State of Art in Artificial Neurals Networks Compression

Pierre Gabin FODOP GUMETE

ENSTA Bretagne
pierre.fodop@ensta-bretagne.org

May 8, 2021

Abstract

Les vingt dernières années ont vu une augmentation exponentielle de la puissance de calcul des ordinateurs personnels, en plus de l'explosion des services de cloud et l'augmentation du nombre de fermes à serveurs pour le stockage de données. Cette Augmentation a eu parmi ses conséquences de mettre à la disposition du plus grand nombre un certain nombre de données, mais aussi l'explosion des méthodes de Réseaux de neurones. Ils ont été appliqués à un nombre important de problèmes notamment en traitement d'image, traitement du son, en traitement du langage naturel (...) Domaines dans lesquels ils constituent à ce jour l'état de l'art. D'un autre côté, on a aussi eu une progression importante de l'Internet des Objets, cela a créé le besoin de reseaux de neurones adaptés aux objets peut puissant. L'objectif de ce papier est de présenter les différentes methodes permettant de compresser les réseaux de neurones et par là de stocker les réseaux dans un moindre espace, mais aussi d'accélérer les calculs avec ces derniers.

I. Introduction

L'Histoire des réseaux de neurones débute avec Warren S. McCulloch et Walter Pitts de chercheurs du MIT qui présentent en 1943 un article sur la réalisation des fonctions neuronales par des fonctions électriques et des portes logiques[25]. Leur approche est lié à une perception des fonctions neuronales comme des fonctions ***Tout ou rien***. Cette avancée sera complete plus tard par la définition du perceptron[14] et la decouverte de l'algorithme de retro-propagation du gradiant[5].

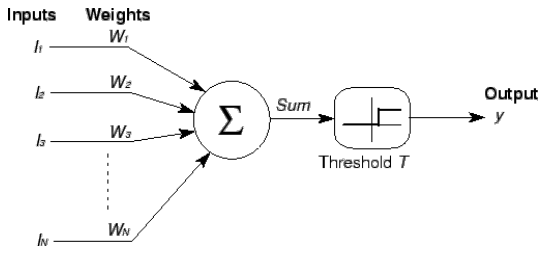

Figure 1: *Modèle Calculatoire de McCulloch et Pitts*

L'*hivers de l'Intelligence arrtificielle* qui commencera à la fin de la décennie 1960 les avancées dans le domaine de l'intelligence artificielle se font plus lentes. À cause d'une réduction des investissements autant publique que privée dans le domaine. Bien que ralenti, c'est dans cette période que sont conçu les premières implémentations à grande échelle des systèmes expert conçu en 1950. Ici la majeur partie des systèmes intelligents fonctionnes avec une base dite *Knowlege Based*(Basé sur la connaisance). On a donc un développement des bases de methodes telque la Programation Logique Inductive (ILP) [22][19]. Qui sera plus tard compilé par S.H. Muggleton dans son artice fondateur "Inductive Logic Programming"[12].

Enfin, dans les années 1990, on a un retour en force des méthodes d'intelligence artificielle avec l'augmentation de la puissance de calcul ; cela avec l'avènement des premiers processeurs Pentium par Intel, une généralisation des super calculateurs et enfin l'avènement d'Internet qui permettra un échange de connaissance sans

précèdent. Cette évolution atteindra un premier grand pallié en 2012 avec l'application des réseaux de neurones au challenge de classification d'image ImageNet qui permettra de faire passer le taux le plus faible d'erreur de classification de 25% a 16% grace au réseau AlexNet[20].

Depuis lors, les réseaux de neurones ont été appliqués à un nombre de plus en plus grand de problème. Notement en Vision par ordinateur[17] [21], en traitement du langage naturel[11], en analyse des signaux[10][28] et bien d'autres [13] Avec des resultats meilleurs que les méthodes qui etaient utilisé jusque là. Ceci a crée le besoin de réseaux de neurones adapté à un certain nombre de supports. Notamment les supports issue de L'Internet des Objets.

L'objectif de ce document est de faire un tour des méthodes permettant de compresser des réseaux de neurones et de les appliques dans un certain nombre d'objets, pour ce faire, nous commencerons par faire une présentation des différentes architectures de réseaux de neurones existantes, en suite, nous explorerons les méthodes de pruning, de représentation creuse notamment quantifie, de distillation de connaissance et enfin, nous verrons les méthodes futures

II. Types de Réseaux de Neurones

II A. Le Perceptron

C'est l'expression la plus simple d'un réseau de neurones, il est constitué d'un seul neurone formel[25].

