

**MANIPAL SCHOOL OF INFORMATION SCIENCES**

**(A Constituent unit of MAHE, Manipal)**

**Streamlining Object Detection using TensorFlow for Plant Disease Classification.**

|  |  |  |
| --- | --- | --- |
| **Reg. Number** | **Name** | **Branch** |
| **241058017** | **Sneha V Shet** | **Big Data Analytics** |
| **241058028** | **Anushree S G** | **Big Data Analytics** |
| **241058038** | **Sufiya Taranum** | **Big Data Analytics** |

**Under the guidance of**

**Dr Prathviraj N**

Associate Professor,

Manipal School of Information Sciences,

MAHE, MANIPAL

**12/11/2024**

****

Table of Contents

[**1.** **Introduction** 1](#_Toc118908357)

[**2.** **Objectives** 2](#_Toc118908359)

[**4.** **Specifications** 3](#_Toc118908361)

[**5.** **Architecture/Design/Block or Schematic diagrams / Flowchart Proposed modules** 4](#_Toc118908362)

[**6.** **Work done** 5](#_Toc118908363)

[**7.** **Results** 12](#_Toc118908364)

[**8.** **Conclusions** 13](#_Toc118908365)

[**9.** **Scope for further work** 13**.**](#_Toc118908366)

[**10.** **References** 13.](#_Toc118908367)

**LIST OF FIGURES**

[Figure 1: Dataset 2**.**](#_Toc54001391)

Figure 2: Overview of project flow……………………………………………..…………… 3

# **Introduction**

Farmers face economic losses every year because of various diseases that can happen to fruit/vegetable growing plants.

If farmer can detect these diseases early and apply appropriate treatment then it can save lot of waste and prevent the economic loss.

It is important that to accurately identify what kind of disease is there in that a particular plant. For example – Potato plant has 2 type of disease late blight and early blight. The treatment for both the diseases is different and if farmer could identify early that would prevent him from economic loss.

Conventional machine learning (ML)- based algorithms face challenges, especially in identifying the plants in real-world data due to a lack of features. Deep Learning (DL) approaches use self-learning to extract all potential features that assist in classifying plant diseases accurately.

This paper presents a hybrid Convolutional Neural Network (CNN) model of three state-of-the-art CNNs to classify pre-processing. The proposed model utilizes convolutional neural networks to extract features and classify images. The framework of the proposed method comprises three paramount stages to accomplish the classification key idea, including the data preparation phase, pre-processing phase, and classification phase.

The dataset consists of 39,152 samples with 21 classes, out of while the data is been sent as batch of 32 and the train and test split as 80% images for training and 20% for testing, out of test data 10% for validation and 10% for test. Each dataset is individually tested in the proposed model to evaluate the classification accuracy using a set of standard evaluation metrics including accuracy and F1-score. The total averages of the proposed model on both datasets are 97%, 94% on the accuracy F1-score.





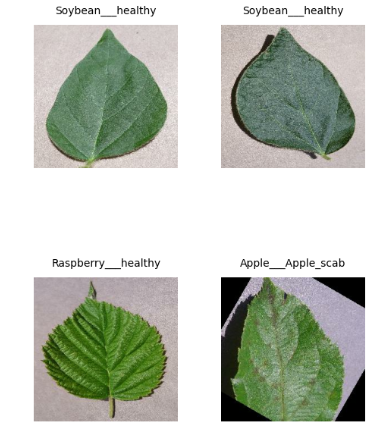


Fig 1: Dataset

# **Objective**

Detect the plant classification effectively, to Enhance classification accuracy, enhancing the capability of feature extraction and representation.

Enhance efficiency and automation through streamlined workflows and scalable resources.

**Improved Feature Extraction**: To overcome the limitations of conventional machine learning models by leveraging a Convolutional Neural Network (CNN) model that automatically extracts relevant features from plant images for precise disease classification.

**Accurate Disease Identification**: To accurately classify different types of diseases affecting plants, such as distinguishing between early blight and late blight in potato plants, enabling targeted treatment.

