Turnitin Originality Report

Processed on: 06-Oct-2024 22:16 IST
ID: 2476587695
Word Count: 3331
Submitted: 1

DL_Assignment.pdf By sanjana dinith

Similarity Index	Similarity by Source	
11%	Internet Sources:	7%
	Publications:	8%
	Student Papers:	7%

1% match (publications)
Mehdi Ghayoumi. "Generative Adversarial Networks in Practice", CRC Press, 2023

1% match (Arild Bergesen Husebo, Huynh Van Khang, Witold Pawlus. "Diagnosis of Incipient Bearing Faults using Convolutional Neural Networks", 2019 IEEE Workshop on Electrical Machines Design, Control and Diagnosis (WEMDCD), 2019)
Arild Bergesen Husebo, Huynh Van Khang, Witold Pawlus. "Diagnosis of Incipient Bearing Faults using Convolutional Neural Networks", 2019 IEEE Workshop on Electrical Machines Design, Control and Diagnosis (WEMDCD), 2019

1% match (student papers from 08-Sep-2024)
Submitted to University of East London on 2024-09-08

1% match (Internet from 02-Jun-2024)
https://wiredspace.wits.ac.za/server/api/core/bitstreams/7f33a5f6-f552-44c6-b364-de3d5b0ef333/content

1% match (student papers from 06-Oct-2024)
Submitted to University of Technology, Sydney on 2024-10-06

1% match (student papers from 14-Mar-2024)
Submitted to Indian Institute of Mangement Rachi Suchana Bhawan on 2024-03-14

< 1% match (Amin Salih Mohammed. "Efficient breast cancer classification using LS-SVM and dimensionality reduction", Soft Computing, 2023)
Amin Salih Mohammed. "Efficient breast cancer classification using LS-SVM and dimensionality reduction", Soft Computing, 2023

< 1% match (student papers from 17-May-2023)
Submitted to Loughborough University on 2023-05-17

< 1% match (Internet from 27-Sep-2023)
https://usercentrics.com/knowledge-hub/artificial-intelligence-ai-and-consent/

< 1% match (student papers from 18-May-2024)
Submitted to UNICAF on 2024-05-18

< 1% match (Internet from 07-Sep-2024)
https://ebin.pub/ai-and-ml-for-coders-a-comprehensive-guide-to-artificial-intelligence-and-machine-learning-techniques-tools-real-world-applications-and-ethical-considerations-for-modern-programmers.html

< 1% match (Internet from 20-Jun-2024)
https://www.scirp.org/xml/131543.xml

< 1% match (student papers from 09-Sep-2024)
Submitted to National University of Ireland, Galway on 2024-09-09

< 1% match (student papers from 11-Dec-2023)
Submitted to Wageningen University on 2023-12-11

< 1% match (Internet from 08-Nov-2023)
https://arxiv.org/pdf/2311.02338.pdf

< 1% match (Internet from 20-Mar-2010)
http://www.ehow.com/facts_6021591_tomato-plant-fungus.html

< 1% match (Internet from 25-Dec-2022)
https://pure.tue.nl/ws/files/142604968/Master_thesis_Sander_Boelders_v1.1.pdf

< 1% match (student papers from 17-May-2024)
Submitted to La Trobe University on 2024-05-17

< 1% match ()
Meihui Li, Yufei Gong, Zhaohui Zheng. "Finger Vein Identification Based on Large Kernel Convolution and Attention Mechanism", Sensors (Basel, Switzerland)

< 1% match (publications)
Narendra Khatri, Ajay Kumar Vyas, Celestine Iwendi, Prasenjit Chatterjee. "Precision Agriculture for Sustainability - Use of Smart Sensors, Actuators, and Decision Support Systems", Apple Academic Press, 2024

< 1% match ()
Nambiar, Nilanjana. "Predicting COVID-19 Fake News", Bryant Digital Repository, 2022

< 1% match (student papers from 26-May-2024)
Submitted to The British College on 2024-05-26

< 1% match (Internet from 23-Aug-2022)
https://www.aphis.usda.gov/brs/aphisdocs/14_09301p.pdf

< 1% match (Internet from 11-Aug-2024)
https://www.geeksforgeeks.org/lightgbm-model-evaluation-metrics/

