Collaborative Edge and Cloud Neural Networks for Real-Time Video Processing

Philipp M. Grulich
University of California, Santa Cruz
Technische Universität Berlin
pgrulich@ucsc.edu

Faisal Nawab
University of California, Santa Cruz
Santa Cruz, CA
fnawab@ucsc.edu

ABSTRACT

The efficient processing of video streams is a key component in many emerging Internet of Things (IoT) and edge applications, such as Virtual and Augmented Reality (V/AR) and self-driving cars. These applications require real-time high-throughput video processing. This can be attained via a collaborative processing model between the edge and the cloud—called an *Edge-Cloud model*. To this end, many approaches were proposed to optimize the latency and bandwidth consumption of Edge-Cloud video processing, especially for Neural Networks (NN)-based methods. In this demonstration. We investigate the efficiency of these NN techniques, how they can be combined, and whether combining them leads to better performance. Our demonstration invites participants to experiment with the various NN techniques, combine them, and observe how the underlying NN changes with different techniques and how these changes affect accuracy, latency and bandwidth consumption.

PVLDB Reference Format:
Philipp M. Grulich and Faisal Nawab. Collaborative Edge and Cloud Neural Networks for Real-Time Video Processing. *PVLDB*, 11 (12): 2046-2049, 2018.
DOI: https://doi.org/10.14778/3229863.3236256

1. INTRODUCTION

Many Internet of Things (IoT) applications rely on the efficient, real-time processing of video streams. This includes Virtual and Augmented Reality (V/AR), video surveillance, self-driving cars, and many more. Recent advances in image processing via Neural Networks are enabling these applications. A challenge that faces these applications, however, is the requirement of real-time processing on edge devices with low computation power. This is especially true for Neural Network (NN) models that are becoming increasingly complex. This forces system designers to opt for one of two extremes: Process images in the cloud but suffer from high wide-area latency and bandwidth costs, or process images in the edge device and relax the accuracy or real-time requirements. A collaborative model between the edge and the cloud (an *Edge-Cloud* model) has the potential of avoiding the disadvantages of the two extremes.

Prior work proposed various Edge-Cloud techniques for video processing [3, 4]. We focus on a subset of these NN techniques that are used for real-time video processing or that are proposed in a general context but are applicable to real-time video processing : (1) Splitting: the neural network is partitioned to an edge partition and a cloud partition [4], (2) Compression: data is compressed in the edge before being sent to the cloud, and (3) Differential communication: only send the difference between the current frame and the previous frame [7, 1]. Each proposed NN technique improves the processing and communication of Edge-Cloud video processing. However, it is not clear whether these NN techniques can be combined to achieve a higher performance. It turns out that combining these NN techniques is a nuanced exercise since some techniques performed tasks that transformed the data to forms that are unusable by the other techniques.

In this demonstration, we propose methods to combine the NN techniques to achieve an overall higher performance. Specifically, we explore the space of all combinations and propose a method for each point in this space. Also, we focus on applying these NN techniques and their combinations to the collaborative Edge-Cloud model. The combination of the NN techniques and fitting them to the Edge-Cloud model results in novel designs that—if done carefully—outperform the various NN techniques individually.

We designed the demonstration to encourage participants to evaluate the performance of the different NN techniques and there combinations. Also, we present an illustration of the underlying NN architecture with various NN techniques and their combinations. This visualization is aimed to provide the intuition of the NN techniques and their performance characteristics.

In this proposal, we present the NN techniques and their combinations (Section 2), the proposed demonstration (Section 3) and evaluations using our framework (Section 4).

