*Project Submission of*

Course-: Statistical Machine Learning (Odd Semester 2024-25)

**Crop Disease Detection:**

**Image processing using Convolutional Neural Networks**

Professor- Dr Nitin Arvind Shelke

By

Shubham Tandon(E23CSEU1730),

Vipin Yadav(E23CSEU1728) and

Abhay Pratap Singh(E23CSEU1720)

Batch- 58



**School of Computer Science Engineering & Technology,**

**Bennett University.**

Submission Date-21st November 2024

**1. Abstract**

Crop diseases significantly impact global agricultural production, reducing yields by up to 40% and causing substantial economic losses. This project aims to detect plant diseases using a machine learning pipeline that classifies images of leaves into healthy or diseased categories across 38 classes. The system leverages a custom Convolutional Neural Network (CNN) and a pre-trained MobileNet model to achieve accurate classification. A dataset from PlantVillage is used for training, testing, and evaluation. Comparative analysis of the two approaches highlights their strengths and limitations, providing practical insights for real-world deployment.

**2. Introduction**

Agriculture forms the backbone of many economies, providing food security and livelihoods to billions of people. However, crop diseases pose a significant challenge to achieving optimal agricultural productivity. According to studies, crop diseases can reduce yields by an average of 40%, with some farmers, especially in developing regions, facing complete crop failure. This leads to severe economic consequences and increased pressure on global food supply chains.

With the global population expected to exceed 9 billion by 2050, the need for innovative solutions in agriculture is paramount. To address this, leveraging machine learning (ML) and computer vision technologies offers promising opportunities. These technologies can automate the detection of diseases from leaf images, providing quick and reliable diagnoses to farmers, thereby preventing severe losses. The proliferation of smartphones, particularly in rural areas, further amplifies the accessibility of such solutions, enabling disease detection through mobile applications.

This project focuses on building a deep learning pipeline to detect diseases in crop leaves using image data. It utilizes the PlantVillage dataset, a comprehensive collection of over 50,000 images of both healthy and diseased leaves across 38 classes. These classes span various plant species, including tomato, potato, apple, and corn, with images provided in different formats—color, grayscale, and segmented—for flexibility and enhanced model performance.

The primary goal is to design and evaluate two ML models:

* Custom CNN Model: A lightweight model specifically tailored for quick training and inference.
* MobileNet Model: A pre-trained architecture using transfer learning, aimed at delivering higher accuracy and generalization.

This project evaluates these models based on their accuracy, computational efficiency, and real-world applicability. A Flask-based web application is also developed to allow users to upload leaf images for disease diagnosis, making the solution accessible to end-users, including farmers and agricultural consultants. Through this approach, the project seeks to demonstrate the potential of deep learning in revolutionizing agricultural diagnostics and reducing the impact of crop diseases on global food production.

**3. Related work Review on Plant Disease Prediction using Machine Learning**

Predicting and diagnosing early diseases in plants is very important for agriculture because it will be in contact with healthy crops as well as reducing the economic loss. Traditional methods rely mostly on expert manual inspection, which is time-consuming and less available in large-scale farms. Because of the advancement of Machine Learning (ML) and DeepLearning(DL) recently, nowadays highly accurate automated plant disease detection systems using images can be developed. This paper discusses what already exists in the field through approaches, methodologies, and gaps.

**Problem Background**

Plant diseases cannot be detected easily because there are visual similarities among the various diseases as well as variations in the manifestation of symptoms based on other environmental factors. Moreover, the identification of disease might be difficult because of a lack of standard symptom databases. The technique utilized, machine learning, addresses these challenges because it offers robust models about image classification that can differentiate between subtle differences existing between disease types.

**Existing Solutions Overview**

In the last few years, several methodologies have been evolved that can overcome the issue of plant disease detection. It ranges from traditional image processing techniques to advanced machine learning models. This section summarizes existing approaches, both the machine learning-based and deep learning-based solution.

