DOCTORAT BRETAGNE LOIRE / MATHSTIC

INSA RENNES

THÈSE DE DOCTORAT DE

L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Traitement du signal*

Par

Théo Ladune

Design of Learned Video Coding Schemes

Thèse présentée et soutenue à Rennes, le 18 octobre 2021
Unité de recherche : VAADER, IETR/INSA

Rapporteurs avant soutenance :

Jenny Benois-Pineau	Professeur, Université de Bordeaux, France
Marc Antonini	Directeur de Recherche CNRS, Université de Nice-Sophia Antipolis, France

Composition du Jury :

Jenny Benois-Pineau	Professeur, Université de Bordeaux, France
Marc Antonini	Directeur de Recherche CNRS, Université de Nice-Sophia Antipolis, France
Frédéric Dufaux	Directeur de recherche CNRS, Centrale Supelec, France
Olivier Déforges	Professeur, IETR/INSA, France (Directeur de thèse)
Pierrick Philippe	Ingénieur Recherche, Orange, France (Co-directeur de thèse)
Wassim Hamidouche	Maître de Conférence, IETR/INSA, France (Encadrant de thèse)

Invitée :

Lu Zhang	Maître de conférence, IETR/INSA, France

ACKNOWLEDGEMENTS

First and foremost, I would particularly like to thank Pierrick Philippe, my PhD supervisor from Orange. I am extremely grateful for the considerable amount of time he has dedicated to me, during which he has provided me with so many advices, recommandations and ideas. Our insightful discussions have been extremely valuable to me and are the source of some of the findings of this thesis.

I am grateful to Jenny Benois-Pineau and Marc Antonini for having accepted to review this manuscript. I also thank the other jury members for accepting to be part of the jury.

I would also like to thank all my colleagues from Orange, who played in key role in the accomplishment of this thesis. In particular, I am indebted to Patrick Boissonade and Rémi Rouaud for the design and the maintenance of the GPU cluster. Without them, it would not have been possible to carry out all the work presented in this manuscript. A special thanks goes to Corentin Kervadec, which has made my stay at Orange even more enjoyable. I have greatly benefited from the valuable exchanges I have had with Félix Henri, Gordon Clare, Matthieu Gendrin and Stéphane Pateux about video coding and deep learning. Finally, I would like to express my gratitude to Didier Tattevin for being always available and for the particular attention he has showed to me, particularly during the Covid-19 outbreak.

I want to thank my PhD supervisors from the VAADER team, Wassim Hamidouche, Lu Zhang and Olivier Déforges for their assistance, especially during the writing of articles and of this manuscript. Our monthly meetings have always been the opportunity to receive relevant comments and interesting ideas.

My final thanks go to Lisette, to my friends and to my parents.

RÉSUMÉ EN FRANÇAIS

Contexte

LA quantité toujours croissante de données transmises sur Internet représente une contribution significative au changement climatique. Avant même la pandémie de la Covid-19, la vidéo représentait environ 80 % du trafic internet [1]. De plus, la consommation de vidéos a encore progressé lors de la pandémie, obligeant des plateformes comme Netflix à réduire la qualité des vidéos afin d'éviter un engorgement du réseau [2]. Selon certaines estimations, la diffusion de vidéos sur Internet est d'ores et déjà responsable de 1 % des émissions mondiales de gaz à effet de serre [3], soit l'équivalent d'un pays comme l'Espagne. Réduire la quantité de données nécessaire à la transmission ou au stockage des vidéos pourrait permettre de réduire les émissions de carbone du streaming vidéo, dans la mesure où un effet rebond (paradoxe de Jevons) est évité [4].

Depuis les années 90, les algorithmes de compression vidéo visent à réduire la taille d'une vidéo tout en garantissant une qualité acceptable pour les utilisateurs. Bien qu'initialement motivés pour répondre aux limitations de débit imposées par les infrastructures existantes, réduire l'impact environnemental du streaming video est une motivation supplémentaire pour concevoir de nouveaux algorithmes de compression. Différents standards de compression vidéo ont été développés par des organismes de standardisation tels que le *Moving Picture Experts Group* (MPEG) et l'*International Telecommunication Union* (ITU-T). Chaque génération de standards (AVC en 2003, HEVC en 2013 et VVC en 2020) présente des améliorations incrémentales qui offrent des gains en compression significatifs, permettant de diviser par deux le débit d'une vidéo à qualité égale.

Ces algorithmes de compression conventionnels traitent les vidéos au travers d'opérations successives, généralement linéaires. De plus, les codeurs conventionnels sont conçus de manière incrémentale, occasionnant une optimisation séparée de leurs différentes opérations. Récemment, les réseaux de neurones sont devenus une réponse populaire à de nombreuses problématiques, grâce à leur capacité à modéliser toute fonction au travers d'un processus d'optimisation. Dans le contexte de la compression, cela permet de réaliser des opérations plus riches et non linéaires. Ces différentes opérations peuvent être

conjointement optimisées afin de mieux minimiser la taille de la vidéo et les dégradations visuelles entraînées par sa compression.

Des travaux récents montrent déjà que les schémas de compression d'image basés sur des réseaux de neurones sont compétitifs avec les meilleurs codeurs conventionnels. Dans ces approches, les réseaux de neurones apprennent les fonctions transformant une image vers une représentation compressée et vice-versa. Cependant, la présence d'une dimension temporelle fait de la compression vidéo une tâche encore difficile pour les approches neuronales. Néanmoins, la réduction de la taille des vidéos est la problématique à laquelle il faut répondre. Dans ce but, cette thèse étudie la conception de codeurs vidéo neuronaux, afin de profiter de leurs capacités prometteuses. Étant donné la nouveauté de ce sujet, le codeur proposé est conçu à partir d'une page blanche et chacun de ces éléments est considéré et motivé.

Partie I – Contexte et état de l'art

La première partie de la thèse présente les notions importantes relatives à la compression d'image et de vidéo. Ces notions sont illustrées à travers leur implémentation par les systèmes de compression conventionnels ainsi que les codeurs d'images neuronaux.

Chapitre 1 : Fondamentaux de compression vidéo

Les algorithmes de compression vidéo visent à réduire le nombre de bits nécessaires à l'envoi de séquences d'images. En effet, les valeurs brutes des pixels d'une séquence vidéo représente une quantité prohibitive de données. Par exemple, une vidéo haute définition est constituée d'une succession de 24 à 60 images par seconde typiquement. Chaque image est composée de 1920×1080 pixels, où la couleur de chaque pixel est codée par un triplet de valeurs 8 bits. Ainsi, le débit nécessaire à la transmission des pixels bruts est de :

$$\frac{24 \text{ images}}{1 \text{ s}} \times \frac{1920 \times 1080 \text{ pixels}}{1 \text{ image}} \times \frac{3 \times 8 \text{ bits}}{1 \text{ pixel}} \simeq 1.20 \text{ Gbit/s}. \tag{1}$$

Une vidéo n'est pas une collection de pixels aux valeurs aléatoires, mais présente au contraire une forte organisation spatio-temporelle. Différentes techniques de compression *sans perte* exploitent cette structure afin de réduire la taille des vidéos, sans pour autant perdre d'information. Ces techniques sont implémentées par un encodeur, qui encode la vidéo dans un flux binaire, et par un décodeur, qui décode du flux binaire la vidéo initiale.

Tout d'abord, les algorithmes de codage entropique exploitent les propriétés statistiques d'une vidéo en assignant moins de bits aux combinaisons de pixels les plus probables. Supposons que $\mathbf{x} = (x_1, \ldots, x_N)$ soit une variable aléatoire discrète suivant une distribution q, représentant un message à transmettre. Par exemple, $\mathbf{x}$ représente les N pixels composant une seconde de vidéo. D'après le théorème de Shannon [5], le débit optimal $R^\star$ est exprimé au travers de l'entropie H, définie comme :

$$R^\star = H(\mathbf{x}) = \mathbb{E}_{\mathbf{x}\sim q}\left[-\log_2 q(\mathbf{x})\right]. \tag{2}$$

Dans l'équation ci-dessus, chaque message (chaque seconde de vidéo) est représenté par un code binaire de longueur $-\log_2 q(\mathbf{x})$. En pratique, la distribution q est souvent inconnue, ce qui nécessite d'estimer q au travers d'une approximation p. L'erreur causée par l'utilisation d'une approximation entraîne un débit R supérieur à $R^\star$:

$$R - R^\star = \overbrace{\mathbb{E}_{\mathbf{x}\sim q}\left[-\log_2 p(\mathbf{x})\right]}^{\text{Longueur de code estimée}} - \overbrace{\mathbb{E}_{\mathbf{x}\sim q}\left[-\log_2 q(\mathbf{x})\right]}^{\text{Longueur de code optimale}}. \tag{3}$$

Ainsi, la performance du codage entropique dépend de la qualité de l'approximation p. Néanmoins, la dimension d'une séquence vidéo (plusieurs millions de pixels par seconde) empêche d'estimer une distribution multidimensionnelle des pixels. En conséquence, les différents pixels sont supposés indépendants et transmis successivement :

$$p(\mathbf{x}) = \prod_{i=1}^{N} p_i(x_i). \tag{4}$$

Bien que nécessaire afin d'implémenter un algorithme de codage entropique, cette hypothèse n'est jamais vérifiée en pratique. Cela entraîne un surcoût de débit, qui augmente avec la dépendence statistique entre les pixels de la vidéo. Pour réduire ce surcoût de débit, deux types d'opérations sont appliqués sur les vidéos.

- Les prédictions visent à réduire les redondances entre les pixels déjà transmis $\mathbf{x}_{<i}$ (mémorisés au décodeur) et ceux à transmettre. Les pixels à transmettre sont prédits (à l'encodeur et au décodeur) grâce à $\mathbf{x}_{<i}$, et seule la partie non-prédite est envoyée ;
- Les transformées sont appliquées sur tous les pixels afin de réduire les dépendences statistiques.

Les standards de compression vidéo emploient prédictions et transformées en amont du codage entropique, permettant de réduire le débit d'un facteur trois [6]. D'après l'équation

(1), le débit est donc abaissé à quelques centaines de Mbit/s, ce qui reste supérieur au débit de quelques Mbit/s utilisés en pratique pour le la transmission de vidéos. Pour atteindre un tel débit la reproduction exacte de la vidéo originale n'est pas possible.

La compression *avec pertes* permet de considérablement abaisser le débit au prix de la perte d'information. Typiquement, cette perte d'information est réalisée au travers d'une étape de quantification, réduisant la précision des valeurs transmises. Elle entraîne une dégradation potentiellement perceptible (distorsion), mesurée par l'intermédiaire de différentes métriques (erreur quadratique, MS-SSIM [7]). Le compromis entre débit R et distorsion D doit être considéré avec attention en fonction du cas d'usage. Il est souvent formulé comme l'optimisation d'une fonction de coût débit distorsion J exprimée à l'aide un Lagrangien, dont le multiplicateur λ établit le compromis entre débit et distorsion :

$$\min\ J_\lambda = D + \lambda R. \tag{5}$$

La Figure 1 montre l'utilisation de techniques de compression sans perte (prédiction P, transformée T et codage entropique AE et AD), couplée avec une quantification Q. Ces principes constituent la base des algorithmes de compression. Ils sont repris aussi bien par les standards de compression conventionnels, présentés dans le Chapitre 2, que par les schémas de compression basés apprentissage développés dans le reste de ce manuscrit.

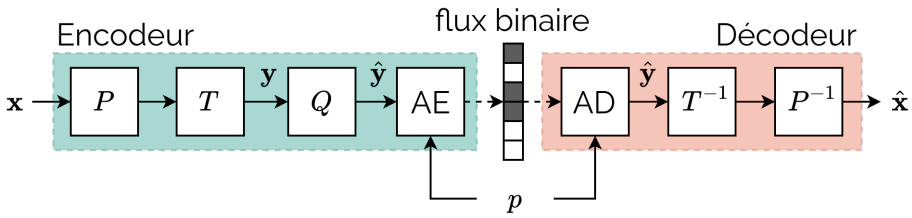


FIGURE 1 : Compression avec pertes d'un message $\mathbf{x}$ avec un modèle de probabilité p.

Chapitre 2 : Algorithmes de compression vidéo conventionnels

Depuis les années 90, le *Moving Picture Experts Group* (MPEG) et l'*International Telecommunication Union* (ITU-T) ont proposé plusieurs standards de compression vidéo. *Advanced Video Coding* (AVC) [8] a été finalisé en 2003, suivi par *High Efficiency Video Coding* (HEVC) [9] en 2013 et récemment *Versatile Video Coding* (VVC) [10] en 2020. Les standards successifs de MPEG/ITU ont continûment amélioré un même paradigme, désigné dans ce manuscrit comme la compression vidéo *conventionnelle*.

Commençons par décrire la façon dont les systèmes de compression conventionnels traitent une image seule, appelée une image intra (I). L'image est tout d'abord partitionnée en différents blocs. Chacun des blocs est ensuite prédit d'après les blocs de l'image déjà transmis. Puis, le résidu (erreur de prédiction) est transmis par un processus de codage composé d'une transformée, d'une quantification et d'un codage entropique. Les étapes de prédiction et de transformée présentent deux caractéristiques importantes :

- Elles sont linéaires, offrant ainsi une complexité limitée ;
- Différentes prédictions et transformées sont disponibles. Les plus adaptées pour le bloc sont choisies par l'encodeur afin d'optimiser le coût débit distorsion.

Pour des raisons pratiques, les différentes étapes sont optimisées séparément, ce qui peut entraîner des choix sous-optimaux par rapport à une optimisation globale du système.

La dimension temporelle d'une vidéo présente de nombreuses redondances, qui doivent être réduites afin d'abaisser le débit total. Pour ce faire, une prédiction temporelle, implémentée par un algorithme de compensation de mouvement, est mise en place. Un mouvement translationnel est appliqué à un bloc de référence issu d'une *autre* image, préalablement transmise. Ce mouvement indique les déplacements entre le bloc de référence et le bloc à coder. Comme il doit être transmis au décodeur, sa valeur et sa précision sont déterminées pour optimiser le coût débit distorsion. Une prédiction *bi-directionnelle* est souvent implémentée afin d'améliorer la precision de la prédiction. Elle repose sur une moyenne de deux compensations de mouvement intermédiaires. Enfin, le résidu issu de la prédiction temporelle est transmis de manière similaire aux images intra.

Les blocs de référence sont issus d'un ensemble restreint d'images de références, défini par la structure de codage. La Figure 2 présente un exemple de structure de codage appelée *Random Access*. Une image s'appuyant sur d'autres images déjà reçues, est appelée une image inter. On différencie ensuite les images P (utilisant une seule image de référence), des images B (utilisant deux images de référence). La structure de codage doit répondre à un certain nombres de contraintes pratiques. Par exemple, l'utilisation d'images de références futures introduit une latence dans le processus de décodage qui peut être prohibitive pour certaines applications (par exemple en visioconférence).

Les systèmes de compression vidéo conventionnels se doivent de répondre à un large éventail de contraintes telles que la complexité ou la latence. En conséquence, les principes exposés dans le Chapitre 1 sont implémentés avec des limitations : opérations linéaires,

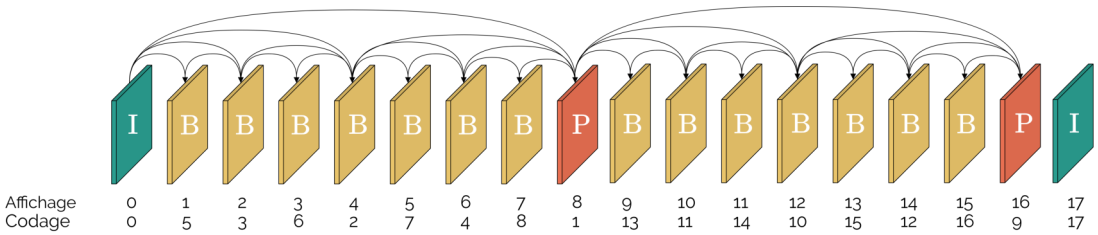


FIGURE 2 : Configuration de codage *Random Access*.

nombre de références limité et optimisation séparée des différentes étapes. Néanmoins, la présence à chaque étape de différents modes de réalisation permet de réaliser des opérations adaptées aux signaux à compresser. Le Chapitre 3 montre comment les réseaux de neurones peuvent être utilisés pour concevoir un système de compression d'image fixe. En particulier, la capacité des réseaux de neurones à représenter des opérations non-linéaires et à être optimisés globalement est exploitée pour obtenir de meilleures performances de compression.

Chapitre 3 : Apprendre à compresser des images

Les réseaux de neurones sont un outil mathématique permettant de représenter n'importe quelle fonction [11]. Ils sont composés d'une succession d'opérations linéaires et non linéaires, dont les paramètres sont appris au cours d'un processus d'optimisation appelé entraînement. Notons $\mathbf{y} = f(\mathbf{x};\boldsymbol{\theta})$ la fonction réalisée par un réseau de neurones qui relie une entrée $\mathbf{x}$ à une sortie $\mathbf{y}$ à l'aide de paramètres $\boldsymbol{\theta}$. Le but du réseau est que $\mathbf{y}$ corresponde à la sortie désirée $\mathbf{y}_{obj}$. Pour ce faire, il est nécessaire de trouver les paramètres optimaux :

$$\boldsymbol{\theta}^{\star} = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}} \left[J\left(f(\mathbf{x};\boldsymbol{\theta}),\, \mathbf{y}_{obj} \right) \right], \tag{6}$$

J est la fonction de coût, qui mesure l'écart entre la sortie attendue et celle observée. Les paramètres $\boldsymbol{\theta}$ sont appris au travers d'un algorithme itératif de descente de gradient stochastique. À l'itération numéro n, plusieurs exemples $\mathbf{x}$ sont tirés aléatoirement depuis un ensemble d'entraînement. La sortie du réseau ainsi que la fonction de coût sont calculées pour chacune des entrées, puis la fonction de coût est moyennée. Enfin, les paramètres $\boldsymbol{\theta}_n$ du réseaux sont mis à jours de la manière suivante :

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \eta \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}} \left[J\left(f(\mathbf{x};\boldsymbol{\theta}),\, \mathbf{y}_{obj} \right) \right],\ \eta \in \mathbb{R}. \tag{7}$$

Depuis 2017 et les travaux de Ballé [12] et Theis [13], les autoencodeurs sont la méthode la plus populaires pour implémenter des systèmes de compression basés réseaux de neurones. La Figure 3 présente l'architecture d'un autoencodeur, pourvue de 2 sous-réseaux réalisant respectivement la transformée d'analyse g_a et de synthèse g_s. Habituellement, les autoencodeurs sont entraînés pour que leur sortie $\hat{\mathbf{x}}$ soit la plus proche possible de l'entrée $\mathbf{x}$, tout en imposant des contraintes sur la représentation latente $\mathbf{y}$. Cela permet d'apprendre une représentation latente pertinente et expressive, offrant un domaine de représentation mieux adapté que celui du domaine vidéo initial. Dans le contexte de la compression, la contrainte imposée sur $\mathbf{y}$ a pour vocation de limiter son entropie, c'est-a-dire son débit.

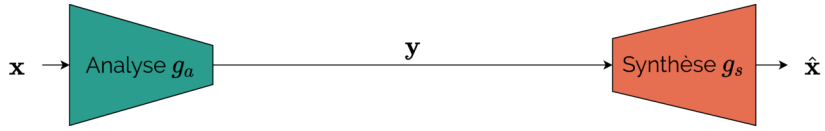


FIGURE 3 : Architecture d'un autoencodeur.

La Figure 4 présente un schéma de compression d'image basé sur un autoencodeur. La variable latente $\mathbf{y}$ obtenue grâce à la transformée d'analyse g_a est quantifiée, puis transmise au décodeur via un codage entropique. Enfin, la transformée de synthèse g_s permet d'obtenir l'image décodée $\hat{\mathbf{x}}$ depuis les latentes quantifiées. Afin de mener à bien l'étape de codage entropique, un modèle de probabilité p des variables latentes doit être disponible. C'est le rôle d'un troisième réseau de neurones, entraîné pour représenter la distribution de $\hat{\mathbf{y}}$. Tous les paramètres du schéma de compression (g_a, g_s et p) sont optimisés conjointement afin de minimiser une fonction de coût débit distorsion :

$$\mathcal{L}_\lambda = \mathbb{E}_{\mathbf{x}} \left[d(\mathbf{x}, \hat{\mathbf{x}}) - \lambda r(\hat{\mathbf{y}}) \right], \tag{8}$$

où d mesure la distorsion, r mesure le débit associé aux variables latentes, et λ est la contrainte de débit permettant d'équilibrer débit et distorsion.

Ainsi, un schéma de compression d'image basé apprentissage est composé de trois éléments principaux : une transformée d'analyse, de synthèse et un modèle de probabilité des variables latentes. Une grande partie des travaux menés depuis 2017 vise à améliorer ces trois éléments, à travers des architectures mieux adaptées. Dans le cadre de cette thèse, un nouveau modèle de probabilité a été proposé. Il permet de mieux représenter la distribution des variables latentes et donc d'abaisser leur débit de 6 % [14].

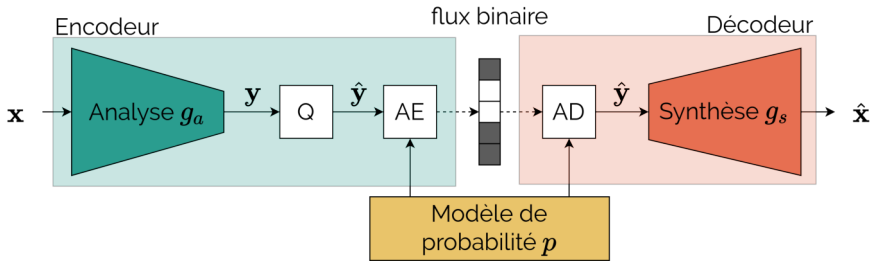


FIGURE 4 : Schéma de compression d'image basé sur un autoencodeur. Q dénote la quantification, AE et AD le codage arithmétique avec p le modèle de probabilité.

La Figure 5 présente les performances d'un codeur d'image basé autoencodeur (AE), avec le modèle de probabilité proposé. Pour mieux apprécier les résultats, les résultats de différents standards de compression (JPEG, HEVC et VVC), sont également présentés. Ces résultats démontrent que la compression d'image basée apprentissage est compétitive avec VVC, le dernier standard de compression. Cela est d'autant plus encourageant au vu de la jeunesse des approches basées apprentissages, qui ont atteint ce niveau de performances en l'espace de quelques années et dont les résultats ne cessent de progresser.

Tout au long de ce chapitre, les autoencodeurs se sont révélés être des architectures génériques, performantes pour transmettre différent types de signaux. Ainsi, ces architectures sont les briques de base des différents systèmes de compression vidéo présentés dans les chapitres suivants.

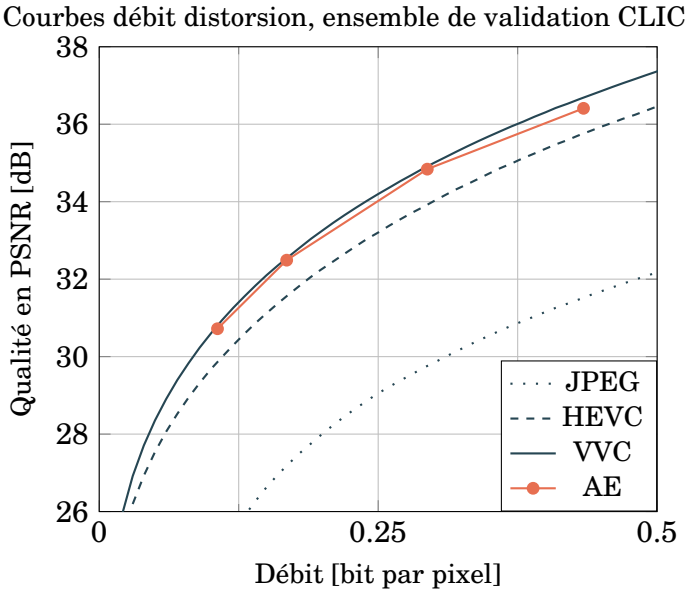


FIGURE 5 : Performance d'un système de compression d'image basé autoencodeur (AE).

Partie 2 – Compression vidéo basée apprentissage

La seconde partie de cette thèse propose de concevoir un codeur vidéo neuronal. Tout d'abord, le cahier des charges du codeur ainsi qu'un cadre expérimental réaliste sont définis. Ensuite, les différentes composantes du codeur sont étudiées.

Chapitre 4 : De la compression d'image à la compression vidéo

En 2019, des travaux précurseurs [15] ont proposé d'étendre directement les autoencodeurs de compression d'image pour traiter de façon groupée T images de vidéo . Cependant, ces approches requièrent un nombre élevé de paramètres et présentent des défauts inhérents au traitement par groupe d'images. En effet, il est nécessaire d'attendre d'avoir T images à traiter pour commencer le codage, entraînant une latence de T images. Puisque ces défauts ne sont pas compensés par des performances encourageantes, la grande majorité des travaux ultérieurs (y compris ceux présentés dans ce manuscrit) ont adopté le paradigme conventionnel d'un traitement image par image. Cependant, cela impose une étape de prédiction temporelle pour réduire les redondances. En pratique, elle est effectuée par compensation de mouvement, de manière similaire aux codeurs conventionnels.

L'ajout d'une étape de compensation de mouvement amène à un schéma de compression en trois étapes, présenté dans la Fig. 6. Supposons que $\mathbf{x}_t$ soit une image à coder et que deux images de références $\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_f$ soient disponibles. Tout d'abord, il est nécessaire d'estimer le mouvement entre $\mathbf{x}_t$ et les références puis de le transmettre au décodeur. Ensuite, la compensation de mouvement calcule la prédiction temporelle $\tilde{\mathbf{x}}_t$. Finalement, seule la partie non-prédictible est transmise pour obtenir l'image reconstruite $\hat{\mathbf{x}}_t$.

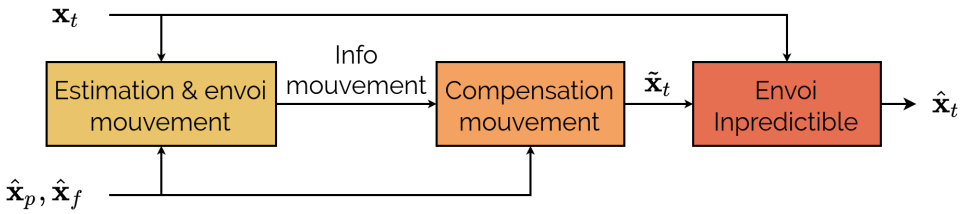


FIGURE 6 : Compression d'une image inter.

Les autoencodeurs sont utilisés ici comme des outils génériques permettant de transmettre les différentes quantités en jeu (informations de mouvement, partie non-prédictible). La meilleure façon de transmettre $\mathbf{x}_t$ étant donné sa prédiction $\tilde{\mathbf{x}}_t$ est étudiée dans le

Chapitre 5. Puis, le Chapitre 6 se concentre sur l'estimation et la transmission des informations de mouvement, afin d'obtenir une prédiction temporelle précise. Afin d'évaluer les différentes solutions proposées, une situation réaliste de codage vidéo est choisie : la tâche de compression vidéo du *Challenge on Learned Image Compression* (CLIC) 2021 [16]. L'objectif de ce challenge est de compresser 100 vidéos haute définition à un débit moyen de 1 Mbit/s tout en obtenant la meilleure qualité, mesurée au travers d'une métrique objective nommée MS-SSIM.

Pour le moment, les meilleures implémentations des standards de compression conventionnels (HM pour HEVC et VTM pour VVC) demeurent les solutions de compression vidéo les plus performantes. Ainsi, elles sont utilisées comme ancrage pour attester de la pertinence des résultats proposés. Puisque le challenge ne comporte pas de contrainte sur la structure de codage, une configuration *Random Access* (voir Fig. 2) issue des conditions de test de VVC est utilisée [17].

La configuration *Random Access* nécessite de savoir traiter des images avec zéro, une ou deux références, c'est-à-dire des images I, P et B. De ce fait, nous proposons d'entraîner nos systèmes afin de coder des triplets d'images, comme illustré par la Fig. 7. Enfin, la totalité des paramètres du système est optimisée de manière conjointe au travers de l'optimisation de la fonction de coût suivante, composée de la somme du coût débit distorsion des images successives :

$$\mathcal{L}_\lambda = \mathbb{E}_{\mathbf{x}} \left[\sum_t d(\mathbf{x}_t, \hat{\mathbf{x}}_t) + \lambda r(\hat{\mathbf{x}}_t) \right]. \tag{9}$$

Pour répondre au challenge, la métrique de distorsion d est logiquement basée sur le MS-SSIM.

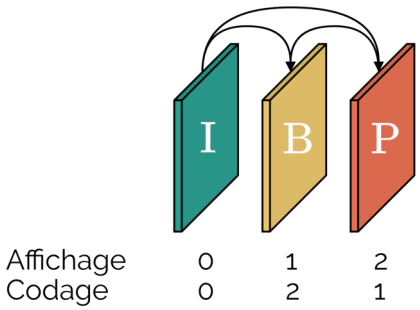


FIGURE 7 : Configuration de codage pour l'entraînement.

Chapitre 5 : Exploitation d'une prédiction

La majorité des systèmes de compression vidéo, aussi bien conventionnels que basés apprentissage, utilise le codage résiduel afin diminuer les redondances entre les images successives. Lorsqu'une prédiction est disponible, elle est soustraite à l'image à coder afin d'obtenir le résidu qui est ensuite transmis. Néanmoins, les réseaux de neurones sont en mesure de mieux exploiter une prédiction qu'au travers d'une simple soustraction. C'est ce qui est démontré dans ce chapitre.

Une prédiction reproductible, sans information de mouvement, est utilisée pour permettre de comparer différentes méthodes d'exploitation de la prédiction. Ainsi, étant donné deux images de références $\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_f$, la prédiction temporelle $\tilde{\mathbf{x}}_t$ utilisée est :

$$\tilde{\mathbf{x}}_t = \beta\hat{\mathbf{x}}_p + (1-\beta)\hat{\mathbf{x}}_f = \begin{cases} \frac{1}{2}\left(\hat{\mathbf{x}}_p + \hat{\mathbf{x}}_f\right) & \text{pour les images B, } \left(\beta = \frac{1}{2}\right) \\ \hat{\mathbf{x}}_p & \text{pour les images P. } \left(\beta = 1\right) \end{cases} \tag{10}$$

La tâche de codage d'une image inter est tout d'abord formalisée . Des informations depuis une quantité présente à l'encodeur $\mathbf{x}_{enc}$, sont envoyées afin d'obtenir $\mathbf{x}_{out}$ au décodeur. Dans le même temps, une quantité $\mathbf{x}_{dec}$, déjà présente au décodeur (par exemple une prédiction) contient des informations pertinentes. Un schéma de codage inter idéal évite de transmettre les informations de $\mathbf{x}_{enc}$ déjà présentes dans $\mathbf{x}_{dec}$. Dans le cas du codage résiduel, la suppression des informations commune entre $\mathbf{x}_{enc}$ et $\mathbf{x}_{dec}$ se fait au travers d'une soustraction, dont le résultat est ensuite fourni à un autoencodeur :

$$\mathbf{x}_{out} = \mathbf{x}_{dec} + \overbrace{g_s(Q(g_a(\underbrace{\mathbf{x}_{enc} - \mathbf{x}_{dec}}_{\text{résidu}})))}^{\text{autoencodeur}}. \tag{11}$$

Une fois transmis, le résidu est combiné avec $\mathbf{x}_{dec}$ au travers d'une addition. Effectuer le mélange via une addition et une soustraction présente deux défauts :

- Ce n'est pas nécessairement la combinaison la mieux adaptée ;
- Cela nécessite des quantités compatibles : image et différence d'images.

Pour remédier à ces défauts, nous proposons une nouvelle architecture appelée *codage conditionnel* [18], présentée Fig. 8. Une transformée d'analyse g_a prend en entrée les informations présentes à l'encodeur et au décodeur et calcule une variable latente $\mathbf{y}$

représentant l'information à transmettre, présente à l'encodeur mais pas au décodeur :

$$\mathbf{y} = g_a(\mathbf{x}_{enc}, \mathbf{x}_{dec}). \tag{12}$$

Comme g_a est un réseau de neurones, ses opérations successives permettent de réaliser un mélange plus riche qu'une simple soustraction.

En codage résiduel, l'information présente au décodeur $\mathbf{x}_{dec}$ est combinée avec la sortie de la transformée de synthèse g_s via une addition. Le codage conditionnel utilise une nouvelle transformée de *conditionnement* g_c qui sert à incorporer les informations de $\mathbf{x}_{dec}$ à la sortie $\mathbf{x}_{out}$. Pour ce faire, g_c extrait une nouvelle variable latente $\mathbf{y}_c$:

$$\mathbf{y}_c = g_c(\mathbf{x}_{dec}). \tag{13}$$

Il est important de noter que la variable latente $\mathbf{y}_c$ issue de la transformée de conditionnement est calculée au décodeur et ne nécessite donc pas de transmission. Enfin, la transformée de synthèse g_s prend en entrée les deux variables latentes pour calculer la sortie désirée :

$$\mathbf{x}_{out} = g_s(\mathbf{y}, \mathbf{y}_c). \tag{14}$$

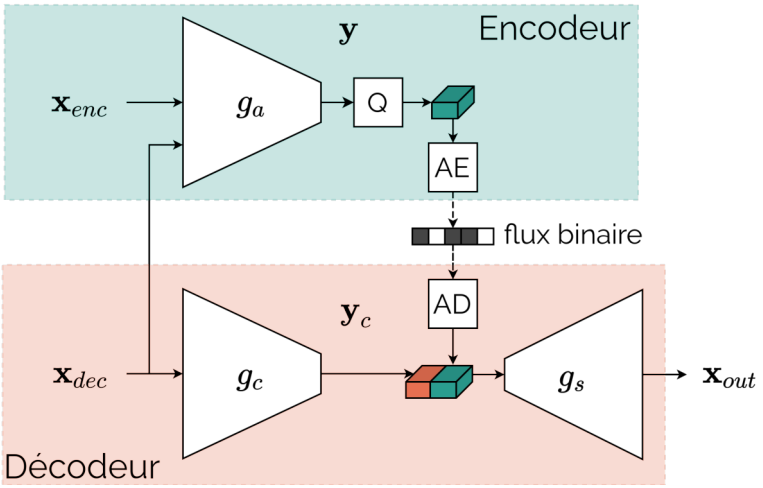


FIGURE 8 : L'architecture du codage conditionnel.

L'architecture des transformées d'analyse g_a et de synthèse g_s demeurent (quasi) identique à celle utilisée en compression d'image. La nouvelle transformée de conditionnement g_c utilise la même architecture que g_a. Enfin, dans le contexte de l'exploitation d'une

prédiction, les quantités $\mathbf{x}_{enc}$, $\mathbf{x}_{dec}$ et $\mathbf{x}_{out}$ se traduisent en :

$$\begin{cases} \mathbf{x}_{enc} = \mathbf{x}_t & \text{image à coder,} \\ \mathbf{x}_{dec} = \tilde{\mathbf{x}}_t & \text{prediction temporelle,} \\ \mathbf{x}_{out} = \hat{\mathbf{x}}_t & \text{image décodée.} \end{cases} \tag{15}$$

Les performances du codage conditionnel sont évaluées sur la tâche de compression décrite dans le Chapitre 4. Afin de servir d'ancrage, un autoencodeur habituel est entraîné à coder des résidus. Pour mieux apprécier ces résultats, les performances de HEVC, *avec* compensation de mouvement, sont fournies au travers de deux implémentations : x265 (preset medium) et le HM. Si le HM est la meilleure implémentation disponible, x265 demeure très utilisé dans la littérature [19]-[21].

Les résultats des différents codeurs sont présentés Fig. 9. Le codage conditionnel diminue le débit de 29 % par rapport au codage résiduel, démontrant son intérêt.

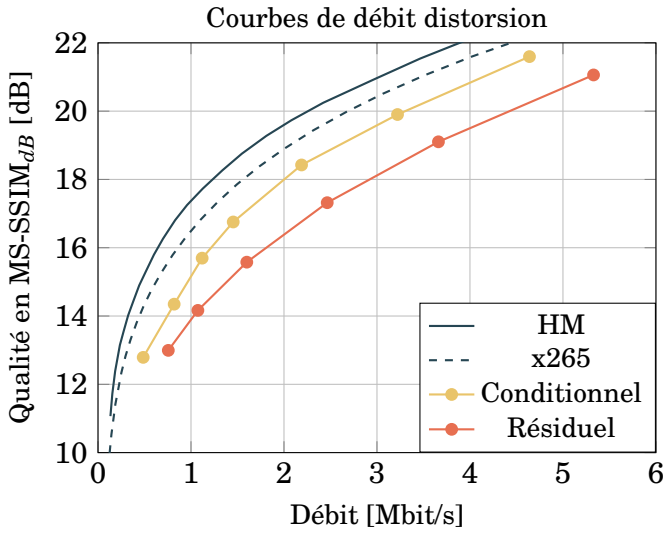


FIGURE 9 : Performances du codage conditionnel.

Les performances du système sont encore améliorées en ajoutant une précision de quantification adaptée au type d'image (I, P ou B). En effet, laisser l'algorithme d'apprentissage fixer la précision de quantification permet de diminuer le débit de 10 %.

Ainsi, ce chapitre a permis d'augmenter significativement les performances du codage résiduel habituellement employé. Cependant, la prédiction temporelle sans information de mouvement n'est pas suffisamment précise et pénalise les performances. Le chapitre 6 se concentre donc sur la réalisation d'une meilleure prédiction.

Chapitre 6 : Calcul d'une prédiction temporelle

Inspiré par les codeurs vidéos conventionnels, la prédiction temporelle est obtenue grâce à une compensation de mouvement bi-directionnelle. Pour calculer cette prédiction, chacune des deux images de références ($\hat{\mathbf{x}}_p$ et $\hat{\mathbf{x}}_f$) dispose d'un flux optique ($\mathbf{v}_p$ et $\mathbf{v}_f$). Un flux optique est une carte de mouvement en deux dimensions, indiquant pour chaque pixel le déplacement horizontal et vertical entre l'image à prédire $\mathbf{x}_t$ et sa référence. Ensuite, une opération de *warping* w permet d'appliquer les flux optiques à leur référence respective. Enfin, les références compensées sont assemblées au travers d'une somme pondérée par $\boldsymbol{\beta}$ pour obtenir la prédiction finale $\tilde{\mathbf{x}}_{tt}$.

$$\tilde{\mathbf{x}}_t = \overbrace{\boldsymbol{\beta} \odot w(\hat{\mathbf{x}}_p; \mathbf{v}_p)}^{\text{Compensation de } \hat{\mathbf{x}}_p} + \overbrace{(1-\boldsymbol{\beta}) \odot w(\hat{\mathbf{x}}_f; \mathbf{v}_f)}^{\text{Compensation de } \hat{\mathbf{x}}_f}, \quad \odot \text{ est un produit pixel par pixel.} \tag{16}$$

Dans ce chapitre, nous allons concevoir *MNet*, un réseau de neurones pour estimer et transmettre les informations nécessaires à la prédiction ($\mathbf{v}_p$, $\mathbf{v}_f$ et $\boldsymbol{\beta}$). Ce réseau vient en amont du processus de compensation, lui-même situé avant le réseau *CNet*, chargé du codage conditionnel de $\mathbf{x}_t$ connaissant la prédiction $\tilde{\mathbf{x}}_t$. La Figure 10 présente ce processus.

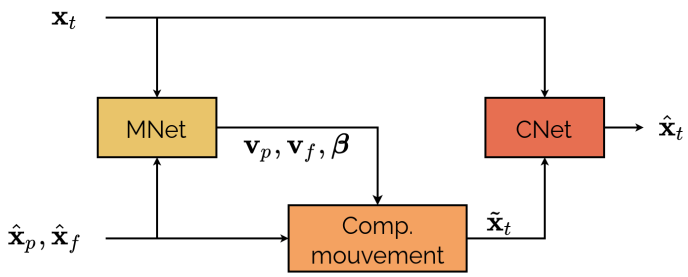


FIGURE 10 : Schéma de codage vidéo présentant une compensation de mouvement.

Dans certaines approches de la littérature [19], [21], MNet est composé d'un réseau d'estimation de flux optique pré-entraîné (SpyNet [22]), suivi d'un autoencodeur dédié à sa transmission. D'autres approches suggèrent que MNet peut être un simple autoencodeur, capable d'estimer et de transmettre les flux optiques directement sans réseau d'estimation dédié [23]-[25]. Dans un soucis de simplicité, cette seconde méthode est choisie.

Quelle que soit la méthode retenue, l'entraînement d'un schéma de codage avec informations de mouvement n'est pas aisé [23]. En particulier, l'obtention de flux optiques au décodeur nécessite souvent d'adapter l'entraînement, avec des réseaux pré-appris ou en intégrant d'autres termes que le débit et la distorsion dans la fonction de coût. Dans les

deux cas, cela peut dévier l'optimisation de l'objectif débit distorsion visé, dégradant les performances. Pour contourner ce problème, nous proposons un nouveau mode de codage appelé *Skip mode* [26] afin d'encourager l'apprentissage d'informations de mouvement.

Le schéma de codage de la Fig. 10 est amélioré au travers de l'ajout du Skip mode, présenté dans la Fig. 11. Ce mode de codage permet d'effectuer une copie directe de certains pixels de la prédiction, au lieu d'être exploitée par CNet. Ainsi, plutôt qu'optimisée au travers CNet, la prédiction peut être apprise directement au travers d'une copie, ce qui stabilise considérablement l'apprentissage. Le choix du mode de codage est réalisé pixel par pixel grâce au mode le codage $\boldsymbol{\alpha}$. Enfin, l'image décodée est reconstruite en additionnant les contributions de CNet et du Skip mode :

$$\hat{\mathbf{x}}_t = \overbrace{c(\boldsymbol{\alpha} \odot \mathbf{x}_t, \boldsymbol{\alpha} \odot \tilde{\mathbf{x}}_t)}^{\text{CNet}} + \overbrace{(1-\boldsymbol{\alpha}) \odot \tilde{\mathbf{x}}_t}^{\text{Skip mode}}, \quad \odot \text{ est un produit pixel par pixel.} \tag{17}$$

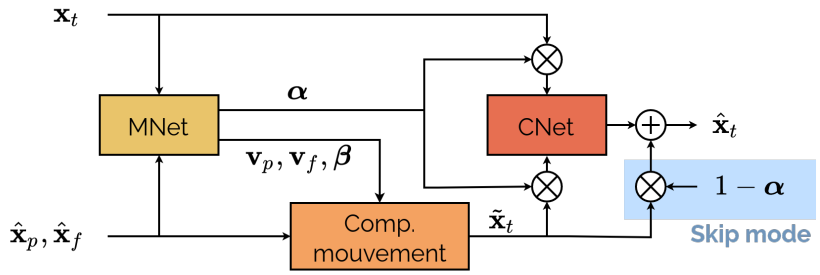


FIGURE 11 : Schéma de codage vidéo avec le Skip mode.

Le mode de codage $\boldsymbol{\alpha}$ doit être transmis au décodeur. Pour ce faire, il constitue une sortie supplémentaire de l'autoencodeur MNet, au côté des informations nécessaires à la prédiction. Afin de diminuer le débit, MNet implémente l'architecture de codage conditionnel développée dans le Chapitre 5. En effet, les deux images de référence présentes au décodeur permettent au codage conditionnel d'interpoler des informations de mouvement (au décodeur) sans nécessairement transmettre d'information depuis l'encodeur.

L'entièreté du schéma de codage (MNet et CNet) est entraîné pour optimiser une fonction de coût débit distorsion, comme présenté dans le Chapitre 4. La Figure 12 montre les gains en compression obtenus par l'ajout de MNet (avec codage conditionnel) et du Skip mode. Au total, cela permet de diminuer le débit de 34 % et d'atteindre des performances en ligne avec celles du HM, l'implémentation la plus performante du standard HEVC.

Néanmoins, différentes expériences ont mis en lumière certaines limitations du schéma

proposé, concernant particulièrement l'estimation des mouvements. L'amélioration du MNet sur ce point permettrait vraisemblablement d'augmenter de manière importante les performances du codeur vidéo proposé. Malgré ces limites, ce codeur vidéo basé apprentissage offre des performances encourageantes dans diverses configurations de codage. Pour cette raison, il a été soumis comme candidat pour la tâche vidéo du challenge CLIC 2021. Le Chapitre 7 détaille notre participation à ce challenge.

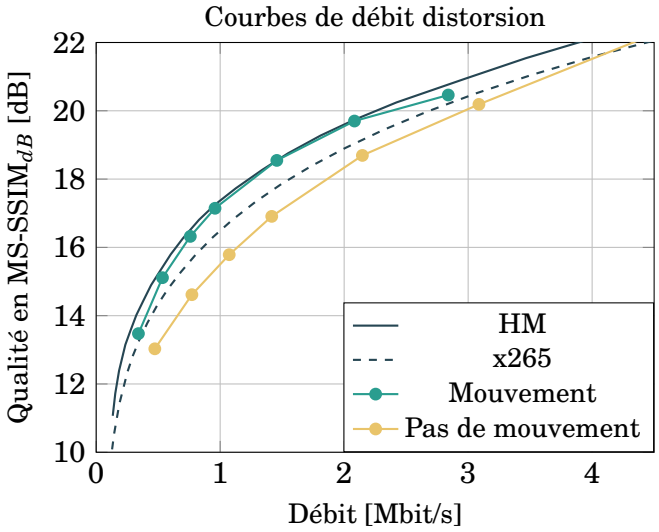


FIGURE 12 : Performances apportée par l'ajout d'information de mouvement.

Chapitre 7 : Vers une compression vidéo utilisable en pratique

Pour la première fois, le challenge CLIC 2021 a proposé une tâche de compression vidéo afin de comparer les systèmes de compression conventionnels et ceux basés apprentissage. L'objectif de ce challenge est de compresser 100 vidéos haute définition à un débit moyen de 1 Mbit/s tout en obtenant la meilleure qualité, mesurée au travers du MS-SSIM. Afin de promouvoir des systèmes réalistes, la métrique de la compétition inclut une pénalité sur la taille des décodeurs. Ainsi, le décodeur conçu dans le Chapitre 6 comporte 27 millions de paramètres, ce qui réduit d'environ 8 % son débit autorisé pour les vidéos.

Les règles du challenge ne portent que sur la moyenne du débit et de la qualité. En conséquence, les performances totales peuvent être améliorées en ajustant séquence par séquence le compromis débit distorsion. Pour ce faire, deux moyens sont mis en place : la compétition de structure de codage et la compétition de débit. Le codeur conçu dans les

chapitres précédents est en mesure de compresser des images I, P et B. Il est donc possible d'organiser ces trois types d'image à volonté, afin de déterminer séquence par séquence la structure de codage optimale.

Pour le moment, tous les codeurs basés apprentissage décrits dans cette thèse sont *mono*-débit. Pour une entrée donnée, les mêmes opérations sont effectuées, amenant la transmission de la même variable latente. Pour obtenir une compétition de débit, il serait nécessaire de mettre en compétition différents systèmes mono-débit ce qui nécessiterait le stockage des différents décodeurs. Cependant, cela entraînerait un nombre de paramètres trop importants selon les règles du challenge. Afin de profiter de la compétition de débit, le codeur conçu dans le Chapitre 6 est modifié pour faire varier son débit grâce à un petit nombre de paramètres supplémentaires. Afin d'adresser N débits différents, le codeur est doté de N précisions de quantification différentes. Chacune de ces précisions est optimisée pour l'un des N débits. À l'exception des précisions de quantification, les autres paramètres du codeur (MNet et CNet) sont communs à tous les débits.

La Figure 13 montre les gains en qualité obtenus à travers la compétition de débits et de structures de codage. Ce graphe présente également les scores obtenus par 3 approches conventionnelles : 2 implémentations d'HEVC (x265 et HM) et une implémentation de VVC (VTM). Si la compétition apporte un gain significatif de +0.47 dB, les performances restent en retrait par rapport aux approches conventionnelles (HM, VTM).

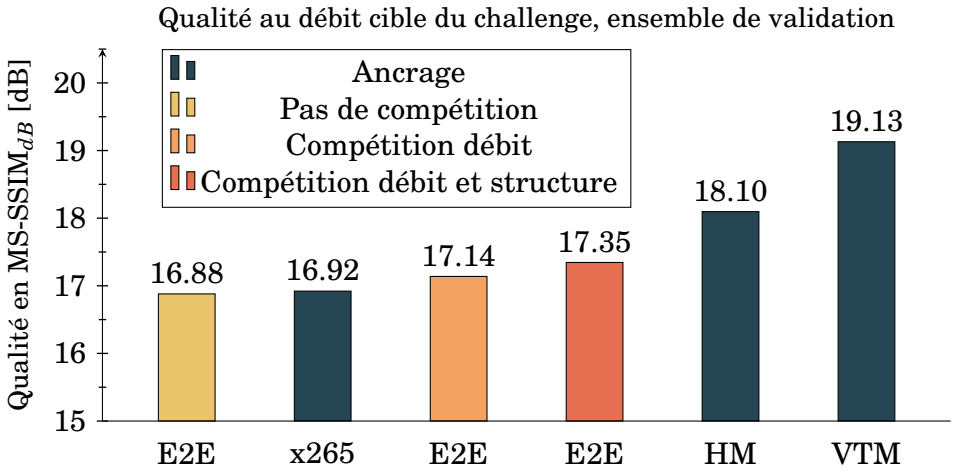


FIGURE 13 : Intérêt de la compétition. E2E désigne les systèmes basés apprentissage.

Il est intéressant de constater que les résultats présentés dans le Chapitre 6 (Fig. 12) montrent que le codeur basé apprentissage est a égalité avec le HM, ce qui n'est plus le cas ici. La différence majeure est qu'à présent, la compétition de stratégies de codage

est intégrée dans les résultats. La grande variété de choix de codage disponibles dans le HM et le VTM permet d'effectuer une compétition accrue, qui augmente les performances de manière importantes. Le codeur basé apprentissage ne bénéficie pas autant de la compétition, car il présente un nombre plus faible de stratégies de codage disponibles. Enfin, la taille d'un décodeur conventionnel est environ 100 fois plus faible que celle d'un décodeur basé apprentissage, ce qui laisse d'autant plus de place pour la représentation compressée des vidéos, permettant une meilleure qualité.

Après quelques ajustements pour permettre un encodage et décodage *cross-platform*, le codeur vidéo basé apprentissage a été proposé comme candidat au challenge. Étant donné les très bonnes performances du VTM, ce dernier a également été soumis comme candidat par nos soins. L'intégralité des résultats du challenge sont présentés Fig. 14. Onze codeurs ont été soumis au challenge, dont 5 systèmes entièrement basés apprentissage. Les autres sont basés sur des codeurs vidéos conventionnels avec éventuellement des modules de post-traitement basés apprentissage.

