

Chapitre 1. Modélisation générative

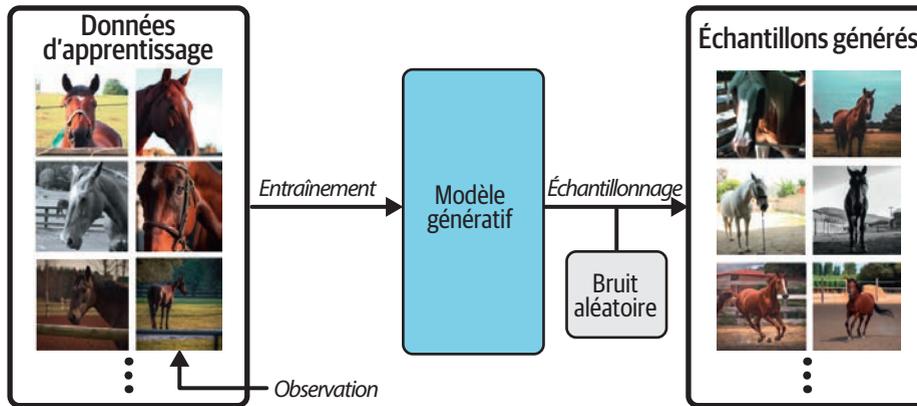


Figure 1.1 : Modèle génératif entraîné à générer des photos réalistes de chevaux.

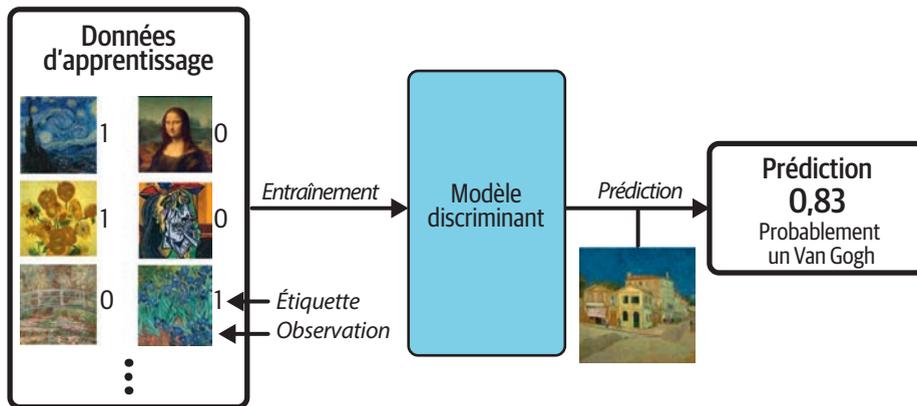


Figure 1.2 : Modèle discriminant entraîné à prédire si une image donnée a été peinte par Van Gogh.

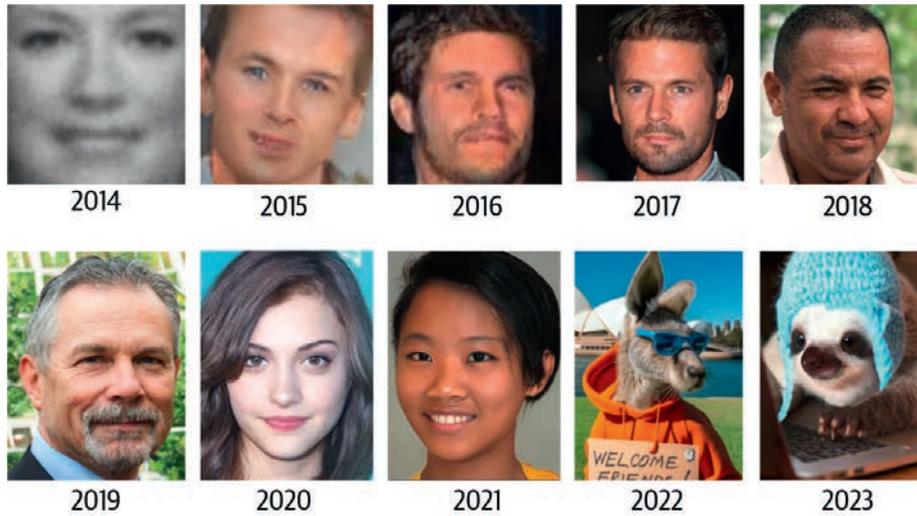


Figure 1.3 : La génération de visages à l'aide de la modélisation générative s'est considérablement améliorée au cours de la dernière décennie (adapté de Brundage et al., 2018).

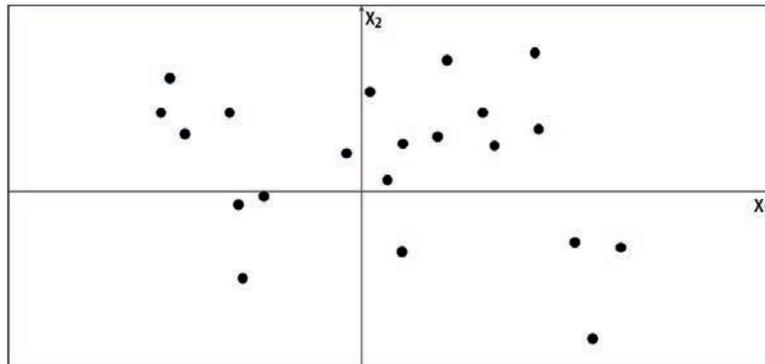


Figure 1.4 : Ensemble de points en deux dimensions, généré par une règle inconnue p_{data} .

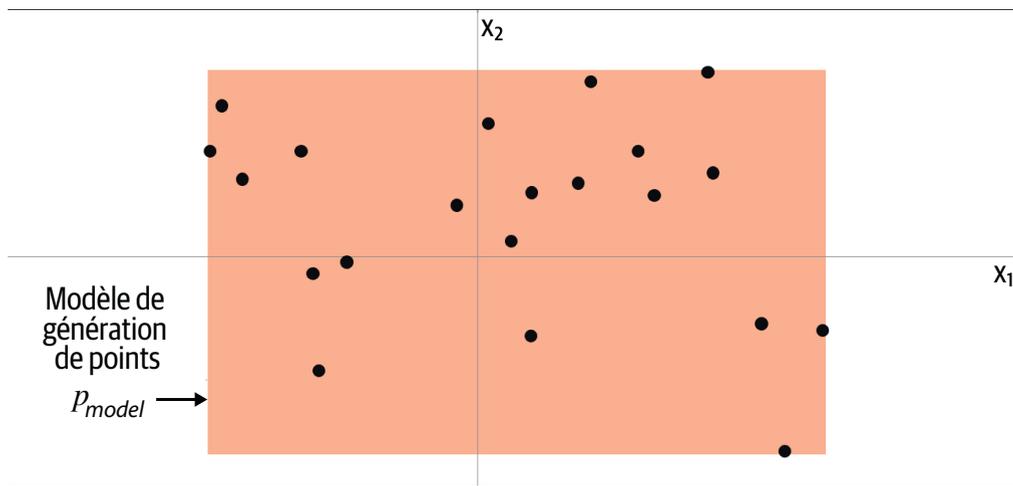


Figure 1.5 : La boîte orange, p_{model} est une estimation de la véritable distribution génératrice de données, p_{data} .

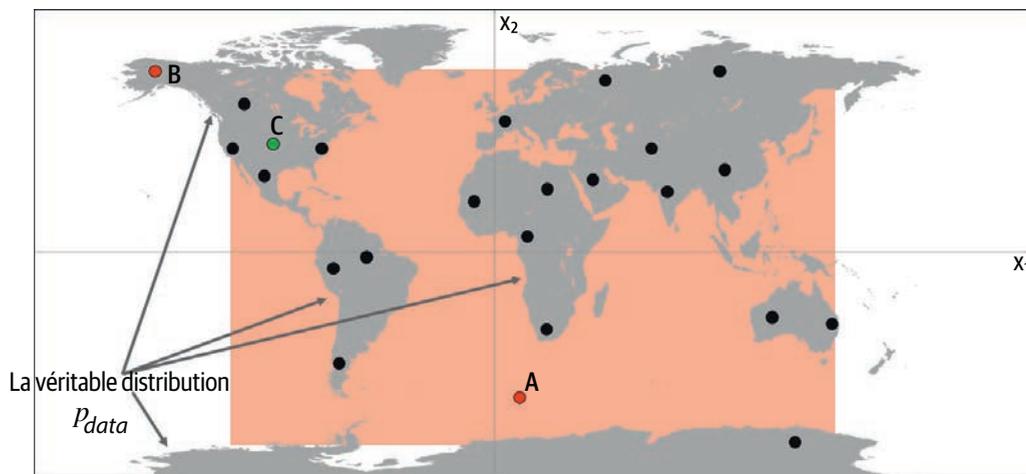


Figure 1.6 : La boîte orange, p_{model} est une estimation de la véritable distribution génératrice de données, p_{data} (la zone grise).

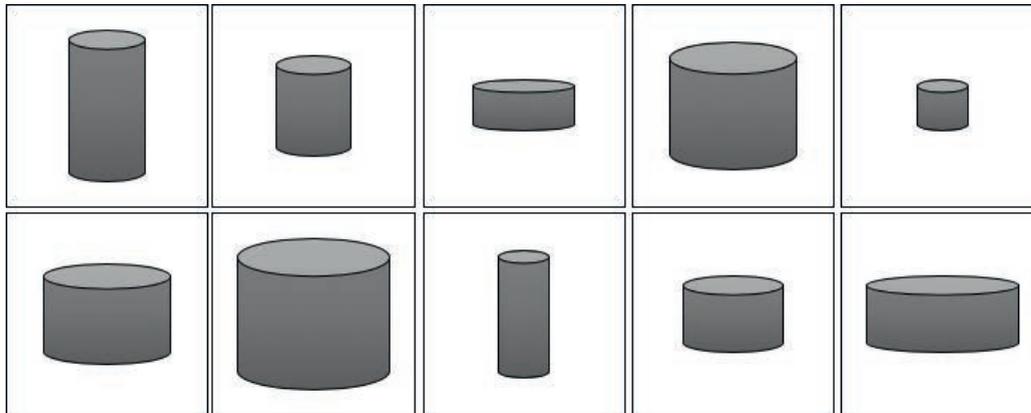


Figure 1.7 : Jeu de données composé de boîtes de biscuits.

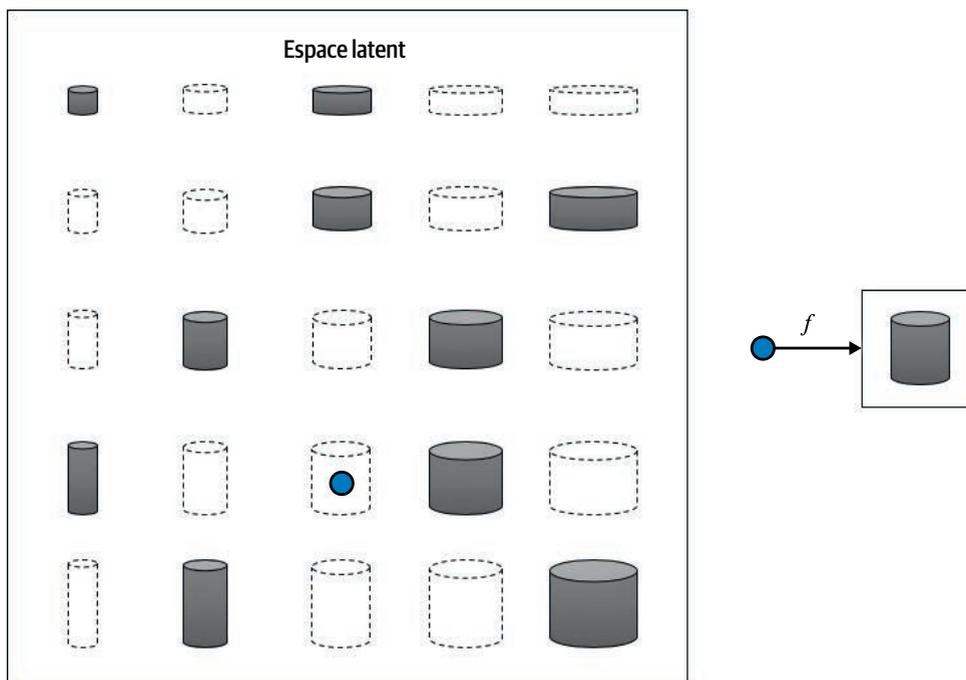


Figure 1.8 : L'espace latent en 2D des boîtes de biscuits et la fonction f qui met en correspondance un point de l'espace latent avec le domaine de l'image originale.

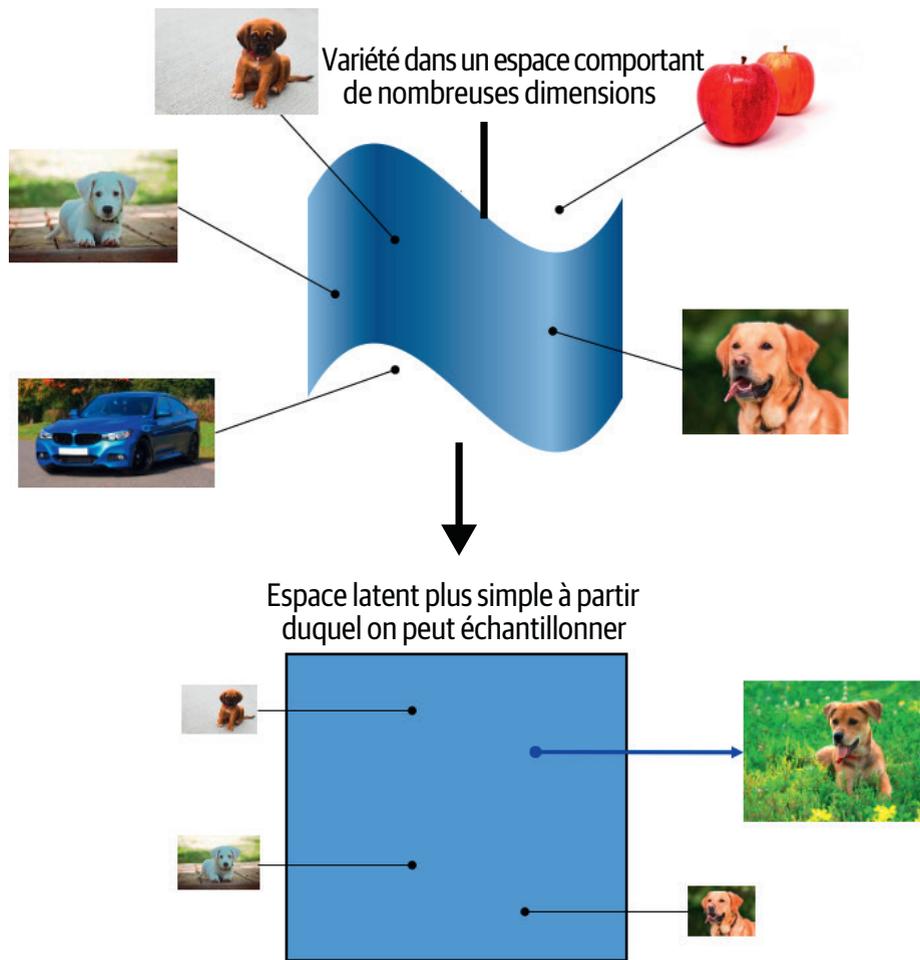


Figure 1.9 : La variété des chiens dans l'espace des pixels comprenant de nombreuses dimensions est cartographiée dans un espace latent plus simple à partir duquel on peut échantillonner.

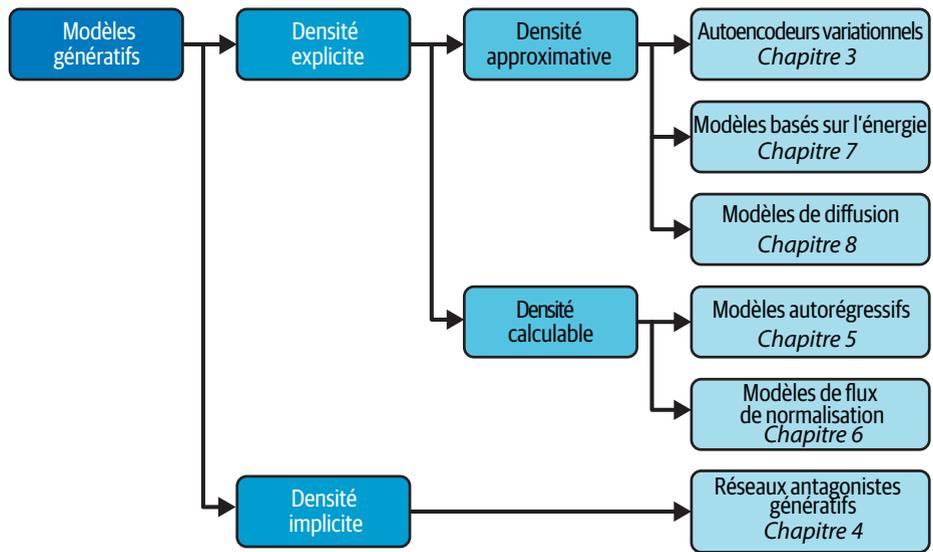


Figure 1.10 : Classification des approches de modélisation générative.

Chapitre 2. Deep learning

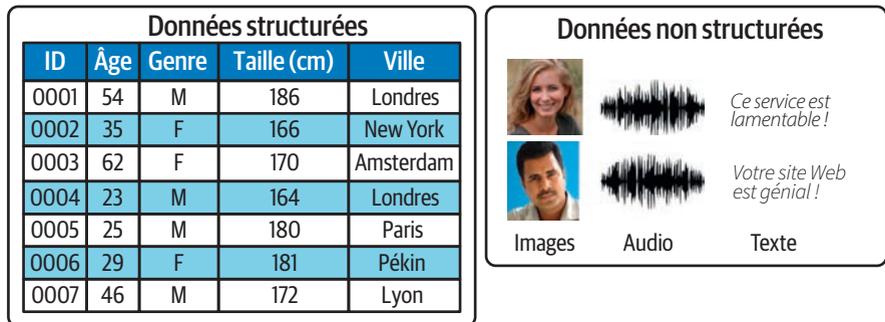


Figure 2.1 : Différence entre données structurées et non structurées.

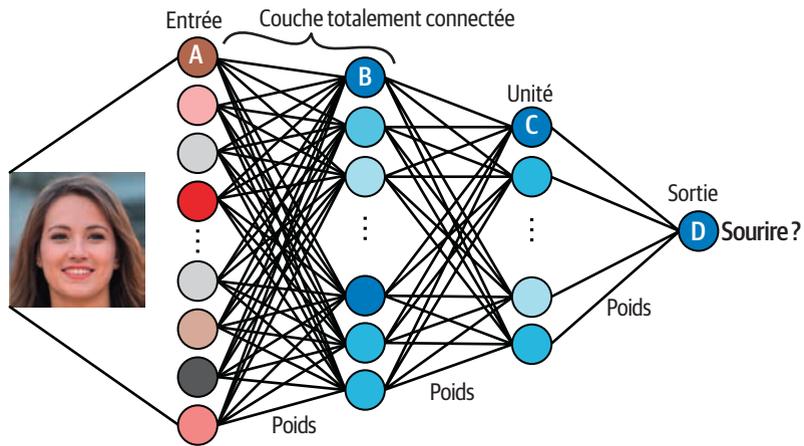


Figure 2.2 : Exemple de perceptron multicouche permettant de prédire si un visage est souriant.



Figure 2.3 : TensorFlow et Keras sont d'excellents outils pour créer des solutions de deep learning.

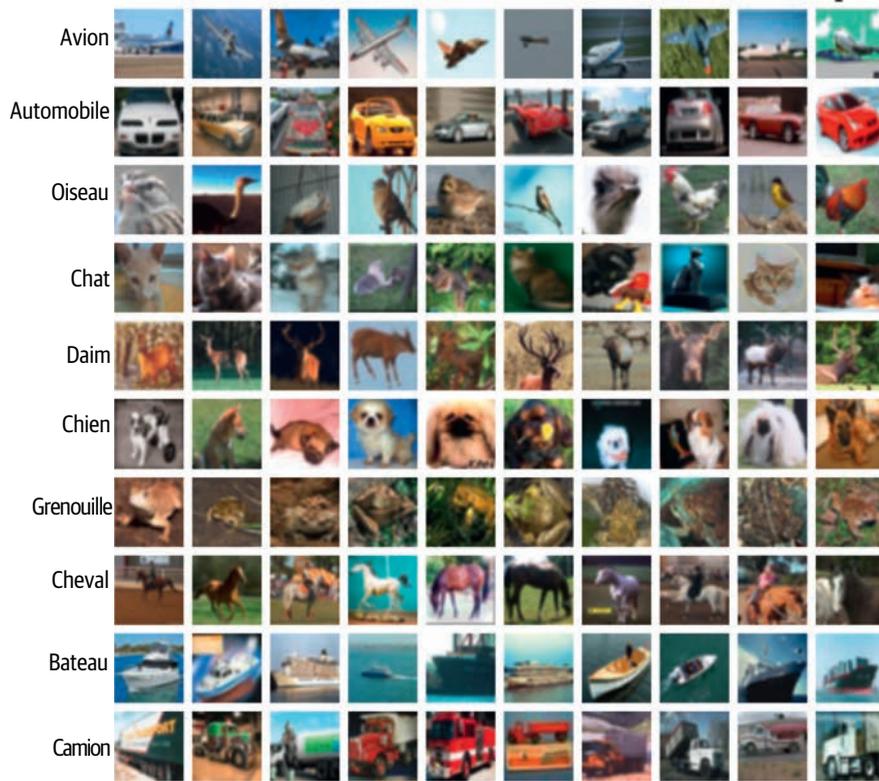


Figure 2.4 : Exemples d'images extraites du jeu de données CIFAR-10 (source : Krizhevsky, 2009).

