

Bilgisayar Mühendisliği Proje Tasarımı

Danışman: Doç. Dr. Yusuf ALTUN

Ocak 2021

Proje; 120 köpek cinsi ve toplam 20580 görselden oluşan Stanford University Dogs Dataset ile eğitilmiş TensorFlow Lite modelinin kullanıldığı bir Android mobil uygulama içermektedir. Bu uygulamada mobil cihazın kamerası kullanılarak cihaza gösterilen herhangi bir köpeğin cinsi ve bu cinse benzerlik oranıyla beraber diğer cinslere benzerlik oranları yüzdelik olarak kullanıcıya sunulmaktadır. Kullanıcı aynı zamanda köpek cinsinin ne kadar sürede uygulama tarafından çıkarımının yapıldığını ve çıkarım işleminin kaç iş parçasında oluştuğunu görüntüleyebilmekte ve değiştirebilmektedir. Aynı zamanda kullanıcı arayüzünden uygulamanın çalışacağı işlemci yükü CPU ve GPU arasında değiştirilebilmektedir.

Bu uygulama; makine öğrenmesi ve görüntü işleme algoritmalarıyla insanların hayatını kolaylaştırmayı ve köpek cinsleri hakkında daha çok bilgi edinilmesini hedeflemektedir. Uygulama geliştirilmesinin ileri safhalarında köpek cinsleri ile ilgili daha detaylı bilgilerinin verilmesi ve kullanıcıların bu konuda daha fazla bilgilendirilmesini öngörmektedir.

ABSTRACT

ANDROID APPLICATION
FOR DOG BREED CLASSIFICATION WITH TENSORFLOW LITE

Mahmut Can KURT

Düzce University

Engineering Faculty

Department of Computer Engineering

Computer Engineering Project Design

Advisor: Assoc. Prof. Dr. Yusuf ALTUN

January 2021

This project: includes an Android mobile application using TensorFlow Lite model trained with Stanford University Dogs Dataset consisting of 120 dog breeds and a total of 20580 images. In this application, the breed of any dog shown to the device using the camera of the mobile device are presented to the user with the similarity rate to the breed as well as the similarity rates to the other breeds. The user can also view and change how long the dog breed has been extracted takes by the application and how many threads occurs. At the same time, the processor load of the application can be changed between CPU and GPU from the user interface.

This application: aims to make people's lives easier and gives people more information about dog breeds with machine learning and image processing algorithms. In the advanced stages of the application development, it is foreseen to provide more detailed information about dog breeds and to inform users more on this topic.

1. GİRİŞ

2020-2021 Eğitim – Öğretim Yılı Güz Dönemi Bilgisayar Mühendisliği Proje Tasarımı Dersi Proje Görevi olarak yapılan proje, 120 köpek cinsi ve toplam 20580 görselden oluşan Stanford University Dogs Dataset [1] ile eğitilmiş Tensorflow Lite [2] modelinin kullanıldığı bir Android mobil uygulamadır. Uygulamada mobil cihaz kamerası kullanılarak cihaza gösterilen herhangi bir köpeğin cinsi ve bu cinse benzerlik oranıyla beraber diğer köpek cinslerine benzerlik oranları kullanıcı arayüzünde sunulmaktadır. Kullanıcı aynı zamanda köpek cinsinin ne kadar sürede uygulama tarafından çıkarımının yapıldığını ve çıkarım işleminin kaç iş parçasında oluştuğunu görüntüleyebilmekte ve değiştirebilmektedir. Kullanıcı arayüzünden uygulamanın çalışacağı işlemci Central Processing Unit (CPU) ve Graphics Processing Unit (GPU) olarak değiştirilebilmektedir.

Uygulama: makine öğrenmesi ve görüntü işleme algoritmalarıyla insanların hayatını kolaylaştırmayı ve köpek cinsleri hakkında daha çok bilgi edinmesini hedeflemektedir. Uygulama geliştirilmesinin ileri safhalarında köpek cinsleri ile ilgili daha detaylı bilgilerinin verilmesi ve kullanıcıların bu konuda daha fazla bilgilendirilmesini öngörmektedir.

2. MATERYAL VE YÖNTEM

2.1. Proje Süresince Kullanılan Materyaller

- Tensorflow 2.3.1
- NumPy 1.18.5
- MatPlotLib 3.2.2
- Anaconda Navigator 1.9.12
- Conda 4.8.3
- Python 3.8.3
- JupyterLab 2.1.5
- Android Studio 4.2
- Android 9.0 (Pie)
- Android SDK Platform-Tools 30.0.05
- NDK 22.0.7026061
- Pixel 2 API 28 Virtual Device
- LG-H850TR Android 8.0.0
- MobileNet V2

3. PROJE TANIMI

Proje TensorFlow Lite ile köpek cinsleri tanıma üzerine geliştirilmiş bir yapay zeka görüntü işleme framework'ü olan TensorFlow [3] algoritma modelinin entegre edildiği, Java kullanılarak Android ile yazılmış bir mobil uygulamadır. Projenin TensorFlow Lite ve Python geliştirmeleri Jupyter Notebook [4] ortamında, Android ve Java geliştirmeleri ise Android Studio [5] ortamında geliştirilmiştir.

TensorFlow Lite, TensorFlow modellerinin mobil cihazlarda, gömülü sistemlerde ve IoT cihazlarda daha verimli çalışması için özelleştirilmiş bir araç setidir. Google tarafından geliştirilmiş olan bu teknoloji sayesinde, yapay zeka uygulamaları derin öğrenme kavramı ile öğrenilebilir ve daha da geliştirilebilmektedir. Cihaz üzerinde makine öğrenmesi çalıştırmak için cihaz ayarları, cihaz üzerindeki uygulamalar için optimize edilmiştir ve küçük boyuttaki bir dizi kernel işlemini destekler. TensorFlow Lite, küçük bir ikili boyuttan oluşur ve hızlı başlayacak şekilde tasarlanmıştır. Ayrıca Andorid ve iOS dahil olmak üzere çeşitli platformlarla uyumludur. Mobil deneyimi geliştirmek için, iyileştirilmiş yükleme süreleri ve donanım hızlandırması olan mobil cihazlar için optimize edilmiştir.

Jupyter Notebook, alınan notları ve hesaplamaları bir arada tutmak için kullanılan en başarılı araçlardandır. Tekrar edilebilir araştırmada ve veri biliminde sıkça kullanılır. Anaconda'nın [6] sağlamış olduğu uyumlu kütüphane ve framework ortamlarını kolay yoldan aynı ortamda kullanmaya yarar.