2. EDGE-CLOUD NEURAL NETWORKS

2.1 System Model

Our System Model consists of edge nodes, which are connected over WAN links to a central cloud node. The edge node consists of a video camera that is recording videos to be processed via a pre-trained NN. The edge node is capable of processing the incoming video frames but has minimal

Proceedings of the VLDB Endowment, Vol. 11, No. 12
Copyright 2018 VLDB Endowment 2150-8097/18/8.
DOI: https://doi.org/10.14778/3229863.3236256

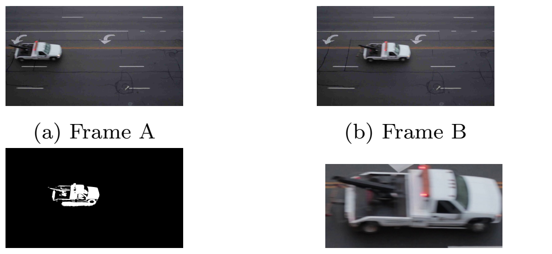

(a) Frame A (b) Frame B (c) Difference image (d) Cropped out difference

Figure 1: Example of differential image transfer

compute capacity and can not rely on special hardware accelerators such as GPUs. Therefore, many techniques for real-time processing of NN are not applicable.

Because the cloud node has—compared to edge nodes—unlimited computational resources, the edge nodes may offload the task of processing the NN to the cloud. However, communication between the edge and the cloud incurs a large latency of up to 100ms and a high bandwidth cost [7]. Therefore we have to combine different NN optimization techniques to reduce the amount of transfered data from the edge to the cloud.

2.2 Neural Networks Techniques

There is a number of techniques to increase the efficiency of NN inference. We focus on NN techniques that are applicable to the Edge-Cloud NN model. The following is a description of these techniques.

(a) Neural Network Splitting. The NN splitting technique [4] is proposed specifically for Edge-Cloud NN. In this technique, the NN is divided into an *edge part* and a *cloud part*. This division occurs at a specific layer of the NN. For example, consider a NN with n layers. A split at the i^{th} layer results in an edge part that consists of the NN's layers 1 to i and a cloud part that consists of the NN's layers $i+1$ to n. Processing the frame is then performed cooperatively between the edge and cloud nodes. The edge node uses the edge part of the NN to compute the output of layer i. Then, the edge node sends the output of layer i to the cloud node that continues the processing using the cloud part of the NN. This splitting enables utilizing the compute resources at the edge which makes the processing at the cloud more efficient. Also, by judiciously choosing the split layer, it is possible to optimize the communication bandwidth, processing latency, and energy consumption.

(b) Differential Communication. In many video processing applications, the typical frame-to-frame difference is a small fraction of the whole frame. Therefore, rather than sending the whole frame to the cloud, it is possible to only send the difference of the frame compared to the previous frame, which leads to savings in communication bandwidth. This is especially useful for applications that uses a fixed camera to record videos, since the background remains mostly unchanged. This concept is also used by a wide range of video compression algorithms. In this work, we focus on techniques that can be applied in a frame-to-frame basis. This will enable a grater flexibility in the image processing pipeline. Figure 1 shows an example. Frame B (Figure 1b) is the new frame where we want to extract the differences to our reference frame, frame A (Figure 1a). We use the Gaussian Mixture-based Background Segmentation to generate a difference map [2] between the two frames. Using this map, we can crop out a small region that represents the difference relative to the original frame (Figure 1d). The edge node can communicate the difference rather than the whole frame, which can save bandwidth. The cloud node can reconstruct the image using the difference only, because it has a copy of the original frame.

(c) Compression. In a system where the images are sent from the edge node to the cloud node, it is possible to compress the image to reduce the communication bandwidth. Compression introduces a computational overhead for compression and decompression. However, the transmission of a smaller image representation might lead to overall better performance. A special case are lossy compression techniques—where the reconstructed image is not identical to the original image. This might lead to a higher compression ratio, but can also influence the prediction accuracy negatively.

2.3 NN Techniques Combinations

In this section, we investigate the combination of the three Edge-Cloud NN techniques: *splitting*, *differential communication*, and *compression*. Some of these combinations are straight-forward. However, others are more nuanced, especially the combination of *splitting* with the other techniques. *Splitting*, unlike the other techniques, is specific to NN. Therefore, combining *splitting* with the other techniques makes the other techniques utilize the specific structure of the NN, which opens more opportunities to optimize performance.

Differential Communication and Compression. In this combination, the edge node extracts the difference between the raw frames. Then, it compresses the difference and sends it to the cloud node. The cloud node decompresses the difference and reconstructs the image that is then processed by the NN. It is possible to further optimize this technique by applying the level of compression selectively across frames or within the same frame.

