Binary Neural Networks

Alasdair Paren[1] **Florian Jaeckle[1]** **Leonard Berrada[1]** **M. Pawan Kumar[1,2]** **Andrew Zisserman[1]**

[1] Department of Engineering Science, *University of Oxford*, [2]*Alan Turing Institute*

{aparen, **florian, lberrada,** pawan, az}@robots.ox.ac.uk

Motivation

Why binary neural networks?

Machine Vision has undergone rapid development during the last 6 years with the state of the art on a range of benchmarks being persistently improved by new techniques, most leveraging convolutional neural networks (CNNs). Large CNNs require graphics processing units (GPUs) to both train and run at inference time because of their computational and memory load. However, the power, cost and space requirements of GPUs prohibit the use of these techniques in many settings. In order to address this issue and improve the applicability of CNNs to real world applications many CNN compression techniques are being developed.

Quantization, where the weights, activations or both within the CCN are quantized from floating point 32-bit numbers to a smaller number of bits, is one of the most promising compression methods.

Our aim was to propose a novel optimization technique for binary neural networks that is easier to use and produces better generalization. We used XNOR-Net as a starting point.

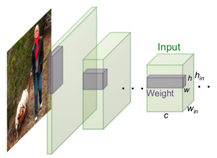
XNOR-Net (Rastegari et al., 2016)

Binary Quantization

XNOR-Net presents an extreme form of quantization where the weights (W) and activations (I) are both binarized. Unlike earlier binary network methods XNOR-Net also preserves magnitude information via a channel wise scaling factor (α).

$$I * W \approx (I_B \oplus W_B)\alpha \qquad I_B = \text{sign}(I) \qquad W_B = \text{sign}(W), \quad \alpha = \frac{1}{N}\sum |W|$$

Here * represents a conventional convolution and $\oplus$ represents a bit wise convolutions computed by using the XNOR and bit count binary operations.



	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Real-Value Weights: 0.12 -1.3 ... 0.41 / -0.2 0.5 ... 0.68	+ , − , ×	1x	1x	%56.7
Binary Weight	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	+ , −	~32x	~2x	%56.8
BinaryWeight Binary Input (**XNOR-Net**)	Binary Inputs: 1 -1 ... -1 / -1 1 ... 1; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	XNOR , bitcount	~32x	~58x	%44.2

Network Structure

In order to reduce information loss when quantizing, full precision weights are used for the first and last layers of XNOR-Nets. Additionally, the following building block is used:

Training XNOR-Nets

Due to the use of the non-continuous sign function in the forward pass of the network it is not possible to train XNOR-Nets using standard stochastic gradient descent or any of its variants. Two tricks are used:

- Approximate the gradients of the sign function using the “straight through estimator” (Matthieu Courbariaux et al.)
- Accumulate the gradients in a second set of weights with full precision and binarize the weights before each forward pass, this can be thought of as modified Mirror Descent

Performance

XNOR-Nets almost match the performance of full precision networks on small scale image datasets such as MNIST and CIFAR10. However, for largescale image data sets, such as ImageNet, their performance is significantly worse ~18%.

Extensions

Many subsequent papers have extended on XNOR-Net; these papers can be grouped into the following areas:

- Increase the number of bits used for quantization, either explicitly or by using multiple binary bases
- Tailoring the network architecture to suit the binary weights and activations, these include increasing the width and add additional residual connections
- Varying methods of including the scaling parameter

Trained Ternary Quantization (Zhu et al., 2017)

Ternary Quantization

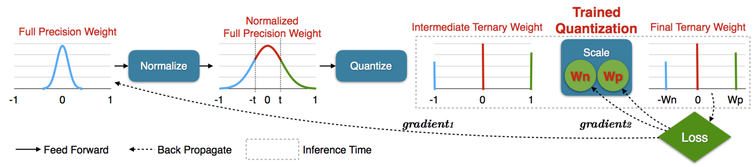
The ternary net proposed by Zhu et. al. reduces the weights of a neural network to ternary values without any loss of accuracy. In fact the ternary networks obtained by Zhu outperform corresponding real-valued networks on both Cifar10 and ImageNet. During training the floating point weights of each layer, $\tilde{w}$, are projected onto the subset of ternary weights using a threshold value $\Delta_l := t \times \max(|\tilde{w}|)$:

$$w_l^t := \begin{cases} W_l^p & \text{if } \tilde{w}_l > \Delta_l \\ 0 & \text{if } |\tilde{w}_l| \le \Delta_l \\ W_l^n & \text{if } \tilde{w}_l < -\Delta_l \end{cases}$$

The scaling coefficients W_l^p and W_l^n are both 32-bit parameters and are trained together with the other weights. The real-valued weights are updated using the gradients with respect to the ternary weights:

$$\frac{\partial L}{\partial \tilde{w}_l} := \begin{cases} W_l^p \times \frac{\partial L}{\partial w_l} & \text{if } \tilde{w}_l > \Delta_l \\ 1 \times \frac{\partial L}{\partial w_l} & \text{if } |\tilde{w}_l| \le \Delta_l \\ W_l^n \times \frac{\partial L}{\partial w_l} & \text{if } \tilde{w}_l < -\Delta_l \end{cases}$$

During inference, only the ternary weights and the real-valued scaling factors are used leading to a 16-fold decrease in the model size.



