UNIVERSITÀ DEGLI
STUDI DI PADOVA

A Comparative Study of Pre-trained Deep Learning Models for Grapevine Leaf Disease Detection

Project presentation
(Vision and cognitive systems 2023-2024)

February 2024

Presented by

Alessandro Canel
Lorenzo D'Antoni
Fateme Baghaei Saryazdi

Abstract

- **Grape leaf disease** classification with **deep-learning approach**
- 4 classes: **Black Rot**, **ESCA**, **LeafBlight**, **Healthy**
- Enhancing a **Kaggle** project by using the **TensorFlow** framework and **adapted** it to the **PyTorch** library
- **evaluated** their **performance**

Introduction

Motivation:

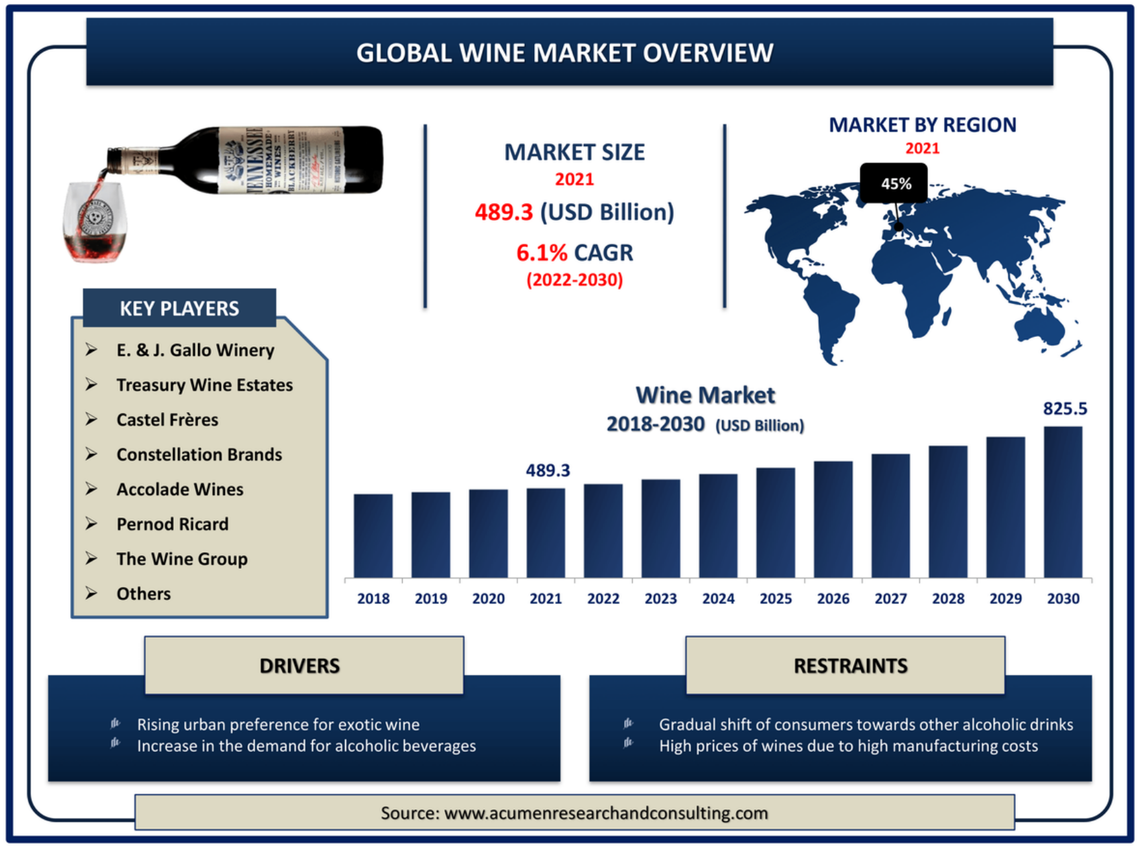
- **Grapes** are one of the **most cultivated** fruits, and they have **many diseases**
- **Early detection** is crucial for the wine industry.

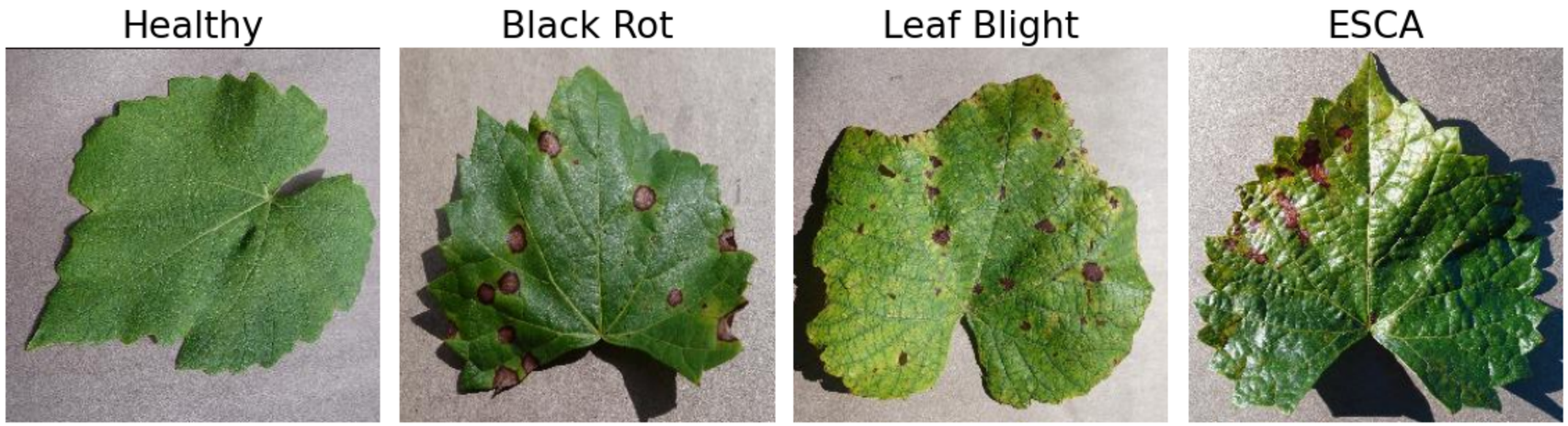
Challenge:

- **Human** inspection is **time-consuming**.

Solution:

- **Computer vision** and **machine learning** approaches can automate and **improve** the process





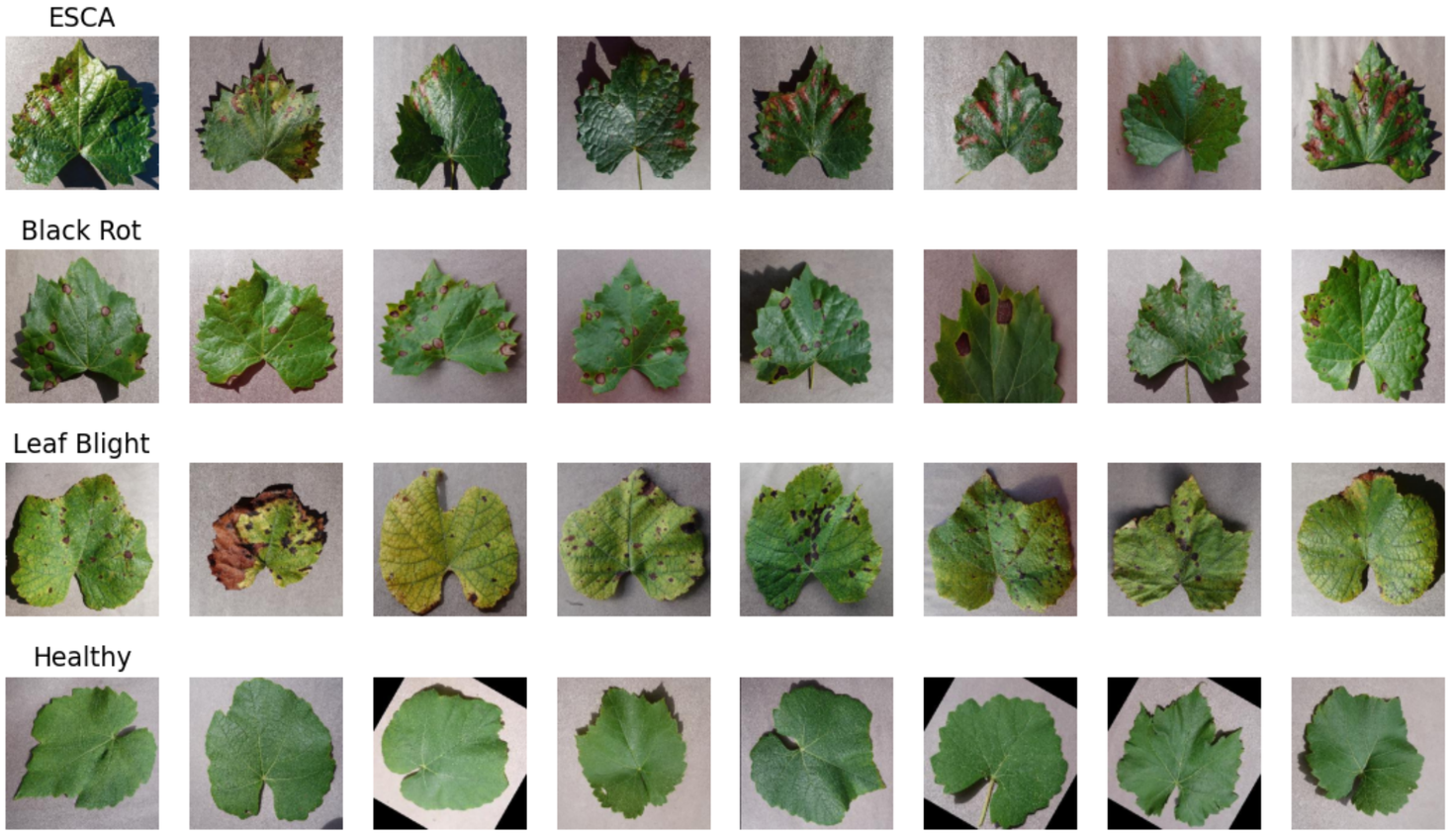
Grape leaf disease classification using **deep learning**, identifying whether the leaf is **infected** by **Black Rot**, **ESCA**, **Leaf Blight** or is **healthy.**