Splitting and Compression. This combination, like splitting, utilizes an edge part and a cloud part of the NN. However, the main difference is that the output of the i^{th} layer is compressed before it is sent to the cloud node. The cloud node decompresses the received information for layer i before applying it to the cloud part of the NN. Compression can have two forms: (1) the output of layer i is compressed as a block of bytes using a compression algorithm, and (2) the value of each neuron in the layer is compressed. Rather than sending the full float value (32bit) of a neuron, we may send a more compact representation in a smaller number of bits and a lower precision. For our work we combine both approaches with the lossy floating point compression library *zfp* [5]. That allows use to configure the number of fixed precision bits and compresses the whole layer at once.

Splitting and Differential Communication. This combination is not as straight-forward as the previous combinations. At a high-level, the problem of combining splitting and differential communication is about finding a way to extract the difference of layer i of frame f_j to layer i of frame f_{j-1}, where f_{j-1} is the frame before f_j. There are some challenges in solving this problem. First, unlike the raw frame images, a layer in the NN does not necessarily exhibit spatial properties—an individual neuron's value depends on input that potentially span the whole image. (there are exceptions to this in some layers of convolutional NNs, but we consider the general case in this paper.) Therefore, it is not feasible to extract a set of neurons that represents the spatial difference between two frames.

We suggest the following heuristic to combine splitting with differential communication: The heuristic selects a subset of neurons in layer i to represent the difference between two frames by using a threshold t. Consider the value of a neuron in frame f_j, n_{f_j}, and its corresponding value for frame f_{j-1}, $n_{f_{j-1}}$. If $\left|n_{f_j} - n_{f_{j-1}}\right| < t$, then we do not send the value of that neuron to the cloud node and the cloud node reuses the value $n_{f_{j-1}}$ for n_{f_j}. The reason for needing a threshold rather than only discarding a neuron's value if it is identical to the previous frame ($t = 0$) is that the neuron values are sensitive to the smallest changes which typically makes all the neurons values different.

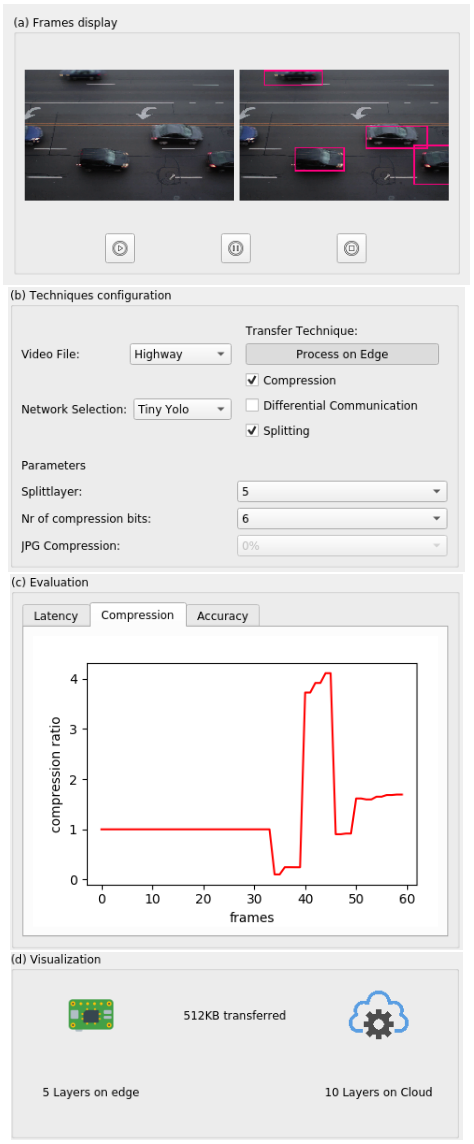
All Techniques. Combining all techniques is similar to the combination of splitting and differential communication. To introduce compression, the derived difference of layer i is compressed at the edge node before it is sent to the cloud.