# **Specifications**

**Software Requirements**

**OS**: Windows

**Python**: Version 3.7+

**Libraries**: TensorFlow 2.x, Keras, OpenCV, NumPy, Matplotlib, TensorFlow Lite

**Cloud Platforms**: GCP, AWS

**Functional Requirements**

1.Model Selection and customization

2.Input Data Handling

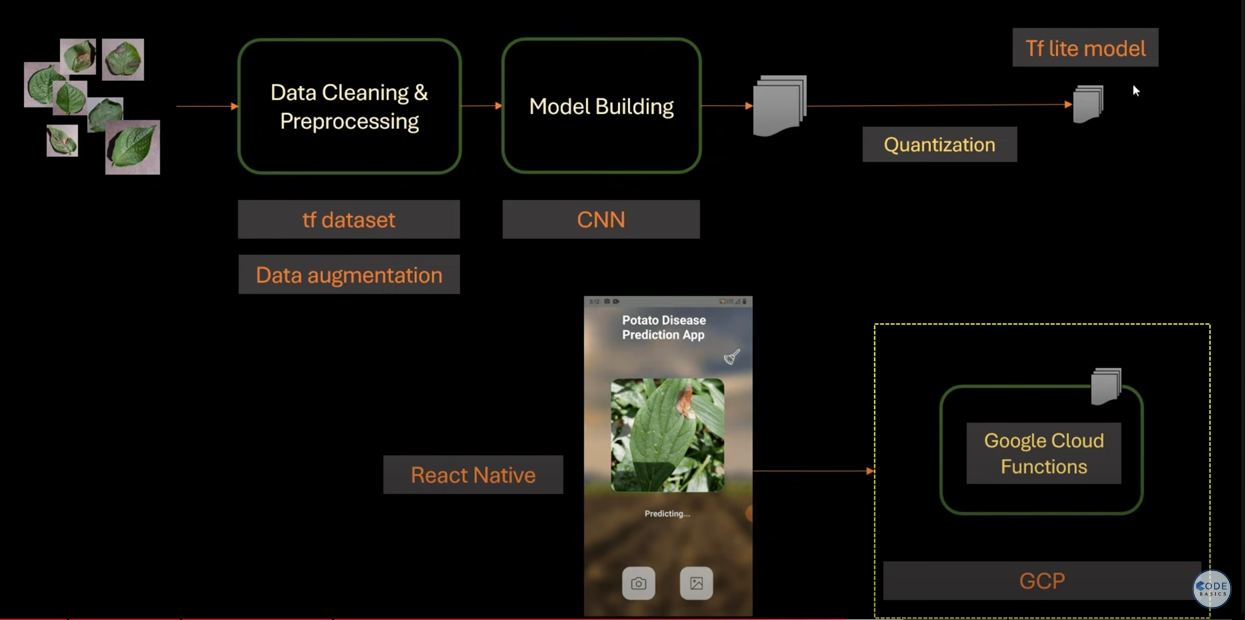
3.Training Framework

4.Model Evaluation and Interface optimization

5.Deployment

6.User Interface

# **Architecture**



Design

Fig 2: Overview of project flow

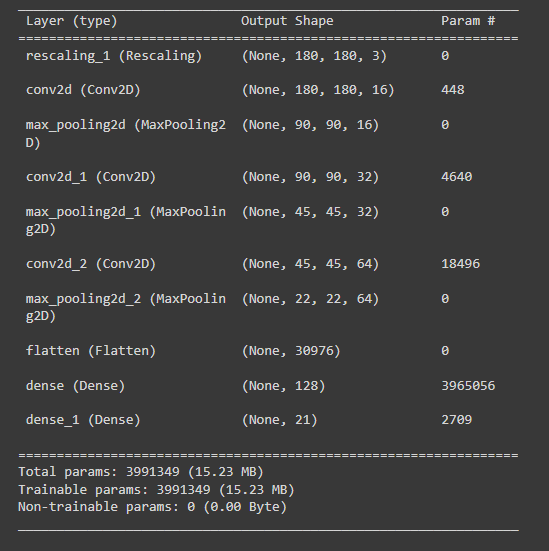
|  |
| --- |
|  |

The image illustrates a three-stage process for image classification using TensorFlow, specifically for tasks like plant disease detection. Here's a summary of each stage:

1. **Stage 1 - Data Preparation**:
   * **Data Collection**: Gathering labelled images for training.
   * **Data Splitting**: Dividing data into training, validation, and test sets.
2. **Stage 2 - Image Pre-processing**:
   * **Image Resizing**: Adjusting image dimensions to fit the input requirements of the model.
   * **Data Augmentation**: Enhancing data by applying transformations like rotation, flipping, or zooming to increase dataset diversity and prevent overfitting.
3. **Stage 3 - Feature Extraction and Image Classification**:
   * The pre-processed image is passed through a **CNN** (Convolutional Neural Network) which extracts features.
   * A fully connected layer then classifies the image into a specific class (e.g., healthy or diseased).
   * The output is a **predicted class** for the input image.

Each stage outputs labelled, resized, or classified images, contributing to the overall classification pipeline

# **Work Done**



This image shows the architecture of a Convolutional Neural Network (CNN) model with details about each layer, its output shape, and the number of parameters:

1. **Rescaling Layer**:
   * Rescales input images to a standard format of shape (180, 180, 3).
   * No trainable parameters.
2. **Convolution and Pooling Layers**:
   * **Conv2D Layer 1**: 16 filters, output shape (180, 180, 16), 448 parameters.
   * **MaxPooling2D Layer 1**: Reduces dimensions to (90, 90, 16), no parameters.
   * **Conv2D Layer 2**: 32 filters, output shape (90, 90, 32), 4,640 parameters.
   * **MaxPooling2D Layer 2**: Reduces dimensions to (45, 45, 32), no parameters.
   * **Conv2D Layer 3**: 64 filters, output shape (45, 45, 64), 18,496 parameters.
   * **MaxPooling2D Layer 3**: Reduces dimensions to (22, 22, 64), no parameters.
3. **Flatten Layer**:
   * Flattens the output to a single vector of 30,976 units for the dense layers.
4. **Dense Layers**:
   * **Dense Layer 1**: 128 units, fully connected layer with 3,965,056 parameters.
   * **Output Dense Layer**: 21 units for classification, with 2,709 parameters.
5. **Model Summary**:
   * Total parameters: 3,991,349 (all trainable).
   * Model size: Approximately 15.23 MB,this architecture is designed for image classification with an output of 21 classes.

|  |
| --- |
| 1. CNN model 2. CNN model after Hyperparameter tuning     Classification Report – 92% training accuracy Classification Report – 93% training accuracy      Testing Accuracy – 80% Testing Accuracy – 94% |
| This comparison highlights the performance of a Convolutional Neural Network (CNN) model before and after hyperparameter tuning:   1. **Initial CNN Model**:    * **Training Accuracy**: 92%    * **Testing Accuracy**: 80%    * Indicates that the model was relatively well-trained but showed a significant gap between training and testing accuracy, suggesting potential overfitting. 2. **CNN Model After Hyperparameter Tuning**:   Hyperparameter tuning means increasing batch size, increasing no of epochs to train the model for better accuracy.   * + **Training Accuracy**: Improved slightly to 93%.   + **Testing Accuracy**: Increased significantly to 94%.   + The higher testing accuracy, close to the training accuracy, indicates that hyperparameter tuning effectively reduced overfitting and improved generalization.   **Summary**: Hyperparameter tuning improved the model's ability to generalize, with testing accuracy increasing from 80% to 94%, bringing it closer to the training accuracy and overall enhancing model performance. |

|  |
| --- |
| 1. EfficientNet Model 4. Resnet50 Model         Training and validation Accuracy and losses. Training and validation Accuracy and losses.  Classification Report – 93% Training Accuracy Classification Report – 99% Training Accuracy    Testing Accuracy - 92% Testing Accuracy – 5% |

This comparison outlines the performance of two models, EfficientNet and ResNet50, based on their training and validation metrics:

1. **EfficientNet Model**:

EfficientNet is a family of CNN architectures introduced by Google in 2019. Its main innovation lies in using a compound scaling method to optimize the network’s accuracy and efficiency. This scaling method allows EfficientNet to scale model dimensions (depth, width, and resolution) in a balanced way, making it highly efficient and powerful.

