Query Time Optimized Video Inference System

Group 6
Mingren Shen, Shuoxuan Dong, Xiuyuan He

December 17, 2018

1 Introduction

Video inference system has received increasing attention due to its potential in various real-world applications. By querying a video data set, we can answer questions like which frames contain objects of certain classes, such as certain types of cars, pedestrians, and bicycles. Answering these questions are very useful in different areas like traffic control, video surveillance, etc [9].

Some may argue that video inference can be done using the same approach for image recognition, however, many existing image recognition methods are too expensive to be directly applied to video inference tasks. Even with state-of-the-art Convolutional Neural Networks (CNN) such as the Faster R-CNN [19] running on powerful GPUs the speed is still relatively low and computational cost is high. According to literature [9], Faster R-CNN runs 30-80 fps on a $ 4000 GPU and it can cost up to $ 250 to query a month-long traffic camera video on cloud. This may seem to be affordable for videos from single camera, but imagine if you have videos from thousands of cameras need to be queried, the cost is will be prohibitively high.

One of the cutting-edge video inference system that provides low cost and low latency is called Focus[9]. It uses an unique architecture that divide the video query problem into two phases: ingesting and querying. A cheap CNN is used in the ingesting phase to quickly generate Top-K indexes which contains K possible classes of the image. An expensive CNN is then used to generate a final and more accurate prediction during query time.

For this project, we build our system based on the Focus architecture as shown in Fig 1. We propose to reuse the intermediate results of the shared layers(red trapezoid in Fig 1) between the cheap and expensive CNN to reduce inference time. Sharing convolution layers between two neural networks has already been used in Faster R-CNN model to speed up object detection where two networks, region proposal neutral network and Fast R-CNN [4], share the beginning several convolution layers [18, 19].

In this report, We show how we use two CNNs, ResNet-50 and ResNet-152, to represent the architecture of Focus [9] and we evaluate the accuracy and latency of it as our baseline. We then develop two different layer-sharing strategies. In one case the two networks share the first 4 blocks with each other, and in another case they share 7 blocks. The results suggest that depends on how many layers the two CNNs share, there is a 10% to 20% improvement for inference latency, as shown in Table 1.

2 Background

2.1 Video Inference System

Video inference system is an important area of computer vision research and application. Video inference systems often try to answer "after-the-fact" queries such as : "In all the recent 24 hours surveillance videos from 1st ave to 5th ave, return all the frames containing a truck." Such types of questions are frequently asked by local police and other agencies [9]. NoScope and Focus are two state-of-the-art systems designed to answer these questions. This report is mainly based on Focus.

As mentioned before, Focus has two phases: ingesting and querying. During ingesting time, Focus uses a cheap CNN model, ResNet-18 [8], to pick the Top-K Indexes for each frame. For example, if the original image is a truck, the index can be a list of classes like [boat, car, truck, bus]. The first prediction on that list may not be the most accurate one. However, the top-K list almost always contain the class "truck". During querying time, a user can query for a certain class like "truck". With the Top-K Indexes from ingesting time, the system identifies the frames that are possibly interesting to the query. An expensive CNN model, ResNet-152, is then used to do predictions only on the selected frame set, whose index contains the specified class user queried. Such an expensive model can cost 8$\times$ more compared to its cheaper variant ResNet-18, but can yield more accurate results.

2.2 Convolutional Neural Networks

Convolutional Neural Networks(CNNs) are a special class of neural network that use many layers of convo-

