T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 401 BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI

BİTKİ SINIFLANDIRMA İÇİN MOBİL UYGULAMA

Rabia ABDİOĞLU

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Tez Danışmanı : Dr. Öğr. Üyesi Seçkin Arı

2022-2023 Güz Dönemi

ÖNSÖZ

Bu çalışmanın hazırlanması sürecinde yardımcı olan, yol gösteren, emeği geçen Dr. Öğr. Üyesi Seçkin Arı hocama, bu zamana kadar bize maddi manevi destek olan aileme, çalışmalarımda bana her türlü yardımı sağlayan, ilim öğreten bütün hocalarıma teşekkürü borç bilirim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

CNN	: Evrişimli Sinir Ağları (Convolutional Neural Network)
IDE	: Tümleşik geliştirme ortamı(Integrated Development Environment)
ML	: Makine Öğrenimi (Machine Learning)
API	: Uygulama programlama arayüzü(Application Programming Interface)

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Görüntü Sınıflandırma, Evrişimli Sinir Ağları, Mobil Uygulama

Bitki tanıma uygulaması kullanıcıların botanik alanında daha bilinçlenmesini amaçlamaktadır. Bitki sınıflandırma bilgisayar görme sistemlerini kullanan aktif bir araştırma alanıdır. Nesne tanıma ve sınıflandırma yöntemlerinin ihtiyaca uygun olarak çeşitleri vardır. Derin sinir ağları ile fotoğraftan bitki tanınması ve sınıflandırılması amaçlanmaktadır.

Bu çalışmada, bitki sınıflandırması için geniş veri seti ile makine öğrenmesi modeli eğitmeyi ve modelin kullanılabilmesi için mobil uygulama geliştirmeyi amaçladım. Mobil için daha uygun olan MobileNet modelinden faydalanıldı. Ayrıca, PlantNet veri seti kullanılarak MobileNet modeli eğitildi. 1080 bitki sınıfından oluşan veri seti, 243.916 eğitim, 31.118 doğrulama,31.112 test için olmak üzere toplam 306.146 fotoğraftan oluşmaktadır.

Bu tasarım çalışması ile farklı veri setleri kullanılarak eğitilmiş, bitki sınıflandırma yapan makine öğrenimi modelleri incelendi. Model oluşturma ve eğitme hakkında denemeler yapıldı.

BÖLÜM 1. GİRİŞ

1.1. Yapay Zeka

Yapay zeka, insan zekası gerektiren görevleri yerine getirebilen akıllı makineler oluşturmakla ilgilenen geniş kapsamlı bir bilgisayar bilimi dalı olarak tanımlayabiliriz. Fakat bu tanımlama yapay zekanın tam olarak ne olduğunu ve makineleri nasıl akıllı yaptığını açıklayamadığından, yapay zeka kavramını sınırlandırıyor. Yapay zeka, birden fazla yaklaşımı olan disiplinler arası bir bilimdir. Ancak makine öğrenimi ve derin öğrenmedeki ilerlemeler, yapay zekanın makine öğrenmesi gibi diğer alanlarında ve teknoloji endüstrisinin neredeyse her sektöründe bir değişikliğe kapı açıyor.

1.2. Tarihi

Yapay zeka terimi 1956'da türetildi, ancak yapay zeka, artan veri hacimleri, gelişmiş algoritmalar ve bilgi işlem gücü ve depolama alanındaki gelişmeler sayesinde bugün daha popüler hale geldi.

1950'lerdeki erken yapay zeka araştırmaları, problem çözme ve sembolik yöntemler gibi konuları araştırdı. Alan Turing, "Bilgisayar Makineleri ve Zeka" adlı makalesini yayınlayarak, bir makinenin zeki olup olmadığını belirleme yöntemi olan Turing Testi yöntemi önerdi. Harvard lisans öğrencileri Marvin Minsky ve Dean Edmonds, ilk sinir ağı bilgisayarı olan SNARC'ı yaptılar.

1960'larda ABD Savunma Bakanlığı bu tür işlere ilgi duydu ve temel insan muhakemesini taklit etmek için bilgisayarları eğitmeye başladı. Örneğin, Savunma İleri Araştırma Projeleri Ajansı, 1970'lerde sokak haritalama projelerini tamamladı. Ve DARPA, 2003'te akıllı kişisel asistanlar üretti.

Bugünlerde yapay zeka makine öğrenimi, doğal dil işleme, yapay sinir ağları, derin öğrenme, bilgisayarla görme gibi branşlara ayrılmıştır.

1.3. Makine Öğrenmesi

Makineler daha akıllı hale gelmeye başladıkça, yapay zeka araştırmacıların ve geliştiricilerin dikkatini çeken en önemli konulardan biri haline geldi. Makinelerin bir insan gibi dış dünyayı görmesi, hareket etmesi ve hatta algılaması için birçok çalışma yapılmaktadır. Makine öğrenimi ile kural ve veri verme ve sonuç olarak çıktı alma klasik programlama yöntemi, veri ve çıktı verme ve çıktı olarak kural alma gibi yeni bir paradigmaya geçmiştir [1]. Bu basit makine öğrenimi paradigması, görüntü sınıflandırmasından konuşma tanımaya kadar geniş bir görev ölçeği arasında kullanılmaktadır [1]. Makine öğrenimi, bilgisayarın belirli bir model aracılığıyla anlamasını sağlamak olarak da tanımlanabilir. Sistem veya yani model, bir dizi örnek üzerinde eğitilir ve verilerden öğrenilebilir hale getirilir.

1.4. Derin Öğrenme

Derin öğrenme, verilerden temsilleri öğrenmek için matematiksel bir çerçeve sağlayan makine öğreniminin alt alanlarından biridir [1]. Verileri doğrusal bir şekilde analiz eden geleneksel programlama biçimine rağmen, derin öğrenme yöntemi hiyerarşik yapısıyla verileri doğrusal olmayan bir şekilde işler [1]. Derin öğrenmede derinlik, birbirini izleyen katmanlardır [1]. Bir veri modeline katkıda bulunan katman sayısı, modelin derinliğidir [1].

Öğrenme sürecinde, bir katman tarafından uygulanan dönüşüm ağırlıklarına göre parametrelendirilir [1]. Örnek girdilerin ağ tarafından ilgili hedeflere doğru şekilde eşlenmesi için bir ağdaki tüm katmanların ağırlıkları için bir dizi değer bulmak anlamına gelir [1]. Derin bir sinir ağında, çıktının ağırlıklar için doğru değeri bulmak için beklenenden ne kadar uzakta olduğunu ölçmeyi gerekli kılan çok miktarda parametre vardır ve bu görev ağın kayıp işlevi tarafından ele alınır [1]. Bir ağın kayıp işlevi, ağın tahminlerini ve belirtilen çıktıyı, yani hedefi kullanarak bir mesafe puanının hesaplanmasını yapar [1].

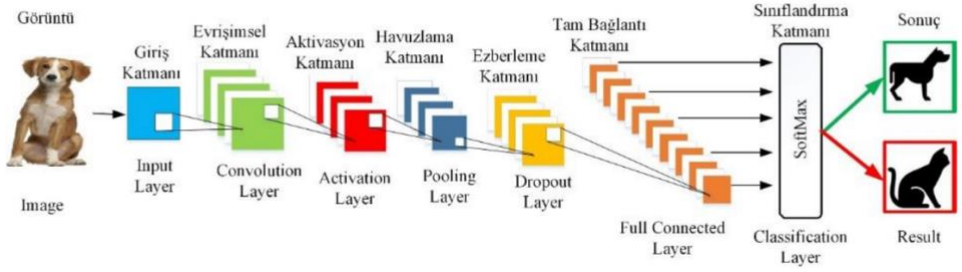
Derin öğrenme nesne sınıflandırma alanında başarısını ilk olarak 2012’ de büyük ölçekli görsel tanıma (ImageNet) [27] yarışmasında ortaya koymuştur. Yüz tanıma

sistemleri, sağlık sektörü, ses tanıma çalışmaları, otonom sistemler, görüntü işleme, gelecek tahmin uygulamaları, doğal dil işleme, metin ve karakter tanıma çalışmaları, savunma ve güvenlik, nesne tanıma, sınıflandırma gibi birçok alanda da derin öğrenme mimarileri kullanılmıştır.

1.5. Evrişimli Sinir Ağları

Evrişimli sinir ağları (CNN'ler) 1970'lerde inşa edildi ve derin sinir ağlarının erken mimarileri olarak önerildi. Derin sinir ağlarının gelişimi, CNN'lerin hızla gelişmesini kolaylaştırmıştır. Nesne algılama, ağaç sayma, palmiye ağacı algılama ve sayma, incir bitkisi segmentasyonu vb. gibi büyük 6 ölçekli bilgisayarla görme çalışmalarında kullanılıyor. Klasik görüntü işlemeden farklı olarak evrişimli sinir ağlarında nesne özellikleri, oluşturulan model tarafından kendi kendine öğrenilerek gerçekleştirilir. CNN mimarisi genel olarak bir veya daha fazla evrişim katmanı, havuzlama katmanı ve bir veya daha fazla tam bağlı katmanlardan oluşmaktadır.

Görüntülerde nesne veya daha spesifik olarak bitki tespiti, yapay zeka ile çevrili günümüz dünyasında önemli bir alan olmuştur. Yetiştiricilerin bitki türünü tespit etmesi, bir uzman bile olsalar zor olabiliyor. Bitkiler büyürken izlenmeli veya üretkenliği en üst düzeye çıkarmak için bakımları düzenli takip edilmelidir. Şekil 1'de CNN mimarisi ve katmanları verilmiştir.

