Multi-Camera System Query Processing

ACM Reference Format:
. 2019. Multi-Camera System Query Processing. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Abundance data is being generated from camera systems nowadays. According to Lucid Motors, about six to twelve cameras are able to produce about 60 to 400 megabytes data. Facing large-scale data, engineers are no long able to manually analyze video data from the cameras. The advancements in deep learning (DL) allow analysts to run video analysis without manual interference. This benefit encourages the emergence of video analytics systems nowadays.

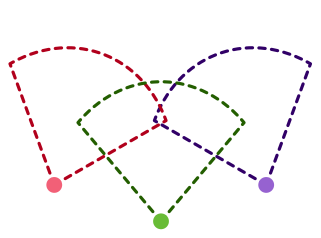
Challenges In current video analytics systems, engineers now face three major challenges.

Latency: DL task itself is very compute intensive. It requires powerful compute resource to quickly complete the DL model inference. However, there always exists mismatch between the availability of computation resource and amount of tasks to be completed. For example, a powerful GPU is able to support 150 frames per second (FPS) for image based DL video analytics system. A typical system can have about six cameras deployed for collecting data. Each camera is able to produce 30 frames per second as its video data. As a result, the powerful GPU is not able to complete the total 180 frames in one second. This dramatically increases the latency that users have to wait for task completion in order to get the prediction results.

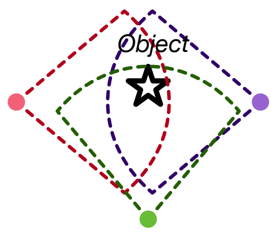
Bandwidth: The second challenge of the system is the networking bandwidth. Because data is usually generated from cameras or sensors, which are usually considered as edge devices. They are not powerful enough to perform the DL prediction fast. In order to run DL algorithm, edge devices have to send data over the network to remote cloud. Nevertheless, because of the amount of data the system sends over the network, the data transfer bandwidth becomes the bottleneck of the system.

Storage: There is still issue once the data arrives on the cloud. First of all, because the system can send gigabytes of data to the server, the data may not fit into the cloud server's memory. Moving data from disk to memory and moving the prediction results from memory back to disk causes overhead to the system. In addition, the system users also want to use data later. Retrieving data back from the disk becomes another issue.

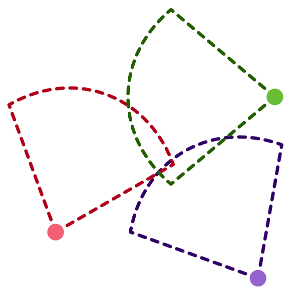
Opportunities: We observe that in a typical camera system, cameras are often deployed at spatially close locations. The image frames generated from those cameras tend to have overlapping data points. For example, cameras deployed on autonomous driving cars often face toward the front of the car. They have overlapped view of road vision in front of the car. Another example can be the surveillance camera system. In order to cover the full view of a public place, cameras are often deployed to have redundant vision perspectives.



(a) Autonomous Driving Cars.



(b) Swarn Robots.



(c) Surveillance Cameras.

Figure 1: Spatially Closed Location Camera System Examples.

Conference'17, July 2017, Washington, DC, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Furthermore, there have been some previous works conducted to optimize the performance of video analytics systems. Nevertheless, they all focus on filtering out the redundant data points in the temporal spectrum. For example, one of the work, BlazeIt, uses specialized model to filter the frame with low probability. Another work Focus uses k-means clustering algorihtm to aggregate similar frames along the time sequence. Both of these work do not consider the spatial property of the camera system. Hence, for this project, we want to explore the opportunities to optimize the spatially close cameras for video analytics system.

2 BACKGROUND

Re-Identification of entities is a core application when it comes to multi-camera systems. It refers to the process of matching a query (image of entity) to related images in the gallery (images in database).

An approach to solve this problem is to perform some form of hand-crafted feature extraction to the images and perform matching based on the distances between entities using these features using algorithms such as K-Nearest Neighbours.

