

**NANYANG  
TECHNOLOGICAL  
UNIVERSITY**  
**SINGAPORE**

AY 2023-2024 Semester 2

CZ4041 Machine Learning

Group Project

## **Cassava Leaf Disease Classification**

<b>Name</b>	<b>Matriculation Number</b>
Tan Jian Tang Julian	U2020428C
Nyan Maw Htun	U2121694G
Ryan Khong Wei Yang	U2022317A
Dexter Voon Kai Xian	U2120267F
Jovian Nursan Ng	U2020037L

# 1. Introduction

The kaggle project chosen by our group is the Cassava Leaf Disease Classification. The goal of this project is to classify cassava leaves into four different disease categories and the healthy leaf category based on the image dataset provided. With the help of classification, farmers may be able to timely identify diseased plants, potentially saving their crops before they inflict irreparable damage to crop yield. Table 1 below shows the labels and the diseases they are associated with.

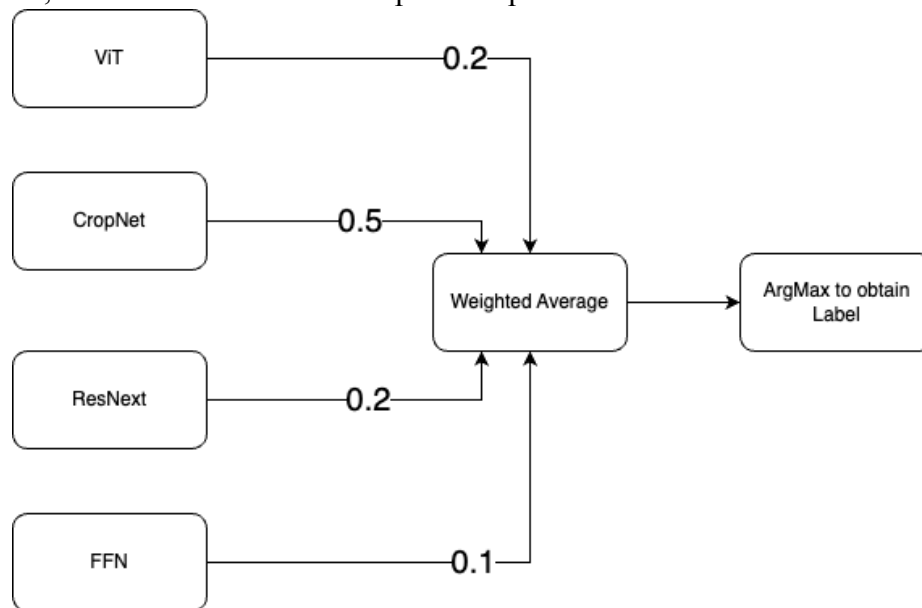
Labels	Diseases
0	Cassava Bacterial Blight (CBB)
1	Cassava Brown Streak Disease (CBSD)
2	Cassava Green Mottle (CGM)
3	Cassava Mosaic Disease (CMD)
4	Healthy

*Table 1: List of Categories*

## 2. Model Architecture

### 2.1 Methodology

The method employed by our group in order to best classify the cassava leaves, was to use an ensemble of pretrained models, as well as our own Feedforward Neural Network trained on the cassava dataset, in order to obtain the best possible prediction of each label for each image.



*Figure 2.1: Illustration of our Model Architecture Ensemble*

Figure 2.1 illustrates our ensemble of models used to obtain our final predictions. In total, we utilised 4 models, 3 pretrained models and 1 of our own trained models. Using the prediction scores at the output of each model, we then took a weighted average based on how important we felt each network was to producing an accurate label. With our final aggregated prediction score, we obtained the final predictions using an Argmax function to return us the labels that will give us the best prediction score. It is noteworthy that although some of the models we tested performed significantly better than others, the benefit of the weighted average approach is that we are able to **harness the network capabilities of different models in identifying the images and place weights on them**. This is as opposed to only using the model with the best validation score, where we perhaps might not be able to tap on the other model's capabilities to capture some intricacies in the data. For example, although CropNet is the best performing standalone model, using it alone means we will not be able to tap on ViT's prediction and model benefits in our final prediction. Hence, we decided to take an ensemble approach to this problem.

