CAM^2 Report

Report on the Trigger Mechanism for reducing transmission of unnecessary data from nodes to the server on a camera network

Amol Moses Jha

July 18, 2018

Rationale behind needing a trigger

When typically transmitting frames over to the server in our designed pipeline, the bottleneck was the neural network which needs to identify features, namely those of pedestrians. While the cameras where transmitting frames at around 30fps, our preliminary testing showed that we could not process frames on our neural network at even half that speed. This would lead to lost frames invariably on multiple ends, namely the server's kernel which would end up buffering incoming data only to an extent, or lost TCP packets, leading to buffering and lost performance on the clients' end. Therfore, an optimization to reduce this invariable loss in performance was needed.An optimization we came up with, which would lead to more robust performance on our feature detection NN would be to simply tramsit those frames which had any associated motion with them. Therefore, we needed a fairly lightweight mechanism to detect frames with motion, since the nodes tend to be low-powered embedded machines with not a lot of hardware capapbility. The newer client, then in theory, detect motion (preferably that of pedestrians), and transmit only those frames which contained meaningful data. This would give our NN more time to process each frame, and also feed it with meaningful data from which to extract features from. Therefore, the nodes would work under a "trigger", and therefore the naming for the same.

HOG + SVM Pedestrian Detector

The first idea of such a filter was to use a Histogram of Gradients(HOG) trained to identify pedestrians in images, and classify such HOGs by using a Support Vector Machine (SVM) on whether they are pedestrians or not. Implementing the same gave us a great accuracy on frames, wherein we could handily identify almost every frame where a pedestrian was present, cutting down by around 50% on our transmisssion, but the performance was very sub-optimal on the nodes, namely the raspberry pi(s) which we used, leading to a transmission of as low as 0.6fps. This would not do, and therefore, we switched to a better a performant method for an acceptable loss in accuracy.

Background Subtraction

The second idea we ended up having was to use background subtraction to detect motion between subsequent frames. A major drawback to this method

would be that *ALL* motion would be detected as opposed to only pedestrians, but as long as we used tight camera angles with controlled lighting, we could eliminate superflous data from our images. Therefore, we decided to give this method a try. Implementation consisted of deleting the current frame with a model background frame, and dialting the frame using Gaussian filters, and then identifying objects different from the background, computing their contours and eliminating noise. Our initial testing seemed to gain great results, with around 30% frames being dropped from vanilla transmisson, at around 17fps. This is very acceptable performance, and we decided to stick with this method for the near future, when it can be subsequently improved upon.

Therefore, we have a working implementation for the clients to act as nodes, which are only "triggered" when motion is present in a video stream. The code is publicly available on our github account, https://github.com/pushyami/person_re-id.