50.035
Computer Vision Team 9

Plant Disease Detection System

Authors	***Student ID***
Isaac Tay Eng Hian	1006327
Natalie Ang Zi Yi	1006131
Teh Hui Yi	1006383
Vainavi	1006119

Table of Contents

1 Introduction

1.1 Background

Agriculture is a vital industry, and plant diseases pose significant risks to crop yield and food security. Farmers often struggle to identify diseases early, which leads to widespread crop loss.

Traditional methods of disease identification require expert knowledge and manual inspection, which can be time-consuming and labour-intensive. However, recent developments in machine learning, especially computer vision, have provided tools to automate this process by analysing visual data.

1.2 Problem Statement

With this in mind, we constructed the following problem statement: How might we develop an **automated plant disease detection system** that assists farmers in **monitoring and detecting various diseases from leaf images**, enabling early intervention to reduce disease spread and minimise crop loss?

1.3 Research Focus

Our project will focus on 2 primary Computer Vision tasks:

1. **Multi-class Image Classification**: To identify and classify the type of disease based on a leaf image input, such that farmers know the necessary measures to take for their crops.
2. **Binary Image Segmentation**: To locate and segment the diseased regions on the leaf image, providing detailed visual feedback on the extent of disease progression.

We would use a different dataset and varied approaches for each task.

2 Datasets

2.1 PlantVillage dataset

The PlantVillage dataset[1] contains 54,305 images of healthy and diseased plant leaves across 20 disease types and 14 plant species. It is an extensive dataset with lab-controlled images of leaves against a static background, making it optimal for learning precise features and achieving higher accuracy scores. Therefore, it was chosen as the basis for our **image classification** task.

Original Class Distribution

The dataset originally had significant class imbalance, with each class ranging from 275 (*Cedar apple rust*) to 15,071 (*healthy*) samples. Notably, the *healthy* class disproportionately outweighed other classes, making up ~28% of the data.

This imbalance would pose a risk of bias during model training, where dominant classes could overshadow minority classes and consequently compromise generalization.

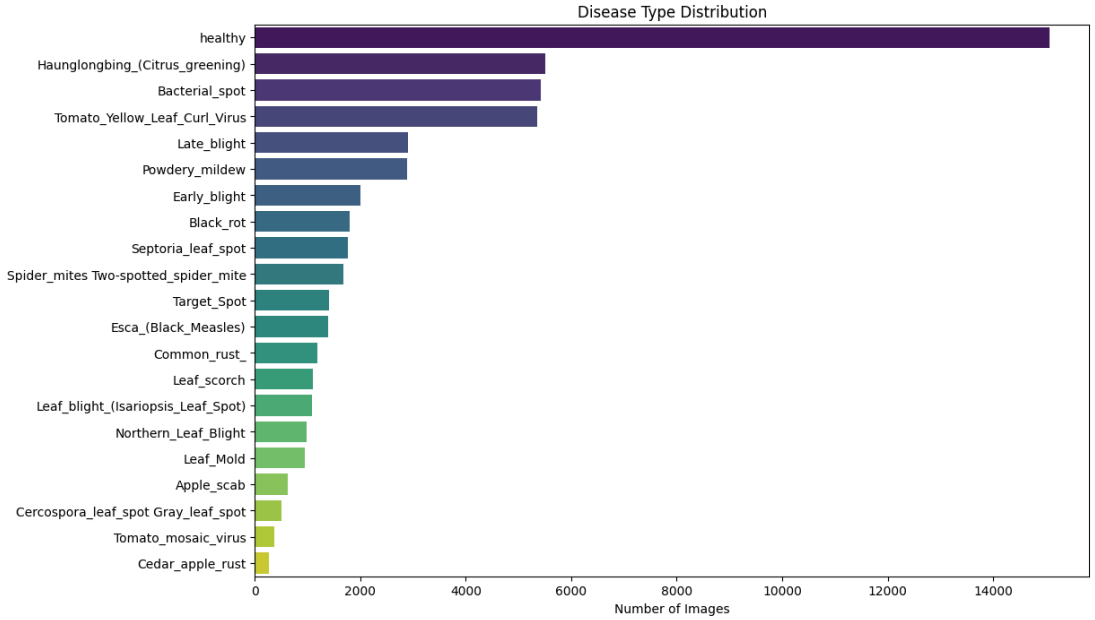


Figure 1: Original class distribution of PlantVillage dataset

Data Resampling

To ensure fairer class representation and allow the models to yield meaningful results, we adopted the following approach to address this imbalance.

[1] *PlantVillage Dataset*. (2019, September 1). Kaggle. https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset

1. Firstly, the *healthy* class was randomly undersampled to 6000 samples, such that it only slightly exceeded the next largest class (*Haunglongbing citrus greening*).

2. We then performed stratified splitting into 3 sets, preserving class proportions:
    - Training set: 70%
    - Validation set: 15%
    - Test set: 15%

3. Further resampling was applied for the **training set only**; keeping the validation and test sets unbiased. Random oversampling for certain classes was performed such that every class would contain at least 2000 samples.

    A balance between over and undersampling is crucial to maintain the integrity of the data. Undersampling of larger classes was avoided as it would result in information loss. On the other hand, we were careful not to overdo the oversampling for minority classes to prevent the model from overfitting or developing a bias to artificial samples.

    Our oversampling approach utilized the keras **ImageDataGenerator**, where we employed data augmentation techniques such as random rotations, flipping and shifting to generate new samples for each class. The final class distribution of the training set would still have some level of imbalance, but is significantly less skewed than before.

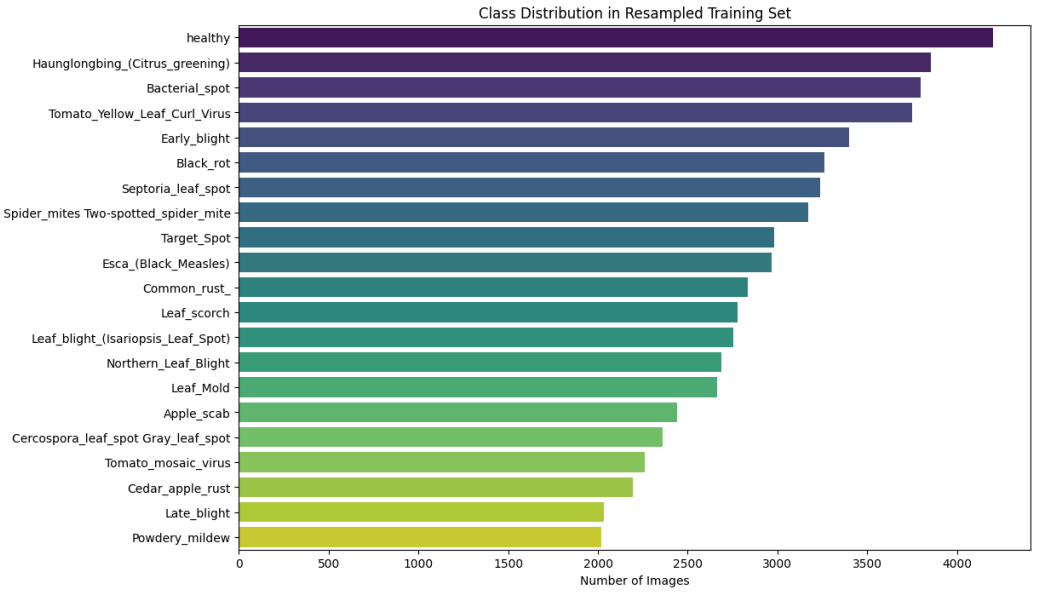


Figure 2: Class distribution of training set after data resampling

All additional data preparation steps are documented in **Data_preparation.ipynb**.

2.2 Leaf Disease Segmentation dataset

For our **image segmentation** task, we used the Leaf disease segmentation dataset[2] which is based on leaf images from the PlantDoc dataset[3]. It contains 588 images of diseased leaves along with their 588 corresponding binary masks.

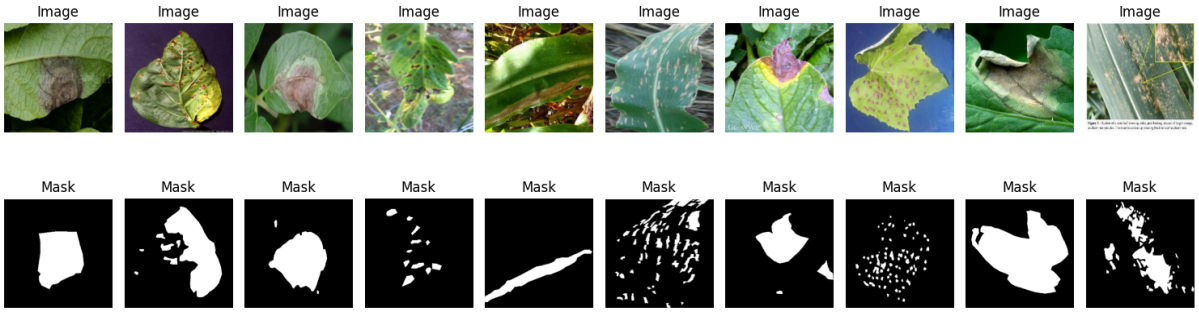


Figure 3: Images and corresponding masks in segmentation dataset

Unlike the PlantVillage dataset which contained lab-controlled images, the original PlantDoc dataset has photographs of leaves on intact plants, captured against natural and realistic backgrounds. This makes the data more representative of practical crop images, hence they are expected to generalize more effectively to real-life scenarios.

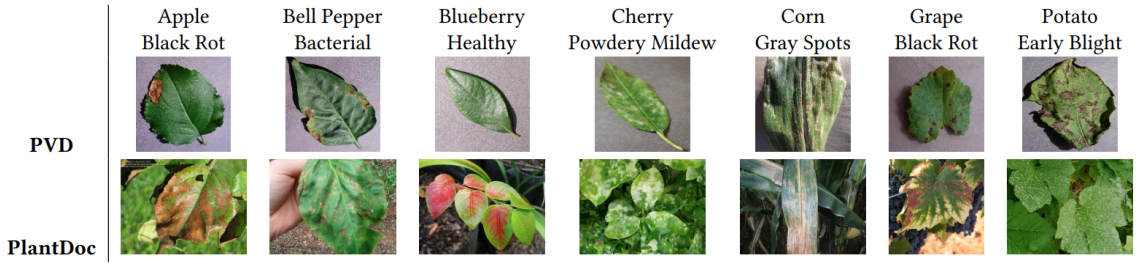


Figure 4: Comparison between PlantVillage and PlantDoc images[3]

[2] *Leaf disease segmentation dataset*. (2021, September 3). Kaggle. https://www.kaggle.com/datasets/fakhrealam9537/leaf-disease-segmentation-dataset

[3] Pratikkayal. (n.d.). *GitHub - pratikkayal/PlantDoc-Dataset.* GitHub. https://github.com/pratikkayal/PlantDoc-Dataset

3 Approach

3.1 Image Classification

The samples from the PlantVillage dataset were already pre-processed and split into training, validation and test sets as explained in the previous section.

Training

We trained each model using **Cross-Entropy Loss**, with further training time augmentations such as: **Random Vertical Flip**, **Random Horizontal Flip**, and **Random Erasing.** Random Erasing helped a lot to improve the **AUROC** of the model as erasing a random patch of the image helped to generalize the model.

The **Adam Optimizer** was utilized for its balance of initial converging speed and ability to fine tune the model.

