工學 學士 學位論文

Sample Thesis Title in English

(학위 논문 제목 한글 예시)

2021년 8월

서울 대학교 공과 대학

電氣·精報工學部

鄭在源

Sample Thesis Title in English

指導教授 @ 0

t 논8D 공학 학¬학 논8<\ œ함.

서울 대학교 공과 대학

電氣·精報工學部

¬원

¬원X 공학 학¬학 논8D x 함.

2020년 6월 11 |

À도교 @ 0 (x)

ABSTRACT

Following the recent success of deep neural networks (DNN) on video computer vision tasks, performing DNN inferences on videos that originate from mobile devices has gained practical signi cance. As such, previous approaches developed methods to o oad DNN inference computations for images to cloud servers to manage the resource constraints of mobile devices. However, when it comes to video data, communicating information of every frame consumes excessive network bandwidth and renders the entire system susceptible to adverse network conditions such as congestion. Thus, in this work, we seek to exploit the *temporal coherence* between nearby frames of a video stream to mitigate network pressure. That is, we propose *ShadowTutor*, a distributed video DNN inference framework that reduces the number of network transmissions through intermittent *knowledge distillation* to a student model. Moreover, we update only a subset of the student's parameters, which we call *partial distillation*, to reduce the data size of each network transmission. Speci cally, the server runs a large and general *teacher* model, and the mobile device only runs an extremely small but specialized *student* model. On sparsely selected *key frames*, the server partially trains the student model by targeting the teacher's response and sends the updated part to the mobile device. We investigate the e ectiveness of ShadowTutor with HD video semantic segmentation. Evaluations show that network data transfer is reduced by 95% on average. Moreover, the throughput of the system is improved by over three times and shows robustness to changes in network bandwidth.

Keyword: deep neural network, distributed inference, knowledge distillation, video semantic segmentation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep learning approaches have proved to be extremely powerful in fields such as computer vision [1], natural language processing [2], speech recognition [3], and sequential recommendation [4]. Due to their applicability to various services, such DNN models sparked interest for performing DNN inference directly on data generated on mobile devices.

Video is one of the most important among such data due to its numerous practical applications. Autonomous vehicles perform road segmentation to traverse the street and detect obstacles [5]. CCTV cameras used for home security can notice movements and detect objects using DNNs [6]. Mobile selfie apps segment humans and the rest to change the background or simulate background blurring effects [7]. Thus, video DNN inferences on comparatively weak devices have become a core mechanism for delivering various services and technologies.

In sum, we make the following contributions:

- We design ShadowTutor, a distributed video DNN inference framework that reduces the number of network transmissions and the data size of each.
- We allow ShadowTutor to be robust to changes in network conditions by adopting asynchronous inference.
- We aid the configuration of ShadowTutor by providing analytic models of its network traffic and throughput in terms of system parameters and component measurements.
- We adapt ShadowTutor to HD video semantic segmentation, a real-life application, and evaluate its performance in terms of throughput, network traffic, accuracy, and robustness.

The rest of this paper is organized as follows. We review previous approaches in chapter 2, and provide background regarding our work in chapter 3. We explain the core idea and mechanism of ShadowTutor in chapter 4, and show how it is adapted speci cally to video semantic segmentation in chapter 5. Then, we evaluate our framework in chapter 6. Finally, we introduce related work in chapter 7 and conclude the paper in chapter 8.

Chapter 2

Previous Approaches

Due to the resource constraints of mobile devices, DNN computations are often o oaded to central cloud servers. Neurosurgeon [8] partitions a given DNN and collaboratively executes it between the cloud and the server. It searches for an optimal partition point in terms of latency and power consumption. Eshratifar et al [9] also seeks to partition DNNs, but by deriving optimal conditions for client power consumption and cloud resource constraints. MCDNN [10] generates modi ed and specialized versions of the original model to trade-o accuracy for improved resource usage. However, it only deals with models that perform image classi cation, requires the execution of the model for every image in the training set for specialization (preforwarding), and the generated model is mostly limited to being a layer-wise modi cation of the original model. Moreover, adopting these approaches still require the communication of every frame across the network, thereby pressuring network tra c greatly.

