RMIT UNIVERSITY

School of Science, Engineering & Technology

COSC2753|Machine Learning Group Project Report

Assignment 2 - Paddy prediction

Lecturer:
Nguyen Thien Bao
Hoang Tuan Anh

Student:
Nguyen Manh Ky Anh - s4029318
Nguyen Phuc Nguyen - s4025306
Nguyen Phu Minh Quang – s3996843

Table of Content

1. Exploratory data analysis

1.1. Metadata integrity

In the metadata file, there are no columns or rows with empty values. There are no duplicate values in the image ID column. In the age column, there are no negative values. Finally, there is no value that is either 'odd' or not in the range of labels given by the requirements.

1.2. Image exploration

For images in the training dataset; By pixels, all images are 480x640. For orientations, all images are portraits, with the exception of four landscape images. Images in the training dataset also have varied brightness and zooms. For metadata; There are 2 categorical columns in the metadata, which is the label (either 'healthy' or the name of a paddy disease) and the variety of the paddy. Both categories contain 10 unique parameters. For labels, the data distribution of unique parameters is highly uneven, with the number of the most common label quintuple the least common label. For variety, the data distribution of unique parameters is also highly uneven, with the number of the most common label sextuple the least common label. Finally, the uneven data distribution is most severe for the age column, as the age ranges from 20 to 80, but there are at most 18 samples of ages in the dataset.

1.3. Data preprocessing

For the metadata, as the categorical features in the metadata are represented in string, label encoder was used to convert them to integers. This helps the tensorflow models interpret different categorical features. Moreover, label encoder also maps the integers to a dictionary, so when required, this dictionary can be used to revert back to string categorical features for humans to understand. For the training images, they must be resized to the same resolution to ensure training consistency and efficiency of machine learning models. Image normalization will also be used to ensure consistency of machine learning models, to reduce training time and to reduce noise in the dataset. Moreover, since the given training dataset is limited to 10.000 images, and the data distribution is highly uneven; data augmentation will be used. For specific details regarding image preprocessing and augmentation, please refer to Image preprocessing and augmentation in the Machine learning section below.

2. Machine learning

2.1. Approach

2.1.1. Data splitting

The training dataset will be divided into 3 fold: 90% is used for training, 5% is used for validating (which is performed in parallel with training), and the remaining 5% - which is never seen by the model - will be used for tuning and testing. Next, to tackle uneven data distribution, stratified data split and redistributing parameters weight will be used.

2.1.2. Stratified split

When using normal train/test split, in most instances, rare parameters will only appear in either training split or testing split. This is detrimental to the model performance, as the model is either not trained to interpret rare parameters, or there are no rare parameters for the model to validate its understanding. By using the stratified split, every generated data folds will contain a sufficient amount of every parameter. However, when creating the image dataset, the image data generator function often randomly redistributes the images. Therefore, to be able to keep the stratified split when creating the image dataset, tensor slice function must be used.

2.1.3. Weight redistribution

Apart from stratified split, parameters weight will also be balanced based on how often it appears in the dataset. Then, this balanced weight is passed to the model only in the training phase. This step increases the model training and testing accuracy by 10%.

2.1.4. Image preprocessing

For reference, when training the model with the original resolution, it takes 3 seconds per step on average, and the model validation accuracy (without image augmentation) is 40%. When considering image downsizing, it takes 20 milliseconds (ms) per step on average, and the model validation accuracy (without image augmentation) is 60%. Therefore, it is mandatory to resize the image to lower values to speed up the training process, and to handle model overfitting. As the dataset is small, images are all resized to 128x128 pixels. Moreover, images are also normalized to range of 0 - 1 for ease of generalization and training efficiency.

2.1.5. Image augmentation

Then, image augmentation is used to tackle the varied image conditions in the training dataset. For classification tasks, brightness and contrast was augmented with a 2% difference margin. For age regression, brightness, contrast, sharpness and saturation augmented with a 3% difference margin. When testing, our group figured that if every augment layer was applied, the model was unable to learn any features. Our group concluded that since the plant images share quite similar features even with different labels, heavy augmentation did not aid in the model training. Below are the similarity histogram of disease parameters for reference.

