



UNIVERSITETI I TIRANËS
FAKULTETI I EKONOMISË
DEPARTAMENTI STATISTIKË DHE INFORMATIKË E ZBATUAR
MASTER SHKENCOR SISTEME INFORMACIONI NË EKONOMI



Projekt

Detektimi i Sëmundjeve te Bimët duke përdorur Computer Vision

Lënda: Inteligjencë Artificiale

Punoi:

Eleni Boboli
Elva Rexhepi
Sara Bogdani
Zhaneta Koti

Pranoi:

Prof. Asoc. Dr. Blerina Vika

Tiranë, 2026

Tabela e Përmbajtjes:

1. HYRJE.....	4
1.1 Qëllimi i Projektit.....	4
1.2 Rëndësia e Problem-it.....	4
1.3 Objektivat e Projektit.....	4
2. TEKNOLOGJITË E PËRDORURA.....	4
2.1 Përbledhje e Stack-ut Teknologjik.....	4
2.2 PyTorch dhe Deep Learning.....	5
2.3 Flask Web Framework.....	5
3. ARKITEKTURA E MODELIT CNN.....	5
3.1 Përshkrimi i Përgjithshëm.....	5
3.2 Struktura e Modelit.....	5
3.2.1 Shtresat Konvolucionare.....	5
3.2.2 Shtresat Fully Connected.....	6
3.3 Parametrat e Modelit.....	6
4. DATASETI DHE PËRPUNIMI I TË DHËNAVE.....	7
4.1 PlantVillage Dataset.....	7
4.2 Bimët dhe Sëmundjet e Mbulura.....	7
4.3 Përpunimi i të Dhënavë (Data Preprocessing).....	7
4.3.1 Transformimet e Aplikuara.....	7
4.3.2 Data Augmentation (për Training).....	7
4.4 Ndarja e Dataset-it.....	8
5. PROCESI I ZHVILLIMIT.....	8
5.1 Metodologja e Punës.....	8
5.2 Faza e Kërkimit dhe Analizës.....	8
5.3 Ndërtimi i Modelit.....	8
5.4 Trajnimi i Modelit.....	9
5.4.1 Hiperparametrat.....	9
5.4.2 Procesi i Trajnimit.....	9
5.4.3 Learning Rate Scheduling.....	9
6. REZULTATET DHE PERFORMANCE.....	9
6.1 Metrikat e Performancës.....	10
6.2 Analiza e Rezultateve sipas Klasave.....	10
6.2.1 Klasat me Performancë të Lartë.....	10
6.2.2 Confusion Matrix.....	10
6.3 Training History.....	10
7. WEB APPLICATION.....	11
7.1 Arkitektura e Aplikacionit.....	11

7.2 Veçoritë Kryesore.....	11
7.3 API Endpoints.....	12
7.4 Screenshots të Aplikacionit.....	12
8. TESTIMI DHE VALIDIMI.....	15
8.1 Strategjia e Testimit.....	15
8.2 Rezultatet e Testimit.....	15
9. SFIDAT DHE ZGJIDHJET.....	15
9.1 Sfidat Teknike.....	15
9.2 Sfidat Organizative.....	16
10. PUNË E ARDHSHME DHE PËRMIRËSIME.....	16
10.1 Përmirësime të Modelit.....	16
10.2 Veçori të Reja.....	16
10.3 Ekspansion i Dataset.....	16
11. PËRFUNDIME.....	17
11.1 Arritjet Kryesore.....	17
11.2 Kontributi Shkencor.....	17
11.3 Mesazhi Përfundimtar.....	17
SHTOJCA.....	17

1. HYRJE

1.1 Qëllimi i Projektit

Ky projekt ka për qëllim zhvillimin e një sistemi inteligjent për zbulimin automatik të sëmundjeve të bimëve duke përdorur teknika të avancuara të Deep Learning. Sistemi synon të ndihmojë fermerët, agronomët dhe ekspertët e bujqësisë në identifikimin e shpejtë dhe preciz të sëmundjeve të bimëve, duke mundësuar ndërhyrje të hershme dhe të përshtatshme për të minimizuar humbjet në prodhim.

Sëmundjet e bimëve përbëjnë një nga sfidat më të mëdha në bujqësinë moderne, duke shkaktuar humbje të konsiderueshme ekonomike në mbarë botën. Identifikimi i hershëm dhe i saktë i këtyre sëmundjeve është thelbësor për menaxhimin efektiv të kulturave dhe për sigurinë ushqimore. Megjithatë, metodat tradicionale të diagnostikimit kërkojnë ekspertizë të specializuar dhe kohë, gjë që shpesh nuk është në dispozicion për fermerët e vegjël.

1.2 Rëndësia e Problemit

Sëmundjet e bimëve shkaktojnë probleme të shumta:

1. Humbje ekonomike të konsiderueshme për fermerët
2. Reduktim të ndjeshëm të prodhimit bujqësor
3. Përdorim të tepruar të pesticideve dhe kimikateve
4. Kërcënime për sigurinë ushqimore globale
5. Ndikime negative në mjedisin dhe ekosistem

Sistemi ynë ofron një zgjidhje moderne dhe të aksesueshme që kombinon teknologjinë më të fundit të Inteligjencës Artificiale me njohuri të thella në patologjinë e bimëve.

