ADDIS ABABA UNIVERSITY
ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

MSc. in ARTIFICIAL INTELLIGENCE

Binary Classification for Insects

Samuel Hailemariam GSE/2031/15, Eyob Teshale GSE/7570/15
Mohammednur Ibrahim GSE/8650/15

Submitted to:

Dr. Natnael Argaw (PhD)

February 2024

1 Introduction

In the realm of computer vision and machine learning, the accurate classification of images is a fundamental task with numerous practical applications. One such application is the classification of insects, which is essential for various fields including agriculture, entomology, and environmental monitoring. This project aims to develop and deploy a machine learning model capable of distinguishing between two common insects: ants and bees.

1.1 Motivation

The motivation behind this project stems from the importance of insect classification in various domains. In agriculture, the identification of pests like ants and bees can aid farmers in implementing targeted pest management strategies, thereby enhancing crop yields and reducing the reliance on pesticides. Furthermore, in ecological studies, understanding the population dynamics of insects such as ants and bees contributes to our comprehension of ecosystem health and biodiversity.

1.2 Goals and Objectives

The primary goal of this project is to build a robust machine learning model capable of accurately classifying images of ants and bees. Specifically, the objectives include:

- Model Development: Develop Convolutional Neural Network (CNN) models for binary classification using TensorFlow/Keras. This involves experimenting with different architectures, hyperparameters, and optimization techniques to achieve optimal performance.
- Data Augmentation: Utilize data augmentation techniques to augment the training dataset, thereby improving the model's generalization ability and resilience to variations in input images.
- Transfer Learning: Investigate the effectiveness of transfer learning by fine-tuning a pre-trained VGG16 model on the insect dataset. Transfer learning can leverage features learned from a large dataset (ImageNet) to enhance the classification performance on the target task.
- Model Evaluation: Evaluate the trained models using standard metrics such as accuracy and loss on separate training, validation, and test datasets. This ensures reliable performance assessment and identifies potential overfitting issues.
- Explainability: Employ explainability techniques, such as Lime, to provide insights into the model's decision-making process. Understanding why the model classifies an image as either an ant or a bee is crucial for building trust and interpreting its predictions.

1.3 Related Work

Previous research in the field of insect classification has predominantly focused on traditional image processing techniques and manual feature extraction. However, with the advancements in deep learning and convolutional neural networks, there has been a paradigm shift towards end-to-end learning approaches, which have demonstrated superior performance in image classification tasks.

1.4 Contributions of the Project

- Development of CNN Models: This project contributes novel CNN architectures tailored for the specific task of insect classification, providing valuable insights into effective model design for similar classification tasks.
- Integration of Transfer Learning: By incorporating transfer learning with the VGG16 model, this project explores the efficacy of leveraging pre-trained models for insect classification, potentially reducing the need for large annotated datasets and computational resources.
- Deployment of Explainability Techniques: The project goes beyond model prediction by integrating explainability techniques like Lime, enabling users to interpret and trust the model's decisions, fostering transparency and accountability in the classification process.

2 Problem Statement

The problem addressed by this project is the accurate classification of images depicting ants and bees. Insect classification is a challenging task due to the visual similarities between different species, variations in appearance within species, and the presence of background clutter in natural environments. The lack of robust automated methods for insect identification hinders applications in agriculture, entomology, and ecological studies, where timely and accurate classification is crucial for decision-making processes.

Specifically, the problem entails developing machine learning models capable of distinguishing between ants and bees based on visual cues extracted from images. Traditional approaches relying on manual feature extraction and rule-based classification systems are labor-intensive, time-consuming, and often lack scalability. Furthermore, the complexity of insect morphology and the variability in pose, lighting conditions, and background clutter pose significant challenges for automated classification systems.

The goal is to leverage advancements in deep learning and convolutional neural networks (CNNs) to address these challenges and develop accurate, robust, and scalable models for insect classification. By training CNN models on labeled datasets of ant and bee images, the project aims to achieve high classification accuracy while ensuring the model's ability to generalize to unseen data.

Key components of the problem statement include:

- Image Classification: The task involves categorizing input images into two classes: ants and bees. Each class represents a distinct insect species with unique visual characteristics.
- Challenges: The problem is compounded by various challenges, including intra-class variations, inter-class similarities, occlusions, background clutter, and variations in image quality.
- Automation: The objective is to develop automated machine learning models capable of accurately classifying insects without the need for manual intervention or human expertise.
- Scalability: The proposed solution should be scalable to handle large volumes of image data efficiently, allowing for real-time or near-real-time classification in practical applications.

