Early detection of Ugandan cassava brown streak virus with machine learning and classification techniques

1. Introduction

Cassava brown streak disease (CBSD) is a damaging disease of cassava plants that greatly affects cassava production. There are two closely related RNA viruses that cause CBSD, one being Ugandan cassava brown streak virus (UCBSV) and the other being Cassava brown streak virus (CBSV). In previous studies [1], a handheld multispectral imaging device was designed to collect multispectral images for infected and uninfected UCBSV leaf samples. Four trials of multispectral images of leaf samples infected and uninfected with UCBSV have been gathered from the experiment. They are named as Experiment 6 (E6, containing 18 plant samples), Experiment 8 (E8, containing 18 plant samples), Experiment 11 (E11, containing 6 plant samples) and Experiment 12 (E12, containing 12 plant samples). In all the four trials, cassava cultivar TME204 were being used and within all plant samples, 14 waveband images of leaf samples were captured.

The objective of this report is to conduct a more detailed investigation into the various machine learning and image processing techniques that could potentially enhance the detection accuracy of these leaf samples infected by UCBSV. Various image pre-processing methods such as segmentation and patch cropping will be employed; two methods of extracting the features will be compared: the first involves taking the average over each wavelength range of patches, while the second utilized a well-tuned Convolutional Neural Network (CNN) model. The extracted features will then be fed into different classifiers for prediction. The results of both intra-trial and cross-trial will be presented.

2. Methods

2.1 Cassava leaf acquisition:

Within the four trials, cassava leaves were detached from TME204 plants at 14, 28, 54 days post inoculation (dpi) in E6; at 7, 14, 28, 52 dpi in E8; at 28, 56, 74, 94 dpi in E11 and at 28, 63, 114 dpi in E12. The detailed experiment design can be viewed from Table 1 below.

Table 1. Experiment design of the 4 trails for cassava TME204

Trial	Days post inoculation (dpi)	Number of uninfected leaf samples	Number of infected leaf samples (UCBSV only)	Number of bands per leaf sample
E6	14	18	18	14
	28	17	18	14
	54	18	18	14

E8	7	18	18	14
	14	18	18	14
	28	18	18	14
	52	18	18	14
E11	28 (original scanner)	6	6	14
	56 (original scanner)	6	6	14
	74 (original scanner set 1)	6	6	14
	94 (original scanner)	7	7	14
E12	28 (0 hr.)	12	12	14
	63	11	12	14
	114	12	12	14

2.2 Data preprocessing:

Following the capture of multispectral scans of cassava leaf images, a process of segmentation will be initiated with the objective of removing the veins and patch cropping where 9 patches with size of 40x40 will be cropped out at random spatial locations on each leaf scan. A sample image of cropped patches can be seen on Figure 1 below.

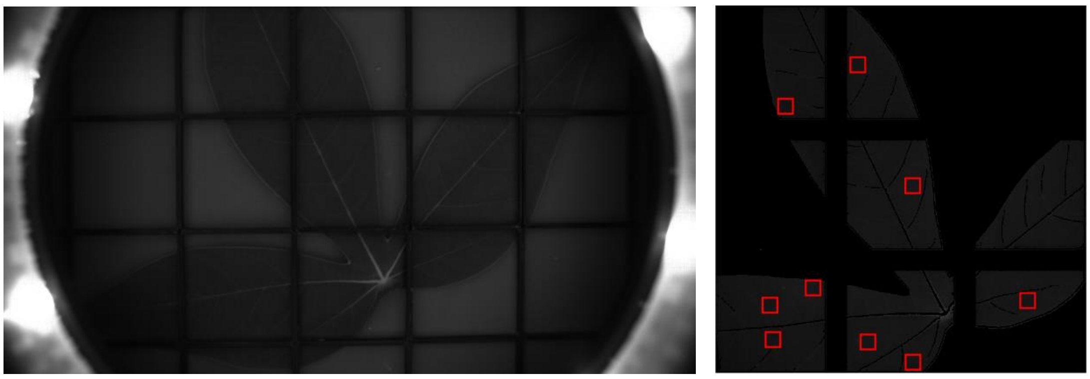

Figure 1. Original leaf sample of one band in Trial 6, 14 dpi, uninfected dataset (Left) and randomly selected 9 patches with size 40x40 cropped on the leaf areas of the same leaf samples (Right)

