Review - Training Binary Neural Nets through learning with noisy supervision

Rishabh Patra

August 2020

1 Introduction

This paper proposes a new method for training in case of binary Neural Networks. Previous method include hard thresholding the weights of a full precision network ($\tilde{Q} = sign(W)$) where $W, \tilde{Q}$ are the binarized and full precision weights respectively. Such a method does not effectively capture the relationship between the weights. The paper proposes a mapping from the full precision weights (taken as a whole) to the binarized weights. The hard thresholded weights are treated as a noisy auxiliary signal, and an unbiased estimator is proposed to mitigate the influence of the noise. Due to a lack of supplementary materials, some assumptions have been made by me while implementing this paper

2 Approach

2.1 Binary Weight Mapping

In the previous approaches, the weights are binarized using a sign function. Let the weights before quantization be $W \epsilon R^{cxkxk}$ where c is the number of channels and k is the kernel size. The binarize weights $\tilde{Q}$ are then given by $\tilde{Q} = sign(W)$. The feature map before convolution $X \epsilon R^{nxcxhxw}$ are also binarized as $B = sign(X)$a and the forward pass then calculates $Y = B * \tilde{Q}$. Then the loss (CE Loss in this case) is calculated (say l_{cls}) and gradients are calculated via a straight through estimator as $\frac{dl_{cls}}{dW} = clip(\frac{dl_{cls}}{d\tilde{Q}}, -1, +1)$. After training, $\tilde{Q}$ are kept for inference

2.2 Learning via noisy supervision

It is proposed that instead of hard thresholding, a mapping model be used to binarize the weights as: $\hat{Q} = f_\theta(W)$, where θ are the parameters of the function (say a neural network). But we only have the classification loss, hence optimizing θ can be problematic. Say we had $\hat{Q}$, then we could minimize $l(\hat{Q}, Q) = \Sigma(q - \star q)^2$ Now to mitigate the effect of noise and learn from the noisy label - $l(\star q, \tilde{q}) =$

$\Sigma(\star q - \tilde{q})^2$ and the flip probability - $P(\tilde{q} = -1|q = +1) = \rho_{+1}$ and $P(\tilde{q} = +1|q = -1) = \rho_{-1}$ (ρ_{+1} and $\rho{-}1$ being hyperparameters). $\hat{Q}$ is expected to be an unbiased estimator of Q, implying that $E(l(\hat{Q}, \tilde{Q})) = l(Q, \tilde{Q})$. From this, $l(\tilde{q}, +1)$ and $l(\tilde{(q)}, -1)$ can be calculated. Say the auxiliary loss for the i-th layer is $l_i = l(\hat{Q}, \tilde{Q})) = \Sigma l(\hat{q}, \tilde{q})$. The gradients $\frac{dl_i}{d\hat{q}}$ are then calculated. The gradients of the auxiliary loss wrt the full precision weights (W) and the mapping model parameters (θ) can then be calculated as - $\frac{dl_i}{d\theta} = \Sigma_{\hat{q}} \frac{dl_i}{d\hat{q}} \frac{d\hat{q}}{d\theta}$ and $\frac{dl_i}{dW} = \Sigma_{\hat{q}} \frac{dl_i}{d\hat{q}} \frac{d\hat{q}}{dW}$ respectively. The total loss to be trained on is then given by $L = l_{cls} + \alpha l_i$ (α being a hyperparameter). The final weight updates to θ and can then be calculated as $W = W - \eta \frac{dL}{dW}$ and $\theta = \theta - \eta \frac{dL}{d\theta}$ (in both cases, $\frac{dl_{cls}}{dW}$ and $\frac{dl_{cls}}{d\theta}$ can be calculated as in normal neural networks).

3 Experiments

3.1 Assumptions made

The explicit structure of f_θ was not given (since the appendix could not be found). Hence 2 approaches could have been taken -
1. Let f_{theta} be a CNN that takes in one channel, and the output shape (mapped weights $\hat{Q}$) is the same as the input shape (full precision weights W). If there are multiple channels present, then each channel is binarized separately and then stacked together to form the final binarized weight matrix.
2. Let f_{theta} be an MLP NN. The full precision weights(W) are flattened and fed as inputs to the MLP, the number of input and output neurons being the same. The output from the MLP($\hat{Q}$) is then reshaped to the size of W.
In both cases, the output layer is passed through a hard tanh non-linearity to ensure that the values of $\hat{Q}$ are between $(-1, 1)$
Moreover, this mapping approach by the authors is used as a fine-tuning technique. First the model is trained using hard thresholding, and then the mapping approach and the hard thresholding approach can be compared.

3.2 CIFAR-10

Using a batch size of 128, the CIFAR-10 dataset is first used to train the hard thresholding model for 400 epochs and then the finetuning is applied for 120 epochs.

3.2.1 Small Resnet

The results in this case are not upto the papers results. The limits on google colab caused me to reduce the number of output channels for each convolution layer.
Update - A small error in the code was found. The results are not in yet. Baseline training - Test CE Loss = , Test Accuracy =
Linear Mapped finetuning - Test CE Loss = , Test Accuracy =

Baseline finetuning - Test CE Loss = , Test Accuracy =

3.2.2 Full Resnet

The results are not in yet.
Baseline training - Test CE Loss = , Test Accuracy =
Linear Mapped finetuning - Test CE Loss = , Test Accuracy =
Baseline finetuning - Test CE Loss = , Test Accuracy =

3.3 ImageNet