* + **Training Accuracy**: 93%
  + **Testing Accuracy**: 92%
  + Shows consistent performance with similar accuracy on training and testing, indicating good generalization and no major overfitting or underfitting.

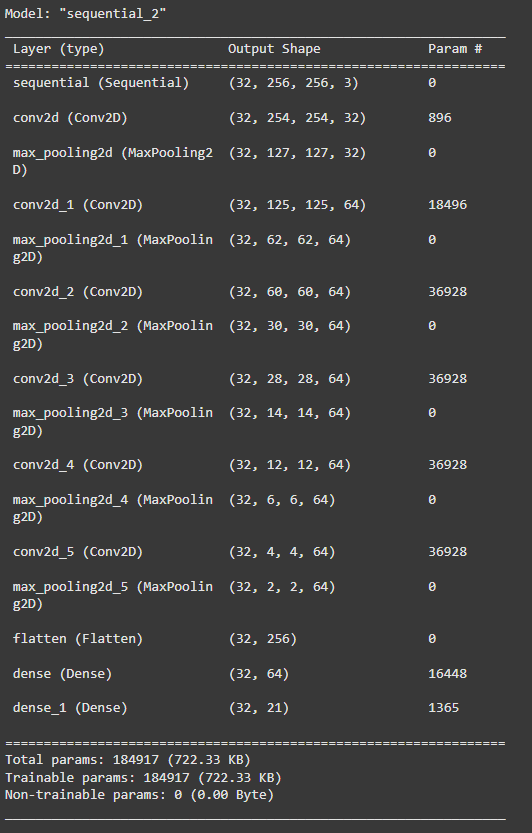
1. **ResNet50 Model**:

ResNet, short for Residual Network, is a deep CNN architecture introduced by Microsoft Research in 2015. It was ground-breaking because it solved the problem of "vanishing gradients," which previously limited the depth of neural networks. ResNet introduced the concept of **residual connections**, allowing it to create very deep networks that learn more complex features.

