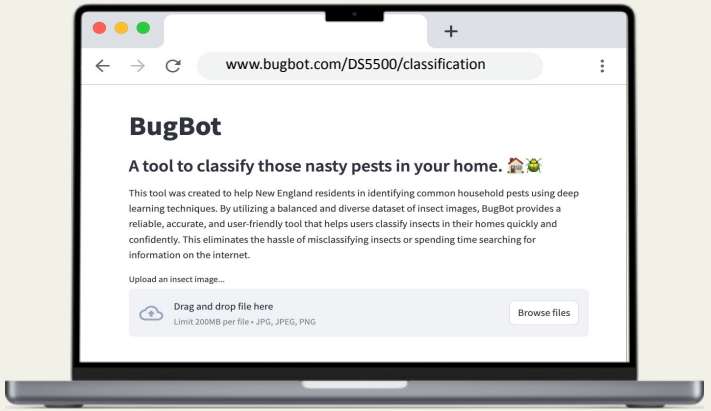
BugBot: Using Deep Learning For Household Pest Image Classification

Team Members: Shirley Fong, Keegan Veazey, Le Ju
Professor: Dr. Fatema Nafa
DS 5500 Capstone Project Presentation



Speaker: Le

Overview

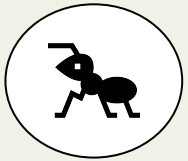
- Problem Specification
- Student & Team Learning Objectives
- Related Work
- Schedule & Timeline
- Demo
- Solution Design
- Data Collection
- Data Preprocessing
- Model Discussion
- Tuning & Model Architecture
- Tools List
- Results
- Postmortem
- Deliverables
- Team Contribution
- Q&A

Speaker: Le

Problem Specification

Problem Description	Intro to the Problem	Scope of Problem	Impact	Challenges & Constraints
Household pests are visually similar, making them hard to identify for non-experts. Misidentification can lead to ineffective treatment and health risks.	Currently no accessible, accurate, image-based tool for identifying common New England household pests using everyday devices like smartphones.	Classify 11 common New England household pests (e.g., rice weevil, subterranean termites) using a deep learning model trained on web-scraped images. Focused on home environments, not agriculture.	A simple architecture and accurate model helps users identify pests early, take informed actions, and potentially reduce treatment costs and health issues.	1. Small dataset 2. Pest visual similarity 3. Balancing performance & efficiency 4. Real-world variability in lighting & image quality





Speaker: Le

Student & Team Objectives

Purpose	Learning Outcomes	Application of Knowledge	Critical Thinking & Problem Solving	Collaboration
To design an accessible, image-based tool that helps users identify common New England household pests using deep learning, solving a practical problem through technology.	• Gained hands-on experience in web scraping, image preprocessing, model developing and tuning, and deploying ML models. • Evaluate models beyond accuracy (i.e., usability & generalizability).	Applied concepts from machine learning, computer vision, and software engineering to build a functional product from scratch, combining theory with practice.	• Tackled challenges like data leakage, and model overfitting. • Iteratively improved results through experimentation and informed decision making.	• Worked as a team to divide tasks effectively, balancing coding, testing, documentation, and design. • Constant feedback and communication were key to staying aligned.

Speaker: Le

Related Work

Context

- Existing pest classification models focus on agriculture, making them unsuitable for household pest identification.
- BugBot addresses this gap by using diverse, real-world images instead of controlled lab images.

Key Works Summarization

Islam & Rahman (2024): Developed a CNN-based jute pest classification model using transfer learning. While their focus is on agricultural pests, their methodology inspired BugBot's classification approach.

Turkoglu et al. (2021): Created PlantDiseaseNet, an ensemble model for plant disease and pest detection. Their dataset used highly controlled images, contrasting BugBot's use of varied, web-scraped images.

Rehman et al. (2019): Explored deep learning for brain tumor classification, highlighting the importance of image preprocessing (flipping, rotation), techniques that BugBot also employs.

Thenmozhi & Reddy (2019): Achieved 95% accuracy in crop pest classification by comparing custom CNN models with pre-trained architectures. Their augmentation methods influenced BugBot's approach.

Rajan et al. (2024): Used SVM for pest classification but demonstrated its limitations, especially with manually designed feature extraction and reliance on controlled images.

Speaker: Shirley

Schedule

Weeks 1-3

- Capstone proposal
- Data collection
- Data pipeline preprocessing

Weeks 4-5

- Data pipeline preprocessing
- Build & train models

Weeks 6-7

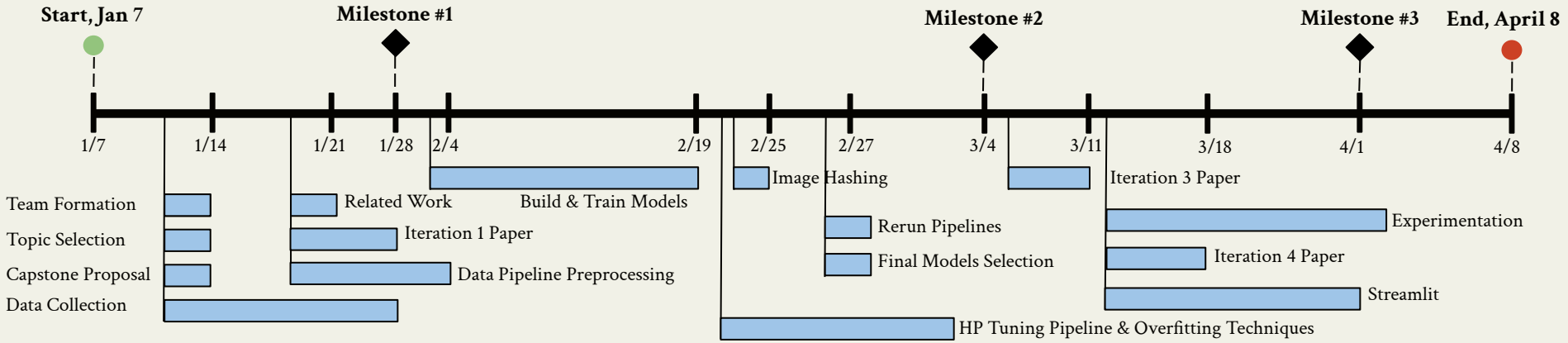
- Hyperparameter tuning
- Overfitting techniques
- Image hashing
- Rerun pipelines
- Final model selection

