Auto-encodeur optimisé au sens débit-distorsion : indépendant de la quantification?

Thierry Dumas, Aline Roumy, Christine Guillemot *

INRIA Rennes Bretagne-Atlantique
263 avenue du Général Leclerc, 35042 Rennes, France
thierry.dumas@inria.fr, aline.roumy@inria.fr, christine.guillemot@inria.fr

Résumé − Ce travail s'inscrit dans le cadre de la compression d'image via une transformée apprise par un auto-encodeur. Il essaie d'adapter la quantification à cette transformée au lieu de la figer. Nous proposons d'une part d'apprendre conjointement la transformée et la quantification. D'autre part, nous analysons si une multitude de pas de quantification peut s'appliquer lors du test sur une transformée apprise pour un pas. Nous montrons que la seconde approche corrige le défaut du meilleur auto-encodeur pour la compression d'image : devoir effectuer un apprentissage par débit de compression.

Abstract − This work relates to image compression via a transform learned by an auto-encoder. It tries to adapt the quantization to this transform instead of fixing it. We propose to jointly learn the transform and the quantization. Moreover, we analyze whether different quantization steps can be applied to a transform learned for one step only. We show that the second approach corrects the flaw of the state-of-the-art auto-encoder for image compression: having to learn one transform per compression rate.

1 Introduction

Les normes de codage d'image et les normes de codage vidéo utilisent une transformée linéaire et inversible pour convertir une image en une représentation plus compacte. Par exemple, dans JPEG, une transformée en cosinus discrète (DCT) est appliquée sur des blocs de pixels. Dans H.265, une DCT est appliquée sur la différence entre des blocs de pixels et leur prédiction intra-image. Un gain en compression pourrait être obtenu en remplaçant cette DCT par une transformée qui extrait de l'image de l'information conceptuelle. Malheureusement, une telle transformée est difficile à définir. Ce problème peut être résolu par l'apprentissage. En effet, certains auto-encodeurs profonds sont capables d'apprendre cette transformée [4, 3].

Les approches de compression d'image basées sur des auto-encodeurs profonds [3, 1] fixent la quantification. C'est-à-dire que les paramètres de quantification ne sont pas optimisés. Ceci est surprenant car la quantification est une étape cruciale dans toutes les normes de codage [7, 5]. À partir de ce constat, deux questions se posent. (i) Est-ce que l'optimisation des paramètres de quantification lors de l'apprentissage de l'auto-encodeur est pertinente ? (ii) Que se passe-t-il lorsque, lors du test, l'auto-encodeur est soumis des quantifications qu'il n'a pas apprises ? Nous proposons une méthode pour transformer les paramètres de quantification en paramètres d'apprentissage. Cette méthode permet de répondre à la première question. Par ailleurs, nous analysons la représentation apprise par un auto-encodeur et cherchons des quantifications adéquates lors du test. Ceci s'attaque à la deuxième question.

1.1 Notations

Les vecteurs sont notés par des lettres minuscules en gras. Les matrices et les tenseurs sont notés par des lettres capitales en gras. $\|\mathbf{X}\|_2$ est la norme de Frobenius de $\mathbf{X}$.

2 Apprentissage joint de la quantification et de l'auto-encodeur

Cette section introduit l'auto-encodeur qui, à ce jour, donne les meilleurs compromis débit-distorsion, c'est-à-dire le mieux adapté à la compression d'image. Ensuite, notre proposition d'apprentissage joint de la quantification et de ce type d'auto-encodeur est expliquée.

2.1 Auto-encodeur pour la compression

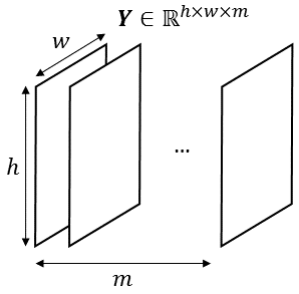
Un auto-encodeur est un réseau de neurones qui est séparé en deux parties. Un encodeur g_e paramétré par $\boldsymbol{\theta}$ prend une image $\mathbf{X}$ et génère une représentation $\mathbf{Y} = g_e(\mathbf{X};\boldsymbol{\theta})$. Un décodeur g_d parametré par $\boldsymbol{\phi}$ prend $\mathbf{Y}$ et donne $\hat{\mathbf{X}} = g_d(\mathbf{Y};\boldsymbol{\phi})$, une reconstruction de $\mathbf{X}$.