Android Studio, Android uygulama geliştirme için resmi entegre geliştirme ortamıdır. Yazılım için Java ile tümleşik bir geliştirme ortamı olan IntelliJ IDEA'ya [7] dayanır ve kod düzenleme ve geliştirici araçlarını içerir. Android Studio, Gradle tabanlı bir oluşturma sistemi, emulator, kod şablonları ve Git [8] entegrasyonu kullanır.

4. UYGULAMA YAPIM AŞAMALARI

4.1. TensorFlow Lite Modeli

4.1.1. Giriş Ardışık Düzeni Kurulumu

Jupyter Notebook'ta açmış olduğumuz proje not defterine öncelikle TensorFlow, NumPy [9], Matplotlib [10] kütüphaneleri ve framework'leri import edilir. Ardından Stanford University Dogs Dataset'in kullanıma sunduğu dosyaları Google Drive'da [11] kaydedilen ortamdan projeye mount edilir. ImageDataGenerator kullanarak mount edilen görüntüler yeniden ölçeklendirilir. Train üreteci oluşturulur ve train veri kümesi dizinin, görüntü boyutunun, toplu iş boyutunun nerede olacağı belirtilir. Flow_from_directory() yöntemiyle train oluşturucuya benzer yaklaşımla validation oluşturucu yaratılır. Son olarak etiketler daha sonra indirilecek bir dosyaya kaydedilir.

4.1.2. Önceden Eğitilmiş Convnet'lerden Temel Model Oluşturma

Google'da geliştirilen MobileNet V2 [12] – [13] modelinden temel ve ImageNet [14] veri kümesi, 1.4 milyon görüntü veri kümesi ve 1000 sınıf web görüntüsü üzerinde önceden eğitilmiş bir model oluşturulur.

İlk olarak, özellik çıkarma için hangi MobileNet V2 ara katmanının kullanılacağı seçilir. Yaygın olan uygulama, "darboğaz katmanı" olarak adlandırılan düzleştirme işleminden önce en son katmanın çıktısını kullanmaktır. Buradaki mantık, aşağıdaki tam bağlantılı katmanların ağın üzerinde eğitildiği görev için çok özel olacağı ve bu nedenle bu katmanlar tarfından öğrenilen özelliklerin yeni bir görev için çok yararlı olmayacağıdır. Bununla birlikte, darboğaz özellikleri büyük ölçüde genelliği korur.

ImageNet üzerinde eğitilmiş ağırlıklarla önceden yüklenmiş bir MobileNet V2 modeli somutlaştırılır. include_top=False argümanı belirleyerek, üst kısımdaki sınıflandırma katmanlarını içermeyen bir ağ yüklenir, bu özellik çıkarımı için en ideal yöntemdir.

4.1.3. Özellik Çıkarma

Önceki adımdan oluşturulan evrişimli tabanı dondurulur ve bu bir özellik çıkarıcı olarak kullanılır, üstüne bir sınıflandırıcı eklenilir ve en üst düzey sınıflandırıcı eğitilir. Bir sınıflandırma başlığı eklendikten sonra model derlenir. Model eğitilmeden önce derlenmelidir. Birden fazla sınıf olduğu için kategorik bir çapraz entropi kaybı kullanılır.

4.1.4. Öğrenme Eğrileri

MobileNet V2 temel modelini sabit özellik çıkarıcı olarak kullanırken eğitim ve doğrulama doğruluğu / kaybının öğrenme eğrileri kontrol edilir.

4.1.5. Üst Katman Ağırlıklarını Eğitmek

Özellik çıkarma deneyinde, bir MobileNet V2 temel modelinin üzerinde yalnızca birkaç katman eğitildi. Önceden eğitilmiş ağın ağırlıklarının eğitim sırasında güncellenmesi gerekmektedir.

Performansı daha da artırmanın bir yolu, eklenilen sınıflandırıcının eğitimiyle birlikte önceden eğitilmiş modelin üst katmanlarının ağırlıklarını eğitmektir. Eğitim süreci, ağırlıkların genel özellikler haritalarından veri setiyle özel olarak ilişkili özelliklere ayarlanmasını zorunlu kılacaktır.

Modelin üst kısımlarının çözülmesi için tek yapılması gereken, base_model'i çözmek ve alt katmanları eğitilemez hâle getirmek. Ardından model yeniden derlenir ve eğitime devam edilir.

Bu aşama sonrasında model çok daha düşük bir eğitim oranı kullanılarak yeniden derlenir. Ardından model kullanılarak tf.saved_model.save kaydedilir ve kaydedilen model tf lite uyumlu bir biçime dönüştürülür. Dönüştürülen model ve etiketler bilgisayar ana dizinine indirilir.

MobileNet V2 temel modelinin son birkaç katmanında ince ayar yaparken ve bunun üzerine sınıflandırıcıyı eğitirken, eğitim ve doğrulama doğruluğu / kaybının öğrenme eğrileri control edilir. Doğrulama kaybı, eğitim kaybından çok daha fazladır, bu sebeple aşırı uyum sağlama görülebilir. Yeni eğitim seti nispeten küçük ve orjinal MobileNet V2 veri setlerine benzer olduğundan öncekine nazaran daha fazla uyum elde edilebilir.

4.1.6. **Özet**

Özellik çıkarma için önceden eğitilmiş model kullanma: Küçük bir veri kümesiyle çalışrken, aynı alandaki daha büyük bir veri kümesi üzerinde eğitilmiş bir model tarafından öğrenilen özelliklerden yararlanılır. Bu, önceden eğitilmiş modelin somutlaştırılması ve üstüne tam bağlantılı bir sınıflandırıcı eklenmesiyle yapılır. Önceden eğitilmiş model dondurulur ve eğitim sırasında yalnızca sınıflandırıcının ağırlıkları güncellenir. Bu durumda, evrişimli taban, her bir görüntüyle ilişkili tüm özellikleri çıkarır ve çıkarılan özellikler kümesi verilen görüntü sınıfını belirleyen bir sınıflandırıcı eğitilir.

Önceden eğitilmiş bir modelin ince ayarı: Performansı daha da artırmak için, önceden eğitilmiş modellerin en üst düzey katmanlarını ince ayar yoluyla yeni veri kümesinde yeniden kullanılabilir. Bu durumda, ağırlıklar, modelin veri kümesine özgü üst düzey özellikleri öğreneceği şekilde ayarlanır. Bu teknik, genellikle eğitim veri kümesi büyük olduğunda ve önceden eğitilmiş modelin eğitildiği orjinal veri kümesine çok benzer olduğunda kullanılır.