Another work by Yang et al. [11] splits video frames into partitions that are distributed to slaves. It utilizes optical ow, i.e. the movement vectors between two frames, to exploit the temporal coherence between frames. However, their system still requires the communication of every single frame between the master and the slaves, since they utilize temporal coherence only to occasionally reduce the amount of computation, rather than reducing network tra c.

Chapter 3

Background

3.1 Knowledge Distillation

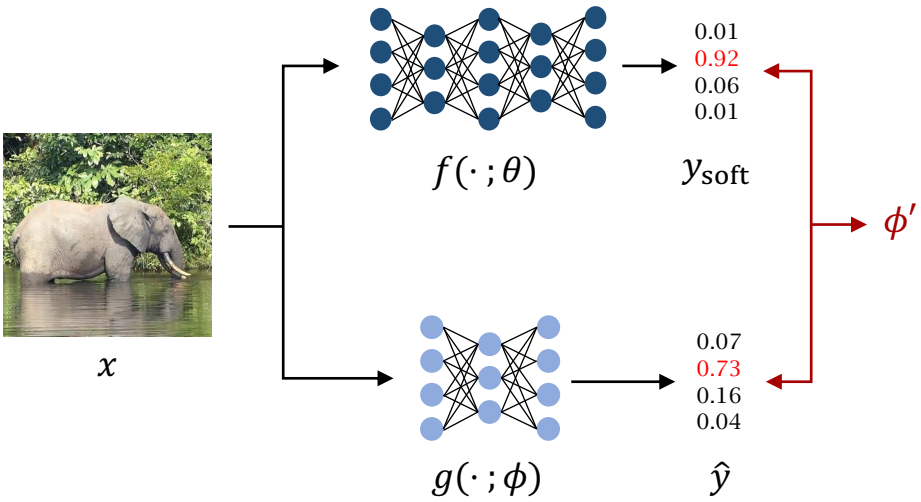


Figure 3.1: Inference time knowledge distillation. The student's parameters are updated by viewing the teacher's ouptut as the label.

Knowledge distillation [12] was initially proposed as a means of model compression. Viewing a large teacher model's output as a *soft target*, it formulated a knowledge distillation loss between the output of a small student model and the soft target. Compared with *hard targets*, which are one-hot vectors from the dataset, soft targets provide much more information about the input data, and allow less variance in the gradients of the student model.

Figure 3.1 illustrates the concept of knowledge distillation applied to image classi cation. Since it assumes inference time, the actual label y_{hard} is not available. Thus, the student learns just from the output of the teacher model y_{soft}. Is is also the case for ShadowTutor.

Chapter 4

ShadowTutor

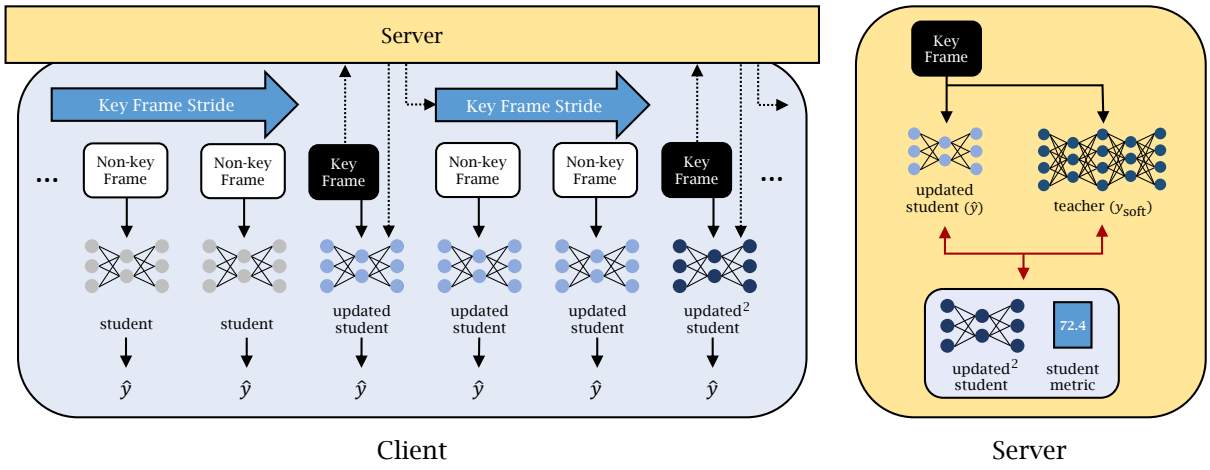


Figure 4.1: High-level view of ShadowTutor. Dotted arrows denote network communication, while other arrows denote data transfers within a device. Each frame is processed one by one sequentially. Non-key frame inferences are done on-device with a small student network. Sparse key frames are used to update the student's weights via knowledge distillation from a teacher.