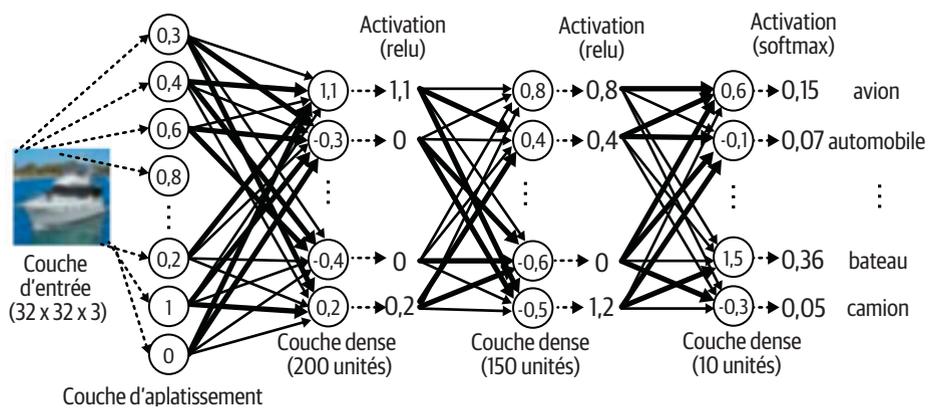


Figure 2.5 : Schéma de l'architecture du MLP.

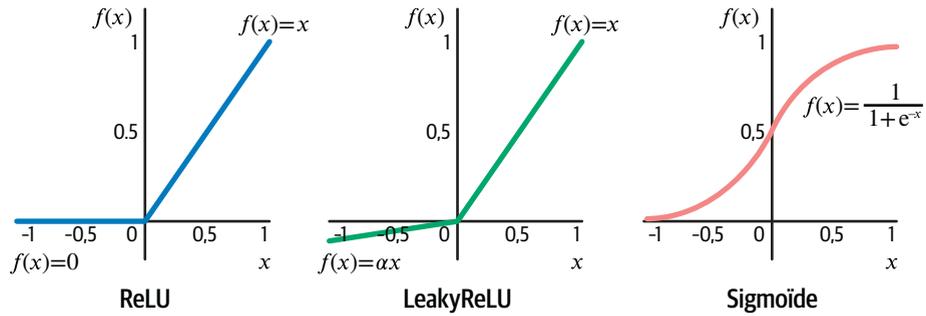


Figure 2.6 : Fonctions d'activation ReLU, LeakyReLU et sigmoïde.

```

model.fit(x_train, y_train, batch_size=32, epochs=10, shuffle=True)
Epoch 1/10
1563/1563 [=====] - 3s 2ms/step - loss: 1.8377 - accuracy: 0.3369
Epoch 2/10
1563/1563 [=====] - 3s 2ms/step - loss: 1.6552 - accuracy: 0.4076
Epoch 3/10
1563/1563 [=====] - 3s 2ms/step - loss: 1.5743 - accuracy: 0.4396
Epoch 4/10
1563/1563 [=====] - 3s 2ms/step - loss: 1.5288 - accuracy: 0.4549
Epoch 5/10
1563/1563 [=====] - 3s 2ms/step - loss: 1.4888 - accuracy: 0.4706
Epoch 6/10
1563/1563 [=====] - 2s 2ms/step - loss: 1.4542 - accuracy: 0.4851
Epoch 7/10
1563/1563 [=====] - 3s 2ms/step - loss: 1.4332 - accuracy: 0.4908
Epoch 8/10
1563/1563 [=====] - 2s 2ms/step - loss: 1.4094 - accuracy: 0.4992
Epoch 9/10
1563/1563 [=====] - 2s 2ms/step - loss: 1.3896 - accuracy: 0.5045
Epoch 10/10
1563/1563 [=====] - 3s 2ms/step - loss: 1.3696 - accuracy: 0.5167
  
```

Figure 2.7 : Sortie de la méthode fit .

```

10000/10000 [=====] - 1s 55us/step
[1.4358007415771485, 0.4896]
  
```

Figure 2.8 : Sortie de la méthode evaluate.



Figure 2.9 : Quelques prédictions faites par le modèle, avec les étiquettes des catégories réelles.

Partie 3 × 3 d'une image Filtre

$$\begin{array}{|c|c|c|} \hline 0,6 & 0,4 & 0,6 \\ \hline 0,1 & -0,2 & -0,3 \\ \hline -0,5 & -0,4 & -0,3 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array} = 2,8$$

$$\begin{array}{|c|c|c|} \hline -0,7 & 0,6 & 0,2 \\ \hline 0,1 & 0,5 & -0,3 \\ \hline -0,3 & -0,4 & 0,5 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array} = -0,1$$

Figure 2.10 : Filtre convolutif 3 × 3 appliqué à deux parties d'une image en niveaux de gris.

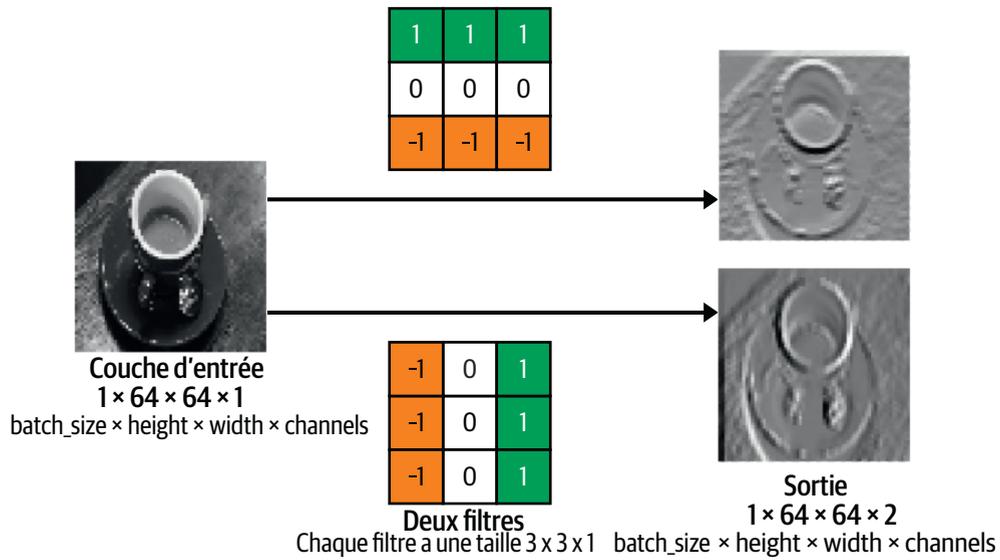


Figure 2.11 : Deux filtres convolutifs appliqués à une image en niveaux de gris.

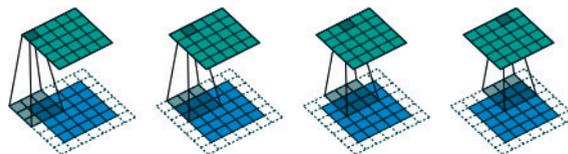


Figure 2.12 : Noyau $3 \times 3 \times 1$ (gris) passé sur une image d'entrée $5 \times 5 \times 1$ (bleu), avec padding = «same» et strides = 1, pour générer la sortie $5 \times 5 \times 1$ (vert) (source : Dumoulin et Visin, 2018).

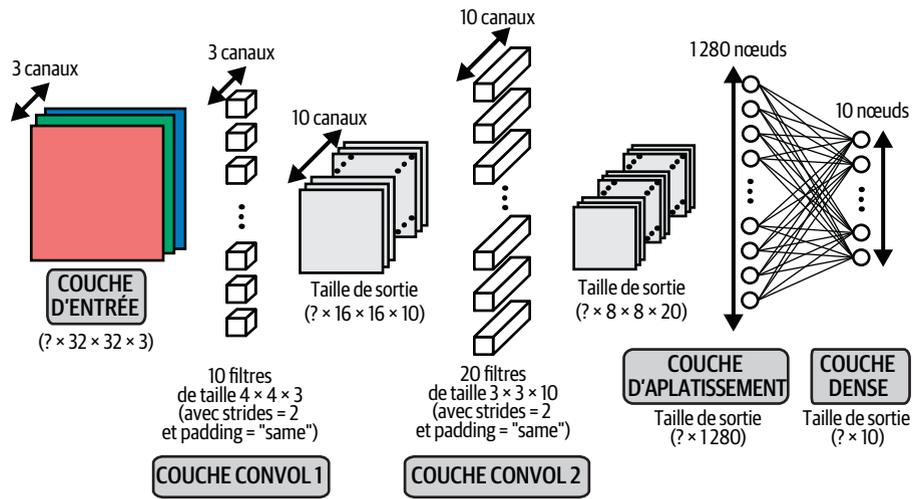
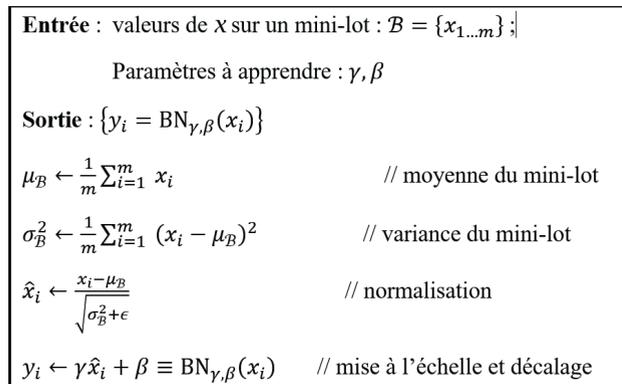


Figure 2.13 : Schéma d'un réseau de neurones convolutif.



Algorithme 1 : transformation de normalisation par lots, appliquée à une fonction d'activation χ sur un mini-lot.

Figure 2.14 : Processus de normalisation par lots (source : Ioffe et Szegedy, 2015).

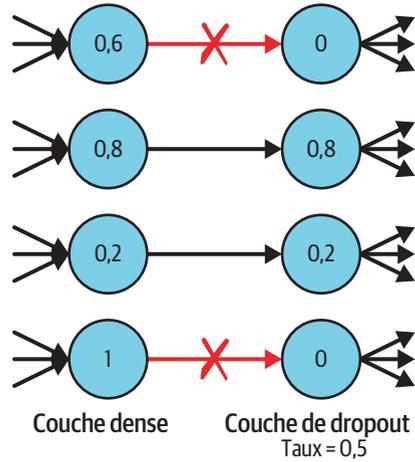


Figure 2.15 : Couche de dropout.

```
model.evaluate(x_test, y_test, batch_size=1000)
10000/10000 [=====] - 15s 1ms/step
[0.8423407137393951, 0.7155999958515167]
```

Figure 2.16 : Performance du CNN.



Figure 2.17 : Prévisions d'un CNN.

Chapitre 3. Autoencodeurs variationnels



Figure 3.1 : Un homme debout devant une penderie infinie en 2D (image créée avec Midjourney).

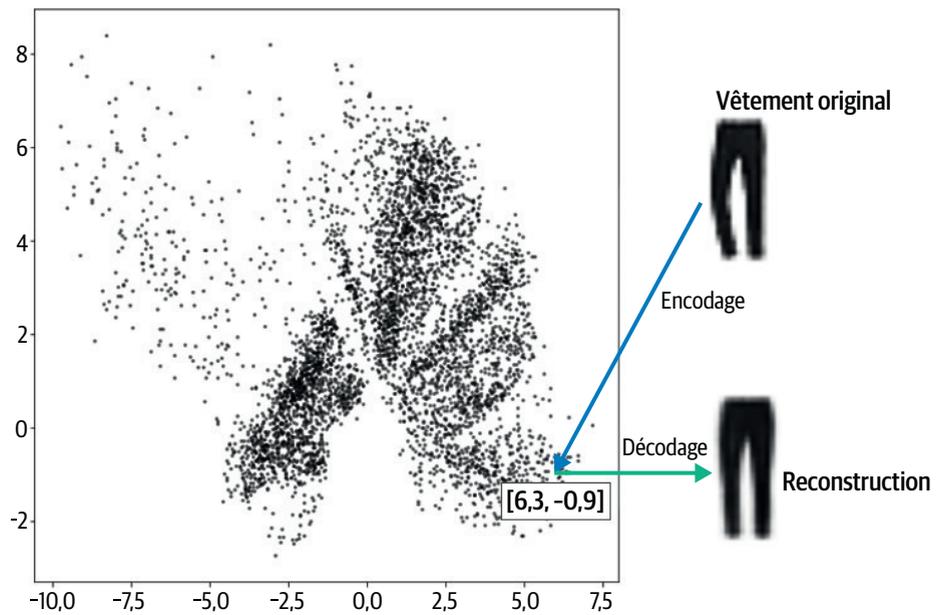


Figure 3.2 : Vêtements de la penderie infinie : chaque point noir représente un vêtement.



Figure 3.3 : Exemples d'images extraites du jeu de données Fashion-MNIST.

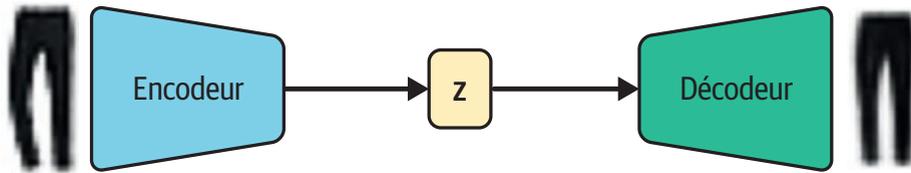


Figure 3.4 : Schéma de l'architecture de l'autoencodeur.

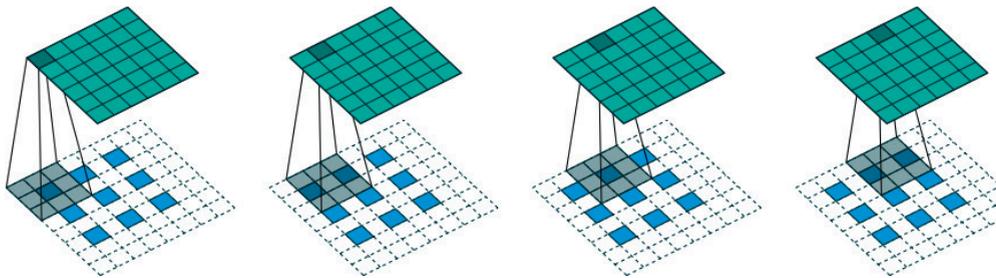


Figure 3.5 : Exemple de couche transposée convolutive (voir l'image en couleur en ligne ; source : Dumoulin et Visin, 2018)¹.

¹ Vincent Dumoulin et Francesco Visin, « A Guide to Convolution Arithmetic for Deep Learning », 12 janvier 2018, <https://arxiv.org/abs/1603.07285>.

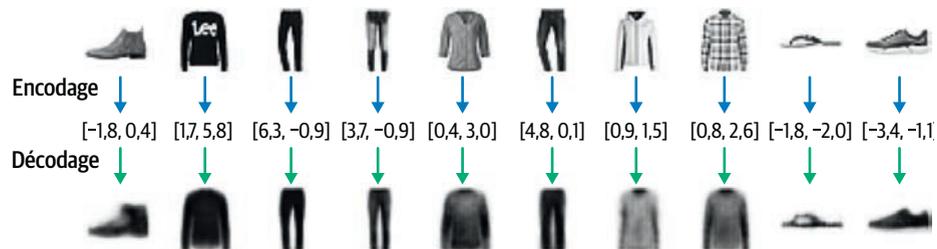


Figure 3.6 : Exemples d'encodage et de décodage de vêtements.

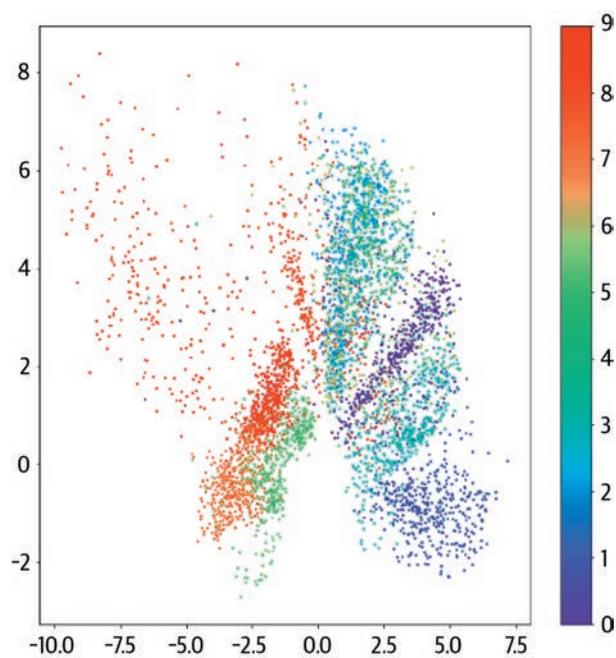


Figure 3.7 : Tracé de l'espace latent, coloré selon la catégorie de vêtement.

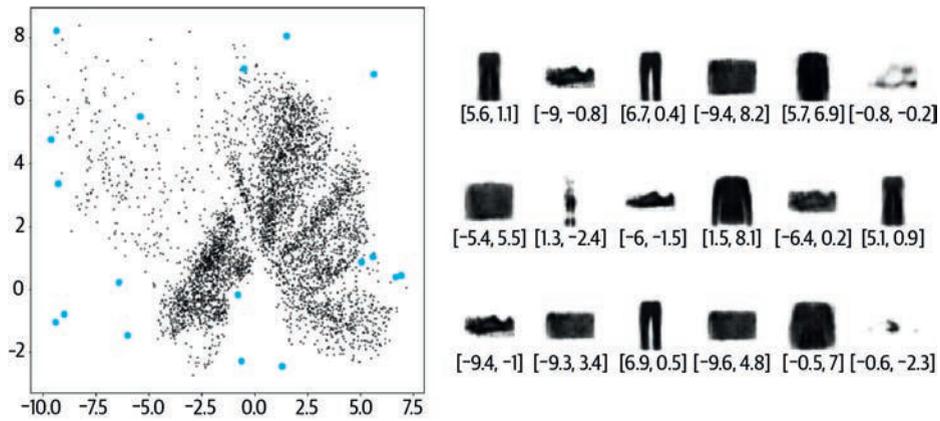


Figure 3.8 : Images générées de vêtements.

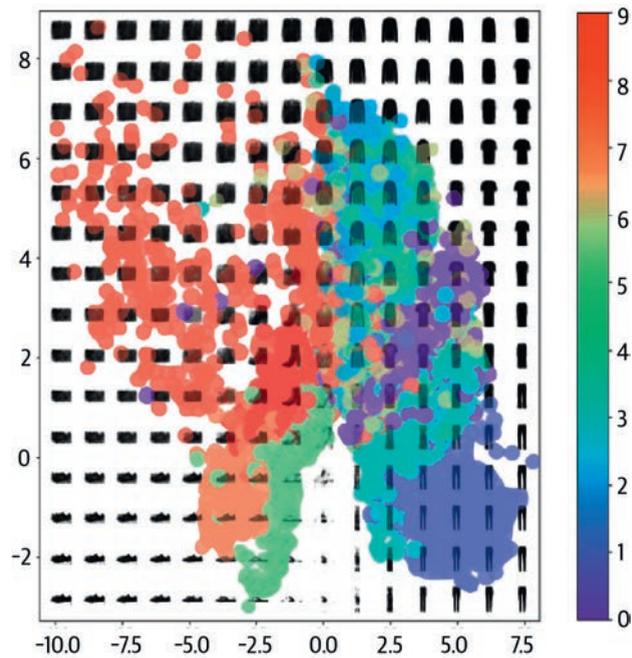


Figure 3.9 : Grille d'embeddings décodés, superposés aux embeddings des images originales du jeu de données, colorés par type d'élément.

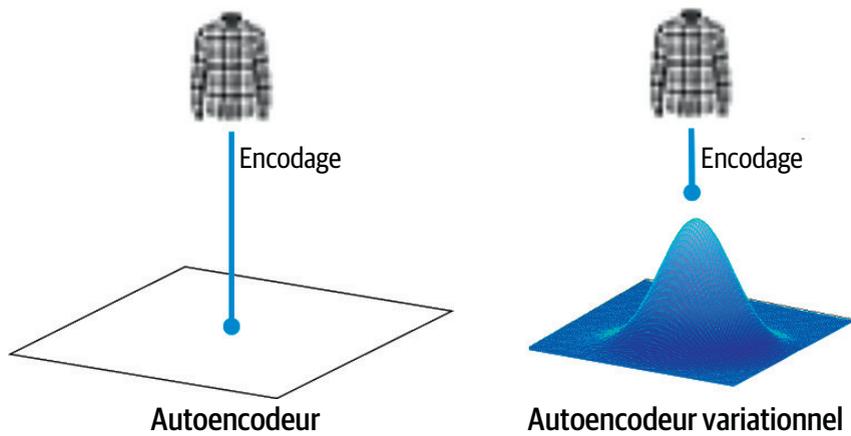


Figure 3.10 : Différence entre les encodeurs d'un autoencodeur et d'un autoencodeur variationnel.