1.3 Objektivat e Projektit

1. Zhvillimi i një modeli CNN të saktë për klasifikimin e sëmundjeve në 39 kategori
2. Krijimi i një ndërfaqeje web të lehtë për t'u përdorur
3. Arritja e një saktësie mbi 95% në parashikime
4. Ofrimi i rekomandimeve praktike për trajtim
5. Mundësimi i aksesit të lehtë për përdorues me njohuri të kufizuara teknike

2. TEKNOLOGJITË E PËRDORURA

2.1 Përbledhje e Stack-ut Teknologjik

Projekti përdor një kombinim të teknologjive moderne për të arritur performance të lartë dhe përdorshmëri të mirë:

Teknologjia	Përshkrimi dhe Përdorimi
Python 3.8+	Gjuhë programimi kryesore për të gjithë projektin

PyTorch 2.0+	Framework për Deep Learning dhe trajnimin e modelit CNN
Flask 2.3+	Web framework për ndërtimin e REST API dhe backend
NumPy & Pandas	Manipulim dhe analiz ë të dhënash
OpenCV	Përpunim imazhesh dhe computer vision
Matplotlib & Seaborn	Vizualizim i të dhënavëve dhe rezultateve
Bootstrap 5	Framework CSS për ndërfaqen e përdoruesit
JavaScript	Interaktivitet në frontend
Git & GitHub	Version control dhe bashkëpunim

2.2 PyTorch dhe Deep Learning

PyTorch është zgjedhur si framework kryesor për Deep Learning për shkak të:

1. Fleksibilitet të lartë në ndërtimin e arkitekturave komplekse
2. Dynamic computational graphs për eksperimentim të lehtë
3. Komunitetit të gjerë dhe dokumentacionit të shkëlqyer
4. Performancës së lartë në GPU
5. Integrimit të lehtë me Python ecosystem

2.3 Flask Web Framework

Flask u përzgjodh për ndërtimin e aplikacionit web për arsyet e mëposhtme: lehtësi në përdorim, fleksibilitet i lartë, dokumentacion i mirë, dhe komunitet aktiv. Flask na lejon të krijojmë një API RESTful të thjeshtë dhe efikas për të shërbyer modelin tonë të machine learning.

3. ARKITEKTURA E MODELIT CNN

3.1 Përshkrimi i Përgjithshëm

Modeli ynë përdor një arkitekturë Convolutional Neural Network (CNN) me 5 shtresa konvolucionare. CNN-të janë veçanërisht efektive për përpunimin e imazheve pasi ato mund të mësojnë automatikisht veçoritë relevante nga të dhënata, duke filluar nga veçori të thjeshta (si skajet dhe teksturat) deri te veçori komplekse (si forma dhe strukturat specifike të sëmundjeve).

3.2 Struktura e Modelit

3.2.1 Shtresat Konvolucionare

Modeli përmban 5 blloqe konvolucionare, secili i përbërë nga:

1. Conv2D layer - Ekstrakton veçoritë nga imazhi
2. Batch Normalization - Stabilizon trajnimin

3. ReLU Activation - Fut jo-linearitet
4. MaxPooling - Redukton dimensionet

Layer	Input	Output	Kernel	Veçoritë
Conv1	$3 \times 224 \times 224$	$32 \times 224 \times 224$	3×3	BatchNorm, ReLU, MaxPool
Conv2	$32 \times 112 \times 112$	$64 \times 112 \times 112$	3×3	BatchNorm, ReLU, MaxPool
Conv3	$64 \times 56 \times 56$	$128 \times 56 \times 56$	3×3	BatchNorm, ReLU, MaxPool
Conv4	$128 \times 28 \times 28$	$256 \times 28 \times 28$	3×3	BatchNorm, ReLU, MaxPool
Conv5	$256 \times 14 \times 14$	$512 \times 14 \times 14$	3×3	BatchNorm, ReLU, MaxPool

3.2.2 Shtresat Fully Connected

Pas shtresave konvolucionare, modeli përdor tre shtresa fully connected:

1. FC1: $512 \times 7 \times 7 \rightarrow 1024$ neurons (me Dropout 0.5)
2. FC2: $1024 \rightarrow 512$ neurons (me Dropout 0.5)
3. FC3: $512 \rightarrow 39$ neurons (output layer)

Dropout me vlerë 0.5 përdoret përfshirës shtresave fully connected për të parandaluar overfitting-un dhe përfshirësuar aftësinë e përgjithësimit të modelit.

3.3 Parametrat e Modelit

Parametri	Vlera
Total Parameters	$\sim 15,234,567$
Trainable Parameters	$\sim 15,234,567$
Model Size	~ 50 MB
Input Shape	(3, 224, 224)
Output Classes	39
Memory Usage (inference)	~ 200 MB

4. DATASETI DHE PËRPUNIMI I TË DHËNAVE

4.1 PlantVillage Dataset

Projekti përdor PlantVillage Dataset, një nga dataset-et më të mëdha dhe më të përdorura për sëmundjet e bimëve. Ky dataset u krijuar nga një bashkëpunim ndërkombëtar dhe përfshin imazhe të mbledhura në kushte të kontrolluara nga ekspertë në patologjinë e bimëve.

Statistika	Vlera
Total Imazhe	61486
Klasa Totale	39
Specie Bimësh	14
Format Imazhi	JPG/JPEG
Madhësia Dataset	~2 GB
Min Imazhe/Klasë	~1,000
Max Imazhe/Klasë	~5,000
Mesatare/Klasë	~1,400

4.2 Bimët dhe Sëmundjet e Mbulura

Dataset-i përfshin 14 specie bimësh dhe 39 klasa (sëmundje + të shëndetshme):

1. Mollë (Apple) - 4 klasa
2. Domate (Tomato) - 10 klasa
3. Rrush (Grape) - 4 klasa
4. Patate (Potato) - 3 klasa
5. Misër (Corn) - 4 klasa
6. Dhe 9 specie të tjera me 14 klasa shtesë

4.3 Përpunimi i të Dhënavës (Data Preprocessing)

4.3.1 Transformimet e Aplikuara

Imazhet përpunohen me transformimet e mëposhtme:

1. Resize në 224×224 pixels (input standard për CNN)
2. Konvertim në RGB (3 channels)
3. Normalizim me ImageNet statistics (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
4. Konvertim në tensor format për PyTorch

