

# Évaluation de la spécialisation d'auto-encodeurs par catégories pour la compression d'images

Julien Roux, Majeed Abdul Baki, Victor Gilbert

Projet Final Hiver 2025 - INF8225 Polytechnique Montréal, Présenté au Prof. Christopher Pal

Code disponible : <https://github.com/Julien9969/INF8225-projet>

**Abstract** - Dans ce travail, nous mettons l'architecture d'auto-encodeur (AE) [2] et d'auto-encodeur variationnel (VAE) [5] à l'épreuve pour des tâches de compression d'images du dataset Image-NET [8] de taille 128 par 128, divisé en 3 catégories, des photos de pizzas (avec beaucoup de ressemblances), de chiens (certains motifs similaires) et des images variées. Nous montrons que, pour notre ensemble de données de taille moyenne, un auto-encodeur est plus performant qu'un auto-encodeur variationnel. Notre étude évalue ensuite plusieurs architectures d'auto-encodeurs en faisant varier la dimension latente de l'auto-encodeur, nous permettant d'isoler le meilleur compromis entre taux de compression et qualité mesurée par les métriques de PSNR et SSIM, tout en comparant les performances à la méthode de compression traditionnelle JPEG. Nous avons également pu montrer que pour les images à forte cohésion et similarité, comme des pizzas, un modèle AE spécialisé sur cette classe d'images lors de son entraînement atteint des images reconstituées de meilleure qualité que le modèle généralisé (entraîné sur les images variées), tandis que ce dernier produit une meilleure qualité d'images reconstituées sur les données de chien que le modèle spécialisé sur des images de chiens, et sur les images variées.

## 1 Introduction

De nos jours, la compression des images est un enjeu important face à l'augmentation exponentielle des données à stocker. Les auto-encodeurs (AE) et auto-encodeurs variationnels (VAE) sont des architectures de réseau de neurones qui permettent d'implémenter une fonction de compression notamment pour des images. Ce modèle va être entraîné à compresser les données en les réduisant à une représentation plus compacte, puis à les reconstruire à partir de cette représentation. On peut alors se demander dans quelle mesure, utiliser des modèles spécialisés pour un certain type d'image permet d'obtenir une meilleure compression et restitution de l'image originale.

## 2 Revue de littérature

La compression d'image est un domaine crucial en traitement de l'information, avec plusieurs algorithmes développés avant l'avènement des réseaux de neurones, comme Huffman, ainsi que DCT et DWT utilisés respectivement dans JPEG et JPEG 2000 [1]. Dans ce

contexte, les auto-encodeurs apparaissent comme une alternative prometteuse, notamment pour la compression avec perte, grâce à leur capacité à apprendre des représentations latentes compactes via un architecture "bottleneck" ou goulot d'étranglement [2].

La compression d'images par auto-encodeur a atteint des taux de compression comparables aux méthodes telle JPEG dans la dernière décennie, tel que démontré par Petscharnig et al. [6] dont la méthode de compression par auto-encodeur convolutionnel a même pu battre JPEG2000, bien qu'un entraînement complexe soit nécessaire. L'année suivante, les progrès et expérimentations sur les architectures d'auto-encodeurs ont continué avec des améliorations comme l'intégration d'analyse en composantes principales (ACP) au modèle apportée par Cheng et al. [7].

Des travaux récents, comme ceux de Kashyap et al. (2024), vont plus loin en combinant compression et sécurité pour des applications de transmission de données. En effet, le modèle encode une image dans une représentation latente difficilement interprétable, offrant une protection implicite [4] semblable à de l'encryption en plus des avantages de réduction du temps de transfert offert par la compression.

Parallèlement, les auto-encodeurs variationnels (VAE) ont suivi, introduisant une dimension probabiliste en modélisant l'espace latent comme une distribution, ce qui permet une meilleure régularisation et un contrôle sur la génération des représentations compressées. Toutefois, ils souffrent parfois de limitations pratiques, comme une qualité de reconstruction inférieure due à une tendance à générer des images floues ou au phénomène de "KL divergence", qui affaiblit l'apprentissage de la représentation latente [5].

