"AUTOMATED CLASSIFICATION OF MANGO AND BANANA RIPENESS STAGES USING EFFICIENTNETB0"

By

Gazi Mehraj Bin Ahmed
ID: 0222210005101020

Md. Shakib
ID: 0222210005101034

Rafiu Anjir
ID: 2104010202271

Batch: 41st ; Section: A

CSE 452: Neural Network & Fuzzy Logic Laboratory

Instructor:

MD Tamim Hossain
Lecturer
Department of Computer Science and Engineering
Premier University

_____
Signature

Department of Computer Science and Engineering
Premier University
Chattogram-4000, Bangladesh

23th November 2025

Automated Classification of Mango and Banana Ripeness Stages using EfficientNetB0

[Gazi Mehraj Bin Ahmed] [0222210005101020][1*],
[Md. Shakib] [0222210005101034][2] and
[Rafiu Anjir] [2104010202271][3]

[1,2,3]Department of Computer Science and Engineering, Premier University, Chattogram.

*Corresponding author(s). E-mail(s): mehrajbinahmed@gmail.com;
Contributing authors: shakibizzraj@gmail.com;
rafiuanjirrifat3@gmail.com;

Abstract

Purpose: Quality control in agriculture, specifically fruit grading, is traditionally a labor-intensive manual process prone to inconsistency and subjectivity. This study aims to automate the classification of mangoes and bananas into distinct ripeness stages to improve supply chain efficiency, reduce food waste, and ensure market consistency.

Methods: We propose a deep learning approach utilizing Transfer Learning with the EfficientNetB0 architecture. The methodology involves a rigorous pre-processing pipeline that extracts individual fruits from YOLO-formatted bounding boxes to create a focused classification dataset. A two-phase training strategy was employed: initially training the custom classification head to stabilize weights, followed by fine-tuning the top layers of the backbone to adapt feature representations to the specific textures of fruit skins. Data augmentation techniques such as rotation, zoom, and translation were employed to enhance model robustness against visual variations.

Results: The model was trained on a dataset of 6,775 cropped images and evaluated on a held-out test set of 1,321 images. The system achieved a final Test Accuracy of 84.41% and a Test Loss of 0.3195. Precision and recall analysis indicates high performance on mango classes (F1-Score $>$ 0.94 for unripe mangoes), with specific challenges identified in distinguishing certain banana ripeness stages due to visual similarity with mangoes.

Conclusion: The results demonstrate that lightweight Convolutional Neural Networks (CNNs) like EfficientNetB0 are viable for real-time agricultural classification tasks. The system balances computational efficiency with high accuracy, making it suitable for deployment in resource-constrained environments without the need for heavy industrial hardware.

Keywords: Computer Vision, Fruit Classification, EfficientNet, Deep Learning, Agriculture, Transfer Learning

1 Introduction

1.1 Background

The global agricultural industry faces significant challenges in maintaining quality consistency across the supply chain. Post-harvest processing, particularly fruit grading, is a critical step that dictates market value, storage logistics, and shelf-life management. Grading involves sorting produce based on visual attributes such as type, size, shape, and ripeness. Traditionally, this process relies heavily on manual labor. Human graders, while adaptable, are susceptible to fatigue, subjective bias, and inconsistency over long shifts. This variability leads to economic losses, as incorrectly graded fruit may ripen prematurely during transport, causing spoilage of entire batches. With the advent of Industry 4.0, the integration of Artificial Intelligence (AI) and Computer Vision (CV) into agriculture—often termed "Precision Agriculture"—has become a priority for modernizing these workflows.

1.2 Problem Statement

Automating fruit ripeness classification from standard RGB images is computationally non-trivial. In real-world packing lines or field settings, images exhibit high variability in lighting conditions (e.g., shadows, overexposure), occlusion (fruits blocking each other), and background clutter (leaves, soil, machinery). Furthermore, the visual distinction between "unripe" (typically green) and "ripe" (yellow/red) stages often involves subtle color gradients and textural changes that vary significantly between species. For instance, a ripe banana and a ripe mango may share similar yellow hues, while an unripe mango and an unripe banana share green hues. A robust automated system must distinguish these inter-class similarities and intra-class variations without requiring computationally prohibitive hardware, ensuring accessibility for deployment in developing regions.

