# Enabling Binary Neural Network Training on the Edge

Erwei Wang[1], James J. Davis[1], Daniele Moro[2], Piotr Zielinski[2], Jia Jie Lim[1], Claudionor Coelho[3], Satrajit Chatterjee[2], Peter Y. K. Cheung[1] and George A. Constantinides[1]
[1]Imperial College London, [2]Google, [3]Palo Alto Networks
erwei.wang13@imperial.ac.uk

## 1. Introduction & Motivation

The ever-growing computational demands of increasingly complex machine learning models frequently necessitate the use of powerful cloud-based infrastructure for their training. Binary neural networks (BNNs) are known to be promising candidates for on-device inference due to their extreme compute and memory savings over higher-precision alternatives. BNNs represent the ideal class of neural networks for edge inference due to their use of XNOR for multiplication, while their compact weights suit systems with limited memory.

Despite featuring binary forward propagation, existing BNN training approaches perform backward propagation using high-precision floating-point data types—typically `float32`—often making training infeasible on resource-constrained devices. Our understanding of the standard BNN training algorithm introduced by Courbariaux & Bengio [1] led to the following observation: high-precision activations and weight gradients should not be used since we are only concerned with weights and activations' *signs*. In this paper, we present a low-memory, low-energy BNN training scheme based on this intuition featuring (i) binary activations only, facilitated through batch normalization modification, and (ii) matrix multiplication devoid of floating-point operations.

By increasing the viability of learning on the edge, this work will reduce the domain mismatch between training and inference—particularly in conjunction with federated learning—and ensure privacy for sensitive applications. Via the aggressive energy and memory footprint reductions they facilitate, our proposals will enable models to be trained without the network access reliance, latency and energy overheads or data divulgence inherent to cloud offloading.

This extended abstract forms a summary of our recent work, which can be found on arXiv[1]. Our open-source training software and memory and energy estimation tools are available on GitHub[2]. A NumPy-based prototype of our work, targeting Raspberry Pis, is currently under development.

[1] https://arxiv.org/abs/2102.04270
[2] https://github.com/awai54st/Enabling-Binary-Neural-Network-Training-on-the-Edge

## 2. Implementation

We now detail the application of aggressive approximation specifically tailored to BNN training. Further to this, and in line with the observation by many authors that `float16` can be used for ImageNet training without inducing accuracy loss, we also switch all remaining variables to this format.

**High-precision activation elimination via batch normalization approximation.** Batch normalization ordinarily sees channel-wise $l_2$ normalization performed on each layer's centralized activations. Since batch normalization is immediately followed by binarization in BNNs, however, we argue that $l_1$ normalization, free of costly squares and square roots, is good enough in this circumstance. Through further quantization and low-cost retention of layer- and channel-wise means, we transform all activations into their binary form. By doing so, we reduce their memory cost by almost $32\times$ and also save energy thanks to the corresponding memory traffic reduction.

**Binarized weight gradients.** Intuitively, BNNs should be particularly robust to weight gradient binarization since their weights only constitute signs. We therefore store weight gradients in binary format for use during weight update.

**Power-of-two activation gradients.** The tolerance of BNN training to weight and gradient binarization further suggests that activation gradients can be aggressively approximated into power-of-two representation without causing high accuracy loss. We hypothesize that this is more suitable than fixed- or block floating-point formats due to the typically high inter-channel variance of activation gradients.

Assuming that the target training platform has native support for only 32-bit fixed- and floating-point arithmetic, matrix multiplications between binary and power-of-two operands can be computed by (i) converting powers-of-two into `int32`s via shifts, (ii) performing sign-flips and (iii) accumulating the `int32` outputs. This consumes far less energy than the all-`float32` standard.

