Larq: An Open-Source Library for Training Binarized Neural Networks

Lukas Geiger[1] **and Plumerai Team**[1]

1 Plumerai Research

Introduction

Modern deep learning methods have been successfully applied to many different tasks and have the potential to revolutionize everyday lives. However, existing neural networks that use 32 bits to encode each weight and activation often have an energy budget that far exceeds the capabilities of mobile or embedded devices. One common way to improve computational efficiency is to reduce the precision of the network to 16-bit or 8-bit, also known as quantization. Binarized Neural Networks (BNNs) represent an extreme case of quantized networks, that cannot be viewed as approximations to real-valued networks and therefore requires special tools and optimization strategies (Helwegen et al., 2019). In these networks both weights and activations are restricted to $\{-1, +1\}$ (Hubara, Courbariaux, Soudry, El-Yaniv, & Bengio, 2016). Compared to an equivalent 8-bit quantized network BNNs require 8 times smaller memory size and 8 times fewer memory accesses, which reduces energy consumption drastically when deployed on optimized hardware (Hubara et al., 2016). However, many open research questions remain until the use of BNNs and other extremely quantized neural networks becomes widespread in industry. `larq` is an ecosystem of Python packages for BNNs and other Quantized Neural Networks (QNNs). It is intended to facilitate researchers to resolve these outstanding questions.

Statement of need

Fortunately, many researchers already publish code along with their papers, though, in absence of a common API to define extremely quantized networks, authors end up re-implementing a large amount of code, making it difficult to share improvements and make rapid progress. Existing tools for BNN research either focus on deployment or require a high learning curve to use and contribute to (Bethge, Bornstein, Loy, Yang, & Meinel, 2018; Zhang, Pan, Yao, Zhao, & Mei, 2019). The API of `larq` is built on top of `tensorflow.keras` (Abadi et al., 2015; Chollet, 2015) and is designed to provide an easy to use, composable way to design and train BNNs. While popular libraries like TensorFlow Lite, TensorFlow Model Optimization or PyTorch focus on 16-bit or 8-bit quantization (Abadi et al., 2015; Paszke et al., 2017), `larq` aims to extend this towards lower bit-widths.

We and many other researchers in the field often encounter major problems when it comes to reproducing existing literature, due to incomplete or even broken code of official implementations. To encourage reproducible research, `larq/zoo` provides tested and maintained implementations and pretrained weights for a variety of popular extremely quantized models (Bethge, Yang, Bornstein, & Meinel, 2019; Hubara et al., 2016; Liu et al., 2018; Rastegari, Ordonez, Redmon, & Farhadi, 2016; Zhou et al., 2016) helping researchers focus on their work instead of spending time on reproducing existing work.

Background: Neural Network Binarization

Binarization of artificial deep neural networks poses two fundamental challenges to regular deep learning strategies, which are all based on backpropagation and stochastic gradient descent (Bottou, 1991; Rumelhart, Hinton, & Williams, 1986): the gradient of the binarization function vanishes almost everywhere and a weight in the set $\{-1, +1\}$ cannot absorb small update steps. Therefore, binarization was long deemed unfeasible.

A solution to both problems was suggested by Hinton, who proposed the Straight-Through Estimator, which ignores the binarization during the backward pass (Hinton, 2012). The viability of this approach was demonstrated by BinaryConnect (Courbariaux, Bengio, & David, 2015) and BinaryNet (Hubara et al., 2016), proving that binarization of networks was possible even for complex tasks.

Since then, the field of BNNs and closely related Ternary Neural Networks has become a prime candidate to enable efficient inference for deep neural networks. Numerous papers have explored novel architectures (Liu et al., 2018; Rastegari et al., 2016; Zhu, Dong, & Su, 2019; Zhuang, Shen, Tan, Liu, & Reid, 2019) and optimization strategies (Alizadeh, Fernández-Marqués, Lane, & Gal, 2019; Helwegen et al., 2019), and the accuracy gap between efficient BNNs and regular DNNs is rapidly closing.