## 3 Théorie et objectifs

Les travaux antérieurs discutés précédemment ont pu démontrer la force des AE et les avantages de la composante variationnelle. Cependant un trait commun de modèles aussi complexes est qu'ils demandent beaucoup de données et de ressources pour l'entraînement, bien l'apprentissage soit non supervisé.

Notre projet cherche, premièrement, à déterminer l'architecture optimale du modèle pour chaque ensemble de données, ce qui correspond aux hyperparamètres de profondeur et de dimension latente du bottleneck de

l'auto-encodeur et de l'auto-encodeur variationnel permettant une compression efficace tout en minimisant la perte d'information.

Deuxièmement, nous évaluons empiriquement l'impact de la spécialisation au contenu des images sur les performances de compression. L'hypothèse que nous souhaitons tester est si un AE peut avoir une compression plus importante en apprenant des motifs spécifiques à une classe d'objets par rapport à un dataset d'images variées.

Les ensembles d'images sélectionnés sont :

- **Pizzas** (contenu homogène, peu de variabilité)
- **Chiens** (variabilité modérée)
- **Images variées** (contenu hétérogène)

Pour chaque dataset, nous testons plusieurs configurations du modèle, en variant initialement la profondeur, c'est-à-dire le nombre de couches convolutives, puis principalement la dimension du bottleneck.

L'objectif est de mesurer dans chaque cas :

- La qualité de reconstruction via les métriques de PSNR et SSIM
- Le quotient de compression (de combien de fois notre image est plus petite)

## 4 Méthodologie

### 4.1 Données et prétraitement

Les trois jeux de données ont été constitués à partir d'Image-NET [8] [9], un dataset populaire pour des applications d'intelligence artificielle. Nous avons divisé chaque jeu de données en deux parties, l'une pour l'entraînement contenant 90% des images, puis l'une pour la validation contenant les 10% d'images restantes.

Des augmentations simples ont été appliquées sur le dataset d'entraînement pour augmenter le nombre d'images de celui-ci. Nous avons appliqué à chaque image des rotations de 90 degrés, ainsi qu'un facteur d'agrandissement (zoom) de 1, c'est-à-dire identique à l'image originale, et de 1.2. La combinaison de ces deux opérations nous ont donc permis d'avoir 8 images à partir de chaque image source.

Enfin, chaque image est redimensionnée en  $128 \times 128$  pixels avant l'entraînement et convertie en tenseur à trois canaux (R, G, B) pour une meilleure compatibilité avec PyTorch [10].

### 4.2 Entraînement du modèle

Les modèles ont été entraînés avec l'optimiseur Adam et un taux d'apprentissage de 0.001 (soit  $10^{-3}$ ) et une taille de batch de 64, pendant 150 époques. Ces paramètres ont

montré une convergence stable et rapide, tout en restant compatible avec des contraintes matérielles notamment en termes de mémoire sur les cartes graphiques (GPU) utilisées.

### 4.3 Fonction de perte

La fonction de perte est le MSE (erreur quadratique moyenne) entre l'image originale et sa reconstruction, auquel est ajouté le résidu de reconstruction, tel que décrit par Naveen et al. [4].

### 4.4 Évaluation

Les performances des modèles sont évaluées avec :

- **PSNR** (*Peak Signal-to-Noise Ratio*) une mesure classique de la fidélité entre deux images
- **SSIM** (*Structural Similarity Index*), plus représentatif de la perception humaine, car il prend en compte la luminance, le contraste et la structure.
- Le **quotient de compression**, calculé par rapport à la taille de l'image non compressée.