## 2.2 CropNet (MobileNetV3)

The first and most specific pretrained model utilized in our ensemble is CropNet. This model architecture is based on MobileNet V3<sup>1</sup>, which is a convolutional neural network (CNN), designed primarily for mobile and embedded devices, where computational resources are limited compared to desktop or server environments. It is an evolution of the MobileNet series, aiming to improve efficiency while maintaining high performance in tasks like image classification, object detection, and semantic segmentation.

CropNet takes in an image as input and outputs the probability of an image belonging to a certain class, there being 5 classes, 4 for each disease and 1 for healthy.

## 2.3 Vision Transformer (ViT) Architecture

Vision Transformer (ViT) is a pure transformer encoder that is directly utilized to sequences of image patches to accomplish image classification responsibilities. We utilized ViT for classifying on Cassava Leaf Disease Dataset. Our Vision Transformer (ViT) model is pre-trained on ImageNet-21k at resolution 224x224, and fine-tuned on ImageNet 2012 at resolution 384x384. We used a base-sized model and a patch size of 16x16 pixels. We used k-fold cross validation where k= 5 with 10 epochs and picked the best model based on validation accuracy.

```
model = ViTForImageClassification.from_pretrained('google/vit-base-patch16-384')
```

Firstly, we split the cassava leaf images into fixed sized patches. These images are then flattened and added with a positional embedding vector. After that, all embedded patches are put into the Transformer Encoder. The output is then passed directly to Feed Forward Neural Network (FNN) to classify and cluster images into five categories: CBB, CBD, CGM, CMD, and Healthy.

---

<sup>1</sup> Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., & Adam, H. (2019, November 20). *Searching for MobileNetV3*. arXiv.org. <https://arxiv.org/abs/1905.02244>