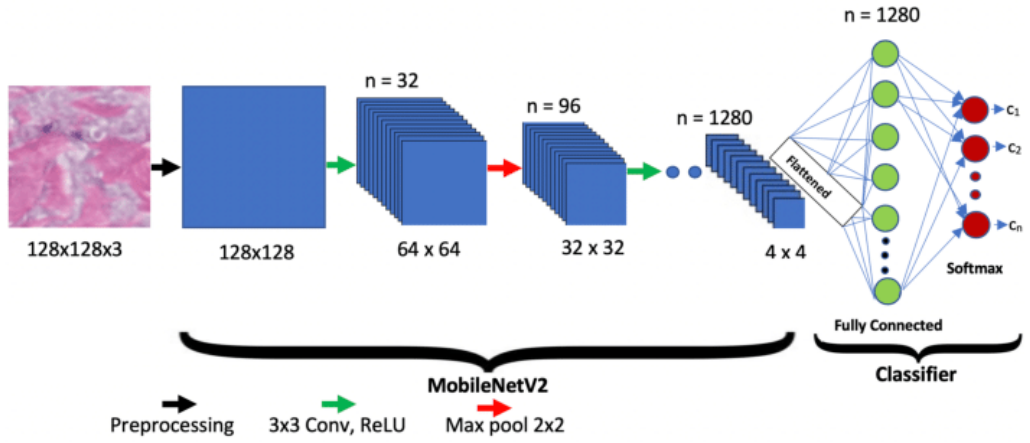


Şekil 1 CNN Mimarisi

1.6. MobileNet Modeli

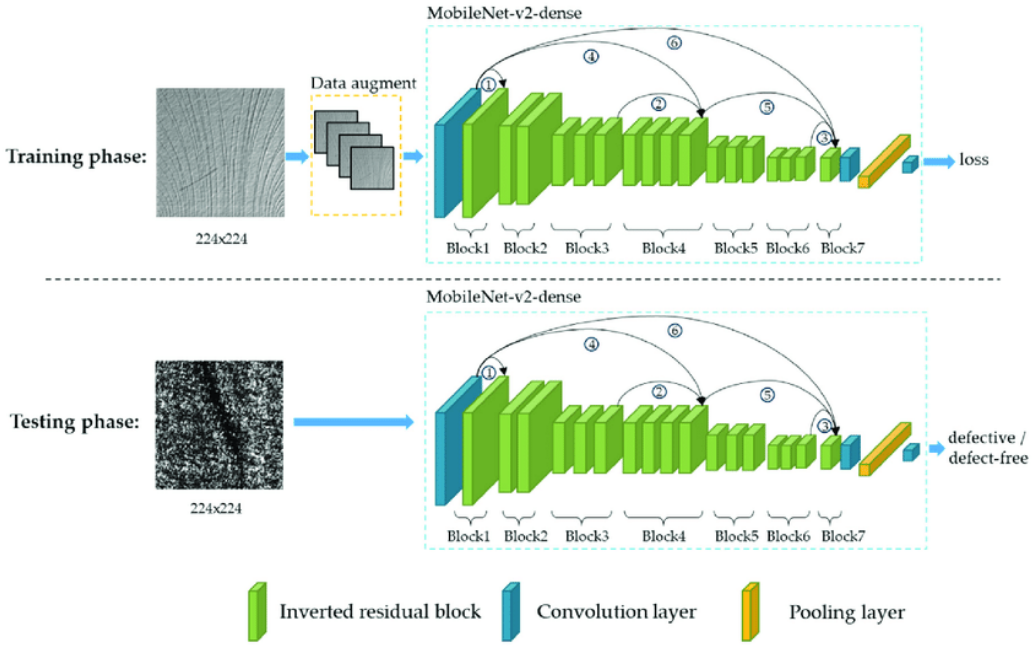
Bu model, mobil ve gömülü görme uygulamaları için tanıtılmış olup derinlik açısından ayrılabilir evrişimlere dayanmaktadır. Model mimarisi 'MobileNet-SSD V2' gerçek

zamanlı nesne tespitinde yaygın kullanılan bir diger modeldir. ‘Single Shot’ anlamı, nesne tespitinin ve sınıflandırmanın tek bir ileri yönlü hesaplamada olmasıdır. Diğer birçok popüler modelden çok daha küçük boyutta ve performans açısından daha hızlı olan, basit bir derin evrişimsel sinir ağı modelidir.



Şekil 2 MobileNet V2 mimarisi

Yüzey hata tespiti için kullanılmış MobileNet modelinin eğitim ve test fazlarının ağ tasarımı Şekil 3’de verilmiştir.



Şekil 3 MobileNet v2 Ağı Tasarımı

1.7. Xcode Editörü ve Swift Dili

Apple, IDE olarak sadece Mac OS X işletim sisteminde çalışan Xcode adlı yazılım geliştirme platformunu tercih etmektedir. Xcode ile iPhone, iPad, Watch ve Mac OS X uyumlu programlar da geliştirilebilir. Geliştirilen programları simülatör aracılığı ile, gerçek bir cihaz üzerinde karşılaşılabilecek çeşitli durumları (bellek kaçakları, cihazın yatay tutulması) test edebilirsiniz. Xcode sadece Mac OS X yüklü bir bilgisayarda çalışmaktadır.

CoreML, matematiksel ve istatistiksel yöntemler kullanarak mevcut verilerden çıkarımlar yapan yöntemler dizisidir. Apple, 2017' de geliştiricilerin kendi uygulamalarına yapay zekâ ekleyebilmeleri için Core ML adında bir kütüphane tanıttı. Xcode üzerinden bir görüntü sınıflandırıcısı, zaten etiketlenmiş olan birçok görüntü örneğini göstererek eğitilebilir ve CoreML modeli oluşturulabilir. Daha önceden eğitilmiş hazır makine öğrenimi modellerini kullanarak uygulama oluşturulabilir.[2]
Swift, Apple platformlarında yazılım geliştirmeyi sağlayan, açık kaynak kodlu bir yazılım dilidir. Otomatik bellek sayma, diğer dillerde çöp toplayıcı, özelliği vardır. Güvenilir ve uygulama geliştirme aşamasını kısaltan bir dildir.

1.8. Anaconda Editörü ve Python Dili

Conda, Windows, macOS ve Linux üzerinde çalışan açık kaynaklı bir paket ve ortam yönetim sistemidir. Conda, paketleri ve bağımlılıklarını hızla kurar, çalıştırır ve günceller. Ayrıca yerel bilgisayarınızdaki ortamları kolayca oluşturur, kaydeder, yükler ve ortamlar arasında geçiş yapar. Python programları için oluşturulmuştur, ancak herhangi bir dil için yazılımı paketleyebilir ve dağıtabilir.[3]

Python, nesne yönelimli, yorumlamalı, birimsel ve etkileşimli yüksek seviyeli bir programlama dilidir. Girintilere dayalı basit söz dizimi, dilin öğrenilmesini ve akılda kalmasını kolaylaştırır. Bu da ona söz diziminin ayrıntıları ile vakit yitirmeden programlama yapılmaya başlanabilen bir dil olma özelliği kazandırır.[4]

BÖLÜM 2. GELİŞTİRİLEN YAZILIM

2.1. Projeye Giriş

Makine öğrenimi, etiketli bitki görüntülerinden oluşan bir veri kümesi üzerinde bir model eğiterek bitkileri sınıflandırmak için kullanılabilir. Model, farklı bitki türlerinin özelliklerini tanımayı öğrenir ve daha sonra yeni, görünmeyen görüntüleri bu özelliklere göre sınıflandırır. Bu amaçla kullanılan popüler bir veri setlerinden, birçok farklı bitki türünün resimlerini ve bunlara karşılık gelen etiketleri içeren PlantNet veri seti kullanılabilir. Veri seti, sinir ağları, SVM gibi çeşitli makine öğrenimi teknikleri ve Resnet, Inception ve Mobilenet gibi mimariler kullanılarak bir modeli eğitmek için kullanılabilir.

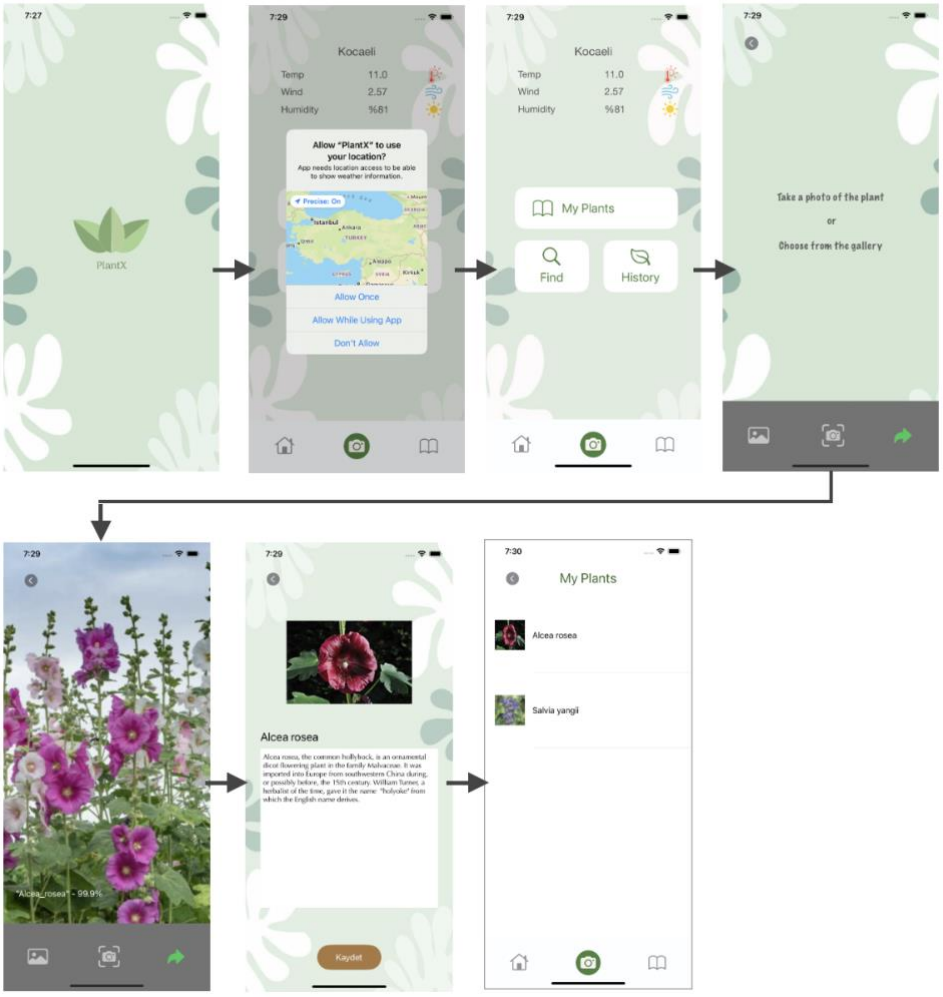
Bitki sınıflandırması için ML kullanmanın avantajlarından biri, büyük miktarda veriyi işleyebilmesi ve düşük ışık veya kısmi tıkanıklık gibi zor koşullarda bile bitkileri tanımlayabilmesidir. Ek olarak, ML modelleri mobil uygulamalara entegre edilebilir ve sahada kolay ve erişilebilir bitki tanımlamasına olanak tanır.

MobileNet mimarisi, hafif ve verimli olacak şekilde tasarlandığından mobil uygulamalarda kullanım için çok uygundur. Bu, parametre sayısını ve gereken hesaplamayı azaltan derinlemesine ayrılabilir konvolüsyonların kullanılmasıyla elde edilir. Ek olarak, model, tamamen bağlı katmanlardaki nöron sayısını azaltarak daha da optimize edilebilir, bu da gerekli hesaplama miktarını daha da azaltır. MobileNet, sınırlı bilgi işlem kaynaklarına sahip mobil ve gömülü cihazlarda çalışacak şekilde tasarlanmıştır. Bu, güç ve bellek kısıtlamalarının önemli hususlar olduğu mobil uygulamalarda kullanım için çok uygundur.

Geliştirilmek istenen uygulama IOS işletim sistemlerinde çalışan bitki tanıma uygulamasıdır. Bitki tanıma ve sınıflandırma için makine öğrenimi kullanıldı. Mobile daha uygun olduğu için evrişimli sinir ağları modeli olarak MobileNet mimarisi kullanıldı. Eğitim için gerekli görseller ve etiketler, PlantNet veri setinden alınmıştır.

2.1.1. Xcode ve kullanıcı arayüzü tasarımı

Figma web uygulamasında daha önce tasarlanan sayfalar swift yazılım dili kullanılarak yapıldı. Sayfa tasarımı yaparken basit, sade ve kullanışlı olmasına özen gösterilmesi gerekir. Uygulamada giriş sayfası ile birlikte beş sayfa tasarlanmıştır. Anasayfa içerisinde hava durumu bilgileri ve diğer sayfalara geçiş için butonlar vardır. Bitkilerim sayfasında kaydedilen, geçmiş sayfasında daha önce taratılan bitkiler listelenir. Arama sayfasında galeri veya kameradan görsel seçimi yapılır. Görsel seçiminden sonra sonuç sayfasında bitkinin türü gösterilir. Sayfalar arası geçiş yaparken veri aktarma ve kontrol sağlamak için segueler kullanıldı ve contrains ile sayfa içi düzen ve diğer cihazlar için duyarlı tasarım sağlandı. Şekil 4'te kullanıcı davranışı örneği vardır.



Şekil 4 Kullanıcı Arayüzü Şeması

2.1.2. Görsel alınması ve model için hazırlanması

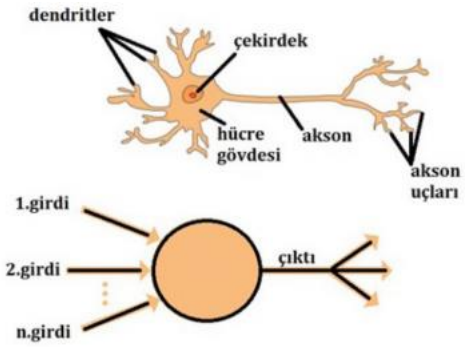
Kullanıcı ‘bul’ sayfasına geldiğinde kameradan veya galeriden görsel edinmeyi seçer. Kullanıcının galeri ve kamera erişimi için izin istenilir. Fotoğraf çekme işleminde ‘FindViewController.swift’, ‘FindViewController+CameraPicker.swift’ içerisinde bulunan ‘cameraPicker’ fonksiyonunu çağırır. Galeriden fotoğraf seçme işleminde ‘FindViewController.swift’, ‘FindViewController+PhotoPicker.swift’ içerisinde bulunan ‘photoPicker’ fonksiyonunu çağırır. Kod karmaşıklığını azaltmak ve bu yol ile farklı parçalara bölmek daha temiz, takip edilmesi ve anlaması kolay yazılım geliştirmeye yardımcı olur. Seçilen veya çekilen fotoğraf sınıflandırma işleminden önce uygun şekle ve boyutlara getirilmelidir. Görsel 224 x 224 boyutlarında ortadan kırpılır. Model girişi için hazır hale gelmiş olur.

2.2. Görüntü Sınıflandırma

Bitkilerin sınıflandırılması ve etiketlenmesi için yapayzekaya ihtiyaç vardır. Gelişmiş mimariler ile model eğitilebilir veya hazır eğitilmiş ağırlıklı modeler kullanılabilir.

2.2.1. Neural network

Bir sinir ağının temel yapısını oluşturan nöronlar yeni bir bilgi aldığında bu bilgi işlenir ve çıktıya dönüşür. Nöral ağda gelen girdi, alınır işlenir ve sonraki işlem için diğer nöronlara gönderilen bir çıktı üretir.

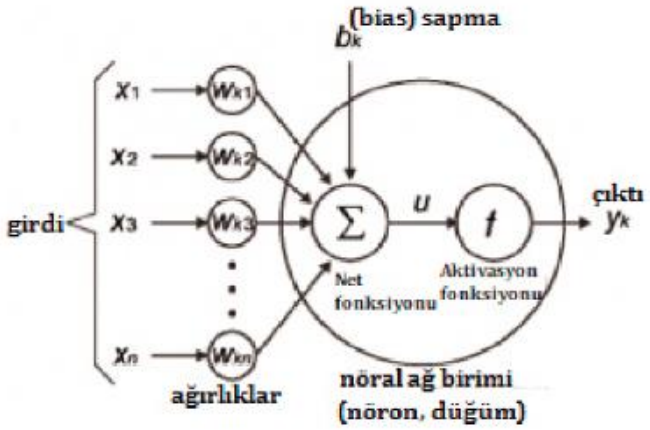


Şekil 5 Sinir Ağı

Girdiler nörona geldiğinde ağırlıklar ile çarpılır. Rastgele başlatılan ağırlıklar, model eğitimi süresince güncellenir. Sinir ağı tarafından eğitim sonrasında daha önemli olduğu düşünülen girdilere daha yüksek ağırlık değeri verilmektedir.

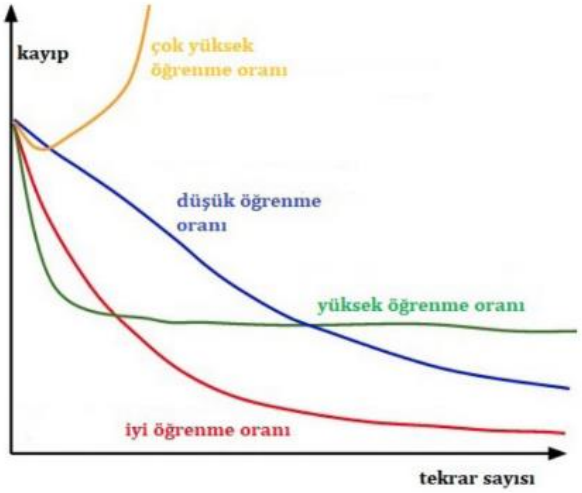
Aktivasyon Fonksiyonu

Giriş sinyallerini, çıkış sinyallerine çeviren aktivasyon fonksiyonunun, lineer bileşime uygulanması; doğrusal bileşeninin girdiye uygulanmasından sonra doğrusal olmayan fonksiyona uygulanması ile gerçekleşir. Aktivasyon fonksiyonu olan f() , uygulama sonrası çıkışı f(a*w1+b) olacaktır.



Şekil 6 Nöral ağ şeması

Ağın girdi, çıktı ve gizli olmak üzere üç katmanı vardır. Oluşturulan ağın amacı doğru değere daha yakın sonuçlar elde etmeye çalışmaktır. Ağın doğruluğu, maliyet fonksiyonları ile hesaplanır. Tahmin doğruluğu artması ve maliyet fonksiyonunun minimize edilmesi amaçlanır. Bu sebeple eğimli iniş algoritması kullanılır. Bir fonksiyonun yerel minimum değerini bulmak için, fonksiyonun negatife eğimine göre ilerlenmelidir.[9]



Şekil 7 Öğrenme oranları

Maliyet fonksiyonunu minimize etmek için yapılan her iterasyonda elde edilen en aza indirgeme miktarı, öğrenme oranıdır. Öğrenme oranları en uygun çözümü elde edebilmek için dikkatli seçilmelidir. Çok yüksek öğrenme oranının, tekrarı az ve kaybı yüksek görünüyor.

2.2.2. Cnn (konvolüsyonel – evrişimli sinir ağı)

Evrişimli sinir ağları 1970' lerde ortaya çıktı ve derin sinir ağlarının erken mimarileri olarak önerildi. Derin sinir ağlarının gelişimi, CNN'lerin hızla gelişmesini kolaylaştırmıştır. Nesne tespiti, ağaç sayımı, palmiye ağacı tespiti ve sayımı, incir bitkisi segmentasyonu vb. gibi geniş ölçekli bilgisayarla görme çalışmaları arasında kullanılırlar.

CNN'lerde her katman üç boyutlu giriş alabilir ve çıkış üretebilir. Giriş katmanının genişlik ve yüksekliği, görüntünün genişlik ve yüksekliği kadardır ve derinliği 3 olabilir (red, green, blue).