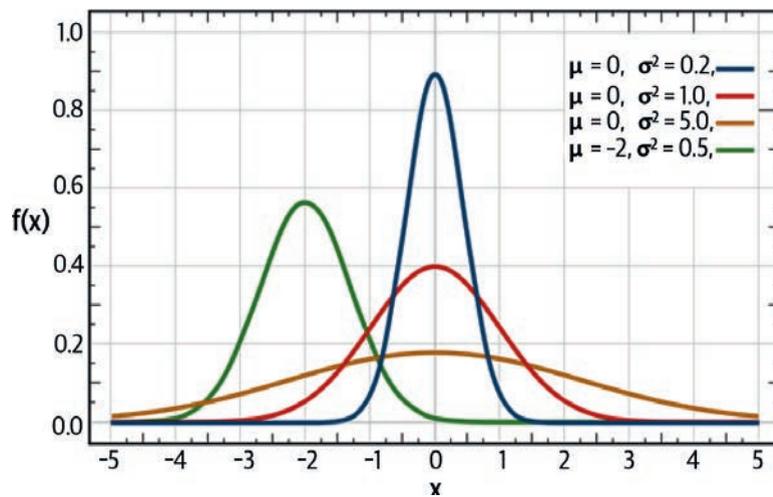


Figure 3.11 : Distribution normale unidimensionnelle (voir l'image en couleur en ligne ; source : Wikipédia ; https://fr.wikipedia.org/wiki/Loi_normale).

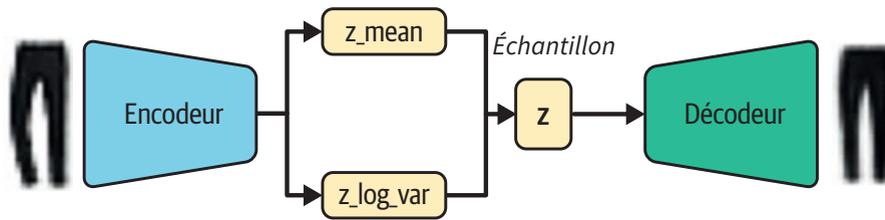


Figure 3.12 : Architecture d'un autoencodeur variationnel.

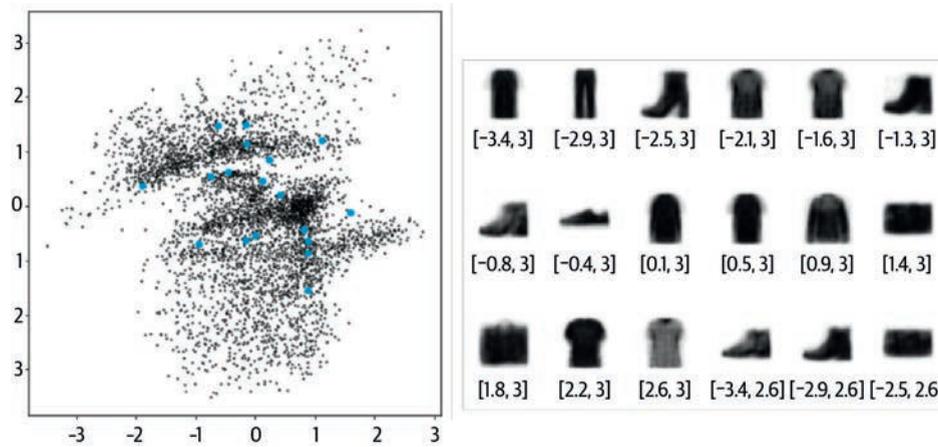


Figure 3.13 : Le nouvel espace latent : les points noirs montrent la valeur z_{mean} de chaque image codée, tandis que les points bleus montrent quelques points échantillonnés dans l'espace latent (avec leurs images décodées à droite). L'image en couleur est disponible en ligne.

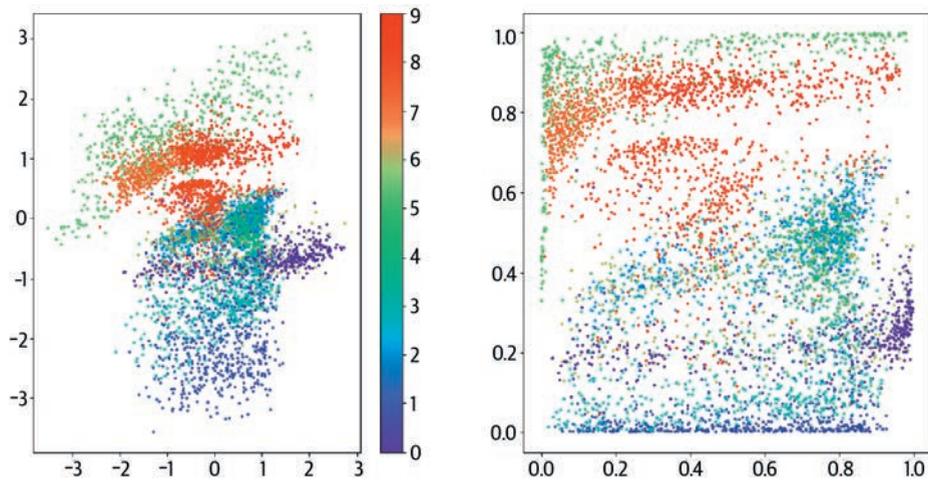


Figure 3.14 : Espace latent du VAE coloré par type de vêtement.



Figure 3.15 : Quelques exemples tirés du jeu de données CelebA (source : Liu et al., 2015).

Exemples de visages réels



Reconstructions



Figure 3.16 : Visages reconstruits, après passage par l'encodeur et le décodeur.

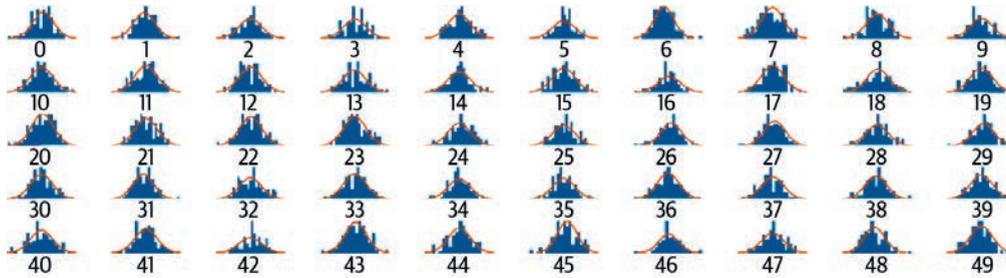


Figure 3.17 : Distributions des points pour les 50 premières dimensions de l'espace latent.

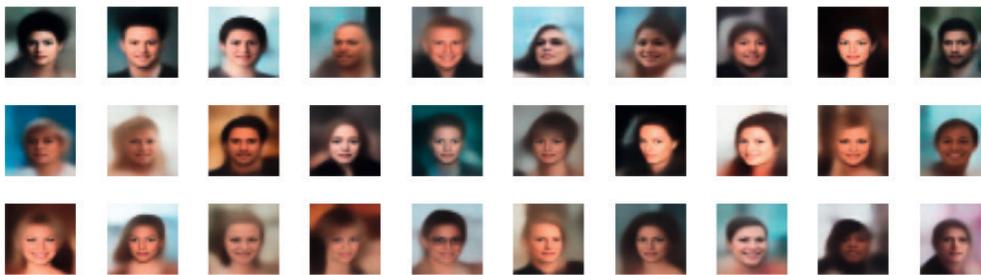


Figure 3.18 : Nouveaux visages générés.

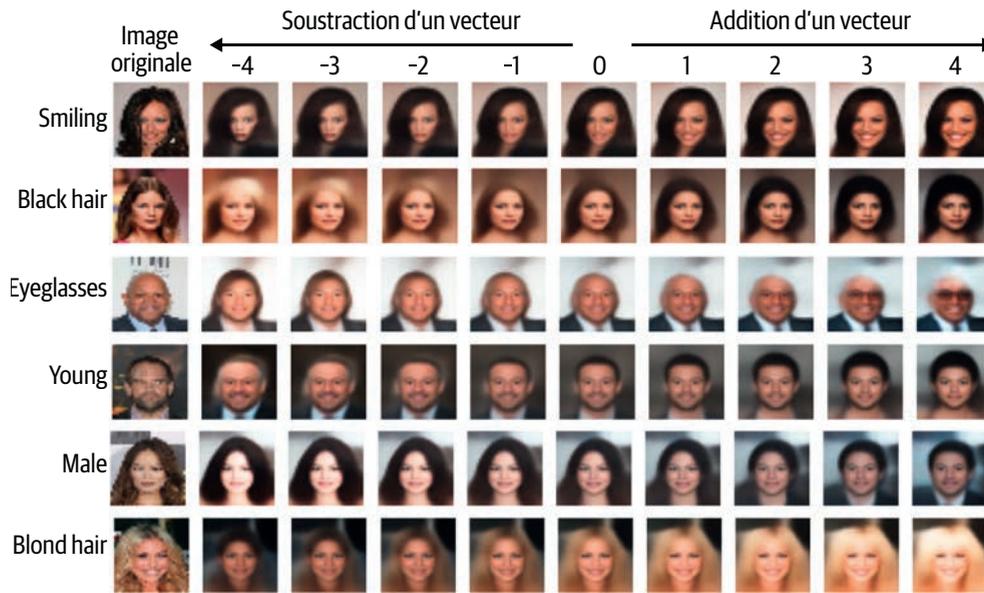


Figure 3.19 : Ajout et soustraction de caractéristiques à des visages et à partir de ceux-ci.

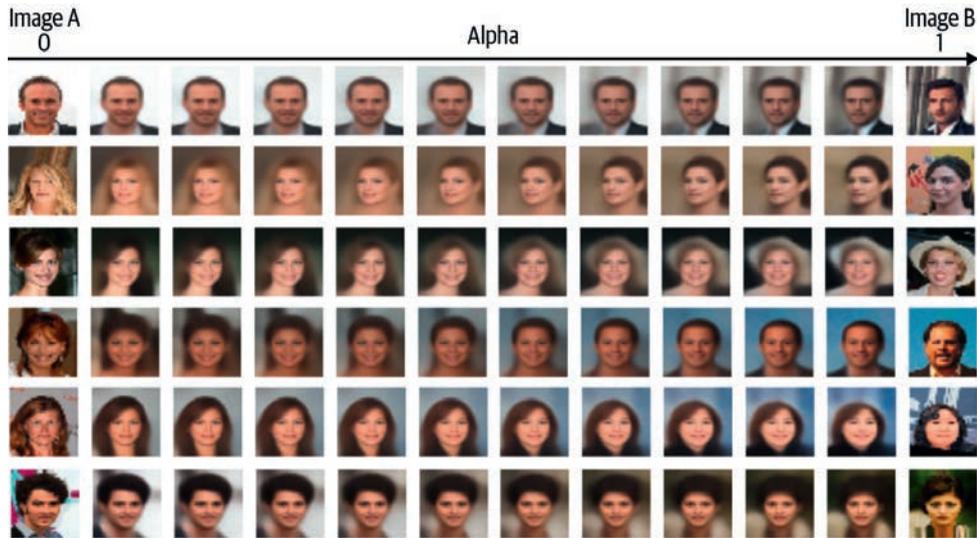


Figure 3.20 : Morphing entre deux visages.

Chapitre 4. Réseaux antagonistes génératifs



Figure 4.1 : Chaîne de production d'une entreprise fabriquant des briques de différentes formes et tailles (image créée avec Midjourney).

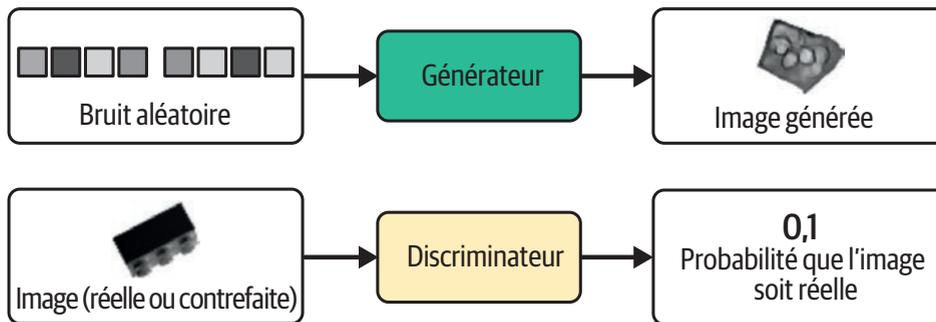


Figure 4.2 : Entrées et sorties des deux réseaux dans un GAN.



Figure 4.3 : Exemples d'images provenant du jeu de données sur les briques Lego.

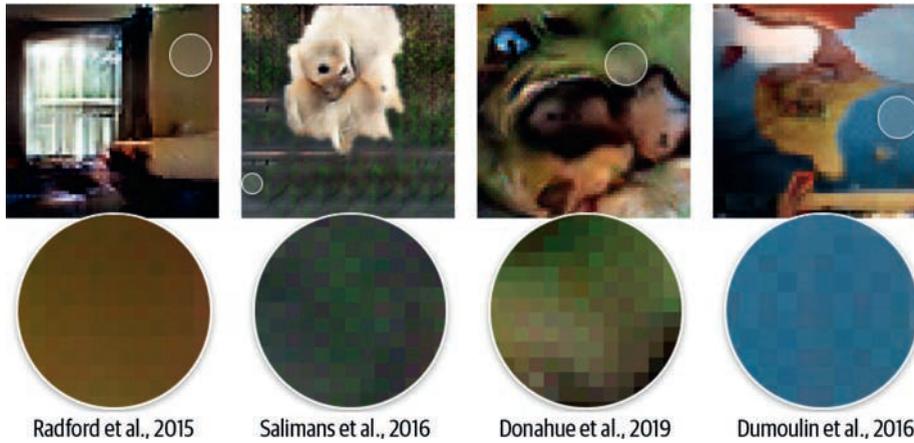


Figure 4.4 : Artefacts lors de l'utilisation de couches transposées convolutives (voir l'image en couleur en ligne ; source : Odena et al., 2016).

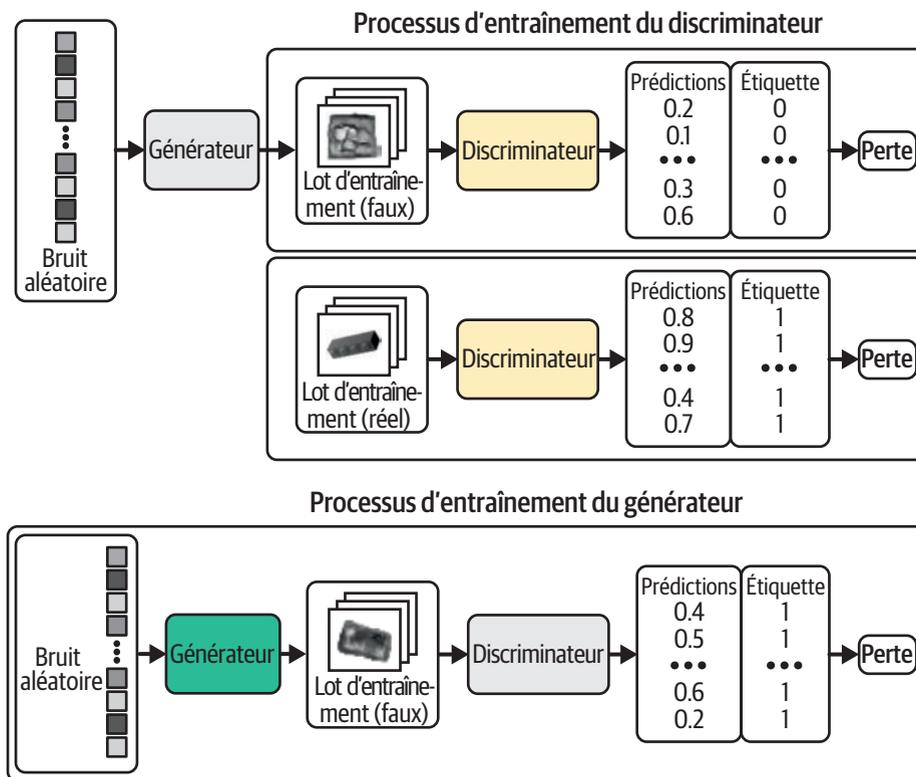


Figure 4.5 : Entraînement du DCGAN ; les cases grises indiquent que les poids sont gelés pendant l'entraînement.

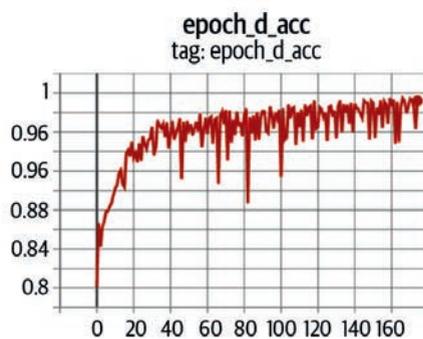
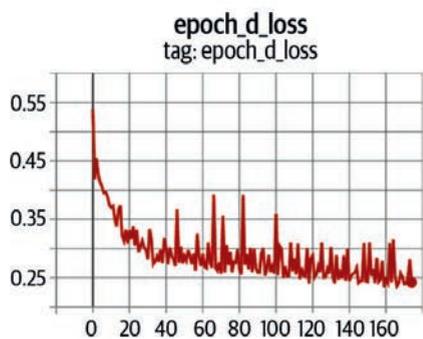
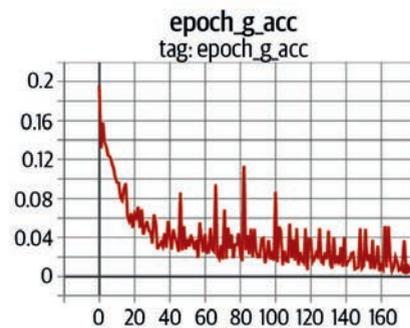
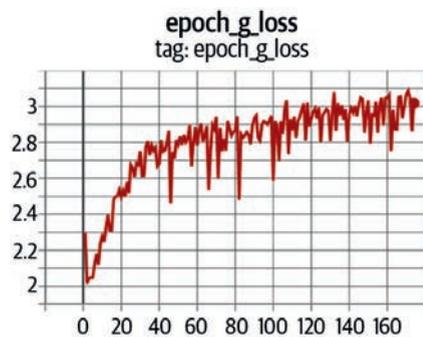


Figure 4.6 : Perte et précision du générateur (en haut) et du discriminateur (en bas) au cours de l'entraînement

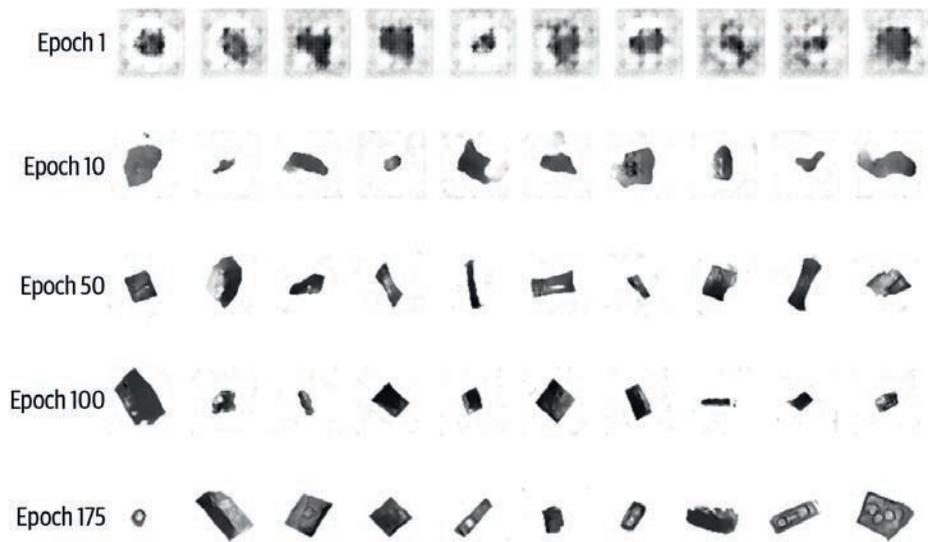


Figure 4.7 : Sortie du générateur à des epochs spécifiques au cours de l'entraînement.

