CS230

Investigation on Robustness of Quantized Ternary Weights for Deep Neural Nets

Ying Hang Seah
Department of Computer Science
Stanford University
yinghang@stanford.edu

Behzad Haghgoo
Department of Computer Science
Stanford University
bhaghgoo@stanford.edu

Golrokh Khoddambashi Emami
Department of Computer Science
Stanford University
golkorh@stanford.edu

Abstract

Quantization of Deep Neural Networks into ternary values is a promising model compression technique as ternary weight networks have been shown to achieve similar accuracy as their full precision counterparts. However, little study has been done to investigate the advantages and disadvantages of using ternary weight networks. Thus, we decided to investigate the relative robustness of ternary weight networks and full precision floating point networks against random and adversarial noise as the robustness against noise is a critical aspect of machine learning models in production usage. To evaluate the relative robustness, we used a ternary weight network and a full precision network with ResNet20 architecture on CIFAR-10 dataset. Our results obtained showed that both networks are equally robust against random noise and adversarial noise. Thus, we have successfully demonstrated that ternarizing a model does not cause any significant degradation regarding its robustness against noise.

1 Introduction

Deep neural networks (DNNs) have the potential to power amazing intelligent technology such as drone systems, smart homes and wearable devices. However, deploying deep learning models onto devices with limited memory and computing power is a challenge as deep neural network models require a large amount of memory to store the weights and running the models are computationally expensive. One possible solution is to compress DNNs using weight quantization, thus allowing the model to get an estimated output that can be computed faster with fewer multiplications. [1-9]

Ternary weight quantization is a new approach proposed by Zhang and Liu to have the weights discretized into 3 values: -1, 0, 1. Ternary weight networks (TWN) appears to be a promising compressing model that has comparable performance compared to the full precision (FP) floating point weight networks.

Although there are studies assessing the performance of TWNs on a variety of tasks, there are not much literature examining their advantages and disadvantages for different purposes. Thus, we intend to compare the performance of TWNs and FP networks when subjected to random noise and adversarial noise to evaluate the relative robustness of the models.

CS230: Deep Learning, Winter 2018, Stanford University, CA.

2 Related work

Quantization of weights has long been studied as a method for compressing networks, with some papers in this area dating back to 1994.[8] However, the performance of these implementations on DNNs were not promising up until recently. Deng et al. and Rastegari et al. proposed binary methods for transforming weights to -1 and 1. However, these models suffered significant loss in performance when compared to FP equivalents. A newer approach was proposed by Zhang and Liu to have the weights discretized into three values: -1,0, and 1. By having a stronger expressive ability than binary methods, TWNs have a huge performance gain relative to their binary counterparts. TWNs are able to attain performance that is comparable to FP weights while still being compressed weights that require 16x to 32x less memory usage than FP networks. Further conducted by Zhu et al. have further confirmed the performance of TWN to be similar to FP. [2]

3 Dataset and Model

To investigate the robustness of TWN vs FP, we are using a CIFAR-10 dataset on ResNet20. CIFAR-10 is an image classification benchmark containing images of size 32 * 32 RGB pixels in a training set of 50000 and a test set of 10000. CIFAR-10 was chosen as the dataset as the small image size allows for faster computation and quicker reiterations of the project.

In the TWN ResNet20 Model, all the layers were quantized into ternary values based of a trained FP ResNet20 model except for the first convolution filter layer and the last fully-connected layer. For quantization we used a method similar to the paper [1] and [2], which is as follows.

We treat quantization as a regularizer that places a ternary constraint on all weight vectors of the network. Since ResNet uses ReLU activations, small perturbations of original weights, result in small changes in each layer. In other words, it is plausible to assume close weight matrices generate close activation functions.

Therefore, our objective in ternarizing weights is to find the closest weight matrix to our floating point weights matrix which satisfies our constraints of being ternary.

In other words

$$\alpha^*, W_i^t* = \arg\min_{\alpha, W_i^t} ||W_i - \alpha W_i^t||^2 \tag{1}$$

We will use the following function for ternarizing

$$W_{i_j}^t = \begin{cases} 1 & \text{if } W_{i_j} \geq \Delta \\ 0 & \text{if } |W_{i_j}| < \Delta \\ -1 & \text{if } W_{i_j} \leq -\Delta \end{cases} \tag{2}$$

We will show that there is no deterministic solution to this problem therefore we treat Δ as a hyper-parameter.

Where W is the original weight matrix of layer i and W_i^t is the weight matrix for that layer.