Addressing the problem statement requires a multidisciplinary approach combining expertise in computer vision, deep learning, image processing, and domain knowledge in entomology. By overcoming the challenges associated with insect classification, the project aims to contribute to advancements in agricultural pest management, ecological monitoring, and biodiversity conservation efforts.

2.1 Methodology

The methodology employed in this project involves a combination of data preprocessing, model development, training, evaluation, and deployment stages. The key methods and techniques used are outlined below:

1. Data Preprocessing
    - Image Loading: Input images depicting ants and bees are loaded from the specified directories using TensorFlow's ImageDataGenerator.
    - Data Augmentation: To increase the diversity of the training dataset and improve model generalization, data augmentation techniques such as rotation, shifting, shearing, zooming, and flipping are applied to the training images.
    - Normalization: The pixel values of the images are rescaled to the range [0, 1] to facilitate faster convergence during model training.
2. Model Development
    - CNN Architecture: Convolutional Neural Network (CNN) architectures are designed using TensorFlow/Keras. Both custom CNN models and transfer learning with the VGG16 pre-trained model are explored.
    - Layer Configuration: The CNN models consist of convolutional layers for feature extraction, max-pooling layers for spatial downsampling, and fully connected layers for classification.

- Activation Functions: Rectified Linear Unit (ReLU) activation is used in convolutional layers, while Sigmoid activation is employed in the output layer for binary classification.
- Dropout: Dropout regularization is incorporated to mitigate overfitting by randomly dropping a fraction of neurons during training.

3. Model Training

   - Compilation: The models are compiled with appropriate loss functions (binary cross-entropy) and optimizers (e.g., Adam) to minimize the training loss.
   - Batch Processing: Training is performed in mini-batches to efficiently utilize computational resources and facilitate parallel processing.
   - Epochs: Training is conducted over multiple epochs, with the number of epochs determined through experimentation and early stopping criteria.

4. Evaluation Metrics

   - Accuracy: The primary evaluation metric used to assess the model's performance is classification accuracy, which measures the proportion of correctly classified instances.
   - Loss: The training and validation loss curves are monitored to assess the convergence and generalization capabilities of the models.
   - Confusion Matrix: Additionally, the confusion matrix is computed to analyze the distribution of true positives, true negatives, false positives, and false negatives, providing insights into the model's behavior across different classes.

5. Experimental Setup

   - Dataset Splitting: The dataset is divided into training, validation, and test sets to facilitate model training, validation, and evaluation, respectively.
   - Hyperparameter Tuning: Hyperparameters such as learning rate, batch size, dropout rate, and model architecture are tuned using techniques such as grid search or random search to optimize model performance.
   - Cross-Validation: Cross-validation techniques may be employed to assess the robustness of the models and mitigate issues related to dataset bias or variance.

6. Model Deployment

   - Model Saving: Trained models with optimal performance are saved in a portable format (e.g., HDF5) for future use and deployment.

- Web Application: A Streamlit web application is developed to interactively deploy the trained models, allowing users to upload images for classification and interpretation of predictions using Lime.
- By following this methodology, the project aims to develop accurate and interpretable models for insect classification while adhering to best practices in machine learning experimentation and evaluation. Iterative refinement of the methodology based on experimental results and feedback enables continual improvement in model performance and applicability.

2.2 Proposed Solution

The proposed solution addresses the problem of accurate classification of ants and bees through the development and deployment of machine learning models tailored for insect classification tasks. The solution consists of several key components working together to achieve the desired outcome:

1. Data Acquisition and Preprocessing
    - Image Dataset: An extensive dataset comprising images of ants and bees is collected from reliable sources or through data scraping methods.
    - Data Preprocessing: The acquired images undergo preprocessing steps such as resizing, normalization, and data augmentation to enhance the quality and diversity of the dataset. Data augmentation techniques help in creating variations of the original images, which aids in robust model training and generalization.
2. Model Development
    - Custom CNN Models: Convolutional Neural Network (CNN) architectures specifically designed for insect classification are developed using TensorFlow/Keras. These models consist of convolutional layers for feature extraction followed by fully connected layers for classification.
    - Transfer Learning: Transfer learning is explored by leveraging pre-trained CNN models such as VGG16, which have been trained on large-scale image datasets like ImageNet. Fine-tuning techniques are applied to adapt the pre-trained models to the task of insect classification.
3. Training and Evaluation
    - Model Training: The developed models are trained on the preprocessed dataset using appropriate optimization algorithms and loss functions. Training is conducted over multiple epochs, with periodic evaluation on a validation set to monitor model performance and prevent overfitting.