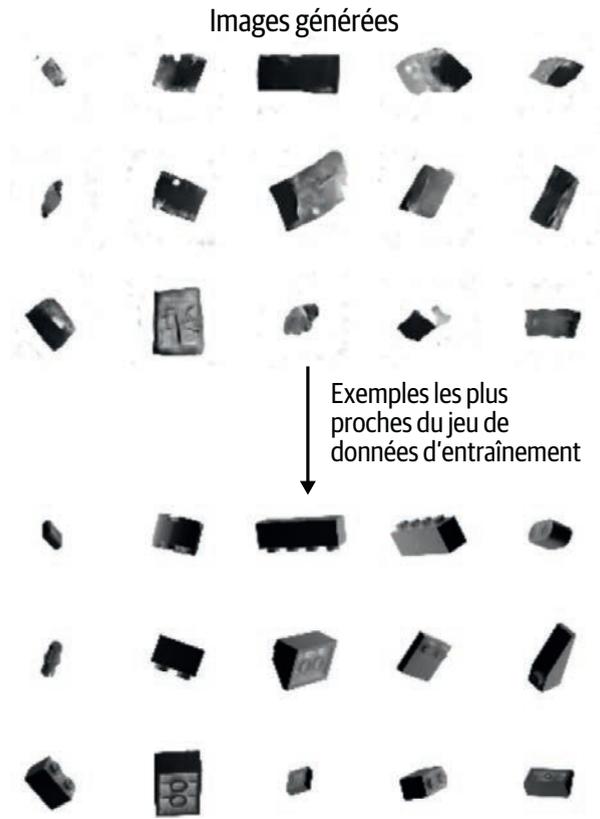


Figure 4.8 : Correspondances les plus étroites entre les images générées à partir du jeu de données d'entraînement.

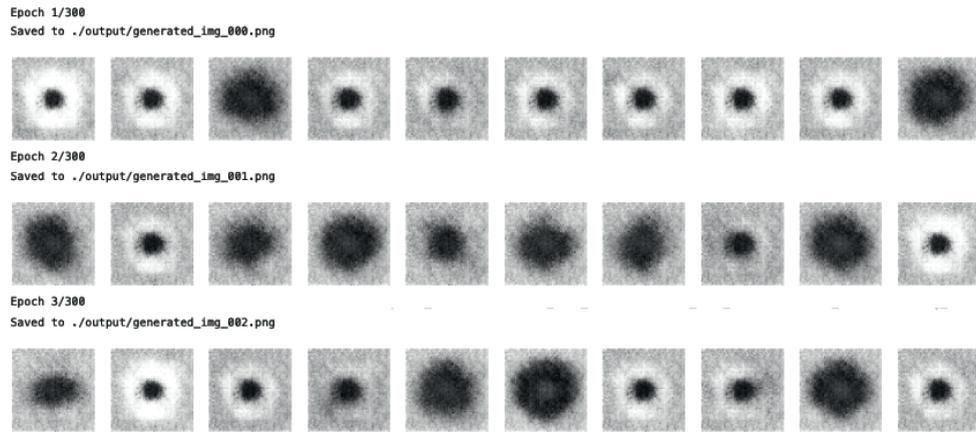


Figure 4.9 : Exemple de sortie lorsque le discriminateur surpasse le générateur.

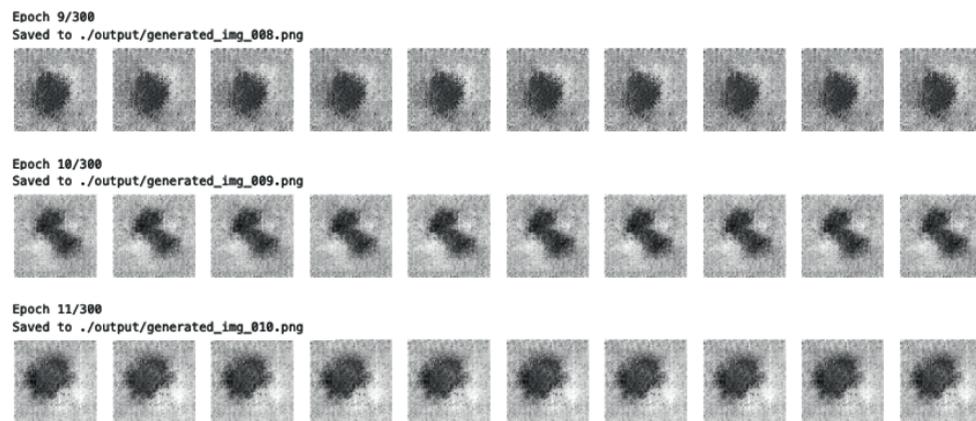


Figure 4.10 : Exemple d'effondrement de mode lorsque le générateur surpasse le discriminateur.

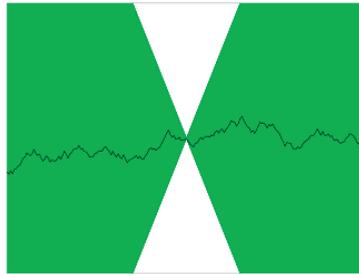


Figure 4.11 : Fonction continue Lipschitz (source : Wikipédia).

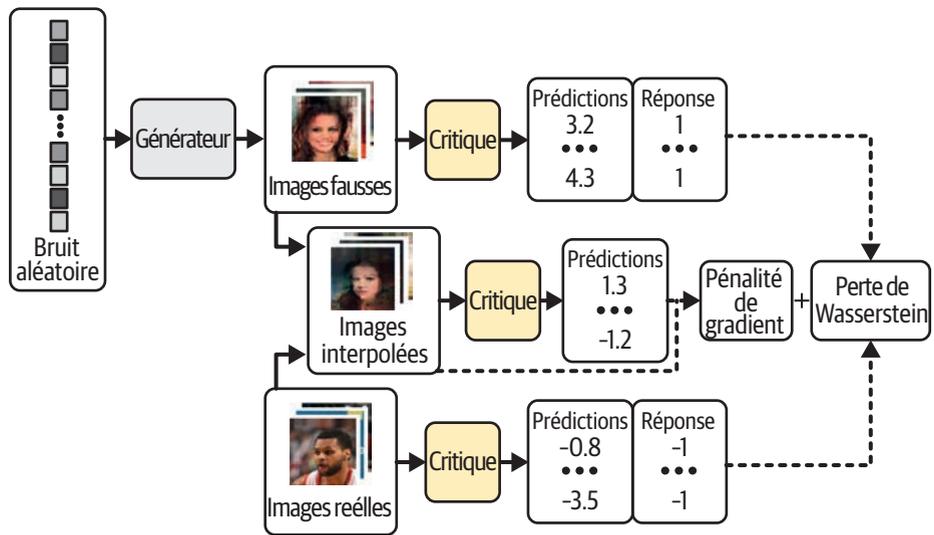


Figure 4.12 : Processus d'entraînement des critiques du WGAN-GP.

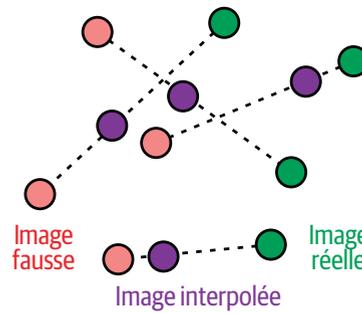


Figure 4.13 : Interpolation entre les images.



Figure 4.14 : Exemples de visages générés par un WGAN-GP.

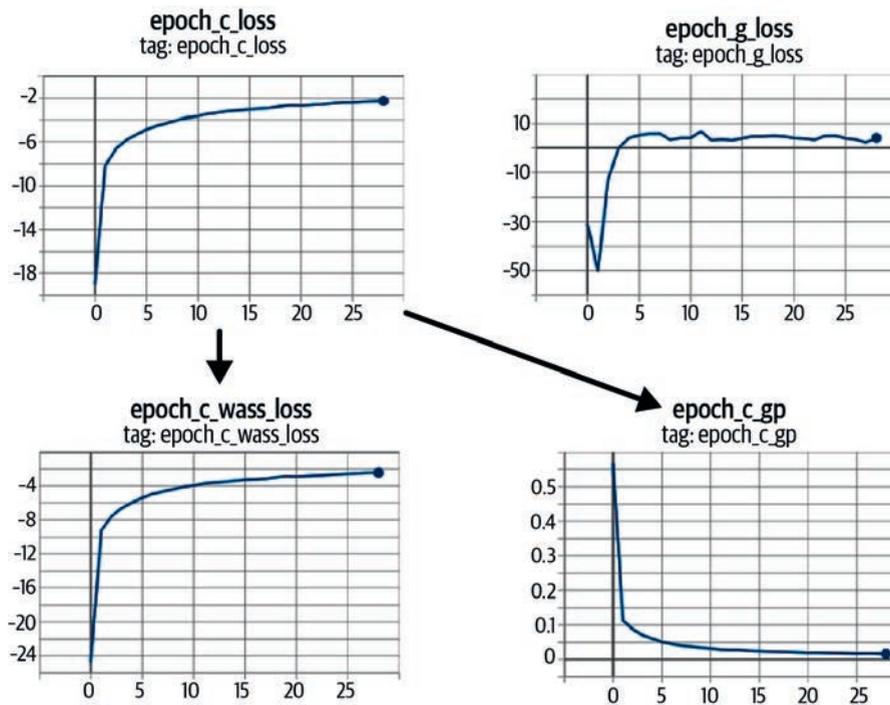


Figure 4.15 : Courbes de perte WGAN-GP : la perte du critique ($epoch_c_loss$) est décomposée en perte de Wasserstein ($epoch_c_wass$) et en perte de pénalité de gradient ($epoch_c_gp$).

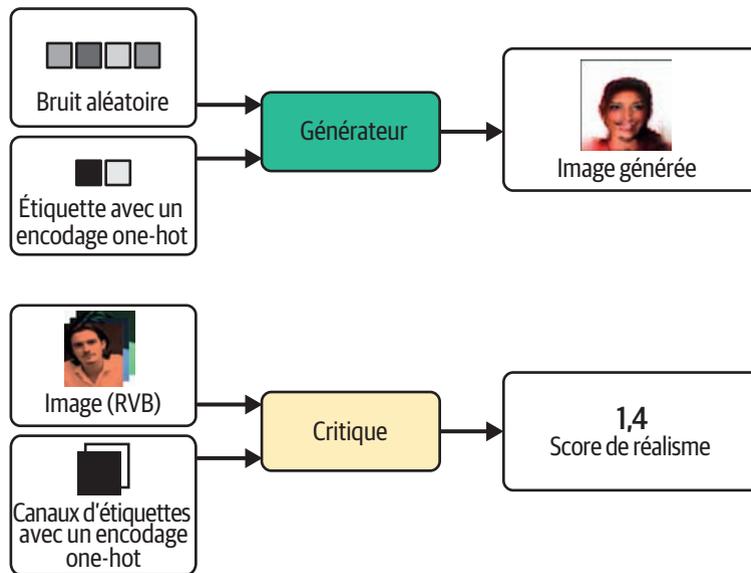


Figure 4.16 : Entrées et sorties du générateur et du critique dans un CGAN.

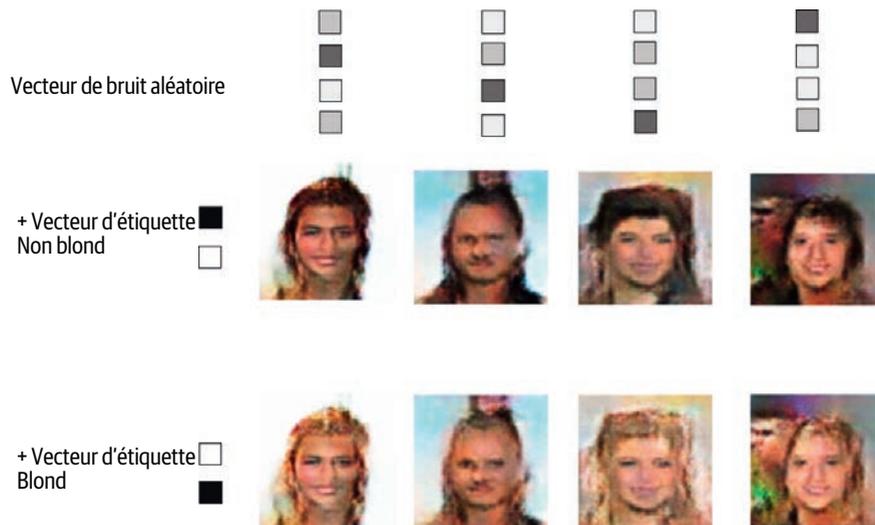


Figure 4.17 : Sortie du CGAN lorsque les vecteurs Blond et Non blond sont ajoutés à l'échantillon latent.

Chapitre 5. Modèles autorégressifs



Figure 5.1 : Grande cellule de prisonniers lisant des livres (image créée avec Midjourney).

Token	Embedding				
0	-0,13	0,45	...	0,13	-0,04
1	0,22	0,56	...	0,24	-0,63
...
9998	0,16	-0,70	...	-0,35	1,02
9999	-0,98	-0,45	...	-0,15	-0,52

Taille du vocabulaire (10 000)

Taille de l'embedding (100)

Figure 5.2 : Une couche Embedding est une table de recherche pour chaque token sous la forme d'un entier.

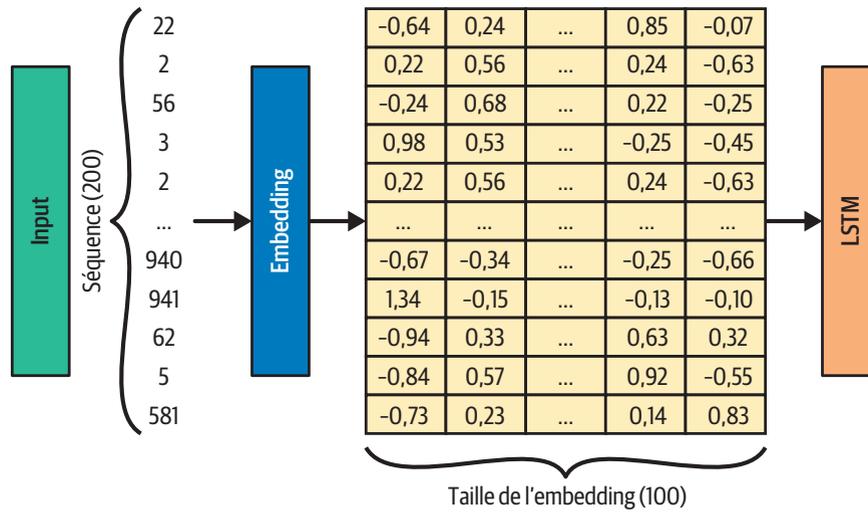


Figure 5.3 : Séquence unique traversant une couche Embedding.

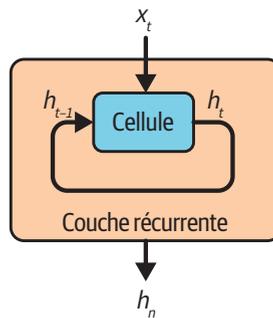


Figure 5.4 : Schéma simplifié d'une couche récurrente.

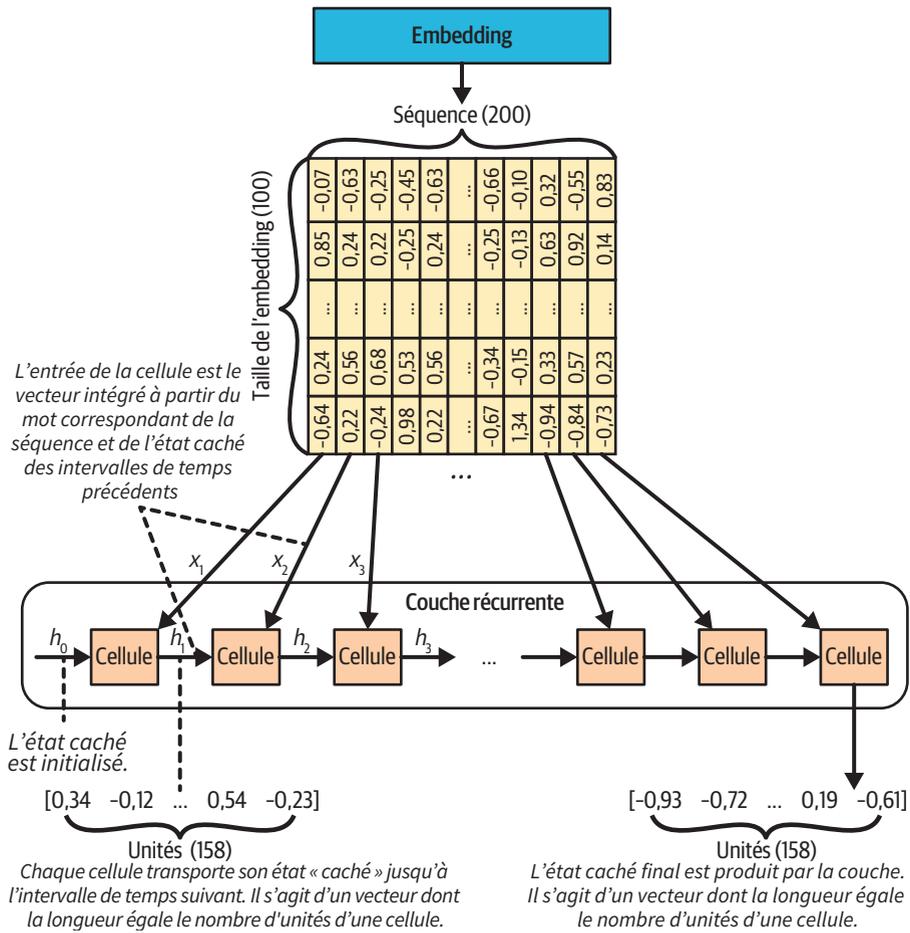
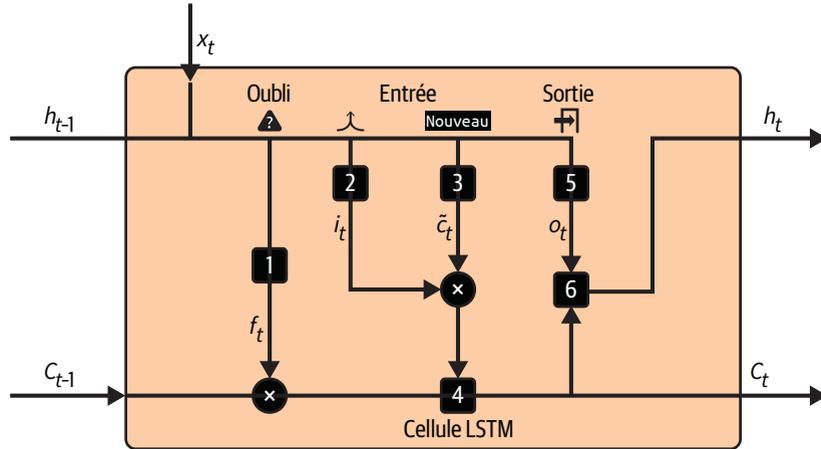


Figure 5.5 : Comment une séquence unique traverse une couche récurrente.



- 1 $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
- 2 $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
- 3 $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
- 4 $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
- 5 $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
- 6 $h_t = o_t * \tanh(C_t)$

Figure 5.6 : Cellule LSTM.

```

Epoch 1/25
628/629 [=====→.] - ETA: 0s - loss: 4.4536
generated text:
recipe for mold salad are high 8 pickled to fold cook the dish into and warm in baking reduced but halves beans
and cut

629/629 [=====] - 29s 43ms/step - loss: 4.4527
Epoch 2/25
628/629 [=====→.] - ETA: 0s - loss: 3.2339
generated text:
recipe for racks - up-don with herb fizz | serve checking thighs onto sanding butter and baking surface in a hea
vy heavy large saucepan over blender ; stand overnight . [UNK] over moderate heat until very blended , garlic ,
about 8 minutes . cook airtight until cooked are soft seeds , about 1 45 minutes . sugar , until s is brown , 5
to sliced , parmesan , until browned and add extract . wooden crumb to outside of out sheets . flatten and prehe
ated return to the paste . add in pecans oval and let transfer day .

629/629 [=====] - 30s 48ms/step - loss: 3.2336
Epoch 3/25
629/629 [=====] - ETA: 0s - loss: 2.6229
generated text:
recipe for grilled chicken | preheat oven to 400*f . cook in large 8 - caramel grinder or until desired are firm
, about 6 minutes

629/629 [=====] - 27s 42ms/step - loss: 2.6229
Epoch 4/25
629/629 [=====] - ETA: 0s - loss: 2.3426
generated text:
recipe for pizza salad with sweet red pepper and star fruit | combine all ingredients except lowest ingredients
in a large skillet . working with batches and deglaze , cook until just cooked through , about 1 minute . meanwh
ile , boil potatoes and paprika in a little oil over medium - high heat , stirring it just until crisp , about 3
minutes . stir in bell pepper , onion and cooked paste and jalapeño until clams well after most - reggiano , abo
ut 5 minutes . transfer warm 2 tablespoons flesh of eggplants to medium bowl . serve .

```

Figure 5.7 : Premières epochs du processus d'entraînement du LSTM.

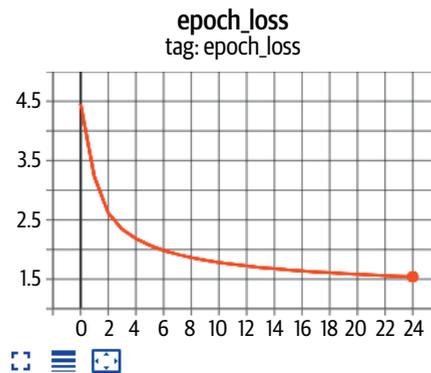


Figure 5.8 : Métrique de perte d'entropie croisée du processus d'entraînement du LSTM par epoch.