Models Tested

Our study evaluated multiple state-of-the-art convolutional neural network architectures to determine the optimal model for our classification task. The following architectures were investigated:

- **SqueezeNet**: A compact architecture achieving AlexNet-level accuracy with significantly fewer parameters through its fire module design, combining squeeze and expand layers for efficient model size reduction.[4]
- **ResNet50**: A deep residual network employing residual connections through 50 convolutional layers to address the vanishing gradient problem and enable effective training of deep architectures through identity mappings.[4]
- **MobileNetV2**: An architecture optimized for mobile and edge devices that utilizes depthwise separable convolutions to reduce computational complexity while maintaining effective feature extraction capabilities.[4]
- **ShuffleNetv2**: A lightweight architecture that implements channel splitting and shuffling operations, designed for mobile devices to achieve optimal balance between computational efficiency and model accuracy.[4]

[4] Iandola, F. N., Jr., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (n.d.). SqueezeNet: AlexNet-level Accuracy with 50X Fewer Parameters and <0.5 MB Model Size. *SqueezeNet: AlexNet-level Accuracy With 50X Fewer Parameters and <0.5 MB Model Size*. https://pdfs.semanticscholar.org/463c/72e39b12690b14ccda377088cc800eb1fc56.pdf

- **EfficientNet-BO**: An architecture developed using compound scaling methodology that optimizes network depth, width, and input resolution simultaneously through compound coefficients.[4]

- **InceptionV3:** A deep convolutional neural network (CNN) architecture designed for image classification and recognition tasks. It is an improved version of the Inception architecture introduced by Google, known for achieving high accuracy while being computationally efficient.[5]

- **Custom Model:**
    - Inspired by lightweight architecture of ShuffleNetV2
    - Initial convolution, max pooling
    - Sequential ShuffleBlocks
    - Final convolution
    - Fully connected

While all models were trained under identical conditions for 50 epochs, SqueezeNet's, EfficientNet-B0's and InceptionV3's training was shortened to 10 epochs due to GPU resource limitations. Training and validation losses, along with metric scores across epochs, were visualised in separate graphs for each model (refer to Appendix 7.1).

To compare performances after training, a comparison of calculated metrics of all models on the test set were displayed as bar graphs, found in Appendix 7.2. Additionally, the performance of individual classes for all models was analyzed using both absolute and proportional confusion matrices (Appendix 7.3).

[5] Iandola, F. N., Jr., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (n.d.). SqueezeNet: AlexNet-level Accuracy with 50X Fewer Parameters and <0.5 MB Model Size. *SqueezeNet: AlexNet-level Accuracy With 50X Fewer Parameters and <0.5 MB Model Size*. https://pdfs.semanticscholar.org/463c/72e39b12690b14ccda377088cc800eb1fc56.pdf

3.2 Image Segmentation

The Leaf disease segmentation dataset was divided into training, validation and test sets with a 80-10-10 split ratio.

Data Pre-processing and Augmentation

For the training set, images and masks were randomly cropped and resized to 256×256 pixels and converted to tensors. Additional augmentation techniques such as horizontal flip and vertical flip were applied from the standard torchvision v2 transforms.

Training

We used a composite loss function that combines two different losses: **Binary Cross-Entropy with Logits Loss** and **Dice Loss**. The first provides a smooth gradient for optimisation of binary classification tasks, while the latter measures the overlap between the predicted and ground truth masks. This combination would leverage the advantages that come with both approaches.

To achieve efficient training, we also used the **Adam optimizer** for segmentation.

Models Tested

Several segmentation models were trained and evaluated to identify the most effective model architecture for our problem. These models include:

- **UNet**: A widely adopted architecture featuring a symmetric encoder-decoder structure with skip connections. Its ability to capture fine and coarse details through skip connections is highly effective in biomedical image segmentation, making it a suitable choice for our segmentation task.[6]
- **UNet++**: An enhanced version of UNet that introduces dense skip connections and deep supervision. This redesigned skip connection pattern creates an ensemble of U-Nets of varying depths, potentially enabling more precise segmentation through better feature fusion.[7] It may potentially perform better than UNet for our task.
- **DeepLabV3** (with ResNet50): An advanced semantic segmentation architecture that employs atrous spatial pyramid pooling (ASPP) to robustly segment objects at multiple scales. The ResNet50 backbone provides strong feature extraction capabilities while maintaining computational efficiency through residual connections.

---

[6] Ronneberger, O., Fischer, P., & Brox, T. (2015, May 18). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. arXiv.org. https://arxiv.org/abs/1505.04597v1

[7] Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., & Liang, J. (2018). *UNet++: A nested U-Net architecture for medical image segmentation*. arXiv. https://arxiv.org/abs/1807.10165

This variant is particularly known for its effective balance between accuracy and computational efficiency.[8]

- **DeepLabV3** (with mobilenet_v2): A lightweight variant of DeepLabV3 utilizing MobileNetV2 as the backbone network. This configuration offers reduced computational overhead while preserving effective semantic segmentation capabilities, making it suitable for resource-constrained environments.
- **MAnet**: A Multi-scale Attention Network that incorporates attention mechanisms at multiple scales. This architecture enhances feature representation by focusing on relevant spatial regions while maintaining awareness of both local and global context.[9]

All models underwent training for 200 epochs under consistent conditions.

[8] Kouidri, A. (2024, September 20). DeepLabV3 Guide: Key to Image Segmentation. *Ikomia*. https://www.ikomia.ai/blog/understanding-deeplabv3-image-segmentation

[9] Li, R., Zheng, S., Zhang, C., Duan, C., Su, J., Wang, L., & Peter M Atkinson. (2021). *Multi-Attention-Network for Semantic Segmentation of Fine-Resolution Remote Sensing Images*. https://lironui.github.io/Files/MANet.pdf

4 Results and Evaluation

4.1 Models Performance

Image Classification Results

For the evaluation of our **image classification** models, the metrics used include:

- **Accuracy**: The proportion of correct predictions (both positive and negative) among all predictions made. This metric provides a straightforward measure of the model's overall performance in correctly classifying images across all classes.
- **AUROC** (Area Under the Receiver Operating Characteristic curve): A performance metric that evaluates the model's ability to distinguish between classes across various classification thresholds. This metric is particularly valuable as it assesses the model's discriminative ability independent of any chosen classification threshold, ranging from 0 to 1, where 1 indicates perfect classification.
- **F1**: A harmonic mean of precision and recall, providing a single score that balances both metrics. This score is particularly useful when dealing with imbalanced datasets as it considers both false positives and false negatives, making it a more comprehensive metric than accuracy alone.

Table 1: Evaluation of image classification models on test set

	Accuracy	AUROC	F1	Average
SqueezeNet	0.9832	0.8242	0.9832	0.9302
ResNet50	0.9721	0.8162	0.9721	0.9202
MobileNetV2	0.9841	**0.8243**	0.9841	0.9308
ShuffleNetV2	0.9637	0.8241	0.9637	0.9172
EfficientNet	0.9866	**0.8243**	0.9866	**0.9325**
InceptionV3	**0.9956**	**0.8243**	**0.9956**	**0.9385**
Custom	0.9844	**0.8243**	0.9844	**0.9310**

Our **custom model, InceptionV3** and **EfficientNet** were chosen as the 3 best-performing classification models as they had the highest average scores. Among the three, our custom

model stands out for its lightweight architecture and lower computational requirements. Built as an improved version of **ShuffleNet**, it performed comparably to **EfficientNet** and **InceptionV3** despite requiring fewer resources.

Considering these advantages, we selected the **custom model** for our prototype due to its balance between computational efficiency and accuracy.

Image Segmentation Results

For the evaluation of our **image segmentation** models, the metrics used include:

- **Pixel F1**: measures a balance between the precision and recall if we treat the segmentation as a binary classification task.
- **IoU (Intersection over Union)**: measures the overlap between the output and ground truth, specifically the ratio of the intersected area of the predicted mask and ground truth mask to their combined area.
- **Dice Coefficient**: measures the similarity between the predicted mask and ground truth mask. It accounts for both precision and recall. As we are working with an imbalanced dataset (background class dominates), this metric is useful as it penalises both false positives and false negatives equally.

Table 2: Evaluation of image segmentation models on test set

	Pixel F1	IoU	Dice	Average
UNet (ResNet 50)	0.6892	0.7222	0.7272	0.7129
UNet++ (ResNet 50)	0.6675	0.7084	0.7336	0.7031
DeepLabV3 (ResNet50)	0.7325	0.7350	0.7465	0.7380
DeepLabV3 (mobilenet_v2)	**0.7364**	**0.7535**	**0.7559**	**0.7486**
MAnet (ResNet 50)	0.7066	0.6930	0.7163	0.7054

DeepLabV3 with mobilenetV2 was selected for our prototype as it outperformed the other segmentation models on all metrics.

4.2 Analysis

Visualizing Classification

To investigate which disease types are prone to misclassification, we visualized the models' performances on each individual class through **confusion matrices**.

We looked at our custom model's proportional confusion matrix (values for each class are normalized to between 0 and 1) for further analysis.

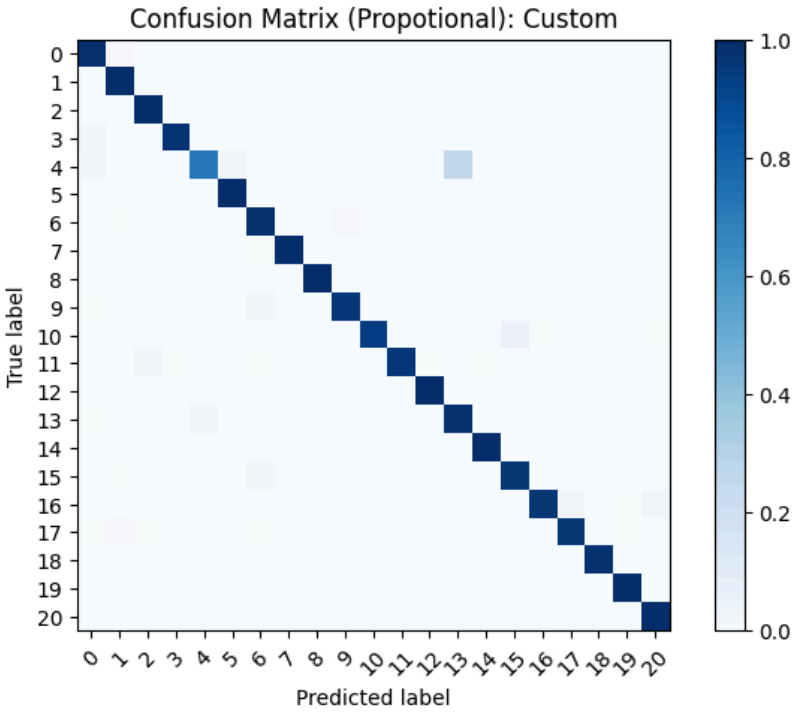


Figure 5: Proportional confusion matrix of custom model

Our custom model showed a **notable tendency to misclassify class 4** (*Cercospora leaf spot*) **samples into class 13** (*Northern leaf blight*); and vice versa to a small extent. Comparatively, other classes exhibited lower levels of misclassification.

This can be attributed to two factors.

- Firstly, both diseases **share visual similarities** as their diseased areas are similar in color and shape, and tend to appear on similarly shaped leaves.
- Secondly, despite us taking a data resampling approach, both classes still have **relatively fewer samples** within the dataset – class 4 is the third smallest class while class 13 is the sixth smallest.

Of the two, visual similarity appears to be the main contributing factor to the misclassifications.

Figure 6: Comparison between a class 4 (left) and class 13 (right) sample

Overall, it can be deduced that the model **has a stronger ability to classify disease types with visual distinctiveness and more training samples**. A good example would be class 8 (*Haunglongbing citrus greening*), the largest disease class which displays visual patterns distinct from other classes. There were little to no misclassifications for class 8.

Visualizing Segmentation

A visual comparison of ground truth and predicted masks for the test set was made for our selected segmentation model.