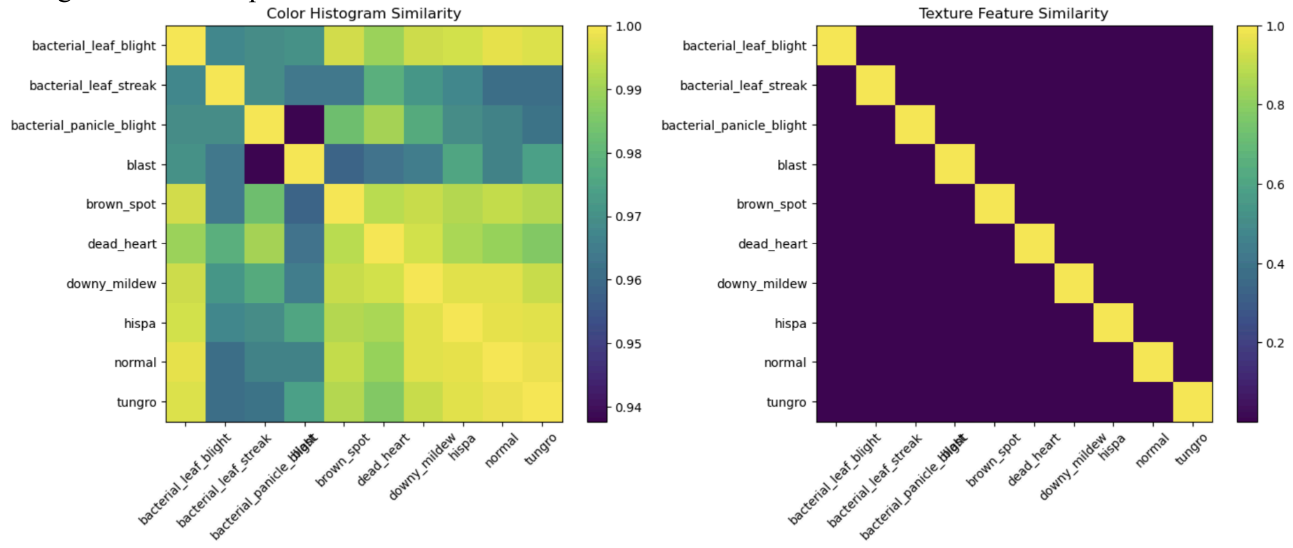


Figure 1. Color and texture similarity histogram

Therefore, the augmentation function will only randomly select one augmentation layer to apply. By applying this step, the model learning curve was smoother, as there is no random large accuracy spike - which indicates the model is overfitting.

2.1.6. Learning rate and optimizer

All models use a learning rate schedule, which makes the learning rate in the first four epochs higher to help the model quickly capture key features. This initial learning rate is 1e-3. Then, the learning rate for the remaining epoch decreased to 1e-4, which allows the model to discover small features. Our group also considered other deeper learning rates, such as 1e-5 and lower; but oftenly, the model overfits to the dataset, rather than learning true features. All three models use the Adam optimizer.

2.2. Training and Testing

2.2.1. Machine learning algorithms

There are three machine learning algorithms that were considered: VGG, EfficientNetB0 and ResNet. All three machine learning algorithms are trained on the same dataset seed, for the same epoch number, and using the same hardware specifications on Google Collab.

2.2.2. Considerations

Before considering raw testing or validating score, there are other aspects that should be considered. VGG is the most interpretable model - which effectively makes VGG the most intuitive to tune hyperparameters. This helps when exploring the suitable epoch, data split, pool size, and l2 regularization. However, VGG is the most 'inefficient' model from the three, as the model is the most data-dependent, generates the most predicting parameters, yet scores the lowest in the performance metric. EfficientNetB0 is the most 'light' model, as it generates the least amount of predicting parameters - which makes EfficentNetB0 the most suitable archetype for mobile devices. However, without pre-trained weight, the model accuracy is affected severely. Finally, ResNet architecture allows each layer to learn from the previous layers instead of learning the input data again. This helps the model train faster, while also increasing the predicting power.

