HW3: DETR Object Matching with Movement in Surveillance Footage
JupyterHub access: **https://csciga-3033-spring.rcnyu.org**

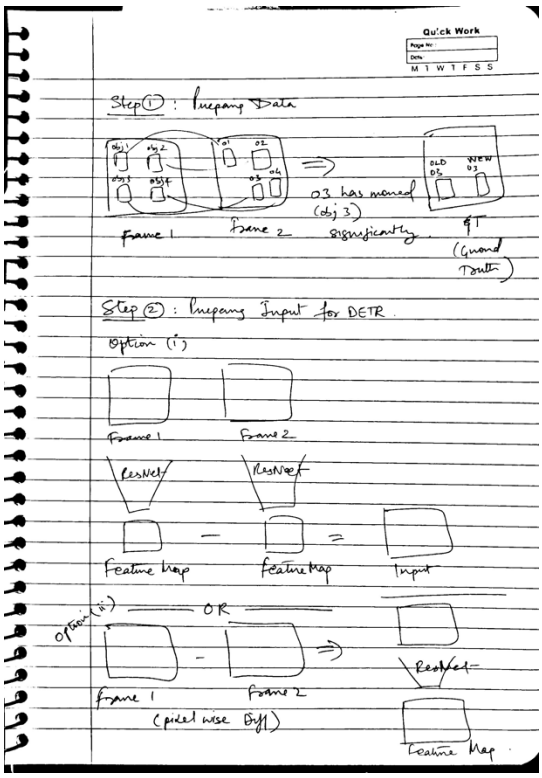
Input: ~/shared/data/cv_data_hw2. It contains pairs and their respective annotation txt files from surveillance camera footage. These frames contain objects labelled: Unknown=0, person=1, car=2, other vehicle=3, other object=4, bike=5. You can read more about the VIRAT dataset (source of these frames and annotations) here:

https://data.kitware.com/#item/56f578dc8d777f753209bdef

I encourage you to read about this dataset just to get an understanding of how you would parse the annotations for respective pairs.

Goal:
This assignment is divided into 2 steps: 1) Preparing the data, 2) Training the transformer

1) Preparing data:
    a) Create ground truth (GT)- (this step is already done for you): I used feature similarity and centroid distances for cost matrix calculation and used it with the Hungarian method of matching objects in 2 frames. After that, I only kept the objects that had significantly moved i.e., with IOU=0. I have given the code for GT generation in the shared folder under ~/shared/GT.pynb, please feel free to tweak the code as per your needs and generate a new set of GTs for your model. The GT will be the annotation files given in ~/shared/hw3/matched_annotations . I have also provided some sample visualizations in ~/shared/hw3/visual_matches with 1 frame per pair, having both the old and new bounding boxes for each object that moved significantly. The visuals are just for your convenience. Note: The object IDs from the original annotation .txt files are not all correct, so for the annotation files for these matched GTs, I have replaced the object IDs with new object IDs (different from those in the actual Kitware dataset). Each annotation file will have 2 rows per object. The first row represents the old coordinates of the object, and the second row represents the new ones. The format of each row is: <object_id> <x> <y> <w> <h> <object_type>. Here, x, y, w, and h are the bounding box coordinates. Please look at the code for the GT to understand more.



b) ***TO-DO*** Next step is to extract feature maps that will be the input to the transformer. You have 2 options for this step (do this for each pair) :

i) Pass both frames in the pair to a ResNet individually and extract a more compact/compressed feature layer out of it. Then take the difference of both the resultant feature layers. This "diff" will be your input for the DETR model in the next phase of the assignment.

- ***Individually** pass both frames through a **pre-trained ResNet**.*
- *Extract **intermediate feature representations** (e.g., from a mid-layer like layer3).*
- *Compute the **difference between these feature maps**.*
- ***Why we do it:***
  This isolates changes in semantic features across frames, helping the model focus on movements or changes in object appearance — especially important in surveillance.
- ***How to do it:***
  Use a pretrained resnet50, extract features using a forward hook or feature extractor, compute feature_diff = abs(f1 - f2). Normalize and resize as needed. You may need to flatten the output.

ii) Alternatively, you can first take the diff of the 2 frames of then use a ResNet on it to get a compact feature layer.

- *Take the **pixel-wise difference** between the two input frames.*
- *Pass the **difference image** through a ResNet to extract a compressed feature map.*
- ***Why we do it:***
  This approach emphasizes direct visual changes and delegates feature abstraction to the network after the change is computed. It's simpler and can capture subtle shifts in pixel data.
- ***How to do it:***
  Use diff_img = abs(img1 - img2) and pass it through resnet50 up to a mid-layer. You may need to normalize and stack the single-channel diff to match ResNet's expected input.

To extract feature layers from a pretrained **ResNet** (e.g., resnet50) in PyTorch, you can either create your own ResNet model and end it at an intermediate layer, say layer3, or you can use a forward hook, which is more flexible but more complex. It lets you tap into any layer dynamically.

2) Training the model:
   - Use an **80-20 train-test split** on the annotated and matched dataset.
   - Fine-tune a **pre-trained DETR** model using the feature_diff (from either option) as input, and matched object pairs (that moved significantly) as output.
   - Please be aware of the memory-intensive nature of DETRs. I would advise you to choose a lightweight model even if you have to make a trade-off elsewhere. Do mention any tradeoffs made in the report.
   - Input size must be consistent, so resize as needed.
   - Evaluation should include precision and recall (and all other relevant metrics) on detected matches.

Submission Checklist:

Report (~1–2 pages):

- Describe your approach, pipeline, which DETR was used, and the rationale behind the chosen approach (option i or ii).
- Include performance metrics, visual output/screenshots wherever helpful.
- Highlight any assumptions or workarounds you implemented.

Code:

- Turn in a complete Jupyter notebook using Nbgrader.
- Skip validation (VERY IMPORTANT, AS THIS HAS BEEN GIVING ERRORS).

As usual, please email me if you need any assistance.