Work-in-Progress: Video Analytics From Edge To Server

Jiashen Cao
Georgia Tech

Ramyad Hadidi
Georgia Tech

Joy Arulraj
Georgia Tech

Hyesoon Kim
Georgia Tech

ABSTRACT

Deep learning algorithms are an essential component of video analytics systems, in which the content of a video stream is analyzed. Although numerous studies target optimizing server-based video analysis, partially processing videos on edge devices is beneficial. Since edge devices are closer to data, they deliver initial insights on data before sending it to cloud. In this paper, we present an edge-tailored video analytics system by using a multi-stage network designed to run on heterogeneous computing resources.

ACM Reference Format:
Jiashen Cao, Ramyad Hadidi, Joy Arulraj, and Hyesoon Kim. 2019. Work-in-Progress: Video Analytics From Edge To Server. In *International Conference on Hardware/Software Codesign and System Synthesis Companion (CODES/ISSS '19 Companion), October 13–18, 2019, New York, NY, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3349567.3351733

1 INTRODUCTION

The advances in deep learning (DL) enables us to accurately evaluate large amount of data. For example, the VGG-16 neural network achieved 6.8% top-5 error rate in ILSVRC-2014 submission. Hence, DL has been widely adopted for many applications. Among these applications, video analytics has a high demand of DL support. This is because a query analyzer for video analytics performs several tasks: image classification, object detection, and action recognition.

Although the state-of-the-art DL algorithm provides high accuracy, it requires powerful computation resources to accomplish its tasks in a timely manner. This limits the wide adoption of video analytics. In addition, video analytics generates streams of data from multiple geographic locations, so the entire system involves large-scale data streams. For example, London city has around half a million surveillance cameras. Every second, gigabytes of videos are sent to data centers for analysis. The current approach for video analytics systems relies on centralized and powerful servers to process incoming streams. These servers must have a good bandwidth to data inputs (i.e., edge) and high availability of computation resources. To accommodate such needs in edge devices, more devices are deployed on the edge, thereby increasing the possibility of performing DL computation in the edge.

Nonetheless, the performance of DL computation on edge devices is a magnitude order slower; currently, the inference tasks are either performed on edge devices or offloaded to a server whenever possible [6]. In this paper, we propose that instead of performing all computation in either edge devices or servers, we split the DL computation network and use both edge and server systems with a dynamic scheduling algorithm. The scheduling algorithm is able to adjust amount of work on edge and server during runtime.

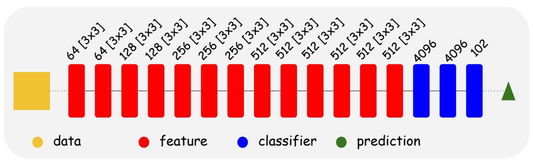
A traditional DL based system assumes that the entire system has one accuracy requirement. However, the assumption breaks in video analytics systems. For instance, police may run a query to look for the license plate number of criminal vehicles. This requires high accuracy to correctly identify the number on the license plate. In contrast, traffic regulators may run a query to check traffic conditions by estimating how many cars drive are on a road during a certain period of time. For these kinds of tasks, the system can still accomplish the task with relatively lower accuracy prediction.

Motivated by that, we propose a *multi-stage deep neural network*. The network is designed to control the execution time based on the demand of the accuracy. The network is designed such that earlier stages can make faster and less accurate predictions but as it progresses, the accuracy increases. If the query demands a higher accuracy, the system continues executing the query to a deeper layer to get fine-grained results. Compared to the traditional deep neural network, our approach provides quality of service to users by user controlled accuracy constraint, and the system chooses according layer to stop. Compared to systems, which use multiple models either lightweight but low accuracy or heavy but high accuracy, our approach only uses a single model during runtime. To run multiple models in a single system, the system needs to allocate additional buffer space and reconstruct the computation graph, which is a heavier task than query inference. By using multi-stage model, our system does not suffer from model switching overhead between queries. As a summary, our contributions in this work as follows: (i) To control the trade-off between accuracy and latency, we propose a multi-stage neural network. (ii) We design a scheduling algorithm with a new cost model to place query from edge to servers.

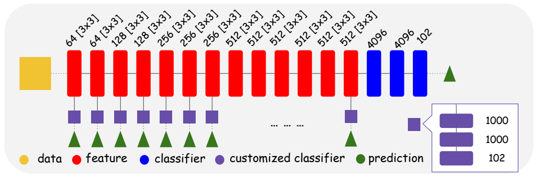
2 RELATED WORK

Neurosurgeon [6] proposes to partition the DL model for a status quo scenario. It searches for the optimal slicing point of the model and places the partitioned model to either edge or server. It focuses on reducing query latency for the end to end system. Hadidi et al. [1–3] propose methods to excute DL models entirely on multiple edge devices in a distributed fashion. Focus [4] proposes to filter out irrelevant image frames during ingest time. It designs a specialized lightweight convolution neural network to improve ingest time performance and uses clustering algorithm to eliminate redundant frames. Noscope [5] instead uses model specialization in which multiple optimized models are created for different tasks to filter out low probable data. Both Focus and Noscope use customized filtering techniques to improve the performance, and only focus on server-only optimization. To the best of our knowledge, our work is the first work that efficiently schedules a DL-based video analytics workload on heterogeneous platforms.

CODES/ISSS '19 Companion , October 13–18, 2019, New York, NY, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6923-7/19/10…$15.00
https://doi.org/10.1145/3349567.3351733



(a) Traditional Network.



(b) Multi-Stage Network.

