İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİ İLE BİTKİ TÜRÜ SINIFLANDIRMA PROJESİ

Yazılım Mühendisliği Ana Bilim Dalı

Dönem Projesi

Can ŞENSU

Y220240091

Proje Danışmanı: Dr. Öğr. Üyesi Mansur Alp TOÇOĞLU

Ocak 2024

MAKİNE ÖĞRENMESİ İLE BİTKİ TÜRÜ SINIFLANDIRMA PROJESİ

Öz

Bu projede, Oxford Üniversitesi tarafından sağlanan 102 kategorili çiçek veri seti üzerinde bir bitki türü tanıma makine öğrenimi modeli geliştirildi. VGG-16 modeli kullanılarak gerçekleştirilen çalışmada, çiçek görüntüleri sınıflandırıldı ve modelin eğitim süreci takip edildi. Eğitim sonuçları, modelin başarılı bir şekilde çiçek türlerini tanıma yeteneğine sahip olduğunu gösterdi. Gelecekteki çalışmalarda, modelin doğruluğunu artırmak ve farklı çiçek türlerini tanıma yeteneğini güçlendirmek üzerine odaklanılabilir. Bu çalışmanın, bitki türü tanıma konusunda makine öğrenimi alanındaki araştırmalara katkıda bulunması amaçlanmaktadır.

Anahtar Sözcükler: Bitki türü tanıma, makine öğrenimi, VGG-16 modeli, çiçek sınıflandırma, Oxford 102 çiçek veri seti.

MACHINE LEARNING-BASED PLANT SPECIES CLASSIFICATION PROJECT

Abstract

In this project, a machine learning model for plant species recognition was developed on the 102-category flower dataset provided by the University of Oxford. Using the VGG-16 model, flower images were classified, and the training process of the model was monitored. The training results demonstrated that the model successfully possesses the capability to recognize flower species. Future work may focus on improving the model's accuracy and enhancing its ability to recognize different types of flowers. This study aims to contribute to research in the field of machine learning for plant species recognition.

Keywords: Plant species recognition, machine learning, VGG-16 model, flower classification, Oxford 102 flower dataset.

Teşekkür

Proje çalışmasına katkılarından dolayı proje danışmanım Doç. Dr. Mansur Alp Toçoğlu'na ve tüm dönem arkadaşlarıma teşekkür ederim.

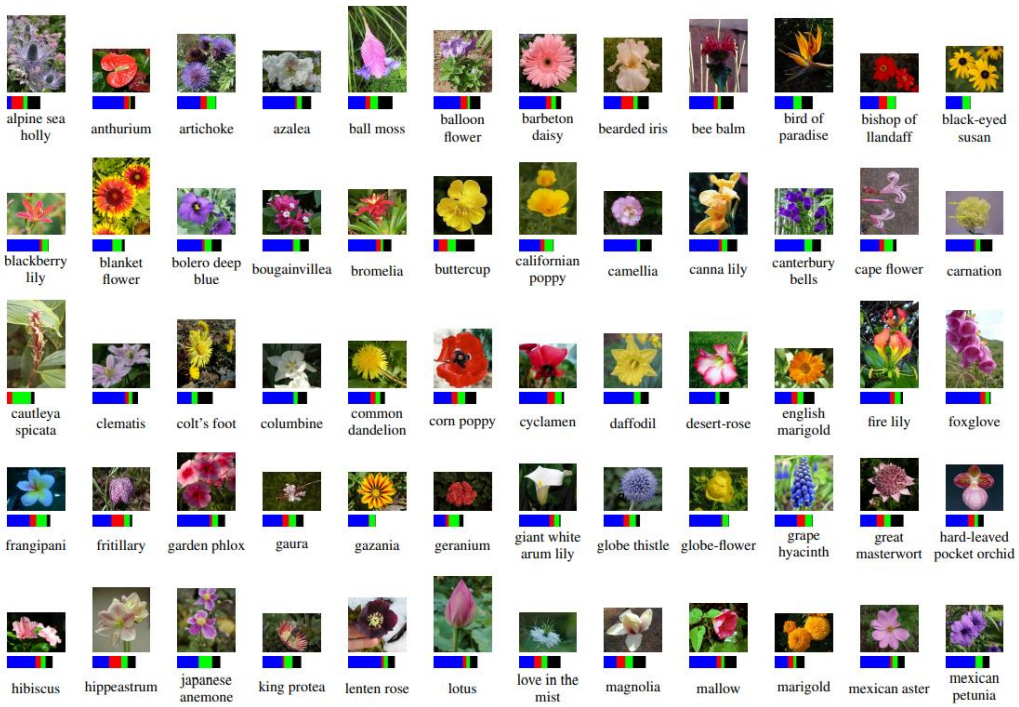
İçindekiler

Şekiller Listesi

Bölüm 1

Giriş

Görüntü tabanlı sınıflandırma sistemleri geliştikçe, nesneleri sınıflandırma görevi daha fazla kategoriye sahip veri setlerine geçmektedir. Bu makalede, birçok farklı kategoriyi tanımak yerine, tek bir kategori içinde birçok sınıfı tanıma sorununu incelemekteyiz - o da çiçeklerin kategorisi. Çiçekleri sınıflandırmak, bisikletler, arabalar ve kediler gibi kategorilerle karşılaştırıldığında ekstra bir zorluk oluşturur, çünkü sınıflar arasında büyük benzerlik vardır. Ayrıca, çiçekler birçok şekilde deforme olabilen esnek nesnelerdir ve bu nedenle sınıflar arasında büyük bir varyasyon bulunmaktadır. Burada, çiçek sınıflandırması için 102 sınıflı bir veri seti tanıtıyoruz. Veri setindeki bazı görüntüler şekil 1'de gösterilmiştir.



Şekil 1. Veri Seti Görselleri

Bir çiçeği diğerinden ayıran şey bazen renk olabilir, örneğin çanak çiçeği vs. ayçiçeği, bazen şekil olabilir, örneğin nergis vs. karahindiba, ve bazen taç yapraklardaki desenler olabilir, örneğin menekşeler vs. zambaklar vb. Zorluk, renk, şekil, desen vb. temsil etmek için uygun özellikleri bulmakta ve ayrıca sınıflandırıcının hangi özelliği veya özellikleri kullanmayı öğrenme kapasitesinde yatar.

Bu makalede, çiçek görüntülerinin oldukça kontrolsüz görüntü durumları altında elde edildiği görüntülerin çoğunlukla web'den indirildiği ve ölçek, çözünürlük, aydınlatma, karma, kalite vb. bakımından önemli ölçüde değiştiği, durumu için bu çoklu çekirdek öğrenme yaklaşımını incelemekteyiz.

Çiçek sınıfına uygun renk, doku ve taç yaprakların (yerel ve global) düzenini yakalayan özellikleri ve buna karşılık gelen çekirdekleri tasarlıyoruz.

Bölüm 2

İlgili Çalışmalar

Bitki türü tanıma konusundaki araştırmalar, gelişen teknoloji ile birlikte çeşitli yenilikçi yöntemlere ev sahipliği yapmaktadır. Bu yazıda, bitki türü tanıma projelerine dair önemli çalışmaların incelenmesi ve bu çalışmalardan elde edilen başarıların detaylı bir değerlendirmesi yapılmaktadır.

Plant Species Recognition Using Convolutional Neural Networks, bu çalışmada, evrişimli sinir ağları (CNN) kullanılarak bitki türü tanıma hedeflenmiştir. Farklı veri setlerinde test edilen CNN modelleri, renk, yaprak deseni ve morfolojik özellikleri öğrenerek yüksek doğruluk elde etmiştir.

Fine-Grained Plant Classification Using Capsule Networks, kapsül ağları ile detaylı bitki sınıflandırma üzerine odaklanan bu çalışma, bitki parçalarının hiyerarşik temsilini öğrenerek benzer türler arasındaki ince ayrıntılara odaklanmıştır.

Deep Transfer Learning for Plant Classification, transfer öğrenme yöntemlerini inceleyen bu çalışma, önceden eğitilmiş derin öğrenme modellerinin bitki türü tanıma görevinde nasıl kullanılabileceğini araştırmış ve az etiketli veri setlerinde model performansını artırmıştır.

Plant Identification Using Leaf Image Retrieval System, yaprak görüntüleri kullanılarak bitki türlerini tanıma amacı güden bu sistem, yaprak deseni ve yaprak kenarları gibi özellikleri kullanarak başarılı bir bitki tanıma işlemi gerçekleştirmiştir.

Automated Plant Species Identification Using Transfer Learning, transfer öğrenme yaklaşımının kullanıldığı bu çalışmada, önceden eğitilmiş modellerin bitki türü tanıma görevinde nasıl adapte edilebileceği üzerinde durulmuş ve az etiketli veri ile yüksek doğruluk elde edilmiştir.