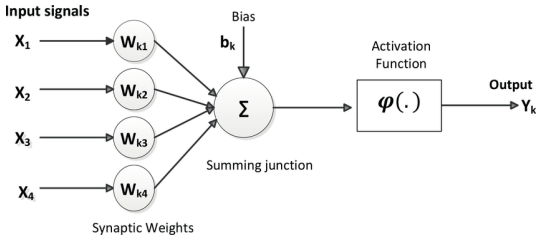

Figure 2: *Perceptron Multi-Couche*

Le neurone prend en entrée un vecteur de valeur x, le multiplie par un vecteur de poids w enfin grace a la fonction d'activation nous retourne la sortie.

$$y = \varphi(\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + b)$$

La fonction d'activation $\varphi()$ doit remplir un certain nombre de condition.

- **L'identite en zero** si $f(x) = 0$ alors $x = 0$ [23] Ces fonctions permettent de faire un apprentissage rapide.
- **Derivee de valeur monotone**[27] pour une meilleur capacite de generalisation et une facilite d'application des methodes d'optimisation convexe.
- **Partout Differentiable**[5] permetant l'application de la descente de gradiant.

Pour mettre a jour les poids w de notre perceptron, nous utiliseron l'algorithme de descente de gradient. Ce dernier permet a partir d'une fonction d'erreur choisie, de faire une mise a jour de ces poids. Soit $F(x)$ une fonction definie et differentiable au voisinage du point a minimum local. L'objectif etant d'estimer la valeur de a; on definira un coeficient d'aprentissage λ. On definira donc la suite suivante avec une intitialisation a a_0.

$$\begin{cases} a_0 & = a_0 \\ a_{n+1} & = a_n - \lambda.\Delta F(a_n) \end{cases}$$

Pour un reseau de neurone avec comme fonction d'activation la sigmiode et comme fonction d'erreur la fonction *cross entropy loss*

$$E(y) = -(y^v.log(y) + (1 - y^v).log(1 - y))$$

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y} * \frac{\partial y}{\partial z} * \frac{\partial z}{\partial w}$$

$$\frac{\partial E}{\partial y} = \frac{y^v}{y} - \frac{1 - y^v}{1 - y} = \frac{y^v - y}{y(1 - y)}$$

$$\frac{\partial y}{\partial z} = y * (1 - y)$$

$$\frac{\partial z}{\partial w} = w^T$$

$$\frac{\partial E}{\partial w} = x * (y^v - y)$$

d'ou on a

$$\begin{cases} w_0 & = w_0 \\ w_{n+1} & = w_n - \lambda.(y^v - y).x \end{cases}$$

la procedure est identique pour la mise a jour des biais.

A sa sortie,un neurone permet de faire une classification d'une classe en deux classe de valeurs. Celles qui actives le neurones et celles qui ne l'activent pas; on peut en utiliser plusieurs pour faire des classifications dans un nombre de classes plus important. On obtiendra alors des perceptron multi-couche

II B. Les perceptrons Multi-Couches

C'est le type de reseaux de neurones le plus basique, il est constitue d'un certain nombre de couche elle meme constitue de de neurone formel[25]. Les informations sont ainsi tansmises de la premieres couche dite couche d'entree a la dernieres couche dite de sortie en passant par les couches cache au travers de connection entre les neurones. Au cas ou chaque neurone d'une couche est lie a tous les neurones de la couche suivante on parle de reseaux entierement connectes,

Actuellement, les perceptrons multi-couches, obtiennent dans les meilleures conditions un taux d'erreur inferieure a 0.35%.[4] L'equivalent mathematique d'un resaux a 784 neurones d'entree 16 caches et 10 de sortie est le suivant:

$$y^1 = \varphi_1\left(\begin{bmatrix} w_1^1 & \dots & w_1^{784} \\ \vdots & \vdots & \vdots \\ w_{16}^1 & \dots & w_{16}^{784} \end{bmatrix} * \begin{bmatrix} x_1 \\ \vdots \\ x_{784} \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_{784} \end{bmatrix}\right)$$

$$y^2 = \varphi_2\left(\begin{bmatrix} w_1^1 & \dots & w_1^{16} \\ \vdots & \vdots & \vdots \\ w_{10}^1 & \dots & w_{10}^{16} \end{bmatrix} * \begin{bmatrix} x_1 \\ \vdots \\ x_{10} \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_{10} \end{bmatrix}\right)$$