Figure 1: Network Comparison.

3 MULTI-STAGE NETWORK

The motivation for designing a multi-stage network comes from the limitation of the traditional DL model, in which a DL model as Figure 1a shows, single inference needs to pass through all layers to reach the prediction result. However, in video analytics systems, not all queries require highly accurate results. Thus, by using the multi-stage network, the system is able to complete prediction earlier. Because of the advantages of the multi-stage network, the system saves bandwidth without sending unnecessary requests and also reduces single query latency if it does not require high accuracy.

Design: As shown in Figure 1b, the multi-stage network has customized classifiers attached to each of its feature layers (convolution layers). The feature layer is able to decide which layer to pass its output either to the next feature layer or to its classifier. During runtime, if the classifier attached to a feature layer is able to reach the accuracy threshold, the feature layer instead sends data to the classifier and receives the prediction result. By doing so, the system avoids extra latency of using later layers.

Implementation: We prototype our idea with the VGG-16 network on the flower-102 dataset. First, we train the original VGG-16 model (all feature layers and final classifier in Figure 1b), and then freeze all weights for the feature layers. As a last step, we train the customized classifiers (early prediction classifiers in Figure 1b) based on the frozen feature weights.

Preliminary Results: Our experiment is conducted on Nvidia TITAN XP GPU. Customized classifiers are appended to the first seven layers of the original VGG-16 neural network. Users have a total of eight options for accuracy and latency, including the original VGG-16 model, as shown in Table 1. For a query that makes prediction immediately after the first layer, it does suffer some performance degradation such that the accuracy drops to 53.7% as opposed to the entire model's accuracy of 89.4%. But it reaches a speed up of 6X compared to the original VGG-16 model. For a query that makes prediction after seven layers, it reaches up to a 1.3X speed up compared to the original model, and the accuracy of seven layers only drops to 83.2%. Combining all those intermediate results (from two layers to six layers), the system offers a wide variety of accuracy and latency trade-off options to users. We did not get a chance to fine-tune our model, but in future, we believe the model will achieve better results with hyper-parameters tuning. The advantage of our approach is that we do not force users to give up accuracy but leave the choice to users. We only provide quality of service, and if users do not require high accuracy, our system is able to optimize the query performance for users.

Number of Feature Layers	1	2	3	4	5	6	7	8 - 12	Last Feature
Early Prediction	✓	✓	✓	✓	✓	✓	✓	✗	✗
Accuracy (%)	53.7	57.8	64.1	69.6	75.7	79.9	83.2	-	89.4
Latency (ms)	0.4	0.9	1.1	1.4	1.6	1.8	2.0	-	2.6

Table 1: Latency and Accuracy of All Classifiers

4 WORKLOAD SCHEDULING

We used a greedy scheduling algorithm from Karnagel et al. [7], which finds an optimal workload placement on multiple devices such as various GPUs and CPUs. The basic idea is a greedy approach to find a close-to-optimal workload placement distribution. We plan to extend the idea to our heterogeneous video analytics system. First, we need to design a new cost model for the scheduler. The model will be stored in a lookup table, in which it shows the cost of running a specific partition on a specific device. The scheduler will use the cost model to choose the best partition combination.

5 CONCLUSIONS AND FUTURE WORK

Figure 1 shows that the current multi-stage network achieves a relatively fine-grained options of accuracy but between each layer it still has about a 5% accuracy gap. Therefore, as for future research, we target to leverage the gap between each accuracy level. The possible approaches are to use thinner but deeper network such as ResNet-50 and to add more specialized classifiers to the network. Other future directions for us are to extend to state-of-art datasets and also extend to other DL tasks, like object detection. We have tried other datasets and network structures, so we strongly believe that the multi-stage network applies to generalized datasets.

Regarding the workload scheduling, because the video analytics system has different constraints, including query accuracy, query latency, and bandwidth usage, we need to design a new cost model based on those parameters. The proposed scheduling algorithm in the paper by Karnagel et al. is slow to reach the optimal placement solution. The alternative solution proposed in this paper is to use the best placement found within a certain time constraint. In order to achieve better results, we will redesign a new scheduling algorithm with the new cost model. After we find a promising scheduling algorithm, we plan to test its functionality on the simulator first and then deploy it to actual devices.

ACKNOWLEDGEMENT

We thank anonymous reviewers for their valuable comments. This work was supported by NSF CSR 1815047.

REFERENCES

[1] Jiashen Cao et al. 2019. An Edge-Centric Scalable Intelligent Framework To Collaboratively Execute DNN. *Demo for SysML Conference, Palo Alto, CA* (2019).
[2] Ramyad Hadidi et al. 2018. Distributed Perception by Collaborative Robots. *IEEE Robotics and Automation Letters (RA-L)* 3, 4 (Oct 2018), 3709–3716.
[3] Ramyad Hadidi et al. 2018. Real-Time Image Recognition Using Collaborative IoT Devices. In *ReQuEST '18*. Article 4.
[4] Kevin Hsieh et al. 2018. Focus: Querying Large Video Datasets with Low Latency and Low Cost. arXiv:cs.DB/1801.03493
[5] Daniel Kang et al. 2017. NoScope: Optimizing Neural Network Queries over Video at Scale. arXiv:cs.DB/1703.02529
[6] Yiping Kang et al. 2017. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. In *ASPLOS '17*. 615–629.
[7] Karnagel et al. 2017. Adaptive Work Placement for Query Processing on Heterogeneous Computing Resources. *VLDB* 10, 7 (March 2017), 733–744.