More recently, this feature extraction process has been performed with deep learning methods. Deep neural networks can provide powerful representations which can better discriminate between images than hand-crafted features. There are two ways to learn these representations using different learning objectives.

2.1 Softmax Loss

Here we training a neural network to correctly classify an image into one of the unique classes present in our database. After training process, we discard the classification layer and utilize the previous layer as our feature representation. Despite the large number of classification outputs (>1000 unique classes) this approach performs well.

2.2 Triplet Loss

In this approach we construct training examples in form of triplets. Each triplet consists of one anchor image, positive image and a negative image. The anchor image and positive image come from the same class while the negative image comes from a different class. The network is trained to maximize the distance between the feature representations of the anchor/negative images and minimize the distance between the anchor/positive images. This directly aligns with our objective.

$$\mathcal{L}(A, P, N) = max(0, \|F(A) - F(P)\|^2 - \|F(A) - F(N)\|^2 + \alpha)$$

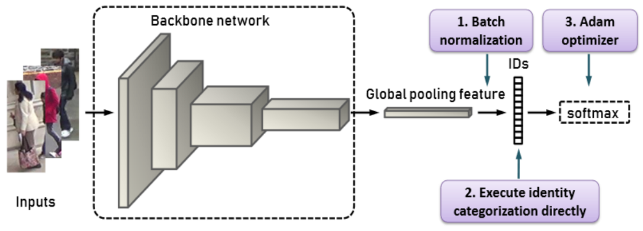


Figure 2: Softmax based feature extraction

3 DATASETS

3.1 Market-1501 Dataset

The market-1501 dataset consists of images of persons collected across 6 cameras in front of a campus supermarket. It consists of 1501 unique labelled identities across 32,000 images. To further evaluate the robustness of re-identification models, it also provides 500,000 "distractor" images.

It also comes annotated with metadata about the person in the image such as shirt color, if the person is holding a bag, wearing a hat, approximate age etc which can be incorporated to improve model accuracy.

3.2 NVIDIA AI City Challenge Dataset

This It is a dataset consists of images of vehicles collected across 40 cameras. It consists of about 37,000 training images. These images are provided in the form of tracks where each track consists of images of the same vehicles across different cameras/timestamps. The goal is to perform re-identification given an image of a vehicle. Each image also has the associated camera ID that captured it.

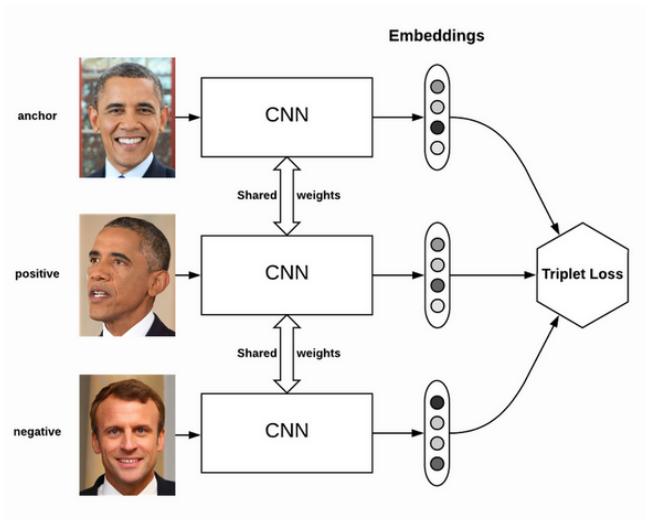


Figure 3: Triplet based model

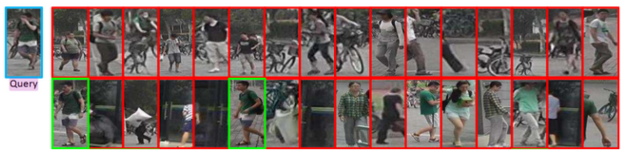


Figure 4: Market-1501 Dataset

Various approaches to this solving this problem incorporate metadata about the image such as vehicle type, 3D - orientation etc to further boost the accuracy.

