Implementation of Accurate Binary Convolutional Network

E4040.2021Fall.ABCN.Report
Mohammed Aqid Khatkhatay(mk4427), Sanket Sunil Gokhale(sg3904), Abhishek Debashish Sinha(ads2251)
Columbia University

***Abstract*—The project is based on the NIPS 17 paper [1] that brings Binary Neural Networks to CNNs and studies the top 1% and 5% accuracy. The goal of Binary networks is to reduce memory storage requirements, improve power efficiency and reduce floating point operations (especially multiplication in convolution). Binary CNNs also increase the parallelization ability of the computation and can work better with specialized hardware The approximated model accuracy is quite near the actual model for small datasets but drops for larger datasets. A tabular representation or a metric to study the computational gain will be modeled as part of the project. More recent approaches and future scope are discussed and are likely to be explored as the project progresses.**

***Index Terms*—Binarization, Approximation, AlexNet, LeNet, VGGNet**

I. Introduction

The use of the convolutional blocks to detect individual features was first implemented in 1989 and the performance on handwritten digits dataset was a new benchmark in neural networks[1]. However the preliminary CNNs faced big problems due to the data hungry nature of neural networks and additionally the computation power required to scale up to large datasets. This changed with increasing popularity of neural networks in recent years and the CNN as we know it was popularized again by Alex et. al. [2] in their famous 2012 Imagenet problem solution which achieved 84.7% accuracy. This soon became the industry standard for deep classification models in vision.

The common problems associated with CNNs are - the requirement of a large dataset for back-propogation to converge (while humans can classify with just a few reference instances), overfitting to in-sample distribution and not generalizing well to out-of-distribution samples, vanishing gradient problems (due to the large number of layers the gradient update may tend to zero as layers increase), etc. A foundational problem with CNNs is the loss of information due to the large number of max-pooling layers. It does result in translational invariance but at the cost of some highly undesirable generalizations. A significant problem is also created by adversarial examples to which many of the state-of-the-art algorithms are highly vulnerable [3].

The problem that we focus on is reducing the computational complexity associated with CNNs. The biggest computing factor is the floating point multiplication operations during forward pass. Another large computation requirement is the storage of millions of these floating point weight activations. With increasing Machine Learning possibilities in mobile applications, there is a need for algorithms (sometimes called a part of TinyML) that can compute the task (say classification) at hand with the limited computational resources available. We focus on this problem in this project by trying to limit the weight activations to {-1,1} values which significantly reduces the memory requirements. The accuracy of models is less than precision models -the improvement of which is an active research area. Sparser representations also improve the power efficiency of the task.

$$\text{f(x)=Sign}(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

The main paper that this project draws inspiration from this equation deals with the approximation of the full-precision CNN weights with linear combinations of binary weight bases. Note that the convolution operation now requires only addition and subtraction (not multiplication). If activations are binary too, then a bitwise operator is required. Binarization is highly parallelizable, and will only improve in performance on specialized hardware (eg. FPGAs, GPUs, etc.) The main contribution of the paper is the weight approximation algorithm, that binarizes the precision-weights which are also updated during training (but only the binary weights are used at test time). After weight approximation, the activation function used bounds the values to {-1,1} as shown in the figure above.

The idea of a compact representation of neural network weights is not new, with Binary Neural Networks existing since 1990. The interest was revived in 2016 with [4] achieving near state-of-the-art performance on MNIST datasets with highly reduced time complexity and memory consumption. Current industry standards in Binary Neural Networks are [5] [6] [7]that build on these concepts. Apart from binarization, quantization has been implemented with [8] being state-of-the-art in that paradigm. Other dense-Net models exist too which are under active academic research [9]. Integrating these ideas and other recent research (in CNNs) is an extended scope of this project.

[1]https://www.cs.toronto.edu/ kriz/cifar.html

II. Summary of Original Paper

A. Methodology of Original Paper

The original paper first starts out with training a neural network at complete precision. Following this, it leverages M binary filters from B_1 to B_M to approximate the complete weight matrix W. This weight matrix is approximately equal to a linear combination of the aforementioned M filters. Each of the filters is computed using the formula: $B_i = sign(W - mean(W) + u_i * std(W))$ where u_i is the shift parameter. The filters are then fitted to also compute their coefficients α_1 to α_M. Now we have,

$$W = \sum_{i=1}^{M} \alpha_i B_i$$

As the filters are all binary, all operations may now be performed with simple operations such as addition. Using binarized activation functions further enables the usage of basic bit-wise operations for computations. For example, multiplication can now be performed using XNOR and POPCOUNT operations. Activation functions are constrained to a binary {-1, +1} using the transformation function $2I_{h_v(R)\geq 0.5} - 1$ where the bounded activation function $h_v(x) = clip(x + v, 0, 1)$, v is the shift parameter and R is the real-valued activation function which is a linear combination of binarized activation functions. Notice here that when $h_v(x)$ is greater than 0.5, the transformation resolves to 1 and when it is not, it resolves to -1. Batch normalization is used before this to ensure stability and backpropagation is done using a straight through estimator (as if function were identity). The described methodology is shown in Figure 2.

