Optimisation débit-distorsion de la quantification dans un encodeur vidéo par apprentissage profond de faible complexité

Pierre-Alain AFRO[1,2] Loïc STRUS[1] Laurent BONNAUD[2] Alice CAPLIER[2] Frédéric ROBIN[1]

[1]Allegro DVT, 30 rue Lavoisier, 38330 Montbonnot-Saint-Martin, FRANCE

[2]GipsaLab, 11 Rue des Mathématiques, 38400 Saint-Martin-d'Hères, FRANCE

Résumé – L'optimisation débit-distorsion de la quantification ou RDOQ (*Rate Distorsion Optimized Quantization*) est un outil important présent dans différents codecs vidéo comme H.265/HEVC ou AV1. Même si cette fonction réduit significativement le débit, sa complexité limite son usage dans des solutions d'encodage matériel. Plusieurs études ont cherché à la simplifier notamment dans le modèle HM, l'implémentation logicielle de référence d'H.265. Cependant sa structuration itérative et séquentielle liée au calcul du débit reste difficilement compatible pour une mise en oeuvre matérielle. Avec l'émergence du *machine learning*, plusieurs modèles de réseaux de neurones ont été proposés afin d'imiter le comportement de la RDOQ et de pouvoir paralléliser cette fonction. Mais les réseaux proposés restent très complexes avec des performances assez éloignées du modèle simplifié de référence HM-RDOQ. En vue d'une implémentation matérielle, nous proposons une solution minimisant la complexité, tant au niveau du nombre de paramètres que de l'architecture du réseau. Notre modèle possède moins de 2000 paramètres et atteint 2.90 % de réduction de BD-rate sur la luminance par rapport à l'algorithme HM sans RDOQ dépassant même les résultats de HM-RDOQ.

Abstract – RDOQ (*Rate Distorsion Optimized Quantization*) is an efficient tool used in several video codecs such as H.264/AVC, H.265/HEVC or AV1. Although this function can significantly reduce the bitrate, its complexity is a limitation for hardware video coding solutions. Previous studies have succeeded in simplifying RDOQ allowing its integration into HM, the software reference implementation of H.265/HEVC. However, its iterative and sequential structure remains non-hardware-friendly. With the advent of machine learning, neural networks (NN) have been introduced to mimic the RDOQ algorithm in a parallel way. Proposed NNs are still highly complex with results below the simplified model HM-RDOQ. In order to use it into hardware implementations, our approach targets a balanced sizing in terms of the number of parameters and architecture reuse. Our 2000-parameter NN reaches a 2.90 % BD-Rate saving on luminance with respect to the HM algorithm without RDOQ, even exceeding HM-RDOQ.

1 Introduction

La vidéo constitue environ 80 % du flux Internet mondial, ce qui fait de la compression vidéo un enjeu majeur. Afin de proposer des codecs toujours plus performants, plusieurs outils sont proposés à chaque nouvelle génération. Même si ces outils peuvent amener à des gains très intéressants, certains sont assez peu exploités dans le domaine industriel. Implémenter des codecs comme HEVC (*High Efficiency Video Coding*) [1] dans des solutions d'encodage matériel nécessite une faible complexité et des opérations parallélisables afin de limiter la surface et la bande passante des puces électroniques. Il existe des codecs plus récents comme VVC (*Versatile Video Coding*) mais qui ne sont pour le moment pas utilisés dans ce type de solutions.

Avec l'essor de l'apprentissage automatique, plusieurs travaux ont cherché à améliorer les codecs vidéo traditionnels avec de l'apprentissage profond (*deep learning*) via des approches hybrides : en remplaçant certains blocs comme des filtres de boucle [2] ou encore, en optimisant des choix d'encodage comme la prédiction de mode intra [3]. Dans notre approche, c'est le bloc de quantification d'HEVC que l'on améliore avec du *deep-learning*, mais dans une optique de performance matérielle.