2.3 Feature extracting:

1. Averaging: one of the attempts to represent the spatial features is to take the average of the number of pixels of every patch over each wavelength range. Therefore, 9 features will be calculated from every waveband image (one per patch) and 126 features per leaf samples. These features will then be fed into a designed classifier.
2. CNN model: another attempt of extracting the features is by building a Convolutional Neural Network (CNN) model. However, CNN models are typically not effective at predicting unseen data when there are insufficient training samples. This is primarily due

to CNNs' tendency to memorize the training data, which may lead to overfitting. To address this issue, an alternative approach involves training the model with input samples, and then replacing the last fully connected layer with a designed classifier. This classifier is subsequently used to making predictions. This method is promising as classifiers such as Support Vector Machines (SVMs) excel at handling small datasets with their strong regularization capabilities, which may prevent overfitting on the datasets. The last layer before the fully connected layer will be viewed as the feature layer. Details of the design of the custom CNN model can be seen in Table 2. Such model is designed with repeated tests to carefully tune the parameters in order to prevent overfitting.

Table 2. Design of the custom CNN model

Convolutional Neural Networks Configuration
5 weight layers custom model
40x40x14 input multispectral images
Conv2-32 (L2 regularization = 0.005, kernel initializer=he normal)
Batch Normalization + PReLU + MaxPooling
Conv2-64 (L2 regularization = 0.005, kernel initializer=he normal)
Batch Normalization + PReLU + MaxPooling
Conv2-128 (L2 regularization = 0.005, kernel initializer=he normal)
Batch Normalization + PReLU + MaxPooling
Flatten
Dropout 0.5
FC-256
Dropout 0.5
FC-128
Sigmoid

Kernel regularization: It is applied to all of three 2D convolutional layers using L2 regularization, with the regularization strength set to be 0.005. This serves to prevent overfitting by introducing a penalty for large weight values in the loss function and in this L2 regularization, such penalty has the value of 0.005 multiply by the square of magnitude of the weights and will be added to the original loss. The number 0.005 is selected from repeated turning to enable more precise control over the strength of the penalty, preventing both over-regularization and under-regularization.

'he normal' Kernel initializer: It is used to initializes the weights with values drawn from a truncated normal distribution with a mean of 0 and a standard deviation proportional to the number of inputs in the previous layer. By applying this initializer to the activation function, it will help maintain the variance of activations constant across layers, which therefore prevents the issue of having too many zeros in the extracted features, also known as the 'dying' neurons problem caused by overfitting. In other words, the use of 'he normal' initializer allows more useful features with non-zero values to be extracted from the feature layer.

Parametric ReLU (PReLU): The application of PReLU is also an attempt to solve the 'dying' neuron problem in the extracted feature layer. It is a variation of ReLU that introduces a learnable parameter for the slope when the input is negative. Its function can be viewed below in Equation 1.

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha \cdot x & \text{if } x \leq 0 \end{cases} \quad (1)$$

Where α is a learnable parameter that determines how much negative input is passed through the function, in our case, it is set to be 0.25 as default. Unlike ReLU, where negative values are completely disregarded, PReLU retains some information from negative activations, which can help remain a lot of information for negative neurons and therefore, more non-zero features can be extracted from the feature layer.

Batch Normalization: This is applied after each convolution and activation function to stabilize and accelerate the training process. It works by normalizing the activations of a layer to have a mean of 0 and a standard deviation of 1, based on the statistics of the current batch. This normalization step helps address internal covariate shift, where the distribution of inputs to a layer changes during training, causing instability.

Dropout layers and the feature layer: The two dropout layers are placed above and below the fully connected layer named FC-256. This is a regularization technique to randomly ignore a certain percentage of neurons, in this case, the dropout rate is set to be 0.5, meaning that 50% of neurons will be randomly ignored while training. By doing so, the network can be less sensitive to the specific weights of individual neurons and learn to be more robust. 128 features will then be extracted from the feature layer below, the value 128 is set to capture a richer set of patterns, through the tuning of the model, feeding 128 features to the SVM classifier will result in an increase in the final prediction accuracy of around 5%.

Validation loss checkpoint: The final strategy to prevent overfitting is by applying a validation loss check point to the model. This act will save the weights of the model with the smallest validation loss. In this experiment, it will pick up the weights with smallest validation loss in the 100 training epochs. Therefore, after training, features can be extracted from such model as it will have the best performance for unseen data.