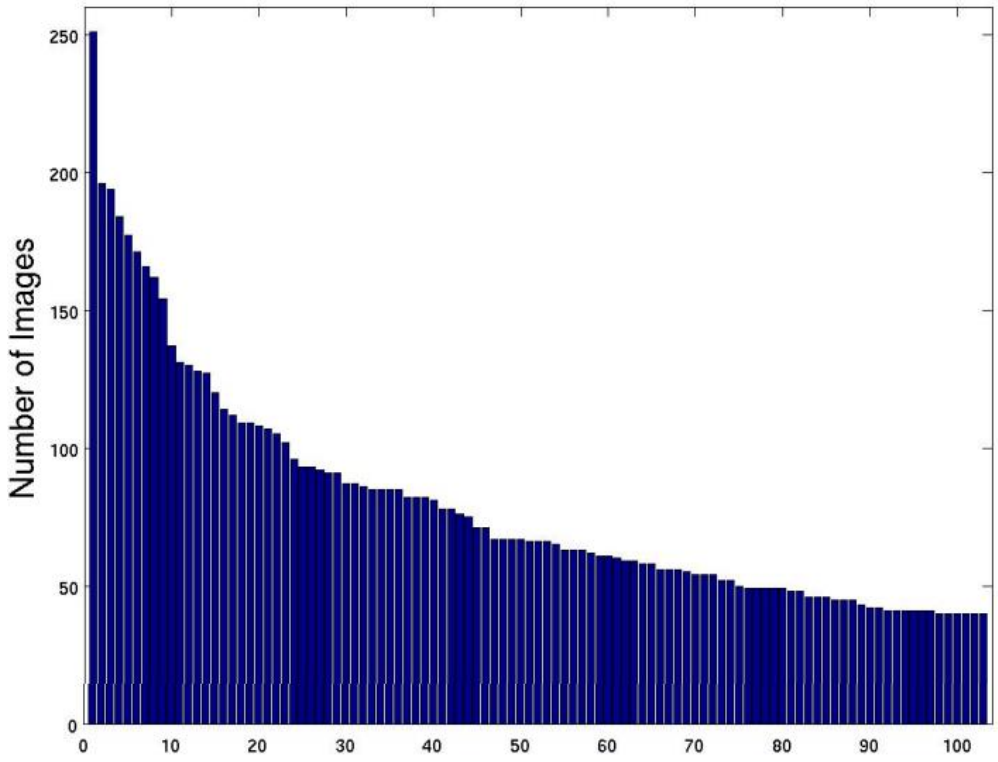
Bitki türü tanıma projelerinde kullanılan çeşitli tekniklerin ve metodolojilerin önemli bir çeşitliliğe sahip olduğunu göstermektedir. Derin öğrenme, transfer öğrenme ve özellik mühendisliği gibi yöntemlerin bir araya geldiği bu çalışmalar, bitki türü tanıma alanında önemli başarılar elde etmiştir.

Bölüm 3

Veri İncelemesi

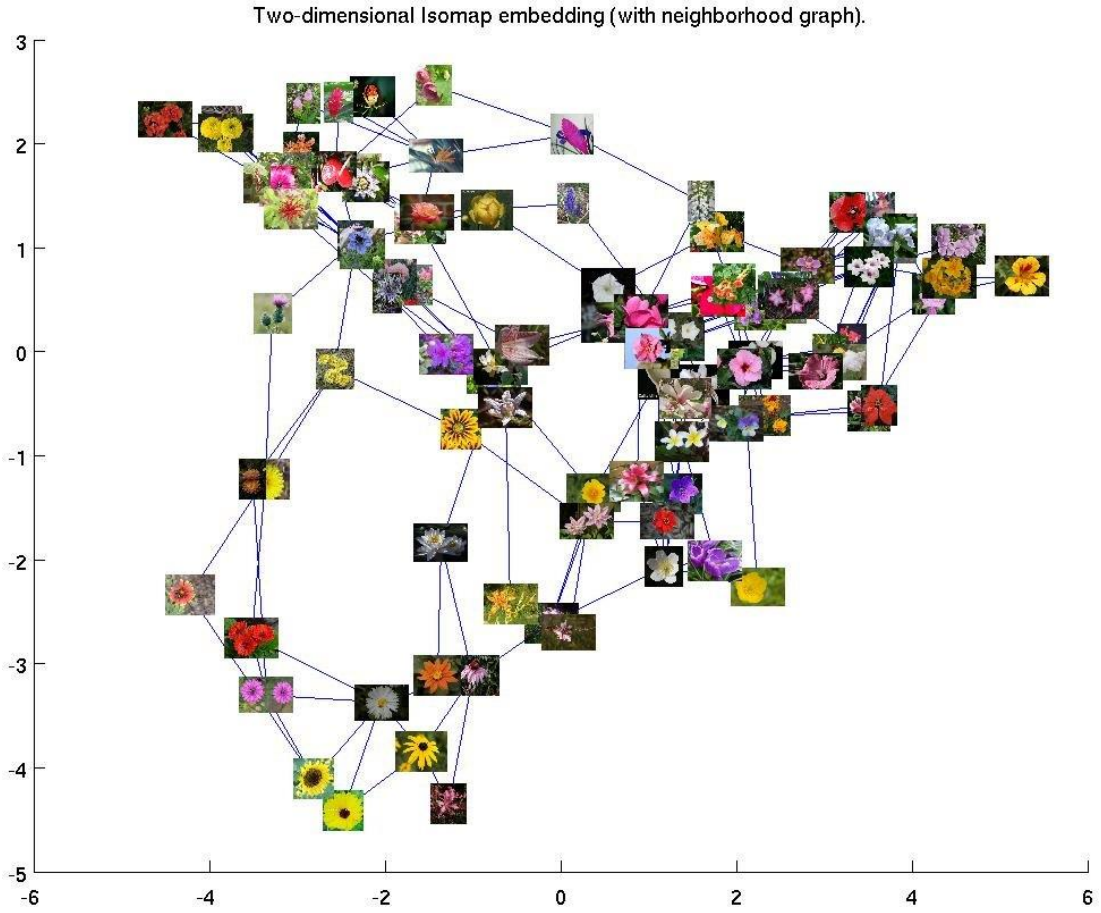
3.1 Kullanılan Veriler

Bu makalede, 102 çiçek sınıfına ayrılmış toplam 8189 görüntü içeren bir veri setini tanıtıyoruz. Bu çiçekler, genellikle Birleşik Krallık'ta bulunan çiçekler arasından seçilmiştir. Görüntülerin çoğu web üzerinden toplandı; ancak küçük bir sayıda görüntüyü kendimiz çekerek elde ettik. Her bir sınıf, 40 ila 250 arasında değişen sayıda görüntü içermektedir. Şekil 2, tüm sınıflar üzerindeki görüntü sayısının dağılımını göstermektedir. Passion flower en fazla görüntüye sahipken, eustoma, mexican aster, celosia, moon orchid, canterbury bells ve primrose en az görüntüye sahip sınıflardır, yani her birinde 40 görüntü bulunmaktadır. Görüntüler, en küçük boyutun 500 piksel olduğu şekilde ölçeklendirilmiştir.



Şekil 2. 102 Sınıf Üzerindeki Görüntü Sayısının Dağılımı

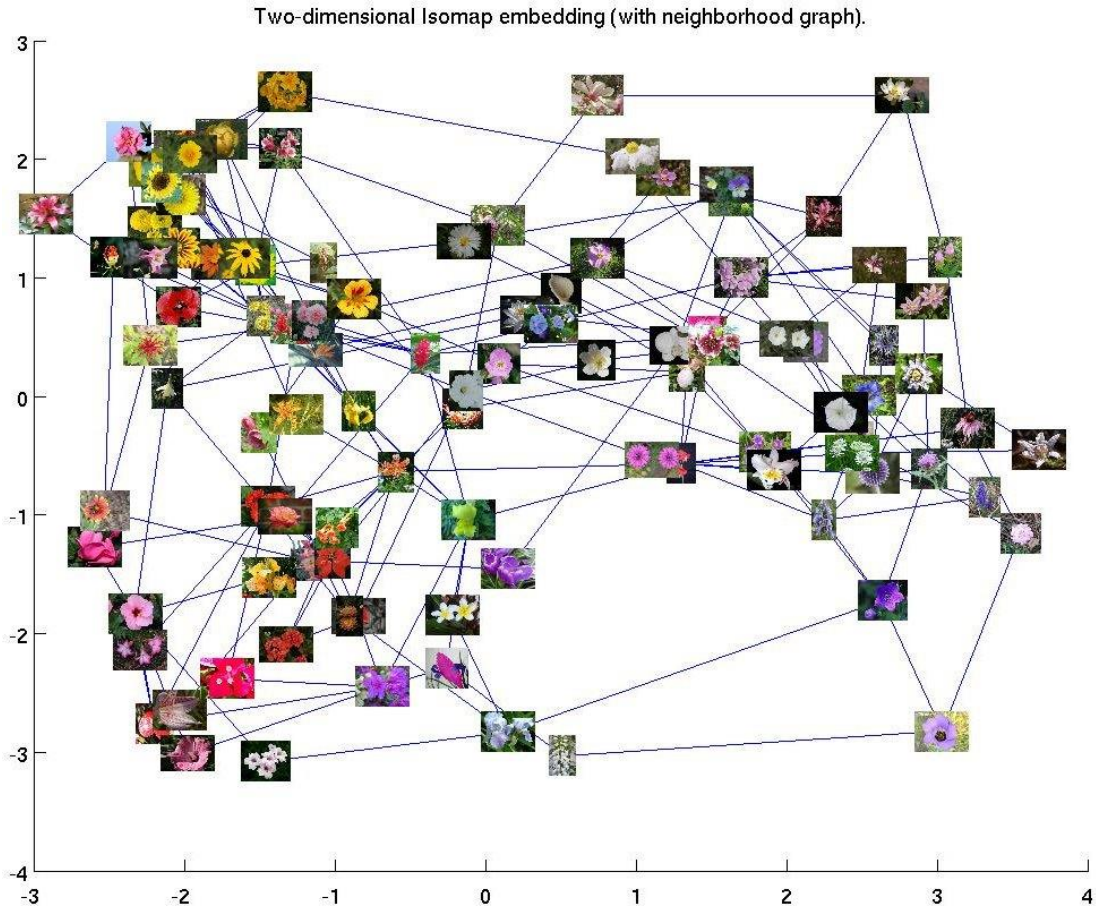
Şekil 3’de görüldüğü gibi, Bitki türlerinde Shape Isomap gibi yöntemler, bitki formları veya organları arasındaki morfolojik benzerlikleri anlamak ve ölçmek için kullanılabilir. Bitki türleri genellikle farklı formlarda ve yapısal özelliklerde gelir. Örneğin, yaprak şekilleri, çiçeklerin düzeni ve gövdelerin morfolojisi bitki türlerini ayırt etmek için kullanılan özelliklerdir. Shape Isomap, bu tür morfolojik özellikleri koruyarak ve vurgulayarak, bitki türlerini birbirinden ayırmak ve benzerliklerini anlamak için etkili bir araç olabilir. Bu sayede bitki bilimcileri ve biyologlar, bitki türlerinin evrimsel ilişkilerini ve çeşitli ekolojik faktörlere olan adaptasyonlarını daha iyi anlayabilirler.