L'objectif de la compression vidéo est de trouver le meilleur compromis entre compression et qualité selon les contraintes du support. On parle d'optimisation sous contraintes du débit et de la qualité que l'on peut noter RDO (*Rate Distorsion Optimization*) [4] et que l'on peut représenter sous la forme d'un lagrangien à minimiser, voir équation 1. Avant de compresser la vidéo, le codec HEVC partitionne chaque image en blocs de différentes tailles (4x4, 8x8, 16x16 et 32x32) en fonction du niveau de détail local. Chaque bloc va alors passer dans la chaîne d'encodage, présentée sur la figure 1. Premièrement le codec élimine les redondances spatiales et temporelles en encodant l'erreur entre la source et une prédiction. Puis les différences obtenues sont envoyées dans le processus de transformation-quantification. Cette étape consiste à concentrer l'information apportée par les résidus grâce à une transformée (*Discrete Cosine/Sine Transform* dans le cadre d'HEVC) avant de quantifier chaque coefficient. L'étape de quantification est cruciale puisqu'elle introduit la seule perte irréversible qui joue un rôle important dans la RDO.

La RDOQ (*Rate Distortion Optimized Quantization*) est une procédure alternative de quantification conforme au standard d'HEVC qui obtient de meilleurs résultats que la quantification scalaire (SQ) traditionnellement utilisée [5]. Le principe de la RDOQ consiste à trouver les niveaux optimaux de quantification afin de réaliser le meilleur compromis entre débit et distorsion. Autrement dit, la RDOQ quantifie chaque coefficient transformé de sorte à minimiser le lagrangien :

$$J = D(x_{source}, x_{recon}) + \lambda R(q) \quad (1)$$

où D représente une mesure de distorsion entre le bloc source x_{source} et le bloc reconstruit noté x_{recon}. Le terme R représente la contribution au débit de ce bloc quantifié noté q. En

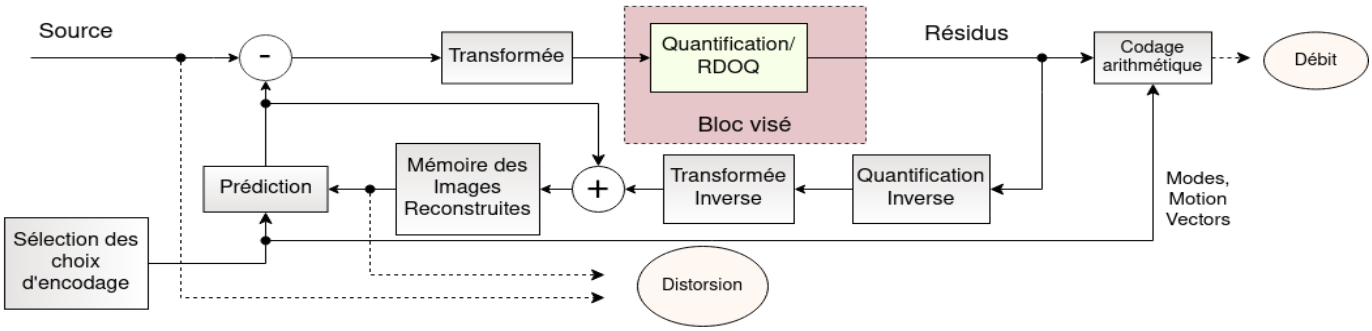

FIGURE 1 : Schéma détaillé de la chaîne d'encodage

fonction de l'importance que l'on veut apporter à la qualité ou au débit, on introduit un facteur λ pour pondérer le terme lié au débit. L'approche théorique et optimale de la RDOQ nécessite de tester tous les niveaux de quantification possibles pour chaque bloc de coefficients transformés ce qui n'est pas envisageable. Plusieurs études ont été menées comme dans [5] afin de simplifier la RDOQ de façon à pouvoir l'implémenter dans le modèle de référence HM 16.25 d'HEVC. Cette version simplifiée présente dans le HM (HM-RDOQ) effectue une estimation pour calculer le débit c'est à dire le terme $R(q)$ dans l'équation 1. Cette estimation utilise des tables entropiques mises à jour à chaque passage dans la fonction RDOQ.

