Cracking open the DNN black-box: Video Analytics with DNNs across the Camera-Cloud Boundary

John Emmons•, Sadjad Fouladi•, Ganesh Ananthanarayanan◇,
Shivaram Venkataraman★, Silvio Savarese•, Keith Winstein•
•*Stanford University,* ◇*Microsoft Research,* ★*University of Wisconsin–Madison*

ABSTRACT

Advancements in deep neural networks (DNNs) and widespread deployment of video cameras have fueled the need for video analytics systems. Despite rapid advances in system design, existing systems treat DNNs largely as "black boxes" and either deploy models entirely on a camera or compress videos for analysis in the cloud. Both these approaches affect the accuracy and total cost of deployment. In this position paper, we propose a research agenda that involves opening up the black box of neural networks and describe new application scenarios that include joint inference between the cameras and the cloud, and continuous online learning for large deployments of cameras. We present promising results from preliminary work in efficiently encoding the intermediate activations sent between layers of a neural network and describe opportunities for further research.

ACM Reference Format:
John Emmons, Sadjad Fouladi, Ganesh Ananthanarayanan, Shivaram Venkataraman, Silvio Savarese, Keith Winstein. 2019. Cracking open the DNN black-box: Video Analytics with DNNs across the Camera-Cloud Boundary. In *2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo'19), October 21, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3349614.3356023

1 INTRODUCTION

The pervasive deployment of cameras has made analytics on their video streams important for applications such as traffic planning and retail store management. Video analytics is thus becoming an important problem in the systems community, with efficient systems being built for processing on live videos [1–3]. These systems are powered by advances in deep neural networks (DNNs) on a variety of computer vision tasks, including object detection and classification [4].

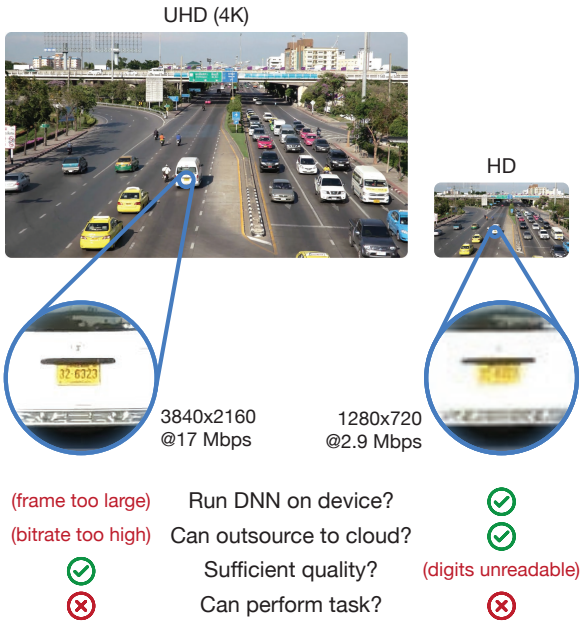

Figure 1: Camera-only and cloud-only solutions. Image resolution plays an important role: The 4K UHD image (left) provides sufficient quality to detect the license plate, but is too big to be processed on camera or sent over a network. The compressed HD version (right) is small enough to be processed on device or sent to the cloud, but leads to inaccurate reading of the license plate.

Despite the rapid advances in the area of video analytics systems, current systems treat DNNs largely as "black boxes". DNNs are generally trained offline with pre-collected data in a datacenter, then deployed in one atomic piece wherever the video can be analyzed, e.g., a datacenter or increasingly cameras in the field, with the latter trend fueled by cameras equipped with compute capacities on board [5, 6].

The treatment of DNNs as atomic units of execution impacts their inference. For DNN inference, either sufficient network links have to be provisioned to get video to the cloud before they are inferred, or the DNN models have to be compressed to fit inside the cameras (at the expense of lower accuracy).