Figure 4.1 shows an overview of ShadowTutor. All frames are processed sequentially in temporal order. Determined by the key frame scheduling policy, if the current frame is a non-key frame, its inference is handled entirely on-device by a lightweight student model. If the current frame is a key frame, it is sent to the server. Using the key frame, the server updates the a copy of the student's weights via knowledge distillation from the teacher, and returns the updated weights to the client. Network communication occurs only at sparse key frames, allowing ShadowTutor to reduce the amount of network tra c drastically.

4.1 System Components

Stripping away the distributed nature from ShadowTutor, there are ve essential components that comprise the system: video data, teacher inference, student inference, student training, and key frame striding.

4.1.1 Video data

Video data reside in the mobile device, and the majority of the frames do not leave its home.

4.1.2 Teacher inference

By *teacher*, we refer to a heavy but high-quality neural network deployed to the server.

4.1.3 Student inference

By *student*, we refer to a lightweight neural network designed to run on the mobile device.

4.1.4 Student training

We emphasize that this process be distinguished from student pre-training; student training is done repetitively during system runtime, and student pre-training is done exactly once when the system is rst organized.

4.1.5 Key frame striding

After training the student on the current key frame, the distance to the next key frame must be determined.

4.2 ShadowTutor

Now, we aggregate the system components into ShadowTutor. Algorithm 1 and 2 shows the runtime operations of the server and the mobile device (client) respectively. ShadowTutor of- oads teacher inference and student training to the server. Needless to say, teacher inference is an unacceptable workload for mobile devices, so o oading is a natural choice.

Algorithm 1: ShadowTutor-Server

Input: Student model *student*, teacher model *teacher*

```
1 ToClient(student)
2 while forever do
3 |   FromClient(frame)
4 |   label ← Forward(teacher, frame)
5 |   student; metric ← Train(student, frame, label)
6 |   ToClient(UpdatedPart(student), metric)
7 end
```

Algorithm 2: ShadowTutor-Client

Input: Target video stream *video*
Output: Student predictions

```
1  stride ← MIN_STRIDE
2  step ← stride                                  // First frame as key frame
3  updated ← true                                 // Whether student recv is done
4  FromServer(student)
5  foreach frame in video do
6  |   if step = stride then                                          // Key frame
7  |   |   ToServerAsync(frame)
8  |   |   async_recv ← FromServerAsync(student_di , metric)
9  |   |   step ← 0
10 |   |   updated ← false
11 |   end
12 |   prediction ← Forward(student, frame)
13 |   step ← step + 1
14 |   if not updated then
15 |   |   if step = MIN_STRIDE then
16 |   |   |   stride ← NextStride(stride, metric)
17 |   |   |   updated ← true
18 |   |   end
19 |   end
20 end
```

4.3 Network Traffic and Throughput Bounds

In this section, we model ShadowTutor's network tra c (amount of data transferred per unit time) and throughput (number of frames processed per unit time) and derive formulae for their lower and upper bounds. We use the notations de ned in table 4.1. Note that t_{net} and s_{net} regard the transmission of one key frame and the corresponding updated student parameters.

Both network tra c and throughput rely on the system's total execution time, which we rst model. ShadowTutor is designed to perform asynchronous inference for at most MIN_STRIDE many frames after a key frame. However, a mobile device may either be able to execute stu-

Table 4.1: Notations used for ShadowTutor. Those in the first block are identified after system execution, and the second based on system component decisions.