2.2.3. Testing methods

Everytime after a model finishes training, a function is called to plot the learning curve history of the trained model. An instance of the learning curve from a ResNet model can be found below.

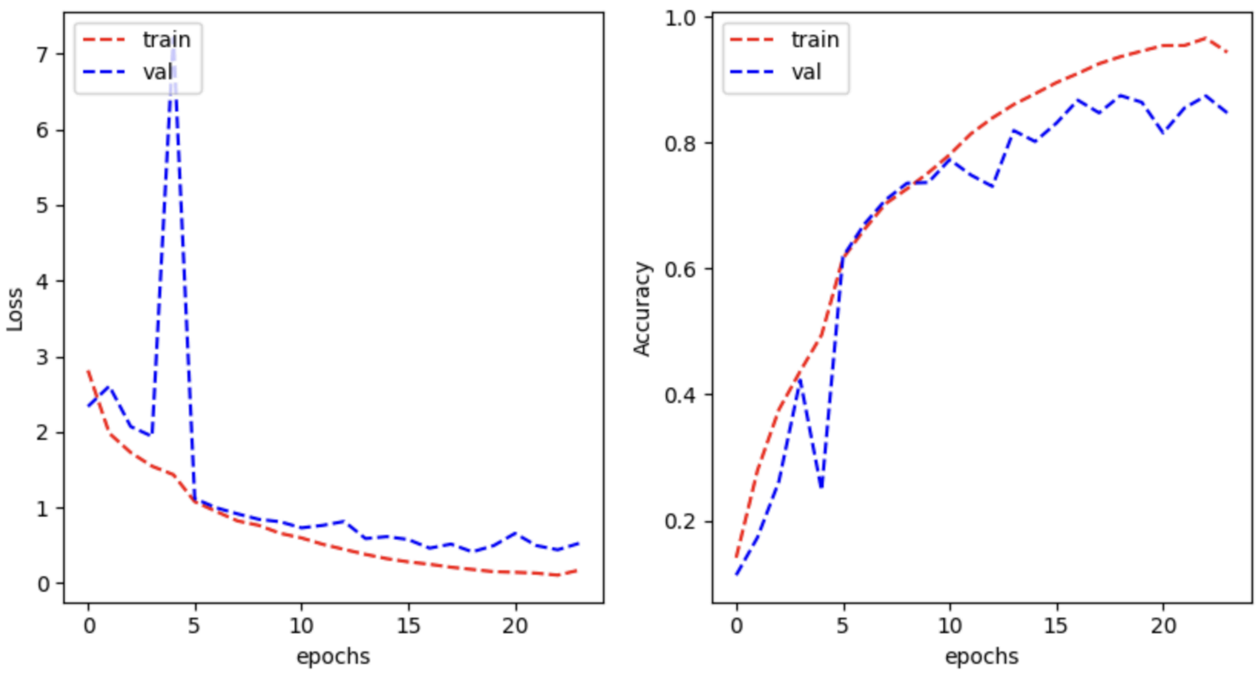
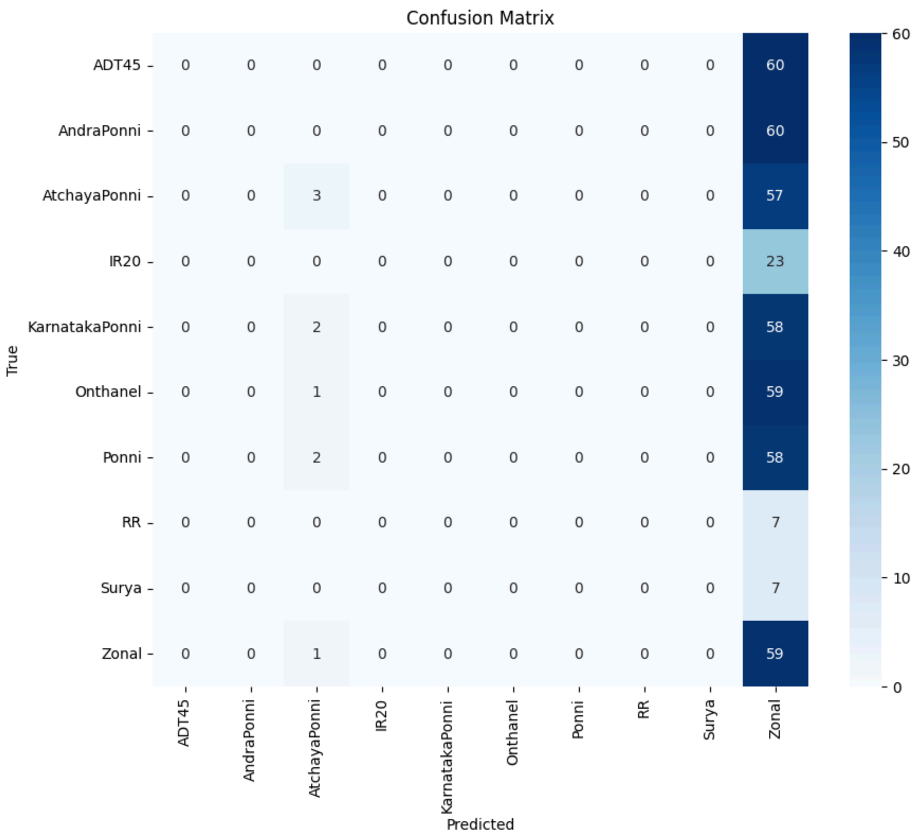