**Machine Learning Approaches**

Dataset Usage

Different datasets have been exploited to train the models. One uses much more commonly the PlantVillage dataset, which is a labelled dataset of diseased and healthy leaves. Such kind of dataset is really plagued with imbalance and diversity issues.

1. <https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset>

2. <https://www.kaggle.com/datasets/rashikrahmanpritom/plant-disease-recognition-dataset>

Model Types

Convolutional Neural Networks are more commonly applied in image classification. Architectures like ResNet, Inception, and MobileNet have been popular and commonly used for plant disease detection. These models can achieve high performance with complex patterns in visuals and are implemented in disease identification.

1. Using Deep Learning: <https://github.com/suraj4502/DL_project_Plant_Disease_prediction>

<https://www.kaggle.com/code/abdallahwagih/plant-village-disease-classification-acc-99-6\>

2. Using CNN: <https://github.com/maheshpikki/Plant-Disease-Detection>

**Existing Research and Studies**

There has been a plethora of studies reported so far on ML/DL-based approaches for the detection of plant diseases. The following are some summary results of a number of recent state-of-the-art research works that particularly refer to transfer learning, multi-class classification techniques, and the benchmark accuracy rates achieved by different models.  
  
1. <https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2024.1356260/full>

2. <https://ieeexplore.ieee.org/document/9388488>

3. <https://www.researchgate.net/publication/379492215_Plant_Leaf_Disease_Detection_Using_CNN>

**Gaps and Limitations in Current Work**

Despite great strides, there are yet many challenges in plant disease detection. One such limitation is that most models have a rather generalizing capability in different plant species, mainly due to being trained on small datasets. High computational cost and overfitting in small datasets often go together too.

**4. Methodology**

This project employs a systematic methodology to design, train, and evaluate machine learning models for crop disease detection. The process is divided into the following key stages:

**4.1 Data Preprocessing**

Effective preprocessing ensures the dataset is ready for model training by improving the model's ability to learn relevant patterns from the data.

* Dataset Overview
  + The dataset comprises 38 classes of leaf images, including healthy and diseased categories for various crops.
  + Images are available in three formats: color, grayscale, and segmented. The color format is used for this project to maintain visual detail.
* Image Resizing
  + Custom CNN: Images are resized to 150x150 pixels.
  + MobileNet: Images are resized to 224x224 pixels, aligning with MobileNet's input size requirements.
* Normalization
  + Pixel values are scaled to the range [0, 1] to normalize data and improve training stability.
* Data Augmentation
  + Augmentation techniques include:
    - Random rotations (up to 40°).
    - Horizontal and vertical flips.
    - Zooming (up to 20%).
    - Shifts (horizontal and vertical by 20%).
  + Augmentation increases dataset diversity, reducing overfitting and improving generalization.
* Splitting the Dataset
  + The dataset is divided into training (80%), validation (10%), and testing (10%) subsets to evaluate model performance effectively.

**4.2 Model Training**

**Custom CNN**

* Architecture Design
  + A lightweight, 3-layer convolutional neural network was built to process the input images.
  + Layers:
    - **Conv2D + ReLU**: Extracts features using a kernel size of 3x3.
    - **MaxPooling2D**: Reduces spatial dimensions, retaining essential features.
    - Dropout layer (rate: 0.5) to prevent overfitting.
  + Fully connected layers (Dense layers) reduce the extracted features to the desired output size.
* Hyperparameters
  + Optimizer: Adam optimizer with a learning rate of 0.001.
  + Loss Function: Categorical crossentropy for multi-class classification.
  + Batch Size: 32.
  + Epochs: 10.
* Advantages
  + Simple architecture allows faster training.
  + Requires fewer computational resources.