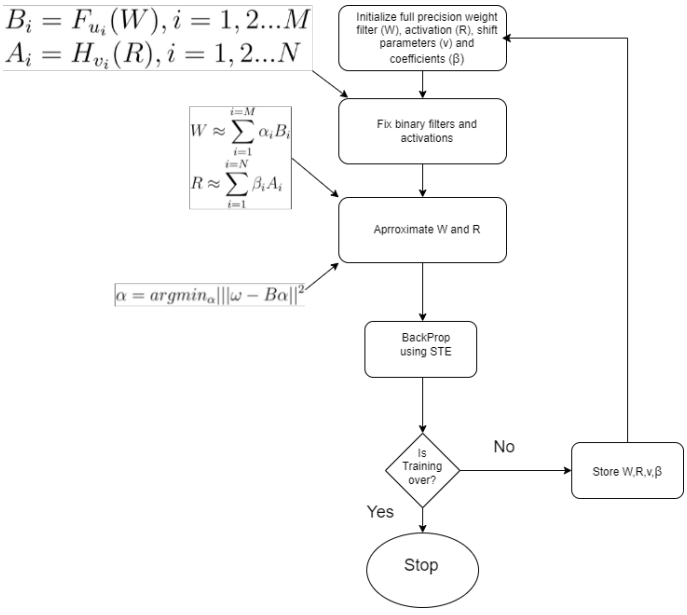

Fig. 1. Original Paper Methodology

The original paper compares the performance of the binarized network against a full precision one for various state-of-the-art architectures including Resnet-18, DoReFa-Net, et al. They perform a space exploration of various configurations of M and N values as model parameter settings for optimal hyperparameter tuning.

B. Key Results from Original Paper

The key finding of the paper is that convolution neural networks with binary filters and full precision networks give highly comparable performances and prediction accuracy's. Binarization was performed in two ways - of the weights, and of the activation functions. On using full precision activation's, it can be noted that increasing the number of binary filters M increases model performance. Furthermore, the converse is also true. On keeping filters the same, increasing the number of binary activations N also increases model performance.

	BWN	M = 1	M = 2	M = 3	M = 5	M = FP
Top - 1	60.8%	62.8%	63.7%	66.2%	68.3%	69.3%
Top - 5	83.0%	84.4%	85.2%	86.7%	87.9%	89.2%

TABLE I
Paper Experimentation Results

III. Methodology

A. Objectives and Technical Challenges

The first objective of our project was to show that the binarization of convolutional layers can be implemented to achieve a good approximation for corresponding convolutional layers. To achieve this objective, we first trained a simple CNN with 2 Conv2D layers and 3 Dense layers. We then trained a neural network with the same number of layers replacing the Conv2D layers with BinConv2D layers(BinConv2D layer was our implementation of the custom Binarized Convolutional Layer). Also, we trained another model with the custom layers, the only difference being the reduced number of binary filters. After successfully compiling these models and getting favourable results(Discussed in Results Section), we continued further with our experiments.

B. Problem Formulation and Design Description

The project will implement the Binary CNN from scratch inspired by [1]. The model architecture implemented in [1] (called as the Approximate Binarized CNN) using the ResNet-18 model with images resized to 224 by 224 prior to training. Top 1% accuracy is modeled. We use a smaller CNN model (with a toy dataset) that we create to simplify the implementation from scratch. We compute the size and computation time of our approximate (binarized) CNN model as opposed to a precision-weighted (fully weighted) CNN model. Accuracies are compared and configuration space is explored. The general methodology can be summarized:

1) Build a model with full precision weights:
   ResNet-18 model with L2 regularization and Batch Normalization is instantiated in the paper. We use a fully-weighted CNN with 2 Convolution layers and 3 dense layers as shown in Figure (1).