Figure 2. A ResNet model learning curve

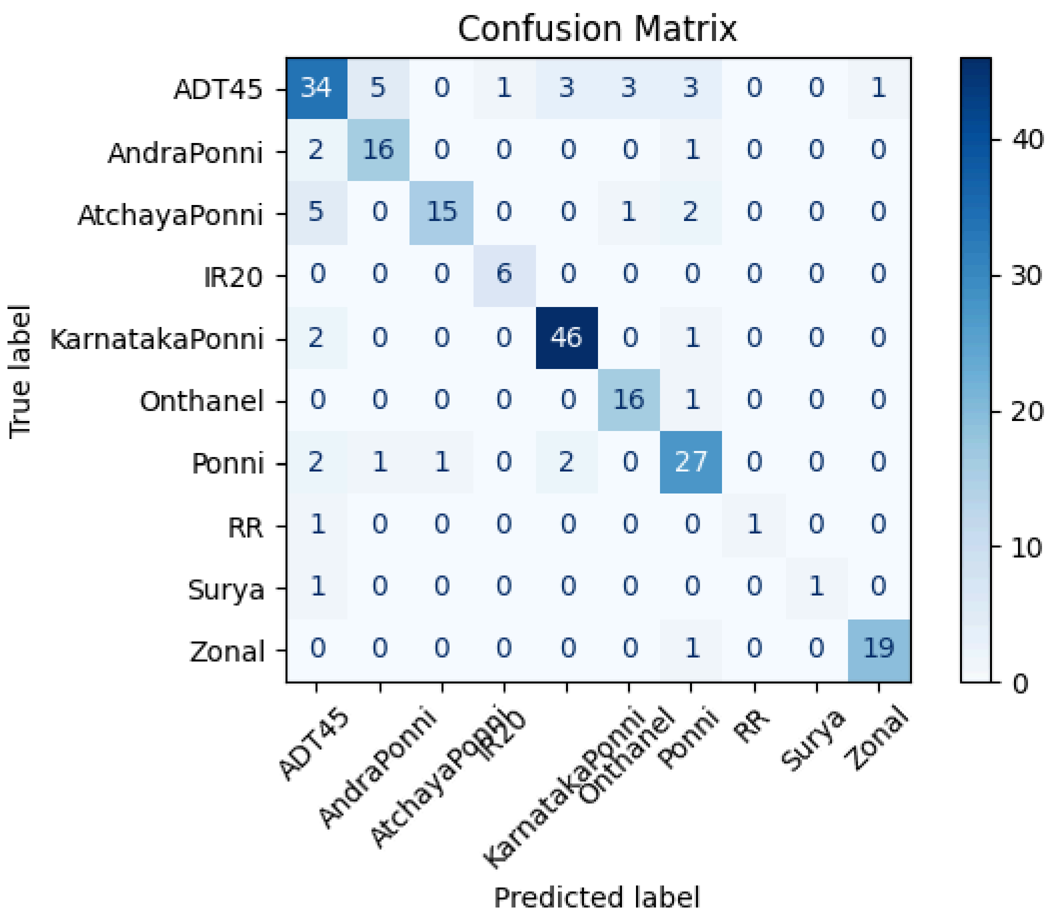
This helps our group detect potential problems. In the example above, the model is having a high learning rate for some first epoch, which could explain the sudden drop in accuracy and 'confidence' of the model at the first 5 epochs. Only after the model learning rate is decreased does the model make stable progression. Next, after any potential performing model has been trained, our group would fit the model to the unseen test dataset and graph its prediction. This helps our group to double-check if the model is underfitting/overfitting, and how the model is performing in general. For reference, below are two before-and-after prediction graphs of an underfitting model that our group was able to detect and adjust based on the generated graph.