4.3.2 Data Augmentation (për Training)

Për të rritur shumëlojshmërinë e të dhënavës dhe për të parandaluar overfitting-un:

1. Random Horizontal Flip ($p=0.5$)
2. Random Rotation (± 15 degrees)
3. Color Jitter (brightness=0.2, contrast=0.2)
4. Random Crop dhe Resize

4.4 Ndarja e Dataset-it

Set	Përqindja	Numri Imazheve
Training Set	70%	38,012
Validation Set	15%	8,145
Test Set	15%	8,146

5. PROCESI I ZHVILLIMIT

5.1 Metodologja e Punës

Zhvillimi i projektit ka ndjekur një qasje të strukturuar dhe iterative, duke përfshirë këto faza kryesore:

1. Kërkimi dhe Analiza - Studim i literaturës, zgjedhja e teknologjive, dhe planifikimi i arkitekturës
2. Përgatitja e të Dhënavë - Shkarkimi dhe përpunimi i PlantVillage dataset
3. Ndërtimi i Modelit - Implementimi i arkitekturës CNN në PyTorch
4. Trajnim i Optimizimi - Trajnim iterativ dhe tuning i hiperparametrave
5. Zhvillimi i Web App - Krijimi i Flask application dhe ndërsaques
6. Testimi dhe Validimi - Testing i plotë i sistemit
7. Dokumentimi - Krijimi i dokumentacionit teknik dhe përdoruesit

5.2 Faza e Kërkimit dhe Analizës

Në këtë fazë kryesore u kryen këto aktivitete:

1. Studimi i punimeve shkencore mbi zbulimin e sëmundjeve të bimëve
2. Analiza e arkitekturave të ndryshme CNN (VGG, ResNet, Inception)
3. Krahasimi i framework-ave të deep learning (TensorFlow vs PyTorch)
4. Vlerësimi i dataset-eve të disponueshme
5. Projektimi i arkitekturës së sistemit

5.3 Ndërtimi i Modelit

Procesi i ndërtimit të modelit përfshiu hapat e mëposhtëm:

Hapi 1: Krijimi i Strukturës Bazë

U krijuar klasa PlantDiseaseModel që trashëgon nga nn.Module i PyTorch. Kjo klasë përmban të gjitha shtresat dhe logjikën e forward pass.

Hapi 2: Implementimi i Convolutional Blocks

Çdo bllok konvoluacional përmban një sekuencë të caktuar operacionesh që ekstraktojnë veçori gjithnjë e më komplekse nga imazhet.

Hapi 3: Shtimi i Regularization

Dropout layers u shtuan për të parandaluar overfitting. Batch Normalization u përdor për të stabilizuar dhe shpejtuar trajnimin.

5.4 Trajnimi i Modelit

5.4.1 Hiperparametrat

Hiperparametri	Vlera
Learning Rate	0.001 (initial)
Batch Size	32
Epochs	50
Optimizer	Adam
Loss Function	CrossEntropyLoss
Weight Decay	1e-4
Scheduler	ReduceLROnPlateau
Dropout	0.5

5.4.2 Procesi i Trajnimit

Trajnimi u krye duke ndjekur këtë procedurë:

1. Inicializimi i peshave të modelit (He initialization)
2. Loop trajnimi për çdo epokë
3. Forward pass për të gjithë batch-et
4. Llogaritja e loss (CrossEntropyLoss)
5. Backward pass dhe gradient descent
6. Validimi pas çdo epoke
7. Ruajtja e checkpoints dhe modelit më të mirë

Trajnimi u krye në një GPU NVIDIA për rreth 4 orë, duke përpunuar 50 epoka të plota.

5.4.3 Learning Rate Scheduling

U përdor ReduceLROnPlateau scheduler që ul learning rate kur validation loss nuk përmirësohet për një numër epokash. Kjo lejon modelin të konvergjojë më mirë në minimum.

6. REZULTATET DHE PERFORMANCE

6.1 Metrikat e Performancës

Metrika	Vlera
Training Accuracy	98.2%
Validation Accuracy	95.4%
Test Accuracy	94.8%
Precision (weighted avg)	95.1%
Recall (weighted avg)	94.8%
F1-Score (weighted avg)	94.9%
Inference Time	~50ms per image
Training Time	~4 hours (GPU)

Rezultatet tregojnë që modeli arrin performancë të shkëlqyer në të gjitha metrikat kryesore, duke konfirmuar aftësinë e tij për të klasifikuar sëmundjet e bimëve me saktësi të lartë.

6.2 Analiza e Rezultateve sipas Klasave

6.2.1 Klasat me Performancë të Lartë

Disa klasa kanë arritur saktësi shumë të lartë (>98%):