Şekil 3. Şekil Haritalama (Shape Isomap)

Şekil 4’te görüldüğü gibi, renk bilgisini içeren bitki türleri görüntülerinde "Color Isomap" olarak adlandırılan spesifik bir yöntem veya teknik bulunmamaktadır. Ancak, bitki türü tanıma veya sınıflandırma projelerinde renk bilgisinin kullanılması oldukça yaygındır. Bu tür projelerde, renk özellikleri genellikle bitki türlerini ayırt etmek için

önemli bir görsel özellik olabilir. Renk bilgisini kullanarak yapılan analizler, bitki türleri arasındaki farklılıkları vurgulamak ve sınıflandırma doğruluğunu artırmak için kullanılabilir.



Şekil 4. renk haritalaması (Color Isomap)

3.1.1 Görüntü İşleme

Birçok makale [6, 14, 12, 13], çiçek görüntüsünün otomatik olarak çiçek olarak öne çıkacak şekilde ve geri kalanını arka plan olarak segmente etmek için özel yöntemler önermiştir. Bu çalışmada, Nilsback ve Zisserman [13] tarafından önerilen bölütleme şeması kullanılmıştır. İlk olarak, genel (sınıf spesifik olmayan) ön plan ve arka plan renk dağılımlarını kullanarak başlangıç çiçek segmentasyonu elde edilir. Bu dağılımlar, her sınıfın veri setindeki birkaç eğitim görüntüsündeki pikselleri ön plan (yani çiçeğin bir parçası) veya arka plan (yani bitkinin bir parçası) olarak etiketleyerek öğrenilir ve sonra tüm sınıflar üzerinde bu dağılımların ortalaması alınır. Bu genel ön plan ve arka plan dağılımları kullanılarak, başlangıç ikili bölütleme, kontrast bağımlı önceki MRF maliyet fonksiyonu kullanılarak ve graf kesimleriyle optimize edilerek elde edilir. Bu bölütlenme mükemmel olmayabilir, ancak genellikle çiçeğin dış sınırlarının en az bir kısmını çıkarmak için yeterlidir. Ardından, genel bir çiçek şekil modeli bu başlangıç segmentasyona uyarlanır ve taç yaprakları algılamak için kullanılır. Model, gevşek bir geometrik tutarlılığa sahip olan taç yapraklarını bir affine invariant Hough benzeri prosedür kullanarak seçer. Geometrik olarak tutarlı olduğu düşünülen taç yaprakları için görüntü bölgeleri, yeni bir görüntü spesifik ön plan renk modelini elde etmek için kullanılır. Ön plan renk modeli, görüntü spesifik ön plan modelini genel ön plan modeli ile karıştırarak güncellenir. MRF bölütleme, bu yeni renk modeli kullanılarak tekrarlanır. Başlangıç bölütlenmenin mükemmel olmadığı durumlarda, görüntü spesifik ön planın kullanımı çoğu zaman çiçeğin daha fazlasını toplar. Şekil modeli uydurma ve görüntü spesifik ön plan öğrenme adımları, iki ardışık iterasyon arasında hiç veya çok az değişiklik olana kadar yinelenebilir.

3.1.2 Sınıflandırma

Bu projede, sınıflar arasında ayırt edici bir sınıflandırıcı elde etmek, aynı sınıfa ait tüm örnekleri doğru bir şekilde sınıflandırabilmektir. Sınıflandırıcı, ayçiçeğini papatyadan ayırt etmenin renk kullanışlı bir ipucu olduğunu ancak şeklinin oldukça zayıf olduğunu ve tersine, semizotunu karahindibadan ayırt etmenin şeklinin çok daha kullanışlı olduğunu ancak renk kullanışlı olmadığını temsil edip öğrenebilmelidir. Bu bölümde önce, ön plan çiçek bölgelerini temsil etmek için tasarlanmış dört özelliği açıklıyoruz ve ardından her biri bir özelliğe karşılık gelen tek karşıt SVM sınıflandırması için kullanılan çekirdeklerin lineer kombinasyonunu açıklıyoruz.

Bölüm 4

Model İncelemesi

4.1 Kullanılan Model

VGG-16, Visual Geometry Group (VGG) tarafından geliştirilen ve 2014'te ImageNet Large Scale Visual Recognition Challenge yarışmasında büyük bir başarı elde eden bir derin öğrenme modelidir. Model, Oxford Üniversitesi'ndeki araştırmacılar tarafından oluşturulan bir dizi derin evrişimli sinir ağı (CNN) modellerinden biridir. VGG-16, görsel tanıma görevlerinde kullanılmak üzere tasarlanmış ve özellikle geniş bir veri kümesi olan ImageNet üzerindeki 1.4 milyon etiketli görüntüyü sınıflandırmak için eğitilmiştir.

VGG-16 modelinin temel özellikleri:

- Derinlik, "16" ifadesi, modelin toplam katman sayısını belirtir. Bu katmanlar arasında 13 evrişimli katman ve 3 tam bağlantılı katman bulunmaktadır.
- Evrişimli Katmanlar, VGG-16'nın evrişimli katmanları, küçük boyutlu filtreler (3x3 çekirdek boyutu) kullanır ve ardışık olarak sıralanmıştır. Bu yapı, daha derin ve karmaşık özelliklerin öğrenilmesine olanak tanır.
- Maksimum Havuzlama Katmanları, evrişimli katmanların ardından, maksimum havuzlama katmanları kullanılarak elde edilen özellik haritaları boyutsal olarak küçültülür. Bu, öğrenilen özelliklerin ölçeklenebilir ve translasyon invariant olmasına yardımcı olur.
- Tam Bağlantılı Katmanlar, evrişimli katmanlardan sonra gelen tam bağlantılı katmanlar, özellikleri bir araya getirerek sınıflandırma yapar. VGG-16, bu amaçla iki adet 4096 nöronlu tam bağlantılı katman içerir.
- Aktivasyon Fonksiyonları, modelin evrişimli katmanlarında genellikle ReLU (Rectified Linear Unit) aktivasyon fonksiyonları kullanılmıştır.

VGG-16, özellikle transfer öğrenme bağlamında popülerdir. Önceden eğitilmiş ağırlıkları kullanarak çeşitli görsel tanıma görevlerinde iyi performans gösterebilir. VGG-16, derin evrişimli sinir ağı mimarilerinin anlaşılması ve yaygınlaşması açısından önemli bir kilometre taşıdır.

4.2 Deneysel Prosedür

102 çiçek sınıfına ayrılmış toplam 8189 görüntü içeren bir veri setinde her bir sınıf, 40 ila 250 arasında değişen sayıda görüntü içermektedir. Veri seti, bir eğitim seti, bir doğrulama seti ve bir test setine ayrılmıştır. Eğitim seti ve doğrulama seti, her biri sınıf başına 10 görüntü içerir (toplamda 1030 görüntü). Test seti, kalan 6129 görüntüyü içerir (her sınıfta minimum 20 görüntü). Doğrulama seti, her bir özellik için görsel kelimelerin, SIFT özelliklerinin yarıçapının ve aralığının optimize edilmesi için kullanılır.

Hem doğrulama hem de test setleri için performans, her bir sınıf için ölçülür. Yani nihai performans, tüm sınıflar üzerinden ortalama yapılan sınıflandırmadır.

4.3 Doğrulama Setinde Optimizasyon

Özellik türüne bağlı olarak iç SIFT özellikleri için ızgara aralığı veya k-means kümeleme için k gibi parametrelerin en uygun değerleri, standart yöntemle doğrulama setinde performansı optimize edilerek öğrenilir.

Örneğin, iç SIFT özellikleri için sınıflandırıcı, yalnızca bu özellik için çekirdek kullanılarak eğitilir. Kelime dağarcığındaki en uygun sayı, 1000 ile 10000 arasında bir aralıkta aranır ve doğrulama setinde maksimum sınıflandırma performansı bulunarak belirlenir. Her k-means kümeleme işlemi 3 kez tekrarlanır ve en iyi sonuçlar saklanır. Renk özellikleri için arama, renk özelliklerinin SIFT'ten çok daha az boyuta sahip olması nedeniyle 100 ile 5000 arasında yapılır, bu nedenle bunları açıklamak için daha az kelime kullanmayı bekleriz. Hem ızgara aralığı M hem de dairesel destek yamaları R, 5 ile 50 piksel arasında bir aralıkta aranır. Performansın doğrulama setinde eniyilenmesi, her değişken için ayrı ayrı gerçekleştirilir.

Renk özellikleri için optimum kelime sayısı 1000, tüm ön plan bölgesi için SIFT için 8000, sadece sınır bölgesindeki SIFT için 3000 ve HOG özellikleri için 1500'dir. Optimum boyut yarıçapı, içsel ve sınır SIFT için sırasıyla 15 ve 10'dur. Optimum aralık, içsel SIFT için 10 ve sınır SIFT için 10'dur.

Parametreler doğrulama setinde belirlendikten sonra, sınıflandırıcılar tüm mevcut eğitim verileri kullanılarak (hem eğitim hem de doğrulama setleri), Varma ve Ray'in optimizasyon yöntemi kullanılarak ağırlıklar belirlenir.

Bölüm 5

Kodun İncelemesi

5.1 Kütüphanelerin Yüklenmesi

Bu projede, çeşitli çiçek türlerini tanımak amacıyla derin öğrenme yöntemlerini kullanıyoruz. İlk adım olarak, gerekli olan kütüphaneleri yüklüyoruz. Bu kütüphaneler arasında görüntü işleme, veri manipülasyonu ve derin öğrenme için gereken PyTorch modülleri bulunmaktadır.

Import Libraries

```
import matplotlib
from torch._C import NoneType

# General utilities
import os
import glob
import time
import subprocess
import scipy.io
import pandas as pd
import numpy as np
from tqdm.notebook import tqdm
from sklearn import preprocessing

# Torch and Torchvision
import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader, Dataset
from torchvision.utils import make_grid
from torchvision.datasets import ImageFolder
import torchvision.models as models
from torchsummary import summary
import torch.nn as nn

# Visualisation
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import ImageGrid
from PIL import Image
print("CUDA available: ", torch.cuda.is_available())
device = 'cuda' if torch.cuda.is_available() else 'cpu'
```

```
CUDA available:  True
```

Şekil 5. Kütüphanenin Yüklenmesi

5.2 Veri Setinin Hazırlanması

Download Dataset, Çiçek veri setini indirmek için gerekli dosyaları ve etiketleri çekiyoruz. Veriyi içeren dosyaları belirli bir klasöre çıkartıyoruz.