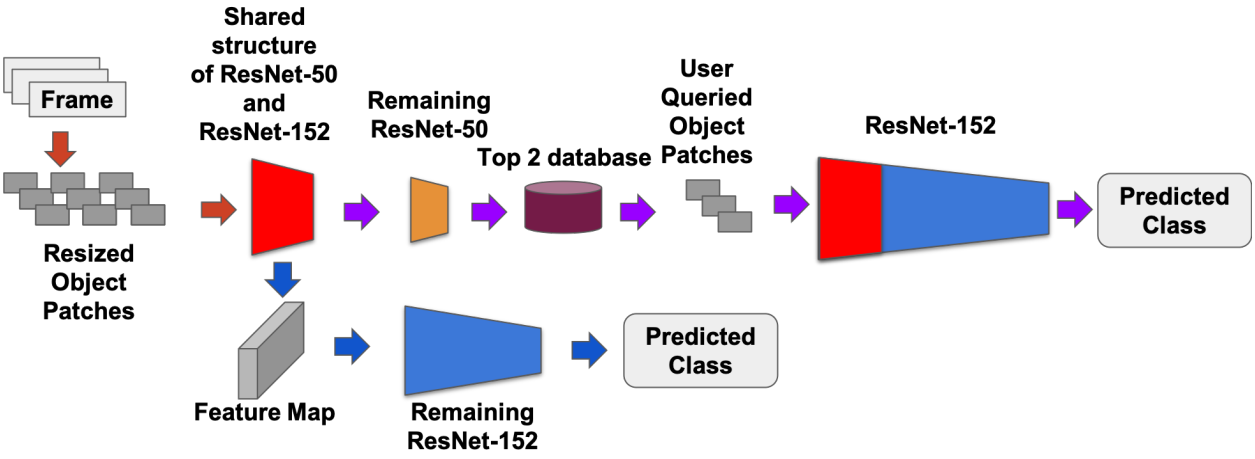

Figure 1: Project System Design

lution filters to automatically extract hierarchical helpful features for the final task like classification. For example, for image classification task, beginning layers of CNNs try to learn lower-level features such as dots, lines and edges. Layers in the middle tend to build object parts based on those lower-level features and layers near the output are learning the most semantic and high-level object related information. Finally, a score is computed for each image class as output [13].

Since Krizhevsky et al. won ImageNet competition in 2012 [12], CNNs are the most popular methods for image related tasks [13]. In 2015, a new deep residual learning method called ResNet [8] even surpassed human performance on image classification tasks of ImageNet. ResNet contains many stacked basic units called blocks to solve the degradation problem encountered in training deep neural network [8]. The central idea is that by attaching an identity mapping with skip connections, even very deep layers in ResNet can get the identical input from previous layer and the newly added layer should not perform worse than the shallower model [8].

CNNs are not only useful for image related tasks, they can also be used for videos [9, 17, 11] inference and tracking, recommender systems [21], natural language processing [3], and physics [1] or chemistry [20, 7] related problems .

2.3 Traffic Video Characteristic

One important characteristic of surveillance or traffic camera videos is that large portion of the frames from the video is duplicate and useless, for example, if the scene is nearly static. So processing each frame is less efficient. To address this problem, NoScope [11] uses a difference detector that measures the Mean Square Error (MSE) between images to determine whether the content of the frame has changed. Compared with NoScope, Focus uses a different approach to solve the duplicate frame problem. It clusters the frames based on the output of the penultimate (i.e., previous-to-last) layer [9].

We create our data-set based on the traffic camera video frames from Urban Tracker Dataset [10]. The patches containing objects are extracted from the original images using background subtraction. The patches are labeled with 3 classes including pedestrians, cars and some subtracted background. The models are trained with manually labeled image patches.

2.4 Feature Map

Feature Maps or Feature Vectors is the intermediate output from certain CNN layers. They are abstract visual features and high-level representation of the original images. Feature maps can gradually capture higher level features that consist of lower level features. Caching feature maps in memory or save them to disk can save inference time for further uses. Xu et al. developed an effective CNN cache system to store these feature maps and reuse them at fine spatial granularity to get high performance when doing inference in continuous mobile vision [22]. Indeed, the data volume of feature maps is large, so there might be a trade off between efficiency(inference time) and storage (memory) to reuse feature maps.