4.2. Android Uygulama Modeli

4.2.1. Uygulamaya Dair

Projede kullanılan uygulama, TensorFlow'un kullanıcılarına sunduğu örnek bir uygulamadan esinlenilerek geliştirilmiştir.[15] Bu uygulama, cihazın arkaya bakan kamerasından gördüğü her şeyi sürekli olarak sınıflandırmak için görüntü sınıflandırmasını kullanır. Uygulama cihazda veya emülatörde çalışabilir.

Çıkarım TensorFlow Lite Java API [16] ve TensorFlow Lite Android Destek Kitaplığı [17] kullanılarak gerçekleştirilir. Demo uygulaması, kareleri gerçek zamanlı olarak sınıflandırarak en olası sınıflandırmaları gösterir. Kullanıcının kayan nokta veya nicelleştirilmiş model arasında seçim yapmasına, iş parçacığı seçmesine ve CPU veya GPU üzerinden çalışıp çalışmayacağına karar vermesine olanak tanır.

4.2.2. Uygulama Oluşturma ve Geliştirme Aşamaları

Bir TensorFlow Lite (TFLite) modelini içe aktarmak için:

1- TFLite modelinin kullanılmak istendiği modülde File > New > Other > Tensorlow Lite Model kısmından yeni bir TFLite modeli oluşturulur.
2- TFLite dosyasının konumu seçilir. build.gradle, ML Model baglama ile modülün bağımlılığını yapılandırır ve tüm bağımlılıklar otomatik olarak Android modülün build.gradle dosyasına eklenir. GPU hızlandırma kullanılmak istenirse TensorFlow GPU'yu içe aktarmak için ikinci onay kutusu seçilir ve Finish butonuna tıklanır.
3- İçe aktarma başarılı olduktan sonra açılan ekranda modeli kullanmaya başlamak için Java seçilir, kod Sample Code bölümünün altına yapıştırılır.

4.2.3. TensorFlow Lite Görev Kitaplığı Kullanımı

TensorFlow Lite Görev Kitaplığı [18], uygulama geliştiricilerin TFLite ile makine öğrenimi deneyimleri oluşturmaları için bir dizi güçlü ve kullanımı kolay göreve özgü kitaplık içerir. Görüntü sınıflandırması, soru ve cevap gibi popüler makine öğrenimi görevleri için optimize edilmiş, kullanıma hazır model arabirimleri sağlar. Model arabirimleri, en iyi performansı ve kullanılabilirliği elde etmek için her görev için özel olarak tasarlanmıştır. Görev kitaplığı, çapraz platformda çalışır ve Java, C++ ve Swift'de desteklenir.

Destek Kitaplığını Android uygulamada kullanmak için, sırasıyla Görev Vizyon kitaplığı ve Görev Metni kitaplığı için. JCenter'da barındırılan AAR [19] kullanılır.

4.2.4. TensorFlow Lite Android Destek Kitaplığı Kullanımı

TensorFlow Lite Android Destek Kitaplığı, modelleri uygulamaya entegre etmeyi kolaylaştırır. Ham girdi verilerini modelin gerektirdiği forma dönüştürmeye yardımcı olan ve modelin çıktısını yorumlayarak gerekli standart kod miktarını azaltan üst düzey API'ler sağlar.

Görüntüler ve diziler dahil, girişler ve çıkışlar için ortak veri formatlarını destekler. Ayrıca, görüntüyü yeniden boyutlandırma ve kırpma gibi görevleri gerçekleştiren ön ve son işlem birimleri sağlar.

Destek Kitaplığını Android uygulamada kullanmak için, JCenter'da [20] barındırılan TensorFlow Lite Destek Kitaplığı AAR kullanılır. Bu AAR, tüm Android ABI'leri [21] için ikili dosyalar içerir. Yalnızca desteklenmesi gereken ABI'leri dahil ederek uygulamanın ikili dosyasının boyutu azaltılabilir.

4.2.5. TensorFlow Lite'ın Android Uygulamasına Eklenmesi

Daha önce eğitilen TensorFlow Lite modeli model.tflite ve label.txt dosyası assets klasörü altına kopyalanır. Ardından build.gradle güncellenmesi yapılır, TensorFlow Lite uygulamanın bağımlılıklarına eklenir.

Android uygulama ikili dosyasını oluştururken TensorFlow Lite model dosyalarını sıkıştırmasını önlemek için noCompress komutu kullanılır. Modelin çalışması için bu komutun eklenmesi zorunludur. Ardından değişiklikleri uygulamak için Sync Now tıklanır.

4.2.6. TensorFlow Lite Yorumlayıcısının Başlatılması

ClassifierFloatMobileNet.java dosyası altında çıkarım için kullanılacak model.tflite ve labels.txt dosyaları eklenir.

Classifier.java dosyasında Classifier sınıfında bir TFLite yorumlayıcısı bildirilir. Ardından Classifier sınıf yapıcısının altında yorumlayıcının bir örneği oluşturulur. TensorFlow Lite çıkarımının çalıştırılma kısmı hazırlanır ve yorumlayıcı örneği eklenir.

4.2.7. TensorFlow Lite ile Çıkarım Yapmak

TensorFlow Lite yorumlayıcısı bir önceki başlıkta kuruldu, bu yüzden giriş görüntüsündeki bazı köpekleri tanımak için kod yazılır. ByteBuffers kullanarak görüntüleri işlemek için birçok satır kod yasmak yerine, TensorFlow Lite, görüntü ön işlemeyi basitleştirmek için kullanışlı bir TensorFlow Lite Destek Kitaplığı sağlıyor. Ayrıca, TensorFlow Lite modellerinin çıktılarını işlemeye ve TensorFlow Lite yorumlayıcının kullanımını kolaylaştırmaya yardımcı olur. Özetle:

- Girdi önceden işlenir, girdi görüntülerini yeniden boyutlandırmak ve döndürmek için bir görüntü işlemcisi oluşturulur.
- TensorFlow Lite ile çıkarım yapılır.
- Çıktının sonradan işlenmesi: Olasılık dizisi TensorLabel kullanılarak kullanıcı tarafından okunabilir bir dizeye dönüştürülür.