- Evaluation Metrics: Model performance is evaluated using standard metrics such as accuracy, loss, and confusion matrices. These metrics provide insights into the model's ability to correctly classify ants and bees while avoiding misclassifications.

4. Explainability and Interpretability

   - Lime Integration: To enhance model interpretability, Lime (Local Interpretable Model-agnostic Explanations) is integrated into the solution. Lime provides local explanations for individual predictions made by the model, highlighting the important features contributing to each classification decision.
   - Visualization: Lime-generated explanations are visualized alongside the original images, allowing users to understand and interpret the model's predictions with confidence.

5. Model Deployment

   - Streamlit Web Application: The trained models, along with Lime-based explainability, are deployed in a user-friendly Streamlit web application. Users can upload images of ants or bees to the application, which then predicts the insect type and provides explanations for the predictions.
   - Real-time Inference: The deployed models support real-time inference, enabling users to obtain instant feedback on the classification of uploaded images.

6. Iterative Improvement

   - Feedback Loop: The proposed solution operates within an iterative improvement framework, where feedback from users and model performance metrics guide further refinements. This feedback loop facilitates continuous enhancement of model accuracy, interpretability, and user experience.
   - By integrating these components into a cohesive solution, the proposed approach aims to deliver accurate, interpretable, and user-friendly insect classification capabilities, with potential applications in agriculture, entomology, environmental monitoring, and beyond.

3 System Architecture

The system architecture is designed to facilitate the seamless integration and interaction of various components involved in insect classification, explainability, and deployment. It comprises the following key components:

1. Data Pipeline

- Data Acquisition: An image dataset containing pictures of ants and bees is acquired from reliable sources or generated through data scraping methods.
- Preprocessing: The acquired images undergo preprocessing steps such as resizing, normalization, and data augmentation to prepare them for model training.

2. Model Development

   - Custom CNN Models: Convolutional Neural Network (CNN) architectures tailored for insect classification are developed using TensorFlow/Keras. These models are trained on the preprocessed dataset to learn discriminative features for distinguishing between ants and bees.
   - Transfer Learning: Alternatively, pre-trained CNN models such as VGG16 can be employed for insect classification. Transfer learning techniques are applied to fine-tune these models on the target insect dataset.

3. Training and Evaluation

   - Training Module: The training module trains the custom CNN models or fine-tunes the pre-trained models on the insect dataset. It utilizes optimization algorithms and evaluates the model's performance using evaluation metrics such as accuracy, loss, and confusion matrices.
   - Evaluation Module: This module assesses the trained models' performance on validation and test datasets, providing insights into their generalization capabilities and potential overfitting issues.

4. Explainability Module

   - Lime Integration: The Lime (Local Interpretable Model-agnostic Explanations) framework is integrated into the system to provide local explanations for individual predictions made by the models. Lime highlights the important features contributing to each classification decision, enhancing model interpretability.

5. Deployment

   - Streamlit Web Application: The trained models, along with the Lime-based explainability module, are deployed in a user-friendly Streamlit web application. Users can upload images of ants or bees through the web interface, which then predicts the insect type and provides explanations for the predictions in real-time.
   - Real-time Inference: The deployed models support real-time inference, allowing users to obtain immediate feedback on the classification of uploaded images without significant latency.

6. User Interface

    - Streamlit Interface
        - The Streamlit web application serves as the user interface, providing an intuitive and interactive platform for users to interact with the deployed models. Users can upload images, view predictions, and explore explanations generated by the Lime framework.
    - Interaction Flow
        - Users interact with the system through the Streamlit web application by uploading images of ants or bees.
        - The uploaded images are processed by the deployed models, which predict the insect type (ant or bee) and generate explanations using Lime.
        - The predictions and explanations are displayed to the users in the web application interface, allowing them to interpret the model's decisions and gain insights into the classification process.

4 Implementation Details

The implementation of the proposed system involves several technical considerations, including hardware specifications, software dependencies, and development environments. Below are the detailed implementation aspects:

1. Hardware

    - Development Environment: The development of the system can be carried out on a standard laptop or desktop computer with sufficient computational resources.
    - Training and Inference: For model training and inference, a machine with a dedicated GPU (e.g., NVIDIA GeForce GTX/RTX series or NVIDIA Tesla series) is preferable to accelerate computation and reduce training time.