Weeks 8-9

- Final experimentation
- Organizing and cleaning up scripts
- Setup and deploy Streamlit

Speaker: Shirley

Timeline



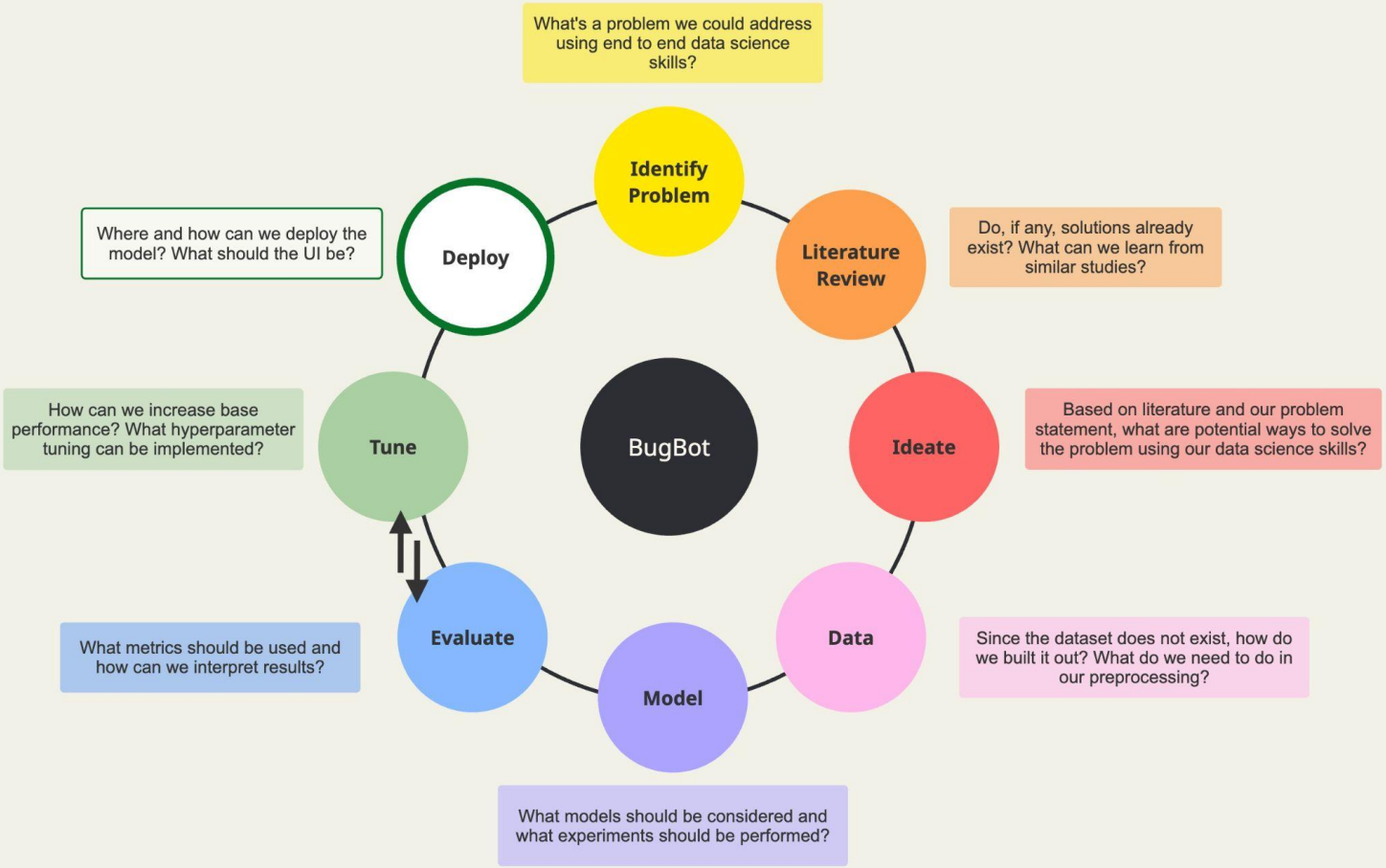
Speaker: Shirley

Demo

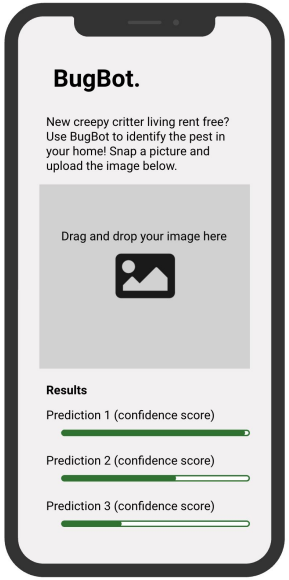
🔗 Link to video

Speaker: Keegan

Solution Design: Process

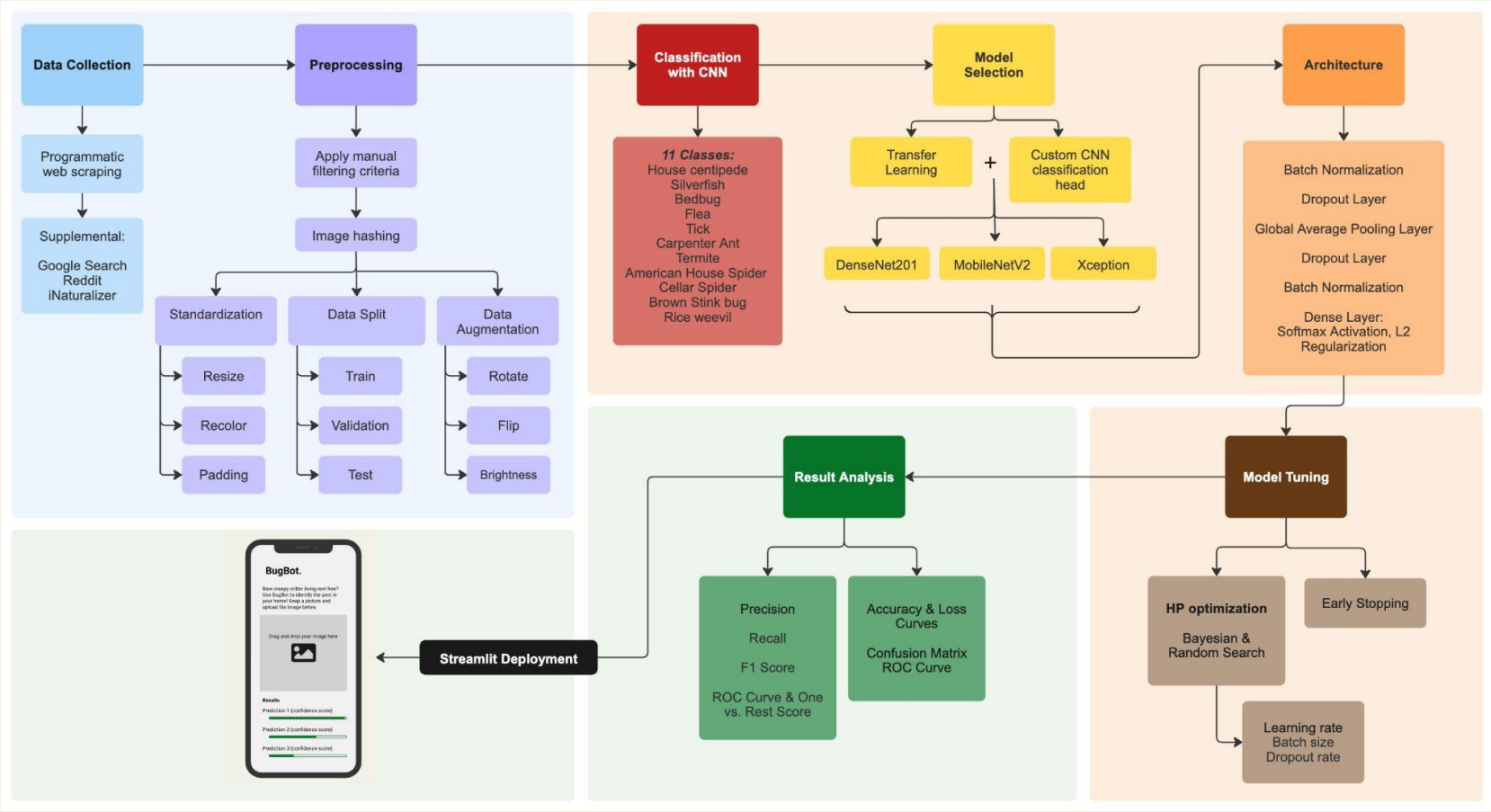


Mock Up



Speaker:
Keegan

Solution Design: Technical



Speaker: Shirley

Data Collection

1 Sources

- Majority of data sourced using a Bing Image Search API tool, supplemented with manual Google search results
- Focus on acquiring images relevant to home environments (e.g., infestations in homes)

2 Filtering Criteria

1. The insect must be present in the image
2. The image must show at least 75% of the insect's body
3. The photo is not a drawing, cartoon, or an AI generated image