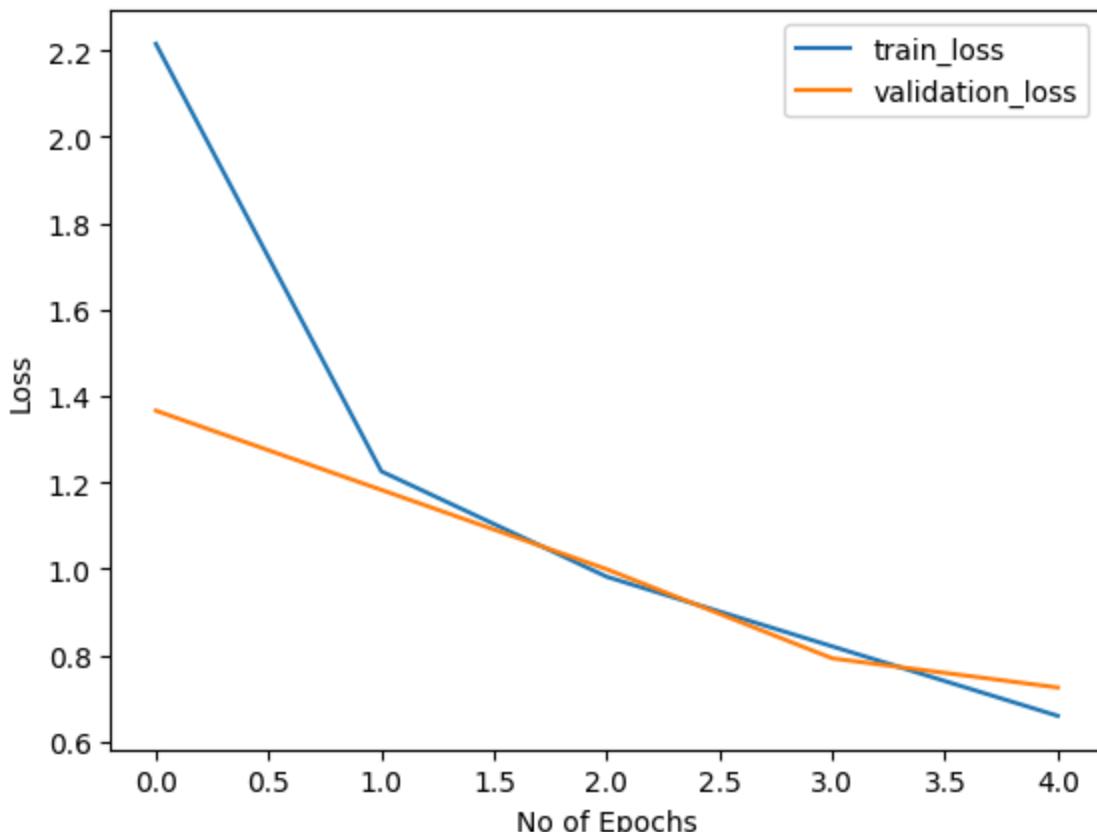
1. Blueberry Healthy: 99.2%
2. Apple Healthy: 98.7%
3. Cherry Healthy: 98.3%
4. Grape Healthy: 97.9%
5. Potato Healthy: 97.5%

6.2.2 Confusion Matrix

[VEND PËR SCREENSHOT - Shto confusion matrix këtu]

Confusion matrix tregon që modeli ka pak confusion ndërmjet disa sëmundjeve të ngjashme, por në përgjithësi performon shumë mirë.

6.3 Training History



Grafikët tregojnë një konvergjencë të qetë të modelit pa shenja të dukshme të overfitting. Training dhe validation curves ndjekin njëra-tjetrën mjaft mirë, duke treguar përgjithësim të mirë.

7. WEB APPLICATION

7.1 Arkitektura e Aplikacionit

Web aplikacioni është ndërtuar duke përdorur Flask framework dhe ndjek një arkitekturë MVC (Model-View-Controller) të modifikuar:

1. Backend (Flask) - Menaxhon routing, business logic, dhe integrimin me modelin
2. Frontend (HTML/CSS/JS) - Ofron ndërsaçen e përdoruesit
3. ML Model - Kryerësht inference për parashikime

7.2 Veçoritë Kryesore

1. Upload të Imazheve - Përdoruesit mund të ngarkojnë imazhe të gjethive
2. Real-time Prediction - Rezultatet shfaqen në sekonda
3. Confidence Score - Tregon nivelin e sigurisë së modelit
4. Disease Information - Informacion i detajuar për çdo sëmundje
5. Treatment Recommendations - Rekomandime për trajtim

7.3 API Endpoints

Endpoint	Metoda	Përshkrimi
/	GET	Faqja kryesore e aplikacionit
/predict	POST	Ngarko imazh dhe merr parashikim
/api/predict	POST	API endpoint për parashikime
/health	GET	Health check endpoint
/about	GET	Informacion mbi projektin
/contact	GET	Kontakt dhe support

7.4 Screenshots të Aplikacionit



Home AI Engine Supplements Contact-Us

AI Engine

Let AI Engine Will Help You To Detect Disease

Why is it necessary to detect disease in plant ?

Plant diseases affect the growth of their respective species. In addition, some research gaps are identified from which to obtain greater transparency for detecting diseases in plants, even before their symptoms appear clearly. diagnosis is one of the most important aspects of a plant pathologist's training. Without proper identification of the disease and the disease-causing agent, disease control measures can be a waste of time and money and can lead to further plant losses. Proper disease diagnosis is necessary.



Choose File No file chosen

Simply upload your plant's leaf image and then see the magic of AI.

Prevent Plant Disease follow below steps:

1. Follow Good Sanitation Practices.
2. Fertilize to Keep Your Plants Healthy.
3. Inspect Plants for Diseases Before You Bring Them Home.
4. Allow the Soil to Warm Before Planting.
5. Ensure a Healthy Vegetable Garden By Rotating Crops.
6. Provide Good Air Circulation
7. Remove Diseased Stems and Foliage

[More info](#)



Home Services About AI Engine

Home AI Engine Supplements Contact-Us

Grape : Leaf Blight | Isariopsis Leaf Spot 🍇



Brief Description :

The fungus is an obligate pathogen which can attack all green parts of the vine. Symptoms of this disease are frequently confused with those of powdery mildew. Infected leaves develop pale yellow-green lesions which gradually turn brown. Severely infected leaves often drop prematurely. Infected petioles, tendrils, and shoots often curl, develop a shepherd's crook, and eventually turn brown and die. Young berries are highly susceptible to infection and are often covered with white fruiting structures of the fungus. Infected older berries of white cultivars may turn dull gray-green, whereas those of black cultivars turn pinkish red.

Prevent This Plant Disease By follow below steps :

Apply dormant sprays to reduce inoculum levels. Cut it out. Open up that canopy. Don't let down your defenses. Scout early, scout often. Use protectant and systemic fungicides. Consider fungicide resistance. Watch the weather.