Sri Lanka Institute of Information Technology Year 4, Semester I, 2024 Assignment Potato Disease Classification Using CNN Architectures on the Plant Village Dataset Deep Learning (SE4050) BSc (Hons) in Information Technology Specializing in Software Engineering Group Details Student Registration Number Student Name Group IT21251900 Rajapaksha R.M.S.D. Y4.S1.WE.SE.01 IT21302862 Sri Samadhi L.A.S.S. Y4.S1.WE.SE.02 IT21178054 Kumari T.A.T.N. Y4.S1.WD.IT.01 IT21360428 Monali G.M.N. Y4.S1.WE.SE.02 GitHub Link IT21251900/DL-Project (github.com) YouTube Presentation Link https://www.youtube.com/playlist?list=PLOC02hssbXJ99RsQVUi8lepMcHTRT8xwH Table of Contents Introduction

37 Introduction Problem Statement Potato crops are a crucial agricultural commodity, feeding millions worldwide. However, they are susceptible to various diseases, which can drastically affect yield and quality. Among the most notorious are Early Blight and Late Blight, both of which can cause significant losses if left unchecked. Identifying these diseases early and accurately is essential for effective intervention and treatment. In this project, we tackle the problem of classifying potato leaf diseases using image data and deep learning techniques. Specifically, we employ supervised learning methods to classify leaf images into categories such as "Healthy," "Early Blight," and "Late Blight." This project leverages the Plant Village dataset, which is widely recognized for plant disease classification research, offering a large and diverse set of annotated images. By building robust deep learning models, the system aims to automate the detection of these diseases from images of potato leaves. Objective The goal of this project is to develop a machine learning solution to classify potato leaf diseases. Specifically, we aim to: • Develop and Compare CNN Models: Train and evaluate four distinct Convolutional Neural Networks (CNNs) for accuracy and generalization. • Provide a Disease Detection API: Deploy a FastAPI backend to process image uploads and provide real-time disease predictions. • Deploy a React Frontend: Build a user-friendly React interface for easy image uploads and viewing of classification results. Supervised Learning Approach Supervised learning was chosen for this project because the problem we are tackling involves labeled data, where the input images are associated with known categories (e.g., healthy or diseased classes). In supervised learning, the model is trained to learn the mapping between the input data and its corresponding labels, allowing it to predict the correct class when new, unseen data is presented. This fits well with our objective, which is to classify images into predefined categories based on their features. The key reasons for choosing supervised learning are: 1. Labeled Dataset: Our dataset comes with ground truth labels, meaning each input image is associated with a known class. Supervised learning algorithms are specifically designed to work with labeled data, learning from examples and improving their performance over time. 2. High Predictive Accuracy: Supervised learning techniques, especially convolutional neural networks (CNNs), have been widely used in image classification tasks and have shown superior performance. CNNs are highly effective in automatically learning spatial hierarchies of features from images, which are crucial in our classification task. 3. Clear Performance Metrics: Since supervised learning involves labeled data, it is easier to evaluate the model's performance through established metrics like accuracy, precision, recall, and F1-score. These metrics provide a clear understanding of how well the model generalizes to unseen data. 4. Control Over Training Process: Supervised learning allows fine-tuning of the model by adjusting hyperparameters and providing more control over the training process. This flexibility ensures that we can iteratively improve the model's performance by adjusting the architecture, learning rate, and other factors. Background Information on Supervised Learning and CNN Supervised learning is a machine learning approach where models learn from labeled datasets. In the context of this project, the labels represent different categories of plant diseases, and the input data consists of images. CNNs, a type of deep learning model particularly well-suited for image classification tasks, are employed in this project. CNNs automatically extract features from images, learning patterns such as edges, textures, and colors, which are crucial for identifying plant diseases. CNNs consist of several layers: • Convolutional layers: Extract features using convolution operations. • Pooling layers: Reduce the spatial dimensions of the data. • Fully connected layers:

Combine features to classify images into the correct category. This architecture is ideal for the classification of plant disease images, where intricate patterns in leaf textures and colors need to be identified to distinguish between healthy plants and various diseases. Scope • Data Preprocessing: Image cleaning, resizing, normalizing, and augmentation using the Plant Village dataset. • Model Development: Exploring four CNN architectures with techniques like batch normalization, dropout, and transfer learning. • API Development: FastAPI will be used to deploy trained models for real-time image classification. • Frontend Development: A React-based application will allow users to upload images and receive predictions. Dataset Overview Dataset Description The dataset used for this project is the Plant Village Dataset, specifically focusing on potato diseases. This dataset is well-suited for tasks like image classification in plant pathology, as it contains high-quality images of plant leaves, labeled for various diseases. • Dataset Name: Plant Village Dataset (Potato Disease Subset). • Size: The subset contains 2152 images across 3 classes. These classes include different plant diseases, but for the purpose of this project, we focus on images of potato leaves with three specific labels: 1. Healthy 2. Early Blight 3. Late Blight Each image represents a close-up shot of a potato leaf, either healthy or affected by one of these two diseases. The large number of images allows for robust training, testing, and validation of the models, reducing the chances of overfitting. • Source: The dataset is publicly available on Kaggle. It is one of the most widely used datasets in plant disease classification tasks due to its diversity and quality. • Image Format: All images are colored and provided in JPEG format. They vary in size and quality, necessitating preprocessing steps like resizing and normalization. • Classes of Interest: o Healthy: These images depict healthy potato leaves without any visible signs of disease. o Early Blight: Caused by the fungus Alternaria solani, Early Blight appears as small dark brown or black spots on the leaves, often surrounded by a yellow halo. o Late Blight: Caused by the oomycete Phytophthora infestans, Late Blight manifests as large dark patches on leaves, with a water-soaked appearance. This disease is particularly devastating and can destroy crops rapidly if left unchecked. • Attributes: o Input: Images of size 256x256, with 3 color channels (RGB). o Output: Labels corresponding to disease types. Feature Selection and Preprocessing Techniques Feature selection for image data involves the extraction of visual features such as textures, shapes, and colors. In CNNs, this is done automatically by the convolutional layers. The preprocessing steps we applied to the dataset include: • • Resizing: All images were resized to a uniform size of 256x256 pixels. Normalization: Pixel values were scaled to the range [0, 1] to improve model convergence during training. • Data Augmentation: Techniques such as random rotation, flipping, and zooming were applied to artificially expand the dataset and improve model generalization. Challenges Despite the benefits of the Plant Village dataset, several challenges arose during model training: • Class Imbalance: The "Healthy" class is overrepresented, leading to biased models. This was addressed with data augmentation and a tuned loss function. • Image Quality: Variations in lighting and resolution affected consistency. Normalization and preprocessing were applied to mitigate these issues. • Leaf Occlusion: Some images had partially occluded leaves or overlapping patterns, complicating the identification of disease symptoms. Model Architectures Model 1 This model uses a Sequential CNN architecture with six Conv2D layers, which is effective for image classification tasks. CNNs are chosen for their ability to automatically detect important features such as edges, textures, and patterns in images. • Layers Chosen: Multiple Conv2D layers followed by MaxPooling layers enable downsampling and feature extraction. The model flattens before the Dense layers to connect to the final classification task. • Activation Functions: ReLU is used for all convolutional layers because of its ability to mitigate the vanishing gradient problem. The Softmax function is applied at the output layer for multi-class classification. This model is a more complex CNN with five Conv2D layers, using Adam as the optimizer for faster convergence. • Layers Chosen: With increasing filter sizes, each convolutional layer captures progressively more complex patterns. MaxPooling downscales feature maps to reduce computational complexity. • Activation Functions: Again, ReLU for hidden layers and Softmax for the final output layer due to the classification task. This model includes Batch Normalization, which stabilizes learning and improves performance in deep architectures. • Layers Chosen: Conv2D layers capture image features, and Batch Normalization ensures stable and faster training. • Activation Functions: ReLU for hidden layers, Softmax for output. This model focuses on generalization using four convolutional layers and dropout to prevent overfitting. • Layers Chosen: Each Conv2D block is followed by MaxPooling, batch normalization, and Dropout for regularization. • Activation Functions: ReLU and Softmax for classification. Results Model 1 • Test Accuracy: Achieved a test accuracy of 97.63% after 50 epochs.. • Performance: The model showed strong performance due to multiple Conv2D layers that captured diverse image features effectively. Loss decreased steadily during training, indicating good model convergence. • Challenges: There was a noticeable improvement in validation accuracy