Binary Neural Network

Apprentissage des Réseaux de Neurones avec des poids et activations binaires

Fatou Kiné SOW
Green AI UPPA

January 23, 2022

Overview

1. Introduction
2. Binary Connect : Training Neural Network with weight binarize during propagations
3. Binary Neural Network : Binarisation des poids et activations
4. Résultats

Introduction

- DNN très coûteux en temps de calcul et consommation énergétique
- Besoin d'embarquer les DNN sur des appareils grand public
- Diminution des opérations de multiplication : Binary Neural Network

BinaryConnect

Restreindre les poids à -1 et +1

Déterministe

$$x_b = \begin{cases} +1 & \text{si } x \geq 0 \\ -1 & \text{si } x \text{ sinon} \end{cases}$$

Stochastique

$$x_b = \begin{cases} +1 & \text{avec une probabilité } p = \sigma(x) \\ -1 & \text{avec une probabilité } 1 - p \end{cases}$$

où σ est la fonction "hard sigmoid"

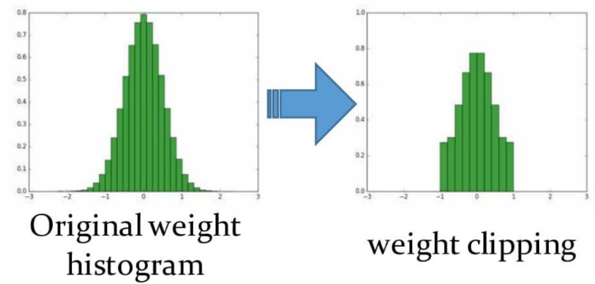
$$\sigma(x) = clip(\frac{x+1}{2}, 0, 1) = max(0, min(1, \frac{x+1}{2}))$$

BinaryConnect : Algorithme

Entrées : minibatch(inputs,targets), w_{t-1}, b_{t-1} and η
Calcul de w_t **et** b_t

1. Forward propagation
   $w_b = binarize(w_{t1})$
   For $k = 1$ to L, compute a_k knowing a_{k1}, w_b and b_{t1}
2. Backward propagation
   Initialize output layer's activations gradient $\frac{\partial C}{\partial a_l}$
   For $k = L$ to 2, compute $\frac{\partial C}{\partial a_{k-1}}$ knowing $\frac{\partial C}{\partial a_k}$ and w_b
3. Update parameters
   Compute $\frac{\partial C}{\partial w_b}$ and $\frac{\partial C}{\partial b_{t-1}}$ knowing $\frac{\partial C}{\partial a_k}$ and a_{k-1}
   $w_t = clip(w_{t-1} - \eta \frac{\partial C}{\partial w_b})$
   $b_t = b_{t-1} - \eta \frac{\partial C}{\partial b_{t-1}}$

BinaryConnect : Algorithme



Binary Neural Network : Approches

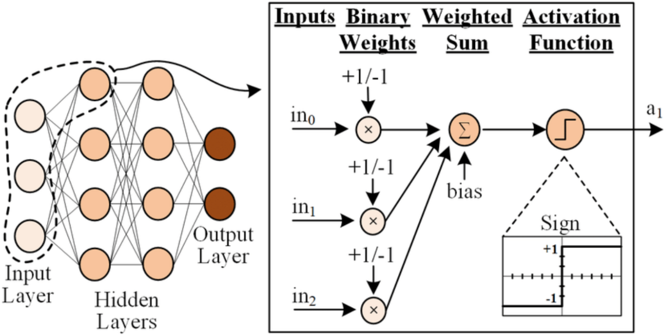


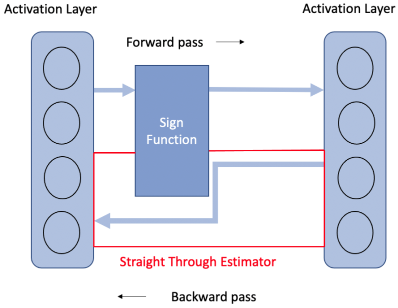
Figure: source Boolean Masking of an Entire Neural Network

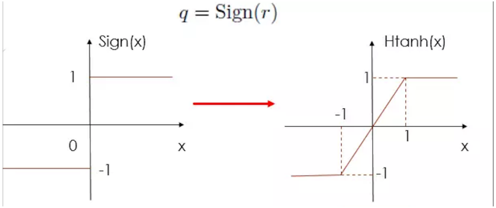
Binary Neural Network : Algorithme

Calcul du gradient des paramètres : Forward propagation

for $k = 1$ to L **do**
  $W_k^b \leftarrow \text{Binarize}(W_k)$
  $s_k \leftarrow a_{k-1}^b W_k^b$
  $a_k \leftarrow \text{BatchNorm}(s_k, \theta_k)$
  if $k < L$ **then**
    $a_k^b \leftarrow \text{Binarize}(a_k)$
  end if
end for