Overall, the design of this custom model is to improve the robustness and preventing too many zero values in the extracted features. Its performance will be examined together with some other CNN models by connecting a patch-based (with no further processing of patches from the same leaf sample) SVM classifier after those features are extracted from the training and testing data. Its prediction accuracy will be listed below in Table 3. This is the accuracy using default parameters of the SVM classifier (Regularization parameter C=1.0, kernel=rbf, degree=3 and gamma=scale) and purely patch-based to give a quick approximate estimation of the final accuracy. It is used to identify how well features are extracted from those CNN models and prove that the custom model is a well-designed model specifically for this problem. Noticing that the model will be repeated training and testing 3 times (100 epochs in each run) to get the mean and standard deviation of the test accuracy. Training and testing dataset are being separated by a method called Leave-One-Group-Out Cross-Validation, further explanation of this method can be found on *Section 3.2 Cross-trial results*.

Table 3. Patch-based SVM predictions of the features extracted from different CNN models

Training data	**Testing data (%)**	**8 weight layers Alexnet (%)**	**16 weight layers VGG16 (%)**	**2D-CNN (%)**	**5 weight layers custom model (%)**
E8, E11, E12	E6	87.59 ± 2.40	91.98± 0.45	90.57 ± 2.03	91.72 ± 0.72
E6, E11, E12	E8	75.80 ± 1.47	72.26 ± 2.58	70.74 ± 0.89	75.24 ± 1.16
E6, E8, E12	E11	47.78 ± 3.75	53.78 ± 1.04	53.11 ± 0.12	56.42 ± 2.39
E6, E8, E11	E12	53.46 ± 2.25	54.09 ± 1.62	53.46 ± 0.15	78.26 ± 1.63

* Details about the designed structure of Alexnet, VGG16 and 2D-CNN can be found in *Section 7. Appendix.*

From Table 3, the first three approaches of applying proven CNN structural designs to extract features perform adequately on training sets E8, E11, E12 and testing set E6, all results are approximately 90%, where VGG 16 and custom model have relatively the best prediction accuracy of 91.72% ± 0.72%. These findings suggest that this dataset is already well-optimized. In contrast, for results from training on E6, E11, E12 and testing on E8, Alexnet and custom model has the best accuracy of around 75%. This indicates that with a proper design of classifier, this dataset has the potential for improved performance. For results from training on E6, E8, E12 and testing on E11, the lowest prediction accuracy was observed, suggesting that none of the classifiers extracted the features very well. This could be attributed to noise interference during the capture of multispectral images of leaf samples in E11, which significantly impacted the classifier's predictions. Further proofs are provided in *Results Section*. In terms of model performance, the custom CNN model continued to outperform others, with a prediction accuracy of 56.42%. When evaluating results from training on E6, E8, E11 and testing on E12, only the custom model can reach an accuracy of 78.26%, which was approximately 45% higher than the other models. The reason for this improved performance is that other models tended to overfit while sampling this group of datasets. However, the custom model, by incorporating ‘he normal’ Kernel initializer, PReLU, and L2 regularization, significantly reduced overfitting and subsequently produced a better prediction after being connected to the SVM classifier. Overall, since the custom model has the best prediction accuracy for all groups of datasets, the features extracted from this model will then be fed into other well-designed classifiers.

2.4 Classifiers:

Both methods of extracting features, *Averaging* and *custom CNN model*, require a well-designed classifier to make the final predictions. In this report, 4 different designs of the classifier will be listed. Noticing that three of the classifiers listed below are designed for the cross-trial experiment, for the Intra-trial experiment, only the first classifier, which is the fundamental SVM classifier will be used as the aim of intra-trial experiment is to examine the quality of each dataset and prove that different designed models are able to distinguish between infected samples and uninfected samples. As a result, except the intra-trial of E11, all other intra-trial experiments reached a prediction accuracy of above 90% with the fundamental classifier. This suggests that the other experimental groups contain leaf samples of high quality and the fundamental SVM classifier by itself is able to distinguish infected and uninfected leaves easily. Therefore, there is no need to further apply other classifiers to these samples as these classifiers are all improvements based on the fundamental SVM classifier, they can be used to solve more complex cross-trial classification problems. All of the four designs of classifiers are listed below:

1. Fundamental SVM classifier: This SVM classifier is the fundamental design of all the other classifiers designed. Its implementation involves combining all patch-generated features in one leaf together, instead of treating each patch individually. This will enable the classifier to learn from the dataset in a cohesive manner and therefore generate a leaf-based predictions. A threshold of 5 is also used in the leaf-based prediction, meaning that if 5 out of 9 patches in one leaf sample are predicted correctly, the prediction will be viewed as correct prediction for this specific leaf sample. Additionally, grid search is employed to explore a wide range of parameters to better tune the SVM classifier and diffusion matrix is utilized at the end to evaluate the classifier’s performance. They are listed in Table 4 below. Accuracy is calculated as the ratio of predicted samples to the total number of samples as shown in Equation 2. For features calculated from *Averaging*, this classification process will be iterated 300 times to get the average accuracy of the prediction, therefore, allowing us to better assess the stability and robustness of this model. For features extracted from *CNN models*, due to significant computational power required to train and test them, 30 times of iterations

will be applied to the classification. This represents a trade-off between time and accuracy.