3. DEMONSTRATION

The main goal of our demonstration is to illustrate the intricacies involved in using and combining NN techniques. Specifically, a NN technique might act differently when applied to different videos. Also, a NN technique needs to be configured and finding the right configuration is challenging. Combining NN techniques complicates the issue further. It is not always clear how to combine various techniques. After combining the techniques, what remains to be understood is whether combining techniques would yield an improvement of certain performance metrics while degrading others. These issues, questions, and challenges are what we attempt to illustrate in our demonstration.

Figure 2 is a mock-up of our proposed demonstration. The demonstration invites participants to observe the processing of a video using a neural network. The participant observes the frames that are being processed (section (a) of the demonstration). Also, the participant has the choice of configuring the demonstration (section (b) in the demonstration). This includes the selection and configuration of the particular NN techniques whereby the participant can explore the trade-off between data reduction and accuracy. We also provide two video samples of the same application—videos of traffic—that nonetheless affect the processing and NN techniques differently. The two videos were selected to demonstrate that the success of a techniques also depends on the processed video.

Depending on which NN techniques are selected, the participant is prompted to enter the parameters to configure the NN techniques. For example, in Figure 2, both compression and splitting are selected. Because of this selection, the participant is also prompted to enter the neuron value size, which is a parameter corresponding to the use of compression and splitting. (the neuron value size is the number of bytes used to represent each neuron's value in the split layer when it is sent from the edge node to the cloud node.) Also, the participant is prompted to enter the split layer, which is a parameter corresponding to the use of the splitting technique. (the split layer decides which layers are processed in the edge node and which layers are processed at the cloud node.) If differential communication is selected, the participant would also be prompted to enter the compression



Video File: Highway

Network Selection: Tiny Yolo

Transfer Technique: Process on Edge

✓ Compression

Differential Communication

✓ Splitting

Parameters

Splitlayer: 5

Nr of compression bits: 6

JPG Compression: 0%

512KB transferred

5 Layers on edge

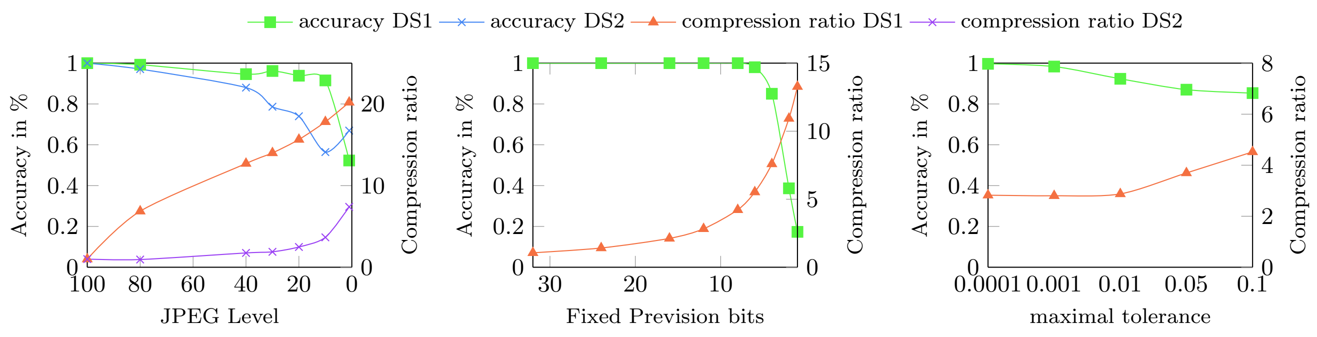
10 Layers on Cloud

Figure 2: The interface of the proposed demonstration

level. In our case, the compression level may correspond to: (1) JPEG compression levels for raw images or (2) The number of bits to represent neurons for neuron layers.

The performance metrics that are evaluated in this demonstration are latency (the time to process the image and apply the NN technique on the edge), compression ratio (the ratio between the size of the original image to the size of the compressed image), and accuracy (what percentage of the images detected by the original NN are detected by the NN after applying the techniques). Section (c) in the demonstration displays the evaluated metrics over time. Throughout the demonstration, the participant can change the used techniques and configuration parameters. This allows comparing the effect of the various techniques and configuration. In the figure, for example, the participant turned on compression at around frame 40. Therefore, the compression ratio increases to around 4 when this change is made.