4. The image depicts the adult/matured insect

3 Distribution

- Original dataset has 1,760 images distributed across 11 insect classes (160 images per class)
- After data augmentation, training set increased to 6,612 images, while the testing set contains a total of 220 images, and the validation set at 440 images

Speaker: Shirley

Data Description

	Size	Number of Images
Original Pre-Augmentation Dataset	700 MB	~1760
Train Dataset	53.7 MB	~6612
Validation Dataset	3 MB	440
Test Dataset	1.6 MB	220
Final dataset	79 MB	8742

Data Source: link

Speaker: Shirley

Data Preprocessing Steps

Image Hashing: Remove duplicate images

Standardization:

- Pad rectangular images
- Color standardization (RGBA → RGB)
- Resize to fixed dimension (224x224)

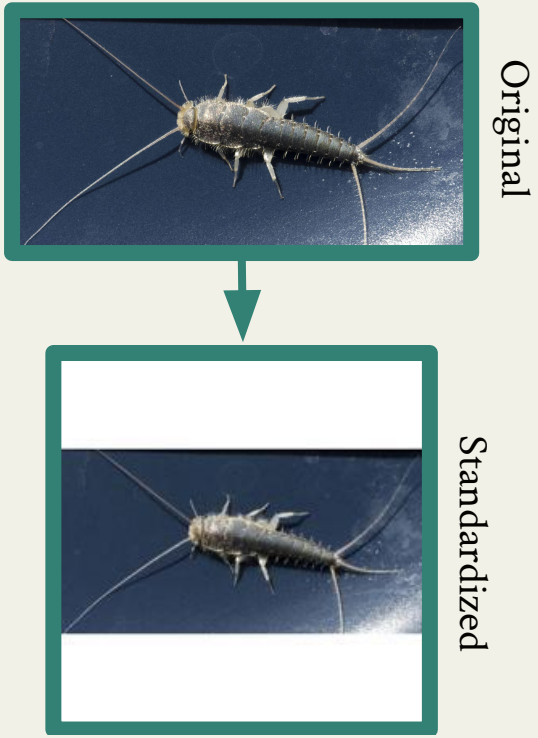
Data split: Test, Validation, Training sets