1.3 Objectives

The primary objective of this project is to develop a robust Convolutional Neural Network (CNN) system to classify mangoes and bananas into four specific categories: *Mango Ripe*, *Mango Unripe*, *Banana Ripe*, and *Banana Unripe* (labeled as class_3). Specifically, we aim to:

1. Implement a robust pre-processing pipeline to extract regions of interest (ROI) from raw images using bounding box annotations.
2. Utilize Transfer Learning with the **EfficientNetB0** architecture to minimize training time and computational resources while maximizing accuracy.
3. Analyze the performance using standard metrics including Accuracy, Precision, Recall, and F1-Score to determine the system's viability for industrial application.

2 Related Work

Computer vision in agriculture has evolved significantly, moving from hand-crafted feature extraction to deep representation learning. Early approaches relied on color histograms, texture filters (e.g., Gabor filters), and morphological operations combined with classifiers like Support Vector Machines (SVM) or k-Nearest Neighbors (k-NN). While effective for controlled environments with uniform backgrounds, these methods often failed to generalize to complex, real-world scenarios where lighting and orientation vary.

The introduction of Convolutional Neural Networks (CNNs) revolutionized the field. Architectures such as AlexNet and VGG16 (9) demonstrated superior performance in fruit sorting tasks by automatically learning hierarchical features—from simple edges to complex shapes. However, these older models are parameter-heavy; for example, VGG16 has approximately 138 million parameters, making it slow for real-time inference and difficult to deploy on edge devices like smartphones or Raspberry Pi systems.

Recent studies have shifted towards Residual Networks (ResNet) (4) to address the vanishing gradient problem in deeper networks. More importantly, Tan and Le (2) introduced EfficientNet, a family of models that scale network width, depth, and resolution uniformly using a compound coefficient. In the domain of fruit classification, recent works have demonstrated that Transfer Learning—using weights pre-trained on large datasets like ImageNet (10)—significantly outperforms training from scratch, especially when labeled agricultural data is scarce. This project builds upon these findings by applying EfficientNetB0, the lightest variant, to a multi-class ripeness detection problem, focusing explicitly on computational efficiency.

3 Dataset

3.1 Data Source

The dataset employed in this study was obtained strictly from **Mendeley Data** (). It consists of high-quality RGB images containing mangoes and bananas in various environments. The data was originally annotated for object detection in the YOLO (You Only Look Once) format, providing bounding box coordinates for every fruit instance alongside the class labels.

- **Source Name:** Mango and Banana Dataset (Ripe Unripe) : Indian RGB image datasets for YOLO object detection (Version 3)
- **URL:** https://data.mendeley.com/datasets/y3649cmgg6/3
- **License:** CC BY 4.0

3.2 Data Preprocessing Pipeline

Since the objective is classification, feeding full images with complex backgrounds (leaves, baskets, soil) into the CNN would introduce significant noise and reduce model focus. We implemented a comprehensive crop extraction pipeline to convert the object detection annotations into a clean classification dataset.

3.2.1 Coordinate Conversion and ROI Extraction

The original annotations were provided in normalized YOLO format: (x_c, y_c, w, h), where values range from 0 to 1. To extract the crops, we converted these to absolute pixel coordinates. To ensure the entire fruit was captured, including edges that might be cut off by tight bounding boxes, we applied a relative padding of 5%. The crop was then extracted from the source image using array slicing.

3.2.2 Filtration and Resizing

Not all annotations represent valid data. We filtered out "micro-crops" (area $< 16 \times 16$ pixels) which typically represent noise or distant background objects. Finally, all valid crops were resized to 224×224 pixels using bicubic interpolation to match the input requirements of the EfficientNetB0 architecture.

3.3 Visual Samples

The following figures demonstrate the variety within the dataset after the cropping and resizing process.

3.4 Exploratory Data Analysis (EDA)

Following extraction, we conducted a thorough analysis of the class distribution. Imbalanced datasets can lead to model bias, where the network over-predicts the majority class. As shown in Table 1, our dataset is reasonably balanced, although the 'Unripe Banana' (class_3) category has a slightly higher representation.