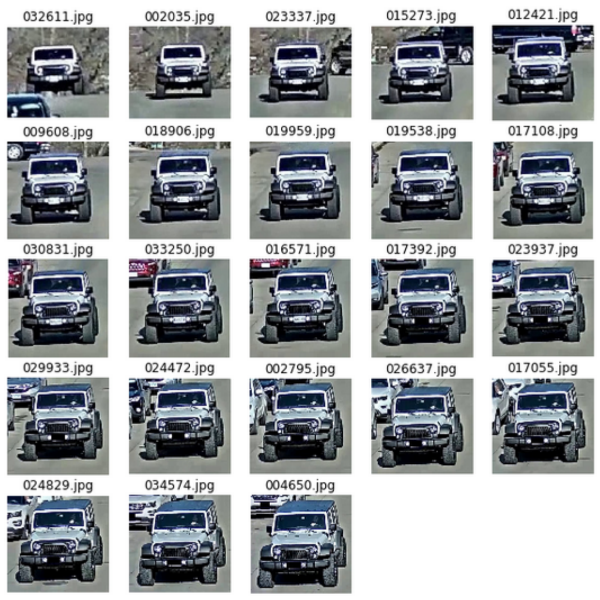


Figure 5: AI City Challenge Dataset - Example Track

4 CHECKPOINT 1

4.1 Person Re-identification

We setup person re-identification (REID) pipeline as our baseline along with Market-1501 dataset. The task of person REID is to detect the person appearing in the query, who also have appeared in the gallery. The query is a new image, but the gallery contains multiple base images the task uses to do REID. For the person REID task, we use ResNet-50 as our baseline model for feature extractions. The pipeline of baseline person REID task is consisted of three steps. 1) Extract features of gallery by using ResNet-50 model. 2) Extract features of query with the same ResNet-50 model. 3) Label the query with the label of the closest feature in gallery. It is basically a un-supervised learning task with the favor of deep neural networks (DNNs).

We conducted timing analysis for those three steps in the baseline pipeline. The first, second and third steps take 48, 8 and 12 seconds respectively. We make two observations based on our timing analysis.

Observation 1: The gallery has multiple classes along with multiple images. During the phase of gallery feature extraction, the pipeline needs to execute DNN inferences on every image. Due to the size of gallery, the feature extraction becomes very slow. In addition, there is a unique feature associated with one image, the storage cost of that approach is also high. We think opportunities exist in this pipeline to optimize both computation and storage. Since gallery has less number of classes compared to number of images, finding a feature associated with a class is faster and more storage friendly. Therefore, in this case, we want to find the most representative feature of a class, thereby avoiding iterating through all images in a class.

Observation 2: During the evaluation phase, each query is compared against all features found in the gallery. This approach does not scale as the gallery size grows. In other words, the computation complexity grows as the gallery size grows. To optimize this, we want to constrain the gallery dataset size, so that it does not store highly similar features in duplicate. In addition, we want to explore using lightweight computation to filter out unrelated queries to speedup the computation.

4.2 Vehicle Re-identification

For this dataset, we train a triplet loss model which consists of blocks of convolution. Each block consists of A 3x3 Convolution layer, followed by Batch Normalization, followed by relu activation into a 2x2 maxpool layer. Our embedding model consists of 5 such blocks followed by an adaptive average pooling layer. This allows us to have uniform feature representation, even if the image dimensions vary across images.

We train this model on the AI City Challenge dataset with 66% / 33% split between training and testing for 10 epochs with Adam optimizer with learning rate of 0.002. We are yet to evaluate the model on the testing set.

To compute the features for the entire gallery, it takes about 2 minutes and 30 seconds for roughly 37,000 images.

5 NEXT STEP

❶: Since we have identified the bottleneck existing in the multi-camera pipeline for person re-identification task, the next step is to try some optimization methods. One approach we have thought about is to construct only one feature per class. By doing so, the pipeline saves both computation and storage resources. The only concern is one feature is not good enough to represent all potential features for one class, so that it hurts the accuracy performance.

