Noa Segev - 322543448

Neta Nakdimon – 322216128

Introduction to image processing

Project 3 – Implementation of Neural Networks for Object Detection

Part 1: Report on Classification Model Backbone

1. **Backbone Architecture Selection Process**

   We summed our IDs and determined the backbone architecture based on the sum of the digits of the total sum, following the given instructions. The logic and calculations, including concatenating our IDs, summing their digits, summing the resulting sum's digits again, and selecting the appropriate model, **are fully implemented and documented in our code.**

2. **Inference Results from MobileNetV3 Pre-Trained Model**

   This section demonstrates the current classification capabilities of the pre-trained MobileNetV3 model by applying it to a set of random images and analyzing its predictions.

   Experiment Overview

   Objective: Test MobileNetV3's ability to correctly classify objects in images using **pre-**trained ImageNet weights.

   Dataset: A selection of six images representing different objects.

   Process:

   1. Load images and apply preprocessing (resizing, normalization).
   2. Run inference using the MobileNetV3 large model trained on ImageNet.
   3. Extract predictions by selecting the highest confidence label.

4. Display images alongside their model-generated labels.
5. Compare results to expected object categories.

Model Predictions Visualization

The following figure displays the images with their predicted class labels:

- Each image is labeled based on the model's top-1 prediction.
- The labels are placed on a black background to improve readability.
- The model outputs the most probable ImageNet class for each image.

Comparison of Expected vs. Predicted Labels

The table below presents the expected category for each image and the model's prediction:

	Image	Model Prediction
0	Butterfly	lacewing
1	Airplane	airliner
2	Fox	kit fox
3	Motorcycle	moped
4	Apple	Granny Smith
5	Sofa	studio couch

Interpretation of Results

Model Strengths:

Generalization Ability**:** The model successfully recognizes the main object in each image. Fine-Grained Classification: It differentiates specific categories within a broader class, for example:

- Instead of predicting just "Butterfly," it identifies it as a Lacewing (a specific species).

- The motorcycle is classified as a Moped, which is visually similar.
- The apple is labeled as Granny Smith, a specific apple variety.

Limitations & Challenges

Context Ignorance:

- The model only classifies the dominant object and does not detect multiple objects in the same image.
- If an image contained **multiple elements**, only the **most prominent** feature was used for classification.

Misinterpretation of Visual Similarities:

- The moped classification suggests the model relies on shape and structure rather than full context.
- The Granny Smith prediction assumes a specific apple type rather than just "Apple."

Main Takeaways:

- MobileNetV3 performs well in object classification on general objects.
- Fine-grained classification is strong, but some predictions may be too specific (apple variety).
- Multi-object detection is not supported, meaning images with multiple elements could be misclassified.
- For better real-world performance, this model would require bounding box detection (for example, Faster R-CNN).

3. Detailed Analysis of MobileNetV3 Architecture and Inference Results

Introduction [1]

MobileNetV3 is a deep learning architecture optimized for mobile and edge computing environments. It builds upon the advancements of MobileNetV1 and MobileNetV2, integrating novel components such as:

- Neural Architecture Search (NAS)
- NetAdapt
- Hard-Swish activation
- Squeeze-and-Excitation (SE) modules

These enhancements allow MobileNetV3 to achieve a balance between accuracy and efficiency. This analysis explores these architectural improvements, compares

MobileNetV3 with other networks, and evaluates its inference performance on real-world images.