HotEdgeVideo, October 2019, Los Cabos, Mexico

ACM ISBN 978-1-4503-6928-2/19/10…$15.00
https://doi.org/10.1145/3349614.3356023

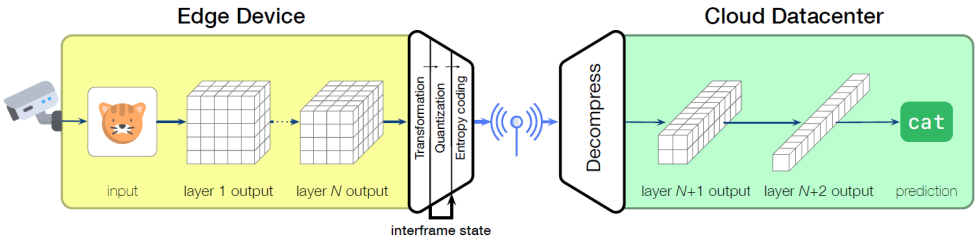

Figure 2: The split-brain architecture, in which the DNN is split into two parts. The edge device evaluates the first part, which consists of the first N layers of the DNN. The resulting activations at the end of this part are compressed and sent over to the cloud. Upon arrival, the cloud decompresses the activations and resumes the evaluation of the DNN. The compression is stateful and exploits the similarities with previously transmitted data to compress the activations further.

In this position paper, we propose a research agenda that involves opening up the black box of neural networks. If the techniques of networked systems and mobile communications can successfully be brought to bear on the signals sent between neurons in a DNN, it would allow for a practical realization of new applications. These include joint inference between the cameras and the cloud given limits on local computation and on communications bandwidth, and continuous online learning for large deployments of cameras.

Split-brain inference. Existing video analytics systems might run the vision DNNs entirely on the cameras (or other edge device) – but will be computationally limited in their resolution, frame rate, and model size. This compromises their ability to recognize small, distant, or short-lived visual features. On the other hand, the camera might instead compress the video and send it to a datacenter to evaluate the DNN. However, compressing the entire source video to meet bandwidth limits will also compromise end-to-end accuracy (see Figure 1).

We propose "split-brain" inference, where video is processed partly on the camera, subject to a limit on computation. Then, intermediate values are transmitted, limited by network capacity, to a cloud datacenter for further DNN inference. While prior work has proposed splitting work between mobile devices and the cloud (e.g., [7]) as well as geo-distributed data analytics [8–10], a key difference in DNN inference is *data amplification*. The size of intermediate data (activations of the DNNs) is far higher than the input size (by up to 4500×), in contrast to mobile and data analytics systems, making it challenging to identify a "split point" for the cloud.

New research is required to develop compression schemes and to train an encoder for DNN intermediates, subject to constraints on local compute, data rate, and overall accuracy. These schemes will have to be optimized for DNN accuracy as opposed to the traditional focus on human viewing (like the H.264 encoder). Preliminary results, with the compression scheme we are developing, show a promising two to three orders of magnitude reduction in the size of the intermediate data. As Figure 3 shows, we also bring the compute-communication tradeoff into an *interesting* region where the data communicated after splitting the DNN is lower than the cloud-only JPEG-compressed baseline and the computation needed on the camera (including compression costs) is less than running the entire DNN; details in §3 and §4.

2 BACKGROUND AND RELATED WORK

Background on DNNs. DNNs achieve state of the art accuracy on many salient computer vision tasks, e.g. objection detection, action recognition, and semantic segmentation, with results surpassing classical computer vision solutions and often matching those by humans. DNNs are composed of many "layers"; the neurons in each layer consume the output of one or more of the layers preceding it. A DNN consists of many different kinds of layers (e.g., convolutions or fully-connected layers) and each layer transforms its input according to model parameters of the DNN and is typically followed by a nonlinear activation function (e.g., ReLU, TanH). The intermediate output produced by each layer is commonly referred to as its *activations*. As the layers successively process the input, they gradually build up high level semantic information until a final prediction is produced. The size of intermediate outputs through the layers is typically *amplified*, i.e., significantly larger than the input image. (More in §3.)

Contemporary DNNs have millions of model parameters (e.g. ResNet50, a relative small DNN, has 25M parameters) and are compute and memory intensive. The large model sizes introduce network constraints when they are shipped to the camera. Modern DNNs require special purpose hardware for fast and power-efficient inference and training.

Related Work.

Mobile DNN Architectures. [11–13] investigate changes to the convolutional layer to reduce its expense. While the proposed optimizations improve compute and memory efficiency, they often come at the expense of accuracy [4].

DNN Model Compression and Specialization. DNN model parameters and activations have conventionally been stored as 32-bit floats (FP32). [14–16] explore approaches to reduce the numerical precision of data and operations in DNNs. While effective, these methods are not sufficient for implementing tasks such as split-brain inference (see Figure 4). Model specialization [1, 17] also attempts to achieve much smaller DNNs or DNNs with small activation sizes. This, however, requires continuous updating of the models to maintain accuracy.