Classifier.java dosyasında ilk olarak, kamera tarafından yakalanan görüntüler önceden işlenir. Ardından girdi görüntüsünü TensorFlow Lite modelinin girdi boyutuyla eşleşecek şekilde yeniden boyutlandırılması ve ardından modelin beslenmesi için RGB biçimine dönüştürülmesi gerekiyor. TensorFlow Lite Destek Kitaplığından ImageProcessor kullanarak, kolayca görüntü boyutlandırma ve dönüştürme yapılabilir. Bu aşamadan sonra TensorFlow Lite yorumlayıcısı ile çıkarım yapılabilir. Ön işlenmiş görüntü TensorFlow Lite yorumlayıcısına eklenir. Son olarak, model çıktısından etiketlerin haritasını ve olasılıkları alınır. labeledProbability bulunan kod bloğunda bulunan labeledProbability, her etiketi olasılığıyla eşleyen nesnedir. TensorFlow Lite Destek Kitaplığı, model çıktısından insan tarafından okunabilir bir olasılık haritasına

dönüştürmek için kullanışlı bir yardımcı program sağlar. Daha sonra labeledProbability'den getTopKProbability(…) metodu ile top-K en olası etiketleri çıkarılır.

4.2.8. Uygulamanın Testi ve Çalıştırılması

Uygulama, gerçek bir Android cihazda veya Android Studio Emulator'da çalışabilir. Tarafımca yapılan çalışmalarda uygulama, 3 farklı Android cihazda ve 2 farklı Android emülatörde test edilmiştir.

Android cihaz kurulumu için cihaz ayarlarından Geliştirici Modu ve USB Hata Ayıklama etkinleştirilmelidir. Aksi takdirde uygulama Anroid Studio'dan cihaza yüklenemez.

Android Studio ortamında emulator kurmak geliştiriciler tarafından oldukça fazla tercih edilmektedir. Bu uygulama kamerayı kullandığından, varsayılan kurulum aşamalarında emülatör, bilgisayar kamerasını kullanmak zorundadır.

Emülatör kurulumu için, Android Virtual Devices (AVD) Manager içinde yeni bir cihaz oluşturulması gerekmektedir. Ana AVDM sayfasından Sanal Cihaz Oluştur seçeneğine tıklanır. Ardından, sanal cihaz kurulumunun son sayfası olan "Yapılandırmayı Doğrula" sayfasında, "Gelişmiş Ayarları Göster" seçilir. Gösterilen gelişmiş ayarlarla, her iki kamera kaynağı da ana bilgisayarın web kamerasını kullanacak şekilde ayarlanılabilir.

Uygulamada herhangi bir değişiklik yapılmadan önce, TensorFlow'un repository'si ile birlikte gelen sürüm çalıştırılır. Derleme ve yükleme işlemini başlatmak için Android Studio'da bir Gradle senkronizasyonu çalıştırılır. Sonrasında Run butonuna tıklanır. Açılır pencerede kullanılacak olan cihazın seçilmesi gerekmektedir. Uygulama açıldığında kamera kullanma ve depolama alanına erişmek için iki izin isteyecektir, uygulamanın sağlıklı çalışması için gelen izin ekranlarında onay verilmesi gerekmektedir.

Bu aşamadan sonra cihaz kamerası herhangi bir köpeğe doğru tutulduğunda köpek cinsini uygulama tahmin etmeye başlayacaktır.

4.2.9. GPU Delegesi ile Çıkarım Süresini Hızlandırma

TensorFlow Lite, mobil cihazlarda çıkarımı hızlandırmak için çeşitli donanım hızlandırıcıları destekler. GPU, TensorFlow Lite'ın bir delege mekanizması aracılığıyla yararlanabileceği hızlandırıcılardan biridir ve kullanımı oldukça kolaydır:

- İlk olarak strings.xml dosyası açılır.
- Kullanıcı arayüzünde "GPU" nun görünmesi için XML dosyasına bir GPU dizesi eklenir.
- Sonra Classifier.java dosyası açılır.
- Classifier sınıfına GpuDelegate fonskiyonu null şekilde eklenir.
- Daha sonra kullanıcı GPU kullanmayı seçtiği durumu ele alınır.
- GPU delegesini, tercümana bağlanabilmesi için TFLite seçeneklerine eklenir.
- Kullandıktan sonra GPU temsilcisi kapatılır.

Orta / üst düzey bir mobil cihazda GPU, CPU'dan çok daha hızlıdır. Düşük uç cihazlar daha yavaş GPU'lara sahip olma eğilimindedir. Bu nedenle görülen hızlanmalarda değişiklikler görülebilir.

5. SONUÇ

Bilgisayar Mühendisliği Proje Tasarımı dersi için şahsım, Mahmut Can Kurt, tarafından hazırlanmış olan projede, giriş bölümünde ve diğer bölümlerde de belirtildiği gibi TensorFlow Lite, Android Destek Kitaplığı, Java ve Python teknolojileri kullanılmıştır.

Bu projeyi hazırlarken lisans eğitimim boyunca öğrenmiş olduğum bu teknolojileri uygulamada yetkinlik kazandım. Aynı zamanda bu teknolojilerin farklı kullanım alanlarını öğrendim ve gerek akademik alanda gerek sektörel anlamda öneminin ne kadar yüksek olduğu hakkında tecrübe edindim. Bilgisayar bilimlerinin ilerleyen safhalarda yapay zeka teknolojileriyle beraber ne kadar genişleyip insanlığa faydalı olacağına dair ufkum genişledi.

Proje boyunca yaptığım çalışmalar esnasında fikir ve bilgi alışverişinde bulunduğum akademik personeller ve sektörde çalışan kişilerle olan iletişimim sayesinde bağlantılarımı genişlettim ve gerek mesleki gerek akademik alanda ilerleyen zamanlarım için referanslar edindim.

Lisans öğrenimim boyunca geçirdiğim 7 dönemin ardından öğrenmiş olduğum Android, Görüntü İşleme, Yapay Zeka, Makine Öğrenmesi, Nesneye Dayalı Programlama, Algoritma Analizi ve Veri Yapıları bilgilerimi pratiğe döktüm. Bütün bunların yanında araştırmış olduğum makaleler ve sektörel alandaki yenilikler sayesinde tecrübe edindim ve kendimi bu alanda geliştirdim.

Bilgisayar Mühendisliği Proje Tasarımı dersinde bana ön ayak olan Düzce Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'ne, lisans öğrenimimde ve bu projenin hazırlanmasında gösterdiği her türlü destek ve yardımından dolayı çok değerli hocam Doç. Dr. Yusuf ALTUN'a, proje çalışmam boyunca değerli katkılarını esirgemeyen eş danışmanım Dr. Öğr. Üyesi Sinan TOKLU'ya ve bu çalışma boyunca yardımlarını ve desteklerini esirgemeyen sevgili aileme ve çalışma arkadaşlarıma sonsuz teşekkürlerimi sunarım.

6. EKLER

Projeye ait kodlarda önemli etkisi bulunan dosyalar ve kullanıldığı alanlar aşağıdaki gibidir. Gerekli olduğu takdirde paylaşılan proje dosyalarında, raporda paylaşılmayan dosyalara ve kod bloklarına ulaşılabilir.