Table 4. Description of confusion matrix

Confusion matrix	Description
True Negatives (TN)	Correctly predicted negative samples
False Positives (FP)	Incorrectly predicted positive samples
False Negatives (FN)	Incorrectly predicted negative samples
True Positives (TP)	Correctly predicted positive samples

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

2. SVM classifier with normalized features: If some of the features are extremely larger than others, this will cause the problem of such kind of features dominating the rest, leading to too much weight biased to them. The problem can be prevented by applying standard normalization techniques to the features. Its equation can be shown below in Equation 3.

$$X_{normalized} = \frac{X - \mu}{\sigma} \qquad (3)$$

   Where, X is the original feature, μ is the mean of the features for each feature patterns, σ is the standard deviation of the features and $X_{normalized}$ is the normalized features. After standard normalization, each feature pattern will be scaled to have a mean of 0 and a standard deviation of 1. After applying standard normalization to all training features, they are subsequently fed into the fundamental SVM classifier. It is important to note that the normalization parameters are retained and applied to the testing features, ensuring that the testing data is scaled consistently with the training data.

3. Ensemble learning: Two techniques of ensemble learning is employed: bagging and boosting. Bagging classifier aims to reduce variance by averaging the predictions of multiple models. Each model, referred as an estimator, will be trained on different subsets of the training data and the final predictions will be made by majority vote. For the purpose of this experiment, 10 SVM classifiers will be constructed, each trained on a distinct random subset of the training features, with the final aggregated predictions made based on majority voting. The second technique used is the boosting classifier, specifically Adaptive Boosting (AdaBoost). It seeks to reduce both bias and variance by sequentially training models, with each model designed to correct the errors of the previous one. The algorithm adjusts the weight of each training feature based on whether it was correctly classified in previous iterations, placing greater emphasis on misclassified features. In this experiment, decision trees will be used as the base classifier for AdaBoost, alongside SVM due to its greater interpretability, making it easier to analyze and understand the impact of the model's predictions. In both bagging and boosting implementations, leaf-based prediction design, diffusion matrix and prediction accuracy will be constructed the same as the

fundamental SVM classifier.

4. Isolation forest: It is a machine learning algorithm that detects outliers in the dataset by constructing binary trees. It works by randomly partitioning the data points, isolating data points based on how easily they can be separated from the rest of the data. In this experiment, the number of trees in the isolation forest is set to be 300 to ensure stable detection of the outliers. Once detected, these outliers will then be replaced by the median values of each feature patterns. The median is chosen because it is a robust statistic that is not significantly influenced by extreme values, making it an ideal replacement for outliers. This act will neutralize the impact of outliers and avoid skewing the dataset. The processed training data will then be connected to the fundamental SVM classifier to make the final prediction.

3. Results

3.1 Intra-trial results

The aim of measuring intra-trial results is to examine how well the model perform within a sample group. The design of the intra-trial model is the fundamental SVM classifier, with 2 leaf samples randomly selected from the dataset as test sample, and the rest being training samples. Such design is also able to examine how well the features are extracted from the two different methods, *Averaging* and from *CNN models*. Four intra-trail experiment will be conducted utilizing these two methods. They will be E6 intra-trial, E8 intra-trial, E11 intra-trial and E12 intra-trial. Their results are listed below in Table 4. All experiment will be run for 300 iterations and repeated 3 times to get the average value of prediction accuracy and standard deviation.