Cloud offloads from mobile devices. Project MAUI was one of the earliest works to explore the idea of offloading compute from mobile devices to the cloud [7, 18–20]. MAUI's focus was on fine-grained code offload to nearby servers with minimal programmer effort. All these systems focused on tasks *without* data amplification; as a result, data transfer during the offload was not a major concern. [21] presents a solution for offloading DNNs for older architectures (e.g., AlexNet, VGG); these DNNs offer a natural split point at their fully connected (FC) layers and do not suffer from data amplification to the same degree as comtemporary DNNs. State-of-the-art DNNs (e.g., ResNet, MobileNet) make use of global average pooling (which reduces the computational costs of FC layers) and have much larger intermediate activations, so splitting before the FC layers is no longer a practical option.

Geo-distributed Data Analytics. Geo-distributed data analytics systems run big data jobs over many datacenters and edge servers connected via wide-area links [8–10]. While these systems balance between compute and communication constraints at each site, they often rely on the property of big data jobs where the amount of intermediate data reduces with the stages of the job (not amplifies, like in a DNN execution).

Compression/network codesign. Past work has focused on codesigning video-compression systems with video applications to optimize *human-perceptible* measures of quality [22]. We suspect similar gains may be had in codesigning video compression with DNN pipelines to optimize arbitrary ML loss functions.

3 SPLIT-BRAIN INFERENCE OF DNNS

System designers today use one of two approaches to meet the demands of resource-hungry DNNs in camera systems.

- *Camera-only*: Fully equip the camera with the compute and memory resources needed to run the DNN locally.
- *Cloud-only*: Outsource the computation of the DNN to a remote datacenter by compressing and sending video frames over the Internet.

These two approaches present extreme points on the design spectrum and we next outline the disadvantages with each of the design approaches. We propose splitting DNN inference across devices in the field and datacenters to allow for greater flexibility in navigating the tradeoffs. This approach is illustrated in Figure 2 and makes it possible to consider the entire design spectrum taking into account tradeoffs between communication, computation and DNN accuracy.

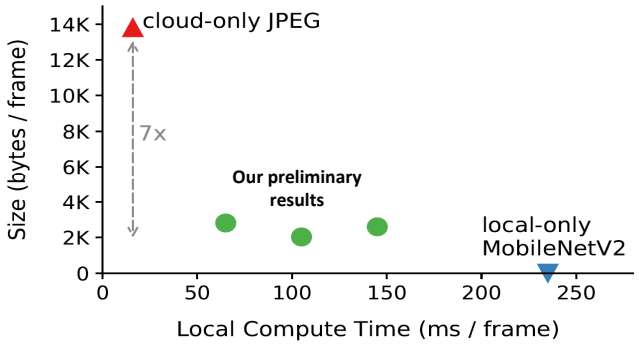

Figure 3: Preliminary results for split-brain DNN inference. We plot the local computation (x-axis, using a single Xeon E5-2687W core) and data rate across the split (y-axis); all points have the same end-to-end accuracy. Shown above are various ways to split execution of the MobileNet DNN [11] to detect and localize objects in a video frame. It can run entirely locally ("local-only"). The device could JPEG-compress the frames and send them to the cloud ("cloud-only"). Our results using non-linear quantization, entropy coding and transforms to the DNN intermediates are promising in the compute and communication costs relative to the two baselines.

Camera-only DNN Inference. The *camera-only* approach requires the cameras to be capable of running a target DNN without assistance using only local compute and memory resources. Modern DNNs, however, require special purpose hardware accelerators (e.g. GPUs, FPGAs [23], TPUs [24]) to run in real time (≥ 30 frames/sec) on HD videos (≥ 720p resolution). Even with recent developments in DNNs optimized for mobile processors, models like SSDLite [11] used for object detection run at less than 5 fps on very low resolution videos; operating on 720p video would make them over 15× slower. Fitting cameras with beefy accelerators would considerably increase their costs and energy consumption.

Compressing models to run entirely on wimpier hardware on the cameras can also lead to degraded accuracy. For example, when the popular object detector, YOLO, is compressed for its "tiny" version, the mean average precision drops by over 50% to achieve the computational savings [4].