Ci-dessous la formule permettant de calculer le quotient de compression entre l'image non compressée et la représentation latente en sortie de l'encodeur.

$$H \times W \times R \div (\frac{H}{2^{C-1}} \times \frac{W}{2^{C-1}} \times b) \\ = R \times \frac{2^{2(C-1)}}{b}$$

Équation 1: Calcul du quotient de compression à partir des paramètres de l'image et du modèle

L'Équation 1 utilise les variables suivantes:

- **H** : hauteur de l'image,
- **W** : largeur de l'image,
- **R** : nombre de canaux (RGB: 3)
- **C** : Nombres de couches de convolution (-1 car la dernière couche a un pas, ou *stride*, de 1),
- **b** : Valeur de bottleneck (ou dimension latente)

## 5 Architecture

Nous avons d'abord exploré une architecture VAE [5], mais les résultats se sont révélés décevants : les reconstructions étaient souvent floues, avec des déformations marquées des couleurs, tel que montré sur la Figure 1 qui suit. Bien que des techniques existent pour atténuer ces effets, elles impliquent une complexité supplémentaire et un temps de développement important. Nous avons donc préféré nous concentrer sur des auto-encodeurs classiques, dont l'architecture plus simple offre de bonnes performances de reconstruction, suffisantes pour nos objectifs d'évaluation.

L'architecture auto-encodeur [4] retenue est composé de deux parties :

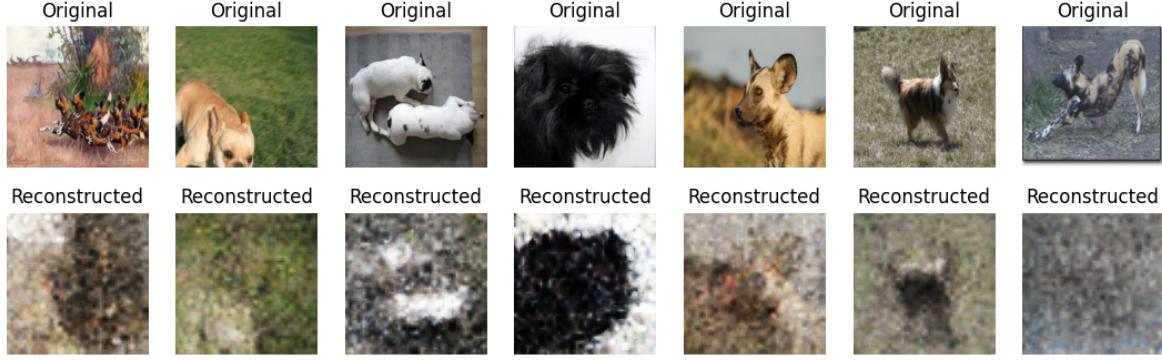


Figure 1 : Images originales et reconstruites par un modèle VAE simple

→ L'encodeur transforme l'image d'entrée en un vecteur latent de faible dimension à l'aide de plusieurs couches de convolution successives.

→ Le décodeur a pour objectif de reconstruire l'image initiale à partir de cette représentation latente. Il repose sur des couches de *transposed convolutions* symétriques à l'encodeur qui permettent de restaurer la structure spatiale de l'image en augmentant la résolution à chaque étape. Une fonction d'activation sigmoïde est appliquée en sortie, afin que les valeurs des pixels générés se situent dans un intervalle entre 0 et 1.

Chaque couche de convolution avec un *stride* de 2 divise la résolution en largeur et hauteur de l'image par 2, permettant une réduction progressive de la dimensionnalité. Pour une image de taille  $128 \times 128$ , trois convolutions successives réduisent cette taille à  $16 \times 16$ , avant projection dans un espace latent de dimension réduite contrôlée par l'hyper paramètre du bottleneck.

La diminution de la valeur du bottleneck permet d'atteindre un meilleur taux de compression, mais cela peut se faire au détriment de la qualité de reconstruction. L'enjeu est donc de trouver un équilibre optimal entre taux de compression et fidélité de l'image reconstituée, afin de voir si l'entraînement d'un modèle spécialisé permet d'atteindre une meilleure

compression pour une qualité similaire par rapport à un modèle généraliste.