Pour faire une distribution probabilistique des sorties, nous pouvonts utiliser un certaint nombre de fonction, dans ce cas, nous utiliserons

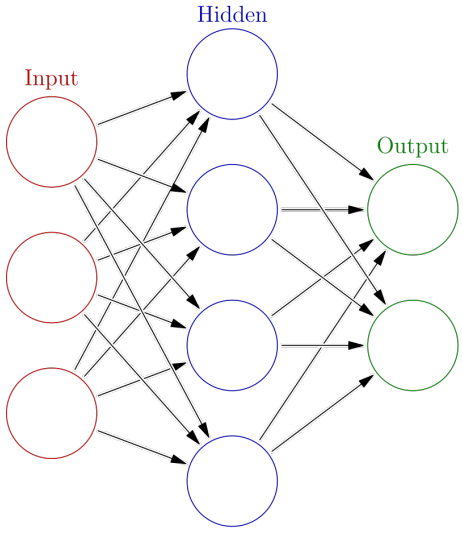

Figure 3: *Perceptron Multi-Couche*

Une fonction softmax.

$$out = softmax(y^2)$$

on aura donc un verceur de 10 probabilites, qui signifient chacune si la classe represente est la classe adequate.

II C. Les Reseaux de Neurones Convolutionnel (CNN)

Le plus grand defaut des reseaux de type perceptron multi couche est leur incapcite a traiter une image en tant qu'information globale a conduit au developpement des reseaux de neurones Convolutionnel.

Le filtrage pour une image est une operation mathematique qui permet a base d'une matrice appele kernel de faire un certain nombre d'operation sur une image. On peut citer parmi ces derniere le floutage, la detection de contour. Cela se fera par convolution d'une matrice appelle noyau avec une notre image. sa representation mathematique est la suivante:

$$g(x,y) = w * f(x,y)$$

$$w * f(x,y) = \sum_{dx=-a}^{a} \sum_{dx=-b}^{b} w(dx,dy) f(x+dx, y+dy)$$

Dans le cadre de la detection de contour qui est une des base des reseaux de neurones convolutionnel, On utilisera de facon preferentiel du filtrage de Canny [2]

Figure 4: *Filtrage de Canny*

Pour une couche convolutionnelle, nous appliquerons a notre image un certain nombre de filtre convolutionnel qui permetrons de saisir un maximum d'information sur l'image. Nous pourons appliquer un Pooling sur les image filtees obtenu pour en reduire le nombre de dimenssion.

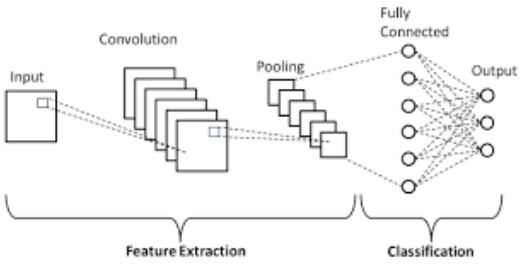

Figure 5: *Reseau Convolutionnel*

Et a la fin du reseau, nous appliquerons un reseau de type Multi layer perceptrons pour classifier les images. Il est aussi possible d'avoir un reseau entierement convolutionnel.

Les application des reseaux de types convolutionnels sont nombreuses, autant en traitement d'image [1] (VGG16, LeNet, ...), qu'en traitement du langage naturel[24], ou en traitement des signaux audios[6]. Bien que les reseaux de neuronnes convolutionnels completent les reseaux perceptron, ils presentent la limite de ne pas prendre en compte des valeurs temporelle et sont assez peux utiles pour analyser des sequences tels que des sequence video.

II D. Recurent Neural Network

Les reseaux recurents, on ete cree pour permettre un transfert de l'information dans les cellules en prenant en compte l'infulence du temps. Ici on parle d'une double propagation spaciaux temporelle. Cela donne la capacite a ces neuronnes d'intervenir dans l'analyse des phenomene evoluant dans le temps comme les videos, les series temporelles ...