Symbol	Definition
n	number of frames processed
d	number of distillation steps taken
k	number of key frames
t_{si}	latency of student inference
t_{sd}	latency of one student distillation step
t_{ti}	latency of teacher inference
t_{net}	network latency associated with one key frame
s_{net}	networked data size associated with one key frame

dent inference and network operations entirely in parallel, or it may not support any form of concurrency. Therefore, t_c, the execution time of `MIN_STRIDE` frames after a key frame, is within the following bounds:

$$\begin{aligned} t_c &\geq \max(\texttt{MIN\_STRIDE} \times t_{si}; t_{net} + t_{ti}) \\ t_c &\leq \texttt{MIN\_STRIDE} \times t_{si} + t_{net} + t_{ti} \end{aligned} \tag{4.1}$$

Then, with t_c, the total execution time for processing n frames can be modelled as follows:

$$t_{tot} = (n - k \times \texttt{MIN\_STRIDE})t_{si} + dt_{sd} + kt_c. \tag{4.2}$$

Now, we obtain a general formula for network traffic by dividing the total size of data transfer by the total execution time:

$$\frac{ks_{net}}{t_{tot}} = \frac{ks_{net}}{(n - k \times \texttt{MIN\_STRIDE})t_{si} + dt_{sd} + kt_c}. \tag{4.3}$$

Chapter 5

ShadowTutor for Video Semantic Segmentation

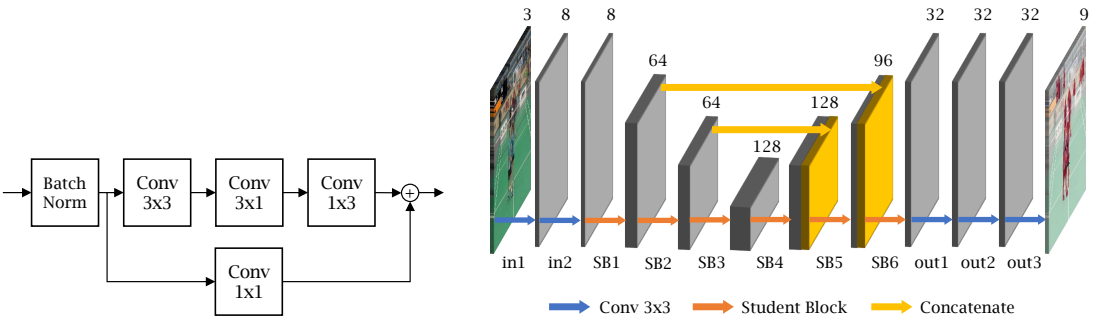
5.1 Experiment setup

As the server, we use a desktop computer equipped with an AMD Ryzen 7 3700X CPU, one NVIDIA RTX 2080ti GPU, and 32 GB memory. As the client, we use the NVIDIA Jetson Nano embedded board [13], equipped with a quad-core ARM A57 CPU, a 128-core Maxwell GPU, and 4 GB memory. Jetson Nano can deliver up to 472 GFLOPS for 32-bit oating points, which is not an unrealistic number for modern mobile devices. For example, Google Pixel 4's Qualcomm Snapdragon 855 can deliver up to 954.7 GFLOPS (32-bit) with its built-in Adreno 640 GPU [14]. As to network con gurations, we limit both uplink and downlink bandwidth to 80 Mbps, assuming strong Wi-Fi connection. We implement the system with OpenMPI [15], PyTorch [16], and Detectron2 [17].

5.2 System Component Decisions

We target videos with 25{30 FPS from the Long Video Segmentation (LVS) dataset [18].

The student neural network is a simple fully-convolutional network.



(a) Operations of a student block

(b) Student architecture

Figure 5.1: The student model is a small fully-convolutional network.

5.3 Algorithm Parameter Decisions

We first determine THRESHOLD using the performance of existing video semantic segmentation models.

Chapter 6

Evaluation

Using the configurations from chapter 5, we show ShadowTutor's effectiveness. Specifically, we investigate the advantages of ShadowTutor in terms of throughput, network traffic, and accuracy in sections 6.1, 6.2, and 6.3, respectively. Then, we push ShadowTutor to the limits in terms of network bandwidth in section 6.4 and temporal coherence in section 6.5.

6.1 Throughput

Table 6.1 summarizes the latency of one distillation step (ms) and the mean number of distillation steps for partial and full distillation. Partial distillation reduces both the latency and the number of distills, contributing to the throughput of the entire system. This result empirically supports the claim that since the number of training steps is limited, it is quicker to exploit a fixed distribution of features than to explore for better ones.