Un algorithme de compression doit s'appliquer à des images de n'importe quelle taille. Dans les auto-encodeurs

*Ce travail est en partenariat avec la DGA.

avec des couches entièrement connectées, le nombre de paramètres dépend de la taille de l'image. Ceci oblige à entraîner un auto-encodeur par taille d'image. Par conséquent, pour la compression d'image, les architectures avec uniquement des couches convolutives et des opérateurs non-linéaires sont préférables. Dans ce cas, $\mathbf{Y} \in \mathbb{R}^{h \times w \times m}$ est une pile de matrices, voir Figure 1. $m \in \mathbb{N}_+^*$ correspond au nombre de noyaux dans la dernière couche convolutive de l'encodeur.

FIGURE 1 – $\mathbf{Y}$ dans un auto-encodeur convolutif.



La méthode classique d'apprentissage des auto-encodeurs minimise l'erreur de reconstruction de l'image. [1] cherche en plus à minimiser l'entropie de la représentation de l'image après quantification. Ceci valorise un codage entropique de la représentation quantifiée. Plus formellement, supposons que, pour $i \in [|1, m|]$, les $n = h \times w$ coefficients $\{y_{ij}\}_{j=1...n}$ dans la $i^{\text{ème}}$ matrice de $\mathbf{Y}$ sont des réalisations d'une variable aléatoire continue Y_i de densité de probabilité p_i, voir Figure 1. Insérons une quantification $\hat{\mathbf{Y}} = \mathcal{Q}(\mathbf{Y})$ entre l'encodeur et le décodeur. Pour $i \in [|1, m|]$, les coefficients $\{\hat{y}_{ij}\}_{j=1...n}$ dans la $i^{\text{ème}}$ matrice de $\hat{\mathbf{Y}}$ sont modélisés comme des réalisations d'une variable aléatoire discrète $\hat{Y}_i = \mathcal{Q}(Y_i)$ de fonction de masse $\hat{p}_i$. Avec ces notations, la minimisation à la fois de l'erreur de reconstruction de l'image et de l'entropie de la représentation de l'image après quantification est (1).

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$$
$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}\left[\|\mathbf{X} - g_d(\mathcal{Q}(g_e(\mathbf{X}; \boldsymbol{\theta})); \boldsymbol{\phi})\|_2^2 + \gamma \sum_{i=1}^{m} H_i\right]$$
$$H_i = -\frac{1}{n}\sum_{j=1}^{n} \log_2(\hat{p}_i(\hat{y}_{ij})), \gamma \in \mathbb{R}_+^* \quad (1)$$

Pour $i \in [|1, m|]$, H_i est l'entropie estimée de $\hat{Y}_i$. L'espérance $\mathbb{E}[.]$ est approximée par une moyenne sur une base d'apprentissage. Malheureusement, la quantification rend (1) inutilisable. En effet, la dérivée de n'importe quelle fonction de quantification $\mathcal{Q}$ par rapport à son entrée est zéro en tout point. Par conséquent, $\boldsymbol{\theta}$ ne peut pas être appris via des méthodes basées gradient [8]. Pour contourner ce problème, [1] choisit pour $\mathcal{Q}$ une quantification scalaire uniforme de pas 1 et donne une approximation de $\mathcal{Q}$ dont la dérivée n'est pas nulle partout. Comme alternative, nous proposons une approximation pour n'importe quelle quantification scalaire uniforme qui peut être apprise. C'est le propos de la section 2.2.