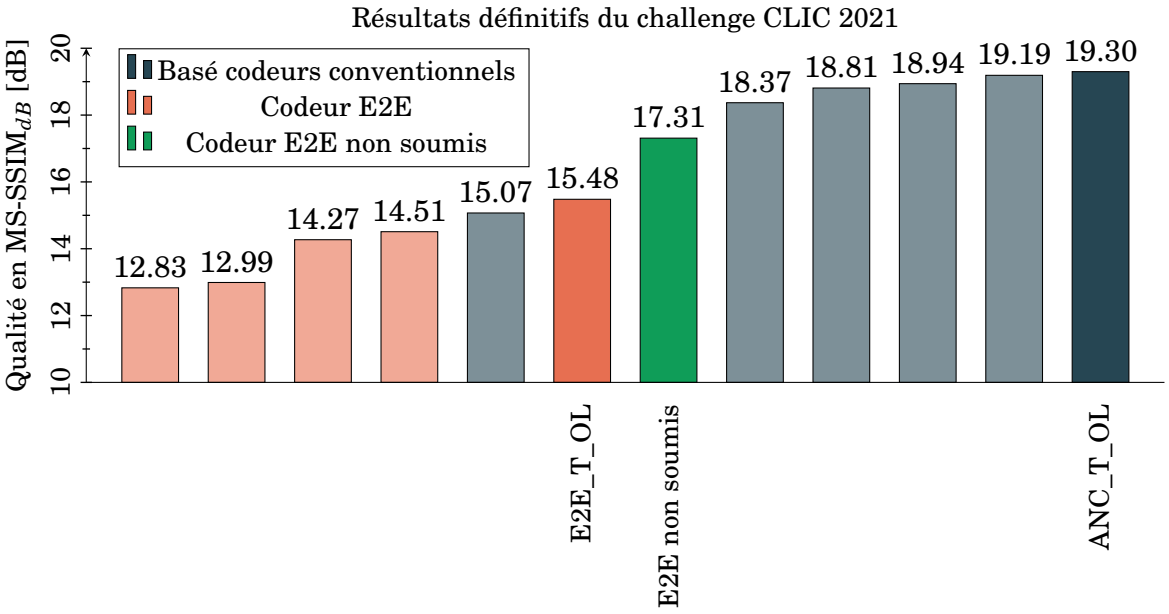


FIGURE 14 : Les entrées nommées correspondent à nos soumissions. E2E désigne les systèmes basés apprentissage

Les résultats du challenge fournissent plusieurs enseignements. Tout d'abord, les approches conventionnelles sont actuellement les plus performantes. C'est même notre soumission *ANC_T_OL* basée VTM, sans *aucun* module basé apprentissage, qui remporte

le challenge. Parmi la catégorie des codeurs basés entièrement apprentissage, le codeur *E2E_T_OL* conçu dans le cadre de cette thèse obtient les meilleures performances. Cela légitime les différents choix techniques motivés dans cette thèse.

Conclusion et perspectives

Cette thèse a reconsidéré entièrement les schémas de compression vidéo existants afin de concevoir un codeur vidéo basé apprentissage. Différentes contributions (architecture de codage conditionnel, ajout d'un mode de codage) ont permis d'obtenir un codeur vidéo basé apprentissage performant, compétitif avec les algorithmes conventionnels (HEVC) sous différentes configurations de codage. Ces résultats démontrent la pertinence des approches basées apprentissage. De plus, la rapidité des progrès de ce type de codeur laisse à penser qu'ils dépasseront bientôt, par un affinement de leurs éléments constituants, les algorithmes conventionnels.

Ainsi, tout au long de cette thèse, différents axes d'amélioration des performances ont été entrevus. Il a été mis en évidence que l'estimation de mouvement par le MNet est un élément sensible du système. Ce composant pourrait être amélioré au travers de l'utilisation de techniques issues de la littérature dédiée à l'estimation de flux optiques (*cost volume*). De plus, dans ce travail, les systèmes de compression vidéo sont entraînés à l'aide d'une configuration de codage comprenant 3 frames (voir Fig. 7). Adapter le codeur en le ré-entraînant pour sa configuration de codage cible améliorerait certainement les performances.

Les algorithmes conventionnels de compression mettent en place une compétition entre différents modes de codage, ce qui leur permet d'effectuer des opérations adaptées au signal à transmettre. Si certains éléments du codeur proposé vont dans ce sens (Skip mode, précision de quantification variable), l'adaptation au contenu des systèmes basés apprentissage demeure trop limité. Pour autant, la nature même des réseaux de neurones est de s'adapter aux données d'entraînement. Ainsi, entraîner un codeur sur une séquence vidéo particulière permettrait d'obtenir un codeur optimal, même si les paramètres du codeur sont alors à transmettre avec la vidéo. Des travaux récents [27], [28] ont montré que cette idée semble prometteuse.

Enfin, les codeurs présentés dans cette thèse ont été conçu avec peu de contraintes de complexité et comportent à ce titre plusieurs dizaines de millions de paramètres. Sur du matériel dédié, il est tout de même possible d'encoder et de décoder plusieurs images

720p par seconde. Néanmoins, les architectures proposées sont symétriques et requièrent donc autant de puissance de calcul pour l'encodeur que pour le décodeur. En pratique, le décodeur est souvent embarqué dans de nombreux appareils à faible consommation d'énergie (téléphone, *set-top boxes*). Mettre en place de nouvelles architectures asymétriques serait un pas important vers une utilisation pratique des schémas de codage basés apprentissage.

TABLE OF CONTENTS

List of Figures

List of Tables

INTRODUCTION

Context

THE ever-growing quantity of data conveyed over the Internet has a substantial impact on climate change. Even before the Covid-19 outbreak, video data represented around 80 % of the internet traffic [1]. Furthermore, the pandemic has made video consumption progress even further, obliging some streaming services (Netflix and YouTube) to reduce the video quality in order to avoid network congestions [2]. According to some estimates, video streaming already accounts for 1 % of the overall greenhouse gas emissions [3], resulting in the same impact than a country like Spain. As such, reducing the amount of data required to transmit or store videos could reduce the carbon emissions of video streaming, as long as the so-called rebound effect or Jevons paradox is avoided [4].

From the 90s onwards, video compression algorithms aim to reduce the size of videos while maintaining an acceptable quality for the users. Even though the primary motivation for compression algorithms was to cope with the limited bandwidth imposed by existing infrastructures, mitigating the environmental impact of online video streaming provides an additional motivation to design better compression algorithms. Video compression standards have been designed by standardization organisms such as the Moving Picture Experts Group (MPEG) and the International Telecommunication Union (ITU-T). Each generation of standards (AVC in 2003, HEVC in 2013 and VVC in 2020) introduced incremental refinements leading to substantial compression gains *i.e.* halving the rate required for equivalent quality.

These traditional coding algorithms process videos through successive operations, which are often linear. Given the incremental design of traditional coders, their operations are optimized separately. In recent years, deep neural networks have been used to solve a variety of problems, thanks to their ability to model virtually any function through a cost function optimization. In the context of compression, this allows to perform richer, non-linear operations which are jointly optimized to minimize both the size of the video and the visual degradations brought by its compression.

Recent works show that neural-based still image compression can be competitive

with state-of-the-art coding standards, by learning the functions mapping back and forth the image to a compressed representation. Due to the additional challenges brought by the temporal dimension, video coding remains a more demanding task for neural-based approaches. Yet reducing the size of video sequences is the actual issue to tackle. Consequently, this thesis studies the design of neural-based video compression schemes, to leverage the promising abilities of neural networks. Given the novelty of this field, the proposed neural-based coding scheme starts from a blank page and each element is reconsidered and motivated.

Structure of the thesis

Part I – Context and state of the art

The first part of the thesis presents important notions for image and video compression. Then, these notions are illustrated by their implementation in traditional coding algorithms and neural-based image coders.

Chapter 1 introduces the main concepts of lossless and lossy compression from an information theory standpoint. The usage of predictions, transforms and entropy coding is justified. Rate-distortion optimization is presented to balance the size and quality of compressed videos.

Chapter 2 illustrates the habitual implementation of the concepts expounded in the first chapter by traditional compression algorithms *e.g.* HEVC and VVC.

Chapter 3 starts with a presentation of deep neural networks. Then, they are used to implement a neural-based image coder, which is successively refined to reach state-of-the-art performance.

Part II – Learned video coding

The second part of the thesis offers to design a neural-based video coder from a blank page. The requirements of the coder and a realistic experimental framework are introduced. Then, the different components of the coder are thoroughly studied.

Chapter 4 motivates the requirements of the proposed neural-based video coder. A real-life video coding situation and relevant anchors are proposed to assess the performance of the neural coder.

Chapter 5 reconsiders the exploitation of a prediction. Using neural networks abilities, a novel architecture called conditional coding is motivated to better leverage prediction than residual coding.

Chapter 6 adds motion information to the neural coder to compute a more accurate prediction. The optimization of the motion-related components is fostered through the introduction of an additional coding mode. Finally, conditional coding is also implemented for motion information.

Chapter 7 evaluates the resulting neural-based coder through the participation to a video compression challenge. Due to the challenge constraints, the proposed coder is modified to operate under a continuous and variable rate constraint.

Contributions

The work carried out in this thesis contains several contributions to the field of learned image and video compression.

- Enhance entropy coding with a more expressive probability model;
- Reconsider the usage of residual coding to exploit prediction;
- Propose a generic architecture to leverage decoder-side information;
- Introduce an additional coding mode to both improve performance and ease the optimization of motion-related components.

More generally, this thesis presents a method to design a flexible neural-based video coder, competitive with modern video coding standards.

Part I

Context and State of the Art

CHAPTER 1

FUNDAMENTALS OF VIDEO CODING

1.1 Introduction

VIDEO coding algorithms aim to reduce the number of bits required to transmit or store sequences of images. Indeed, transmitting raw values of all pixels in the video would require a prohibitive amount of bits to be sent. For instance, a high-definition video is made of successive images composed of 1920×1080 pixels, where each pixel is usually represented as three 8-bit values indicating its colour, while the number of images per second often varies between 24 to 60. Therefore, conveying the raw pixel values requires to transmit:

$$\frac{24 \text{ images}}{1 \text{ s}} \times \frac{1920 \times 1080 \text{ pixels}}{1 \text{ image}} \times \frac{3 \times 8 \text{ bits}}{1 \text{ pixel}} \simeq 1.20 \text{ Gbit/s}. \tag{1.1}$$

This bandwidth exceeds the usual capacity of computing devices and highlights the need to reduce the number of bits representing a video. This is made possible by the fact that most videos exhibit a spatio-temporal structure. Different techniques leverage this organization to code videos with fewer bits, without losing any information. First, entropy coding algorithms exploit the statistical properties of videos by assigning less bits to the more probable pixel combinations. Second, invertible mathematical operations such as predictions and transforms are performed to reduce redundancies among video pixels. These techniques compose the lossless coding framework presented in the first part of this chapter.

In many cases, the size of a losslessly coded video exceeds the available bit budget of practical applications. Lossy coding algorithms allows to reach further data size reduction, at the expense of information loss. Due to this loss, the compressed video exhibits some form of distortion compared to the original one. This distortion is measured and balanced against the rate reduction offered, in a process called rate-distortion optimization. The second part of this chapter introduces the different notions related to lossy coding.

1.2 Lossless coding

1.2.1 Optimal rate and information theory

This section lays out the information theory framework for lossless coding. Let the message to code be a discrete random variable $\mathbf{s} = (s_1, \ldots, s_N)$ composed of N symbols following a distribution q, such as an N-pixel image. Lossless coding performs a *reversible* mapping from the message $\mathbf{s}$ to a set of K bits $\mathbf{b} = (b_1, \ldots, b_K)$. The process of mapping the message into a bitstream is called encoding and is performed by an encoder. Similarly, the decoder performs the inverse mapping, decoding the bitstream back to the initial message. To decrease the number of transmitted bits, the mapping is designed to leverage statistical properties of the distribution q. Indeed, the Shannon's source coding theorem states that the optimal number of bits for $\mathbf{b}$ is equal to the information I of the message:

$$I(\mathbf{s}) = -\log_2 q(\mathbf{s}), \text{ with } I(\mathbf{s}) \text{ in bits.} \tag{1.2}$$

Consequently, the most frequent messages are represented with fewer bits. When transmitting many successive messages drawn from q (*e.g.* successive N-pixel images), the average number of bits for the messages is called the rate. It is expressed in bits per message which translates in practice to bits per pixel, bits per image, bits per second, *etc.* Equation 1.2 yields that the minimal rate $R^\star$ is expressed through the Shannon entropy H [5], defined as:

$$R^\star = H(\mathbf{s}) = \mathbb{E}_{\mathbf{s}\sim q}\left[I(\mathbf{s})\right] = \mathbb{E}_{\mathbf{s}\sim q}\left[-\log_2 q(\mathbf{s})\right]. \tag{1.3}$$

In most cases, the underlying data distribution q is unknown. As such, it is not possible to compute $I(\mathbf{s})$ to determine the optimal mapping for each message. This issue is overcome by estimating q through an approximation p. Each message is now mapped to a stream of $-\log_2 p(\mathbf{s})$ bits. Thus, the rate R needed to code messages drawn from q becomes:

$$R = \mathbb{E}_{\mathbf{s}\sim q}\left[-\log_2 p(\mathbf{s})\right]. \tag{1.4}$$

The estimate p is an approximation of q, causing a non-optimal mapping. The resulting

rate overhead is often expressed by D_{KL}, the Kullback-Leibler divergence:

$$\begin{aligned} R - R^\star &= \overbrace{\mathbb{E}_{\mathbf{s}\sim q}\left[-\log_2 p(\mathbf{s})\right]}^{\text{Estimated code length}} - \overbrace{\mathbb{E}_{\mathbf{s}\sim q}\left[-\log_2 q(\mathbf{s})\right]}^{\text{Optimal code length}} \\ &= \mathbb{E}_{\mathbf{s}\sim q}\left[\log_2 q(\mathbf{s}) - \log_2 p(\mathbf{s})\right] \\ &= D_{KL}(q \,||\, p). \end{aligned} \tag{1.5}$$

The Kullback-Leibler divergence measures the mismatch between two probability distributions. This highlights that achieving efficient lossless coding comes down to performing accurate modelling of the data distribution of $\mathbf{s}$.

1.2.2 Arithmetic coding

Entropy coding algorithms are used to implement the mapping from a message to a stream of bits, based on the estimated distribution p. Among them, Huffman coding [29] reaches the theoretical rate expressed in Eq. (1.4) only when all the probabilities $p(\mathbf{s})$ are equal to the inverse of a power of two. This constraint on the distribution p often leads to a greater mismatch between q and p, increasing the rate overhead. Arithmetic coding [30] is more effective as it handles a probability model not constrained to inverse powers of two. It maps multiple messages into a single number $w \in [0,1[$, allowing the practical rate to converge to the theoretical one when the number of messages increases.

To illustrate the arithmetic coding process, let us suppose that a message $\mathbf{s}$ has a set of possible values Ω_s with a finite number of elements denoted as $\text{card}(\Omega_\mathbf{s})$. Figure 1.1 depicts the arithmetic coding algorithm. Each message is sequentially encoded by subdividing an interval $[u,v[$, initialized at $u = 0$ and $v = 1$. At each step, this interval is divided into $\text{card}(\Omega_\mathbf{s})$ segments, one for each possible values of $\mathbf{s}$, denoted as $\mathbf{s}^{(j)}$. The length l_j of each segment is proportional to the probability of its corresponding value:

$$l_j = \frac{1}{v-u} \cdot p\left(\mathbf{s}^{(j)}\right) \tag{1.6}$$

Finally, the interval boundaries u, v are set to the ones of the interval corresponding to the current message value and the next message is processed. Once the messages are processed, w is set to a value in $[u,v[$ and converted to $\mathbf{b}$, a set of bits. $\mathbf{b}$ must have enough bits so that any base-2 values starting with the bits $\mathbf{b}$ belongs to $[u,v[$.

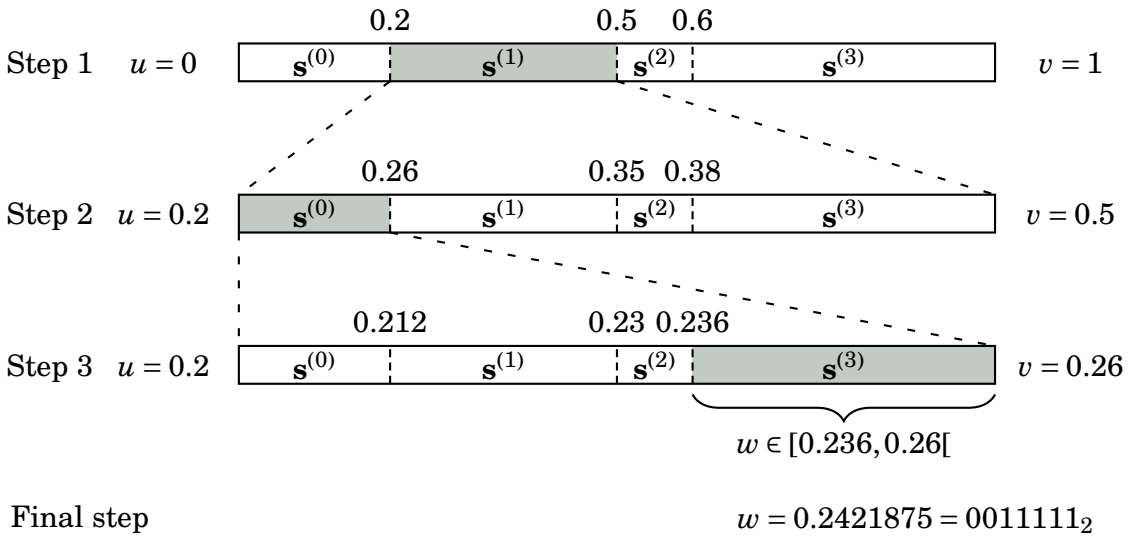


Figure 1.1: Arithmetic coding of 3 successive messages into a single number c. Each message has 4 possible values: $\Omega_s = \{\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \mathbf{s}^{(3)}\}$, whose respective probability are $\frac{2}{10}$, $\frac{3}{10}$, $\frac{1}{10}$ and $\frac{4}{10}$. At the end of the arithmetic coding process, the value $w = 0.2421875$ represents the three successive messages $\{\mathbf{s}^{(1)}, \mathbf{s}^{(0)}, \mathbf{s}^{(3)}\}$.

1.2.3 Practical limitations of arithmetic coding

The arithmetic algorithm might be intractable in practice. Indeed, if $\mathbf{s}$ represents a 8-bit greyscale image with N pixels, the number of possible values is $\text{card}(\Omega_{\mathbf{s}}) = 256^N$. Dividing an interval into that many segments is not practicable. One of the solutions to mitigate this issue is to rely on a *factorized* probability model, where the symbols composing $\mathbf{s}$ are assumed independent, modelled with their own distribution p_i:

$$p(\mathbf{s}) = \prod_{i=1}^{N} p_i(s_i). \tag{1.7}$$

This allows the arithmetic coding process to be conducted symbol-wise instead of message-wise. A message is now composed of a single scalar symbol, instead of N symbols. Continuing the example, arithmetic coding now works pixel-by-pixel, dividing the interval into $\text{card}(\Omega_{\mathbf{s}}) = 256$ segments at each step, which can be performed in practice. Some real-life arithmetic coders, *e.g.* the Context Adaptive Binary Arithmetic Coding (CABAC) [31], makes the additional assumption that the symbol takes binary value, leading to an even smaller number of possible values: $\text{card}(\Omega_{\mathbf{s}}) = 2$. This requires a binarization step, which decomposes each symbol into several binary values.

Despite being necessary to perform arithmetic coding, the assumption of symbols independence almost never holds in practice and results in a suboptimal probability model leading to a significant rate overhead. Let us illustrate the rate overhead through the lossless coding of a greyscale N-pixel image $\mathbf{x} = (x_1, \ldots, x_N)$. The minimal rate required to transmit the pixels is obtained thanks to Eq. (1.3):

$$R^\star = H(\mathbf{x}) = H(x_1, \ldots, x_N). \tag{1.8}$$

Yet, the pixels of $\mathbf{x}$ cannot be jointly coded due to the practical concerns previously raised. Instead, the pixels are independently coded and their probability models are assumed independent. This leads to the following rate:

$$R = \sum_{i=1}^{N} H(x_i) \geq R^\star = H(x_1, \ldots, x_N). \tag{1.9}$$

The rate overhead brought by the assumption of independent pixels can be expressed through the mutual information I [32]:

$$\begin{aligned} R - R^\star &= \sum_{i=1}^{N} H(x_i) - H(x_1, \ldots, x_N) \\ &= D_{KL}(p(x_1, \ldots, x_N) \,||\, \prod_{i=1}^{N} p(x_i)) \\ &= I(x_1, \ldots, x_N) \geq 0. \end{aligned} \tag{1.10}$$

As illustrated by the Kullback-Leibler divergence, the mutual information assesses the degree to which random variables are independent. If the variables are independent, there is no mutual information and no rate overhead. However, one pixel in a video sequence is likely correlated to its neighbouring pixels. This results in a non-negligible mutual information, increasing the rate.

To lower the resulting rate, bijective functions are applied to $\mathbf{x}$ prior to arithmetic coding. The purpose of these functions is to remove as much statistical redundancy as possible among $\mathbf{x}$ components, reducing the mutual information. The next two sections introduce two complementary techniques for redundancy removal, based either on prediction or on transform.

1.2.4 Predictive coding

In practice, a N-pixel video sequence $\mathbf{x} = (x_1, \ldots, x_N)$ is often decomposed into smaller sets of pixels, in order to make its processing more convenient. One natural decomposition is to split the video image-by-image and into small blocks of pixels. Yet, this leads to a rate overhead, illustrated on a toy example where the video $\mathbf{x}$ is split into two sets of pixels. The first M pixels $\mathbf{x}_{dec} = (x_1, \ldots, x_M)$ of the video have already been transmitted (*i.e.* available at the decoder) while the others pixels $\mathbf{x}_{enc} = (x_{M+1}, \ldots, x_N)$ are still to be sent. According to Eq. (1.10), transmitting $\mathbf{x}_{enc}$ without leveraging the information contained in $\mathbf{x}_{dec}$ leads to a rate increase of $I(\mathbf{x}_{dec}, \mathbf{x}_{enc})$.

Predictive coding palliates the effects of independent arithmetic coding for each set of pixels through a two-step process. First, the pixels $\mathbf{x}_{enc}$ are predicted thanks to the already transmitted pixels $\tilde{\mathbf{x}}_{enc} = f(\mathbf{x}_{dec})$, so that the decoder is able to reproduce the prediction. Second, $\tilde{\mathbf{x}}_{enc}$ and $\mathbf{x}_{enc}$ are combined through a *bijective* function m to obtain a prediction residual $\mathbf{r} = m(\mathbf{x}_{enc}, \tilde{\mathbf{x}}_{enc})$. Usually the function m is a simple difference between the prediction and the symbols to transmit *i.e.* $\mathbf{r} = \mathbf{x}_{enc} - \tilde{\mathbf{x}}_{enc}$. With an ideal prediction, the prediction residue $\mathbf{r}$ does not longer contain information predictable from $\mathbf{x}_{dec}$, meaning that $\mathbf{x}_{dec}$ and $\mathbf{r}$ are independent. The residue $\mathbf{r}$ is transmitted instead of $\mathbf{x}_{enc}$, yielding an optimal rate without overhead:

$$R^\star = H(\mathbf{x}_{dec}) + H(\mathbf{r}) = \overbrace{H(\mathbf{x}_{dec}, \mathbf{r})}^{R^\star} + \overbrace{I(\mathbf{x}_{dec}, \mathbf{r})}^{=0}. \tag{1.11}$$

Since the function m is bijective, no information is lost when transmitting the prediction residue $\mathbf{r}$ instead of $\mathbf{x}_{enc}$, that is: $H(\mathbf{x}_{dec}, \mathbf{x}_{enc}) = H(\mathbf{x}_{dec}, \mathbf{r})$. Following the transmission of $\mathbf{x}_{dec}$ and $\mathbf{r}$, the prediction is computed at the decoder side and the original pixel $\mathbf{x}_{enc}$ is retrieved from $\mathbf{r}$ and $\tilde{\mathbf{x}}_{enc}$.

Predictive coding aims to remove statistical dependencies between the two sets of pixels, so that sending $\mathbf{x}_{dec}$ and $\mathbf{x}_{enc}$ separetely does not bring a rate overhead. However, it does not target the statistical dependencies within each pixel blocks. According to Eq. (1.10), the pixel-by-pixel arithmetic coding of the residue $\mathbf{r} = (r_{M+1}, \ldots, r_N)$ results in a rate overhead of $I(r_{M+1}, \ldots, r_N)$ bits. This additional rate is targeted by transform coding, which aims to remove statistical relationship inside each pixel block. As such, the role of the prediction and of the transform is complementary. It is theoretically possible to feed the entire video to a transform (*i.e.* $\mathbf{x}_{enc} = \mathbf{x}$ and $\mathbf{x}_{dec} = \emptyset$), so that the transform removes

all the statistical relationship within the image, making the prediction step irrelevant. It is also possible to rely only on the prediction step, by predicting and transmitting each pixel x_i successively, using all the previously sent pixels: $\mathbf{x}_{enc} = \{x_i\}$ and $\mathbf{x}_{dec} = \{x_1, \dots, x_{i-1}\}$. In this case, it is the transform which becomes irrelevant. In practice, using both the prediction and the transform through a relevant partitioning of the video often leads to better performance.

1.2.5 Transform coding

While predictive coding focuses on removing statistical relationships between pixel blocks, transform coding targets the dependencies inside a set of pixels. Let us suppose that the current pixel block is a $H \times W$ greyscale image $\mathbf{x} = (x_1, \dots, x_{HW})$. It is losslessly compressed using arithmetic coding. As such, the $H \times W$ symbols are independently processed, yielding a rate overhead derived from Eq. (1.10):

$$R - R^\star = I(x_1, \dots, x_{HW}). \tag{1.12}$$

This issue is addressed by reducing the statistical dependencies among the pixels, leading to a lower mutual information. To this end, a *bijective* transform g is applied to $\mathbf{x}$.

Without loss of generality, the distribution of $\mathbf{x}$ is assumed to be centered, with a covariance matrix denoted as $\mathbf{\Sigma_x}$. Let us make the additional assumption that the transform is linear, such as it is expressed through a matrix $\mathbf{A}$:

$$\mathbf{y} = g(\mathbf{x}) = \mathbf{A}\mathbf{x}, \text{ with } \mathbf{A} \in \mathbb{R}^{HW \times HW}. \tag{1.13}$$

The covariance matrix of $\mathbf{y}$ can now be expressed as:

$$\mathbf{\Sigma_y} = \mathbb{E}\left[(\mathbf{A}\mathbf{x})(\mathbf{A}\mathbf{x})^\top\right] = \mathbf{A}\mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right]\mathbf{A}^\top = \mathbf{A}\mathbf{\Sigma_x}\mathbf{A}^\top. \tag{1.14}$$

Since $\mathbf{\Sigma_x}$ is real-valued and symmetric, there exists a matrix $\mathbf{A}$, composed of the eigenvectors of $\mathbf{\Sigma_x}$, which results in $\mathbf{\Sigma_y}$ being diagonal. Therefore, the different symbols of $\mathbf{y}$ are decorrelated *i.e.* there is no linear relationship between them. If the image follows a Gaussian distribution, this also translates into independent symbols [5] for $\mathbf{y}$. In this case, the rate overhead expressed by the mutual information in Eq. (1.12) is null. For non-Gaussian data, decorrelating the symbols reduces, but does not remove the statistical dependencies, still lowering the rate overhead.

The decorrelation matrix $\mathbf{A}$ is orthogonal *i.e.* $\mathbf{A}^{-1} = \mathbf{A}^{\top}$, ensuring the bijective nature of the transform and allowing to retrieve the original image at the decoder side, after the arithmetic coding step:

$$\mathbf{x} = g^{-1}(\mathbf{y}) = \mathbf{A}^{\top}\mathbf{y}. \tag{1.15}$$

This transform is known as the Karhunen-Loève Transform (KLT) [33]. Although limited to the removal of linear relationships, variations of this transform such as the Discrete Cosine Transform (DCT) are widely used in video coding. It should be noted that non-linear transforms could remove additional statistical relationships. This could be achieved by neural networks, which are known to effectively model non-linear transforms.

1.2.6 Conclusion

Lossless coding performs a reversible mapping from a message to a bitstream. Theoretically, leveraging the statistical properties of the message allows to reach a minimal number of bits, expressed by the message entropy.

In practice, the message is a video, which is usually decomposed into smaller sets of pixels corresponding to successive images, often additionally partitioned into blocks of pixels. These smaller sets of pixels are transmitted one after another as successive independent messages. The mapping from a message to a bitstream is performed through arithmetic coding, which assumes that the different symbols composing the message are independent. Yet, neither the messages nor their symbol are independent, leading to a significant rate overhead compared to the video entropy.

This issue is addressed by performing two complementary operations on the messages: prediction and transform. Predictive coding aims to remove dependencies between the successive sets of pixels, while transform coding fosters the independence between the symbols of a given set of pixels. To ensure the coding process remains lossless, these bijective operations are implemented at the encoder, and their inverses at the decoder. Figure 1.2 presents a typical example of a lossless encoding and decoding process.

1.3 Lossy compression

Modern lossless video coding algorithms offer a rate reduction up to a factor of three [6], often resulting in a rate of a few hundred of Mbit/s. The usual rate target for a video sequence is far smaller, being in the order of a few Mbit/s for a 1080p video. To

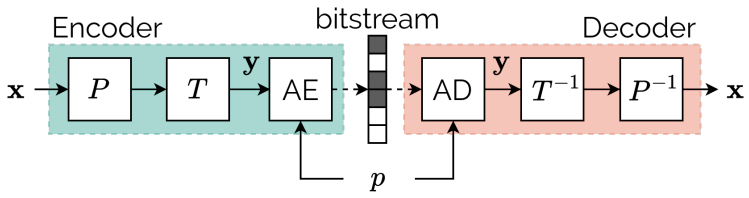


Figure 1.2: Block diagram for the lossless coding of a message $\mathbf{x}$ with a probability model p. The function P corresponds to prediction and function T corresponds to transform. AE and AD stand for arithmetic encoding and decoding.

this end, lossy coding removes information from the video before its transmission. This decreases the symbols entropy, allowing for lossless transmission at a lower rate. Yet, this information loss introduces distortion *i.e.* differences between the original and the compressed data. The trade-off between rate and distortion needs to be adjusted according to the application.

1.3.1 Discarding information

Discarding information is achieved by using a non-bijective (*i.e.* non-invertible) surjective function g during the encoding process. Indeed, such function is allowed to map several different inputs to an identical output. This increases the probability of the different possible output values, lowering the resulting entropy:

$$\mathbf{y} = g(\mathbf{x}) \Rightarrow H(\mathbf{y}) < H(\mathbf{x}), \text{ for a non-bijective} g. \tag{1.16}$$

The typical example of a function discarding information is the uniform scalar quantization Q (see Fig. 1.3), which maps any number $x \in \mathbb{R}$ to the closest multiple of $\Delta \in \mathbb{N}$:

$$Q_\Delta(x) = \Delta \, \texttt{round}\left(\frac{x}{\Delta}\right). \tag{1.17}$$

The quantization parameter Δ is called the quantization step. Setting a bigger value of Δ increases the amount of discarded information.

Besides quantization, others stages of the coding process can also be made lossy by relaxing the constraint on the bijective nature of the operations. For instance, the Principal Component Analysis (PCA) is a transform reducing the data dimensionality. It is similar to the KLT but the PCA matrix is composed of a subset of the most important eigenvectors of the data covariance matrix. The PCA is a lossy transform, discarding the

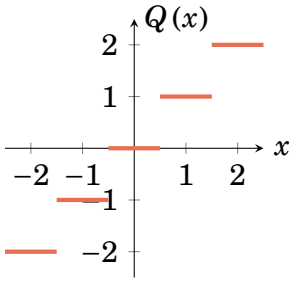


Figure 1.3: A uniform scalar quantizer Q with a quantization step $\Delta = 1$.

less relevant dimensions of the signal. Figure 1.4 presents a lossy coding scheme using both a quantization step and a non-invertible transform step. Since the transform is no longer constrained to be bijective, T^{-1} is not guaranteed to exist. As such, the encoder and decoder transforms are respectively denoted T_a and T_s.

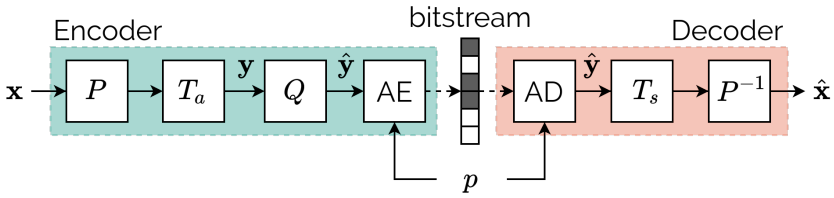


Figure 1.4: Lossy coding of a message $\mathbf{x}$ with a probability model p.

1.3.2 Distortion metrics

Discarding some information during the encoding process decreases the rate, at the expense of decoding a distorted version of the original data. The error between the original data $\mathbf{x}$ and the decoded data $\hat{\mathbf{x}}$ is called the distortion denoted as $D(\mathbf{x},\hat{\mathbf{x}})$. Among all the existing distortion metrics, two are considered in this manuscript. First, the Mean Squared Error (MSE):

$$D(\mathbf{x},\hat{\mathbf{x}}) = \text{MSE}(\mathbf{x},\hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2, \tag{1.18}$$

with N the dimension of $\mathbf{x}$ and $\hat{\mathbf{x}}$. For images and videos, it is common to translate the MSE (the distortion) into the Peak Signal to Noise Ratio (PSNR), which is to be maximized:

$$PSNR(\mathbf{x},\hat{\mathbf{x}}) = 10\log_{10}\frac{max^2}{\text{MSE}(\mathbf{x},\hat{\mathbf{x}})}. \tag{1.19}$$

The constant max is the maximum possible value for $\mathbf{x}$ *i.e.* 255 for 8-bit images and videos considered in this work. The PSNR translates the quality of the reconstruction $\hat{\mathbf{x}}$ compared to the original data $\mathbf{x}$.

The PSNR only considers pixel-wise error *i.e.* the energy of noise introduced in compressed images. Yet, the human visual system is not equally sensitive to the presence of noise in an image. Indeed, we are more tolerant when the noise occurs in high-contrast areas, while we are more sensitive to noisy low-contrast areas.

This is the reason behind the second quality metric used in this work: the Multi-Scale Structural Similarity Metric (MS-SSIM) [7]. This metric computes statistics on the entire image, at different scales, to better assess the perceived degradations due to the presence of noise. In short, the MS-SSIM measures whether the overall structure of the image is conserved *i.e.* a low-contrast area should not become a high-contrast one due to noise presence. The MS-SSIM returns a quality score ranging from 0 (worst quality) to 1 (best quality). Throughout this work, the MS-SSIM is often expressed on a logarithmic scale:

$$\text{MS-SSIM}_{dB}(\mathbf{x},\hat{\mathbf{x}}) = -10\log_{10}(1-\text{MS-SSIM}(\mathbf{x},\hat{\mathbf{x}})). \tag{1.20}$$

Figure 1.5 presents how the repartition of the same amount of noise is qualified by the PSNR and the MS-SSIM. In Fig. 1.5b, the noise is added to a low-contrast area (the girl's face). In Fig. 1.5c, the noise is added to a high-contrast area (the beanie). Since the noise added has approximately the same energy, the PSNR remains the same. However, the MS-SSIM penalizes significantly the introduction of noise in low-contrast areas. Adding the same amount of noise exclusively to high-contrast areas is better tolerated by the MS-SSIM, which is consistent with the human visual system. While some distortion metrics are more advanced than others, there is no perfect one and the choice of the distortion metric remains the subject of many discussions.

1.3.3 Rate-distortion optimization

The objective of lossy coding can be turned into the minimization of the distortion under a rate constraint of R_c bits:

$$\min\ D(\mathbf{x},\hat{\mathbf{x}}) \text{ subject to } R(\hat{\mathbf{x}}) < R_c, \tag{1.21}$$

(a) Original image

(b) Noise on the face
PSNR = 16.51 dB
MS-SSIM$_{dB}$ = 6.65 dB

(c) Noise on the beanie
PSNR = 16.83 dB
MS-SSIM$_{dB}$ = 10.76 dB

Figure 1.5: Influence of two different noise repartitions evaluated with two quality metrics: PSNR and MS-SSIM$_{dB}$ (higher means better). Example image is *ad249bba099568403dc6b97bc37f8d74* from the test set of the *Challenge on Learned Image Compression 2020* [16].

where $D(\mathbf{x},\hat{\mathbf{x}})$ is the distortion between the original data $\mathbf{x}$ and the compressed one $\hat{\mathbf{x}}$, and $R(\hat{\mathbf{x}})$ is the rate associated to the transmission of $\hat{\mathbf{x}}$. This optimization task is usually solved through Lagrangian optimization where the distortion term is balanced against the rate term. Using a Lagrangian formulation, the problem becomes:

$$\min\ J_\lambda = D(\mathbf{x},\hat{\mathbf{x}}) + \lambda R(\hat{\mathbf{x}}), \tag{1.22}$$

where λ is called the Lagrange multiplier. Thanks to the Lagrangian formulation [34], each solution of Eq. (1.22) for a given Lagrange multiplier λ corresponds to an optimal solution of the original Eq. (1.21) under a particular rate constraint R_c.

The value of J_λ is called the rate-distortion (RD) cost. Optimization of the RD cost is universally used in video coding. For instance, each coding possibility (choice of the prediction and transform) is rated with its RD cost. Similarly, learning-based approaches presented in this manuscript are trained to minimize a RD cost.

1.4 Conclusion

This chapter has introduced the theoretical framework for lossy video compression. Predictions and transforms are performed to obtain a compact representation of the video, more suited for compression. The encoding is then made lossy by selecting the information

that has to be kept, through a quantization step or a non-bijective transform. Finally, the information is mapped to a bitstream using a lossless coding algorithm such as arithmetic coding.

Lossy compression algorithms aim to minimize the distortion under a given rate constraint. This is achieved by optimizing a rate-distortion cost, expressed using a Lagrange formulation. The next two chapters introduce two implementations of lossy compression—traditional and learning-based one—both following closely the theoretical framework presented in this chapter.

CHAPTER 2

TRADITIONAL VIDEO CODING ALGORITHMS

2.1 Introduction

FROM the 90s onwards, standardization organisms such as the Moving Picture Experts Group (MPEG) and the International Telecommunication Union (ITU-T) have introduced several video coding standards. Advanced Video Coding (AVC) [8] was finalized in 2003, followed by High Efficiency Video Coding (HEVC) [9] in 2013 and recently Versatile Video Coding (VVC) [10] in 2020. These standards address the decoding process (*i.e.* from the bitstream to the decoded video). The encoder is required to produce a bitstream compliant with the decoding algorithm, resulting in an encoding process which tends to mirror the behaviour of the decoder.

The successive MPEG/ITU standards have continuously improved a coding algorithm based on the same paradigm, which is refered to as *traditional* video coding in this work. This chapters introduces how MPEG/ITU standards practically implement the principle expounded in Chapter 1 for individual images and video sequences.

2.2 Hybrid image coding

This section describes how traditional video coders compress a standalone image. In video coding, a standalone image is often called an Intra (I)-frame, since it focuses exclusively on the removal of the statistical dependencies within the image.

2.2.1 Representing the colour: the RGB and YUV colour spaces

A colour image is modelled by expressing each pixel value into a colour space. For instance, the RGB colour space decomposes a colour into three primary colours: red, green and blue.

Video sequences usually use the YUV colour space, due to its better correlation with the human visual system. Indeed, humans are more sensitive to variation of luminance (the quantity of light) than to variation of colours. As such, the YUV colour space dedicates the Y component to the representation of the luminance and leaves the colour representation for the U and V components.

Since the human visual system is less sensitive to the colour, a pre-processing step often reduces the resolution of the UV colour components prior to the compression process. This allows to reduce the rate with little visual degradations. For example, the YUV 420 format used in this work refers to a Y component of size $N \times N$ for a U and V components of size $\frac{N}{2} \times \frac{N}{2}$. In some cases, particularly for still images compression, the UV components are not sub-sampled. Such data format is called YUV 444. Since the Y channel represents most of the rate, the rest of this chapter focuses on the Y channel compression. The U and V channels are often independently coded using the same principles than the Y channel.

2.2.2 Overview of the video coding structure

In order to process an image, traditional coders split it into non-overlapping blocks of variable size and shape. Gathering pixels with similar properties within a block allows to perform adapted operations (prediction and transform) on this block. Figure 2.1 illustrates the partitioning of an I-frame: some large blocks gather low-frequency areas (background) while smaller ones isolate the edges of the players. Each block is processed with a different prediction and transform, yet all blocks follow a coding pipeline similar to the one introduced in Chapter 1.

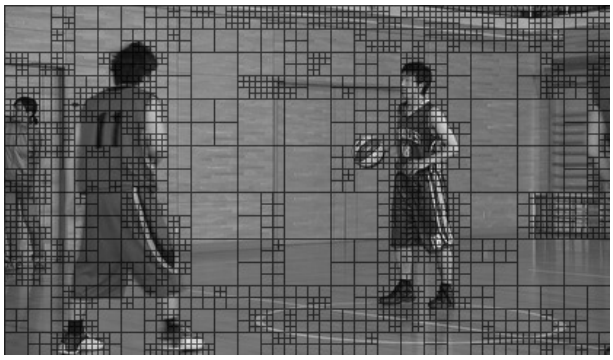


Figure 2.1: Partitioning of an intra frame in HEVC. Reproduced from [35].

Figure 2.2 presents the coding pipeline for a block $\mathbf{x}$. It consists of a prediction step (P) to reduce statistical dependencies between the current block and previously coded

ones. Then a transform step (T) removes spatial redundancies inside the block. Finally a quantization step (Q) selects the information transmitted using arithmetic coding.

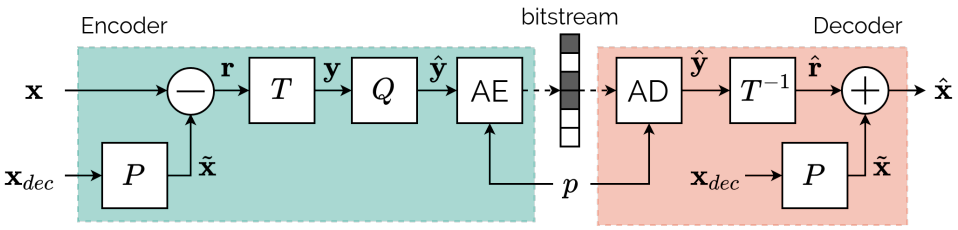


Figure 2.2: Block diagram of the coding of a block $\mathbf{x}$ using the reference pixels $\mathbf{x}_{dec}$ to compute the prediction P. T is an invertible linear transform, Q a quantization operation. The arithmetic encoding and decoding (AE and AD) rely on a probability model p.

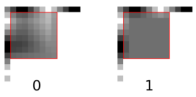
Video encoders implement multiple coding choices *i.e.* different ways of partitioning, predicting or transforming the signal. During the coding process, different coding choices are evaluated and the one minimizing the rate-distortion cost is selected. The competition mechanism enables the coder to arbitrate among very different behaviours according to the rate constraint. Thanks to the wide range of coding choices, competitive performance can be achieved on a large range of rate targets and video contents.

2.2.3 Intra prediction

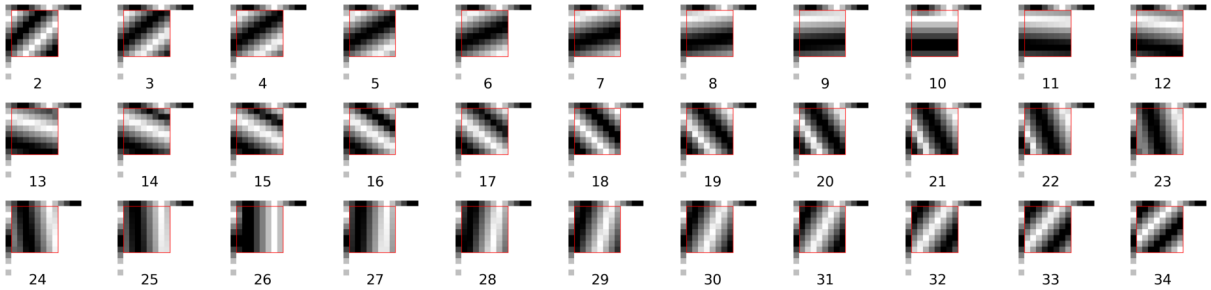
For intra frames, the purpose of the (intra) prediction step is to remove the redundancies between the block $\mathbf{x}$ and the previously transmitted blocks, which serve as reference. Since the blocks are encoded in a lexical order, the reference pixels available at the decoder $\mathbf{x}_{dec}$ to perform the prediction are located on the left and above of $\mathbf{x}$.

Multiple intra prediction modes are available, which correspond to different functions f when computing the prediction $\tilde{\mathbf{x}} = f(\mathbf{x}_{dec})$. Figure 2.3 gives an illustration of the 35 different prediction modes available in HEVC [9], illustrated for 8×8 blocks. Among them, 33 are directional prediction modes aiming to propagate directional patterns. The 2 others compute either a gradient (*Planar* mode) or predict the mean value (*DC* mode). All the intra prediction modes compute linear operations on the reference pixels.

Once the prediction $\tilde{\mathbf{x}}$ is available, it is subtracted from the block to code to obtain a residue $\mathbf{r} = \mathbf{x} - \tilde{\mathbf{x}}$. Thanks to the prediction, the residue exhibits less redundancies regarding its spatial or temporal neighbours, reducing the rate. As multiple prediction modes are available, the selected one must be signalled to the decoder.



(a) $Planar$ and DC modes



(b) Angular Modes

Figure 2.3: The 35 intra prediction modes of HEVC, illustrated on the Y channel of a 8×8 block. Reproduced from [35].

2.2.4 Transform

While the (intra) prediction step reduces the dependencies between the current block and its neighbours, the transform step aims to reduce the statistical redundancies within the block. This is achieved through a linear transform of the residual block $\mathbf{r}$.

Section 1.2.5 has introduced the KLT, which is the optimal linear transform. The KLT of a $N \times N$ residual block (reshaped as a vector of N^2 elements), is an $N^2 \times N^2$ matrix, resulting in N^4 multiplications for one block. For complexity considerations, video coders trade this optimal two-dimensional transform for 2 one-dimensional transforms, applied successively on the rows and columns of the residual block. The transform resulting from these 2 one-dimensional transforms is called a separable transform and requires only $2N^3$ multiplications. Yet, lowering the number of computation scomes at the price of sub-optimality. Indeed, separable transforms only remove statistical correlations across the horizontal and vertical axis and do not decorrelate pixels in the other directions (*e.g.* along the diagonals).

From the legacy image coding standard JPEG [36] to the recent VVC, one of the most used transforms is the Discrete Cosine Transform of type II (DCT-II) [37]. It is based on the average statistical properties of natural images, which are known to closely follow a first order auto-regressive model. In such model, a pixel value x_n is assumed to be a

combination of the neighbouring pixel x_{n-1} and a noise term ϵ_n:

$$x_n = \rho x_{n-1} + \epsilon_n, \tag{2.1}$$

with ρ the correlation between x_n and x_{n-1}, ϵ_n is an independent noise. When $\rho \to 1$, the KLT yields the DCT-II, a one-dimensional transform, decomposing a signal $\mathbf{x} \in \mathbb{R}^N$ into N frequential basis $B_k,\ k \in \{0, \dots N-1\}$:

$$B_k(n) = \frac{\xi(k)}{\sqrt{N}} \cos\left(\frac{(2n+1)k\pi}{2N}\right), \tag{2.2}$$

$$\text{with } n,\ k = \in \{0, \dots N-1\} \text{ and } \xi(k) = \begin{cases} 1 & \text{if } k = 0, \\ \sqrt{2} & \text{otherwise.} \end{cases}$$

Applying the DCT-II successively on the rows and columns of a block gives an equivalent two-dimensional transform, whose basis are shown in Fig. 2.4 for $N = 8$.

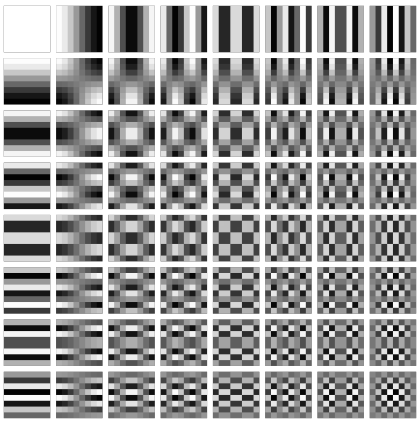


Figure 2.4: The 64 two-dimensional frequential basis of the DCT-II for a 8×8 signal.

More recent works focus on the statistical properties of the residual blocks, according to the type of prediction used to obtain them [38]. This allows to derive adapted transforms, such as the Discrete Sine Transform of type VII (DST-VII), which are implemented in HEVC and VVC. As for the partitioning and the prediction step, the transform type is selected through the minimization of the rate-distortion cost.

Once the residue has been transformed, a scalar quantization step Q takes place to discard information in order to lower the entropy of the residue. The amount of discarded information is parameterized by the quantization step Δ, which is itself set through the quantization parameter (QP): $\Delta \propto 2^{\frac{\text{QP}}{12}}$.

2.2.5 CABAC: Context Adaptive Binary Arithmetic Coding

At the end of the encoding process, all the values required for the decoding process are transmitted using the Context Adaptive Binary Arithmetic Coding (CABAC) algorithm. This encompasses signal data (*i.e.* the quantized and transformed residue) and multiple side-information such as the partitioning, the prediction mode or the transform type.

Due to the practical consideration presented in the previous chapter, the arithmetic coder operates only on binary messages, requiring all the values to be binarized beforehand. The probability model of a binary message is parameterized through a single parameter $P \in [0,1]$, corresponding to the probability of being equal to one. One benefit of the CABAC is to adjust P throughout the encoding process by adapting P to the context of the message. For instance, having many consecutive messages equal to 1 increases P. Since there are different sorts of message (residue, motion vectors, intra prediction mode), several independent contexts are used simultaneously, each one dedicated to a type of message.

2.3 From coding still images to video sequences

A video sequence exhibits a significant level of statistical dependencies along the temporal axis. As such, leveraging the relevant information contained in already sent frames allows to lower the amount of data to send for the current frame to code. Frames which depend on previously transmitted frames are called inter frames. Inter frame coding closely follows the processing described for intra frames. Inter frames are partitioned into blocks, which are processed using the prediction-transform-quantization pipeline, followed with arithmetic coding to transmit the selected information. However, an important additional prediction mode is available: the inter prediction.

2.3.1 Inter prediction

While the intra prediction focuses on the removal of spatial relationship between blocks, the inter-prediction targets temporal redundancies. It is achieved through a process called motion compensation. It usually applies a translational motion to a reference block, extracted from one of the already transmitted frames. The motion is parameterized by a motion vector $\mathbf{v} = \left(v_x, v_y\right)$, indicating the horizontal and vertical displacements between the block to code and its reference block. The motion vector value and accuracy is selected to optimize the rate-distortion cost. It is sent to the decoder to ensure proper decoding.