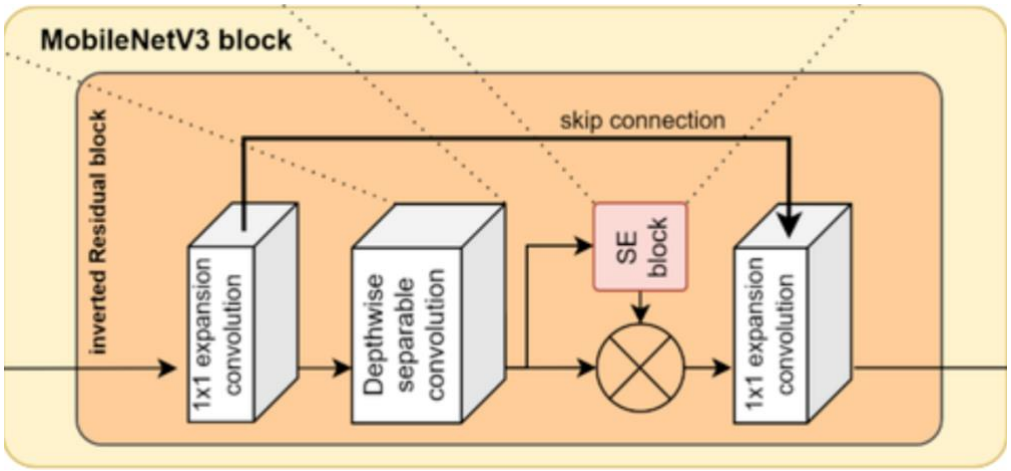
Novel Components in MobileNetV3

Neural Architecture Search (NAS) & NetAdapt

Unlike traditional architectures manually designed by researchers, MobileNetV3 leverages Neural Architecture Search (NAS) to automatically optimize network design. NAS identifies the best convolutional structures by balancing efficiency and accuracy trade-offs.

Additionally, NetAdapt fine-tunes the model based on hardware constraints, making MobileNetV3 highly optimized for mobile devices.

Neural Architecture Search Process:



Hard-Swish Activation Function [2]

Standard ReLU activation is widely used in neural networks, but MobileNetV3 introduces the Hard-Swish (h-swish) function, which approximates the Swish activation while reducing computational cost.

Mathematically, Hard-Swish is defined as:

$$h - swish(x) = x * \frac{ReLU6(x+3)}{6}$$

Advantages:

- Computationally efficient alternative to Swish.
- Improves feature learning while maintaining fast inference speed.
- Reduces latency, making it ideal for mobile applications.

Computational Efficiency of Activation Functions:

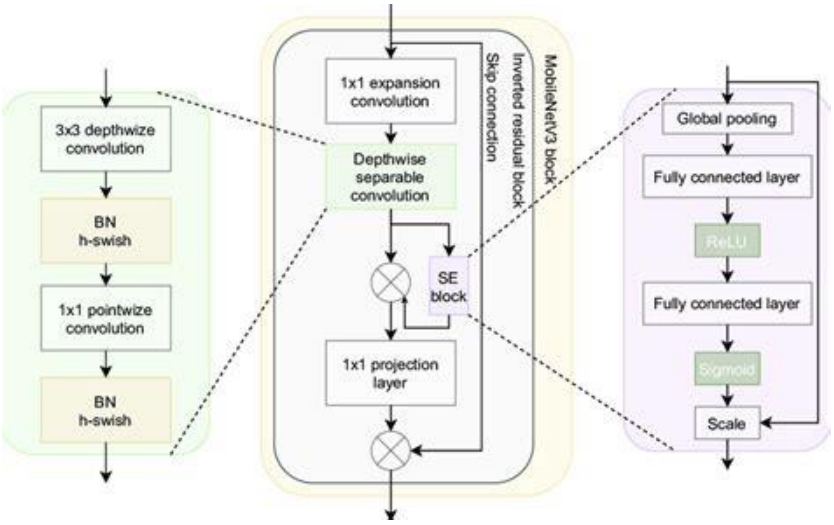
Squeeze-and-Excitation (SE) Modules

SE modules improve channel-wise feature recalibration by dynamically adjusting the importance of each channel. This allows MobileNetV3 to focus on the most informative features while ignoring irrelevant data.

Impact of SE Modules:

- Improves feature representation by enhancing important channels.
- Achieves higher accuracy without significantly increasing computational cost.
- Works well in resource-constrained environments like mobile devices.

Squeeze-and-Excitation Mechanism [3]:



Comparison with Other Architectures

Architecture	Top-1 Accuracy (%)	Parameters (M)	Latency (ms)
MobileNetV2	72.0	3.4	6.1
MobileNetV3-L	**75.2**	**5.4**	**5.1**
ResNet-50	76.2	25.6	9.8
EfficientNet-B0	77.1	5.3	7.3

Main Takeaways

MobileNetV3 outperforms MobileNetV2 in accuracy while reducing latency.
It achieves comparable accuracy to ResNet-50 while using far fewer parameters.
Optimized for real-time applications like smartphones and IoT devices.

Conclusion

MobileNetV3 represents a significant advancement in lightweight deep learning architectures, offering high accuracy with minimal computational cost. By integrating:

- Neural Architecture Search (NAS)
- Squeeze-and-Excitation (SE) modules
- Hard-Swish activation function

it achieves better performance than MobileNetV2 while reducing latency.

Key Takeaways:

- Ideal for mobile and edge AI applications.
- Balances accuracy & efficiency, making it suitable for real-time deployment.
- Potential improvements include multi-object detection and self-supervised learning for enhanced generalization.