Table 4. Intra-trial prediction accuracy from the fundamental SVM classifier

	Prediction accuracy for features calculated from *Averaging* (%)	**Prediction accuracy for features extracted from the custom CNN model (%)**
E6 intra-trial	98.26 ± 0.10	90.86 ± 0.33
E8 intra-trial	97.89 ± 0.31	80.50 ± 0.71
E11 intra-trial	59.93 ± 0.14	63.67 ± 0.94
E12 intra-trial	92.32 ± 0.72	80.60 ± 1.88

The table shows that the intra-trial prediction accuracies on dataset E6, E8 and E12 for features calculated from *Averaging* all reached above 90%, with E6 having the best prediction accuracy of 98.26%. Although for features extracted from the CNN model, the accuracy for these trials all dropped by around 10%, 18% and 13% respectively, this could be due to the fact that CNN features are harder to classify and require a more complicated classifier to train. The later cross-trial experiments on dataset E6, E8 and E12 with the implementations of more complex methods suggest that with a properly designed method added to the fundamental SVM classifier, the cross-trial prediction accuracy for CNN extracted features will be able to improve. Returning to the intra-trial results, although there might be some differences of prediction accuracies between

Averaging and *custom CNN model* extracted features, for dataset E6, E8 and E12, neither experiment is lower than 80%. This suggests that the multispectral images captured in these experiments are clean enough to be able to identify each other within the trials, leading to the conclusion that both methods of extracting features are feasible and robust. However, for dataset E11, the fundamental SVM classifier is not able to identify the features extracted by either method well. The accuracy is only around 60% for both methods. This indicates that this experiment may have been conducted in a relatively noise environment where the multispectral imagines being captured varied a lot that even within the same dataset, they are hard to be classified. The Cross-trial results in *Section 3.2* for dataset E11 are also able to prove this assumption.

3.2 Cross-trial results

To evaluate how well the model predicts on new samples from the provided dataset, a strategy called Leave-One-Group-Out Cross-Validation is employed. The method is similar to K-fold cross-validation, but instead of choosing a number of folds (k), it sets aside one entire group as the validation dataset while using the other groups for training. By doing this repeatedly with different groups, a robust and unbiased estimate of how well the model will perform on new data will be gained. In our case, there are 4 groups of data, therefore, there will be 4 cross-validation tests run for each model. More specific experiment groups will be listed below in Table 4.

Table 4. Leave-One-Group-Out Cross-Validation experiment design

	Train data	**Test data**
Group 1	E6, E8, E12	E11
Group 2	E6, E8, E11	E12
Group 3	E8, E11, E12	E6
Group 4	E6, E11, E12	E8

The experiment accuracy results for features extracted from *Averaging* are provided below in Table 5. All the different classifiers will be compared. Each experiment will be run for 300 iterations and repeated 3 times to get the average value of prediction accuracy and standard deviation.

Table 5. Cross-trial prediction accuracy results for features calculated from *Averaging*

	Fundamental SVM classifier accuracy (%)	**SVM classifier with normalized features accuracy (%)**	**Ensemble learning (Bagging) accuracy (%)**	**Ensemble learning (Boosting) accuracy (%)**	**Isolation forest accuracy (%)**
Group 1	56.22 ± 0.86	51.33 ± 0.33	54.39 ± 0.64	55.61 ± 0.75	56.93 ± 0.50
Group 2	69.28 ± 0.39	70.84 ± 0.34	69.94 ± 0.67	60.22 ± 0.32	69.33 ± 0.62
Group 3	88.87 ± 0.75	89.63 ± 0.50	91.72 ± 0.55	90.50 ± 0.00	92.33 ± 0.24

Group 4	72.72 ± 1.11	75.50 ± 2.33	75.00 ± 0.00	64.83 ± 0.62	75.5 ± 0.00

In Group 1, the normalization, Ensemble, and Isolation Forest methods were unsuccessful in improving the average prediction accuracy, which in all methods, remained at around 55%. This may be attributed to the fact that E11 appears to be noisier than other test datasets. The in-trial prediction accuracy for E11 is also approximately 60%, which is significantly lower than that of the other trials. This further indicates that there may have been issues during the data collection process, causing E11 to vary a lot from the other samples.

In Group 2, the fundamental SVM's prediction accuracy is approximately 20% higher than in Group 1. However, it is evident that the remaining methods, including normalization and Isolation Forest, did not lead to substantial improvements in average accuracy, which remained around 70%. In the case of the Boosting classifier, accuracy even decreased by 13%. This suggests that the characteristics of the E12 test data may be better suited to strong classifiers, such as SVM with grid search, rather than a combination of weaker classifiers.

In Group 3, the accuracy is the best among other groups. The accuracy ranges from 88.87% to 92.33%, this suggests that the this set of data is already well-scaled. As in the intra-trial results, E6, also the testing dataset in Group 6, contains the cleanest data and has the best intra-trail prediction accuracy of 98.26%.