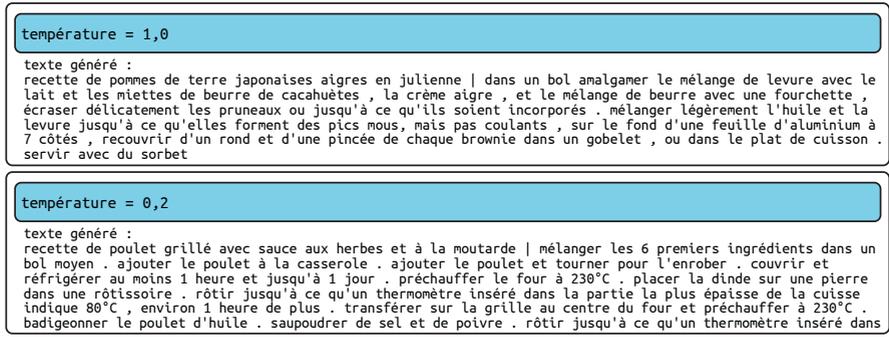


Figure 5.9 : Sorties générées avec le paramètre *temperature* = 1.0 et *temperature* = 0.2.

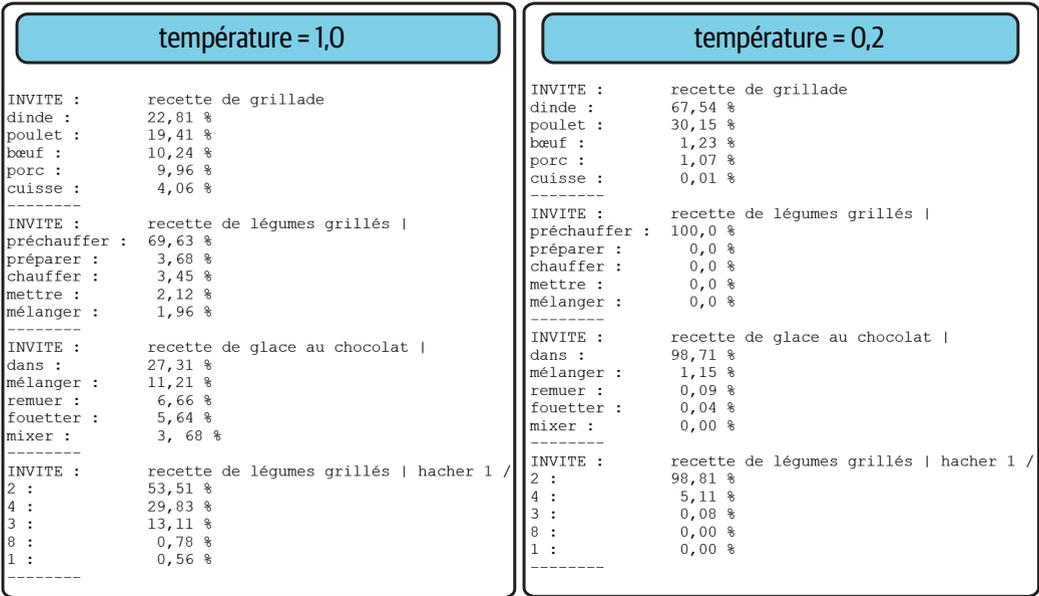


Figure 5.10 : Distribution des probabilités de mots suivant diverses séquences, pour des valeurs de température de 1,0 et 0,2.

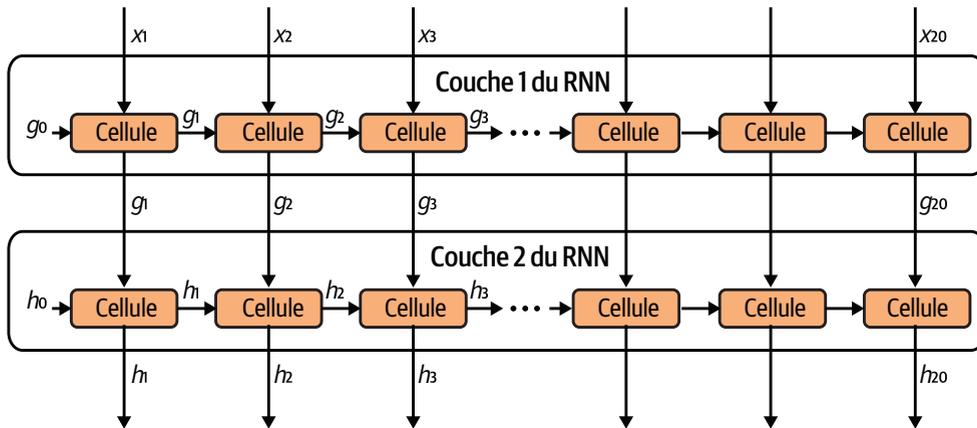


Figure 5.11 : Schéma d'un RNN multicouche : g_t indique les états cachés de la première couche, et h_t les états cachés de la deuxième couche.

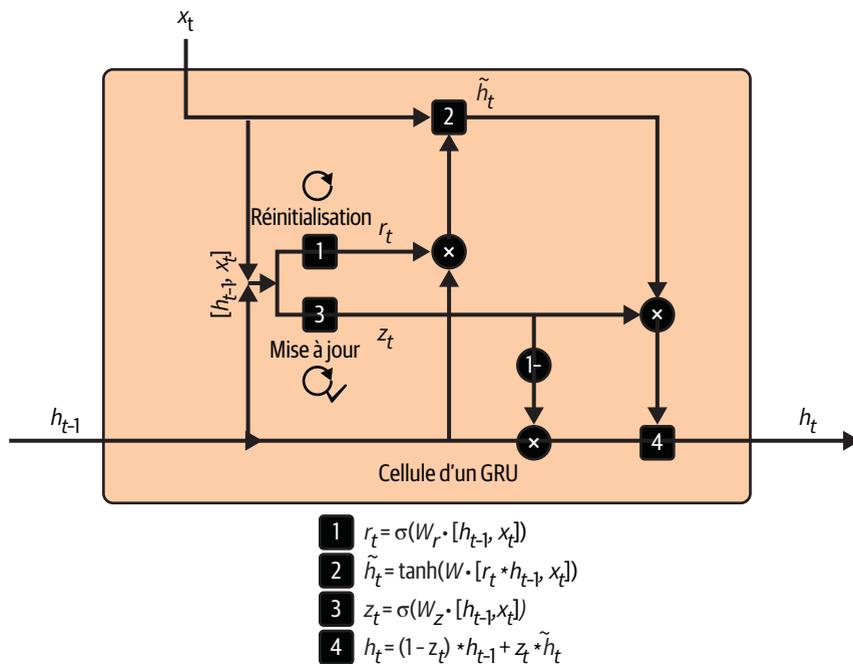


Figure 5.12 : Cellule unique d'un GRU.

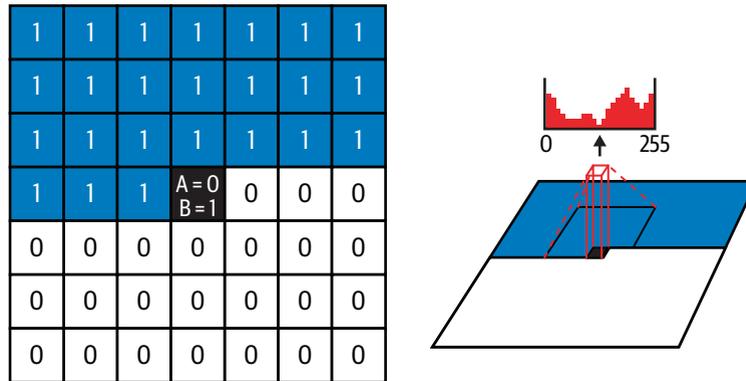


Figure 5.13 : À gauche : masque de filtre convolutif ; à droite : masque appliqué à un ensemble de pixels afin de prédire la distribution de la valeur du pixel central (source : van den Oord et al., 2016).

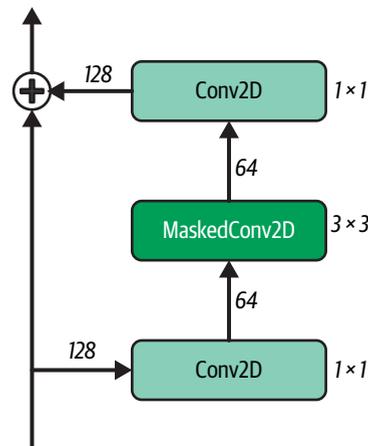


Figure 5.14 : Bloc résiduel PixelCNN (le nombre de filtres est indiqué à côté des flèches et la taille des filtres est indiquée à côté des couches).



Figure 5.15 : Exemples d'images provenant du jeu de données d'entraînement et d'images générées par le modèle PixelCNN.

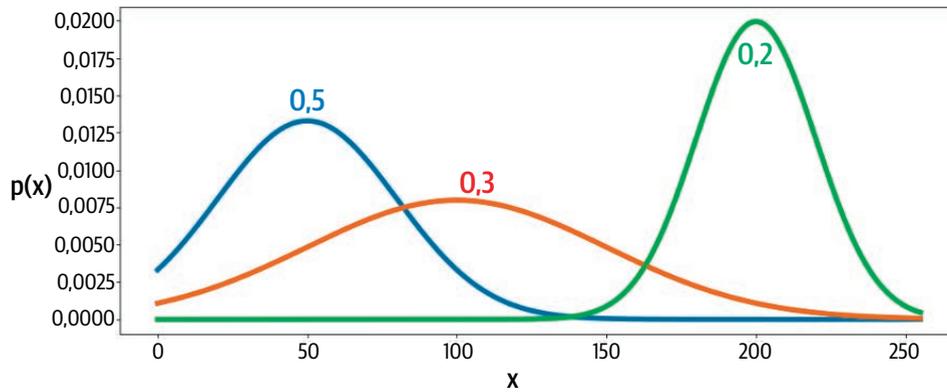


Figure 5.16 : Distribution mixte de trois distributions normales avec des paramètres différents ; la distribution catégorielle sur les trois distributions normales est $[0,5, 0,3, 0,2]$.



Figure 5.17 : Sorties du PixelCNN utilisant une distribution mixte en sortie.

Chapitre 6. Modèles de flux de normalisation



Figure 6.1 : Intérieur d'une boutique du xix^e siècle avec une grosse cloche métallique (image créée avec Midjourney).

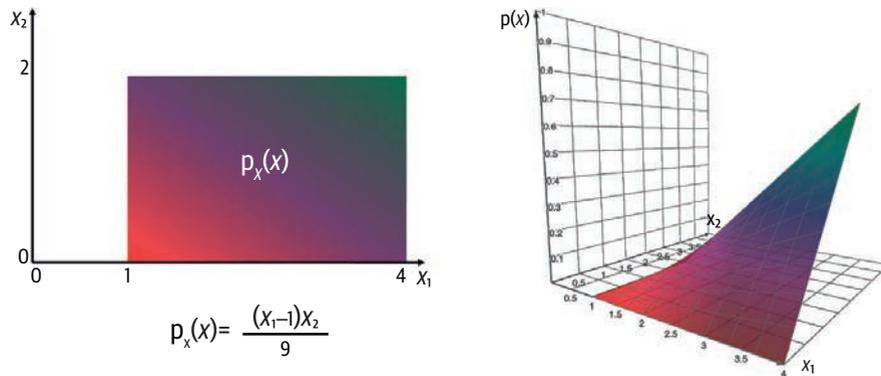


Figure 6.2 : Distribution de probabilité $p_X(x)$ définie sur deux dimensions, représentée en 2D (à gauche) et en 3D (à droite).

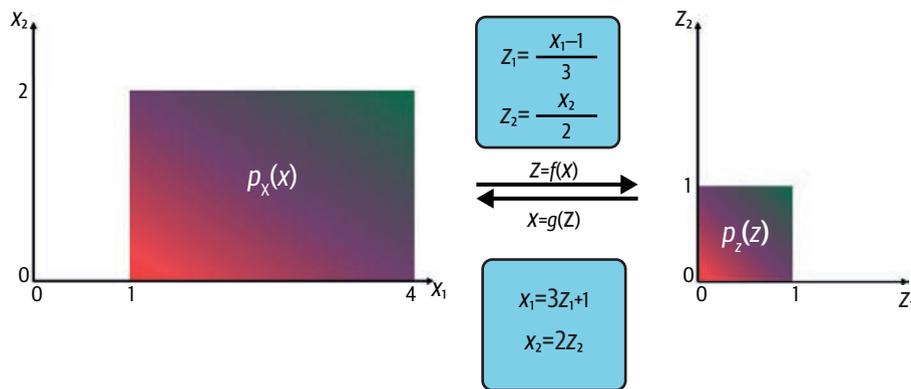


Figure 6.3 : Changement de variables entre X et Z .

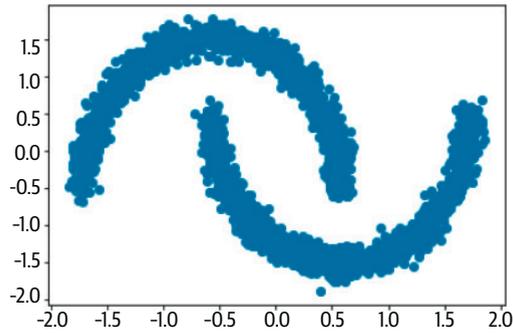


Figure 6.4 : Jeu de données représentant deux lunes en deux dimensions.



Figure 6.5 : Une couche de couplage produit deux tenseurs qui ont la même forme que l'entrée : un facteur d'échelle (s) et un facteur de translation (t).

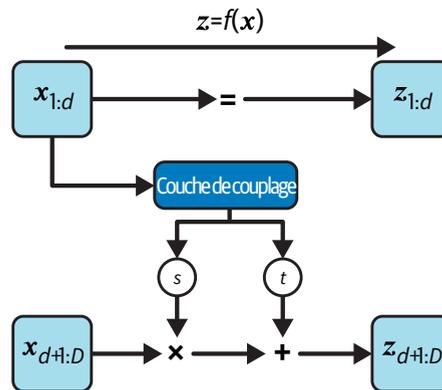


Figure 6.6 : Processus de transformation de l'entrée x à travers une couche de couplage.

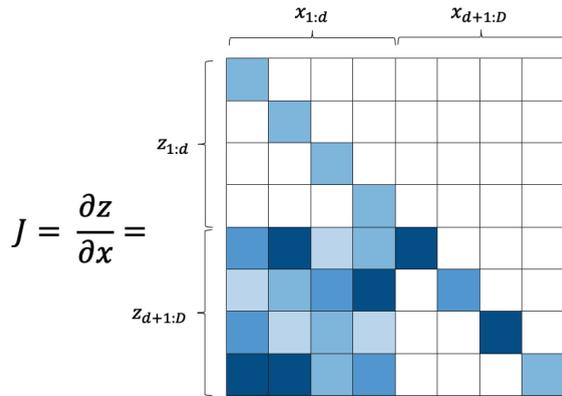


Figure 6.7 : Matrice jacobienne de la transformation : une matrice triangulaire inférieure, dont le déterminant est égal au produit des éléments le long de la diagonale.

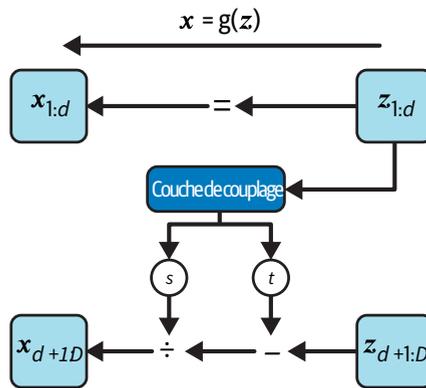


Figure 6.8 : Fonction inverse $x = g(z)$.

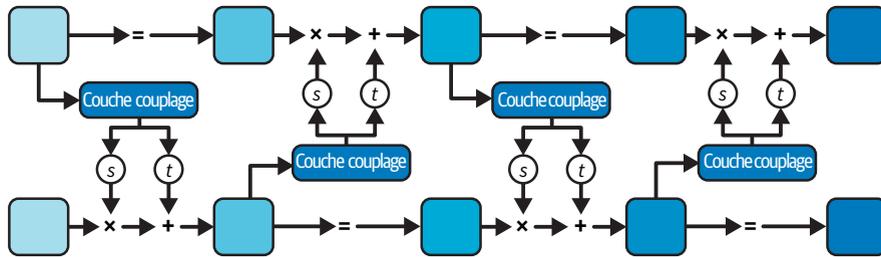


Figure 6.9 : Empilement de couches de couplage, en alternant le masquage avec chaque couche.

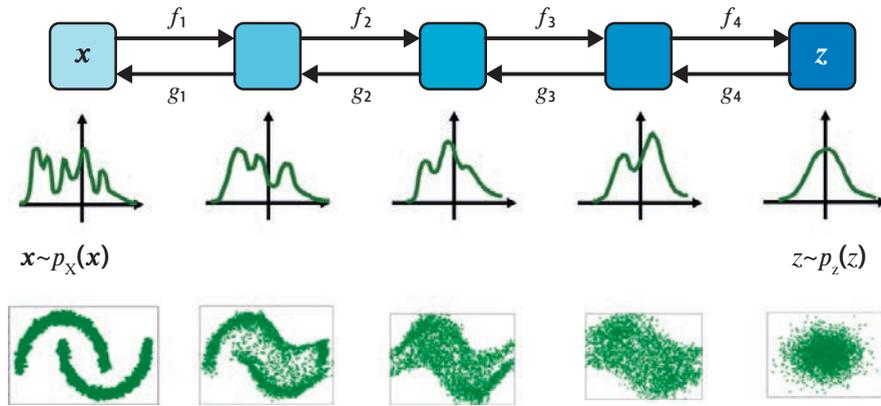


Figure 6.10 : Transformation entre la distribution complexe $p_x(x)$ et une gaussienne simple $p_z(z)$ en 1D (ligne du milieu) et 2D (ligne du bas).

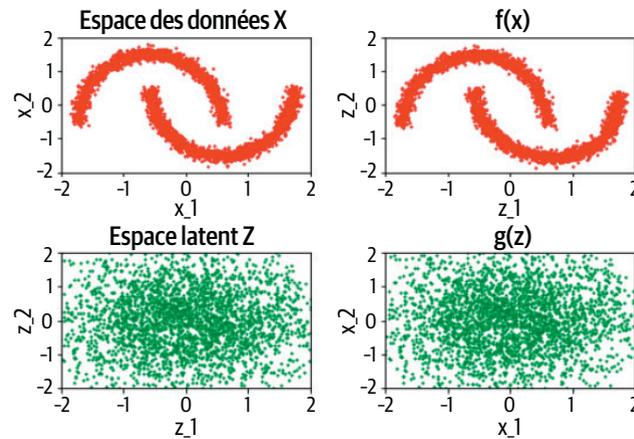


Figure 6.11 : Entrées (à gauche) et sorties (à droite) du modèle RealNVP avant l'entraînement, pour le processus direct (en haut) et le processus inverse (en bas).

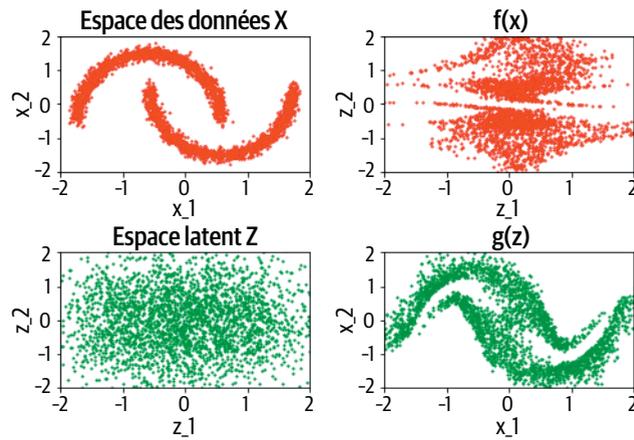


Figure 6.12 : Entrées (à gauche) et sorties (à droite) du modèle RealNVP après entraînement, pour le processus direct (en haut) et le processus inverse (en bas).

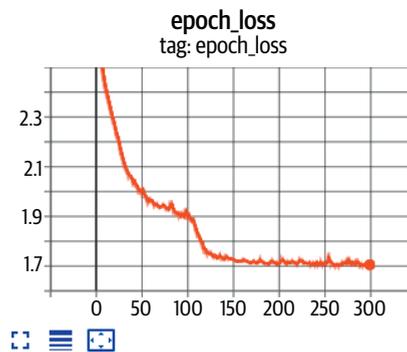


Figure 6.13 : Courbe de perte pour le processus d'entraînement RealNVP.



Figure 6.14 : Échantillons aléatoires à partir du modèle GLOW (source : Kingma et Dhariwal, 2018).

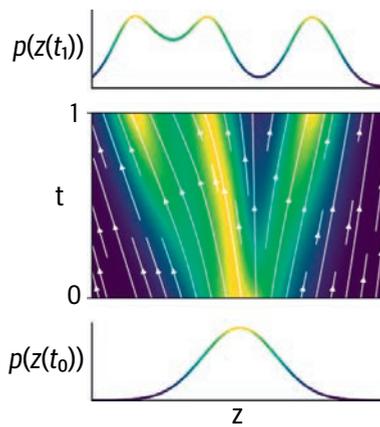


Figure 6.15 : FFJORD modélise la transformation entre la distribution des données et une gaussienne standard via une équation différentielle ordinaire, paramétrée par un réseau de neurones (source : Will Grathwohl et al., 2018).

