Assignment 3 – Moved Object Detection with DETR (Option 2: Pixel Differences)

Computer Vision

Hailemariam Mersha (NetID: hbm9834)

December 12, 2025

Overview

This report describes my implementation of Option 2, where moved-object detection is performed by feeding a pixel-wise RGB difference of two frames into `facebook/detr-resnet-50`. The pipeline consumes only the matched annotation text files and produces direct DETR training targets without requiring any intermediate formats. The model predicts bounding boxes on the second frame, corresponding to the final locations of objects that moved. I describe the dataset construction, preprocessing, training setup, ablation results, and visualization outputs.

Method

Data construction

Dataset samples are created using the matched output generated by `data_ground_truth_labeller.py`. For each pair of frames:

- The two frames are loaded at native resolution.
- Their absolute pixel-wise RGB difference is computed:

$$diff(x, y) = |I_2(x, y) - I_1(x, y)|.$$

- Each annotation file contains two lines of bounding boxes: first-frame boxes and second-frame boxes.

Only the second-frame boxes serve as ground truth, since the task is to detect where the object ends up after moving. Extremely small or invalid boxes are filtered. If an annotation contains no boxes after filtering, the sample is skipped. The dataset is shuffled with a fixed seed and split 80/20 into training and validation subsets.

Preprocessing

All preprocessing is delegated to `DetrImageProcessor`. It performs resizing using the modern `shortest_edge/longest_edge` interface, normalization using DETR's ImageNet statistics, and automatic construction of training targets. The custom collate function handles variable-size inputs, preserves metadata for visualization, and excludes invalid samples. Importantly, the entire workflow preserves the spatial consistency required for accurate bounding-box regression.

Model

The model used is `facebook/detr-resnet-50`, modified with a six-class detection head for:

$$\{unknown, person, car, othervehicle, otherobject, bike\}.$$

The prediction and bounding-box layers are reinitialized accordingly. Four fine-tuning configurations are supported:

1. all: updates the entire DETR model,
2. backbone_only: updates only the convolutional backbone,
3. transformer_only: updates only the DETR transformer layers,
4. head_only: updates only the classifier and box-regressor heads.

Training setup

Training uses AdamW with:

- learning rate 1×10^{-5} (unless overridden in ablations),
- weight decay 1×10^{-4},
- batch size 2,
- 100 epochs for the baseline,
- 100 epochs for each ablation,
- gradient clipping (max-norm 1).

Validation reports DETR loss, precision, and recall using a score threshold of 0.2 and IoU threshold of 0.5. Only the best checkpoint is retained.

Evaluation and visualization

The evaluation script computes precision/recall and produces four-panel visualizations showing:

- ground-truth boxes (green) on the two input frames,
- predicted boxes (red) on the same frames.

These visualizations help interpret qualitative behavior such as false positives from shadows or noisy diff regions. Outputs are saved under `outputs/eval/` and `outputs/eval_vis/`.

Results

Baseline (100 epochs)

Training improved recall steadily while precision varied with confidence distribution. The best checkpoint was at epoch 31.

- best validation loss: 1.0783

- validation precision: 0.453
- validation recall: 0.775

Final evaluation on the held-out portion of the data yielded:

$$Precision = 0.1068, \qquad Recall = 0.6458.$$

Although precision is low, recall remains strong. This behavior is expected with small datasets and a high-capacity model: DETR proposes many candidate boxes, and pixel differences naturally emphasize broader motion regions. Even when inaccurate, these outputs still allow meaningful comparison across fine-tuning strategies.

Ablation study (100 epochs each)

All ablation results below are computed on the validation split using score threshold 0.2 and IoU threshold 0.5.

Setting	LR	Precision	Recall
all_lr1e-5	1×10^{-5}	0.504178	0.754167
all_lr5e-5	5×10^{-5}	0.207858	0.683333
backbone_lr1e-5	1×10^{-5}	0.446281	0.675000
transformer_lr1e-5	1×10^{-5}	0.532544	0.750000
head_lr1e-5	1×10^{-5}	0.207776	0.712500

Interpretation of ablations

The transformer-only setting performs best overall, achieving the highest precision while maintaining strong recall. This suggests that, for pixel-difference inputs, learning improved global correspondence and relational reasoning is more important than adapting low-level convolutional filters. The all-parameters setting at 1×10^{-5} is a close second and achieves the highest recall; however, its precision is slightly lower, indicating that full fine-tuning may introduce more false positives under limited training.

Backbone-only performs worse than transformer-only and full fine-tuning, supporting the idea that motion cues in pixel-diff images are not primarily captured by retuning local texture filters. Head-only achieves decent recall but low precision, consistent with updating only the last layers without improving the underlying representation. Increasing the learning rate to 5×10^{-5} reduces precision substantially while keeping recall moderate, suggesting less stable optimization when more parameters are updated aggressively.

These ablation runs are trained for only 100 epochs and the dataset is limited, so absolute metrics can be noisy. Nevertheless, the comparison is still informative: transformer-only and low-learning-rate full fine-tuning are the most effective choices under this setup.

Qualitative analysis

Visual inspection confirms that pixel differences highlight motion effectively. The model reliably detects moved vehicles and pedestrians, even when frame changes are subtle. Errors typically arise from shadows, lighting variation, or diff noise. Transformer-only runs generally produce cleaner boxes and fewer spurious detections compared to head-only and high-learning-rate full fine-tuning.