Table 1: Dataset Distribution per Class

Class Index	Class Label	Number of Images
0	banana_ripe	1514
1	mango_unripe	1533
2	mango_ripe	1685
3	class_3 (Banana Unripe)	2043
Total		**6775**

The total dataset size of 6,775 images is sufficient for Transfer Learning, as the pre-trained weights allow the model to converge with fewer samples than training from scratch would require.

(a) Variation in Mango Ripeness

(b) Variation in Banana Ripeness

Fig. 1: Sample crops showing the primary classes. Note the textural differences between the smooth mango skin and the faceted banana shape.

4 Methodology

4.1 Theoretical Foundations of CNNs

Our approach is founded on Convolutional Neural Networks (CNNs), a class of deep neural networks designed for processing grid-like topology data such as images. A typical CNN consists of three primary types of layers: convolutional layers, pooling layers, and fully connected layers.

4.1.1 Convolution Operation

The core building block is the convolution layer, which applies a set of learnable filters (kernels) to the input. For an input image I and a filter K of size $f \times f$, the feature map S is computed as:

$$S(i,j) = (I * K)(i,j) = \sum_{m=0}^{f-1} \sum_{n=0}^{f-1} I(i+m, j+n) K(m,n) \tag{1}$$

This operation allows the network to detect local features such as edges, textures, and corners. The weights of the kernel K are learned during training via backpropagation. By stacking multiple convolution layers, the network learns a hierarchy of features: early layers detect simple edges, while deeper layers detect complex shapes like fruit stems or peel spots.

(a) Lighting Variations (Shadows/Highlights)

(b) Occlusion and Orientation challenges

Fig. 2: Complex samples illustrating the challenges of the dataset, including varying lighting conditions and non-standard orientations.

4.1.2 Non-Linear Activation

After convolution, a non-linear activation function is applied to introduce non-linearity into the model, allowing it to learn complex decision boundaries. We rely on the activation functions embedded in EfficientNet (Swish) and our custom head (ReLU/Softmax). The Rectified Linear Unit (ReLU) is defined as:

$$f(x) = \max(0, x) \tag{2}$$

4.1.3 Pooling Layers

Pooling layers reduce the spatial dimensions (Width $\times$ Height) of the feature maps, thereby reducing the number of parameters and computation. They also provide translation invariance, meaning the network can recognize a banana regardless of its exact position in the crop. Global Average Pooling (GAP), used in our model head, computes the average value of each feature map:

$$GAP_k = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} S_k(i, j) \tag{3}$$

4.2 EfficientNet Architecture

We utilized **EfficientNetB0** as our backbone. Unlike traditional scaling methods that arbitrarily increase depth (number of layers, as in ResNet) or width (number of channels, as in WideResNet), EfficientNet uses a Compound Scaling method.

4.2.1 Compound Scaling

EfficientNet scales depth (α), width (β), and resolution (γ) uniformly using a compound coefficient ϕ. The intuition is that if the input image is bigger, the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image.

$$\text{depth: } d = \alpha^{\phi}, \quad \text{width: } w = \beta^{\phi}, \quad \text{resolution: } r = \gamma^{\phi} \tag{4}$$

subject to $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ and $\alpha \geq 1, \beta \geq 1, \gamma \geq 1$. For EfficientNetB0, $\phi = 0$, providing a baseline model that is highly optimized for efficiency.

4.2.2 MBConv Blocks

The fundamental building block of EfficientNet is the Mobile Inverted Bottleneck Convolution (MBConv). It differs from standard residual blocks in two ways:

1. **Inverted Residuals:** The block expands the channel dimension using a 1×1 convolution, performs depth-wise convolution, and then compresses the channels back. This connects the "bottlenecks" (thin layers) with residuals, which is more memory efficient.
2. **Depthwise Separable Convolutions:** Standard convolution performs spatial and channel-wise computation simultaneously. Depthwise convolution splits this into two steps: a spatial convolution for each channel independently, followed by a 1×1 pointwise convolution to mix channels. This drastically reduces the FLOPs (Floating Point Operations).