3 Design

3.1 Dataset Generation

A video from the Urban Tracker Dataset [10] is used to generate object patches. To convert video into object patches, we use background subtraction to detect the moving objects (cars, pedestrians, etc.). The detector we use is BackgroundSubtractorGMG in OpenCV, which is an algorithm that combines per-pixel Bayesian segmentation with statistical background image estimation [6]. We generate 2275 patches with 3 classes: cars, pedestrians and background images (noise generated by background sub-tractor), as shown in Fig 2. All patches are re-scaled to 224x224 pixels.

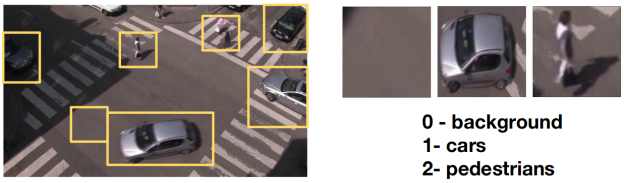

Figure 2: Background Subtraction and Generated Patches

3.2 Building Baseline

We jointly use two networks to mimic the two CNNs architecture of Focus [9]. Like Focus, we use a cheap CNN for ingest phase, during which top-k indexes are generated. During query time, the user queries frames containing a specified class. An expensive CNN is used to classify the images only if the top-k index of the image contains the desired object class.

3.3 Feature Map Reuse

Focus uses ResNet-18 and 152 as the cheap and expensive CNN, whose structure is shown in Fig 3. From that figure, we can find that the two CNNs have some same structure layers since the input layer. Considering there are two different residual blocks in ResNet [8], the basic block and the bottleneck block, we only reuse shared layers if the two models are built upon the same residual blocks. For example, if comparing ResNet-18 and ResNet-34, the two model are both built with basic blocks, and the first two residual blocks have the exact same structure. Another example would be ResNet-50 and ResNet-152, both are built with bottleneck blocks and they share the same structure for the first seven blocks. By reusing the shared structure of two networks, we try to reduce computation amount during the query phase.

3.4 Overall Experiment Design

We run our experiment based on ResNet-50 and 152, in this way we can skip more blocks, compared with ResNet-18 and 34.

During ingest time, we classify all the patches using ResNet-50 and generates Top-3 indexes. At the same time, we save the feature map for the first several residual blocks and reuse it as the input of the expensive CNN. Here we tested two cases: one is reusing the first 4 blocks and another is reusing the first 7 blocks. For the query phase, we run the expensive model to predict on the feature map. We evaluate the performance with and without retrain the remaining layers of the model.

4 Evaluation

A ResNet-50 model and a ResNet-152 are trained on the data-set we mentioned above[1]. The ratio of training set and testing set is 4:1. We also retrain the remaining part of ResNet-152 with the feature maps that are output from the shared layers of the trained ResNet-50. All training curves are shown in the Appendix. The models are able to achieve almost 1 on training data-set and 0.95 to 0.97 on the testing data-set.

The developed system with trained model parameters is tested on a laptop computer to evaluate the latency of inference[2]. The performance results of the inference system are shown in table 1. For each scenario, the test is done 5 times to show average and standard deviation of latency.

It is observed that the feature map reusing models with weights trained from original structures cause huge loss in accuracy, which means that the feature maps from the shared layers of ResNet-50 model can not be fully understood by ResNet-152 although they have the same dimension. So, retrain the remaining layers of ResNet-152 would be the proper way to build the system without accuracy sacrifice.

We are able to achieve a 9.6% reduction of query time for the shallow cut retrained model with a compromise of 0.9% accuracy. Moreover, we get 17.5% reduction of query time for the deep cut retrained model with no accuracy reduction.

[1] Our system is built on the basis of a Keras-ResNet repository on Github https://github.com/raghakot/keras-resnet

[2] Dell Inspiron 7000 laptop with a i7-6700hq CPU and 16 GBs of memory.