Table 6.2 lists the actual throughput of the system (frames processed per second) and the total execution time. As expected, partial distillation outperforms full distillation in every category. Moreover, ShadowTutor shows an improvement greater than 3x over naive offloading. This is especially because ShadowTutor only communicates with the server on key frames, and thus drastically reduces the latency for networking.

6.2 Network Traffic

We investigate the reduction of network traffic in terms of the amount of data transfer per key frame and the ratio of key frames to all frames.

Table 6.1: Execution time and mean number of distillation steps

Distillation	Partial	Full
One step (ms)	13	18
Mean # of steps	3.83	4.44

Table 6.2: Frames processed per second (FPS) and execution time (s) in parenthesis

Camera	Scene	Partial	Full	Naive
fixed	animals	6.55(762.5)	6.21(804.5)	2.09(2391.3)
fixed	people	**6.60(757.4)**	6.43(777.0)	2.09(2391.3)
fixed	street	6.50(768.8)	5.95(840.5)	2.09(2391.3)
moving	animals	6.57(760.5)	6.27(796.5)	2.09(2391.3)
moving	people	**6.59(758.5)**	6.36(785.8)	2.09(2391.3)
moving	street	6.41(780.2)	5.55(901.0)	2.09(2391.3)
egocentric	people	6.57(760.5)	5.89(848.5)	2.09(2391.3)
average		**6.54(764.1)**	6.08(822.0)	2.09(2391.3)

Table 6.3: Data transmitted on each key frame (MB).

Direction	Partial	Full	Naive
To Server	2.637	2.637	2.637
To Client	**0.395**	1.846	0.879
Total	**3.032**	4.483	3.516

Table 6.3 shows the amount of data transfer (in MB) per key frame. Since it suffices to send only the updated part of the student, partial distillation reduces network traffic compared with full distillation. Against naive offloading, which sends the teacher prediction to the client, ShadowTutor reduces the amount of data transfer by 13.77% per key frame, because the size of the student is even smaller than one video frame.

Table 6.4 summarizes the proportion of key frames and the actual network traffic in Mbps. The smaller the key frame proportion, the less frequent the network communication. Partial distillation generally performs better than full distillation, and strictly better than naive offloading. Especially, for the fixed-people category, the number of network communications is only 1.96% compared with the naive offloading scheme, yielding a surprising 98% reduction.

The effects of the two reductions are multiplicative. Thus, compared with naive offloading, ShadowTutor reduces the amount of network transfer per frame by 98.3% at most, and 95.3% on average.

On the other hand, the reduction in network traffic (amount of networked data per unit

Table 6.4: Key frames ratio (%) and network tra c (Mbps)

Camera	Scene	Key frame ratio			Network tra c	
		Partial	Full	Naive	Partial	Naive
xed	animals	4.73	4.60	100.0	7.51	58.51
xed	people	**1.96**	2.42	100.0	**3.14**	58.51
xed	street	7.78	7.43	100.0	12.27	58.51
moving	animals	2.55	2.29	100.0	4.06	58.51
moving	people	**3.45**	4.12	100.0	5.51	58.51
moving	street	11.70	11.48	100.0	18.19	58.51
egocentric	people	**5.46**	9.75	100.0	8.70	58.51
average		**5.38**	6.01	100.0	**6.19**	58.51

Table 6.5: Mean IoU of various settings. Wild = pre-trained student on its own, P = partial distillation, F = full distillation, digit (1 or 8) = number of delayed frames before receiving updated student weights.

Camera	Scene	Wild	P-1	P-8	F-1	Naive
xed	animals	14.34	74.31	73.27	74.47	100.0
xed	people	13.91	81.69	81.39	81.36	100.0
xed	street	17.28	**70.26**	69.01	63.60	100.0
moving	animals	22.31	74.94	73.80	75.21	100.0
moving	people	17.62	74.82	74.06	75.55	100.0
moving	street	18.65	**60.48**	58.61	52.94	100.0
egocentric	people	14.80	**70.42**	68.87	61.41	100.0
average		16.99	**72.42**	71.29	69.22	100.0

time) is coupled with the improvement in throughput, showing a reduction of 89.4% on average. We especially note that network tra c has improved even if throughput had a threefold improvement.