Problems:

- **Scarcity** of annotated **datasets**
- Need for **real-time detection**
- **Diversity** and **Complexity** of **disese spots**

Results

Achieved high accuracy on the test set, but **struggled** to **distinguish** between **Black Rot** and **ESCA**, which are similar diseases



Related works

Before 2016:

- **K-means** clustering:
  - find diseased regions (segmentation)
- extract **color** and **texture** features
- **PCA**
  - reduce the dimensionality of the plant image features and retain the most important ones
- **SVM**:
  - detect the type of leaf disease

Problems

- manually engineered features:
  - **Time-consuming**
  - **Incomplete**
  - No **generalization**
- **extensive preprocessing** to deal with image variations and **noise**

After 2016: CNN

- **automatically** learn **discriminative** features from **images**
- **avoiding complicated** preprocessing
- **preserving** relevant **information**
- capture **hierarchical** spatial **features**
- extracting **complex patterns** and structures
- computationally **efficient** (parameter sharing)
- **translation invariant**

Two Main Papers

Faster R-CNN and **ResNet**:

- enhanced the base model with **state-of-the-art** deep-learning modules
- **improving** the multiscale diseased **spot extraction** and **detection speed**
- **trained** and tested the **model** with **simple** and **complex backgrounds**
- **augmented** the **dataset** using **digital image processing** (to prevent the overfitting problem)

YOLOXS model:

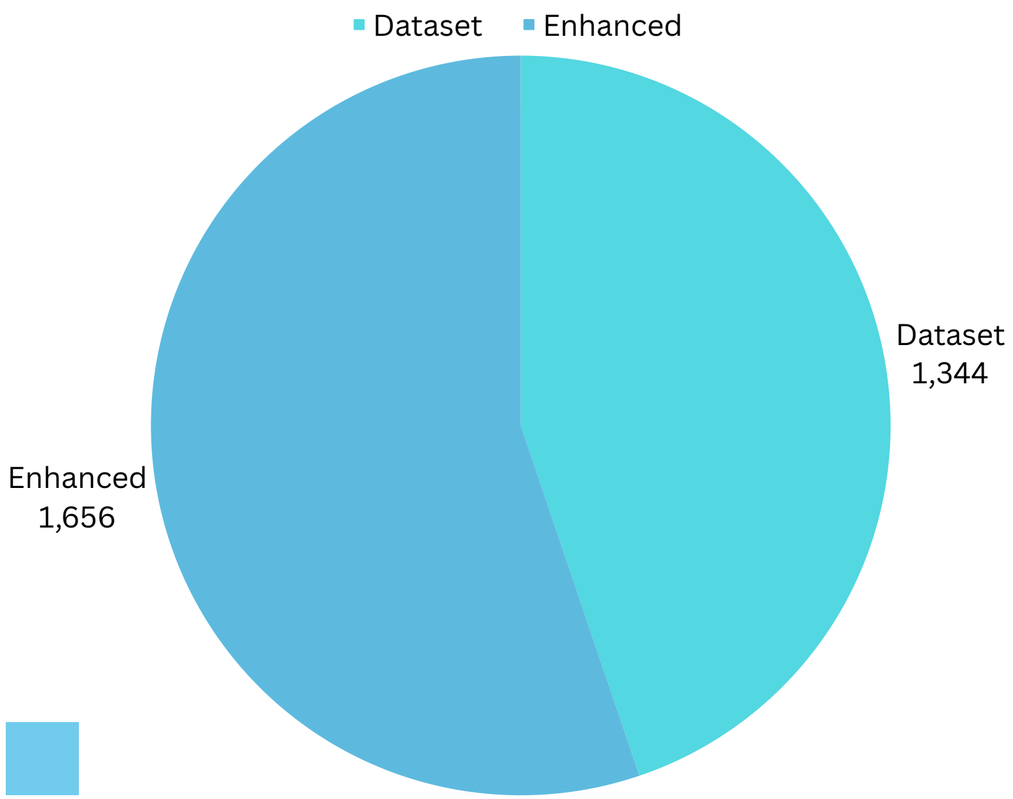
- **collected** and used a dataset of 15 leaf diseases in natural environments
- **improved** the **generalization** and **speed** of the existing models using an improved YOLOXS model

