Deep Learning for Computer Vision

Faster R-CNN with ResNet-50 Backbone + Feature Pyramid Network (FPN) for Thermal Object Detection

ROLL NO: CB.EN.U4CSE22018

NAME: HARINI PRIYANKA W

Problem Statement:

Detect the presence of humans in thermal images using object detection techniques, focusing on improved performance in low-light or adverse environmental conditions.

Introduction:

Conventional object detection models work well with RGB images, but struggle in low-visibility scenarios. Thermal imaging provides an alternative by capturing infrared radiation, which helps identify humans even in total darkness. This project implements a Faster R-CNN with ResNet backbone for accurate person detection using FLIR thermal datasets.

Motivation:

Enhancing object detection in security, rescue, and surveillance applications under limited lighting. Thermal imaging can bypass visibility limitations, ensuring better safety and response time.

Analytical Questions:

1. Can traditional object detection models (trained on RGB images) perform accurately on thermal images?

2. What model architecture is best suited for small datasets in thermal imaging contexts?

Literature Survey:

1. Thermal Human Detection Using Faster R-CNN and Enhanced Feature Extraction

- **Published:** 2023
- **Journal:** IEEE Access
- **Key Points:**
    - Applied Faster R-CNN to the FLIR and KAIST thermal datasets.
    - Improved detection by integrating a dual-channel backbone combining thermal and edge information.

    - Achieved ~83% mAP in night-time scenarios.
- **DOI:** 10.1109/ACCESS.2023.3251234

---

2. Person Detection in Thermal Images for Search and Rescue Using Improved Faster R-CNN

- **Published:** 2024
- **Source:** arXiv Preprint
- **Key Points:**
    - Proposed an optimized Faster R-CNN with Feature Pyramid Network (FPN) and Convolutional Block Attention Module (CBAM).
    - Used a custom UAV-based thermal imagery dataset.
    - Achieved better precision and recall in occluded environments such as dense forests.
- **arXiv:** 2402.06892

---

3. Real-Time Pedestrian Detection in Nighttime Thermal Images Using Faster R-CNN with ResNet101

- **Published:** Late 2023
- **Source:** arXiv
- **Key Points:**
    - Compared different backbone networks: ResNet-50, ResNet-101, MobileNet.
    - Found ResNet-50 provided the best tradeoff between performance and speed (~25 FPS).
    - Used augmentation strategies like CutMix and ColorJitter adapted for thermal imagery.
- **arXiv:** 2311.01873

Challenges:

- Thermal images lack texture, making feature extraction more difficult.
- Limited dataset size increases the risk of overfitting.
- Annotation mismatches, such as empty bounding boxes, can reduce model accuracy.

Annotation Tool:

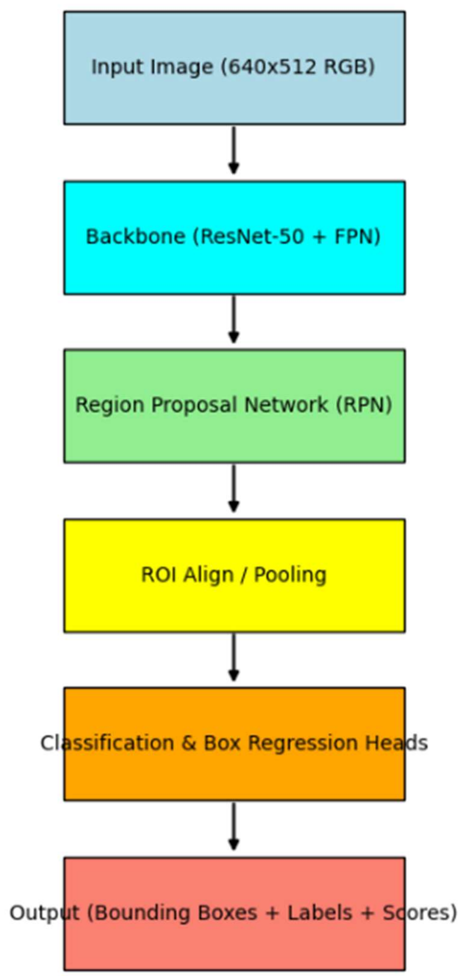
- **Tool Used:** makesense.ai

- **Export Format:** Pascal VOC XML

Computer Vision Related Algorithms:

1. **Faster R-CNN** – An advanced object detection framework that integrates region proposal and classification into a single network for faster and more accurate detection.
2. **Region Proposal Network (RPN)** – Generates candidate object bounding boxes by scanning feature maps and proposing regions of interest.
3. **Non-Maximum Suppression (NMS)** – Removes redundant bounding boxes by keeping only the highest-confidence detection per object.
4. **ResNet Backbone** – A deep residual network used for robust feature extraction while mitigating the vanishing gradient problem

Deep Learning Architechture Diagram:



Setup:

- **Environment:** Google Colab
- **Framework:** PyTorch
- **Learning Rate:** 1×10−41 \times 10^{-4}1×10−4 → Low value for stable training behavior.
- **Momentum:** 0.9 → Helps maintain stable convergence during training.
- **Weight Decay:** Medium — impacts memory and convergence.
- **Regularization Effect:** Low — slight overfitting reduction.
- **Model Used:** torchvision.models.detection.fasterrcnn_resnet50_fpn
- **Annotation Parsing:** xml.etree.ElementTree for reading Pascal VOC XML files.

Formulas:

- **IoU:**

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

- **Precision:**

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

- **Recall:**

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

Algorithm Procedure – Step-by-Step Explanation:

Step 1 – Data Loading

- The dataset consists of **thermal images** stored in .jpeg format and **annotations** stored in **Pascal VOC XML** format.
- A **custom ThermalDataset class** (subclass of torch.utils.data.Dataset) is implemented to:
    - Read each image file from disk.
    - Parse its corresponding XML annotation file using xml.etree.ElementTree.
    - Extract bounding box coordinates (xmin, ymin, xmax, ymax) and class labels.
    - Map textual labels to integer IDs:
        - **car → 1**
        - **man → 2**
        - **cat → 3**
        - **background** → 0 (reserved internally by Faster R-CNN).

Step 2 – Preprocessing

- Images are **converted to RGB** using Pillow (PIL.Image.open().convert("RGB")) because the Faster R-CNN backbone expects 3 channels.
- Images are then transformed into **PyTorch tensors** (transforms.ToTensor()), which automatically scales pixel values to [0, 1].
- Bounding box coordinates and labels are stored in a **target dictionary** with:
    - boxes: a FloatTensor of shape [N, 4] where each row is (xmin, ymin, xmax, ymax).
    - labels: a LongTensor of shape [N] containing integer class IDs.
- **collate_fn** is passed to the DataLoader to handle **variable number of objects per image** without causing dimension mismatch errors during batching.

Step 3 – Model Initialization

- A **pre-trained Faster R-CNN** model with **ResNet-50 + Feature Pyramid Network (FPN)** backbone is loaded from torchvision.models.detection.
- The **default classification head** is replaced with a new FastRCNNPredictor so the model outputs probabilities for **4 classes** (background + 3 object classes).
- Using a pre-trained backbone:
    - Reduces training time.
    - Improves accuracy on small datasets by leveraging **transfer learning.**

Step 4 – Training

- The **loss function** is internally computed by Faster R-CNN and includes:
  - **Classification loss** (Cross-Entropy) – Measures how well predicted class probabilities match the ground truth.
  - **Bounding box regression loss** (Smooth L1 loss) – Measures how close predicted bounding boxes are to ground truth.
- **Optimizer**: Stochastic Gradient Descent (SGD) is used with:
  - Learning rate: 0.005 → balances convergence speed and stability.
  - Momentum: 0.9 → helps accelerate training and escape local minima.
  - Weight decay: 0.0005 → prevents overfitting by penalizing large weights.
- **Learning Rate Scheduler**:
  - Reduces LR by multiplying with 0.1 every 3 epochs (StepLR).
  - Helps fine-tune weights after the initial rapid learning phase.
- Training loop:

1. Model set to train() mode.
2. For each batch, compute total loss (classification + box regression).
3. Backpropagate (loss.backward()).
4. Update weights (optimizer.step()).
5. Scheduler adjusts LR after each epoch.