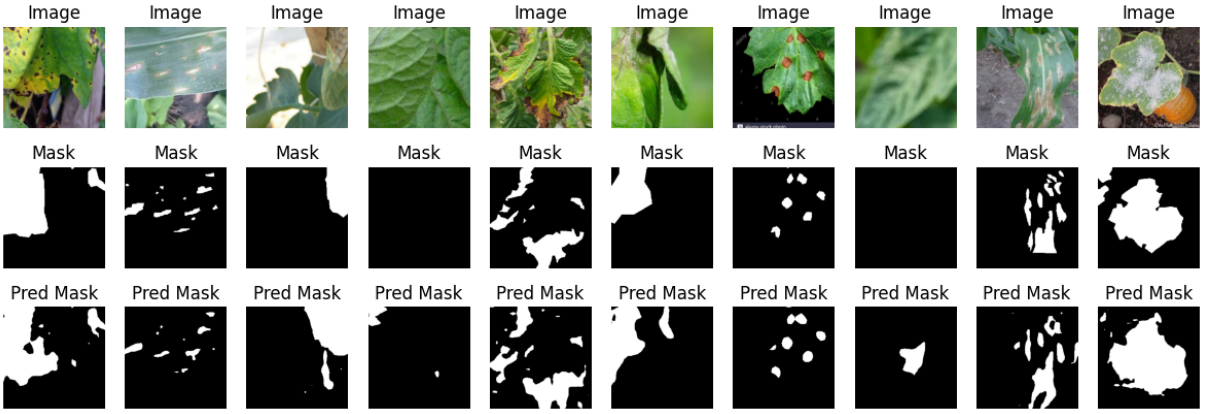


Figure 7: Test set samples, ground truths and predicted masks by DeepLabV3

It can be observed that the model successfully segmented the rough shapes of diseased areas for the majority of the samples, albeit with some imperfections:

- **Objects in the background** were occasionally misidentified as diseased areas.
- For healthy images, the model sometimes **identified non-existent diseased regions.**

It can be deduced that the model **segments more accurately for diseases with smaller affected areas** and **colors distinct from the leaf and background**. In contrast, segmentation results were less precise for diseases with large affected areas or colors that blend in with the leaf itself. In such cases, predicted shapes of diseased regions were less defined.

5 System Design

5.1 Implementation

Our prototype requires a functionality that allows the user to conveniently upload photos of their plant leaf and receive instant feedback generated by our models.

To achieve this, we utilized PyTorch to save our selected classification and segmentation models after training, the **custom model** and **DeepLabV3 with mobilenetV2** respectively. We then integrated them into a simple web application built with Streamlit. The application's workflow is illustrated below.

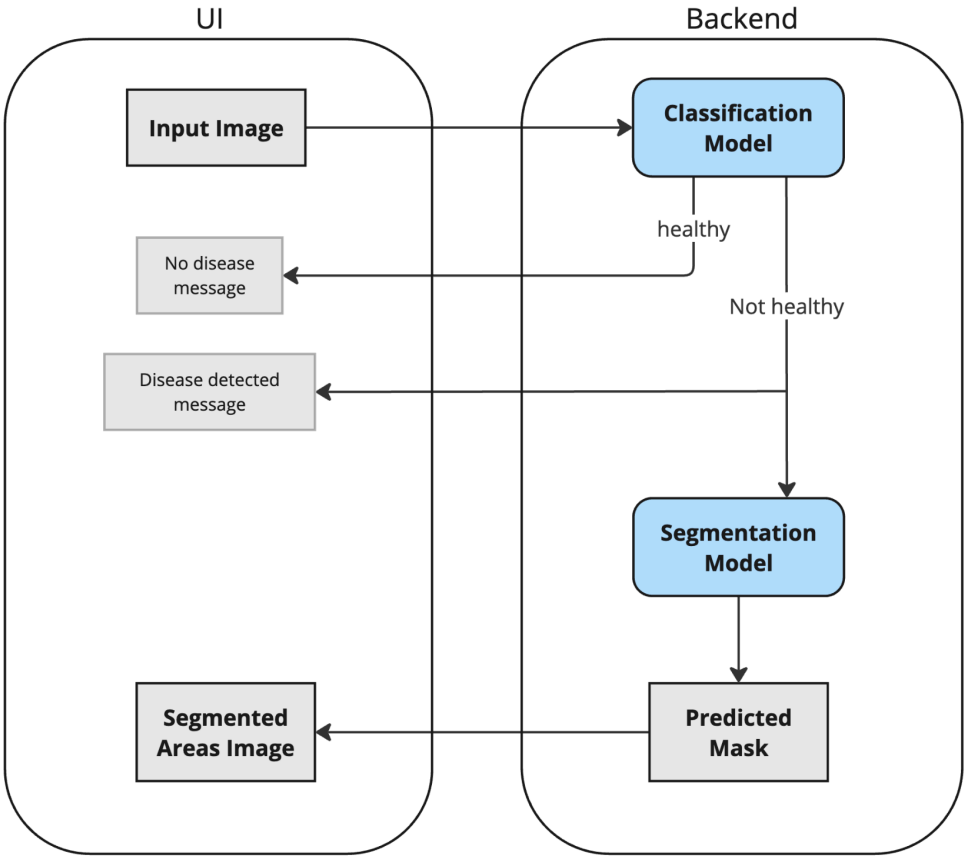


Figure 8: System diagram of prototype

The process begins with the user uploading an image on the user interface, after which a button labeled *'Does my plant have a disease?'* appears. Upon clicking it, the image goes through this pipeline:

- The classification model identifies the plant disease (if any), and generates a message.
- If the model identifies it to be healthy, the pipeline ends. If a disease is detected, the image goes into the segmentation model to generate a predicted mask, which is used to produce an image mapping the diseased areas of the leaf.

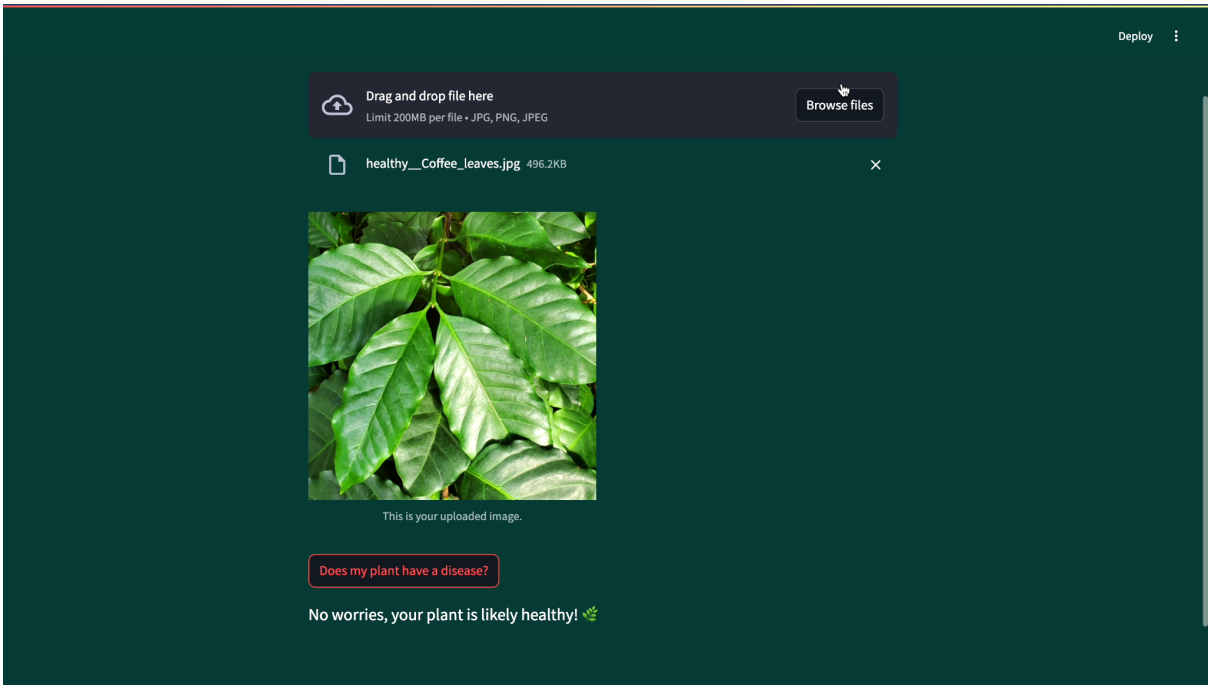


Figure 9: Screenshot 1 of prototype (healthy image)

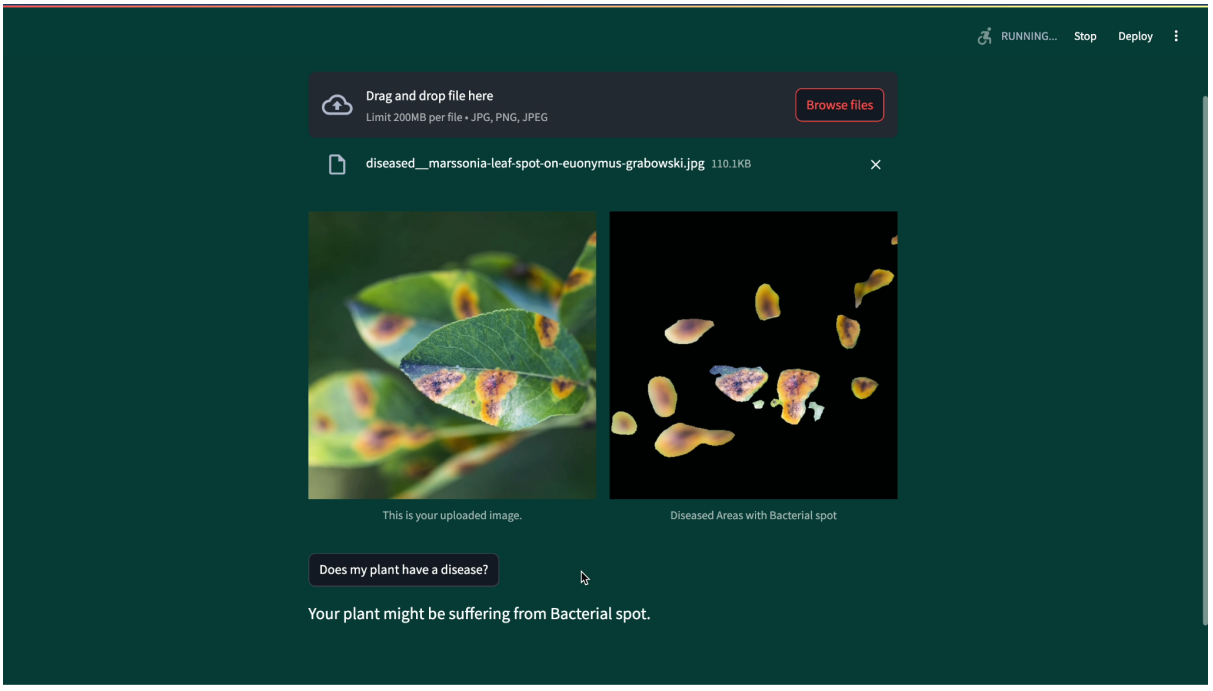


Figure 10: Screenshot 2 of prototype (diseased image)

5.2 Source Code and Instructions

Our runnable prototype source code, demo video and relevant files can be found in this GitHub repository: https://github.com/natnotnet/50.035-Plant-Disease-Detection

Instructions to run the code and file-related details are documented in **README.md**.

Note: If you face any issues with the link or have questions, please reach out to us.

6 Conclusion

In this project, we developed a simple plant disease detection system leveraging image classification and image segmentation models to help address the challenge of early plant disease identification.

Our solution aligns with several of the **UN's Sustainable Development Goals**[10]:

1. **Goal 2: Zero Hunger** – By enabling early plant disease detection, our solution addresses food security and enhances crop yield, contributing to more productive agricultural practices.
2. **Goal 12: Responsible Consumption and Production** – By facilitating targeted disease management, our solution promotes more responsible usage of resources such as pesticides, reducing waste and environmental impact.
3. **Goal 8: Decent Work and Economic Growth** – As our solution aims to minimise crop losses, it helps improve farmers' economic stability, thereby contributing to sustainable livelihoods and economic growth in agricultural communities.

Future Work

While our system represents a step towards addressing our identified problem, improvements can be made in several areas to enhance its performance, usability and impact.

For performance improvements, we can explore using Generative Adversarial Networks (GANs) for better/more advanced generation of synthetic samples in data resampling. This would allow the models to learn more useful features during training. We may also employ transfer learning or fine-tuning on our models to further optimize their accuracies.