## 6 Présentations des résultats

Tous les paramètres de *batch* ainsi que la normalisation des jeux de données, présentés lors de la méthodologie, sont gardés identiques à travers les différentes expériences pour avoir des résultats comparables.

La première étape de nos expériences a été de déterminer les hyperparamètres de notre architecture. Nous avons notamment comparé différents nombres de couches pour notre Auto-Encodeur Convolutif (CAE), pour déterminer le nombre de couches idéal permettant de compresser nos images avec une dégradation raisonnable.

Nous remarquons, grâce aux résultats présentés à la Table I, qu'avoir un modèle avec 5 couches de convolutions dégrade significativement la qualité des images, notamment par la métrique de SSIM qui est en moyenne à 0.88 et peut même atteindre 0.53 pour certaines images. Nous avons alors décidé, dans la suite des tests, de laisser tomber l'architecture à 5 couches convolutives. Les figures de la page suivante confirment visuellement l'effet décrit, notamment la comparaison entre les modèles à 3 et 4

Nombre de couches convolutives	Bottleneck	PSNR				SSIM				Quotient de compression
		Moyenne	STD	MIN	MAX	Moyenne	STD	MIN	MAX	
3	128	44.37	3.55	34.85	55.95	0.99	0	0.96	1	0.375
4	<b>128</b>	<b>35.143</b>	<b>3.328</b>	<b>26.374</b>	<b>48.805</b>	<b>0.961</b>	<b>0.021</b>	<b>0.857</b>	<b>0.997</b>	<b>1.5</b>
5	128	29.4	3	21.47	43.53	0.88	0.05	0.53	0.99	6

Table I : Comparaison des reconstructions en fonction du nombre de couche de convolution  
Dataset Chien

couches (Figures 3 et 4), dont la qualité est meilleure que celle du modèle à 5 couches (Figure 5).

Concernant le modèle à 3 couches convolutives, bien que la qualité des images reconstituées est supérieure, le nombre de couches n'est pas suffisant pour effectuer une compression de l'image avec 128 filtres, tel que montré par le quotient de compression de 0.375 inférieur à 1.

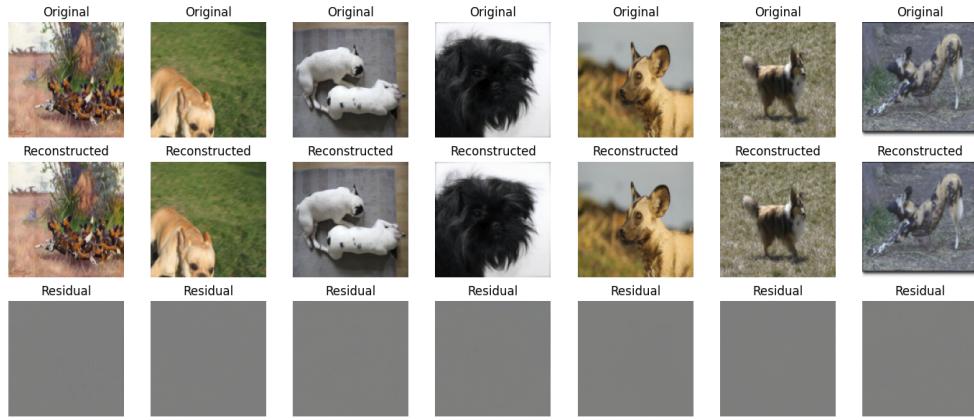


Figure 3 : Images originales, reconstruites et résidus pour un AE à 3 couches (Dataset Chiens)