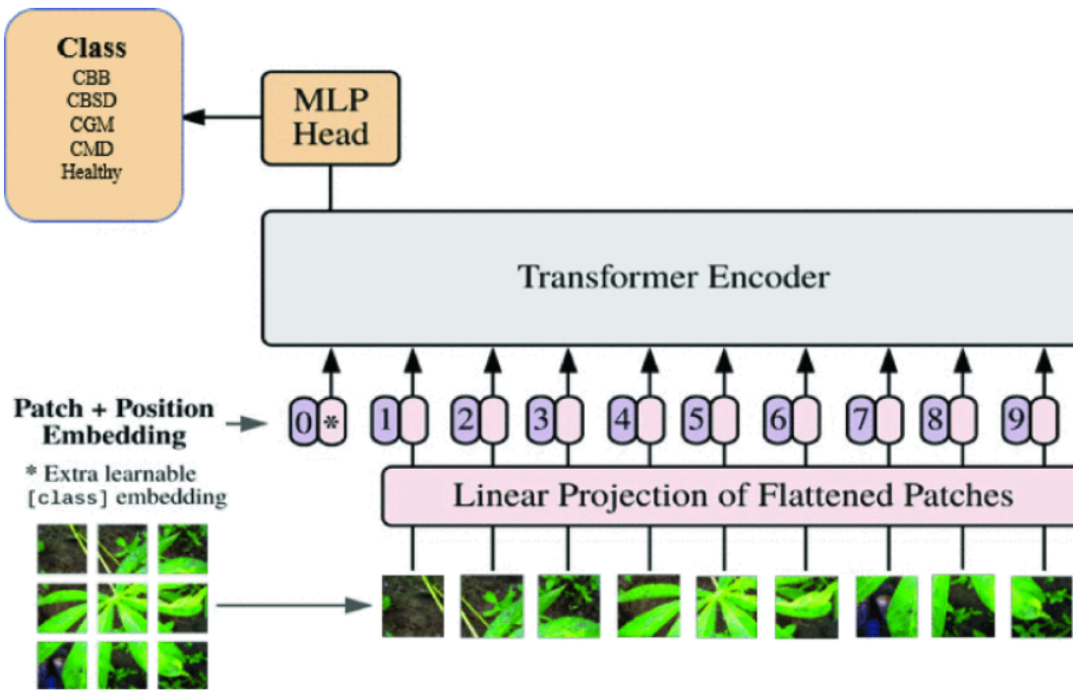


Figure 2.3a Illustration of ViT Architecture. Source<sup>2</sup>

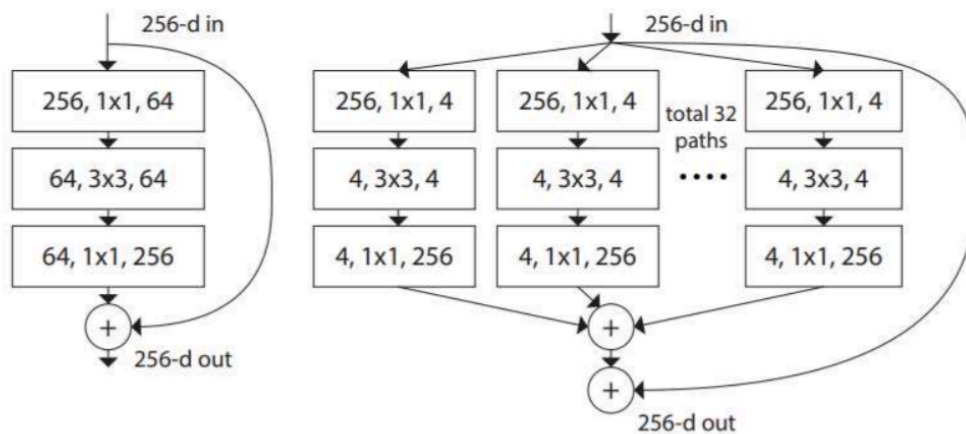
### 2.3.1 Preprocessing

We used ViTImageProcessor to resize (or rescale) and normalise images for the model. Images are resized and rescaled to the same resolution (384x384 during fine-tuning) and normalized across the RGB channels with mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225) since the Vision Transformer requires all images to have the same resolution and size.

<sup>2</sup>H. -T. Thai, N. -Y. Tran-Van and K. -H. Le. (2021, November 15). Artificial Cognition for Early Leaf Disease Detection using Vision Transformers.  
[https://ieeexplore-ieee-org.remotexs.ntu.edu.sg/mediastore\\_new/IEEE/content/media/9598167/9598172/9598303/thai2-p6-thai-large.gif](https://ieeexplore-ieee-org.remotexs.ntu.edu.sg/mediastore_new/IEEE/content/media/9598167/9598172/9598303/thai2-p6-thai-large.gif)

## 2.4 ResNeXt<sup>3</sup>

ResNeXt is a convolutional neural network (CNN) architecture that builds upon the ResNet (Residual Network) architecture, introducing a novel structure for learning feature representations. Traditional CNN architectures often improve performance by increasing the depth and width of the network. However, this can result in higher computational complexity and susceptibility to overfitting. ResNeXt tackles these challenges by introducing a novel concept called "cardinality," which signifies the number of parallel pathways within each ResNeXt block. By enhancing the cardinality, ResNeXt can capture more diverse and nuanced feature representations without substantially increasing computational demands.