Currently, our system is limited to classifying 20 disease types across 14 plant species, only addressing the problem to an extent. We can consider utilizing data from multiple datasets to cover a wider range of diseases common to plant crops. Furthermore, sourcing for realistic

[10] United Nations. (n.d.). *The 17 goals*. United Nations. Retrieved December 9, 2024, from https://sdgs.un.org/goals

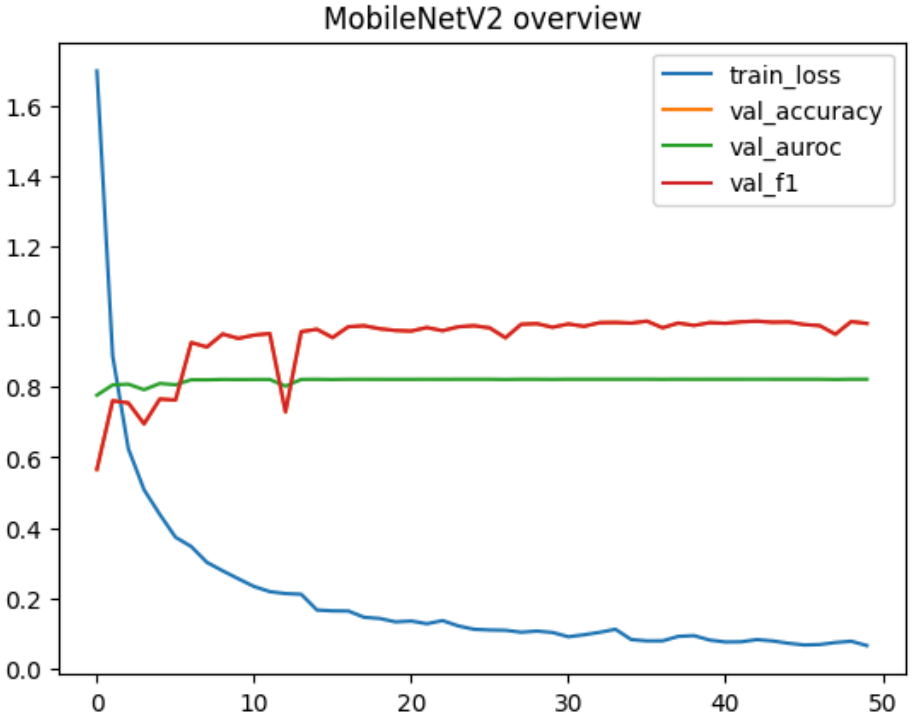
samples rather than only lab-controlled images for our classification models will allow our system to generalize better to field conditions.

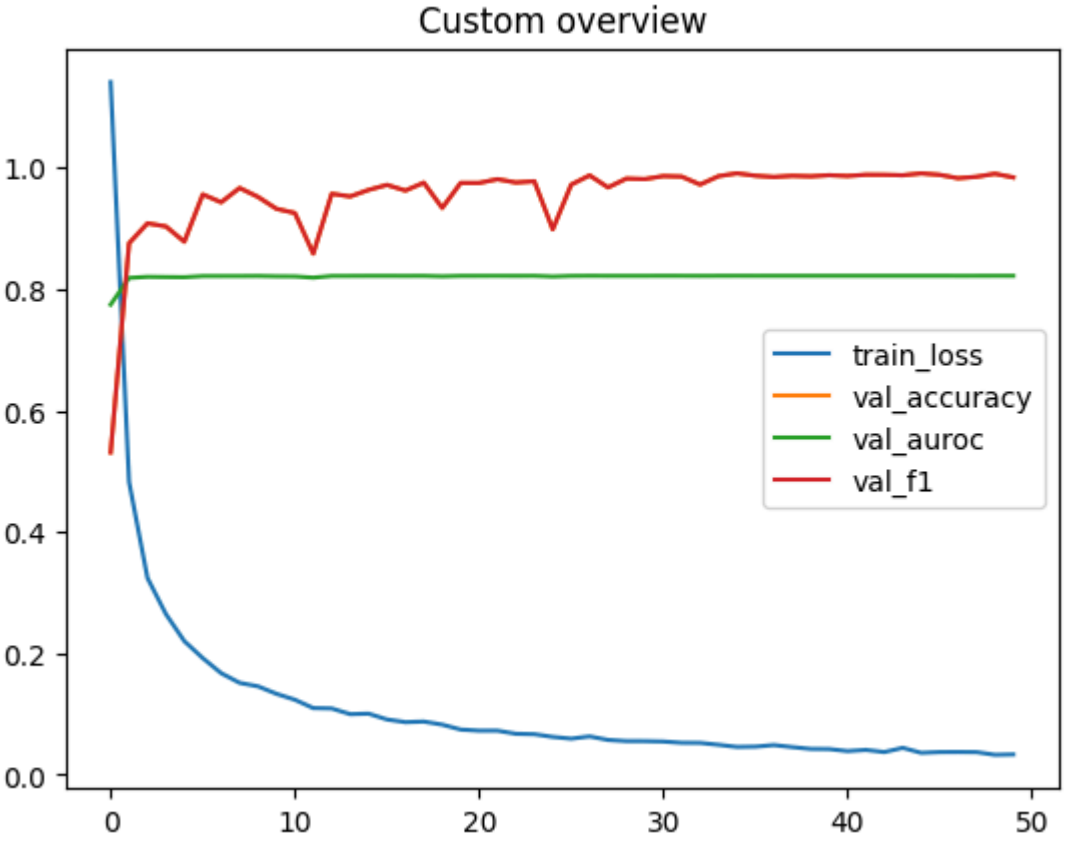
Our prototype is functional but not yet a practical tool for prolonged use. We can increase its scale in the future by integrating a feature for users to store a collection of photos over time, to clearly monitor the development of crop health.

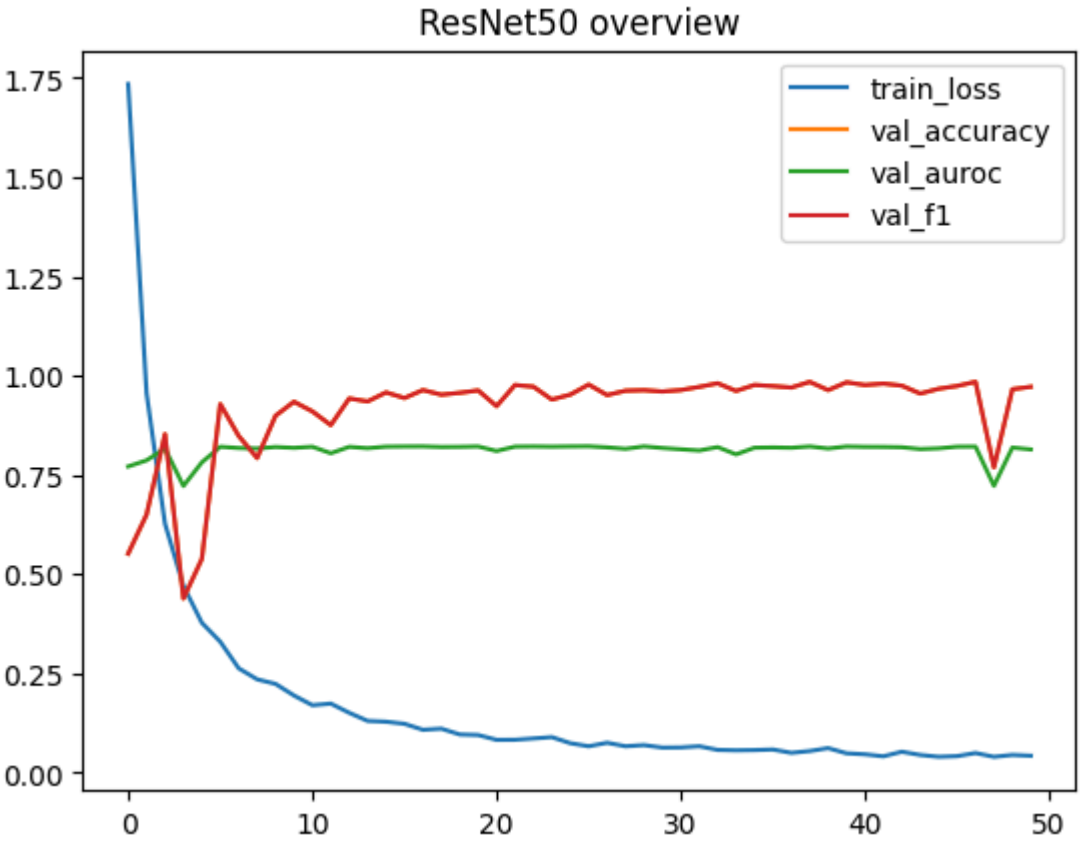
Finally, we may consider developing an additional segmentation model that segments the entire leaf in the image. This will allow us to calculate the percentage of the leaf's surface area affected by disease, providing a quantitative measure of the disease progression and aiding in more informed decision-making.