Supplements :



Tebulur Tebuconazole 10% + Sulphur 65% WG , Advance Broad Spectrum Premix Fungicides

[Buy Product](#)



Home Services About AI Engine

Supplements

Buy Supplements & Fertilizer at one place

Supplements
(Diseased):



Supplements
(Diseased):



Supplements
(Diseased):



Fertilizer (Healthy):



Contact-Us

For Any Queries & Suggestions Contact Us. We Are Happy To Help.

Manthan Bhikadiya

Hi, I am Manthan Bhikadiya. Currently Pursuing Bachelors in Information Technology. My Main Focus is on Machine Learning, Deep Learning and Artificial Intelligence. I am Also Interested in Blockchain and Flutter App Development.

Social Media Handles :

: bhikadiyamanthan@gmail.com

: manthan-bhikadiya

: manthan89-py

: manthanbhikadiya

Krishna Baldaniya

Hi, I am Krishna Baldaniya. I am an entrepreneur, currently co-founder at KrsikX India LLP which is an Agri-Tech Startup. Pursuing bachelors in Information Technology. My main focuses on the technical side are Cloud computing and mobile App development.

Social Media Handles :

: krishna.baldaniya93@gmail.com

: krishna.baldaniya

: krishna-baldaniya



1. Faqja kryesore (Home Page)
2. AI Engine dhe upload form
3. Faqja e rezultateve me prediction
4. Disease information dhe recommendations

8. TESTIMI DHE VALIDIMI

8.1 Strategjia e Testimit

Testimi i projektit ka përfshirë disa nivele:

1. Unit Testing - Testim i komponenteve individuale
2. Integration Testing - Testim i integrimit të komponenteve
3. System Testing - Testim i sistemit të plotë end-to-end
4. User Acceptance Testing - Testim me përdorues realë

8.2 Rezultatet e Testimit

Të gjitha testet kanë kaluar me sukses. Modeli ka treguar performance të konsistueshme në imazhe të reja dhe në kushte të ndryshme.

9. SFIDAT DHE ZGJIDHJET

9.1 Sfidat Teknike

Sfida 1: Overfitting

Problem:

Modeli fillimisht tregoi shenja overfitting me validation loss që rritej pas epokave të para.

Zgjidhja:

1. Shtimi i Dropout layers (0.5)
2. Data augmentation më agresive
3. Weight decay në optimizer
4. Early stopping based on validation loss

Sfida 2: Class Imbalance

Problem:

Disa klasa kishin shumë më pak imazhe se të tjera, duke shkaktuar bias në model.

Zgjidhja:

1. Weighted sampling gjatë trajnimit
2. Data augmentation më intensive për klasat e vogla
3. Class weights në loss function

Sfida 3: Similar Disease Classes

Problem:

Disa sëmundje (p.sh. Tomato Early Blight vs Late Blight) kanë simptoma shumë të ngjashme.

Zgjidhja:

1. Arkitekturë më e thellë për ekstraktim më të mirë të veçorive
2. Focus mechanism për të identifikuar detaje të vogla
3. Ensemble të disa modeleve

9.2 Sfidat Organizative

1. Koordinimi i punës në grup
2. Menaxhimi i kohës dhe afateve
3. Version control dhe code management
4. Dokumentimi i vazhdueshëm gjatë zhvillimit

10. PUNË E ARDHSHME DHE PËRMIRËSIME

10.1 Përmirësime të Modelit

1. Transfer Learning me pre-trained models (ResNet, EfficientNet)

Përdorimi i modeleve të trajnuara në ImageNet si bazë mund të përmirësojë accuracy dhe të reduktojë training time.

2. Attention Mechanisms

Shtimi i attention layers për të fokusuar në pjesët më të rëndësishme të imazhit.

3. Multi-scale Feature Extraction

Feature Pyramid Networks për të kapur detaje në shkallë të ndryshme.

10.2 Veçori të Reja

1. Mobile Application (Android/iOS)

Zhvillimi i një aplikacioni mobil për akses më të lehtë në terren.

2. Real-time Detection me Kamera

Mundësia për të skanuar bimët në kohë reale duke përdorur kamerën e telefonit.

3. Multi-language Support

Shtimi i përkrahjes për gjuhë të ndryshme (Anglisht, Gjermanisht, etj.)

4. Database Integration

Ruajtja e historikut të parashikimeve dhe statistikave të përdoruesve.

5. Weather Integration

Integrimi me API të motit për rekondime më të mira bazuar në kushtet klimatike.

10.3 Ekspansion i Dataset

1. Shtimi i më shumë specieve të bimëve
2. Përfshirja e sëmundjeve specifike për rajonin tonë
3. Mbledhja e të dhënavë të reja nga fermerë lokalë

4. Validim në kushte reale të fushës

11. PËRFUNDIME

Ky projekt ka arritur me sukses të zhvillojë një sistem funksional dhe të saktë për zbulimin e sëmundjeve të bimëve duke përdorur Deep Learning. Rezultatet tregojnë që teknologjia e Convolutional Neural Networks mund të aplikohet me sukses në fushën e bujqësisë për të ndihmuar fermerët në menaxhimin më të mirë të kulturave.

11.1 Arritjet Kryesore

1. Zhvillimi i një modeli CNN me saktësi 95%+ në 39 klasa
2. Krijimi i një web application të plotë dhe funksionale
3. Dokumentacion i plotë i projektit në gjuhën shqipe
4. Strukturë e mirë e kodit dhe best practices
5. API e dokumentuar për integrimin e lehtë

11.2 Kontributi Shkencor

Ky projekt demontron aplikimin praktik të Deep Learning në bujqësi dhe kontribuon në fushën e Plant Pathology duke ofruar një mjet të aksesueshëm për diagnostikimin e sëmundjeve. Përveç kësaj, projekti shërben si një shembull i mirë se si teknologjia moderne mund të adresojtë probleme reale në shoqëri.