6.1. TensorFlow Lite Dog Breeds Recognition İşlemleri

Projede köpek tanıma algoritmasının eğitilme işleminin yapıldığı, Jupyter Notebook'da yazılan dog_breeds_tf_lite.ipynb dosyası içerisinde bulunan kodlar aşağıda görüldüğü gibidir:

```
"""
Proje süresince kullanılacak olan
TensorFlow, NumPy ve MatPlotLib kütüphaneleri
projeye import edilir
"""
import tensorflow as tf
assert tf.__version__.startswith('2')
import os
import numpy as np
import matplotlib.pyplot as plt

# TensorFlow versiyon kontrolü yapılır.
tf.__version__

"""
Google'ın sağlamış olduğu drive kullanılarak muhafaza ettiğimiz veri seti
projeye eklenir.
"""
from google.colab import drive
drive.mount('/content/gdrive')
image_dir = '/content/gdrive/My Drive/dataset/Images'
```

```
# Görüntü boyutları eğitim için eşit hale getirilir
IMAGE_SIZE = 224
BATCH_SIZE = 64
datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1. / 255,
    validation_split=0.2)
train_generator = datagen.flow_from_directory(
    image_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='training')
val_generator = datagen.flow_from_directory(
    image_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='validation')
for image_batch, label_batch in train_generator:
    break
image_batch.shape, label_batch.shape
print(train_generator.class_indices)

# Görüntülere ait başlıklar sınıflandırılır
labels = '\n'.join(sorted(train_generator.class_indices.keys()))
with open('labels.txt', 'w') as f:
    f.write(labels)
get_ipython().system('cat labels.txt')
IMG_SHAPE = (IMAGE_SIZE, IMAGE_SIZE, 3)

# Önceden eğitilmiş olan MobileNet V2'den temel model oluşturulur
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
```

```
                                    include_top=False,
                                    weights='imagenet')
base_model.trainable = False
model = tf.keras.Sequential([
   base_model,
   tf.keras.layers.Conv2D(32, 3, activation='relu'),
   tf.keras.layers.Dropout(0.2),
   tf.keras.layers.GlobalAveragePooling2D(),
   tf.keras.layers.Dense(120, activation='softmax')
])
model.compile(optimizer=tf.keras.optimizers.Adam(),
            loss='categorical_crossentropy',
            metrics=['accuracy'])
model.summary()

# Eğitilebilir tüm görseller
print('Number of trainable variables = {}'.format(len(model.trainable_variables)))
%%time
epochs = 25
history = model.fit(train_generator,
                  steps_per_epoch=len(train_generator),
                  epochs=epochs,
                  validation_data=val_generator,
                  validation_steps=len(val_generator))
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()), 1])
plt.title('Training and Validation Accuracy')
plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0, 1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
base_model.trainable = True

# Temel modelde kaç katman olduğu incelenir
print("Number of layers in the base model: ", len(base_model.layers))
# Bu katmandan itibaren ince ayar yapılır
fine_tune_at = 100

# "fine_tune_at" katmanından önceki katmanlar dondurulur
for layer in base_model.layers[:fine_tune_at]:
   layer.trainable = False
model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(1e-5),
              metrics=['accuracy'])
```

```
model.summary()
# Eğitilebilir tüm görseller
print('Number of trainable variables = {}'.format(len(model.trainable_variables)))
history_fine = model.fit(train_generator,
                         steps_per_epoch=len(train_generator),
                         epochs=5,
                         validation_data=val_generator,
                         validation_steps=len(val_generator))

"""
Eğitilen görsellerin bulunduğu "model.tflite" dosyası ve
bu görsellere ait başlıkların bulunduğu "labels.txt" dosyası indirilir
"""
saved_model_dir = 'save/fine_tuning'
tf.saved_model.save(model, saved_model_dir)
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
tflite_model = converter.convert()
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
from google.colab import files

files.download('model.tflite')
files.download('labels.txt')
acc = history_fine.history['accuracy']
val_acc = history_fine.history['val_accuracy']

loss = history_fine.history['loss']
val_loss = history_fine.history['val_loss']

plt.figure(figsize=(8, 8))
```

```
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()), 1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0, 1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```

6.2. Android “build.gradle” Dosyası

TensorFlow ile Android uygulamasını birbirine bağlamak için kullanılan “build.gradle” dosyasındaki kodlar şu şekildedir:

```
apply plugin: 'com.android.application'
project.ext.ASSET_DIR = projectDir.toString() + '/src/main/assets'
assert file(project.ext.ASSET_DIR + "/model.tflite").exists()
assert file(project.ext.ASSET_DIR + "/labels.txt").exists()

android {
   compileSdkVersion 28
   defaultConfig {
      applicationId "org. tensorflow.lite.examples.classification"
      minSdkVersion 21
      targetSdkVersion 28
      versionCode 1
      versionName "1.0"
   }
   buildTypes {
      release {
         minifyEnabled false
         proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
      }
   }

   // TFLite model dosyasını sıkıştırmamak için opsiyon ekleme
   aaptOptions {
      noCompress "tflite"
   }
   compileOptions {
```

```
        sourceCompatibility = '1.8'
        targetCompatibility = '1.8'
    }
    lintOptions {
        abortOnError false
    }
}
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.0.0'
    implementation 'androidx.coordinatorlayout:coordinatorlayout:1.0.0'
    implementation 'com.google.android.material:material:1.0.0'

    // TFLite bağlılığı
    implementation('org.tensorflow: tensorflow-lite:0.0.0-nightly') { changing = true }
    implementation('org. tensorflow: tensorflow-lite-gpu:0.0.0-nightly') { changing = true }
    implementation('org. tensorflow: tensorflow-lite-support:0.0.0-nightly') { changing = true }

}
```

6.3. Android “ClassifierFloatMobileNet.java” Dosyası

Uygulamada kullanılacak görsellerin bulunduğu “models.tflite” ve etiketlerin bulunduğu “labels.txt” dosyasının projeye eklenmesi ve tanınması için kullanılacak “ClassifierFloatMobileNet.java” dosyasının içinde bulunan kodlar aşağıdaki gibidir:

```
package org.tensorflow.lite.examples.classification.tflite;

import android.app.Activity;
import java.io.IOException;
import org.tensorflow.lite.examples.classification.tflite.Classifier.Device;
import org. tensorflow.lite.support.common.TensorOperator;
import org. tensorflow.lite.support.common.ops.NormalizeOp;