Confusion Matrix

True \ Predicted	ADT45	AndraPonni	AtchayaPonni	IR20	KarnatakaPonni	Onthanel	Ponni	RR	Surya	Zonal
ADT45	0	0	0	0	0	0	0	0	0	60
AndraPonni	0	0	0	0	0	0	0	0	0	60
AtchayaPonni	0	0	3	0	0	0	0	0	0	57
IR20	0	0	0	0	0	0	0	0	0	23
KarnatakaPonni	0	0	2	0	0	0	0	0	0	58
Onthanel	0	0	1	0	0	0	0	0	0	59
Ponni	0	0	2	0	0	0	0	0	0	58
RR	0	0	0	0	0	0	0	0	0	7
Surya	0	0	0	0	0	0	0	0	0	7
Zonal	0	0	1	0	0	0	0	0	0	59

Figure 3. Model A underfitting in the unseen testing data



	ADT45	AndraPonni	AtchayaPonni	IR20	KarnatakaPonni	Onthanel	Ponni	RR	Surya	Zonal
ADT45	34	5	0	1	3	3	3	0	0	1
AndraPonni	2	16	0	0	0	0	1	0	0	0
AtchayaPonni	5	0	15	0	0	1	2	0	0	0
IR20	0	0	0	6	0	0	0	0	0	0
KarnatakaPonni	2	0	0	0	46	0	1	0	0	0
Onthanel	0	0	0	0	0	16	1	0	0	0
Ponni	2	1	1	0	2	0	27	0	0	0
RR	1	0	0	0	0	0	0	1	0	0
Surya	1	0	0	0	0	0	0	0	1	0
Zonal	0	0	0	0	0	0	1	0	0	19

Figure 4. Model A starting to perform better in the unseen testing data

2.2.4. Testing results

Below is the table for comparing some matrics of each model architecture.

	VGG	ResNet	EfficientNetB0
Model storage size	40Mb	9Mb	5Mb
Training time (time per step)	300ms	40ms	18ms
Training accuracy	0.80	0.92	0.74
Validation accuracy	0.76	0.89	0.64
Testing accuracy	0.77	0.88	0.70