4.2.3 Squeeze-and-Excitation (SE)

EfficientNet incorporates Squeeze-and-Excitation blocks within the MBConv structure. The SE block adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels. It "squeezes" global spatial information into a channel descriptor (via Global Average Pooling) and then "excites" (reweights) the channels via a fully connected layer with a sigmoid activation. This allows the network to emphasize informative features (e.g., the color yellow) and suppress less useful ones (e.g., background noise).

4.3 Proposed Model Architecture

Our complete architecture consists of the EfficientNetB0 backbone connected to a custom classification head designed for our 4-class problem:

- **Input Layer:** Accepts tensors of shape $(224, 224, 3)$.

- **Backbone:** EfficientNetB0 (include_top=False, weights='imagenet').
- **Global Average Pooling 2D:** Reduces the tensor $H \times W \times C$ to a vector of size $1 \times 1 \times C$.
- **Dropout (0.3):** A regularization technique that randomly sets 30% of input units to 0 during training updates. This prevents the network from becoming overly reliant on specific neurons, thus reducing overfitting.
- **Dense Output Layer:** A fully connected layer with 4 neurons and **Softmax** activation. The Softmax function converts the raw logits into probability distributions:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{5}$$

  where z is the input vector and $K = 4$ is the number of classes.

4.4 Data Augmentation Strategy

To improve model generalization and simulate real-world variability, we integrated a dynamic augmentation pipeline using Keras preprocessing layers. These transformations are applied on-the-fly during training, meaning the model never sees the exact same image twice.

- **RandomFlip ("horizontal"):** Simulates the fruit facing different directions (left/right).
- **RandomRotation (0.07):** Simulates slight camera tilts or fruit orientation changes (approx. $\pm 25°$).
- **RandomZoom (0.07):** Simulates varying distances between the camera and the fruit.
- **RandomTranslation (0.04):** Shifts the image slightly horizontally or vertically to ensure the model doesn't rely on the fruit being perfectly centered.
- **RandomContrast (0.06):** Adjusts the contrast to simulate varying lighting intensities in the field or packing house.

5 Training Procedure

5.1 Experimental Setup

The training was conducted in a controlled cloud environment to ensure reproducibility.

- **Platform:** Kaggle Notebooks
- **Accelerator:** NVIDIA Tesla P100-PCIE-16GB GPU
- **Language:** Python 3.11.13
- **Framework:** TensorFlow 2.18.0 / Keras
- **Random Seed:** Fixed at `42` for all random number generators (NumPy, TensorFlow, Python random).

5.2 Loss Function

We utilized **Categorical Crossentropy** as the objective function. This is the standard loss for multi-class classification problems where labels are one-hot encoded. The loss L quantifies the difference between the predicted probability distribution $\hat{y}$ and the true distribution y:

$$L(y, \hat{y}) = -\sum_{c=1}^{M} y_{o,c} \log(\hat{y}_{o,c}) \tag{6}$$

where M is the number of classes (4). Minimizing this loss maximizes the log-likelihood of the correct class.

5.3 Optimizer

We chose the **AdamW** optimizer (Adam with Weight Decay). Standard Adam (Adaptive Moment Estimation) computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. However, standard Adam implementation of L2 regularization is not identical to weight decay. AdamW decouples the weight decay from the gradient update, which has been shown to improve generalization performance and training stability.

- Learning Rate Schedule: Fixed for each phase.
- Weight Decay: $1e-5$
- Beta 1: 0.9, Beta 2: 0.999

5.4 Two-Phase Training Schedule

Directly fine-tuning a pre-trained model with random weights in the head can destroy the learned features of the backbone due to large gradient updates. Therefore, we adopted a two-phase strategy:

5.4.1 Phase 1: Head Training (Warm-up)

- **Goal:** Initialize the random weights of the custom Dense layer without disturbing the pre-trained backbone weights.
- **Configuration:** Backbone (EfficientNetB0) layers are **Frozen** (trainable=False).
- **Epochs:** 6
- **Learning Rate:** $1e^{-3}$
- **Observation:** Validation accuracy typically increases rapidly in this phase as the linear classifier aligns with the feature extraction.