From table 6.4, all network tra c values obey the bounds, and the bounds are quite tight, proving their usefulness.

6.3 Accuracy

Table 6.5 shows the mean Intersection over Union (mIoU) of various experiment settings. The mIoU of *every* frame (key and non-key frames) is averaged to show that the student can leverage temporal coherence to accurately perform inference on non-key frames. Note that all accuracy values are evaluated against the teacher (Mask R-CNN) output, which is why the naive approach always achieves perfect accuracy. However, we emphasize that the LVS dataset has been labelled with the Mask R-CNN. Thus, in our case, we are measuring the accuracy

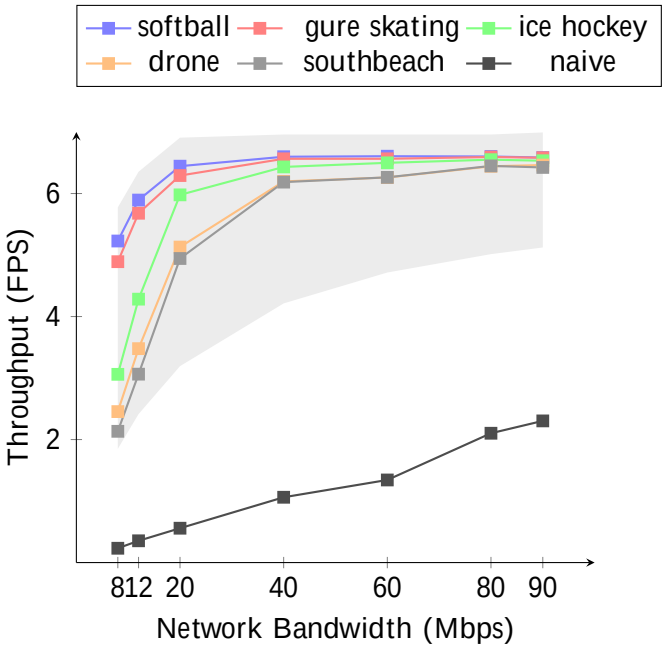


Figure 6.1: Network bandwidth and system throughput

against the label in e ect.

First, to show the need for shadow education, we run the pre-trained student on every frame without any supervision from the teacher (denoted as Wild). As expected, its accuracy su ers greatly, approaching the accuracy of random guessing. This is because the student is too small to generalize to all kinds of scenes.

Next, we show the e ect of shadow education. Recall that the mobile device receives the updated weights in a non-blocking fashion, mitigating the e ect of delays in network transfer. Thus, we measure accuracy when there is the least delay (1 frame, P-1) and the most (8 frames, P-8). ShadowTutor approaches the accuracy of the teacher with a student 100x smaller, proving the e ectiveness of knowledge distillation. Moreover, asynchronous inference hardly hurts accuracy, showing that slightly outdated weights are still useful due to temporal coherence.

Lastly, we compare partial distillation (P-1) and full distillation (F-1). Overall, partial distillation is more accurate. When partial distillation is better, it outperforms full distillation signi cantly, thereby providing an overall stable level of accuracy.

6.4 Robustness to Network Conditions

Fluctuations often happen during network communications between the cloud data center and the client. Thus, in this experiment, we investigate the e ect of reduced available network

bandwidth. Specifically, we set the bandwidth of the system to 90, 80, 60, 40, 20, 12, and 8 Mbps, and examine the throughput of the system.

Figure 6.1 shows the change in throughput against network bandwidth for ShadowTutor and naive offloading. For ShadowTutor, we selected five video streams with different key frame proportions; softball has the least key frames (1.72%), and southbeach (street CCTV) has the most (12.4%).

The throughput of naive offloading decreases immediately in the face of low network bandwidth because it has no mechanism to mitigate the increase in network latency. On the contrary, the throughput of ShadowTutor remains remarkably stable until 40 Mbps, which is half of the original bandwidth. For videos that have a small proportion of key frames, throughput is retained even until 20 Mbps, since network latency takes up only a small fraction among all latency components. Videos with more key frames lose throughput more quickly, but only by 3x even if the network bandwidth is 10x narrower.