In Group 4, the accuracy trend is similar to that observed in Group 2. The application of normalization, bagging, and isolation forest methods resulted in slight improvements in the average accuracy, increasing from 72.72% to 75.5%, 75% and 75.5% respectively. However, the accuracy in the boosting classifier decreased by around 12%. This suggests that similar to E12, the E8 testing dataset also relies more on strong classifiers rather than weak learners.

By averaging the four prediction accuracies in all of the five methods, the final best prediction accuracy for Leave-One-Group-Out Cross-Validation is 73.52% ± 0.34% in Isolation Forest. This is around 2% higher than the accuracy from fundamental SVM classifier, which is 71.77% ± 0.78%.

Overall, all these attempts have very limited improvements in the prediction accuracy compared to the fundamental classifier. Therefore, another approach is introduced: extracting features from the custom CNN model and subsequently applying those classifier designs to the features to assess their impact on the final prediction accuracy. The results are provided in Table 6 below. Each experiment will be run for 30 iterations due to the fact that they cost a huge amount of computing power to run, and each experiment will be repeated 3 times, the same as the features calculated from *Averaging* to get the average value of accuracy and standard deviation.

Table 6. Cross-trial prediction accuracy results for features extracted from *CNN*

model

	Fundamental SVM classifier accuracy (%)	SVM classifier with normalized features accuracy (%)	Ensemble learning (Bagging) accuracy (%)	Ensemble learning (Boosting) accuracy (%)	Isolation forest accuracy (%)
Group 1	56.67 ± 2.36	51.67 ± 2.89	56.67 ± 2.89	49.33 ± 0.72	57.56 ± 0.79
Group 2	76.11 ± 1.57	75.00 ± 0.00	81.11 ± 1.57	82.67 ± 1.19	83.17 ± 2.32
Group 3	88.67 ± 0.98	89.44 ± 0.79	87.00 ± 1.52	91.67 ± 2.36	95 ± 0.00
Group 4	73.52 ± 2.50	71.00 ± 1.41	68.05 ± 0.39	67.72 ± 1.23	77.67 ± 1.55

In Group 1, the average accuracy ranges from 49.33% to 56.67%. The normalization, Ensemble and Isolation Forest attempt all failed to improve the accuracy. In comparison to the prediction accuracy in Table 5, there is also no improvement. The best average accuracy in Table 5 for group 1 is the one from fundamental classifier, which is 56.83%, however, for classifiers in Table 6, none of these prediction accuracies exceed that accuracy. As illustrated before, this dataset is probably much nosier than others, possible modifications will be suggested in *Discussion section* in order to increase the accuracy of the accuracy of E11.

In Group 2, compared to the fundamental SVM classifier, the Ensemble learning, and Isolation Forest methods show a 9% improvement. Additionally, compared to the fundamental classifier in Table 5, the improvement is over 20%. This indicates that for this specific dataset, the designed CNN model is more capable of accurately extracting features and is more precise than the *averaging* method. Thus, classifiers with more complex structures and outlier detections can achieve more accurate classification.

In Groups 3 and 4, similar to Group 2, the Isolation Forest method resulted in improvements of approximately 7% and 6%, respectively, compared to the fundamental SVM classifier. Furthermore, when compared to the fundamental classifier in Table 5, this method provides both Group 3 and Group 4 with an improvement of around 7%. For Group 3, this detection accuracy is 95%, suggesting that the model is well-tuned for this specific dataset and there are very few spaces for improvement. For Group 4, although the best average detection accuracy did not improve in a percentage as much as Group 2, such increase in accuracy is still a strong proof that CNN extracted features can be really suitable for detecting multispectral images for UCBSV on cassava plants.

By averaging the four prediction accuracies across all five methods, the final best prediction accuracy achieved through Leave-One-Group-Out Cross-Validation is 78.35% ± 1.17% using the Isolation Forest method. This accuracy represents an improvement of approximately 6.25% over the fundamental SVM classifier, which achieved an accuracy of

73.74% ± 1.85% using the same feature extraction method. Additionally, in comparison to fundamental classifier and Isolation Forest method using features calculated from *Averaging*, it exceeded by 9.16% and 6.57% respectively.

