Contraction d'une image en une petite séquence de tokens avec des modules d'attention croisée

Natacha Luka, Romain Negrel, David Picard

Contraction d'une image en une petite séquence de tokens avec des modules d'attention croisée

Natacha LUKA[1] Romain NEGREL[1] David PICARD[1]

[1]LIGM, École des Ponts, Université Gustave Eiffel, ESIEE Paris, CNRS, Marne-La-Vallée, France

Résumé – Dans cet article, nous introduisons une méthode de contraction d'une image en une petite séquence entièrement basée sur des modules d'attention de type *transformer* décodeur. Nous introduisons le concept d'*image queries* : des tokens prototypes appris qui vont agréger l'information contenue dans l'image d'entrée, préalablement découpée en une séquence de patchs, formant ainsi une séquence de taille réduite utilisable pour d'autres applications. Contrairement aux autres modèles d'apprentissage profond utilisant des modules d'attentions en compression d'image, notre architecture ne contient aucune convolution.

Abstract – In this paper, we introduce a novel deep image sequence reduction method solely based on attention *transformer* decoder modules. Contrarily to common deep learning image compression using attention mechanism, our architecture does not use any convolution. Besides, we introduce our concept of learned *image queries*, which aggregates the information of the input image previously partitioned in a sequence of patchs, to obtain a smaller sequence which can be re-used for various downstream tasks.

1 Introduction

Avec l'omniprésence et la croissance exponentielle des images numériques, développer des méthodes de compression offrant le meilleur compromis entre taille et qualité devient fondamental. Cela permet entre autre d'optimiser l'espace de stockage nécessaire à leur conservation et la bande passante requise pour leur transfert. Alors que le grand public continue d'utiliser majoritairement le format JPEG, de nouveaux algorithmes ont été développés dont JPEG2000, WebP [9] ou BPG [2] (basé sur le standard de compression vidéo HEVC [12]).

Parallèlement, des méthodes de compression fondées sur les méthodes d'apprentissage profond se développent, notamment celles s'appuyant sur des RNNs [14], des réseaux génératifs adverses [1] ou des CNNs. Les méthodes utilisant les CNNs montrent des résultats compétitifs. Cependant, les CNNs sont fortement concurrencés dans de nombreuses tâches par des architectures de type *transformer* [15]. Il est donc légitime de se demander si ce type d'architecture peut être aussi efficace en compression.

Dans les approches basées sur les CNNs, des auto-encodeurs convolutionels ont d'abord été combinés avec une quantification et un codage entropique [3]. Ballé *et al.* [4] ont ensuite proposé d'ajouter un réseau de neurones pour prédire des priors pour le codage entropique et en améliorer l'efficacité. Puis des architectures hybrides combinant couches convolutionelles et modules d'attention ont fait leur apparition [6, 5], ou même des modules de type *transformer*[11, 10].

Dans cet article, nous proposons un encodeur d'image entièrement basé sur des modules d'attention sans aucun bloc de convolution. Nous n'avons pas implémenté les opérations de quantification et de codage entropique, qui pourront toutefois se greffer sur l'architecture proposée. Notre encodeur contracte simplement une séquence de patchs en une nouvelle séquence de taille réduite.

Nos principales contributions sont 1) une architecture encodeur-décodeur d'images **composée uniquement de modules d'attentions de type *transformer*** [15] pour la compression d'une image, représentée comme une séquence de patchs 2) l'introduction des *images queries*, qui sont des *tokens* prototypes appris servant à agréger l'information contenue dans une image divisée en une séquence de *patchs*. La réduction de la taille de la séquence vient du fait que le nombre d'*images queries* est beaucoup plus petit que le nombre de *patchs*, ce qui présente également un avantage sur le plan calculatoire.

Dans la suite de cet article, nous introduirons en 2 les modules d'attention du *transformer*, voûte de notre architecture qui sera présentée en 3 avec notre méthode d'entraînement. Avant de conclure, nous exposerons en 4 différents éléments de résultats qui montrent que notre méthode uniquement basée sur l'attention est prometteuse si la méthode d'entraînement et l'architecture sont bien optimisées. Nous verrons également que nos *images queries* se spécialisent spatialement.

2 Transformers

Les modules de type *transformer* ont d'abord été introduits en traitement automatique des langues [15, 7] avant de l'être en vision [8] où ils ont progressivement remplacé les réseaux de convolutions dans un grand nombre de tâches.