2.2 Apprentissage de la quantification

$\mathcal{Q}$ est divisée en $\mathcal{Q}_1, ..., \mathcal{Q}_m$. Pour $i \in [|1, m|]$, $\mathcal{Q}_i$ est une quantification scalaire uniforme de pas $\delta_i \in \mathbb{R}_+^*$. $\mathcal{Q}_i$ s'applique à la $i^{\text{ème}}$ matrice de $\mathbf{Y}$. Pour $i \in [|1, m|]$, $\hat{Y}_i = \delta_i \lfloor Y_i / \delta_i \rceil$ où $\lfloor . \rceil$ arrondit à l'entier le plus proche. Soit $\hat{\mathcal{Y}}_i = \{..., -\delta_i, 0, \delta_i, ...\}$ l'ensemble des symboles de $\hat{Y}_i$. Pour $i \in [|1, m|]$, pour $q \in \hat{\mathcal{Y}}_i$,

$$\hat{p}_i(q) = \int_{q-0.5\delta_i}^{q+0.5\delta_i} p_i(t)\, dt = \delta_i (p_i * f_i)(q)$$

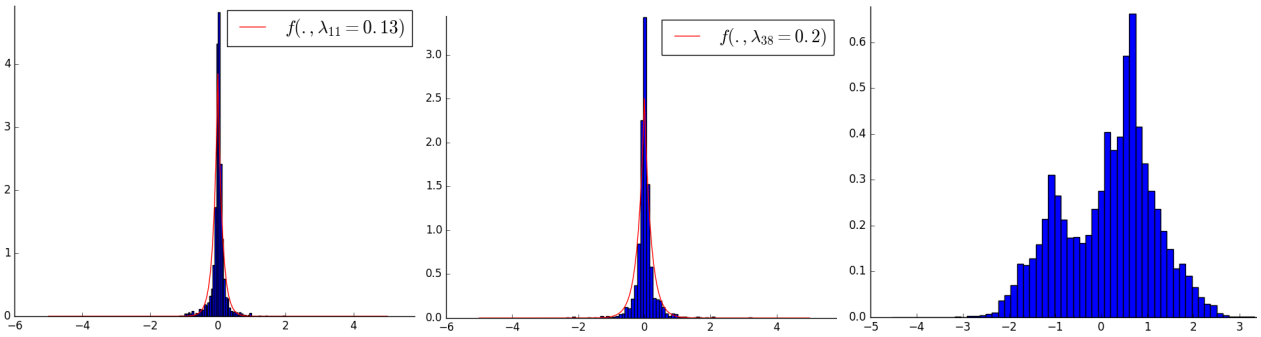
f_i est la densité de probabilité de la loi uniforme continue de support $[-0.5\delta_i, 0.5\delta_i]$. Pour $i \in [|1, m|]$, $\tilde{p}_i = p_i * f_i$ est la densité de probabilité de la variable aléatoire continue $\tilde{Y}_i = Y_i + \mathcal{E}_i$ où $\mathcal{E}_i$ est une variable aléatoire continue de densité de probabilité f_i. Avec les relations précédentes, (1) peut être approximée par (2).

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \delta_1, ..., \delta_m)$$
$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \delta_1, ..., \delta_m) = \mathbb{E}\Bigg[\|\mathbf{X} - g_d(g_e(\mathbf{X}; \boldsymbol{\theta}) + \boldsymbol{\mathcal{E}}; \boldsymbol{\phi})\|_2^2 + \gamma\left(\sum_{i=1}^{m} \tilde{h}_i - \sum_{i=1}^{m} \log_2(\delta_i)\right)\Bigg]$$
$$\tilde{h}_i = -\frac{1}{n}\sum_{j=1}^{n} \log_2(\tilde{p}_i(y_{ij} + \varepsilon_{ij})) \quad (2)$$

Pour $i \in [|1, m|]$, la $i^{\text{ème}}$ matrice de $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{h \times w \times m}$ contient n réalisations $\{\varepsilon_{ij}\}_{j=1...n}$ de $\mathcal{E}_i$. Pour $i \in [|1, m|]$, $\tilde{h}_i$ est l'entropie différentielle de $\tilde{Y}_i$ dans sa forme estimée. Dans les deux termes de la fonction à minimiser, la quantification $\mathcal{Q}$ a été remplacée par une approximation dont la dérivée par rapport à son entrée ne s'annule nulle part. Il est désormais possible d'apprendre $\boldsymbol{\theta}$ par des méthodes basées gradient. Par contre, on ne peut pas encore apprendre $\delta_1, ..., \delta_m$. Les réalisations de $\mathcal{E}_i$ dépendent de manière implicite de δ_i. $\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \delta_1, ..., \delta_m)$ n'est donc pas dérivable par rapport à δ_i. Ceci se résout grâce au changement de variable $\mathcal{E}_i = \delta_i \mathcal{T}_i$ où $\mathcal{T}_i$ est une variable alétoire suivant la loi uniforme continue de support $[-0.5, 0.5]$. Maintenant, une minimisation sur $\delta_1, ..., \delta_m$ est possible, voir (3).

