Reducing Residual Neural Network Size with Hybrid Binarization by Augmenting Pre-activation Layers

Mani Amani
UCSD
mamani@ucsd.edu

Samuel Brege
UCSD
sbrege@ucsd.edu

Abstract

Binarized Neural Networks (BNNs) offer low memory usage and high inference speeds at a cost of accuracy. However, they often require large, wide architectures (NIN, VGG) in order to provide state of the art accuracy. Furthermore, to fully take advantage of Binary computations, specialized hardware and software is needed. As a result, the benefits of BNNs are somewhat mitigated, particularly for the average user. We propose an alternate, hybrid approach (i.e. both binary and floating-point layers) that allows for smaller networks to save on memory and inference time without overly impeding performance or relying on specialized software, improving versatility.

1 Introduction

Data storage and computational power has been on the rise year after year, a rise that shows no sign of slowing down.. As such, Machine Learning (ML) models are becoming larger and deeper to take advantage of improving technology. However, there are still cases where memory and computational power conservation is, and will continue to be, highly important. Notably, Edge Computing and Internet of Things (IoT). For example, current state of the art Flash Translation Layers use linear regression as optimization models to improve SSD address mapping (Sun et al. (2023)). The computational overhead of current deep learning models and other strong but computationally expensive algorithms is discouraging current implementations from utilizing them. Therefore, we believe exploring different models that reduce the computational overhead of these models is a worthwhile endeavour

Binarized Neural Networks (BNNs) are a subset of Artificial Neural Networks (ANNs) wherein layer weights are constrained to either $+1$ or -1 (Courbariaux et al. (2016)). The effect is a significant reduction in parameter size once trained, theoretically resulting in lower memory usage and a higher inference speed. This is not without cost, however. Firstly, BNNs generally suffer longer training times compared to ANNs of similar complexity. Furthermore, 1-bit binary weights offer a significant reduction in precision relative to 32-bit standard full-precision (FP) weights (i.e. information theory), which can result in a lower potential for accuracy.

That being said, an extremely complex model may not always be necessary: if so, utilizing FP weights could be a waste of resources in those situations. Binarizing common Convolutional Neural Network (CNN) models is an area that has shown promise (Rastegari et al. (2016), Hubara et al. (2016)). These models are capable of achieving accuracies comparable to FP weighted models of the same architecture, whilst greatly reducing memory and inference time.

Another method of reducing memory usage and inference time is through reducing model size (say through removing a layer, or lowering the number of neurons), albeit with a much more direct reduction in accuracy compared to binarization. We are curious if using a tempered approach, with both binarization and model reduction, we might achieve better results than one method alone. We propose a hybrid CNN model using a mixture of Binary and FP weighted layers, hereto named BlinkNet.

2 Methodology

In this paper we implement BlinkNet: a hybrid CNN model based off ResNet architecture. We compare this model to non-hybrid models with the same architecture, as well as to two popular binary models of different architecture: XNOR-Net(Chakraborty et al. (2019)) and BNN (Hubara et al. (2016)). Exact binarization method for BlinkNet is based off (Courbariaux et al. (2016)).

$$w_{ib} = sign\left(w_i\right) \tag{1}$$

2.1 Models and Architecture

In this paper we compare 5 different CNN models. BlinkNet (Hybrid), FP BlinkNet (Full-precision), Binary BlinkNet (Binary), XNOR-Net (Hybrid), and BNN (Binary). The BlinkNet series of models make use of a modified ResNet architecture: we introduced Convolutional pre-residual layers before the residual blocks. This expands on previously developed methods on identity mappings, which implements pre-activation at the beginning of residual blocks to improve training and generalizability (He et al. (2016)). In the hybrid and binary models this convolutional layer is binarized, and left alone in the FP model. All Blink models have FP activation functions, batch normalization layers, and a dropout layer. Three residual blocks were included in the architecture, as well as one linear layer. See Figure 1) for illustration. All Blink models were trained on the CIFAR-10 dataset. The training process utilizes L1 regularization and dropouts in an effort to prevent overfitting and reaching the global minimum. The model was trained using the Adam optimizer.

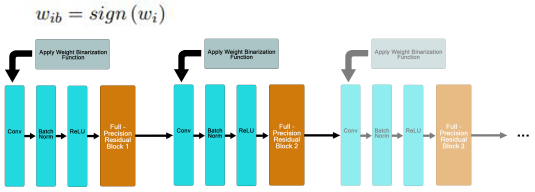

Figure 1: BlinkNet Architecture

The implementation of XNOR-Net we used in this paper makes use of NIN architecture (Lin et al. (2014)). XNOR-Net is not a fully binarized model, somewhat similar to BlinkNet, since it's first and last layers utilize 32-bit parameters. The BNN uses the VGG architecture (Simonyan and Zisserman (2015)) and is entirely binary, aside from pooling and normalization layers. These models were both pre-trained (See code availability). Models' memory usage, inference speed, and accuracy were then compared.

2.2 Data

For our investigation we made use of the well-known CIFAR-10 dataset. It consists of 60000 labeled images, with each 32 by 32 image belonging to one of 10 classes (Evenly distributed at 6000 images per class). 50000 of these images are used in training, with the remaining 10000 reserved for testing. Our primary reason for choosing this dataset was its common use as a benchmark in the ML community (itself a result of the data being in a sweetspot in both difficulty and complexity). It was important to choose a dataset that already had many state of the art pre-trained models to allow for a fair comparison.

2.3 Pre-processing and Data Augmentation

In order to ensure better generalizability of our model, we implemented a data augmentation mechanism that randomly flips and rotates images. We also implement a random crop that crops a 32x32 image from the original dataset to increase the scale invariance. Afterwards, we normalize the data point through a mean gray-scale function. These pre-processing techniques have had significant benefits for our models convergence and overfitting.

2.4 Software and Code Availability

All code was written in Python, making use of the PyTorch package (Paszke et al. (2019)) to run our neural networks efficiently using a GPU. BlinkNet code used in this paper can be found on our github: https://github.com/samuelbrege/Cogs181. In addition, code for the XNOR model was sourced from: https://github.com/jiecaoyu/XNOR-Net-PyTorch.

Code for the BNN model was sourced from: https://github.com/itayhubara/BinaryNet.pytorch (Hubara et al. (2016))

3 Results

Our results show a statistically significant change in both memory usage and speed between both full-precision architecture and the hybridized variant of it. Blink was able to predict the CIFAR-10 dataset with 88.62% accuracy. There was a statistically significant difference between the mean inference time of BlinkNet and the full precision variant ($t(498) = -5.879$, $p < 0.05$) and between Blinknet and the full Binary variant of the model ($t(498) = 7.898$, $p < 0.05$). However, the highest accuracy that could be achieved through the full Binary variant was 27.72%. The full-precision variant was able to achieve 89.00% accuracy being 0.38% more accurate than the BlinkNet. BlinkNet was also significantly faster than FP BlinkNet in the training process ($t(50) = -43.408$, $p < 0.05$). The code was trained and tested on UCSD DSMLP servers using dedicated GPUs.

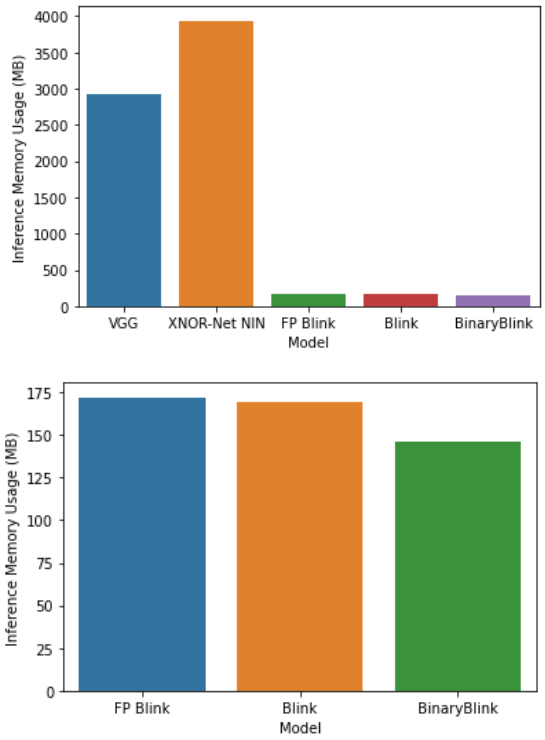

Figure 2: Inference Max Memory Usage

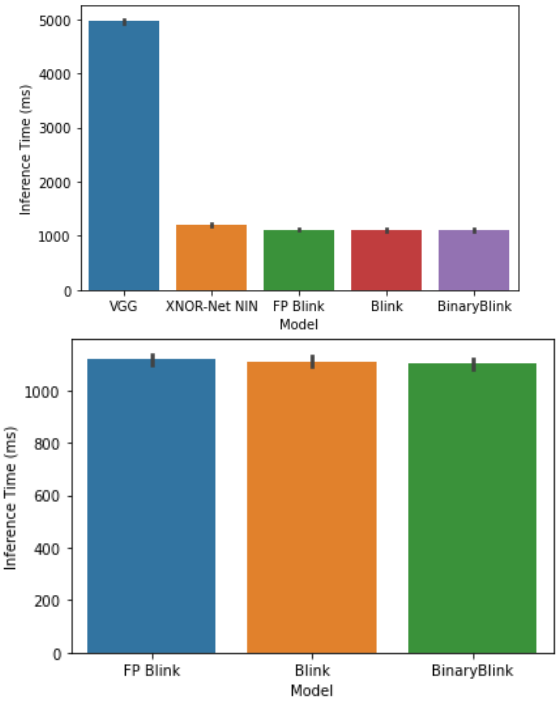

Figure 3: Inference Time

Model	Parameter Memory Usage (Bits)	Accuracy
BNN VGG	3.79×10^7	89.85%
BlinkNet	1.51×10^7	88.62%
FP BlinkNet	1.80×10^7	89.00%
XNOR NIN	1.55×10^6	86.28%
Binary BlinkNet	6.51×10^5	27.72%

Table 1: Parameter Memory Usage and Accuracy

4 Discussion

Since the differences between inference time and memory usage is heavily dependent on software and hardware, we chose to compare theoretical memory usage through memory required to store the entirety of the model's parameters. For the purpose of this paper, we refer to this value as effective parameters. As shown in Table 1 we can see that BlinkNet is able to have competitive accuracy compared to it's counterparts while having fewer effective 1-bit parameters. The XNOR-NIN model is able to have a slightly lower, but nonetheless competitive accuracy, with far fewer effective parameters. However, as can be seen in Figures 2 and 3, we could not achieve that theoretical memory usage and speed compiling on standard software. This may be due to the fact that Binary models need a specialized computing kernel to take advantage of the architecture Xu and Pedersoli (2019). This requirement can be a potential obstacle if the computational environment does not allow these extensions cannot be implemented. In particular, some kernels currently require a GPU, which would not be available in low memory applications. Another potential downside of fully binarized neural networks is the difficulty in their training. Issues such as gradient mismatch from gradient approximation through STEKim et al. (2020) and hyperparameter tuning could require more intensive training compared to it's counterparts so it can reach to it's optimal convergence.

Like everything in deep learning, the architecture and the method of choice is highly dependent on the task at hand and the implementation that it demands. The Blink method specifically looks to serve as an alternative method to optimize computational overhead where specific deep binary networks, such as binary VGG, cannot be implemented due to either hardware or software compatibility. The advantage that Blink has over current XNOR binary neural nets is that it does not need proprietary frameworks in order to take advantage of the bit-wise operations and the speed that XNOR methods bring, giving the model more versatility and compatibility through different hardware and software compatibilities. Furthermore, fully binary networks require large architecture to achieve reasonable results. The effect of using full

binarization in conjunction with relatively small architecture (like the one that underlies Blink), can be seen in Table 1. Despite our best efforts, the Binary Blink model was unable to achieve a higher accuracy than 30%. Therefore, hybrid binarization of pre-activation layers might have implementations in edge cases where a small architecture is necessary.

There is also the concern of overfitting: when the model tunes too well to the training data and suffers during testing. Deep models are often prone to this phenomenon and none of the models we tested were immune. This could be controlled somewhat through L1 regularization, high dropout rate, batch normalization and data augmentation. That being said, compared to FP BlinkNet and a standard ResNet18 model, the Hybrid BlinkNet suffered far less from overfitting once the above methods were added. Throughout the training process, both the training and testing accuracy remained relatively the same. The FP models tested would frequently ended with training accuracies >5% higher than the testing. Full binary models (XNOR-net and BNN) suffer significantly from overfitting and undershooting due to binary activation functions not being differentiable. However, due to time-constraints we were unable to train our own XNOR-Net and BNN models, so we cannot give a fair comparison between BlinkNet and XNOR-Net models' training requirements.

4.1 Why Binarized Pre-activation

When researching hybrid binarized models, we were unable to find a model that integrated pre-activation, so we found it to be a good avenue to experiment and explore further. Though we attempted adding binarization to the residual blocks instead, we noticed a steep drop in performance compared to binarizing convolutional layers outside the residual block. Therefore, we chose to add a Binarized convolutional layer before pre-activation; a pre-residual layer. This way, we could reduce the computational overhead without intefering with the general pre-activation model.

5 Future Work

Further research is necessary to test the hybrid binarzation approach through pre-residual layers on deeper networks and assessing it's effect on accuracy and speed. This study mainly focused on CNNs and their application in image processing using convolutions. We plan to apply the concept of weight binarization and its applications in different types of neural networks and deep learning models such as SNNs and LTSMs. Further research is also required applying these models in IoT devices and moving away from the theoretical memory use decrease and accuracies, gaining better insight on the applicability and versatility of BlinkNet and BlinkNet-like models on resource-limited environments.

Given the relatively small architecture and the existence of significant changes in both usage memory and inference time, further work is warranted in exploring these options in deeper and wider neural networks such as ResNet 50 and Residual Dense Networks (RDNs).Another implementation that is worth investigating having hybridized traditional residual blocks and using full precision pre-activated residual blocks and testing to understand the effects of the skip connection and identity mapping through binary convolutions and it's interactions with FP blocks. Blink's limitation arises from the fact that it could only optimize the algorithms to a certain extent without fully binarizng the algorithm and morphing into lesser versions of current existing architectures. However this creates the opportunity for the designer to choose a position between being fully binarized or using full-precision neural network, by choosing to binarize certain pre-activation layers at will. Further research is required to fully understand and evaluate the effects of selective binarization of pre-activation in deeper residual networks.

6 Conclusion

Blink was able to maintain a high prediction accuracy and decrease computational overhead from it's full precision counterpart. it was also able to achieve a highly competitive accuracy with other BNN's. These results show that the existence and binarization of pre-residual block layers can be beneficial for the purposes of optimizing current existing architectures. The simplicity of Blink gives the design flexibility and versatility given that no proprietary frameworks or third party APIs are need to be able to take full advantage of the model.

7 Bonus Points

We feel we deserve to be considered for bonus points because, though the potential utility is low, we did create something novel. We extensively

studied available literature and learned a great deal during this project, even if our results were not overly exciting. We consciously chose to pursue a project that was more difficult in the hopes of improving our research skills. We implemented multiple models and tried our utmost to keep testing and comparisons scientific.

References

Indranil Chakraborty, Deboleena Roy, Aayush Ankit, and Kaushik Roy. 2019. Efficient hybrid network architectures for extremely quantized neural networks enabling intelligence at the edge.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Hyungjun Kim, Kyungsu Kim, Jinseok Kim, and Jae-Joon Kim. 2020. Binaryduo: Reducing gradient mismatch in binary activation network by coupling binary activations. *CoRR*, abs/2002.06517.

Min Lin, Qiang Chen, and Shuicheng Yan. 2014. Network in network.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition.

Jinghan Sun, Shaobo Li, Yunxin Sun, Chao Sun, Dejan Vucinic, and Jian Huang. 2023. Leaftl: A learning-based flash translation layer for solid-state drives. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS 2023, page 442–456, New York, NY, USA. Association for Computing Machinery.

Xianda Xu and Marco Pedersoli. 2019. A computing kernel for network binarization on pytorch. *CoRR*, abs/1911.04477.