Les architectures fondées sur les *transformers* reposent sur un enchaînement de blocs qui implémentent un processus d'attention sur deux séquences de vecteurs. L'idée est d'agréger les éléments d'une séquence de vecteurs *values* (rassemblés dans une matrice V), dans une autre séquence de vecteurs *queries* (rassemblés dans une matrice Q). L'agrégation est faite en fonction de certaines caractéristiques définies par une projection de la séquence de *values* pour donner une séquence de *keys* (rassemblés dans une matrice K). Ainsi, les vecteurs de K contiennent les caractéristiques que les vecteurs *queries* et *values* doivent avoir en commun pour être agrégés ensemble. Pour évaluer la présence plus ou moins importante de ces caractéristiques dans un vecteur de Q, il faut calculer des scores de similarité entre un vecteur de K et ce vecteur de Q. Un score de similarité entre deux vecteurs est simplement une corréla-

tion entre ces deux vecteurs. Les corrélations entre les vecteurs de *queries* et les vecteurs de *keys* vont servir de poids pour sommer les éléments de *values*. Puis, grâce à une connexion résiduelle, les *queries* vont au fur et à mesure agréger les *values* qui lui sont le plus similaires selon les caractéristiques de K. Pour la couche n, nous avons donc l'équation suivante :

$$Q^{n+1} = Q^n + \underbrace{\text{softmax}\left[Q^n \left(K^n\right)^T\right]}_{A^n = \left(a_{ij}^n\right)_{i,j}} V^n \tag{1}$$

La matrice A^n est la matrice d'attention. En effet, par l'application du softmax, la matrice contient sur chaque ligne i des poids d'attention $\{a_j\}_j$ qui mesurent l'importance que la *query* i doit accorder à la *key* j. Une somme pondérée est ensuite effectuée :

$$Q^{n+1} = Q^n + A^n V^n \tag{2}$$

$$= Q^n + \left(\sum_k a_{ik}^n v_{kj}^n\right)_{i,j} \tag{3}$$

Il existe deux types de module d'attention : ceux d'auto-attention, durant lesquels *queries* et *values* sont identiques à une projection près, et que nous noterons $\text{SA}\left[Q\right]$. Les seconds sont ceux d'attention croisée, durant lesquels les *queries* sont différentes des *values*. Nous les noterons $\text{CA}\left[Q, V\right]$. Au sein de ces modules, l'attention peut être multi-têtes si plusieurs projections K sont considérées pour pouvoir sélectionner plusieurs *values* et les agréger au sein d'une même *query*.

3 Méthode

Notre méthode consiste à se passer de convolutions dans l'encodeur et le décodeur d'image pour n'utiliser que des blocs de type *transformer* décodeur[15]. Un bloc consiste en un module de SA suivi d'un module de CA et d'un perceptron multicouches.

Architecture L'architecture est présentée à la figure 1. Une image I est découpée en P patchs de taille $h_p \times w_p$ qui sont convertis en une séquence de vecteurs $\{\varphi\left(I\right)\}_{1 \leqslant p \leqslant P}$ à l'aide d'une transformation φ. Dans notre cas, le rôle de φ est juste d'aplatir les patchs de dimension $h_p \times w_p \times c$ vers une dimension $h_p w_p c$. c correspond au nombre de canaux de l'image ($c = 3$ dans la suite), Ainsi, nous disposons d'une séquence de taille P qui va servir, selon la terminologie des *transformers* et après projection, de *keys*/*values* (K_E et V_E sur la figure) pour notre encodeur d'image E. Celui-ci est un enchaînement de l blocs de type *transformer* décodeur.

Dans notre cas, les N *queries* données en entrée du *transformer* sont des vecteurs prototypes appris que nous appelons *image queries* et que nous notons Q_E^{in}. À l'origine, ils sont donc identiques quelque soit l'image. Cependant, au fur et à mesure des l blocs, les *queries* vont agréger l'information contenue dans l'image via les équations 4, 5 et 6, avec FFW un perceptron multicouches :