Training extremely quantized neural networks with `larq`

Quantization is the process of mapping a set of continuous values to a smaller countable set. BNNs are a special case of QNNs, where the quantization output x_q is binary:

$$x_q = q(x), \quad x_q \in \{-1, +1\}, x \in \mathbb{R}$$

`larq` is built around the concept of quantizers $q(x)$ and quantized layers that compute

$$\sigma(f(q_{\text{kernel}}(\boldsymbol{w}), q_{\text{input}}(\boldsymbol{x})) + b)$$

with full precision weights $\boldsymbol{w}$, arbitrary precision input $\boldsymbol{x}$, layer operation f (e.g. $f(\boldsymbol{w}, \boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{w}$ for a densely-connected layer), activation σ and bias b. q_{kernel} and q_{input} are quantizers that define an operation for quantizing a kernel and inputs, respectively, and a pseudo-gradient used for automatic differentiation so that operations of the layer can be executed in reduced precision. The source code and documentation be found at github.com/larq/larq and larq.dev respectively.

While `larq` can be used to train networks with arbitrary bit-widths, it provides tools specifically designed to aid in BNN development, such as specialized optimizers, training metrics, and profiling tools.

Summary

The flexible yet easy to use API of `larq` is aimed at both researchers in the field of efficient deep learning and practitioners who want to explore BNNs for their applications. Furthermore, `larq` makes it easier for beginners and students to get started with BNNs. We are working to expand the audience by adding support for deploying BNNs on embedded devices, making `larq` useful for real applications. By building a community-driven open source project, we hope to accelerate research in the field of BNNs and other QNNs to enable deep learning in resource-constrained environments.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Retrieved from https://www.tensorflow.org/

Alizadeh, M., Fernández-Marqués, J., Lane, N. D., & Gal, Y. (2019). An empirical study of binary neural networks' optimisation. In *7th international conference on learning representations, ICLR 2019, new orleans, la, usa, may 6-9, 2019*. Retrieved from https://openreview.net/forum?id=rJfUCoR5KX

Bethge, J., Bornstein, M., Loy, A., Yang, H., & Meinel, C. (2018). Training competitive binary neural networks from scratch. *CoRR*, *abs/1812.01965*. Retrieved from http://arxiv.org/abs/1812.01965

Bethge, J., Yang, H., Bornstein, M., & Meinel, C. (2019). Back to simplicity: How to train accurate bnns from scratch? *CoRR*, *abs/1906.08637*. Retrieved from http://arxiv.org/abs/1906.08637

Bottou, L. (1991). Stochastic gradient learning in neural networks. *Proceedings of Neuro Nimes*, *91*(8), 12.

Chollet, F. (2015). Keras. https://keras.io.

Courbariaux, M., Bengio, Y., & David, J.-P. (2015). BinaryConnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems 28* (pp. 3123–3131). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/5647-binaryconnect-training-deep-neural-networks-with-binary-weights-during-propagations.pdf

Helwegen, K., Widdicombe, J., Geiger, L., Liu, Z., Cheng, K.-T., & Nusselder, R. (2019). Latent weights do not exist: Rethinking binarized neural network optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 7531–7542). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/8971-latent-weights-do-not-exist-rethinking-binarized-neural-network-optimization.pdf

Hinton, G. (2012). Neural networks for machine learning. *Coursera Video Lectures*.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems 29* (pp. 4107–4115). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/6573-binarized-neural-networks.pdf

Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., & Cheng, K. (2018). Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer vision – eccv 2018* (pp. 747–763). Cham: Springer International Publishing. doi:10.1007/978-3-030-01267-0_44

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., et al. (2017). Automatic differentiation in PyTorch. In *NIPS autodiff workshop*.

Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). XNOR-net: ImageNet classification using binary convolutional neural networks. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision – eccv 2016* (pp. 525–542). Cham: Springer International Publishing. doi:10.1007/978-3-319-46493-0_32

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. doi:10.1038/323533a0

Zhang, J., Pan, Y., Yao, T., Zhao, H., & Mei, T. (2019). DaBNN: A super fast inference framework for binary neural networks on arm devices.

Zhou, S., Ni, Z., Zhou, X., Wen, H., Wu, Y., & Zou, Y. (2016). DoReFa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, *abs/1606.06160*. Retrieved from http://arxiv.org/abs/1606.06160

Zhu, S., Dong, X., & Su, H. (2019). Binary ensemble neural network: More bits per network or more networks per bit? In *2019 ieee/cvf conference on computer vision and pattern recognition (cvpr)* (pp. 4918–4927). doi:10.1109/CVPR.2019.00506

Zhuang, B., Shen, C., Tan, M., Liu, L., & Reid, I. (2019). Structured binary neural networks for accurate image classification and semantic segmentation. In *The ieee conference on computer vision and pattern recognition (cvpr)* (pp. 413–422). doi:10.1109/CVPR.2019.00050