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi},\delta_1,...,\delta_m} \tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \delta_1, ..., \delta_m)$$
$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \delta_1, ..., \delta_m) = \mathbb{E}\Bigg[\|\mathbf{X} - g_d(g_e(\mathbf{X}; \boldsymbol{\theta}) + \boldsymbol{\Delta\mathcal{T}}; \boldsymbol{\phi})\|_2^2 + \gamma\left(\sum_{i=1}^{m} \tilde{h}_i - \sum_{i=1}^{m} \log_2(\delta_i)\right)\Bigg]$$
$$\tilde{h}_i = -\frac{1}{n}\sum_{j=1}^{n} \log_2(\tilde{p}_i(y_{ij} + \delta_i \tau_{ij})) \quad (3)$$

FIGURE 2 – Histogramme normalisé de la $i^{\text{ème}}$ matrice de $\mathbf{Y}$.



(a) Auto-encodeur via (3), $i = 11$. (b) Auto-encodeur via (3), $i = 38$. (c) Auto-encodeur classique, $i = 10$.

Pour $i \in [|1, m|]$, la $i^{\text{ème}}$ matrice de $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{h \times w \times m}$ contient n réalisations $\{\tau_{ij}\}_{j=1...n}$ de $\mathcal{T}_i$. Tous les coefficients dans la $i^{\text{ème}}$ matrice de $\boldsymbol{\Delta} \in \mathbb{R}^{h \times w \times m}$ sont égaux à δ_i. Un détail a été laissé de côté jusqu'ici. Pour $i \in [|1, m|]$, $\tilde{p}_i$ est inconnu. Comme dans [1], on peut choisir une fonction linéaire par morceaux $\tilde{f}_i$ de paramètres $\boldsymbol{\psi}_i$ et apprendre $\boldsymbol{\psi}_i$ afin que $\tilde{f}_i$ approxime $\tilde{p}_i$.

Au final, il y a trois groupes de paramètres : $\{\boldsymbol{\theta}, \boldsymbol{\phi}\}$, $\{\delta_1, ..., \delta_m\}$ et $\{\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_m\}$. Ces trois groupes sont appris en alternant trois descentes de gradient stochastique différentes. Le lecteur trouvera le détail des heuristiques d'apprentissage sur le site internet [1].

Le but de la section 2 est d'apprendre $\delta_1, ..., \delta_m$. Ensuite, lors du test, $\delta_1, ..., \delta_m$ seront inchangés. Avant de passer aux expériences, l'approche complémentaire est expliquée. $\delta_1, ..., \delta_m$ sont figés lors de l'apprentissage. Par contre, la quantification évolue lors du test. Ceci est la section 3.

3 Quantifier au moment du test

Afin de comprendre comment quantifier la représentation d'une image donnée par l'auto-encodeur appris, on analyse la distribution des coefficients dans cette représentation.

On commence par construire un auto-encodeur convolutif. $g_e(.,\boldsymbol{\theta})$ et $g_d(.,\boldsymbol{\phi})$ sont une succession de trois couches convolutives et d'opérateurs non-linéaires. m est égal à 64. Cet auto-encodeur est entraîné sur une base de 24000 images de luminance de taille 256x256 créée à partir d'ImageNet [2]. L'objectif d'optimisation est (3) sauf que, $\forall i \in [|1, m|]$, $\delta_i = 0.8$ n'est pas appris. Après l'apprentissage, des images de luminance de taille 512x768 de la base Kodak [2] sont insérées dans l'auto-encodeur. Par exemple, si $\mathbf{X}$ désigne la $3^{\text{ième}}$ image de luminance de Kodak, l'auto-encodeur fournit $\mathbf{Y} = g_e(\mathbf{X}, \boldsymbol{\theta})$. Figure 2(a)(b) montre l'histogramme normalisé de la $11^{\text{ième}}$ matrice de $\mathbf{Y}$ et celui de la $38^{\text{ième}}$ matrice. Toutes les matrices de $\mathbf{Y}$, à l'exception de 2 parmi les 64, ont des histogrammes normalisés semblables à ceux affichés. Pour préciser l'étude, écrivons la densité de probabilité de la distribution de Laplace de moyenne 0 et de paramètre d'échelle $\lambda \in \mathbb{R}_+^*$:

$$f(x, \lambda) = \frac{1}{2\lambda} \exp\left(-\frac{|x|}{\lambda}\right)$$

Pour toutes les matrices de $\mathbf{Y}$, à l'exception de 2 parmi les 64, il existe $\lambda_i \in [0.05, 0.6]$ tel que l'histogramme normalisé de la $i^{\text{ème}}$ matrice s'accorde avec la courbe de $f(., \lambda_i)$. Par exemple, $\lambda_{11} = 0.13$ pour la $11^{\text{ième}}$ matrice et $\lambda_{38} = 0.2$ pour la $38^{\text{ième}}$. La distribution de Laplace se retrouve dans les transformées qui offrent de bons compromis débit-distorsion lorsque la représentation de l'image subit une quantification évolutive [6]. C'est pourquoi il semble judicieux de faire évoluer la quantification scalaire uniforme qui est appliquée sur la représentation donnée par l'auto-encodeur lors test. Ceci sera vérifié dans la section 4.

Qu'est-ce qui a engendré l'apprentissage de ces distributions de Laplace ? Nous avons entraîné des architectures avec différents opérateurs non-linéaires (Leaky ReLU et GDN [1]). Rien ne change. En revanche, si un auto-encodeur classique est appris, i.e en ne minimisant que l'erreur de reconstruction de l'image, ces distributions de Laplace disparaissent. Par exemple, Figure 2(c) montre l'histogramme normalisé de la $10^{\text{ième}}$ matrice de $\mathbf{Y}$ généré par l'auto-encodeur classique. La contrainte sur l'entropie donne donc naissance à ces distributions de Laplace.

4 Expériences

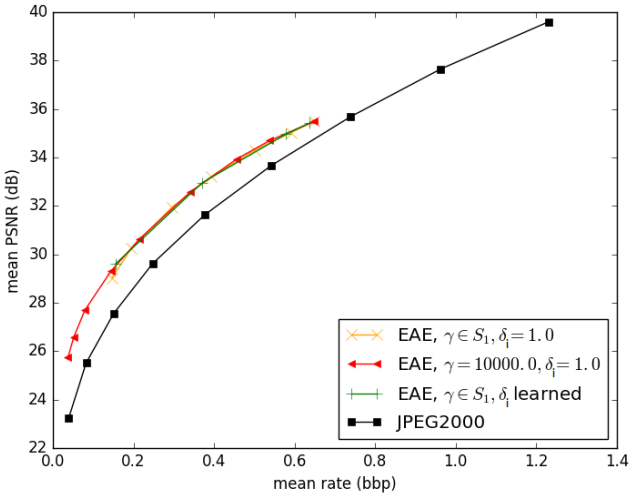
Pour répondre aux questions traitées par les sections 2 and 3, trois approches sont comparées.

La première est une référence. Elle se base sur le protocole dans [1]. Plus précisément, un auto-encodeur est appris pour chaque valeur du coefficient $\gamma \in S_1 = \{10000.0, 12000.0, 16000.0, 24000.0, 40000.0, 72000.0, 96000.0\}$ qui nivelle la contrainte de distorsion par rapport à la contrainte d'entropie. Ainsi, chaque auto-encodeur est dédié à un débit de compression. $\forall i \in [|1, m|], \delta_i = 1.0$ n'est pas appris et reste inchangé au moment du test.