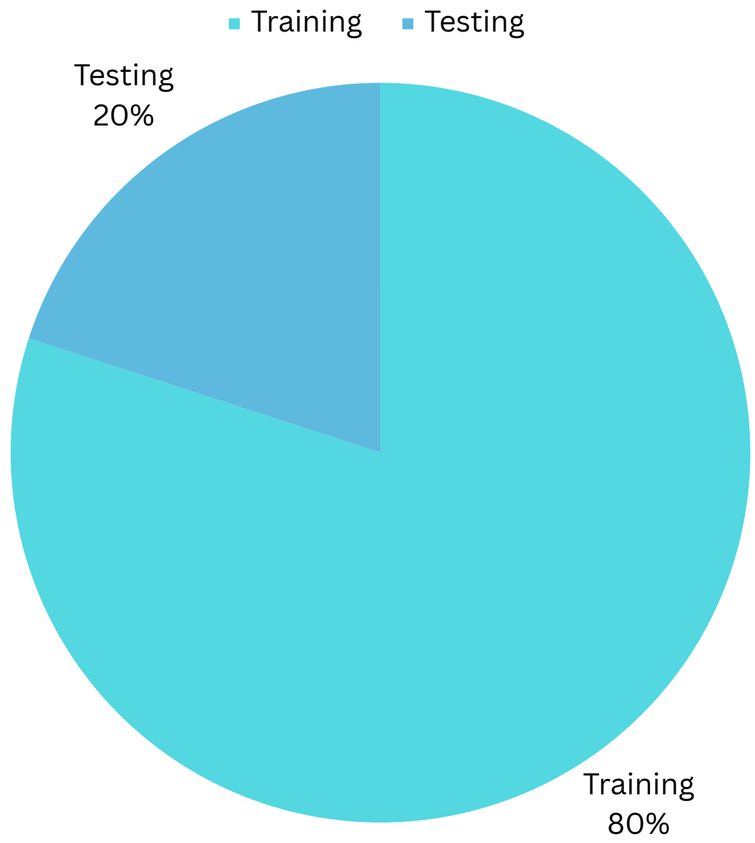
Back to Agenda

Dataset

Enhancing

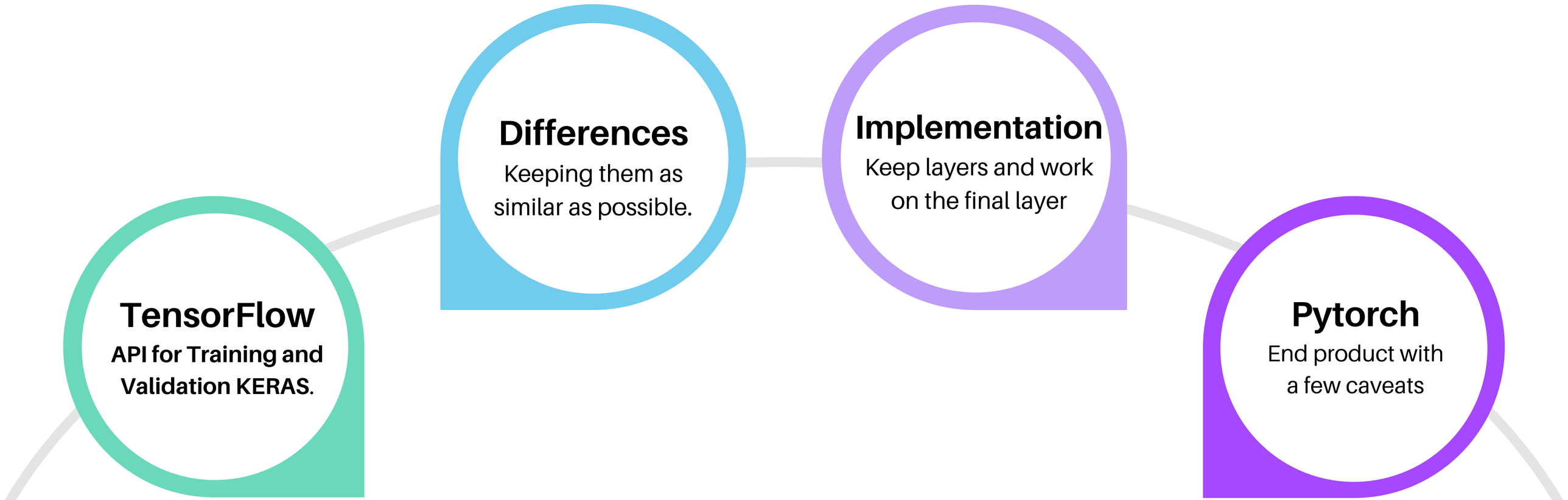
Understanding our Kaggle Dataset



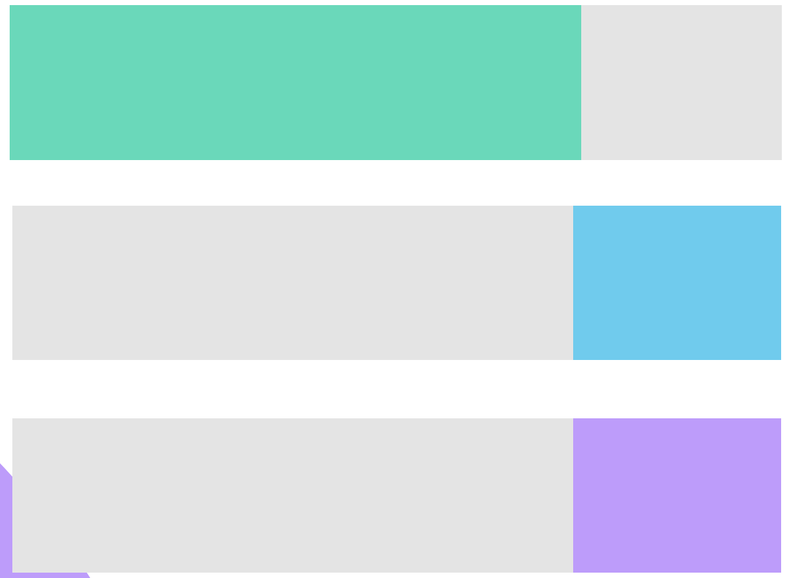


From TensorFlow to Pytorch

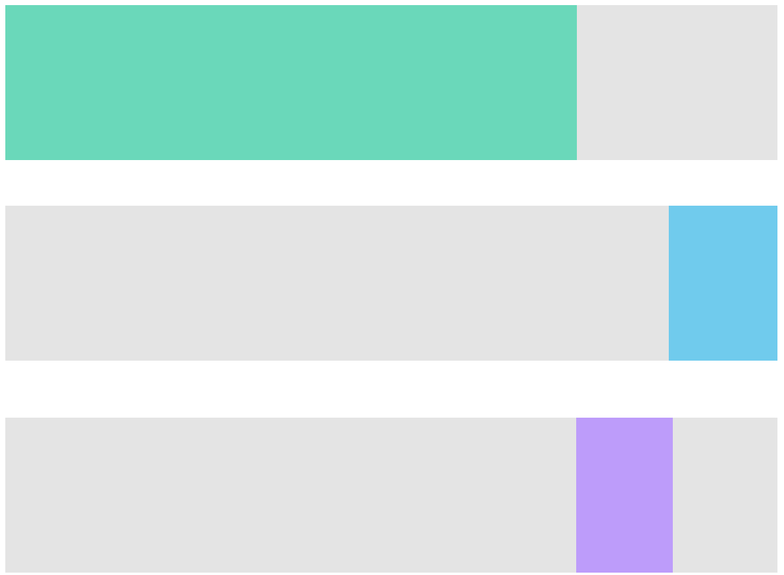
TensowFlow to Pytorch



Testing and Validation



Paper’s solution



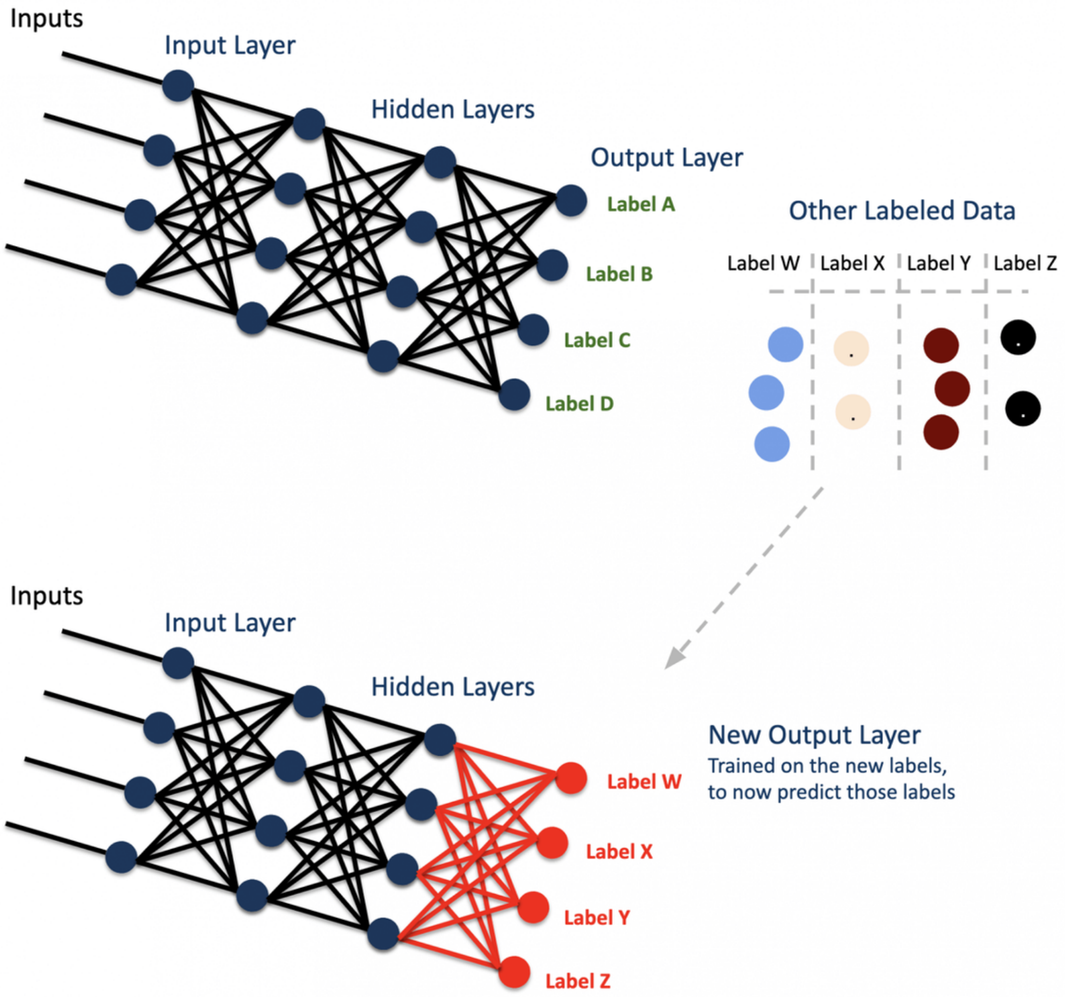
Our solution

Train

Validation

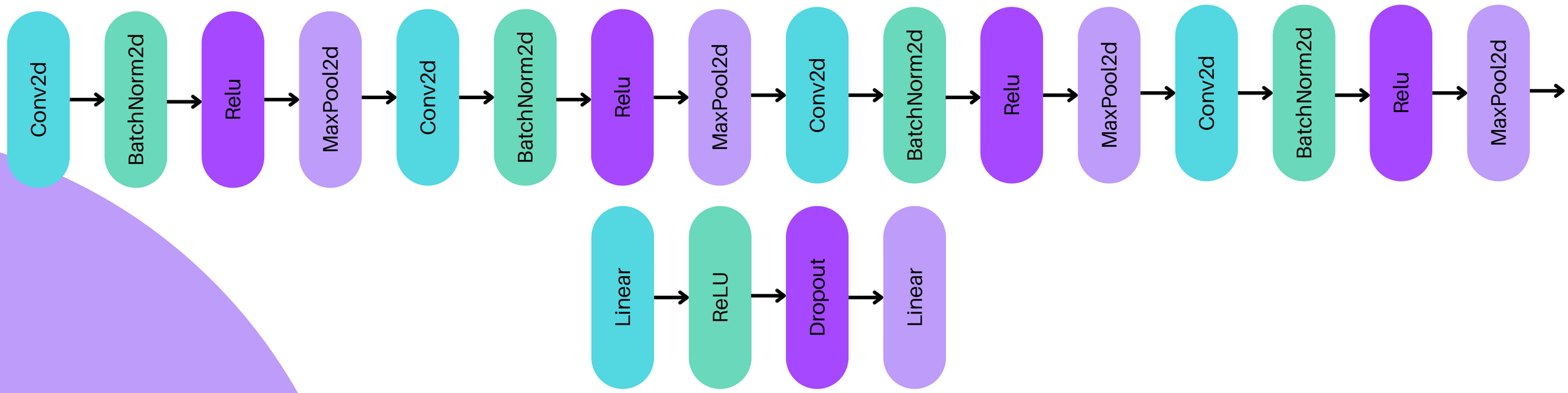
Test

Transfer Learning

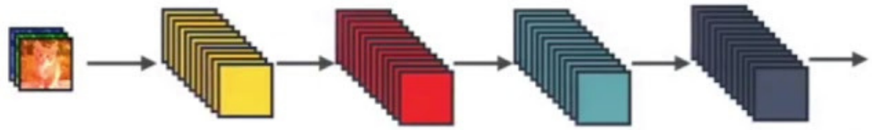


CNN	DenseNet	EfficentNet	MobileNet	ResNet	VGG	Ensemble model
Baseline simple model	Layer-to-layer	Scaling along with parameters	Separable convolutions	Residual blocks	Deep CNN architecture	
	Transition Layers	Adapts to resource limits	Split convolution into depthwise and pointwise	Skip connections	Conv layers followed bymx pooling	
	Control complexity and cost	Might overfit	Less intensive but good performance	Learns residual mappings	Reduces spatial dimensions	
	Many variations	Many variations	MobileNetV2, MobileNetV3, and Mo bileNetV3 Small	ResNet-18, ResNet-34, ResNet-50, ResNet 101 and ResNet-152	Authors used VGG16	
VERSION CHOSEN	**Densenet-121**	**EfficentNet-B7**	**MobileNetV2**	**ResNet-50**	**VGG11**	Best three models

Our CNN

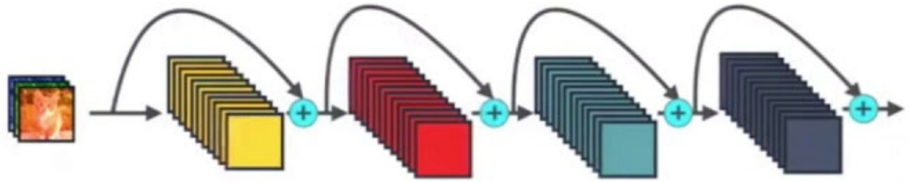

Conv2d
BatchNorm2d
Relu
MaxPool2d
Conv2d
BatchNorm2d
Relu
MaxPool2d
Conv2d
BatchNorm2d
Relu
MaxPool2d
Conv2d
BatchNorm2d
Relu
MaxPool2d
Linear
ReLU
Dropout
Linear