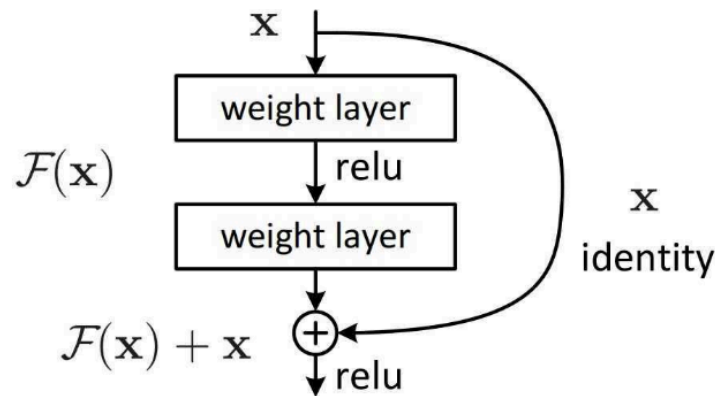
**Left:** A block of ResNet. **Right:** A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

Figure 2.4a In the illustration above, the architecture on the left represents ResNet, while on the right, it is ResNeXt. Source<sup>4</sup>

<sup>3</sup> Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017, April 11). *Aggregated residual transformations for deep neural networks*. arXiv.org. <https://arxiv.org/abs/1611.05431>

<sup>4</sup>Kouidri, A. (2024, December 1). ResNeXt explained: Enhancing cnns with Cardinality & Efficiency. RSS. <https://www.ikomia.ai/blog/resnext-cnn-cardinality-efficiency-explained#:~:text=The%20key%20innovation%20in%20ResNeXt,for%20scaling%20up%20neural%20networks.>

Similar to ResNet's defining feature of residual connections, ResNeXt also performs skip connections to allow for the building of very deep networks by facilitating the gradient's flow during training and avoiding the vanishing gradient problem in deep networks.



*Figure 2.4b In the illustration above, the residual connection creates a shortcut route by adding the value at the beginning of the block,  $x$ , directly to the end of the block ( $\mathcal{F}(x) + x$ ). Source<sup>4</sup>*

In our model, we used k-fold cross validation where  $k = 5$  with 15 epochs. We select the best model with the highest validation accuracy after 15 training epochs.

```
model = timm.create_model('resnext50_32x4d', pretrained=True)
```

### 2.4.1 Preprocessing

Images were resized to the same resolution (512x512) with the help of Albumentations Python library. Similarly, we transformed and normalised the images with mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225).

## 2.5 Feedforward Neural Network

Besides using the aforementioned pretrained models, our team also wanted to train our own model and include it in our ensemble architecture as part of making our solution novel. We experimented with different architectures for this model.

Initially we implemented a Logistic regression layer, with the intention of filtering out just healthy leaves from diseased leaves, using a binary cross entropy loss function. We had the hypothesis that healthy leaves would be much easier to distinguish from the 4 classes of diseased leaves, therefore we wanted to have a model to separate those leaves first. Thereafter, we intended to pass the remainder of leaves that were classified as diseased through our ensemble architecture, making it a 2 phase architecture.

However, this method did not result in a promising validation score. There are several likely reasons for this. First of all, a simple logistic regression layer might be too simple of an architecture to capture underlying features in the data. Secondly, our group felt that this approach restricts us from fully utilising the capabilities of our pretrained models. CropNet is able to classify not only diseases but healthy leaves. Our group felt that if we took the initial approach of using a logistic regression layer, it would be improbable that the model would successfully filter most of the healthy leaves out, and that since CropNet would be capable of distinguishing that, we changed our approach from a 2 phase architecture, to a single ensemble with our own Feedforward Neural Network as part of the weight average prediction score.

### ***2.5.1 Dataset Cleaning and preprocessing***

#### *Cleaning*

When preparing the dataset for training, our group first analysed the dataset to check if dataset cleaning was necessary. Upon our own inspection of the image files, we initially suspected that some of the images could be irrelevant to training because they did not contain cassava leaves, as shown below in Figure 2.5.1. Additionally, there were some images that were prone to a larger amount of noise.





*Figure 2.5.1 Examples of images that did not contain Cassava Leaves*

Hence, our group wanted to remove these images from the training dataset. However, after removing these images, the accuracy of our model did not improve, but conversely worsened. Our group realised that a highly likely reason for this is because these fruits are unique to class “1”: Cassava Brown Streak disease (CBSD). If we remove these images from the dataset, our model might not have enough data to learn on how to classify this disease. For the noisy datasets, our group also did not want to remove them completely from the dataset as if we manage to learn how to identify noisy images, it could improve our validation score. Hence, we changed our strategy from simply filtering out images of fruits and noisy images to a combination of data augmentation and the ensemble method that we employed.