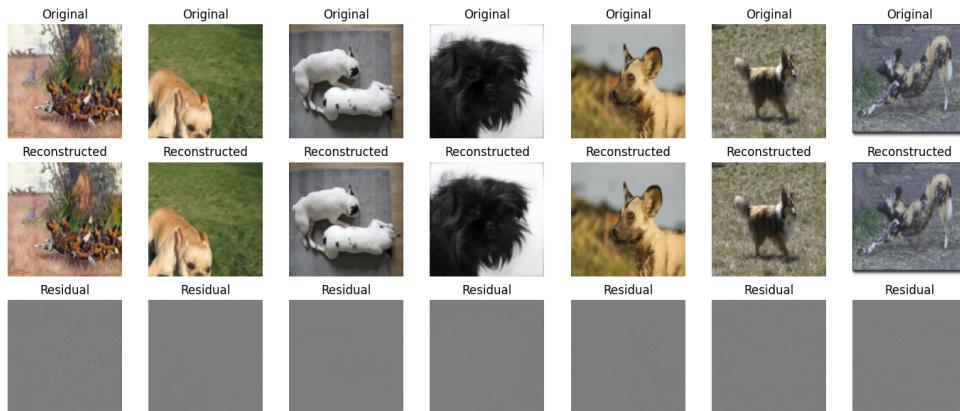


Figure 4 : Images originales, reconstruites et résidu pour un AE à 4 couches

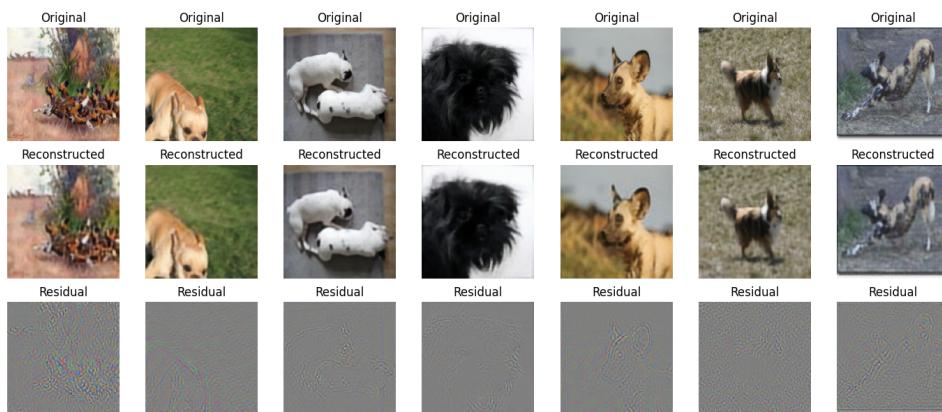


Figure 5 : Images originales, reconstruites et résidu pour un AE à 5 couches

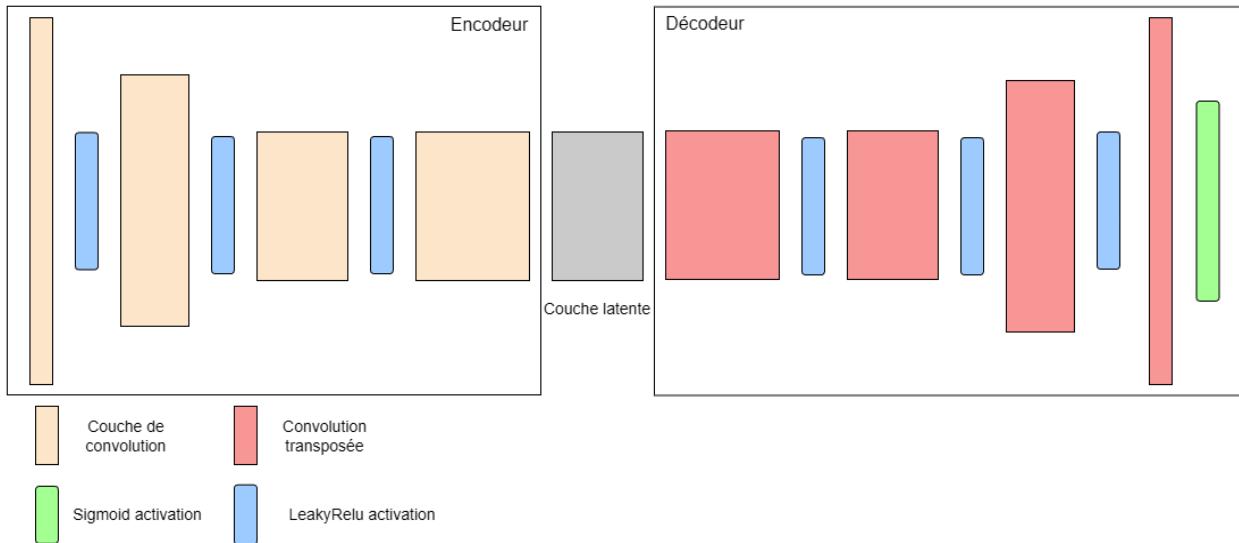


Figure 2 : Schéma de l'architecture d'auto-encodeur retenue

Nous avons donc décidé d'utiliser une architecture d'Auto-Encodeur à 4 couches convolutives, que nous présentons à la Figure 2 incluant les fonctions d'activation et types de convolutions.

Les résultats présentés dans les Tables II, III et IV permettent d'évaluer l'impact de la taille du bottleneck et du type de données sur la qualité de reconstruction. On observe systématiquement qu'une augmentation de la taille du bottleneck améliore à la fois le PSNR et le SSIM. Ce comportement est attendu : un bottleneck plus large réduit la compression appliquée, permettant une meilleure préservation de l'information visuelle. Par exemple, pour le dataset "chien", le PSNR passe de 34.014 (bottleneck 32) à 35.143 (bottleneck 128), et le SSIM de 0.952 à 0.961. Des tendances similaires sont visibles pour les datasets "pizza" et "images variées". Cela confirme que la capacité de reconstruction du modèle est corrélée à la taille du bottleneck .

Cependant, quelle que soit la taille du bottleneck, l'algorithme JPEG reste supérieur en termes de qualité de reconstruction, avec des PSNR et SSIM systématiquement plus élevés, comme l'indiquent les résultats de la Table V. L'avantage moyen en PSNR varie de 1.015 à 1.144 fois celui de l'auto-encodeur selon les datasets et tailles de bottleneck, et un avantage en SSIM plus modéré (proche de 1.015 à 1.035).

