Paddy Leaf Disease Classification using SOTA CNN and Transformer Models

Report

Bindela Jaswanth Kumar Reddy

MindFlix AI, Bangalore

March 10, 2025

Contents

Abstract

This report details the development, experimental evaluation, and comparative analysis of two state-of-the-art (SOTA) deep learning models—a convolutional neural network (CNN) and a Transformer-based model—for classifying paddy leaf diseases. The models were trained on a combined dataset sourced from the *Paddy Doctor: Paddy Disease Classification* and *Rice Leaf Disease Image Samples* datasets. The report discusses the methodologies, training metrics, hyperparameter tuning strategies, and recommendations on the optimal model choice.

1 Introduction

Paddy leaf disease classification is a vital task for crop health monitoring and disease management. In this project, we implement and compare two SOTA models:

1. **CNN-based model:** Built on a ResNet50 architecture, optimized via fine-tuning.
2. **Transformer-based model:** Implemented using a Vision Transformer (ViT) to capture global image features.

The goal is to classify each paddy leaf image into one of nine disease categories or as a normal leaf. The models are evaluated on a test dataset consisting of 3,469 images.

2 Datasets

Two datasets were combined for model training:

1. **Paddy Doctor: Paddy Disease Classification**
   Source: `https://www.kaggle.com/c/paddy-disease-classification/data`

   This dataset contains 10,407 labeled paddy leaf images organized in subfolders by label (9 diseases + 1 normal). Annotations are provided in the CSV file `paddy_disease_train.csv` with columns: `image_id, label, variety, age`.

2. **Rice Leaf Disease Image Samples**
   Source: `https://data.mendeley.com/datasets/fwcj7stb8r/1`

   This dataset consists of 5,932 images of four rice leaf diseases. Annotations are provided in `rice_leaf_disease_images.csv` with columns: `image_id, label`.

3 Methodology

3.1 Data Preprocessing

Images are resized to 224x224 pixels and normalized using ImageNet statistics. Data augmentation techniques such as random horizontal flips and rotations are applied to improve model robustness.

3.2 CNN Model

A pre-trained ResNet50 model is used as the backbone. The final fully connected layer is replaced to output predictions for the 10 classes. The feature extraction layers are frozen to expedite training and reduce overfitting.

3.3 Transformer Model

The Transformer model employs a Vision Transformer (ViT) architecture, accessed via the `timm` library. As with the CNN, only the classification head is fine-tuned after freezing most of the pre-trained parameters.

3.4 Implementation Details

The complete code is implemented in the Jupyter Notebook `backend.ipynb`. Key components include:

- **Dataset Loader:** The custom `LeafDiseaseDataset` class loads images and annotations.
- **Model Builders:** Functions `get_cnn_model` and `get_transformer_model` initialize the respective models.
- **Training Framework:** The `Trainer` class handles model training, evaluation, and logging of metrics.

A sample snippet from the notebook is provided below:

```
train_transforms = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(20),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
combined_train_dataset = combine_datasets(csv_files, root_dirs,
transform=train_transforms)
train_dataset, val_dataset = random_split(combined_train_dataset,
   [train_size, val_size])
```

Listing 1: Dataset loading and augmentation

4 Experimental Results

Experiments were conducted for 3 epochs for both models. The recorded training and validation metrics are as follows.

4.1 CNN Model Performance

Epoch	Training Accuracy	Validation Accuracy	Validation Loss
1	55.83%	65.15%	1.0669
2	65.01%	68.57%	0.9779
3	67.27%	67.53%	0.9530
Final Metrics	**Accuracy: 67.90%**	**Precision: 72.09%**	**Recall: 67.90%**

Table 1: CNN Model Performance Metrics

Figure 1 shows the CNN model's training progress in terms of loss and accuracy.

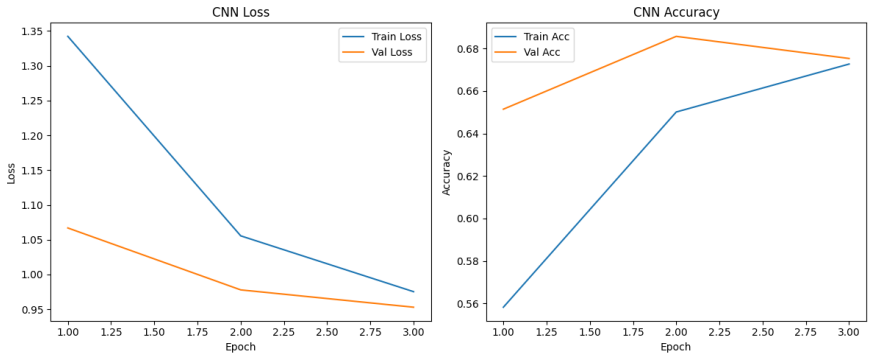

Figure 1: CNN Model: Loss vs. Epoch and Accuracy vs. Epoch

4.2 Transformer Model Performance

Epoch	Training Accuracy	Validation Accuracy	Validation Loss
1	71.10%	78.03%	0.6628
2	81.46%	81.85%	0.5612
3	83.54%	83.84%	0.4890
Final Metrics	**Accuracy: 83.78%**	**Precision: 85.49%**	**Recall: 83.78%**

Table 2: Transformer Model Performance Metrics

Figure 2 illustrates the training dynamics for the Transformer model.