/** Bu TensorFlowLite sınıflandırıcı, float MobileNet modeliyle çalışır. */
public class ClassifierFloatMobileNet extends Classifier {

  /** Float MobileNet, kullanılan girişin ek olarak normalleştirilmesini gerektirir. */
  private static final float IMAGE_MEAN = 0f;
  private static final float IMAGE_STD = 255f;
  /**
   * Float modeli, işlem sonrası işlemde dekuantizasyona ihtiyaç duymaz.
   * Normalizasyonu atlamak için ortalama ve std'nin sırasıyla 0.0f ve 1.0f olarak ayarlanması.
   */
  private static final float PROBABILITY_MEAN = 0.0f;
  private static final float PROBABILITY_STD = 1.0f;

  /**
   * Initializes a {@code ClassifierFloatMobileNet}.
   * @param activity
   */
```

```
  public ClassifierFloatMobileNet(Activity activity, Device device, int numThreads)
      throws IOException {
    super(activity, device, numThreads);
  }

  // Model dosyası olarak model.tflite ve etiket dosyası olarak labels.txt belirtilir.
  @Override
  protected String getModelPath() {
    return "model.tflite";
  }
  @Override
  protected String getLabelPath() {
    return "labels.txt";
  }
  @Override
  protected TensorOperator getPreprocessNormalizeOp() {
    return new NormalizeOp(IMAGE_MEAN, IMAGE_STD);
  }
  @Override
  protected TensorOperator getPostprocessNormalizeOp() {
    return new NormalizeOp(PROBABILITY_MEAN, PROBABILITY_STD);
  }
}
```

6.4. Android “Classifier.java” Dosyasında Bulunan Önemli Noktalar

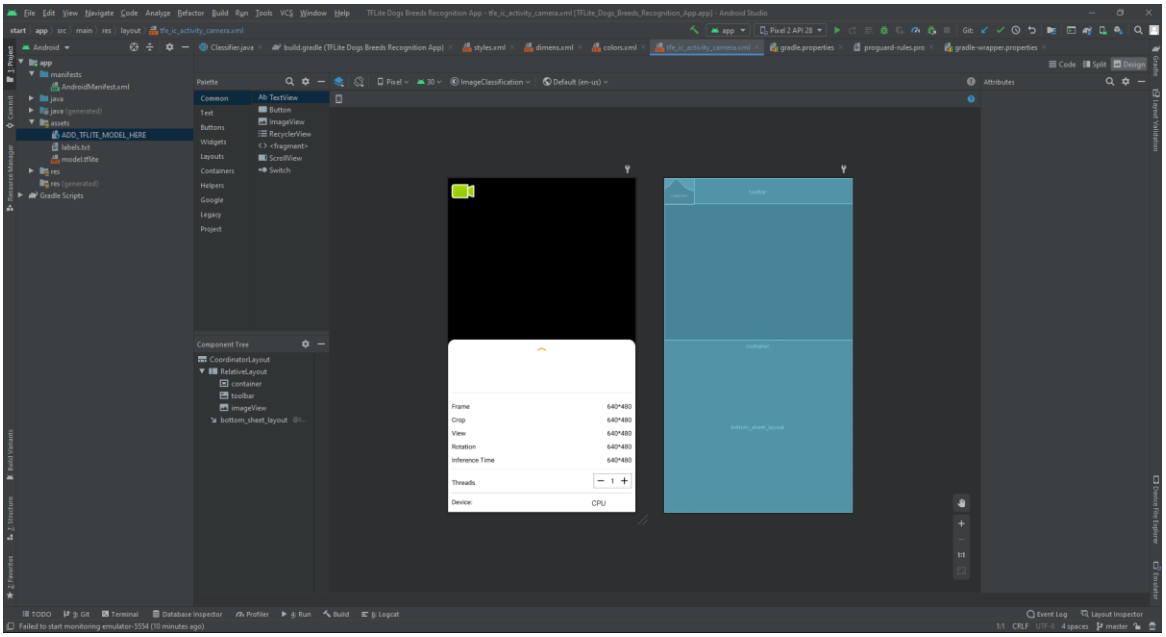
GPU delegeliği, TFLite çıkarımcısının tanımlandığı ve TensorFlow tercümanlığının işlenmiş olduğu “Classifier.java” dosyasında bulunan önemli ve işlevli kod blokları aşağıda görüldüğü gibidir:

```
// TFLite tercümanlığının eklendiği kod bloğu:
protected Interpreter tflite;

// TFLite yorumlayıcı örneğinin oluşturulduğu kod bloğu:
tflite = new Interpreter(tfliteModel, tfliteOptions);

// Yorumlayıcının kapatıldığı kod bloğu:
tflite.close();
tflite = null;

// Ön İşleme yapılması için TFLite Destek Kitaplığından
// Görüntü İşlemcisi Tanımlama:
ImageProcessor imageProcessor =
        new ImageProcessor.Builder()
            .add(new ResizeWithCropOrPadOp(cropSize, cropSize))
            .add(new ResizeOp(imageSizeX, imageSizeY,
ResizeMethod.NEAREST_NEIGHBOR))
            .add(new Rot90Op(numRoration))
            .add(getPreprocessNormalizeOp())
            .build();
return imageProcessor.process(inputImageBuffer);

// TFLite çıkarımcısının çalıştırılması:
tflite.run(inputImageBuffer.getBuffer(), outputProbabilityBuffer.getBuffer().rewind());
```

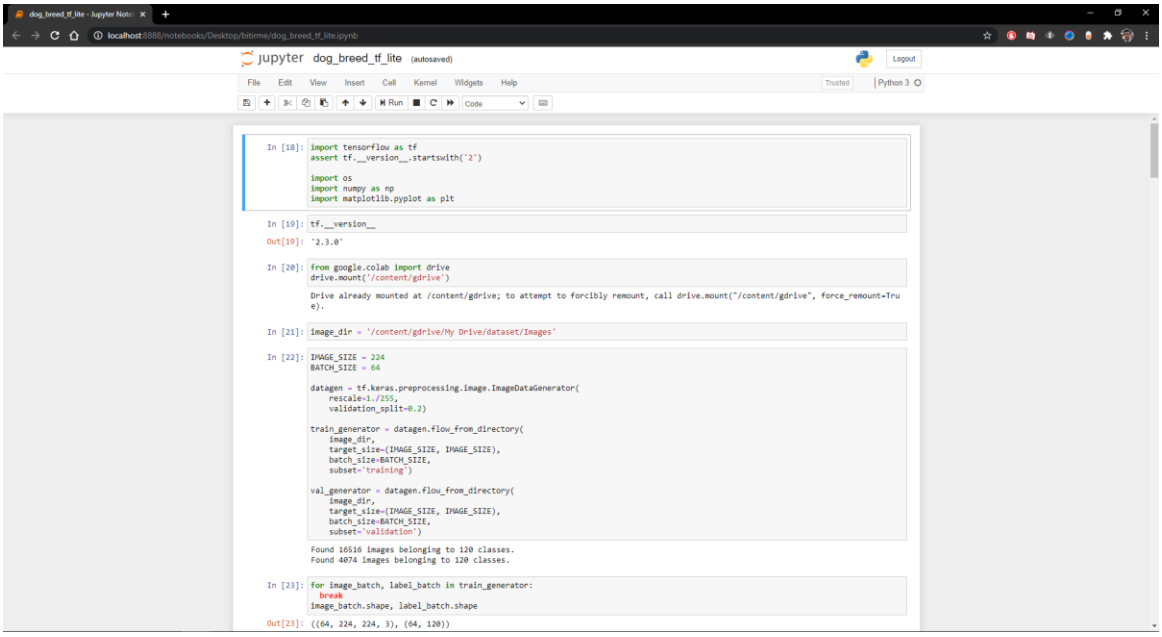
```
// Olasılıkların kategorize edilmesi ve etiketlerle ilişkilendirilmesi:
Map<String, Float> labeledProbability =
   new TensorLabel(labels, probabilityProcessor.process(outputProbabilityBuffer))
       .getMapWithFloatValue();