2) Develop a Weight Approximation paradigm :
The paper [1] employs a linear combination of M binary filters to learn the real valued weight matrix W. The dimensions of B and M are the same. Due gradient mismatch, backpropogation is not used directly. Instead, the high-precision weights are assumed to have a non-sparse Gaussian distribution. Hence the mean and standard deviation are used to calculate the binary weight approximations. Here a shift parameter is used. Now it is a simple linear regression algorithm that can be solved via backpropogation.
For our methodology, we implement full precision approximate Alexnet, LeNet5 and VGG16 along with proposed binarised and ternerized models of the same.

$$B_i = F_{u_i}(W) := \text{sign}\left((W - \mu) + u_i\sigma(W)\right)$$
$$\min_{\alpha,\beta} J(\alpha,\beta) = \|w - B\alpha\|^2, \text{ s.t. } B_{ij} \in \{-1, 1\}$$

3) Setting the activation function:
The activation function h used here is a simple clipping function with input x and shift parameter v. The shift parameter is used to further separate values that are close in high precision weight vector space.

$$h_v(x) = \text{clip}(x + v, 0, 1)$$

The real valued binary activation function is then transformed to a binary activation function.

$$H_v(R) := 2\mathbb{I}_{h_v(R)\geq 0.5} - 1$$

The forward and backward passes are straightforward now and can be represented in matrix addition and subtraction operations. It actually involves a Hadamard product but due to the nature of binary weights we do not have to compute the multiplication.

$$\text{Forward: } H_v(R)$$
$$\text{Backward : } \frac{\partial c}{\partial A} \odot I_{0\leq R-v\leq 1}$$

4) Test the approximate model using binary weight approximations:
The binary-weight model trained on the trained high-precision weights of the model are used on validation images and classification accuracy is studied. We also model the amount of memory consumption saved as a result of binary test weights.

IV. Implementation

A. Data

Parent paper [1] uses the ImageNet [2] dataset. We use a toy-dataset, namely, MNIST digits [10] for purposes of the project. The MNIST digits dataset is a collection of grayscale images of numbers from 0 to 9 often used for benchmarking prototype models. The MNIST dataset has around 60,000 training and 10,000 test examples. The image size is [28,28,1] and this follows since the images are grayscale. The code however is directly scalable to color images.

B. Deep Learning Network

Our proposed methodology architecture consists of one of our proposed model or state of the art binarized, ternarized and full precision approximate model consisting of convolution, activation, and other Keras API functionalities. We perform experiments by varying the binary and ternary filters and compare their accuracies and weights.

C. Software Design

In order to be efficient with memory and compute, the proposed design uses (Modified National Institute of Standards and Technology database) MNIST dataset, loaded through tensorflow datasets library. We do not perform any augmentation since we have to benchmark on an ideal scenario which is the basic training. Our goal is also not to improve the training and testing accuracy but to make the overall model efficient. Binarization and Ternarization

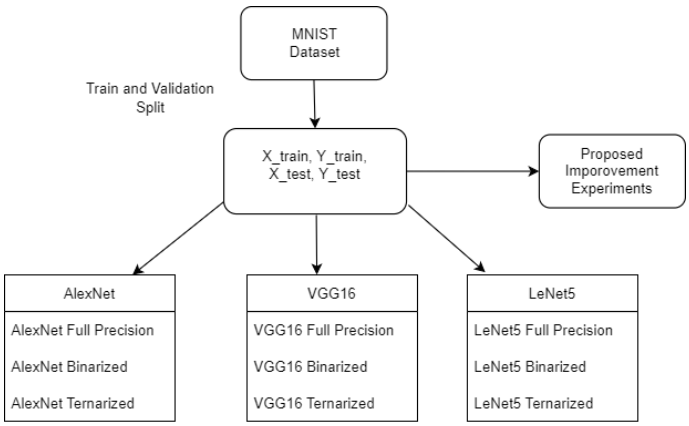

Fig. 2. Our Architecture

V. Results

A. Project Results

After succesfully setting up and comparing our base model with a 2-Conv2D layered Neural Network, we tested out how binarization as well as ternarization of Convolutional layers affects their performances

Architect	Training Times (s)	Accuracy
AlexNet Full Precision	131	0.9719
AlexNet-Binary	312	0.9353
AlexNet - Ternery	219	0.8479
VGGNet Full Precision	351	0.9816
VGGNet-Binary	396	0.9439
VGGNet - Ternery	460	0.9555
LeNet5 Full Precision	16	0.5528
LeNet5-Binary	21	0.7566
LeNet5-Ternery	18	0.6565

TABLE II
Our Model Results in terms of training time and accuracy

The information for Table III was compiled for a 2-layered convolutional neural network(MyModel in our repository). As

Architecture	Weights
Model1 Full Precision	239 KB
Model2 Binary	10KB
Model3 Binary Filter Reduced to half	2KB

TABLE III
MODEL FILE SIZE AS A FUNCTION OF BINARY FILTERS

expected the binarized neural network is much lighter than the normal CNN. Also, the binarized neural network with half the binary filters is exponentially smaller than the binarized neural network due to the squaring of number of operations that are carried out with the addition of each filter in the approximate layer. Table II also shows expected results; binarized neural networks require more time to train due to an increase in number of binary operations.