**MobileNet**

* Transfer Learning Approach
  + MobileNet, pre-trained on ImageNet, is fine-tuned to classify the PlantVillage dataset.
  + Transfer learning uses the base layers of MobileNet for feature extraction and custom top layers for classification.
* Architecture Modifications
  + Global Average Pooling (GAP) replaces fully connected layers, reducing parameters.
  + Custom Dense layers (128 neurons) with Dropout (rate: 0.5) for classification.
* Hyperparameters
  + Optimizer: Adam optimizer with a learning rate of 0.001.
  + Loss Function: Categorical crossentropy.
  + Batch Size: 32.
  + Epochs: 5.
* Advantages
  + Pre-trained layers provide robust feature extraction.
  + Better generalization on unseen data due to extensive pre-training.

**4.3 Model Evaluation**

* Training and Validation Accuracy
  + Both models are trained on the training subset, and performance is monitored on the validation subset after each epoch.
* Loss Metrics
  + Training and validation loss curves are analyzed to detect underfitting or overfitting.
* Confusion Matrix
  + Provides insights into model performance for each class, highlighting misclassified instances.
  + Helps identify specific classes where the model struggles (e.g., visually similar diseases).
* Comparison of Models
  + Custom CNN vs. MobileNet in terms of accuracy, computational efficiency, and robustness.

**4.4 Experimental Setup**

* Hardware
  + Training was conducted on a GPU-enabled environment to speed up computations.
  + Testing was performed on a standard CPU to simulate real-world deployment.
* Software
  + TensorFlow and Keras were used for model implementation.
  + Python libraries such as NumPy and Matplotlib were utilized for data handling and visualization.

**4.5 Deployment**

* Web Application
  + A Flask-based application allows users to upload an image of a leaf.
  + Users can select either the Custom CNN or MobileNet model for classification.
  + The app returns the predicted disease class along with confidence scores.
* GitHub Repository
  + Contains code, trained models, and documentation.
  + Enables reproducibility and collaborative development.

**4.6 Comparative Analysis**

* Training Speed
  + Custom CNN trained faster (~15 seconds per epoch) due to its smaller architecture.
  + MobileNet required more time (~25 seconds per epoch) due to its deeper architecture.
* Accuracy
  + Custom CNN: Achieved 78.4% validation accuracy, suitable for lightweight applications.
  + MobileNet: Achieved 87.6% validation accuracy, demonstrating superior performance.
* Limitations
  + Custom CNN: Prone to overfitting on small datasets and struggles with visually similar classes.
  + MobileNet: Computationally intensive, making it less ideal for edge deployment without optimization.

**5. Hardware and Software Requirements**

This project requires specific hardware and software resources to ensure smooth execution of tasks like data preprocessing, model training, and deployment.

**Hardware Requirements**

**1. Training Environment**

* **GPU**: NVIDIA Tesla K80 (minimum) or RTX 3080 (recommended) for accelerated training.
* **CPU**: Intel Core i5 (minimum) or i7/Ryzen 7 (recommended).
* **RAM**: 8 GB (minimum) or 16 GB (recommended).
* **Storage**: At least 50 GB free space; SSDs recommended for faster data access.

**2. Deployment Environment**

* **Cloud Hosting**: Platforms like AWS or Google Cloud with GPU support.
* **Local Deployment**: Standard systems with multi-core CPUs and 8 GB RAM for testing.

**3. Testing Environment**

* Any system with Intel Core i3 or better and 8 GB RAM.

**Software Requirements**

**1. Programming Tools**

* **Python**: Version 3.9+ for development.
* **Jupyter Notebook**: For experimentation and iterative coding.

**2. Machine Learning Frameworks**

* **TensorFlow**: For implementing and training models.
* **Keras**: High-level API for constructing deep learning architectures.

**3. Libraries**

* **NumPy and Pandas**: For data manipulation.
* **Matplotlib and Seaborn**: For visualizing data and model performance.
* **OpenCV**: For image preprocessing.

**4. Deployment Tools**

* **Flask**: For creating a web application interface.
* **GitHub**: For hosting code, models, and project documentation.

**5. Optional Tools**

* **Docker**: For containerized deployment.
* **TensorFlow Lite**: For optimizing MobileNet for mobile devices.