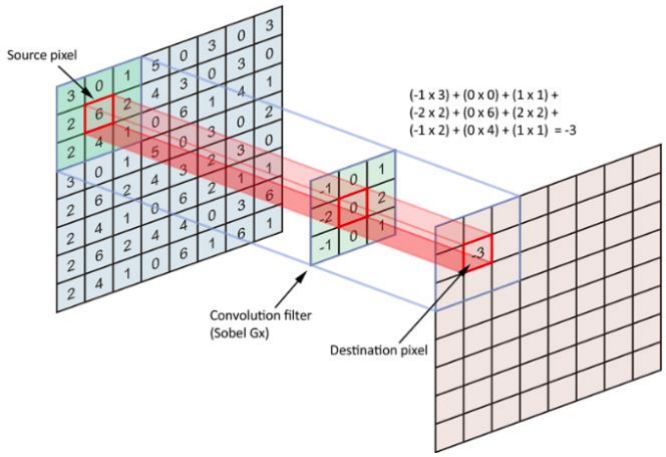
CNN'ler, katmanlar olarak düzenlenir ve her katman bir özellik öğrenir.

1. Giriş Katmanı: Girdi olarak resimleri alır ve ön işleme yapmaz. Bu katman resimleri veri setinde bulunan şekline göre boyutlandırır ve sadece özellikleri alır. Verilerin ağa sunulmak üzere standart hale getirilip, ağa sunulduğu

katmandır. Genellikle bu katmanda kullanılan mimariye ve veri setine bağlı olmak üzere normalizasyon işlemi, aşağıdaki denklem ile gerçekleşir.

$$x = \frac{x - \min(x)}{\max(x) - \min(x)}$$

2. Convolutional Katmanları: Her görüntü, genişlik, yükseklik ve derinlik boyutlarına sahip bir 3B matris olarak temsil edilir . Derinlik, bir görüntüde kullanılan renk kanallarından (RGB) dolayı bir boyuttur. Bu katmanlar, girdi resmi ile önceden tanımlanmış bir filtre kullanarak özellikleri öğrenir. Her işlemin farklı bir filtre kullandığı çok sayıda evrişim gerçekleştirilir. Bu, farklı özellik haritalarıyla sonuçlanır. Sonunda, tüm bu özellik haritalarını alıp evrişim katmanının nihai çıktısı olarak bir araya getirir.



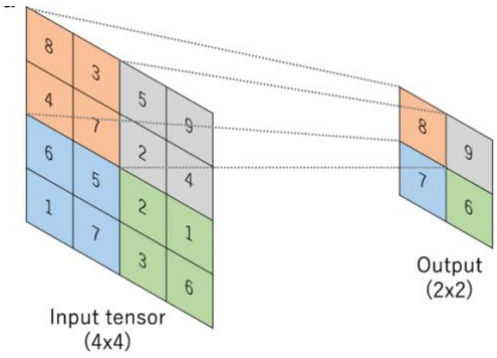
Şekil 8 Konvolüsyon Katmanı

Tıpkı diğer sinir ağları gibi çıktıyı doğrusal olmayan hale getirmek için bir aktivasyon işlevi kullanılır. Evrişimli Sinir Ağı durumunda, evrişimin çıktısı aktivasyon fonksiyonundan geçirilecektir. Bu, ReLU aktivasyon işlevi olabilir. Görüntü üzerinde dolaştırılacak filtrenin, ağın başarısına doğrudan etkisi bulunmaktadır. Örneğin ağ üzerinde dolaştırılacak matris çok büyük boyutlu ve büyük sayılabilecek rakamlardan oluşması durumunda, ağın eğitilmesi çok uzun sürebilir, hata ile karşılaşılabilir.

3. Pooling Katmanları: Bu katmanlar, önceki katmandan gelen özellikleri sıkıştırmak ve boyutlandırmak ve arasındaki bağımlılığı azaltmak için kullanılır. Genellikle max pooling veya average pooling kullanılır.

   Max pooling, verilen bir bölgede en yüksek değerli nöronu seçer ve diğer nöronları atlar. Average pooling ise, verilen bir bölgede tüm nöronların değerlerini toplar ve bölge boyutuna böler. Bu sayede, özellikler arasındaki bağımlılık azaltılır ve özellikler daha az boyutlu hale getirilir.

   Havuzlama katmanları, özellikleri sıkıştırmak ve özellikler arasındaki bağımlılığı azaltmak için kullanılır. Bu sayede, model daha az veri ile daha iyi özellikler öğrenebilir ve daha az overfitting riski oluşur.

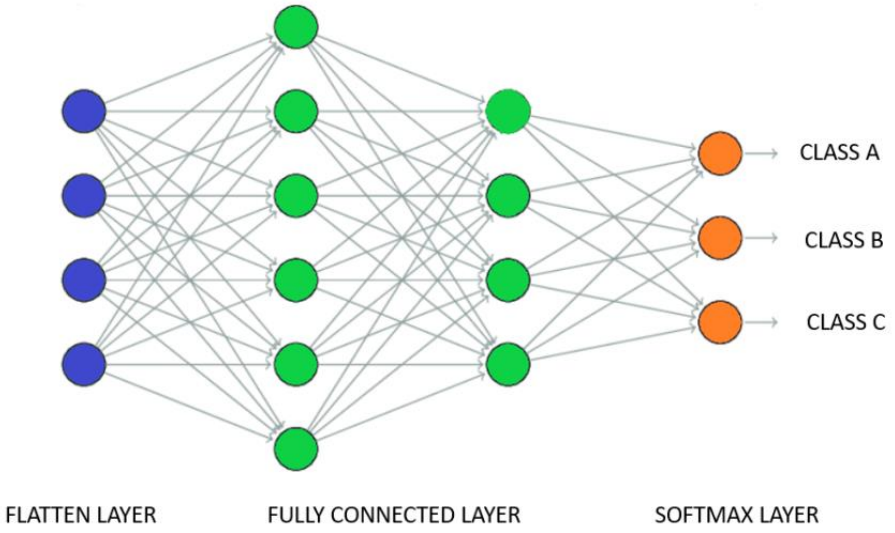


Şekil 9 Maks Havuzlama

4. Fully Connected Katmanları: Bu katmanlar, önceki katmanlardan gelen özellikleri kullanarak sınıflandırma yapar. Bu katmanlar, önceki katmanların özelliklerini birleştirir ve sınıflandırma yapmak için kullanır. Konvolüsyon ve havuzlama katmanlarından sonra, sınıflandırma katmanı birbiriyle tamamen bağlantılı birkaç katmandan oluşuyor. Tamamen bağlantılı katmanlar yalnızca 1 boyutlu verileri kabul edebilir.

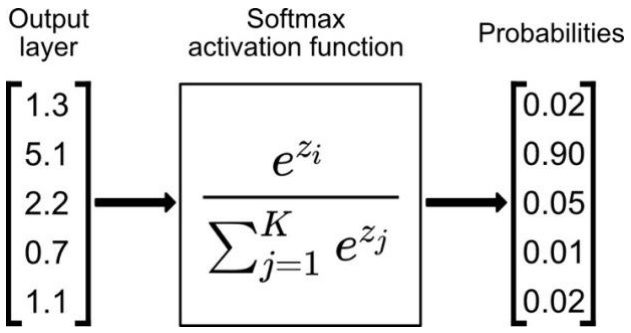
   Evrişimli bir CNN'nin son katmanları tamamen bağlantılı katmanlardır. Tamamen bağlantılı bir katmandaki nöronlar, önceki katmandaki tüm aktivasyonlarla tam bağlantılara sahiptir. Bu kısım prensip olarak normal bir

sinir ağı ile aynıdır. Şekil 10'da yeşil ile gösterilen katmanlar, tam bağlantılı katmanlardır.



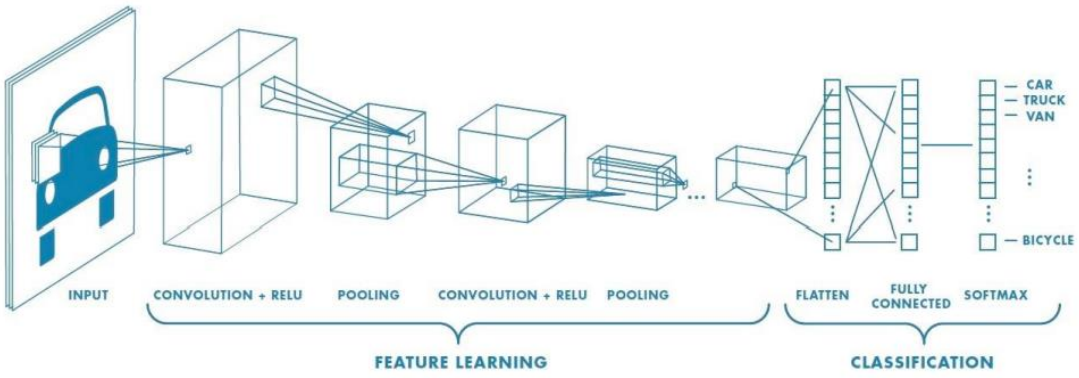
Şekil 10 Sınıflandırma Katmanları

5. Çıkış Katmanı: Bu katman, sınıflandırma yapar ve çıktı olarak sınıf etiketlerini verir. Modelin son çıktısını üreten son katmandır. Son tahmini veya sınıflandırmayı yaptığı için "sınıflandırma katmanı" olarak da bilinir. Çıkış katmanı, sınıflandırma problemindeki her sınıf için tipik olarak bir nörona sahiptir. Her nöronun çıktısı, girdi görüntüsünün o belirli sınıfa ait olma olasılığını temsil eder. Çıktı katmanı, tüm çıktıların toplamı 1'e eşit olacak şekilde her bir nöronun çıktısını 0 ile 1 arasında bir değere eşleyen bir olasılık dağılım işlevi olan bir softmax aktivasyon işlevini kullanır. Bu, çıktının farklı sınıflar üzerinde bir olasılık dağılımı olarak yorumlanmasına izin verir ve en yüksek olasılığa sahip sınıf tahmin olarak seçilebilir.



Şekil 11 Sınıflandırma Katmanı

Özet olarak; art arda çok sayıda konvülasyon ve havuzlama yapılabilir. Tam bağlantılı katmanı üç boyutlu girişi tek boyuta indirgeyerek alır ve bir sınıf etiketi elde eder. Softmax katmanı çıkış sınıflarının olasılık dağılımını hesaplar.