Les reseaux recurents on vu le jour avec le reseau de Hopfield[9]. Ce dernier introduit la memoire dans les reseau permetant un passage au traver le temps des informations.Ils connaitrons quelques evolutions jusqu'a l'arive d'un autre type de reseaux de neurone, Le *Long Short Term Memory*[8]

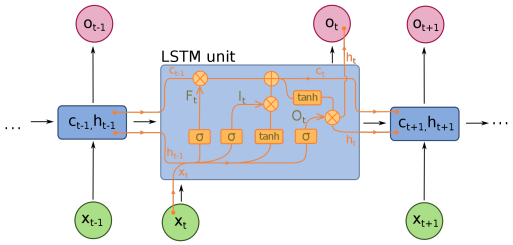

Figure 6: *Cellule LSTM*

Une celule LSTM prend en entree 3 valeurs valeurs, les sorties de la celule precedente H_t l'etat cache et C_t l'etat de la celule; mais aussi la valeur actuelle d'entree X_t.

la premiere porte que nous verons est porte de l'oublie. la fonction de la porte d'oublie est la suivante:

$$f_t = \sigma(U^f X_t + W^f h_{t-1})$$

Les portes suivantes permettent de calculer l'etat d'entree et l'etat de sortie

$$i_t = \sigma(U X_t + W h_{t-1})$$

$$O_t = \sigma(U^\circ X_t + W^\circ h_{t-1})$$

La porte de sortie

$$g_t = tanh(U^g X_t + W^g h_{t-1})$$

etat final de la cellule

$$c_t = i_t g_t + f_t c_{t-1}$$

Et l'etat cache suivant

$$h_t = tanh(c_t)o_t$$

Enfin en 2014 seront introduit les celule dites Gated Recurrent Unit (GRUs)[3].

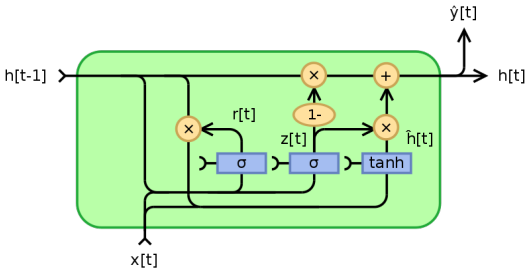

Figure 7: *Cellule GRU*

La cellule GRU prend en entre 2 variables et genere deux sorties. Les entrees sont h_t l'etat cache provenant de la precedente cellule x_t la valeur actuelle de l'entree ; les sorties sont quant a elles, y^est la valeur de sortie estimee et h_t l'etat cache estime.

Du point de vue mathematique on a:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$

$$h_t^{est} = \phi_h(W_h x_t + U_h(r_t * h_{t-1}) + b_h)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * h_t^{est}$$

ou x_t est le vecteur d'entree, h_t le vecteur de sortie, z_t le vecteur de mise a jour des protes, r_t vecteur de reinitialisation, h_t^{est} Vecteur candidat d'acivaion. W, U et b sont les marices de parametre. σ_g est la fonction sigmoide et ϕ_h est la tangente hyperbolique.

Plus recement, les GRU ont ete modifie pour donner les reseaux Minimal Gated Unit [7][29]. Ces reseaux presentent l'avantage d'avoir un minimum d'entree de sortie et de porte logiques.

II E. Autres reseaux

Les types de reseaux de neurones presente plus haut on tous la particularite d'etre en

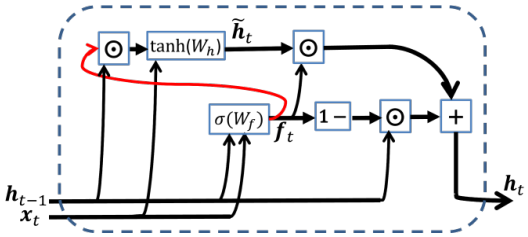

Figure 8: *Cellule MRU*

apprendtissage supervise. Ce genre de reseaux est generalement utilise pour la classification. Dans cette partie nous nous interesserons au cas des autres types de reseaux dont l'apprentissage est generalement non supervisee.

les premiers de ces reseaux seront, **Les Machines de Bolztmann**. l'origine de ces reseaux est l'article de Hinton et Sejnowski de 1983[15], ils sont utilise pour avoir des estimation de distribution probabiliste d'un jeux de donnee[18]. Ces derniers sont generalement utilise sous la forme de machine de bolzmann restreinte a deux couches de neurones.