Finally, even when it is possible to fully provision the cameras with the resources necessary to run DNNs at high frame-rate and resolution, many applications have bursty

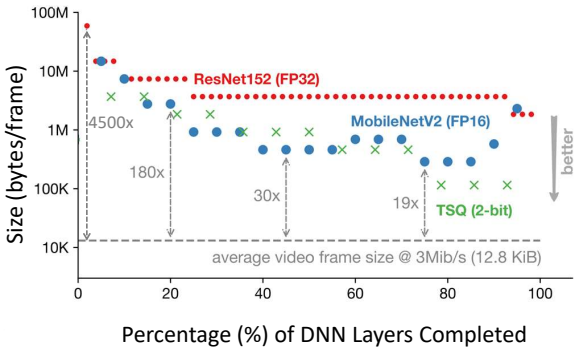

Figure 4: When performing inference on 1280x720 video, the intermediate results produced by the layers of DNNs are 19-4500x larger than the compressed input video. ResNet152 [25] is a conventional DNN and its intermediates are stored with 32 bits per element. MobileNetV2 [11] and TSQ [16] were both designed for mobile/low-memory environments, and use 16 bits and 2 bits per element respectively. Despite the improvement compared to conventional DNNs, the size of the intermediates are too large to feasibly send in place of the compressed video.

usage patterns. Provisioning such cameras with the full hardware capabilities to perform inference locally is wasteful.

Cloud-only DNN Inference. In the *cloud-only* approach, devices compress and send video frames over a bandwidth-limited communications channel (e.g. 3G, LTE, Wi-Fi) to a remote cloud location for DNN inference. This approach eliminates the need for DNN inference to run on cameras, but necessitates provisioning network links with sustained high capacity for HD videos.

Not performing any of the inference work on the camera proportionately increases the processing requirements on cloud infrastructure, which increases the overall costs (in addition to the network costs). For reference, performing object recognition using ResNet152 [25] in the cloud costs $250 per month per camera [26].

Finally, DNNs can only perform high accuracy inference if the compressed video sent from the cameras to the cloud faithfully preserves the salient visual features of the raw video. In environments with poor network link capacities (70% of all mobile connections in 2017 transited 2G and 3G networks [27]), the visual artifacts introduced by compression can reduce the accuracy of DNNs considerably.

Split-brain Inference between camera and cloud. We propose a new approach for DNN inference in intelligent systems, the "split-brain" architecture, where the computation of a DNN is divided between the camera in the field and a cloud datacenter. The objective is to not necessarily make the DNN fit within the camera's constraints but instead rely on the cloud for "overflow" capacity. The intermediate result from applying part of a DNN is transmitted between the two collaborating parties during operation; see Figure 2.

A naïve implementation of the split-brain approach would be impractical; many common DNNs (e.g. AlexNet [28], ResNet [25], YOLO [29], MaskRCNN [30]) represent their intermediates as an uncompressed 3D-array of 32-bit floating point numbers. As a result, the size of the DNN intermediates is considerably higher than the input, a phenomenon we refer to as *data amplification*.

The intermediate output produced by the first layer of ResNet152 is (640×360×64) FP32 numbers is 21× the size of the raw (1280×720×3) RGB frame input. Further, as illustrated in Figure 4, compared to the average frame size when compressed with a standard video codec (as is usually the case) like H.264, the intermediates are up to 4500× bigger. (Note that the y-axis in Figure 4 is log-scale.) We observe two more noteworthy characteristics from Figure 4: (1) The size of the intermediates continue to be high throughout the different layers (x-axis). (2) Even with DNNs that are optimized for weak devices, e.g., MobileNetV2 and TSQ, the size of the intermediates are still one to two orders of magnitude higher than the average video frame size.

Data amplification is a key difference in our problem setup on splitting DNN executions compared to many prior works that split computations between devices and the cloud [7, 18–20]. In mobile applications as well as big data jobs [8–10], the key to the splitting is intelligently identifying the point in the computation pipeline where the amount of data reduces. However, similar techniques are unlikely to be effective in DNNs due to data amplification.

Preliminary Results: We start by applying simple techniques from signal processing including: (a) quantization, where intermediates are encoded using fewer floating point bits (b) transformations like discrete cosine transform (DCT) and KL transform that concentrate the signal in a few components and (c) compressing the results using techniques like Huffman coding and arithmetic coding.