Standard Connectivity



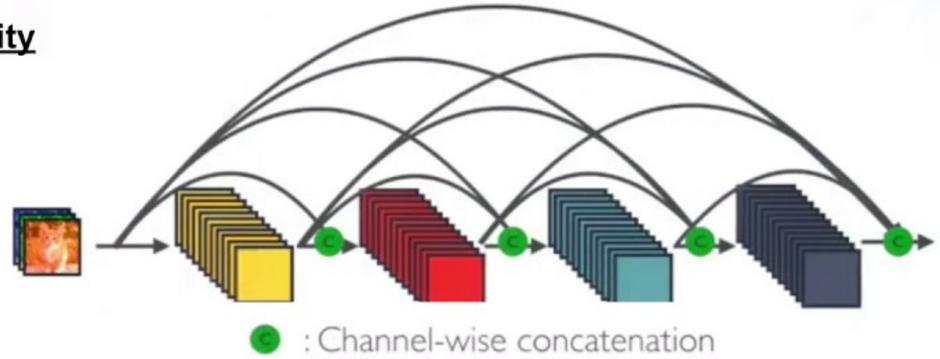
Successive convolutions

Resnet Connectivity



Element-wise feature summation

DenseNet Connectivity



Feature concatenation

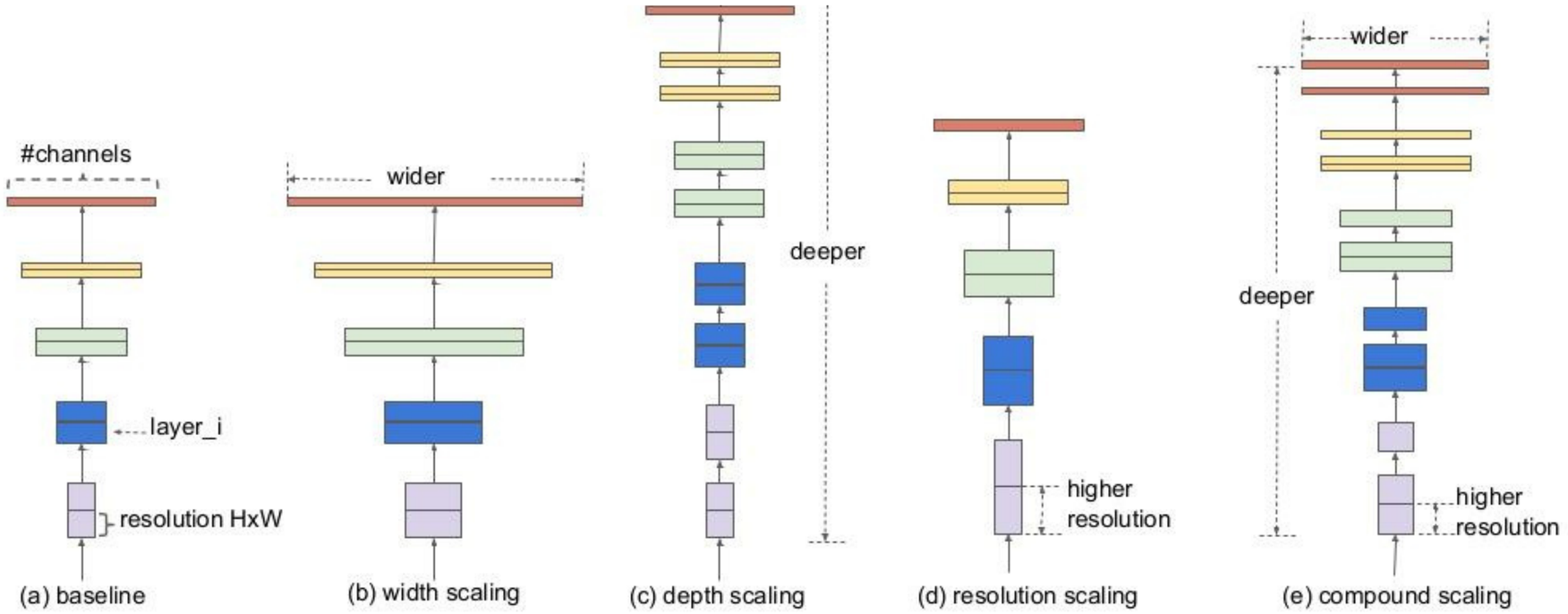
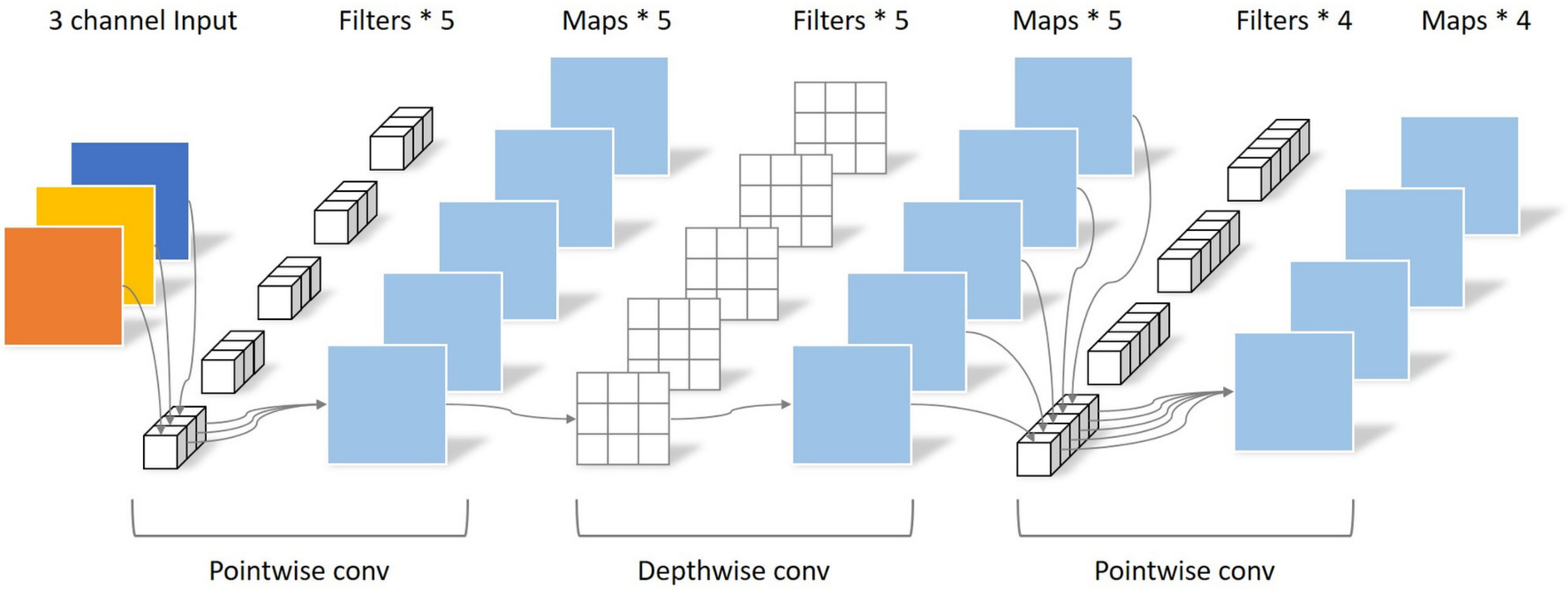
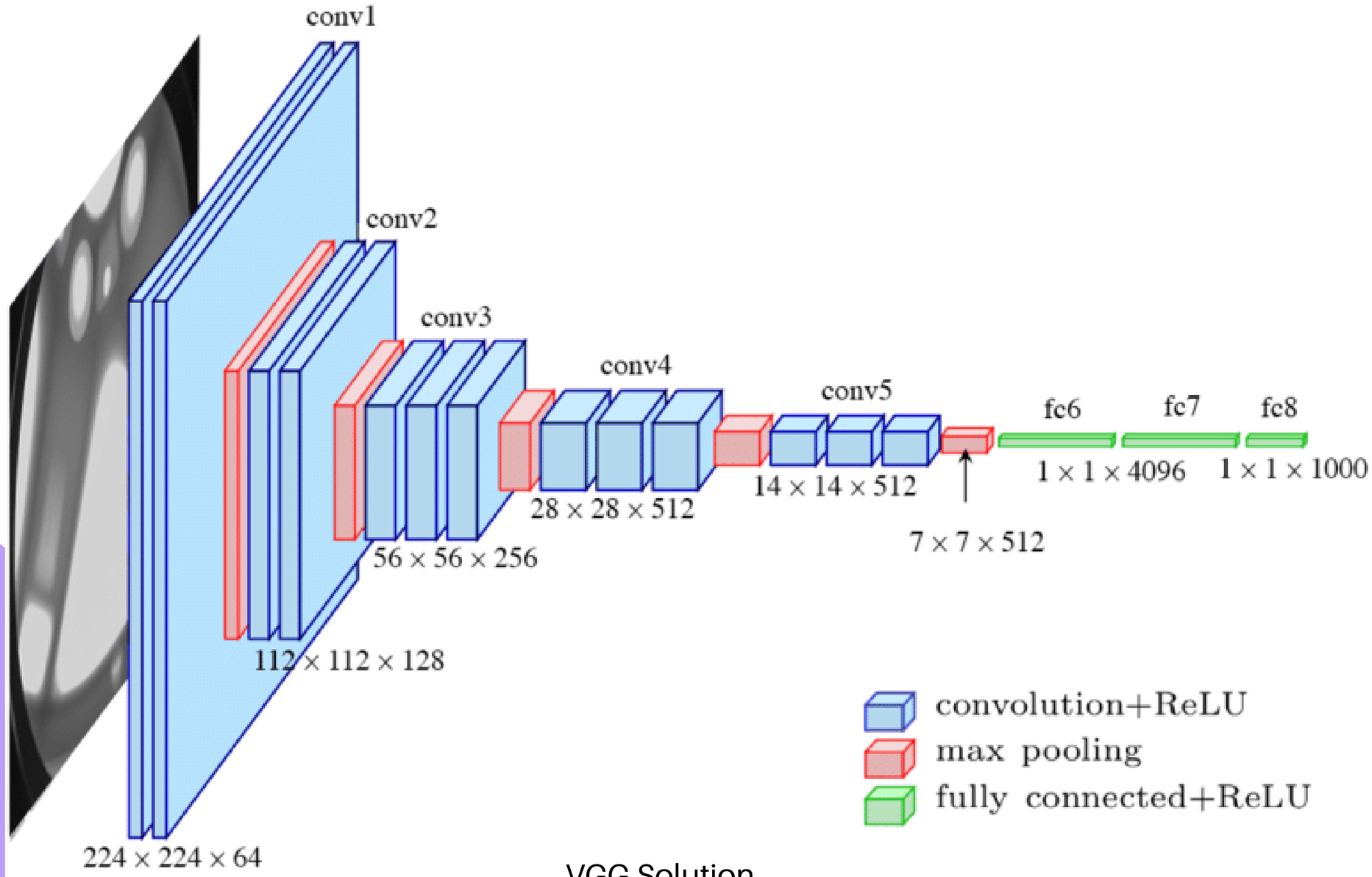


