Introduction

In this assignment we have implemented the paper "Unsupervised Learning of Visual Representations using Videos" by wang et. al. on the given dataset.

Method Overview

In this section we have briefly described the method adopted in the paper for learning features using videos. In next section we will describe the modifications we have made and our approach. The method involves tracking one object per video for a small duration (close to 30 frames). To find objects for tracking, first optical flow is computed for SURF points of the first frame. Then a sliding window is used to find a box having maximum number of points with a shift of $>= 0.5$ pixels between two frames. This box is then tracked using KCF tracker. A triplet is obtained by taking object crop at first and last frame of tracked object and another random crop of some other object. Such triplets are used to build a large scale database. A Siamese Network is then trained with Triplet Loss function. After training, the network learns some good features for objects.

Implementation Details (Our Approach)

Since we have a small dataset (only five videos), instead of tracking one object per video, we tracked multiple objects per video and instead of taking just two crops, one at beginning and one at end, we haven taken multiple crop of objects at an interval of five frames to increase the dataset size. We have used videos from four cameras for training and last video for evaluation.

Finding Objects to Track

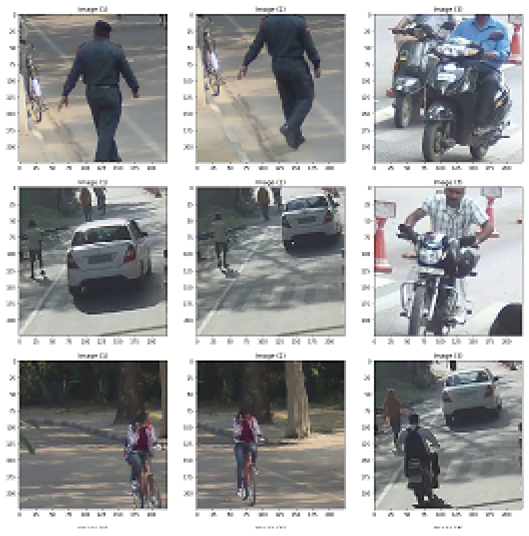
To find multiple objects for tracking in a video, we used sliding windows to find all the boxes having atleast 20 SURF points with optical flow between 10 to 15 pixels. We then apply NMS to get different windows with no intersection. We do this step for every other 5^th frame (skipping 4 frames in between). So this also takes into account and tracks new objects that appear in video after first frame.

Tracking Objects

We have implemented a naive algorithm for tracking. After locating all object in current frame (as described above) , we find correspondence between boxes of current frame and last frame processed. For a given box in last frame processed, we find one box in current frame such that it has maximum corresponding/shifted SURF points of the box in last frame and atleast 20 such points. Such pairs of boxes are considered same object. Any box in current frame having no corresponding box in previous frame is considered new object. All the bounding boxes in every 5^{th} frame are cropped and stored.

Dataset

First we delete noisy objects i.e having only one crop. After that to get a triplet, we take two random objects. For the first object we chose two random crops in the range [1,number of crops of that object]. For second object again we chose any crop of it randomly. This gives us a triplet (Below are few examples). We have extracted objects from four cameras.We got a total of 133 objects and total of 832 object crops. We have used crops of 100 objects for training and rest of the crops for validation.