As highlighted earlier in Figure 3, the key result from our preliminary investigation is that we can split the DNN inference such that the amount of communication out of the camera and compute on the camera is less than the cloud-only and camera-only baselines. This makes DNN splitting an amenable solution and we also plot results from splitting at three different points with varying compute and communication costs in Figure 3. We achieve this despite data amplification with our preliminary compression techniques providing *two to three orders of magnitude* (depending on the split-point) gains compared to MobileNet DNN [11].

It is to be noted that prior attempts at reducing the size of DNN intermediates [16, 31] do not compare favorably with existing image codecs and result in the intermediates, *after compression*, still being larger than the cloud-only baseline

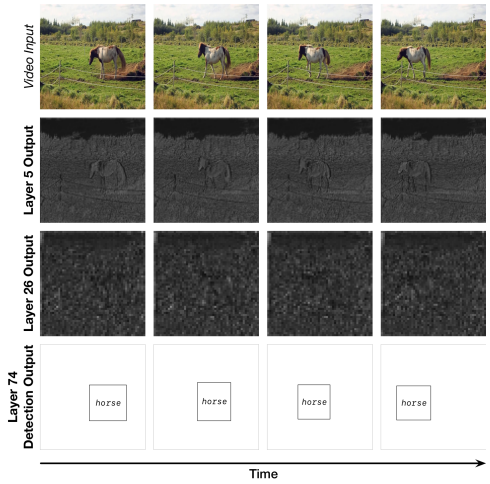

Figure 5: Intermediates of a DNN at frames that are one second apart. Like the input video, there are temporal similarities across DNN intermediates, which presents opportunities for compression.

(i.e., above the red point in Figure 3). Thus, we believe that our result of bringing the intermediate data into the useful trade-off region is a promising initial result.

4 CHALLENGES AND OPPORTUNITIES

We next outline a number of opportunities to further improve the split-brain approach described in §3 and some of the challenges in deploying a such a split-brain model.

1) Encoding across channels. Our results in the previous section looked at applying traditional compression approaches to DNN intermediates. DNN intermediates are typically the outputs from a bank of convolution filters and are hence three-dimensional. For example, in case of the Resnet-152, starting with images of size 224x224 with 3 (RGB) channels, each convolutional layer consists of a number of filters (each 7x7 or 3x3 depending on the layer). State-of-the-art DNNs often have a number of filters at each layer; the third convolutional batch in Resnet-152 has 128 filters leading to intermediate outputs of size 28x28x128. We refer to this third dimension as *channels*.

In our results in §3, compression algorithms are applied to each channel independently. However, as not all feature detectors learn unique features at each layer and often there is considerable similarity *across channels*, which we also verify empirically in our experiments. We intend to leverage this similarity in our encoding algorithms for compression.

2) Inter-frame Encoding. Video codecs like H.264 exploit repeated patterns between frames while encoding videos. We believe similar inter-frame encoding techniques could be applied to DNN intermediates. As the set of active objects changes relatively infrequently over time, DNN intermediates are also similar across frames, likely with higher similarity than the raw frames themselves.

Figure 5 shows an example where we see strong temporal correlations between DNN intermediates of temporally adjacent frames. As deeper layers of a DNN represent coarser features about objects in the frame, there is often more similarity and hence more room for compression of intermediates at deeper layers.

3) Beyond visual encoding. We predict that in the future, most video and images will never be seen by a human. Instead, the data produced by these systems will be consumed solely by algorithms. Current approaches that encode and lossily compress visual data (e.g., JPEG, MP3, and H.264) were designed to optimize figures of merit relevant to a human viewer (perceptual picture quality, smoothness, etc.), and not the high-level objective of a DNN.

We believe that there is potential for new encoding schemes that will be designed for the scenario where data is only consumed by algorithms. For example, if we have a object detector in a traffic camera that is deployed to get real-time traffic volumes, any background data like roads and trees can be eliminated by the encoder during compression if it does not affect the accuracy of the traffic counts. We plan to approach this as an end-to-end optimization problem where the encoding scheme and the object recognition models are jointly trained for high accuracy and compression.