We can expand this to get

$$\alpha^*, W_i^t* = \arg\min_{\alpha, W_i^t}(||W_i||^2 - \alpha||W_i||||W_i^t|| + ||W_i^t||^2) \tag{3}$$

If we define $I_\Delta = \{j | W_{i_j}^t = 1\}$, where W_{i_j} is the j^{th} index of W_i^t (after reshaping it to a vector). Then we can simplify equation (2) to become.

$$||W_i^t|| = \sum_{j \in I_\Delta} ||W_{i_j}||\alpha \tag{4}$$

$$\Delta^*, W_i^t* = \arg\min_{\alpha, W_i^t}(\alpha^2 |I_\Delta| - \alpha \sum_{j \in I_\Delta} ||W_i|| + \sum_{j \in I_\Delta} ||W_{i_j}||^2) \tag{5}$$

And we can see that the α that minimizes this function is

$$\alpha^* = \frac{1}{|I_\Delta|} \sum_{j \in I_\Delta} ||W_{i_j}|| \tag{6}$$

By substituting this in equation (4) we get

$$\Delta^* = \arg\max_{\alpha, W_i^t} (\frac{1}{|I_\Delta|} \sum_{j \in I_\Delta} ||W_{i_j}||) \tag{7}$$

But there is no deterministic solution to this problem [13]. Therefore, this is a hyperparameter of the network. It can be shown that $\Delta = 0.7E(|W|)$ is a good approximation [1].

It is important to note that we didn't rely on having the three possible weights as $\{-1, 0, 1\}$. Any other {negative, zero, positive} tuple is acceptable. Therefore there is no necessity to have the positive and negative values equal. In fact, this is what we do in this paper as shown in Figure 1.

In each step of training, we calculate gradients as normal but with weights that zeros and ones, then change the weights accordingly, ternarize with respect to Δ and calculate the cost.

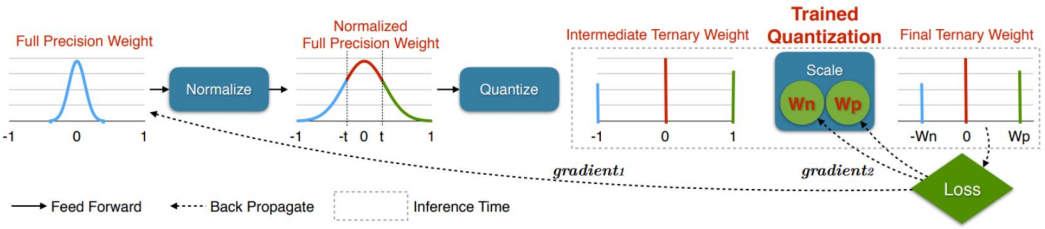

Figure 1: Backpropagation quantization technique (Figure excerpted from Trained Ternary Quantization)

4 Methods

To investigate the relative robustness of the networks, we will evaluate the performance of the network against test images with random and adversarial noise added. On the original dataset with no noise, the FP ResNet20 achieved 91.75% top1 accuracy and the TWN ResNet20 achieved 91.71% top1 accuracy.

To study the robustness of the network against random noise, Gaussian noise, Poisson noise, Salt and Pepper noise, and Speckle noise were added to the test set. Images with noise were then propagated to both the FP network and TWN and the performance were then recorded.

Original Gaussian Poisson Salt and Pepper Speckle

Figure 2: CIFAR-10 images with different forms of random noise

To assess the robustness of the networks against adversarial noise, an image was first passed through the network and multiple iterations of back-propagation was performed to the original image to maximize the loss of the network.

$$\delta = -\alpha \nabla J \tag{8}$$

Where J is the cost function, α is learning rate and δ is the noise matrix which is added to the image.

$$\eta_i = mean_{image_i}(|\delta|) \tag{9}$$

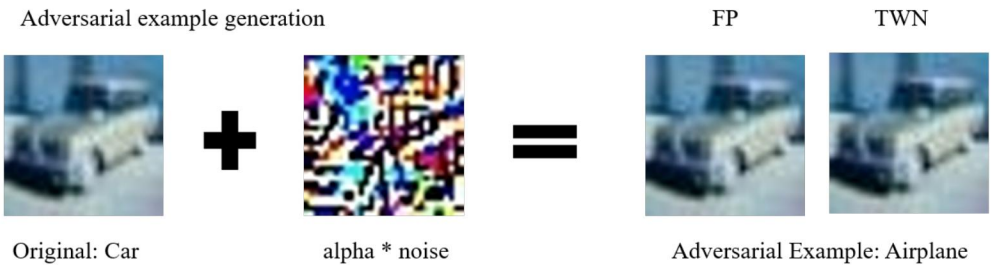

Figure 3: Adversarial example generation

Once the adversarial example is successfully generated, we then compute the mean absolute differences between the original image and the adversarial image as shown in equation (7). A network with a higher mean absolute difference indicates that a network is more robust against adversarial examples as more adversarial noise need to be added to fool the network into generating a false output.

5 Results and Discussion

After experimenting with different types of noises and varying the intensity of the noise added, we managed to obtained interesting results that indicates the performance of the TWN can rival that of FP even under noisy datasets.

Dataset	FP Top 1(%)	TWN Top 1(%)	FP Top 5(%)	TWN Top 5(%)
Original	91.75	91.71	99.76	99.75
Gaussian	19.09	19.93	72.54	72.52
Poisson	18.76	19.58	72.48	72.40
Salt and Pepper	51.09	49.53	79.56	79.11
Speckle	10.34	10.43	52.38	52.59

Figure 4: Top 1 and Top 5 accuracy for FP and TWN against random noise

The Top 1 and Top 5 accuracy of FP and TWN are comparable even after noise was added to the test set. This shows that the robustness of both FP networks and TWN have equivalent robustness against random noise.