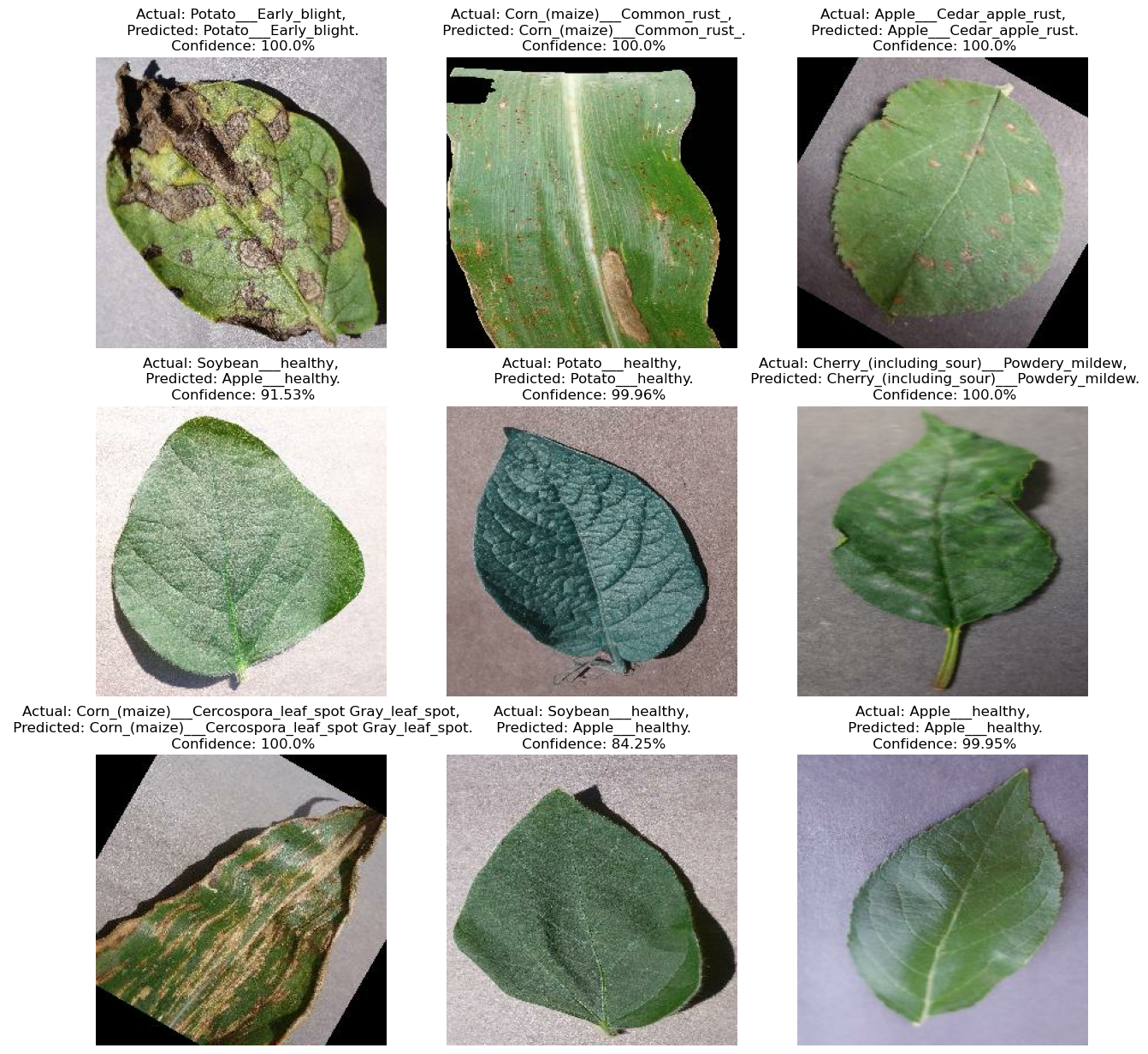
* + **Training Accuracy**: 99%
  + **Testing Accuracy**: 5%
  + The model achieves high training accuracy but extremely low testing accuracy, indicating severe overfitting where the model memorized training data but failed to generalize to new, unseen data.

**Summary**: The EfficientNet model demonstrates balanced and reliable performance across training and testing datasets, making it a better choice for this task. In contrast, the ResNet50 model, despite high training accuracy, performs poorly on testing data, suggesting it needs regularization or tuning to address overfitting.