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

EfficentNet Solution



MobileNet solution



VGG Solution

Experiments

Experiments

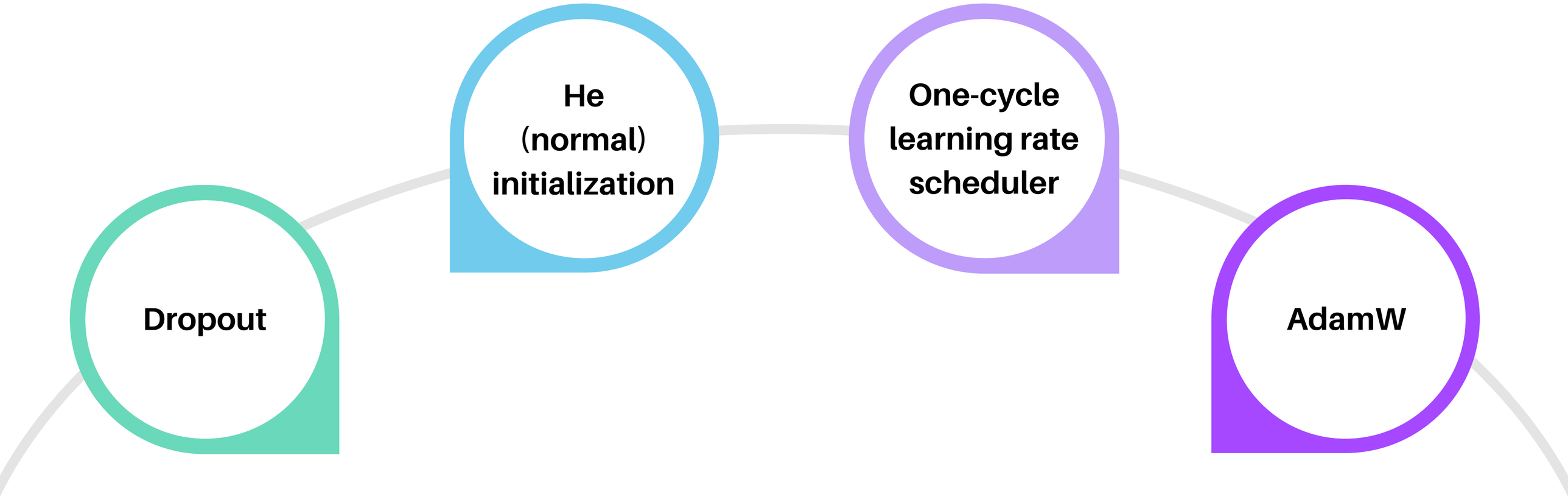
Parameters

- training epochs = 25
- batch size = 64
- patience (Early Stopping) = 8
- Adam = default values
- Cross-Entropy loss function

Evaluation metrics

- accuracy score
- confusion matrix
- precision, recall, f1-score
- visualization of wrong predictions

Optimization techniques



Baseline CNN

Model	Accuracy
Our VGG11	0.9979
Our VGG16	0.9948
ResNet50	0.9927
Our Baseline CNN	**0.9927**
Our EfficientNetB1	0.9927
Our ResNet50	0.9927
Baseline CNN	**0.9917**
VGG16	0.9906
Our ResNet18	0.9896
MobileNetV2	0.9833
DenseNet121	0.9816
Our MobileNetV2	0.9812
Our DenseNet121	0.9795
EfficientNetB7	0.9656

Model	Number of parameters
Our VGG16	134,285,380
VGG16	134,276,932
Our VGG11	128,788,228
EfficientNetB7	63,797,204
ResNet50	23,516,228
Our ResNet50	23,516,228
Our ResNet18	11,178,564
DenseNet121	6,957,956
Our DenseNet121	6,957,956
Our EfficientNetB1	6,518,308
Our Baseline CNN	**6,485,956**
MobileNetV2	2,228,996
Our MobileNetV2	2,228,996
Baseline CNN	**364,580**

Model	Wrong Predictions
Our VGG11	2
Our VGG16	5
ResNet50	7
Our Baseline CNN	**7**
Our ResNet50	7
Baseline CNN	**8**
DenseNet121	8
Our DenseNet121	8
VGG16	9
Our EfficientNetB1	10
Our ResNet18	10
MobileNetV2	16
Our MobileNetV2	16
EfficientNetB7	37

DenseNet

Model	Accuracy
Our VGG11	0.9979
Our VGG16	0.9948
ResNet50	0.9927
Our Baseline CNN	0.9927
Our EfficientNetB1	0.9927
Our ResNet50	0.9927
Baseline CNN	0.9917
VGG16	0.9906
Our ResNet18	0.9896
MobileNetV2	0.9833
DenseNet121	**0.9816**
Our MobileNetV2	0.9812
Our DenseNet121	**0.9795**
EfficientNetB7	0.9656

Model	Number of parameters
Our VGG16	134,285,380
VGG16	134,276,932
Our VGG11	128,788,228
EfficientNetB7	63,797,204
ResNet50	23,516,228
Our ResNet50	23,516,228
Our ResNet18	11,178,564
DenseNet121	**6,957,956**
Our DenseNet121	**6,957,956**
Our EfficientNetB1	6,518,308
Our Baseline CNN	6,485,956
MobileNetV2	2,228,996
Our MobileNetV2	2,228,996
Baseline CNN	364,580

Model	Wrong Predictions
Our VGG11	2
Our VGG16	5
ResNet50	7
Our Baseline CNN	7
Our ResNet50	7
Baseline CNN	8
DenseNet121	**8**
Our DenseNet121	**8**
VGG16	9
Our EfficientNetB1	10
Our ResNet18	10
MobileNetV2	16
Our MobileNetV2	16
EfficientNetB7	37

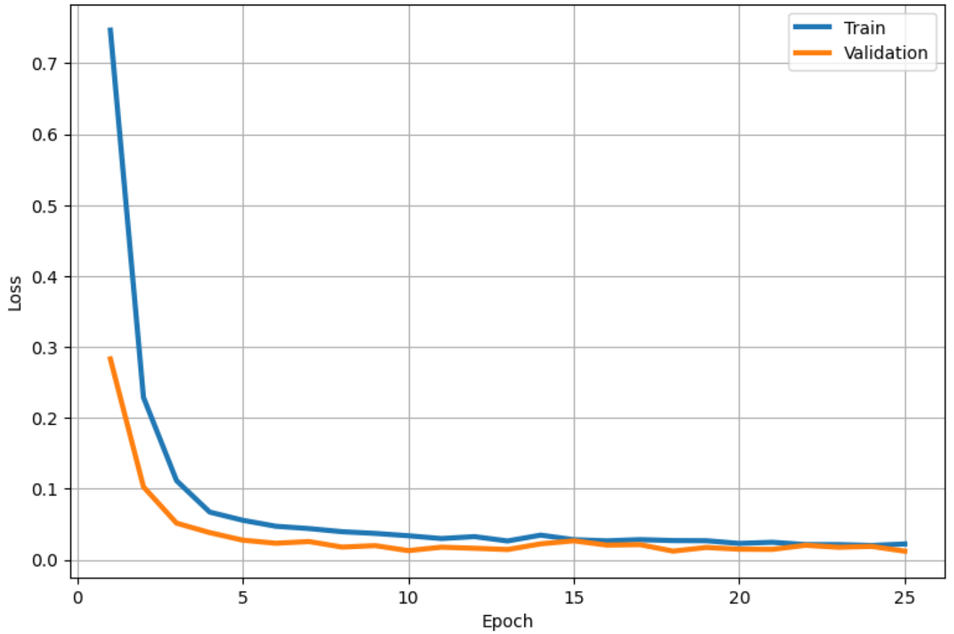
EfficientNet

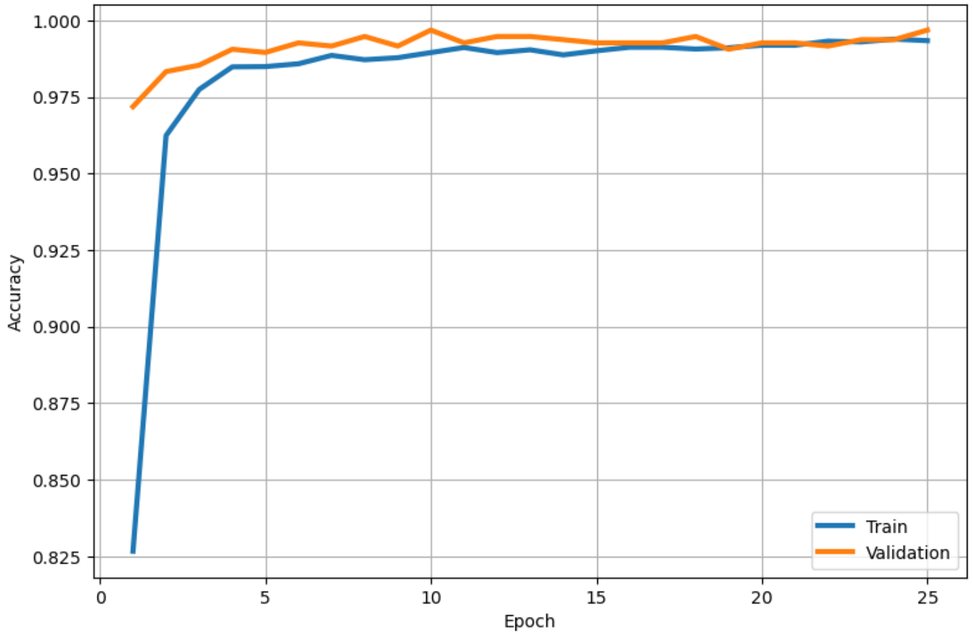
Model	Accuracy
Our VGG11	0.9979
Our VGG16	0.9948
ResNet50	0.9927
Our Baseline CNN	0.9927
Our EfficientNetB1	**0.9927**
Our ResNet50	0.9927
Baseline CNN	0.9917
VGG16	0.9906
Our ResNet18	0.9896
MobileNetV2	0.9833
DenseNet121	0.9816
Our MobileNetV2	0.9812
Our DenseNet121	0.9795
EfficientNetB7	**0.9656**