En suite nous avons les **chaines de Markov**, Il s'ait ici d'un reseaux qui a base de connaisance de l'etat actuelle et d'un ensemble de probabilite connue peut predire les etat futur. Elles sont utilise sur les processus a temps discret, ou a temps continu et a espace d'etats discret.

Enfin nous classeron un certain nombre de reseaux notemment les auto encodeur (auto-encodeur variationnel, auto encodeur de debruitage), les Reseaux generatifs adversariaux (...) dans la cathegorie des reseaux composees en cela qu'il sont issue de la composition d'un certain nombre de couche de neuronne issue des reseaux mentionne plus haut.

III. Pourquoi la compression des reseaux de neurones

L'origine des objets connectes remonte a 1994 quand la start-up Violet a lance le concept de la lampe connecte en wifi DAL. Et depuis lors leur nombre n'a pa arrete de croitre, leur usage aussi. D'un autre cote le nombre d'outils

informatique a lui aussi augmente de facons considerable, avec l'invention des smartphones, des tabletes et autres.

Ces objets connnectes ont pour point commun de ne disposer ni d'une grande puissance de calcul ni d'un grand espace de stockage. Cette situation les eloignes des possiblite d'aplication des reseaux de neurones qui quand a eu on besoin d'une puissance de calcule importante et d'un espace de stockage lui aussi grand.

DNN	Size(MB)	Temps(Milion)
AlexNet	200	720
VGG-16	550	15 300
GoogLeNet	50	1550
ResNet	170	11 300

Table 1: *Les Poids et temps de calcul de differents reseaux*

Pour rendre ces objet plus intelligent il est donc important pour nous de pouvoir developper des reseaux de neurones adaptés. Pour ce faire, un certain nombre de moyens et de methodes ont ete developpe. Parmi les methodes qui s'attaquent a reduire le poid des reseaux on a, Le pruning, la quantification, la distillation de connaisance. D'autres approches visent une repartition des calclus entre plusieurs objets comme l'apprentissage federe. Nous explorerons ici en profondeur les approches qui s'attellent a reduire la taille des reseaux et nous ferons une introduction de l'aprentissage federe. Enfin , Dans la mesure ndu possible, nous presenterons les resultas que nous avons eu en modelisant les methodes.

IV. Le Pruning

IV A. C'est quoi le Pruning

L'une des premieres idee pour reduire la taille des reseaux en memoire, est de suprimer les connections inutiles entre les neurones. En effet lors de l'entrainement des reseaux de neurones et leur utilisation, ils existent potentiellement des connections qui se sont jamais activee, leurs supression fait donc gagner de l'espace de stockage.

Le Pruning est la methode de reduction de taille des reseaux de neurones qui consiste a suprimer les poids inferieure a un certain seuil.

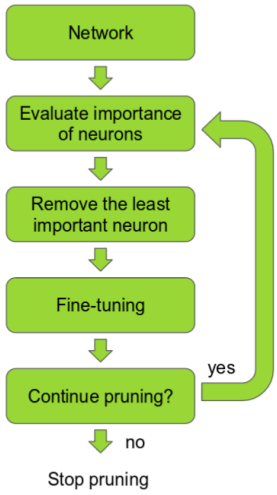

Figure 9: *etapes de Pruning*

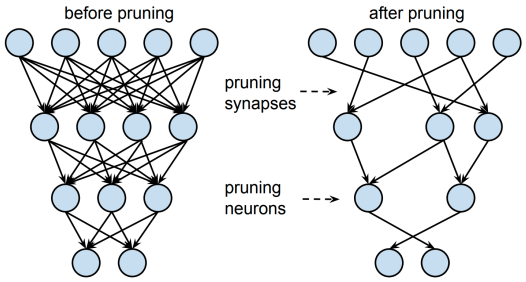

Figure 10: *Pruning*

Les recherches dans ce dommaine sont oriente vers la resolution du probleme compression/ efficacite qui en decoule.