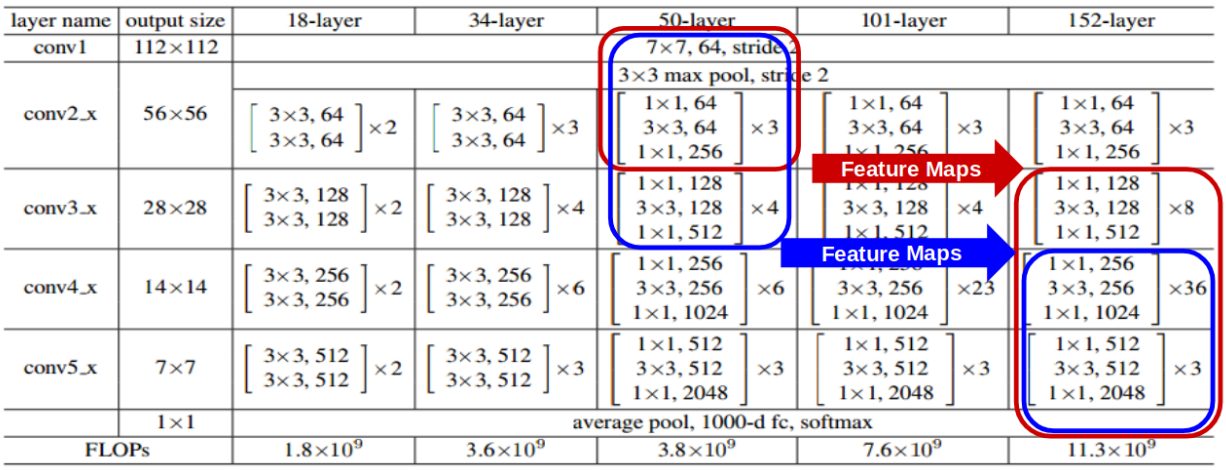

Figure 3: ResNet architecture and two different cuts.The red color reuses 4 blocks and the blue one reuses 7 blocks.

Model	Accuracy	Accuracy Reduction	Latency(sec/image)	Latency Standard Deviation	Latency Reduction
ResNet-50 + 152 (Focus Baseline)	0.967	N/A	0.3611	0.0038	N/A
Reuse Feature Map (shallow cut)	0.518	46.4%	0.3277	0.0072	10.6%
Reuse Feature Map (shallow cut, retrained)	0.958	0.9%	0.3266	0.0069	9.6%
Reuse Feature Map (deep cut)	0.391	59.6%	0.3058	0.0078	15.3%
Reuse Feature Map (deep cut, retrained)	0.969	0%	0.2978	0.0070	17.5%

Table 1: System performance of difference cuts

5 Related Work

5.1 Object Detection

Most existing object detection techniques, including two-stage and single-stage methods, rely on Convolution Neural Networks [14]. A two-stage method is a region proposal based method. It first generates region proposals and then predicts bounding boxes based on these proposed regions. Many researchers have been focusing on improving the performance and speed of two-stage method like R-CNN [5], Fast R-CNN [4], Faster R-CNN [18, 19], etc. Compared to the two-stage method, the single-stage method is usually faster and simpler [14]. So, the single-stage method is popular in speed demanding object detection tasks. For example, YOLOv1 [16] could perform real-time object detection in videos when most two-stage methods fail.

5.2 Other Video Inference Systems

Besides Focus, NoScope is another system for accelerating neural network query speed over videos [11]. To reduce the cost of doing Neural Network inference, NoScope uses a specialized shallower model that is less generalized but faster. The specialized model estimates the class of the objects in the frame with a confidence score. If the confidence is not high enough, a more accurate but higher cost model will be called to generate the final class for this object. However, NoScope is limited since it relies binary classifiers which is hard to scale.

6 Future work

First, our system needs to be tested on other datasets with more images and more classes. The current dataset contains just small patches of 3 different classes with a low resolution. Testing on more higher resolution datasets

can prove the robustness of the system. We would also like to explore strategies of reusing layers on different ResNet models, e.g. combine ResNet-18 and 152, on the inference time. A generic methodology for deciding where to cut the layers would be useful which may need the statistics of the time consumed on each layer. Moreover, we currently only focus on ResNet architecture, which imposes restrictions for the application of this system. Other types of models should also be considered.