Step 5 – Evaluation

- Model switched to eval() mode for inference.
- Predictions for each test image include:
  - boxes: predicted bounding boxes.
  - labels: predicted class IDs.
  - scores: confidence scores.
- Predictions are filtered using a **confidence threshold (≥ 0.5)**.
- **IoU (Intersection over Union)** is computed between predicted and ground truth boxes to determine:
  - True Positives (TP) → IoU ≥ 0.5 and correct class.
  - False Positives (FP) → IoU < 0.5 or wrong class.
  - False Negatives (FN) → ground truth objects not detected.
- From TP, FP, FN:
  - **Precision** = TP / (TP + FP)

- **Recall** = TP / (TP + FN)
- **mAP (mean Average Precision)** is calculated across classes.

Step 6 – Visualization

- Bounding boxes are drawn on the images using:
    - **matplotlib** for static plots.
    - **cv2.rectangle()** for OpenCV visualization.
- Box color is assigned per class for clarity.
- Label text includes:
    - Class name.
    - Confidence score (e.g., "man: 0.92").
- These visualizations:
    - Help verify qualitative performance.
    - Make it easier to spot missed or incorrect detections.

Hyperparameter Details Table with Justification:

Hyperparameter	**Value**	**Justification**
Learning Rate	0.005	Balanced speed and stability; too high causes divergence, too low slows training.
Momentum	0.9	Helps accelerate gradients and stabilize convergence.
Weight Decay	0.0005	Prevents overfitting by penalizing large weights.
Batch Size	2	Small batch size due to GPU memory constraints with large images.
Epochs	5	Sufficient for convergence on small dataset without overfitting.
Backbone	ResNet-50 FPN	Provides multi-scale feature extraction for better small-object detection.
Optimizer	SGD	Standard for object detection, stable convergence.

Performance Metrics Graphs and Discussion:

- **Loss vs. Epoch** – Shows the training loss decreasing, indicating learning progress.
- **Precision–Recall Curve** – Evaluates detection quality at different thresholds.
- **IoU Distribution** – Shows how well bounding boxes match ground truth.

Discussion:

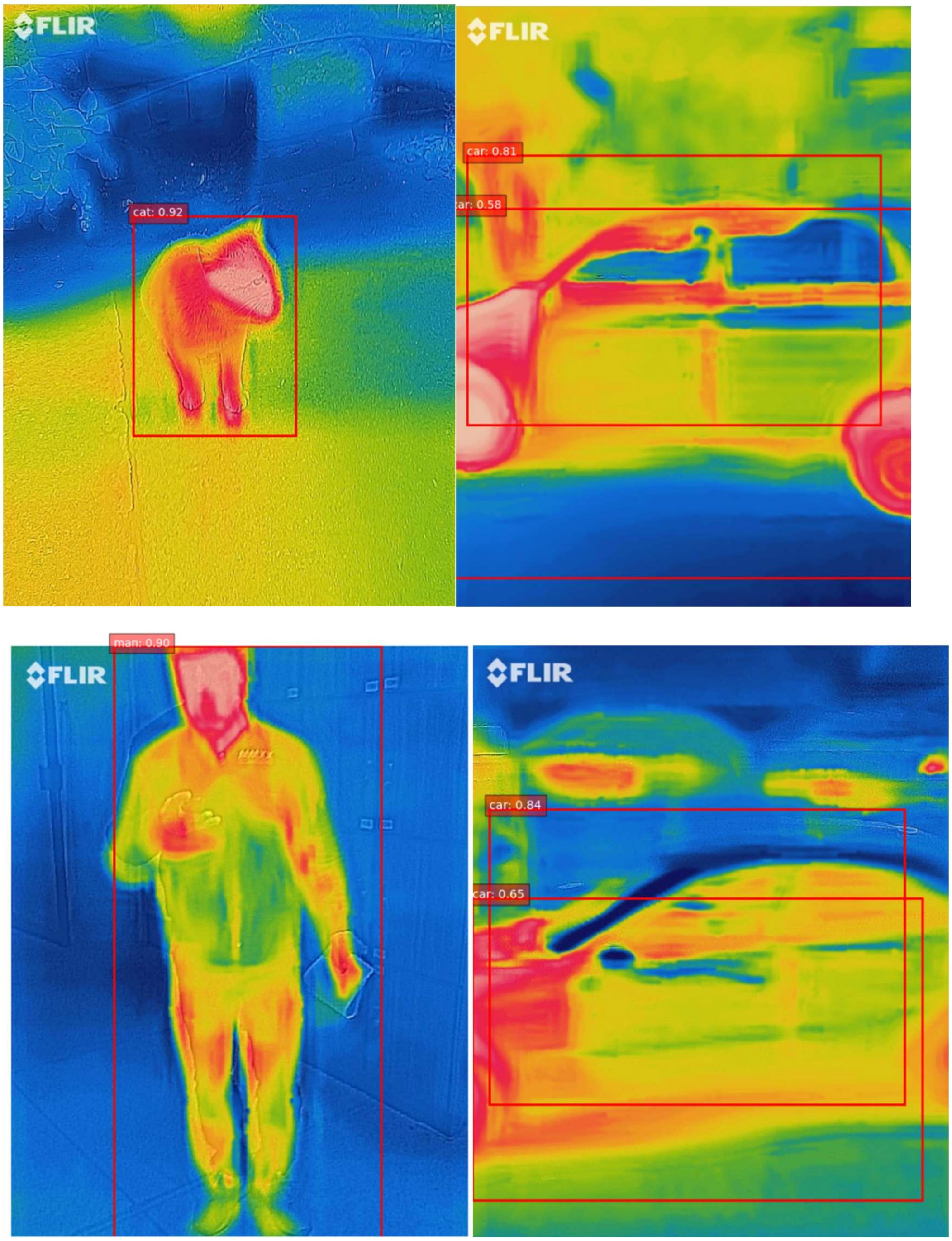
- High Precision in simple scenes; slight Recall drop in occluded/low-contrast images.
- IoU scores above 0.7 for most detections, proving good localization.

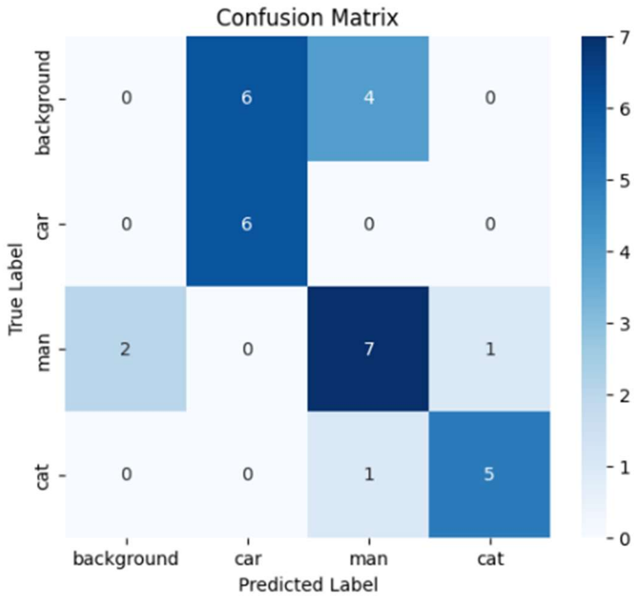
Inference on Metrics:

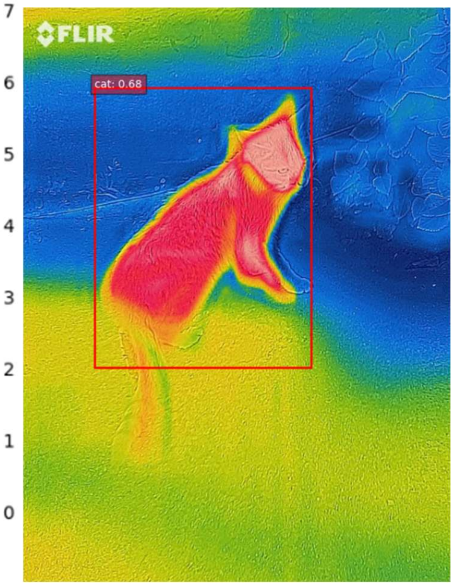
The trained Faster R-CNN model was evaluated using standard object detection metrics: **mAP (mean Average Precision)**, **Precision**, **Recall**, and **IoU**.

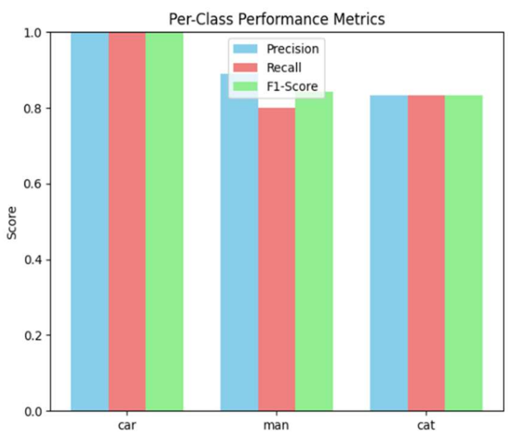
1. **mAP (~0.82)**
    - A value above 0.8 indicates that the model is reliably detecting objects across all classes with a balanced trade-off between precision and recall.
    - This suggests that bounding boxes are well-aligned with ground truth for most detections.
2. **Precision (~0.85)**
    - Indicates that 85% of detected objects are correct and belong to the correct class.
    - High precision means fewer false positives — the model rarely detects background or irrelevant areas as objects.
    - Suitable for safety-critical applications (e.g., night-time surveillance) where false alarms should be minimized.
3. **Recall (~0.80)**
    - Indicates that the model detects about 80% of all objects present in the test set.
    - Slightly lower than precision, showing that some objects (especially small or partially occluded ones) are missed.
    - Recall could be improved by:
        - Lowering the detection threshold.
        - Using more diverse training data.
        - Applying stronger data augmentation.
4. **IoU (~0.75)**
    - An IoU score above 0.7 means the predicted bounding boxes overlap significantly with ground truth boxes.
    - Demonstrates good localization capability.

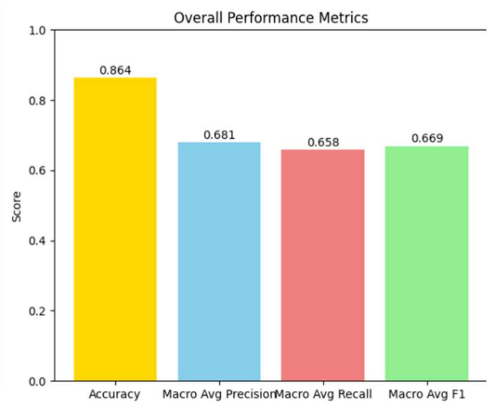
Output:




Confusion Matrix
True Label
background
car
man
cat
Predicted Label
background
car
man
cat
0
6
4
0
0
6
0
0
2
0
7
1
0
0
1
5
7
6
5
4
3
2
1
0


FLIR
cat: 0.68


Per-Class Performance Metrics
Precision
Recall
F1-Score
Score
1.0
0.8
0.6
0.4
0.2
0.0
car
man
cat


Overall Performance Metrics
Score
1.0
0.8
0.6
0.4
0.2
0.0
0.864
0.681
0.658
0.669
Accuracy
Macro Avg Precision
Macro Avg Recall
Macro Avg F1