Adversarial	FP	TWN
Mean Absolute Difference	73.819317	73.819370

Figure 5: Mean absolute difference of adversarial noise

Since both FP network and TWN obtained very similar values for the mean absolute difference, it shows that there is no significant difference between the relative difficulty to generate new adversarial examples between FP networks and TWN. One explanation for this can be that as stated in section 3, the linear part of ReLU causes close weight matrices to result in close outcomes. Therefore if the scaled matrix αW_i^t is close to W we will get similar calculations in both networks and therefore similar behavior. Apparently, because of its low performance outcomes, a binary compression is not complex enough to make the matrices close enough [9] but TWNs' high accuracy can mean the matrices are close enough. Therefore, it is not surprising that results are similar.

6 Conclusion

As a regularizer, ternary weight networks bring major advantages in memory and computation cost for small devices. In our project, we successfully demonstrated that both FP networks and TWN are equally robust against random and adversarial noise. Our work alongside Zhu et al.'s research

can prove that ternary weight networks can be a reasonable replacement for floating point weight networks without any trade-offs. Thus, compressing a model using ternarizing appears to be a promising technique for machine learning applications on small devices with limited computing power as it does not result in any degradation in its robustness against both random and adversarial noise.

For further research, this project can be extended to more comprehensive classifier networks such as AlexNet to observe how well ternarizing can generalize across various architectures. Another interesting step would be to visualize the activations to see how ternary weight networks and floating point weight networks differ in perceiving inputs with adversarial noise.

7 Contributions

Prior to working on evaluating the relative performance of ternary and full precision networks, we were experimenting with the concept of meta-networks, a network which takes in the input of a classifier and then outputs the weights for another network. From a general perspective, our idea was to give neural nets the power to generalize their knowledge. A vision for our idea was to make a meta-network that can generate a network that plays a game when the parameters of the game have changed. However, after long discussions and experimenting, we decided to switch our project as we discovered that there was very application for such a network. As such, we have detailed the contributions made for the meta-network and ternary weight network on the tables below.

Work	Ying Hang	Behzad	Golrokh
Researching on Topic	✓	✓	✓
Developing the Model		✓	
Literature Reviews	✓	✓	✓
Weight Extraction from DarkNet	✓	✓	✓
ImageNet Crawler	✓		
Weight Alteration Code	✓		
Resizer Code	✓	✓	✓
Writing the Milestone		✓	✓
Editing the Milestone	✓	✓	

Figure 6: Breakdown of member contributions on Meta-Networks Project

Work	Ying Hang	Behzad	Golrokh
Researching on Topic	✓	✓	✓
Literature Reviews	✓	✓	✓
Recreating Ternary Network	✓		
Random Noise Generator	✓	✓	
Adversarial Noise Generator	✓	✓	
Coming up with Evaluation Metric		✓	
Poster Preparation	✓	✓	✓

Figure 7: Breakdown of member contributions on Investigation of Robustness of TWN and FP

8 References

[1] Li, F., Zhang, B., & Liu, B. (2016). Ternary weight networks. arXiv preprint arXiv:1605.04711.

[2] Zhu, C., Han, S., Mao, H., & Dally, W. J. (2016). Trained ternary quantization. arXiv preprint arXiv:1612.0106

[3] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Klingner, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint

arXiv:1609.08144.

[4] Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., & Zou, Y. (2016). DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160.

[5] Deng, L., Jiao, P., Pei, J., Wu, Z., & Li, G. (2017). Gated XNOR Networks: Deep Neural Networks with Ternary Weights and Activations under a Unified Discretization Framework. arXiv preprint arXiv:1705.09283.

[6] Sung, W., Shin, S., & Hwang, K. (2015). Resiliency of deep neural networks under quantization. arXiv preprint arXiv:1511.06488.

[7] Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017). A Survey of Model Compression and Acceleration for Deep Neural Networks. arXiv preprint arXiv:1710.09282.

[8] Khan, A. H., & Hines, E. L. (1994). Integer-weight neural nets. Electronics Letters, 30(15), 1237-1238.

[9] Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, October). Xnor-net: Imagenet classification using binary convolutional neural networks. In European Conference on Computer Vision (pp. 525-542). Springer, Cham.

[10] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

[11] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[12] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).

[13] Hwang, K., & Sung, W. (2014, October). Fixed-point feedforward deep neural network design using ternary weights. In Signal Processing Systems (SiPS). 2014 IEEE Workshop on (pp. 1-6). IEEE.

[14] pytorch. (2018). PyTorch. Retrieved from https://github.com/pytorch/pytorch

[15] czhu95. (2017). Trained Ternary Quantization (TTQ). Retrieved from https://github.com/czhu95/ternarynet

[16] utkuozbulak. (2017). Convolutional Neural Network Adversarial Attacks. Retrieved from https://github.com/utkuozbulak/pytorch-cnn-adversarial-attacks

[17] seahyinghang8. (2018). CS230 Ternary Operators. Retrieved from https://github.com/seahyinghang8/cs230-ternary