4. Discussion

In the previous experiments, features extracted from *the custom CNN model* has proven to reach higher prediction accuracies than the features calculated from *Averaging* in Group 2, 3 and 4. This is mainly due to the fact that the custom CNN model has utilized a lot of methods to handle unseen data and prevent overfitting. Therefore, by applying such trained weight to the testing data, a proper designed SVM will be able to separate the infected and uninfected leaf samples better. However, to further improve the prediction accuracy for these groups or to increase the prediction accuracy of Group 1, there are still several potential modifications that can be done. They are listed below:

Modifying the patch cropping algorithm for CNN method: The original design of the patch cropping algorithm includes a step where, if the leaf is too small to crop 9 patches with size 40x40 on it, the patch size is automatically reduced to 25x25. This adjustment poses no issues when using the *Averaging* method, as averaging results will lead to a loss of information about size; thus, a smaller patch among larger ones does not significantly affect the outcome. However, to input the patches into the CNN model, a fixed size is required. Therefore, the part of the algorithm that changes the size of the patch was removed for all experiments using the *CNN* method. This modification introduces a new challenge: if the algorithm cannot identify an appropriate leaf area for patch cropping, it may instead search for areas in the dark background. An example of this issue is illustrated in Figure 2 below.

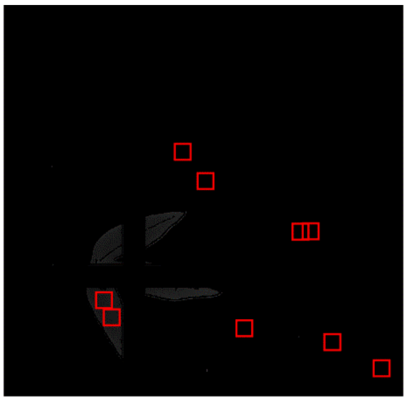

Figure 2. 9 randomly selected patches with some of them being cropped on the background instead of the leaf

The patches being cropped on the background will contain no information about whether the leaf is infected or not, therefore, they will be useless data being input into the CNN model. Although instances of this issue are expected to be minimal, as most of the leaf samples are sufficiently large enough to accommodate 9 patches of size 40x40, they may still cause a slight impact of the final result of the data. To eliminate this issue, the patches should also be resized to 25x25 for the *CNN* method, and further modification such as expanding the size of them to 40x40 before feed into the CNN model or adding some extra pixels with value of zero around them (also known as the 'zero padding' method) can be applied to these resized patches. In the end, these patches need to contain enough valid information while maintaining a fixed size.

Data augmentation (rotation and zooming): A rotation range of ± 20° was applied on the training patches before inputting into the CNN model has been examined. However, there is no significant change observed in the prediction accuracy. Consequently, this adjustment was not included in the design of the custom CNN model. However, zooming on the leaf samples has not been applied and more adjustments can be made on the rotation angles. Such attempt could theoretically improve the diversity of the training data, given that there are only 4 trials of data for training and testing, as listed in Table 1, data augmentation should have some improvements on the accuracy if optimized properly.

CNN models for multispectral images: Only one of the CNN models listed on Table 3 has been proven to have great ability in identifying multispectral images, which is the 2D-CNN model. It was a baseline model being used in previous studies[2] to demonstrate the generalizability of the developed spatial–spectral machine learning method to multispectral image applications such as the Indian Pine dataset, a public bench mark for these type of problems. For other model examined in Table 3, such as Alexnet and VGG16, they are all primarily designed for solving problems of standard RGB (3 channel) images, instead of 14 channel multispectral images in this case. Although some modifications have been made to these models to allow them accepting more input channels, their prediction accuracy are not robust enough as shown in Table 3. Therefore, several other CNN models designed primarily for solving multispectral image problems can be investigated to increase the robustness of the model. Models like 3D-CNN-LR, Feature-ensemble ND-SVM and SSFNet$_{2D}$ are all proven to have high prediction accuracies on Indian Pine multispectral image dataset [3].

More iterations in one run for classifiers: The experiment was conducted with 300 iterations for cross-trial SVM classifiers using features calculated from *Averaging* method, and 30 iterations for classifiers using features extracted from *CNN model*. However, these iteration counts may not be sufficient, as additional iterations could potentially enhance the classifiers' stability and accuracy. Further experiment with higher iteration counts, such as 500 or 1000, for all designed classifiers using both feature extraction methods, could be assessed to see whether the model's performance changes or plateaus after higher iteration.

Combination of different methods and classifiers: Although accuracy of different designs of classifiers has been tested, they have not been combined. For example, Isolation Forest algorithm can be applied to ensemble classifier as a preprocessing step. Once the outliers are detected and replaced by the median value, the ensemble classifier may result in better prediction accuracy. Another way of combining the classifier is by voting.