// GPU delege tanımlama kod bloğu:
private GpuDelegate gpuDelegate = null;

// GPU delege örneği oluşturuma ve bunu yorumlayıcı seçeneklerine ekleme:
gpuDelegate = new GpuDelegate();
tfliteOptions.addDelegate(gpuDelegate);

// GPU delegeliğinin kapatılması
if (gpuDelegate != null) {
  gpuDelegate.close();
  gpuDelegate = null;
}
```

7. PROJEYE AİT GÖRSELLER

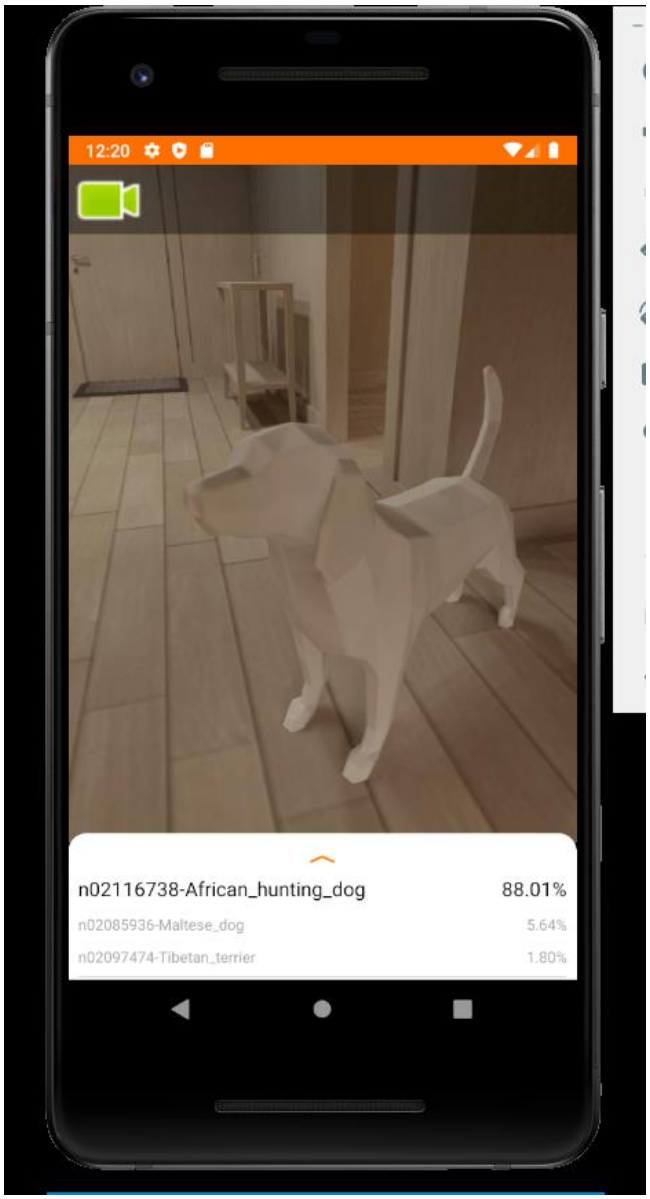
Projenin yapım aşamalarında kullanılan programlar ve uygulamanın ekran görüntüleri aşağıdaki gibidir:



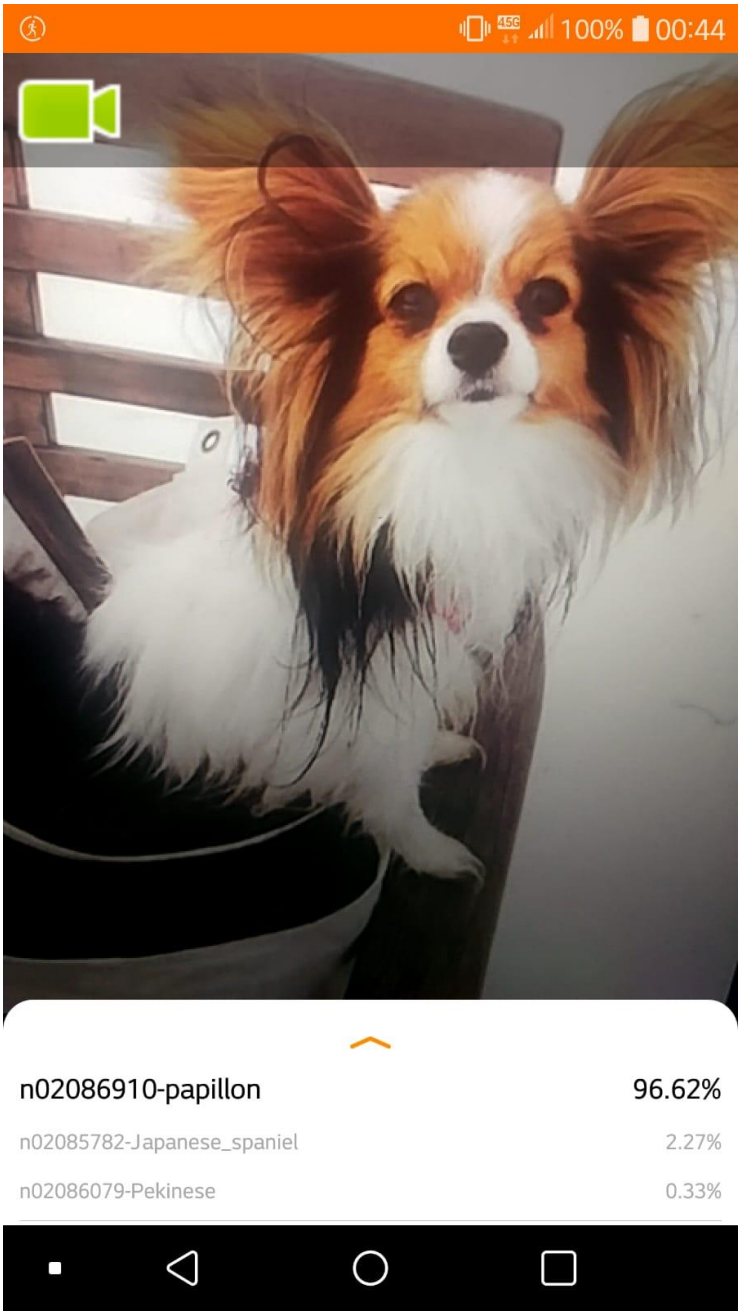
Görsel 1- Projenin Android Uygulama Kısmının Yazıldığı Android Studio Ortamı



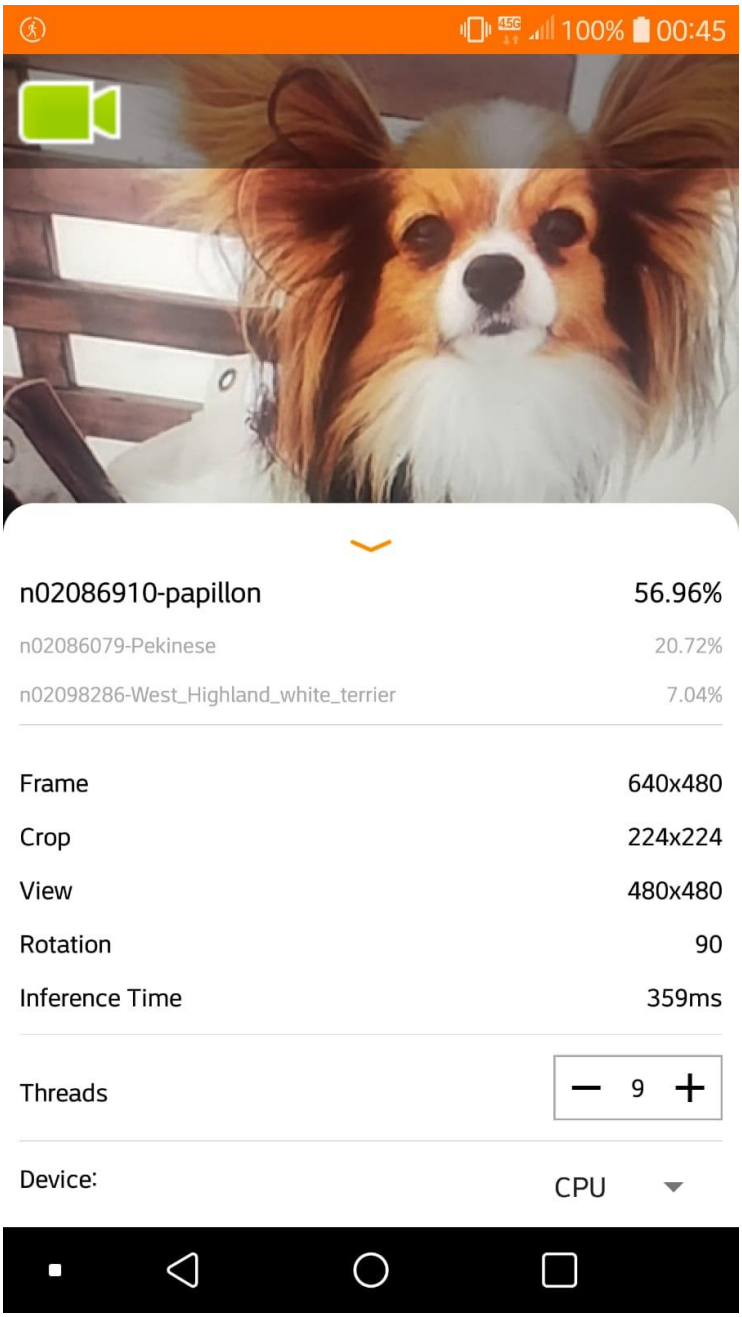
Görsel 2- TensorFlow Lite Modelinin Yazıldığı Jupyter Notebook Ortamı



Görsel 3- Pixel 2 API 28 Sanal Cihaz'da Uygulamanın Test Aşamasındaki Ekran Görüntüsü



Görsel 4- LG-H850TR Cihazında Uygulamanın Test Aşamasındaki Ekran Görüntüsü



Görsel 5- LG-H850TR Cihazında Uygulamada Bulunan Alt Menüye Dair Ekran Görüntüsü

KAYNAKÇA

[1] – http://vision.stanford.edu/aditya86/ImageNetDogs/

[2] – https://www.tensorflow.org/lite

[3] – https://www.tensorflow.org/

[4] – https://jupyter.org/