Data Augmentation: Rotate, flip horizontal + vertical, brightness

Normalization:

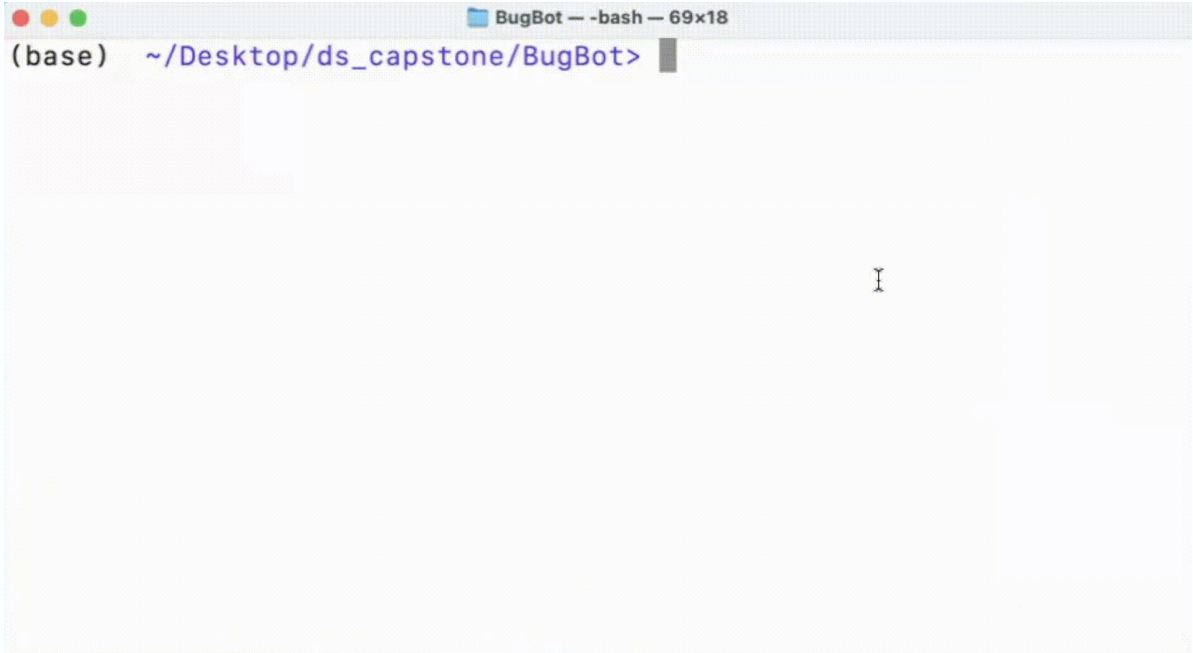
- Pixel values normalized to [-1, 1] & [0, 1]

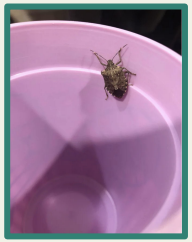
Libraries: *Undouble, PIL Image, ImageEnhance, Numpy, Random, etc.*