❷: The second task remaining is the baseline for Nvidia AI Challegen dataset. Though we have setup the dataset, we haven't constructed any baseline pipeline for tracking or re-identification task. Hence, as for next step, we want to finish the baseline for that dataset, and identify the performance bottleneck in the system as well.

6 GOAL

75% goal: The 75% goal of this project is to successfully setup the baseline for our project. After constructing the baseline, we want to identify the system issues existing in the baseline approaches. We think we have achieved our 75% goal as for now. We have two datasets ready for now. One is Market-1501 dataset and the other is Nvidia AI City Challenge dataset. Both of these datasets are related to multi-camera tracking or re-identification. In addition to that, we have also conducted analysis for re-identification task on Market-1501 dataset, in which we analyzed the bottleneck of the system and the opportunities to optimize the pipeline.

100% goal: The 100% goal of this project is to propose an optimization solution based on our identified issue. The solution can be one optimization of three major challenges that we mention. As for now, since we have already found the bottleneck of multi-camera person re-identification system, we want to first focus on gallery computation optimization.

125% goal: The 125% goal is that after we implement our proposed solution, we successfully evaluate it on our customized dataset. For now, the optimal goal of our project is to evaluate our proposed optimization technique for gallery dataset and check its resulted performance gain.

7 CHECKPOINT 2

7.1 Person Re-identification

We have tried two approaches for person re-identification tasks. The first approach is to use specialized model to speedup the computation on feature extractions. The second approach is to use cluster algorithm to reduce the gallery feature size and speedup the evaluation process.

Specialized Model: We designed a specialized deep neural network models, such that the feature inference is able to stop in the middle. However, the resulted feature will be less accurate compared to the original models. Hence, there is a trade-off between accuracy and performance. The advantages of this approach are that we are able to improve both gallery and query feature extractions processes because we reduce the computation overhead. Nevertheless, the evaluation process cannot be optimized because the dimensions of distance metrics are not changed.

Clustering: The second idea is that we want to actually reduce the gallery dataset size. The reason is because in gallery dataset, a single class has many images. In the pipeline, the feature extraction needs to be applied to all images. Therefore, instead of using all images, use single image to represent a class can speedup the pipeline. We use a simple clustering algorithm as proof of our point. The results demonstrate that the pipeline gains speedup on gallery feature extraction step and the evaluation step. The below table demonstrates the results for those two approaches.

Table 1: Latency of our designed methods compared to baseline.

Approach	Gallery	Query	Evaluation	mAP
Baseline	48.20 s	8.29 s	12.40 s	0.72
DNN 1	42.41 s	7.30 s	12.40 s	0.71
DNN 2	37.63 s	5.94 s	12.40 s	0.57
DNN 3	33.89 s	5.14 s	12.40 s	0.33
Clustering	10.57 s	8.45 s	2.40 s	0.00

7.2 Vehicle Re-identification

We have performed an evaluation on the Nvidia AI city challenge dataset use the model proposed in section 4.2. We obtained a mAP score of 0.1868. For reference the rank 60 team in the leaderboard had an mAP score of 0.25 [3]. So we are able to obtain reasonable results, despite not using any of the post processing / re-ranking steps taken by most of the teams on the leaderboard.

The bottleneck for performing these computations are threefold Computing the embeddings for each image, computing pairwise distances and sorting the distances. For our baseline we notice that it takes about 12 seconds to compute the full nearest neighbor search. Which is prohibitive for real-time inference. The embeddings themselves can be computed offline and stored.

To overcome this bottleneck we use approximate nearest neighbor (ANN) search. It is designed to allow trade-offs between speed and accuracy for nearest neighbor search. For minimal loss in accuracy, we can get orders of magnitude in speedups for inference. It achieves this speed-up through multiple key optimizations mainly being Product quantization and Look-up tables. Facebook provides a library: Faiss [5] which allows an easy interface for ANN with numpy tensors. It requires us to compute the embeddings one time and Faiss will create indices required for ANN search.