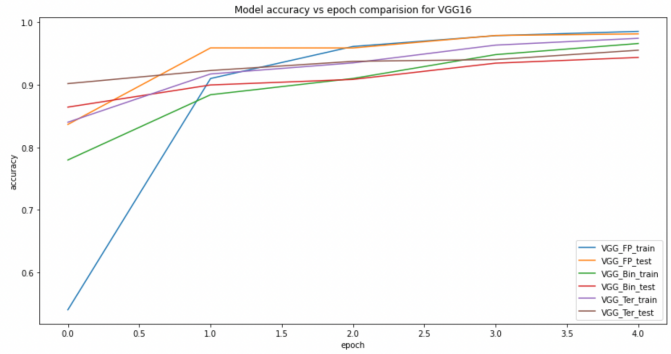

Fig. 3. VGG-16: Accuracy vs Epoch

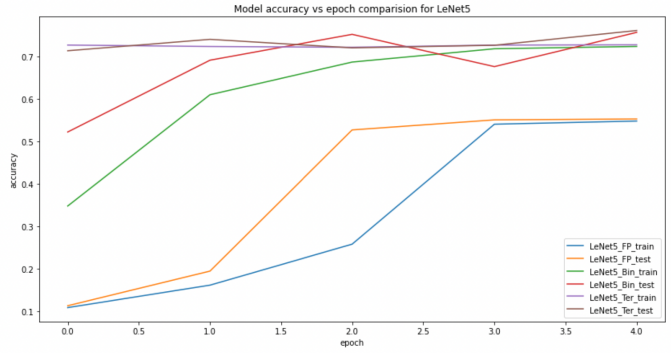

Fig. 4. LeNet-5: Accuracy vs Epoch

B. Comparison of Results

For the Alexnet model (Fig 5.) we observe that the model in the paper performs better than our proposed Binary and Ternary Models, even with the same number of trainable parameters. For the Lenet5 model (Fig 4.) we observe that our binarized model performs better than the model in the paper. For the VGG16 model (Fig 3.) we observe that the model in the paper performs slightly better than our proposed Binary and Ternary Models, even with the same number of trainable parameters. There was slight improvement in our proposed

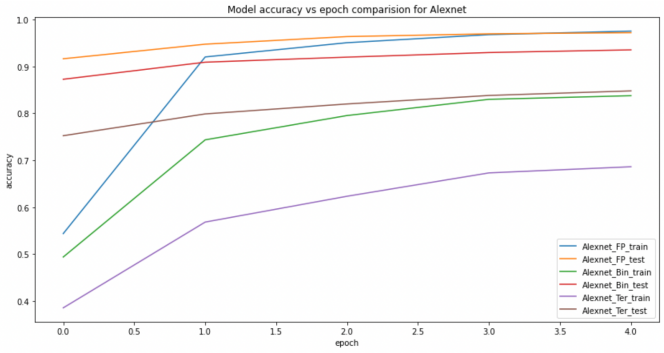

Fig. 5. AlexNet: Accuracy vs Epoch

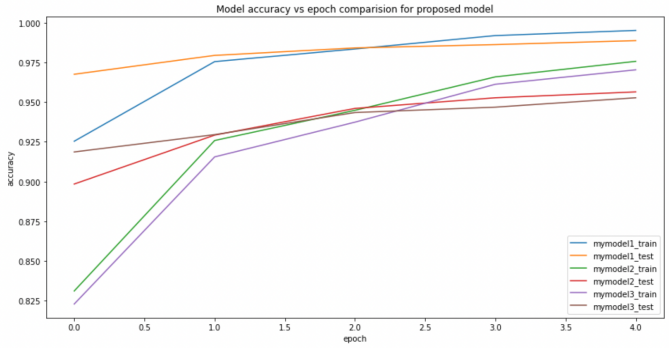

Fig. 6. My Model: Accuracy vs Epoch

model in terms of Letnet5 model which is a small model which might be due to the fact that it is a small architecture and not very deep.