The last part of the demonstration is an illustration of the underlying NN after applying the NN techniques (section (d)



(a) JPEG Image Compression (b) Layer Compression (c) Layer Diff Compression

Figure 3: Evaluation of the single NN compression techniques

in the demonstration). For example, Figure 2 shows what would be shown if compression and splitting are chosen as NN techniques. In this case, the edge NN consists of layers 1 to 6 (6 is chosen as the split layer in (b)) and the cloud NN consists of the remaining layers. Also, the communication between the edge and cloud nodes is compressed which is shown as a comment on the arrow between the two nodes. The NN illustration changes according to the chosen NN techniques and configured parameters.

4. EVALUATION

In the course of the demonstration we will guide the participant in exploring the various challenges and trade-offs in using NN techniques and their combinations. To this end, we prepare a set of evaluations that we invite the participant to explore and recreate. In this section, we present three sets of these evaluations to show what we aim to demonstrate to the participants. **Evaluation Setup:** For the demonstration and the evaluation we use the Tiny-YOLO NN Model, which is a smaller and faster version of the original YOLO2 network [6]. This is mainly due to our target of low power edge devices, which are not able to compute the original YOLO2 network efficiently. We use a processor with 2 cores @ 1.6 Ghz each without any hardware accelerators such as GPUs.

Image compression: To evaluate the compression of single frames we use standard JPEG compression, because it is available on low-end devices with a reasonable performance. In Figure 3a, we evaluate the effects of different compression levels with two different datasets, DS1 and DS2. For both datasets, we get a higher compression factor with lower JPEG Levels. For DS1, we get an accuracy over 95% up to compression level 30 and a very high compression ratio of nearly 15 at that point. The accuracy of DS2 drops more rapidly (after compression level 80). The main reason for this is that DS2 contains images with smaller details that cannot be recognized with a moderate compression level. In summary, frame compression can improve the compression ratio significantly without suffering from a drastic loss of accuracy. However, the benefits of frame compression depend highly on the dataset.

Differential communication on Images: As we have discussed in Section 2.3, the extraction of differences between frames is promising for data reduction. Our experiments on DS1 highlight that we can achieve an average data reduction ratio of 18 with a latency of 80ms per frame. This will not lead to any reduction in accuracy because we still reconstruct the original frame at the cloud node.

Splitting + Compression: For this experiment we compress the output of a particular layer. In figure 3b we evaluate how the number of fixed precision bits influences the accuracy of the NN model. We observe that we can reduce the fixed bits drastically and reach a compression ratio of 5 (Original data is 5 times bigger) with only a small loss of accuracy.

Splitting + Differential communication: Here, we combine the differential communication approach with the lossy layer compression. In figure 3c we show how the tolerance factor will influence the accuracy. We notice that from a tolerance factor $t > 0.01$ the compression ratio increases up to a factor of 2. However, this leads also to a reduction in accuracy. Therefore, this technique is only applicable if a small improvement on top of compression is desired even if it leads to a larger loss of accuracy.

5. SUMMARY AND CONCLUSION

We propose a demonstration that highlights the subtleties of using and combining NN techniques for real-time video processing. We focus on three techniques: splitting, differential communication, and compression, and propose methods to combine them. Our demonstration aims to provide an intuition about the challenges and trade-offs in combining NN techniques and the significance of correct configuration.

6. REFERENCES

[1] N. Fraser. Differential synchronization. In *Proceedings of the 9th ACM symp. on Document eng.*, 2009.

[2] P. Kaewtrakulpong and R. Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection. 2001.

[3] D. Kang et al. Noscope: optimizing neural network queries over video at scale. *PVLDB*, 10(11):1586–1597, 2017.

[4] Y. Kang et al. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *ASPLOS*, 2017.

[5] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE transactions on visualization and computer graphics*, 20(12):2674–2683, 2014.

[6] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[7] A. Vulimiri et al. Wanalytics: Geo-distributed analytics for a data intensive world. In *SIGMOD*, 2015.