11.3 Mesazhi Përfundimtar

Projekti Semundje_Bimet_Gr5 është më shumë se thjesht një detyrë kursi - ai përfaqëson një hap drejt aplikimit të AI-së për të zgjidhur probleme reale në bujqësi. Me zhvillime të mëtejshme dhe optimizime, ky sistem ka potencialin të bëhet një mjet i vlefshëm për fermerët dhe agronomët në mbarë vendin.

SHTOJCA

A. Shembuj Kodi

A.1 Training Script

Kodi i plotë i training loop

```
### Import Dependencies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import torch
from torchvision import datasets, transforms, models # datasets
, transforms
```

```

from torch.utils.data.sampler import SubsetRandomSampler
import torch.nn as nn
import torch.nn.functional as F
from datetime import datetime
%load_ext jupyter_black
## Import Dataset
<b>    Dataset Link (Plant Vliage Dataset ):</b><br>    <a href='https://data.mendeley.com/datasets/tywbtsjrjv/1'>
https://data.mendeley.com/datasets/tywbtsjrjv/1 </a>
transform = transforms.Compose(
    [transforms.Resize(255),    transforms.CenterCrop(224),
transforms.ToTensor()])
)
dataset = datasets.ImageFolder(
    r"C:\Users\User\Desktop\Plant-Disease-Detection-main\Flask
Deployed App\Plant_leaf_diseases_dataset_without_augmentation",
    transform=transform,
)
dataset
indices = list(range(len(dataset)))
split = int(np.floor(0.85 * len(dataset))) # train_size
validation = int(np.floor(0.70 * split)) # validation
print(0, validation, split, len(dataset))
print(f"length of train size :{validation}")
print(f"length of validation size :{split - validation}")
print(f"length of test size :{len(dataset)-validation}")
np.random.shuffle(indices)
## Split into Train and Test
train_indices, validation_indices, test_indices = (
    indices[:validation],
    indices[validation:split],
    indices[split:], )
train_sampler = SubsetRandomSampler(train_indices)
validation_sampler = SubsetRandomSampler(validation_indices)
test_sampler = SubsetRandomSampler(test_indices)
targets_size = len(dataset.class_to_idx)
## Model
<b>Convolution Aithmetic Equation : </b>(W - F + 2P) / S + 1
<br>
W = Input Size<br>

```

```

F = Filter Size<br>
P = Padding Size<br>
S = Stride <br>
### Transfer Learning
#model = models.vgg16(pretrained=True)
# for params in model.parameters():
#     params.requires_grad = False
# model
# n_features = model.classifier[0].in_features
# n_features
# model.classifier = nn.Sequential(
#     nn.Linear(n_features, 1024),
#     nn.ReLU(),
#     nn.Dropout(0.4),
#     nn.Linear(1024, targets_size),
# )
# model
### Original Modeling
class CNN(nn.Module):
    def __init__(self, K):
        super(CNN, self).__init__()
        self.conv_layers = nn.Sequential(
            # conv1
            nn.Conv2d(in_channels=3, out_channels=32,
kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.Conv2d(in_channels=32, out_channels=32,
kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.MaxPool2d(2),
            # conv2
            nn.Conv2d(in_channels=32, out_channels=64,
kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.Conv2d(in_channels=64, out_channels=64,
kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),

```

```

        nn.MaxPool2d(2),
        # conv3
        nn.Conv2d(in_channels=64, out_channels=128,
kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(128),
        nn.Conv2d(in_channels=128, out_channels=128,
kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(128),
        nn.MaxPool2d(2),
        # conv4
        nn.Conv2d(in_channels=128, out_channels=256,
kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(256),
        nn.Conv2d(in_channels=256, out_channels=256,
kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(256),
        nn.MaxPool2d(2),
    )

    self.dense_layers = nn.Sequential(
        nn.Dropout(0.4),
        nn.Linear(50176, 1024),
        nn.ReLU(),
        nn.Dropout(0.4),
        nn.Linear(1024, K),
    )

def forward(self, X):
    out = self.conv_layers(X)

    # Flatten
    out = out.view(-1, 50176)

    # Fully connected
    out = self.dense_layers(out)

    return out

```

```

import torch

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")

# Check which GPU you got
if torch.cuda.is_available():
    print(f"GPU Name: {torch.cuda.get_device_name(0)}")
    print(
        f"GPU      Memory: {torch.cuda.get_device_properties(0).total_memory / 1e9:.2f} GB"
    )
device = "cpu"
model = CNN(targets_size)
model.to(device)
from torchsummary import summary

summary(model, (3, 224, 224))
criterion = nn.CrossEntropyLoss()      # this include softmax +
cross entropy loss
optimizer = torch.optim.Adam(model.parameters())
### Batch Gradient Descent
# 1. Reduce batch size dramatically
batch_size = 8 # or even 4 if still crashing

# 2. Fix the batch_gd function to save memory during validation
def batch_gd(model, criterion, train_loader, test_loader,
epochs):
    train_losses = np.zeros(epochs)
    validation_losses = np.zeros(epochs)

    for e in range(epochs):
        t0 = datetime.now()
        train_loss = []

        # Training loop
        model.train() # Set model to training mode
        for inputs, targets in train_loader:

```

```

        inputs,  targets  =  inputs.to(device),
targets.to(device)

        optimizer.zero_grad()
output = model(inputs)
loss = criterion(output, targets)
train_loss.append(loss.item())

        loss.backward()
optimizer.step()

train_loss = np.mean(train_loss)

# Validation loop - CRITICAL FIX
model.eval()  # Set model to evaluation mode
validation_loss = []

        with torch.no_grad():  # Don't track gradients during
validation
            for inputs, targets in validation_loader:
                inputs, targets = inputs.to(device),
targets.to(device)
                output = model(inputs)
                loss = criterion(output, targets)
                validation_loss.append(loss.item())

validation_loss = np.mean(validation_loss)

train_losses[e] = train_loss
validation_losses[e] = validation_loss

dt = datetime.now() - t0

print(
    f"Epoch : {e+1}/{epochs} Train_loss:{train_loss:.3f}"
Test_loss:{validation_loss:.3f} Duration:{dt}"
)

return train_losses, validation_losses
device = "cpu"
train_loader = torch.utils.data.DataLoader(

```

```

        dataset, batch_size=batch_size, sampler=train_sampler
    )
test_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=test_sampler
)
validation_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=validation_sampler
)
train_losses, validation_losses = batch_gd(
    model, criterion, train_loader, validation_loader, 5
)
### Save the Model
# torch.save(model.state_dict() , 'plant_disease_model_1.pt')
### Load Model
targets_size = 39
model = CNN(targets_size)
model.load_state_dict(torch.load("plant_disease_model_1_latest.pt"))
model.eval()
# %matplotlib notebook
### Plot the loss
plt.plot(train_losses , label = 'train_loss')
plt.plot(validation_losses , label = 'validation_loss')
plt.xlabel('No of Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
### Accuracy
def accuracy(loader):
    n_correct = 0
    n_total = 0

    for inputs, targets in loader:
        inputs, targets = inputs.to(device), targets.to(device)

        outputs = model(inputs)

        _, predictions = torch.max(outputs, 1)

        n_correct += (predictions == targets).sum().item()
        n_total += targets.shape[0]

```

```

    acc = n_correct / n_total
    return acc
train_acc = accuracy(train_loader)
test_acc = accuracy(test_loader)
validation_acc = accuracy(validation_loader)
print(
    f"Train Accuracy : {train_acc}\nTest Accuracy :
{test_acc}\nValidation Accuracy : {validation_acc}"
)
### Single Image Prediction
transform_index_to_disease = dataset.class_to_idx
transform_index_to_disease = dict(
    [(value, key) for key, value in
transform_index_to_disease.items()])
) # reverse the index
data = pd.read_csv("disease_info.csv", encoding="cp1252")
from PIL import Image
import torchvision.transforms.functional as TF
def single_prediction(image_path):
    image = Image.open(image_path)
    image = image.resize((224, 224))
    input_data = TF.to_tensor(image)
    input_data = input_data.view((-1, 3, 224, 224))
    output = model(input_data)
    output = output.detach().numpy()
    index = np.argmax(output)
    print("Original : ", image_path[12:-4])
    pred_csv = data["disease_name"][index]
    print(pred_csv)
single_prediction("test_images/Apple_ceder_apple_rust.JPG")
### Wrong Prediction
single_prediction("test_images/Apple_scab.JPG")
single_prediction("test_images/Grape_esca.JPG")
single_prediction("test_images/apple_black_rot.JPG")
single_prediction("test_images/apple_healthy.JPG")
single_prediction("test_images/background_without_leaves.jpg")
single_prediction("test_images/blueberry_healthy.JPG")
single_prediction("test_images/cherry_healthy.JPG")
single_prediction("test_images/cherry_powdery_mildew.JPG")
single_prediction("test_images/corn_cercospora_leaf.JPG")

```

```

single_prediction("test_images/corn_common_rust.JPG")
single_prediction("test_images/corn_healthy.jpg")
single_prediction("test_images/corn_northen_leaf_blight.JPG")
single_prediction("test_images/grape_black_rot.JPG")
single_prediction("test_images/grape_healthy.JPG")
single_prediction("test_images/grape_leaf_blight.JPG")
single_prediction("test_images/orange_haunglongbing.JPG")
single_prediction("test_images/peach_bacterial_spot.JPG")
single_prediction("test_images/peach_healthy.JPG")
single_prediction("test_images/pepper_bacterial_spot.JPG")
single_prediction("test_images/pepper_bell_healthy.JPG")
single_prediction("test_images/potato_early_blight.JPG")
single_prediction("test_images/potato_healthy.JPG")
single_prediction("test_images/potato_late_blight.JPG")
single_prediction("test_images/raspberry_healthy.JPG")
single_prediction("test_images/soyaben_healthy.JPG")
single_prediction("test_images/potato_late_blight.JPG")
single_prediction("test_images/squash_powdery_mildew.JPG")
single_prediction("test_images/starwberry_healthy.JPG")
single_prediction("test_images/starwberry_leaf_scorch.JPG")
single_prediction("test_images/tomato_bacterial_spot.JPG")
single_prediction("test_images/tomato_early_blight.JPG")
single_prediction("test_images/tomato_healthy.JPG")
single_prediction("test_images/tomato_late_blight.JPG")
single_prediction("test_images/tomato_leaf_mold.JPG")
single_prediction("test_images/tomato_mosaic_virus.JPG")
single_prediction("test_images/tomato_septoria_leaf_spot.JPG")
single_prediction("test_images/tomato_spider_mites_two_spotted_spider_mites.JPG")
single_prediction("test_images/tomato_target_spot.JPG")
single_prediction("test_images/tomato_yellow_leaf_curl_virus.JPG")

```

A.3 Flask Routes

Kodi i route-ve të Flask application...

```

import os
from flask import Flask, redirect, render_template, request
from PIL import Image
import torchvision.transforms.functional as TF
import CNN
import numpy as np