$$Q^{n+1/3} = \text{SA}\left[Q^n\right] \tag{4}$$

$$Q^{n+2/3} = \text{CA}\left[Q^{n+1/3}, \varphi(I)\right] \tag{5}$$

$$Q^{n+1} = Q^{n+2/3} + \text{FFW}^n(Q^{n+2/3}) \tag{6}$$

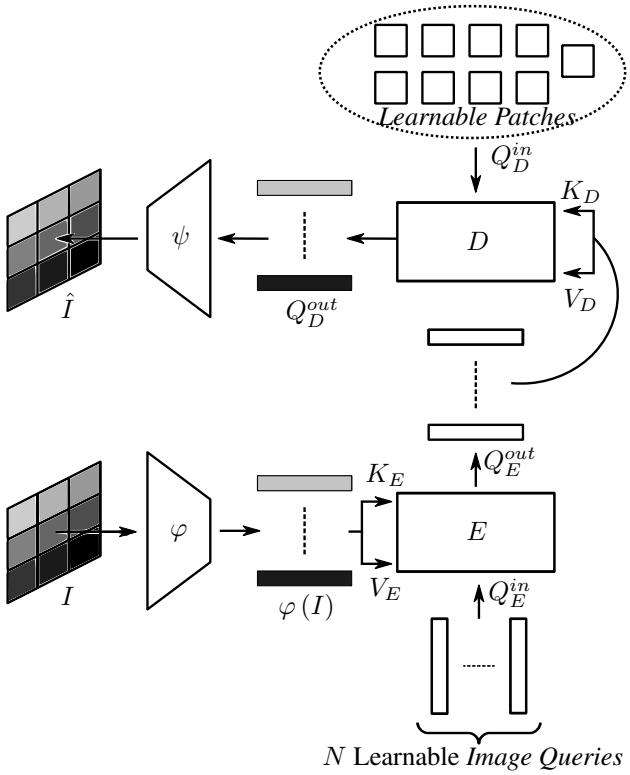


FIGURE 1 : Architecture de l'encodeur d'image.

Une fois ce processus répété l fois, les *queries* transformées, notées Q_E^{out}, contiennent l'image dans une séquence de taille réduite.

Pour le décodage, le processus est identique mais les *queries* sont inversées avec les *keys*/*values* : les *queries* obtenues en sortie de E sont utilisées comme *keys*/*values* dans le décodeur D, qui est également un ensemble de l blocs de type *transformer* décodeur dimensionnés comme ceux de E. Des vecteurs patchs prototypes – des « patchs *queries* » – sont de la même manière appris pour servir de *queries* à D. Ces patchs prototypes vont alors agréger l'information contenue dans les *queries* de sortie de E, qui contiennent l'image compressée. En redimensionnant les patchs et en les ré-assemblant via ψ nous pouvons reconstruire $\hat{I}$, l'image décodée de I. Nous avons ici systématiquement utilisé des architectures à 4 têtes avec un nombre de blocs $l = 4$.

Entraînement Pour l'apprentissage, nous avons optimisé dans un premier temps une loss perceptuelle (implémentation lpips [17] avec le backbone VGG-16 pré-entraîné). En effet, cette fonction de coût a montré son efficacité pour la synthèse d'images naturelles. Son but est de minimiser une norme L2 sur les features extraites des premiers blocs de VGG-16. La taille des images utilisées étant de 128×128, l'optimisation n'était pas optimale car VGG a été entraîné sur des images plus grandes. Une meilleure optimisation a été obtenue en agrandissant les images d'entrée avec une interpolation bilinéaire vers une taille 256×256 (section 4).

Durant cette première phase, nous nous sommes servi d'un ordonnanceur de pas d'apprentissage avec une augmentation linéaire pendant les $30k$ premières itérations jusqu'à un maximum de 10^{-4} avant de décroître, linéairement, durant $70k$ itérations. Le modèle a été entraîné sur ImageNet avec des images re-dimensionnées de taille 128×128.

Dans une seconde phase, l'erreur quadratique moyenne

Taille de Patch	PSNR ↑	MS − SSIM ↑	lpips ↓
8x8 ($d = 192$)	18.10	0.664	0.365
16x16 ($d = 768$)	**21.56**	**0.856**	**0.236**

TABLE 1 : Comparaison entre différentes tailles de patch à $77.5k$ itérations - *ImageNet-val*

(MSE) a été optimisée pendant $100k$ itérations avec un taux d'apprentissage constant pour améliorer le PSNR.

4 Résultats

Taille de patch Nous avons mené une étude pour déterminer le meilleur compromis entre la largeur de l'architecture (dimension interne) et la taille de patch $h_p \times w_p$. Par le choix de φ, la dimension intérieure d est égale à $h_p w_p c$ et la taille P de la séquence de patchs est de $\frac{hw}{h_p w_p}$: il s'agit donc de choisir, à budget binaire constant sur le Q_E, entre beaucoup de tokens Q_E de faible dimension ou peu de tokens Q_E de plus grande dimension (plus h_p et w_p croîssent, plus d croît, mais plus le nombre de Q_E diminue et vice versa).