La seconde approche découle de la section 2. Un auto-encodeur est également appris pour chaque valeur $\gamma \in S_1$

1. `www.irisa.fr/temics/demos/quantization_ae/quantizationAE.htm`
2. *r0k.us/graphics/kodak/*

FIGURE 3 – Courbes de débit-distorsion moyennées sur les 24 images de luminance de Kodak.



mais, cette fois-ci, $\forall i \in [|1, m|], \delta_i$ est appris. La valeur de δ_i lors du test est celle obtenue à la fin de l'apprentissage.

La troisième approche correspond à la section 3. Un unique auto-encodeur est appris pour $\gamma = 10000.0$ et, $\forall i \in [|1, m|]$, $\delta_i = 1.0$ n'est pas appris. Par contre, δ_i croît au moment du test. Dans ce cas uniquement, un seul auto-encodeur est utilisé pour plusieurs débits.

Tous les auto-encodeurs convolutifs ont une architecture identique à [1]. Ces auto-encodeurs sont appris sur 24000 images de luminance de taille 256x256 créées à partir d'ImageNet. Puis, au moment du test, les 24 images de luminance de Kodak sont passées dans ces auto-encodeurs. Le débit est évalué via l'entropie empirique des coefficients quantifiés en supposant que ces coefficients sont i.i.d, voir (1). Figure 3 montre des courbes de débit-distorsion moyennées sur ces 24 images. La courbe de JPEG2000 est obtenue en utilisant ImageMagick [3]. Il n'y a quasiment pas de différence entre la première approche et la seconde. Nous avons observé que, pendant l'apprentissage, l'auto-encodeur profond arrive sans souci à dilater ou contracter les distributions dans la représentation de l'image. De cette façon, il trouverait des compromis débit-distorsion similaires que la quantification soit imposée ou apprise. En revanche, nous voyons que la troisième approche est aussi performante que la première en ne demandant qu'un apprentissage. Ceci est critique puisqu'un apprentissage prend 2 jours avec un GPU NVIDIA Quadro K6000.

5 Conclusion

Utiliser une quantification variable au moment du test permet, avec un seul auto-encodeur, de faire une compression aussi performante qu'en apprenant un auto-encodeur par point de débit. En revanche, l'apprentissage joint de l'auto-encodeur et de la quantification scalaire uniforme semble peu intéressant. Pour creuser ce point, cette approche devra être généralisée à des quantifications non uniformes. Par ailleurs, la base d'apprentissage est construite à partir d'images encodées en JPEG. Il faudra analyser si ceci biaise l'auto-encodeur appris.

3. www.imagemagick.org/script/index.php

Références

[1] Johannes BALLÉ, Valero LAPARRA et Eero P. SIMONCELLI : End-to-end optimized image compression. *In ICLR*, 2017.

[2] Jia DENG, Wei DONG, Richard SOCHER, Li-Jia LI, Kai LI et Fei-Fei LI : ImageNet : a large-scale hierarchical image database. *In CVPR*, 2009.

[3] Karl GREGOR, Frederic BESSE, Danilo Jimenez REZENDE, Ivo DANIHELKA et Daan WIERSTRA : Towards conceptual compression. *arXiv preprint arXiv : 1604.08772*, 2016.

[4] Karl GREGOR, Ivo DANIHELKA, Alex GRAVES et Daan WIERSTRA : DRAW : a recurrent neural network for image generation. *In ICLR*, 2015.

[5] Gary J. SULLIVAN, Jens-Rainer OHM, Woo-Jin HAN et Thomas WIEGAND : Overview of the High Efficiency Video Coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22 (12): 1649–1668, December 2012.

[6] Edmund Y. LAM et Joseph W. GOODMAN : A mathematical analysis of the DCT coefficient distributions for images. *IEEE Transactions on Image Processing*, 9 (10):1661–1666, October 2000.

[7] Michael W. MARCELLI, Margaret A. LEPLEY, Ali BILGIN, Thomas J. FLOHR, Troy T. CHINEN et James H. KASNER : An overview of quantization in JPEG 2000. *Image Communication*, 17 (1):73–84, January 2002.

[8] David E. RUMELHART, Geoffrey E. HINTON et Ronald J. WILLIAMS : Learning representations by back-propagating errors. *Nature*, 323 (9):533–536, October 1986.