Chapitre 7. Modèles basés sur l'énergie



Figure 7.1 : Coach entraînant des athlètes pour la compétition (image créée par Midjourney).

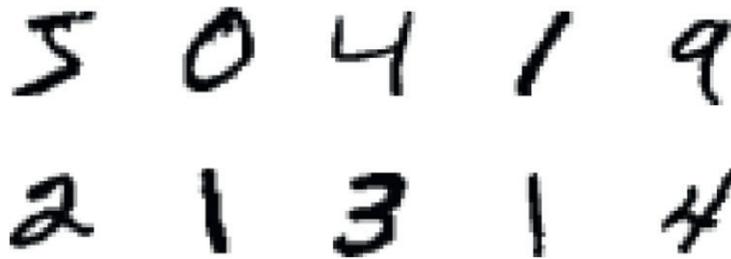


Figure 7.2 : Exemples d'images tirées du jeu de données MNIST.

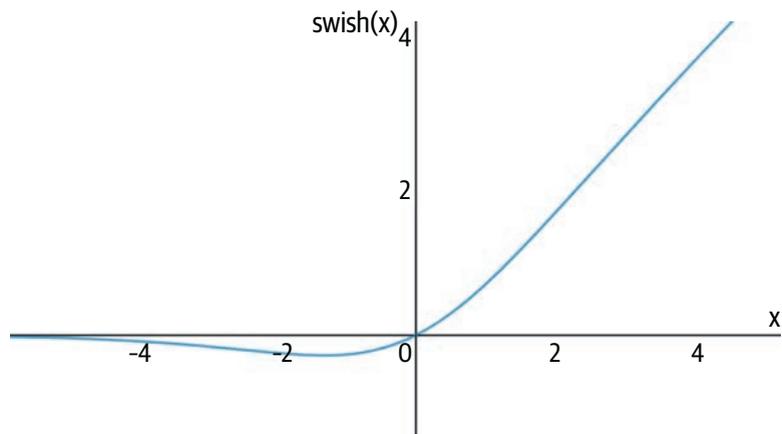


Figure 7.3 : Fonction d'activation swish.

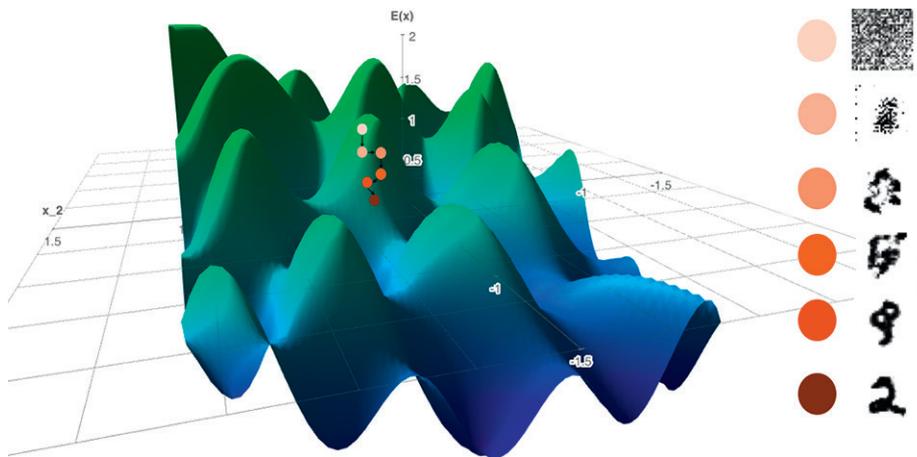


Figure 7.4 : Descente de gradient utilisant la dynamique de Langevin.

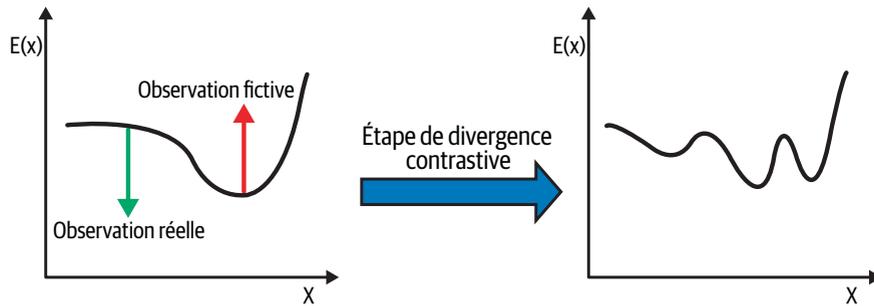


Figure 7.5 : Étape de divergence contrastive.

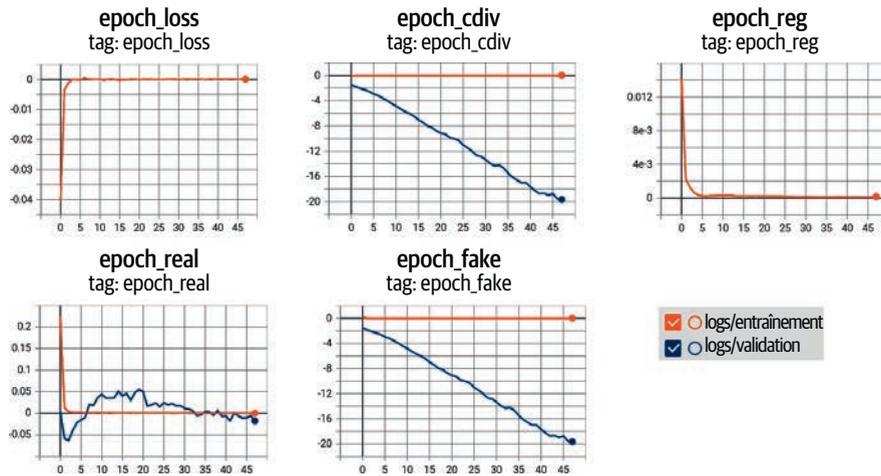


Figure 7.6 : Courbes de perte et métrique du processus d'entraînement de l'EBM.



Figure 7.7 : Exemples produits par l'échantillonneur de Langevin utilisant un EBM pour diriger la descente de gradient.



Figure 7.8 : Instantanés d'une observation à différentes étapes du processus d'échantillonnage de Langevin.

Chapitre 8. Modèles de diffusion



Figure 8.1 : Longue rangée de téléviseurs connectés dans une allée d'un magasin (image créée avec Midjourney).

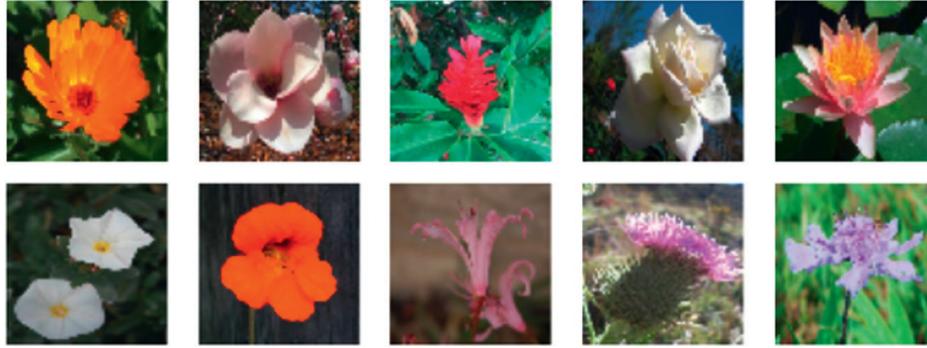


Figure 8.2 : Exemples d'images extraites du jeu de données Oxford 102 Flower.

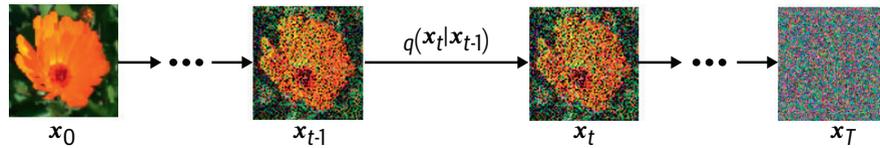


Figure 8.3 : Processus de diffusion direct q .

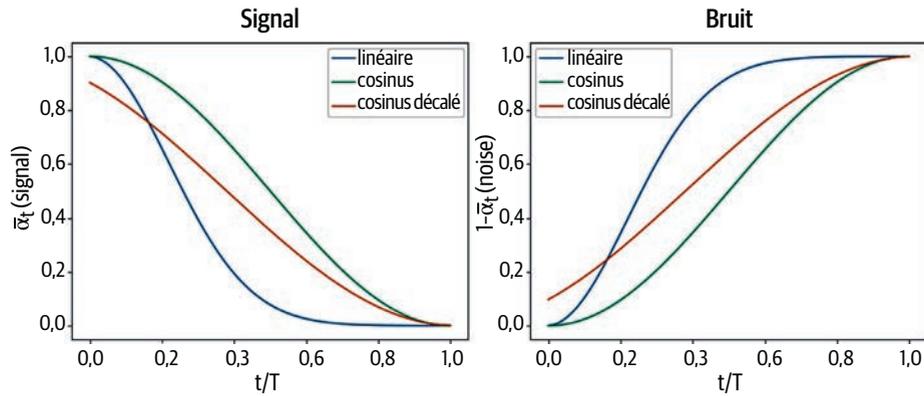


Figure 8.4 : Signal et bruit à chaque étape du processus de bruitage, pour les ordonnancements de diffusion linéaire, en cosinus et en cosinus décalé.



Figure 8.5 : Image dégradée par des ordonnancements de diffusion linéaire (en haut) et en cosinus (en bas), à des valeurs de t espacées de manière égale de 0 à T (source : Ho et al., 2020).

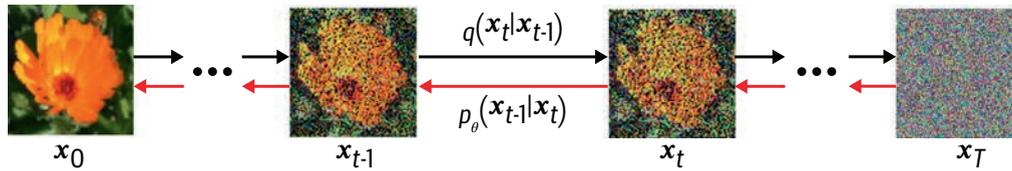


Figure 8.6 : Le processus de diffusion inverse $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ tente d'annuler le bruit produit par le processus de diffusion direct.

Algorithme 1 Entraînement

1 : Répéter

2 : $\mathbf{x}_0 \sim q(\mathbf{x}_0)$

3 : $t \sim \text{Uniform}(\{1, \dots, T\})$

4 : $\epsilon \sim N(\mathbf{0}, \mathbf{I})$

5 : Exécuter l'étape de descente de gradient à

$$\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2$$

6 : Jusqu'à ce que ça converge

Figure 8.7 : Processus d'entraînement d'un modèle de diffusion de débruitage (source : Ho et al., 2020).

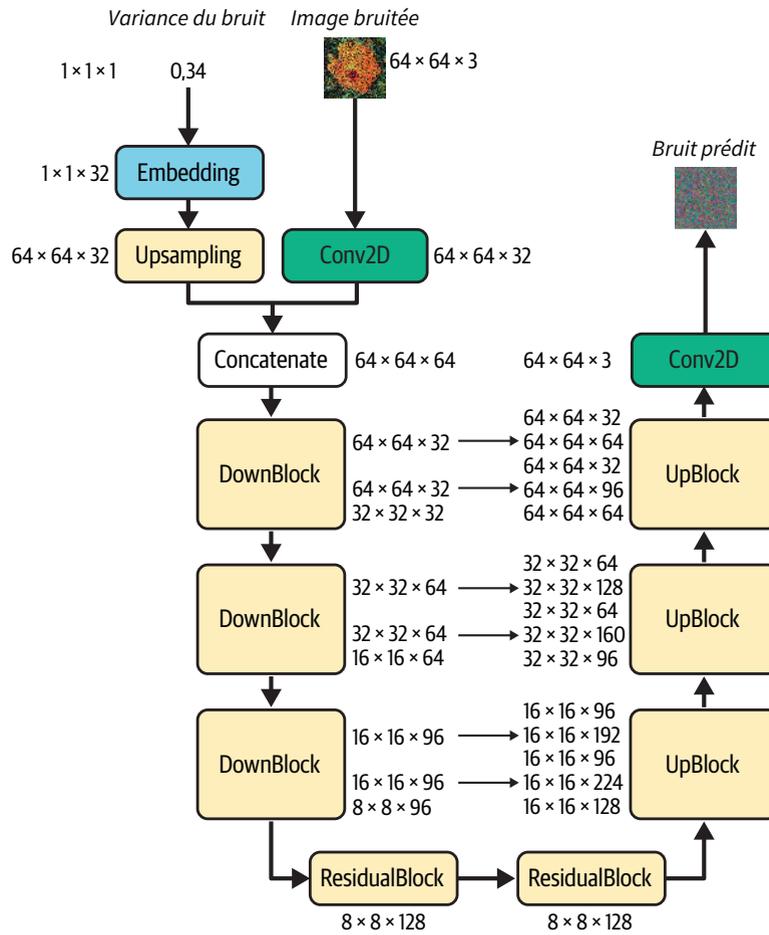


Figure 8.8 : Schéma de l'architecture U-Net.

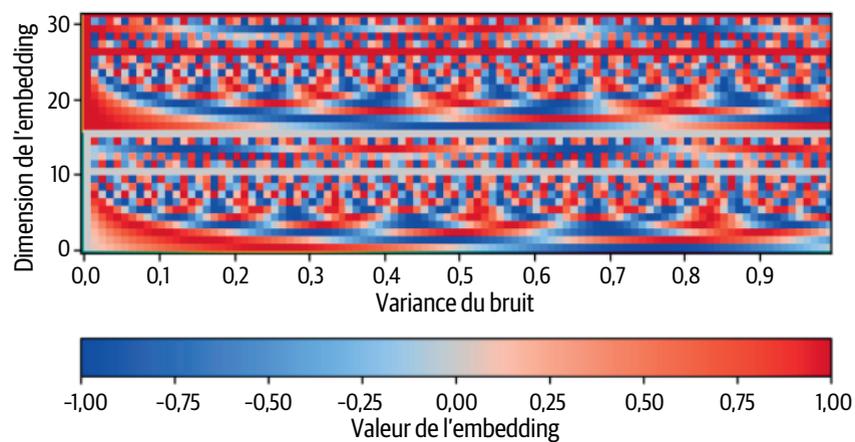


Figure 8.9 : Modèle d'embedding sinusoïdal pour des variances de bruit allant de 0 à 1.

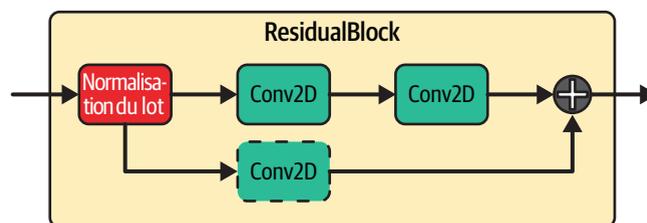


Figure 8.10 : ResidualBlock dans un réseau U-Net.

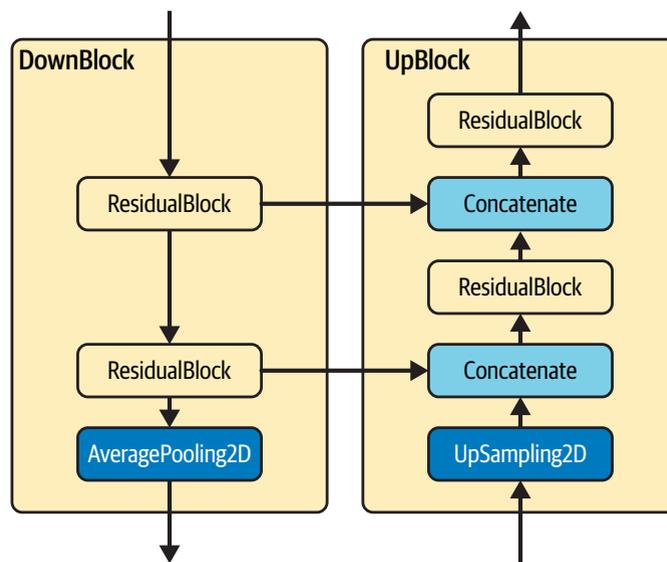


Figure 8.11 : **DownBlock** et **UpBlock** correspondants dans un U-Net.



Figure 8.12 : Courbe de perte de l'erreur absolue moyenne du bruit, par epoch.

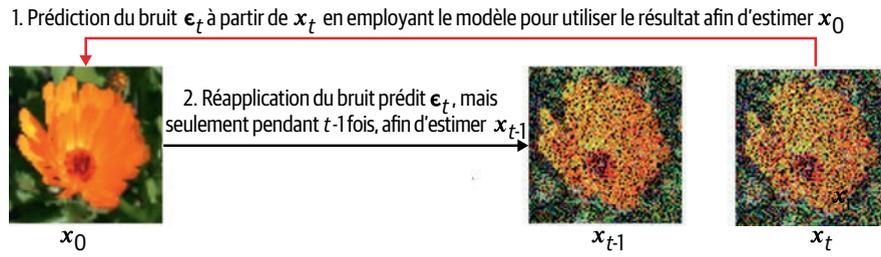


Figure 8.13 : Étape du processus d'échantillonnage pour notre modèle de diffusion.



Figure 8.14 : Échantillons du modèle de diffusion à différentes epochs du processus d'entraînement.

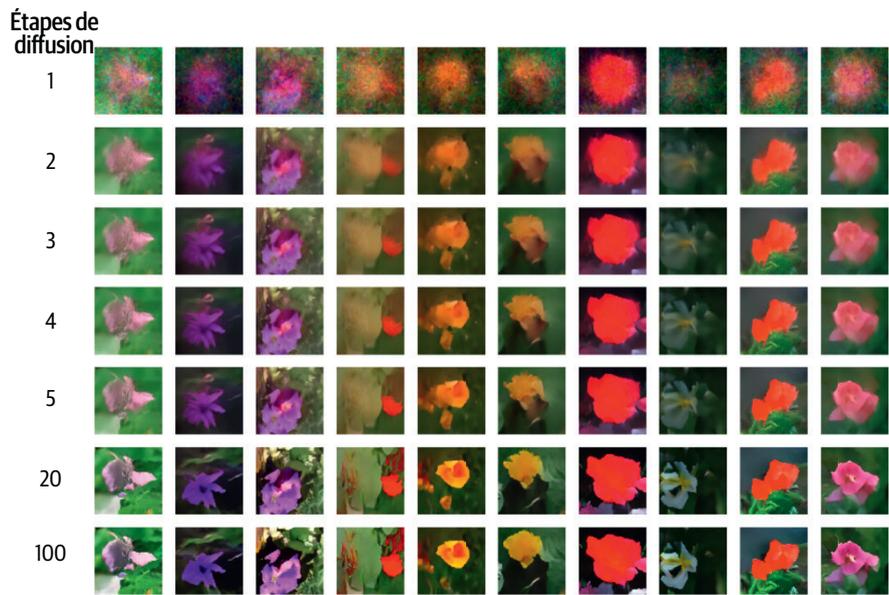


Figure 8.15 : La qualité de l'image s'améliore si l'on augmente le nombre d'étapes de diffusion.

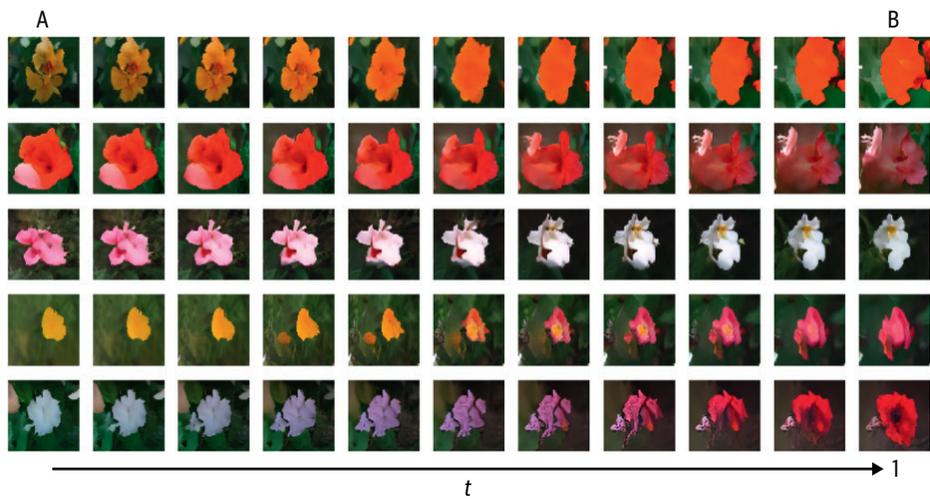


Figure 8.16 : Interpolation entre des images à l'aide d'un modèle de diffusion de débruitage.