The region colored in gray represent the throughput bounds. All throughput values obey the bounds. Especially, for low bandwidth settings, network latency dominates among all latency components, reducing the variation in throughput brought about by the degree of concurrency supported by the mobile device.

ShadowTutor's robustness to the reduction in network bandwidth comes from asynchronous inference. In effect, as long as the network latency is shorter than the inference latency of `MIN_STRIDE` many frames, ShadowTutor can hide the network latency almost completely. However, when the network latency is longer, the reduction in network bandwidth begins to take a more direct impact to the system's throughput since asynchronous inference can no longer serve as a buffer.

6.5 Feasibility of Real-Time Inference

Finding promise from 25–30 FPS videos, we test ShadowTutor with videos with less temporal coherence. Specifically, for every video, we re-sample the frames such that all videos have an FPS of 7. Thus, by matching the input video's frame rate with ShadowTutor's throughput, we simulate the *real-time inference* of frames fetched from the mobile device's camera.

Table 6.6 shows the mean IoU and key frame proportion of the first 5000 frames of the re-sampled videos. Surprisingly, even if the time distance between adjacent frames is elongated by four times, ShadowTutor yields an average accuracy drop of less than 6%p and key frame

Table 6.6: Mean IoU and key frame ratio for 7 FPS videos. Digit (1 or 8) = number of frame delays before receiving updated student weights. Key frame proportion is in %.

Camera	Scene	Partial-1	Partial-8	Key frame
xed	animals	62.72	61.86	6.59
xed	people	80.44	80.08	1.97
xed	street	63.78	62.51	8.9
moving	animals	68.63	66.78	4.84
moving	people	73.66	72.91	4.15
moving	street	48.92	46.99	12.34
egocentric	people	67.57	66.09	5.44
average		66.53	65.31	6.32

proportion increase of less than 1%p. Therefore, this shows the feasibility of applying ShadowTutor to real-time inference applications. With optimized students, e.g. those that utilize weight quantization/pruning or employ more e cient operations, ShadowTutor will be able to handle higher frame rate videos in real-time, and its accuracy will increase further due to stronger temporal coherence.

Chapter 7

Related Work

We brie y survey works related to extending or replacing parts of ShadowTutor.

Chapter 8

Conclusion and Future Work

In this work, we developed ShadowTutor, a distributed video DNN inference framework that encodes the temporal coherence in the video at hand into the parameters of a small student model through intermittent partial knowledge distillation. Evaluations show its advantages in terms of network tra c, throughput, accuracy, and robustness. Moreover, ShadowTutor achieves this through a simple and elegant split between the server and the client, and without complex DNN partitioning, server-side scheduling, or model-level optimizations tailored to the experiment devices.

While our work focused on developing a distributed framework for video data, ShadowTutor can also be extended to tasks other than video computer vision. That is, there are plenty of *sequence data*, i.e. a collection of *data points* that are temporally coherent, that are handled with DNNs. Examples of such sequence data include speech signals from a single speaker, a sentence that requires translation, or a series of item recommendation requests from a single user. Thus, by exploiting the temporal coherence embedded in various types of sequence data through intermittent knowledge distillation, ShadowTutor has the potential to be extended to any type of sequence data. Upgrading the knowledge distillation process, designing appropriate model architectures for the teacher and the student, and resolving the issues that may arise in the process will serve as a good future research direction.

Reference

[1] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961{2969, 2017.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998{6008, 2017.

[3] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960{ 4964. IEEE, 2016.

[4] Balazs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[5] Mennatullah Siam, Sara Elkerdawy, Martin Jagersand, and Senthil Yogamani. Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*, pages 1{8. IEEE, 2017.

[6] Jerry Liu. Analyze real-time cctv images with convolutional neural networks. `https://developer.ibm.com/patterns/iot-devicesensor-damage-detection-with-edge-analytics/`, Dec 2018.

[7] V. Bazarevsky and A. Tkachenka. Mobile real-time semantic segmentation. `https://ai.googleblog.com/2018/03/mobile-real-time-video-segmentation.html`, Mar 2018.

[8] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615{629, 2017.

[9] Amir Erfan Eshratifar and Massoud Pedram. Energy and performance e cient computation o oading for deep neural networks in a mobile cloud computing environment. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, pages 111{116, 2018.