Since we are building our system on top of Focus [9]. The other features in Focus should be realized in our system and some of these features can even be optimized. For example, the clustering of objects in Foucs can help reduce duplication of object patches, only uses simple incremental clustering method which is well-studied in [9, 15, 2]. However, this approach has some drawbacks. First, it relies on the hyper-parameters manually set before running the algorithm. Second, the clustering parameter T, the distance threshold between clusters, has a direct impact on both recall and precision. This requires the user has background knowledge about how to configure the clustering algorithm. Also, more clustering algorithms can be tested for better performance.

References

[1] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 2014.

[2] F. Cao, M. Estert, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM, 2006.

[3] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[4] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[6] A. B. Godbehere, A. Matsukawa, and K. Goldberg. Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. In *American Control Conference (ACC), 2012*, pages 4305–4312. IEEE, 2012.

[7] G. B. Goh, N. O. Hodas, and A. Vishnu. Deep learning for computational chemistry. *Journal of computational chemistry*, 38(16):1291–1307, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] K. Hsieh, G. Ananthanarayanan, P. Bodik, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu. Focus: Querying large video datasets with low latency and low cost. *arXiv preprint arXiv:1801.03493*, 2018.

[10] N. S. Jean-Philippe Jodoin, Guillaume-Alexandre Bilodeau. Urban tracker: Multiple object tracking in urban mixed traffic. *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[11] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope: optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment*, 10(11):1586–1597, 2017.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[14] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*, 2018.

[15] L. O'callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high-quality clustering. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 685–694. IEEE, 2002.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[17] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[18] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[19] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1137–1149, 2017.

[20] M. H. Segler, M. Preuss, and M. P. Waller. Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698):604, 2018.

[21] A. Van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

[22] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu. Deepcache: Principled cache for mobile deep vision. *arXiv preprint arXiv:1712.01670*, 2017.

7 Appendix

7.1 Training Curves

The models converge after about 50 to 100 learning epochs and the accuracy of evaluation comes to or above 95%. The learning curves for ResNet-50, ResNet-152 and the remaining part of ResNet-152 after two cuts is shown in Figure 4, Figure 5, Figure 6, Figure 7.

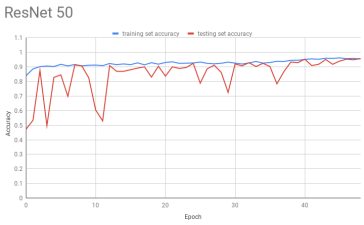

Figure 4: Training Curves of ResNet-50

7.2 Source Code of this Project

We provide all the source code of this project on Github and you check the code from this link,https://github.com/iphyer/FocousIngestOpt_FinalProject_CS744Fall2018.

The code is built on Python3 and Keras with Tensorflow as back-end engine.

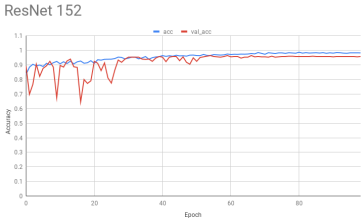

Figure 5: Training Curves of ResNet-152

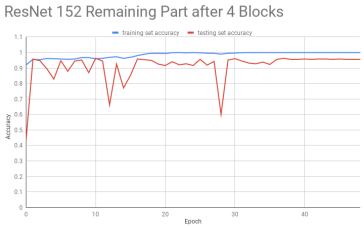

Figure 6: Training Curves of ResNet-152 cutting 4 blocks

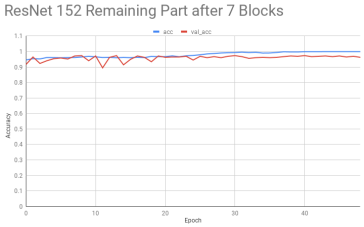

Figure 7: Training Curves of ResNet-152 cutting 7 blocks