Dans la section qui suit, nous presenterons l'etat de l'art de la methode de pruning cela en nous attardant sur certaines publications que nous avons selectionnee.

IV B. Etat de l'art de la methode Pruning

La premiere publication majeure qui a pose les jalon de l'application de la methode de pruning

nous date des annee 1991[16]. l'idee principale de ce dernier est d'utiliser le mean square error associer a la norme de Forbenius pour

IV C. Le futur du Pruning

IV D. Presentation de mes resultats

V. Quantification

La quantifictation se definit comme la division d'une valeur continue en un ensemble de valeurs discretes. De facon reele, tout element contenu dans un systeme electronique est d'une facon ou d'une autre quantifié les ordinateurs ne pouvant contenir que des valeurs discretes. Dans ce catre la la quantification va consister en la reduction du nombre de bits utilisées pour stocker l'information. Dans un premier temps elle a ete appliqué à l'echantillonagez de valeur dans les signaux et la la compression de ceux ci ou elle a montré une cappacité interessante. l'equation de la quantifictation sera la suivante:

$$Q(x) = \Delta * E[\frac{x}{\Delta} + \frac{1}{2}]$$

La Quantification d'un signal introduit un certain type d'erreur qui est due au difference entre les valeurs reele et les valeurs une fois compressé, la formule de calcul de cette derniere est la suivante:

$$SNR_{dB} = 20\frac{Log_{10}(\frac{2*q}{2\sqrt{2}})}{\frac{q}{2\sqrt{3}}}$$

Un reseaux de neurone est peut etre modelisé comme une fonction qui prend en entree une valeur x, son label correspondant y et un ensemble de parametre θ qui nous retourne une une valeur correspondant à la classe que nous souhaitons predire.

$$L(\theta) = \frac{1}{N}\sum_{n=1}^{\infty} l(x_i, y_i, \theta)$$

Notre objectif sera de quantifier les valeurs contenues dans θ sans pour autant reduire l'efficacité de notre reseaux de neurones

V A. Methode de Quantification

V A.1 Quantification uniforme

L'idee ici est de definir un sous ensemble de valeur dans le quel projeter uniformement les poids, la formule est la suivante[26]:

$$Q(r) = E(\frac{r}{S}) - Z$$

Ou Q est l'operateur de quantification, E la valeur entiere, r la valeur reel de l'entrée et Z la valeurs qui sera concidéré comme le centre de nos valeur projetées.

On peut adjoindre à cette fonction de quantification une fonction de *de-quantification* qui permet de revenir à une estimation de la valeur vraie

$$\vec{r} = S(Q(r) + Z)$$

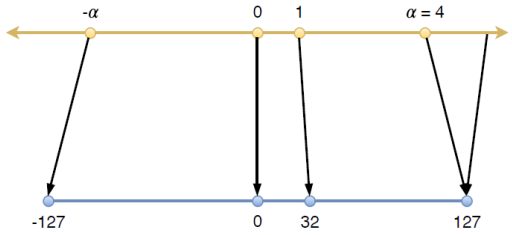

Figure 11: *Symetric Quantification*

V A.2 Quantification asysmetrique

Cette methode de quantification est quasiment identique à la quantification uniforme a la diference qu'ici, l'espace de projection est un intervale $[\alpha, \beta]$ avec $-\alpha$!= β la formule de quantification est la suivante [26]:

$$Q(r) = E(\frac{r}{S})$$

ou

$$S = \frac{\beta - \alpha}{2^b - 1}$$

le meileur choix en général est

$$S = \frac{2\max(|r|)}{2^n - 1}$$

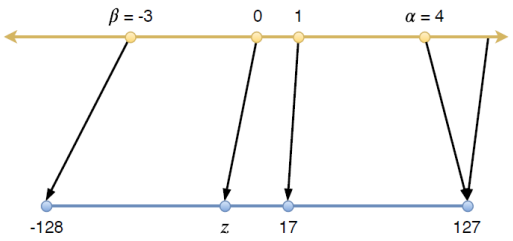

Figure 12: *Symetric Quantification*

La quantification uniforme et la quantification asymetrique peuvent etre appliqué à diffrent probleme selon certains mechanismes. Parmis les mechanismes possible on a:

- **la Quantification Dynamique** qui consiste a appliquer durant l'inference chacunes des methodes precedentes de facon adapté à chaque couche du reseaux.
- **La quantifiaction statique**, ici le choix de la methode de quantification est fait en amont du processus d'inference et appliqué identiquement à chaque couche.