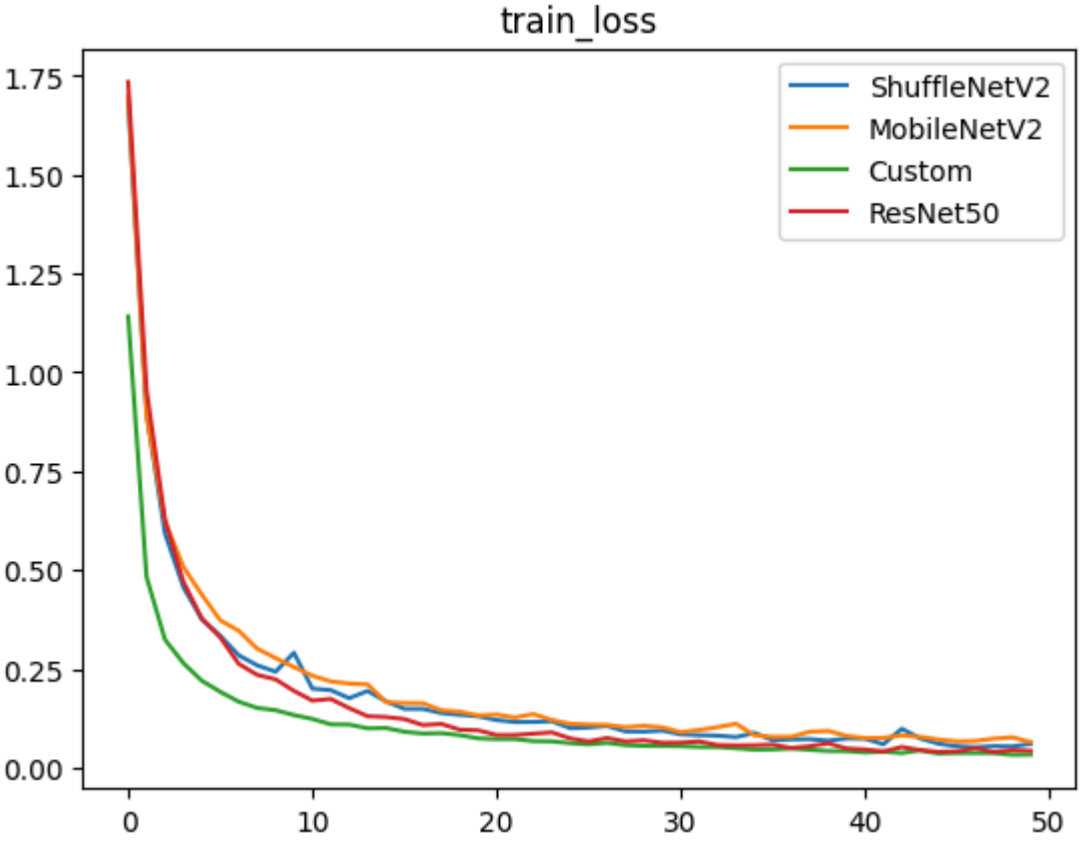
7 Appendix

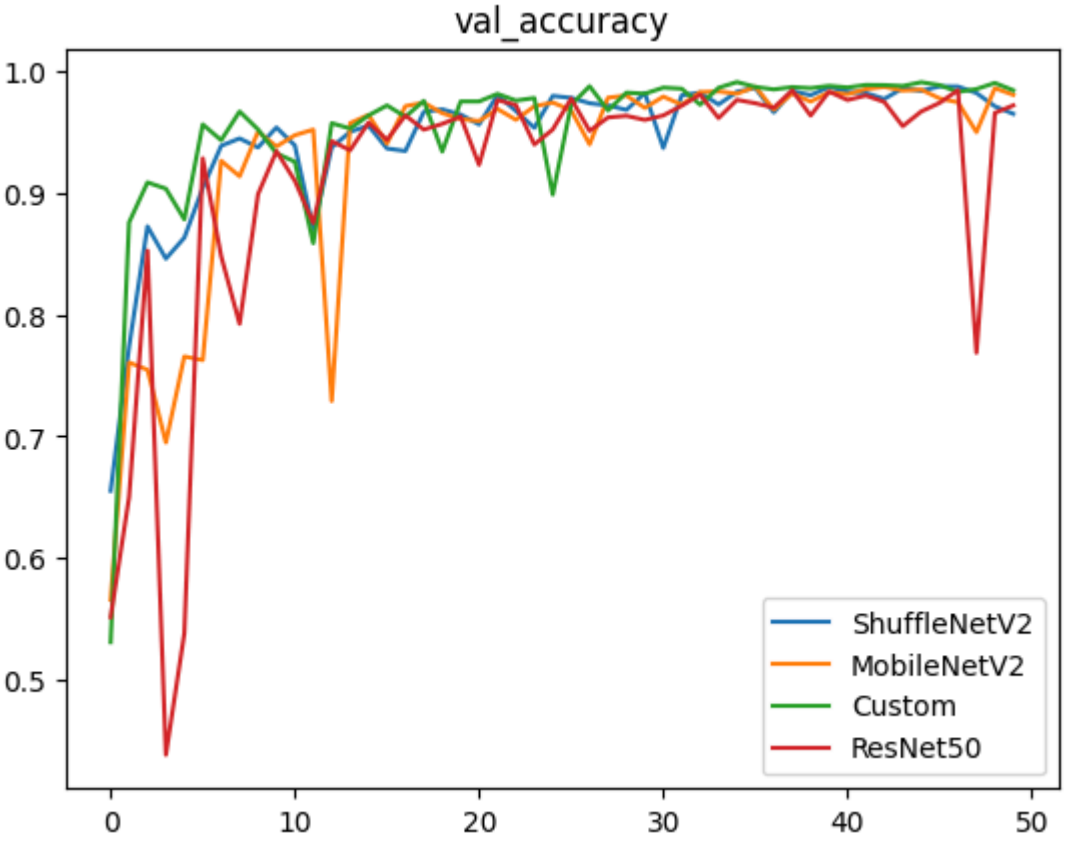
7.1 Training Graphs

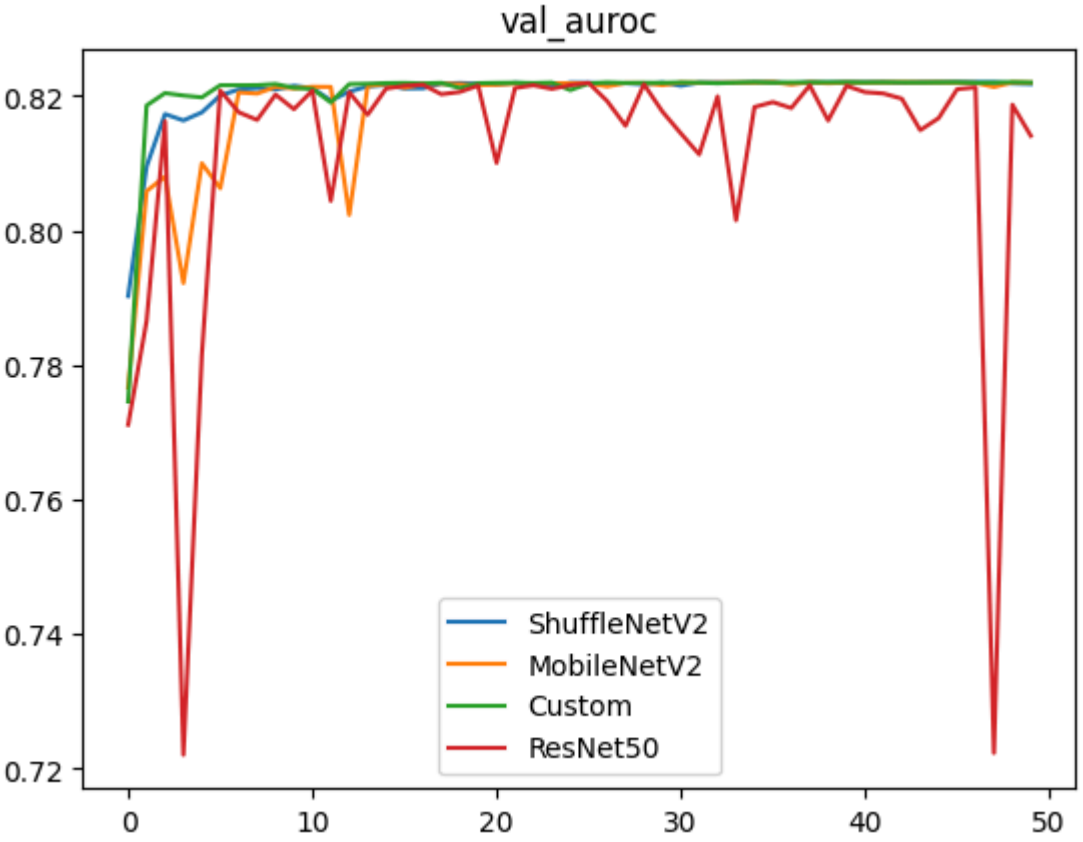


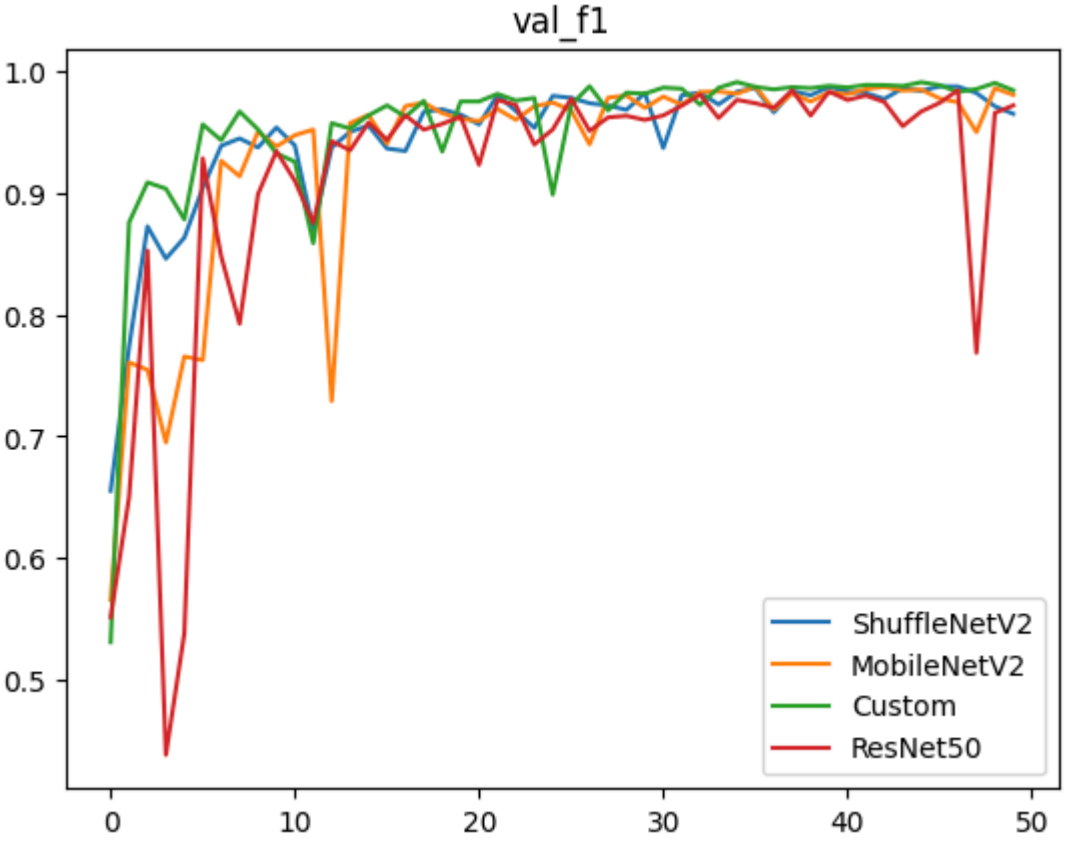

Custom overview
train_loss
val_accuracy
val_auroc
val_f1
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50


ResNet50 overview
train_loss
val_accuracy
val_auroc
val_f1
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50

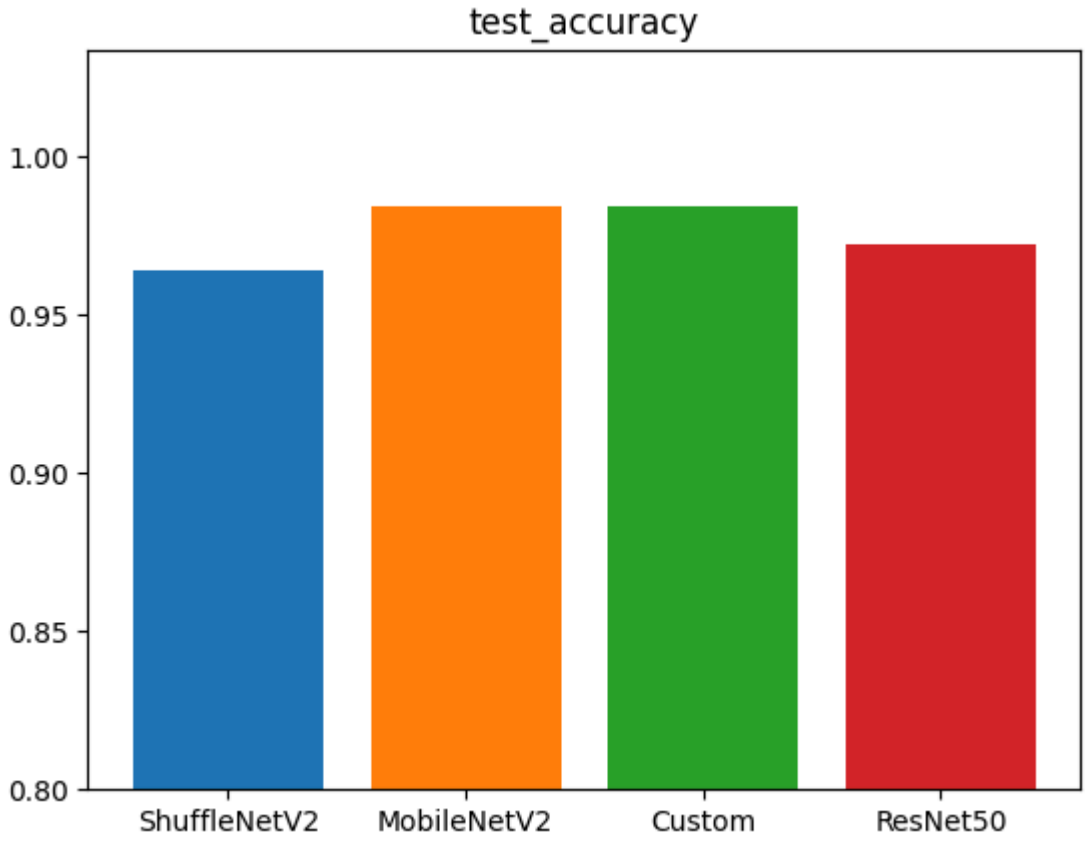

train_loss
ShuffleNetV2
MobileNetV2
Custom
ResNet50
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50