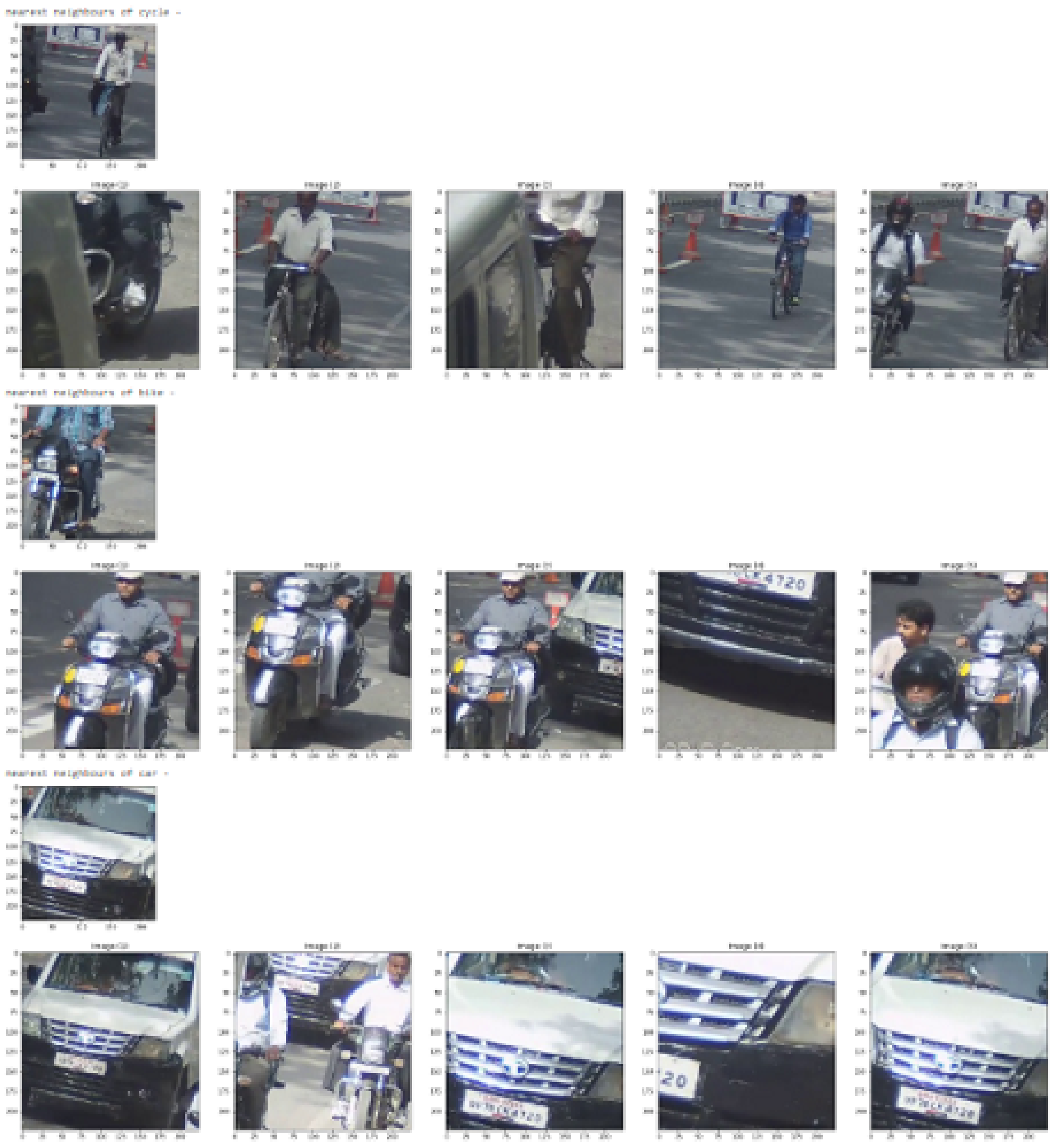


Training

We have used resnet-18 with random initialization for training. We remove last layer with 1000 outputs. We take a batch of size 32 for training. We have used the same triplet loss given in the paper as our loss function. Training does not converge properly although we do get good values for loss function and save models having low loss on validation set.

Results

To find how effective the trained model is, we found image vectors for object crops obtained from video of camera 5 (which was not used for training or validation) and used them to find nearest neighbour on object crops (only positive results included). Below are the results -



```
nearest neighbours of cycle -
nearest neighbours of bike -
nearest neighbours of car -
```

References:

Unsupervised Learning of Visual Representations using Videos Xiaolong Wang, Abhinav Gupta

https://arxiv.org/pdf/1505.00687//

Pytorch for training models

pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

Optical Flow from OpenCV -

https://docs.opencv.org/3.4/d7/d8b/tutorial_py_lucas_kanade.html