Accelerate Deep Learning on Mobile Devices

YI ZHAO

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Additional Key Words and Phrases: datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Yi Zhao. 2018. Accelerate Deep Learning on Mobile Devices. 1, 1 (March 2018), 3 pages. https://doi.org/10.1145/1122445.1122456

There are two directions to accelerate deep learning models on resource-constrained devices: model compression and caching. Model compression reduces the computation, storage and memory cost of models. Caching reuse previous results to reduce redundant computation caused by similar input.

1 MODEL COMPRESSION

First, we introduce several techniques to compress DNN models and then the system design to apply these techniques to mobile devices.

1.1 Model Compression Techniques

The classification is from Liu et al..

- Weight Compression. Remove weights that are not important [1, 6, 11].
- Convolution Decomposition. Approximate convolution layers with less-computation-intensive layers [6, 8, 13].
- Special architecture layers. Replace traditional layers with newly designed light-weight layers [10, 12].

1.2 System Design to Apply Compression Techniques

Accommodating these techniques to various needs in mobile scenarios needs extra efforts on system design.

- Balance user-specified performance goals and resource constraints. Different platforms have different resource constraint. Different applications have different performance needs. AdaDeep[14] automatically determines the optimal combination of DNN compression techniques considering the user-specified performance requirements and platform-imposed resource constraints on accuracy, latency, storage, and energy consumption.
- Runtime resource-aware dynamic model compression. The resources available in a mobile device are usually dynamic, for example when users open a new application or close some applications. In order to achieve a higher resource utilization and the optimal trade-off between performance and resource cost, Fang et al. design NestDNN,

Author's address: Yi Zhao.

Manuscript submitted to ACM

a framework that *takes the dynamics of runtime resources into account to enable resource-aware multi-tenant on-device deep learning for mobile vision systems*. NestDNN devises a descendent model with varying capacity by pruning the unimportant filters iteratively.

2 CACHING

Similar inputs are fed into DNN models in many application scenarios:

- **Inside an application:** A continuous video stream has much redundancy between consecutive frames. These works will be discussed in Sec. 2.1.
- **Among different applications in the same device:** Different applications on the same device may process the same video stream at the same time. MCDNN [7] shares the lower layers across DNN models: *the layers of a DNN can be viewed as increasingly abstract representations, so it is conceivable the representations captured by lower levels are shareable across many high-level tasks*. Potluck [5] reuses the results of function calls across different applications. *When an application calls certain processing functions, it first queries the cache for any existing results. The input data are turned into a feature vector, which serves as the key. Then a lookup attempt is made with the key and the function name.*
- **Among different devices:** Same applications can be installed on multiple devices, which generate similar requests when in a similar context. FoggyCache [4] *observes cross-device fuzzy redundancy in mobile and IoT scenarios and eliminate them by approximate computation reuse on edge servers.*

2.1 Caching between Consecutive Video Frames

Consecutive frames usually share large area of similar contents. Processing these frames with DNN models introduces redundant computation. Several works reduce the resource consumption of processing current frame by reusing the computation results of processing last frame.

- **CBinfer [2]:** Pixel-wise reuse and detect reusable regions for each DNN layer. The pixel value at the same position in consecutive frames are compared. Only the affected areas of changed pixels are re-calculated. The reusable regions are detected for every DNN layer.
- **DeepMon [9]:** Block-wise reuse and cascade reusable regions through layers. The frames are segmented into fixed blocks and blocks at the same position are compared by color histogram to decide if they are reusable. The reusable regions are only detected in the raw image and the reusable regions of following convolution layers are derived accordingly.
- **DeepCache [15]:** Flexible block-wise reuse and cascade reusable regions through layers. The algorithm *Diamond Search* is used to find the similar regions between frames, even at different positions. Then these areas are aligned to form larger reusable areas.

The three works are compared in Table 1

REFERENCES

[1] Sourav Bhattacharya and Nicholas D. Lane. 2016. Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM (SenSys '16)*. ACM, New York, NY, USA, 176–189. https://doi.org/10.1145/2994551.2994564

[2] Lukas Cavigelli, Philippe Degen, and Luca Benini. 2017. CBinfer: Change-Based Inference for Convolutional Neural Networks on Video Data. In *Proceedings of the 11th International Conference on Distributed Smart Cameras (ICDSC 2017)*. ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/3131885.3131906

Table 1. Accelerate DNN on Mobile Vision Applications

	Cache Key	Cache Value	Key Match Algorithm	Match position	Layer cascade
DeepMon	Image blocks	Feature map blocks	Histogram based	Fixed	Yes
CBinfer	Pixel or feature values	Feature map values	Threshold	Fixed	No (Cache for each layer)
DeepCache	Image blocks	Feature map blocks	Diamond search and merge	Flexible	Yes

[3] Biyi Fang, Xiao Zeng, and Mi Zhang. 2018. NestDNN: Resource-Aware Multi-Tenant On-Device Deep Learning for Continuous Mobile Vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)*. ACM, New York, NY, USA, 115–127. https://doi.org/10.1145/3241539.3241559

[4] Peizhen Guo, Bo Hu, Rui Li, and Wenjun Hu. 2018. FoggyCache: Cross-Device Approximate Computation Reuse. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)*. ACM, New York, NY, USA, 19–34. https://doi.org/10.1145/3241539.3241557

[5] Peizhen Guo and Wenjun Hu. 2018. Potluck: Cross-Application Approximate Deduplication for Computation-Intensive Mobile Applications. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '18)*. ACM, New York, NY, USA, 271–284. https://doi.org/10.1145/3173162.3173185

[6] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[7] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. 2016. 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 (MobiSys '16)*. ACM, New York, NY, USA, 123–136. https://doi.org/10.1145/2906388.2906396

[8] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[9] Loc N. Huynh, Youngki Lee, and Rajesh Krishna Balan. 2017. DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '17)*. ACM, New York, NY, USA, 82–95. https://doi.org/10.1145/3081333.3081360

[10] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).

[11] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. 2016. DeepX: A Software Accelerator for Low-power Deep Learning Inference on Mobile Devices. In *Proceedings of the 15th International Conference on Information Processing in Sensor Networks (IPSN '16)*. IEEE Press, Piscataway, NJ, USA, Article 23, 12 pages. http://dl.acm.org/citation.cfm?id=2959355.2959378

[12] Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *arXiv preprint arXiv:1312.4400* (2013).

[13] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. 2015. Sparse Convolutional Neural Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] Sicong Liu, Yingyan Lin, Zimu Zhou, Kaiming Nan, Hui Liu, and Junzhao Du. 2018. On-Demand Deep Model Compression for Mobile Devices: A Usage-Driven Model Selection Framework. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '18)*. ACM, New York, NY, USA, 389–400. https://doi.org/10.1145/3210240.3210337

[15] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. 2018. DeepCache: Principled Cache for Mobile Deep Vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)*. ACM, New York, NY, USA, 129–144. https://doi.org/10.1145/3241539.3241563