Nous avons choisi un ratio de contraction de 1.25 :1 pour réduire le nombre de bits de l'image sans avoir une perte d'information trop importante dans le cadre de cette étude. Notons que ce facteur pourrait être augmenté par de la quantification et du codage entropique (non traités ici). Quant à la taille des patchs, nous avons choisi $h_p \times w_p = 8 \times 8$ et $h_p \times w_p = 16 \times 16$. Le calcul du nombre de Q_E s'effectue donc de la manière suivante (l'architecture est en 32bits, une valeur est donc codée sur 4 octets) :

$$N = \left\lfloor \frac{1}{1.25} \left(\frac{hwc}{4d} \right) \right\rfloor = \left\lfloor \frac{hwc}{5d} \right\rfloor = \left\lfloor \frac{P}{5} \right\rfloor \qquad (7)$$

Soit $N = 51$ ou $N = 12$ si la taille des patchs est respectivement de 8×8 ou de 16×16.

Nos résultats sont dans le tableau 1. Les métriques utilisées sont lpips[17] ainsi que le PSNR et la MS-SSIM [16] (implémentation de **torchmetrics**). Du fait de la taille des images, nous avons supprimé une échelle lors du calcul de la MS-SSIM pour ne garder que les 4 plus petites.

Nous obtenons un net gain en utilisant des patchs de taille 16×16, ce qui correspond à une petite séquence Q_E de grande dimension. Notre hypothèse est que la dimension dans une tête du module d'attention est trop faible dans le cas des patchs 8×8 : pour préserver la largeur du *transformer* à toutes les étapes, la dimension dans une tête est de $\frac{d}{H}$ avec H le nombre de tête. Ainsi dans le cas des patchs de taille 8×8, les dimensions par tête pourraient être un facteur limitant.

Comparaison entre lpips vs lpips avec augmentation et gain grâce à la MSE La taille de patch étant fixé à 16×16, nous avons entraîné notre modèle de trois manières différentes : 1) avec lpips sans agrandissement, 2) avec lpips entre les images 128×128 à reconstruire et celles reconstruites, après leur agrandissement à la taille 256×256, 3) en optimisant la MSE en second lieu, après optimisation de lpips avec agrandissement. En effet, si dans un premier temps lpips s'avère être une fonction de coût intéressante, nous avons constaté qu'elle devient difficile à optimiser après les $100k$ premières itérations. En revanche, la MSE s'optimise mieux après ces itérations initiales et permet d'obtenir un gain de 5dB sur le PSNR mais aussi des gains sur la MS-SSIM et lpips (non optimisées), comme le montre le tableau 2.

Entraînement	PSNR ↑	MS − SSIM ↑	lpips ↓
lpips	15.42	0.643	0.196
lpips_{up}	21.56	0.856	0.236
lpips_{up} & MSE	**26.646**	**0.953**	**0.131**

TABLE 2 : Comparaison entre lpips, lpips avec agrandissement 256×256 (lpips_{up}) et lpips_{up} suivi d'une optimisation de la MSE (lpips_{up} & MSE) sur *ImageNet-val*

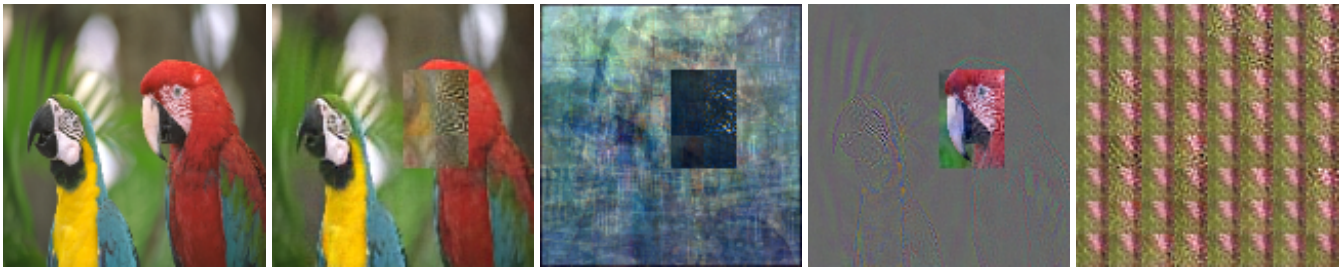
Flottants des Q_E	PSNR ↑	MS − SSIM ↑	lpips ↓
16bits	28.58	0.983	0.138
32bits	28.58	0.983	0.138