Patches may go through both *Averaging* and *CNN model* and the generated features can be feed into different classifiers; these classifiers then vote (most votes from the models will be chosen) to make the final prediction. Such attempt may help improve accuracy if the models complement each other.

5. Conclusion

Overall, the two feature extraction methods, *Averaging* and *custom CNN models* both shown to be feasible for classifying a number of the multispectral images of cassava plants infected and uninfected with UCBSV, with the *custom CNN* model has better overall detection accuracy than the *Averaging* method. The combination of custom CNN models and fundamental SVM model with Isolation Forest algorithm resulted in the best average prediction accuracy of 78.35% ± 1.17% for the 4 groups of cassava plants infected with UCBSV. This is 9.16% higher than the fundamental SVM model with features calculated from *Averaging*. Meanwhile, the experiment group with the best quality of data is E6, it reached an intra-trial prediction accuracy of 98.26% ± 0.10%. The cross-trial results of Group 3, training on E8, E11, E12 and testing on E6 also have great accuracy ranges from 87.00% to 95.00%. The experiment group with the worst quality of data may probably be E11, the intra-trial prediction accuracy is only 59.93% ± 0.14% and cross-trial accuracy ranges from 46.33% to 56.93%. Several additional image processing and machine learning methods have been suggested in the *Discussion section* to improve the quality of input data and the robustness of model. However, these ideas still need to be supported by more experimental data. Presently, E11's experiential results vary a lot from the rest of trials, further research is needed on E11 and more leaf samples both infected and uninfected with UCBSV can be collected from the laboratory as a comparison to further explore the leaf samples of E11. Those features can also be plotted together in a feature diagram by different trials or by different DPI (Days Post Inoculation) to analyze the differences between the characteristics of features generated by different models.

6. References

[1] Peng, Y., Dallas, M.M., Ascencio-Ibáñez, J.T. *et al.* Early detection of plant virus infection using multispectral imaging and spatial–spectral machine learning. *Sci Rep* **12**, 3113 (2022) Methods Section. https://doi.org/10.1038/s41598-022-06372-8

[2] Peng, Y., Dallas, M.M., Ascencio-Ibáñez, J.T. *et al.* Early detection of plant virus infection using multispectral imaging and spatial–spectral machine learning. *Sci Rep* **12**, 3113 (2022) Discussion Section. https://doi.org/10.1038/s41598-022-06372-8

[3] Peng, Y., Dallas, M.M., Ascencio-Ibáñez, J.T. *et al.* Early detection of plant virus infection using multispectral imaging and spatial–spectral machine learning. *Sci Rep* **12**, 3113 (2022) Table 3. Class-specific accuracies (%) on Indian Pines dataset. https://doi.org/10.1038/s41598-022-06372-8

7. Appendix

Designed CNN model structures in *Table 3, Section 2.3*.

Convolutional Neural Networks Configuration		
Alexnet	**VGG16**	**2D-CNN**
40x40x14 input multispectral images		
Conv2D-96	conv2-64 conv2-64	Conv Block 1 Conv2D-8
Batch Normalization + MaxPooling	Batch Normalization + MaxPooling	Batch Normalization + ReLU
Conv2D-256	conv2-128 conv2-128	Conv Block 2 Conv2D-8
Batch Normalization + MaxPooling	Batch Normalization + MaxPooling	Batch Normalization + ReLU
Conv2D-384 Conv2D-384 Conv2D-256	conv2-256 conv2-256 conv2-256	Residual Block 1 Residual Block (Filters = 16) MaxPooling (2 × 2)
Flatten	Batch Normalization + MaxPooling	Residual Block 2 Residual Block (Filters = 32) MaxPooling (2 × 2)
FC - 4096	conv2-512 conv2-512 conv2-512	Residual Block 3 Residual Block (Filters = 64) MaxPooling (2 × 2)
Dropout 0.5	Batch Normalization + MaxPooling	Residual Block 4 Residual Block (Filters = 128) MaxPooling (2 × 2)
FC - 4096	conv2-512 conv2-512 conv2-512	Residual Block 5 Residual Block (Filters = 256) MaxPooling (2 × 2)
Dropout 0.5	Flatten	Flatten
FC - 128 (Feature layer)	Dropout 0.5	Dropout 0.5
Sigmoid	FC-4096	FC-512
	Dropout 0.5	Dropout 0.5

	FC-2048	FC-128 (Feature layer)
	Dropout 0.5	Sigmoid
	FC-128 (Feature layer)	
	Sigmoid	