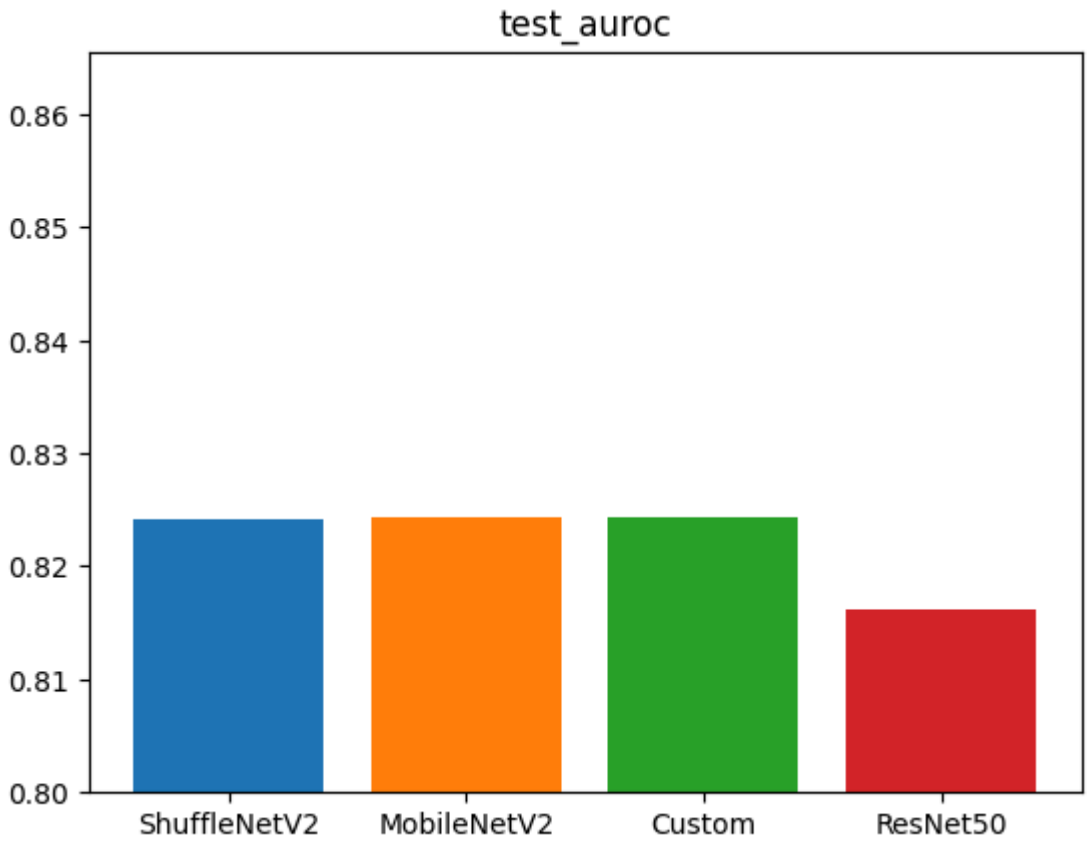

val_accuracy
ShuffleNetV2
MobileNetV2
Custom
ResNet50
1.0
0.9
0.8
0.7
0.6
0.5
0
10
20
30
40
50

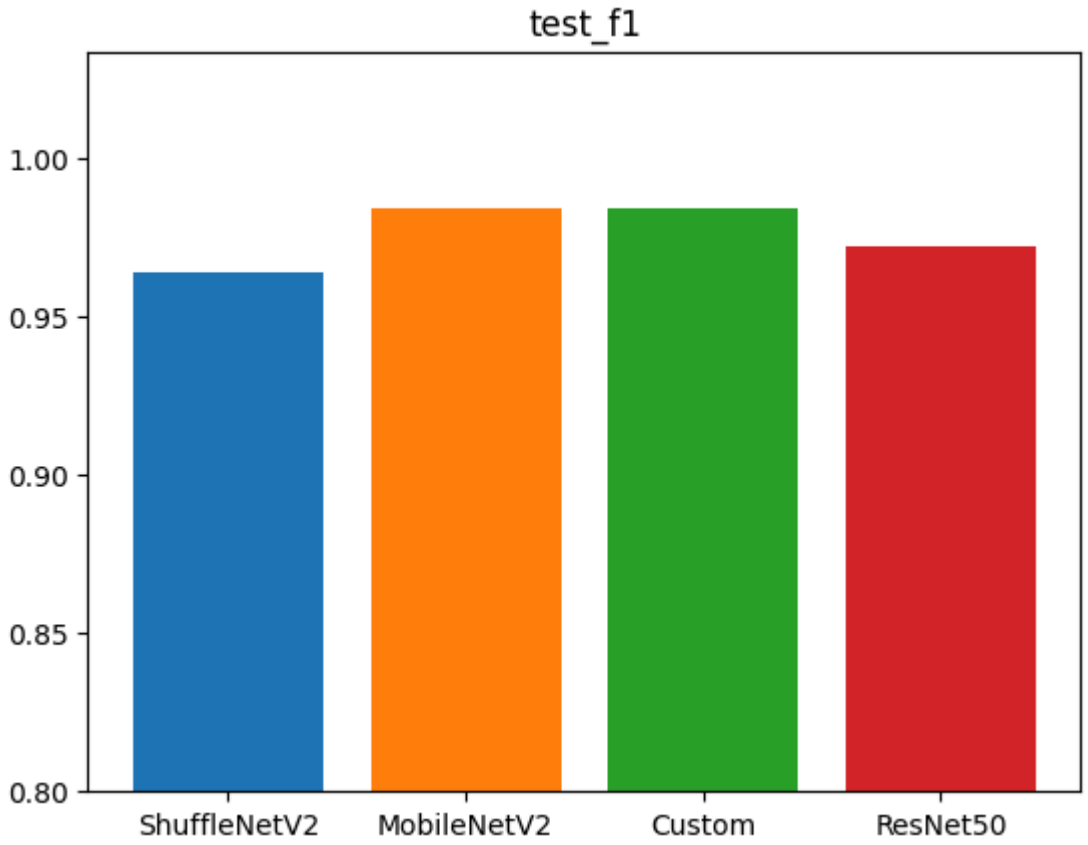

val_auroc
0.82
0.80
0.78
0.76
0.74
0.72
0
10
20
30
40
50
ShuffleNetV2
MobileNetV2
Custom
ResNet50


val_f1
1.0
0.9
0.8
0.7
0.6
0.5
0
10
20
30
40
50
ShuffleNetV2
MobileNetV2
Custom
ResNet50