4) Communication/Computation Trade-off. Deploying a split-brain architecture would involve determining the optimal *split point* given a camera deployment and DNN model. While some aspects like the amount of camera compute available can be determined before deployment, other aspects like the wide-area bandwidth available [8] or price of cloud computing resources will fluctuate over time. Thus, techniques to dynamically change the split point in DNN will need to account for compute and network resources currently available. In doing so, we intend to dynamically tweak our compression such that it results in, say, lower data transmission potentially at the expense of slightly reduced accuracy of the DNN. We also plan to draw up prior work in mobile systems in estimating the available resources before splitting [7, 18, 20].

5 ACKNOWLEDGEMENTS

This work was supported by the Toyota Research Institute, NSF grants CNS-1528197 and CNS-1838733 and by Google, Huawei, VMware, Dropbox, Facebook, Amazon, and the Stanford Platform Lab.

REFERENCES

[1] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Noscope: Optimizing neural network queries over video at scale. *Proc. VLDB Endow.*, 10(11):1586–1597, August 2017.

[2] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. Live video analytics at scale with approximation and delay-tolerance. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 377–392, Boston, MA, March 2017. USENIX Association.

[3] Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A. Lee. Awstream: Adaptive wide-area streaming analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 236–252, New York, NY, USA, 2018. ACM.

[4] Yolo: Real-time object detection. https://pjreddie.com/darknet/yolo/.

[5] AWS DeepLens: Deep Learning enabled video camera. https://aws.amazon.com/deeplens/.

[6] Qualcomm Vision Intelligence Platform. https://www.qualcomm.com/products/vision-intelligence-400-platform.

[7] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: Making smartphones last longer with code offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 49–62, New York, NY, USA, 2010. ACM.

[8] Ashish Vulimiri, Carlo Curino, Philip Brighten Godfrey, Thomas Jungblut, Konstantinos Karanasos, Jitendra Padhye, and George Varghese. Wanalytics: Geo-distributed analytics for a data intensive world. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1087–1092, New York, NY, USA, 2015. ACM.

[9] Qifan Pu, Ganesh Ananthanarayanan, Peter Bodik, Srikanth Kandula, Aditya Akella, Paramvir Bahl, and Ion Stoica. Low latency geo-distributed data analytics. *SIGCOMM Comput. Commun. Rev.*, 45(4):421–434, August 2015.

[10] Chien-Chun Hung, Ganesh Ananthanarayanan, Leana Golubchik, Minlan Yu, and Mingyang Zhang. Wide-area analytics with multiple resources. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, pages 12:1–12:16, New York, NY, USA, 2018. ACM.

[11] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[12] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017. cite arxiv:1707.01083.

[13] Lukasz Kaiser, Aidan N. Gomez, and Francois Chollet. Depthwise separable convolutions for neural machine translation. *arXiv*, 2017.

[14] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[15] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

[16] Peisong Wang, Qinghao Hu, Yifan Zhang, Chunjie Zhang, Yang Liu, Jian Cheng, et al. Two-step quantization for low-bit neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4376–4384, 2018.

[17] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. Accelerating machine learning inference with probabilistic predicates. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, pages 1493–1508, New York, NY, USA, 2018. ACM.

[18] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: elastic execution between mobile device and cloud. In *EuroSys*, 2011.

[19] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. *Proceedings - IEEE INFOCOM*, 945-953:945–953, 03 2012.

[20] Mark S. Gordon, D. Anoushe Jamshidi, Scott Mahlke, Z. Morley Mao, and Xu Chen. COMET: Code offload by migrating execution transparently. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 93–106, Hollywood, CA, 2012. USENIX.

[21] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, pages 615–629, New York, NY, USA, 2017. ACM.

[22] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 267–282, Renton, WA, 2018. USENIX Association.

[23] Project Brainwave. https://www.microsoft.com/en-us/research/project/project-brainwave/.

[24] Norman P. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 1–12, New York, NY, USA, 2017. ACM.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[26] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodík, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. Focus: Querying large video datasets with low latency and low cost. *CoRR*, abs/1801.03493, 2018.

[27] The mobile economy 2018. https://www.gsma.com/mobileeconomy/wp-content/uploads/2018/05/The-Mobile-Economy-2018.pdf.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[29] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[30] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[31] Jong Hwan Ko, Taesik Na, Mohammad Faisal Amir, and Saibal Mukhopadhyay. Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms. *CoRR*, abs/1802.03835, 2018.