Modern coders refine the motion compensation process through bi-directional prediction. It consists in feeding two reference blocks (from two already received frames) alongside two motion vectors to the motion compensation process. The final predicted frame is the weighted average of the two motion compensations. This allows to improve the relevance of the prediction by reducing its noise, as illustrated in the following toy example. Let us suppose that a prediction operation w is applied on two reference blocks $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_f$ using their respective motion vectors $\mathbf{v}_p$ and $\mathbf{v}_f$. The prediction function allows to retrieve the block to code $\mathbf{x}_t$ up to noise terms $\boldsymbol{\epsilon}_p$ and $\boldsymbol{\epsilon}_f$:

$$w(\hat{\mathbf{x}}_p;\mathbf{v}_p) = \mathbf{x}_t + \boldsymbol{\epsilon}_p \ ; \ w(\hat{\mathbf{x}}_f;\mathbf{v}_f) = \mathbf{x}_t + \boldsymbol{\epsilon}_f. \tag{2.3}$$

The terms $\boldsymbol{\epsilon}_p$ and $\boldsymbol{\epsilon}_f$ denote two independent noises, whose variance σ_w^2 is assumed to be identical since they are obtained through the same prediction process w. These two intermediate predictions are combined through a sum weighted by $\beta \in [0,1]$. This yields the bi-directional prediction $\tilde{\mathbf{x}}$:

$$\tilde{\mathbf{x}}_t = \beta w(\hat{\mathbf{x}}_p;\mathbf{v}_p) + (1-\beta) w(\hat{\mathbf{x}}_f;\mathbf{v}_f) = \mathbf{x}_t + \beta\boldsymbol{\epsilon}_p + (1-\beta)\boldsymbol{\epsilon}_f. \tag{2.4}$$

The bi-directional prediction weighting β allows to select the contribution of each reference block. Typically, the value of β is taken from a discrete set of possible values. Very often, β is set to $\frac{1}{2}$, but other values can be useful in peculiar cases *e.g.* fading transitions from one scene to another. The purpose of the bi-directional prediction is to decrease the noise in the prediction error $\mathbf{r}$. From Eq. (2.4) it comes that the prediction error is:

$$\mathbf{r} = \mathbf{x}_t - \tilde{\mathbf{x}}_t = \beta\boldsymbol{\epsilon}_p + (1-\beta)\boldsymbol{\epsilon}_f. \tag{2.5}$$

Since $\boldsymbol{\epsilon}_p$ and $\boldsymbol{\epsilon}_f$ are independent noises with and identical variance σ_w^2, the energy of the prediction error is:

$$\sigma_r^2 = \beta^2\sigma_w + (1-\beta)^2\sigma_w = (2\beta^2 - 2\beta + 1)\ \sigma_w \leq \sigma_w \quad \forall\ \beta \in [0,1]. \tag{2.6}$$

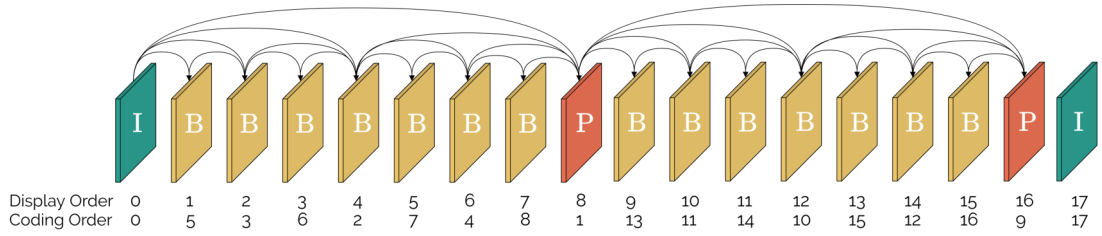
As such, the energy of the prediction error is smaller when using two reference blocks.

2.3.2 Coding structure

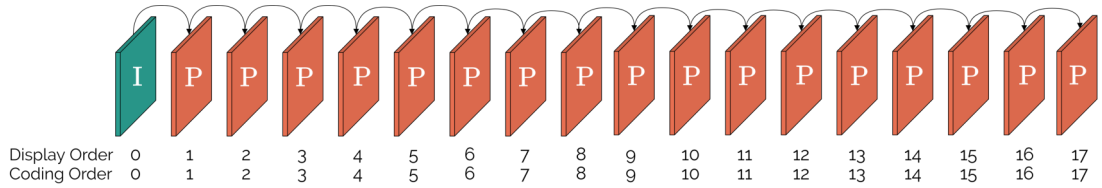
Encoders select the reference blocks from a restrained set of already transmitted frames, called reference frames, which are defined by the coding structure. The coding structure is a fixed repetitive pattern, often called a Group of Pictures (GOP). The number of frames in a GOP indicates the maximal temporal distance available for an inter prediction. Among the inter frames, a distinction is made between B-frames and P-frames. While B-frames have two reference frames available, P-frames have a single one.

Besides the performance aspect, the coding structure has to comply with a significant set of practical constraints. In most video streaming use cases, the Random Access (RA) coding configuration is used. This coding structure is depicted in Fig. 2.5a. The intra period which corresponds to the number of inter frames between two intra frames sets the maximum latency before the first decoded frame is displayed. Indeed, when an user wants to access a video (*e.g.* on his TV), the content can only be accessed within an I-frame. The common test conditions (CTC) of VVC [17] conditioned the intra period on the frame-rate in order to approximately have one I-frame per second. For instance, the VVC CTC states that a video at 30 frames per second has an intra period of 32 frames.

Some peculiar use cases such as video conferencing impose strong latency constraints. As it is not practical to wait for future frames to be decoded, specific coding structures are designed to compress frames without relying on future frames. This reduces the latency at the expense of compression efficiency. Such structures include the low-delay P (LDP) configuration, presented in Fig. 2.5b.



(a) Random access coding configuration. The GOP size is 8 and the intra period is 16.



(b) Low-delay P coding configuration.

Figure 2.5: Random access and Low-delay P coding configuration.

2.4 Conclusion

This chapter has introduced traditional video coding, which implements the operations motivated from a source coding perspective in the first chapter: prediction, transforms, quantization and arithmetic coding. The traditional video coding scheme is a particular implementation of the lossy video coding framework. This traditional implementation has to address a large set of constraints (complexity, handcrafted and linear operations), that restricts the compression performance. Moreover, each step of the coding scheme has to be separately designed through potentially misaligned proxy metrics, which can yield to suboptimal results.

The next chapter introduces deep neural networks (DNN) as a novel set of tools for image and video coding. Thanks to their ability to represent non-linear operations designed to optimize any (differentiable) functions, DNNs relax the constraints on the different steps of the compression process. Furthermore, DNNs enable end-to-end optimization of the whole coding scheme. Consequently, DNN-based coding scheme should reach further coding performance.

CHAPTER 3

LEARNING TO COMPRESS IMAGES

3.1 Introduction

NEURAL networks have recently seen a surge in their popularity and are at the heart of state-of-the-art systems for many different applications, such as image classification, natural language processing or generative modelling. This is due to the neural network ability of being an universal function estimator [11], allowing them to fulfil virtually any tasks. Neural networks consist of a succession of linear and non-linear operations, whose parameters are set to optimize a particular objective function. This allows to design powerful non-linear functions, achieving better performance than the more simple handcrafted (usually linear and optimized through proxy metrics) functions.

In the context of compression, neural networks are used as transforms, predictions and probability models. This chapter introduces the concepts and techniques of learning-based image compression, which will be extended to video coding in the next chapters. The first half of this chapter presents the main operations performed by neural networks as well as their training process. The second half of the chapter discusses neural networks applied to image coding. The overall framework of learned image coding is explained. Then, different implementations are proposed, successively increasing the coding scheme performance.

3.2 Basics of neural networks

3.2.1 Building a neural network

A neural network performs a function $\mathbf{y} = f(\mathbf{x})$, mapping an input $\mathbf{x}$ to an output $\mathbf{y}$. Such mapping can consist in labelling an image into different categories, removing noise in an input or obtaining a compressed representation, suited for arithmetic coding. All the neural networks presented in this manuscript are modelled as a succession of functions f_i

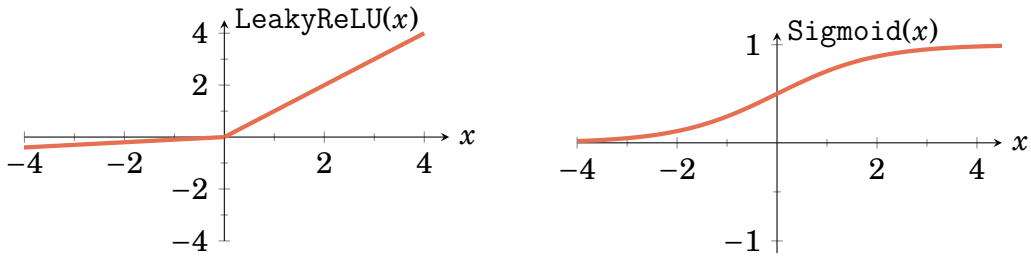
called layers. The network output is obtained by applying successively all the layers:

$$\mathbf{y} = f(\mathbf{x}) = f_L \circ f_{L-1} \circ \ldots \circ f_1 (\mathbf{x}), \tag{3.1}$$

where $\mathbf{y}$ is the network output and $L \in \mathbb{N}$ denotes the number of layers. The organization and the operations performed by all the layers are called the network architecture. Each layer f_i outputs a hidden state $\mathbf{h}_i$, *i.e.* a vector acting as an internal representation of the input data. The hidden states represent prominent features of the data *e.g.* edges or textures for images. In most cases, a layer f_i is decomposed as a linear operation, represented by a matrix $\mathbf{W}_i$ and a bias $\mathbf{b}_i$, followed by a non-linear operation σ_i:

$$\mathbf{h}_i = f_i (\mathbf{x}) = \sigma_i (\mathbf{W}_i \mathbf{x} + \mathbf{b}_i). \tag{3.2}$$

Some of the most common non-linearities σ such as the Leaky Rectified Linear Unit (ReLU) or the sigmoid functions are illustrated in Fig. 3.1.



(a) $\texttt{LeakyReLU}(x) = \max(\epsilon x, x)$ with $\epsilon \ll 1$.

(b) $\texttt{Sigmoid}(x) = \frac{1}{1+e^{-x}}$

Figure 3.1: The LeakyReLU and Sigmoid non-linearity functions.

The values of the matrix $\mathbf{W}_i$, the bias $\mathbf{b}_i$, as well as the optional parameters of σ_i, form the layer parameters, denoted as $\boldsymbol{\theta}_i$. The set of all the layers parameters constitutes the network parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L\}$. The values of all the parameters are set during the training stage, according to the function $f(\mathbf{x};\boldsymbol{\theta})$ the network aims to model.

3.2.2 Training a neural network

The training objective is to find the optimal set of parameters $\boldsymbol{\theta}^\star$ to represent the desired function. Ideally, this function maps an input $\mathbf{x}$ to an output $\mathbf{y} = f(\mathbf{x};\boldsymbol{\theta})$, which has to be close to the desired output $\mathbf{y}_{obj}$. Moreover, the network must generalize to (*i.e.* perform well on) all inputs $\mathbf{x}$ drawn from a distribution q, such as the natural images distribution.

These requirements yield the following optimization problem:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim q}\left[J\left(f(\mathbf{x};\boldsymbol{\theta}),\, \mathbf{y}_{obj}\right)\right], \tag{3.3}$$

where $\mathcal{L}$ is the loss function to minimize and J is a per-example cost, measuring the network performance on one single example. The minimization problem is solved using an iterative gradient descent algorithm [39]. After a random initialization of the network parameters $\boldsymbol{\theta}_0$, the parameters are updated as follows:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}),\ \eta \in \mathbb{R}. \tag{3.4}$$

Here, $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$ denotes the gradient of the loss function with respect to the network parameters $\boldsymbol{\theta}$. This equation states that at each step, the parameters are updated towards the steepest descent of the average loss function. The learning rate η sets the displacement magnitude. In theory, the gradient descent algorithm converges towards the optimal set of parameters $\boldsymbol{\theta}^\star$ as long as the loss function is convex [40].

Two issues arise in practice. First, there is no guarantee that the loss function is convex. Second, it is not possible to compute the exact expectation of the per-example cost J over the whole distribution of $\mathbf{x}$. The first issue remains an open problem, sometimes tackled through the adaptation of the network architecture, which affects the convexity of f. The second issue is solved using the Stochastic Gradient Descent (SGD) [39], which estimates the gradient based on a reduced subset of samples $\{\mathbf{x}^{(i)},\ i = 1,\ldots,B\}$, called a batch. As a result, the average loss function becomes:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{B}\sum_{i=1}^{B} J\left(f(\mathbf{x}^{(i)};\, \boldsymbol{\theta}),\, \mathbf{y}_{obj}^{(i)}\right). \tag{3.5}$$

Estimating the derivative of the loss with respect to each network parameter is achieved using the backpropagation algorithm [11]. It consists in propagating the gradient backward, from the output to the input layers. This is often called the backward pass. By analogy, the process of computing the network output $\mathbf{y} = f(\mathbf{x};\boldsymbol{\theta})$ is called the forward pass. Let us illustrate the backward pass on the L-layer network described in Eq. (3.1). For the sake of brevity, the final network output y is supposed to be scalar and each layer f_i is assumed to have a single scalar parameter θ_i and a scalar hidden state h_i. The loss

gradient with respect to a parameter θ_i is expressed using the chain rule as:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{\partial \mathcal{L}}{\partial y} \times \frac{\partial y}{\partial h_{L-1}} \times \frac{\partial h_{L-1}}{\partial h_{L-2}} \times \ldots \times \frac{\partial h_{i+1}}{\partial h_i} \times \frac{\partial h_i}{\partial \theta_i}. \tag{3.6}$$

The successive multiplications represent the gradient propagation from the output layer to the i-th layer. This equation highlights a major constraint of neural networks training: all layers as well as the loss function must be (piece-wisely) differentiable with respect to both their input and their parameters.

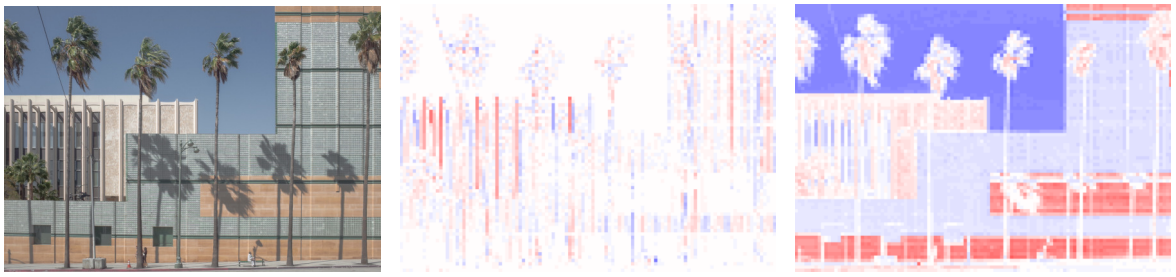
3.2.3 Convolutional neural networks

A layer composed of a generic matrix multiplication followed by a non-linearity, as described in Eq. (3.2) is called a fully connected layer. Similarly, a neural network made of fully connected layers is a Fully Connected Network (FCN). It is the most generalist network type and can theoretically learn to perform any function, with sufficent number of parameters and layers. However, the FCN universality suffers from an increased complexity. Let us consider the case where the network input is a $N \times N$ greyscale image. Suppose that the first layer of the network is a fully connected layer without dimensional reduction *i.e.* represented by a matrix $\mathbf{W} \in \mathbb{R}^{N^2 \times N^2}$. Then, each row of $\mathbf{W}$ contains N^2 parameters, used only once to weight each input pixels in order to compute a single output pixel. This results in a prohibitive number of parameters *e.g.* more than 4 billion parameters for a single layer applied on a 256×256 input image.

The number of parameters is imposed by the dependence on the spatial position. Indeed, each output pixel of a fully connected layer is obtained by its own set of parameters, applied on all input pixels. As such, detecting the same feature for all $N \times N$ input pixels requires to learn a dedicated set of $N \times N$ parameters for each output pixel, even though the task to perform is identical for all pixels. This issue is solved by trading the generic matrix multiplication for a convolution between the input pixels and a small set of parameters called a kernel. A unique kernel is slid over the input, computing each output pixel through a weighted sum of the neighbouring pixels. As such, the convolutional layer is invariant to translation [11]. That is, the same processing is applied regardless of the position of the input pixels within the image. Insightful convolutional layers visualisations are available in [41].

The output of the convolution with one kernel is called a feature map. It is a filtered version of the input, aiming to extract relevant features (*e.g.* edge detection). Usually,

more than one feature is extracted from the input by applying F different convolutional kernels in parallel. Successive convolutional layers are often applied to refine the extracted features. In order to process a three-dimensional input (F feature maps with a $N \times N$ spatial dimension), each kernel is three-dimensional and composed of $F \times k \times k$ weights. Fig. 3.2 presents different feature maps extracted from an input image.



(a) Input image.

(b) A feature map extracting vertical edges.

(c) A colour-related feature map.

Figure 3.2: Different feature maps extracted from an input image. Red and blue denote respectively positive and negative values.

Thanks to the small spatial dimension of the kernel (usually $k \in \{3,5,7,9\}$), the number of parameters remains reasonable. Besides the reduction of the memory footprint, this eases the training of the network. Indeed the same parameters are applied multiple times on an image, making their average gradient less noisy. Moreover, convolutional layers present additional interesting properties. Sharing the convolution parameters across all spatial positions enables the processing of variable size inputs. Finally, convolutions provide a simple means of performing spatial downsampling or upsampling of the input data. Downsampling is easily achieved by sliding the convolution kernel with a non-unitary stride. Upsampling is often implemented through sub-pixel convolutions [42] or transposed convolutions [43].

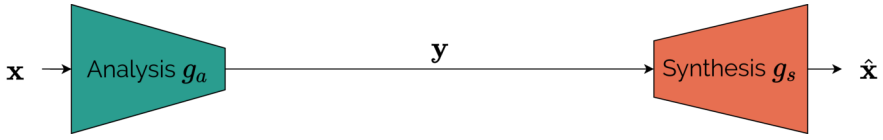
3.3 Learning-based lossy image coding

3.3.1 Autoencoders

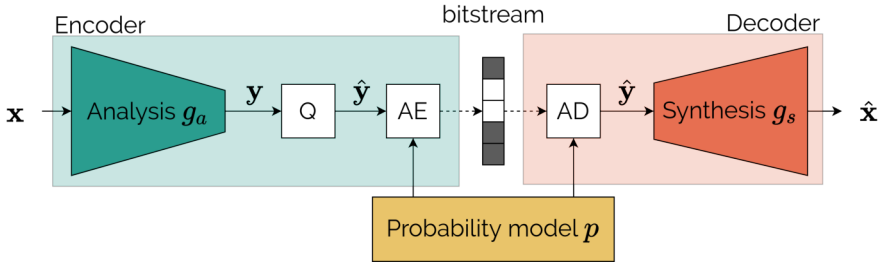
Since Ballé [12] and Theis' work [13] in 2017, most learned compression systems [44]–[49] operate using autoencoders. Autoencoders are specific neural network architecture,s whose purpose is to find a latent representation $\mathbf{y}$ of an input data $\mathbf{x}$. Ideally, $\mathbf{y}$ is a meaningful

low-dimensional representation of $\mathbf{x}$, which encodes the most important properties of the input. Beside compression, this simplified representation can be used for a wide variety of task *e.g.* it is a more convenient input for classification or noise removal.

To compute the latent representation, an autoencoder is composed of two sub-networks: the analysis and the synthesis transforms, presented in Fig. 3.3a. First, the analysis transform g_a maps an image-domain input $\mathbf{x}$ to a latent domain variable $\mathbf{y}$. Then the synthesis transform g_s maps back the latent variable $\mathbf{y}$ to the input domain by computing $\hat{\mathbf{x}}$. To foster the learning of a meaningful representation, constraints are imposed on the latent variable $\mathbf{y}$ [11]. In the context of compression, the constraint targets the entropy (*i.e.* the rate) of the latent variable.



(a) A vanilla autoencoder.



(b) Autoencoder-based image coding scheme. Q denotes a uniform scalar quantization function, AE and AD stand for arithmetic encoding and decoding with p the latent probability model.

Figure 3.3: The autoencoder architecture.

Similarly to autoencoders, lossy compression schemes aims to obtain a low-entropy representation encoding the most important features of the input. As such, autoencoders are a straightforward architecture to implement learned compression systems. The analysis is fed with the input image, computing a latent variable to be losslessly sent to the decoder. At the decoder, the synthesis uses the latent variable to reconstruct the input as accurately as possible. Such learning-based systems are presented in Fig. 3.3b.

In order to be efficiently sent to the decoder, the latent variable is losslessly compressed using arithmetic coding. Consequently, the processing pipeline of a compression autoencoder features two additional components: a quantization step and a latent probability model. The quantization step is present since arithmetic coding only operates on discrete

data. The latent probability model is required to allocate less bits to the most probable latent variable values. The synthesis and analysis transforms are not constrained to be the inverse of each other. Thus, the quantization step is not the sole lossy step in the coding pipeline: the analysis transform also removes information. The resulting autoencoder-based architecture yields a coding scheme similar to the abstract lossless coding pipeline presented in Fig. 1.4. The analysis g_a and synthesis g_s transforms represent the transforms T_a and T_s in Fig. 1.4. So far, no prediction step is used in the learned coding scheme.

3.3.2 Training a learned coding scheme

The analysis, the synthesis and the probability model are implemented with neural networks, whose parameters are learned during a training stage. This enables the joint optimization of the entire coding scheme, in an *end-to-end* fashion. This is different from traditional approaches, where each step (prediction, transform, arithmetic coding) is separately and incrementally optimized potentially leading to suboptimal systems.

Yet, according to Eq. (3.6), the training process requires each operation of the coding scheme to be differentiable with respect to its input. This is not the case for the arithmetic coding algorithm which operates on discrete data. Additionally, the quantization step derivative is defined almost everywhere, but is always equal to zero (see Fig. 1.3). Using it as such would prevent the gradient propagation to the other layers, as the chain rule equation (3.6) would feature multiplications by zero. In the literature, three main methods are proposed to circumvent the quantization differentiability issue during the training:

- the straight-through estimate [13] sets the quantization derivative to 1 during the backward pass while still performing the rounding operation in the forward pass;
- the soft quantization [50] approximates the quantization with a differentiable function (sum of sigmoids) for both forward and backward passes;
- the additive noise [12], [46] models the quantization by adding an independent uniform noise for the forward and backward passes. At high rate, this is known to mimic the quantization effect [32].

Once the training stage is done, all three methods use the real quantization to actually compress images. All the systems presented in this manuscript rely on the additive uniform noise model for their training, as it usually offers better performance. Indeed,

recent works [51] hint that it may be due to the regularization effect of the stochastic noise addition, helping the learning of relevant and expressive latent variables. In this setting, the quantized latent variable $\hat{\mathbf{y}}$ is replaced by the *relaxed* latent variable $\tilde{\mathbf{y}}$:

$$\tilde{\mathbf{y}} = \mathbf{y} + \mathbf{u}, \text{ with the noise } \mathbf{u} \sim \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right). \tag{3.7}$$

Arithmetic coding operates exclusively on discrete data and is thus non-differentiable. Since it is a lossless step within the coding pipeline, it is removed during the training stage, without altering the value of the decoded image. As explained in Section 1.2.2, arithmetic coding rate can be estimated by the cross entropy:

$$R = \mathbb{E}_{\hat{\mathbf{y}} \sim q_{\hat{\mathbf{y}}}} \left[-\log_2 p_{\hat{\mathbf{y}}}(\hat{\mathbf{y}}) \right]. \tag{3.8}$$

Yet, there is no quantized latent variable $\hat{\mathbf{y}}$ during the training stage due to the additive noise model. Instead, we have the relaxed latent variable $\tilde{\mathbf{y}}$, whose probability density function (PDF) is:

$$\begin{aligned} p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}}) = \left(p_{\mathbf{y}} * p_{\mathbf{u}}\right)(\tilde{\mathbf{y}}) &= \int_{\tilde{\mathbf{y}}-\frac{1}{2}}^{\tilde{\mathbf{y}}+\frac{1}{2}} p_{\mathbf{y}}(a)\, \mathrm{d}a \\ &= \int_{-\infty}^{\tilde{\mathbf{y}}+\frac{1}{2}} p_{\mathbf{y}}(a)\, \mathrm{d}a - \int_{-\infty}^{\tilde{\mathbf{y}}-\frac{1}{2}} p_{\mathbf{y}}(a)\, \mathrm{d}a \\ &= c_{\mathbf{y}}\left(\tilde{\mathbf{y}} + \frac{1}{2}\right) - c_{\mathbf{y}}\left(\tilde{\mathbf{y}} - \frac{1}{2}\right), \end{aligned} \tag{3.9}$$

with $p_{\mathbf{u}}$ the PDF of the continuous uniform distribution introduced in Eq. (3.7). The above equation allows to express the PDF of the relaxed latent variable through $c_{\mathbf{y}}$, the cumulative distribution function of the non-quantized latent variable $\mathbf{y}$. Moreover, it highlights that the $p_{\tilde{\mathbf{y}}}$ is an interpolation of $p_{\hat{\mathbf{y}}}$, with equality for all integers:

$$p_{\tilde{\mathbf{y}}}(n) = p_{\hat{\mathbf{y}}}(n), \ \forall \ n \in \mathbb{Z}. \tag{3.10}$$

Consequently, modelling the non-quantized latent variable distribution $p_{\mathbf{y}}$ allows to obtain the distribution of the quantized (and relaxed) latent variable, as illustrated in Fig. 3.4. Finally, the continuous $p_{\tilde{\mathbf{y}}}$ is used as a proxy of $p_{\hat{\mathbf{y}}}$ during the training stage, yielding the training rate:

$$R_{train} = \mathbb{E}_{\tilde{\mathbf{y}} \sim q_{\tilde{\mathbf{y}}}} \left[-\log_2 p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}}) \right]. \tag{3.11}$$

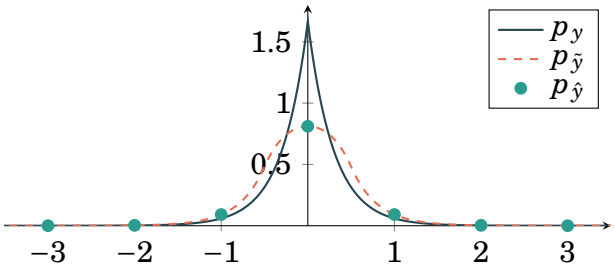


Figure 3.4: Distribution of the relaxed ($p_{\tilde{y}}$) and quantized ($p_{\hat{y}}$) latent variables.

The coder objective is to minimize both the rate and the distortion. The latter is denoted as a function d, *e.g.* the mean-squared error or the MS-SSIM. The rate constraint is expressed through a Lagrange multiplier λ, yielding the loss function:

$$\mathcal{L}_\lambda = \mathbb{E}_{\mathbf{x}} \left[d(\mathbf{x}, \hat{\mathbf{x}}) - \lambda \log_2 p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}}) \right] \tag{3.12}$$

3.3.3 Initial architecture

Because of their desirable behaviour for image processing, convolutional neural networks are used to implement the analysis transform g_a and the synthesis transform g_s. A typical architecture of a learning-based image coder [12], is presented in Fig. 3.5. Varying the number of kernels F in the internal convolutions provides a simple—yet effective—way of trading computational complexity for performance. Since g_a and g_s are convolutional transforms, the latent variable $\mathbf{y} = g_a(\mathbf{x})$ consists of multiple two-dimensional feature maps. To foster the reduction of the latent variable entropy, the analysis transform performs a spatial downsampling while the synthesis transform upsamples the latent variable to restore the original input dimension. For an $H \times W$ input image, it is common [44]–[46] to have latent feature maps with a spatial dimension of $\frac{H}{16} \times \frac{W}{16}$. In practice, the number of latent feature maps F_y is set big enough to not constrain the autoencoder (*i.e.* the constraint is imposed on the entropy of the latent, not their dimensionality).

An additional benefit of using convolutional transforms is the seamless processing of colour images. Traditional approaches consider the different colour channels separately. Here, the convolutional nature of the learned coding scheme allows to naturally process colour images. Indeed, a colour image (*e.g.* a RGB image or a YUV video sequence) is seen by the first convolutional layer as a three-feature-maps input.

The non-linearity function implemented in neural image coder is usually a variation of the Generalized Divisive Normalization (GDN) [52]. The GDN is designed as a

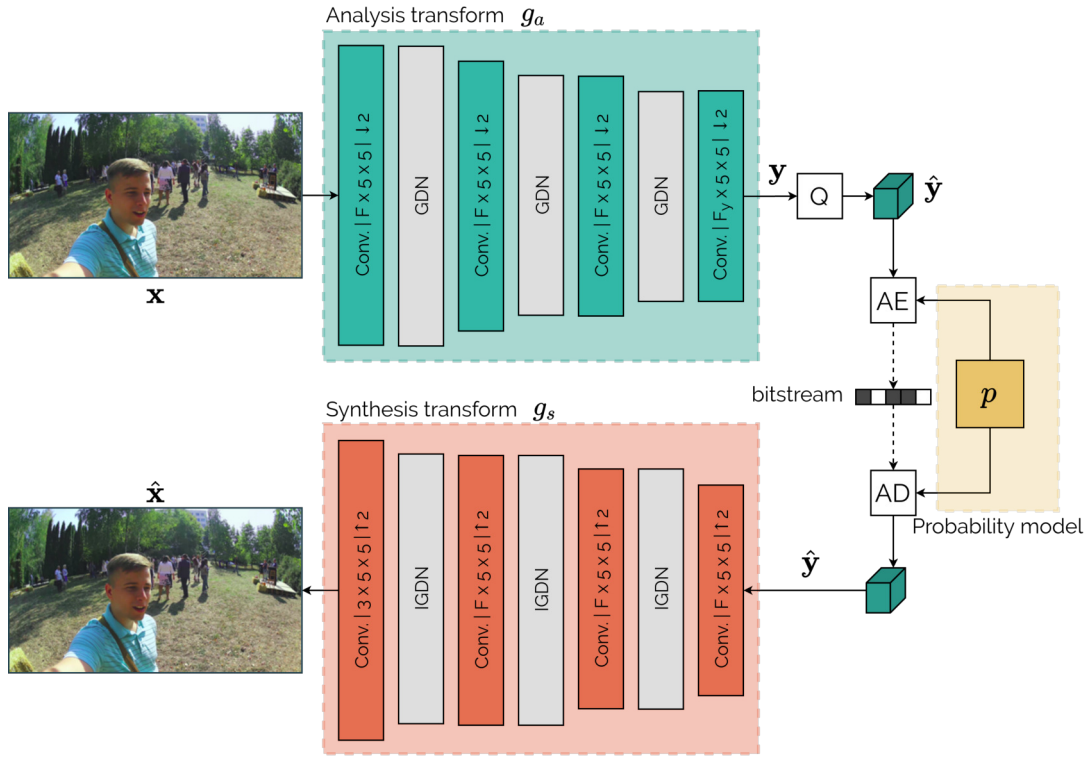


Figure 3.5: Architecture of a typical learning-based image coder. Conv $F \times 5 \times 5$ denotes a convolutional layer with F output feature maps and a 5×5 kernel. Q is a scalar uniform quantizer, AE and AD respectively denote arithmetic encoding and decoding with a probability model p.

Gaussianizing transform, with the objective of producing a set of marginally independent Gaussian variables. Here, the Gaussian distribution of the variables is not of primary interest but the independence between the different latent feature maps is important, as they are transmitted independently. Let us denote $w_i(k)$ the i-th pixel in the k-th feature map produced by a convolutional layer. The purpose of the GDN operation is to reduce the statistical dependencies among all the i-th pixels across the F feature maps *i.e.* for $k = 1, \ldots, F$. It is achieved through the following transform:

$$u_i(k) = \frac{w_i(k)}{\left(\beta_k + \sum_{j=1}^{F} \gamma_{k,j} w_i(j)^2\right)^{\frac{1}{2}}} \tag{3.13}$$

where $u_i(k)$ is the output pixel and $\beta_k,\ \gamma_{k,j} \in \mathbb{R}^+$ are learned parameters. Thus, a GDN transform requires $F^2 + F$ parameters for F feature maps. Similarly, the inverse GDN

(IGDN) is defined as:

$$u_i(k) = w_i(k)\left(\beta_k + \sum_{j=1}^{F} \gamma_{k,j} w_i(j)^2\right)^{\frac{1}{2}}. \tag{3.14}$$

Unlike the non-linear functions presented in Fig. 3.1, the (I)GDN is not a point-wise non linearity. As it mixes different inputs with several learnable parameters, the GDN is able to represent richer functions yielding better rate-distortion performance [12]. It is worth noting that common deep learning techniques such as the batch normalization [53] do not bring any improvement on a GDN-based system [46]. This is due to the normalization effect of the GDN, which is able to center and re-scale the data. For all these reasons, the GDN and its inverse are found in a substantial number of works [47].

In order to perform arithmetic coding or to estimate the rate during the training process, a probability model p of the latent is required. To tackle the dimensionality issue mentioned in Chapter 1, the latent pixels are coded independently:

$$p_{\mathbf{y}}(\mathbf{y}) = \prod_{k,i} p_{y_{k,i}}(y_{k,i}), \tag{3.15}$$

With $y_{k,i}$ the i-th pixel in the k-th feature map of $\mathbf{y}$. Yet, estimating a probability model $p_{y_{k,i}}$ for each latent pixel is not desirable. It would require a significant number of parameters and forbid the processing of variable size inputs (as the dimension of the latent feature maps depends on the input dimension). Ballé et al. [46] propose to learn one single probability model p_{y_k} per feature map:

$$p_{\mathbf{y}}(\mathbf{y}) = \prod_{k,i} p_{y_k}(y_{k,i}). \tag{3.16}$$

Each distribution p_{y_k} is represented by its cumulative c_{y_k}, see Eq. (3.9), parameterized by a fully connected network. In order to represent a cumulative c (indices omitted for clarity), the neural network is designed to fulfil the following constraints:

$$c(-\infty) = 0 \;\; ; \;\; c(\infty) = 1 \;\; ; \;\; \frac{\partial c(x)}{\partial x} \geq 0. \tag{3.17}$$

3.3.4 Training and rate-distortion results

In order to measure their performance, learning-based coding schemes have to undergo a training stage. As such, the rate-distortion results achieved by a given architecture vary significantly according to the training parameters. The appendix B.1 provides additional

information regarding the training parameters. Unless specified otherwise, all the results presented in this manuscrit result from a personal training by the author in order to enable consistent results. The learning-based coding scheme presented in Fig. 3.5 is trained end-to-end, to minimize the rate-distortion cost presented in Eq. (3.12) with the Mean Squared Error (MSE) as the distortion metric. In order obtain rate-distortion curves with $N = 4$ points, N individual systems are learnt under N different rate constraints λ. The quality of the reconstructed image is measured through the PSNR, and the rate is expressed in bits per pixel (bpp), computed as:

$$R_{bpp} = \frac{-\sum_{\text{images}} \log_2 p(\hat{\mathbf{y}})}{\text{Number of pixels}}. \tag{3.18}$$

The learned image coder is evaluated on the CLIC 2020 validation set, against existing image/video coding standards. More details on the evaluation and the anchors are available in the appendix B.2.

Figure 3.6 presents the rate-distortion performance of the learning-based coding scheme with $F = 192$ internal convolution features. The interest of non linear transform is illustrated by training the exact same system without (I)GDN layers. The Bjøntegaard rate delta (BD-rate) [54] accurately compares different systems. It measures the variation of rate required to achieve equivalent quality between two codecs. Compared to a linear autoencoder, the non-linear behaviour brought by the GDN allows to have a BD-rate of -40 %, *i.e.* the rate can be decreased by 40 % while maintaining the same quality (same PSNR). This proves the relevance of using non-linear transforms. Although the non-linear systems outperform JPEG, they underperform compared to HEVC and VVC. The next section details how to improve the system to reach performance competitive with VVC.

3.4 Advanced learned coding schemes

3.4.1 Hyperprior to refine the probability model

According to Eq. (1.5), a mismatch between the latent variable probability model p and its actual distribution q brings a rate overhead of $D_{KL}(q \,||\, p)$ bits. As such, an efficient way of improving the rate-distortion performance of the system is to reduce this mismatch through a better probability model estimation. The current probability model introduced previously presents two main weaknesses. First, it is assumed to be spatially invariant *i.e.*

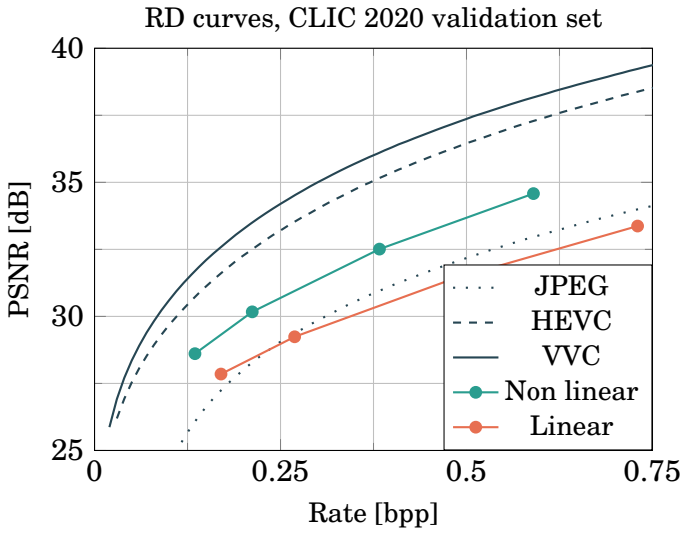


Figure 3.6: Rate-distortion comparison between a GDN-based and a linear system.

the same model p_{y_k} is used, regardless of the pixel index i. Second, the probability model remains identical whatever the input image.

Figure 3.7 presents the same latent feature map for two different input images. It is clear that the statistics (mean value and variance) of the latent variable vary both according to the spatial dimension and the input image.

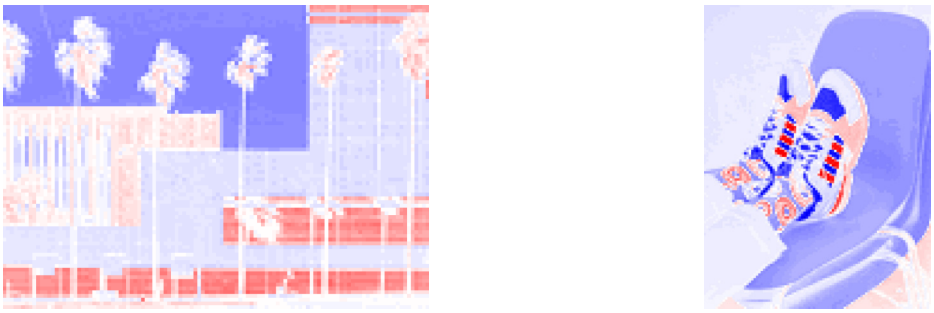


Figure 3.7: The 116$^{\text{th}}$ feature map for two different pictures.

The hyperprior mechanism [46] proposes to condition the probability model p on both the spatial position and the input image. This is achieved by conditioning p on the side information $\hat{\mathbf{z}}$, called the hyperprior. This hyperprior is sent using an additional autoencoder, with an adapted analysis transform h_a and synthesis transform h_s. The hyperprior analysis transform computes $\mathbf{z}$ using the latent variable:

$$\mathbf{z} = h_a(\mathbf{y}). \tag{3.19}$$

The hyperprior $\mathbf{z}$ is then quantized and transmitted using arithmetic coding. Its probability model $p_{\mathbf{z}}$ is a feature-wise cumulative function as explained before for the latent variable $\mathbf{y}$. The hyperprior synthesis transform takes the quantized hyperprior as input to compute $\boldsymbol{\psi}$, a vector of parameters conditioning the latent variable distribution:

$$\boldsymbol{\psi} = h_s(\hat{\mathbf{z}}). \tag{3.20}$$

In Ballé and Minnen's initial work [46], [48], the latent distribution parameters consists of a pixel-wise mean value and standard deviation, $\boldsymbol{\psi} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$. For a latent variable $\mathbf{y} \in \mathbb{R}^{F \times H \times W}$, $\boldsymbol{\psi} \in \mathbb{R}^{2F \times H \times W}$ which correspond to two parameters μ_i and σ_i per latent pixel y_i. Consequently, each latent pixel y_i is modelled with its probability distribution p_{y_i}, with a closed form cumulative distribution function, typically a Gaussian or a Laplace distribution. In this thesis, a Laplace distribution $\mathcal{L}$ is used:

$$p_{y_i} \sim \mathcal{L}\left(\mu_i, \sigma_i\right). \tag{3.21}$$

The hyperprior mechanism makes the latent probability model p_{y_i} vary according to the pixel index i (the spatial dimension). As $\boldsymbol{\psi}$ is computed from the latent variable, it is dependent on the input image and able to match the image statistics. Figure 3.8 shows an implementation of the hyperprior-based system, derived from [46]. Since the hyperprior $\hat{\mathbf{z}}$ is sent as side information, its associated rate has to be taken into account in the loss function. Equation (3.12) becomes:

$$\mathcal{L}_\lambda = \mathbb{E}_{\mathbf{x}}\left[d(\mathbf{x}, \hat{\mathbf{x}}) - \lambda\left(\log_2 p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}}) + \log_2 p_{\tilde{\mathbf{z}}}(\tilde{\mathbf{z}})\right)\right]. \tag{3.22}$$

Hyperprior-based systems (Fig. 3.8) are evaluated against fixed probability systems (Fig. 3.5) with a similar number of internal feature maps $F = 192$. The rate-distortion results are shown in Fig. 3.9. The hyperprior-based systems achieve a BD-rate of -32 % compared to the fixed probability systems, significantly improving the coding performance.

Besides the performance increase, the relevance of the hyperprior can be visually assessed. As each latent pixel is independently coded, the system must strive to make them independent to avoid a significant rate overhead. Figure 3.10b presents one feature map of $\mathbf{y}$, where each latent pixel exhibits a high spatial correlation with its neighbours. The Fig. 3.10c shows the inter pixel correlation captured by $\boldsymbol{\mu}$ by representing $\mathbf{y} - \boldsymbol{\mu}$. Similarly, Fig. 3.10d indicates the structure captured by $\boldsymbol{\sigma}$ through the visualisation of

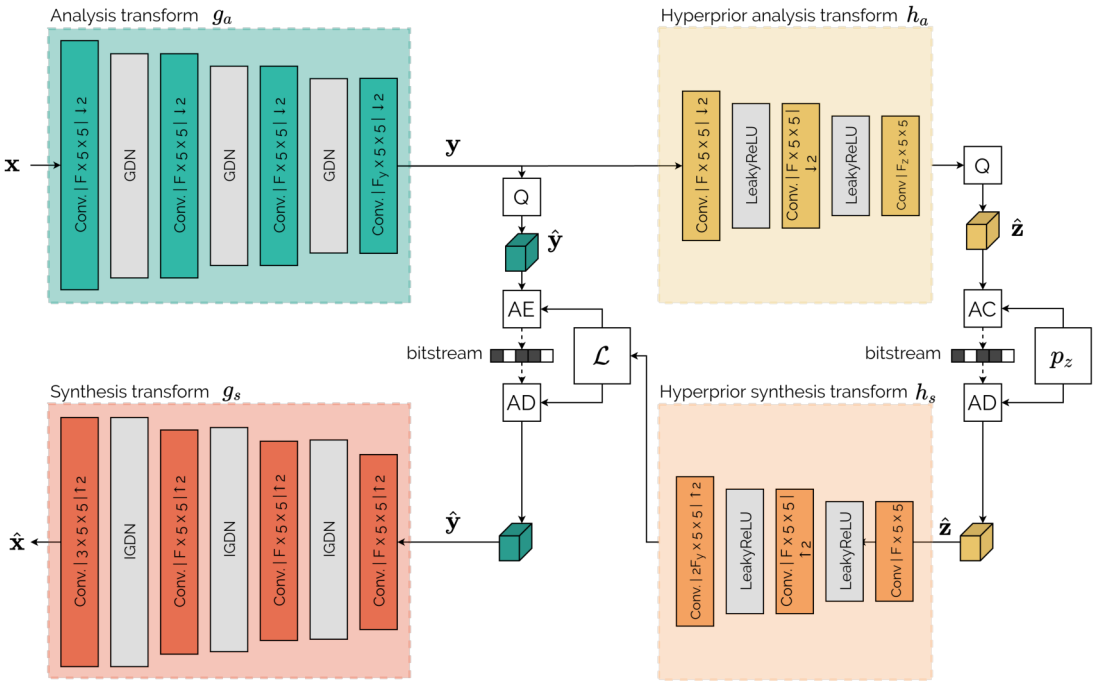


Figure 3.8: Image coding scheme with a hyperprior. The cumulative-based hyperprior probability model is denoted as p_z.

$\frac{\mathbf{y}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}$. This last illustration reveals significantly less spatial correlation than the original feature map, proving the relevance of the hyperprior.

There is an interesting relationship between modelling the latent variable through an adaptive probability model and performing adaptive operations on the signal:

$$y_i \sim \mathcal{L}(\mu_i, \sigma_i) \Leftrightarrow y_i - \mu_i \sim \mathcal{L}(0, \sigma_i). \tag{3.23}$$

This equation gives a meaningful interpretation to the quantities $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. The expectation $\boldsymbol{\mu}$ aims to predict the value of the latent variable $\mathbf{y}$, while the standard deviation $\boldsymbol{\sigma}$ indicates the incertitude of the prediction. This highlights that $\boldsymbol{\mu}$ acts as a prediction of the signal, conditioned on some previously sent values: the hyperprior $\mathbf{z}$. This latent domain prediction is a significant improvement over the naive architecture, presented in Section 3.3.3, which lacks a prediction step.

Unlike the autoencoders previously introduced, the hyperprior autoencoder does not aim to copy its input. Indeed, the hyperprior autoencoder maps the latent variable $\mathbf{y}$ to the hyperprior $\mathbf{z}$ and then to the probability distribution parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. That is, the input of the (hyperprior) analysis h_a and the output of the synthesis h_s do not lie

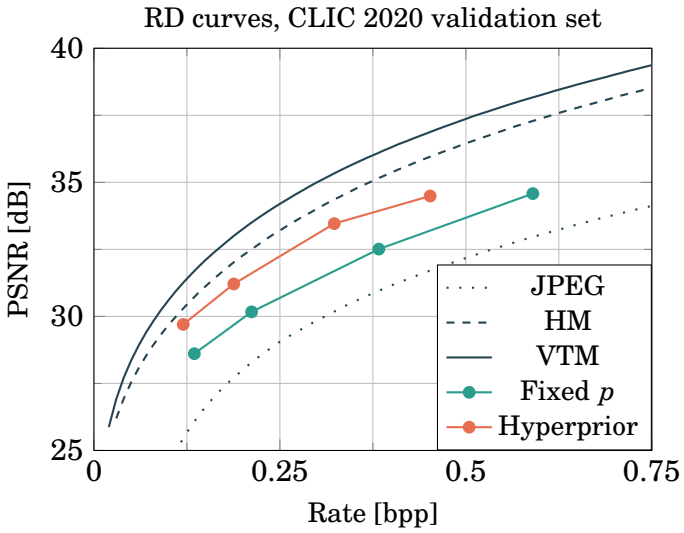


Figure 3.9: Performance of the hyperprior-based coding scheme.

in the same domain. Even though this is not a conventional autoencoder, such analysis-synthesis architecture will be called autoencoder in this manuscript. Nevertheless, this is an important difference with the conventional autoencoders, where the synthesis output aims to replicate the analysis input *e.g.* image-domain input and output. One insightful lesson of the hyperprior mechanism is that the autoencoder architecture is able to simultaneously:

1. Compute any quantity (in this case probability parameters and in future systems, motion information, coding mode selection *etc.*) based on image-domain inputs;

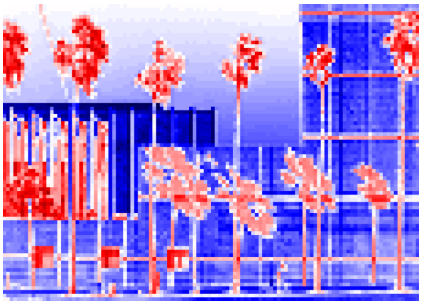
2. Transmit this quantity as a low-entropy latent variable suited for arithmetic coding.

This will be used in all further systems to efficiently estimate and transmit any desired quantity that has to be computed at the encoder and available at the decoder.

3.4.2 Auto-regressive probability model

The hyperprior probability model performance proves the relevance of modelling each latent pixel with its own Gaussian or Laplace distribution. Auto-regressive models (ARM) [48], [49] have been proposed to improve hyperprior-based systems by also conditioning the latent variable distribution on previously transmitted values. Indeed, the i-th quantized latent pixel $\hat{y}_i$ is likely correlated with the previously transmitted pixels $\hat{\mathbf{y}}_{<i}$.

(a) Input image.



(b) One feature map $\mathbf{y}_i$

(c) The centered version $\mathbf{y}_i - \boldsymbol{\mu}_i$

(d) The centered and normalized version $\frac{\mathbf{y}_i - \boldsymbol{\mu}_i}{\boldsymbol{\sigma}_i}$

Figure 3.10: Visualisations of the hyperprior.

Adding an ARM module allows to leverage the already received latent pixels to more accurately model the current pixel distribution parameters $\psi_i = \{\mu_i, \sigma_i\}$. Similarly to the hyperprior, the ARM enables a latent domain prediction of the image, based on the already transmitted values. A principle diagram of a coding scheme featuring hyperprior and ARM mechanisms is shown in Fig. 3.11. The already received latent pixels $\hat{\mathbf{y}}_{<i}$ are fed to the ARM h_r, which extracts relevant information. Finally, information coming from the hyperprior synthesis h_s and the ARM h_r are fused using a fusion module f:

$$\psi_i = f\left(h_s(\hat{\mathbf{z}}), h_r(\hat{\mathbf{y}}_{<i})\right). \tag{3.24}$$

In practice, the ARM is a PixelCNN-like network [55], [56] *i.e.* a convolutional neural network with masked kernels processing only causal pixels. Its detailed architecture is available in appendix B.3. The fusion module is also a PixelCNN-like network, whose architecture is depicted in appendix B.3.

Figure 3.12 presents the rate-distortion gain brought by the addition of an ARM. Compared to hyperprior-based system, using both hyperprior and ARM yields a BD-rate of -7 %. Such results are consistent with the literature [48], [49] where rate gains are

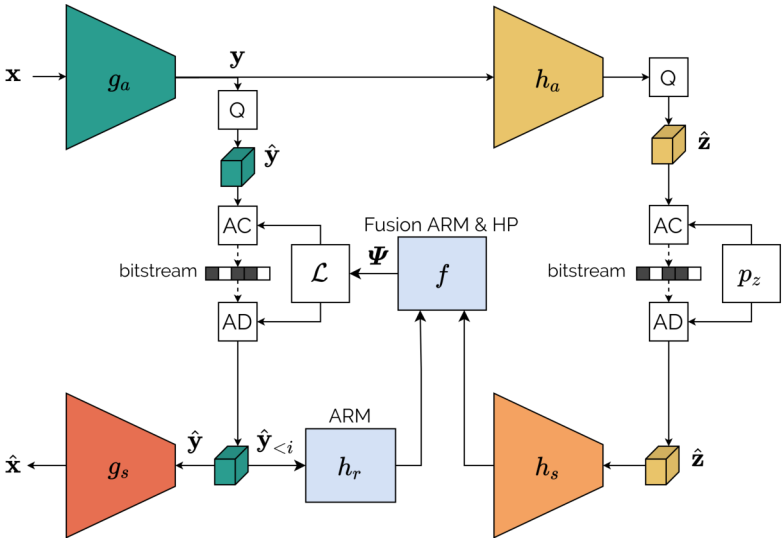


Figure 3.11: Image coding scheme featuring both hyperprior and ARM.

claimed to be around 10 %.