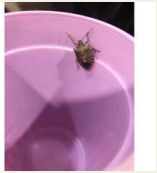
Speaker: Shirley

Image Preprocessing





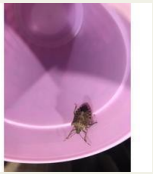
Original



Standard

Bright

Flip 180



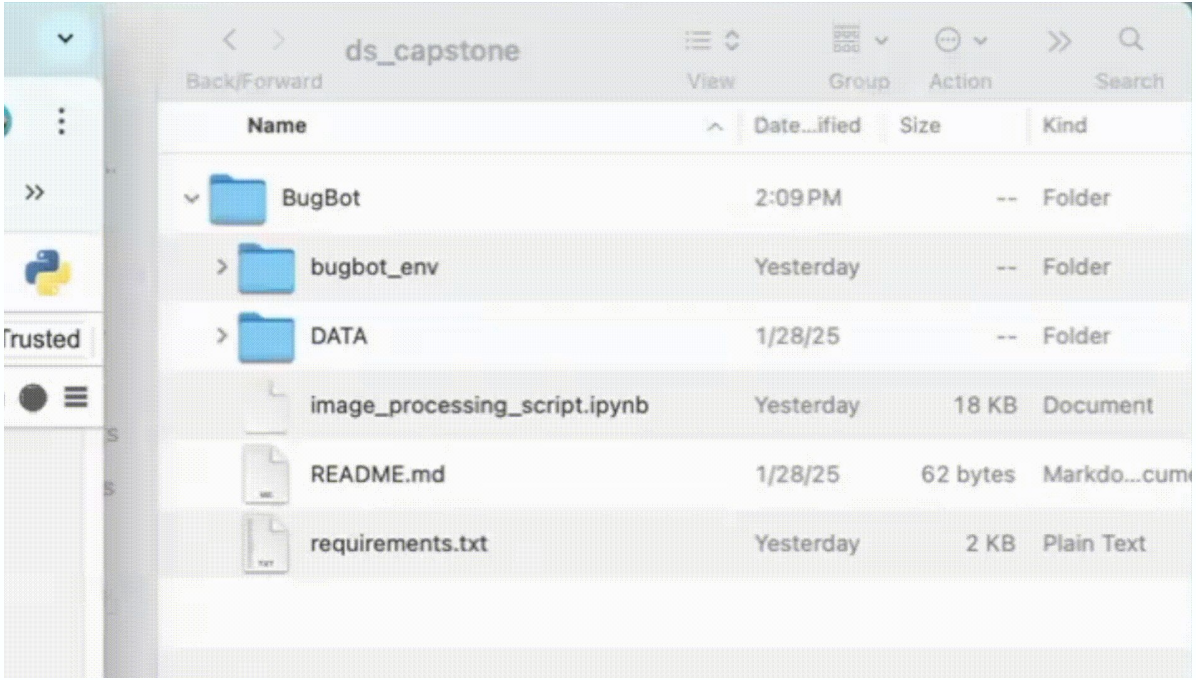
H Flip

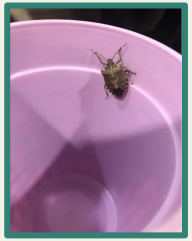
V Flip

Rotate 90

Speaker: Shirley

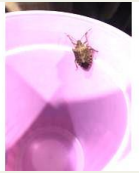
Image Preprocessing





Original

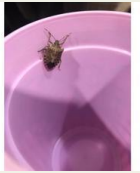
Standard



Bright

Flip 180

H Flip



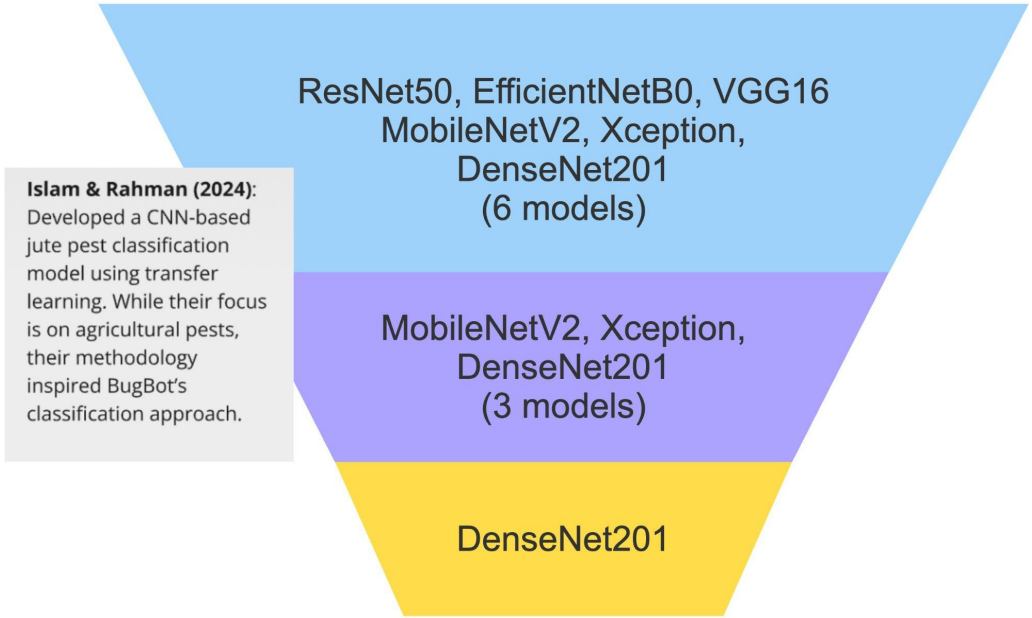
V Flip

Rotate 90

Speaker: Keegan

Model Selection

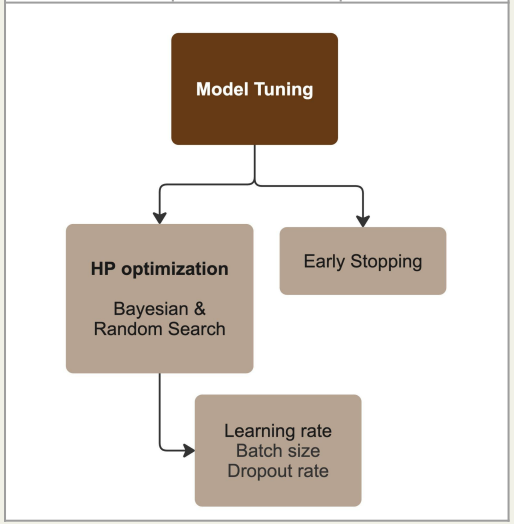
- Use consolidated set of models across literature review
- Narrow to top 3 performing models based on accuracy
- Complete model tuning for each of the 3 models and build the best model for each
- Use multiple metrics and visuals to identify best of the best



Speaker: Keegan

Model Tuning

LR	Dropout Rates	Batch Sizes
[0.01, 0.001, 0.0001]	[0.2, 0.3, 0.4, 0.5]	[16, 32, 64]

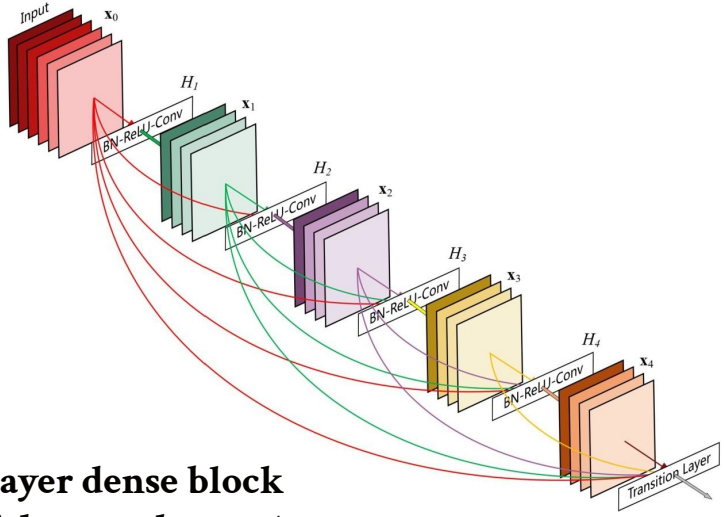


HP Tuning Algorithm	**Grid Search**	**Bayesian Optimization**	**Random Search**
Intuitive Definition	Brute force method that finds the best parameter pairs by trying every possible parameter combination	Uses Bayesian statistics to find best pairs based on history of previously evaluated pairs using validation loss	Randomly samples a subset of parameter combinations and evaluates pairs in real time using validation loss to make "steps in the right direction"
Used in BugBot	*No* Too slow due due to exhaustive search and constrained resources	*Yes* Runs on local compute, proved to be fastest option	*Yes* Able to run on local compute, but took longer than BO. Used in tandem with BO as check & as resource for further hp experimentation