Il est important de noter que les différences de PSNR, bien que parfois modestes en valeur absolue, doivent être interprétées avec prudence car le PSNR est exprimé en décibels, une échelle logarithmique : de petites différences

peuvent correspondre à des écarts significatifs en termes de qualité perçue.

Concernant le quotient de compression, JPEG présente également un meilleur compromis, en atteignant un quotient de 6.12 pour une qualité conservée équivalente à la version auto-encodeur avec un bottleneck 96 / 128 qui ont un quotient de compression bien plus faible.

En ce qui concerne le type de données, l'analyse par dataset révèle que les performances sont relativement homogènes, quel que soit le type d'images. Sur les données "chiens", le SSIM maximal atteint 0.978, légèrement supérieur aux données "images variées" (0.974) et aux "pizzas" (0.958).

Cependant, les écarts restent faibles (moins de 0.02 en valeur absolue), ce qui indique que la nature spécialisée des données n'offre pas d'avantage clair et systématique. En particulier, le dataset "pizzas" montre des résultats légèrement inférieurs, probablement en raison de la grande similarité visuelle entre les images, rendant l'apprentissage de détails fins plus complexe pour un modèle généraliste.

Lors de la validation croisée dans les Tables VI, VII, le modèle entraîné sur les "images variées" montre les meilleures capacités de généralisation, sur les données "chiens", il atteint un SSIM moyen de 0.97, supérieur au modèle spécialisé "chiens" (0.961). Sur les données "pizzas", il obtient 0.95, légèrement inférieur au modèle spécialisé "pizzas" (0.957).