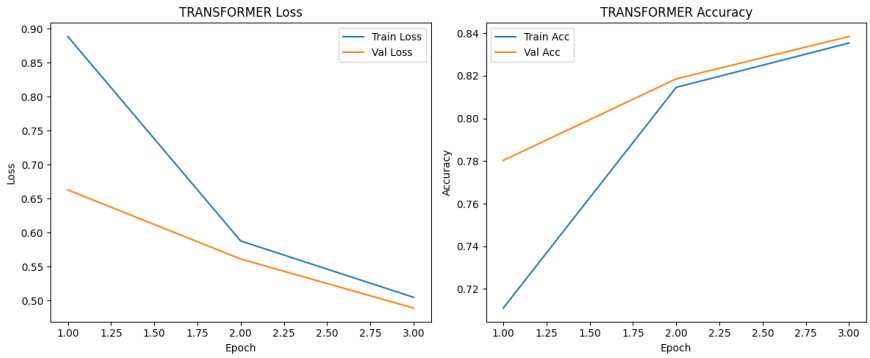

Figure 2: Transformer Model: Loss vs. Epoch and Accuracy vs. Epoch

5 Comparative Evaluation

The comparative analysis reveals that the Transformer model outperforms the CNN model on all measured metrics. Specifically:

- **Accuracy:** Transformer achieved 83.78% vs. 67.90% for the CNN.
- **Precision:** 85.49% for the Transformer compared to 72.09% for the CNN.
- **Recall:** Transformer recorded 83.78% while the CNN reached 67.90%.

These results are in line with the principles outlined in the Google ML Crash Course on Accuracy, Precision, and Recall [**?**]. Figures 1 and 2 provide visual confirmation of the training progress for both models.

6 Hyperparameter Tuning

Hyperparameter tuning is critical for improving model performance. In this project, the following tuning strategies were applied or are recommended:

- **Learning Rate:** An initial learning rate of 1e-3 was used with a StepLR scheduler (reducing the rate by a factor of 0.1 every 7 epochs) to ensure gradual convergence.
- **Batch Size:** A batch size of 32 was chosen to balance training stability with computational efficiency.
- **Number of Epochs:** Although experiments were run for only 3 epochs, extending the training duration may improve performance.
- **Feature Extraction vs. Fine-tuning:** The strategy of freezing pre-trained layers helped mitigate overfitting and reduce training time. Future experiments could involve selective unfreezing of layers for more granular fine-tuning.

7 Recommendations and Future Directions

Our experimental results from `backend.ipynb` indicate that the Transformer model—achieving 83.78% accuracy, precision, and recall—is the optimal choice for applications demanding peak performance, thanks to its superior ability to capture global image features. In contrast, the ResNet50-based CNN is a viable alternative when operating under constrained computational resources.

The following snippet from `backend.ipynb` illustrates our flexible model selection approach:

```
if model_type == 'cnn':
    model = get_cnn_model(num_classes, feature_extract=True)
elif model_type == 'transformer':
    model = get_transformer_model(num_classes, feature_extract=True)
```

Listing 2: Model Selection Logic from backend.ipynb

Future enhancements include longer training durations for improved convergence, refined hyperparameter tuning (exploring various learning rates, batch sizes, and regularization techniques), and ensemble methods that could combine the strengths of both models.

In summary, while the Transformer model is recommended for high-accuracy requirements, the CNN model remains an attractive option for resource-limited scenarios, with several avenues available for further performance optimization.

8 Conclusion

This report presented a detailed study of paddy leaf disease classification using two SOTA models: a CNN based on ResNet50 and a Transformer-based model using ViT. The experiments demonstrated that the Transformer model achieved significantly higher performance metrics compared to the CNN. Hyperparameter tuning proved essential in optimizing model performance, and future work should explore more extensive tuning and model unfreezing strategies to further enhance classification accuracy.

References

1. Kaggle: `https://www.kaggle.com/c/paddy-disease-classification/data`
2. Mendeley Data: `https://data.mendeley.com/datasets/fwcj7stb8r/1`
3. Google ML Crash Course on Accuracy, Precision, and Recall: `https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall`