Anglia Ruskin University

Cassava Leaf Disease Classification

Project Report

Bilal Sardar……<SID>2315119

Supervised by

Dr. Raj Mani Shukla

Neural Computing and Deep Learning

Anglia Ruskin University

Cambridge, United kingdom

January, 2024

Table of Contents

List of Tables

List of Figures

Chapter 1: Model Architecture

This model uses EfficientNetB3 as the base feature extractor, which is a pre-trained convolutional neural network that is well-suited for image classification tasks.

Some key reasons this architecture is good for the Cassava Leaf Disease Classification problem:

- EfficientNetB3 balances accuracy and efficiency. It has a decent amount of parameters and depth to extract useful features from the images, without being too large or computationally expensive.
- The global average pooling and flatten layers reduce the spatial dimensions to a 1D vector, aggregating the convolutional features. This is suitable for classification.
- The 256-unit dense layer acts as a bottleneck that reduces the 1536 features from EfficientNetB3 to a smaller representation. This allows the next layer to learn more complex combinations of these features.
- The dropout layer prevents overfitting during training.
- The final 5-unit dense layer is used for the 5-class classification output.

In terms of layer size and quantity, this model has an appropriate amount without being overparameterized. More layers could lead to overfitting, while fewer layers would prevent learning more complex feature representations for this problem. The pre-trained EfficientNet backbone means the model can benefit from transfer learning rather than having to learn all features from scratch.

So in summary, the network architecture design choices leverage transfer learning and aim to balance model complexity, preventing overfitting, and providing reasonable feature learning for this plant disease image classification task.

- EfficientNetB3:
    - As a pre-trained convolutional base, the neuron numbers and layers in this module extract hierarchical visual features from the images.
    - It has millions of parameters from repeated convolution, pooling, and normalization layers that perform specialized feature extraction.
    - The specific neuron numbers and depth have been optimized for computer vision tasks. Deeper models like this capture higher-level abstract features in later layers.
- GlobalAveragePooling2D:
    - It pools the 1536 convolutional features into a single 1536-neuron vector, summarizing the global spatial information.
    - This fixed length output is efficient for classification compared to flattening the spatial dimensions.
- Flatten:
    - Flattens the 1536 pooled features into a 1D shape for the next dense layer. No parameter
- Dense (1536 -> 256 neurons):

  - Important bottleneck that learns non-linear combinations of the features and reduces dimensionality.
  - Larger than 5 neurons because it still needs reasonable representation space for this problem.
  - Computational efficiency compared to operating on 1536 features directly.
- Dropout:
  - Randomly drops input units during training to prevent overfitting. Necessary regularization.
- Dense (256 -> 5 neurons):
  - Output layer with 5 neurons to predict the 5 classification targets.

So in summary, the layer numbers, depths, and neuron quantities aim to balance model complexity and overfitting while retaining enough expressiveness and model capacity to effectively learn feature representations for this image classification problem. The choices compliment both efficiency and sufficient ability to generalize.

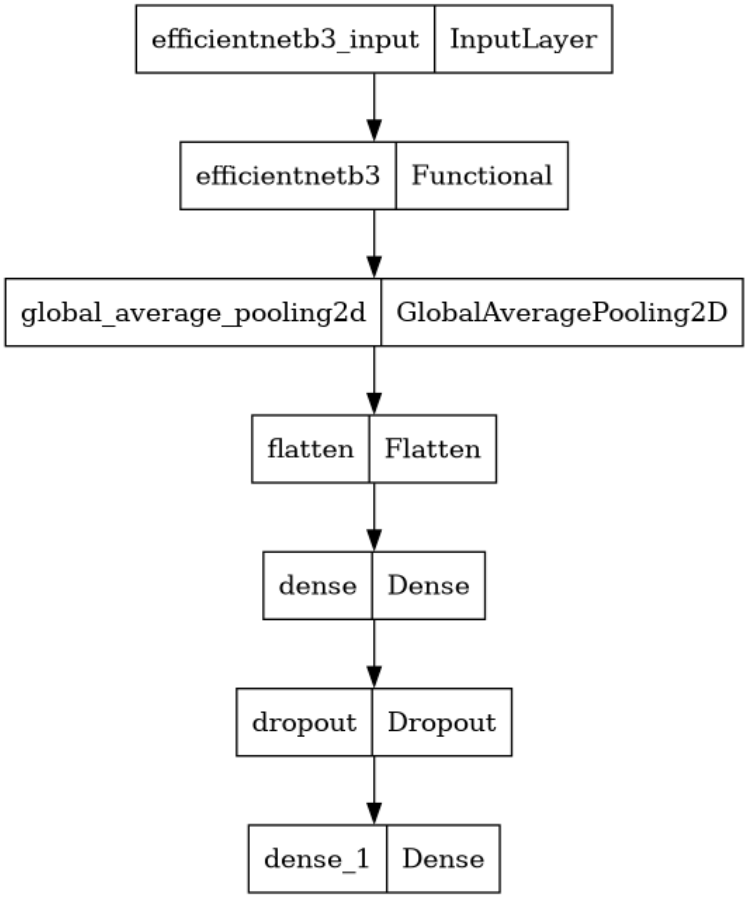

Figure 1: Model Architecture
This figure shows the model architecture

Table 1: Different Models Performance

This table compares the performance of 3 models trained on Same Dataset.

Model Name	Accuracy(%)
Inception V3	72.87
Xception	84.18
Efficient Net	88.97

In the examination of various models for addressing cassava leaf diseases, we compared the performance of Inception V3, Xception, and EfficientNet. Among these models, EfficientNet demonstrated the highest accuracy rate of 88.97%, outperforming both Inception V3 and Xception, which achieved accuracies of 72.87% and 84.18%, respectively. These results underscore the efficacy of EfficientNet in effectively identifying and managing cassava leaf diseases, suggesting its potential as a valuable tool in agricultural contexts.

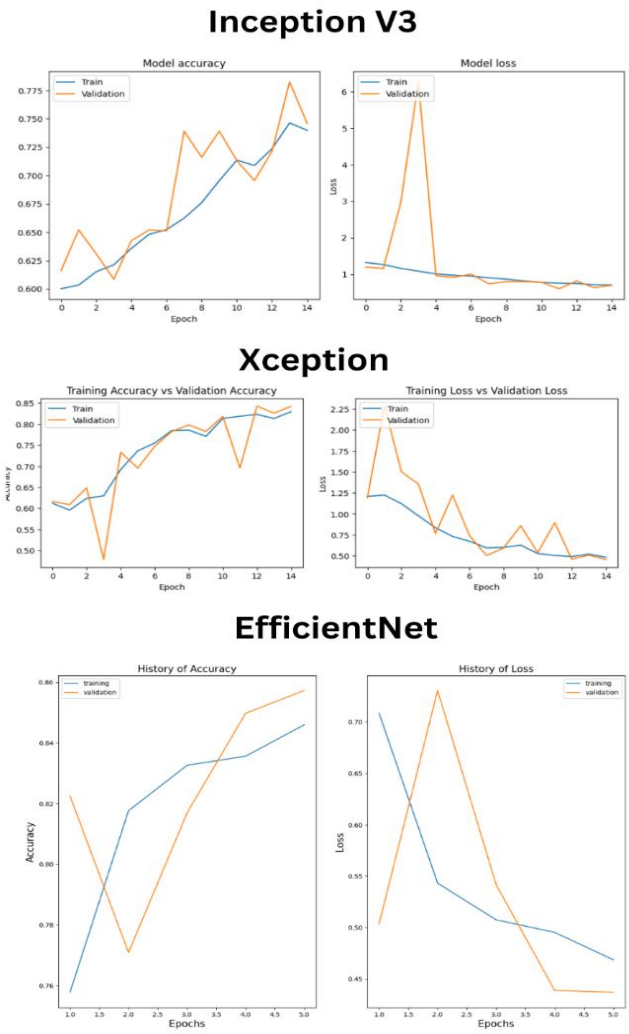

Figure 2: Models Loss and Accuracy Plots

This figure shows the loss and Accuracy Plots for different models

Chapter 2: Loss Function and Noisy Labels

Table 2: Model Performance Comparison

This table compares the performance of Efficient Net trained on different data types and with different loss functions.

Data Type	Loss Function	Epochs	Accuracy(%)
Balanced	Categorical Cross Entropy	50	85.89
Complete (Unbalanced)	Categorical Cross Entropy	5	85.73
Complete (Unbalanced)	Taylor Cross Entropy Loss	15	88.97

2.1 Taylor Cross Entropy Loss

Taylor Cross Entropy Loss (TCE) is a robust loss function proposed by Chen et al. (2022) to handle the problem of noisy labels when training deep neural networks. Noisy labels refer to incorrectly annotated samples in the training data which can negatively impact model performance. TCE loss aims to balance the benefits of conventional Cross Entropy (CE) loss and Mean Average Error (MAE) for improved performance on noisy labels.