Figure 5. Evaluation table of selected model

2.3. Evaluation

Based on the evaluation table above, it is clear that EfficientNetB0 is the 'lightest' model. However, since EfficientNetB0 is designed to train in relation with pre-weights like 'ImageNet', the accuracy drawback is quite evident, as pre-weight training was not used pursuant to the project requirements. The VGG model has higher accuracy compared to EfficientNetB0, but the model takes up the most space and takes the most time for each step - which effectively makes VGG the least efficient model. Finally, ResNet is the highest performing model in terms of accuracy, since the model utilizes 'residual' information from the last layer to train and make predictions. The model is also lightweight, as the 'residual' information is used to reduce the size of the input when training and prediction also. However, pre-built ResNet is often hard to interpret. Therefore, our group came up with a solution: a VGG-based model with residual net implementation. This allows custom and intuitive layer definition, which can tailor to the specific limited nature of this dataset. Moreover, since pre-trained weight and transfer learning is not allowed, the layers could also be reduced to cope with the requirements. Finally, the model is lightweight, while also not compromising on accuracy due to the residual net in play. For details regarding the model prediction accuracy on unseen testing dataset, please refer to Figure 6 in the appendix. With the final model implemented, our group has also developed a Graphical User Interface (GUI) for users in general and farmers specifically to interact and use our model without requiring low-level knowledge of the terminal or any programming language. The GUI is a web interface, ensuring usability and portability with an open standard that could be accessed from any device with internet. It communicates with the pretrained models for prediction through a FastAPI server with RESTful JSON response, opening the opportunity for other clients, such as mobile applications to connect to the same server.

3. Discussion

3.1. Challenges and limitations

3.1.1. <u>Dataset problem</u>

In general, for every image machine learning problem, the quality of the dataset is the most important factor when it comes to the performance of the model. However, since the dataset quality is often based on the labeler, not the engineers; it is often impossible to augment or preprocess the image to the highest quality, if the source of the data is invalid at the beginning. For the given dataset for this project, assume the data label is 100% valid - which is an extremely rare case, the dataset size is too small in comparison with other image machine learning projects. Moreover, the image label distribution is often uneven, which leads to bias and unfair prediction in the mode. This is even more evident in the regression problem. Finally, when deploying the model for use, it is almost improbable that the input images will have a similar 'pattern' like the training, or even the testing dataset. This will affect the performance of the model in a real-world scenario. Thus, one way to tackle this problem is to minimize the error of data labeling right from the beginning. Moreover, increasing the amount of images is also preferred over image augmentation. Finally, in a scenario where the techniques involved in training the model were not limited, it is always better to utilize pretrained weight from sources like Imagenet. Ensembling, which involved more than three models could also be utilized for a more generalized, unbiased and accurate prediction. Finally, transfer learning could also be used, as mentioned in [4] that transfer learning increases the accuracy of the model by 17%.the model architecture should prioritize normalization and regularization.

3.2. Related work

One work with similar data characteristics and goals that our group could find was from [2] by Charles O'Neil. While our group project and Charles both utilizes these following techniques: cyclical learning rate and slight data augmentation; Charles utilizes a variety of other techniques, such as progressive resizing - where the image input size is progressively different to help with generalization, model ensembling - where a variety of model was used to predict a single output, and mixed-precision training - where computations is performed in both 32 float and 16 float interchangeably, as proposed in [3]. Collectively, Charles' project was able to achieve a 98% accuracy in both training and testing sets. However, our group did not include the three aforementioned techniques, as due to the limitations of our Google Collab setup, and due to the limitation of the project requirements.

3.3. Final decision

In conclusion, our group has developed a VGG-based model with residuals to tackle the regression and classification requirements; as well as coping with the specific nature of the requirements, such as limited dataset, no pre-trained weight and no transfer learning. Moreover, our group has also developed a GUI for users to interact and use the model without requiring any technical skills such as Python or terminal interaction.

4. Reference

[1] Leslie N. Smith, “Cyclical Learning Rates for Training Neural Networks”. U.S. Naval Research Laboratory, 2017. [Online]
Available: https://arxiv.org/pdf/1506.01186
[Accessed: May 13, 2025].

[2] Charles O’Neil, “Rice Paddy Disease Classification Using CNNs”. [Online]
Available: https://arxiv.org/pdf/2303.08415
[Accessed: May 8th, 2025].