Another approach to nearest neighbors is to use K-D trees which have been designed to tackle nearest neighbor searches. Spotify's Annoy [8] provides such an implementation for ANN. This also requires us to index our embeddings into it, but each embedding must be added individually this

Table 2: Baseline vs Approximate NN

	Baseline	Faiss	Annoy
Embeddings	62s	62s	62s
Indexing	0	0.001s	1.41s
Pairwise distances	2.24s	0	0
Sorting	9.63s	0	0
Nearest neighbors	0	1.17s	3.87s
mAP	0.1868	0.1868	0.1868
Finetuned mAP	0.5802	0.5801	0.6052

causing the index building step to be slower compared to Faiss. The speed-up provided during inference is also slower than Faiss.

We notice that the speed-ups compared to the baseline are very good (about 10x speedup using Faiss and 2x speedup from Annoy) with almost not loss in mAP scores. This is a promising avenue for improving performance in re-identification tasks in real-time settings.

8 CHECKPOINT 2: NEXT STEPS

8.1 Person Re-identification

We will try different clustering approach to increase the accuracy. In addition to that, we expect to expand this work to a more complicated pipeline.

8.2 AI City Challenge

We still have room for improving the mAP scores. Most of the existing approaches leverage existing datasets for vehicle re-identification and use transfer learning models to improve results on the AI city challenge. We can utilize some of the existing datasets [2] and train the model on them and later fine-tune it for the AI city challenge.

9 FINAL UPDATE

9.1 Bag of Tricks

After checkpoint 2, we realize that the person re-identification baseline does not generalize well to other tasks. Therefore, we decided to switch to use the baseline from the bag of tricks paper. It provides a more robust pipeline and also better accuracy.

9.2 Early Inference

After changing the pipeline, we re-apply the early inferencing technique on the pipeline and compare the results with ResNet18 and ResNet50. We discover that the deeper early inferencing point supports similar to ResNet50 accuracy but much less latency, so this is a feasible optimization for the pipeline. However, other two shallow early inferencing points provide even worse than ResNet18 accuracy but with similar latency. Hence, for future work, we need to optimize those two early inferencing points accuracy.

9.3 Specialized Model Per Camera

In addition to the idea of early inferencing, we also prototype lightweight specialized model for different camera perspectives. Our idea is a single model is not robust enough to generalize to different perspectives of cameras. We expect to reach higher accuracy but lower latency with specialized lightweight models. However, the accuracy of each model is relatively low for now, so we want to optimize that as well in the future.

9.4 Approximate Nearest Neighbours

After our initial attempt at tackling the NVIDIA AI City Challenge, we originally obtained an mAP Score of 0.19. By adding a learning rate scheduling using PyTorch's ReduceLROnPlateau scheduler on the mAP score, we obtain a massive improvement of 0.58 mAP 2. We also were able to train for longer amount of epochs without getting stuck in a local minima after setting up the learning rate schedule from just 5 epochs to 30 epochs.

With the ANN setup, we notice that the performance starts deviating from the exact nearest neighbours with our finetuned embeddings. Faiss remains close to our exact nearest neighbours results but Annoy ends up performing better. This could be better generalization occurring due to the approximations occuring from the KD Trees used in Annoy.

9.5 Knowledge Distillation

Another avenue explored when it comes to improving inference speed is to just simply use smaller models. Models such as a squeezenet and mobilenet have been proposed to reduce the number of model parameters while retaining performance of larger models such as resnet.

But smaller models have been shown to not perform as well in practice. An approach to combat this situation is to use knowledge distillation. It is a training process in which we try to extract the knowledge learned by a larger model into a smaller model. First we start by training the larger model to desired accuracy. We then train a smaller model with the loss function being a combination of the cross entropy loss with respect to ground truth labels and the cross entropy loss with the outputs of the larger model with the same input. An analogy for knowledge distillation can be thought of as learning from a professor/teacher instead of learning purely on the textbook.