Şekil 12 CNN aşamaları

Bu projede bitki sınıflandırması için CNN nin mimarilerinden MobileNet kullanıldı. Diğer mimarilerden, RestNet doğru sonuç verme performansı ve daha çok parametre sayısına sahiptir. Bununla beraber çıkarım aşamasında daha yavaş performans gösterir. DenseNet-121 ve MobileNet-V2, parametre sayılarını azaltan (sırasıyla 7,98M ve 3,50M) ve mümkün olduğunca yüksek sınıflandırma doğruluğunu korurken çıkarım süresini hızlandıran gelişmiş mimarilerdir.

2.2.3. Mobilenet V2 modeli

Bu çalışmada ev bitkilerini sınıflandırmak için çalışmalar yaptım. Bitki sınıflandırma mobil uygulaması için önce uygun model seçilmesi ve oluşturulması gerekir. Bu işlem için mobil cihazlara daha uygun olacağı için MobileNet tabanlı bir ağ kullandım. Şekil 1'de gösterilen yoğun evrişimli ağ her katmanın bağlanmasında ileri besleme yapan önceden eğitilmiş bir modeldir ve bu çalışmada ince ayar için temel model olarak kullandım. Tablo 1'de MobileNet modeline ait katman yapısı gösterilmiştir. Giriş 224x224 ve çıkış, tam bağlantılı listenin girişi , 1280 adet 1 boyutlu kanallar olarak görülüyor.

Tablo 1 MobileNetv2 katman yapısı. t: genişleme faktörü, c: çıkış kanal sayısı, n: tekrar sayısı, s: adım.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Şekil 1'de kullandığım modelin katmanları Tablo 1'de daha açıklayıcı şekilde gösterilmiştir.

MobileNetV2 modelinin önceden eğitilmiş ağırlıkları Tensorflow veya Pytorch frameworkleri ile yüklenebilir. Bu projede model oluşturulup sıfırdan eğitime başlamak tercih edildi. Modeli eğitirken, kayıp işlevini, optimize ediciyi ve diğer eğitim parametreleri belirlenmelidir. Görüntü sınıflandırma görevleri için en yaygın kayıp fonksiyonu, çapraz entropi kaybıdır.

Optimize edici, kayıp fonksiyonunu en aza indirmek için modelin parametrelerini ayarlamak için kullanılan bir yöntemdir. Amacı eğitim sırasında kayıp fonksiyonunu en aza indiren en uygun parametre setini bulmaktır. SGD, Adam, Adagrad, Rmsprop ve Adadelta optimize edicileri vardır. Bu projede Rmpsprop kullanıldı.

MobileNetV2 diğer CNN modellerine göre daha az parametre ve hesaplama işlemi yapılır. Bu şekilde mobil cihazlarda çalışmaya uygun bir modeldir. Görüntüdeki nesnelerin ayrıntılarını ve gecikme sürelerini kontrol edebilir. Bu özellikleri derin öğrenme mimarilerinin sahip olduğu konvolüsyon yapıları ile sağlanmaktadır. Derinlemesine konvolüsyon(DC), derinlemesine ayrılabilir konvolüsyon(DWC), noktasal 20 konvolüsyon(PWC) yapıları vardır. MobileNetV2 mimarisi derinlemesine ayrılabilir konvolüsyon yöntemini kullanmaktadır. Ağırlık sayılarının azaltılması için çıkış kanallarının daraltıldığı bir darboğaz oluşturulduğu görülmektedir.

Darboğaz, modelin parametre sayısını ve hesaplama maliyetini azaltmak için kullanılan bir tekniktir. Bu, 3x3 derinlemesine evrişim katmanından önce 1x1 evrişim katmanı kullanılarak elde edilir. "Noktasal evrişim" olarak da adlandırılan bu 1x1 evrişimli katman, özellik haritasındaki kanal sayısını azaltarak derinlemesine evrişimli katmandaki parametre sayısını etkili bir şekilde azaltır. Parametre sayısını azaltarak, model daha hafif ve mobil ve gömülü cihazlarda çalışması için daha hızlı hale getirilebilir ve yine de iyi bir doğruluk düzeyi korunabilir.

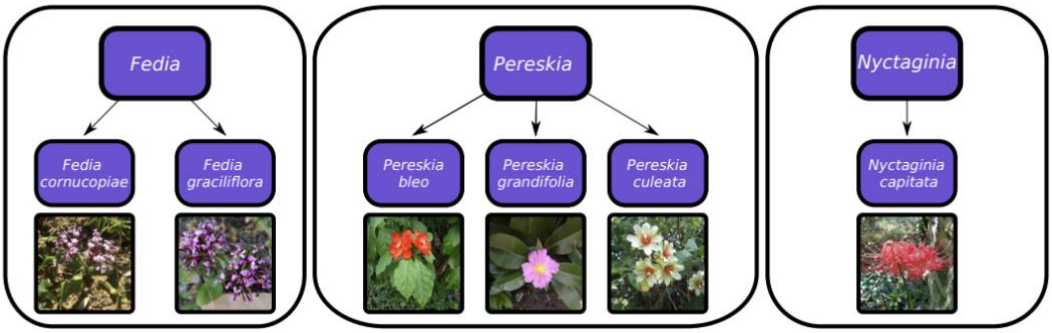
2.3. PlantNet Veri Seti

PlantNet araştırmacılarının 'P@ntNet' vatandaş gözlemevi veri tabanından oluşturulmuş, yüksek içsel belirsizliğe ve uzun kuyruklu bir dağıtıma sahip yeni bir görüntü veri seti sunmaktadır. 1.081 türü kapsayan 306.146 bitki görüntüsünden oluşur.

Veri setinin, görüntülerin elde edilme biçimine ve bitki morfolojisinin içsel çeşitliliğine özgü iki özel özellik vardır. Birincisi veri setinde güçlü bir sınıf dengesizliği olması, yani, görüntülerin çoğunu birkaç tür oluşturuyo. İkinci olarak birçok tür görsel olarak benzerdir, bu da uzman gözü için bile tanımlamayı zorlaştırır. Bu iki özellik, mevcut veri setini, set değerli sınıflandırma yöntemlerinin ve algoritmaların değerlendirilmesi için çok uygun hale getirir. Bu nedenle, veri kümesiyle ilişkili iki küme değerli değerlendirme metriği (makro ortalama üst-k doğruluğu ve makro-ortalama ortalama-k doğruluğu) öneriyoruz ve çapraz entropi kaybını kullanan derin sinir ağlarını eğiterek oluşturulan temel sonuçları sağlıyor.[12]

2.3.1. Görüntü Etiketleme

Taksonomi, biyolojide sınıflandırma bilimidir. Canlıların sınıflandırılması ve bu sınıflandırmada kullanılan kural ve prensipleri içerir.

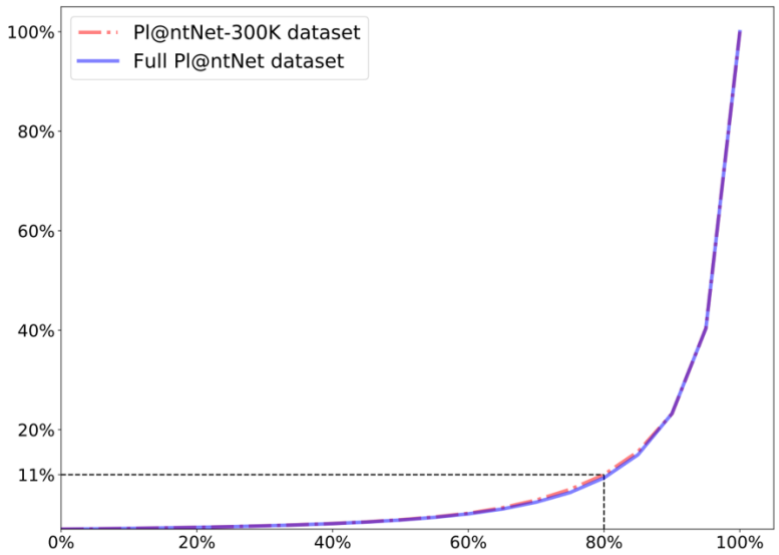


Şekil 13 Cins taksonomisi örneği

Taksonomide, türler, her cins bir veya daha fazla tür içeren ve farklı cinsler örtüşmeyen cinsler halinde düzenlenir. PlantNet veri setinde rastgele seçilen cinsler ve bu cinslere ait tüm türler tutulmuştur. Bu seçim, aynı cinse ait türler görsel özellikleri paylaşma eğiliminde olduğundan, orijinal veri tabanında bulunan büyük miktardaki belirsizliği korumayı amaçlar.

2.3.2. Epistemik (model) belirsizlik

Veri kümesinin yapısı, sınıf dengesizliğini korur. Dengesizliği göstermek için Pl@ntNet-300K veri setinin Lorenz eğrilerini Şekil 10 'da verilmiştir. Epistemik belirsizlik, temel olarak koşullu olasılıkları doğru bir şekilde tahmin etmek için gerekli verilerin eksikliğini ifade eder.



Şekil 14 Tüm türlerin kümülatif payı

Türlerin %80'inin toplam görüntü sayısının yalnızca %11'ini oluşturduğunu görülüyor [Şekil 14]. Bu nedenle, makine öğrenimi modellerini eğitmek, böyle bir veri kümesi için zorlayıcıdır, çünkü birçok sınıf için model üzerinde eğitilecek az görüntü vardır ve bu türler için tanımlamayı zorlaştırır.

2.3.3. Diğer Veri Setleri ve PlantNet

PlantNet-300K özellikleri onu diğer çeşitli görevler için ideal bir aday haline getiriyor. Güçlü sınıf dengesizliği, araştırmacılar tarafından sınıf dengesizliğinin üstesinden gelmek için özel olarak tasarlanmış yeni algoritmaları değerlendirmek için kullanılabilir.