On peut trouver comme autre methodes de quantification

- **Quantification couche par couche**
- **Quantification groupe par groupe**
- **Quantification par chanel**
- **Quantification par sous chanel**

V B. Binarization

VI. Distillation de connaissance

VII. Autres Methodes

VIII. Conclusion

References

[1] Matthew Browne and Saeed Ghidary. Convolutional neural networks for image processing: An application in robot vision, 12 2003.

[2] John Canny. A computational approach to edge detection. *Transaction on pattern analysis and machine Intelligence*, 8(6), 1986.

[3] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[4] Dan Cirean, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep big multilayer perceptrons for digit recognition. *springer*, 01 2012.

[5] Ronald J. Williams David E. Rumelhart, Geoffrey E. Hinton. Learning representations by back-propagating errors. *Nature*, 323(9), 1986.

[6] Fernando Gama, Antonio G. Marques, Geert Leus, and Alejandro Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67(4):1034–1049, Feb 2019.

[7] Joel Heck and Fathi M. Salem. Simplified minimal gated unit variations for recurrent neural networks, 2017.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[9] J.J Hopfield. Neural networks and physical systems with emergent collecive computational abilities. *Biophysics*, 79, 1982.

[10] Mohamed Ibnkahla. Application of neural networks to digital communications - a survey. *Signal Processing*, 80:1185–1215, 2000.

[11] Kun Jing and Jungang Xu. A survey on neural network language models, 2019.

[12] Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8:295–318, 1991.

[13] Alexander Poznyak, Isaac Chairez, and Tatyana Poznyak. A survey on artificial neural networks application for identification and control in environmental engineering: Biological and chemical systems

with uncertain models. *Annual Reviews in Control*, 48:250–272, 2019.

[14] F. RosenBlatt. The perceptron: A probaabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 1958.

[15] Rumelhart, E. David, Mcclelland James, and L. James. *Parallel distributed processing: explorations in the microstructure of cognition. Volume 1. Foundations*. The MIT Press, 01 1986.

[16] Yu Hen Hu S. Y. Kung. A forbenius approximation reduction method (farm) for determining optimal number of hidden unints. *Proceedings of the IJCNN-91*, 1991.

[17] Madhusmita Sahu and Rasmita Dash. A survey on deep learning: Convolution neural network (cnn). *Annual Reviews in Control*, pages 317–325, 01 2021.

[18] Ruslan Salakhutdinov and Geoffrey Hinton. Learning and evaluaing deep bolztmann machines. *MLR press*, 04 2008.

[19] Ehud Y. Shapiro. inductive inference of theories from facts. Technical report, Yale University, 1981.

[20] Rajat Vikram Singh. Imagenet winning cnn architectures - a review. Technical report, andrew cmu, 2016.

[21] Madasamy Sornam, Muthu Subash Kavitha, and Vanitha Venkateswaran. A survey on image classification and activity recognition using deep convolutional neural network architecture. *Annual Reviews in Control*, page 1, 12 2017.

[22] WRAY BUNTINE STEPHEN MUGGLETON. Machine invention of first-order predicates by inverting resolution. In John Laird, editor, *Machine Learning Proceedings 1988*, pages 339–352. Morgan Kaufmann, San Francisco (CA), 1988.

[23] David Sussillo and L. Abbott. Random walk initialization for training very deep feedforward networks. *arXiv: Neural and Evolutionary Computing*, 2014.

[24] W. Wang and J. Gang. Application of convolutional neural network in natural language processing. In *2018 International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pages 64–70, 2018.

[25] Walter Pitts Warren S. McCulloch. Logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52(1/2):99–115, 1943.

[26] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *CoRR*, abs/2004.09602, 2020.

[27] Huaiqin Wu. Global stability analysis of a general class of discontinuous neural networks with linear growth activation functions. *Information Sciences*, 179(19):3432–3441, 2009.

[28] Sha Zhang Xiaofan Li, Fangwei Dong. A survey on deep learning techniques in wireless signal recognition. *Hindawi*, 2019(12), 2019.

[29] Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. Minimal gated unit for recurrent neural networks, 2016.