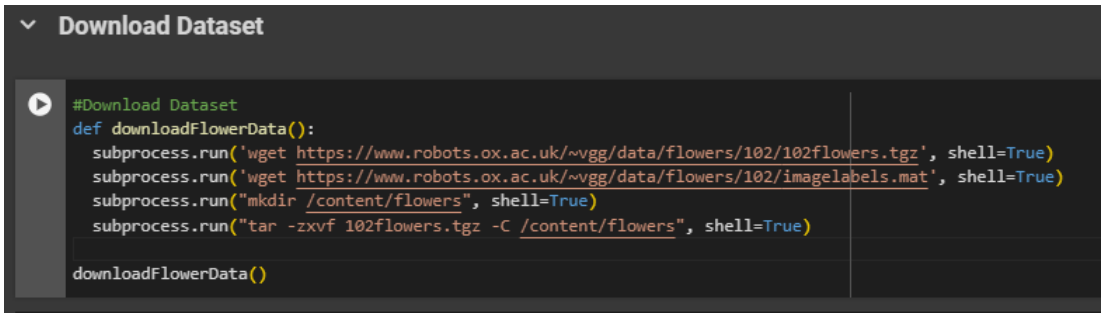
Etiket Eşleme İçin Sözlük Oluşturma, Etiketleri çiçek türleri isimleriyle eşleştirmek için bir sözlük oluşturuyoruz. Örneğin, '21' etiketi 'fire lily' olarak eşleştirilir.

Data Frame Oluşturma, Etiketler ve ilgili görüntü dosyalarının bulunduğu bir Data Frame oluşturuyoruz. Bu, veriyi düzenli bir şekilde işlememize olanak tanır.

Veri Seti ve Veri Yükleyici Oluşturma, Oluşturduğumuz Data Frame'i kullanarak eğitim ve doğrulama veri setlerini oluşturuyoruz. Görüntülerin ve etiketlerin yüklenmesini sağlayan özel bir veri yükleyici (Data Loader) oluşturuyoruz.

5.2.1 Veri Setinin Yüklenmesi (Download Dataset)

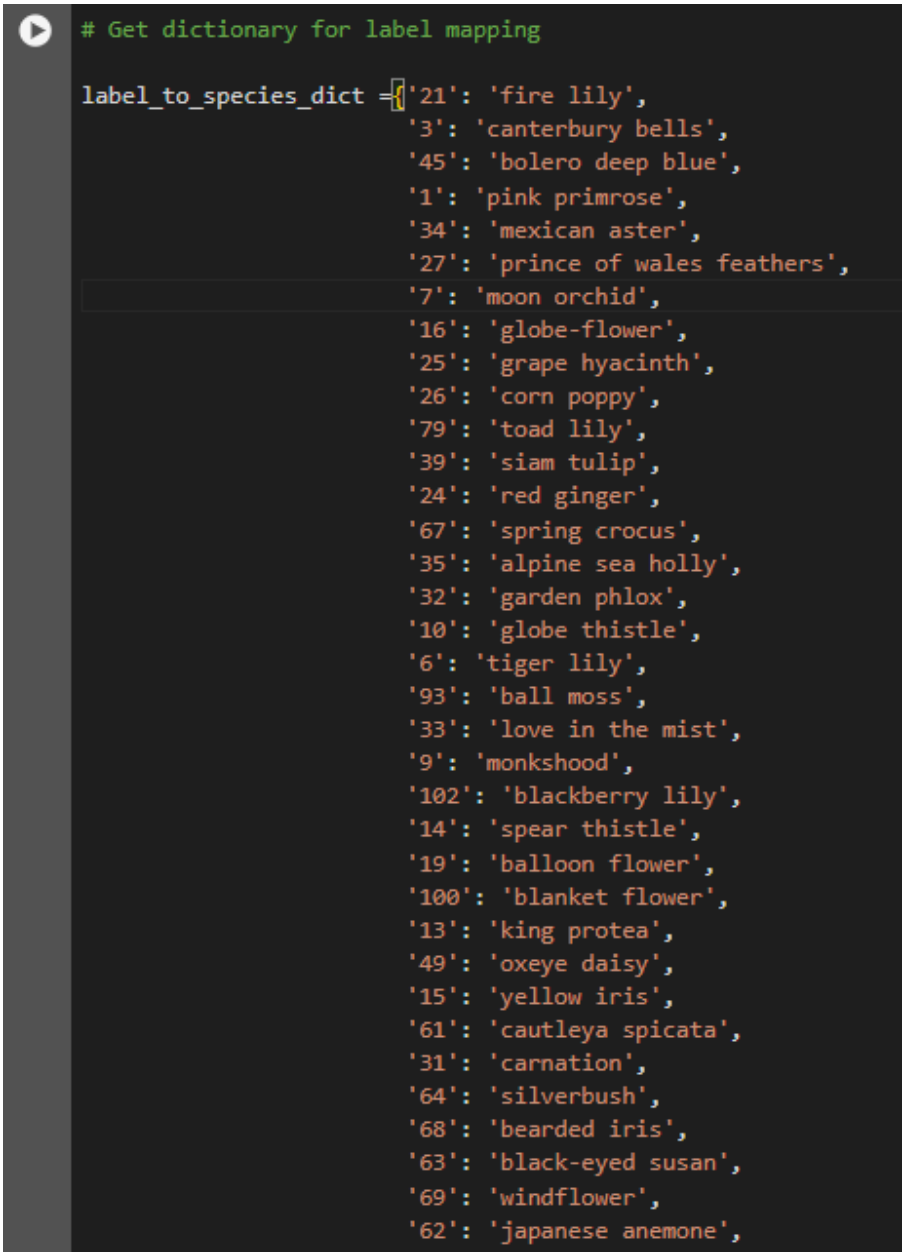
Projemizin temelini oluşturan çiçek veri setini indiriyoruz. Bu veri seti, İngiltere'de yaygın olarak bulunan 102 farklı çiçek türünü içermektedir.



Şekil 6. Veri Setinin Yüklenmesi

5.2.2 Etiket Eşleme İçin Sözlük Oluşturma

Çiçek türlerini sayısal etiketlere karşılık gelen isimlere eşleyen bir sözlük oluşturur. Her bir etiket, belirli bir çiçek türünü temsil eder. Örneğin, '21' etiketi 'fire lily' (ateş zambak) çiçeğini temsil eder. Bu sözlük, veri setindeki her bir etiketin anlamını daha anlaşılır ve yorumlanabilir hale getirir. Etiket ve çiçek isimleri, bir Python sözlüğünde eşleştirilmiştir.

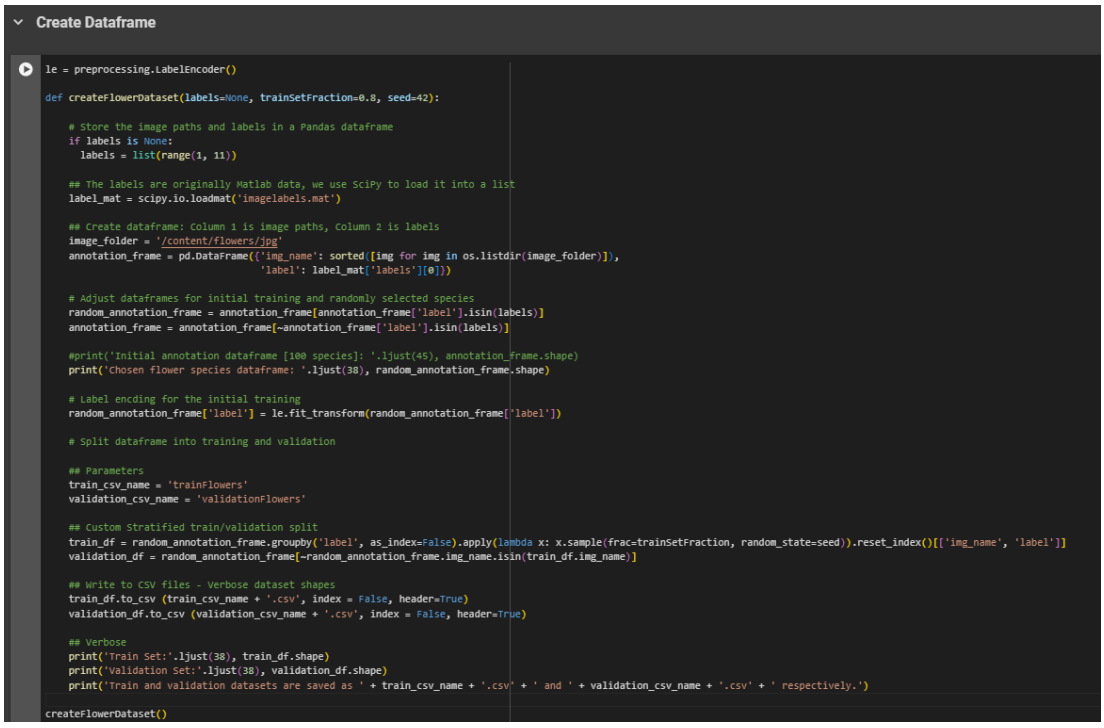


```
# Get dictionary for label mapping

label_to_species_dict ={'21': 'fire lily',
                        '3': 'canterbury bells',
                        '45': 'bolero deep blue',
                        '1': 'pink primrose',
                        '34': 'mexican aster',
                        '27': 'prince of wales feathers',
                        '7': 'moon orchid',
                        '16': 'globe-flower',
                        '25': 'grape hyacinth',
                        '26': 'corn poppy',
                        '79': 'toad lily',
                        '39': 'siam tulip',
                        '24': 'red ginger',
                        '67': 'spring crocus',
                        '35': 'alpine sea holly',
                        '32': 'garden phlox',
                        '10': 'globe thistle',
                        '6': 'tiger lily',
                        '93': 'ball moss',
                        '33': 'love in the mist',
                        '9': 'monkshood',
                        '102': 'blackberry lily',
                        '14': 'spear thistle',
                        '19': 'balloon flower',
                        '100': 'blanket flower',
                        '13': 'king protea',
                        '49': 'oxeye daisy',
                        '15': 'yellow iris',
                        '61': 'cautleya spicata',
                        '31': 'carnation',
                        '64': 'silverbush',
                        '68': 'bearded iris',
                        '63': 'black-eyed susan',
                        '69': 'windflower',
                        '62': 'japanese anemone',
```