Tablo 2 Birkaç veri setinin Pl@ntNet-300K ile karşılaştırılması

	Döngü etiketlemede insan	Uzun kuyruklu dağıtım	Sınıf içi değişkenlik	Odaklanmış etki alanı	Belirsizliği koruyan örnekleme
Plant disease dataset	✗	✗	✗	✓	✗
CUB200	✗	✗	✗	✓	✗
Oxford flower dataset	✗	✗	✓	✓	✗
Aircraft dataset	✓	✗	✗	✓	✗
Compcars	✗	✗	✗	✓	✓
Census cars	✗	✗	✗	✓	✓
ImageNet	✗	✗	✓	✗	✗
iNat2017	✓	✓	✓	✗	✗
Pl@ntNet-300K	✓	✓	✓	✓	✓

“Odaklanmış alan”, veri setinin tek bir kategoriden (yani arabalardan) oluşup oluşmadığını ve “Belirsizliği koruyan örnekleme”, veri setinin oluşturulmasında, sınıf hiyerarşisinde aynı ebeveyne ait tüm sınıfların tutulup tutulmadığını gösterir. (bizim durumumuzda, ebeveyn cins düzeyine karşılık gelir).

2.4. Model Oluşturma Ve Eğitim

2.4.1. Veri seti içeriği

Toplam boyutu 30gb olan klasör içeriğinde “train”, “val” ve “test” adlı üç alt klasör bulunmakta. Bu klasörlerin her biri 1.081 alt klasör içerir. ‘plantnet300K_species_names.json’ dosyasında alt klasörlerin isimleri ile sınıfların

isimleri arasındaki yazışma sağlanıyor. Modelden sağlanan sayısal çıktı bu dosyadaki tür adı ile eşleşiyor. Ayrıca her görüntü için "plantnet300K metadata.json" adlı bir meta veri dosyasında, tür tanımlayıcısı (sınıf), bitkinin organı (çiçek, yaprak, ağaç kabuğu, . . .), yazarın adı, lisans ve bölme (ör. tren, doğrulama veya test seti) bilgileri yer alıyor.

```
(base) rabiaabdioglu@Rabia-MacBook-Air PlantNet % ls
PlantNet_Train.ipynb    model                   plantnet_300K.zip.1
Untitled.ipynb          plantnet_300K
(base) rabiaabdioglu@Rabia-MacBook-Air PlantNet % cd plantnet_300K
(base) rabiaabdioglu@Rabia-MacBook-Air plantnet_300K % LS
README                          plantNet.ipynb
images_test                     plantnet300K_metadata.json
images_train                    plantnet300K_species_names.json
images_val
(base) rabiaabdioglu@Rabia-MacBook-Air plantnet_300K %
```

Şekil 15 Plantnet_300K dosya hiyerarşisi

2.4.2. Model oluşturma

Eğitim için önce train, validation ve test klasörleri oluşturuldu. Tensorflow kullanılarak, 'Mobilenet v2' mimarisi ve 'imagenet' ağırlıkları ile model oluşturuldu.

```
base_model = tf.keras.applications.MobileNetV2(
    input_shape = (224,224,3),
    alpha=1.0,
    include_top=False,
    weights="imagenet",
    pooling=None,
    classifier_activation="softmax",
)
```

Şekil 16 Model oluşturma

Son üç katman, oluşturulan temel modele eklenir. Bunlar, temel modeli dondursak bile kendi özel verilerimizi sınıflandırmamıza yardımcı olacaktır. Ayrıca son katman, sınıflandırma etiketlerimizin boyutuyla eşleşecek 1081 düğüme sahiptir. Belirli verileri öğrenmek için eğitilebilir son katmanları eklenir.

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation="relu")(x)
predictions = Dense(1081, activation="softmax")(x)
```

Şekil 17 Eğitilebilir katman ekleme (softmax)

Optimize ediciler, kayıpları azaltmak için sinir ağınızın ağırlıklar ve öğrenme hızı gibi niteliklerini değiştirmek için kullanılan algoritmalar veya yöntemlerdir.

```
OPTIMIZER = 'rmsprop'

opt = 'rmsprop'

# compile the model (should be done *after* setting layers to non-trainable
model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics= ["accuracy", 'top_k_categorical_accuracy'])
```

Şekil 18 Optimizer kullanımı

Son olarak model oluşturma bitirilir ve orta katmanlar dondurulur.

```
FREEZE_LAYERS = False

base_model.trainable = not FREEZE_LAYERS

print("Model is trainable: " + str(base_model.trainable))
```

```
Model is trainable: True
```

Şekil 19 Model orta katmanları dondurma

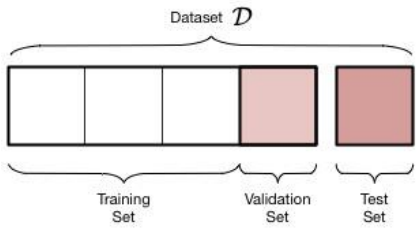
2.4.3. Eğitim

Epoch, makine öğreniminde kullanılan bir terimdir ve makine öğrenme algoritmasının tamamladığı tüm eğitim veri kümesinin geçiş sayısını gösterir. Veri kümeleri özellikle veri miktarı çok büyük olduğunda partiler halinde gruplanır. Epoch 5 alınması istenilen sonuçları veremeyecektir. Donanım yetersizliklerinden ve veri setinin büyük olmasından kaynaklı, epoch değeri 5 verilmiştir.