Pour éviter un comportement séquentiel, les auteurs de [6] et de [7] proposent d'utiliser un réseau de neurones imitant l'algorithme de RDOQ. Le réseau prend une décision à la volée sans dépendre d'éléments extérieurs. L'architecture obtenant les meilleurs résultats est le FCNN (*Fully Convolutional Neural Network*) en atteignant 45 % de la performance de HM-RDOQ en termes de BD-Rate [8]. Toutefois le réseau proposé est de l'ordre de plusieurs millions de paramètres avec une architecture différente pour chaque taille de bloc. En plus de la complexité, plusieurs éléments viennent à manquer dans les deux précédentes études, [6] ne précise pas les gains en termes de BD-Rate et [7] ne considère pas les blocs 32x32 et les métriques utilisées sont peu explicitées.

On se propose dans ce travail de rappeler premièrement le fonctionnement de l'algorithme de référence HM-RDOQ ainsi que des approches utilisant des réseaux de neurones. On introduira ensuite notre modèle, le LC-FCNN que l'on comparera à deux autres modèles inspirés des précédents travaux [6] et [7]. Nous comparerons les résultats en terme de taux de réduction d'erreur ou ERR (*Error Reduction Ratio*) avant d'intégrer notre modèle dans l'implémentation HM afin d'évaluer ses performances en termes de BD-Rate.

2 Implémentations de la RDOQ

2.1 Approche simplifiée HM-RDOQ

L'algorithme de la RDOQ implémenté dans le HM comprend trois étapes comme rappelé dans [5] :

1. Étape itérative : Pour chaque coefficient quantifié $q_{i,j}$ issu d'une quantification scalaire donnée par l'équation 2, on teste différentes valeurs alternatives en partant du dernier coefficient non nul avec Q_{step} un terme lié au paramètre de quantification (QP), $C_{i,j}$ les coefficients transformés et Θ un *offset* ajusté en fonction de l'arrondi désiré.

$$q_{i,j} = \left\lfloor \frac{|C_{i,j}|}{Q_{step}} + \Theta \right\rfloor \tag{2}$$

   Pour savoir si on doit faire un changement, on teste les différents candidats $l_{i,j} \in L_{i,j} = \{0, q_{i,j}, max(0, q_{i,j} - 1)\}$.

2. La deuxième phase consiste à éliminer tout un groupe de coefficients (dit CG).

3. La dernière étape consiste à trouver la meilleure position pour le dernier coefficient non nul. Cette étape est nécessaire pour encoder les coordonnées de ce dernier.

Pour chacune de ces étapes les choix sont faits selon l'équation 1 où la distorsion est calculée dans le domaine fréquentiel ce qui évite d'avoir à effectuer la quantification et transformation inverses. Le débit est estimé à partir de tables entropiques mises à jour après chaque passage dans la RDOQ, ce qui d'un point de vue matériel, n'est pas souhaitable.

2.2 Approche par apprentissage automatique

L'idée est d'utiliser des modèles à base d'apprentissage automatique pour imiter le comportement de l'algorithme HM-RDOQ : on parle d'*imitation learning*. Le réseau de neurones va apprendre à prédire les mêmes changements que la vérité terrain, ici HM-RDOQ, sans avoir à calculer un lagrangien complexe, qui peut nécessiter des estimations via des tables entropiques mises à jour itérativement.

Les réseaux de neurones convolutifs (dits CNN) semblent être les plus adaptés à cette tâche. En effet, comme expliqué dans [6], la raison pour laquelle on peut utiliser des CNN pour prédire la RDOQ est qu'il existe une corrélation locale au niveau des coefficients quantifiés. L'étape traitant chaque groupe de coefficients 4x4 peut également être approchée par des filtres. Ainsi les CNN peuvent apprendre à prédire des coefficients quantifiés de la RDOQ en se basant sur les caractéristiques locales et contextuelles d'un bloc.

Le réseau proposé par [6] prend en entrée les coefficients transformés notés x ainsi que les coefficients quantifiés et prédit la variation à appliquer à ces derniers pour obtenir la RDOQ. Plus précisément si q_{SQ} représente les coefficients issus de la quantification scalaire, alors le réseau ajuste de Δ ces derniers afin d'imiter q_{RDOQ} d'où : $\hat{q} = \Delta + q_{SQ} \approx q_{RDOQ}$. Les

valeurs d'ajustement Δ sont comprises dans l'ensemble $\{0, 1\}$. Le réseau prédit ainsi pour chaque coefficient la probabilité que Δ soit égal à 0 ou 1. La fonction de coût est l'entropie croisée définie par $L = -E_{x,q_{SQ}}[\log P(\Delta|x, q_{SQ})]$.