### *Preprocessing*

Images are loaded and preprocessed using ImageDataGenerator, from tensorflow.keras, which allows for on-the-fly image augmentation and preprocessing. The package rescales pixel values using specified parameters that apply augmentation techniques like rotation and flip. The exact values of these hyperparameters are found below in 2.5.2.

The aforementioned images in Figure 2.5.1, i.e. images with Cassava Brown Streak Disease (CBSD), undergo additional augmentation, such as rotation and brightness adjustment. This is done so that there is greater attention placed upon such images. We specify these images by storing the identification numbers of these image files, as seen below in Figures 2.5.2 and 2.5.3

```
# Focus on augmenting CBSD images specifically
cbstd_focus_ids = {str(x)+".jpg" for x in [
    274726002, 9224019, 159654644, 199112616, 226533928, 262902341, 269713568,
    384390206, 390601409, 421035788, 457405364, 600736721, 580111608, 616718743,
    695438825, 723564013, 826231979, 847847826, 927165736, 1004389140, 1008244905,
    1338159402, 1339403533, 1359893940, 1366430957, 1689510013, 4269208386, 4239074071,
    3810809174, 3652033201, 3609350672, 3609986814, 3477169212, 3435954655, 3425850136,
    3251960666, 3252232501, 3199643560, 3126296051, 3040241097, 2981404650, 2925605732,
    2839068946, 2698282165, 2604713994, 2415837573, 2382642453, 2321669192, 2320471703,
    2278166989, 2276509518, 2262263316, 2182500020, 2139839273, 2084868828, 1848686439,
    1689510013, 1359893940
]}
```

*Figure 2.5.2 Storing the file names of images which contain CBSD*

```
def data_preprocessing_function(img, focus_ids, filename):
    img = img_to_array(img) # Convert PIL image to numpy array

    if filename in focus_ids:
        # Apply specific augmentations for focused CBSD images
        # Random rotation
        angle = np.random.randint(-15, 15)
        h, w = img.shape[0], img.shape[1]
        M = cv2.getRotationMatrix2D((w / 2, h / 2), angle, 1)
        img = cv2.warpAffine(img, M, (w, h))

        # Random brightness adjustment
        factor = 0.5 + np.random.random() # Factor between 0.5 and 1.5
        img = cv2.convertScaleAbs(img, alpha=factor, beta=0)

        # Gaussian blur
        if np.random.rand() > 0.7: # Apply with 30% probability
            img = cv2.GaussianBlur(img, (5, 5), cv2.BORDER_DEFAULT)

    img = preprocess_input(img) # MobileNetV2-specific preprocessing
    return array_to_img(img)

def preprocessing_function_wrapper(img):
    # Get access to the filename and apply conditional preprocessing
    return data_preprocessing_function(img, cbstd_focus_ids, img.filename) # Assuming 'img' object has 'filename' attribute
```

*Figure 2.5.3 Data processing for CBSD images*

Generators, train and validation generators, are set up to yield batches of preprocessed images and corresponding labels from a dataframe, with specified target size for resizing.











The Feedforward model employs a Sequential architecture with a Flatten layer to convert 2D image data to 1D vectors, followed by fully connected layers with ReLU activation and dropout for feature extraction and regularisation, and a softmax output layer for multi-class classification.

The model is trained using the Adam optimizer and categorical cross-entropy loss, with callbacks like early stopping and model checkpointing for monitoring and optimization.