Chapitre 9. Transformeurs

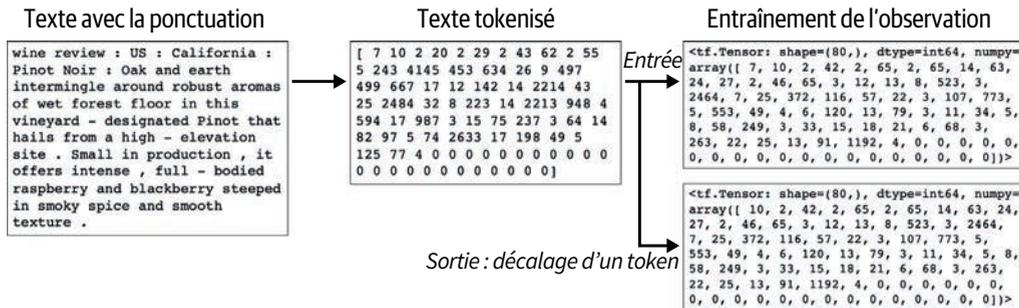


Figure 9.1 : Traitement des données pour le transformeur.

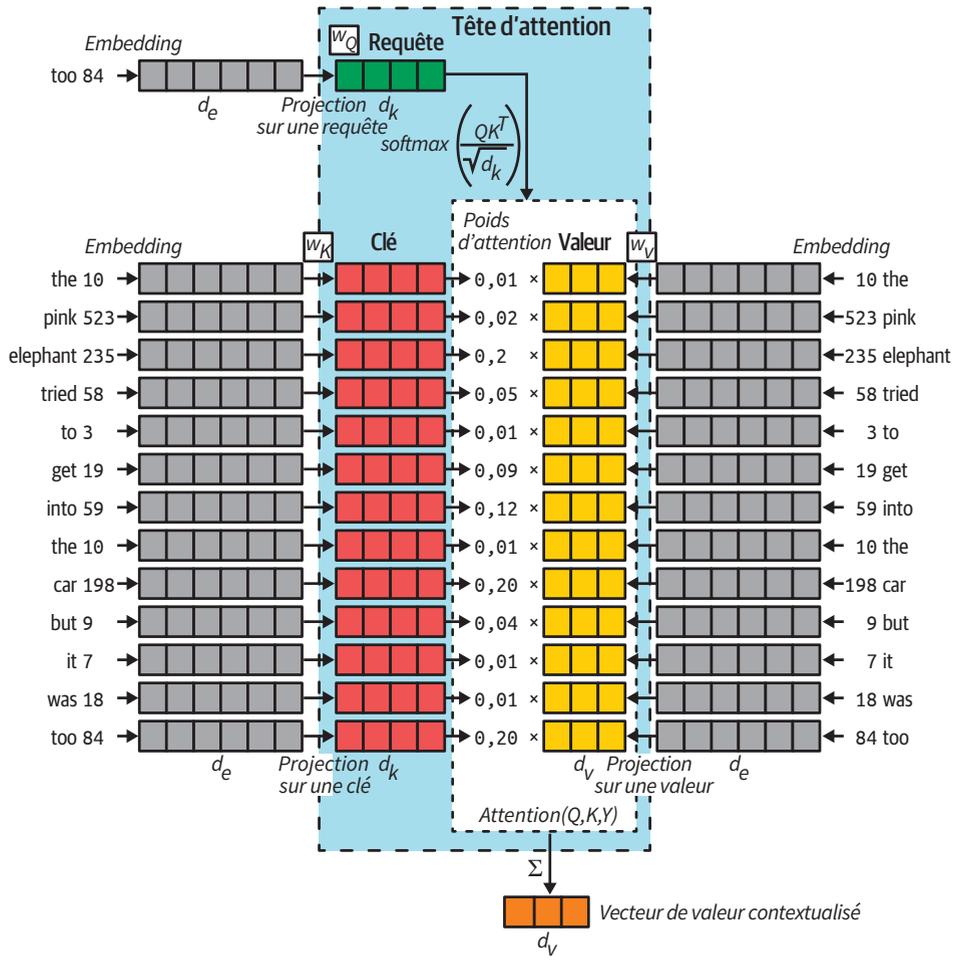


Figure 9.2 : Mécanisme de tête d'attention.

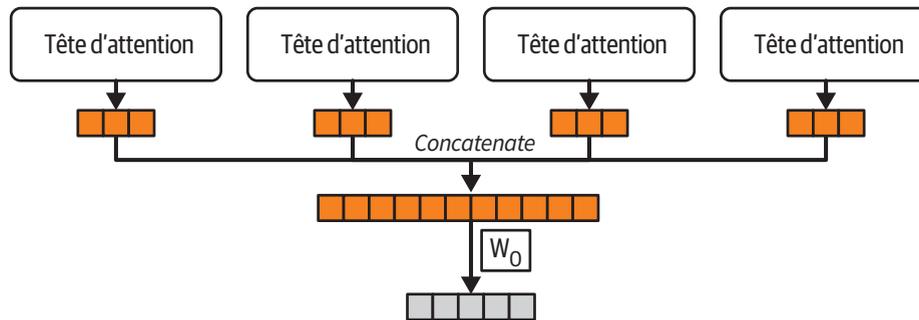


Figure 9.3 : Couche d'attention à quatre têtes.

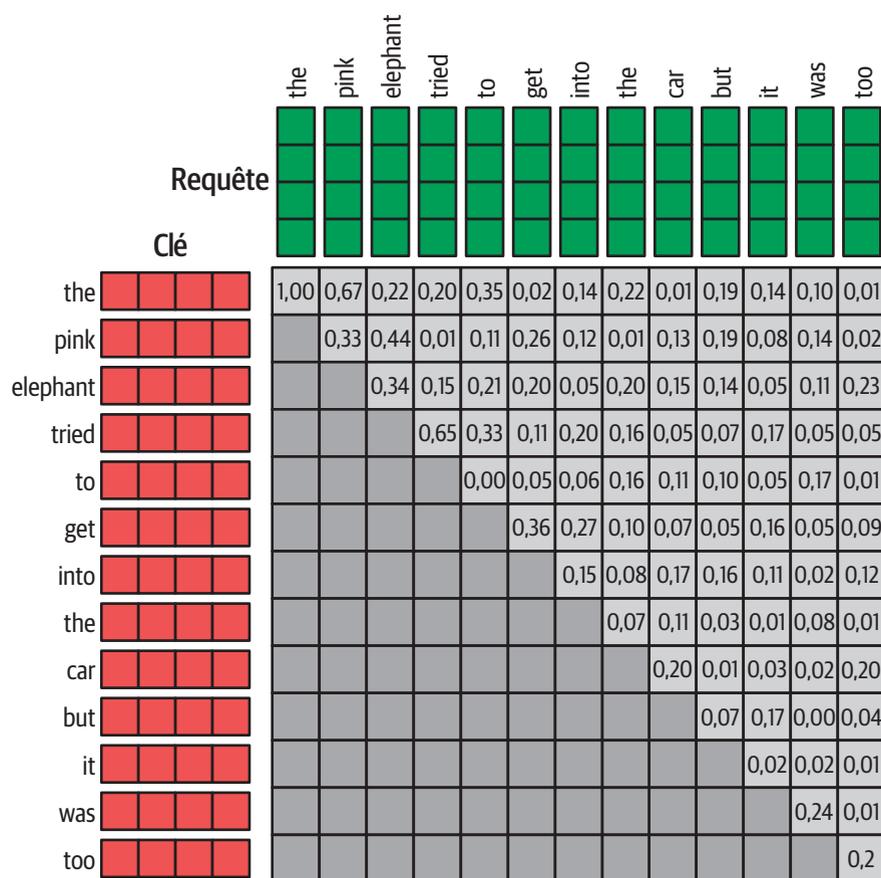


Figure 9.4 : Calcul matriciel des scores d'attention pour un lot de requêtes d'entrée, en utilisant un masque causal d'attention afin de masquer les clés qui ne sont pas disponibles pour la requête (parce qu'elles viennent plus tard dans la phrase).

```
array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [0, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [0, 0, 1, 1, 1, 1, 1, 1, 1, 1],
       [0, 0, 0, 1, 1, 1, 1, 1, 1, 1],
       [0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
       [0, 0, 0, 0, 0, 1, 1, 1, 1, 1],
       [0, 0, 0, 0, 0, 0, 1, 1, 1, 1],
       [0, 0, 0, 0, 0, 0, 0, 1, 1, 1],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 1],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]], dtype=int32)
```

Figure 9.5 : Masque causal sous la forme d'un tableau *numpy* ; -1 signifie non masqué et 0 signifie masqué.

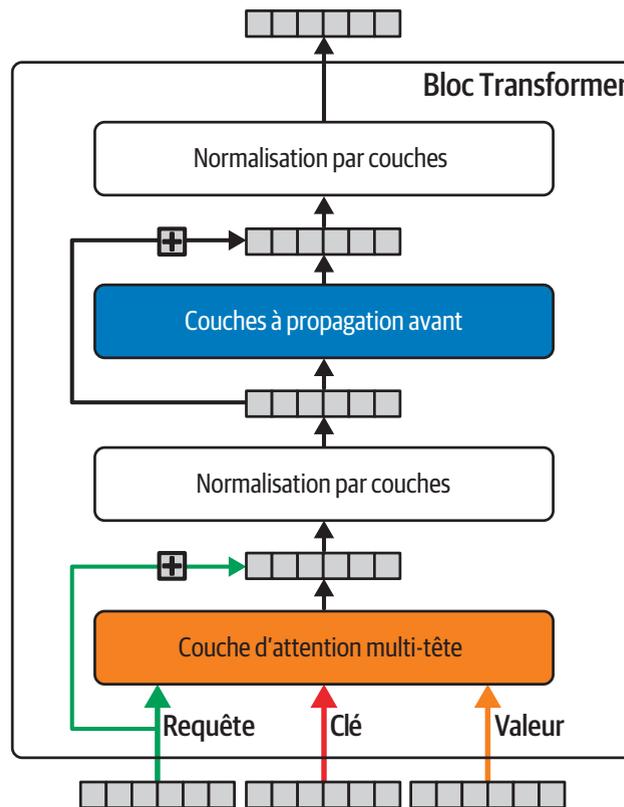


Figure 9.6 : Bloc Transformer.

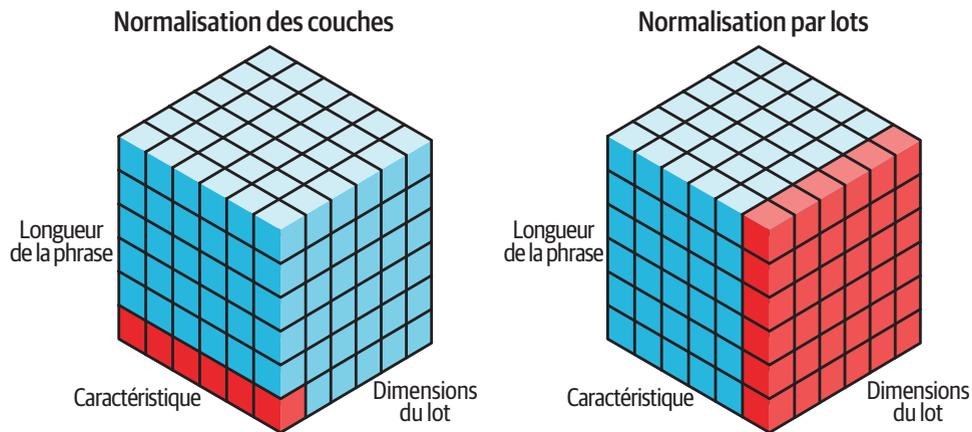


Figure 9.7 : Normalisation par couches contre normalisation par lots ; les statistiques de normalisation sont calculées dans les cellules bleues (source : Sheng et al., 2020).

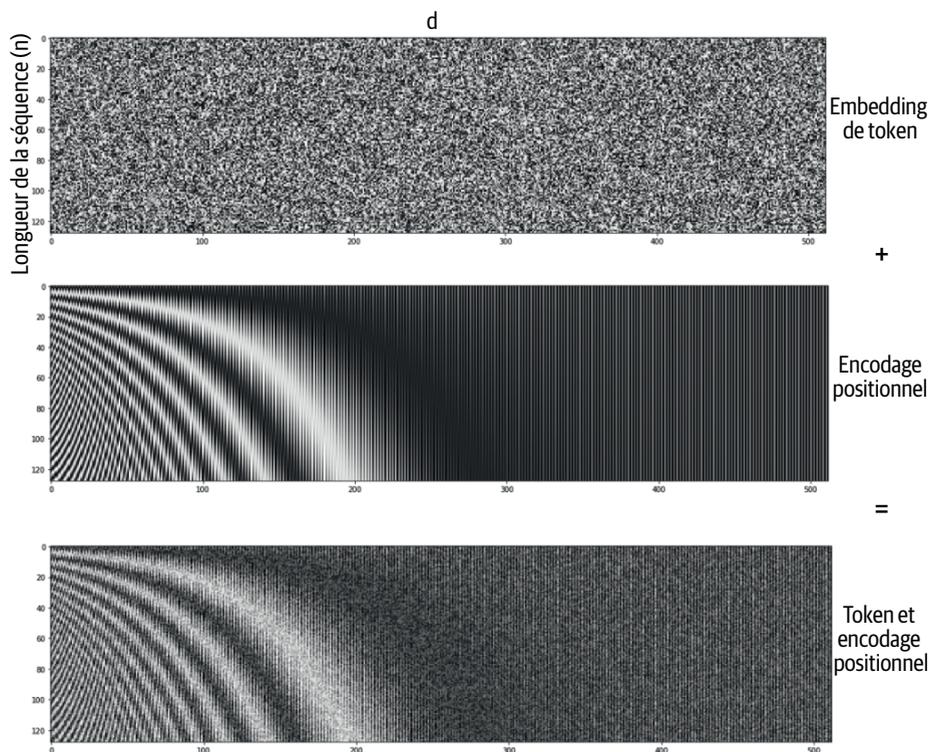


Figure 9.8 : Les embeddings de token sont ajoutés aux embeddings de position pour obtenir l'encodage positionnel des tokens.

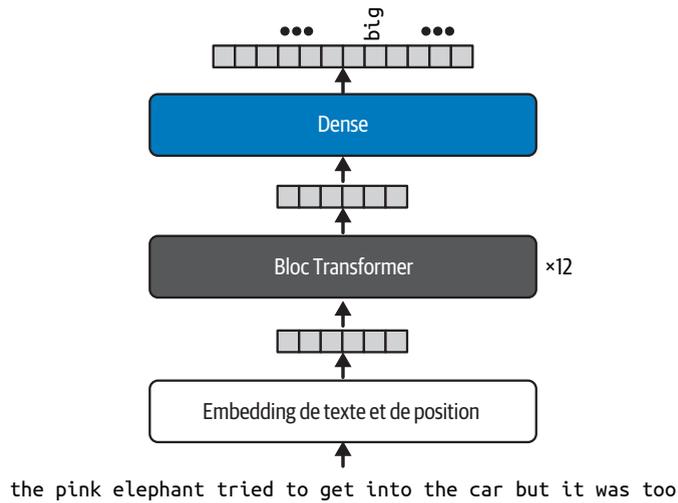


Figure 9.9 : Architecture simplifiée du modèle GPT.

température = 1,0

Texte généré :
 Critique de vin : USA : Washington : chenin blanc : un vin léger, moyennement corsé, ce vin expressif léger n'est pas très agréable à boire. Il est simple avec des arômes de beurre et de vanille qui se mélangent à des notes de fruit expressives. Il est juteux et acidulé avec une note persistante de citron en fin de bouche.

température = 0,5

Texte généré :
 Critique de vin : Italie : Piémont : nebbiolo : ce vin s'ouvre sur des arômes de chêne français, de menthol et un soupçon de pain grillé. Le palais simple offre des cerises rouges, des framboises noires et un soupçon d'anis étoilé, ainsi que des tanins fermes mais plutôt fugaces. À boire jusqu'en 2016.

Figure 9.10 : Sorties générées avec les paramètres *température = 1,0* et *température = 0,5*.

wine review : germany :

pfalz:	51.53%
mosel:	41.21%
rheingau:	4.27%
rheinhessen:	2.16%
franken:	0.44%

wine review : germany : rheingau : riesling : this is a ripe , full - bodied

riesling:	46.56%
,:	27.78%
wine:	16.88%
and:	4.58%
yet:	1.33%

wine review : germany : rheingau : riesling : this is a ripe , full - bodied riesling

with a touch of residual sugar . it ' s a slightly

sweet:	94.23%
oily:	1.25%
viscous:	1.09%
bitter:	0.88%
honeyed:	0.66%

Figure 9.11 : Distribution de probabilités de mots suivant différentes séquences.

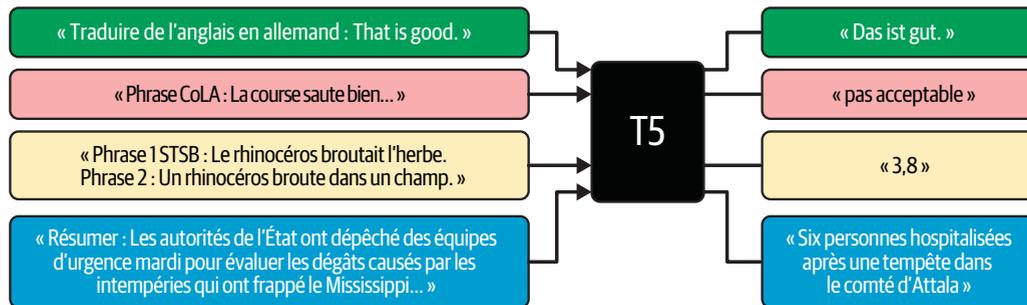


Figure 9.12 : Exemples de la façon dont T5 redéfinit une série de tâches (traduction, acceptabilité, similarité et résumé) dans un framework textuel (source : Raffel et al., 2019).

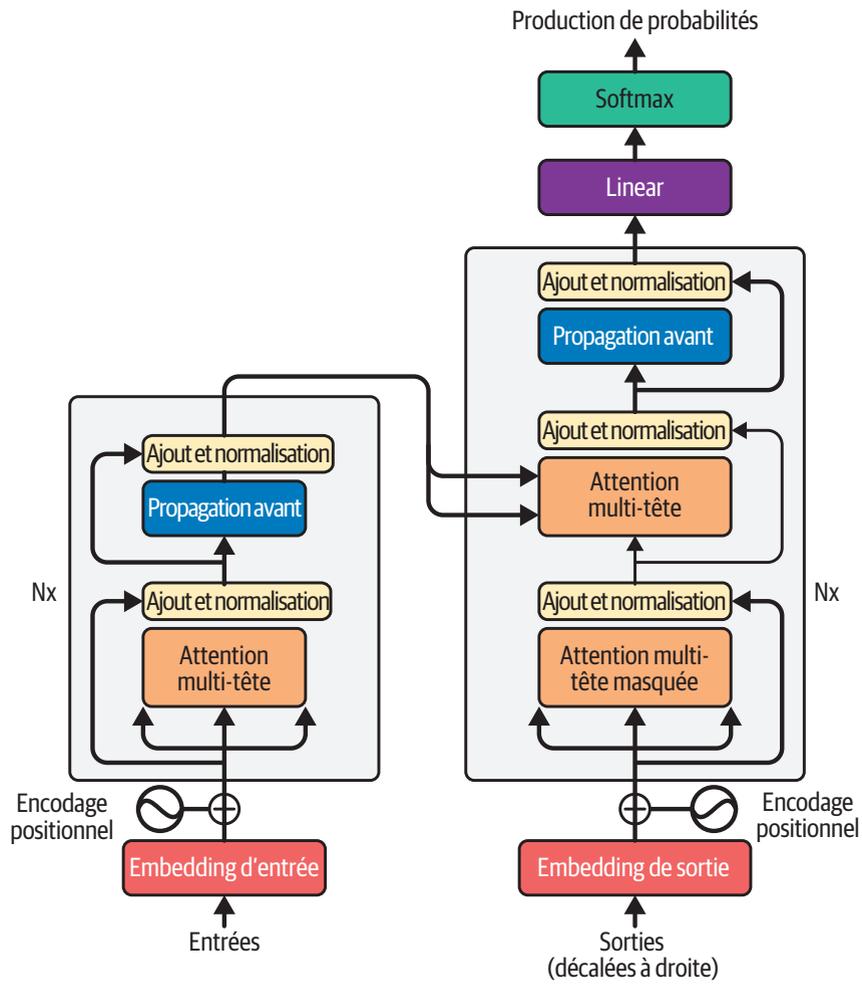


Figure 9.13 : Modèle de transformeur encodeur-décodeur : chaque encadré gris est un bloc Transformer (source : Vaswani et al., 2017).