Şekil 7. Etiket Eşleme Sözlüğü Oluşturma

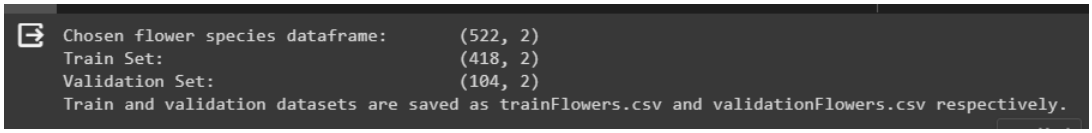
5.2.3 Veri Çerçevesi Oluşturma

Bu bölümde, çiçek veri setinden eğitim ve doğrulama setlerini oluşturan bir işlevi tanımlıyoruz. İlk olarak, eğer belirli bir etiket listesi belirtilmemişse, varsayılan olarak etiket listesini 1 ile 10 arasındaki sayılar olarak belirliyoruz. Daha sonra, çiçek veri setinin etiket bilgilerini içeren "imagelabels.mat" dosyasını SciPy kullanarak yüklüyoruz. Sonrasında, çiçek görüntülerinin dosya yolları ve etiketleri içeren bir Pandas veri çerçevesi oluşturuyoruz. Bu veri çerçevesini, ilk eğitim için seçilen etiketlere göre ayarlıyoruz ve diğer türleri içeren başka bir veri çerçevesine ayırıyoruz. Seçilen etiketlerin ilk eğitim için sayısal olarak kodlanmasını LabelEncoder kullanarak gerçekleştiriyoruz. Daha sonra, veri çerçevemizi eğitim ve doğrulama setlerine bölmek için özel bir Stratified yöntemini kullanıyoruz. Bu işlemi gerçekleştirirken belirtilen kesir oranına ve rastgele bir tohum değerine göre bölme işlemini gerçekleştiriyoruz. Elde edilen eğitim ve doğrulama setlerini CSV dosyalarına yazarak bu işlemi tamamlıyoruz.



Şekil 8. Veri Seti Hazırlama

"Chosen flower species dataframe" 522 çiçek görüntüsünü içerir ve seçilen etiketlere aittir. "Train Set" 418 çiçek görüntüsünden oluşur, özel bir Stratified bölme ile seçilen etiketleri içerir. "Validation Set" 104 çiçek görüntüsünden oluşur ve eğitim setinden ayrılmıştır. Her iki set de dosya adları ve etiket bilgilerini içeren CSV dosyalarına kaydedilmiştir: "trainFlowers.csv" ve "validationFlowers.csv".



Şekil 9. Eğitim ve Doğrulama Veri Setleri

5.2.4 Veri Kümeleri ve Veri Yükleyicileri Oluşturma

Veri kümesini oluşturmak için "FlowerDataset" adlı özel bir sınıf tanımlanmıştır. Bu sınıf, çiçek görüntülerinin dizinini, etiketlerin bulunduğu bir CSV dosyasını, dönüşümleri ve hedef dönüşümleri içerir. __len__ ve __getitem__ metotları, veri kümesinin uzunluğunu ve belirli bir öğeyi döndürme işlemlerini yönetir. getDataLoader fonksiyonu, veri kümesini ve belirtilen dönüşümleri kullanarak bir veri yükleyici oluşturur. Bu yükleyici, belirtilen parti boyutu, işçi sayısı ve bellek pimlemesi gibi parametrelere sahiptir. Son olarak, eğitim ve doğrulama veri yükleyicileri oluşturulmuştur. Eğitim veri yükleyicisi, çeşitli dönüşümleri içerirken, doğrulama veri yükleyicisi sadece boyutlandırma ve normalizasyon içerir. Bu yapı, veri kümesi oluşturma ve veri yükleyici tanımlama işlemlerini basitleştirir ve eğitim/validation süreçlerinde kullanılmaya hazır veri setleri sağlar.

```python
from torchvision import transforms

# Custom Flower Dataset Class
class FlowerDataset(Dataset):
    def __init__(self, image_directory, annotation_file, transform=None, target_transform=None):
        self.annotations = pd.read_csv(annotation_file)
        self.image_directory = image_directory
        self.transform = transform
        self.target_transform = target_transform

    def __len__(self):
        return len(self.annotations)

    def __getitem__(self, index):
        img_path = self.annotations.iloc[index, 0]
        image = Image.open(os.path.join(self.image_directory, img_path)).convert("RGB")
        label = self.annotations.iloc[index, 1]
        if self.transform is not None:
            image = self.transform(image)
        if self.target_transform is not None:
            label = self.target_transform(label)
        return image, label


def getDataLoader(imageFolder, annotationFile, transformation, batchSize):
  dataset = FlowerDataset(imageFolder, annotationFile, transform=transformation)
  dataLoader = DataLoader(dataset=dataset, shuffle=True, batch_size=batchSize, num_workers=2, pin_memory=True)
  return dataLoader

trainDataLoader = getDataLoader(imageFolder='/content/flowers/jpg',
                                annotationFile='trainFlowers.csv',
                                transformation=transforms.Compose([transforms.Resize((224, 224)),
                                                                   transforms.RandomHorizontalFlip(),
                                                                   transforms.RandomVerticalFlip(),
                                                                   transforms.ToTensor(),
                                                                   transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])]),
                                batchSize=8)

validDataLoader = getDataLoader(imageFolder='/content/flowers/jpg',
                                annotationFile='validationFlowers.csv',
                                transformation=transforms.Compose([transforms.Resize((224, 224)),
                                                                   transforms.ToTensor(),
                                                                   transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])]),
                                batchSize=8)
```

Şekil 10. Veri Kümeleri ve Veri Yükleyicileri Oluşturma

5.3 Modelin Çalışması

5.3.1 Derin Öğrenme Modeli VGG-16

Bitki türü tanıma projesi için VGG-16 mimarisini kullanarak bir derin öğrenme modeli oluşturmayı amaçlamaktadır. Proje, Oxford Üniversitesi tarafından sağlanan 102 farklı bitki türünün yer aldığı bir veri seti üzerine odaklanmaktadır. İlk olarak, VGG-16 mimarisi seçilmiştir. Eğer kullanıcı tarafından belirtilirse (`pretrained=True`), önceden eğitilmiş VGG-16 ağırlıkları kullanılır ve sınıflandırma katmanı projedeki sınıf sayısına göre güncellenir. Eğer önceden eğitilmiş ağırlıklar kullanılmayacaksa (`pretrained=False`), VGG-16 mimarisi kullanılarak yeni bir model oluşturulur ve sınıflandırma katmanları elle tanımlanır. İlk olarak, PyTorch kütüphanesi kullanılarak gerekli modüller ve cihaz (GPU veya CPU) seçimi import edilir. Daha sonra, `createModel` fonksiyonu tanımlanır. Bu fonksiyon, `pretrained` ve `nClasses` parametrelerini alarak VGG-16 modelini oluşturur ve projeye özgü olarak uyarlar. Eğer önceden eğitilmiş bir model kullanılıyorsa, VGG-16 modeli `models.vgg16(pretrained=True)` ile çağrılır ve sınıflandırma katmanı projedeki sınıf sayısına uyacak şekilde güncellenir. Eğer önceden eğitilmiş model kullanılmayacaksa, `models.vgg16(pretrained=False)` ile yeni bir model oluşturulur ve sınıflandırma katmanları elle tanımlanır.

Modelin bellek yönetimi için GPU'ya taşınması sağlanır (`torch.cuda.empty_cache()`). Oluşturulan modelin özeti ekrana yazdırılır ve son olarak oluşturulan model fonksiyon tarafından döndürülür. Projede kullanılmak üzere örnek bir model, `pretrained=True` ve `nClasses=10` parametreleri ile `createModel` fonksiyonuna gönderilir. Bu, önceden eğitilmiş VGG-16 modelini kullanarak, projedeki 10 farklı bitki türünü tanıyabilen bir model oluşturur.

```python
def createModel(pretrained, nClasses):

  # Define model
  if pretrained:
    model = models.vgg16(weights='VGG16_Weights.DEFAULT')
    model.classifier[6] = nn.Linear(4096, nClasses)
  else:
    model = models.vgg16(weights=None)
    model.classifier = nn.Sequential(nn.Linear(25088, 4096, bias = True),
                                     nn.ReLU(inplace = True),
                                     nn.Dropout(0.4),
                                     nn.Linear(4096, 2048, bias = True),
                                     nn.ReLU(inplace = True),
                                     nn.Dropout(0.4),
                                     nn.Linear(2048, nClasses))

  # Get device and load model to the device
  torch.cuda.empty_cache()
  model.to(device)
  print('VGG16 model created successfully, summary; \n', model)
  return model

modelVGG = createModel(pretrained=True, nClasses=10)
```

Şekil 11. Model Kodu

5.3.2 Eğitim Modeli

Bitki türü tanıma modelinin eğitimini gerçekleştiren bir eğitim döngüsünü içermektedir. Eğitim döngüsü, önceden tanımlanmış bir VGG-16 modelini, eğitim veri kümesi (trainDataLoader) ve doğrulama veri kümesi (validationDataLoader) üzerinde belirli bir öğrenme oranı (learningRate) ve belirli bir epoch sayısı (epochs) için eğitir. Eğitim döngüsü, Adam optimizer'ı ve CrossEntropyLoss kaybını kullanarak modelin parametrelerini günceller. Ayrıca, her epoch için eğitim ve doğrulama verileri üzerinde kayıp (loss) ve doğruluk (accuracy) değerlerini hesaplar. Eğitim süreci ilerledikçe bu değerleri train_loss, train_accuracy, valid_loss, ve valid_accuracy listelerine ekler.