Model	Number of parameters
Our VGG16	134,285,380
VGG16	134,276,932
Our VGG11	128,788,228
EfficientNetB7	**63,797,204**
ResNet50	23,516,228
Our ResNet50	23,516,228
Our ResNet18	11,178,564
DenseNet121	6,957,956
Our DenseNet121	6,957,956
Our EfficientNetB1	**6,518,308**
Our Baseline CNN	6,485,956
MobileNetV2	2,228,996
Our MobileNetV2	2,228,996
Baseline CNN	364,580

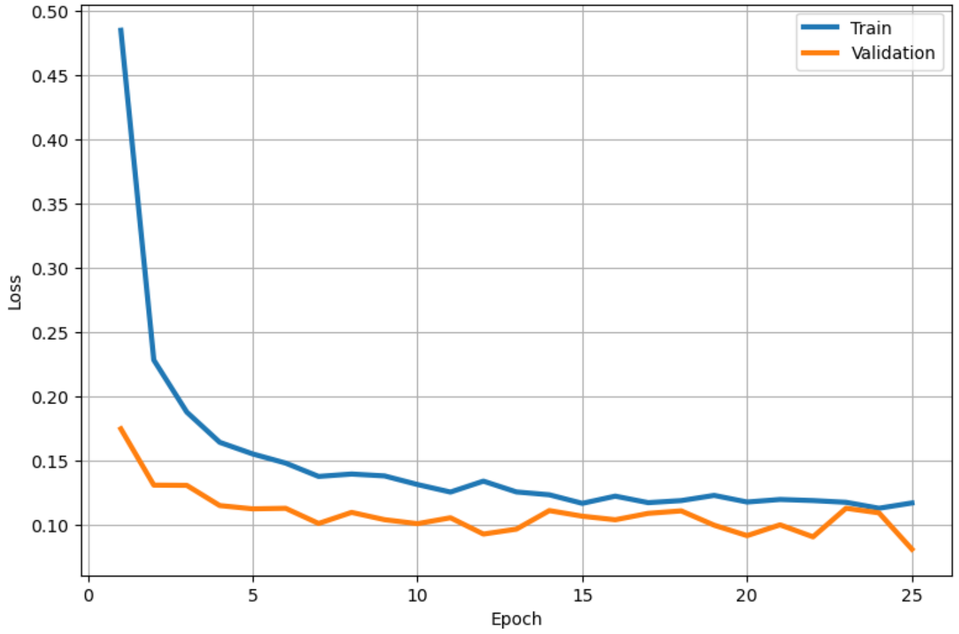
Model	Wrong Predictions
Our VGG11	2
Our VGG16	5
ResNet50	7
Our Baseline CNN	7
Our ResNet50	7
Baseline CNN	8
DenseNet121	8
Our DenseNet121	8
VGG16	9
Our EfficientNetB1	**10**
Our ResNet18	10
MobileNetV2	16
Our MobileNetV2	16
EfficientNetB7	**37**

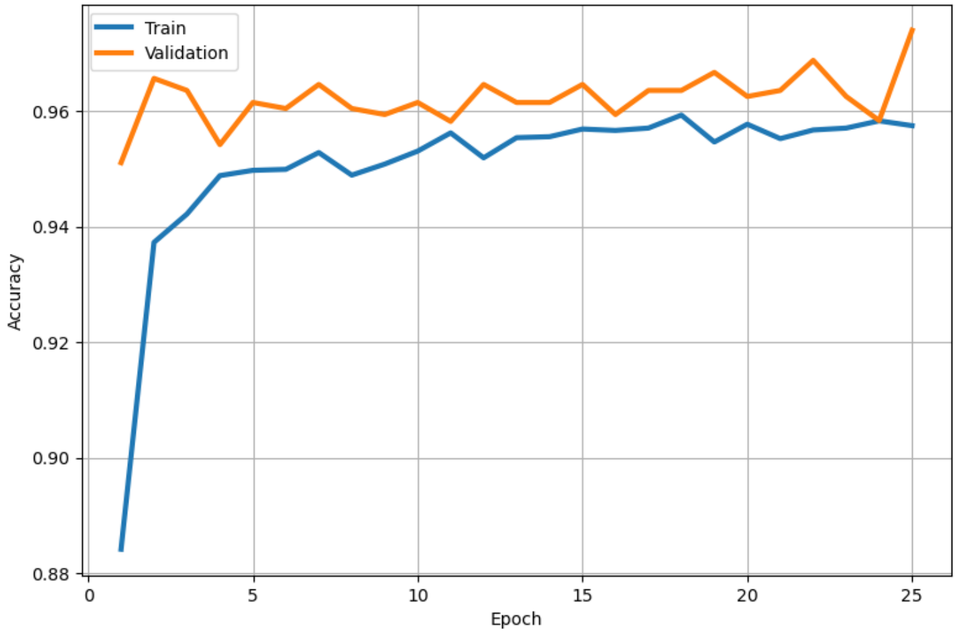
Our (EfficientNetB1)





Original (EfficientNetB7)





MobileNet

Model	Accuracy
Our VGG11	0.9979
Our VGG16	0.9948
ResNet50	0.9927
Our Baseline CNN	0.9927
Our EfficientNetB1	0.9927
Our ResNet50	0.9927
Baseline CNN	0.9917
VGG16	0.9906
Our ResNet18	0.9896
MobileNetV2	**0.9833**
DenseNet121	0.9816
Our MobileNetV2	**0.9812**
Our DenseNet121	0.9795
EfficientNetB7	0.9656

Model	Number of parameters
Our VGG16	134,285,380
VGG16	134,276,932
Our VGG11	128,788,228
EfficientNetB7	63,797,204
ResNet50	23,516,228
Our ResNet50	23,516,228
Our ResNet18	11,178,564
DenseNet121	6,957,956
Our DenseNet121	6,957,956
Our EfficientNetB1	6,518,308
Our Baseline CNN	6,485,956
MobileNetV2	**2,228,996**
Our MobileNetV2	**2,228,996**
Baseline CNN	364,580

Model	Wrong Predictions
Our VGG11	2
Our VGG16	5
ResNet50	7
Our Baseline CNN	7
Our ResNet50	7
Baseline CNN	8
DenseNet121	8
Our DenseNet121	8
VGG16	9
Our EfficientNetB1	10
Our ResNet18	10
MobileNetV2	**16**
Our MobileNetV2	**16**
EfficientNetB7	37

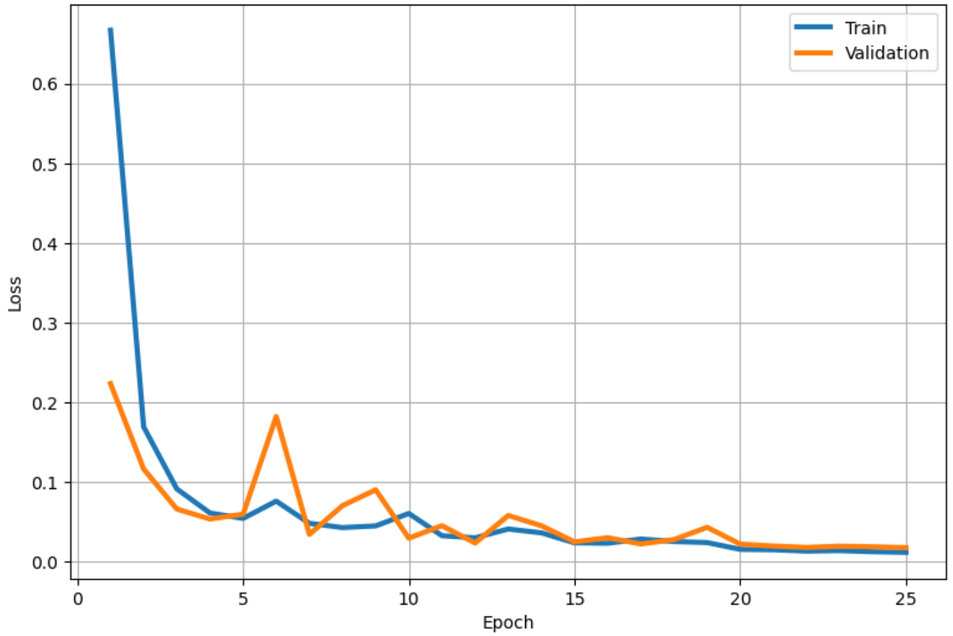
ResNet

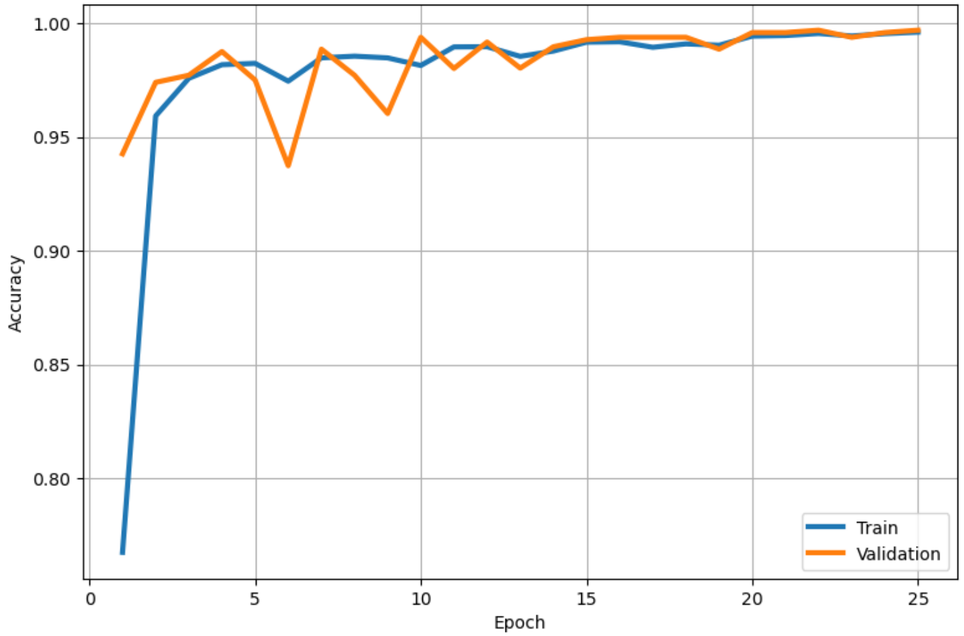
Model	Accuracy
Our VGG11	0.9979
Our VGG16	0.9948
ResNet50	**0.9927**
Our Baseline CNN	0.9927
Our EfficientNetB1	0.9927
Our ResNet50	**0.9927**
Baseline CNN	0.9917
VGG16	0.9906
Our ResNet18	**0.9896**
MobileNetV2	0.9833
DenseNet121	0.9816
Our MobileNetV2	0.9812
Our DenseNet121	0.9795
EfficientNetB7	0.9656