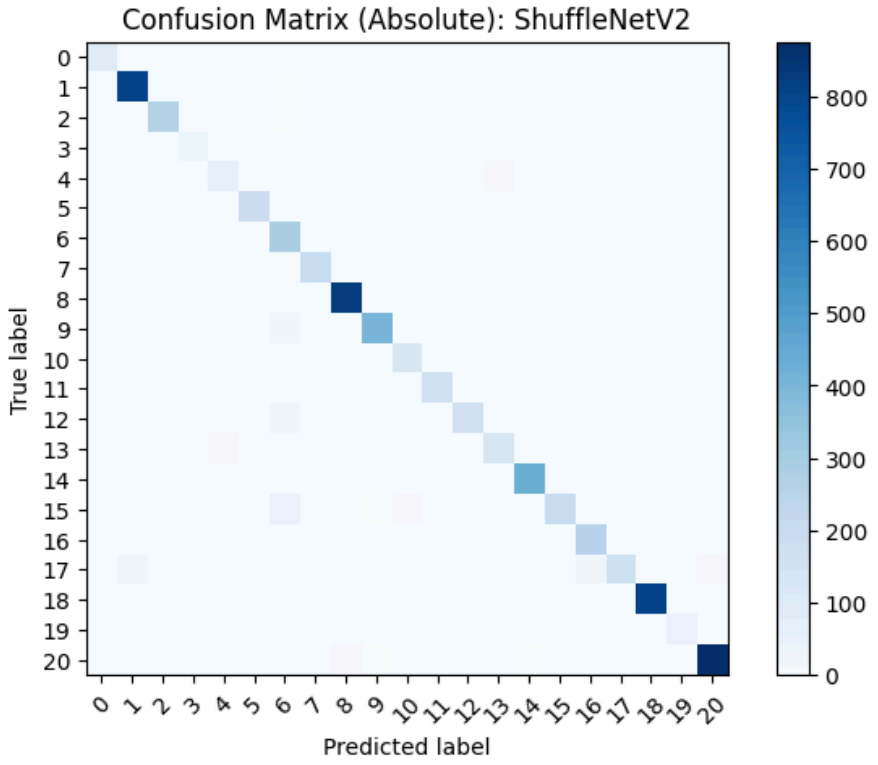
7.2 Metrics Graphs

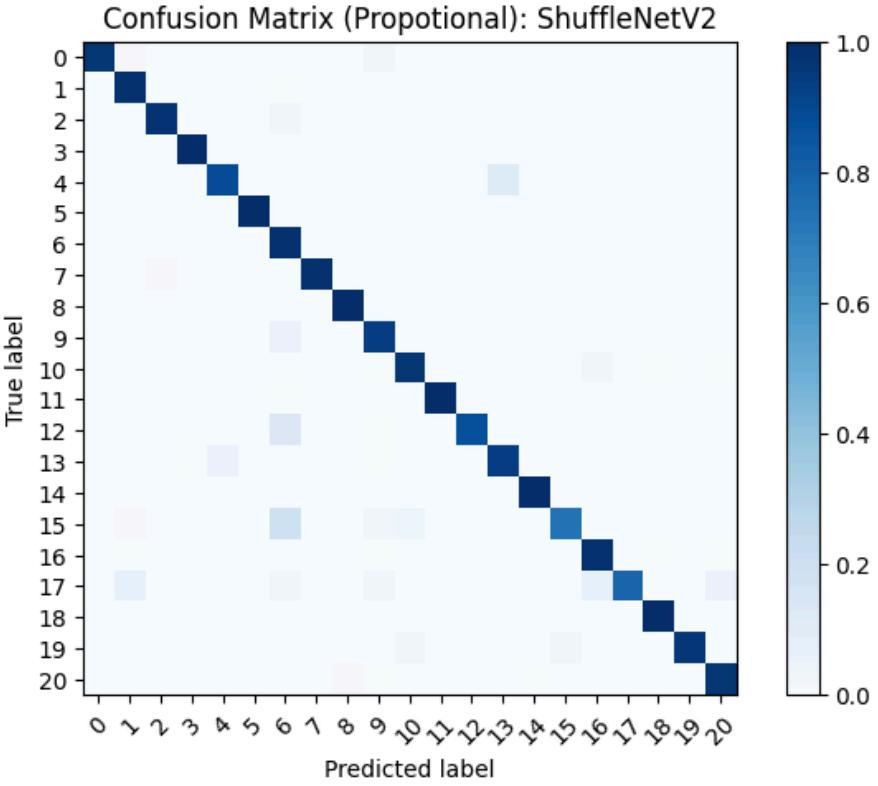


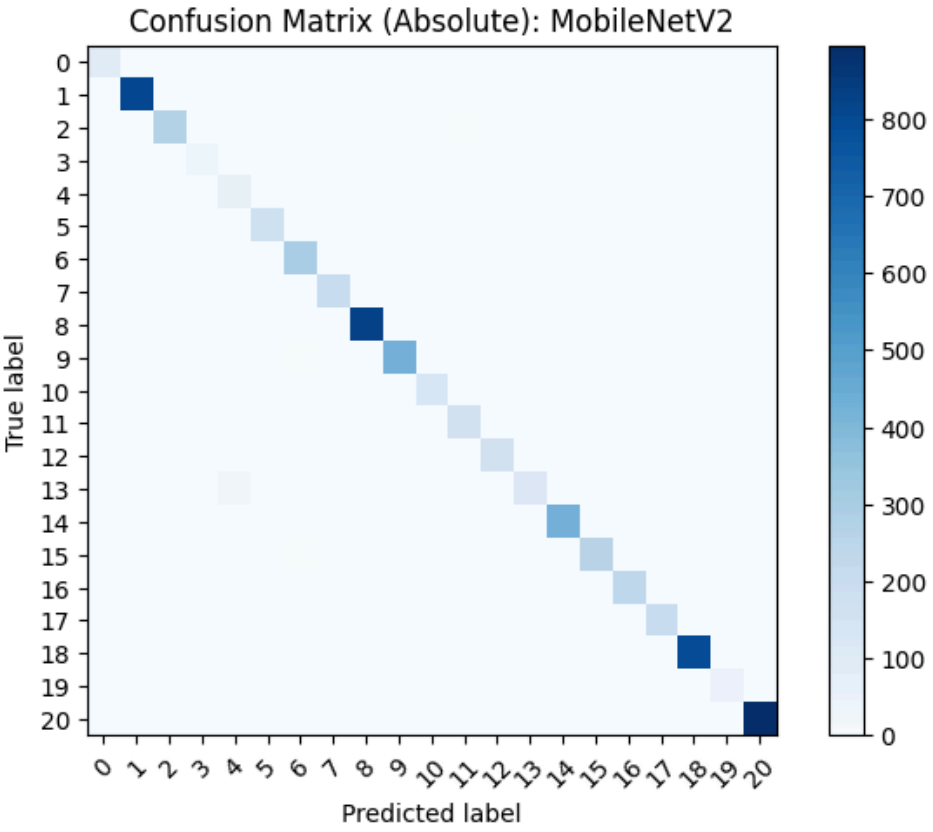


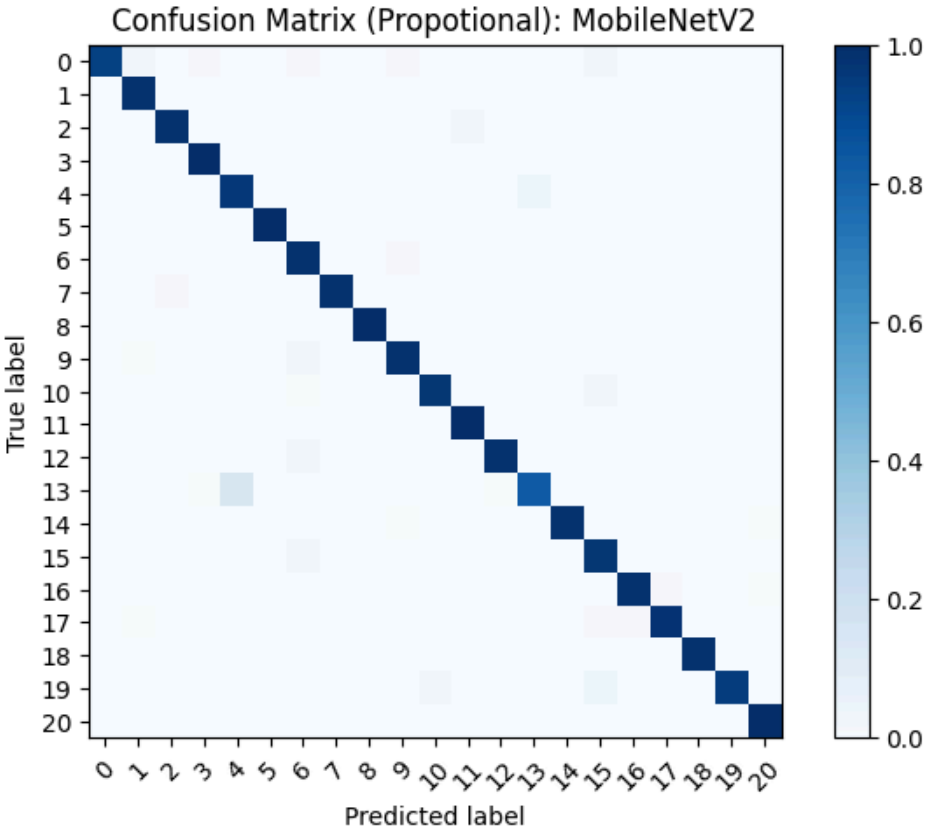

test_f1
1.00
0.95
0.90
0.85
0.80
ShuffleNetV2
MobileNetV2
Custom
ResNet50

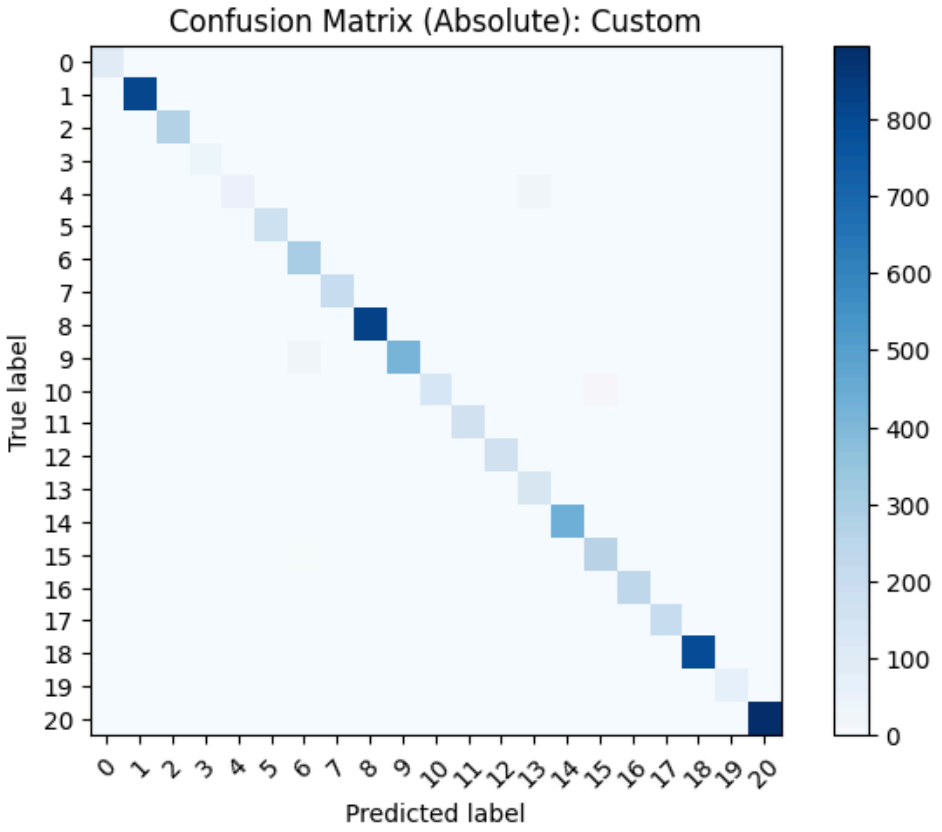
7.3 Confusion Matrices

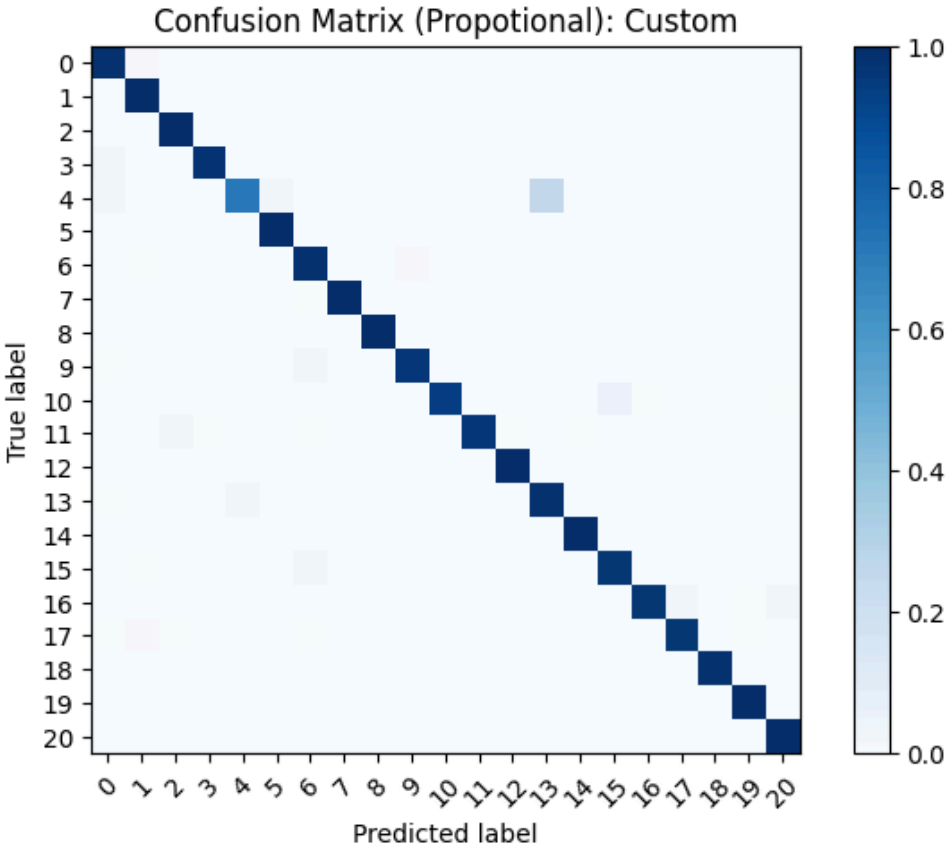


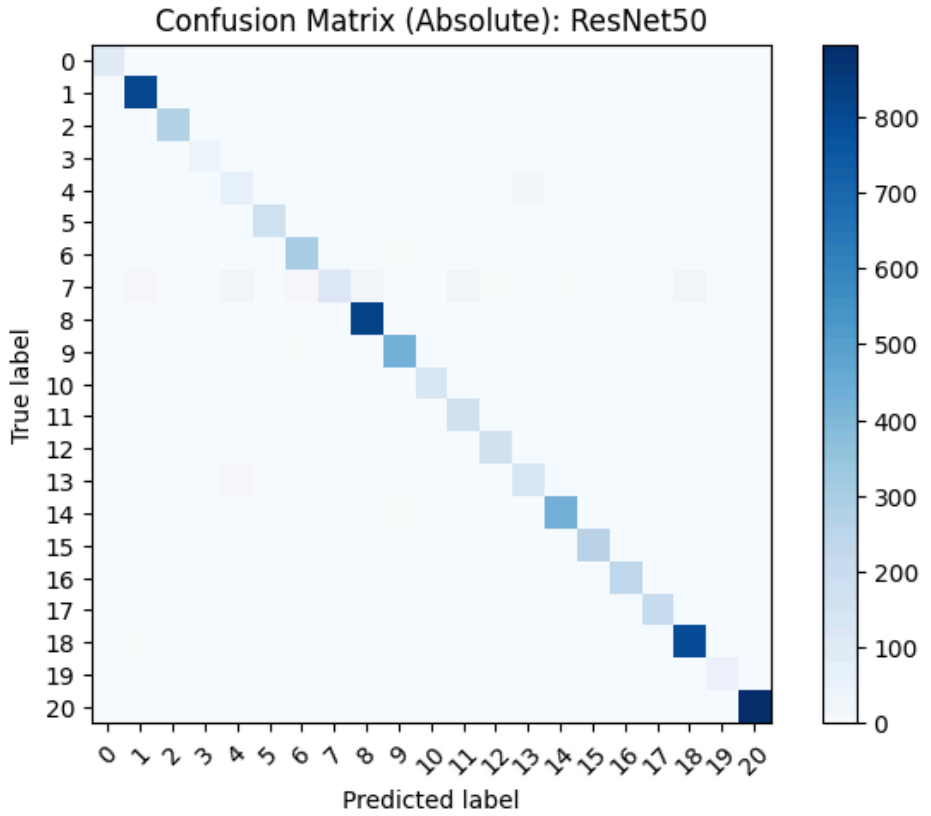


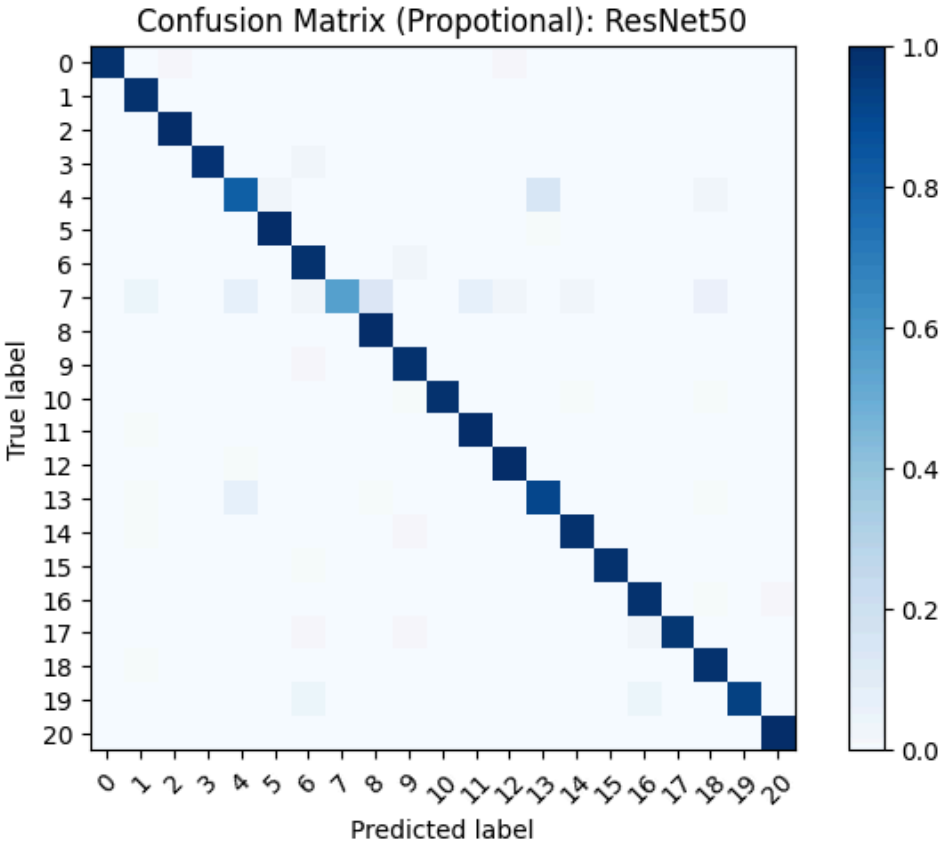

Confusion Matrix (Absolute): MobileNetV2
True label
Predicted label
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
800
700
600
500
400
300
200
100
0