across epochs, with minimal overfitting thanks to the inclusion of data augmentation. • Test Accuracy: Achieved a test accuracy of 99.36% after 50 epochs.. • Performance: Model 2 performed the best among the four models, likely due to its deeper architecture with five Conv2D layers. The increasing filter sizes and more complex feature extraction likely contributed to this high accuracy. • Challenges: The model required a longer training time due to the larger number of parameters (over 4 million), but this led to significant accuracy improvements. • Test Accuracy: Achieved a test accuracy of 76.42% after 50 epochs. • Performance: Although the architecture was simpler than Model 2, the inclusion of Batch Normalization improved training stability. However, the performance was lower compared to Model 2, likely due to underfitting or insufficient feature extraction in the deeper layers. • Challenges: The model struggled to generalize on the validation set, suggesting that additional tuning or a more complex architecture might be required for better performance. • Training Accuracy: Achieved a test accuracy of 97.33% after 50 epochs. • Performance: This model showed potential but demonstrated underfitting, with the validation accuracy lagging behind the training accuracy by a significant margin. This suggests that either the model requires more epochs to train or more advanced data augmentation techniques to improve generalization. • Challenges: The model's simplicity compared to Models 1 and 2 may have contributed to its lower accuracy. Additionally, it might have benefitted from more complex feature extraction or regularization methods to reduce overfitting. Comparison of Results Model 2 achieved the highest test accuracy of 99.36%, making it the best performer. Its deeper architecture with five Conv2D layers and more complex feature extraction likely contributed to this superior performance. Despite the larger number of parameters and longer training time, it delivered the most accurate results. Model 1 followed closely with a test accuracy of 97.63%. It showed strong performance with minimal overfitting, due to effective data augmentation techniques. Its six Conv2D layers allowed it to capture diverse image features efficiently. Model 4 achieved a test accuracy of 97.33%, slightly lower than Model 1. This model demonstrated potential but exhibited signs of underfitting, where the validation accuracy lagged behind the training accuracy. It may have benefitted from more training epochs or advanced data augmentation to improve generalization. Model 3 had the test accuracy of 76.42% which is lower than comparing with other 3 models. While Batch Normalization improved training stability, the simpler architecture resulted in insufficient feature extraction, leading to underfitting. This model struggled to generalize well compared to the others, indicating the need for architectural improvements or additional tuning. Training & Validation Accuracy and Loss • Model Performance: It helps assess how well the model generalizes to unseen data. • Overfitting Detection: A large gap between training and validation metrics signals overfitting, where the model performs well on training data but poorly on new data. • Hyperparameter Tuning: Tracking these metrics informs adjustments to improve model performance, such as tuning learning rate or batch size. • Early Stopping: Monitoring allows for early stopping to prevent overfitting and save computational resources. Confusion Matrix Visualizing training and validation metrics plays a crucial role in understanding how a machine learning model evolves during training. To ensure effective learning and avoid overfitting, it's essential to monitor these metrics closely through graphical representations. In this project, Matplotlib was used to create detailed visualizations that illustrate training accuracy, validation accuracy, training loss, and validation loss over time. These plots provide a clear visual summary of the model's performance, making it easier to spot trends, identify overfitting or underfitting, and understand the impact of hyperparameter adjustments. All team members relied on these visualizations to assess and improve the performance of their individual models designed to classify potato leaf diseases. The graphical representations served as an indispensable tool for evaluating model convergence, guiding model adjustments, and ensuring that each model maintained a healthy balance between accuracy and generalization. • Potato Early Blight: Predicted 110 correctly, 4 misclassified as Potato Late Blight, 0 misclassified as Healthy. • Potato Late Blight: Predicted 122 correctly, 2 misclassified as Healthy, 0 misclassified as Early Blight. • Potato Healthy: Predicted 17 correctly, 1 misclassified as Late Blight. • Potato Early Blight: Predicted 110 correctly, 0 misclassified as Potato Late Blight, 0 misclassified as Healthy. • Potato Late Blight: Predicted 128 correctly, 1 misclassified as Healthy, 0 misclassified as Early Blight. • Potato Healthy: Predicted 17 correctly, 1 misclassified as Late Blight. • Potato Early Blight: Predicted 134 correctly, 48 misclassified as Potato Late Blight, 6 misclassified as Healthy. • Potato Late Blight: Predicted 44 correctly, 10 misclassified as Healthy, 48 misclassified as Early Blight. • Potato Healthy: Predicted 11 correctly, 3 misclassified as Late Blight, 6 misclassified as Early Blight. • Potato Early Blight: Predicted 129 correctly, 5 misclassified as Potato Late Blight, 0 misclassified as Healthy. • Potato Late Blight: Predicted 102 correctly, 0 misclassified as Healthy, 0 misclassified as Early Blight. • Potato Healthy: Predicted 17 correctly, 2 misclassified as Late Blight, 1 misclassified as Early Blight. Overall