#	Nombre de couches	Bottleneck	PSNR				SSIM				Quotient de compression
			Moyenne	STD	MIN	MAX	Moyenne	STD	MIN	MAX	
1	4	32	34.014	3.207	25.456	45.248	0.952	0.026	0.835	0.997	6
2	4	48	34.343	3.296	25.639	47.667	0.954	0.024	0.841	0.997	4
3	4	64	34.156	3.264	25.466	47.425	0.953	0.025	0.838	0.996	3
4	4	96	35.139	3.314	26.493	48.44	0.961	0.021	0.856	0.997	2
5	4	128	35.143	3.328	26.374	48.805	0.961	0.021	0.857	0.997	1.5
6	JPEG	X	38.621	2.801	30.432	50.874	0.978	0.011	0.893	0.997	6.12

Table II : Résultats des reconstructions pour le dataset chien sur l'ensemble de validation

#	Nombre de couches	Bottleneck	PSNR				SSIM				Quotient de compression
			Moyenne	STD	MIN	MAX	Moyenne	STD	MIN	MAX	
1	4	32	31.28	2.30	26.08	38.34	0.937	0.023	0.768	0.975	6
2	4	48	31.85	2.3	26.34	38.88	0.944	0.022	0.779	0.977	4
3	4	64	33.89	2.37	27.34	40.71	0.963	0.018	0.817	0.986	3
4	4	96	34	2.35	27.35	40.67	0.964	0.017	0.816	0.986	2
5	4	128	33.16	2.32	27.04	39.95	0.957	0.019	0.806	0.983	1.5
6	JPEG	X	34.15	1.85	30.07	38.32	0.958	0.019	0.848	0.988	6.12

Table III : Résultats des reconstructions pour le dataset pizza sur l'ensemble de validation

#	Nombre de couches	Bottleneck	PSNR				SSIM				Quotient de compression
			Moyenne	STD	MIN	MAX	Moyenne	STD	MIN	MAX	
1	4	32	33.042	3.856	25.542	45.926	0.941	0.031	0.806	0.994	6
2	4	48	34.662	3.889	26.874	46.864	0.957	0.024	0.847	0.995	4
3	4	64	34.664	3.715	26.998	44.727	0.959	0.024	0.849	0.995	3
4	4	96	34.218	3.847	26.567	46.599	0.953	0.026	0.838	0.995	2
5	4	128	35.008	3.871	27.119	47.89	0.96	0.022	0.858	0.996	1.5
6	JPEG	X	37.806	3.611	29.782	49.169	0.974	0.014	0.928	0.997	6.12

Table IV : Résultats des reconstructions pour le dataset images variées sur l'ensemble de validation

Données	Bottleneck	JPEG est X fois meilleur		
		PSNR	SSIM	
Chien	128	1.099	1.018	
	32	1.135	1.027	
	48	1.125	1.025	
Pizza	128	1.031	1.001	
	32	1.092	1.022	
	48	1.072	1.014	
Aléatoire	128	1.015	1.015	
	32	1.144	1.035	
	48	1.091	1.018	

Table V : Comparaison de résultats des reconstructions par rapport à JPEG

Modèle:		PSNR				SSIM			
		Bottleneck	Moyenne	STD	MIN	MAX	Moyenne	STD	MIN
Varié (repris de la Table IV)	128	35.008	3.871	27.119	47.89	0.96	0.022	0.858	0.996
Chiens	128	34.36	3.97	26.7	47.81	0.95	0.03	0.83	0.999
Pizza	128	34.815	3.752	26.986	45.801	0.959	0.023	0.85	0.995

Table VI : Résultats validation sur le dataset images variés

Validation sur les données:	Bottleneck	PSNR				SSIM			
		Moyenne	STD	MIN	MAX	Moyenne	STD	MIN	MAX
Chiens	128	35.62	3.25	26.74	47.93	0.97	0.02	0.87	0.99
Pizzas	128	32.29	2.29	26.83	39.54	0.95	0.02	0.81	0.98

Table VII : Résultats de validation avec le modèle images variés sur les autres datasets

Cela suggère que le modèle généraliste, entraîné sur des données diversifiées, est plus robuste et polyvalent, tout en maintenant des performances compétitives sur des domaines spécifiques. Toutefois, pour des cas où les images présentent des motifs très homogènes (comme les pizzas), un modèle spécialisé peut conserver un léger avantage.