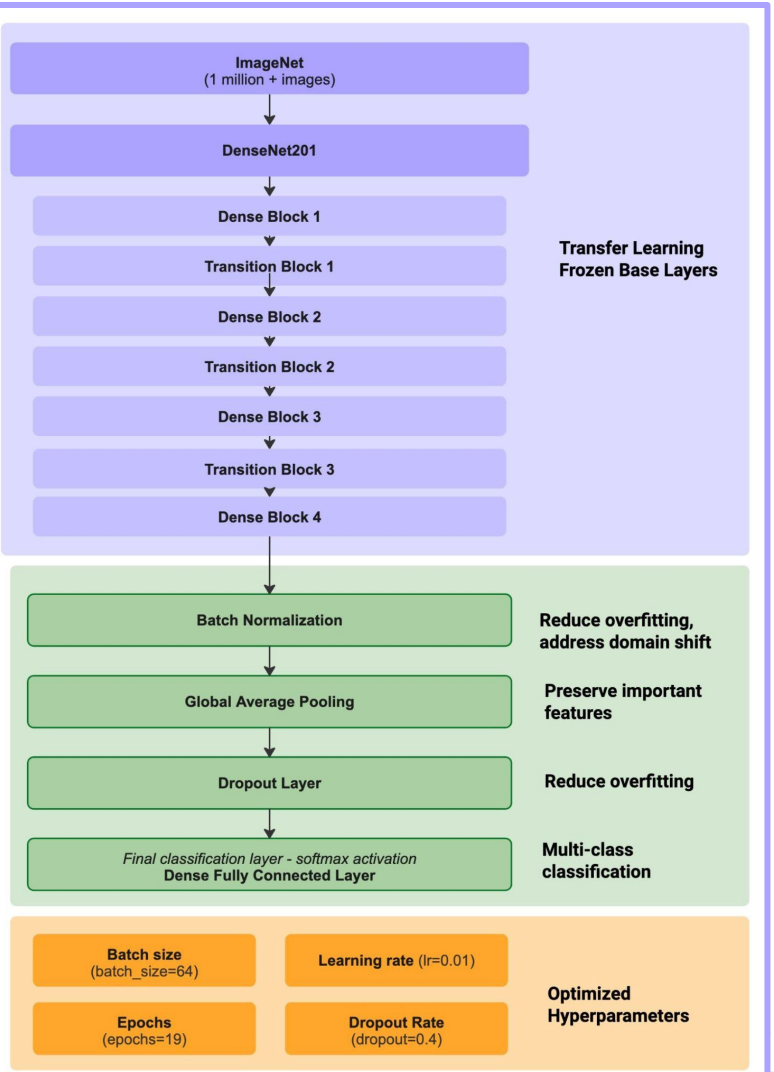
Speaker: Keegan

Final Model Architecture

- Use categorical cross entropy as objective function
- Reference literature review neural network architectures
- Use best hyperparameters found from model tuning



5 Layer dense block
With growth rate 4



Speaker: Keegan

Tools List

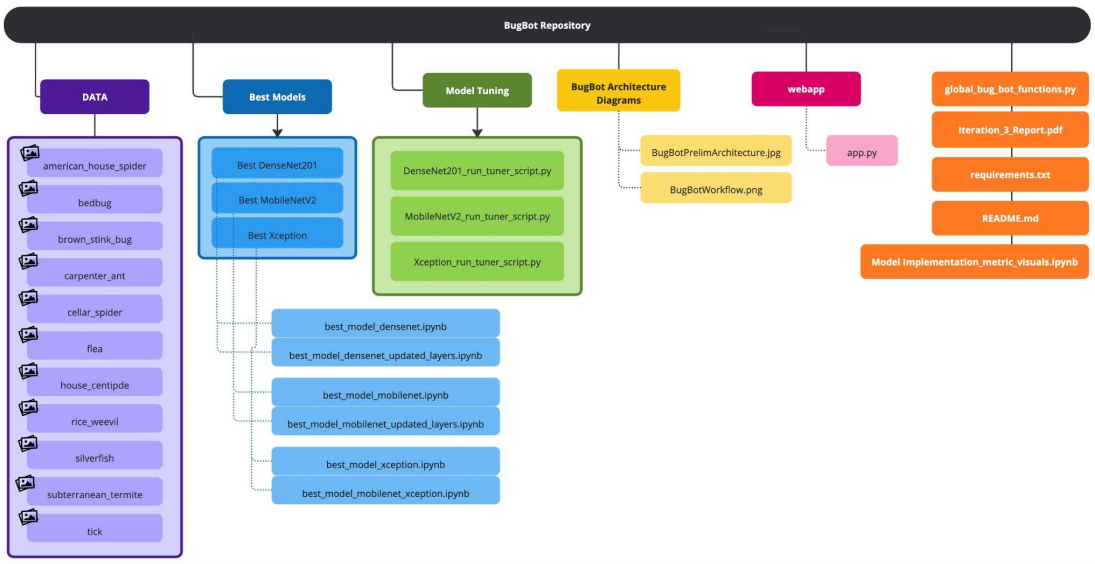
Core Tools

- ★ Python
- ★ Jupyter Notebook
- ★ Miro
- ★ Keras
- ★ Numpy
- ★ Undouble
- ★ Scikit-Learn
- ★ Streamlit
- ★ Bing Image API
- ★ Matplotlib