**Summary**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Category** | | --- | | **CPU** | | **GPU** | | **RAM** | | **Software** | | | **Minimum** | | --- | | Intel Core i5 or equivalent | | NVIDIA Tesla K80 | | 8 GB | | Python 3.9+, TensorFlow 2.0+, Flask | | | **Recommended** | | --- | | Intel Core i7/Ryzen 7 or higher | | NVIDIA RTX 3080 or higher | | 16 GB or higher | | Docker, TensorFlow Lite (Optional) | |

**6. Experimental Results**

In this section, we provide a comprehensive analysis of the experimental results, which include the training process, model performance evaluation, and comparison of both models used in the project. We also analyze their effectiveness in the task of crop disease detection using leaf images.

**6.1 Training Process and Performance Evaluation**

**Custom CNN Model**

The Custom CNN was trained on the PlantVillage dataset with the following training setup:

* **Training Details**:
  + Epochs: 10
  + Batch Size: 32
  + Optimizer: Adam (learning rate = 0.001)
  + Loss Function: Categorical Crossentropy
  + Metrics: Accuracy
* **Training Progress**:  
  During training, the model showed consistent improvement in accuracy with each epoch. The training accuracy steadily increased, while the validation accuracy also improved but at a slower rate, indicating that the model was learning generalizable features from the dataset.
* **Final Results**:
  + **Training Accuracy**: 85.61%
  + **Validation Accuracy**: 91.97%
  + **Training Loss**: 0.4501
  + **Validation Loss**: 0.2524

The gap between training and validation accuracy indicates that while the model is performing well, there might be some overfitting, though it's not severe.

**MobileNet Model (Pre-trained)**

The MobileNet model was used in a transfer learning setup, where the base layers of MobileNet (pre-trained on ImageNet) were kept frozen, and only the top layers were trained on the PlantVillage dataset.

* **Training Details**:
  + Epochs: 5
  + Batch Size: 32
  + Optimizer: Adam (learning rate = 0.001)
  + Loss Function: Categorical Crossentropy
  + Metrics: Accuracy
* **Training Progress**: MobileNet exhibited a faster convergence rate due to the pre-trained weights, requiring fewer epochs to reach a higher accuracy level. The validation loss decreased rapidly, and the accuracy stabilized quickly.
* **Final Results**:
  + **Training Accuracy**: 87.82%
  + **Validation Accuracy**: 93.64%
  + **Training Loss**: 0.3653
  + **Validation Loss**: 0.1527

MobileNet outperformed the Custom CNN model in terms of accuracy. This is attributed to the power of transfer learning, where MobileNet leverages pre-trained features that are robust to various visual tasks.

**6.2 Model Comparison**

The two models—Custom CNN and MobileNet—were evaluated based on multiple performance metrics to assess their strengths and weaknesses.

**Comparison of Accuracy**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Training Accuracy** | **Validation Accuracy** | **Final Loss (Validation)** |
| Custom CNN | 85.61% | 91.97% | 0.2524 |
| MobileNet | 87.82% | 93.64% | 0.1527 |

* **MobileNet** demonstrated a significant advantage in terms of both training and validation accuracy. This is due to the pre-trained weights and the more complex architecture, which helps in better feature extraction and classification.

**Custom CNN Accuracy Graph**A graph of a line

Description automatically generated

**MobileNet CNN Accuracy Graph**A graph of a line

Description automatically generated

**Comparison of Training Time**

* **Custom CNN**: Training on 10 epochs took approximately **15 seconds per epoch** (total training time: ~2.5 minutes).
* **MobileNet**: Training on 5 epochs took approximately **25 seconds per epoch** (total training time: ~2.08 minutes).

Although MobileNet requires slightly more time per epoch due to its deeper architecture, it converged faster and required fewer epochs, leading to a quicker training process overall in comparison to the Custom CNN.