Deep Frank-Wolfe (Berrada et al., 2018)

Optimizer for deep networks with an optimal step size calculation

We tried to improve the optimization procedure of the binary and ternary networks by replacing Stochastic Gradient Descent (SGD) with the Deep Frank-Wolfe (DFW) algorithm. Unlike SGD, DFW requires only a single hyper-parameter while yielding better generalization performances than most of SGD's adaptive variants. DFW computes an optimal step-size in closed-form at each time-step which is one of its main benefits.

DFW is based on a formulation which linearizes the model $f(I_j)$ and the regularization (ρ) while preserving the loss function ($\mathcal{L}$):

Each step of stochastic gradient decent (SGD) can be thought of as solving the following proximal problem:

$$W_{t+1} = argmin\left\{\frac{1}{2\eta_t}\|w - w_t\|^2 + \mathrm{T}_{w_t}\left[\rho(w) + \mathcal{L}\left(f(I_j)\right)\right](w)\right\}$$

Here $T_{w_t}[\ ]$ is the Taylor expansion of a function around the current point w_t, $f(I_j)$ is the output of the neural network on the j^{th} sample and η_t is the step size. Deep Frank Wolfe (DFW) instead solves a different proximal problem at each step where only the network not the loss function is linearized. This gives a convex problem at each step. The dual problem is solved using Block Coordinate Frank Wolfe which is a batch variant of conditional gradients.

$$W_{t+1} = \underset{w\in\mathbb{R}^p}{\text{argmin}}\left\{\frac{1}{2\eta_t}\|w - w_t\|^2 + \mathrm{T}_{w_t}[\rho(w)] + \mathcal{L}\big(\mathrm{T}_{w_t}[f(I_j)]\big)(w)\right\}$$

DFW for Binary Neural Networks

In order to use DFW to optimize XNOR-Nets the above minimization was modified, to include constraints on the value of w. The resulting proximal problem is a relaxation of the integer program where the weights are constrained to be in the set {-1,1}. This optimization is then used in the place of SGD or is variants in the XNOR-Net optimization scheme.

$$W_{t+1} = \underset{w\in[-1,1]^p}{\text{argmin}}\left\{\frac{1}{2\eta_t}\|w - w_t\|^2 + \mathrm{T}_{w_t}[\rho(w)] + \mathcal{L}\big(\mathrm{T}_{w_t}[f(I_j)]\big)(w)\right\}$$

Results

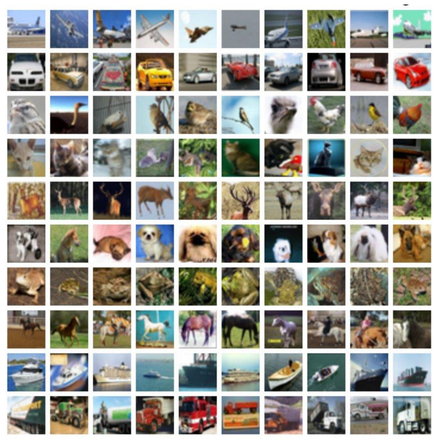
Objective

We wanted to compare DFW against the Adam and SGD optimizers for quantized neural networks, specifically XNOR-Nets and Trained Ternary Quantization Networks. We used the cifar10 data set for our first experiments as it is relatively cheap to train on.

CIFAR-10

Number of classes: 10

Examples: “airplane”, “automobile”, “bird”, “cat”, “dear”, etc.



Model: ResNet20 with XNOR and Trained Ternary Quantization Variations

Loss: SVM

Optimizers: Adam, SGD with momentum, and DFW

Performance

DFW for Binary Neural Networks is currently outperforming Adam for training XNOR-Networks, and is ~1% off current state of the art optimizers for Trained Ternary Networks.

Quantization	Optimizer	% Test Acc
Floating Point	SGD w M	91.94
Ternary Network	SGD w M	90.66
Ternary Network	DFW	89.06
XNOR-Net	Adam	72.89
XNOR-Net	DFW	75.87

Future Work

Further Testing of using XNOR-Nets

We aim to submit this work to CVPR, and hence more thorough testing of our method is required especially on larger datasets such as ImageNet and using a variety of architectures, including large state of the art models such as DenseNets and Wide ResNets.

DFW for Quantized Networks

We’re currently working on generalizing the DFW algorithm to work on all quantized networks. Some quantized networks, such as the ternary net, achieve the same accuracy as the corresponding real-valued networks while still achieving a reduction in memory space at inference time. In practice these might have a wider range of applications than the XNOR-net.

References

- Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, Ali Farhadi, XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks ECCV 2016
- Matthieu Courbariaux, Yoshua Bengio, Jean-Pierre David BinaryConnect: Training Deep Neural Networks with binary weights during propagations Neural Information Processing Systems 2015
- Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. International Conference on Learning Representations 2017
- Leonard Berrada, Andrew Zisserman, M. Pawan Kumar, Deep Frank-Wolfe For Neural Network Optimization, in processigns ICLR 2019