2. Software

    - Operating System: The system can be implemented on various operating systems, including Windows, Linux, or macOS.
    - Programming Language: Python is the primary programming language used for implementing the system due to its extensive libraries and frameworks for machine learning and web development.
    - Libraries and Frameworks
        - TensorFlow and Keras: Used for building, training, and deploying deep learning models, including custom CNN architectures and pre-trained models.

- Streamlit: Employed for developing the web application interface, allowing users to interact with the deployed models.
- Lime: Integrated into the system for generating explanations for model predictions and enhancing interpretability.
- NumPy, Pandas, Matplotlib: Auxiliary libraries utilized for data manipulation, visualization, and analysis.

3. Development Environment

   - Integrated Development Environment (IDE): IDEs such as PyCharm, Visual Studio Code, or Jupyter Notebook can be used for writing, debugging, and testing the code.
   - Package Management: Python package management tools like pip or conda are used to install and manage dependencies required by the system.

4. Deployment

   - Cloud Platform: The trained models and web application can be deployed on cloud platforms such as Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure for scalability and accessibility.
   - Containerization: Docker containers can be utilized to package the application along with its dependencies, facilitating easy deployment and reproducibility across different environments.
   - Web Server: The Streamlit web application can be hosted on a web server such as Nginx or Apache for serving web traffic and handling user requests.

5. Data Management

   - Data Storage: The image dataset and trained model files can be stored on local storage or cloud storage services such as Amazon S3 or Google Cloud Storage.
   - Data Versioning: Version control systems like Git can be employed for managing changes to the codebase, data, and model files, ensuring reproducibility and collaboration among team members.

6. Continuous Integration/Continuous Deployment (CI/CD)

   - CI/CD Pipelines: Automated CI/CD pipelines can be set up using tools like Jenkins, Travis CI, or GitHub Actions to streamline the development, testing, and deployment processes, ensuring code quality and reliability.

5 Results

The results of the project demonstrate the effectiveness of the proposed methods and techniques in insect classification, explainability, and deployment. Key performance metrics, limitations, and potential areas for improvement are discussed below:

1. Model Performance

    - Accuracy: The accuracy of the trained models in classifying ants and bees is evaluated using separate test datasets. The custom CNN models and transfer learning-based models achieve high classification accuracies, typically exceeding 90%.
    - Loss: The training and validation loss curves indicate the convergence of the models during training, with minimal overfitting observed.
    - Confusion Matrix: Analysis of the confusion matrix provides insights into the distribution of true positives, true negatives, false positives, and false negatives, highlighting any misclassifications or areas for improvement.

2. Explainability

    - Lime Explanations: The Lime framework generates local explanations for individual predictions made by the models, highlighting the important features contributing to each classification decision. These explanations enhance model interpretability and build user trust.

3. Deployment and User Interaction

    - Streamlit Web Application: The deployed Streamlit web application provides an intuitive interface for users to upload images, obtain predictions, and explore explanations in real-time. User feedback and engagement with the application are monitored to assess usability and satisfaction.
    - Limitations:
    - Data Imbalance: Imbalance in the distribution of ant and bee images within the dataset may affect model performance, particularly in scenarios where one class is underrepresented.
    - Generalization: The trained models may exhibit limitations in generalizing to unseen data or handling variations in insect species, poses, or environmental conditions not represented in the training dataset.
    - Explainability Constraints: While Lime provides valuable insights into model predictions at the local level, its effectiveness may be limited in scenarios with complex or non-linear decision boundaries.

6 Future Directions

- Data Augmentation: Further exploration of data augmentation techniques and strategies for addressing data imbalance can enhance model robustness and generalization.
- Transfer Learning Variants: Experimentation with different pre-trained models, fine-tuning strategies, and architectures can potentially improve classification performance and adaptability to diverse insect species.
- Model Interpretability: Integration of additional explainability techniques, such as SHAP (SHapley Additive exPlanations), Grad-CAM (Gradient-weighted Class Activation Mapping), or attention mechanisms, may provide complementary insights into model predictions and decision-making processes.
- Real-world Testing: Conducting real-world field tests and validation studies to assess the system's performance in practical insect classification scenarios, such as agricultural pest management or ecological monitoring, can provide valuable feedback and validation of the proposed solution.
- Overall, while the results demonstrate promising performance in insect classification and explainability, addressing the identified limitations and pursuing future research directions can further enhance the efficacy and applicability of the proposed system.