3 CNN de faible complexité

Nous avons décidé de tester 3 modèles en tenant compte des précédents travaux. Notre domaine s'inscrit dans le cadre de la segmentation sémantique où le but est d'associer une classe à un pixel donné. Les architectures connues dans ce domaine peuvent être utilisées dans notre cas. La différence réside dans le type de données puisqu'on travaille ici avec des blocs fréquentiels. Pour les 3 modèles nous avons uniquement utilisé *ReLU* ainsi que *Sigmoid* sur la dernière couche.

Le premier modèle est de type VGG comme le propose [6]. Nous avons opté pour une implémentation plus épurée de la version originale du VGG comprenant deux blocs de convolutions/maxpooling avec un nombre de canaux resp. de 64 et 128, suivi d'un réseau dense. Comme ce modèle comporte un réseau dense, son nombre de paramètres augmente avec la taille des blocs limitant ainsi son utilisation pour des blocs de tailles 16x16 et 32x32. Nous testerons ainsi ce modèle uniquement sur les blocs de taille 4x4 et 8x8.

Le deuxième modèle est inspiré de [7] et reprend un FCNN sans opération de maxpooling. L'avantage d'une architecture FCNN est qu'elle ne dépend pas de la taille des entrées. On peut donc ré-utiliser la même architecture pour les 4 types de blocs afin d'alléger une potentielle implémentation matérielle. Nous avons choisi une implémentation proche de [7] avec 4 étages de convolutions avec 256 canaux chacun (excepté sur le dernier étage) pour un total de 1 185 281 paramètres.

Enfin nous proposons un modèle LC-FCNN, (voir figure 2), avec plus de couches de convolutions que le deuxième modèle mais avec beaucoup moins de canaux par couche. En augmentant la taille des champs récepteurs (*receptive fields*), le réseau prend en compte plus de voisins pour le choix final justifiant l'augmentation du nombre d'étages au profit du nombre de canaux. Nos expérimentations ont montré qu'un modèle possédant 5 couches de convolutions comptant 8 canaux par étage était un bon compromis entre performance et complexité. Cela réduit le nombre de paramètre a 1913.

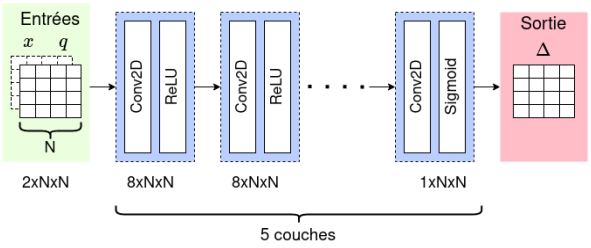

FIGURE 2 : Schéma du LC-FCNN

4 Expérimentations

4.1 *Dataset*

Pour les données (*datasets*) d'entraînement nous avons utilisé des séquences issues de Xiph [9] à plusieurs résolutions et contenant toutes le même nombre d'images. Nous en avons retenu 112 en privilégiant la diversité de contenu. Chaque séquence est encodée pour 4 valeurs de QPs : 22, 27, 32 et 37 en utilisant uniquement le mode intra pour l'encodage et en ne travaillant qu'avec des coefficients de luminance pour simplifier l'étude. On obtient alors 16 *datasets*, un pour chaque taille de blocs et pour chaque QP. Chaque *dataset* contient les données suivantes : coefficients transformés, quantifiés ainsi que la vérité terrain associée, à savoir les coefficients en sortie de l'algorithme d'HM-RDOQ. La table 2 récapitule le nombre de données par *dataset*.