TABLE 3 : Résultats sur Kodak avec Q_E^{out} en 32bits ou convertis en 16bits sont identiques

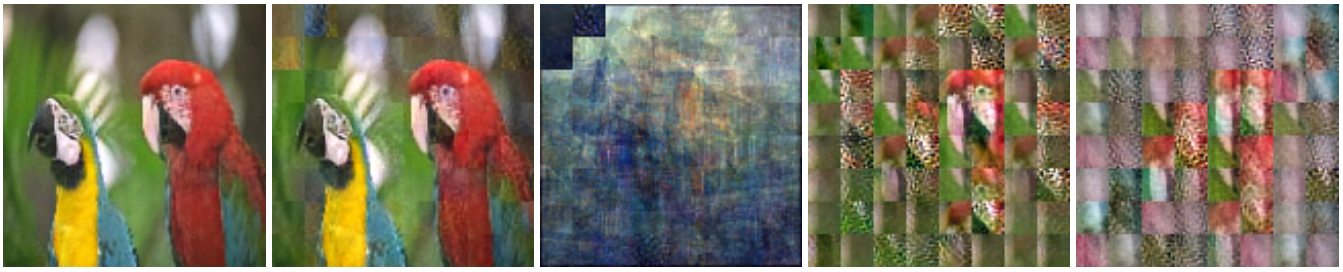
Ce qui est surprenant, c'est que quand nous utilisons la méthode 2), le PSNR et de la MS-SSIM augmentent mais la valeur de lpips augmente également par rapport à la méthode 1). Nous émettons l'hypothèse que sur des images de taille 128×128, ne correspondant pas à la taille des images sur lesquelles VGG-16 est pré-entraînée, les réponses des filtres sont différentes. Par ailleurs, les éléments prépondérants dans lpips peuvent être différents par échelle : des défauts inaperçus par VGG-16 à petite résolution ne pénalisent pas la métrique mais pénalisent le PSNR et la MS-SSIM. Par contre, ces mêmes défauts peuvent être davantage pris en compte par lpips en plus haute résolution.

Résultats sur Kodak Un exemple de reconstruction sur une image du jeu de données de Kodak [13] avec le modèle entraîné selon la méthode 3 (lpips_{up} & MSE), est visible en figure 2f. En moyenne, nous obtenons des images visuellement correctes pour l'oeil humain avec un PSNR moyen de 28.58dB. L'ensemble des résultats est dans le tableau 3. Avec de la quantification et un codage entropique, le taux de compression réel de notre méthode pourrait s'avérer bien plus élevé que le 1.25 :1 choisi pour l'architecture et donc avoir des applications intéressantes en compression. Par ailleurs, les Q_E^{out} sont probablement quantifiables car la conversion des flottants 32 bits vers 16 bits ne change pas les métriques.

Analyse de l'information codée par les tokens de la séquence $\mathbf{Q_E}$ Pour analyser l'information encodée par chaque token de la séquence Q_E, nous avons procédé à la reconstruction d'images avec la même configuration que précédemment (méthode 3) en supprimant un token ou avec un unique token. La figure 2 montre que les tokens encodent bien de l'information spatiale. Cependant, seuls, ils ne suffisent pas pour reconstituer la zone qu'ils encodent : ils ont besoin des informations contenues dans les autres tokens. Sur la figure 2e, une partie du bec du perroquet et du vert provenant probablement de l'arrière-plan sont décodées sur tous les patchs mais la zone encodée par la *query* n'est pas décodée convenablement. En pratique, il faut au moins deux *queries* pour décoder une information spatiale comme le montrent les figures 2i et 2j. Plus particulièrement, les figures 2g,2c, 2i et 2j nous enseignent que la *query* $\{Q_E^{out}\}_9$ semble encoder une information plus globale que les autres, en contrepartie d'une faible information spatiale.