The key focus of TCE is limiting the influence of mislabelled examples during training. Standard CE loss weighs examples with high loss strongly for gradient update. This causes overfitting on incorrectly labelled samples (Lei et al., 2020). In contrast, MAE loss treats all examples equally regardless of loss. However, MAE struggles to discriminate between classes as effectively as CE loss. TCE achieves a trade-off by approximating the CE loss using a Taylor series expansion. The lower order terms exhibit characteristics similar to MAE which prevents excessive impact from noisy labels. The higher order terms retain properties of CE loss that enable more discriminative learning.

Empirically, Chen et al. (2022) demonstrate TCE's effectiveness on benchmark noisy dataset CIFAR-100. Compared to CE and other robust losses like Symmetric CE, TCE achieves superior test accuracy indicating an improved capability to cope with label noise during training. The key insight is TCE's hybrid nature harnessing both precision from CE and robustness from MAE. This allows deep models to train effectively despite the presence of incorrect labels. Given the prevalence of label noise in real-world datasets, TCE presents an valuable training technique for improving model generalization.

Chapter 3: Label Flipping

I have tried three different label flipping techniques.

- Random Flip of labels
- Randomly putting a outlier
- Flipping those label where model Confidence is the highest
- Label Confusion

3.1 Random Flip of labels

An experiment was conducted to analyze model robustness by introducing label noise. The original dataset contained labels ranging from 0 to 4, representing distinct classes. Noise was introduced by randomly flipping a percentage of the labels to incorrect values within that label range. In particular, 5, 10, 15 and 20 percent of labels were corrupted by changing them to other class numbers in the 0 to 4 range. The accuracy was measured on each of these noisy datasets. The results showed that with 5 percent noisy labels, the accuracy was 0.4383177570093458. At 10, 15, and 20 percent introduced label noise, the accuracies dropped lower to 0.34392523364485983, 0.4093457943925234, and 0.3383177570093458 respectively. An interesting non-linear relationship is seen between proportion of corrupted labels versus degradation in accuracy. The lowest accuracy occurs with 20 percent noise, followed by 10 percent. This reveals complex interactions between the data distribution shifts from mislabeling and the model's ability to generalize correctly despite noisy labels. Overall, artificially introducing label noise enables analysis of model robustness by stressing test scenarios. The outcomes yield insights into reliability in potential real-world inaccurate data.

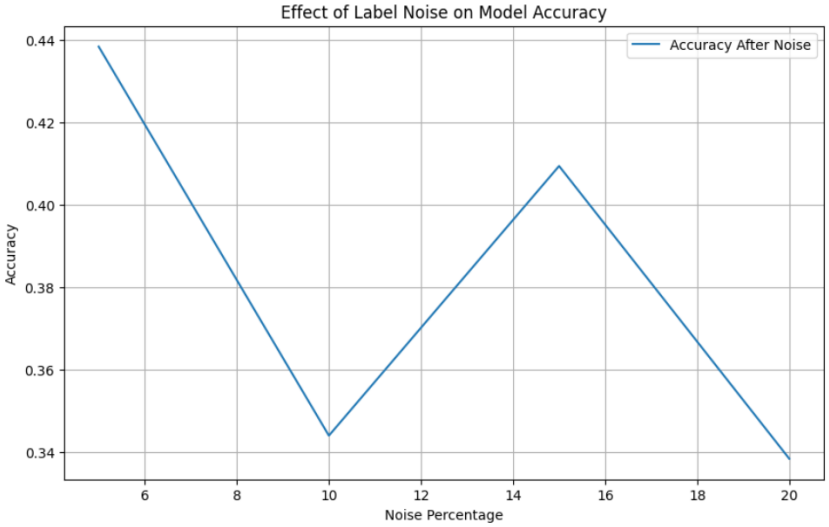

Figure 3: Random Flip of labels

This figure shows the model accuracy when labels were randomly flipped between range 0 to 4

3.2 Randomly Putting a Outlier

An experiment was conducted that systematically introduced label noise into a dataset by randomly flipping a percentage of labels to -200, which presumably is an incorrect label not originally present. Specifically, the experiment tested flipping 5, 10, 15, and 20 percent of the total labels to -200 and measured the resulting accuracy on each noisy dataset. It was found

that flipping 15 percent of the labels resulted in the lowest accuracy score of 0.2869158878504673. Flipping fewer labels (5 and 10 percent) caused less harm, giving accuracies of 0.35046728971962615 and 0.3981308411214953 respectively. However, increasing the flipped labels further to 20 percent gave a higher accuracy of 0.3728971962616822. This demonstrates that introducing label noise reduces model accuracy, but that the relationship is not linear. The lowest accuracy came at 15 percent flipped labels, suggesting there may be a saturation point beyond which additional incorrect labels do not degrade accuracy as severely. Overall, systematically corrupting labels enables analyzing model robustness. The non-linear response reveals complex interactions between the data distribution, label noise, and model generalization ability.