5.4.2 Phase 2: Fine-Tuning

- **Goal:** Adapt the top layers of the backbone to fruit-specific features. While ImageNet has fruit categories, they may not match the specific varieties or ripeness stages in our dataset.
- **Configuration:** The top 20 layers of the backbone are **Unfrozen** (trainable=True).

- **Epochs:** 25 (with Early Stopping patience=8).
- **Learning Rate:** $1e^{-4}$. We reduce the learning rate by a factor of 10 to prevent "catastrophic forgetting" of the pre-trained features.

6 Results

6.1 Evaluation Metrics

To rigorously assess performance, we employed the following metrics:

- **Accuracy:** The ratio of correctly predicted observations to the total observations.
- **Precision:** The ability of the classifier not to label as positive a sample that is negative.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7}$$

- **Recall (Sensitivity):** The ability of the classifier to find all the positive samples.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{8}$$

- **F1-Score:** The harmonic mean of Precision and Recall.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

6.2 Quantitative Analysis

The model was evaluated on an independent test set of 1,321 images.

- **Final Test Loss:** 0.3195
- **Final Test Accuracy:** 84.41%

Table 2 presents the detailed classification report. The model achieves near-perfect performance on `class_3` (F1: 0.9783) and strong performance on `mango_unripe` (F1: 0.9452).

Table 2: Detailed Classification Report (Test Set)

Class	Precision	Recall	F1-Score	Support
banana_ripe	0.8222	0.4253	0.5606	261
class_3 (Unripe Banana)	1.0000	0.9575	0.9783	400
mango_ripe	0.6261	0.9734	0.7620	301
mango_unripe	0.9791	0.9136	0.9452	359
Accuracy			**0.8441**	**1321**
Macro Avg	0.8568	0.8175	0.8115	1321
Weighted Avg	0.8740	0.8441	0.8375	1321

6.3 Visual Analysis

6.3.1 Learning Dynamics

The learning curves shown in Figure 3 demonstrate the effectiveness of the two-phase strategy. During Phase 1 (Epochs 0-6), the accuracy rises sharply as the head weights align. At the transition to Phase 2 (indicated by the vertical line), there is a secondary, more gradual improvement in accuracy as the backbone adapts. The validation loss remains close to the training loss, suggesting that our regularization strategies (Dropout and Data Augmentation) successfully prevented overfitting.

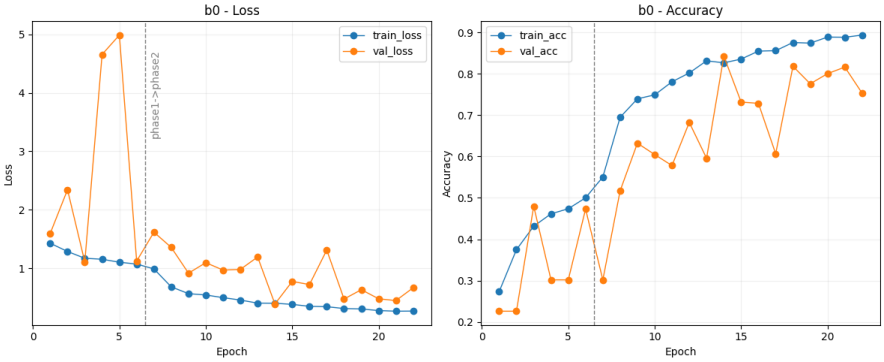

Fig. 3: Training and Validation Loss/Accuracy curves over epochs. The vertical line marks the transition from frozen-backbone training to fine-tuning.

6.3.2 Error Analysis via Confusion Matrix

The confusion matrix (Figure 4) provides deeper insight into misclassifications.

- **Mangoes:** The model distinguishes between ripe and unripe mangoes with high accuracy.
- **Bananas:** The primary error source is the `banana_ripe` class. A significant number of these instances are misclassified as other classes. This is likely due to the visual similarity between yellow ripe bananas and yellow ripe mangoes in specific crops where shape context is lost (e.g., a zoomed-in square crop of yellow skin looks identical for both fruits).