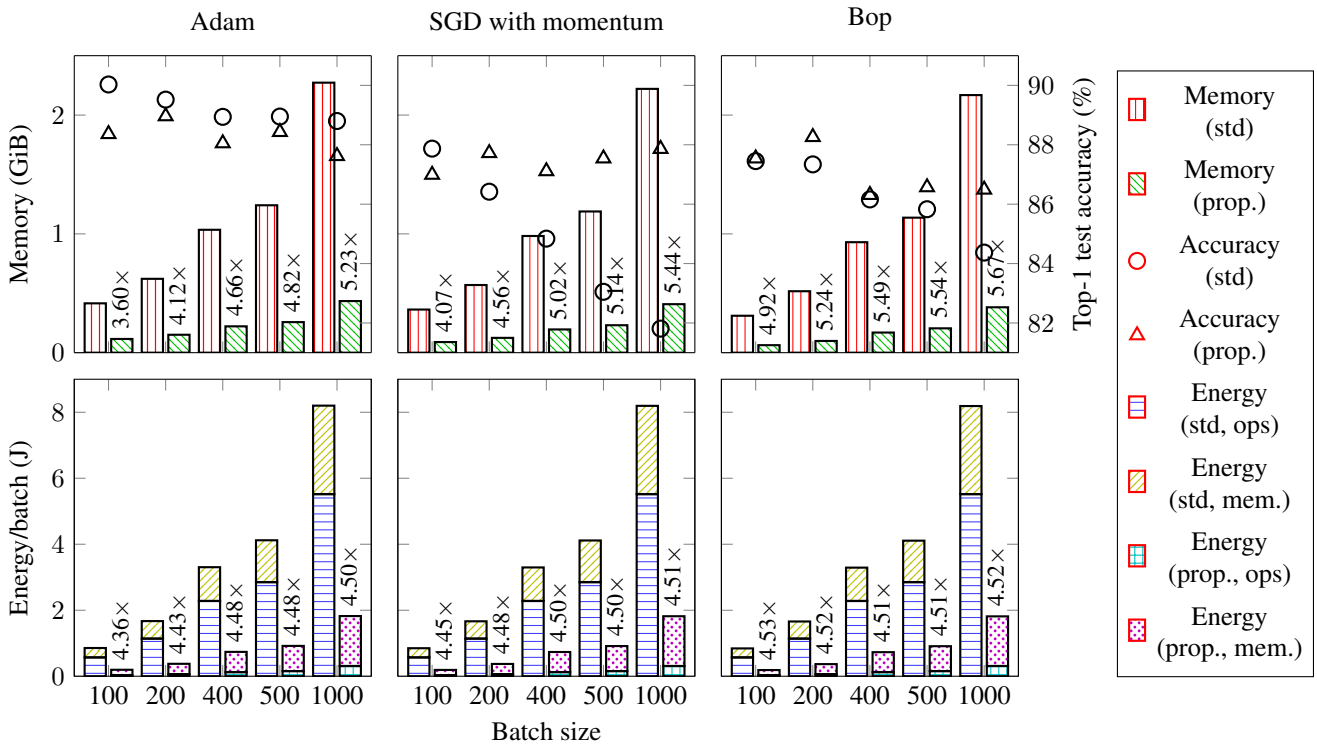


Figure 1. Batch size vs training memory footprint, achieved test accuracy and per-batch training energy consumption for BinaryNet with CIFAR-10 using the standard and our proposed training flows. In the lower plots, total energy is split into compute- and memory-related components. Annotations show reductions vs the standard approach.

## 3. Evaluation

We implemented our BNN training method using Keras and TensorFlow, and experimented with the small-scale MNIST, CIFAR-10 and SVHN datasets, as well as large-scale ImageNet, using a range of network models.

Figure 1 shows the memory footprint savings from our proposed BNN training method for different optimizers and batch sizes for BinaryNet with the CIFAR-10 dataset. Across all of these, we achieved a geomean reduction of 4.86×. For all three optimizers, movement from the standard to our proposed BNN training allows the batch size used to be increased by 10×, facilitating faster convergence, without a material increase in memory consumption. With respect to energy, we saw an estimated geomean 4.49× reduction, split into contributions attributable to arithmetic operations and memory traffic by 18.27× and 1.71×. Test accuracy did not drop significantly due to our approximations, with only Adam inducing small drops (geomean 0.87 pp).

We also trained ResNetE-18, a mixed-precision model with most convolutional layers binarized, to classify ImageNet. In this setting, our approximations delivered memory and energy reductions of 3.12× and 1.17× in return for a 2.25 pp drop in test accuracy.

## 4. Conclusion

In this work, we introduced the first training scheme tailored specifically to BNNs. Moving first to 16-bit floating-point representation, we selectively and opportunistically approximated beyond this based on careful analysis of the standard training algorithm presented by Courbariaux & Bengio [1]. With a comprehensive evaluation conducted across multiple models, datasets, optimizers and batch sizes, we showed the generality of our approach and reported significant memory and energy reductions vs the prior art, challenging the notion that the resource constraints of edge platforms present insurmountable barriers to on-device learning.

## Acknowledgement

The authors are grateful for the support of the United Kingdom EPSRC (grant numbers EP/P010040/1 and EP/S030069/1). They also wish to thank Sergey Ioffe and Michele Covell for their helpful suggestions.

## References

[1] Matthieu Courbariaux and Yoshua Bengio. BinaryNet: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016. 1, 2