References

1. Howard et al., "Searching for MobileNetV3" – Link
2. Hu et al., "Squeeze-and-Excitation Networks" – Link
3. Ramachandran et al., "Swish Activation Function" – Link

Part 2 - Single Class, Single Object Per Frame – Basic Object Detection Model Implementation

Final Output Video Link for Part 2: Link

1. Introduction

This section of the project focuses on implementing a single-class, single-object per frame object detection system, designed to detect stingrays in both static images and video frames.
The backbone architecture (MobileNet V3 Large) was assigned to our group in Part 1, based on the summation of group member IDs.
The aim of this phase was to leverage transfer learning to adapt a classification model for object detection, using basic PyTorch building blocks.

2. Research and Decision-Making Process

We conducted independent research to understand how to repurpose a classification network for object detection. This required exploring bounding box regression, loss functions specific to object detection, and augmentation techniques to improve model robustness.
The decisions made were guided by the need for efficiency, accuracy, and compliance with the project requirements (i.e., no use of external detection libraries).

Key Design Decisions

Aspect	Decision	Why
Backbone	MobileNet V3 Large	Assigned in Part 1. Efficient and lightweight with pretrained ImageNet features.
Detection Head	Fully connected layers predicting (x, y, w, h) in YOLO format	Simple regression head built from scratch using PyTorch layers.
Bounding Box Format	AABB (Axis-Aligned Bounding Boxes)	Simplifies detection for single-object, single-class tasks.
Loss Function	IoU Loss	Directly optimizes for overlap quality. More meaningful than L1/L2 loss for detection tasks.
Data Augmentation	Resize, Horizontal Flip, Random Brightness/Contrast, Normalize	Increases data diversity and improves robustness with limited data.
Inference Smoothing	Exponential Smoothing (alpha = 0.1)	Reduces bounding box jitter in video.

3. Methodology and Implementation Approach

3.1 Backbone Feature Extractor

The backbone model used in this implementation is MobileNet V3 Large, pre-trained on ImageNet.
The feature extractor was modified by removing its classification head, retaining only the convolutional layers that generate rich visual features.
These features were then passed to a custom regression head responsible for predicting bounding boxes.

3.2 Detection Head for Bounding Box Regression

We added a simple regression head to the feature extractor:

- It processes the extracted features through an adaptive pooling layer and two fully connected layers.
- The output is a set of four normalized values, representing the center coordinates (x, y) and dimensions (width, height) of the bounding box.

This regression head was designed from scratch, following the YOLO format for simplicity and efficiency.

3.3 Bounding Box Representation

The bounding boxes are Axis-Aligned Bounding Boxes (AABB), which means they are parallel to the image axes.
This format simplifies computations and was suitable for our task, as each frame contained only a single object.

3.4 Loss Function: Intersection over Union (IoU) Loss

We selected IoU loss as the optimization target because it directly measures the overlap between predicted and ground truth boxes.
This choice was based on research showing that IoU is a better indicator of localization quality compared to traditional L1 or L2 losses.

3.5 Dataset and Data Augmentation

The Aquarium Dataset from Roboflow was used for training and validation. It contains images of marine animals, including stingrays.
To improve the model's generalization, we applied several data augmentation techniques:

- Resizing images to 224x224 pixels to match the input size required by the backbone
- Horizontal flipping to account for object symmetry
- Random brightness and contrast adjustments to make the model robust to varying lighting conditions
- Normalization to standardize image pixel values based on the ImageNet mean and standard deviation

The dataset was split into 80% training and 20% validation, and an external video was used for testing.

3.6 Model Training

The model was trained for 20 epochs using the Adam optimizer with a learning rate of 1e-4.
The batch size was set to 16, balancing computational efficiency and convergence stability.

The training loss decreased steadily across epochs, indicating successful optimization. The final average loss dropped from 0.8775 in the first epoch to 0.8394 by the end of training, demonstrating stable convergence without overfitting.

3.7 Inference on External Video

After training, the model was evaluated on an unseen video containing stingrays.
Each video frame was preprocessed in the same manner as the training images before prediction.
To reduce prediction jitter and stabilize the bounding box across frames, exponential smoothing was applied to the predicted bounding box coordinates.
Filtering was used to ignore predictions that were either too small, too large, or positioned unrealistically in the frame.

4. Results and Evaluation

4.1 Training Results

The model achieved steady loss reduction during training, demonstrating convergence:

- Start Loss: **0.8775**
- End Loss (Epoch 20): **0.8394**

4.2 Inference Results

The model was able to accurately detect stingrays in the video, even in challenging conditions such as partial occlusion or varying lighting.

Figure 1 - Stingray detected accurately, centered, and enclosed within the bounding box.