Design decisions

- Pixel-wise differences provide a simple yet effective representation of motion.
- Using `DetrImageProcessor` ensures consistent resizing and normalization.
- Training directly from matched text annotations reduces overhead.
- A lower score threshold is appropriate for small datasets, where confidence calibration differs from large-scale pretraining.
- Ablations help isolate which DETR components contribute most to motion understanding.

Reproducibility

Experiments can be reproduced via:

1. `python data_ground_truth_labeller.py`
2. `python train.py --strategy all`
3. `python eval_detr_moved.py --ckpt <path>`
4. `python run_ablations.py`

   All outputs are saved under `outputs/checkpoints/`, `outputs/eval/`, and `outputs/ablation_metrics/`.

Conclusion

The Option 2 approach demonstrates that DETR can detect moved objects using only pixel-wise frame differences. Despite limited data and short ablation runs, the ablation study indicates that fine-tuning the transformer layers is the most effective strategy, with low-learning-rate full fine-tuning being a strong alternative. Pixel-diff DETR provides interpretable visual results and reliably captures motion-related patterns, even when precise localization remains challenging.

Sample outputs

Figure 1: *

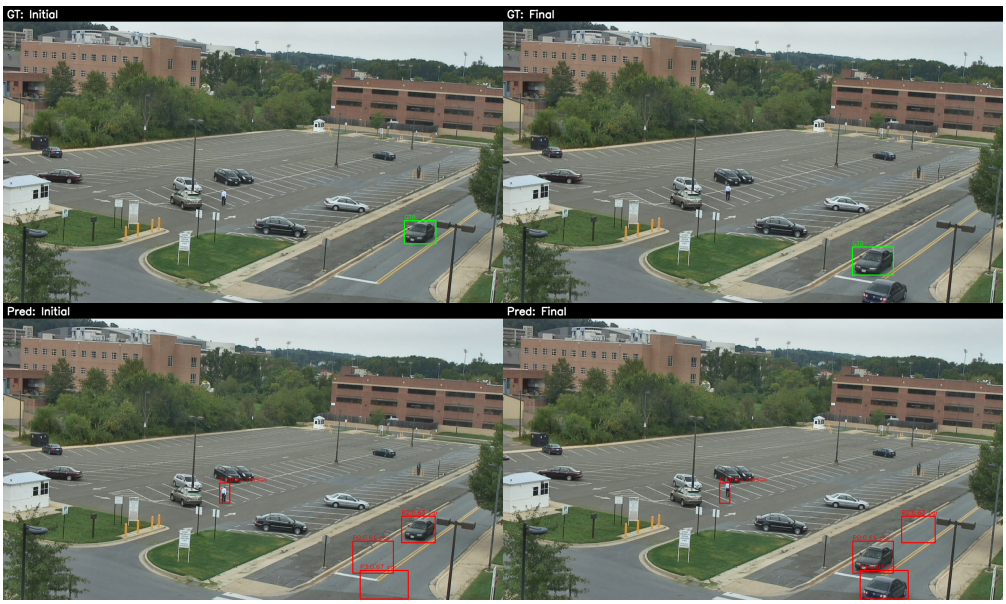

Figure 2: *

Figure 3: *

Figure 4: *

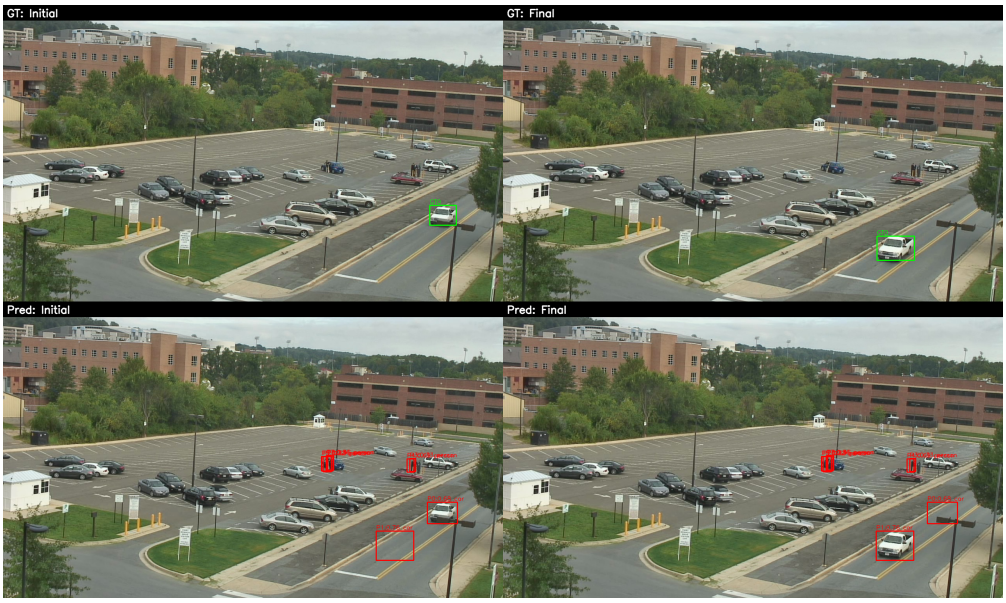

Figure 5: *

Figure 6: *