Yet, the ARM makes the entire decoding process sequential, as one latent pixel can not be processed while all the previous one have been decoded. While the hyperprior mechanism allows to obtain the probability parameters $\boldsymbol{\psi}$ all at once, the ARM requires to perform one iteration per (dependent) latent pixels, often leading to a number of iterations in the order of $H \times W$. This results in a prohibitive decoding time [57], even though some approaches aims to decrease the number of dependent pixels [58]. In this work, we consider that the relatively minor performance gain was not worth the significant increase in complexity.

3.4.3 Attention modules for better networks

Moreover recent approaches [44], [59]–[61] propose to improve the rate-distortion performance through more powerful transforms. This is achieved through residual blocks and attention modules inside the different transforms of the coding scheme. More information on these architectures are available in appendix B.4. Residual attention modules and residual blocks can be seamlessly integrated between the convolutional layers, increasing the representational capacity of the transforms at the expense of an increase in complexity and memory footprint. The appendix B.5 presents an attention-based architecture inspired from Cheng *et al*. [44]. Its performance is assessed against the plain hyperprior system, without residual block or attention module. The rate-distortion results are shown in Fig.

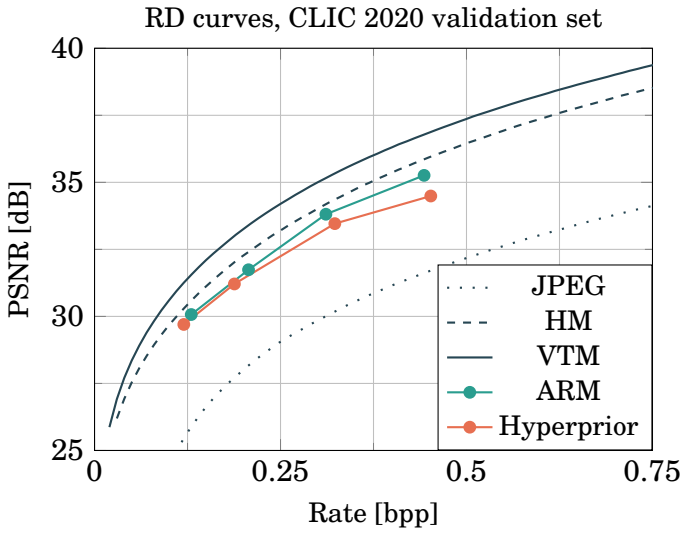


Figure 3.12: Performance brought by adding an ARM to an hyperprior-based system.

3.13. The attention-based system offers a BD-rate of −31 %, allowing to significantly outperform HEVC. This highlights the relevance of using more advanced architectures.

It is interesting to note that even such rich architectures exhibit slightly worse performance at higher rates. This is a known problem in the literature, which may be due to the intrinsic nature of the operations performed within the coding pipeline. Indeed, the transforms g_a and g_s are not lossless. Constraining the operations to be invertible (*i.e.* $g_s = g_a^{-1}$), have been shown to offer competitive high-rate performance [62] .

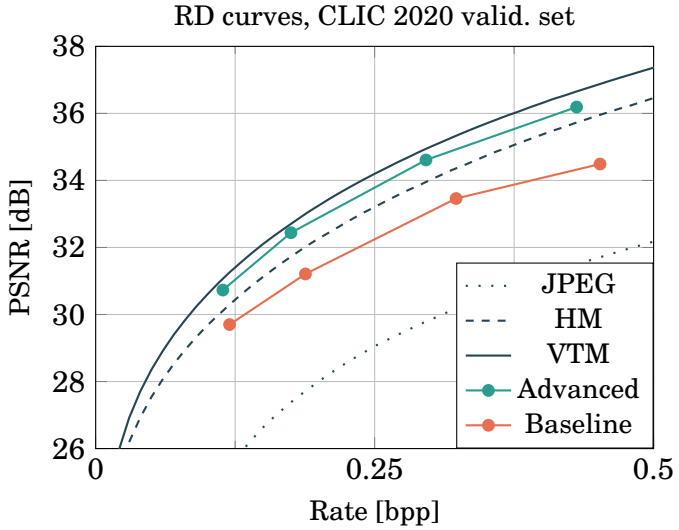


Figure 3.13: Baseline architecture vs. advanced one (see appendix B.5).

3.4.4 Binary Probability Model

The successive introduction of the hyperprior and the ARM indicates that a better probability model for the latent variable is an important source of performance improvement. Following this trend, some recent work propose to add more parameters to the probability model, allowing to more accurately represent the latent distribution. Usually, this is achieved by using mixtures of Gaussian distributions [63], [64]. One of the early contribution of this thesis is to propose the binary probability model (BPM): an alternative means of refining the latent variable probability model [14].

So far, hyperprior-based systems model each latent pixel with a Laplace distribution. As such, the centered latent pixel $\bar{y}_i = y_i - \mu_i$ follows:

$$y_i \sim \mathcal{L}\left(\mu_i, \sigma_i\right) \Leftrightarrow \bar{y}_i \sim \mathcal{L}(0, \sigma_i) \tag{3.25}$$

As the most frequent values of $\bar{y}_i$ tend to be around zero, it is particularly important to precisely model the centered latent pixel distribution in this range to avoid a significant rate overhead. Following this intuition, the binary probability model (BPM) features additional parameters to explicitly model the probability of $\bar{y}_i$ being quantized to the value -1, 0 or 1. This allows to achieve a more accurate probability model.

The BPM is implemented by modifying the signalling process of each latent variable. Instead of simply being conveyed at once through arithmetic coding based on a Laplace distribution, the latent variable is decomposed into different quantities. First, each latent variable is centered and quantized. For sake of clarity, the latent variable index is omitted and y stands for any y_i. The value to convey is:

$$\hat{y} = Q\left(\bar{y}\right), \text{ with } \bar{y} = y - \mu. \tag{3.26}$$

The signalling process of one latent variable is shown in Fig. 3.14. Since $\bar{y}$ is assumed centered, it focuses on $|\hat{y}|$ *i.e.* amplitude of $\hat{y}$ while the sign is conveyed has an additional bit. The signalization of the amplitude is inspired by the binarization scheme of the CABAC. That is, $|\hat{y}|$ is decomposed into two binary flags:

$$G_0 = |\hat{y}| > 0 \text{ and } G_1 = |\hat{y}| > 1. \tag{3.27}$$

The flag G_0 indicates whether $\hat{y}$ is greater than 0. Similarly the flag G_1 indicates whether it is greater than one. These two binary flags are transmitted using arithmetic

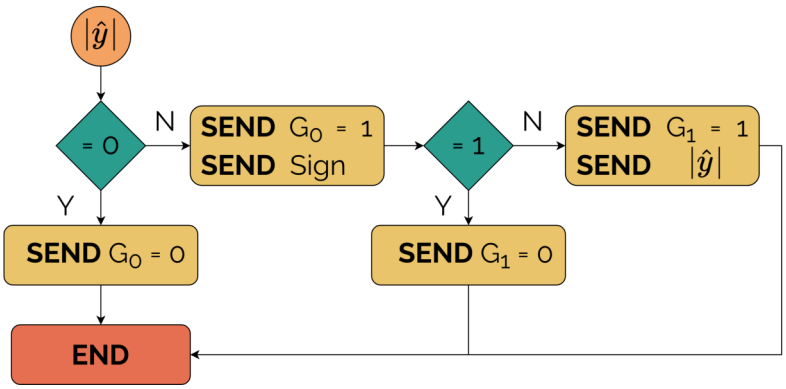


Figure 3.14: Signalling process of a centered latent variable $\hat{y}$.

coding, which requires to model their respective probabilities with two parameters P_0 and P_1. Finally, if the amplitude is greater than one *i.e.* if $|\hat{y}| = k > 1$, then the value k is explicitly conveyed. In this case, the probability used for the arithmetic coding of k is:

$$p(k) = p(|\hat{y}| = k \mid |\hat{y}| > 1) = \frac{2\int_{k-0.5}^{k+0.5} p_{\bar{y}}(a)\,\mathrm{d}a}{\int_{-1.5}^{1.5} p_{\bar{y}}(a)\,\mathrm{d}a}, \tag{3.28}$$

The probability density function of the centered latent variable $p_{\bar{y}}$, is modelled with a centered Laplace distribution *i.e.* $p_{\bar{y}} \sim \mathcal{L}(0,\sigma)$. The BPM can thus be thought of as an enhancement of the Laplace model, as it introduces two additional parameters (P_0 and P_1) to more accurately model the latent distribution, allowing to decrease the rate. The BPM is implemented through a single modification of hyperprior-based systems, such as those presented in Fig. 3.8 or in appendix B.5. The two additional parameters P_0 and P_1 are additionally decoded from the hyperprior $\hat{\mathbf{z}}$.

The interest of the BPM is assessed by implementing the architecture presented in appendix B.5 for the Laplace distribution or the BPM. The rate-distortion results are presented in Fig. 3.15. Compared to the Laplace model, the BPM offers a rate reduction of -6 %, yielding an image coding scheme competitive with VVC. Since the overall architecture remains the same, this performance gain comes with virtually no increase in complexity. This performance enhancement was published through a conference paper presented at the IEEE *International Conference on Acoustics, Speech and Signal Processing* (ICASSP) in 2020 [14].

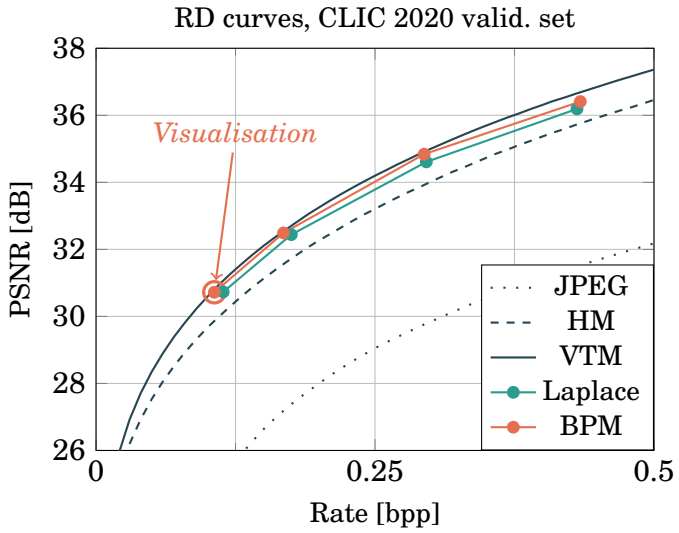


Figure 3.15: Laplace distribution vs. binary probability model (BPM).

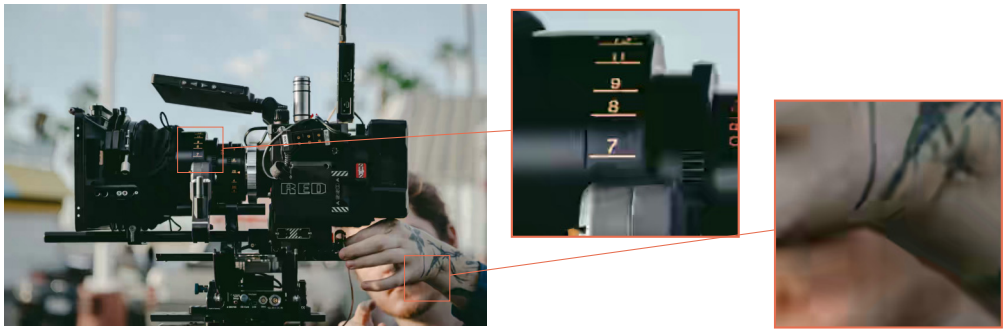
3.5 Visualisations

Figure 3.16 presents a visual example of a learned image coder. Detailed visual comparisons for different images and rate constraints are available in Appendix B.6. The original image, shown in Fig. 3.16a is fed to the coder analysis transform, which computes a latent variable sent to the decoder. The spatial distribution of the latent variable rate is illustrated in Fig. 3.16d. As expected, the high-frequency areas (edges, text) require more rate to be transmitted, while low-frequency area (the blurry background) costs very little rate. Finally, the synthesis transform uses the latent variable to generate the compressed image presented in Fig. 3.16b.

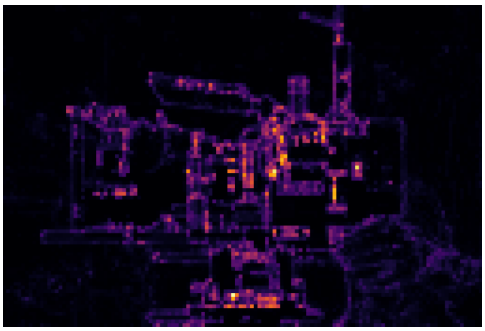
In order to respect a tight rate constraint, the learned coder discards some information. It mostly concerns the details, as it is measured as less harmful according to the PSNR (the training quality metric). For instance, the heavy grain of the original image presents a heavy grain (visible on the camera) is removed by the compression process to reduce the rate. Similar visual degradations are observed on the hands, where most of the details are lost during the compression. For comparison, the visual result obtained by coding the image with the VTM at a similar rate is presented in Fig. 3.16c, allowing to note some difference concerning the nature of the visual degradations. Due to its convolution operations, the learned coder tends to lose most of the sharpness, resulting in smooth images. On the other hand, the VTM performs a block-based processing which causes more discontinuities in the compressed image.

(a) Original image $\mathbf{x}$.

(b) Reconstructed image $\hat{\mathbf{x}}$. PSNR = 33.47 dB, Rate = 0.050 bpp.



(c) Image compressed by the VTM. PSNR = 34.08 dB, Rate = 0.053 bpp.



(d) Spatial distribution of the rate.

Figure 3.16: Visual comparison between a learned coder and the VTM. The test image is *400984b87394ada6d9627ed918908986* from the CLIC20 dataset.

3.6 A word on complexity

Most of the improvements introduced in this chapter come at the cost of additional parameters, increasing both the memory footprint and the complexity. Yet , the number of parameters of a given design can be tuned by modifying the number of internal convolutional kernels. This enables to modulate the complexity without dramatically altering the overall architecture. Applying this technique to all the architecture allows to obtain performance-complexity curves, that exhibits whether a design is intrinsically better at similar complexity (*e.g.* whether hyperprior models are better than fixed-pdf models) or if this is only due to the presence of additional parameters.

Although the number of parameters is an important factor of the overall complexity, it is not a satisfactory measurement. For instance, a single convolutional layer with lots of input and output feature maps can have the same number of parameters as many small successive layers. On appropriate computing devices, it is possible to efficiently parallelize the big convolutional layer while the successive small ones are inherently sequential, leading to a more important execution time. The encoding and decoding time could be a relevant measurement of the complexity. Yet, it omits an important dimension of learning-based systems: the training stage. Ideally, one would like to compare systems by measuring their performance and execution time once the training stage has properly converged. In practice, some systems still improves after weeks of training, resulting in impractical convergence time.

Consequently, the complexity is measured through the training time. This somehow reflects the execution time, as one iteration and thus the whole training will be shorter if the execution time of the architecture is small. The performance of an architecture is computed by averaging the rate-distortion cost $\overline{J}$ for different rate-constraint λ_i:

$$\overline{J} = \frac{1}{N} \sum_{i=1}^{N} D + \lambda_i R. \tag{3.29}$$

Figure 3.17 presents loss-training time curves for all the proposed designs, except the autoregressive-based ones which has already be discarded due to complexity issue. This graph shows that even at equivalent complexity (equivalent training time), more advanced approaches (hyperprior, attention, BPM) are consistently better. This proves that the successive improvements presented in this Chapter perform better due to their intrinsic quality, and not because they feature more parameters and more training time.

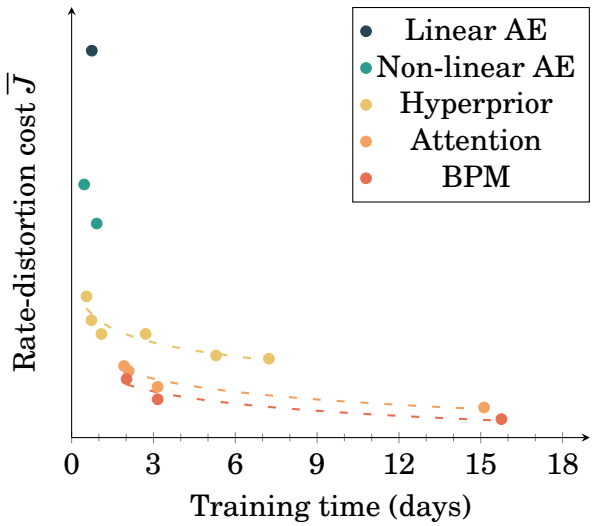


Figure 3.17: Average rate-distortion cost (lower is better) as a function of the training time for different architectures. Architectures are detailed in Table 3.1.

3.7 Conclusion

In this chapter, the main concepts of learned compression have been introduced. The most common techniques of the literature (hyperprior, ARM, attention mechanism) as well as one contribution of this thesis (the binary probability model) have been evaluated to assess their coding performance. Table 3.1 summarises the performance gains brought by the successive improvements and highlights two mains levers of performance improvement. The first one is the enhancement the analysis and synthesis transforms through non-linearities and attention modules. The second is the refinement of the latent probability model using hyperprior, ARM and the binary probability model. The resulting learned coding scheme achieves a BD-rate of −55.6 % compared to the initial non-linear autoencoder *i.e.* it requires 55.6 % less rate to obtain an equivalent quality.

End-to-end training of a coding scheme implementing all these components allows to reach image coding performance competitive with VVC, achieving a BD-rate of +1.4 % when compared to VVC. These are compelling results, as VVC represent the state-of-the-art of the standardized compression algorithms, which uses a conventional approach. Furthermore, learned approaches are still quickly improving. In a span of a few years, their performance has gone from JPEG to VVC. Yet, the performance of learned systems comes at the expense of an important complexity and memory footprint. For instance, the best performing approaches presented in this chapter require 20 million parameters to

Table 3.1: BD-rates of the different coding schemes. AE stands for autoencoder, ARM for autoregressive module and BPM for binary probability model.

Codec	Non-linear	HP	ARM	Attention	BPM	BD-rate	
Linear AE						+63.7 %	+285.9 %
Non-linear AE	✓					Reference	+132.5 %
Hyperprior	✓	✓				−32.1 %	+56.7 %
ARM	✓	✓	✓			−36.7 %	+45.0 %
Attention	✓	✓		✓		−52.9 %	+7.4 %
BPM	✓	✓		✓	✓	−55.6 %	+1.4 %
JPEG						+46.2 %	+236.6 %
HEVC						−44.8 %	+28.1 %
VVC						−56.9 %	Reference

achieve performance on par with VVC. So far, such convolutional networks still requires high-end computing devices to achieve realistic encoding/decoding time.

Finally, the different designs presented in this chapter turn out to be generic architecture that can be applied to efficiently transmit any two-dimensional signal. As such, these architectures are the building blocks of the learned video schemes designed in the next chapters of this manuscript.

PART II

Learned Video Coding

CHAPTER 4

FROM LEARNED IMAGE CODING TO LEARNED VIDEO CODING

4.1 Introduction

THE presence of the temporal dimension makes video coding a more challenging task than still image coding, due to the supplementary statistical redundancies. On one hand, the well-established traditional video coders remove these redundancies using a temporal prediction performed through a motion compensation process. On the other hand, learned video coding is a brand new field allowing to reconsider some historical design choices of traditional coders. Since the initial work published in 2019, different strategies of temporal redundancies removal have been considered in the literature.

The first part of this chapter consists of a brief review of the literature to motivate the usage of motion compensation to remove temporal redundancies. Consequently, the learned approaches proposed in this thesis are composed of three successive stages: estimating and transmitting the motion information, computing a temporal prediction based on the motion information and finally conveying only the unpredicted part. These three stages are implemented by neural networks. In order to carefully compare among different possible network architectures, strict experimental conditions are used. The second part of the chapter presents these experimental conditions, implemented to evaluate the different proposed systems.

4.2 Temporal dependencies reduction

4.2.1 Need of an explicit motion compensation

Let us model a video sequence $\mathcal{V}$ as set of T frames: $\mathcal{V} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$. Each video frame $\mathbf{x}_t$ is represented as a tensor of dimension $3 \times H \times W$. Early work from 2019 by Habibian

et al. [15] proposes to straightforwardly extend the learned image coding schemes to video coding. That is, all the T video frames are fed to the analysis transform g_a which computes a single latent variable $\mathbf{y}$ representing all the stacked frames. The latent variable undergoes a quantization step Q and it is sent to the decoder using arithmetic coding. In the end, the synthesis transform g_s reconstructs the T frames in its entirety:

$$\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_T = g_s(\hat{\mathbf{y}}), \text{ with } \hat{\mathbf{y}} = Q\left(g_a(\mathbf{x}_1, \ldots, \mathbf{x}_T)\right). \tag{4.1}$$

In practice, the analysis and synthesis transforms are implemented using 3D convolutions [65]. We recall that the usual 2D convolutions are fed with $F \times H \times W$ input data, that is F two-dimensional feature maps of size $H \times W$. 2D convolutions use several $F \times k_H \times k_W$ kernels to compute the output feature maps. On the other hand, 3D convolutions process $F \times T \times H \times W$ inputs, *i.e.* F three-dimensional feature maps of size $T \times H \times W$, with T the number of frames. They use several $F \times k_T \times k_H \times k_W$ kernels to compute the output feature maps. The additional dimension of their kernel, makes 3D convolutions able to process the supplementary temporal dimension.

3D convolutions present two main weaknesses. First, compared to the successive processing of T frames with 2D convolutions, 3D convolutions require k_T times more parameters and multiplications, with k_T often greater than 3. Second, the processing of T frames together prevent to tune the coding structure according to the application needs. For instance, the coding latency is by design T frames. This can be troublesome in some context (*e.g.* in video-conferencing setup), where the latency has to be as low as possible. In such case, video coders resorts to *low-delay* coding configurations, featuring a coding latency of a single frame. These two limitations are particularly concerning as the 3D convolution-based approaches have not been shown to bring compelling performance. Consequently, this thesis follows most of the publications from the literature and discards the 3D convolution-based approaches in favour of 2D convolution-based ones, which process each frame separately. This reduces the complexity and gives more flexibility on the coding configuration.

Yet, naively processing each frame separately leads to a rate overhead. As motivated in the first chapter (see Section 1.2.4), a prediction step is introduced to lower the mutual information between the frames, reducing the rate overhead. In practice, it is achieved through a motion compensation process, similarly to traditional coders.

4.2.2 inter frame coding with neural networks

The introduction of a motion compensation process leads to the decomposition of the learned coding schemes in three main steps, presented in Fig. 4.1. The first step is to estimate and efficiently convey the motion between the frame to code and its references. Then, the motion compensation takes place to compute a temporal prediction through the application of the motion information to the reference frames. Lastly, only the remaining *unpredicted* part of the frame needs transmission. This unpredicted part exists for two reasons: either the content is simply unpredictable (a new object appearing on the image for instance), or the motion compensation step is not accurate enough to achieve a satisfactory temporal prediction.

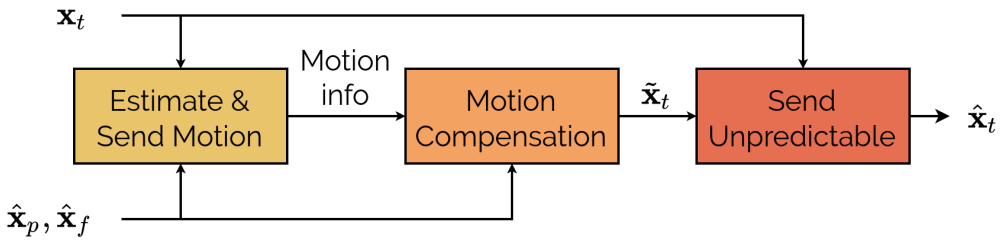


Figure 4.1: Conceptual processing pipeline of an inter frame $\mathbf{x}_t$ relying on two reference frames $\hat{\mathbf{x}}_p$,$\hat{\mathbf{x}}_f$ to compute a temporal prediction $\tilde{\mathbf{x}}_t$.

As presented in the learned image coding chapter, autoencoders are generic tools to efficiently transmit a quantity from the encoder to the decoder. Thus, they are used to transmit the motion information as well as the unpredicted part. The other steps are implemented in several ways in the literature. Table 4.1 summarises the main differences among existing learned video coders. The *motion estimation* column indicates how the motion information is obtained. The *Bi-pred* column details whether a bi-directional prediction is available. If not, the coders only relies on P-frames, leading to reduced compression efficiency. Then, the *Image residual* entry details how the non-predictable information is computed and sent to the decoder. The last column indicates whether the same coder (*i.e.* the same network with the same parameters) is used for intra and inter frames, or if a dedicated intra coder is required.

Information presented in Table 4.1 are discussed in the next two sections. The objective is to identify the limitations of existing learned video coding schemes in order to design a better learned coder.

Table 4.1: Summary of the different learned video coders in the literature. AE stands for autoencoder, Bi-pred for bi-directional prediction.

	Motion		**Image**	**Same intra**
	Estimation	**Bi-pred.**	**residual**	**and inter coder**
Habibian, ICCV 19 [15]	3D convolutions			
Djelouah, ICCV 19 [66]	PWCNet	✓	latent residual	✓
Lu, CVPR 19 [19]	SpyNet		✓	
Yilmaz, ICIP 20 [67]	SpyNet	✓	✓	
Hu, ECCV 20 [20]	SpyNet		✓	
Lu, ECCV 20 [68]	SpyNet		✓	
Yang, CVPR 20 [69]	SpyNet	✓	✓	
Golinski, ACCV 20 [23]	AE†		✓*	
Liu, AAAI 20 [24]	AE†		✓	
Agustsson, CVPR 20 [25]	AE		✓	
Yang, STSP 21 [21]	SpyNet		✓	
Hu, 21 [70]	?		latent residual	

**Not explicitely computed at the encoder-side, but the decoder features residue + prediction.*
†*Requires an explicit loss term to converge.*

4.2.3 Motion estimation, transmission and compensation

Most learned video coders from the literature use a single reference frame, restraining their performance. In order to obtain a more accurate prediction, bi-directional prediction is preferred:

$$\tilde{\mathbf{x}}_t = \beta w(\hat{\mathbf{x}}_p; \mathbf{v}_p) + (1-\beta) w(\hat{\mathbf{x}}_f; \mathbf{v}_f), \text{ with } \beta \in [0,1]. \tag{4.2}$$

Here, w denotes the warping function used to apply a motion $\mathbf{v}$ to a reference frame $\hat{\mathbf{x}}$. The two intermediate warpings are combined through a sum weighted by β, the bi-directional prediction weighting. For the vast majority of works as well as for the work presented in this thesis, w is a linear warping *i.e.* pixels of the motion compensated image are sampled from a linear interpolation of the original image.

In order to compute a relevant prediction, accurate motion information must be available at the decoder. In most existing schemes, a two-step process takes place.

1. Motion is estimated at the encoder using a pre-trained network such as SpyNet [22] or PWCNet [71];

2. An autoencoder is used to convey the motion information to the decoder.

An other approach is to straightforwardly estimate and transmit the motion at once with a single autoencoder. This last method offers the benefit of a simpler design and will be chosen in this manuscript. Once the motion information is available at the decoder, it must be applied to the reference frames through the motion compensation function. This is usually achieved with a bilinear warping. Chapter 6 details how neural networks are used to estimate, transmit and apply the motion information.

4.2.4 Transmitting the unpredicted part

Once a temporal prediction $\tilde{\mathbf{x}}_t$ is available, only the part of $\mathbf{x}_t$ missing from $\tilde{\mathbf{x}}_t$ is conveyed. To this end, $\tilde{\mathbf{x}}_t$ is combined with the frame to code $\mathbf{x}_t$ through a function m. This function aims to remove the predictable part from $\mathbf{x}_t$, to avoid conveying redundant information. To this end, traditional video coders, as well as most learned ones rely on residual coding. That is, m is a simple subtraction in the spatial (image) domain:

$$m(\mathbf{x}_t, \tilde{\mathbf{x}}_t) = \overbrace{\mathbf{x}_t - \tilde{\mathbf{x}}_t}^{\text{Residual}}. \tag{4.3}$$

Then, the decoder recovers $\hat{\mathbf{x}}_t$ as a sum between the prediction $\tilde{\mathbf{x}}_t$ and the transmitted residual. We will argue in Chapter 5 that performing a simple difference between the frame to code and its prediction can be improved from a more advanced combination of these two quantities. Indeed, the representational power of neural networks will be leveraged to design different mixing operations m. Different designs of m are compared, to determine the most effective way of leveraging a temporal prediction.

4.3 Experimental conditions

4.3.1 CLIC 21 video track

The next two chapters carefully consider the design of the motion estimation, transmission and compensation as well as the more effective means of leveraging the temporal prediction (*i.e.* improving the legacy residual coding). In order to accurately assess the different designs introduced, a real-life video coding task is used: the video track of the *Challenge on Learned Image Compression* (CLIC) 2021 [16]. The objective of this challenge is to compress 100 videos at an average rate of 1 Mbit/s, while getting the highest MS-SSIM. All videos have 60 frames with a resolution of 720 lines with 30 frames per second.

4.3.2 Anchors

For the next two chapters, the different design choices are assessed recent modern video coders: HEVC and VVC. This decision is motivated by their performance and their flexibility. So far, the vast majority of the learned systems in the literature are assessed against a fast encoder implementation of HEVC of HEVC *e.g.* x265 using the very fast preset. Consequently, the performance achieved by the best implementations of modern video coders (such as the HM for HEVC or VTM for VVC) remains an upper bound for learned approaches [72].

Then, most learned coding schemes tend to implement restrained coding configurations. Indeed, they are often still limited to P-frames and target higher rates [21], [70], with a frequent intra period. This makes difficult to achieve proper comparison between the approaches presented in this manuscript, which targets *real-life* video coding and the literature.

4.3.3 Coding configuration

The coding structure is of primary importance to obtain the best rate-distortion trade-off. The bi-directional prediction, motivated in section 2.3.1, offers a more accurate prediction which usually results in better coding efficiency. Since, the bi-directional prediction is only available for B-frames, the coding structure must feature as many B-frames as as possible.

For fair comparisons between the proposed learned coders and existing coders, the coding structure from the VVC Common Test Conditions [17] are selected. As such, the Intra Period is set to 32 frames (approximately one intra frame per second at 30 fps). Since there is no constraint on the latency, the best performing coding configuration is chosen. This configuration is called the random access configuration and is presented in Fig. 4.2 (with a GOP size of 8). It presents B-frames wherever it is possible, leading to better compression results. To further maximise the number of B-frames, a GOP size of 32 is used.

4.3.4 End-to-end training

Random access coding features I-frames, P-frames and B-frames. As such, one need to process a frame with zero, one or two reference frames with the same set of parameters. Indeed, at first sight, it is not desirable to have a dedicated intra coder as most existing systems do in the literature. To this end, the training stage has to prepare the coder

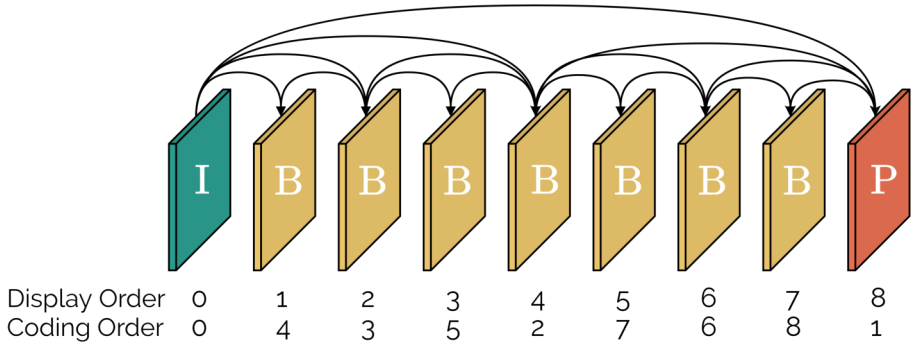


Figure 4.2: Random Access coding configuration, with a GOP size of 8.

to process these different types of frame. Consequently, the coding configuration for the training is set to the smallest configuration featuring I, P and B-frames: a random access configuration with a GOP size of 2 as presented in Fig. 4.3. One training example is therefore composed of a triplet of frames.

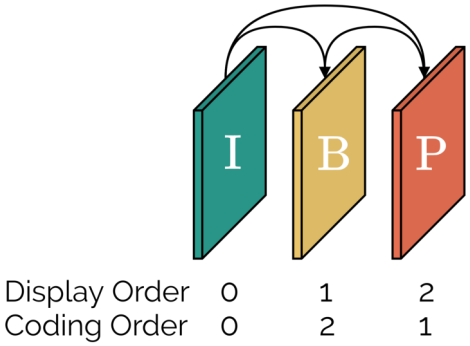


Figure 4.3: The training coding configuration.

In order to achieve the highest coding performance, the different components of the coder are jointly optimized according to the actual rate-distortion objective. To this end, all the frames within a GOP are successively coded and the following loss is minimized by the learning algorithm:

$$\mathcal{L}_\lambda = \mathbb{E}_{\mathbf{x}} \left[\sum_t d(\mathbf{x}_t, \hat{\mathbf{x}}_t) + \lambda r(\hat{\mathbf{x}}_t) \right]. \tag{4.4}$$

To comply with the experimental conditions of the CLIC video track, the distortion metric d is based on the MS-SSIM. The rate metric r measures the rate required to reconstruct the frame $\hat{\mathbf{x}}_t$ at the decoder and the rate constraint λ sets the rate-distortion trade-off.

Framing the optimization task at a GOP level by minimizing the loss function on a whole GOP allows to more accurately represent the propagation of the compression noise

among frames. This enables the model to learn the impact of having compressed (*i.e.* noisy) reference frames, as well as preparing the system to work with compressed references. In the end, it should result in better performance.

Following the common data format in video compression, each frame is represented in YUV 420. For convenience, all three channels are converted to the same spatial resolution. Several methods are explored in the literature [73]. Unlike [73], we choose the simple yet efficient nearest neighbour upsampling of the U and V channels. Although this method has a slightly superior computational cost, it is more convenient as it allows to align the spatial dimensions of U and V with those of the Y channel. As a result, each frame is presented as a tensor of dimension $3 \times H \times W$, that is three channels with a spatial dimension of $H \times W$ pixels. During training, the frames are crops of size 256×256. The appendix C details the composition of the training dataset.

4.4 Conclusion

In this introductory chapter, a thorough review of the learned video coding literature has been carried out. Based on this literature review, a coding strategy has been chosen to comply with requirements regarding the complexity, the performance and the flexibility of the coder. The most significant design choice is to perform a frame-by-frame processing, which in turns implies a motion compensation step. As such, the processing of a single frame is split into three successive steps: motion estimation and transmission, motion compensation and sending the remaining unpredicted part.

Throughout the literature review, it has been noted that most existing learned coding schemes perform these steps using usual tools such as residual coding or pre-trained motion estimation networks. The two next chapters reconsider these design choices, in the light of the possibilities offered by neural networks. All the neural-based designs presented in the following chapters will have to be accurately evaluated. To this end, a real-life coding task is selected: the CLIC21 video track. In order to ensure the relevance of the learned coding schemes, state-of-the-art anchors are proposed. To this day, traditional approaches (HEVC, VVC) remains the best performing systems. For fair comparison with these anchors, the random access coding configuration of the CTC [17] is chosen. Finally, a training process is devised to prepare the systems for the target coding configuration.

CHAPTER 5

EXPLOITATION OF A PREDICTION

5.1 Introduction

VIDEO coders rely on temporal prediction to identify the statistical relationship within the data. Once the prediction of a frame is available, it is subtracted to the current frame to code to remove some of the redundancies. This process is called residual coding and it is implemented by most traditional and learned coders. Yet, neural networks are able to learn how to exploit the prediction beyond a simple difference.

This chapter compares different mixing functions between the frame to code and its prediction. Existing approaches from the literature are compared and a novel architecture called conditional coding is proposed to enhance the compression efficiency. Finally, we propose to condition operations in the coding scheme based on the type of the current frame (I, P or B), increasing the rate-distortion performance.

5.2 Baselines and experimental conditions

5.2.1 Naive prediction

This chapter aims to investigate the most effective means to exploit a temporal prediction. That is, we are interested in the design of the last step presented in Fig. 4.1 which consists in transmitting the unpredicted part of the frame. In this context, let us suppose that a bi-directional prediction process is used to obtain $\tilde{\mathbf{x}}_t$, a temporal prediction of $\mathbf{x}_t$:

$$\tilde{\mathbf{x}}_t = \beta w(\hat{\mathbf{x}}_p; \mathbf{v}_p) + (1-\beta) w(\hat{\mathbf{x}}_f; \mathbf{v}_f), \text{ with } \beta \in [0,1]. \tag{5.1}$$

The function w applies the motion information $\mathbf{v}_p$ (resp. $\mathbf{v}_f$) to the reference frame $\hat{\mathbf{x}}_p$ (resp. $\hat{\mathbf{x}}_f$) and β is the bi-directional prediction weighting. For fair comparison between different methods, the temporal prediction must remain identical. To this effect, a naive

prediction without any motion information is used. That is, the prediction is simply a combination of the raw reference frames:

$$\tilde{\mathbf{x}}_t = \beta\hat{\mathbf{x}}_p + (1-\beta)\hat{\mathbf{x}}_f = \begin{cases} \frac{1}{2}\left(\hat{\mathbf{x}}_p + \hat{\mathbf{x}}_f\right) & \text{for B-frames, } \left(\beta = \frac{1}{2}\right) \\ \hat{\mathbf{x}}_p & \text{for P-frames, } \left(\beta = 1\right) \end{cases}. \tag{5.2}$$

For I-frames, $\tilde{\mathbf{x}}_t = 0$ since they do not have any reference frame available.

5.2.2 Learned and traditional baselines

As presented in Table 4.1, residual coding in the image (*i.e.* spatial) domain is the usual method to exploit the temporal prediction. This is also the case in all the ITU/MPEG video coding standards. Figure 5.1 shows the autoencoder-based residual coding scheme. It first mixes the current frame and its prediction through a simple subtraction. Then, only the prediction error $\mathbf{x}_t - \tilde{\mathbf{x}}_t$ is sent. Its architecture is based on the components known to perform well for learned image coders (hyperprior, attention modules), with $F = 192$ internal features. Their exact implementation are available in appendix D.1. To better appreciate the performance of the learned residual coder, a pure image coder is also implemented. It relies on exactly the same architecture than the residual coder, except that it transmits the frame $\mathbf{x}_t$ without reference.

Different configurations and implementations of traditional coding algorithms are used as anchors to verify the performance of the learned approaches. First, the All Intra (image coding only) configuration of HEVC is tested through the HM (HEVC test Model) 16.22. This serves as a lower bound for the required performances of learned approaches. Indeed, even the learned image coding scheme should outperform HEVC All Intra (as in Chapter 3). Conversely, we used a fully fledged configuration of HEVC, *with* an actual motion compensation, to represent the performance of a modern video coder. This is achieved using the HM 16.22 and x265 (medium preset). While the former is arguably the best implementation available, the latter is very often used in the learned compression literature [19]–[21].

5.2.3 Training and testing the baselines

For the learning-based residual and image coding scheme, the training process follows the principles described in the learned image coding chapter. Namely, the quantization is approximated by an additive noise and the rate is measured through the entropy of

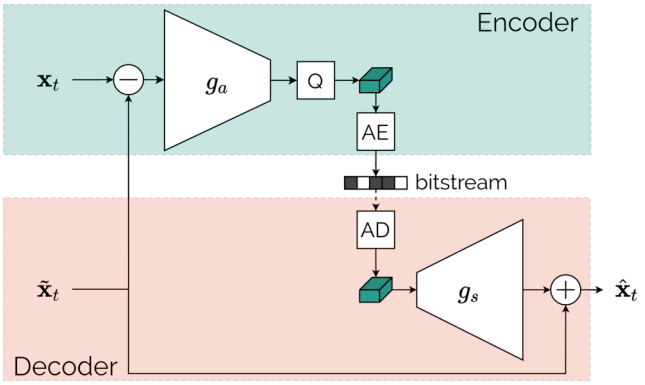


Figure 5.1: Autoencoder-based residual coding scheme. The latent variable probability model is omitted, Q stands for quantization, AE and AD for arithmetic encoding and decoding.

the latent variables. As presented in section 4.3.4, each training example is a GOP of size 2 composed of one I-frame, one P-frame and one B-frame and the training process aims to minimize the rate-distortion cost based on the MS-SSIM. For each coding scheme, N different rate constraints λ are used to learn N systems operating at different rate targets. At the beginning of the training, the compressed frames are not relevant enough to be used as references. Relying on them would result in an irrelevant prediction that would not allow the optimization algorithm to converge. This is avoided through the usage of non-compressed reference frames during the first 20 training epochs out of 50.

After training, the systems are evaluated on the CLIC21 validation set, using a Random Access coding structure with a GOP size of 32 and an intra period of 32. The rate-distortion results are shown in Fig. 5.2. The learned image coder outperforms the HM in All Intra configuration. This is coherent with the image coding result presented in Chapter 3. Since the residual coder has access to a (naive) temporal prediction, it logically outperforms the image coder. The rest of the chapter aims to investigate ways of better leveraging the prediction, without changing the way it is computed, defined in Eq. (5.2).

5.3 Conditional coding

5.3.1 Motivations

Let us generalize the task of inter frame coding. Information from an encoder-side quantity $\mathbf{x}_{enc}$ is sent, while a variable $\mathbf{x}_{dec}$ containing relevant information is already available at

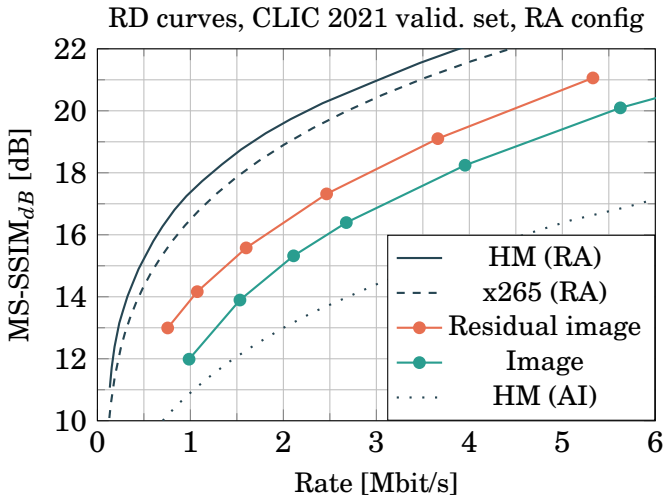


Figure 5.2: Different baselines comparison. MS-SSIM$_{dB} = -10\log_{10}(1 - \text{MS-SSIM})$ is the quality (higher is better). RA stands for Random Access, AI for All Intra.

the decoder-side (*e.g.* a prediction or a reference frame). An ideal inter frame coding scheme is able to leverage all the decoder information. As such, it avoids conveying redundant information *i.e.* information from $\mathbf{x}_{enc}$ which is already available at the decoder, in $\mathbf{x}_{dec}$. The main idea of residual coding is that the decoder information can be removed from $\mathbf{x}_{enc}$ by simply subtracting $\mathbf{x}_{dec}$ to it. The results of this subtraction is called the residue. It is conveyed by an autoencoder, through the usual analysis-quantization-synthesis pipeline. In the end, $\mathbf{x}_{dec}$ is added back to obtain the output $\mathbf{x}_{out}$, as in Fig. 5.1:

$$\mathbf{x}_{out} = \mathbf{x}_{dec} + \overbrace{g_s(Q(g_a(\underbrace{\mathbf{x}_{enc} - \mathbf{x}_{dec}}_{\text{residue}})))}^{\text{autoencoder}}. \tag{5.3}$$

This equation highlights two weaknesses of residual coding. First, the mixture between $\mathbf{x}_{enc}$ and $\mathbf{x}_{dec}$ is limited to a simple subtraction. Second, the fact that $\mathbf{x}_{out}$ results from a final addition between $\mathbf{x}_{dec}$ and the autoencoder output requires these quantities to have *compatible* nature such as image and image difference. Here, we propose to design a novel architecture to overcome these two issues. Its purpose is to condition the transmission of $\mathbf{x}_{enc}$ on the decoder information $\mathbf{x}_{dec}$, to transmit as few bits as possible. We call this architecture *conditional coding*. It must comply with two design constraints:

1. Learn the optimal combination of $\mathbf{x}_{enc}$ and $\mathbf{x}_{dec}$, to identify the information unavailable at the decoder, *i.e.* which has to be transmitted;

2. Be resilient to any nature of data for $\mathbf{x}_{enc}$, $\mathbf{x}_{dec}$ and $\mathbf{x}_{out}$.

5.3.2 Conditional coding principles

The proposed conditional coding architecture is depicted in Fig. 5.3. At the encoder-side, the analysis transform g_a is fed with the concatenation of the feature maps from $\mathbf{x}_{enc}$ and $\mathbf{x}_{dec}$. It computes latent variable $\mathbf{y}$, representing the information to be sent *i.e.* the information present at the encoder but missing at the decoder:

$$\mathbf{y} = g_a(\mathbf{x}_{enc}, \mathbf{x}_{dec}). \tag{5.4}$$

The analysis transform g_a is a convolutional network. As such its successive layers intertwine the encoder and decoder quantities in a more general way than a simple subtraction. This allows to learn an adapted mixture through the training process.

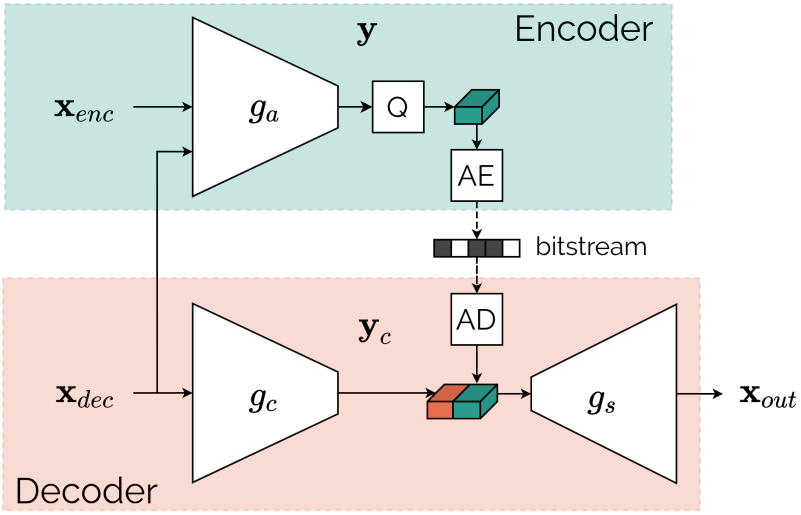


Figure 5.3: The conditional coding architecture.

In residual coding, the decoder-side information $\mathbf{x}_{dec}$ is combined with the synthesis transform output through an addition. Here, conditional coding uses a new conditioning transform g_c to incorporate the decoder-side information into the output $\mathbf{x}_{out}$. The conditioning transform extracts the relevant information within $\mathbf{x}_{dec}$ as additional *conditioning* latent variable $\mathbf{y}_c$:

$$\mathbf{y}_c = g_c(\mathbf{x}_{dec}). \tag{5.5}$$

Since the conditioning latent variable is computed at the decoder-side, it does not

require any transmission. Through the training process, the conditioning transform learns to compute the conditioning latent variable $\mathbf{y}_c$ in a form that suits the synthesis transform g_s. Indeed, the latent variables of both the analysis and the conditioning transforms are concatenated before being fed to the synthesis transform, which learns to combine the encoder and decoder information and computes the desired output $\mathbf{x}_{out}$:

$$\mathbf{x}_{out} = g_s(\mathbf{y}, \mathbf{y}_c). \tag{5.6}$$

5.3.3 Implementation and rate-distortion performance

The conditional coding mechanism is implemented to replace residual coding. Therefore, in the context of inter frame coding:

$$\begin{cases} \mathbf{x}_{enc} = \mathbf{x}_t & \text{the frame to code,} \\ \mathbf{x}_{dec} = \tilde{\mathbf{x}}_t & \text{the temporal prediction,} \\ \mathbf{x}_{out} = \hat{\mathbf{x}}_t & \text{the decoded frame.} \end{cases} \tag{5.7}$$

Compared to residual coding, the analysis g_a and synthesis g_s transforms remain unchanged, except for the number of input features. For the analysis, residual coding has a 3-feature input (residual for the Y, U and V channels). Here $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_t$ are concatenated, resulting in a 6-feature input (two Y, two U and two V channels). Similarly, the synthesis transform is now fed with the concatenation of two latent variables, $\mathbf{y}$ and $\mathbf{y}_c$. The conditioning transform replicates the architecture of the analysis, with a 3-feature input (only $\tilde{\mathbf{x}}_t$). Similarly to the residual coder, all transforms use $F = 192$ internal feature maps.

The training and the inference are performed as for the residual coder. Figure 5.4 shows the performance of conditional coding compared to residual coding. It significantly improves the performance, reaching equivalent quality at a rate 30 % lower. This proves the relevance of performing a richer mixture than a simple difference. Yet, this enhancement comes at the cost of the additional conditioning transform. In this case, the residual coder has 15 million parameters, while the conditional one has 21 million parameters.

5.3.4 Visualisation

This section aims to detail the inner behaviour of the conditional coding. To this effect, the output of the conditional coder is computed while disabling some latent variables (by setting them to zero). Namely, the analysis and conditioning latent variables are

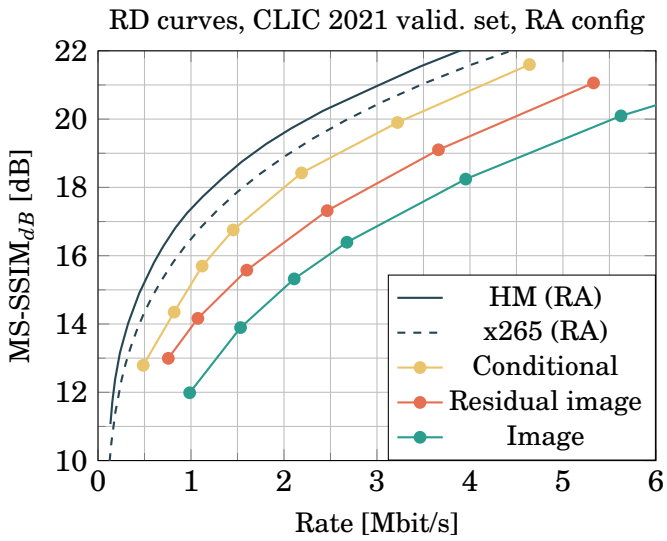


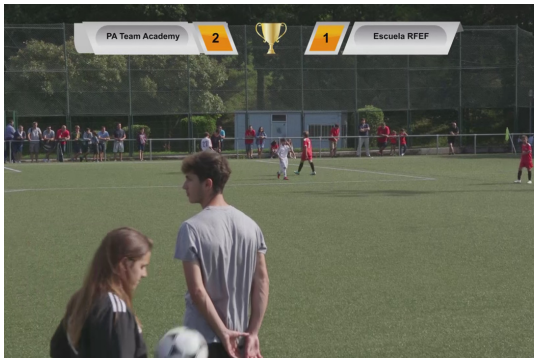
Figure 5.4: Conditional coding performance.

alternatively set to zero, as explained in Table 5.1. The visual results are presented in Fig. 5.5, in a B-frame coding context.