C. Discussion of Insights Gained

The accuracies of the binarized convolutional neural network are as expected less than those of the precision-based CNN from which it was approximated. However, the difference is not too large and with deeper networks (97% accuracy on MNIST with only 2 Convolutional layers) ; it can be presumed that the accuracy will only increase. The accuracy of the Binarized CNN also depends on the number of binary filters used which is a hyperparameter and while increasing the number of filters will improve the accuracy, it will increase the number of computations and memory requirements over the entire network.

Secondly, we chose to compare the space occupied by binary weighted CNN weights as opposed to the fully-weighted CNN. Looking at the drastic drop in size when it comes to the approximate binarized CNN weights, it is fair to say that for a small reduction in accuracy, we have obtained a much lighter weight file. Please note that these are not real or raw tensors that are stored in memory but the tensorflow checkpoints using tensorflow save feature. Hence, the real model and weights will occupy even less memory.

It is fair to note that the biggest contribution of [1] is to reduce (even eliminate) most matrix multiplication operations and simply use bitwise operations. [1] tested

their model (approximated using the vastly superior ResNet 18 architecture) and obtained faster computation times and competitive classification accuracy (top 1found the validation runs of the Approximate Binarized CNN to take more time than the fully weighted models, we have used Google Colab notebooks for implementation that use CPU for computation. [1] has tested the algorithm on highly parallelized hardware (ASICs and FPGAs) which definitely speed up computation time for bitwise operations. [1] also talks about the reduction in power consumption using this paradigm.

VI. Future Work

The project can be extended to implementation of current state-of-the-art models [5] [6] [7]. The project can involve quantization models along with the binary CNN model and the tradeoff between increased accuracy and increased computation cost. An interesting direction of the research is the XNOR-Net [11] which claims to provide a 58% speedup while reducing network size by 32%. Ensemble models [10] An obvious direction of future research is the application of compact networks in object detection (an area of active research). Finally, TinyML applications can be studied in detail to further explore areas where a binary representation will be useful.

VII. Conclusion

Implementation of a Binarized and Ternerized Convolutional Neural Network was studied and explored in this project. We could test out some of the additional claims made in the original paper and explored it further. We see that a binarized and ternerized model certainly reduces the memory requirement of storing network weights to a good extent but not as good as an approximated model. The accuracies obtained are competitive with fully precision accuracy models which implies that we can look seriously into these architectures for model deployment. Another thing we observed was that in the model in the paper is not good for smaller networks like LeNet5 and becomes expensive computationally and physically.

VIII. Acknowledgements

We want to thank Professor Zoran Kostić, as well as the TAs in this course for their continuous help and support this semester.

IX. Appedix

	mk4427	sg3904	ads2251
Last Name	Khatkhatay	Gokhale	Sinha
Fraction of (useful) total contribution	1/3	1/3	1/3
Model Architecture	LeNet5 Architecture	VGG-16 Architecture	AlexNet Architecture
Miscellaneous	Binary helper functions	Visualizations and Parameter tuning	Ternary helper functions
Documentation work	Introduction and Methodology	Results	Original methodology and Conclusion

TABLE IV
Contributions

References

[1] Lin, Xiaofan, Cong Zhao, and Wei Pan. "Towards accurate binary convolutional neural network."
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Conference on Neural Information Processing Systems, 2012
[3] Ian J. Goodfellow and Jonathon Shlens and Christian Szegedy,"Explaining and Harnessing Adversarial Examples"
[4] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830, 2016
[5] . Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., Cheng, K.T.: Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In: Proceedings of the European conference on computer vision (ECCV). pp. 722–737 (2018)
[6] Zechun Liu, Zhiqiang Shen, Marios Savvides, and KwangTing Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In European Conference on Computer Vision, pages 143–159. Springer, 2020.
[7] Xu, W., Chen, Q., He, X., Wang, P., and Cheng, J., "Improving Binary Neural Networks through Fully Utilizing Latent Weights", ¡i¿arXiv e-prints¡/i¿, 2021.
[8] Yang, Jiwei, et al. "Quantization networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
[9] Huang, G., Liu, S., Van der Maaten, L., & Weinberger, K. Q. (2018). Condensenet: An efficient densenet using learned group convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2752-2761).
[10] Zhu, S., Dong, X., Su, H. (2019). Binary ensemble neural network: More bits per network or more networks per bit?. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern
[11] Rastegari M., Ordonez V., Redmon J., Farhadi A. (2016) XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9908. Springer, Cham. Recognition (pp. 4923-4932).