[10] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 123{136, 2016.

[11] Hsuan-Kung Yang, Tsu-Jui Fu, Po-Han Chiang, Kuan-Wei Ho, and Chun-Yi Lee. A distributed scheme for accelerating semantic video segmentation on an embedded cluster. In *2019 IEEE 37th International Conference on Computer Design (ICCD)*, pages 73{81. IEEE, 2019.

[12] Geo rey Hinton, Oriol Vinyals, and Je Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[13] NVIDIA. Nvidia jetson nano. `https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/`, 2020.

[14] Andrew G. Mobile gpu approaches to power e ciency. `https://www.highperformancegraphics.org/wp-content/uploads/2019/hot3d/mobile_gpu_power_and_performance.pdf`, Jul 2019.

[15] Edgar Gabriel, Graham E Fagg, George Bosilca, Thara Angskun, Jack J Dongarra, Jeffrey M Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, et al. Open mpi: Goals, concept, and design of a next generation mpi implementation. In *European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting*, pages 97{104. Springer, 2004.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024{8035. 2019.

[17] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[18] Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, and Kayvon Fatahalian. Online model distillation for e cient video inference. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3573{3582, 2019.

秒錄

\근 D 오 ô è 0 D 8 해결Ð ì 5 à경Ý t p 성공D 거P t서, ¨ | t ¤ Ð서 Ý성 D 오Ð 대해 ì 5 à경Ý X ″ ` 결과| » ″ 것t ä용 x 관 Ð서 ü © D 고 ˆ ä. t Ð 0 |, t X 근ä @ t |우Ü 서„ \ t ø À X ì 5 à경Ý ″ ` 계° É D 덜´내″) • < \ ¨ | t ¤ X Á대 < \ H } 한 계° 성¥ D ô완하였ä. 그ì나 t ø À가 D Ì D 오 데t 0 ″ ä t ä. ¨ à 프 „ X ô | 네¸워l µ à D µ해 교X하게 t 네¸워l ¬용É t 극도\ •가할 Ð Ì t D È |, ´ Ü ¤ \ t 네¸워l < ¡ 0 X Á i Ð è } 해À 0 L 8 t ä. 그ð t Ð서, t 논8 @ D 오 프 „간X Ü간 4 성D \용하ì 네¸워l €하| t ″ 데Ð 그 X X | T ä. t | 해 t 논8 Ð선 ShadowTutor | ″ 프 „웍D H한ä. t ″ „ ° ô è X경Ð서 ì 5 à경Ý X ″ ` D ä è p, 학Ý à경Ý < \ X 간헐 x À Ý • X 0 • D µ해 네¸워l µ à X Ÿ | 감소Ü ¨ ä. 한, € „ À Ý • X | ″ 0 • D \용하ì, 각 네¸워l µ à X 데t 0 É 한 감소Ü ¨ ä. 구´ < \ ″ ä L과 같ä. 서„ Ð ″ 용É t l고 대€ „ X 데t 0 Ð 대해 ‹ @ 성¥ D 내″ 선Ý à경Ý D, ¨ | t ¤ Ð ″ ‘ À Ì ¬ 데t 0 Ð 8 T 학Ý à경Ý D T ä. t Ä, Ü 8 Ü 8 선Ý ″ ¤ 프 „ Ð 대해, 서„ ″ 선Ý à경Ý X ″ ` 결과| 데t 0 X | ¨ \ ô고 학Ý à경Ý 가 X X | € | È (Ü ¨ ¤ ¨ | t ¤ Ð게 뀐 € „ D 송한ä. t 논8 Ð선 고T È D 오X X ø ` „할D µ해 ShadowTutor X ¨ 과성D … • 하고 • 한ä. ä험 결과, 네¸워l µ à X 데t 0 용É t É균 < \ 95% 감소하였ä. 한, Ü ¤ \ X è Ü간ù ˜ ¬ É t 30 t Á •가하였< p, 네¸워l 대 í í X 감소Ð 대해서도 ˜ ¬ É D À하″ 견고성D ô였ä.

주요어: ì 5 à경Ý, „ ° ″ `, À Ý • X, D 오 X ø ` „할