```
epochs = 5
steps_per_epoch = None
# val_dataset = None
history = model.fit(train_dataset, validation_data=val_dataset, epochs=epochs, steps_per_epoch=steps_per_epoch)
```

Şekil 20 Model eğitimi

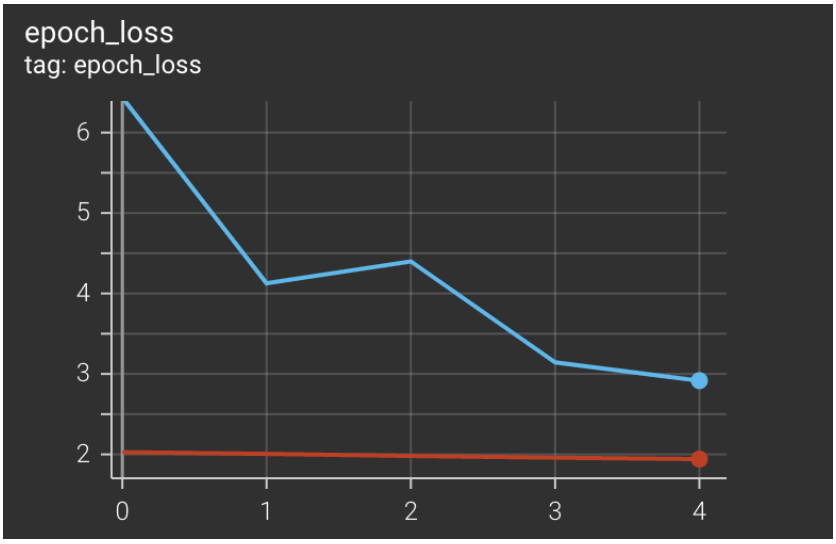
Validation set, test veri setinin öğrenme sırasındaki temsili halidir. Training set içerisinden seçilir fakat optimizasyon sırasında kullanılmaz. Bu nedenle öğrenme sürecini doğrudan etki eder. Her iterasyon sonunda (epoch) validation set ve test set ile modelin başarımı ayrı ayrı ölçülür. Validation set ile elde edilen sonuç geri yayılım işleminde kullanılırken, test veri setinin sonucu öğrenme işleminin hiçbir basamağında kullanılmaz. Test veri seti, öğrenme sürecinde adım adım modelin başarımını ölçen tamamen bağımsız bir test kümesidir. Buna karşı validation set her bir iterasyonda training setin farklı bir parçası olacak şekilde yenilenir ve modelin öğrenme sürecine doğrudan katkı sağlar. Plant Net veri setinde 300.000 görselin, %81 eğitim, %10 doğrulama (validation), %9 test için ayrılmıştır.



Şekil 21 Veri Seti Oranları

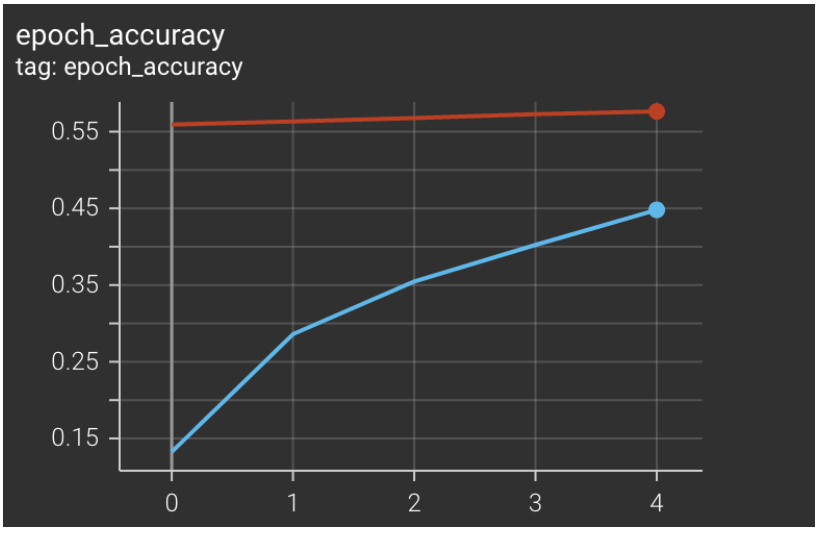
Loss, modelin eğitim verilerinde ne kadar doğru tahmin yaptığını ölçer. Daha düşük bir loss değeri, modelin daha iyi performans gösterdiği anlamına gelir. Validation loss ise, eğitim verilerinden ayrı olarak kullanılan doğrulama verilerinde modelin ne kadar doğru tahmin yaptığını ölçer. Bu değer, modelin eğitim verilerinde iyi performans göstermesine rağmen doğrulama verilerinde iyi performans gösterip göstermediğini gösterir. Yapılan eğitim sonucunda kayıp değeri 2.027 ile başlayıp 1.941'e yakınsarken, doğrulama(validation) kayıp değeri 6.43 ile başlayıp 2.917' e düşmüştür.

Şekil 22 ve 23 te görülen grafiklerde, kırmızı çizgiler eğitim, mavi çizgiler doğrulama verilerini gösteriyor.



Şekil 22 Eğitim kayıp grafiği

Epoch accuracy, bir modelin eğitim sürecindeki doğruluk değerini ifade eder. Bu değer, eğitim verilerinde modelin kaç tahmininin doğru olduğunu ölçer. Örneğin, eğitim verilerinde 100 örnek varsa ve model 80 tanesini doğru tahmin etti, epoch accuracy değeri 0.8 olur. Bu değer genellikle epoch bazında hesaplanır, yani model her epoch için ayrı ayrı hesaplanır. Bu, modelin eğitim süresi boyunca nasıl ilerlediğini gösterir.



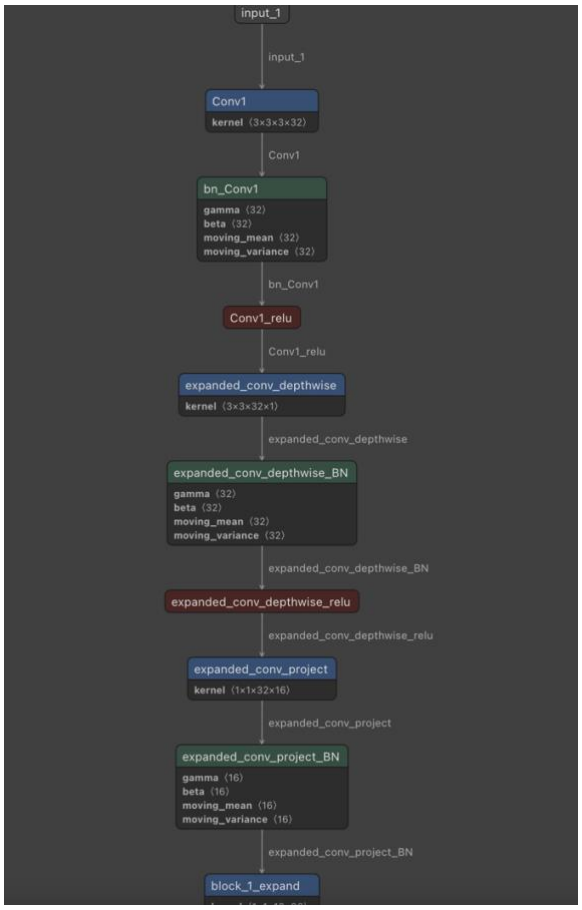
Şekil 23 Eğitim doğruluk grafiği

Eğitim bittikten sonra Keras olarak dondurulan model, Coreml modele dönüştürülmüştür.

2.4.4. Oluşturulan modelin katmanları

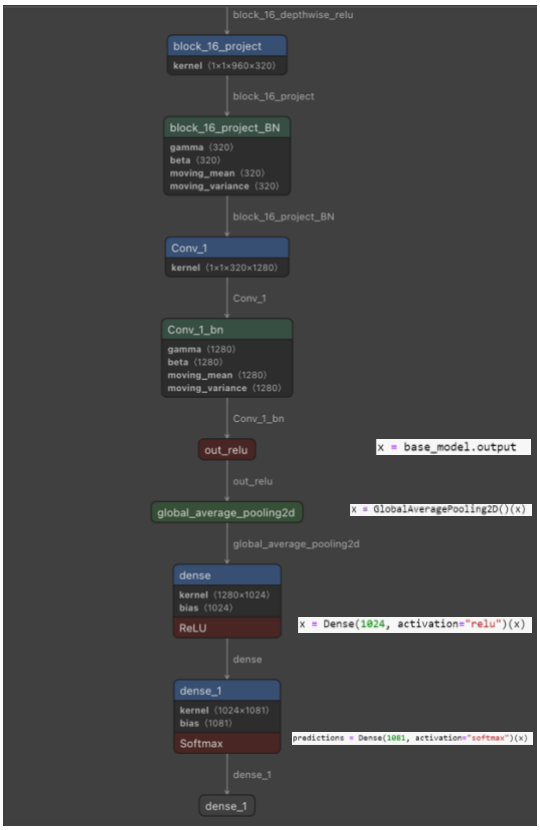
Netron üzerinden eğitilen modele ait katmanlar görülüyor. Şekil 24 ve 25 convolutional katmanları, Şekil 26 da ortalama havuzlama ve softmax katmanlarını görebiliriz.

Conv_1 girişte (3x3x3x32) , çıkışta (1x1x320x1280) olmuştur. Şekil 25 te görülen döngü 15 kez gerçekleşmiştir.



Şekil 24 Model Katmanı I

Şekil 25 Model Katmanı II



Şekil 26 Model Katmanı III

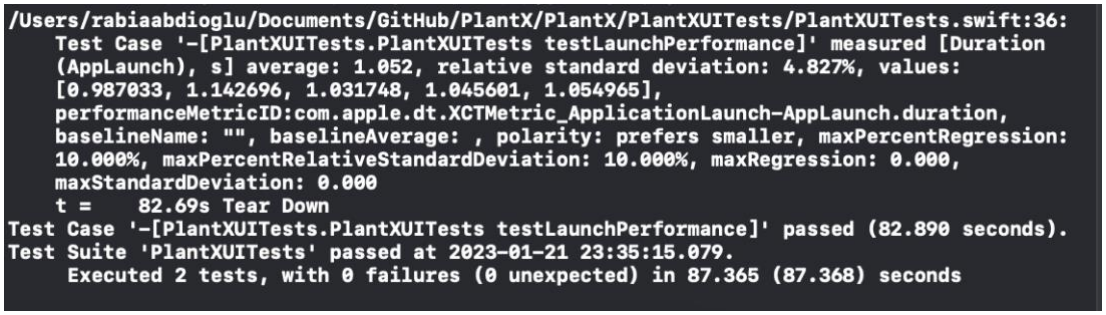
2.5. Versiyon Kontrolü

Sürüm kontrolü olarak Github kullanıldı. Daha hızlı, sade ve geri dönülebilir olduğu için kullanımı önemlidir. Projeye ait tüm kaynaklar, kullanılan teknolojiler ve AppStore linki, Github sayfasında bulunmaktadır. [14]

Proje geliştirildiği süre boyunca IOS uygulamasının yazılım versiyon takibi Git ile yapılmıştır.

2.6. Yazılımın Ve Uygulamanın Testi

Proje kullanıcı testi Apple connect tarafından ve TestFlight ile yapılmıştır. Yazılım testi Xcode üzerinden yapılmıştır. iPad ve iPhone cihazlarında, IOS 16 ve üzeri için yapılmıştır.



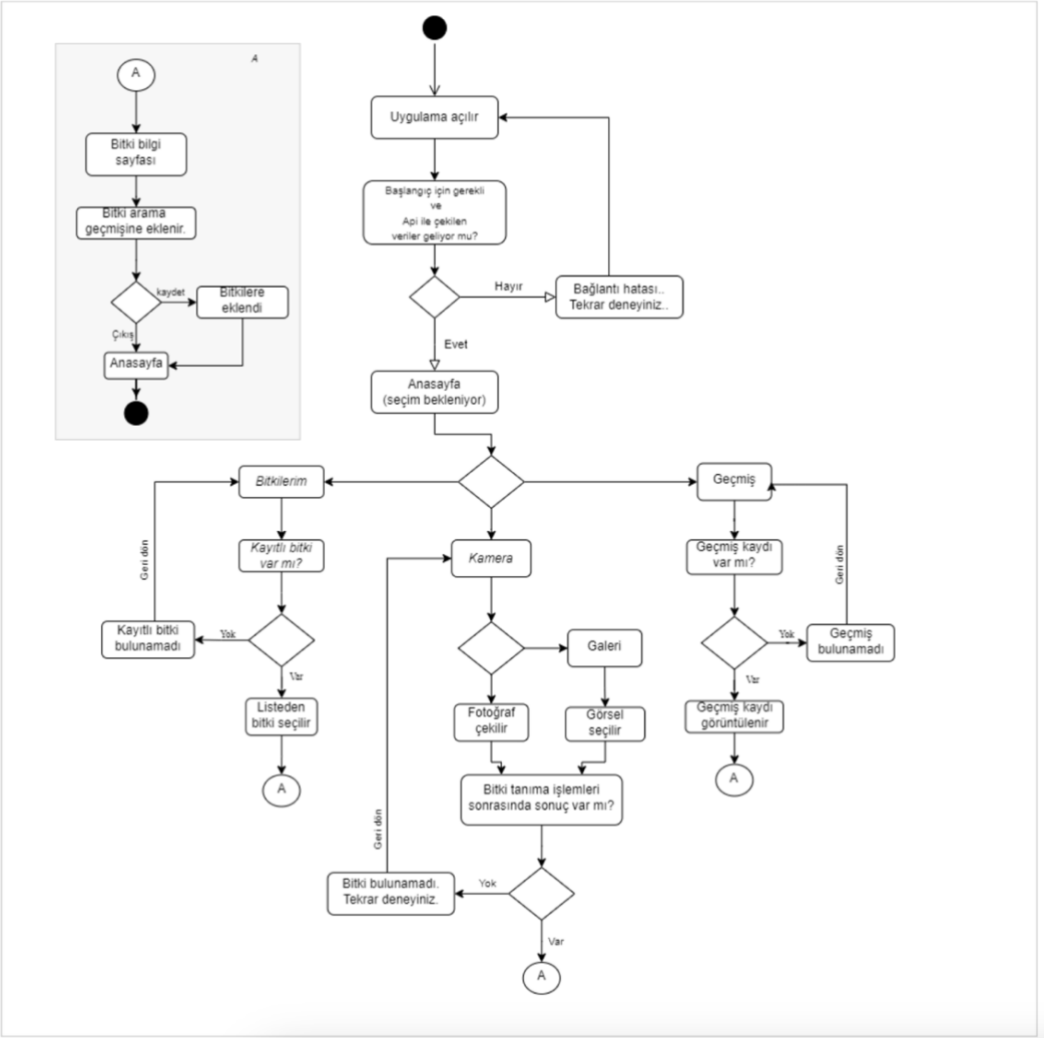
Şekil 27 Xcode Performans Testi

Uygulama yayınlama süresi boyunca yazılım iyileştirmeleri yapılmış. Hatalar giderilmiştir. Uzun süren bu süreçte, uygulama, test ekibi tarafından birçok kez geri bildirim almış ve son olarak kabul edilmiştir.

2.7. Kullanıcı Kullanım Durum Ve Veritabanı Diyagramı

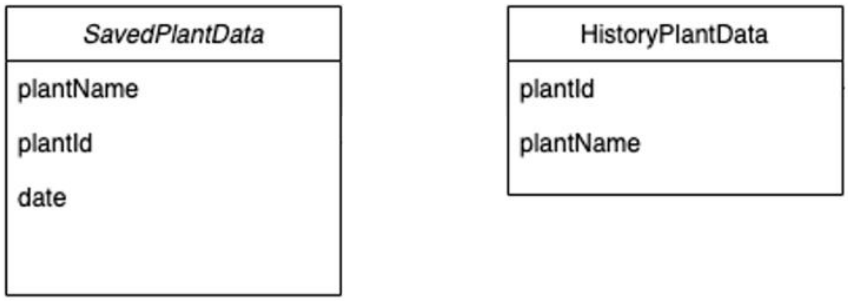
Şekil 4'teki örnek kullanıcı davranışı için Uml diyagramında izlenecek yol:
"Uygulama açılır, Başlangıç için lokasyon ve hava durumu geliyor mu? , Anasayfa, Kamera (Bul) sayfası, Galeri, Görsel Seçilir, Bitki tanıma işlemi sonuç var mı? (Alcea

Rosea), A, Bitki bilgi sayfası, Bitki arama geçmişine eklendi, Kaydet, Bitkilere eklendi, Bitkilerim sayfası”