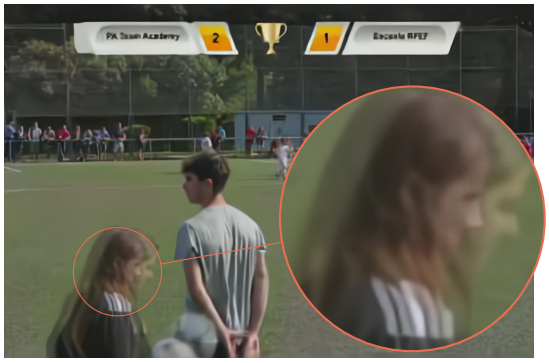
For now, the prediction $\tilde{\mathbf{x}}_t$ is a simple average of the reference frames. As such, it presents some artifacts around moving objects *e.g.* the girl in Fig. 5.5b. This temporal prediction is fed to to the conditioning transform to obtain the conditioning latent $\mathbf{y}_c = g_c(\tilde{\mathbf{x}}_t)$. Conversely, Fig. 5.5d presents the synthesis transform g_s output when computed only from the conditioning latent variable, allowing to represent the information contained in $\mathbf{y}_c$. In residual coding, the prediction would be directly added to the synthesis transform output to obtain the reconstructed frame. Thus, any artifacts present in the temporal prediction would be found in the output frame. Here, the conditioning transform allows to remove some of the artifacts. This is particularly visible around the girl's face (Fig. 5.5d), where most of the visual degradations visible in the temporal prediction are absent from the conditioning-only synthesis. This mechanism is particularly useful at low-rate as it

Table 5.1: Different configurations for the conditional coding ablation.

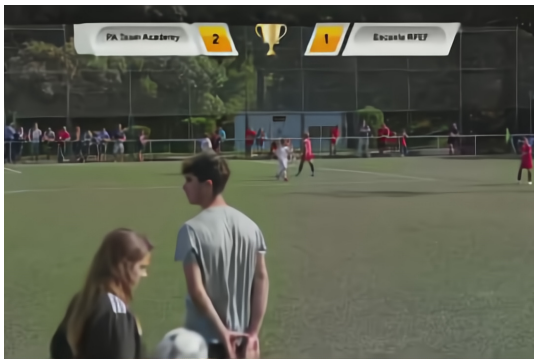
	Latent variables		**Synthesis**
	Analysis	**Conditioning**	**operation**
Normal	✓	✓	$g_s(\mathbf{y},\mathbf{y}_c)$
Conditioning only		✓	$g_s(0,\mathbf{y}_c)$
Analysis only	✓		$g_s(\mathbf{y},0)$



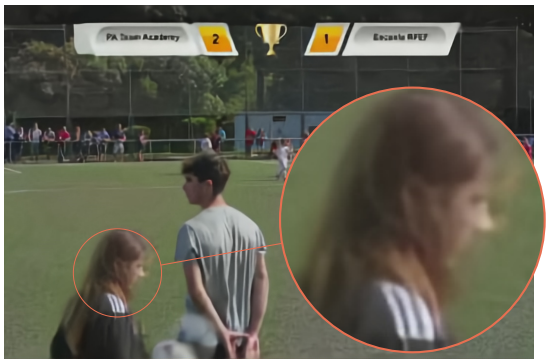
(a) Original frame to code $\mathbf{x}_t$.



(b) The temporal prediction $\tilde{\mathbf{x}}_t = \frac{1}{2}\left(\hat{\mathbf{x}}_p + \hat{\mathbf{x}}_f\right)$.

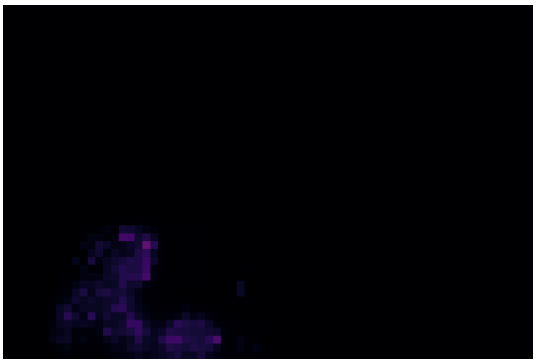


(c) Reconstructed frame $\hat{\mathbf{x}}_t = g_s(\mathbf{y}, \mathbf{y}_c)$.



(d) Conditioning transform only: $g_s(0, \mathbf{y}_c)$.

(e) Analysis transform only: $g_s(\mathbf{y}, 0)$.



(f) Spatial distribution of the rate.

Figure 5.5: Low-rate conditional coding illustrated on a B-frame from the CLIC sequence *Sports_1080P-6710*. This frame costs 2 481 bits and has a MS-SSIM of 13.35 dB.

avoids sending corrective terms through the analysis transform.

Figure 5.5e presents the synthesis of the analysis latent variable $\mathbf{y}$ only. This latent variable is obtained by applying the analysis transform on both the frame to code and its prediction: $\mathbf{y} = g_a(\mathbf{x}_t, \tilde{\mathbf{x}}_t)$. As such, it represents the unpredicted part of the frame *i.e.* the correction that has to be sent. In this examples, the unpredicted areas concern mostly the girl and the ball. The analysis latent is the only quantity requiring to be sent to the decoder. Consequently, the spatial distribution of the rate (see Fig. 5.5f) shows that bits are spent only for the areas requiring information from the analysis transform. Finally, Fig. 5.5c depicts the usual output of conditional coding *i.e.* when the latent variables from both the analysis and the conditioning transforms are fed to the synthesis transform. This illustrates that the combination of the encoder-side and decoder-side information is properly achieved by conditional coding, without noticeable artifacts (except the obvious loss in quality due to lossy compression).

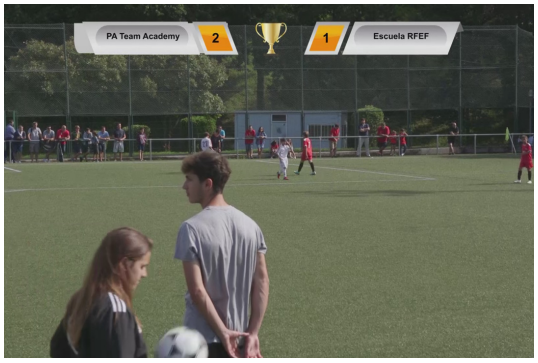
Those illustration shows that the decoder-side information $\tilde{\mathbf{x}}_t$ is well exploited by the proposed conditional coding. Both for lower rates (Fig. 5.5e) and for higher rates (Fig. 5.6e), the coder conveys only what is not present at the decoder-side.

The set of Figures 5.6 presents the same visualisations as the previous one except they are obtained from a higher-rate network. When the rate-constraint is decreased, the analysis latent variable $\mathbf{y}$ are allowed to carry more information. This is clearly visible in Fig. 5.6e, which exhibits more details and is active in more areas than its low-rate counterpart. This translates into a more important rate, as illustrated through the spatial rate distribution, in Fig. 5.6f. As the higher-rate network relies more on the analysis transform, the conditioning transform tends to be less important. For instance, Figure 5.6d shows that it removes less artifacts from the temporal prediction.

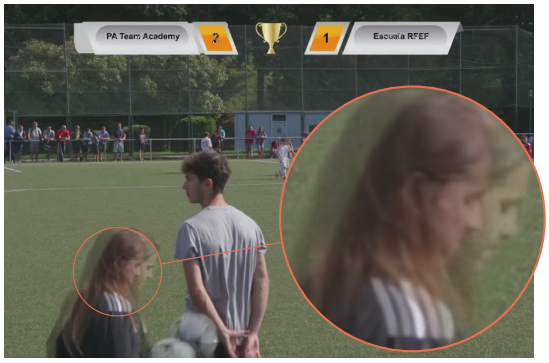
5.4 Latent domain residual coding

The main idea of conditional coding is the combination $\mathbf{x}_{enc}$ and $\mathbf{x}_{dec}$ in the latent domain, allowing to perform a richer mixture of these quantities. Likewise, the latent-domain residual approach [66] proposes a similar but not identical method, summarised in Fig. 5.7. In this coding scheme, the analysis transform g_a is applied successively on $\mathbf{x}_{enc}$ and $\mathbf{x}_{dec}$, yielding two latent variables:

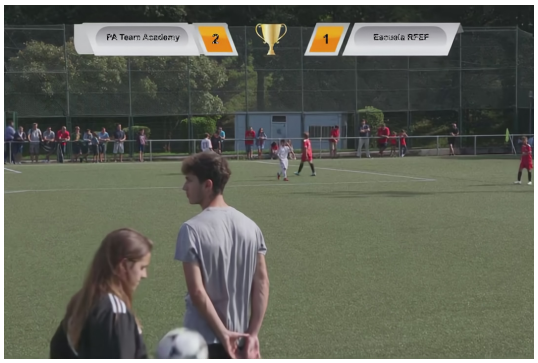
$$\mathbf{y}_{enc} = g_a(\mathbf{x}_{enc}) \;;\; \mathbf{y}_{dec} = g_a(\mathbf{x}_{dec}). \tag{5.8}$$



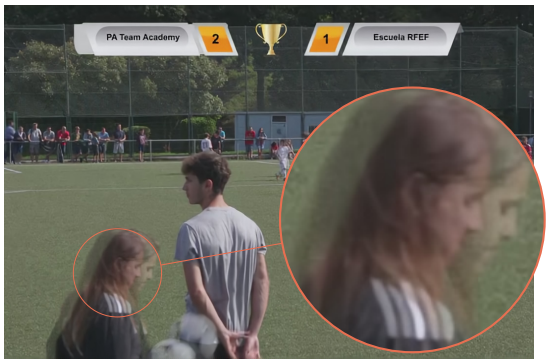
(a) Original frame to code $\mathbf{x}_t$ (Identical to Fig. 5.5a).



(b) The temporal prediction $\tilde{\mathbf{x}}_t$ (Identical to Fig. 5.5b).

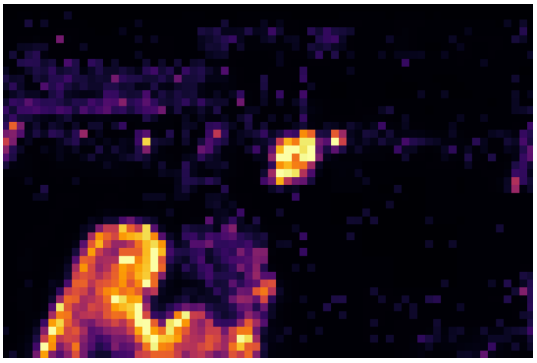


(c) Reconstructed frame $\hat{\mathbf{x}}_t = g_s(\mathbf{y}, \mathbf{y}_c)$.



(d) Conditioning transform only: $g_s(0, \mathbf{y}_c)$.

(e) Analysis transform only: $g_s(\mathbf{y}, 0)$.



(f) Spatial distribution of the rate.

Figure 5.6: High-rate conditional coding illustrated on a B-frame from the CLIC sequence *Sports_1080P-6710*. This frame costs 33 712 bits and has a MS-SSIM of 21.48 dB.

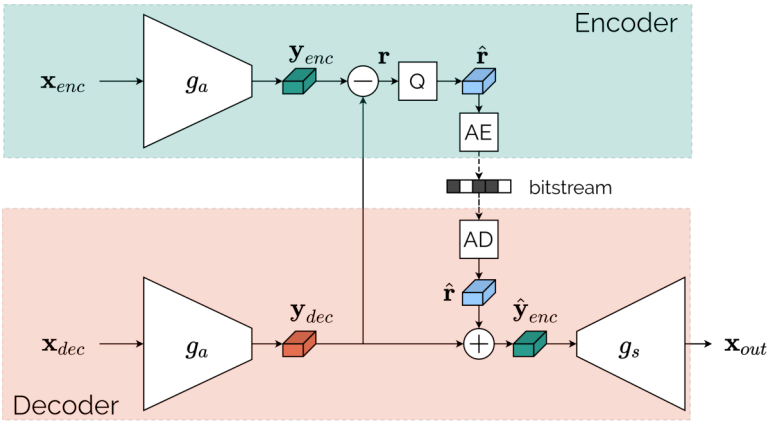


Figure 5.7: Latent-domain residual.

Since the latent variable $\mathbf{y}_{dec}$ is computed based solely on $\mathbf{x}_{dec}$, it is available at the decoder without transmission, enabling to compute a latent-domain residue:

$$\mathbf{r} = \mathbf{y}_{enc} - \mathbf{y}_{dec}. \tag{5.9}$$

This residue is quantized and sent to the decoder. At the decoder side, $\mathbf{y}_{dec}$ is added back to the quantized residue to retrieve a noisy version of the encoder latent $\hat{\mathbf{y}}_{enc}$, fed to the synthesis transform g_s to compute the desired output:

$$\mathbf{x}_{out} = g_s(\hat{\mathbf{y}}_{dec}). \tag{5.10}$$

As with conditional coding, the encoder and decoder information is mixed in the latent domain. Like conditional coding, this approach requires additional decoder-side computation (in order to obtain $\mathbf{y}_{dec} = g_a(\mathbf{x}_{dec})$).

Figure 5.8 presents the latent-domain residual coding performance, which offer significantly better results than spatial-domain residual coding. Yet, latent-domain residual coding slightly underperforms compared to conditional coding. Two reasons explain why conditional coding is a more interesting architecture than latent-domain residual coding. First, the encoder and decoder quantities are still combined through an addition, even though it takes place in the latent domain. Compared with conditional coding which simply concatenates the quantities, it may hinder the expressiveness of the combination. Second, the same analysis transform g_a is used at the decoder-side, instead of learning a specialized conditioning transform g_c. Recall that the transform g_a aims to reduce the entropy of its input (due to the rate constraint). When performed at the decoder to achieve

latent residual coding, it purposely removes information from the temporal prediction. This might lead to a suboptimal mixture.

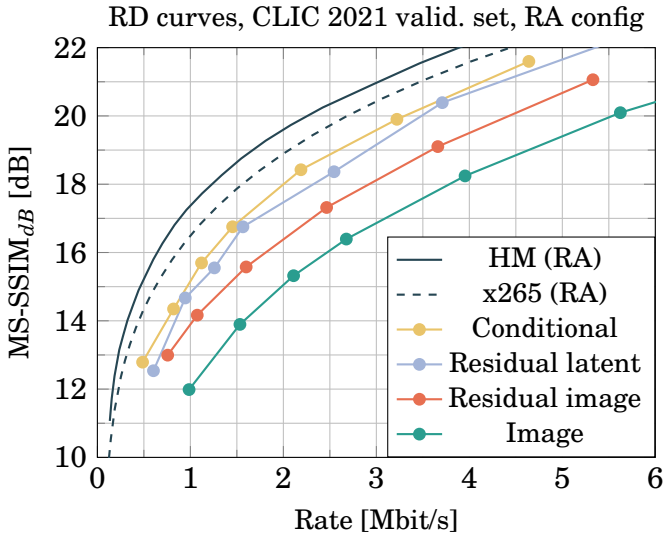


Figure 5.8: Latent residual coding performance.

Besides the performance aspect, latent-domain residual coding presents a second drawback. As $\mathbf{x}_{enc}$ and $\mathbf{x}_{dec}$ are fed to the same transform g_a, with the same parameters, they must have the same nature and dimension. In the current setup (*i.e.* $\mathbf{x}_{enc} = \mathbf{x}_t$ and $\mathbf{x}_{dec} = \tilde{\mathbf{x}}_t$), this constraint is acceptable. As we will see in the next chapter, it will not be the case for coding of the motion information. In the end, conditional coding is the most effective and the most flexible architecture to exploit decoder-side information. It will be implemented in most of the proposed video coding scheme to exploit decoder-side information as much as possible.

5.5 Do we need a dedicated intra frame coder?

Most existing learned video coders feature a dedicated intra frame coder (see the literature review in Table 4.1). This is due to the usage of a image-domain residual coding, where the nature of the conveyed data varies significantly between intra and inter frames. For inter frames, the autoencoder is fed with a difference image, while intra frames require to convey image data. Thus, it looks sensible to learn one dedicated coder for each nature of content. To assess whether a separate intra coder is required, the different systems previously learned are evaluated in an All Intra configuration. For this experiment, the

image coder constitutes the upper bound for the performance. Indeed, it is only able of coding intra frames and is thus trained for this task.

Fig. 5.9 shows the All Intra coding results of the different architectures. While the image-domain residual coder struggles to properly code images, both latent-domain residual coding and coding coding achieve All Intra performance on par with the pure image coder. This demonstrates that conditional coding avoids the need of a dedicated intra coder. Similar conclusions are drawn by Djelouah *et al.* [66], who states that performing the combination of encoder and decoder information in the latent domain prevents the need of designing a dedicated intra coder.

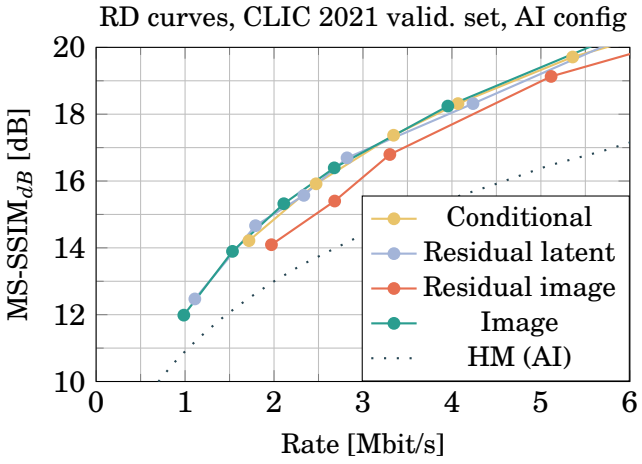


Figure 5.9: All Intra coding performance.

5.6 Variable quantization steps

5.6.1 Motivation and implementation

During the processing of a video sequence, the role of the analysis transform g_a varies significantly. For intra frames, there is no contribution from the conditioning transform g_c and the frame is transmitted through the latent variable $\mathbf{y} = g_a(\mathbf{x})$. In the case of an inter frame, a prediction $\tilde{\mathbf{x}}$ is available and the conditioning transform is used. As such, far less information are sent through the analysis latent variable $\mathbf{y}$, as most of the relevant information is already available from the conditioning transform.

Since the importance of the analysis latent variable $\mathbf{y}$ depends on the frame type, we propose to apply a linear operation on $\mathbf{y}$, conditioned on the frame type. The proposed

method is inspired by the one introduced by Guo *et al.* for variable rate coders [74] and is presented in Fig. 5.10. For each frame type $f \in \{\text{I, P, B}\}$, a feature-wise pair of quantization gains $\{\Gamma_f^{enc}, \Gamma_f^{dec}\}$ is learnt. Each quantization gain $\Gamma \in \mathbb{R}^{F_y}$ with F_y the number of feature maps composing the analysis latent variable $\mathbf{y}$. The feature-wise quantization gains are applied prior to quantization at the encoder-side and at the decoder-side after arithmetic decoding. Seen from the synthesis transform, the received analysis latent is:

$$\hat{\mathbf{y}} = \Gamma_f^{dec} \odot Q(\Gamma_f^{enc} \odot \mathbf{y}). \tag{5.11}$$

where $\odot$ denotes the feature-wise multiplication and Q the quantization function.

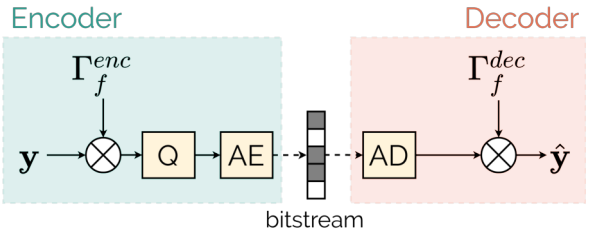


Figure 5.10: Feature-wise quantization gains.

Conceptually, these frame-type dependent quantization gain can be compared to the Δ_{QP} of traditional coders, which sets different quantization accuracies depending on the frame type and position in a GOP. Yet, the proposed quantization gains offers more flexibility than the usual Δ_{QP}:

- A different gain is learned per latent feature maps, compared to a single scalar gain;
- There is no constraint imposed such as $\Gamma_f^{dec} = \frac{1}{\Gamma_f^{enc}}$;
- The quantization gains can be learned to operate as a *switch*, enabling or disabling particular latent feature maps according to the frame type;
- All the quantization gains are set through an end-to-end optimization of the entire video coding schemes.

5.6.2 Experimental results

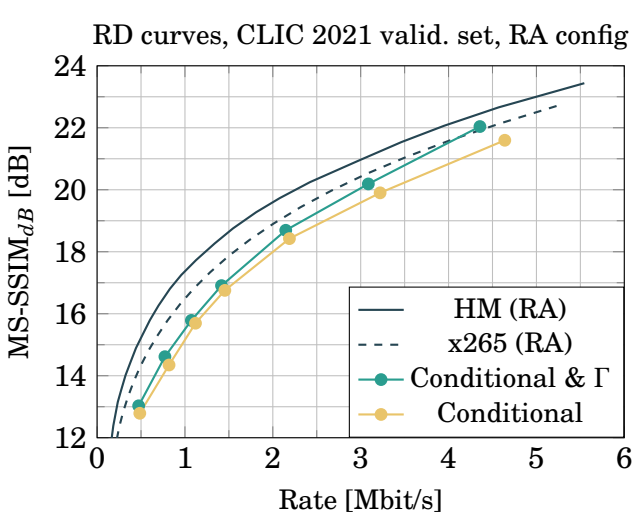
The quantization gains are implemented on the conditional coding scheme. While the conditional coder has 25 million parameters, the three pairs of feature-wise quantization

gains represent a few parameters:

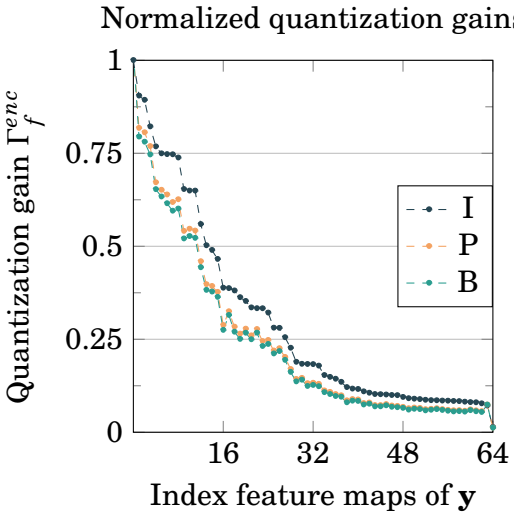
$$\begin{array}{ccccccc} \text{Encoder \& Decoder} & \times & \text{I, P \& B} & \times & F_y & = \text{Number of parameters} \\ 2 & \times & 3 & \times & 256 & = 1\,536 \end{array} \tag{5.12}$$

The training stage of the system is carried out similarly to the other coding schemes and its results are presented in Fig. 5.11a. For a negligible increase in complexity, the introduction of quantization gains conditioned on the frame type allows to save 9.6 % rate.

To better understand the behaviour of the quantization gains, Fig. 5.11b shows the normalized values of Γ^{enc} for one system. As expected, the quantization gain *i.e.* the quantization accuracy is different for the intra frames (I) and the inter frames (P & B). For a given feature map, its associated quantization accuracy is almost always smaller for an inter frame than for an intra frame. Together with the rate-distortion results, this proves the relevance of learning dedicated quantization accuracy according to the type of the frames. Similar visualisations for all rate targets are presented in Appendix D.3, where the behaviour of the quantization gains remain similar regardless the rate target.



(a) Quantization gains performance.



(b) Encoder quantization gains Γ_f^{enc} for f a I, P or B-frames.

Figure 5.11: Performance and analysis of the quantization gains.

5.7 Conclusion

In this chapter, several ways of leveraging a temporal prediction have been investigated and the conditional coding architecture has been introduced. Table 5.2. summarises the rate saving brought by each element. Conditional coding couplet with quantization gains conditioned on the frame type, achieves a BD-rate of −35.9 % over the usual spatial residual coding *i.e.* it obtains equivalent quality at a rate 35.9 % lower. This demonstrates the relevance of the different techniques introduced in this Chapter.

Table 5.2: BD-rates of the different coding schemes in Random Access configuration.

	Residual		Conditional	Q.	BD-rate	
	Image	Latent	Coding	gains		
HM (All Intra)*			/		+158.2 %	+580.3 %
Image					+36.7 %	+184.0 %
Spatial res.	✓				Reference	+129.0 %
Latent res.		✓			−21.2 %	+71.9 %
Cond.			✓		−29.3 %	+60.8 %
Cond., Γ			✓	✓	−35.9 %	+42.9 %
x265 medium† (*motion*)			/		−46.0 %	+22.8 %
HM† (*motion*)			/		−56.3 %	Reference

**: evaluated at QP 36, 41, 46, 51.*
†*: evaluated at QP 22, 27, 32, 37.*

Besides its compelling results, conditional coding allows to process intra frames and inter frames with the same parameters, preventing the need for a dedicated intra coder. Thanks to its performance and flexibility, the conditional coding mechanism has been used throughout the work carried out in this thesis. Conditional coding was introduced in a conference paper presented at the *IEEE International Workshop on Machine Learning for Signal Processing* (MLSP) 2020 [18].

Yet, the naive temporal prediction of the coding scheme hinders its performance, which explains the significant gap with HEVC (BD-rate of +42.9 % when compared to the HM). In order to further improve the compression efficiency, a better temporal prediction has to be available. To this end, the next chapter focuses on the design of a proper motion compensation process, which implies to accurately estimate and efficiently transmits motion information.

CHAPTER 6

COMPUTATION OF A TEMPORAL PREDICTION

6.1 Introduction

VIDEO coders compute a prediction of frames in order to only convey the unpredicted content. Consequently, achieving a more accurate temporal prediction allows to reduce the amount of unpredicted content, reducing the overall rate. The successive video frames usually present similar content with small displacements among frames. As such, it is possible to compute a relevant prediction by performing a motion compensation, based on the previously received frames. In order to be accurate, the motion compensation step relies on motion information which needs to be properly estimated and efficiently transmitted so that the prediction improves the rate-distortion cost of the codec.

This chapters builds upon the coding scheme designed in the previous chapter based on a naive prediction. In the first half of the chapter, motion information is integrated into the existing coding scheme, improving the compression efficiency. Then, the conditional coding architecture is implemented for the motion information to further increase the performance. Lastly, the resulting coding scheme is evaluated in details to assess its strength and weakness.

6.2 Coding scheme featuring a prediction step

6.2.1 Bidirectional prediction

This chapter aims to enhance the coding scheme introduced in the previous chapter. Let us call CNet, the conditional coder which implements the operation $c(\mathbf{x}_t, \tilde{\mathbf{x}}_t)$ *i.e.* it conveys the frame $\mathbf{x}_t$ while leveraging a temporal prediction $\tilde{\mathbf{x}}_t$ available at the decoder. In order to improve the compression performance, we propose to move forward the naive

prediction of the previous chapter by achieving an actual motion compensation process. To this end, the coding scheme now transmits some motion information, conditioning the motion compensation process. In the learned video compression literature, bidirectional predictions remain less common than simple single-reference prediction. Indeed, only 3 out of the 12 coding schemes presented in Table 4.1 feature a bidirectional prediction. Yet, Section 2.3.1 and Eq. (2.6) remind that a bidirectional prediction step allows to reduce the prediction noise, *i.e.* improving its accuracy and thus the entire coding scheme. As such, we propose to implement a bidirectional prediction within the coding scheme. Fig. 6.1 details the bi-directional motion compensation step.

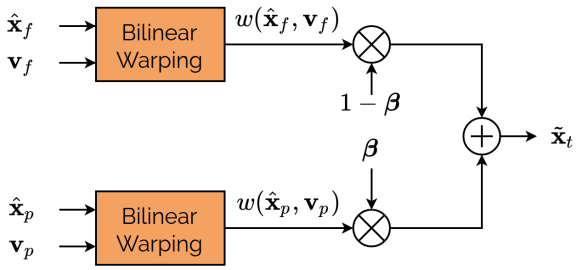


Figure 6.1: Bi-directional temporal prediction.

Let us model the frame to code as $\mathbf{x}_t$, a $C \times H \times W$ tensor. Bidirectional prediction computes $\tilde{\mathbf{x}}_t$, a prediction of $\mathbf{x}_t$ based on two reference frames, $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_f$, and their respective motion relative to $\mathbf{x}_t$. This motion information is represented by two optical flows $\mathbf{v}_p$ and $\mathbf{v}_f$ *i.e.* motion maps representing the pixel-wise two-dimensional motion from $\mathbf{x}_t$ to one of its reference frames. The two optical flows are applied separately on the two reference frames through w, a warping function which performs motion compensation. Here, sub-pixel motions are achieved through a bilinear interpolation. As such, w is called a bilinear warping function. Finally, the two intermediate motion compensated references are combined through a sum weighted by $\boldsymbol{\beta} \in [0,1]^{H \times W}$, a pixel-wise bidirectional weighting (see Section 2.3.1). This yields the bidirectional prediction equation:

$$\tilde{\mathbf{x}}_t = \overbrace{\boldsymbol{\beta} \odot w(\hat{\mathbf{x}}_p; \mathbf{v}_p)}^{\text{Compensation from } \hat{\mathbf{x}}_p} + \overbrace{(1-\boldsymbol{\beta}) \odot w(\hat{\mathbf{x}}_f; \mathbf{v}_f)}^{\text{Compensation from } \hat{\mathbf{x}}_f}, \quad \odot \text{ is a pixel-wise product.} \tag{6.1}$$

The temporal prediction has to be present at the decoder so that only the unpredicted content is sent by the encoder. To this end, the two optical flows and the bidirectional weighting must be conveyed to the decoder. Figure 6.2 presents the overall coding scheme.

The first component, *MNet*, estimates and conveys the motion information. Then, *Motion Comp.* implements the motion compensation described in Eq. 6.1. Finally, *CNet* performs the conditional coding of $\mathbf{x}_t$ conditioned on $\tilde{\mathbf{x}}_t$, as presented in the previous Chapter.

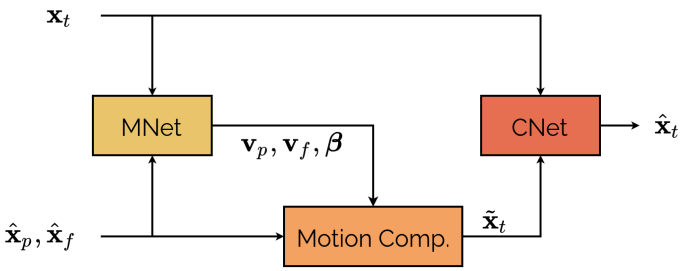


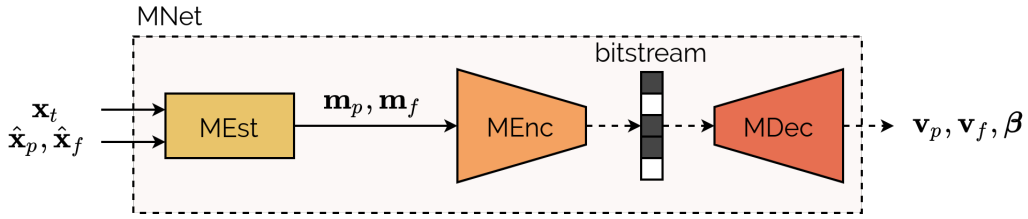
Figure 6.2: Video coding scheme with an actual motion compensation step.

6.2.2 Motion information at the decoder side

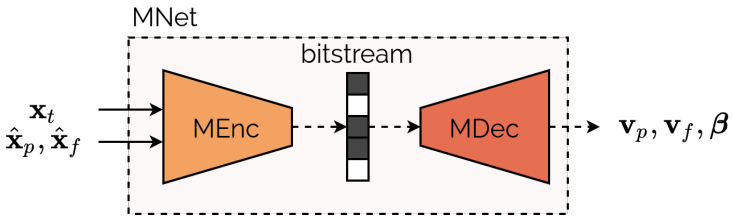
Figure 6.3 presents two means to obtain relevant motion information at the decoder side. The first one is a two-step method, with a separate motion estimation network located at the encoder which computes encoder-side motion maps $\mathbf{m}_p$ and $\mathbf{m}_f$. Then, these motion maps are fed to a motion autoencoder which performs lossy coding and retrieves decoder-side motion information $\mathbf{v}_p$ and $\mathbf{v}_f$. The bidirectional weighting $\boldsymbol{\beta}$ is obtained as an additional output of the motion autoencoder, as in [66].

The second method consists of a single step, based on an autoencoder fed with the current frame and the reference frames. The autoencoder both estimates the motion and perform lossy coding *i.e.* finding a low-entropy representation suited for transmission allowing to retrieve motion information at the decoder. In the literature, the single-step approach is only found for P-frame coding [23]–[25]. Since we strive to design a simple video coder, the single-step approach is chosen and extended to B-frame coding by adding a supplementary output for $\boldsymbol{\beta}$, see Fig. 6.3b. Thanks to this approach, we avoid relying on a dedicated and pre-trained motion estimation network, which can often results in a cumbersome training process and heavier systems.

Regardless of the method used to obtain motion information, the training of the overall coding scheme is not trivial. In particular, obtaining proper motion information at the decoder is notoriously difficult [23]. In the case of the two-step method, this is tackled by using an off-the-shelf network for the motion estimation such as SpyNet [22] or PWCNet



(a) Two-step MNet, with MEst a dedicated motion estimation network.



(b) Single step MNet.

Figure 6.3: Two main trends for the MNet architecture. MEnc and MDec denotes a learned encoder-decoder pair.

[71]. We argue that integrating pre-trained motion estimation networks is not the ideal solution for several reasons:

- It brings about additional parameters (*e.g.* 8 million weights for PWCNet);
- Such networks are mostly trained on synthetic datasets, for which the underlying motion is known (*e.g.* MPI-Sintel [75]). They are not necessarily tuned for natural video content;
- They aim to predict the most accurate motion information as they are often trained through the minimization of the L_2 error of the motion. Yet, the objective of motion estimation in an actual video coding scheme is not to be the most accurate. Its goal is to obtain motion information leading to the best rate-distortion cost, once integrated in the whole coding chain.

Since pre-trained motion estimation networks add supplementary parameters and require a retraining to be adapted to the video coding objective and content, they are not used in our coding scheme. On the other hand, the single-step method very often requires some sort of pre-training or a dedicated loss term to foster convergence [23], [24]. Yet, adding dedicated loss terms to the rate-distortion objective leads to an optimization

objective which deviates from the targeted rate-distortion optimization. This might lead to suboptimal systems. For both convenience and performance-related motivations, we advocate for an actual end-to-end rate-distortion training, with no pre-trained component or supplementary loss terms. To this end, an additional coding mode called the *Skip mode* is introduced to promote the learning of relevant motion information.

6.3 Skip mode: an additional coding mode

6.3.1 Principles

In this thesis, the coding scheme is enhanced through a supplementary coding mode: the Skip mode. It is inspired to an existing Skip mode in conventional video coding, where the temporal prediction is used without the residual. Here, the proposed Skip mode consists in a pixel-wise copy of the temporal prediction. Offering the possibility of copying the prediction acts as a strong incentive to learn relevant motion information. Indeed, in a system without the Skip mode, the motion information is learned to obtain a proper prediction which performs well when fed to the CNet. Thus, the training of the MNet happens through CNet, making it more difficult to converge. Here, the Skip mode is a shortcut: an accurate prediction can now be exploited directly (*i.e.* by being copied) instead of going through the CNet. This considerably stabilizes the training process, allowing to train the whole coding scheme from scratch using the rate-distortion loss.

The reconstructed frame $\hat{\mathbf{x}}_t$ is computed by adding the contributions of both the CNet and Skip mode. This sum is arbitrated by the *continuous* pixel-wise coding mode selection $\boldsymbol{\alpha} \in [0,1]^{H \times W}$, with H and W the spatial dimension of $\mathbf{x}_t$:

$$\hat{\mathbf{x}}_t = \overbrace{c(\boldsymbol{\alpha} \odot \mathbf{x}_t, \boldsymbol{\alpha} \odot \tilde{\mathbf{x}}_t)}^{\text{CNet}} + \overbrace{(1-\boldsymbol{\alpha}) \odot \tilde{\mathbf{x}}_t}^{\text{Skip mode}}, \ \odot \text{ is a pixel-wise product.} \tag{6.2}$$

The coding mode selection applied on the CNet input (at the encoder) to select the areas of $\mathbf{x}_t$ that have to be conveyed through the conditional coder. Its role is to zero areas from $\mathbf{x}_t$ before coding them through CNet, allowing to save their associated rate. Conversely, $\boldsymbol{\alpha}$ applied at the decoder side to obtain the Skip mode contribution. The coding mode selection is continuous, allowing blending between the two coding modes.

Figure 6.4 shows an example of coding mode selection $\boldsymbol{\alpha}$. In this example, the man's hand in the middle as well as the left man's shoulder are moving fast, resulting in an

inaccurate temporal prediction. Consequently, this prediction can not be copied from the Skip mode and these areas rely on CNet to be transmitted (red areas on the mode selection map). The other areas are well predicted and use mostly the Skip mode (blue areas).

(a) Past reference $\hat{\mathbf{x}}_p$. (b) Current frame $\mathbf{x}_t$. (c) Future reference $\hat{\mathbf{x}}_f$. (d) Mode selection $\boldsymbol{\alpha}$. *blue*: skip, *red*: CNet.

Figure 6.4: Coding mode selection $\boldsymbol{\alpha}$ for a B-frame from the CLIC sequence *TelevisionClip_1080P-4c94*.

In order to be available at the decoder, $\boldsymbol{\alpha}$ must be transmitted. Since it is applied on CNet inputs, its transmission must occur prior to CNet. To this end, an additional output is added to the motion autoencoder MNet to retrieve $\boldsymbol{\alpha}$ at the decoder side. Figure 6.5 presents the block diagram of a coding scheme featuring the Skip mode. CNet implements the conditional coding mechanism presented in Section 5.3. On the other hand, the MNet implements the usual analysis-synthesis architecture, shown in Fig. 6.6. The MNet transforms do not share the same parameters than those of the CNet and their architecture can be slightly different. To highlight this fact, the MNet transforms are denoted with a superscript m.

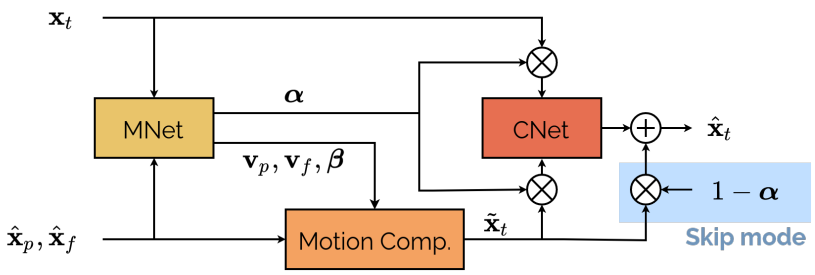


Figure 6.5: Block diagram of a coding scheme featuring the Skip mode.

MNet features all the relevant techniques required to obtain a proper coding network: hyperprior, attention modules, *etc.* The analysis g_a^m is fed with the frame to code and both

reference frames. It computes a latent variable, a compact representation of a relevant coding mode selection, bidirectional weighting and optical flows. After quantization, this latent variable is sent to the decoder and fed to the synthesis in order to compute the coding mode $\boldsymbol{\alpha}$ as well as both optical flows $\mathbf{v}_p$, $\mathbf{v}_f$ and the bidirectional weighting $\boldsymbol{\beta}$. The detailed implementation of MNet is presented in appendix E.1. Its transforms are implemented using $F = 192$ internal feature maps, resulting in 15 million parameters for MNet and 36 million parameters for the entire coding scheme.

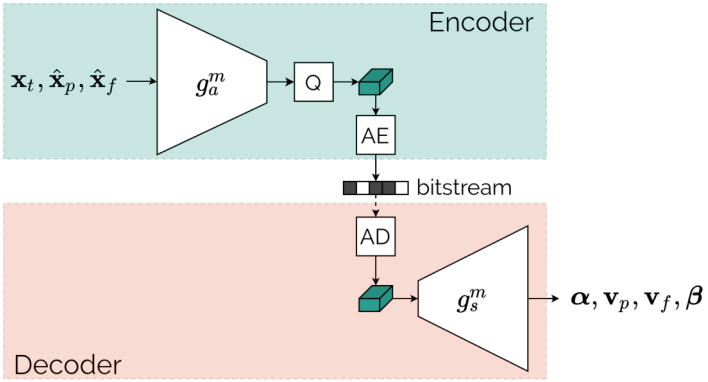


Figure 6.6: The MNet architecture.

6.3.2 Training

The entire coding scheme is trained in an end-to-end fashion to minimize the actual rate-distortion loss. This is particularly important to note that there is no dedicated loss term for $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\mathbf{v}_p$ or $\mathbf{v}_f$. The MNet autoencoder learns through the rate-distortion cost to efficiently transmit a relevant and low-rate coding mode selection and motion information. As such, a simple rate-distortion training allows to perform a relevant partitioning between Skip mode and CNet. Furthermore, the learning step also enables to automatically share the rate budget between MNet and CNet.

Yet, there is a subtlety at the beginning of the training. During the first training iterations, CNet and Skip mode are not ready to compete. Indeed, CNet has to be trained, while the Skip mode is already able to copy the prediction. For this reason, $\boldsymbol{\alpha}$ is set to zero on one half of the image and to one on the other half. The zeroed half relies on the Skip mode, fostering the learning of a relevant motion compensation provided by MNet. The other half uses only CNet, preparing CNet for the competition with the Skip mode.

6.3.3 Behaviour

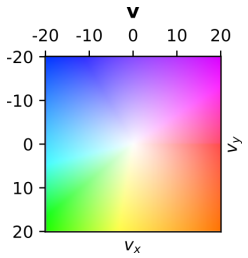
The new quantities introduced in this Chapter are illustrated through a comprehensive visual example of B-frame coding. The first step is the coding pipeline is the MNet, whose inputs and outputs are shown in Figure 6.7. MNet inputs are the frame to code $\mathbf{x}_t$ (Fig. 6.7b) and two reference frames $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_f$(Fig 6.7a and 6.7c). MNet uses these inputs to compute a latent variable, conveyed to the decoder, where the MNet synthesis computes different outputs: two optical flows $\mathbf{v}_p$, $\mathbf{v}_f$ (Fig. 6.7d and 6.7f), the bidirectional prediction weighting $\boldsymbol{\beta}$ (Fig. 6.7g) and the coding mode selection $\boldsymbol{\alpha}$ (Fig. 6.7h).

(a) Past reference $\hat{\mathbf{x}}_p$.

(b) Original frame to code $\mathbf{x}_t$.

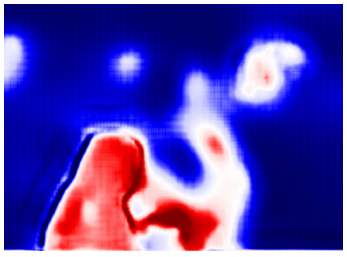
(c) Future reference $\hat{\mathbf{x}}_f$.

(d) Optical flow $\mathbf{v}_p$: motion from $\mathbf{x}_t$ to $\hat{\mathbf{x}}_p$.

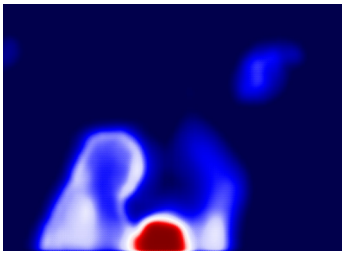


(e) Optical flow representation. Motion is expressed in pixel.

(f) Optical flow $\mathbf{v}_f$: motion from $\mathbf{x}_t$ to $\hat{\mathbf{x}}_f$.



(g) Bidirectional prediction weighting $\boldsymbol{\beta}$.*blue*: warp from $\hat{\mathbf{x}}_f$, *red*: warp from $\hat{\mathbf{x}}_p$.



(h) Coding mode selection $\boldsymbol{\alpha}$. *blue*: skip, *red*: CNet.

Figure 6.7: The inputs and outputs of MNet.

The optical flows are shown through a colour-based representation as presented in Fig. 6.7e. The intensity of the colour indicates the magnitude of the motion, while the colour

itself indicates the direction of the motion. Here, one optical flow represents the motion from $\mathbf{x}_t$ to $\hat{\mathbf{x}}_p$, while the other represents the motion from $\mathbf{x}_t$ to $\hat{\mathbf{x}}_f$. As it is often the case, both optical flows are approximately the opposite of each other. Consequently, the colours are not identical between the two optical flows as the motion does not go in the same direction. These two optical flows demonstrate that the proposed system is able to learn accurate optical flows through a pure rate-distortion training. Indeed, the optical flows are precise in position, their edges are relevant and the motion direction and magnitude is accurate. An additional example of optical flows presenting a complex background motion (zoom) is available in Appendix E.2.

The next set of illustrations in Fig. 6.8 presents the actual motion compensation step, based on the reference frames, the optical flows and the bidirectional prediction weighting already introduced in the previous visualisations. Figures 6.8a and 6.8b show the difference between the frame to code and each of the reference frames. The optical flows are used to compute two intermediate motion compensated frames, shown in Fig. 6.8d and 6.8e. Finally, the two intermediate motion compensations are gathered through the sum weighted by $\boldsymbol{\beta}$, yielding the final temporal prediction (Fig. 6.8f). The bidirectional prediction interest is illustrated through the computation of the PSNR relative to the current frame. Compared to the intermediate motion compensations, the bidirectional prediction increases the PSNR by around 2 dB. This leads to a prediction with little error, except for areas with important motion such as the ball, as shown in Fig. 6.8c.

The bidirectional weighting $\boldsymbol{\beta}$ exhibits an interesting behaviour in Fig. 6.7g. In this example, most of the frame is composed of motionless background. For these areas, the value of $\boldsymbol{\beta}$ is of limited importance, as $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_f$ are almost identical and both can serve as reference. Here the system uses $\boldsymbol{\beta} \simeq 0.75$, which corresponds to most of the medium blue areas. All the people in the frame are moving. For these areas, the system relies on $\boldsymbol{\beta} \simeq 0.5$ (in white), decreasing the prediction error by averaging the intermediate predictions.

The thin dark blue and dark red areas around the girl at the foreground allows the system to elegantly handle disocclusions. Indeed, the girl moves from the left to the right. As such, the area located at its left side is only visible in the future reference frame $\hat{\mathbf{x}}_f$. Similarly, the area at the right of the girl is only visible in the past reference $\hat{\mathbf{x}}_p$. Using $\boldsymbol{\beta} \simeq 0$ (dark blue) or $\boldsymbol{\beta} \simeq 1$ (dark red) enables the usage of a unique reference frame for the motion compensation process, allowing to properly handle disocclusions. This is an interesting behaviour, which is not achieved by the weighted prediction of traditional video coders. More detailed examples of disocclusions are presented in appendix E.3.

(a) Error before compensation $\mathbf{x}_t - \hat{\mathbf{x}}_p$.

(b) Error before compensation $\mathbf{x}_t - \hat{\mathbf{x}}_f$.

(c) Prediction error $\mathbf{x}_t - \tilde{\mathbf{x}}_t$.

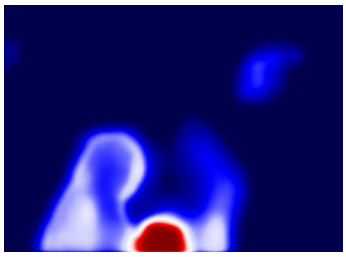
(d) Intermediate motion comp. $w(\hat{\mathbf{x}}_p; \mathbf{v}_p)$. PSNR = 28.39 dB.

(e) Intermediate motion comp. $w(\hat{\mathbf{x}}_f; \mathbf{v}_f)$. PSNR = 27.91 dB.

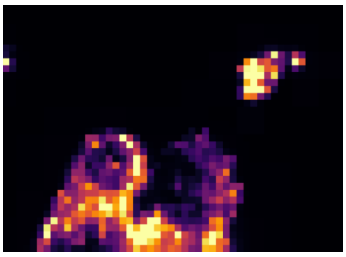
(f) Prediction $\tilde{\mathbf{x}}_t$, computed using Eq. (6.1). PSNR = 30.25 dB.

Figure 6.8: Motion compensation through a bidirectional prediction process.

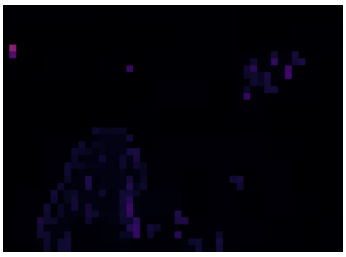
Once a temporal prediction is available, the coding mode selection $\boldsymbol{\alpha}$ (Fig. 6.7h) arbitrates each pixel between the two coding modes: CNet and Skip. Figure 6.9d shows the areas selected for the Skip mode. Since the videos are in the YUV colour space, zeroed areas appear in green. Thanks to the accuracy of the temporal prediction, most of the reconstructed frame is a simple copy of the prediction, without any transmission. Only some areas difficult to predict (the ball and the girl) require to be conveyed through CNet (Fig. 6.9e). Finally, the contributions of both CNet and the Skip mode are summed to obtain the final reconstructed frame (Fig. 6.9f). As expected, CNet rate distribution (Fig. 6.9b) is only active for the areas using CNet and not those relying on the Skip mode. Furthermore, CNet still implements the conditional coding mechanism, allowing to further reduce the overall rate. Lastly, Fig. 6.9c presents the rate distribution of the MNet. Since it only conveys side-information (motion, coding mode), its rate is significantly lower than CNet rate. MNet usually represents less than 5 % of the overall rate.



(a) Mode selection $\boldsymbol{\alpha}$. *blue*: skip, *red*: CNet.

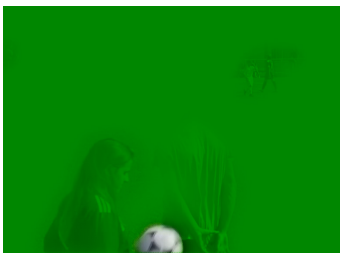


(b) CNet rate, 13 278 bits or 0.015 bit per pixel.



(c) MNet rate, 1 759 bits or 0.002 bit per pixel.

(d) Skip mode $(1-\boldsymbol{\alpha}) \odot \tilde{\mathbf{x}}_t$.



(e) CNet $c(\boldsymbol{\alpha} \odot \mathbf{x}_t, \boldsymbol{\alpha} \odot \tilde{\mathbf{x}}_t)$.

(f) Reconstructed frame $\hat{\mathbf{x}}_t$.