Binary Neural Network : Algorithme





$$g_r = g_q 1_{|r| \leq 1}.$$

$$\text{Htanh}(x) = \text{Clip}(x, -1, 1) = \max(-1, \min(1, x))$$

Binary Neural Network : Algorithme

Calcul du gradient des paramètres : Backward propagation

{Please note that the gradients are not binary.}
Compute $g_{a_L} = \frac{\partial C}{\partial a_L}$ knowing a_L and a^*
for $k = L$ to 1 **do**
  if $k < L$ **then**
    $g_{a_k} \leftarrow g_{a_k^b} \circ 1_{|a_k| \leq 1}$
  end if
  $(g_{s_k}, g_{\theta_k}) \leftarrow \mathrm{BackBatchNorm}(g_{a_k}, s_k, \theta_k)$
  $g_{a_{k-1}^b} \leftarrow g_{s_k} W_k^b$
  $g_{W_k^b} \leftarrow g_{s_k}^\top a_{k-1}^b$
end for

Binary Neural Network : Algorithme

Modification des paramètres

{2. Accumulating the parameters gradients:}
for $k = 1$ to L **do**
  $\theta_k^{t+1} \leftarrow \mathrm{Update}(\theta_k, \eta, g_{\theta_k})$
  $W_k^{t+1} \leftarrow \mathrm{Clip}(\mathrm{Update}(W_k, \gamma_k \eta, g_{W_k^b}), -1, 1)$
  $\eta^{t+1} \leftarrow \lambda \eta$
end for

Binary Neural Network : Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Binary Neural Network : Shift based Batch Normalization

Require: Values of x over a mini-batch: $B = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Ensure: $\{y_i = \mathrm{BN}(x_i, \gamma, \beta)\}$

$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ {mini-batch mean}

$C(x_i) \leftarrow (x_i - \mu_B)$ {centered input}

$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(C(x_i) \ll\gg AP2(C(x_i)))$ {apx variance}

$\hat{x}_i \leftarrow C(x_i) \ll\gg AP2((\sqrt{\sigma_B^2 + \epsilon})^{-1})$ {normalize}

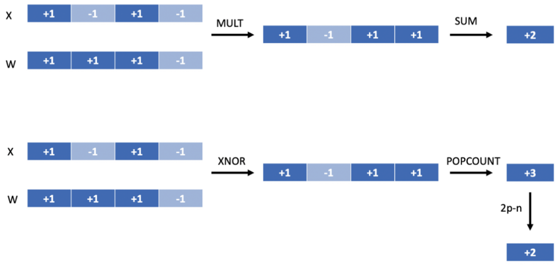
$y_i \leftarrow AP2(\gamma) \ll\gg \hat{x}_i$ {scale and shift}

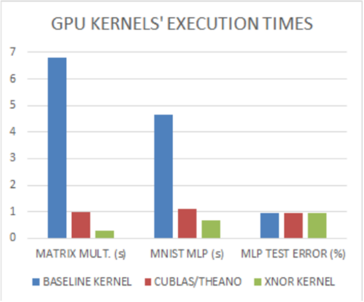
Exéperiences sur MNIST, CIFAR-10 et SVHN

Data set	MNIST	SVHN	CIFAR-10
Binarized activations+weights, during training and test			
BNN (Torch7)	1.40%	2.53%	10.15%
BNN (Theano)	0.96%	2.80%	11.40%
Committee Machines' Array (Baldassi et al., 2015)	1.35%	-	-
Binarized weights, during training and test			
BinaryConnect (Courbariaux et al., 2015)	1.29± 0.08%	2.30%	9.90%
Binarized activations+weights, during test			
EBP (Cheng et al., 2015)	2.2± 0.1%	-	-
Bitwise DNNs (Kim & Smaragdis, 2016)	1.33%	-	-
Ternary weights, binary activations, during test			
(Hwang & Sung, 2014)	1.45%	-	-
No binarization (standard results)			
Maxout Networks (Goodfellow et al.)	0.94%	2.47%	11.68%
Network in Network (Lin et al.)	-	2.35%	10.41%
Gated pooling (Lee et al., 2015)	-	1.69%	7.62%

Gain en énergie

Pour les GPU utilisant la technique SWAR (Single Instruction Multiple Data within a register), 32 variables binaires peuvent être concaténées dans des registres de 32 bits pour accélérer les opérations bit à bit (XNOR).





References

Matthieu Courbariaux and Yoshua Bengio and Jean-Pierre David (2015)
BinaryConnect: Training Deep Neural Networks with binary weights during propagations
Journal CoRR abs/1511.00363.

Matthieu Courbariaux and Yoshua Bengio (2016)
Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to $+1$ or 1
Journal CoRR abs/1602.02830.

Dubey, Anuj and Cammarota, Rosario and Aysu, Aydin (2020)
Boolean Masking of an Entire Neural Network

Rahul Varma (2021)
Binary Neural Networks — Future of low-cost neural networks