7 Discussion

7.1 Interpretation

The proposed EfficientNetB0 model successfully automated the grading of mangoes and bananas with an overall accuracy of 84.41%. The high F1-scores for Unripe Mangoes (0.94) and Unripe Bananas (0.97) demonstrate that the model is highly effective

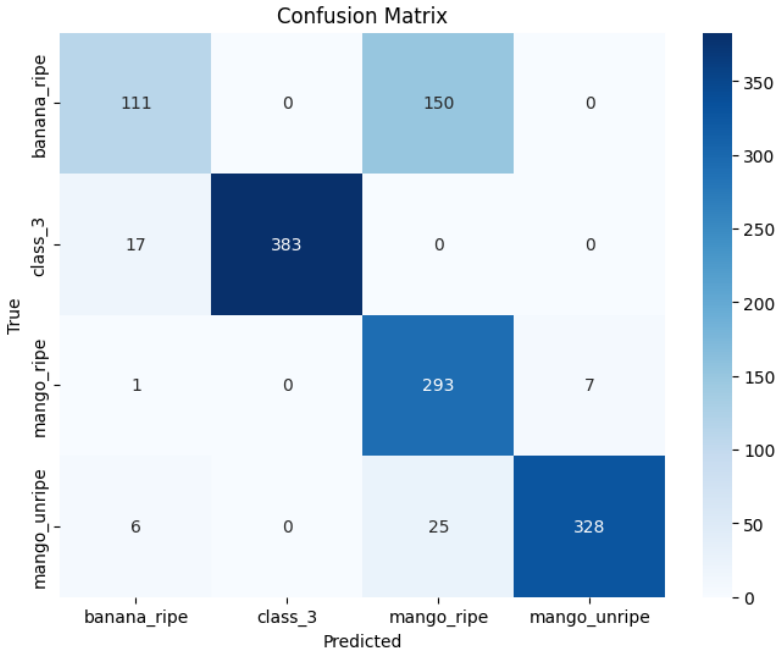

Fig. 4: Confusion Matrix Heatmap. The diagonal elements represent correct predictions, while off-diagonal elements represent errors.

at identifying early-stage fruit. The "green" texture features are evidently distinct and well-captured by the CNN filters. The struggle with identifying Ripe Bananas (Recall 0.42) highlights a limitation in feature discrimination when color (yellow) is the dominant feature shared between Ripe Bananas and Ripe Mangoes.

7.2 Limitations

The reliance on square crops (224×224) is a geometric limitation. Bananas are elongated, and square cropping may exclude the ends of the fruit, removing shape context that distinguishes them from round mangoes. Additionally, the dataset, while diverse, may not fully represent the lighting extremes found in open-field farming.

7.3 Ethical Considerations

While this technology promises increased efficiency, it poses ethical questions regarding agricultural labor. Automated grading could displace low-skilled workers who rely on manual sorting for income. It is imperative that such systems are deployed as "assistive" technologies—enhancing human productivity and reducing eye strain—rather than as replacements.

8 Conclusion and Future Work

This project presented a comprehensive deep learning framework for fruit ripeness classification. By leveraging the efficiency of the EfficientNetB0 architecture and a robust two-phase transfer learning strategy, we achieved a viable accuracy of 84.41% with minimal computational cost. The study highlights the importance of careful data preprocessing and the effectiveness of fine-tuning pre-trained models for agricultural domains.

Future Work:

- **Architectural Improvements:** Investigating Vision Transformers (ViT) to better capture global shape context.
- **Object Detection Integration:** Moving to a full YOLOv8 pipeline to handle detection and grading simultaneously.
- **Mobile Deployment:** Quantizing the model to TFLite for Android deployment.

References

References

[1] Sutar, A., Naikare, A., Jadhav, P., & Kute, R. (2023). Mango and Banana Dataset (Ripe Unripe): Indian RGB image datasets for YOLO object detection. *Mendeley Data*, V3. https://doi.org/10.17632/y3649cmgg6.3

[2] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *International Conference on Machine Learning (ICML)*.

[3] Keras Team. (2024). *Keras Applications Documentation*. Available at: https://keras.io/api/applications/

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[5] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *CVPR*.

[6] Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*.

[7] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *CVPR*.

[8] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *CVPR*.

[9] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR*.

[10] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. *CVPR*.