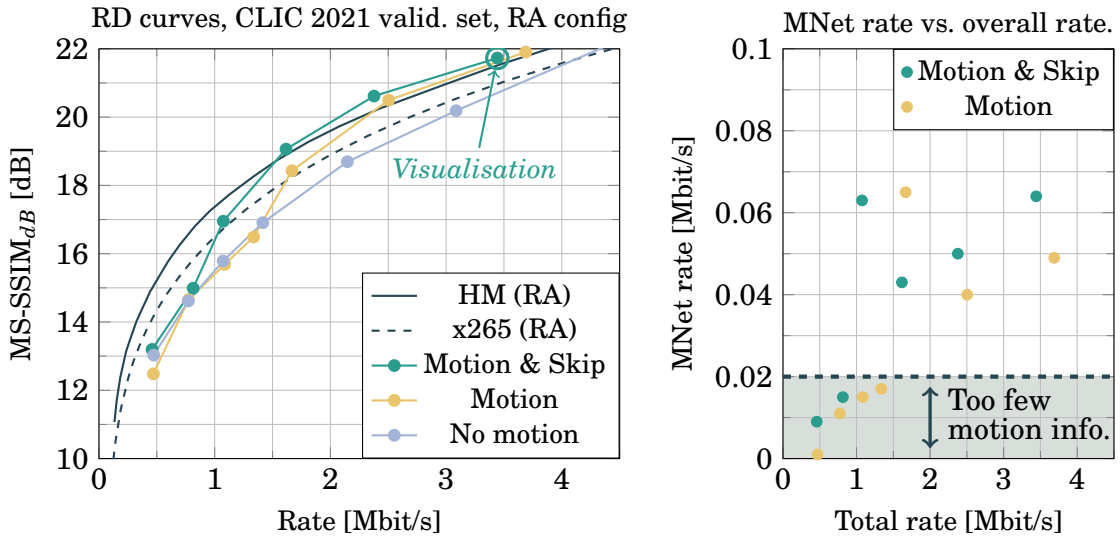
Figure 6.9: CNet and Skip mode contributions. This frame costs 15 037 bits, for a quality of MS-SSIM$_{dB}$ = 22.60 dB. The PSNR is equal to 38.46 dB

6.3.4 Rate-distortion results

The performance brought by the addition of motion information and the Skip mode is presented in Fig. 6.10a. Additionally, performance of systems featuring motion information but no Skip mode are presented. Recall that the absence of the Skip mode prevents an actual rate-distortion end-to-end training from scratch. In order to obtain *skipless* systems, pre-trained systems with motion information and Skip mode are used as starting points for the training process. Then, the Skip mode is disabled by forcing $\boldsymbol{\alpha} = 1$ (CNet only) and the resulting skipless system is finetuned by extending the training phase.

Throughout the evaluated rate range, systems with a Skip mode are equivalent or better than systems without one. This further justifies the presence of the Skip mode, which not only helps the convergence of motion information, but also improves the coding performance. This improvement is mainly explained by the fact that Skip mode takes the majority of the frame area, as shown in the preceding visual examples. This leaves CNet with only a small fraction of the frame, allowing it to be specialized for this smaller set of data through the learning of more suited parameters, which results in better performance.

At high rate, the presence of motion information significantly improves the compression



(a) Performance gain through the addition of motion information and Skip mode.

(b) MNet rate for different total rates.

Figure 6.10: Rate-distortion results for coding schemes featuring motion information.

performance, allowing to outperform the HM. This validates the accuracy of the optical flow, learned through a pure rate-distortion training. Yet, at low rate, the introduction of motion information does not lead to better results. A rationale for this behaviour is presented in Fig. 6.10b, which shows the rate dedicated to MNet as function of the overall rate. At low-rate, the training converges towards a rate distribution allocating virtually no rate to the MNet *i.e.* few information is transmitted about the optical flows, the coding mode selection and the bidirectional prediction weighting. Given our current MNet, this may be due to the fact that the motion information retrieved at the decoder is not accurate enough when operating at a low rate. Consequently, investing bits into motion information at low rate might lead to a worse rate-distortion trade-off.

To improve the performance of the proposed coding scheme, especially at low rate, the conditional coding mechanism is implemented for the MNet in the next section.

6.4 Conditional coding for the MNet

6.4.1 Principles

One of the important requirements in the design of the conditional coding mechanism was its flexibility. Namely, being able to use any decoder-side information to lower the rate

required to transmit any quantity from the encoder to the decoder. Here, we propose to implement conditional coding for MNet, with the intuition that the reference frames are capable to infer most of the motion information at the decoder-side. Indeed, having one past and one future reference allows to interpolate the motion information and thus to compute relevant optical flows.

Figure 6.11 presents the new MNet architecture, featuring the additional conditioning transform g_c^m. As for CNet, the conditioning transform g_c^m replicates the architecture of the analysis transform g_a^m with different number of input and output features. MNet conditioning transform aims to extract, from the two reference frames, relevant information regarding the motion, as well as $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Consequently, the MNet analysis g_a^m only transmits the information missing at the decoder, decreasing the rate. The addition of conditional coding for MNet increases the overall number of parameters from 36 million to 42 million.

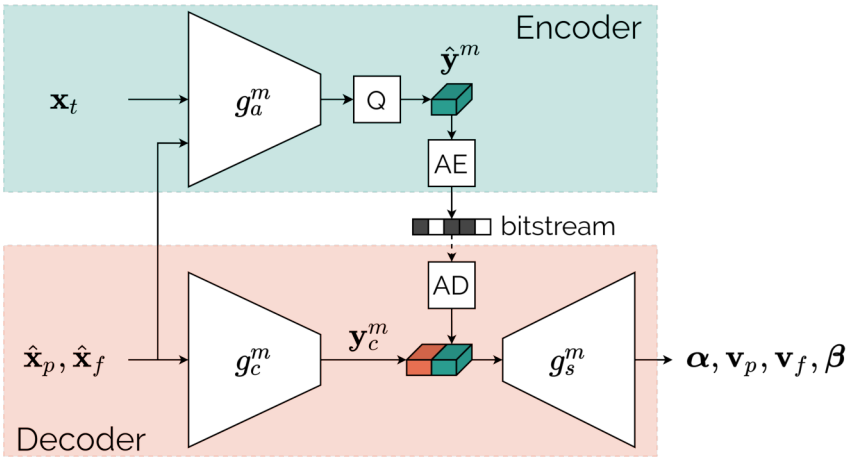


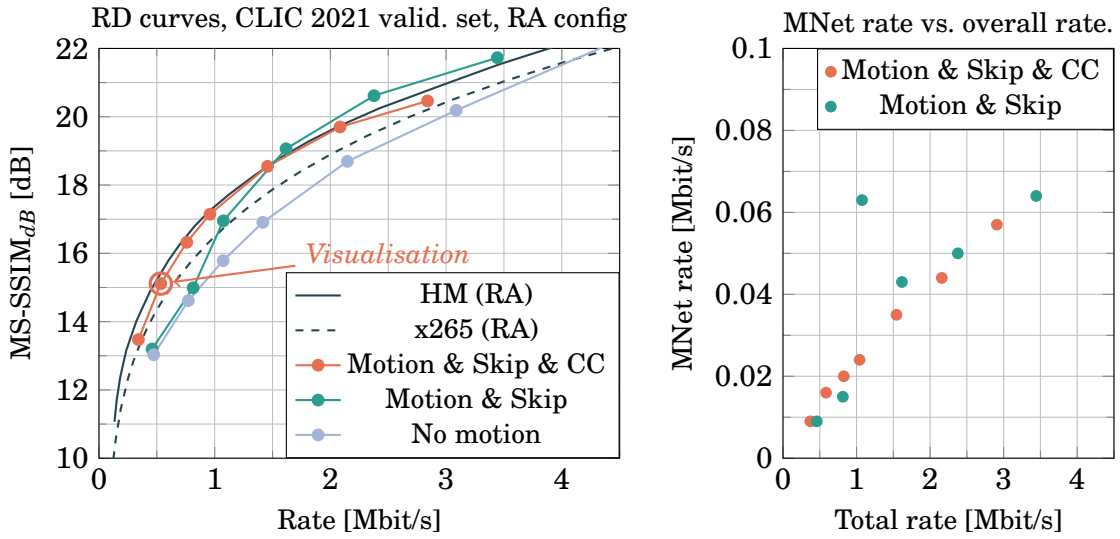
Figure 6.11: The conditional coding MNet architecture.

6.4.2 Training and rate-distortion results

The complete system, featuring conditional coding for CNet and MNet alongside the Skip mode is trained end-to-end. This yields the rate-distortion results presented in Fig. 6.12a. Adding the conditional coding mechanism to the MNet significantly enhances the compression performance at lower rate, *i.e.* lower than 2 Mbit/s. Thanks to conditional coding, MNet is able to compute relevant motion information at very low rate: even when MNet rate is around 10 kbit/s, it still significantly increases the rate-distortion results (see

Fig. 6.12b). Comparatively, systems without conditional coding for MNet are not able to improve their performance at low-rate, due to the rate constraint being too tight for MNet. Additionally, it seems that adding conditional coding to MNet, leads its rate to exhibit a more regular behaviour (Fig. 6.12b). This may hint that conditional coding makes the convergence of the system easier, which results in a more regular behaviour during the training stage.

However, conditional coding for MNet slightly decreases the performance at higher rate. In theory, this should not happen: if conditional coding becomes counter-productive, the training process should learn to disable the conditioning transform by always computing null conditioning latent variable. It is not the case in practice, leading to degraded compression performance at higher rate. Yet, we believe that conditional coding is a relevant technique to improve low-rate performance, while the higher-rate performance issue can be alleviated through a more suited or longer training policy.



(a) Performance gain through the addition of motion information and Skip mode.

(b) MNet rate for different total rates.

Figure 6.12: Conditional coding (CC) for MNet, rate-distortion results.

6.4.3 Visualisations

In order to illustrate the interest of conditional coding for MNet, a low-rate system (circled in Fig. 6.12a) compresses a video sequence. To understand the behaviour of MNet conditional coding, its output is computed while setting either the analysis or the

conditioning latent variable to zero. This ablation is identical to the one proposed in Chapter 5 for CNet, except that it now concerns MNet. The different configurations of this ablation are summarised in Table 6.1 and the results are presented in Fig. 6.13 in a B-frame coding context.

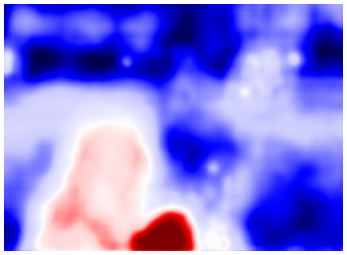
Table 6.1: Different configurations for the MNet conditional coding ablation.

	Latent variables		**Synthesis**
	Analysis	**Conditioning**	**operation**
Complete	✓	✓	$g_s^m(\mathbf{y}, \mathbf{y}_c)$
Conditioning only		✓	$g_s^m(0, \mathbf{y}_c)$
Analysis only	✓		$g_s^m(\mathbf{y}, 0)$

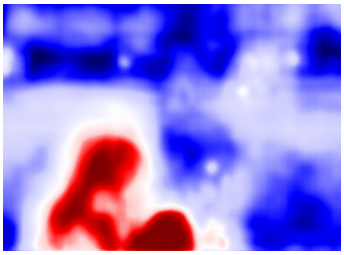
(a) Optical flow $\mathbf{v}_f$.

(b) $\mathbf{v}_f$, from conditioning latent $\mathbf{y}_c^m$ only.

(c) $\mathbf{v}_f$, from analysis latent $\mathbf{y}^m$ only.



(d) Mode selection $\boldsymbol{\alpha}$.



(e) $\boldsymbol{\alpha}$, from conditioning latent $\mathbf{y}_c^m$ only.

(f) $\boldsymbol{\alpha}$, from analysis latent $\mathbf{y}^m$ only.

Figure 6.13: Low-rate MNet conditional coding illustrated on a B-frame from the CLIC sequence *Sports_1080P-6710*. This frame requires 195 bits for MNet.

The first column presents two quantities: the optical flow $\mathbf{v}_f$ (Fig. 6.13a) and the coding mode selection $\boldsymbol{\alpha}$ (Fig. 6.13d). These quantities are obtained through the complete synthesis transform *i.e.* by feeding it with both the conditioning latent and the analysis latent variables. On the second column (Fig. 6.13b and 6.13e), the same quantities are presented when synthesized from the conditioning latent variable only. Conversely, the

last column (Fig. 6.13c and 6.13f) shows these quantities when synthesized from the analysis latent variable. The conditioning-only outputs are almost identical to the normal one, while the analysis-only outputs are virtually empty. This highlights that most of the relevant information is contained in $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_f$ and at the decoder-side without any transmission. Since no information is conveyed, the MNet rate is extremely low, see Fig. 6.12b. Yet, the motion information as well as $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ remain accurate enough, allowing to improve the rate-distortion cost even at low rate.

6.5 Comprehensive evaluation of the final system

6.5.1 Different coding configurations

Throughout this chapter, the proposed video coding scheme has undergone successive enhancements, improving the compression performance and leading to the final proposed system. It features two conditional coders, MNet and CNet, and an additional Skip mode. While most design choices have been evaluated on a Random Access configuration (Fig. 2.5a), the training stage of the systems prepares them to code I, P and B-frames. In order to thoroughly examine the performance of our learned coding scheme, it is evaluated against HEVC (HM 16.22) under three established coding configurations [17]:

- All Intra (AI): only I-frames are used;
- Low-delay P (LDP): P-frames only, except for the first frame;
- Random Access (RA): usual random access configuration with a gop size of 32 and an intra period of 32.

Figure 6.14 presents the rate-distortion results of our learned coding scheme, compared to those of the HM. For all evaluated coding configurations and across the entire rate range, the learned coding scheme achieves performance on par with the best implementation of HEVC, a modern video coder. These are compelling results, proving the flexibility and the relevance of the proposed coder. It is interesting to remark that due to the approximately symmetrical architecture of the coder, the encoding and decoding speed are comparable. For the 720p video sequences of the CLIC dataset, both the encoding and decoding run at a speed of a few frames per second.

It should also be noted that these results are obtained through training for a proxy problem. Indeed, the training stage consists in coding 3-frame video, similar to a Random

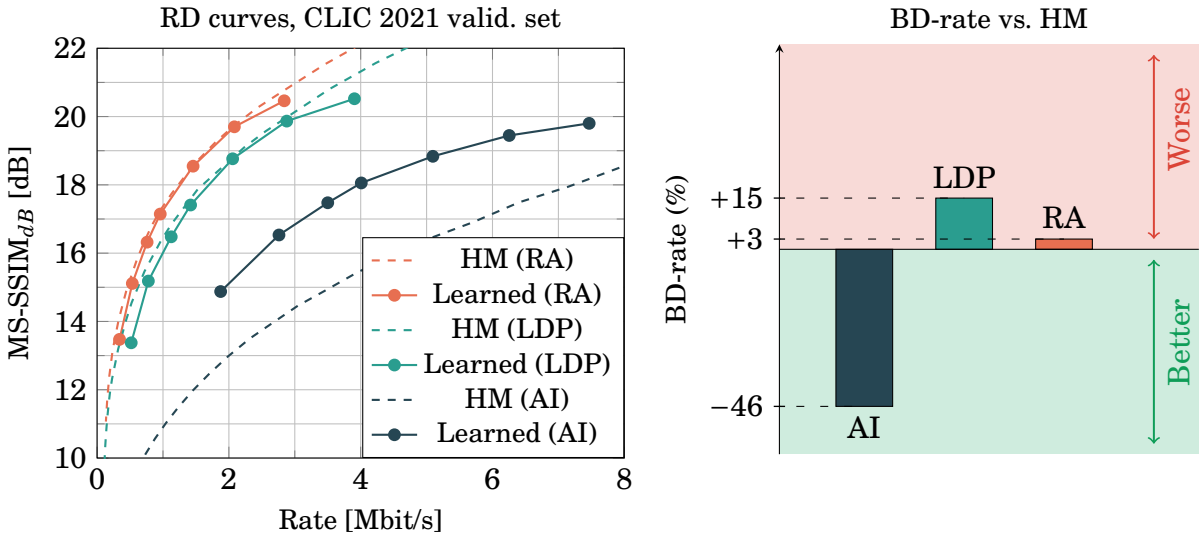


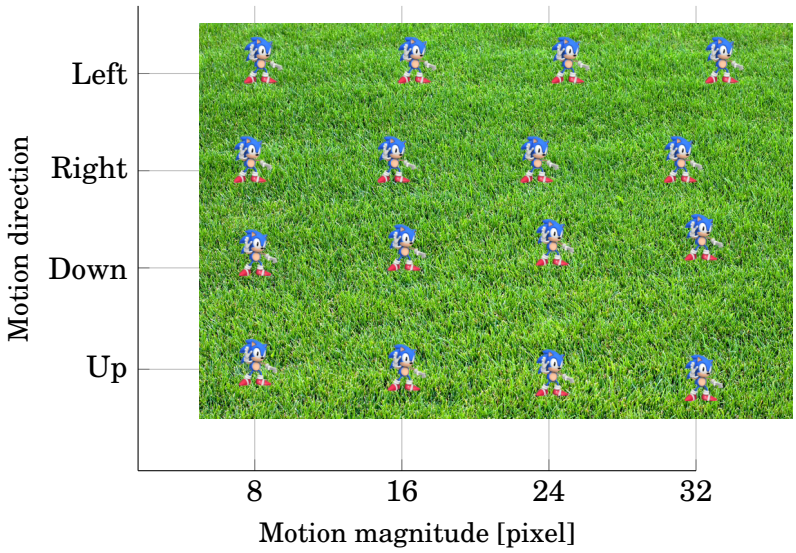
Figure 6.14: Rate-distortion results for different coding configurations.

Access configuration with a GOP size of 2. While it prepares the system for the processing of I, P and B-frames, it also specializes the coder for this peculiar coding configuration. Performing a training closer to the desired coding configuration would allow to improve the compression performance. However, this causes computational issues, as the training time is proportional to the GOP sizes.

Despite the overall encouraging performance of the learned coder, these results highlights some limitations of our coding scheme. On one hand, it significantly outperforms traditional methods for I-frame coding. One the other hand, when the coding configuration is also composed of P or B-frames, it simply achieves results on par with traditional methods. This means that the learned coder has a significant advantage on I-frame coding, but lose some of this benefit on P and B-frame coding. It might be due to weaknesses in the motion estimation and transmission process.

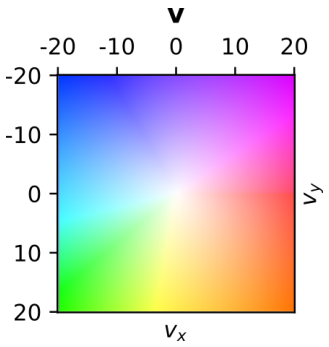
6.5.2 MNet limitations

This section aims to evaluate the ability of MNet to estimate and convey relevant motion information, in order to discover its potential limitations. To this effect, a synthetic video sequence is introduced and shown in Fig. 6.15a. It features a 4×4 grid of objects, moving alongside 4 directions, with a motion magnitude of 8, 16, 24 and 32 pixels. For a 1280×720 video sequence at 30 frames per second, such motions correspond to an object crossing the entire image (horizontally) in 5.3, 2.7, 1.3 and 0.7 seconds.



(a) Motion of $\mathbf{x}_t$ relatively to $\hat{\mathbf{x}}_p$.

(b) Optical flow $\mathbf{v}_p$.

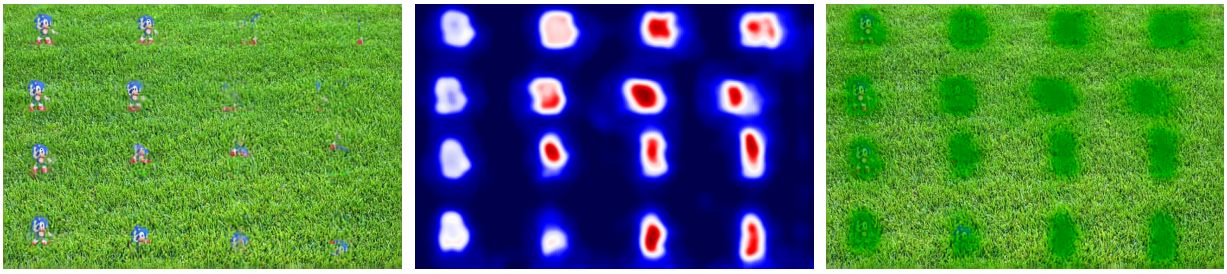


(c) Legend for the optical flow representation. Motion is expressed in pixel.

Figure 6.15: Example of an inaccurate optical flow.

Figure 6.15b presents the optical flow obtained at the decoder *i.e.* after estimation and transmission through the MNet. For the 8-pixel motions, the resulting optical flow is accurate, regardless the motion direction. When motion amplitudes become more important, *e.g.* a 16-pixel motion, the optical flow is properly estimated for horizontal motions, while vertical ones are no longer captured by the MNet. The system is not able to compute any optical flow for motion of 24 pixels or more. Finally, the edges of the optical flows are less accurate when the motion magnitude becomes more important.

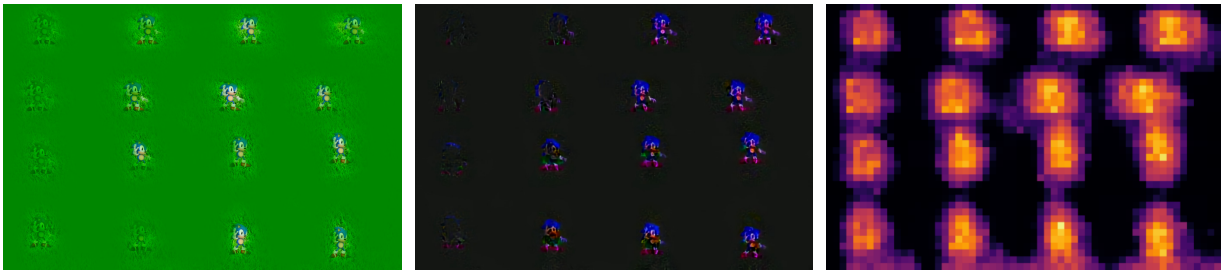
This weakness regarding the estimation and transmission of important optical may be caused by several factors. First, important motions are not often found in video sequences and therefore in the training set. The lack of relevant example prevents the system from learning to identify important motion. Then, the proposed architecture for MNet is mostly derived from an autoencoder. Integrating components from the optical flow estimation literature such as cost volume elements could lead to better results.



(a) Temporal prediction $\tilde{\mathbf{x}}_t$.

(b) Mode selection $\boldsymbol{\alpha}$. *blue*: skip, *red*: CNet.

(c) Skip mode contribution $(1-\boldsymbol{\alpha}) \odot \tilde{\mathbf{x}}_t$.



(d) CNet contribution $c(\boldsymbol{\alpha} \odot \mathbf{x}_t, \boldsymbol{\alpha} \odot \tilde{\mathbf{x}}_t)$.

(e) CNet output, analysis latent $\mathbf{y}_c$ only.

(f) CNet rate, total is 78 293 bits.

Figure 6.16: Consequences of an inaccurate prediction. This frame costs 80 954 bits, for a quality of MS-SSIM$_{dB}$ = 10.65 dB. The PSNR is equal to 20.80 dB

A poorly estimated motion leads to an irrelevant temporal prediction (Fig. 6.16a). The effect of a not accurate prediction is two-fold.

- Skip mode can not be used since the direct copy of a degraded prediction does not result in acceptable quality. This is illustrated by the coding mode $\boldsymbol{\alpha}$ in Fig. 6.16b, and by the Skip mode contribution in Fig. 6.16c. Consequently most of the frame is transmitted through CNet (see Fig. 6.16d), leading to an increased rate.

- The conditional coder CNet can not rely too much on the conditioning transform, as the conditioning input (*i.e.* the prediction) is not accurate enough. As such, most of CNet output comes from the analysis latent variable, that requires an actual transmission (see Fig. 6.16e), further increasing the rate. This can be thought of as intra coding: no relevant information is available in the prediction, resulting in the complete transmission of the image.

Finally, Figure 6.16f presents the spatial distribution of the rate. The right side of the image (*i.e.* the areas with bigger motion) appears brighter, highlighting the relation between an inaccurate temporal prediction and an increased rate.

6.6 Conclusion

In this chapter, we have enhanced the initial coding scheme composed of a conditional coder CNet, through the introduction of several motion-based components. An additional autoencoder MNet is added to estimate and convey motion information, allowing to achieve a relevant temporal prediction. In order to both foster the convergence and increase the performance, an additional coding mode called Skip mode is proposed. Finally, compression efficiency is improved by implementing MNet with the conditional coding architecture.

Table 6.2 summarises the rate spared through the introduction of these different components. The coding scheme featuring motion information and Skip mode through MNet with the conditional coding architecture achieves a BD-rate of -33.9 % compared to the initial motionless system. That is, it obtains equivalent quality requires 33.9 % less rate. Furthermore, the proposed learned video coder achieves a BD-rate of $+3.1$ % against the HM *i.e.* the best implementation of HEVC available. In the end, we have designed a learned video coder, competitive with HEVC under different coding configurations. This demonstrates the relevance of the design choices made for our learned coder.

Despite these compelling results, some elements of the coding scheme could be improved. Based on the performance gap between all intra coding and the other coding configurations, we believe that MNet is the main weakness of the coding scheme. Obtain-

Table 6.2: BD-rates of the different coding schemes in Random Access coding configuration. CC denotes conditional coding.

	MNet	Skip mode	MNet CC	BD-rate	
No motion				Reference	+42.9 %
Motion	✓			−7.6 %	+24.4 %
Motion & Skip	✓	✓		−17.5 %	+14.7 %
Motion & Skip & Cond.	✓	✓	✓	−33.9 %	+3.1 %
x265 medium†		/		−14.5 %	+22.8 %
HM†		/		−37.2 %	Reference

†: *evaluated at QP 22, 27, 32, 37.*

ing better motion information, by using more suited architecture inspired from the motion estimation literature, would likely improve the overall coding performance.

Skip mode was initially introduced for performance purpose within a motionless coding scheme in a conference paper presented at the *IEEE International Workshop on Machine Learning for Signal Processing* (MLSP) 2020 [18]. A following paper, further justified skip mode to allow for an easier learning of motion information. This paper was presented at the *IEEE International Workshop on Multimedia Signal Processing* (MMSP) 2020 [26]. Finally, the implementation of MNet with conditional coding as well as the flexibility of the overall coder for different coding configuration was presented at the *Neural Compression Workshop, International Conference on Learning Representations* (ICLR) 2021 [76].

CHAPTER 7

TOWARDS PRACTICAL VIDEO CODING

7.1 Introduction

THE *Challenge on Learned Image Compression* (CLIC) 2021 [16] was organized as a workshop of the IEEE *Conference on Computer Vision and Pattern Recognition* (CVPR) 2021. For the first year, a video coding track was proposed in order to compare learned and traditional coders. The learning-based coder finalized in the previous chapter, was submitted to the challenge to assess its performance.

Besides the evaluation of our system against other learned coders, this challenge highlights the importance of having a flexible coder. Indeed, due to the challenge rules, using the optimal coding strategy for each video sequence improves the overall performance. To this end, this chapter proposes different dimensions of coding strategies competition.

7.2 CLIC21 video track

7.2.1 Challenge presentation

The objective of this challenge is to obtain the highest quality (measured with the MS-SSIM) while compressing 100 high-resolution (720p) video sequences, with a frame rate of 30 frame per second. The overall bit budget given for the challenge is:

$$\text{decoder size} + \frac{\text{data size}}{0.019} \leq 1\,309\,062\,500 \text{ bytes}. \tag{7.1}$$

Since each of the 100 video sequences lasts two seconds, the rate constraint corresponds approximately to an *average* rate of 1 Mbit/s. For convenience, the allocated bit budget constraint is turned into a rate target R_T:

$$R_T = R_{dec} + R_{data}, \text{ with } R_T \simeq 1 \text{ Mbit/s}. \tag{7.2}$$

To discourage the participants to come with huge neural network models, the decoder size is taken into account in the bit budget. For instance, one decoder of the final model proposed in Chapter 6 has around 27 million parameters. As such, it requires 108 MBytes of storage. This yields $R_{dec} = 0.08$ Mbit/s lowering the data rate to $R_{data} = 0.92$ Mbit/s: the impact with respect to the bitrate efficiency (*e.g.* BD-rate) is in the range of 8 %.

7.2.2 Sequence-wise competition

The challenge rules do not impose sequence-wise constraints concerning the rate or quality of each individual sequences. Thus, it is possible to improve the overall results through a relevant sequence-wise quality selection. For instance, sparing a few bits on a sequence might lead to a minimal quality loss while investing these bits on some other sequence dramatically improves its quality. This rationale justifies the competition between different coding strategies for each sequence, in order to optimize an overall rate-distortion cost.

To formalize this problem, let us denote $J_\lambda(\mathbf{v}_i;\mathcal{C})$ the rate-distortion cost of the video sequence $\mathbf{v}_i$ compressed using the coding strategy $\mathcal{C}$:

$$J_\lambda(\mathbf{v}_i;\mathcal{C}) = D(\mathbf{v}_i;\mathcal{C}) + \lambda R(\mathbf{v}_i;\mathcal{C}). \tag{7.3}$$

Here D and R denote the distortion and the rate associated to the current video and coding strategy. For each video sequence $\mathbf{v}_i$, the objective is to find the optimal coding strategy $\mathcal{C}_i^\star$, yielding the minimal rate-distortion cost:

$$\mathcal{C}_i^\star = \underset{\mathcal{C}}{\arg\min}\ J_\lambda(\mathbf{v}_i;\mathcal{C}). \tag{7.4}$$

Searching the best coding strategy for all the N video sequences of the dataset allows to obtain the optimal average rate $\overline{R}$ and distortion $\overline{D}$:

$$\overline{R} = \frac{1}{N}\sum_{i=1}^{N} R(\mathbf{v}_i;\mathcal{C}_i^\star) \ \text{ and } \ \overline{D} = \frac{1}{N}\sum_{i} D(\mathbf{v}_i;\mathcal{C}_i^\star). \tag{7.5}$$

The value of the rate control parameter λ is set empirically so that the resulting rate $\overline{R}$ respects the desired rate target.

We propose to perform coding strategies competition using the final system designed in the previous chapter (*i.e.* with Skip mode and conditional coding for MNet and CNet). Recall that several instances of the proposed learned coder have been trained for different

rate target (*i.e. different* λ). There are two different means of varying its coding strategy:

- Tuning the coding structure *i.e.* the organization of the I, P and B frames. Since the systems are prepared to process I, P and B-frames, they can be arranged in any desired order, enabling the compression of the video using any frames structure;
- Tuning the rate of the compressed video, by choosing one coder-decoder pair among the different systems available.

So far, a learned coder is *mono-rate i.e.* it is only able to target one specific rate per sequence. Indeed, it always performs the same operations for a given input, leading to the same quantization accuracy and the same data to be transmitted. Therefore, the competition between N different rates requires to store the parameters of N different mono-rate decoders, resulting in a overall coding scheme with N times more parameters.

7.3 Rate competition

7.3.1 With mono-rate coders

There is a great diversity of video sequences within the CLIC dataset. Some are almost motionless, some exhibit few textured areas while others feature important motions or many high-frequency contents. This results in a wide variety of rate-distortion costs, presented in Fig. 7.1. The important rate range advocates for rate competition, as a mono-rate coder can not be optimal for all the sequences. It is possible that the sequences far from the average rate-distortion point of the coder would benefit from being compressed with an other coder (*i.e.* at an other rate-distortion point) in order to improve their quality (for low-rate sequences) or reduce their rate (for high-rate sequences).

Rate competition is implemented through the coding of all the sequences using all the previously trained coders, shown in Fig. 7.2a. Seven different mono-rate coders are available, leading to seven coding choices for each sequence. The best coding strategy (*i.e.* the best rate target out of the seven available) is chosen sequence-wise according to Eq. (7.4). Adjusting the rate control parameter λ allows to obtain the *7-rate competition* curve, which presents a clear improvement over the absence of competition.

Yet, the performance improvement yielded by rate competition comes at the cost of an increase in the overall decoder size. Indeed, the rate competition between N decoders requires a decoder rate $R_{dec} = N \times 0.08$ Mbit/s which contributes significantly to the

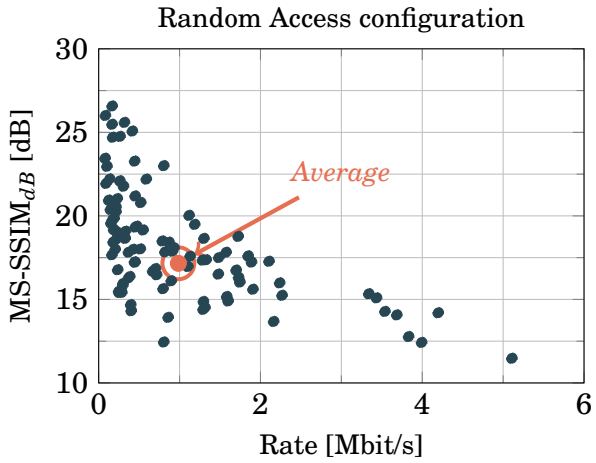
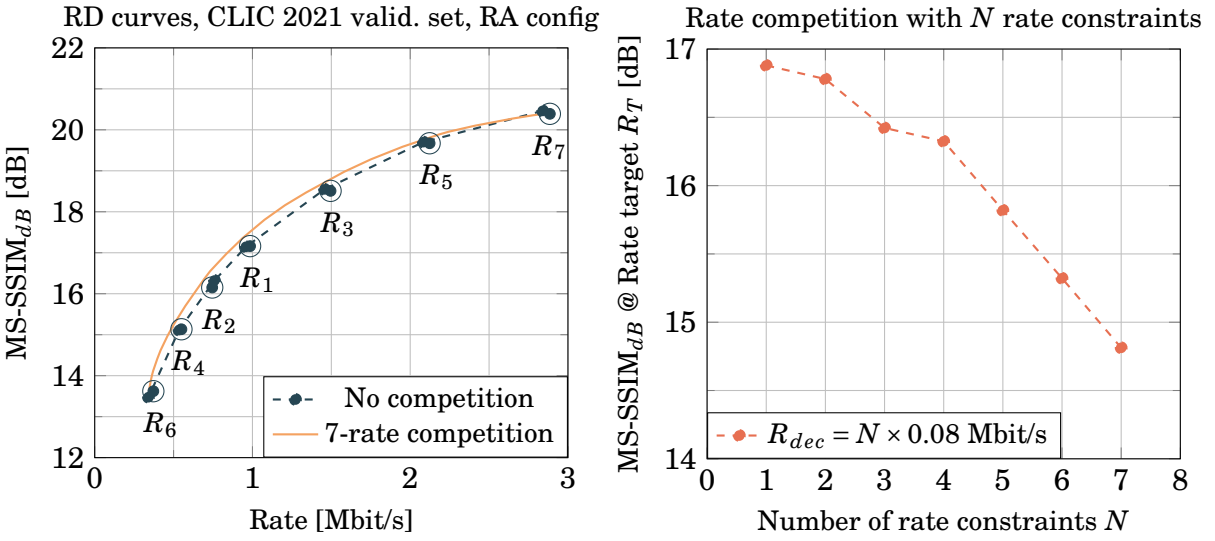


Figure 7.1: Sequence-wise rate-distortion for a low-rate learned encoder-decoder pair.

overall rate target or 1 Mbit/s, decreasing the amount of data available for the video sequences. Figure 7.2b presents the actual benefit of rate competition by measuring the quality obtained through the competition of N mono-rate coders, when accounting for the decoders size. The order in which the different decoders are used is indicated in Fig. 7.2a. For instance, the competition between $N = 3$ coders relies on the systems whose rate is R_1, R_2 and R_3.



(a) Performance of rate competition.

(b) Influence of the model size on the rate target.

Figure 7.2: Sequence-wise rate competition with mono-rate coders.

While rate competition does bring some performance improvement, this benefit is

cancelled by the significant amount of additional parameters required. To fully benefit from rate competition, the number of additional has to be mitigated in order to be competitive according to the challenge rules.

7.3.2 Design of a multi-rate coder

In order to benefit from rate competition between N coders, the size of those systems must remain small. We propose to transform a mono-rate system into a multi-rate one, with a negligible increase in the number of parameters. For all rates addressed by the multi-rate system, the main components of the coder remains the same. That is, MNet and CNet transforms (analysis, synthesis, conditioning and hyperpriors) remain identical, regardless of the rate target. In the proposed multi-rate system, the single operation conditioned on the rate target is the CNet quantization gains. In Section 5.6, quantization gains were used to achieve different quantization accuracies according to the type of the frame (I, P and B). To obtain multi-rate systems, CNet quantization gains are now also conditioned on the rate constraint, as proposed in [74]. In order to address N rate constraints, $3 \times N$ pairs of encoder-decoder quantization gains are learned:

$$\Gamma^{enc}_{f,i}, \Gamma^{dec}_{f,i} \text{ with } \begin{cases} f \in \{I,P,B\} & \text{the frame type,} \\ i \in \{1,\dots,N\} & \text{the rate constraint index.} \end{cases} \tag{7.6}$$

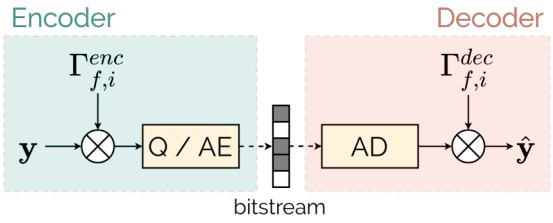


Figure 7.3: Principles of CNet multi-rate quantization gains.

Each quantization gain is a feature-wise gain *i.e.* $\Gamma_{i,f} \in \mathbb{R}^{F_y}$, with $F_y = 256$ the number of feature maps of CNet analysis latent variable $\mathbf{y}$. They are applied to $\mathbf{y}$ through feature-wise multiplication, as presented in Fig. 7.3. Quantization gains allow to obtain additional rate targets available for a minor increase in the overall number of decoder parameters. Let us denote $M \simeq 27 \times 10^6$ the number of parameters for the decoder of a mono-rate system. The number of parameters required to address $N = 6$ different rates becomes:

$$\textit{(multi-rate)} \quad M + \overbrace{N \times 3 \times 256}^{\text{quantization gains}} \ll M \times N \quad \textit{(mono-rate)} \tag{7.7}$$

Each additional rate constraint adds only $3 \times 256 = 756$ parameters. In the end, addressing N different rates is achieved with virtually N times less parameters thanks to multi-rate systems.

7.3.3 Training

With multi-rate coders, the training objective is to prepare the coder to compress I, P and B frames, under different N rate constraints $\{\lambda_1, \ldots, \lambda_N\}$. As for mono-rate systems, the optimization is performed according to a rate-distortion criterion and similarly, each training example is a 3-frame videos compressed using an IBP configurations. In order to learn the different quantization gains, each training iteration is composed of the following steps:

1. Randomly draw a rate constraint index $i \in \{1, \ldots, N\}$;

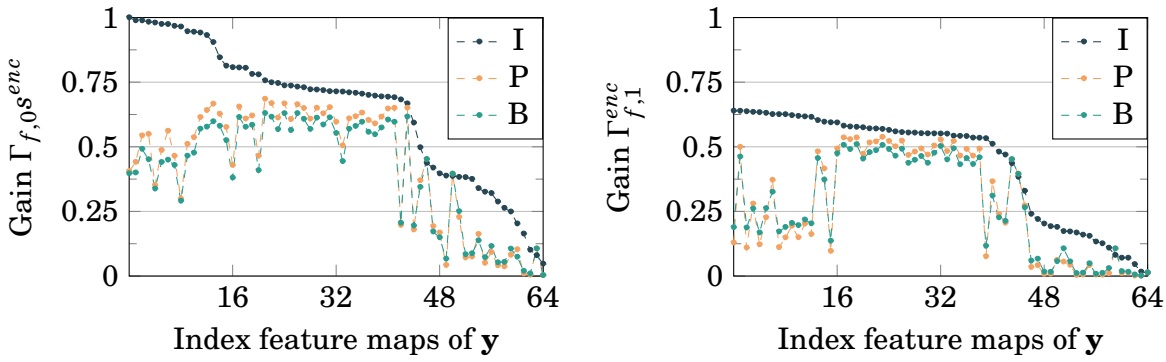
2. Process example with the corresponding quantization gains $\Gamma^{enc}_{f,i}, \Gamma^{dec}_{f,i}$ (and the other parameters common to all rate constraints);

3. Gradient descent to minimize the corresponding rate-distortion cost: $D + \lambda_i R$.

Figure 7.4 presents the encoder quantization gains obtained for two rate constraints. Similarly to mono-rate coders, the encoder gains dedicated to intra frames are more important which translates into an improved quantization accuracy for the intra frames. As expected, when the rate constraint becomes tighter, the encoder quantization gains decreases leading to a less accurate quantization, see Fig. 7.4b.

Moreover, once the training stage is achieved, the resulting coder is able to compress a video sequence at any *continuous* rate target, which conveniently allows to freely set the rate of a sequence. This is achieved through a geometric averaging of the quantization gains. Let us suppose that one wants to address the continuous rate constraint $r \in [1, N]$. In such case, the encoder and decoder quantization gains are expressed as follows:

$$\Gamma_{f,r} = \left(\Gamma_{f,\lfloor r \rfloor}\right)^{1-l} \odot \left(\Gamma_{f,\lceil r \rceil}\right)^{l}, \text{ with } l = r - \lfloor r \rfloor. \tag{7.8}$$



(a) Rate constraint λ_0 (higher rate).

(b) Rate constraint λ_1 (lower rate).

Figure 7.4: Encoder quantization gains for two rate constraints.

7.3.4 Rate-distortion results

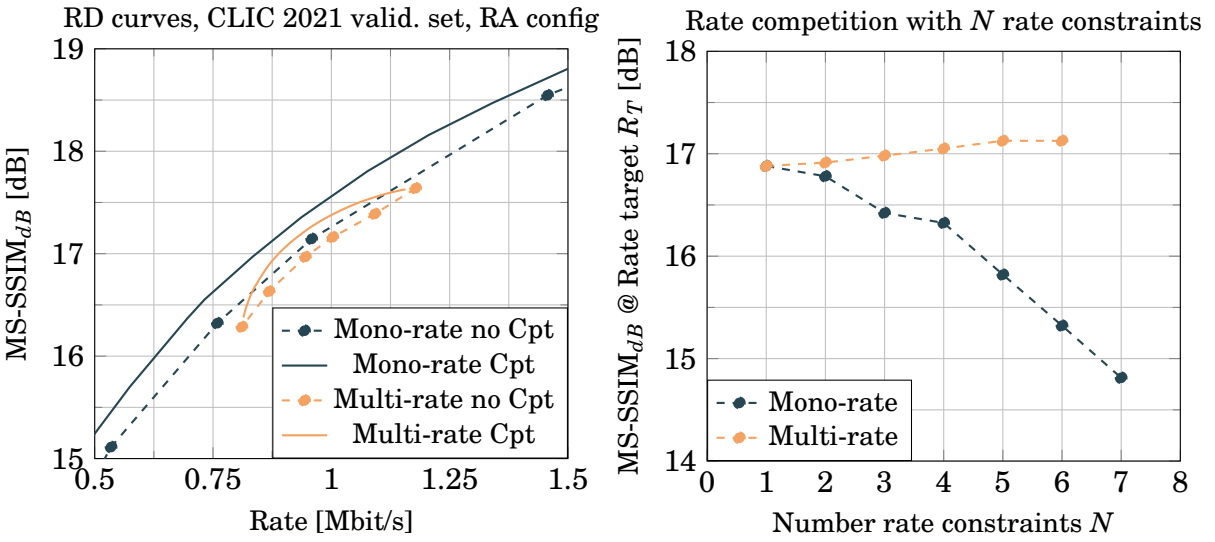
To evaluate the benefits of multi-rate systems, the CLIC dataset is compressed, with and without rate competition. The results are presented in Fig. 7.5a. The dashed curves indicate the performance of the systems without rate competition. While the blue one is obtained through the training of multiple mono-rate coders, the orange curve result from a single multi-rate system. While the multi-rate system does not cover a rate range as wide as the mono-rate ones, its performance remains competitive around the rate target of 1 Mbit/s. Activating rate competition, the multi-rate system reaches performance competitive with the mono-rate ones, while having a significantly smaller decoder.

Finally, Fig. 7.5b presents the quality at the challenge rate target R_T, for mono-rate and multi-rate systems with competition. As explained before, rate competition with mono-rate system is not a relevant choice as the size of the additional decoders exceeds the benefit of rate competition (the blue curve on the graph). This is not the case for a multi-rate system, where the additional parameters is negligible, making possible to actually benefit from rate competition. Rate competition using multi-rate system improves the quality obtained at the challenge rate by +0.26 dB.

7.4 Participation to the CLIC21 video track

7.4.1 Proposed systems

Our submission to the CLIC challenge was a multi-rate system, with rate competition to achieve better compression performance. By design, this system is able to process I, P and



(a) Performance of rate competition.

(b) Influence of the decoder size on the rate target.

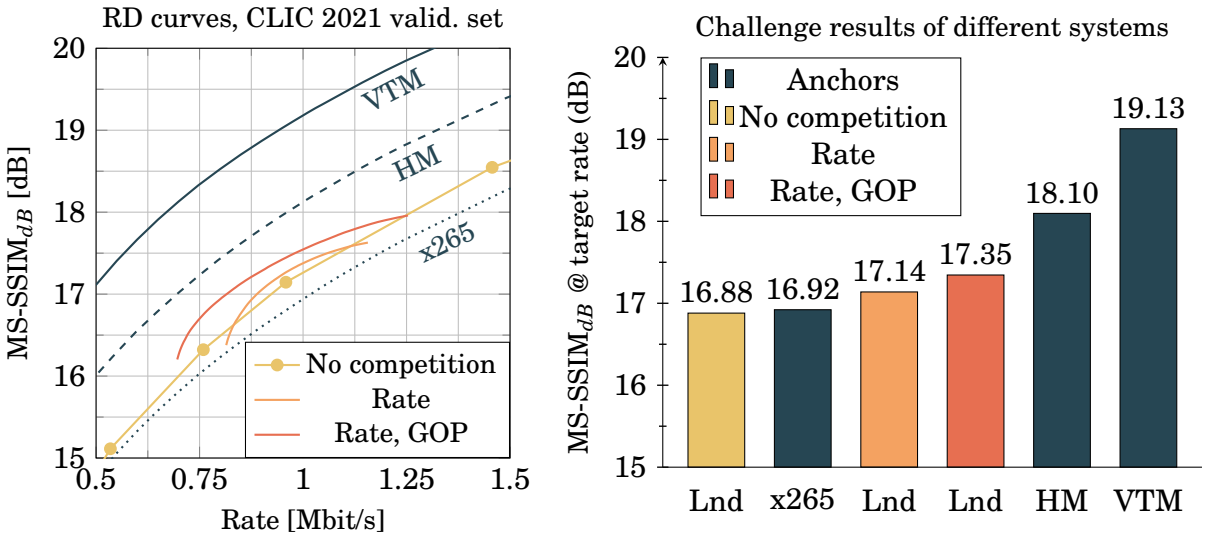
Figure 7.5: Sequence-wise competition for a multi-rate system. Cpt stands for competition.

B-frames. As such, it is possible to organize these types of frame as desired to achieve any coding structure. This is relevant, due to the wide variety of motion exhibited by the CLIC videos. Indeed some videos with important motions do not benefit from being coded using distant reference frames. For such sequences, using a coding structure with a smaller GOP size mitigates the prediction issues and improves the compression performance.

Coding structure competition is performed sequence-wise and the optimal structure is chosen according to Eq. (7.4). Since the encoding and decoding speed of the learned coder remains reasonable (a few frames per second for a 720p video sequence), it is possible to test a great variety of coding configurations and rates. To assess the performance of the proposed system, traditional coders are evaluated. In addition to HEVC implementations (x265 medium and HM), VTM (test model of VVC) is also tested. For fair comparison, the traditional anchors also performs sequence-wise rate and coding structure competition. Figure 7.6a shows the different rate-distortion results and Fig. 7.6b presents the MS-SSIM obtained by each system at the challenge rate target.

The interest of competition is clearly illustrated in these rate-distortion results. Each additional competition (rate, coding structure) leads to further performance improvement, yielding an overall increase of +0.47 dB in quality. The resulting learned coder, with full competition, achieves performances better than x265 medium but worse than the HM.

In Chapter 6, our learned coder has been shown to be on par with the HM, which is no



(a) Different coders with competition.

(b) Challenge results. Lnd stands for learned coder.

Figure 7.6: Performance of different systems with a focus on the quality at the challenge target rate. *GOP* stands for coding structure competition.

longer the case with the challenge results presented here. This can be explained by two main factors.

- Traditional coders (HM, VTM) feature an important number of different coding strategies: different rate constraints, coding configurations, dedicated tools for certain contents. This makes them particularly suited for sequence-wise competition.

- The size of traditional decoder is significantly smaller than the learned one, requiring around 1 MByte of storage (against 100 MBytes for learned decoders). As such, the decoder rate is negligible, leaving $R_{data} \simeq 1$ Mbit/s for the video sequences (against $R_{data} = 0.92$ Mbit/s for learned decoders).

7.4.2 Challenge results

Due to its compelling performance, we submitted the VTM alongside the learned coder to the CLIC video track. The objective of the VTM submission is to represent the state-of-the-art in traditional video coding.

In order to ensure cross-platform arithmetic encoding and decoding, it is not straightforwardly possible to use the hyperprior mechanism to describe the latent variable probability distribution. At that time, the proposed coder had to be slightly degraded by using a fixed

probability distribution instead of the hyperprior mechanism. Additionally, decreasing the number of internal convolutional feature maps from 192 to 128 resulted in a better trade-off between the model size and the compression performance. This leads to the following proposals the CLIC video track, summarised in the Table 7.1 below.

System	Submission name	MS-SSIM$_{dB}$ [dB]	Decoder size [MBytes]
VTM	ANC_T_OL	19.13	1
Learned (128 ft., fixed PDF)	E2E_T_OL	15.53	43
Learned (192 ft., HP)	Not submitted	17.35	108

Table 7.1: CLIC21 leaderboard results validation set. ft. stands for feature maps.

Coders from 11 different participants were proposed to the challenge. Among them, 6 are based on traditional video coders, with some additional learning-based enhancements (mostly post-processing filters). The other 5 systems are end-to-end learned coders. To prevent overfitting from the learned systems, a new dataset is used to determine the challenge winner. Figure 7.7 presents the definitive results of the CLIC 2021 video track.