7 Deployment

Deploying the insect classification system in a real-world setting involves several steps to ensure accessibility, scalability, and reliability. The deployment process includes the following key considerations:

1. Hosting Environment
    - Cloud Infrastructure: Utilize cloud platforms such as Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure for hosting the deployed system. Cloud infrastructure offers scalability, flexibility, and reliability, allowing the system to handle varying levels of traffic and workload demands.
    - Virtual Machines or Containers: Deploy the system on virtual machines (VMs) or containers (e.g., Docker) to isolate the application environment and ensure consistency across deployments.
2. Web Application Deployment
    - Streamlit Application Deployment: Deploy the Streamlit web application on the chosen hosting environment. Streamlit provides built-in deployment options, such as Streamlit Sharing or Heroku, which simplify the deployment process and streamline application hosting.

- Web Server Configuration: Configure a web server (e.g., Nginx or Apache) to serve the Streamlit application and handle incoming HTTP requests. Configure domain settings and SSL certificates for secure communication.

3. Model Serving

   - Model Hosting: Host the trained machine learning models on the cloud platform using services such as AWS S3, GCP Storage, or model hosting platforms like TensorFlow Serving or AWS SageMaker. Ensure the models are accessible and scalable for making predictions in real-time.
   - Endpoint Configuration: Configure endpoints for the deployed models to handle prediction requests from the Streamlit web application. Use RESTful APIs or gRPC for communication between the web application and the model serving infrastructure.

4. Scalability and Performance

   - Auto-scaling: Implement auto-scaling policies to automatically adjust the compute resources based on demand. Configure scaling triggers based on metrics such as CPU utilization, request latency, or incoming traffic.
   - Load Testing: Conduct load testing to evaluate the system's performance under different levels of concurrent user traffic. Identify and address potential bottlenecks or performance issues to ensure optimal system responsiveness.

5. Monitoring and Logging

   - Monitoring Tools: Utilize monitoring and logging tools such as AWS CloudWatch, GCP Stackdriver, or Prometheus to monitor system performance, resource utilization, and application health metrics.
   - Alerting: Configure alerting mechanisms to notify administrators of any critical issues, anomalies, or performance degradation in real-time.

6. Security and Compliance

   - Access Control: Implement robust access control mechanisms to restrict access to sensitive data and resources. Use IAM (Identity and Access Management) policies to manage user permissions and roles effectively.
   - Data Encryption: Encrypt data at rest and in transit using encryption standards such as TLS/SSL to ensure data confidentiality and integrity.

- Compliance: Ensure compliance with relevant data protection regulations (e.g., GDPR, HIPAA) and industry standards to protect user privacy and maintain regulatory compliance.

7. Continuous Integration/Continuous Deployment (CI/CD)

   - Automated Deployment Pipelines: Implement CI/CD pipelines using tools like Jenkins, GitLab CI/CD, or GitHub Actions to automate the deployment process. Streamline the integration, testing, and deployment of changes to the system codebase.

8. User Training and Support

   - Documentation and Training: Provide comprehensive documentation and training materials to users and administrators on how to use the deployed system effectively. Offer tutorials, user guides, and troubleshooting resources to facilitate user adoption and support. By following these deployment guidelines, the insect classification system can be successfully deployed in a real-world setting, enabling users to access and utilize the system for insect identification and analysis.

8 Conclusion

In conclusion, this project has demonstrated the successful development and deployment of a machine learning-based system for insect classification, leveraging deep learning techniques, explainable AI, and web application development. Through the implementation of custom CNN models and transfer learning with pre-trained architectures, the system achieves high accuracy in distinguishing between ants and bees, with additional insights provided by the Lime framework for model interpretability.

The deployment of the system in a real-world setting via a Streamlit web application enables users to upload images of insects and obtain rapid predictions, along with explanations for the model's decisions. By harnessing cloud infrastructure, auto-scaling capabilities, and continuous monitoring, the deployed system ensures reliability, scalability, and performance under varying levels of demand.

The implications of this work extend to various domains, including agriculture, ecology, and biodiversity conservation, where accurate insect classification is essential for pest management, species monitoring, and ecosystem analysis. The user-friendly interface and interpretability features enhance the usability and trustworthiness of the system, facilitating its adoption by researchers, practitioners, and stakeholders in relevant fields.

9 References

1 Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Ghemawat, S. (2016). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

2 Brownlee, J. (2020). Deep Learning for Computer Vision. Machine Learning Mastery. [Online]. Available: https://machinelearningmastery.com/deep-learning-for-computer-vision/.

3 Ribeiro, M. T., Singh, S., Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

4 Streamlit. (n.d.). Streamlit: The fastest way to build custom ML tools. [Online]. Available: https://www.streamlit.io/.

5 TensorFlow. (n.d.). TensorFlow Documentation. [Online]. Available: https://www.tensorflow.org/.