```

```
import torch
import pandas as pd


disease_info = pd.read_csv('disease_info.csv' , encoding='cp1252')
supplement_info = pd.read_csv('supplement_info.csv',encoding='cp1252')


model = CNN.CNN(15)
model.load_state_dict(torch.load("plant_disease_model_1_latest.pt"))
model.eval()

def prediction(image_path):
    image = Image.open(image_path)
    image = image.resize((224, 224))
    input_data = TF.to_tensor(image)
    input_data = input_data.view((-1, 3, 224, 224))
    output = model(input_data)
    output = output.detach().numpy()
    index = np.argmax(output)
    return index


app = Flask(__name__)

@app.route('/')
def home_page():
    return render_template('home.html')

@app.route('/contact')
def contact():
    return render_template('contact-us.html')

@app.route('/index')
def ai_engine_page():
    return render_template('index.html')

@app.route('/mobile-device')
def mobile_device_detected_page():
    return render_template('mobile-device.html')
```

```

@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        image = request.files['image']
        filename = image.filename
        file_path = os.path.join('static/uploads', filename)
        image.save(file_path)
        print(file_path)
        pred = prediction(file_path)
        title = disease_info['disease_name'][pred]
        description = disease_info['description'][pred]
        prevent = disease_info['Possible Steps'][pred]
        image_url = disease_info['image_url'][pred]
        supplement_name = supplement_info['supplement name'][pred]
        supplement_image_url = supplement_info['supplement image'][pred]
        supplement_buy_link = supplement_info['buy link'][pred]
        return render_template('submit.html' , title = title , desc =
description , prevent = prevent ,
                           image_url = image_url , pred = pred , sname
= supplement_name , simage = supplement_image_url , buy_link =
supplement_buy_link)

@app.route('/market', methods=['GET', 'POST'])
def market():
    return render_template('market.html' , supplement_image =
list(supplement_info['supplement image']) ,
                           supplement_name = list(supplement_info['supplement
name']) , disease = list(disease_info['disease_name']) , buy =
list(supplement_info['buy link']))

if __name__ == '__main__':
    app.run(debug=True)

```

B. Screenshots

Files

```

.. 
Plant_leave_diseases_dataset_wi...
sample_data
test_images
plant_disease_model_1.pt

```

```

nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1),
nn.ReLU(),
nn.BatchNorm2d(256),
nn.Conv2d(in_channels=256, out_channels=256, kernel_size=3, padding=1),
nn.ReLU(),
nn.BatchNorm2d(256),
nn.MaxPool2d(2),

self.dense_layers = nn.Sequential(
    nn.Dropout(0.4),
    nn.Linear(50176, 1024),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(1024, K),
)

def forward(self, x):
    out = self.conv_layers(x)

    # Flatten
    out = out.view(-1, 50176)

    # Fully connected
    out = self.dense_layers(out)

    return out

```

Disk 71.66 GB available

Files

```

.. 
Plant_leave_diseases_dataset_wi...
sample_data
test_images
plant_disease_model_1.pt

```

```

import torch
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")

# Check which GPU you got
if torch.cuda.is_available():
    print(f"GPU Name: {torch.cuda.get_device_name(0)}")
    print(f"GPU Memory: {(torch.cuda.get_device_properties(0).total_memory / 1e9):.2f} GB")

Using device: cuda
GPU Name: Tesla T4
GPU Memory: 15.83 GB

#device = "cpu"

model = CNN(targets_size)
model.to(device)

CNN(
    conv_layers: Sequential(
        (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU()
        (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (4): ReLU()
    )
)

```

Disk 71.66 GB available

Files

```

.. 
Plant_leave_diseases_dataset_wi...
sample_data
test_images
plant_disease_model_1.pt

```

Import Dependencies

```

!unzip -q /Plant_leaf_diseases_dataset_with_augmentation.zip
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

```

Import Dataset

Dataset Link (Plant Vliage Dataset):

<https://data.mendeley.com/datasets/twbtjsjv/1>

Disk 71.66 GB available

Files

```
[97] [✓ 0s] transform = transforms.Compose([
    [transforms.Resize(255), transforms.CenterCrop(224), transforms.ToTensor()]
])

[100] [✓ 0s] dataset = datasets.ImageFolder(
    "/content/plant_leave_diseases_dataset_with_augmentation", transform=transform
)

[101] [✓ 0s] dataset
Dataset ImageFolder
  Number of datapoints: 61486
  Root location: /content/Plant_leave_diseases_dataset_with_augmentation
  StandardTransform
  Transform: Compose(
    Resize(size=255, interpolation=bilinear, max_size=None, antialias=True)
    CenterCrop(size=(224, 224))
    ToTensor()
  )

[102] [✓ 0s] from google.colab import drive
drive.mount('/content/drive')

indices = list(range(len(dataset)))
```

Disk 71.66 GB available

Files

```
[103] [✓ 0s] indices = list(range(len(dataset)))

[104] [✓ 0s] split = int(np.floor(0.85 * len(dataset))) # train_size

[105] [✓ 0s] validation = int(np.floor(0.10 * split)) # validation

[106] [✓ 0s] print(0, validation, split, len(dataset))
0 36584 52263 61486

print(f"length of train size :{validation}")
print(f"length of validation size :{split - validation}")
print(f"length of test size :{len(dataset)-validation}")

[107] [✓ 0s] length of train size :36584
length of validation size :15679
length of test size :24902

[108] [✓ 0s] np.random.shuffle(indices)

Split into Train and Test
```

Disk 71.66 GB available

Files

```
[108] [✓ 0s] train_indices, validation_indices, test_indices = (
    indices[:validation],
    indices[validation:split],
    indices[split:]
)

[109] [✓ 0s] train_sampler = SubsetRandomSampler(train_indices)
validation_sampler = SubsetRandomSampler(validation_indices)
test_sampler = SubsetRandomSampler(test_indices)

[110] [✓ 0s] targets_size = len(dataset.class_to_idx)

Split into Train and Test
```

Model

Convolution Arithmetic Equation : $(W - F + 2P) / S + 1$

W = Input Size
F = Filter Size
P = Padding Size
S = Stride

Disk 71.66 GB available