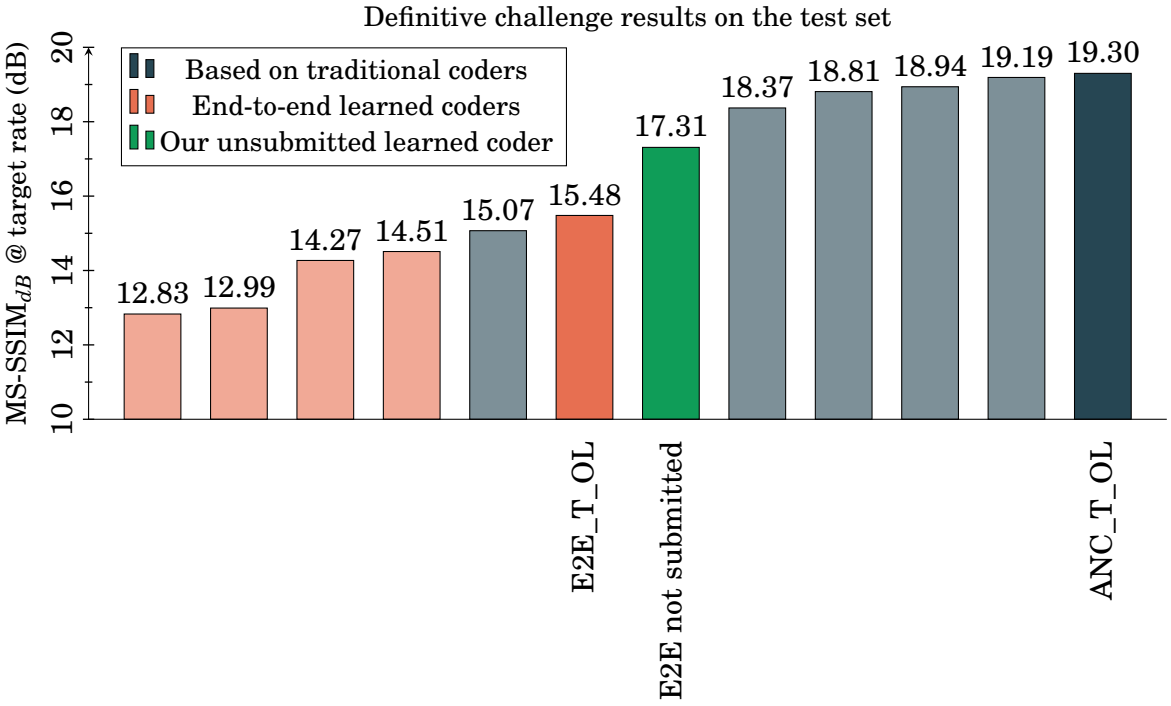


Figure 7.7: Challenge test set leaderboard. The two named systems are our submissions.

The challenge results gives several insightful information regarding learned video coding. First of all, submissions based on traditional coders significantly outperform learning-based ones. Indeed, the winner of the challenge is the VTM-based approach we submitted, *without learned component*. Among the end-to-end category, our submission obtains the best performance, even though its performance is altered due to the removal of the hyperprior mechanism. The actual results of our best learned coder are shown in green, even though it was not submitted to the challenge. The design choices of this thesis are thus further justified, since it leads to state-of-the-art a learned video coder.

7.5 Conclusion

This chapter has strived to address a more practical video coding situation through the participation of a compression challenge. Due to wide variety of video sequences, we have proposed to resort to coding strategies competition, namely rate and coding structure competition. Furthermore the decoder size constraint has brought us to come up with multi-rate systems in order to benefit from rate competition while avoiding an increase in the decoder size. Thanks to its enhanced flexibility, the proposed coder obtains compelling results, being ranked first among all the learned systems submitted to the challenge. More importantly, this chapter has shown than the learned coder can be configured to achieve different coding strategies. This is an important step towards a more practical use case of video coding, where real-life constraints (latency, intra period, rate) are more prominent.

This challenge is a snapshot of the video coding landscape in 2021. As all systems are evaluated using an identical method, results are more comparable than self-reported performance under different test methods, as it is often the case in the literature. While learned coding performance is continuously and quickly improving, traditional coders such as the VTM remains more competitive. So far, traditional coding systems achieve better performance and present more flexibility due to their many coding strategies available. For instance, the VTM is able to achieve state-of-the-art performance on a wide-range of rate. This remains a challenge for multi-rate learned systems.

Through the limited scope of this challenge, the learned multi-rate system meets most of our expectation, as it allows to use rate competition. In a more realistic setup, it is desirable to address a larger range of rate using a multi-rate system. Empirically, the quantization gains conditioned on the rate constraints does not seem to be powerful enough to operate at widely different rate targets. We believe that other elements of our

system would have to be conditioned on the rate constraint in order to obtain a satisfactory multi-rate system. In traditional video coding, the quantization gains is one of the many operations (partitioning, prediction mode *etc.*) conditioned on the rate target.

Conclusions and Future Work

Thesis objectives

The ever-increasing consumption of images and videos has to be met with improved compression efficiency to mitigate the environmental impact of the infrastructures required to convey them. To this end, this thesis proposes to study the design of end-to-end learned video coding schemes. While the tools and techniques required to obtain state-of-the-art learned image coders are known and presented in Chapter 3, learned video coding remains a more challenging task. So far, existing coders from the literature underperform compared to traditional coders such as HEVC and VVC. Furthermore, learned coders often exhibit less flexibility compared to traditional coders. For instance, they feature a constrained frame organization or they require a dedicated intra frame coder.

Given the novelty of learned video coding, this thesis has reconsidered the entire design of a learned video coder. For instance, the computation of a temporal prediction and the exploitation of this prediction are thoroughly investigated to incrementally build a learned coder. The design process is carried out with the objective of obtaining a practical video coder, competitive with traditional approaches when evaluated under different coding configurations and rate constraints.

Learned compression implements the usual compression techniques (entropy coding, prediction, transform) with neural networks. In this context, the benefits of neural networks are two-fold. First, they are able to perform non-linear operations achieving better results than the linear ones of traditional coders. Second, the different steps of the compression process can be jointly optimized through an end-to-end learning process, allowing to further improve the performance.

Learned image coding is tackled in Chapter 3 as the first step towards the design of learned video coders. The different elements of the compression pipeline are motivated from an information theory standpoint (Chapter 1) and from their usage in traditional coders (Chapter 2). The learned image coder proposed in this thesis achieves performance competitive with the image coding configuration of VVC, the latest video coding standard.

The handling of the temporal dimension is discussed in Chapter 4, leading to a two-step coder. A temporal prediction of the current frame is first computed through a motion compensation process, based on previously received frames. Then, the prediction is exploited to only convey the unpredicted part of the current frame. In both traditional and learned coders from the literature, this is often achieved through residual coding. Chapter 5 introduces conditional coding to better leverage the prediction. It is designed as a generic technique to use any decoder-side information available. Conditional coding is implemented based on components from the learned image coding literature and reduces the rate by 30 % compared to residual coding. Unlike most learned approaches from the literature, conditional coding prevents the need for a dedicated intra frame coder, leading to a more factorized system.

Motion compensation is used to compute a relevant temporal prediction. As such, the motion between successive video frames has to be accurately estimated and efficiently transmitted. We propose to perform this two tasks through a single autoencoder. To foster the convergence of this additional encoder a coding mode is introduced: the Skip mode which consists in a direct copy of the temporal prediction. Unlike most existing coding schemes, the addition of the Skip mode allows to learn the overall coding scheme through an end-to-end rate-distortion training. Finally, conditional coding is implemented to reduce the motion information rate. The resulting video coder achieves performance competitive with the best implementation of HEVC under various coding configurations and rate constraints.

Finally, the proposed coder is assessed through a participation to the video track of the CLIC 2021 challenge. The coder flexibility is demonstrated by performing sequence-wise competition of coding structures and rate targets, in order to improve the average performance on the entire test set. Due to a specific challenge rule on the decoder size, a multi-rate system is introduced. It allows to target different rates for a video sequence at the cost of few additional parameters. The resulting coder was submitted to the challenge and obtained the best score among the end-to-end learned coders.

Additionally, we also proposed a traditional solution for the CLIC 2021, which won the challenge. This shows that in 2021, traditional coders remains more performant than learning-based ones when evaluated in a realistic video coding situation. Nevertheless, our proposed coder is competitive with HEVC, demonstrating the practicability of learned video coding. Regarding the speed of improvements for learning-based coding schemes, there is no doubt that these approaches will soon outperform traditional ones.

Future works

Further performance improvement

This thesis presents evidence of the neural networks ability to outclass handmade operations of traditional coders. For instance, this is the case in image coding where learned approaches already achieve state-of-the-art results, or for conditional coding which outperforms residual coding. Yet, learned approaches seem to struggle for one key step of the coding pipeline: computing a temporal prediction. Indeed, Section 6.5.2 has shown that MNet fails to capture important motions. We believe that this step could be significantly enhanced. So far, the motion autoencoder MNet shares the same architecture than the signal autoencoder CNet. Using known techniques from the optical flow estimation literature, such as the addition of a cost volume component, might yields important improvements.

All the video coding trainings done in this thesis have been performed using a proxy coding configuration, composed of one I-frame, one P-frame and one B-frame. The rationale behind this choice is to prepare the model to process all three types of frame. However, finetuning the system using the target frame organization would allow to obtain a better coder, specialized for the desired coding structure.

More practical learned coding

The models presented in this thesis are designed with few complexity constraints, resulting in coders with dozens of million of parameters. In video coding, the complexity constraint is often tighter on the decoder than on the encoder, since the decoder has to be embedded in a variety of low-power devices (smartphones, set-top boxes). Yet, learned coder architectures are often symmetrical. For instance, both the encoding and decoding speed of the proposed system are around a few frame per second for a 720p video. This is different from traditional video coding, where the encoding time significantly exceeds the decoding time. Investigating different balance of complexity between the encoder and decoder would be a important step towards a real-life usage of learned compression.

Likewise, most of the work carried out on learned compression relies on mono-rate coders. This requires to store multiple coders to adjust the rate of the video sequences regarding the available bandwidth. The basic multi-rate coder presented in Chapter 7 offers a solution to this issue. Yet, it is not able to reach competitive performance on a range of rates as wide as traditional coders. Conditioning more operations and coding

modes on the rate constraint would allow to obtain better multi-rate coders.

Content adaptation

In traditional coding, the purpose of the encoding stage is to find the tools which are the most suited to the current video, within the set of tools provided by the standard. These tools have been designed beforehand and target different types of signal: low-frequency signals, screen content signals *etc.* On the other hand, the nature of neural networks is to adapt to their training data. As such, performing the training on the data to compress would allow to obtain an optimal coder for this particular video sequence. Even though the parameters would have to be sent alongside the video, several recent works have hinted this idea as a promising means of improving compression efficiency [27], [28].

Performing operations adapted to the signal is one key factor for improvement and might be one of the explanation regarding the better performance of traditional coders. Throughout this thesis, several means of adapting the operations to the signal have been introduced:

- The hyperprior mechanism allows to use an adapted probability model for each latent pixel;
- The Skip mode offers the choice between two coding modes;
- The quantization gains conditioned on the frame type enable to accurately set the quantization accuracy.

We believe that offering even more possibilities of adaptation to the signal to compress would yield an important increase in performance.

BIBLIOGRAPHY

[1] *Cisco predicts more ip traffic in the next five years than in the history of the internet*, 2017. [Online]. Available: https://newsroom.cisco.com/press-release-content?type=webcontent%5C&articleId=1955935.

[2] *Netflix to cut streaming quality in europe for 30 days*, 2020. [Online]. Available: https://www.bbc.com/news/technology-51968302.

[3] *The shift project, climate crisis: the unsustainable use of online video*, 2020. [Online]. Available: https://theshiftproject.org/en/article/unsustainable-use-online-video/.

[4] R. York, "Ecological paradoxes: william stanley jevons and the paperless office", *Human Ecology Review*, vol. 13, Dec. 2006.

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006, ISBN: 0471241954.

[6] T. Nguyen, T. Ma, M. Ikeda, H. Jang, and X. Zhao, *Jvet ahg report jvet-r0014: lossless and near-lossless coding (ahg14)*, 2020.

[7] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multi-scale structural similarity for image quality assessment", *in in Proc. IEEE Conf. on Signals, Systems, and Computers*, 2003, pp. 1398–1402.

[8] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, *7*, pp. 560–576, 2003. DOI: 10.1109/TCSVT.2003.815165.

[9] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, *12*, pp. 1649–1668, 2012. DOI: 10.1109/TCSVT.2012.2221191.

[10] B. Bross, J. Chen, J.-R. Ohm, G. J. Sullivan, and Y.-K. Wang, "Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc)", *Proceedings of the IEEE*, pp. 1–31, 2021. DOI: 10.1109/JPROC.2020.3043399.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[12] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression", *in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France*, 2017. [Online]. Available: https://openreview.net/forum?id=rJxdQ3jeg.

[13] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders", *in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France*, 2017. [Online]. Available: https://openreview.net/forum?id=rJiNwv9gg.

[14] T. Ladune, P. Philippe, W. Hamidouche, L. Zhang, and O. Déforges, "Binary probability model for learning based image compression", *in 2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, IEEE, 2020, pp. 2168–2172. DOI: 10.1109/ICASSP40776.2020.9053997. [Online]. Available: https://doi.org/10.1109/ICASSP40776.2020.9053997.

[15] A. Habibian, T. van Rozendaal, J. M. Tomczak, and T. Cohen, "Video compression with rate-distortion autoencoders", *in 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, IEEE, 2019, pp. 7032–7041. DOI: 10.1109/ICCV.2019.00713. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00713.

[16] *Workshop and challenge on learned image compression, https://www.compression.cc/*.

[17] F. Bossen, J. Boyce, K. Suehring, X. Li, and V. Seregin, "VTM common test conditions and software reference configurations for sdr video", *in JVET-T2010-v1*, Oct. 2020.

[18] T. Ladune, P. Philippe, W. Hamidouche, L. Zhang, and O. Déforges, "Modenet: mode selection network for learned video coding", *in 30th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2020, Espoo, Finland, September 21-24, 2020*, IEEE, 2020, pp. 1–6. DOI: 10.1109/MLSP49062.2020.9231841.

[19] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: an end-to-end deep video compression framework", *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 2019*, 2019, pp. 11 006–11 015. DOI: 10.1109/CVPR.2019.01126. [Online]. Available: http://openaccess.thecvf.com/content%5C_CVPR%5C_2019/html/Lu%5C_DVC%5C_An%5C_End-To-End%5C_Deep%5C_Video%5C_Compression%5C_Framework%5C_CVPR%5C_2019%5C_paper.html.

[20] Z. Hu, Z. Chen, D. Xu, G. Lu, W. Ouyang, and S. Gu, "Improving deep video compression by resolution-adaptive flow coding", *in Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., ser. Lecture Notes in Computer Science, vol. 12347, Springer, 2020, pp. 193–209. DOI: 10.1007/978-3-030-58536-5_12. [Online]. Available: https://doi.org/10.1007/978-3-030-58536-5%5C_12.

[21] R. Yang, F. Mentzer, L. V. Gool, and R. Timofte, "Learning for video compression with recurrent auto-encoder and recurrent probability model", *IEEE J. Sel. Top. Signal Process.*, vol. 15, *2*, pp. 388–401, 2021. DOI: 10.1109/JSTSP.2020.3043590. [Online]. Available: https://doi.org/10.1109/JSTSP.2020.3043590.

[22] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network", *CoRR*, vol. abs/1611.00850, 2016. arXiv: 1611.00850. [Online]. Available: http://arxiv.org/abs/1611.00850.

[23] A. Golinski, R. Pourreza, Y. Yang, G. Sautière, and T. S. Cohen, "Feedback recurrent autoencoder for video compression", *in Computer Vision - ACCV 2020 - 15th Asian Conference on Computer Vision, Kyoto, Japan, November 30 - December 4, 2020, Revised Selected Papers, Part IV*, H. Ishikawa, C. Liu, T. Pajdla, and J. Shi, Eds., ser. Lecture Notes in Computer Science, vol. 12625, Springer, 2020, pp. 591–607. DOI: 10.1007/978-3-030-69538-5_36. [Online]. Available: https://doi.org/10.1007/978-3-030-69538-5%5C_36.

[24] H. Liu, H. Shen, L. Huang, M. Lu, T. Chen, and Z. Ma, "Learned video compression via joint spatial-temporal correlation exploration", *in The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY,*

USA, February 7-12, 2020, AAAI Press, 2020, pp. 11 580–11 587. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/6825.

[25] E. Agustsson, D. Minnen, N. Johnston, J. Balle, S. J. Hwang, and G. Toderici, "Scale-space flow for end-to-end optimized video compression", *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[26] T. Ladune, P. Philippe, W. Hamidouche, L. Zhang, and O. Déforges, "Optical flow and mode selection for learning-based video coding", *in 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, 2020, pp. 1–6. DOI: 10.1109/MMSP48831.2020.9287049.

[27] E. Dupont, A. Golinski, M. Alizadeh, Y. W. Teh, and A. Doucet, "COIN: compression with implicit neural representations", *CoRR*, vol. abs/2103.03123, 2021. arXiv: 2103.03123. [Online]. Available: https://arxiv.org/abs/2103.03123.

[28] T. van Rozendaal, I. A. M. Huijben, and T. Cohen, "Overfitting for fun and profit: instance-adaptive data compression", *in 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=oFp8Mx%5C_V5FL.

[29] D. A. Huffman, "A method for the construction of minimum-redundancy codes", *Proceedings of the IRE*, vol. 40, *9*, pp. 1098–1101, 1952. DOI: 10.1109/JRPROC.1952.273898.

[30] G. G. Langdon, "An introduction to arithmetic coding", *IBM Journal of Research and Development*, vol. 28, *2*, pp. 135–149, 1984. DOI: 10.1147/rd.282.0135.

[31] M. Wien, "Entropy coding", *in High Efficiency Video Coding: Coding Tools and Specification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 251–282, ISBN: 978-3-662-44276-0. DOI: 10.1007/978-3-662-44276-0_10. [Online]. Available: https://doi.org/10.1007/978-3-662-44276-0_10.

[32] B. Kleijn, "A basis for source coding", 2003.

[33] K. R. Rao and P. Yip, *The Transform and Data Compression Handbook*. USA: CRC Press, Inc., 2000, ISBN: 0849336929.

[34] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression", *IEEE Signal Processing Magazine*, vol. 15, *6*, pp. 74–90, 1998. DOI: 10.1109/79.733497.

[35] K. Reuzé, “Adaptive Coding of Intra Prediction Modes in the Future Video Coding”, Theses, INSA de Rennes, 2018.

[36] G. K. Wallace, “The jpeg still picture compression standard”, *Commun. ACM*, vol. 34, *4*, pp. 30–44, Apr. 1991, ISSN: 0001-0782. DOI: 10.1145/103085.103089. [Online]. Available: http://doi.acm.org/10.1145/103085.103089.

[37] N. Ahmed, T. Natarajan, and K. Rao, “Discrete cosine transform”, *IEEE Transactions on Computers*, vol. C-23, *1*, pp. 90–93, 1974. DOI: 10.1109/T-C.1974.223784.

[38] A. Saxena and F. C. Fernandes, “Dct/dst-based transform coding for intra prediction in image/video coding”, *Trans. Img. Proc.*, vol. 22, *10*, pp. 3974–3981, Oct. 2013, ISSN: 1057-7149. DOI: 10.1109/TIP.2013.2265882. [Online]. Available: https://doi.org/10.1109/TIP.2013.2265882.

[39] S. Ruder, *An overview of gradient descent optimization algorithms*, 2017. arXiv: 1609.04747 [cs.LG].

[40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 2004, ISBN: 0521833787.

[41] V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, 2018. arXiv: 1603.07285 [stat.ML].

[42] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, *Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network*, 2016. arXiv: 1609.05158 [cs.CV].

[43] L. Xu, J. S. Ren, C. Liu, and J. Jia, “Deep convolutional neural network for image deconvolution”, *Advances in neural information processing systems*, vol. 27, pp. 1790–1798, 2014.

[44] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, *Learned image compression with discretized gaussian mixture likelihoods and attention modules*, 2020. arXiv: 2001.01568 [eess.IV].

[45] F. Mentzer, G. D. Toderici, M. Tschannen, and E. Agustsson, “High-fidelity generative image compression”, *in Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 11 913–11 924. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/8a50bae297807da9e97722a0b3fd8f27-Paper.pdf.

[46] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior", *in 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada*, 2018. [Online]. Available: https://openreview.net/forum?id=rkcQFMZRb.

[47] Y. Hu, W. Yang, Z. Ma, and J. Liu, "Learning end-to-end lossy image compression: a benchmark", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. DOI: 10.1109/TPAMI.2021.3065339.

[48] D. Minnen, J. Ballé, and G. Toderici, "Joint autoregressive and hierarchical priors for learned image compression", *in Conference on Neural Information Processing Systems 2018, NeurIPS„ Montréal, Canada.*, 2018, pp. 10 794–10 803. [Online]. Available: http://papers.nips.cc/paper/8275-joint-autoregressive-and-hierarchical-priors-for-learned-image-compression.

[49] J. Lee, S. Cho, and S. Beack, "Context-adaptive entropy model for end-to-end optimized image compression", *in International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 2019*. [Online]. Available: https://openreview.net/forum?id=HyxKIiAqYQ.

[50] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, *Soft-to-hard vector quantization for end-to-end learning compressible representations*, 2017. arXiv: 1704.00648 [cs.LG].

[51] Z. Guo, Z. Zhang, R. Feng, and Z. Chen, *Soft then hard: rethinking the quantization in neural image compression*, 2021. arXiv: 2104.05168 [eess.IV].

[52] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation", *in 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1511.06281.

[53] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift", *in Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html.

[54] G. Bjontegaard, “Calculation of average psnr differences between rd-curves”, *in ITU-T Q.6/16, Doc. VCEG-M33*, Mar. 2001.

[55] A. van den Oord, N. Kalchbrenner, L. Espeholt, k. kavukcuoglu koray, O. Vinyals, and A. Graves, “Conditional image generation with PixelCNN decoders”, *in Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, Curran Associates, Inc., 2016.

[56] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, “Pixelcnn++: improving the pixelcnn with discretized logistic mixture likelihood and other modifications”, *in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=BJrFC6ceg.

[57] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, “Practical full resolution learned lossless image compression”, *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[58] D. Minnen and S. Singh, “Channel-wise autoregressive entropy models for learned image compression”, *in 2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 3339–3343. DOI: 10.1109/ICIP40778.2020.9190935.

[59] Z. Guo, Z. Zhang, R. Feng, and Z. Chen, *Causal contextual prediction for learned image compression*, 2021. arXiv: 2011.09704 [cs.CV].

[60] Z. Zhong, H. Akutsu, and K. Aizawa, “Channel-level variable quantization network for deep image compression”, *in Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, C. Bessiere, Ed., ijcai.org, 2020, pp. 467–473. DOI: 10.24963/ijcai.2020/65. [Online]. Available: https://doi.org/10.24963/ijcai.2020/65.

[61] J. Liu, G. Lu, Z. Hu, and D. Xu, *A unified end-to-end framework for efficient deep image compression*, 2020. arXiv: 2002.03370 [eess.IV].

[62] L. Helminger, A. Djelouah, M. Gross, and C. Schroers, “Lossy image compression with normalizing flows”, *in Neural Compression: From Information Theory to Applications – Workshop @ ICLR 2021*, 2021. [Online]. Available: https://openreview.net/forum?id=NQJ9pMf9id.

[63] J. Liu, G. Lu, Z. Hu, and D. Xu, "A unified end-to-end framework for efficient deep image compression", *CoRR*, vol. abs/2002.03370, 2020. arXiv: 2002.03370. [Online]. Available: https://arxiv.org/abs/2002.03370.

[64] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned lossless image compression with A hyperprior and discretized gaussian mixture likelihoods", *in 2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, IEEE, 2020, pp. 2158–2162. DOI: 10.1109/ICASSP40776.2020.9053413. [Online]. Available: https://doi.org/10.1109/ICASSP40776.2020.9053413.

[65] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, *1*, pp. 221–231, 2013. DOI: 10.1109/TPAMI.2012.59.

[66] A. Djelouah, J. Campos, S. Schaub-Meyer, and C. Schroers, "Neural inter-frame compression for video coding", *in The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2019.

[67] M. A. Yilmaz and A. M. Tekalp, "End-to-end rate-distortion optimization for bi-directional learned video compression", *2020 IEEE International Conference on Image Processing (ICIP)*, Oct. 2020. DOI: 10.1109/icip40778.2020.9190881. [Online]. Available: http://dx.doi.org/10.1109/ICIP40778.2020.9190881.

[68] G. Lu, C. Cai, X. Zhang, L. Chen, W. Ouyang, D. Xu, and Z. Gao, "Content adaptive and error propagation aware deep video compression", *in Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., ser. Lecture Notes in Computer Science, vol. 12347, Springer, 2020, pp. 456–472. DOI: 10.1007/978-3-030-58536-5_27. [Online]. Available: https://doi.org/10.1007/978-3-030-58536-5%5C_27.

[69] R. Yang, F. Mentzer, L. V. Gool, and R. Timofte, "Learning for video compression with hierarchical quality and recurrent enhancement", *in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, IEEE, 2020, pp. 6627–6636. DOI: 10.1109/CVPR42600.2020.00666. [Online]. Available: https://doi.org/10.1109/CVPR42600.2020.00666.

[70] Z. Hu, G. Lu, and D. Xu, "FVC: A new framework towards deep video compression in feature space", *CoRR*, vol. abs/2105.09600, 2021. arXiv: 2105.09600. [Online]. Available: https://arxiv.org/abs/2105.09600.

[71] D. Sun, X. Yang, M. Liu, and J. Kautz, "Pwc-net: CNNs for optical flow using pyramid, warping, and cost volume", *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA*, 2018, pp. 8934–8943. DOI: 10.1109/CVPR.2018.00931. [Online]. Available: http://openaccess.thecvf.com/content%5C_cvpr%5C_2018/html/Sun%5C_PWC-Net%5C_CNNs%5C_for%5C_CVPR%5C_2018%5C_paper.html.

[72] T. Ladune and P. Philippe, "Coding standards as anchors for the CVPR clic video track", *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops 2021*.

[73] H. Egilmez, A. Singh, M. Coban, and M. Karczewicz, "Jvet-u0079: a DNN architecture for intra-frame coding in yuv 4:2:0 format with cross-component prediction", *in JVET-U0079-v1*, Jan. 2021.

[74] T. Guo, J. Wang, Z. Cui, Y. Feng, Y. Ge, and B. Bai, "Variable rate image compression with content adaptive optimization", *in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 533–537. DOI: 10.1109/CVPRW50498.2020.00069.

[75] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation", *in European Conf. on Computer Vision (ECCV)*, A. Fitzgibbon et al. (Eds.), Ed., ser. Part IV, LNCS 7577, Springer-Verlag, Oct. 2012, pp. 611–625.

[76] T. Ladune, P. Philippe, W. Hamidouche, L. Zhang, and O. Déforges, "Conditional coding for flexible learned video compression", *in Neural Compression: From Information Theory to Applications – Workshop @ ICLR 2021*, 2021. [Online]. Available: https://openreview.net/forum?id=uyMvuXoV1lZ.

[77] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: dataset and study", *in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jul. 2017.

[78] T. J. A. dataset, *Https://jpegai.github.io/3-datasets/*.

[79] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders", *in Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16, Barcelona, Spain: Curran Associates Inc., 2016, pp. 4797–4805, ISBN: 9781510838819.

[80] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", *in 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90.

[81] V. Hosu, F. Hahn, M. Jenadeleh, H. Lin, H. Men, T. Szirányi, S. Li, and D. Saupe, "The konstanz natural video database (konvid-1k)", *in 2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, IEEE, 2017, pp. 1–6.

[82] R. Yang, Y. Yang, J. Marino, and S. Mandt, "Hierarchical autoregressive modeling for neural video compression", *in 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=TK%5C_6nNb%5C_C7q.

[83] A. Mercat, M. Viitanen, and J. Vanne, "UVG dataset: 50/120fps 4k sequences for video codec analysis and development", *in Proceedings of the 11th ACM Multimedia Systems Conference, MMSys 2020, Istanbul, Turkey, June 8-11, 2020*, L. Toni, A. C. Begen, Ö. Alay, and C. Timmerer, Eds., ACM, 2020, pp. 297–302. DOI: 10.1145/3339825.3394937. [Online]. Available: https://doi.org/10.1145/3339825.3394937.

APPENDIX A

LIST OF PUBLICATIONS

Conference papers

Binary probability model for learning based image compression

IEEE ICASSP 2020

Authors: Théo Ladune, P. Philippe, W. Hamidouche, L. Zhang, O. Déforges

Abstract

In this paper, we propose to enhance learned image compression systems with a richer probability model for the latent variable. Previous works model the latent variable with a Gaussian or a Laplace distribution. Inspired by binary arithmetic coding, we propose to signal the latent variable with three binary values and one integer, with different probability models.A relaxation method is designed to perform gradient-based training. The richer probability model results in a better entropy coding leading to lower rate. Experiments under the Challenge on Learned Image Compression test conditions demonstrate that this method achieves 18 % rate saving compared to Gaussian or Laplace models.

ModeNet: Mode selection network for learned video coding

IEEE MLSP 2020

Authors: Théo Ladune, P. Philippe, W. Hamidouche, L. Zhang, O. Déforges

Abstract

In this paper, a mode selection network (ModeNet) is proposed to enhance deep learning-based video compression. Inspired by traditional video coding, ModeNet purpose is to enable competition among several coding modes. The proposed ModeNet learns and conveys a pixel-wise partitioning of the frame, used to assign each pixel to the most suited

coding mode. ModeNet is trained alongside the different coding modes to minimize a rate-distortion cost. It is a flexible component which can be generalized to other systems to allow competition between different coding tools. ModeNet interest is studied on a P-frame coding task, where it is used to design a method for coding a frame given its prediction. ModeNet-based systems achieve compelling performance when evaluated under the Challenge on Learned Image Compression 2020 P-frame coding track conditions.

Optical flow and mode selection for learning-based video coding

IEEE MMSP 2020

Authors: Théo Ladune, P. Philippe, W. Hamidouche, L. Zhang, O. Déforges

Abstract

This paper introduces a new method for inter frame coding based on two complementary autoencoders: MOFNet and CodecNet. MOFNet aims at computing and conveying the Optical Flow and a pixel-wise coding Mode selection. The optical flow is used to perform a prediction of the frame to code. The coding mode selection enables competition between direct copy of the prediction or transmission through CodecNet.The proposed coding scheme is assessed under the Challenge on Learned Image Compression 2020 P-frame coding conditions, where it is shown to perform on par with the state-of-the-art video codec ITU/MPEG HEVC. Moreover, the possibility of copying the prediction enables to learn the optical flow in an end-to-end fashion i.e. without relying on pre-training and/or a dedicated loss term.

Note: This publication received the *Best Paper Award* at the IEEE MMSP 2020 conference.

Conditional coding for flexible learned video compression

ICLR 2021, Neural Compression workshop

Authors: Théo Ladune, P. Philippe, W. Hamidouche, L. Zhang, O. Déforges

Abstract

This paper introduces a novel framework for end-to-end learned video coding. Image compression is generalized through conditional coding to exploit information from reference

frames, allowing to process intra and inter frames with the same coder. The system is trained through the minimization of a rate-distortion cost, with no pre-training or proxy loss. Its flexibility is assessed under three coding configurations (All Intra, Low-delay P and Random Access), where it is shown to achieve performance competitive with the state-of-the-art video codec HEVC.

Conditional coding and variable bitrate for practical learned video coding

IEEE CVPR 2021, Challenge on Learned Image Compression workshop

Authors: Théo Ladune, P. Philippe, W. Hamidouche, L. Zhang, O. Déforges

Abstract

This paper introduces a practical learned video codec. Conditional coding and quantization gain vectors are used to provide flexibility to a single encoder/decoder pair, which is able to compress video sequences at a variable bitrate. The flexibility is leveraged at test time by choosing the rate and GOP structure to optimize a rate-distortion cost. Using the CLIC21 video test conditions, the proposed approach shows performance on par with HEVC.

Coding standards as anchors for the CVPR CLIC video track

IEEE CVPR 2021, Challenge on Learned Image Compression workshop

Authors: Théo Ladune, P. Philippe

Abstract

In 2021, a new track has been initiated in the Challenge for Learned Image Compression : the video track. This category proposes to explore technologies for the compression of short video clips at 1 Mbit/s. This paper proposes to generate coded videos using the latest standardized video coders, especially Versatile Video Coding (VVC). The objective is not only to measure the progress made by learning techniques compared to the state of the art video coders, but also to quantify their progress from years to years. With this in mind, this paper documents how to generate the video sequences fulfilling the requirements of this challenge, in a reproducible way, targeting the maximum performance for VVC.

Note: This publication won the video track of the CLIC 2021.

Patent applications

- *Prédiction pondérée d'image, codage et décodage d'image utilisant une telle prédiction pondérée, patent application*, FR 2101632, February 2021
- *Détermination d'au moins un mode de codage d'image ou d'au moins un mode de décodage d'image, codage et décodage d'image utilisant une telle détermination*, patent application, FR 2101633, February 2021

APPENDIX B

ADDITIONAL DETAILS ON LEARNED IMAGE CODING

B.1 Training recipe

B.1.1 Training dataset

The learned coding schemes are trained using a training set composed of 500 000 examples. All training samples are 256×256 crops, extracted from natural images dataset such as the Challenge on Learned Image Coding (CLIC) dataset [16], the DIV2K dataset [77] and the JPEG-AI dataset [78]. These datasets are known to feature miscellaneous contents with diverse resolutions, allowing to learn a generic image coder.

B.1.2 Details on the training stage

The expectation of the loss function is estimated by computing the average loss when feeding the network with a batch of 8 examples. The number of training iterations (*i.e.* network parameters updates) ranges from half a million to two million. While most of the results are achieved during the first half a million steps, prolongation of the training enables to grasp a few additional performance.

The value of the learning rate, see Eq. (3.4) is set to $\eta = 10^{-4}$. Decreasing the learning rate throughout the training allows for a better converge of the gradient descent algorithm. As such, the learning rate is successively decreased to $\eta = 2 \cdot 10^{-5}$ and $\eta = 4 \cdot 10^{-6}$.

The purpose of the training is to optimize a rate-distortion loss, parameterized through a rate constraint λ:

$$\mathcal{L}_\lambda = D + \lambda R. \tag{B.1}$$

Empirically, replacing the rate-constraint λ with a smaller one $\lambda' < \lambda$ during the first iterations of the training often results in better performance.

B.2 Additional information regarding the test stage

B.2.1 Inference dataset

The performance of the system is evaluated on the CLIC 2020 validation set [16]. It is composed of 102 natural images with a resolution ranging from 512×384 to 2021×1518. To provide additional variety, two different categories of images are present in this dataset. Half is made of photos from professional photographers, featuring sharp and in-focus edges as well as bokeh effects. The other half is composed of user generated content, which often exhibits more noise and less sharpness. Figure B.1 presents one example extracted from each subcategory.

B.2.2 Coding standards as anchors

In order to assess the performance of the learned coding schemes, several existing standards are used as anchors. The JPEG image coding standard [36] is evaluated with the following image magick command, relying on *libjpeg 6.2*:

```
convert original.png -quality Q compressed.jpeg
```

Varying the quality parameter `Q` allows to obtain a rate-distortion curve.

Despite being very popular, JPEG is not a state-of-the-art still image coder. The image coding configuration of modern video coding standards (HEVC and VVC) achieves better rate-distortion performance. To ensure a fair comparison, the results of HEVC and VVC are obtained through one of their best implementation, the HM 16.22 (HEVC Test Model) and the VTM 10 (VVC Test Model), using the intra coding configuration for RGB 444.

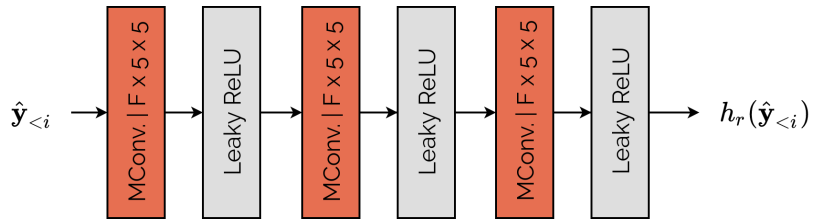
B.3 Detailed ARM architecture

This section details the architecture of the Auto-Regressive Module (ARM) and the fusion module whose results are presented in Section 3.4.2. Figure B.2a presents the architecture of the ARM, while Fig. B.2b shows the architecture of the fusion module for the hyperprior and ARM.

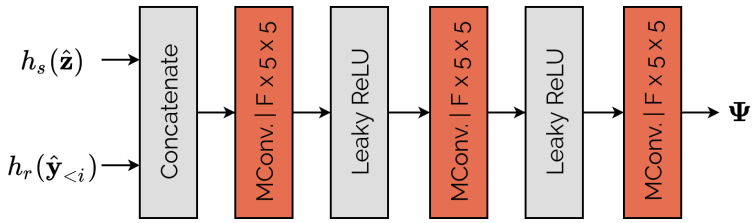
(a) Image *IMG_0426* from the user-generated dataset.

(b) Image *nomao-saeki-33553* from the professional dataset.

Figure B.1: Examples from the professional and user-generated datasets.



(a) ARM



(b) Fusion

Figure B.2: Architecture of the Auto-Regressive Module and the hyperprior/ARM fusion component. MConv $F \times 5 \times 5$ denotes a masked convolutional layer with F output feature maps and kernels of size 5×5. The kernel of masked convolutions have null parameters for anti-causal pixels [79] *i.e.* they only process past (already received) values.

B.4 Residual blocks and attention modules

When training a deep neural network, one may encounter *vanishing gradient* issues. Indeed, it often happens that the gradient of each layer with respect to its input becomes small. This causes the product of the successive gradients to be very close to zero. Consequently the backpropagation algorithm only perform tiny adjustments to the network parameters and the optimization process yields a suboptimal network.

Figure B.3a introduces the architecture of a residual block, which has been shown to ease convergence, allowing to obtain more performant systems [80]. This is due to the fact that a residual block learns a residual mapping:

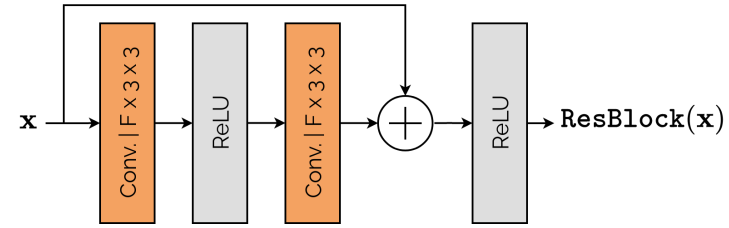
$$\texttt{ResBlock}(\mathbf{x}) = \mathbf{x} + f(\mathbf{x}). \tag{B.2}$$

Consequently the derivative of the residual block with respect to its input is:

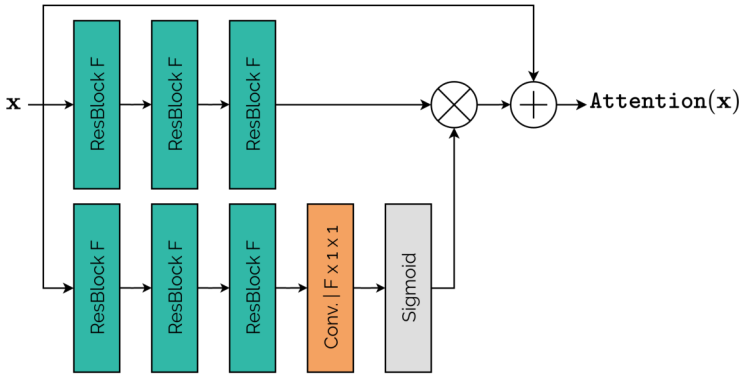
$$\frac{\partial \texttt{ResBlock}(\mathbf{x})}{\partial \mathbf{x}} = 1 + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}. \tag{B.3}$$

As a result, even if the term $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ becomes too small, the overall residual block derivative is close to 1, avoiding the vanishing gradient issue.

The attention mechanism presented in Fig. B.3b allows the system to focus on the more relevant extracted features. The middle branch, called the trunk, is responsible for extracting features from the input, as an usual convolutional networks would do. Yet, the bottom branch computes attention maps based on the inputs, with values in $[0,1]$ thanks to the sigmoid function. These attention maps are applied through a pixel-wise multiplication of the trunk feature maps, allowing to outline the more important features. Finally, a residual connexion is added to ease the training of the overall architecture.



(a) Residual block.



(b) Residual attention module.

Figure B.3: Residual block and a residual attention module.

B.5 Comprehensive architecture of a state-of-the-art image coder

Attention modules and residual blocks are combined to design the different transforms of an hyperprior-based image coder. Figures B.4 and B.5 present the architecture of the coding scheme, used in Section 3.4.3. This architecture is widely inspired by Cheng et *al.* [44] with a few modifications. The overall number of parameters is 20 million.

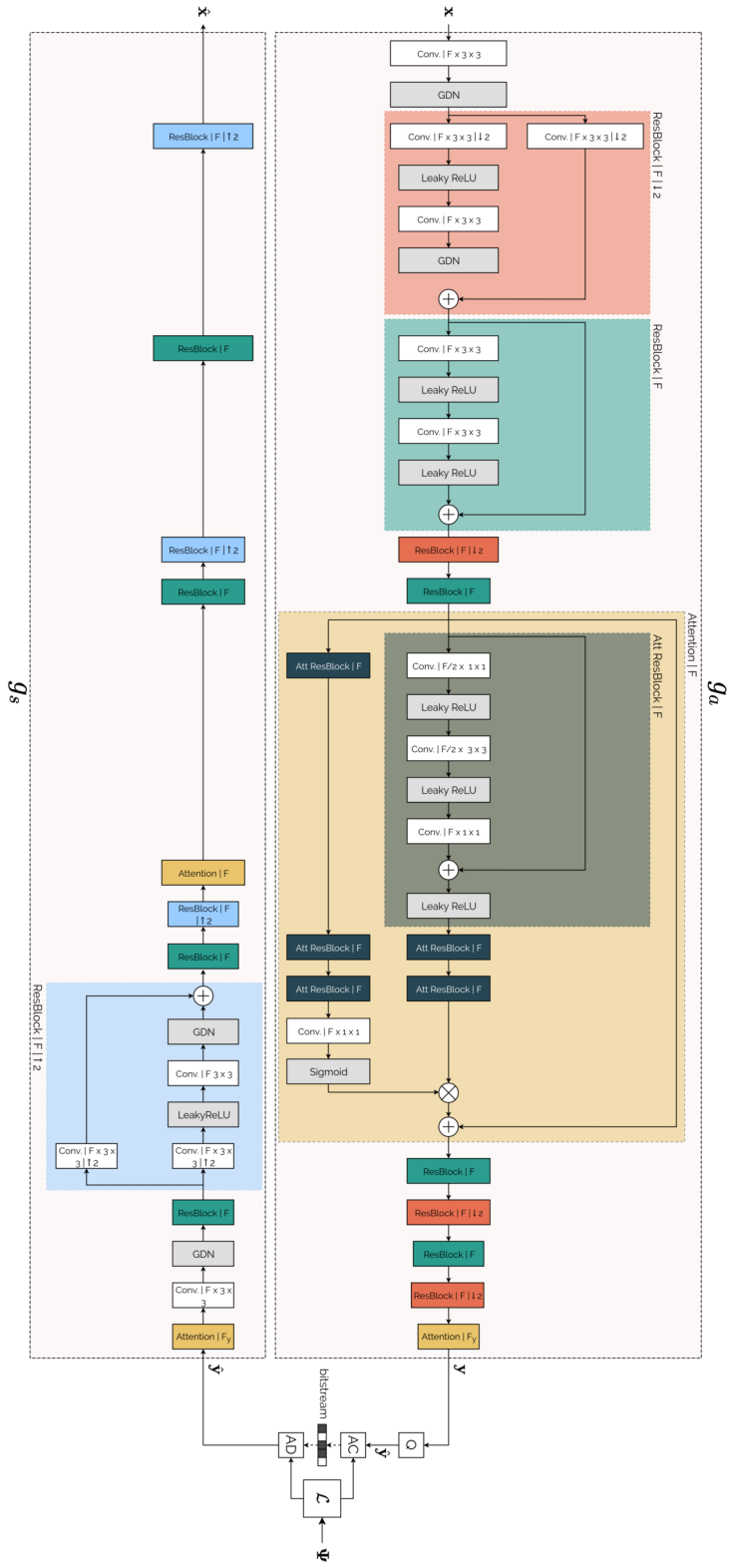


Figure B.4: The analysis and synthesis transforms g_a and g_s. The number of internal features F is set to 192, the number of latent features F_y is set to 256.

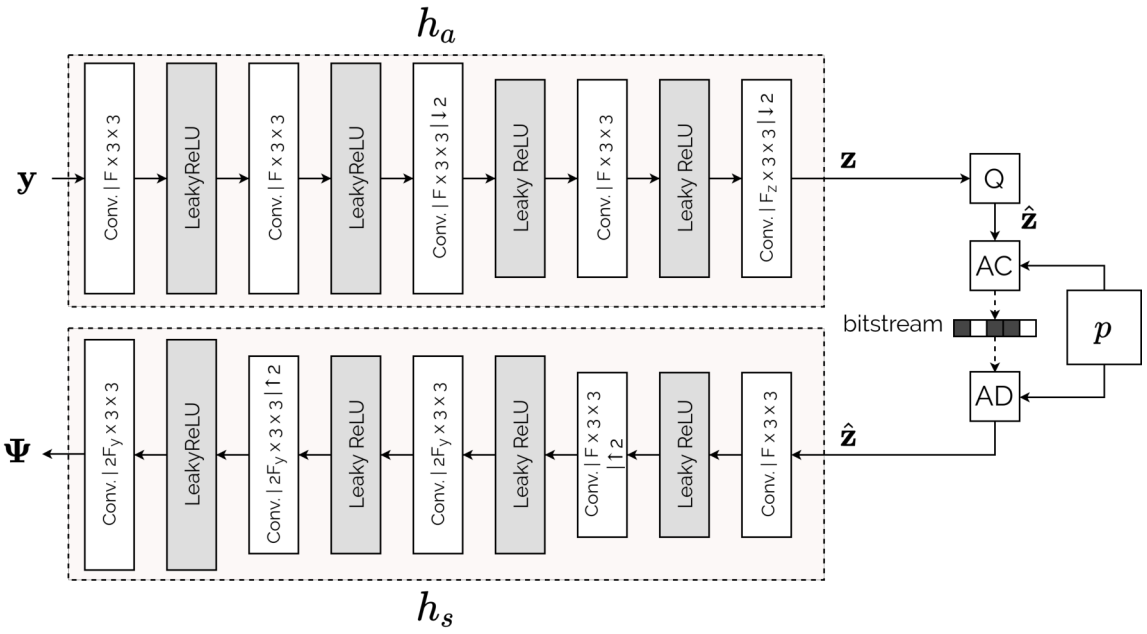


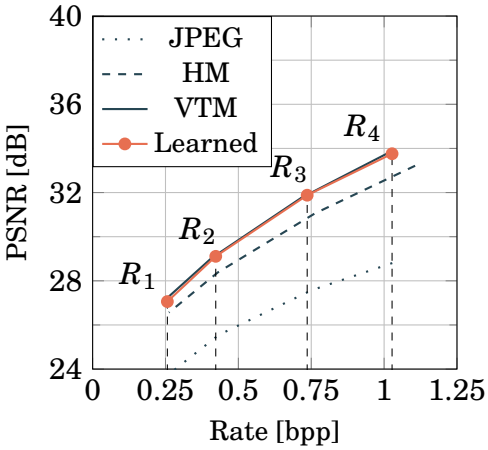
Figure B.5: The hyperprior analysis and synthesis transforms h_a and h_s. The number of internal features F is set to 192, the number of hyperprior features F_z is set to 64.

B.6 Additional visual examples

Figures B.6, B.7, B.8, B.9 and B.10 present visual comparisons between a learned image coder and several traditional anchors (JPEG, HM and VTM) under different rate constraints. Unlike traditional coders, the learned coder does not perform a block based processing. As such, it avoids blocking artifacts and discontinuities, especially at low rate, see Fig. B.7. The convolutional nature of the learned coder also avoids other artifacts *e.g.* banding (the sky with JPEG at rate R_1 in Fig. B.8) or ringing (the sky with HM at rate R_1 in Fig. B.8).

The variety of the examples proposed in these Figures demonstrates that the learned image coder offers visual performance competitive with state-of-the-art traditional coder such as the VTM. The learned coder handles properly low-frequency areas (the girl's face in Fig. B.7 or the sky in Fig. B.8), high-frequency areas (the trees in Fig. B.6 and B.10) as well as areas with small details (the numbers in Fig. B.9).

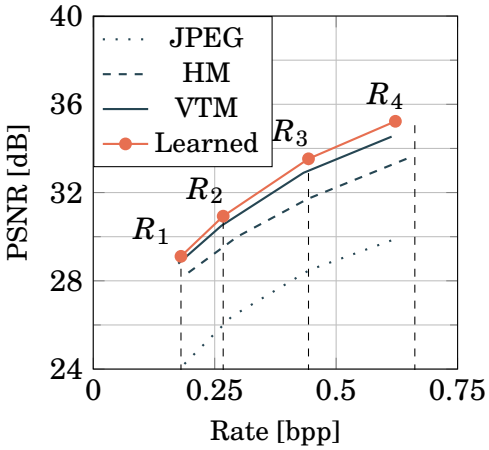
(a) Original image.



(b) Rate-distortion curves.

Figure B.6: Visual comparison for the image *46c1831600829ae8b30c6b06557424ef*.

(a) Original image.



(b) Rate-distortion curves.