(a) Originale en 128×128 (b) Recons. sans $\{Q_E\}_{11}$ PSNR $= 21.85dB$ (c) Erreur de reconstruction moyenne sans $\{Q_E\}_{11}$ sur toutes les images de Kodak (d) Image 2a - 2b (e) Recons. avec $\{Q_E\}_{11}$ seule



(f) Recons. avec $\{Q_E\}_i \, \forall i$ PSNR $= 29.42dB$ (g) Recons. sans $\{Q_E\}_9$ PSNR $= 24.32dB$ (h) Erreur de reconstruction moyenne sans $\{Q_E\}_9$ sur toutes les images de Kodak (i) Recons. avec $\{Q_E\}_9$ et $\{Q_E\}_{11}$ (j) Recons. avec $\{Q_E\}_1$ et $\{Q_E\}_{11}$

FIGURE 2 : Reconstruction d'une image de Kodak avec tous les Q_E ou avec certains des éléments de Q_E manquant. Les tokens de Q_E se spécialisent spatialement mais sont insuffisants à eux seuls pour décoder une zone.

5 Conclusion

Nous avons proposé une nouvelle méthode d'apprentissage profond pour transformer une image en une petite séquence de tokens grâce au processus d'agrégation au sein de nos *image queries* via l'attention croisée. Cette méthode se passe complètement de convolutions et n'introduit donc pas les biais spatiaux inhérents à ce type d'opération. Elle permet de transformer une grande séquence de patchs en une séquence de taille réduite. Le passage à des résolutions supérieures et l'intégration dans une chaîne de compression complète, comprenant notamment quantification et codage entropique, doit être étudié. Enfin, l'utilisation de la séquence réduite produite par l'encodeur pour d'autres tâches de vision doit être explorée car elle apporterait une réduction non négligeable du coût des calculs d'attention de par sa longueur inférieure à la séquence obtenue à partir d'une image découpée en patchs.

6 Remerciements

Ce travail est mené dans le cadre d'une thèse co-financée par l'Agence Innovation Défense (AID). Il bénéficie d'une allocation de l'IDRIS sur le calculateur Jean-Zay (AD011011290).

Références

[1] Eirikur AGUSTSSON, Michael TSCHANNEN, Fabian MENTZER, Radu TIMOFTE et Luc Van GOOL : Generative adversarial networks for extreme learned image compression. *In ICCV*, 2019.

[2] Fabrice BALLARD : Bpg image format. `https://bellard.org/bpg/`.

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

[4] Johannes BALLÉ, David MINNEN, Saurabh SINGH, Sung Jin HWANG et Nick JOHNSTON : Variational image compression with a scale hyperprior. *In ICLR*, 2018.

[5] Tong CHEN, Haojie LIU, Zhan MA, Qiu SHEN, Xun CAO et Yao WANG : End-to-end learnt image compression via non-local attention optimization and improved context modeling. *IEEE Transactions on Image Processing*, 2021.

[6] Zhengxue CHENG, Heming SUN, Masaru TAKEUCHI et Jiro KATTO : Learned image compression with discretized gaussian mixture likelihoods and attention modules. *In CVPR*, 2020.

[7] Jacob DEVLIN, Ming-Wei CHANG, Kenton LEE et Kristina TOUTANOVA : Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, 2018.

[8] Alexey DOSOVITSKIY, Lucas BEYER, Alexander KOLESNIKOV, Dirk WEISSENBORN *et al.* : An image is worth 16x16 words : Transformers for image recognition at scale. *ICLR*, 2021.

[9] GOOGLE LLC : An image format for the web |webp. `https://developers.google.com/speed/webp/`.

[10] Binglin LI, Jie LIANG et Jingning HAN : Variable-rate deep image compression with vision transformers. *IEEE Access*, 2022.

[11] Ming LU, Peiyao GUO, Huiqing SHI, Chuntong CAO et Zhan MA : Transformer-based image compression. *In DCC*, 2022.

[12] 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*, 2012.

[13] THE EASTMAN KODAK COMPANY : The kodak lossless true color image suite. `https://r0k.us/graphics/kodak/`.

[14] George TODERICI, Damien VINCENT, Nick JOHNSTON, Sung JIN HWANG, David MINNEN, Joel SHOR et Michele COVELL : Full resolution image compression with recurrent neural networks. *In CVPR*, 2017.

[15] Ashish VASWANI, Noam SHAZEER, Niki PARMAR, Jakob USZKOREIT *et al.* : Attention is all you need. *In NIPS*, 2017.

[16] Z. WANG, E.P. SIMONCELLI et A.C. BOVIK : Multiscale structural similarity for image quality assessment. *In The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, 2003.

[17] Richard ZHANG, Phillip ISOLA, Alexei A. EFROS, Eli SHECHTMAN et Oliver WANG : The unreasonable effectiveness of deep features as a perceptual metric. *In CVPR*, June 2018.