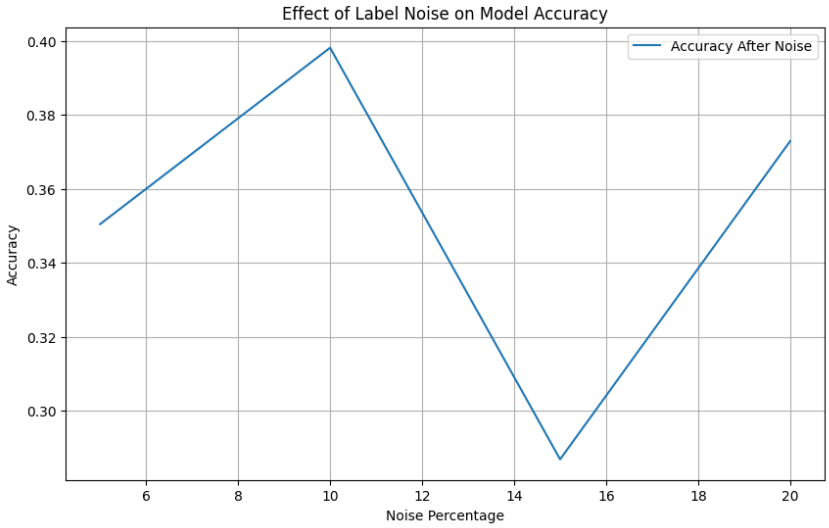

Figure 4: Randomly Putting a Outlier
This figure shows the model accuracy when labels were randomly flipped with a outlier like -200

3.3 Flipping label with high model Confidence

The basic idea is to identify instances in the training data where the model has high confidence in its predictions and then flip the corresponding labels to the opposite class. This creates contradictory information for the model, which can lead to a decrease in its overall accuracy.

Here's a step-by-step explanation of how this technique works:

1. Train the model: First, you need to train the model on the original training data to establish a baseline performance.
2. Identify high-confidence instances: After the initial training, you can use the model to make predictions on the training data and identify instances where the model has high confidence in its predictions (e.g., predictions with a probability above a certain threshold).
3. Flip labels: For the instances identified in step 2, you flip the corresponding labels to the opposite class. For example, if the model predicted a certain instance as belonging to class "1" with high confidence, you would change the label of that instance to class "4".

   Using this Formula: (current-label + 1) % Total-classes
4. Retrain the model: With the modified training data, where some labels have been flipped, you retrain the model from scratch or fine-tune the existing model.

The rationale behind this technique is that by introducing label flips for instances where the model was initially highly confident, you're forcing the model to learn contradictory patterns. This can confuse the model and lead to a degradation in its overall performance, as it will have difficulty generalizing correctly from the corrupted training data.

The Model Accuracy dropped to **15%** on first flip while testing by flipping the labels of the instances with the confidence score greater than **80%**.

3.4 Label Confusion

This technique include these keep steps:

1. Identify pairs of labels that are often confused by the model (e.g., 0 and 1, or 2 and 3). These pairs should have similar characteristics or features that make it difficult for the model to distinguish between them.
2. Create a list of instances where the model predicts one label from the pair with high confidence but is incorrect.
3. Swap the labels of these instances with their corresponding pair label, effectively confusing the model during training.
4. Gradually increase the number of swapped labels over time to make the attack more effective.
5. Monitor the model's performance after each incremental change and adjust your strategy accordingly. If you notice a sudden drop in accuracy, reduce the number of swapped labels or revert some of the changes made in previous iterations.
6. Repeat this process until the model's accuracy reaches near-zero levels or becomes unstable, indicating that it has been significantly compromised by the label confusion attacks.