[5] – https://developer.android.com/studio

[6] – https://www.anaconda.com/

[7] – https://www.jetbrains.com/idea/

[8] – https://git-scm.com/

[9] – https://numpy.org/

[10] – https://matplotlib.org/

[11] – https://www.google.com/intl/en_in/drive/

[12] – https://arxiv.org/abs/1801.04381

[13] – https://keras.io/api/applications/mobilenet/

[14] – http://www.image-net.org/

[15] – https://0fs.me/7868728

[16] – https://www.tensorflow.org/lite/api_docs/java/org/tensorflow/lite/package-summary

[17] – https://github.com/tensorflow/tflite-support/tree/master/tensorflow_lite_support/java

[18] – https://www.tensorflow.org/lite/inference_with_metadata/task_library/overview

[19] – https://gist.github.com/lopspower/6f62fe1492726d848d6d

[20] – https://jcenter.bintray.com/

[21] – https://developer.android.com/ndk/guides/abis

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	:	Mahmut Can Kurt
Doğum Tarihi ve Yeri	:	01.04.1998 – Bakırköy
Yabancı Dili	:	İngilizce
E-Posta	:	is.mahmutcankurt@gmail.com

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul / Üniversite**	**Mezuniyet Yılı**
Lisans	Bilgisayar Müh.	Düzce Üniversitesi	…
Lise	Sayısal	Pendik Fatih AL	2016