4.2 Métrique

Nous avons choisi d'utiliser la métrique dite d'Error Reduction Ratio (ERR) initialement introduite dans [6]. Cette métrique calcule le ratio entre deux distances de Hamming : l'erreur de prédiction du réseau $err_{DL} = \sum \|q_{DL} - q_{RDOQ}\|$ et l'erreur de prédiction du modèle de référence sans RDOQ : $err_{SQ} = \sum \|q_{SQ} - q_{RDOQ}\|$.

$$ERR = \frac{err_{DL}}{err_{SQ}} \tag{3}$$

Dans la suite on considérera plutôt $1 - ERR$ afin d'avoir une métrique qui tende vers 1 quand l'erreur de prédiction diminue. Autrement dit, $1 - ERR$ mesure le pourcentage de changements bien prédits par rapport au modèle de référence. Les résultats en ERR sont bornés contrairement au BD-Rate. Le HM-RDOQ étant simplifié, il est possible que les erreurs du réseau puissent améliorer les résultats du BD-Rate.

4.3 Hyper-paramètres

En terme d'hyper-paramètres on utilise l'optimiseur Adam, avec un *learning rate* de 10^{-4} et un *batch* différent pour chaque type de blocs, 4096 pour les blocs 4x4, 2048 pour les blocs 8x8 et 16x16 et 512 pour les blocs 32x32. Le nombre d'époques est fixé à 10.

5 Résultats

5.1 Résultats en terme d'ERR

Les résultats de la table 1 montrent que notre modèle LC-FCNN a de très bons résultats pour tous les types de blocs atteignant entre 64.9 % et 72.5 % de bonnes prédictions par rapport à la référence, surpassant le modèle VGG et même le FCNN pour les blocs 8x8. Le FCNN obtient les meilleures performances pour les blocs 4x4, 16x16 et 32x32 mais l'écart avec le LC-FCNN est de l'odre de 1 à 3 % d'ERR. Notre modèle LC-FCNN obtient presque les mêmes performances que le FCNN avec moins de 2000 paramètres contre plus d'un million, laissant présager que les modèles proposés dans les précédents articles étaient sur-dimensionnés. On note des résultats similaires pour les autres QPs. Même si l'ERR donne une bonne indication des performances, elle n'indique pas le gain en débit.

5.2 Résultats en BD-Rate

Afin d'obtenir les résultats en BD-Rate de notre modèle, nous l'avons implémenté dans le modèle HM. La table 3 montre que

Taille	4x4			8x8			16x16			32x32		
Modèle	VGG	FCNN	LC-FCNN	VGG	FCNN	LC-FCNN	VGG	FCNN	LC-FCNN	VGG	FCNN	LC-FCNN
BasketDrill	0.533	0.712	0.710	0.743	0.793	0.823	-	0.803	0.780	-	0.742	0.746
BQTerrasse	0.516	0.662	0.656	0.619	0.685	0.733	-	0.728	0.703	-	0.688	0.649
Sunflower	0.579	0.679	0.639	0.666	0.640	0.644	-	0.592	0.562	-	0.516	0.513
Flowervase	0.476	0.642	0.635	0.597	0.629	0.660	-	0.643	0.612	-	0.658	0.637
InToTree	0.610	0.744	0.739	0.710	0.718	0.762	-	0.702	0.670	-	0.693	0.697
Moyenne	0.542	**0.687**	0.676	0.670	0.693	**0.725**	-	**0.694**	0.666	-	**0.660**	0.649

TABLE 1 : Résultats $1 - ERR$ pour QP=32. Plus les valeurs sont proches de 1, meilleur est le résultat.

QPs / Taille de Blocs	22	27	32	37
4x4	73.1	47.9	30.6	17.5
8x8	30.4	20.4	15.4	11.3
16x16	7.83	7.68	6.75	6.38
32x32	2.99	3.71	4.04	4.17

TABLE 2 : Taille des différents *datasets* (en millions)

notre modèle faible complexité obtient des résultats meilleurs que la fonction HM-RDOQ ciblée. Notre modèle obtient de meilleurs résultats que l'état de l'art avec - 2.90 % de BD-Rate par rapport à HM sans RDOQ contre -2.29 % pour HM-RDOQ, tout en ayant moins de 2000 paramètres contre plusieurs millions pour les précédents réseaux proposés. La légère sur-performance du modèle LC-FCNN se situe à haut débit, là où la fonction RDOQ apporte les gains les plus importants, voir figure 3. Cela vient du fait que la fonction simplifiée d'HM-RDOQ n'a pas été optimisée et semble élaguer trop de coefficients à haut débit contrairement au LC-FCNN.