Şekil 28 Use-case diyagramı

Uygulama içerisinde bulunan veya kaydedilen bitkileri cihazda depolamak için CoreData kullanıldı. Geçmiş ve kayıtlı bitkiler için birbirinden bağımsız tablolar oluşturuldu.



Şekil 29 CoreData Diyagramı

2.8. Yazılım Gereksinimleri

- Uygulama, IOS 16 ve üzeri cihazlar için indirilebilir olacaktır.
- Uygulama Coreml model ile aldığı görselden sonuç döndürecektir.
- Kullanıcı sonuç bitkiyi, telefona kaydedebilecektir.
- Kullanıcının daha önce bulduğu bitkiler geçmişe kaydedilecektir.
- Kullanıcı veritabanında tutulan bitki bilgilerini silebilir.
- Kullanıcı uygulamayı silmesi halinde yedek tutulmaz.
- Kullanıcı hava durumu bilgilerini görmek için konum izni vermelidir.
- Kullanıcı fotoğraf çekebilmek için kamera erişimi izni vermelidir.
- Kullanıcı fotoğraf seçebilmek için fotoğraf galerisine erişim izni vermelidir.
- Bitki bilgileri Wikipedia ile getirilecektir.
- Hava durumu bilgileri Openweathermap ile getirilecektir.

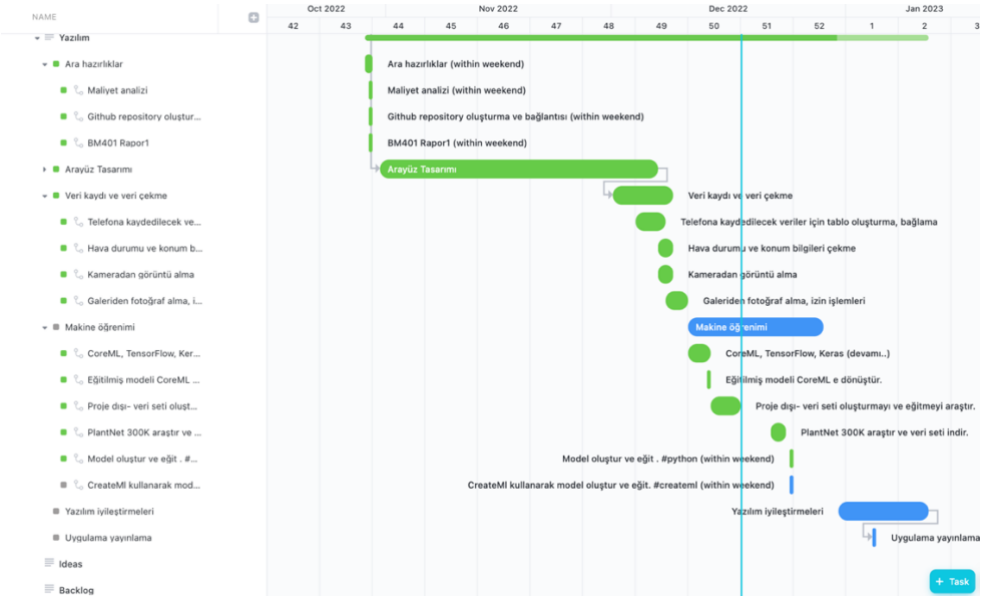
BÖLÜM 3. SONUÇ VE DEĞERLENDİRME

3.1. Yapılan Uygulamadan Beklentiler Ve Sonuçlar

Yapay zeka ve IOS mobil uygulama geliştirme alanlarının beraber kullanımı düşünüldü. Mobil uygulama başarı ile yapıldı ve sonuçlandı. Yapılan kullanıcı ve performans testlerinde bir sorun ile karşılaşılmadı.

Model eğitiminde daha doğru sonuçlar vereceği ve eğitim başarısını yükselteceği için büyük bir veri seti kullanımı planlandı. Oxford 102, iNaturalist ve PlantNet veri setleri denendi. PlantNet veri setinde karar kılındıktan sonra model eğitimi konusunda araştırmalar yapıldı. Veri setinin büyüklüğünden kaynaklı eğitimin süresi uzun olması, donanım yetersizliklerinden dolayı istenilmeyen bir durum oldu. Bu sebep ile epoch değeri 5 verildi. Model eğitimi başarılı fakat yetersiz oldu. Araştırmaya harcanan zaman daha verimli kullanılması durumunda bu zaman eksikliği giderilebilir ve yeterli bir model ortaya konulabilirdi. Modele [18] ve projeye [15] ait kodlara kaynakçada yer verildi.

3.2. Proje Sınırları



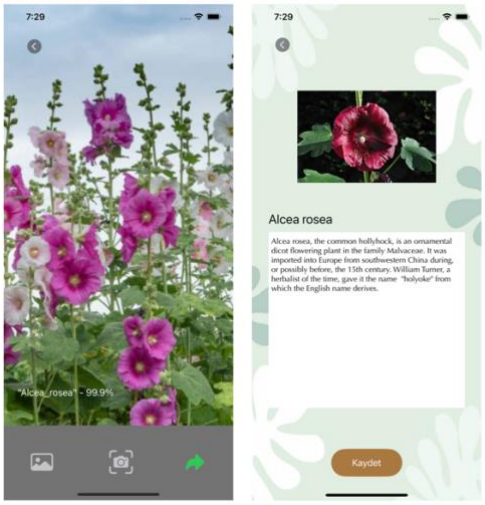
Şekil 30 Clickup Gantt şeması

Tablo 3 Proje sınırları

Hafta	1-4	5-8	9-11	12-14
Yapılan işlemler	Araştırma, Proje sınırları belirleme	Araştırma, Tasarım, Analiz	Yazılım, Xcode tasarım, Yapay zeka	Yazılım, Yayınlama, Test

3.3. Uygulama Çıktıları

Şekil 31 Uygulama Çıktısı I



Şekil 32 Uygulama Çıktısı II

KAYNAKLAR

[1] Chollet, F. 2018, Deep Learning with Python, Manning Publications Co., Shelter Island, 28-31.

[2] https://developer.apple.com/machine-learning/models/ (Kasım 2022)

[3] https://www.anaconda.com/products/distribution (Kasım 2022)

[4] https://tr.wikipedia.org/wiki/Python (Kasım 2022)

[5] https://www.figma.com/file/VGO1adrPSqYxe90HqE6KWe/PlantX?node-id=0%3A1&t=zmYyMILo1nH5J2zW-1

[6] https://developers.google.com/ml-kit (Kasım 2022)

[7] https://cocoapods.org/ (Aralık 2022)

[8] https://tr.wikipedia.org/wiki/TensorFlow (Kasım 2022)

[9] https://www.tensorflow.org/lite/models/trained (Kasım 2022)

[10] https://sharing.clickup.com/36284421/l/h/12ka05-3700/c5bd831d1873432

[11] https://github.com/plantnet/PlantNet-300K (Aralık 2022)

[12] C. Garcin and A. Joly and P. Bonnet and A. Affouard and \JC Lombardo and M. Chouet and M. Servajean and T. Lorieul and J. Salmon , NeurIPS Datasets and Benchmarks 2021

[13] https://tfhub.dev/google/lite-model/aiy/vision/classifier/plants_V1/3

[14] https://openweathermap.org/api

[15] https://github.com/rabiaabdioglu/PlantX

[16] https://github.com/Raureif/WikipediaKit

[17] https://github.com/osmr/imgclsmob/blob/

[18] https://github.com/isakdiaz/mobilenetv2_coreml/blob/main/mobilenet_v2.py

ÖZGEÇMİŞ

Rabia Abdioğlu, 29 Ocak 1998'de İstanbul Bakırköy'de doğmuştur. 2018 yılında İstanbul Aydın Üniversitesinde Bilgisayar Programcılığı bölümüne başlamıştır. 2019 yılında memleketi Trabzonda bulunan Phi Software şirketinde stajlarını yapmıştır. 2019 – 2020 öğrenim yılı boyunca dönem içi stajını Clipart Bilişim Tasarım ve Danışmanlık şirketinde tamamlamış ve daha sonra Bilgisayar Programcılığı Bölümü'nden mezun olmuştur. Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde eğitimine devam etmektedir. 2022 yılında Toyota Tsusho otomotiv şirketinde IT olarak staj yapmıştır.