Model	Number of parameters
Our VGG16	134,285,380
VGG16	134,276,932
Our VGG11	128,788,228
EfficientNetB7	63,797,204
ResNet50	**23,516,228**
Our ResNet50	**23,516,228**
Our ResNet18	**11,178,564**
DenseNet121	6,957,956
Our DenseNet121	6,957,956
Our EfficientNetB1	6,518,308
Our Baseline CNN	6,485,956
MobileNetV2	2,228,996
Our MobileNetV2	2,228,996
Baseline CNN	364,580

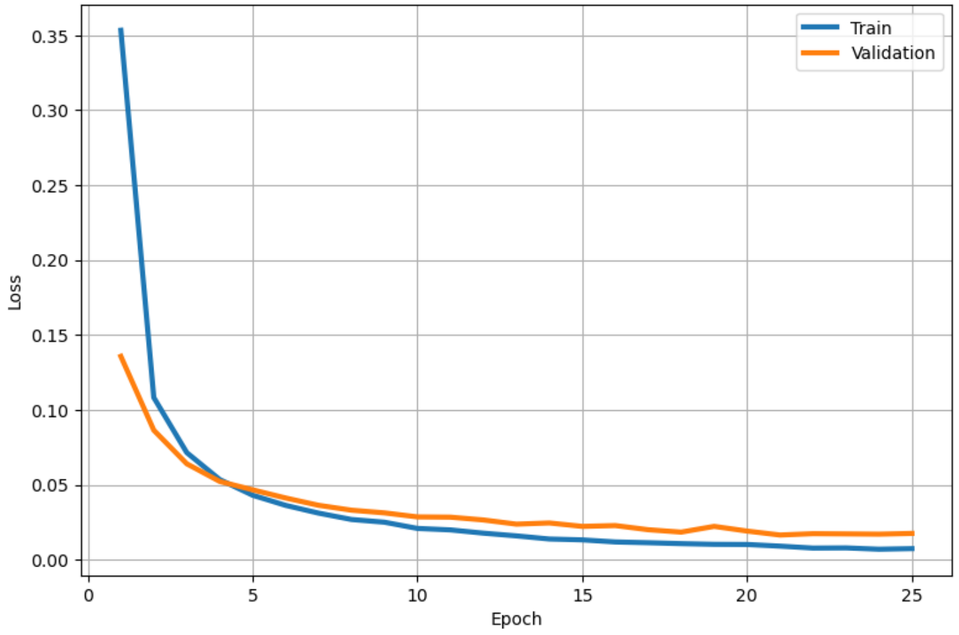
Model	Wrong Predictions
Our VGG11	2
Our VGG16	5
ResNet50	**7**
Our Baseline CNN	7
Our ResNet50	**7**
Baseline CNN	8
DenseNet121	8
Our DenseNet121	8
VGG16	9
Our EfficientNetB1	10
Our ResNet18	**10**
MobileNetV2	16
Our MobileNetV2	16
EfficientNetB7	37

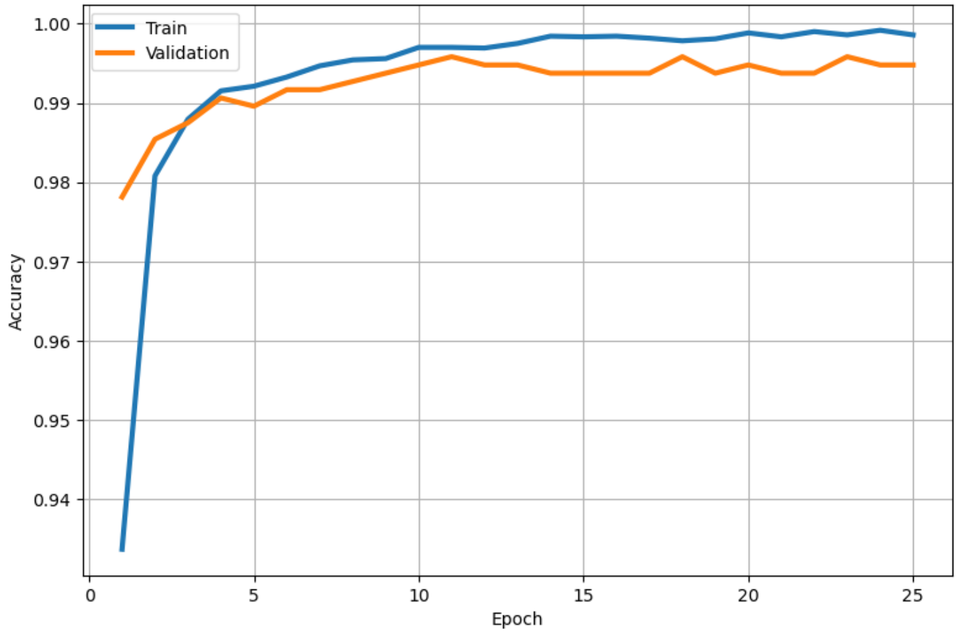
Our (ResNet18)





Original (ResNet50)





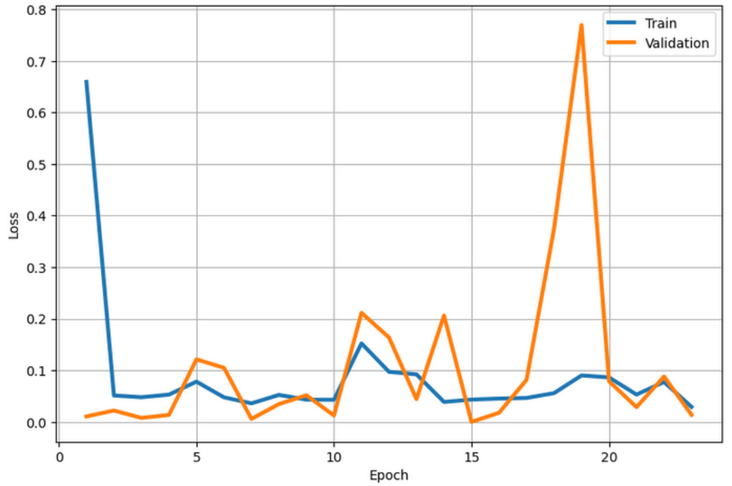
VGG

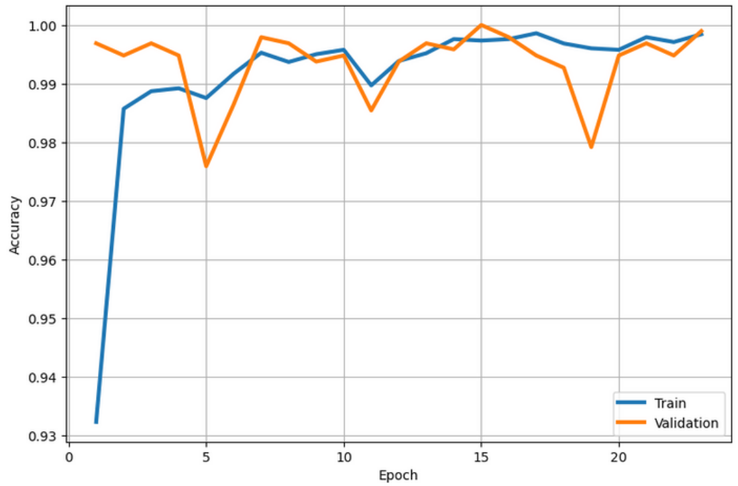
Model	Accuracy
Our VGG11	**0.9979**
Our VGG16	**0.9948**
ResNet50	0.9927
Our Baseline CNN	0.9927
Our EfficientNetB1	0.9927
Our ResNet50	0.9927
Baseline CNN	0.9917
VGG16	**0.9906**
Our ResNet18	0.9896
MobileNetV2	0.9833
DenseNet121	0.9816
Our MobileNetV2	0.9812
Our DenseNet121	0.9795
EfficientNetB7	0.9656

Model	Number of parameters
Our VGG16	**134,285,380**
VGG16	**134,276,932**
Our VGG11	**128,788,228**
EfficientNetB7	63,797,204
ResNet50	23,516,228
Our ResNet50	23,516,228
Our ResNet18	11,178,564
DenseNet121	6,957,956
Our DenseNet121	6,957,956
Our EfficientNetB1	6,518,308
Our Baseline CNN	6,485,956
MobileNetV2	2,228,996
Our MobileNetV2	2,228,996
Baseline CNN	364,580

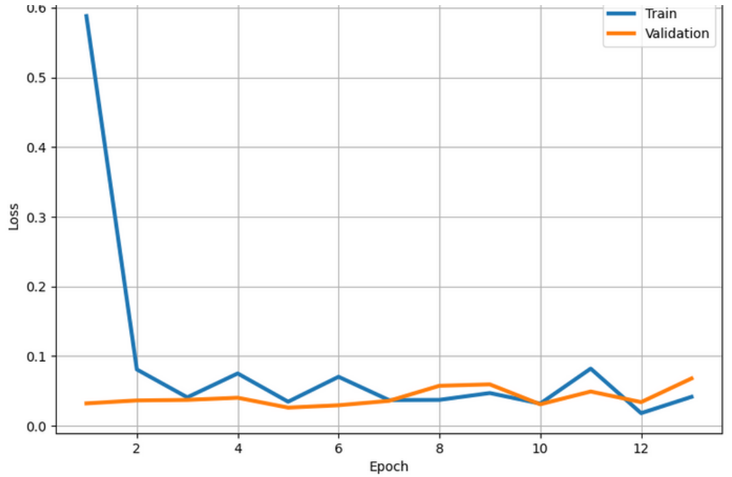
Model	Wrong Predictions
Our VGG11	**2**
Our VGG16	**5**
ResNet50	7
Our Baseline CNN	7
Our ResNet50	7
Baseline CNN	8
DenseNet121	8
Our DenseNet121	8
VGG16	**9**
Our EfficientNetB1	10
Our ResNet18	10
MobileNetV2	16
Our MobileNetV2	16
EfficientNetB7	37