Eğitim döngüsü şu adımları içerir:

- Her bir epoch için eğitim veri kümesi üzerinde ilerlenir.
- Modelin çıkışı alınır ve belirtilen kayıp fonksiyonu (nn.CrossEntropyLoss) kullanılarak kayıp hesaplanır.
- Optimizasyon işlemi gerçekleştirilir: Gradyanları sıfırlanır, kayıp fonksiyonunun gradyanlarına göre türev hesaplanır ve parametreler güncellenir.
- Eğitim doğruluğu ve kaybı toplam epoch boyunca güncellenir.
- Epoch sonunda, doğrulama veri kümesi üzerinde benzer işlemler gerçekleştirilir ve doğrulama doğruluğu ve kaybı toplam epoch boyunca güncellenir.
- Her epoch sonunda eğitim ve doğrulama verileri için kayıp ve doğruluk değerleri ekrana yazdırılır.

Eğitim döngüsü, belirtilen epoch sayısına ulaşıldığında sona erer. Sonuç olarak, history değişkeni içinde eğitim sürecinde elde edilen kayıp ve doğruluk değerleri bulunur. Bu değerlerin incelenmesi, modelin eğitim performansının değerlendirilmesine ve gerektiğinde iyileştirmelere yönelik bilgi sağlar.

Bu eğitim döngüsü, bitki türü tanıma modelinin eğitimini gerçekleştirmek için kullanılabilir ve projede elde edilen sonuçları değerlendirmek için önemli bir bileşenidir.

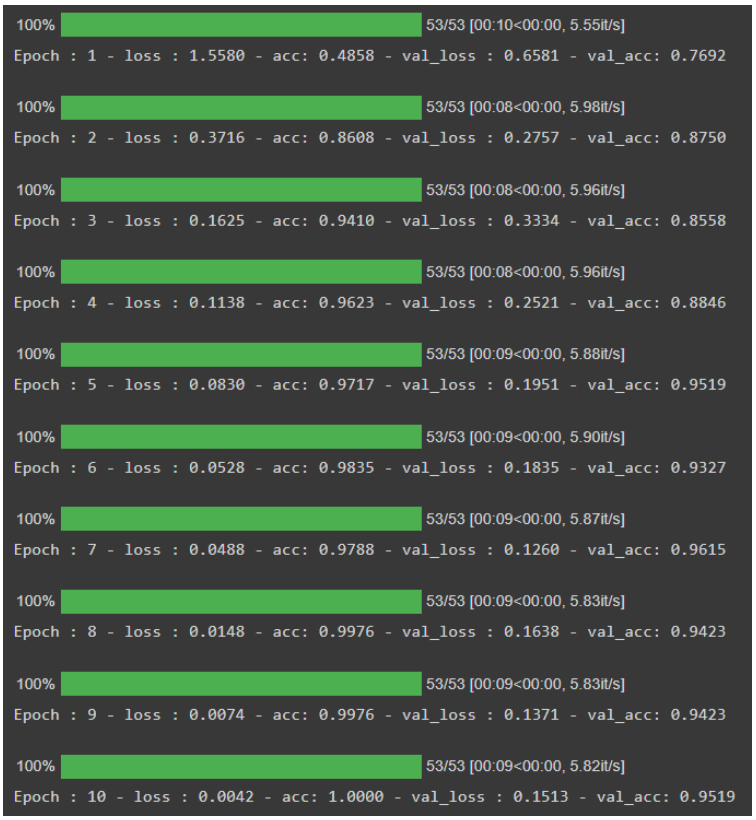
```python
# Training loop

def trainModel(model, trainDataLoader, validationDataLoader, learningRate, epochs):
  # Training parameters
  optimizer = torch.optim.Adam(model.parameters(), lr = learningRate)
  criterion = nn.CrossEntropyLoss()

  ## Define list objects for storing training information
  train_loss = []
  train_accuracy = []
  valid_loss = []
  valid_accuracy = []

  # Training loop
  for epoch in range(epochs):
      epoch_loss = 0
      epoch_accuracy = 0
      for data, label in tqdm(trainDataLoader):
          data = data.to(device)
          label = label.to(device)
          output = model(data)
          loss = criterion(output, label)
          optimizer.zero_grad()
          loss.backward()
          optimizer.step()
          acc = (output.argmax(dim=1) == label).float().mean()
          epoch_accuracy += acc / len(trainDataLoader)
          epoch_loss += loss / len(trainDataLoader)
      train_accuracy.append(epoch_accuracy.item())
      train_loss.append(epoch_loss.item())
      with torch.no_grad():
          epoch_val_accuracy = 0
          epoch_val_loss = 0
          for data, label in validationDataLoader:
              data = data.to(device)
              label = label.to(device)
              val_output = model(data)
              val_loss = criterion(val_output, label)
              acc = (val_output.argmax(dim=1) == label).float().mean()
              epoch_val_accuracy += acc / len(validationDataLoader)
              epoch_val_loss += val_loss / len(validationDataLoader)
          valid_accuracy.append(epoch_val_accuracy.item())
          valid_loss.append(epoch_val_loss.item())
      print(f"Epoch : {epoch+1} - loss : {epoch_loss:.4f} - acc: {epoch_accuracy:.4f} - val_loss : {epoch_val_loss:.4f} - val_acc: {epoch_val_accuracy:.4f}\n")
  return train_loss, train_accuracy, valid_loss, valid_accuracy


history = trainModel(model=modelVGG,
                     trainDataLoader=trainDataLoader,
                     validationDataLoader=validDataLoader,
                     learningRate=2e-5,
                     epochs=20)
```

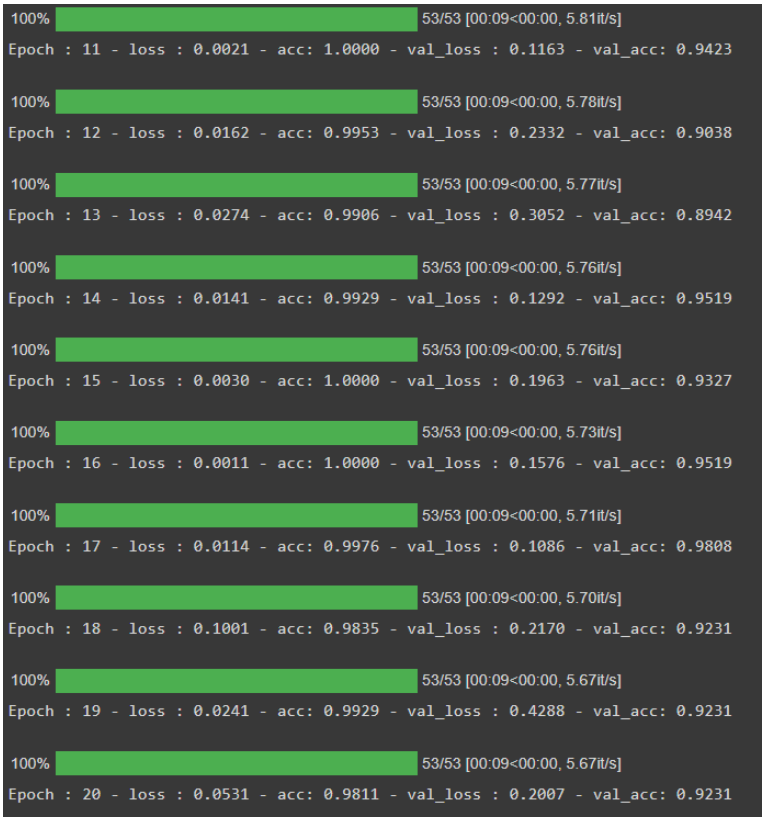
Şekil 12. Eğitim Kodu

Bitki türü tanıma modelinin eğitim sürecinde elde edilen veriler eğitim sürecinin her bir epoch'u boyunca modelin performansındaki değişiklikleri göstermektedir. Eğitim modeli çıktılardan çıkarılabilecek bazı önemli bilgiler şunlar:

- Epoch 1:
  - ✓ Eğitim Kaybı: 1.5580, Eğitim Doğruluğu: %48.58
  - ✓ Doğrulama Kaybı: 0.6581, Doğrulama Doğruluğu: %76.92
  - ✓ İlk epoch sonunda, modelin eğitim ve doğrulama performansı arasında belirgin bir fark vardır. Bu, modelin başlangıçta eğitim verilerine aşırı uyum sağlamış olabileceğini gösterir.
- Epoch 2-5:
  - ✓ Eğitim kaybı sürekli azalırken, eğitim doğruluğu artar.
  - ✓ Doğrulama kaybı ve doğrulama doğruluğu da iyileşir.
  - ✓ Model, genel olarak daha iyi bir performans göstermeye başlar.
- Epoch 6-11:
  - ✓ Eğitim ve doğrulama performansı oldukça yüksek ve benzerdir.
  - ✓ Modelin genelleme yeteneği artar, aşırı uyum (overfitting) belirtileri azalır.
- Epoch 12-20:
  - ✓ Eğitim kaybı artar ve eğitim doğruluğu düşer, ancak bu durumun doğrulama performansını nasıl etkilediği önemlidir.
  - ✓ Modelin genelleme yeteneği nispeten sabit kalır, bu da modelin başka veri setlerinde de iyi performans gösterebileceğini gösterir.
- Genel İzlenim:
  - ✓ Model, eğitim sürecinin başlarında öğrenmeye başlar ve genellikle daha iyi bir performans gösterir.
  - ✓ Doğrulama doğruluğu genellikle eğitim doğruluğundan biraz düşüktür, ancak bu fark zamanla azalır.
  - ✓ Kayıp fonksiyonları genellikle eğitim sürecinin başlarında düşer ve daha sonra artabilir. Bu durum, modelin öğrenme oranının azaldığı veya aşırı uyumun arttığı anlamına gelebilir.

Modelin başarıyla eğitildiği, doğrulama setinde yüksek bir doğruluk elde ettiği ve aşırı uyumun kontrol altında olduğu söylenebilir.





Şekil 13. Eğitim Modeli Çıktıları

5.3.3 Doğrulama Modeli

Bitki türü tanıma modelinin eğitim sürecinde elde edilen kayıp ve doğruluk değerlerini görselleştiren bir grafik oluşturan bir fonksiyon içermektedir. Oluşturulan grafik, eğitim ve doğrulama süreçlerindeki performans metriklerini takip eder ve eğitimin ilerleyişini değerlendirmek amacıyla kullanılır.