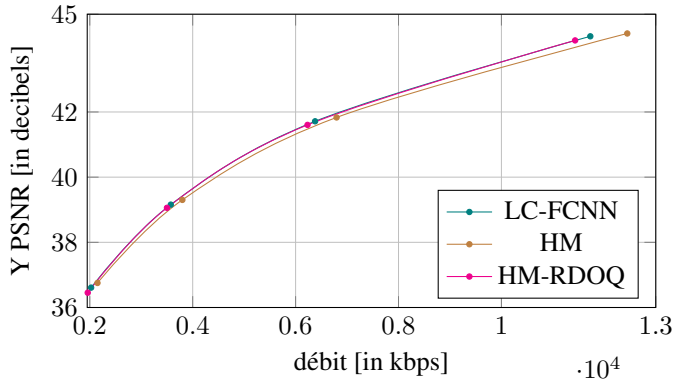

FIGURE 3 : Évolution du PSNR/débit pour vidyo-720p

6 Conclusion

En vue d'une implémentation industrielle, nous avons proposé un modèle de *deep-learning* basse complexité afin d'imiter la fonction de RDOQ. Notre modèle faible complexité atteint -2.90 % de BD-Rate et surpasse même HM-RDOQ pour moins de 2000 paramètres. La prochaine étape est de proposer un modèle multi-QP, c'est à dire d'entraîner un seul réseau fonctionnant quelle que soit la valeur du QP.

Séquence	HM-RDOQ	LC-FCNN
DOTA2-1080p	-1.5262	-2.2124
Factory-1080p	-1.7550	-2.4176
Fallout-1080p	-1.3603	-2.3231
Hearthstone-1080p	-1.5515	-2.3628
KristenAndSara-720p	-2.4407	-2.9790
MobCal-720p	-2.3862	-3.1781
RushFields-1080p	-2.3172	-3.4589
Shields-720p	-2.4759	-3.3276
Stockholm-720p	-3.1613	-3.8335
intotree-1080p	-3.0291	-2.8413
parkrun-720p	-2.4527	-2.8864
pedestrian-1080p	-2.312	-2.7425
vidyo4-720p	-3.0250	-3.2080
Moyenne	-2.2918	**-2.9054**

TABLE 3 : Résultats du modèle en BD-Rate (en %) pour la luminance en mode de prédiction Intra seulement. Chaque séquence est encodée pour 4 QPs (22,27,32 et 37) et comprend les 60 premières images.

Références

[1] G. J. SULLIVAN et al. "Overview of the High Efficiency Video Coding (HEVC) Standard". In : *IEEE Transactions on Circuits and Systems for Video Technology* (déc. 2012).

[2] S. KUANAR et al. "Deep learning based HEVC in-loop filter and noise reduction". In : *Signal Processing : Image Communication* (nov. 2021).

[3] T. LAUDE et J. OSTERMANN. "Deep learning-based intra prediction mode decision for HEVC". In : *Proceedings of the Picture Coding Symposium (PCS)* (déc. 2016).

[4] G. J. SULLIVAN et T. WIEGAND. "Rate-distortion optimization for video compression". In : *IEEE Signal Processing Magazine, vol. 15, no. 6* (nov. 1998).

[5] J. STANKOWSKI et al. "Rate-distortion optimized quantization in HEVC : Performance limitations". In : *Proceedings of the Picture Coding Symposium (PCS)* (juin 2015).

[6] T. N. CANH, M. XU et B. JEON. "Rate-Distortion Optimized Quantization : A Deep Learning Approach". In : *IEEE High Performance Extreme Computing Conference* (sept. 2018).

[7] D. KIANFAR et al. "Parallelized Rate-Distortion Optimized Quantization Using Deep Learning". In : *IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)* (sept. 2020).

[8] G. BJØNTEGAARD. "Document VCEG-M33 : Calculation of average PSNR differences between RD-curves". In : *ITU-T VCEG Meeting, USA, Tech. Rep,* (avr. 2001).

[9] *Xiph.org video test media*. 2017. URL : `https://media.xiph.org/video/derf`.