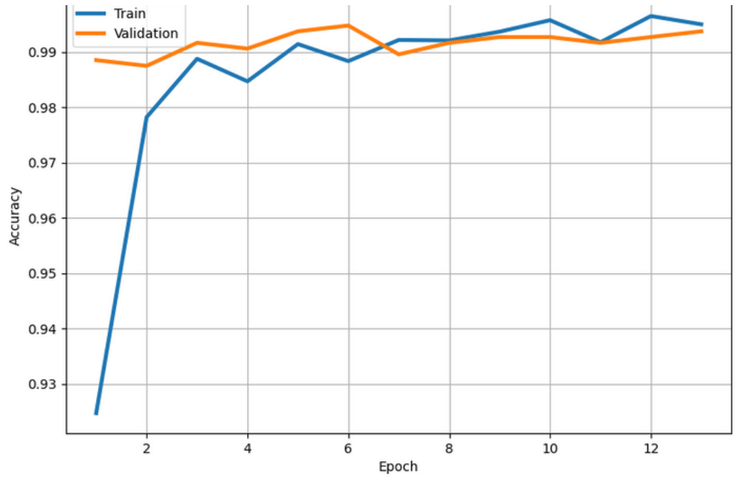
Our
(VGG11)



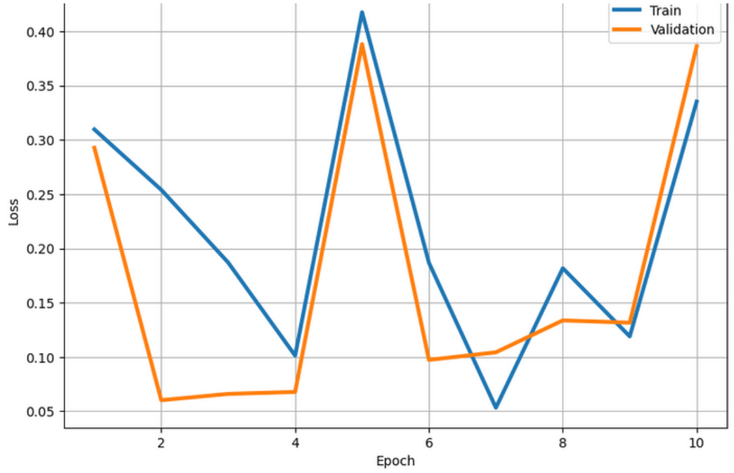


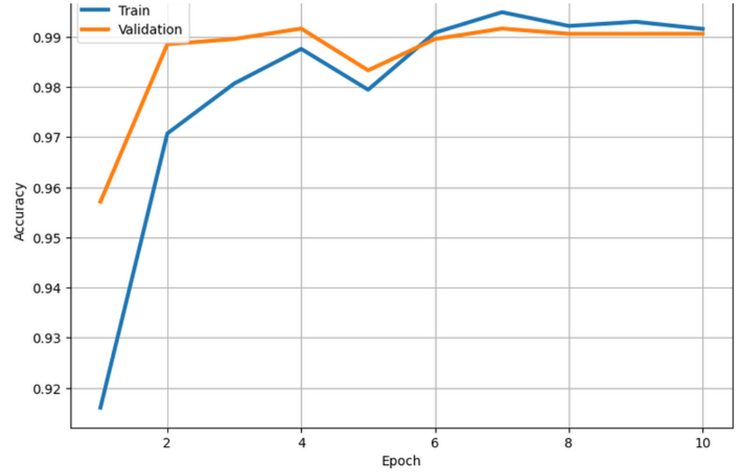
Our
(VGG16BN)





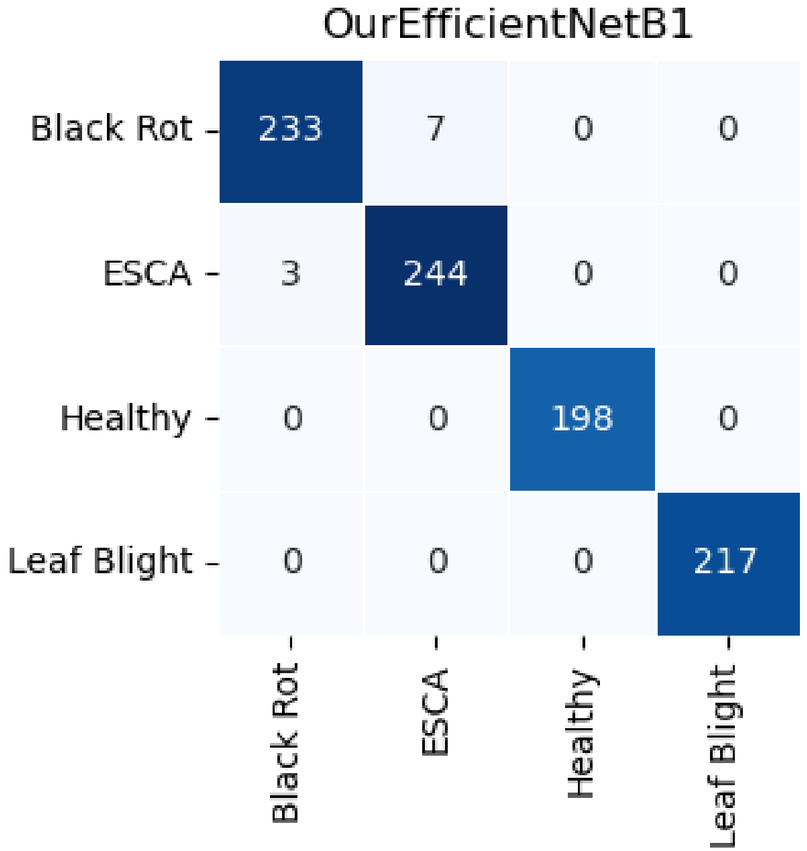
Original
(VGG16)

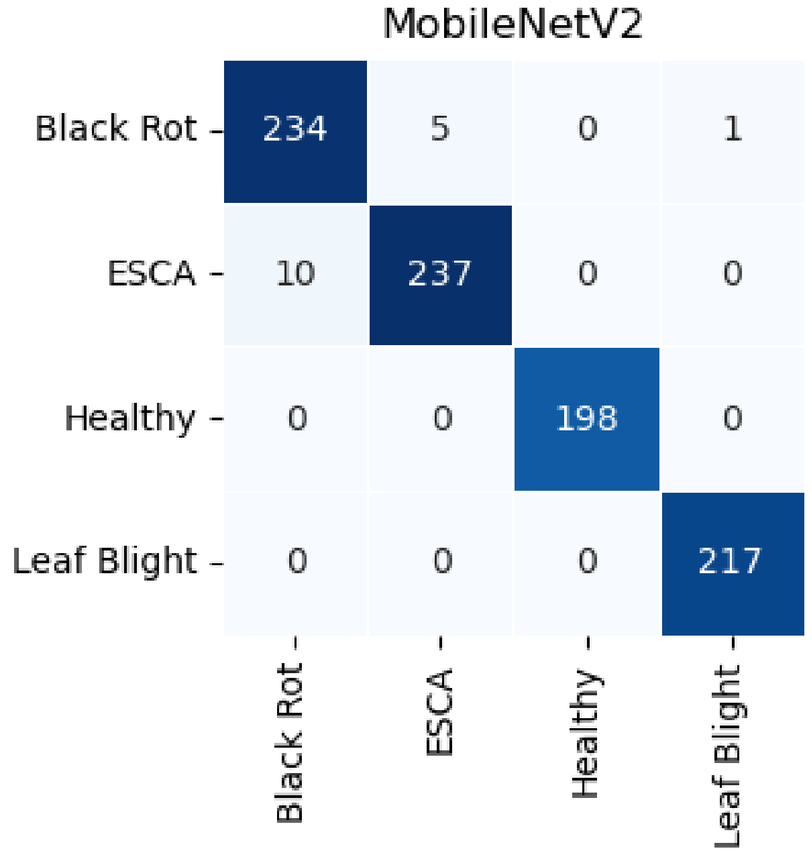




Summing up...

Model	Accuracy
Our VGG11	0.9979
Our VGG16	0.9948
ResNet50	0.9927
Our Baseline CNN	0.9927
Our EfficientNetB1	0.9927
Our ResNet50	0.9927
Baseline CNN	0.9917
VGG16	0.9906
Our ResNet18	0.9896
MobileNetV2	0.9833
DenseNet121	0.9816
Our MobileNetV2	0.9812
Our DenseNet121	0.9795
EfficientNetB7	0.9656





Max-voting ensemble

(using the best 3 models)

Our	Original
• VGG11 • Baseline CNN • EfficientNetB1 • Accuracy score: 99.56% • Misclassifications: 4	• VGG16 • Baseline CNN • ResNet50 • Accuracy score: 98.89% • Misclassifications: 10

Our ensemble

Predicted: ESCA
Actual: Black Rot

Predicted: ESCA
Actual: Black Rot

Predicted: ESCA
Actual: Black Rot

Predicted: ESCA
Actual: Black Rot

Original ensemble

Predicted: ESCA
Actual: Black Rot

Predicted: ESCA
Actual: Black Rot

Predicted: ESCA
Actual: Black Rot

Predicted: ESCA
Actual: Leaf Blight

Predicted: Black Rot
Actual: ESCA

Predicted: Black Rot
Actual: ESCA

Predicted: Healthy
Actual: Black Rot

Predicted: ESCA
Actual: Black Rot

Predicted: ESCA
Actual: Black Rot

Conclusion

What we have done?

- analyzed, implemented, and improved a specific *Kaggle* project
- evaluated many pre-trained models for classifying 4 grape diseases
- translated the code from *TensorFlow* to *PyTorch*
- adjusted the original dataset splitting, experimented with new models, and applied novel optimization techniques

Future directions

- need more comprehensive and robust models
- create a new standard and large dataset (data availability acts as a bottleneck)