**Model Complexity and Computational Efficiency**

* **Custom CNN**:
  + **Model Complexity**: 3 layers (shallow architecture)
  + **Parameters**: Fewer parameters due to its small size, leading to faster inference and lower computational load.
  + **Computational Efficiency**: Ideal for edge devices and mobile applications with limited resources.
* **MobileNet**:
  + **Model Complexity**: Deep architecture with transfer learning, resulting in a higher number of parameters.
  + **Parameters**: 4.2 million parameters (much larger than Custom CNN)
  + **Computational Efficiency**: Requires more computational power for training and inference, but this is mitigated by the use of transfer learning and optimizations like depth-wise separable convolutions, which reduce the parameter size.

**Model Robustness**

* **Custom CNN**: Performed well for basic classification tasks but struggled with more visually similar classes (e.g., diseases with subtle differences like Tomato Early Blight vs. Tomato Late Blight).
* **MobileNet**: Showed better robustness against visually similar disease classes, thanks to the pre-trained weights which help the model generalize better to new data.

**6.3 Confusion Matrix Analysis**

To further analyze the performance of both models, confusion matrices were generated for the validation set. The confusion matrix provides insights into which classes were most frequently misclassified by each model.

* **Custom CNN**:  
  The model tended to confuse classes with visually similar symptoms (e.g., **Tomato Early Blight** and **Tomato Late Blight**), leading to a relatively higher misclassification rate in these categories.
* **MobileNet**:  
  MobileNet showed a lower misclassification rate, especially for the visually challenging classes. The model's ability to generalize better contributed to fewer errors in predicting disease classes.

**6.4 Evaluation Metrics**

For a more granular analysis, additional metrics were calculated, such as:

* **Precision**: Measures the accuracy of the model when predicting positive cases for each class.
* **Recall**: Measures how well the model identifies all actual positive cases for each class.
* **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure for performance.

|  |  |  |
| --- | --- | --- |
| **Model** | **Precision** | **Recall** |
| Custom CNN | 0.78 | 0.75 |
| MobileNet | 0.88 | 0.85 |

MobileNet outperformed the Custom CNN in all metrics, showing better performance in detecting and classifying crop diseases.

**6.5 Key Observations and Conclusion from Results**

* **Accuracy**: MobileNet consistently outperformed the Custom CNN, achieving higher accuracy across both training and validation datasets.
* **Training Efficiency**: While MobileNet required more computational resources, its ability to achieve higher accuracy in fewer epochs made it more efficient overall in terms of model performance.
* **Model Robustness**: MobileNet exhibited greater robustness, especially for difficult-to-classify disease types, likely due to the use of pre-trained layers.
* **Computational Trade-offs**: The Custom CNN model, being computationally lighter, could be deployed in environments with limited hardware resources, such as edge devices or mobile phones.

**7. Future Scope**

The future scope of this project encompasses several directions for further development, optimization, and real-world deployment:

1. **Model Optimization for Edge Devices**:
   * While **MobileNet** showed great promise in terms of accuracy, it still requires significant computational resources. Future work could focus on optimizing MobileNet or other models for mobile devices using techniques such as **quantization**, **pruning**, and **TensorFlow Lite**. These techniques help to reduce the model size and computational load, making it more feasible for real-time, on-device predictions.
2. **Improving Data Augmentation and Preprocessing**:
   * Although the dataset used in this study is quite extensive, enhancing data augmentation techniques could help create more diverse training data, improving model robustness. This could include transformations like rotation, flipping, and scaling, which are especially useful when working with images from different sources (e.g., varying camera angles, lighting conditions, and resolutions).
   * Incorporating more advanced preprocessing techniques, such as **histogram equalization** or **color normalization**, could further improve the model’s ability to generalize across different environmental conditions.
3. **Incorporating Additional Sensors**:
   * While the current project focuses on visual data from leaf images, future work could explore the integration of additional sensor data, such as thermal imaging or multispectral data, to further improve disease detection. Combining visual information with data from other sources can provide a more comprehensive understanding of crop health, enabling more accurate predictions.
4. **Real-Time Deployment and Mobile Application**:
   * The project can be expanded into a **real-time mobile application** for farmers. By using a smartphone’s camera and implementing the trained model, the application can instantly detect plant diseases, provide suggestions for treatment, and give farmers access to relevant resources. Incorporating this into a **cloud-based system** could help collect data from various regions, improving disease prediction models over time by enabling a more robust dataset for training.
5. **Cross-Disease and Multi-Crop Models**:
   * Future work could also aim to develop models that not only classify diseases within a particular crop but also **cross-classify** diseases across multiple crops. By combining models trained on different plants, a more generalized disease detection system could be created, which would be useful for applications targeting farmers growing a variety of crops.
6. **Continuous Learning**:
   * Implementing **online learning** or **incremental learning** techniques would allow the model to adapt and improve over time. As new data is collected from users (farmers), the model could be retrained regularly to ensure that it continues to accurately detect newly emerging diseases or variations within existing diseases.
7. **Integrating with Decision Support Systems**:
   * The output of the disease classification model can be used in conjunction with decision support systems for farmers. Such systems can provide actionable insights, such as the optimal time for applying pesticides or fertilizers, thus contributing to sustainable farming practices and reducing the environmental impact of agriculture.
8. **Exploring Other Advanced Architectures**:
   * Beyond MobileNet and Custom CNN, exploring other advanced neural architectures, such as **ResNet** or **DenseNet**, could potentially lead to better results in crop disease classification tasks. Experimenting with architectures known for their high accuracy on large image datasets might offer improvements in the detection of subtle disease patterns.

**8. Conclusion**

This project aimed to develop a machine learning model for crop disease detection using leaf images, leveraging two different approaches: a custom convolutional neural network (CNN) and a pre-trained MobileNet model. Both models were evaluated on the PlantVillage dataset, containing over 50,000 labeled images across 38 crop disease categories. Here are the key conclusions drawn from the study:

1. **Performance Comparison**:
   * **MobileNet** outperformed the **Custom CNN** in almost all evaluation metrics, including accuracy, precision, recall, and F1-score. This is primarily due to MobileNet’s use of transfer learning, where the model benefits from pre-trained weights on the ImageNet dataset. This allowed it to generalize better to new datasets and handle the complex features of crop diseases more effectively.
   * **Custom CNN**, though simpler and computationally lighter, had a more limited ability to capture the intricate patterns needed for accurate classification, particularly for visually similar diseases. However, its reduced computational requirements make it suitable for applications with limited hardware resources.
2. **Training and Deployment Considerations**:
   * **MobileNet** required more computational power and time to train, due to its deeper architecture and the need for transfer learning. Despite this, it converged faster, requiring fewer epochs to reach optimal performance, which makes it a better choice for real-time or high-accuracy applications.
   * The **Custom CNN** model was more computationally efficient, requiring less time per epoch and being better suited for edge devices and mobile deployment where computational resources might be constrained.
3. **Model Robustness**:
   * MobileNet's ability to handle a wide range of image variations and its greater robustness against overfitting were critical in producing better results for challenging classes, such as distinguishing between diseases that have visually similar symptoms.
   * The **Custom CNN** model, with fewer parameters and simpler architecture, struggled with diseases that were harder to differentiate. This highlighted the importance of model complexity for complex real-world image classification tasks.
4. **Practical Application in Agriculture**:
   * The success of the models in this task shows significant promise for the use of machine learning in crop disease detection. Accurate and efficient identification of plant diseases could lead to improved early diagnosis, reduced crop losses, and better yield prediction, which is crucial for ensuring food security in the face of global challenges.
5. **Limitations**:
   * One limitation of this study was the reliance on only two models. While both models performed well, further research with more advanced models or hybrid approaches could potentially improve accuracy even further.
   * The dataset used in this project is extensive, but real-world scenarios may present challenges such as images taken in different lighting conditions, different stages of disease, and varying image quality from mobile devices. These factors were not fully simulated in the current study.

**9. Github Repository**

[**https://github.com/VipinYadav16/Multimodel-Plant-Disease-Detection**](https://github.com/VipinYadav16/Multimodel-Plant-Disease-Detection)