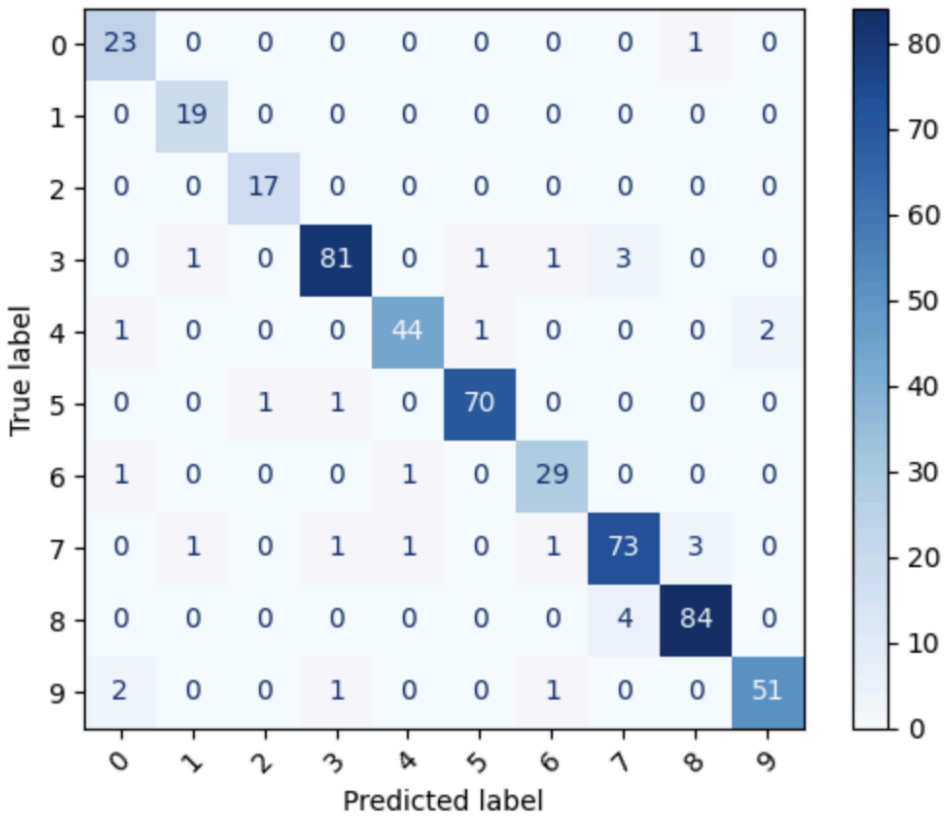
[3] Sharan Narang* , Gregory Diamos, Erich Elsen, Paulius Micikevicius* , Jonah Alben, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, Hao Wu; “Mixed precision training”. ICLR, 2018.
Available: https://arxiv.org/pdf/1710.03740
[Accessed: May 13th, 2025].

[4] Ahad, Md Taimur, Li, Yan, Song, Bo and Bhuiyan, Touhid; “Comparison of CNN-based deep learning architectures for rice diseases classification”. China: Elsevier, 2023.
Available:
https://research.usq.edu.au/download/748987f6260d4fa19fe0906c41d43a8c58bf737bf85dcb0764ac52b29dc84adc/2417933/1-s2.0-S2589721723000235-main.pdf
[Accessed: May 8th, 2025].

[5] Keras, “Image augmentation layers”. [Online].
Available: https://keras.io/api/layers/preprocessing_layers/image_augmentation/
[Accessed: May 6th, 2025].

5. Appendix

Figure:



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	23	0	0	0	0	0	0	0	1	0
1	0	19	0	0	0	0	0	0	0	0
2	0	0	17	0	0	0	0	0	0	0
3	0	1	0	81	0	1	1	3	0	0
4	1	0	0	0	44	1	0	0	0	2
5	0	0	1	1	0	70	0	0	0	0
6	1	0	0	0	1	0	29	0	0	0
7	0	1	0	1	1	0	1	73	3	0
8	0	0	0	0	0	0	0	4	84	0
9	2	0	0	1	0	0	1	0	0	51

Figure 6. ResNet + VGG inspired model accuracy on unseen testing dataset