Observation • Model 1 and Model 2 performed consistently well, with minimal misclassifications across all classes. • Model 3 struggled significantly with Potato Late Blight predictions, misclassifying many cases as Early Blight. • Model 4 had generally good performance, though it had some misclassifications for Early Blight and Late Blight. In conclusion, Model 2 seems to offer the most balanced performance across all categories, while Model 3 may need further refinement to handle the Late Blight class more effectively. Critical Analysis and Discussion How Accuracy Could Be Improved • Hyperparameter Tuning: Adjusting learning rates, batch sizes, and the number of epochs could further improve accuracy. For instance, increasing the number of epochs for Model 4 could help prevent underfitting. • More Data Augmentation: Increasing the variety of augmentations such as zoom, brightness, and contrast adjustments could help improve generalization, especially for underperforming models like Model 4. • Regularization: Implementing techniques like L2 regularization could help prevent overfitting in high-capacity models like Model 2. • Pre-trained Models: Using a pre-trained architecture such as VGG16 or ResNet could improve performance, especially in complex classification tasks with limited training data. Possible Future Work • Transfer Learning: Implementing transfer learning with pre-trained models could yield higher accuracy with fewer training epochs. • Ensemble Methods: Combining predictions from multiple models in an ensemble could improve overall performance by leveraging the strengths of each individual model. • Optimization Algorithms: Experimenting with different optimizers such as RMSprop or SGD could fine-tune model performance and convergence. • Further Hyperparameter Optimization: Automated techniques like Grid Search or Random Search could be used to systematically explore a wider range of hyperparameters. Potato Disease Identification API Overview The Potato Disease Identification API is a web-based API developed using FastAPI. This API utilizes a machine learning model to identify diseases in potato leaves by analyzing uploaded images. The API can classify potato leaf images into three categories: Early Blight, Late Blight, or Healthy. The application is designed to provide quick and accurate predictions, assisting farmers and agricultural professionals in managing crop health. How It Works 1. Image Upload: Users can upload an image of a potato leaf to the /predict endpoint. 2. Preprocessing: The uploaded image is resized and normalized to meet the input requirements of the machine learning model. 3. Model Prediction: A pre-trained TensorFlow model processes the image and predicts the class of disease or indicates that the plant is healthy. 4. Response: The API returns the predicted disease class along with a confidence score. API Endpoints 1. Health Check • Endpoint: /ping • Method: GET • Description: A simple health check endpoint to verify if the API is running. Response: 2. Predict Disease • Endpoint: /predict • Method: POST • Description: Upload an image of a potato leaf to receive a prediction about its health status. Request: • Form Data: o file: The image file of the potato leaf (must be in a supported image format). Response: Early Blight Disease Identification. Late Blight Disease Identification React Application The React application interacts with the FastAPI to facilitate the image upload process and display the results. Below are the key functionalities implemented in the React app: Features 1. User Interface: o The app includes a user-friendly interface for uploading images of potato leaves. 2. API Calls: o The React application makes asynchronous calls to the FastAPI endpoints using Axios or Fetch API. o When the user uploads an image, the app sends a POST request to the /predict endpoint. 3. Displaying Results: o Upon receiving the response from the API, the React application displays the predicted class and confidence score to the user. o The results are shown in a structured format, allowing users to understand the health status of their potato plants at a glance. Early Blight Disease Identification. Healthy Plants Identification Late Blight Disease Identififcation Workflow 1. Upload Image: The user selects a potato leaf image and clicks the "Upload" button. 2. Send Request: The React app sends the image to the /predict endpoint. 3. Receive Response: The API processes the image and returns the prediction. 4. Display Prediction: The React application displays the predicted disease class and confidence score. To run the application 1. run the backend server(FasterAPI - potato-disease-application-python-api) 2. set the environment $env:NODE_OPTIONS="--openssl-legacy-provider" 3. Install the dependencies npm install npm audit fix Conclusion In this study of potato disease identification using various CNN architectures, Model 2 demonstrated the highest accuracy at 99.36%, showcasing the effectiveness of deeper architectures and complex feature extraction. Model 1 and Model 4 also performed well with accuracies of 97.63% and 97.33%, respectively, although Model 4 exhibited signs of underfitting. In contrast, Model 3 underperformed with a test accuracy of 76.42%, indicating a need for further architectural enhancement. Overall, these results highlight the importance of model complexity and the balance between training efficiency and generalization in achieving high performance in image classification tasks. References Plant Village Dataset : https://www.kaggle.com/datasets/arjuntejaswi/plant-village Appendix Contribution IT21251900 IT21302862 IT21178054 IT21360428 Model 2 Model 3 Model 4 Model 2 Model 3 Model 4 Model 1 Model 2 Model 3 Model 4 Model 1 Model 2 Model 3 Model 4