Grafik, iki bölümden oluşur: Model Doğruluğu ve Model Kaybı.

1. Model Doğruluğu Bölümü:
    - Eğitim veri kümesi ve doğrulama veri kümesi için her epoch'ta elde edilen doğruluk değerleri, epoch sayısına göre çizilir.
    - X-ekseni, her bir epoch'ı temsil ederken, Y-ekseni doğruluk değerlerini yüzde cinsinden gösterir.
    - İki çizgi, eğitim doğruluğunu (mavi çizgi) ve doğrulama doğruluğunu (turuncu çizgi) temsil eder.
2. Model Kaybı Bölümü:
    - Eğitim veri kümesi ve doğrulama veri kümesi için her epoch'ta elde edilen kayıp değerleri, epoch sayısına göre çizilir.
    - X-ekseni, her bir epoch'ı temsil ederken, Y-ekseni kayıp değerlerini gösterir.
    - İki çizgi, eğitim kaybını (mavi çizgi) ve doğrulama kaybını (turuncu çizgi) temsil eder.

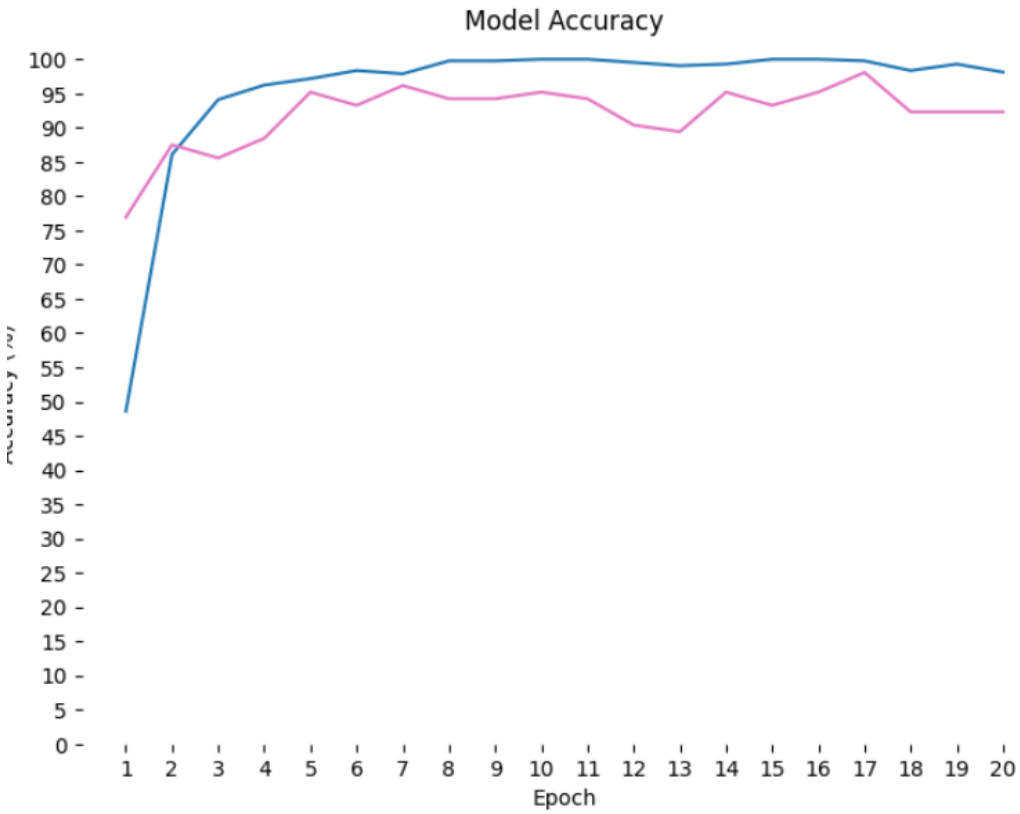
Bu grafik, eğitim sürecinde modelin performansını hızlı bir şekilde değerlendirmek ve aşırı uyuma (overfitting) veya eğitim sürecindeki potansiyel problemleri belirlemek için kullanılır. Eğitim doğruluğu ve kaybının yanı sıra doğrulama doğruluğu ve kaybını bir arada göstererek, modelin genelleme yeteneği ve eğitim sürecindeki tutarlılık hakkında bilgi sağlar.

Bu grafik, projedeki bitki türü tanıma modelinin eğitim sürecinde elde edilen metrikleri anlamak ve modelin performansını değerlendirmek için kullanılabilir. Ayrıca, eğitim sürecinin iyileştirilmesi veya modelin hiperparametrelerinin ayarlanması gerekip gerekmediği konusunda bilgi sağlar.

```
# Model Loss/Accuracy Plot

def plot_model_history(history, model_name, loss_type):

  tableau20 = [(31, 119, 180), (174, 199, 232), (255, 127, 14), (255, 187, 120),
               (44, 160, 44), (152, 223, 138), (214, 39, 40), (255, 152, 150),
               (148, 103, 189), (197, 176, 213), (140, 86, 75), (196, 156, 148),
               (227, 119, 194), (247, 182, 210), (127, 127, 127), (199, 199, 199),
               (188, 189, 34), (219, 219, 141), (23, 190, 207), (158, 218, 229)]

  for i in range(len(tableau20)):
      r, g, b = tableau20[i]
      tableau20[i] = (r / 255., g / 255., b / 255.)

  # Subplots (Vertically stacked)
  fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18, 6))
  fig.suptitle(model_name + ' Training Metrics Plot', fontsize=16, y=1.05)

  ax1.spines["top"].set_visible(False)
  ax1.spines["bottom"].set_visible(False)
  ax1.spines["right"].set_visible(False)
  ax1.spines["left"].set_visible(False)

  ax1.plot(np.arange(1, len(history[1])+1, 1.0), np.array(history[1])*100, color=tableau20[0])
  ax1.plot(np.arange(1, len(history[1])+1, 1.0), np.array(history[3])*100, color=tableau20[12])
  ax1.set_title('Model Accuracy')
  ax1.set_xlabel('Epoch')
  ax1.set_ylabel('Accuracy (%)')
  ax1.set_xticks(np.arange(1, len(history[3])+1, 1.0))
  ax1.set_yticks(np.arange(0, 100+0.05, 5))

  ax2.spines["top"].set_visible(False)
  ax2.spines["bottom"].set_visible(False)
  ax2.spines["right"].set_visible(False)
  ax2.spines["left"].set_visible(False)

  ax2.plot(np.arange(1, len(history[0])+1, 1.0), history[0], color=tableau20[0])
  ax2.plot(np.arange(1, len(history[0])+1, 1.0), history[0], color=tableau20[12])
  ax2.set_title('Model Loss')
  ax2.set_xlabel('Epoch')
  ax2.set_ylabel(loss_type)
  ax2.set_xticks(np.arange(1, len(history[0])+1, 1.0))
  ax2.set_yticks(np.arange(0, max(history[2])+0.05, 0.2))
  ax2.legend(['Train', 'Validation'], loc='upper right')


plot_model_history(history=history,
                   model_name = 'VGG-16',
                   loss_type = 'Cross Entropy')
```

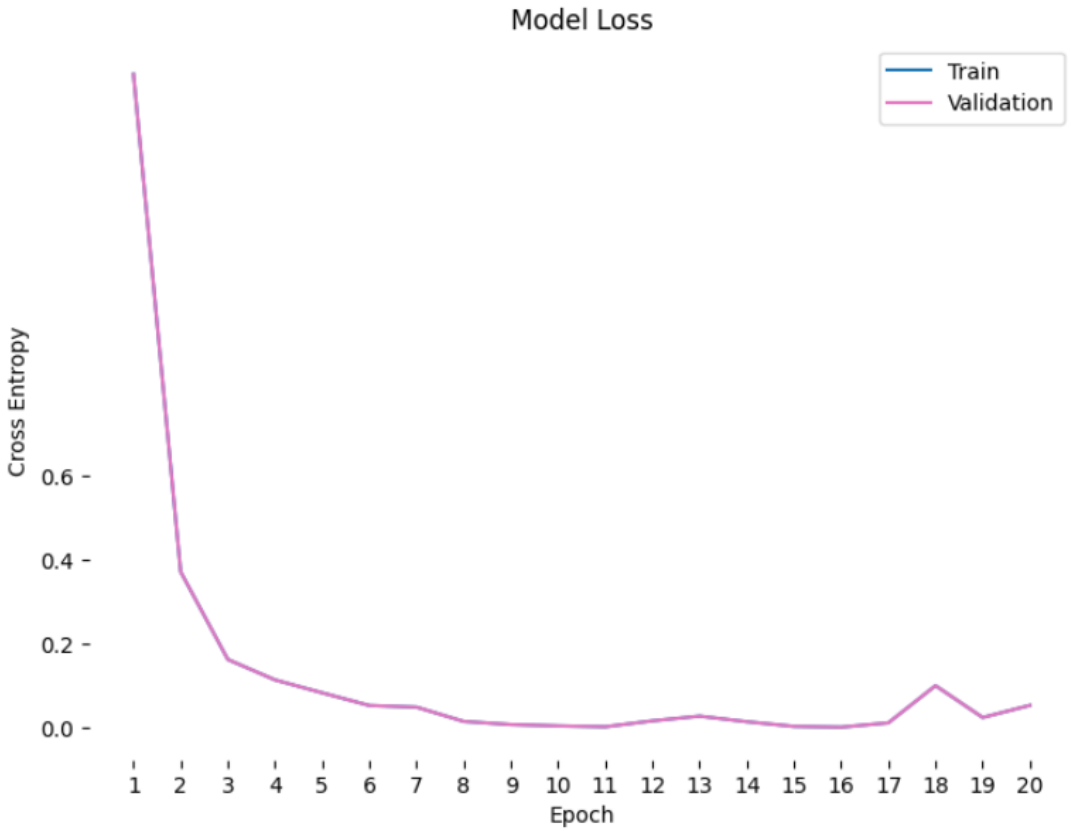
Şekil 14.Doğrulama Modeli

Bu modelde model doğruluğu grafiğinde, doğrulama doğruluğu her epoch'ta giderek artarak bir eğilim göstermiştir. Eğitim ve doğrulama doğrulukları arasındaki benzerlik, modelin genelleme yeteneğini gösterir. İdeal durumda, her iki doğruluk da artmalıdır. Grafiğe baktığımızda eğitim doğruluğu artarken, doğrulama doğruluğu da artıyor, bu modelin iyi genelleme yaptığını gösterir.