## Conclusion et ouverture

Nos expérimentations ont permis de montrer que les auto-encodeurs, même dans une architecture relativement simple, peuvent atteindre des performances intéressantes en compression d'image. Toutefois, malgré l'hypothèse initiale selon laquelle la spécialisation des modèles sur des jeux de données cohérents (ex. uniquement des photos de chiens ou de pizzas) améliorerait significativement la compression pour une meilleure qualité conservée, les résultats ont montré que les modèles entraînés sur des données variées semblent un peu plus performants mais surtout plus flexibles. Une des raisons possibles à cette observation pourrait être le manque de filtrage rigoureux des images dans les datasets spécialisés: certaines photos contiennent des éléments peu pertinents pour la catégorie (comme des humains sur des photos de chiens), ce qui pourrait avoir réduit la capacité du modèle à apprendre les représentations spécifiques. Il est également notable que nos modèles n'atteignent pas encore les performances de standards industriels comme JPEG en termes de rapport compression/qualité, ce qui peut être attribué à la simplicité de l'architecture auto-encodeur utilisée. Pour approfondir cette étude, il serait pertinent de tester des architectures plus avancées comme les compressive autoencoders [3] et d'effectuer un prétraitement rigoureux du jeu de données.

## Répertoire du code

Github : <https://github.com/Julien9969/INF8225-projet>

## References

- [1] IEEE, "Techniques traditionnelles de compression d'image," IEEE Xplore, 2003. [En ligne]. Disponible sur : <https://ieeexplore.ieee.org/document/10212295>
- [2] Umberto Michelucci, "An Introduction to Autoencoders," arXiv, 2022. [En ligne]. Disponible sur : <https://arxiv.org/abs/2201.03898>
- [3] Lucas Theis, Wenzhe Shi, Andrew Cunningham, et Ferenc Huszár, "Lossy Image Compression with Compressive Autoencoders," arXiv, 2017. [En ligne]. Disponible sur : <https://arxiv.org/abs/1703.00395>
- [4] Aryan Kashyap Naveen, Sunil Thunga, Anuhya Murki, Mahati A Kalale, et Shriya Anil, "Autoencoded Image Compression for Secure and Fast Transmission," arXiv, 2024. [En ligne]. Disponible sur : <https://arxiv.org/abs/2407.03990v1>
- [5] Diederik P. Kingma et Max Welling, "An Introduction to Variational Autoencoders," Foundations and Trends® in Machine Learning, 2019. [En ligne]. Disponible sur : <https://arxiv.org/abs/1906.02691>
- [6] S. Petscharnig, M. Lux, et S. Chatzichristofis, "Dimensionality reduction for image features using deep learning and autoencoders," dans le document Proceedings, pages 1-6, 2017.
- [7] Cheng et al., "Deep convolutional autoencoder-based image compression method," arXiv, 2018. [En ligne]. Disponible sur : <https://arxiv.org/abs/1806.05662>

- [8] Stanford Vision Lab, Stanford University, Princeton University, “ImageNet,” [En ligne]. Disponible sur : <https://www.image-net.org>
- [9] “ImageNet Object Localization Challenge,” Données d’ImageNet accessible sur la plateforme Kaggle, 2025. [En ligne]. Disponible sur : <https://www.kaggle.com/c/imagenet-object-localization-challenge/data>
- [10] PyTorch, librairie Python, 2024, disponible sur <https://pytorch.org/>