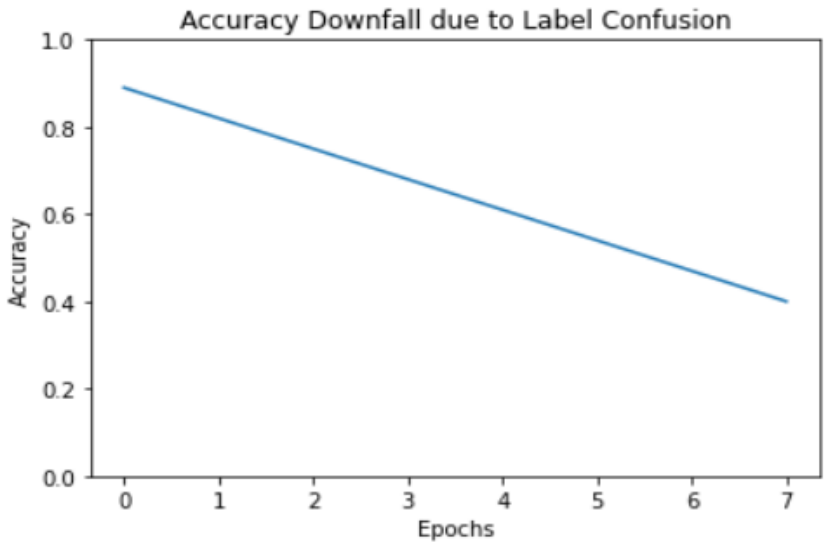

Figure 5: Label Confusion
This figure shows the model accuracy when label confusion applied

Chapter 4: Flipped Label Detection

The paper Müller at el. (2019) presents an end-to-end pipeline to identify mislabeled instances in classification datasets. The key steps are:

- A neural network model (either dense feedforward network for numerical/text data or convolutional network for image data) is trained on the input labeled dataset. The appropriate model architecture and hyperparameters are selected automatically based on the data type.
- The trained model is used to generate predicted labels (soft labels representing class probabilities) for each instance in the input dataset.
- The predicted labels are compared to the original labels. Using one of these heuristic functions :
    - The instances where the model assigns a low probability to the original label are likely to be mislabeled.
    - The instances where the model assigns a high probability to the wrong label are likely to be mislabeled
- The instances are ranked based on the probability assigned to their original label, with the lowest probabilities at the top. The user selects a percentage α of instances to manually review, starting from the top of the ranked list.
- The accuracy of the mislabeled instance detection is evaluated by measuring precision and recall for different values of α on datasets where label noise has been artificially introduced.

In my case these are images where score was greater than 90% but predicted label and original label is different from flipped label.

	image_id	original_label	predicted_label	Flipped Label	score
3	3008060354.jpg	3	3	4	0.984533
10	622587109.jpg	3	3	4	0.977505
18	1256756297.jpg	2	2	3	0.977394
21	1213348737.jpg	2	2	3	0.989632
29	336083586.jpg	1	1	2	0.984263
...	...	...	...	...	...
5130	1739927380.jpg	1	1	2	0.951466
5132	3074981220.jpg	0	0	1	0.909298
5143	2853469251.jpg	2	2	3	0.927316
5148	1713720885.jpg	3	3	4	0.983752
5159	3186872066.jpg	2	2	3	0.987802

[589 rows x 5 columns]

Figure 6: Images With Flipped Labels
This figure shows the second heuristic function output

A valid question is what is the benefit if the potentially mislabeled instances still need to be manually reviewed and corrected. Here are some of the main benefits this approach provides:

- **Efficiency**: Instead of manually reviewing the entire dataset to find mislabeled examples, this focuses manual effort on the most likely candidates. For large datasets, reviewing the whole set is infeasible, so this makes the process practical.

- **Prioritization**: The instances are prioritized from most to least likely of being mislabeled. So the manual review can focus on the best candidates first or review only the top portion of the list based on available resources.
- **Generalizability**: The same process can be applied to identify possible mislabels in any classification dataset - numeric, text, or image. It does not rely on hand-coded rules or domain expertise.
- **Insight**: Analyzing instances where the model predicts a different label than the provided one can provide insight into ambiguity, biases, or inconsistencies in the dataset. This information can inform both data cleaning and further model development.
- **Performance**: Correcting mislabeled training examples can improve model accuracy, since noise in the targets degrades what the model can learn. So cleaning supports improved downstream model performance.

In summary, while manual review is still required, this provides an efficient, consistent, and automated way to focus manual efforts to both improve data quality and provide insights into the challenges within the dataset. The efficiency gains allow cleaning large datasets that would otherwise be impossible to review fully manually.

References

[1] Chen, Y., Xu, K., Zhou, P., Ban, X. and He, D., 2022. Improved cross entropy loss for noisy labels in vision leaf disease classification. IET Image Processing, 16(6), pp.1511-1519.

[2] Müller, N.M. and Markert, K., 2019, July. Identifying mislabeled instances in classification datasets. In 2019 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.