For our experimental setup we chose Market-1501 dataset with resnet-50 as the teacher model, squeezenet and mobilenet as student models. We train all the models for 30 epochs with ADAM optimizer with a learning rate of 0.003.

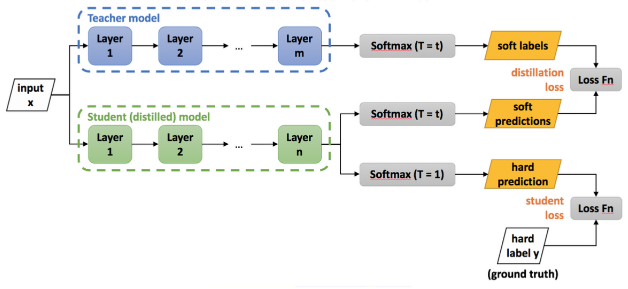


Figure 6: Knowledge Distillation Architecture

Table 3: Model Comparison on Market-1501

	Resnet-50	Squeezenet	Mobilenet
Parameters	25 Million	1.1 Million	5.6 Million
Training Time	1 hour 40 mins	13 mins	19 mins
Size on Disk	98 MB	4.3 MB	22.2 MB
Inference	19 sec	7 sec	17 sec
mAP	67.1%	29.6%	51.9%
Rank-1 Accuracy	83.8%	54.6%	74.0%

Table 4: Knowledge Distillation on Market-1501

	Resnet-50	Squeezenet	Mobilenet
Parameters	25 Million	1.1 Million	5.6 Million
Training Time	1 hour 40 mins	17 mins	25 mins
Size on Disk	98 MB	4.3 MB	22.2 MB
Inference	19 sec	7 sec	17 sec
mAP	67.1%	47.1%	60.7%
Rank-1 Accuracy	83.8%	70.0%	79.4%

For comparison between regular training and knowledge distillation see table 3 and 4.

We are able to obtain an improvement in performance for squeezenet of about 15.6% and about 5.4% in mobilenet. These improvements are obtained at the cost of increased training time of the models themselves as well as the training of a teacher model. We still retain the massive improvements in terms of model size as well as inference time as shown.

10 CONCLUSION

We explored multiple approaches to reducing inference time for re-identification tasks. We attempted the NVIDIA AI City Challenge with satisfactory results (existing results put us in top 25 in leaderboard) and explore additional techniques to improve inference time on person re-identification datasets such as Market-1501.

Avenues of future work include

- Improving accuracy of early inferencing and specialized DNNs
- Exploring different models for knowledge distillation
- Integrate feature extraction approaches to Approximate nearest neighbours

Additional approaches to explore include model pruning and quantization to improve inference speed and size of models.

REFERENCES

[1] Jon Almazan et al. "Re-id done right: towards good practices for person re-identification". In: *arXiv preprint arXiv:1801.05339* (2018).

[2] *Awesome vehicle re-identification Datasets*. https://github.com/knwng/awesome-vehicle-re-identification.

[3] Ming-Ching Chang et al. "AI City Challenge 2019–City-scale video analytics for smart transportation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 99–108.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[5] Jeff Johnson, Matthijs Douze, and Hervé Jégou. "Billion-scale similarity search with GPUs". In: *arXiv preprint arXiv:1702.08734* (2017).

[6] Yutian Lin et al. "Improving Person Re-identification by Attribute and Identity Learning". In: *Pattern Recognition* (2019). DOI: https://doi.org/10.1016/j.patcog.2019.06.006.

[7] Milind Naphade et al. "The 2019 AI City Challenge". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 452–460.

[8] *Spotify ANNOY library*. https://github.com/spotify/annoy.

[9] Liang Zheng, Yi Yang, and Alexander G Hauptmann. "Person re-identification: Past, present and future". In: *arXiv preprint arXiv:1610.02984* (2016).

[10] Liang Zheng et al. "Scalable person re-identification: A benchmark". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1116–1124.