## *2.5.2 Hyperparameters of FFN*

Hyperparameter	Value
Rescale	1./255
Rotation range	40
Width shift range	0.2
Height shift range	0.2
Shear range	0.2
Zoom range	0.2
Validation split	0.2
Batch size	32
Seed	42
Learning rate	0.001
Early_stopping patience	10
epochs	50

### 3. Results and Leaderboard performance

 <b>final_submission_cassava - Version 5</b> Succeeded (after deadline) · 2h ago · Notebook final_submission_cassava   Version 5	<b>0.8974</b>	<b>0.9021</b>	<input type="checkbox"/>
 <b>final_submission_cassava - Version 4</b> Notebook Threw Exception (after deadline) · 3h ago · Notebook final_submission_cassava   Version 4			
 <b>final_submission_cassava - Version 3</b> Succeeded (after deadline) · 3h ago · Notebook final_submission_cassava   Version 3	<b>0.8939</b>	<b>0.8978</b>	<input type="checkbox"/>
 <b>final_submission_cassava - Version 2</b> Succeeded (after deadline) · 4h ago · Notebook final_submission_cassava   Version 2	<b>0.8587</b>	<b>0.8602</b>	<input type="checkbox"/>
 <b>cassava_late_submission - Version 11</b> Succeeded (after deadline) · 4h ago · Notebook cassava_late_submission   Version 11	<b>0.7094</b>	<b>0.7158</b>	<input type="checkbox"/>
 <b>final_submission_cassava - Version 1</b> Notebook Threw Exception (after deadline) · 5h ago · Notebook final_submission_cassava   Version 1			
 <b>cassava_late_submission - Version 10</b> Succeeded (after deadline) · 10h ago · Notebook cassava_late_submission   Version 10	<b>0.7094</b>	<b>0.7158</b>	<input type="checkbox"/>
 <b>cassava_late_submission - Version 9</b> Succeeded (after deadline) · 11h ago · Notebook cassava_late_submission   Version 9	<b>0.7641</b>	<b>0.7698</b>	<input type="checkbox"/>
 <b>cassava_late_submission - Version 8</b> Succeeded (after deadline) · 16h ago · Notebook cassava_late_submission   Version 8	<b>0.7908</b>	<b>0.7892</b>	<input type="checkbox"/>
 <b>cassava_late_submission - Version 7</b> Notebook Threw Exception (after deadline) · 16h ago · Notebook cassava_late_submission   Version 7			

*Figure 3.1: Submission score of final\_submission\_cassava - Version 5*

In our final submission of our result, we managed to attain a private score of 0.8974.

## Cassava Leaf Disease Classification

Late Submission ...

Overview Data Code Models Discussion Leaderboard Rules Team Submissions

486	↗ 311	Kun Hao Yeh		0.8975	11	3y
487	↗ 137	Hou Ric		0.8975	19	3y
488	↗ 79	[Deleted] 61b67ce5-07ae-4d07-8e78-94c71ba858d8		0.8975	20	3y
489	↗ 1177	charmz		0.8975	15	3y
490	↗ 369	dhaqui the kaggler		0.8975	3	3y
491	↗ 347	Akihiro Katsura		0.8975	26	3y
492	↗ 305	Chase Yang		0.8975	21	3y
493	↗ 108	arutema47		0.8974	32	3y
494	↗ 170	KYYYYH		0.8974	84	3y
495	↗ 1108	ReNet		0.8974	25	3y
496	↗ 1238	iezhaoliang		0.8974	25	3y
497	↗ 610	blue0620		0.8974	22	3y
498	↗ 362	Zhuang yh		0.8974	4	3y
499	↗ 170	jdxyw		0.8974	57	3y
500	↗ 284	Nick & Hammad		0.8974	79	3y
501	↗ 254	makaibari tea leaves		0.8974	24	3y
502	↗ 53	km		0.8974	27	3y
503	↗ 120	复旦小垃圾		0.8974	78	3y
504	↗ 238	Giving up from climbing LB		0.8974	27	3y

Figure 3.2 Leaderboard standing

On the leaderboard, the score of 0.8974 corresponds to a ranking of 493 out of 3901 submissions. Our relative ranking is an impressive top 12.6%.