Şekil 15. Model Doğruluğu

Model kaybı grafiğine baktığımızda ise, her epoch'ta doğrulama veri kümesinin kayıpları azalıyor bu durum modelin genelleme yeteneğinin iyi olduğunu gösterir.



Şekil 16. Model Kaybı

5.3.4 Tahminleme

Eğitilmiş derin öğrenme modelini kullanarak tek bir çiçek görüntüsü üzerinde sınıflandırma yapmayı amaçlar.

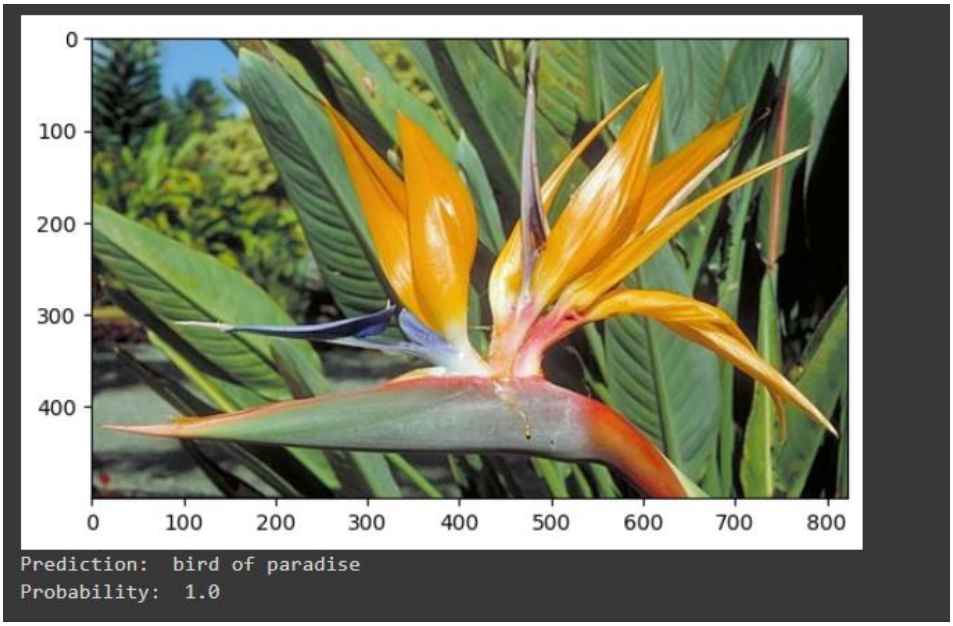
- Fonksiyon Tanımı:
  - getPredictionsSingleImage adlı fonksiyon, modeli, görüntü yolu (imagePath) ve görüntü üzerinde uygulanacak dönüşümleri (transform) parametre olarak alır.
  - Modelin değerlendirme moduna (model.eval()) geçmesini sağlar. Bu, modelin eğitim modundan çıkıp sadece tahmin yapma moduna geçmesini sağlar.
- Görüntü Yükleme ve Gösterme:
  - Image.open(imagePath).convert("RGB") ile belirtilen yol üzerindeki görüntü yüklenir ve renk kanalları RGB'ye dönüştürülür.
  - matplotlib.pyplot.imshow(image) ile görüntü ekrana gösterilir.
- Dönüşüm ve Tahmin:
  - Eğer bir dönüşüm belirtilmişse (transform is not None), görüntü üzerine bu dönüşümler uygulanır.
  - with torch.no_grad(): ifadesi, modelin gradyanlarını hesaplamadan geçici olarak saklar, çünkü bu aşamada sadece tahmin yapılacaktır.
  - predictions = model(image.unsqueeze(dim=0).to(device)) ile model, görüntüyü alır ve sınıflandırma tahminlerini yapar.
  - torch.softmax(predictions, 1).argmax() ile tahminler softmax fonksiyonu ile normalize edilir ve en yüksek olasılığa sahip sınıf belirlenir.
- Sonuçları Gösterme:
  - label_to_species_dict[str(le.inverse_transform([pred_labels.item()])[0])] ile tahmin edilen sınıf etiketi, projede kullanılan etiketlerle eşleştirilir ve ekrana yazdırılır.
  - torch.softmax(predictions, 1).max().item() ile en yüksek olasılık değeri bulunur ve ekrana yazdırılır. Bu, tahminin ne kadar güvenilir olduğunu gösterir.

```python
def getPredictionsSingleImage(model, imagePath, transform=None):
  model.eval()
  image = Image.open(imagePath).convert("RGB")
  matplotlib.pyplot.imshow(image)
  plt.show()
  if transform is not None:
    image = transform(image)
  with torch.no_grad():
    predictions = model(image.unsqueeze(dim=0).to(device))
    pred_labels = torch.softmax(predictions, 1).argmax()
  print('Prediction: ', label_to_species_dict[str(le.inverse_transform([pred_labels.item()])[0])])
  print('Probability: ', round(torch.softmax(predictions, 1).max().item(), 4))


getPredictionsSingleImage(model=modelVGG,
                          imagePath='/content/flowers/jpg/image_03307.jpg',
                          transform=transforms.Compose([transforms.Resize((224, 224)),
                                                        transforms.ToTensor(),
                                                        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])]))
```

Şekil 17. Tahminleme

Bu modelde tahminleme çıktısı: "bird of paradise" (cennet kuşu) olarak belirlenmiştir. Yani, model verilen çiçek görüntüsünün bu bitki türüne ait olduğunu tahmin etmektedir. Tahminleme olasılığı, 1.0 olarak belirlenmiştir. Bu, modelin tahmininin oldukça yüksek bir güven düzeyine sahip olduğunu gösterir. Olasılık değeri 1.0, modelin bu sınıfa ait olduğunu çok yüksek bir güvenle belirttiği anlamına gelir.

Sonuç olarak, modelin bu özel çiçek görüntüsünü "bird of paradise" türüne yüksek bir güvenle sınıflandırdığı söylenebilir. Bu tahmin, modelin eğitildiği veri setindeki örneklerle benzer özelliklere sahip olduğu ve bu nedenle doğru bir tahminde bulunabildiği anlamına gelir.



Şekil 18. Tahminleme Sonucu

Bölüm 6

Sonuç

6.1 Sonuç

Bu projede, Oxford Üniversitesi tarafından sağlanan 102 kategorili çiçek veri seti üzerinde bir bitki türü tanıma makine öğrenimi modeli geliştirildi. Projede, özellikle VGG-16 mimarisi kullanılarak gerçekleştirilen çalışmada, çiçek görüntüleri sınıflandırıldı ve modelin eğitim süreci detaylı bir şekilde takip edildi.

Eğitim sonuçları, modelin çiçek türlerini başarılı bir şekilde tanıma yeteneğini ortaya koydu. 20 epoch boyunca gerçekleştirilen eğitimde, modelin doğruluk oranı artış gösterdi. Eğitim kaybı (loss) değerleri düşerken, doğruluk (accuracy) değerleri ise artış gösterdi. Model, özellikle doğrulama seti üzerinde yüksek bir doğruluk oranı elde etti, bu da genelleme yeteneğinin güçlü olduğunu gösterdi.

Eğitim sonrasında elde edilen model, belirli bir çiçek görüntüsünü doğru bir şekilde sınıflandırma kabiliyetine sahiptir. Örneğin, verilen bir çiçek görüntüsünü "bird of paradise" (cennet kuşu) türüne yüksek bir güven düzeyiyle sınıflandırmıştır.

Projede elde edilen bu olumlu sonuçlar, bitki türü tanıma konusundaki makine öğrenimi alanındaki araştırmalara katkı sağlamaktadır. Gelecekteki çalışmalarda, modelin doğruluğunu artırmak ve daha geniş bir çiçek türü yelpazesini kapsayacak şekilde genişletmek üzerine odaklanılabilir. Bu proje, bitki tanıma teknolojisinin geliştirilmesine katkıda bulunarak doğal yaşamın anlaşılması ve korunması için önemli bir adım oluşturabilir.

Kaynaklar

[1] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In Proc. ICML, 2004.

[2] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In Proc. ICCV, 2007.

[3] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In Proc. CIVR, 2007.

[4] Y. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in ND images. In Proc. ICCV, volume 2, pages 105–112, 2001.

[5] N. Dalal and B. Triggs. Histogram of oriented gradients for human detection. In Proc. CVPR, volume 2, pages 886–893, 2005.

[6] M. Das, R. Manmatha, and E. M. Riseman. Indexing flower patent images using domain knowledge. IEEE Intelligent Systems, 14(5):24–33, 1999.

[7] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In IEEE CVPR Workshop of Generative Model Based Vision, 2004.

[8] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[9] S. Lazebnik, C. Schmid, and J. Ponce. Beyound Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In Proc. CVPR, 2006.

[10] D. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 60(2):91–110, 2004.

[11] T. Malisiewicz and A. Efros. Recognition by association via learning per-exemplar distances. In Proc. CVPR, 2008.

[12] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In Proc. CVPR, volume 2, pages 1447– 1454, 2006.

[13] M.-E. Nilsback and A. Zisserman. Delving into the whorl of flower segmentation. In Proc. BMVC., volume 1, pages 570–579, 2007.

[14] T. Saitoh, K. Aoki, and T. Kaneko. Automatic recognition of blooming flowers. In Proc. ICPR, volume 1, pages 27–30, 2004.

[15] B. Scholkopf and A. Smola. Learning with Kernels. MIT Press, 2002.

[16] M. Varma and D. Ray. Learning the discriminative powerinvariance trade-off. In Proc. ICCV, 2007.

[17] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In Proc. CVPR, volume 2, pages 2126– 2136, 2006.

[18] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. IJCV, 73(2):213–238, Jun 2007.