Data Structures

- ★ Dictionaries
- ★ Numpy Arrays
- ★ Dataframes
- ★ CSV

GitHub



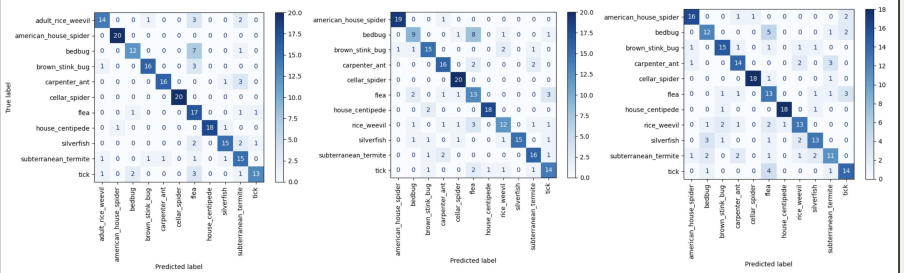
Speaker: Shirley

Results & Evaluation

Summary Chart of Model Performance

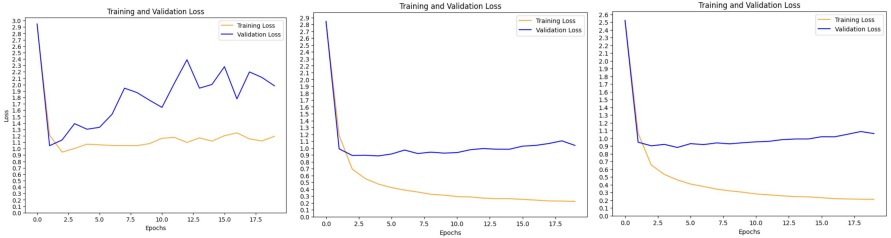
Model Name	Precision (%)	Recall (%)	F1 Score (%)	Accuracy (%)	OVR Score (%)
Xception	72	71	72	71	95
MobileNetV2	78	76	76	76	96
DenseNet201	83	80	81	80	97

Confusion Matrix



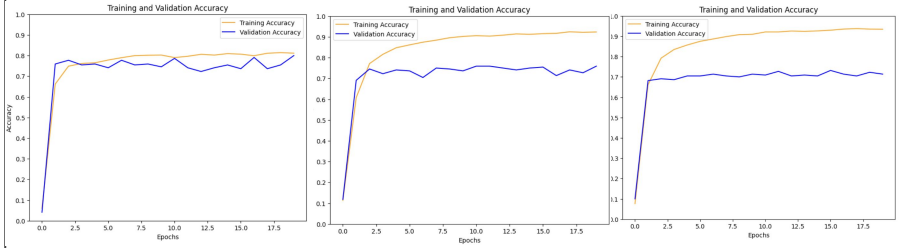
(a) DenseNet201 (b) MobileNetV2 (c) Xception

Loss Curve



(a) DenseNet201 (b) MobileNetV2 (c) Xception

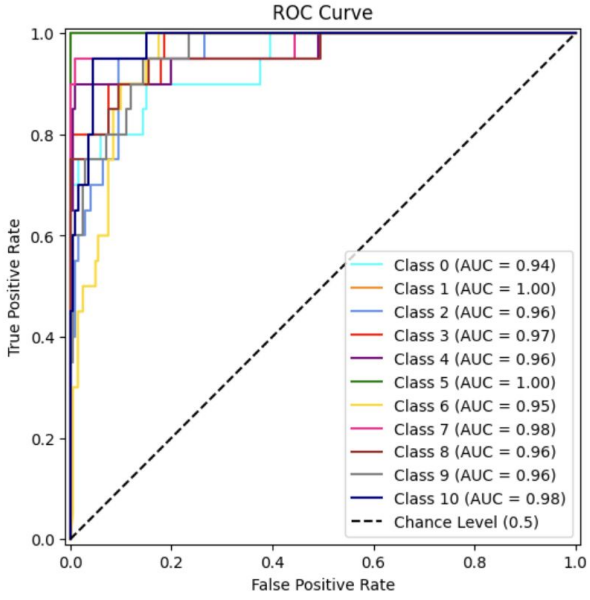
Accuracy Curve



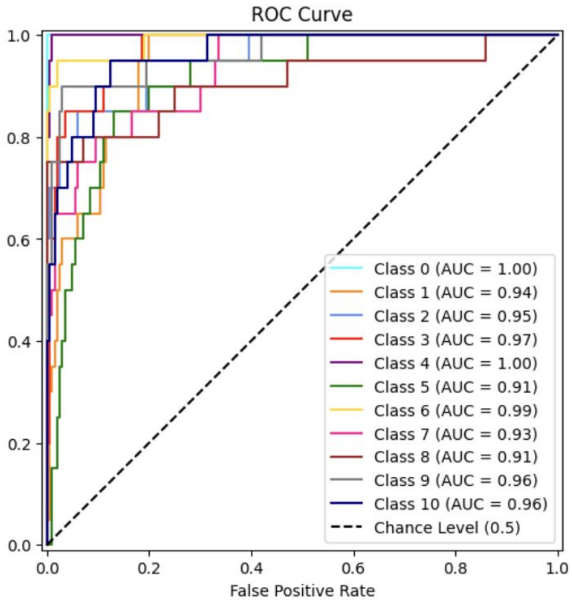
(a) DenseNet201 (b) MobileNetV2 (c) Xception