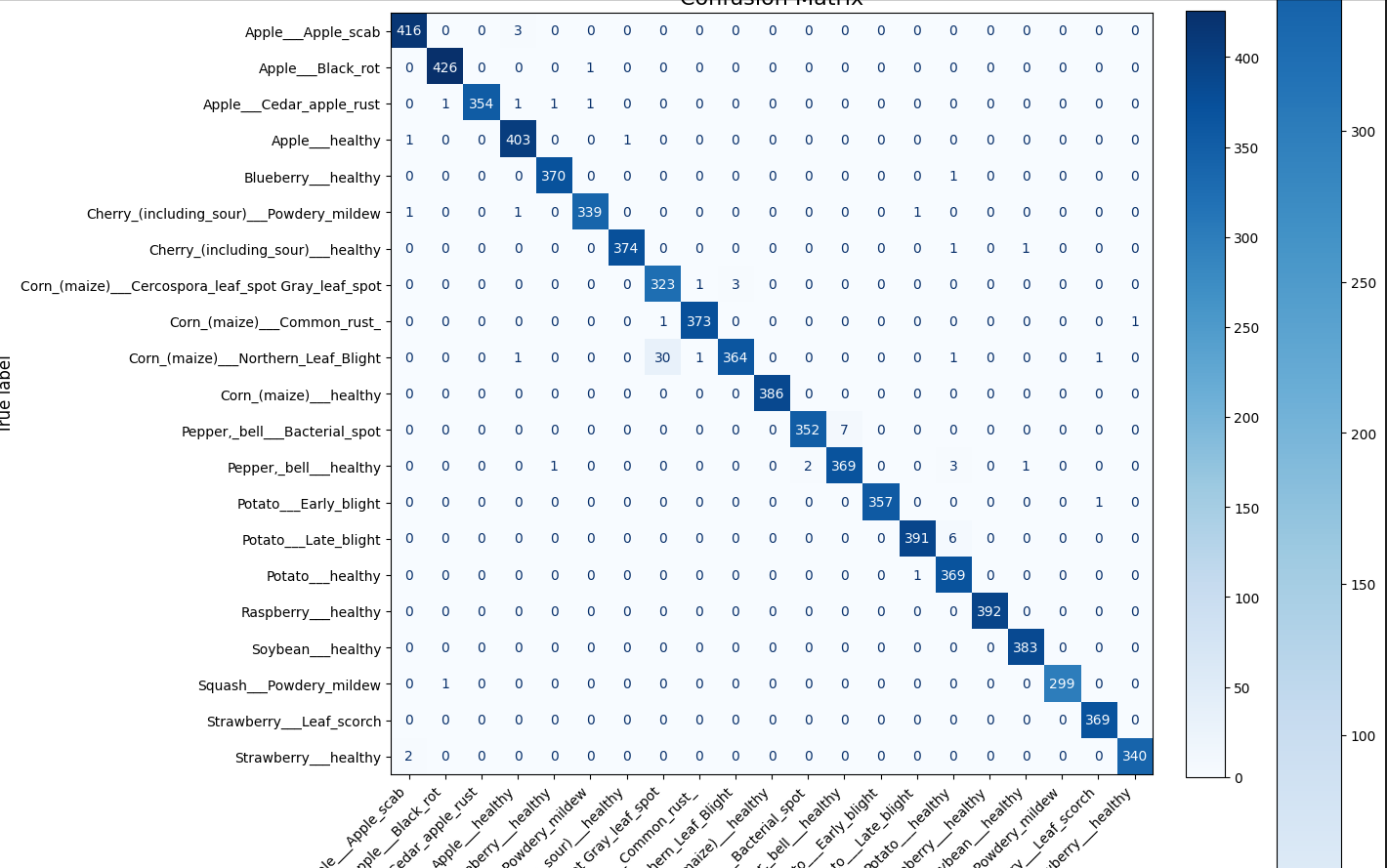
5.CNN with Extra Layers



Training Accuracy – 97%

Testing Accuracy – 94%

Confusion matrix



# **Result**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Training Accuracy (%)** | **Validation Accuracy (%)** | **Training Loss (%)** | **Validation Loss (%)** |
| CNN (2) | 92 | 80 | 0.8 | 0.2 |
| **CNN with extra layers (5)** | **97** | **94** | **0.088** | **0.19** |
| ResNet50 | 99 | 82 | 0.008 | 0.18 |
| EfficientNet | 93 | 92 | 0.7 | 0.8 |

# **Conclusion**

This work introduces pivotal knowledge to the computer vision community. Firstly, it improves the classification methods for plant disease detection in real-world conditions by using a Deep Learning technique. Regarding the agricultural community, this research can be implemented in a treatment planning system. It assists the farmer in alleviating labour-intensive costs, reducing time-consuming tasks, preventing herbicide pollution in the environment, and controlling diseases in plants.

# **Future Scope**

-Deploy in cloud platforms for smooth operations.

-Creating UI where farmer can upload image directly and get to know about the disease and treatment plan.

# **References**

[1] C. Huyen, [Introduction to Machine Learning Interviews](https://huyenchip.com/ml-interviews-book/)(2021), Self-published

[2] V. Tuulos, [Effective Data Science Infrastructure](https://www.manning.com/books/effective-data-science-infrastructure) (2022), Manning Publications Co.

<https://towardsdatascience.com/streamlining-object-detection-with-metaflow-aws-and-weights-biases-b44a14cb2e11>

<https://github.com/EdIzaguirre/plant-object-detection>

<https://www.tensorflow.org/tutorials/images/classification>