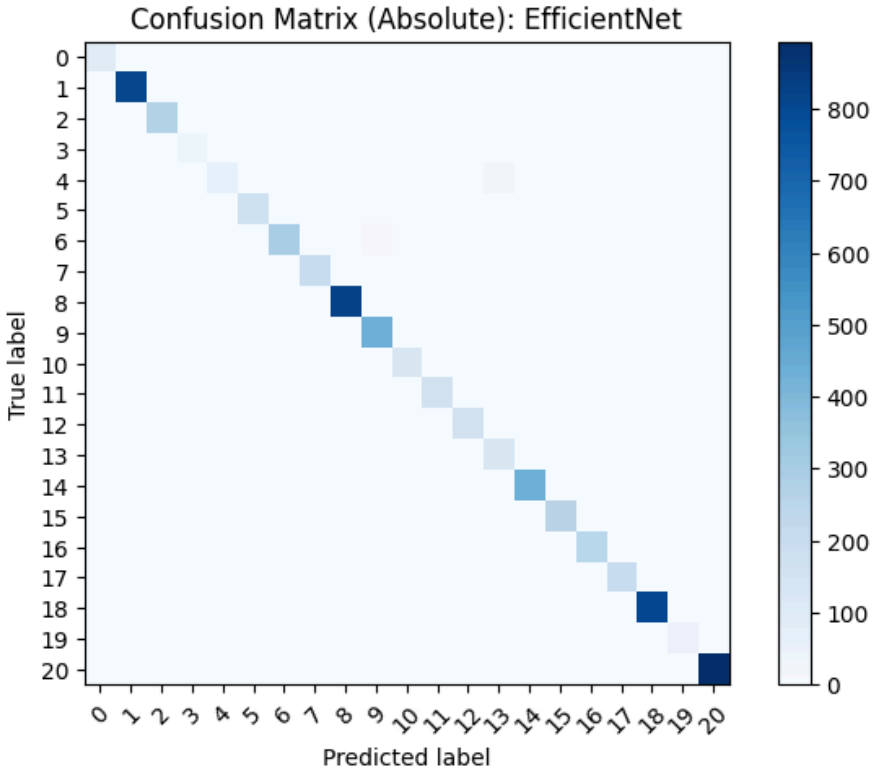

Confusion Matrix (Propotional): MobileNetV2
True label
Predicted label
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1.0
0.8
0.6
0.4
0.2
0.0

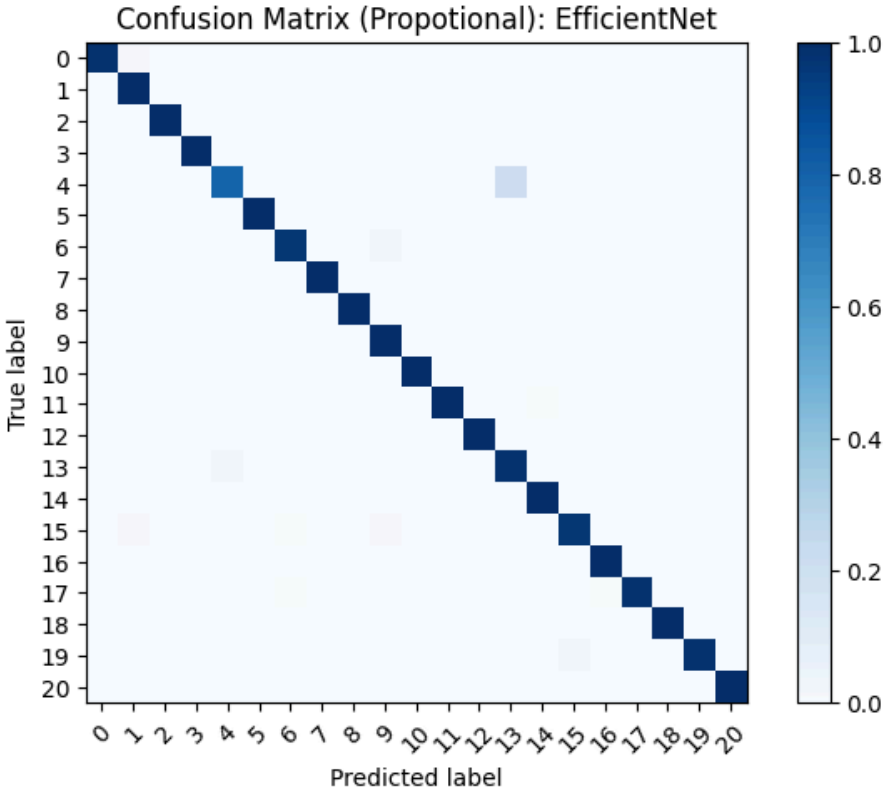

Confusion Matrix (Absolute): Custom
True label
Predicted label
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
800
700
600
500
400
300
200
100
0


Confusion Matrix (Propotional): Custom
True label
Predicted label
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1.0
0.8
0.6
0.4
0.2
0.0

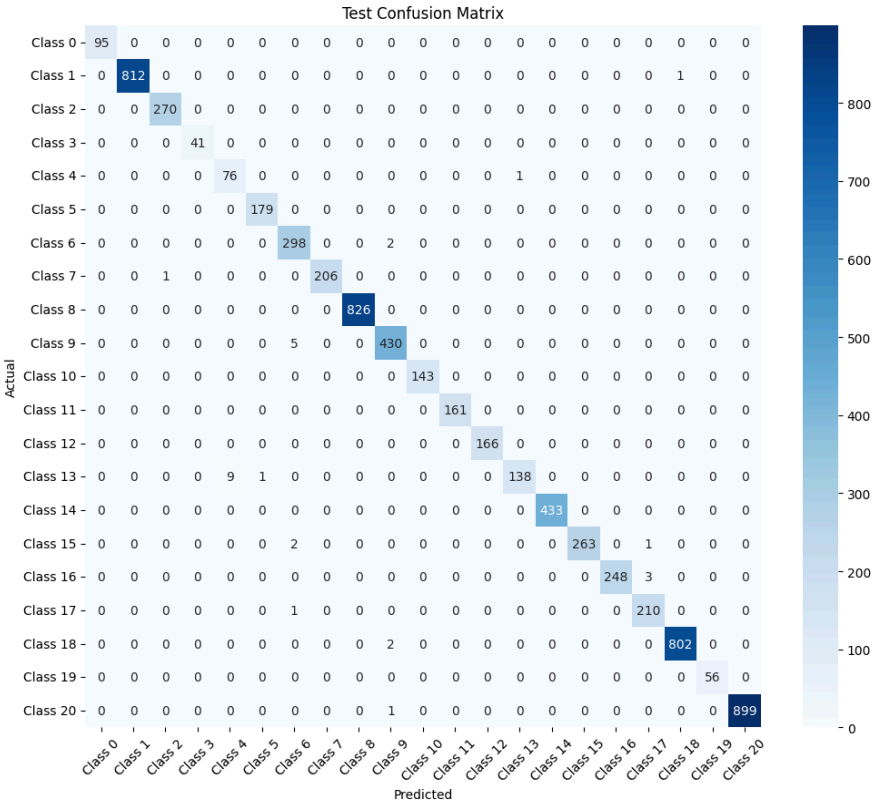

Confusion Matrix (Absolute): ResNet50
True label
Predicted label
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
800
700
600
500
400
300
200
100
0


Confusion Matrix (Propotional): ResNet50
True label
Predicted label
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1.0
0.8
0.6
0.4
0.2
0.0

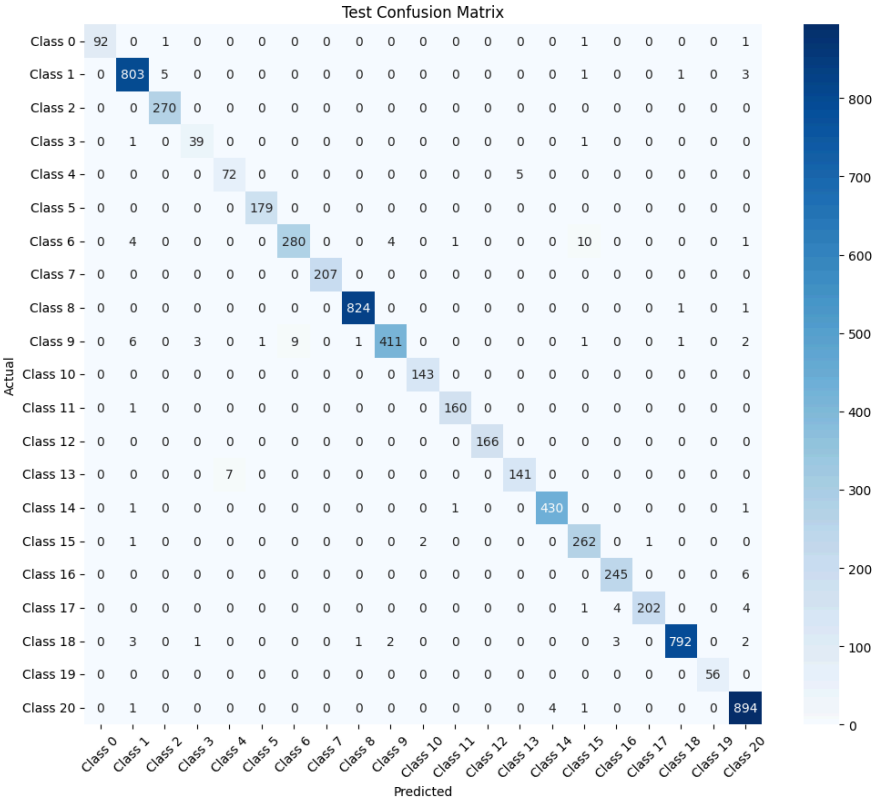

Confusion Matrix (Absolute): EfficientNet
True label
Predicted label
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
800
700
600
500
400
300
200
100
0


Confusion Matrix (Propotional): EfficientNet
True label
Predicted label
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1.0
0.8
0.6
0.4
0.2
0.0

InceptionV3



SqueezeNet



Workload Distribution

Member	Contributions
Isaac Tay Eng Hian	● Image Classification - ResNet50, MobileNetV2, ShuffleNetv2, Custom Model ● Image Segmentation - UNet, UNet++, DeepLabV3 (with ResNet50), MAnet
Natalie Ang Zi Yi	● Data Resampling and Preparation ● Image Segmentation - UNet ● Prototype User Interface
Teh Hui Yi	● Image Classification - SqueezeNet, InceptionNet
Vainavi	● Image Classification - EfficientNet-BO ● Image Segmentation - DeepLabV3 (with mobilenet_v2)

All team members contributed equally to:

- Idea Brainstorming
- Background Research
- Project Idea Selection
- Initial Solution
- Model Selection & Initial Testing
- Integration & Testing
- Documentation - Report Writing and Presentation