Speaker: Shirley

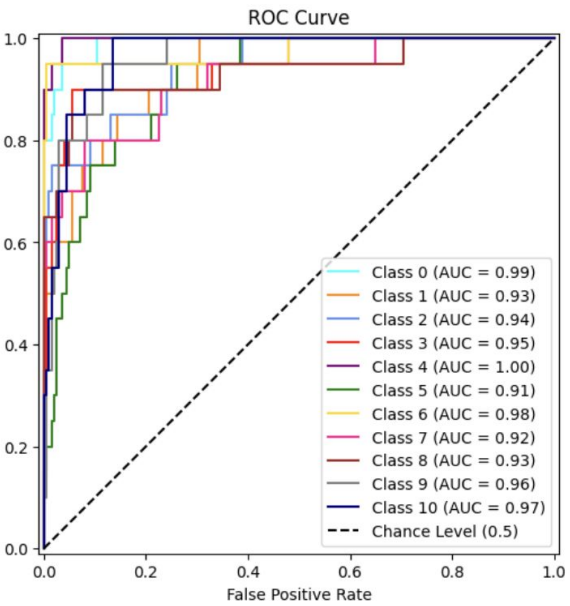
Results & Evaluation



(a) DenseNet201



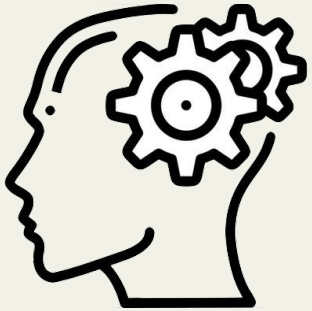
(b) MobileNetV2



(c) Xception

Speaker: Le

Postmortem



- This project helped us learn more about image classification
- Helped us understand how computers process images
- Made us realize how much (diverse) data we need to train the model
- This project helped us learn to value cross checking and referencing multiple resources before implementing new libraries right away
- Taught us to take initiative and problem-solve more confidently, while also improving our teamwork and communication

Speaker: Le

Deliverables

Capstone Proposal & Presentation	PDF & PPT
GitHub Repository	Link
Sample Outputs	Link
BugBot Code Guide	Link
Discussion of Different Approaches	PDF
Streamlit	Link
Google Drive Folder	Link

Team Member Contributions

Shirley Fong
Role: Data Scientist

Skills & Expertise:
- Data engineering
- Project management
- UI design

Data Collection

Ticks

Flea

Rice weevil

Carpenter ant

Data Processing

Data Collection
- Created script to extract image & links, saves links to CSV file

Data Image Preprocessing
- Added fixed image size & made normalization function
- Created image hashing script to remove duplicate images
- Created file sorter script to organize images into respective folders for model implementation

GitHub
- Uploaded images on GitHub using Git LFS

Model & Application

Model Experimentation
- Implemented evaluation metrics (ROC curve, one-vs-rest score, confusion matrix, classification report, accuracy, precision, recall, F1 score)
- Ran 7 models on validation set to select final 3 models (CNN w/o transfer learning, EfficientNet, ResNet50, Vgboost, MobileNetV2, Xception, DenseNet201)
- Ran DenseNet201 and replicated jute pest paper classification portion layers

Application
- Deployed Streamlit application

Team Member Contributions

Keegan Veazey
Role: Data Scientist

Work alignment

Interest in learning about transfer learning + model deployment

Data Collection

Cellar Spider

Marmorated Stink bug

American House Spider

Data Processing

Data Scaling
- Added code to scale via padding to fix rectangular images being warped

Modularization & Automated Pipeline
- Modularized case study data preprocessing notebooks and consolidated into single Python script for fully automated data preprocessing pipeline
- Built custom randomized (reproducible) train, test, validation split unique to BugBot data infrastructure

GitHub
- Established repo, virtual environment, requirements.txt, and instructions for users to run the pipeline on their own
- Consistent github contributions code

Model Training & Tuning

Hyperparameter tuning
Coded scripts for transfer learning hyper parameter tuning and training
Hyperparameters
- LR, batch size, dropout rate

Optimization algorithms implemented
- Bayesian optimization, Random Search

Model Training
Experimented with overfitting mitigation techniques
- Early stopping: pipeline implements argeparse to run script experiments with varied patience & min_delta values
- Experimented with Dropout & Batch Normalization layers, L2 regularization to address bias variance tradeoff

Final Model
Coded notebook for final model build template & ran on test set for Shirely's final evaluation
- Extend dataset by combining validation with training data post-tuning for final model

Team Member Contributions

Le Ju
Role: Data Scientist

Work alignment

Interest in completing an end-to-end data science project

Data Collection

Bed Bug

Household Centipede

Silverfish

Data Processing

Data Augmentation & Preparation

- Applied rotation, height/width shift, and brightness adjustments to improve model robustness
- Manually labeled all images in the validation and test sets, organizing them into pest classes to streamline the training and evaluation process

GitHub

- Pushed code, results, and project files for version control and collaboration

Model Training & Tuning

Model Implementation

- Built and trained (1) an initial CNN model and (2) a CNN model with transfer learning using ResNet, EfficientNet, and VGG16.

Model Training & Hyperparameter Tuning

- Utilized GPU resources and installed CUDA and TensorFlow-GPU (10x faster) to optimize training and hyperparameter tuning.
- Trained DenseNet201 with extensive hyperparameter tuning to determine the best parameters for the final model and generate evaluation metrics.

References

Guo, B., Wang, J., Guo, M., Chen, M., Chen, Y., & Miao, Y. (2024). Overview of pest detection and recognition algorithms. Electronics, 13, 3008–3008. https://doi.org/10.3390/electronics13153008

Islam, M. T., & Rahman, M. S. (2024). An efficient deep learning approach for jute pest classification using transfer learning, 1473–1478. https://doi.org/10.1109/iceeict62016.2024.10534395

Rehman, A., Naz, S., Razzak, M. I., Akram, F., & Imran, M. (2019). A deep learning-based framework for automatic brain tumors classification using transfer learning. Circuits, Systems, and Signal Processing, 39, 757–775. https://doi.org/10.1007/s00034-019-01246-3

Thenmozhi, K., & Srinivasulu Reddy, U. (2019). Crop pest classification based on deep convolutional neural network and transfer learning. Computers and Electronics in Agriculture, 164, 104906. https://doi.org/10.1016/j.compag.2019.104906

Turkoglu, M., Yanikoˇglu, B., & Hanbay, D. (2021). Plantdiseasenet: Convolutional neural network ensemble for plant disease and pest detection. Signal, Image and Video Processing, 16. https : //doi.org/10.1007/s11760-021-01909-2