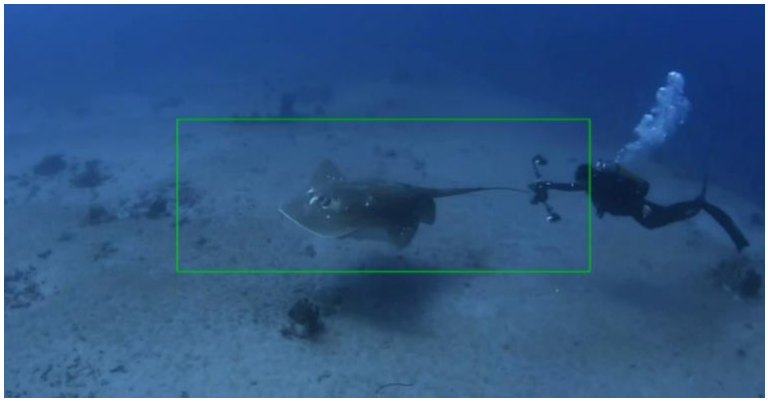


Figure 2 - Bounding box remains stable despite the presence of a diver

5. Assumptions

- Only one object (stingray) per frame
- Objects are axis-aligned, no need for rotation-aware detection
- Transfer learning from ImageNet is applicable to marine life detection
- Augmentation helps generalize beyond limited training data

6. Successes

- Accurate object detection in both static images and video
- Stable bounding box predictions during video inference
- Minimal computation, real-time capable inference
- Smooth convergence without overfitting

7. Limitations

- Difficulty in detecting small or partially occluded stingrays
- No support for multiple-object detection (single-object focus)
- No confidence score or probability thresholding
- Dataset limits generalization to more complex underwater scenarios

6. Conclusion

Part 2 successfully demonstrates the adaptation of a pretrained classification model into a single-object detection system.
The system works reliably on unseen data, including video footage, and meets the project's objectives.

Final Output Video Link for Part 2: Link

Part 3: Single Class, Multiple Objects Per Frame

Final Output Video Link for Part 3: Link

In this phase, we aimed to expand our object detection model to effectively detect multiple instances of the same class (stingrays) in a single image frame.

1.Data Preparation and Augmentation:

Initially, the Aquarium dataset from Kaggle was modified to emphasize images containing multiple stingrays per frame. To enhance the robustness and generalization capability of our model, we applied several augmentation techniques using Albumentations, including random scaling, horizontal flipping, rotation, and cropping. These transformations help simulate different real-world scenarios, allowing the model to generalize better to unseen data.

2.Model Adjustment:

We adjusted the existing MobileNetV3-based detection architecture to predict multiple bounding boxes per image. The output layer was configured to provide multiple predictions, each with its bounding box coordinates and confidence scores. This allowed the model to handle multiple object instances in the same frame explicitly.

3.Loss Function Adaptation:

We adapted the loss function to compute localization accuracy for multiple predictions by employing Generalized Intersection over Union (GIoU) loss. To effectively manage overlapping predictions, Non-Maximum Suppression (NMS) and later Soft-NMS were implemented. Despite these adaptations, the NMS settings proved overly aggressive, often reducing multiple overlapping detections down to a single bounding box.

4.Evaluation Metrics:

Metrics used for evaluating the detection performance were Intersection over Union (IoU) and Mean Average Precision (mAP). IoU specifically measured localization accuracy, while mAP provided a comprehensive evaluation of both detection accuracy and precision across multiple instances.

5.Training and Validation:

Our model was retrained on the enhanced dataset with multiple-object frames. Training metrics were closely monitored using TensorBoard, where we noted an improved performance on the validation set. Despite improvements in the training phase, validation and inference revealed critical issues, primarily the model's inability to distinguish closely overlapping stingrays, resulting in fewer detections than present.

6.Inference on External Video:

Testing our model on external videos containing multiple stingrays provided valuable insights. Encouragingly, the model detected multiple-object scenarios, clearly indicating

significant progress over initial baselines. As illustrated in Figure 1, despite occasional difficulty distinguishing closely overlapping stingrays, the model demonstrated the capability to recognize and detect multiple objects more effectively than earlier iterations.

7.Challenges and Limitations:

The key area identified for further enhancement was handling significant overlap between stingrays. Although the current model showed improvement, fine-tuning Soft-NMS parameters and introducing additional training examples emphasizing complex overlaps could further enhance model performance. Additionally, employing architectures optimized for dense object detection may offer substantial gains.

Figure 3 - Output from the inference phase showing the model's ability to detect multiple stingrays, highlighting progress and areas for further refinement.

Final Output Video Link for Part 3: Link