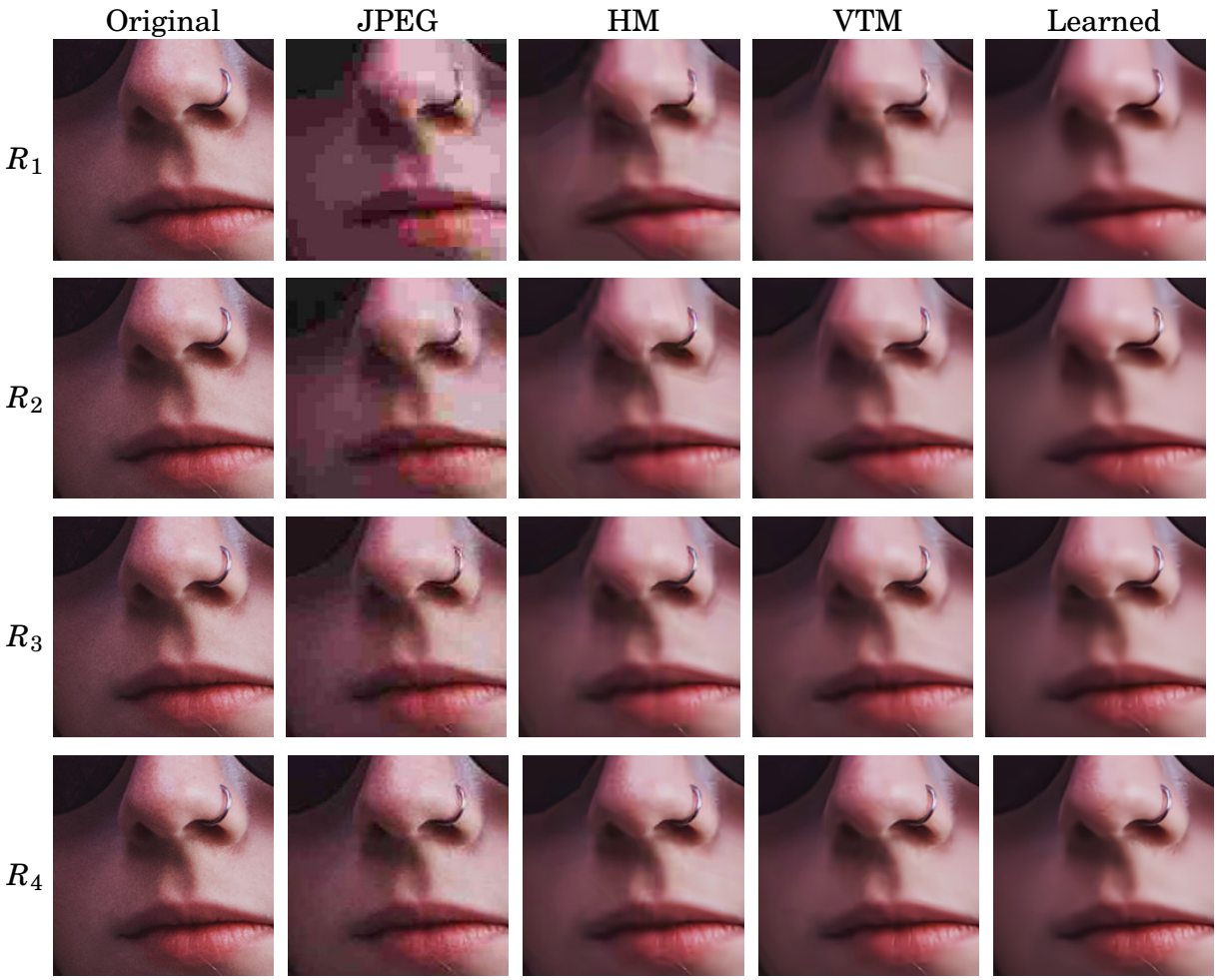
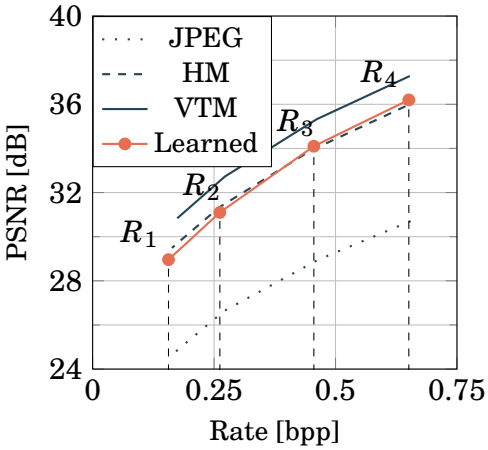


Figure B.7: Visual comparison for the image *ad249bba099568403dc6b97bc37f8d74*.

(a) Original image.



(b) Rate-distortion curves.

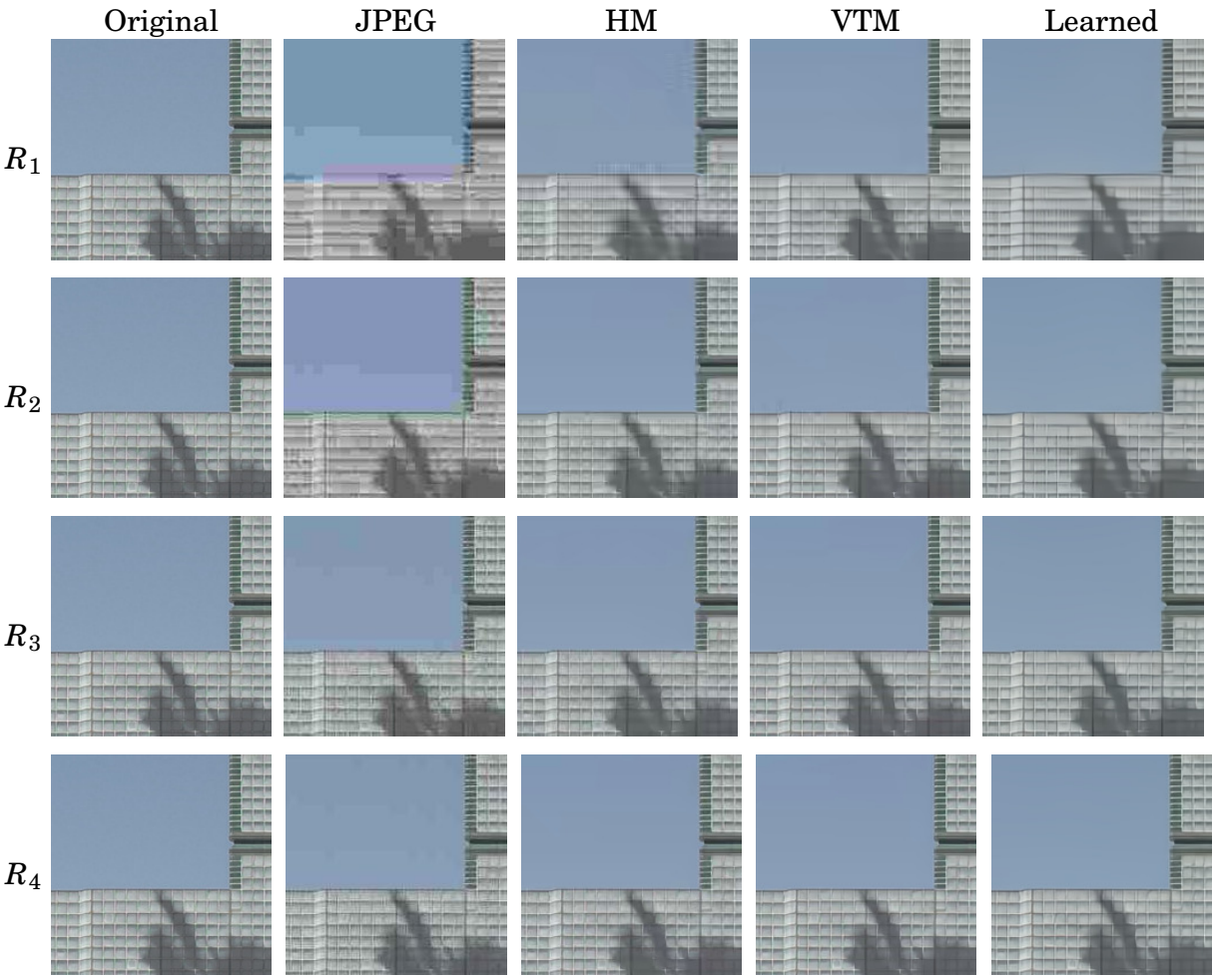
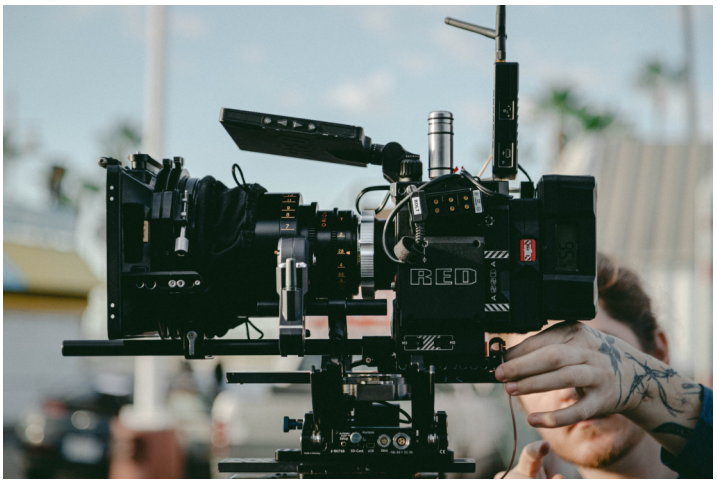
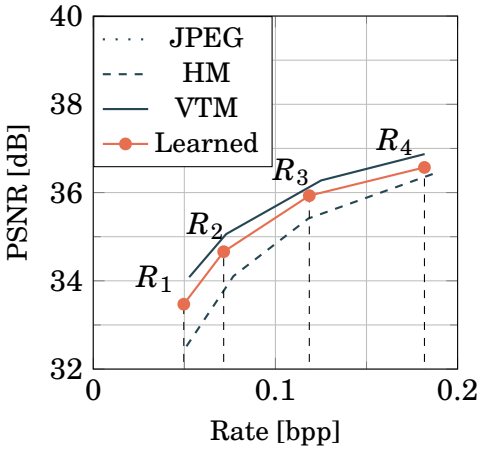


Figure B.8: Visual comparison for the image *08052112d0151f7c9ac4879f838d5a0c*.



(a) Original image.



(b) Rate-distortion curves.

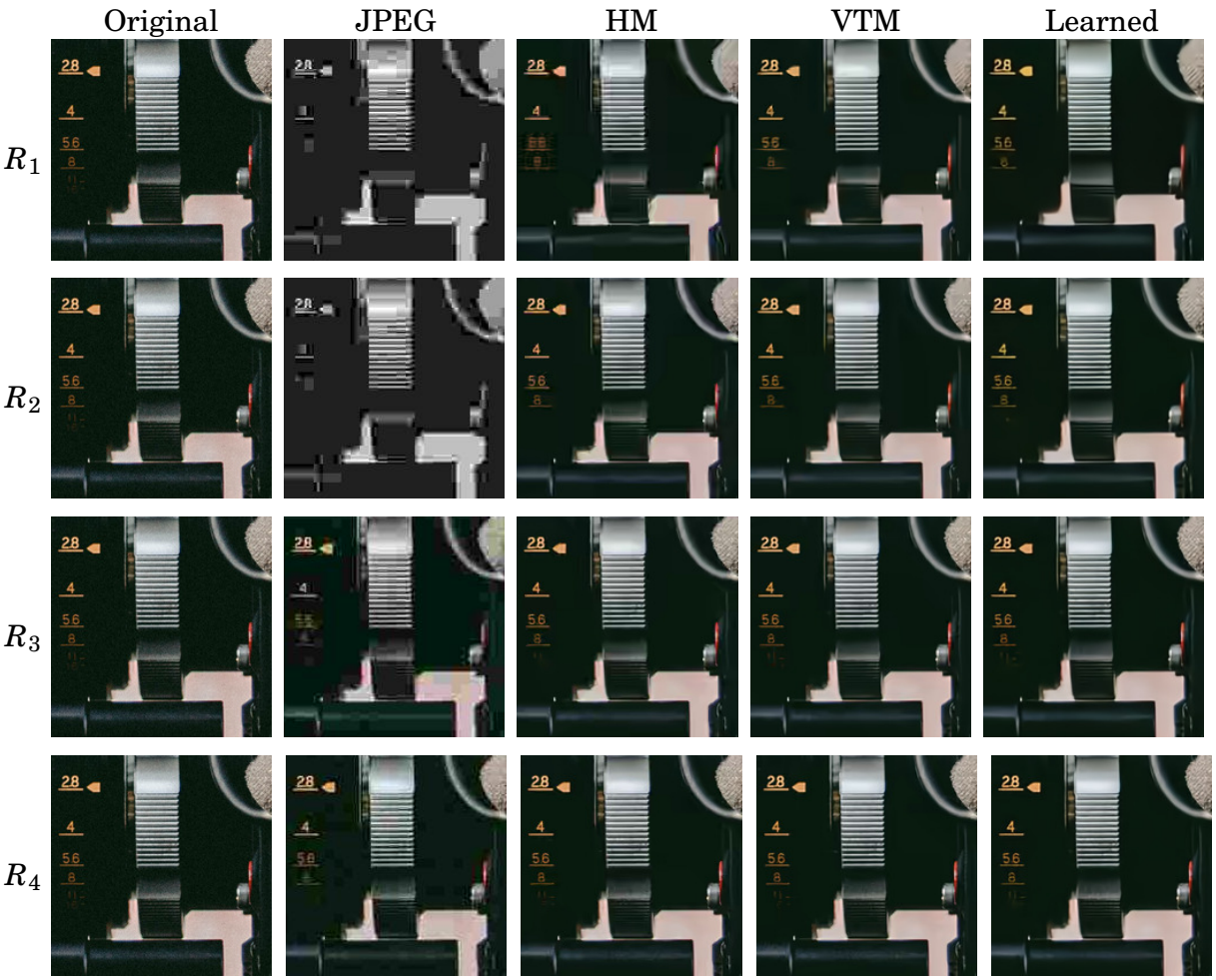
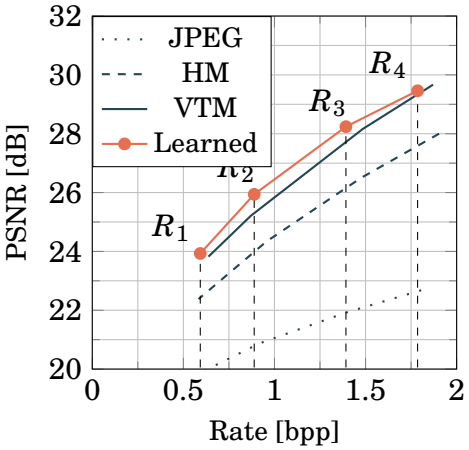


Figure B.9: Visual comparison for the image *400984b87394ada6d9627ed918908986*.

(a) Original image.



(b) Rate-distortion curves.

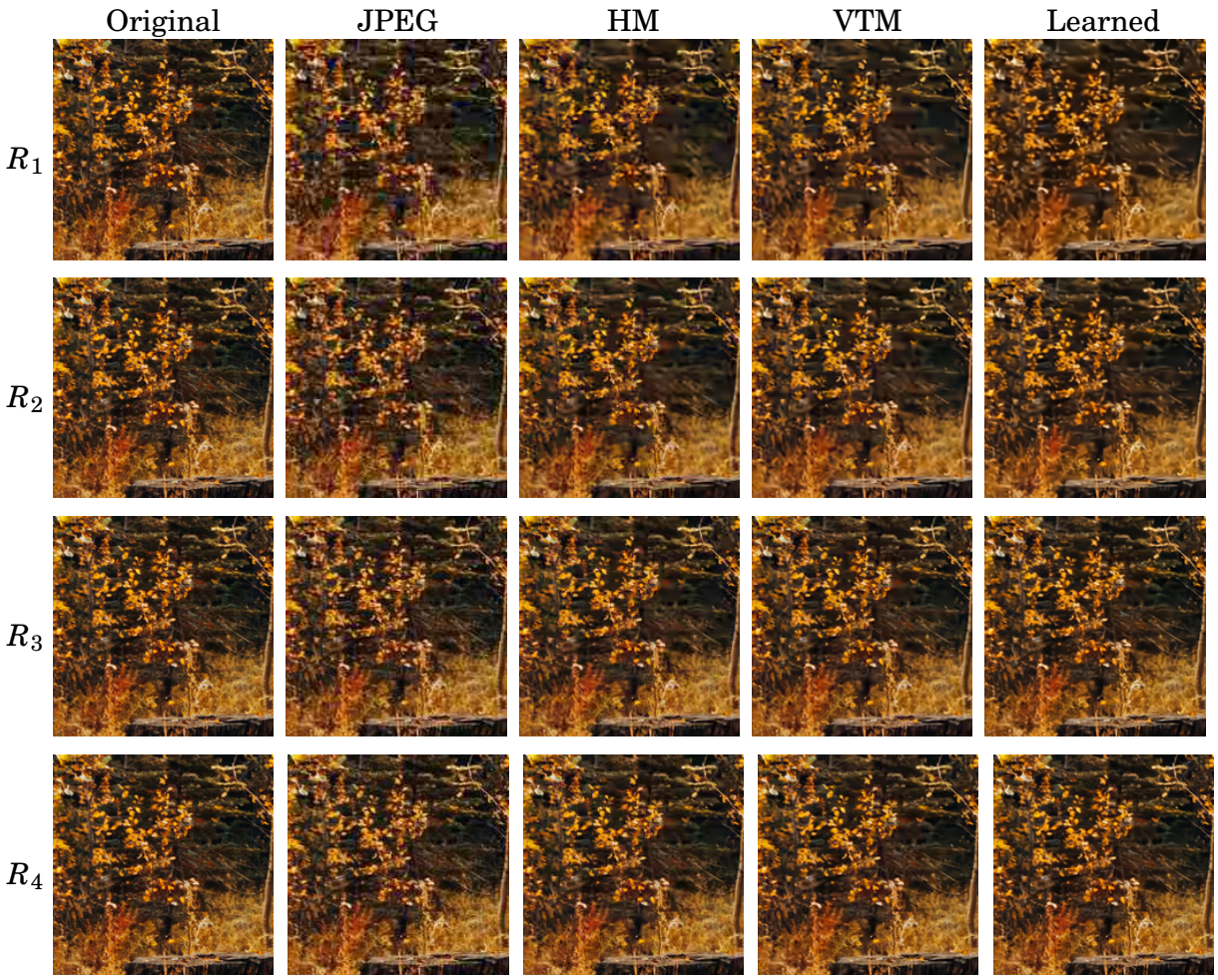


Figure B.10: Visual comparison for the image *72e19343f46a447bea2206c368a9692a*.

APPENDIX C

TRAINING DATASET FOR LEARNED VIDEO CODING

C.1 Requirements

The parameters of the learned coding schemes are optimized through the minimization of a rate-distortion cost for successive training examples. As such, the nature of the training data significantly impacts the performance of the resulting system. At test time, a learned coder performs well on data similar to the training set, while video sequences exhibiting a nature too different from the training data leads to worse performance. In this thesis, the objective is to obtain a generic coder. In particular, the system should perform well on:

- Different natures of video sequences (user-generated contents, professional videos and synthetic sequences);
- Different resolutions, from 540p to 2160p;
- Different frame rates, from 24 to 60 frames per second.

To this end, the training dataset is constructed by gathering different existing video datasets.

C.2 Dataset composition

Section 4.3.4 has introduced the coding configuration used during the training stage: each example is a 3-frame video sequences. To ensure an acceptable memory usage, all examples are 256×256 crops, extracted from pre-existing video datasets. Table C.1 summarises the different datasets used to composed the training set. The resulting training set is composed of 5.2 million video sequences.

Table C.1: Composition of the training dataset. UGC stands for user-generated content and Prof. for professional content.

Dataset	**Nature**				**Resolution**	**Average FPS**	**Number examples**
	UGC	Prof.	Natural	Synthetic			
KoNVid_1k [81]	✓		✓		540p	28	0.9×10^6
CLIC2021 [16]	✓		✓	✓	720p	30	1.4×10^6
YouTube-NT [82]		✓	✓	✓	1080p	25	2.5×10^6
YUV_4K [83]		✓	✓		2160p	58	0.4×10^6
Total							5.2×10^6

APPENDIX D

COMPREHENSIVE EXPERIMENTAL DETAILS FOR CNET

D.1 Residual autoencoder

This section presents the architecture of the autoencoder used for the residual and image coding in the context of video compression. In both case, exactly same architecture is implemented, the difference lies simply on the input. Residual coding input is $\mathbf{x}_t - \tilde{\mathbf{x}}_t$ *i.e.* the subtraction between the frame to code and its prediction. For image coding, the input is the frame to code $\mathbf{x}_t$.

D.2 Quantization gains

Figure D.3 presents the encoder coding gains learned according to different rate constraints. Across the entire range of rate, the quantization gains (*i.e.* quantization accuracy) of the intra frames is always more important than those of the inter frames. Moreover, this example allows to illustrate that the system learns to disable some latent feature maps when the rate constraint becomes too tight. For instance, compare the highest and the lowest rate systems, respectively the systems a and g. For the high-rate system, the 64 latent feature maps are active. For the low-rate system only the first 25 latent feature maps are active. The other ones are always equal to zero, allowing to reduce the overall rate.

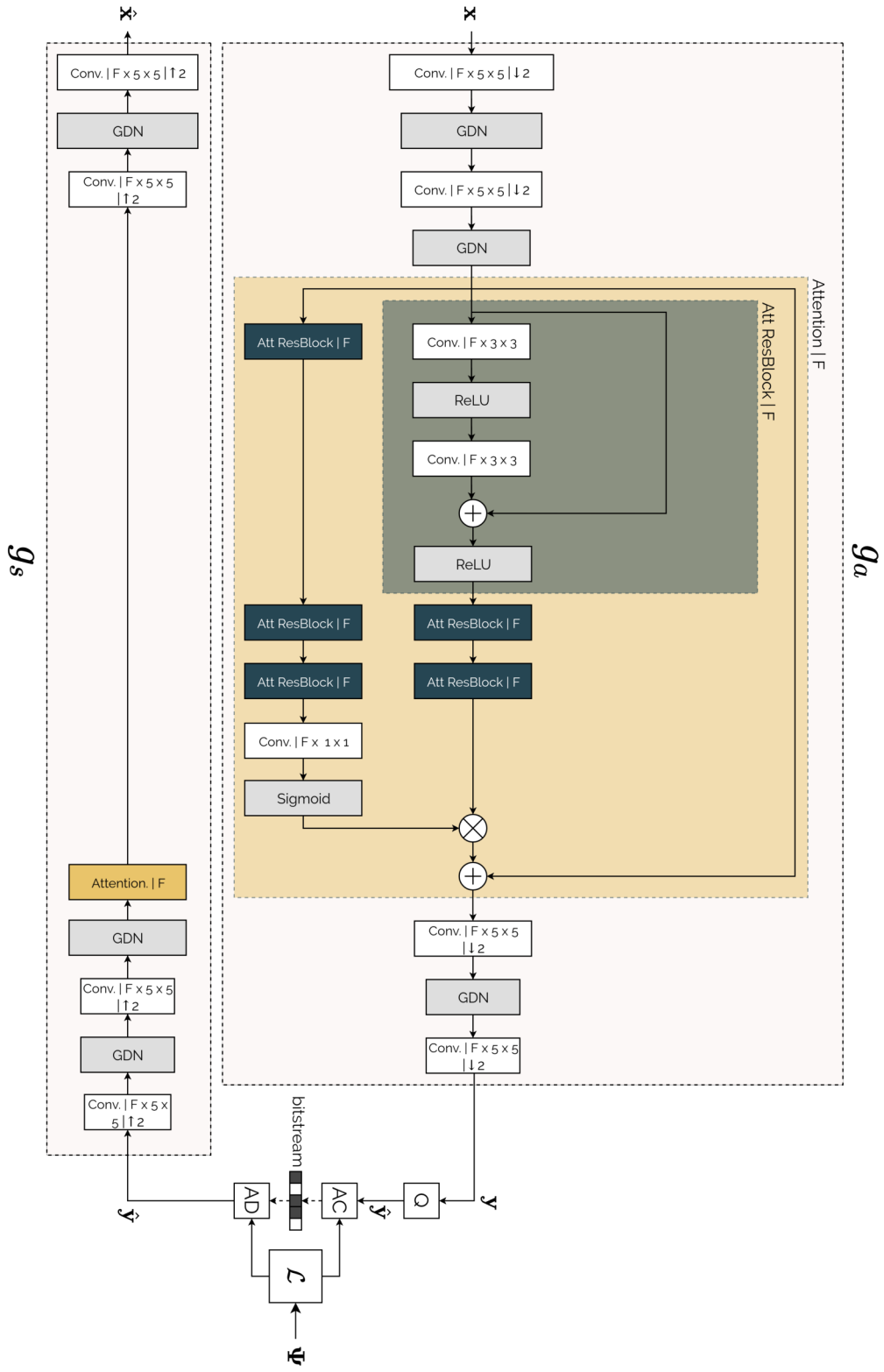


Figure D.1: The analysis and synthesis transforms g_a and g_s. The number of internal features F is set to 192, the number of latent features F_y is set to 256.

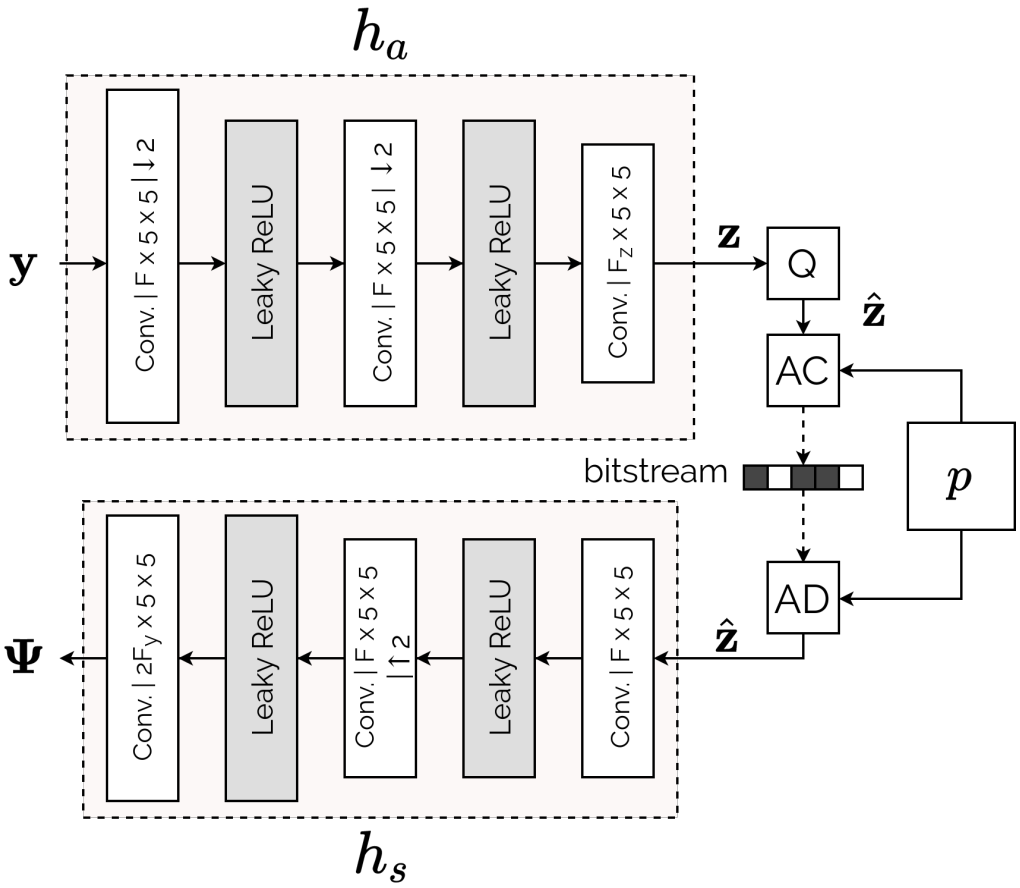


Figure D.2: The hyperprior analysis and synthesis transforms h_a and h_s. The number of internal features F is set to 192, the number of hyperprior features F_z is set to 64.

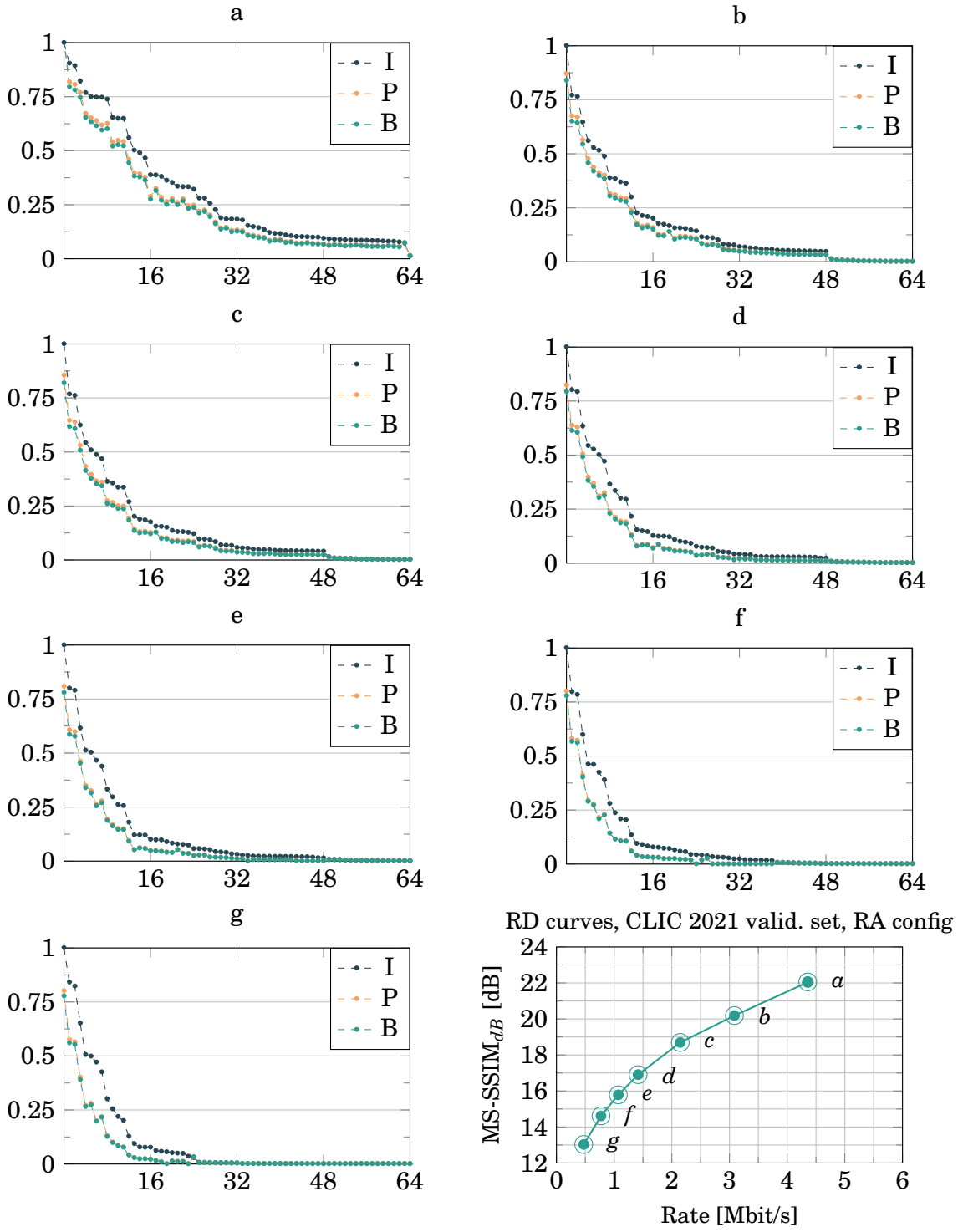


Figure D.3: Encoder quantization gains Γ_f^{enc} for different rate targets.

APPENDIX E

COMPREHENSIVE EXPERIMENTAL DETAILS FOR MNET

E.1 Architecture

This sections gives the detailed architecture of the MNet autoencoder. For convenience, the architecture of MNet is extremely similar to the one of CNet. Figure E.1 and E.2 presents the architecture of the different MNet transforms.

E.2 Additional examples of optical flows

Figure E.3 presents an additional example of optical flows. Here, the camera is zooming out of the man *i.e.* the entire background is moving in different directions, resulting in different colours on the flow visualisations. As the zoom out is centered on the man, the pixels on the borders of the images move faster than those in the center. This can be seen through the darker shade of the colours at the border of the optical flows. While the man's body remains still, its hands are moving vertically. This is well captured in the optical flows. Lastly, the text remains at the same place throughout the sequence. This explains the white areas at the bottom right of the optical flows. This example proves that the system is able to learn to estimate and transmits complex and multiple motions, as well as properly segmenting between different areas (background, the man's hands, the text).

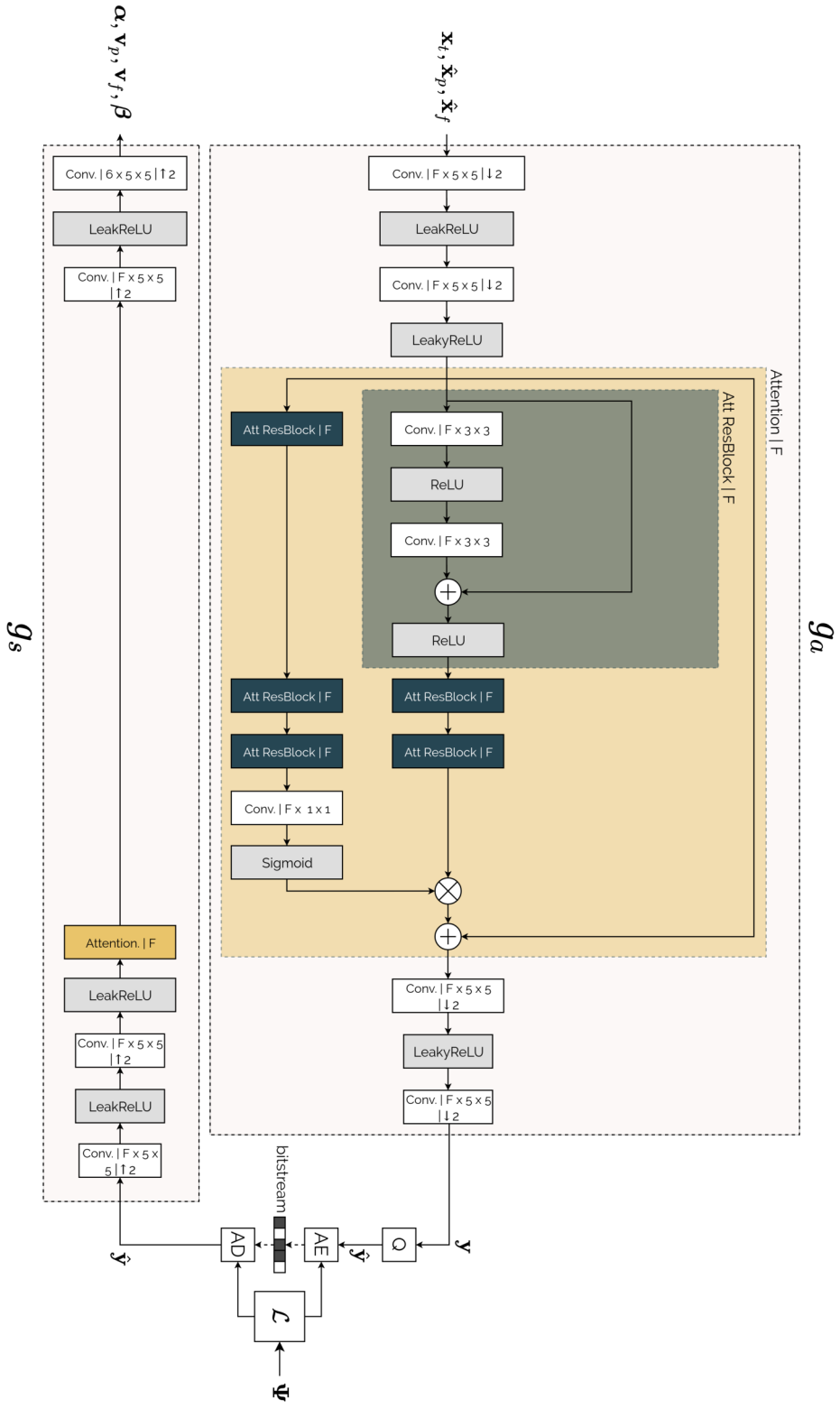


Figure E.1: MNet analysis and synthesis transforms g_a and g_s. The number of internal features F is set to 192, the number of latent features F_y is set to 64.

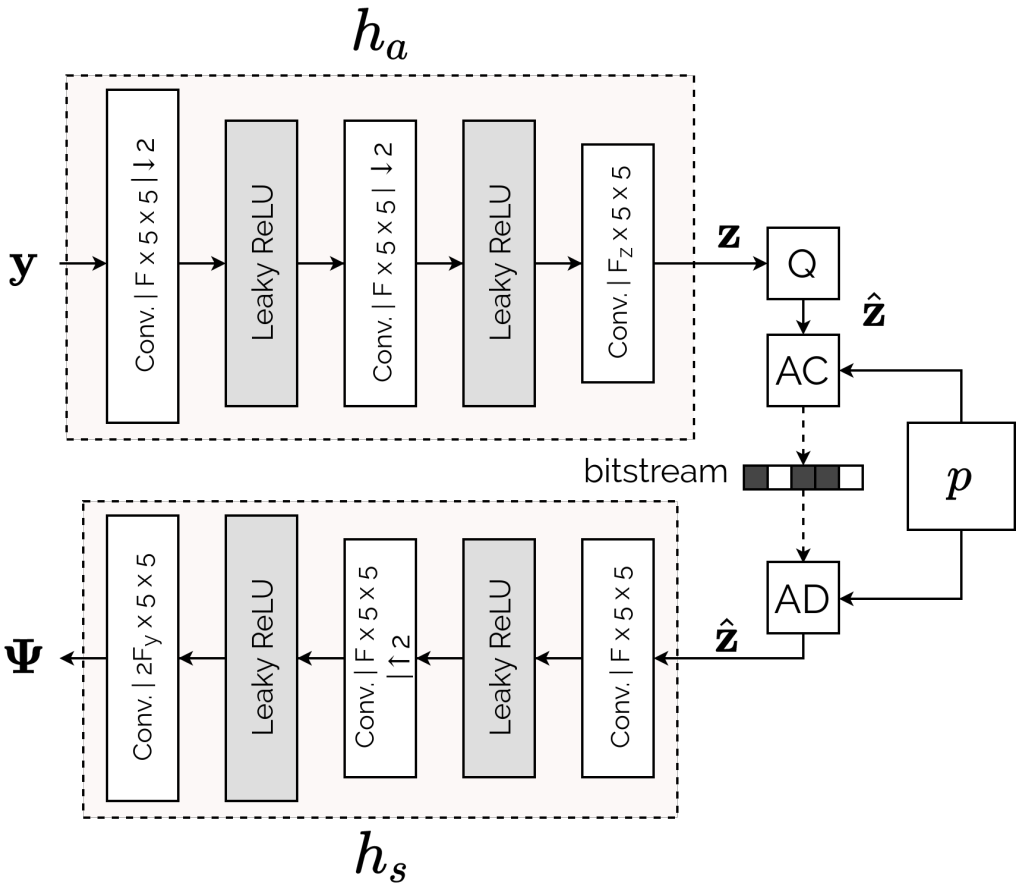
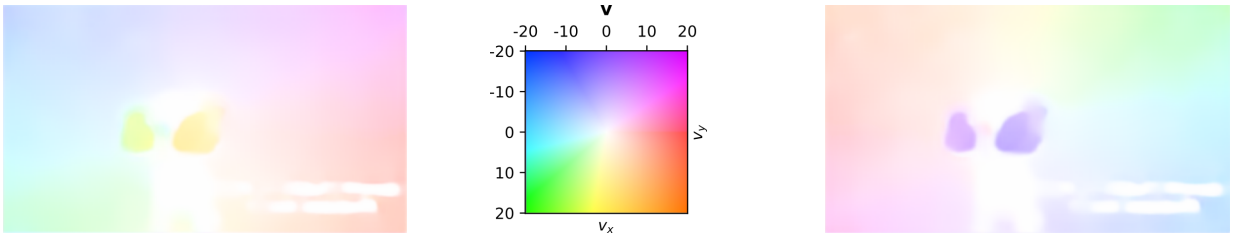


Figure E.2: MNet hyperprior analysis and synthesis transforms h_a and h_s. The number of internal features F is set to 192, the number of hyperprior features F_z is set to 16.

(a) Past reference $\hat{\mathbf{x}}_p$. (b) Original frame to code $\mathbf{x}_t$. (c) Future reference $\hat{\mathbf{x}}_f$.



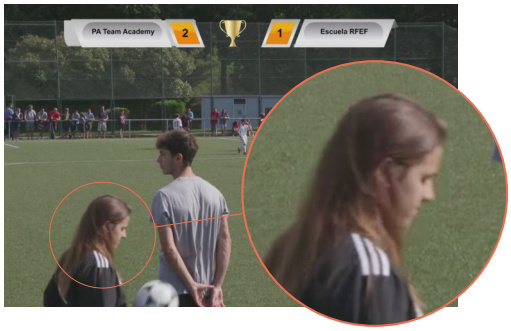
(d) Optical flow $\mathbf{v}_p$: motion from $\mathbf{x}_t$ to $\hat{\mathbf{x}}_p$. (e) Optical flow representation. Motion is expressed in pixel. (f) Optical flow $\mathbf{v}_f$: motion from $\mathbf{x}_t$ to $\hat{\mathbf{x}}_f$.

Figure E.3: Additional examples of optical flows from the sequence *Sports_1080P-08e1*, extracted from the CLIC21 validation set.

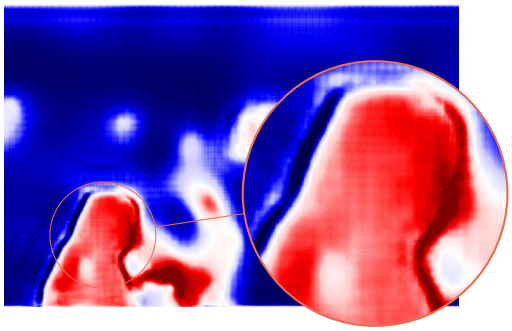
E.3 Bidirectional prediction weighting and disocclusions

Figure E.4 completes the example from Section 6.3.3, where the bidirectional prediction weighting $\boldsymbol{\beta}$ handles disocclusions. The girl in the foreground is moving from left to right. As such, the intermediate prediction (using a single reference) shown in Fig. E.4c and E.4d present visual degradations either at the left or at the right or the girl *i.e.* in the areas occluded in their respective reference frame. Combining the two intermediate predictions through $\boldsymbol{\beta}$ allows to select the non-occluded areas from each of them *i.e.* the areas without visual degradations. This behaviour is illustrated in Figures E.4e and E.4e. In the end, this results in a temporal prediction with fewer visual degradations, see Fig. E.4g.

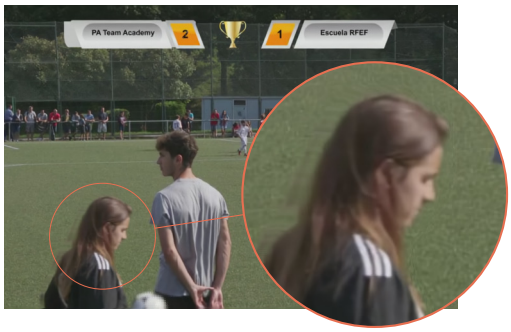
The zoom-out motion of the example presented in Fig. E.3 leads to an interesting behaviour for $\boldsymbol{\beta}$, illustrated in Fig. E.5. Most of the frame uses a $\boldsymbol{\beta} \simeq 0.5$ which corresponds to computing the average of both intermediate motion compensations. Yet, the borders shows $\boldsymbol{\beta} \simeq 0$ *i.e.* the border areas rely only on the future reference frame $\hat{\mathbf{x}}_f$ to compute the final temporal prediction. This is due to the zoom-out motion, where the border pixels are not present in the past reference frame.



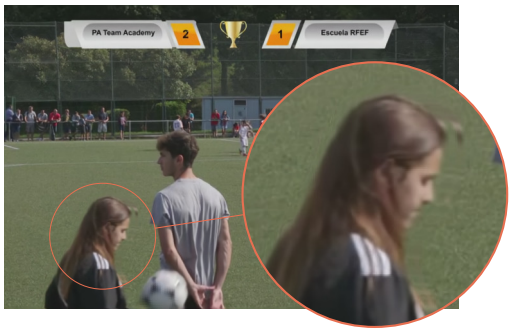
(a) Frame to predict $\mathbf{x}_t$.



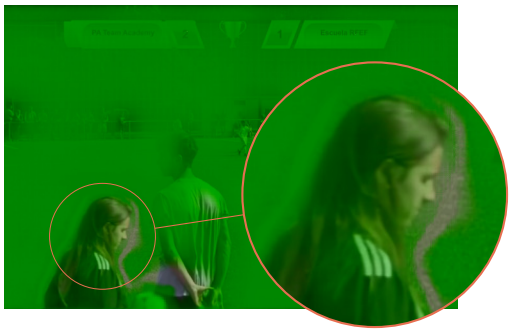
(b) Bidirectional prediction weighting $\boldsymbol{\beta}$.



(c) Intermediate motion comp. $w(\hat{\mathbf{x}}_p;\mathbf{v}_p)$.

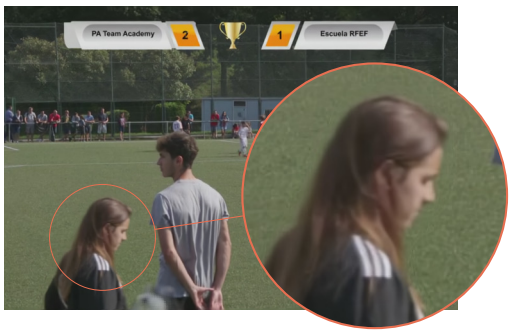


(d) Intermediate motion comp. $w(\hat{\mathbf{x}}_f;\mathbf{v}_f)$.



(e) Weighted motion comp. $\boldsymbol{\beta} \odot w(\hat{\mathbf{x}}_p;\mathbf{v}_p)$.

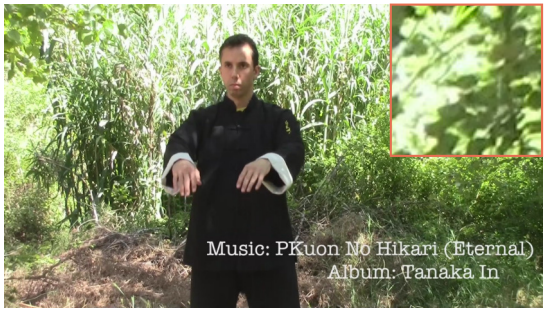
(f) Weighted motion comp. $\big(1-\boldsymbol{\beta}\big) \odot w(\hat{\mathbf{x}}_f;\mathbf{v}_f)$.



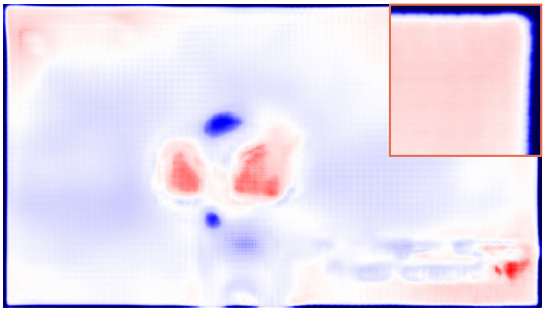
(g) Prediction $\tilde{\mathbf{x}}_t$, computed using Eq. (6.1).

(h) Prediction error $\mathbf{x}_t - \tilde{\mathbf{x}}_t$.

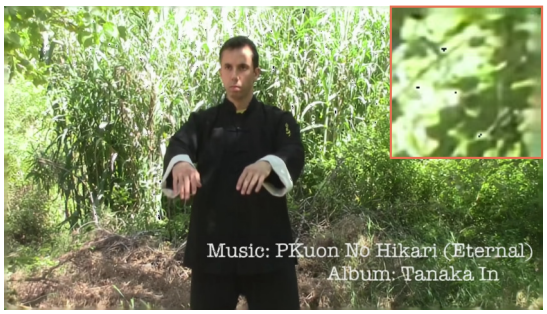
Figure E.4: Example of $\boldsymbol{\beta}$ handling a disocclusion.



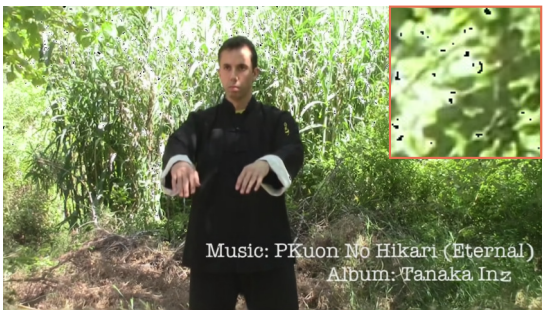
(a) Frame to predict $\mathbf{x}_t$.



(b) Bidirectional prediction weighting $\boldsymbol{\beta}$.



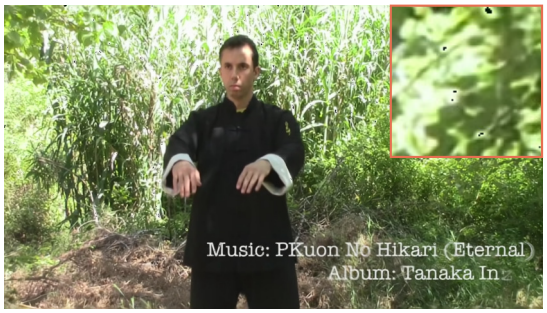
(c) Intermediate motion comp. $w(\hat{\mathbf{x}}_p; \mathbf{v}_p)$.



(d) Intermediate motion comp. $w(\hat{\mathbf{x}}_f; \mathbf{v}_f)$.

(e) Weighted motion comp. $\boldsymbol{\beta} \odot w(\hat{\mathbf{x}}_p; \mathbf{v}_p)$.

(f) Weighted motion comp. $\big(1 - \boldsymbol{\beta}\big) \odot w(\hat{\mathbf{x}}_f; \mathbf{v}_f)$.



(g) Prediction $\tilde{\mathbf{x}}_t$, computed using Eq. (6.1).

(h) Prediction error $\mathbf{x}_t - \tilde{\mathbf{x}}_t$.

Figure E.5: Additional examples of $\boldsymbol{\beta}$ performing dissoclusion.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

INSA RENNES

Titre : Développement de schémas de compression vidéo basés apprentissage

Mot clés : Compression vidéo, apprentissage profond, traitement du signal

Résumé : La quantité toujours croissante d'images et de vidéos échangées sur Internet a un impact substantiel sur le changement climatique. Cet impact peut-être réduit via le développement de meilleurs algorithmes de compression, qui visent à réduire le volume de données représentant les vidéos, tout en conservant une qualité acceptable pour l'utilisateur. Depuis les années 90, des standards de compression vidéo ont été conçus afin de diminuer la taille des vidéos via une succession d'opérations linéaires. Ces standards sont développés de manière incrémentale, entraînant une optimisation séparée des différentes opérations. Récemment, les réseaux de neurones ont émergé comme une réponse pertinente à un grand nombre de problématiques, grâce à leur capacité à apprendre des fonctions non-linéaires au travers d'une optimisation de bout-en-bout. Si la compression neuronale constitue d'ores et déjà l'état de l'art pour le codage d'images fixes, le codage vidéo demeure une tâche plus difficile. Cette thèse propose de concevoir un codeur vidéo basé apprentissage, afin de tirer profit des capacités prometteuses des réseaux de neurones. Étant donné la nouveauté de ce domaine, la conception du codeur démarre d'une page blanche, et les différents éléments le composant sont étudiés en détail. Au final, l'évaluation du codeur proposé sur une tâche de codage vidéo réaliste montre qu'il présente des performances compétitives avec des standards de compression modernes.

Title: Design of learned video coding schemes

Keywords: Video Compression, Deep Learning, Signal Processing

Abstract: The ever-growing amount of images and videos conveyed over the Internet has a substantial impact on the climate change. This impact can be mitigated through the design of better compression algorithms, which aims to reduce the size of videos while maintaining an acceptable quality for the user. Since the 90s, video coding standards have been devised to reduce the video size through a succession of linear operations. Video coding standards are incrementally designed, leading to a separate optimization of the different operations. Recently, neural networks have emerged as a relevant solution for a wide variety of issues, due to their ability to learn non-linear functions through an end-to-end optimization process. While learned approaches already achieve state-of-the-art performance for still image compression, video coding remains a more challenging task. This thesis proposes to design a learned video coder, to leverage the promising abilities of neural networks. Due to the novelty of this field, the design of the proposed learned coding scheme starts from a blank page. The different elements of the coder are thoroughly considered. In the end, it is shown that the proposed learned coder is able to achieve performance equivalent to modern video coding standards in a realistic video coding setup.