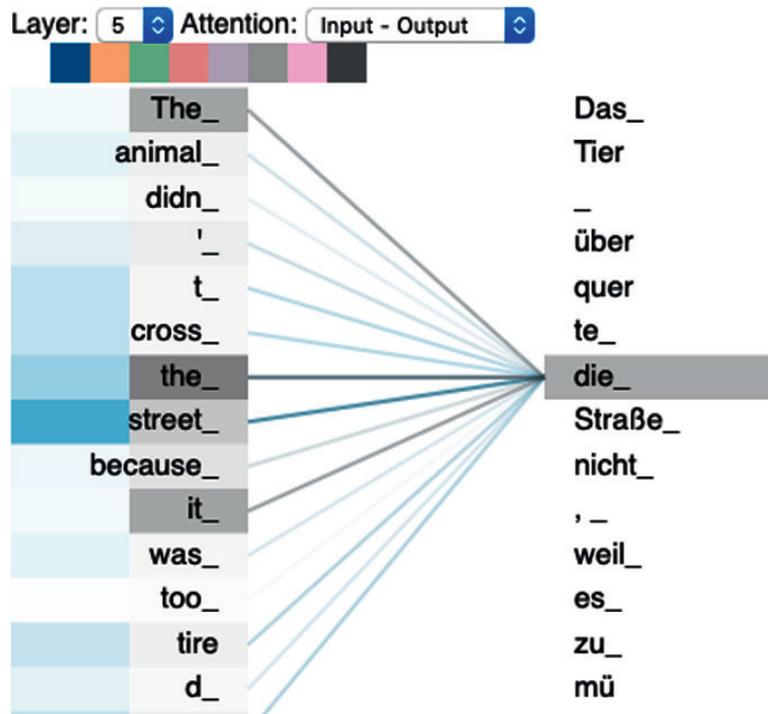


Figure 9.14 : Exemple de la façon dont une tête d'attention s'occupe du mot « the » et une autre du mot « street » afin de traduire correctement le mot « the » par le mot allemand « die » en tant qu'article défini féminin de « Straße ».

Écris-moi une histoire courte sur une licorne qui vainc un dragon maléfique.

Dans la vallée enchantée de Lumios, une licorne majestueuse du nom d'Aurora vivait paisiblement. Un jour, les cieux s'assombrirent lorsque Drakon, un dragon maléfique, envahit la contrée. Ses flammes noires dévastèrent les terres et semèrent la terreur.

Aurora, guidée par son cœur pur, prit courageusement la tête de la résistance. Sa crinière étincelante et sa corne lumineuse irradièrent une aura magique. Elle se dirigea vers la montagne où Drakon résidait, déterminée à sauver Lumios.

Un duel épique éclata entre la licorne lumineuse et le dragon obscur. Les flammes de Drakon cherchaient à dévorer l'éclat pur d'Aurora. Cependant, la licorne esquiva avec grâce, utilisant sa corne pour renvoyer des éclairs magiques.

Guidée par la puissance de son amour pour Lumios, Aurora réussit à briser les chaînes de la noirceur. Sa corne émit une lumière éblouissante qui enveloppa Drakon, le purifiant de sa malice. Le dragon obscur se transforma en une créature bienveillante, libéré de son ancienne malédiction.

La vallée retrouva sa splendeur, et Aurora, avec le dragon transformé à ses côtés, devint la gardienne de Lumios. Leur amitié improbable rappela à tous que même les ténèbres les plus sombres pouvaient être dissipées par la lumière de la bonté.

Figure 9.15 : Exemple de la façon dont GPT-3 peut réagir à un message saisi dans l'invite du système.

You

Qu'est-ce qu'un modèle Transformer et comment est-il utilisé dans l'apprentissage profond génératif ? Peux-tu me donner un résumé en 200 mots ?

ChatGPT

Un modèle Transformer est une architecture de réseau neuronal sans récurrence, introduite par Vaswani et al. en 2017. Il se distingue par son mécanisme d'attention, permettant une efficacité remarquable dans le traitement des séquences de données. Plutôt que de s'appuyer sur des récurrences, le Transformer utilise des mécanismes d'attention pour capturer les relations à longue portée entre les éléments de séquence, rendant son apprentissage parallélisable et accélérant considérablement les performances.

Dans l'apprentissage profond génératif, les Transformers sont utilisés pour générer des données, tels que des modèles de langage. En utilisant des architectures comme le GPT (Generative Pre-trained Transformer), ces modèles apprennent à prédire et générer des séquences de manière créative. Ils sont pré-entraînés sur de vastes corpus de données, capturant la structure et la sémantique du langage, puis affinés pour des tâches spécifiques. Ces modèles génératifs utilisent l'attention pour contextualiser chaque élément de la séquence en fonction de son contexte global, permettant une génération cohérente et contextuellement riche de texte, contribuant ainsi à des applications telles que la création automatique de contenu ou la traduction de langues.

You

Qu'est-ce que l'attention ? Réponds-moi sous forme de limerick.

ChatGPT

Il était une fois, l'attention si fine,
Dans le réseau neuronal, elle câline.
Elle danse entre les nœuds,
Comme un doux flambeau,
Éclairant chaque pensée, divine.

You

Si un Transformer était un animal, quel serait cet animal et pourquoi ?

ChatGPT

Si un Transformer était un animal, il serait probablement un caméléon. Comme le caméléon ajuste ses couleurs pour se fondre dans son environnement, le Transformer adapte son attention et sa compréhension contextuelle pour traiter efficacement les informations. De plus, tout comme le caméléon peut se déplacer rapidement et capturer sa proie avec agilité, le Transformer excelle dans la manipulation et la génération dynamique de données, montrant une adaptabilité similaire à celle de ce reptile étonnant.

Figure 9.16 : Exemple d'utilisation de ChatGPT répondant à des questions sur les modèles Transformer.

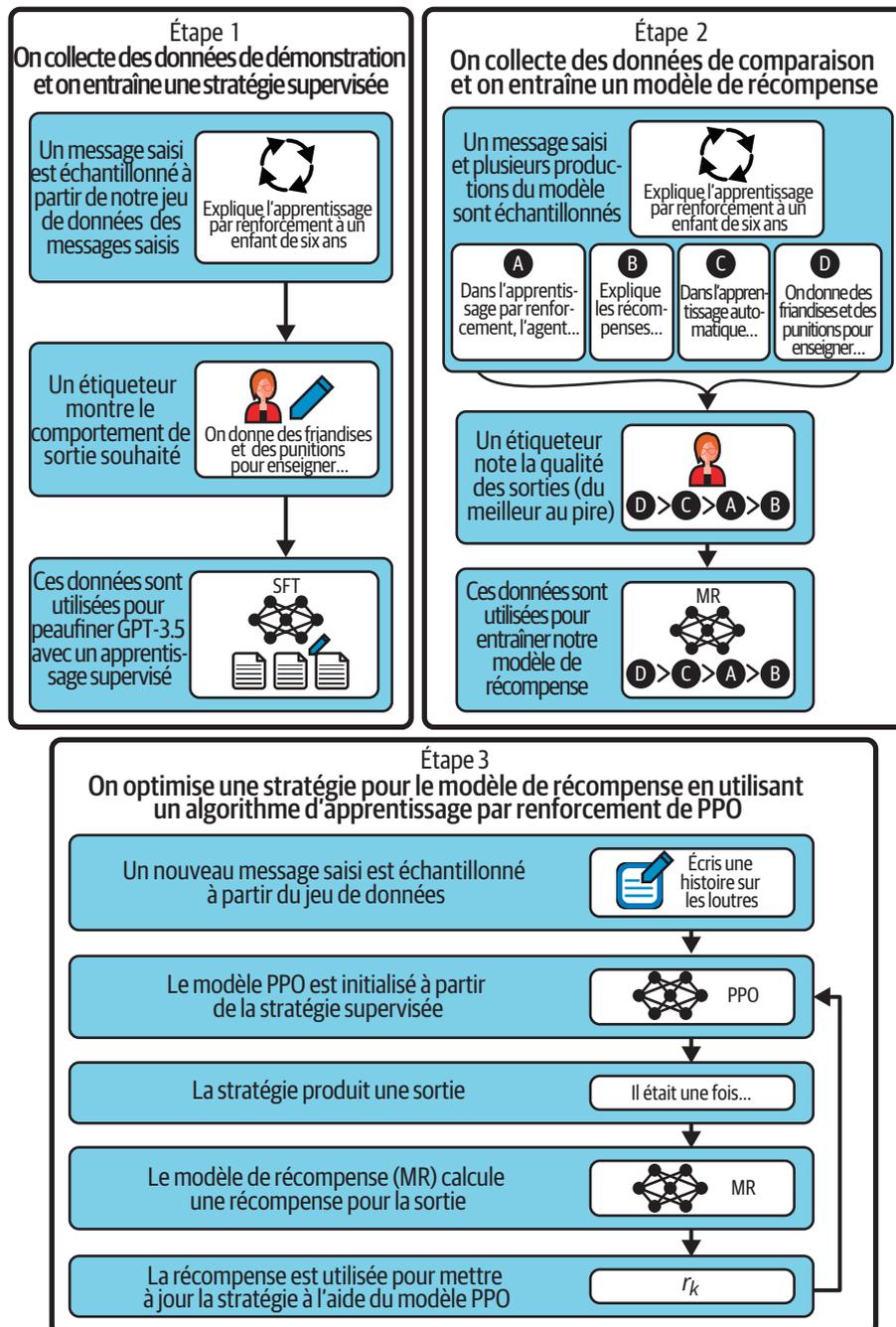


Figure 9.17 : Processus de peaufinage employant un apprentissage par renforcement à partir de la

rétroaction humaine utilisé dans ChatGPT (source : OpenAI ; <https://openai.com/blog/chatgpt>).

Chapitre 10. GAN avancés

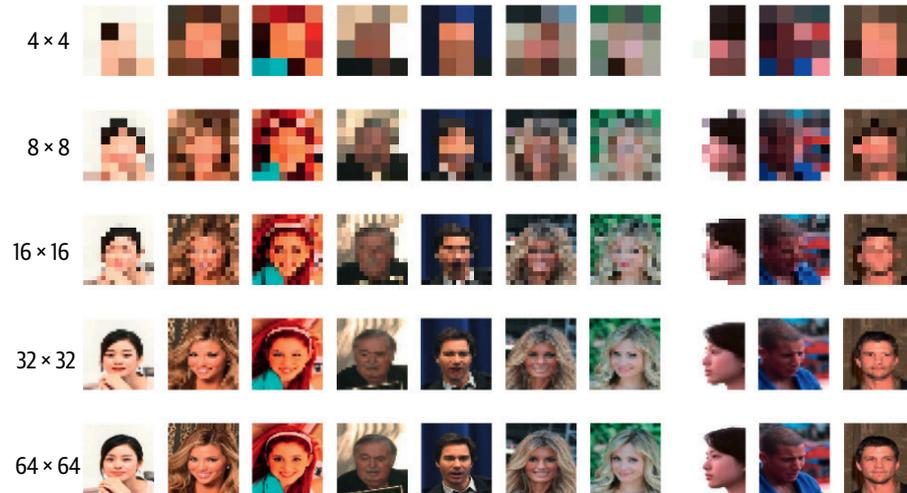


Figure 10.1 : Les images du jeu de données peuvent être compressées en une résolution plus faible par interpolation.

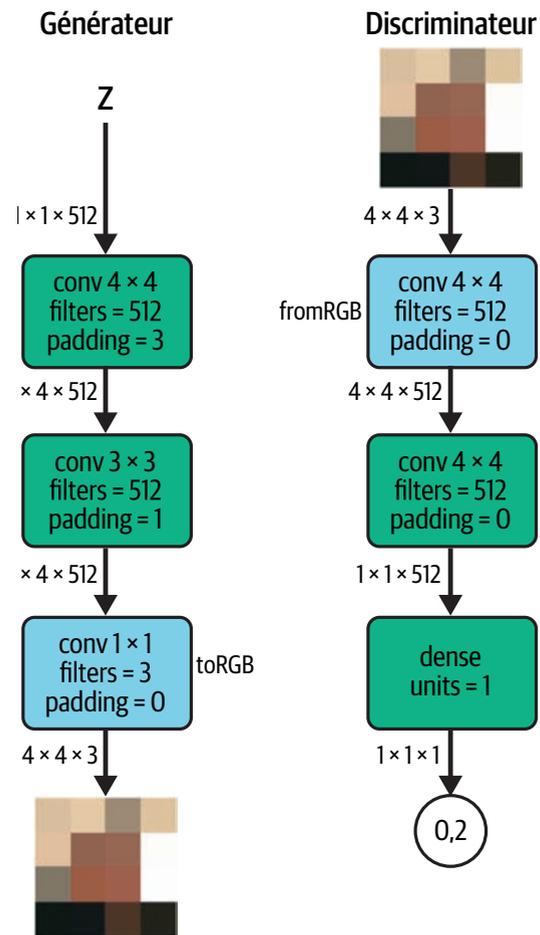


Figure 10.2 : Architectures du générateur et du discriminateur pour la première étape du processus d'apprentissage du ProGAN.

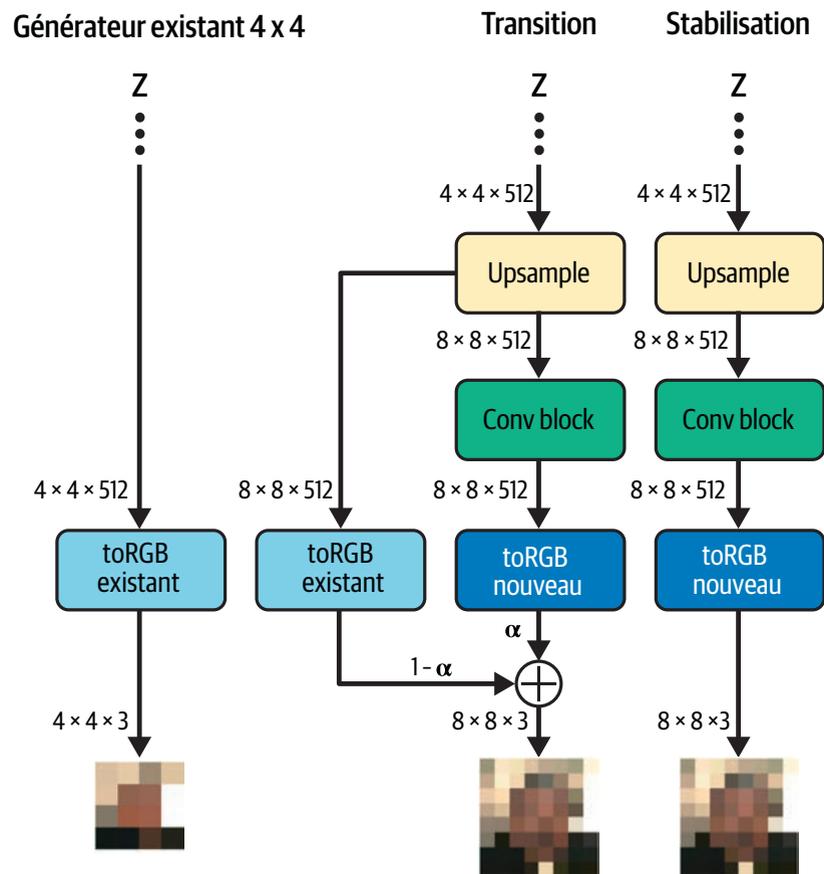


Figure 10.3 : Processus d'entraînement du générateur du ProGAN, étendant le réseau d'images en 4×4 à des images en 8×8 (les lignes pointillées représentent le reste du réseau, qui n'est pas illustré).

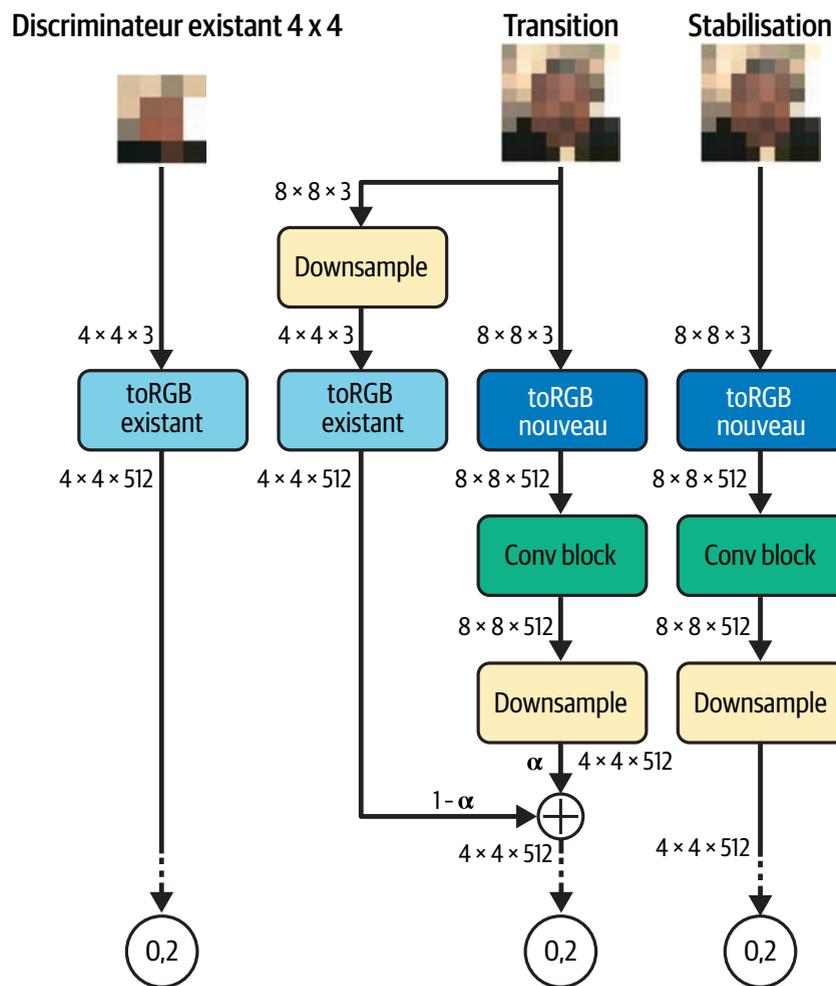


Figure 10.4 : Processus d'entraînement du discriminateur du ProGAN, étendant le réseau d'images en 4×4 à des images en 8×8 (les lignes pointillées représentent le reste du réseau, qui n'est pas illustré).

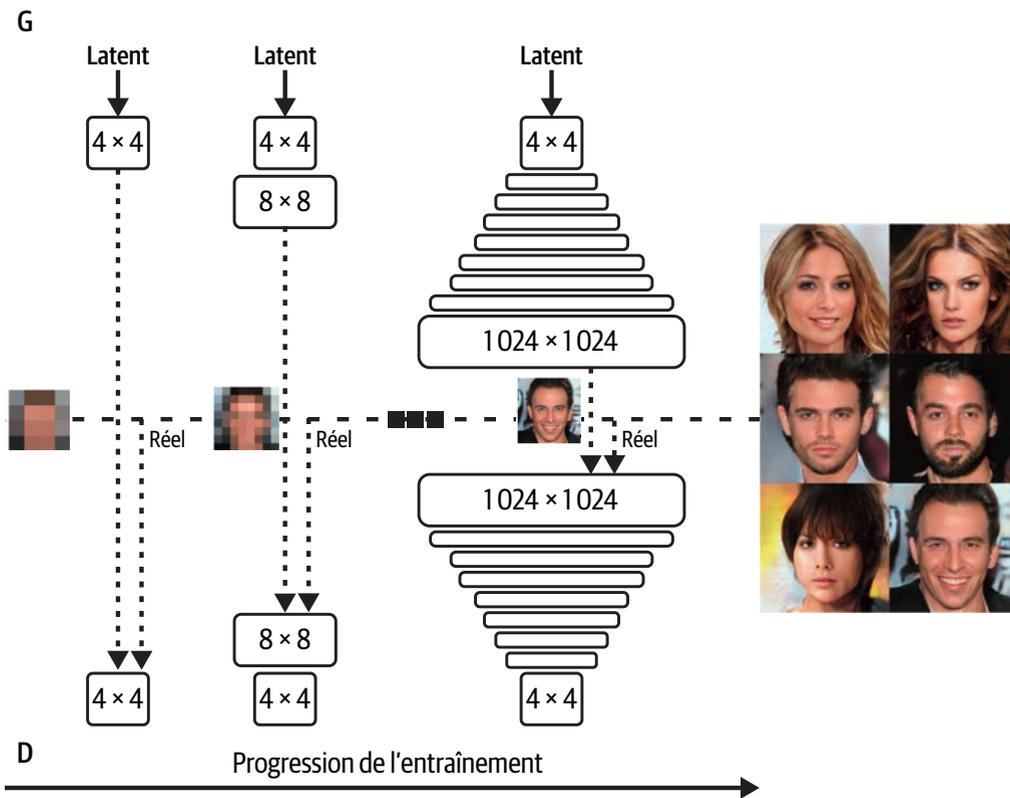


Figure 10.5 : Mécanisme d'entraînement d'un ProGAN, et exemples de visages générés (source : Karras et al., 2017).

Générateur	Activation	Format de sortie	Paramètres	Discriminateur	Activation	Format de sortie	Paramètres
Vecteur latent	—	512 × 1 × 1	—	Image d'entrée	—	3 × 1 024 × 1 024	—
Conv 4 × 4	LReLU	512 × 4 × 4	4,2 M	Conv 1 × 1	LReLU	16 × 1 024 × 1 024	64
Conv 3 × 3	LReLU	512 × 4 × 4	2,4 M	Conv 3 × 3	LReLU	16 × 1 024 × 1 024	2,3 k
Suréchantillonnage	—	512 × 8 × 8	—	Conv 3 × 3	LReLU	32 × 1 024 × 1 024	4,6 k
Conv 3 × 3	LReLU	512 × 8 × 8	2,4 M	Sous-échantillonnage	—	32 × 512 × 512	—
Conv 3 × 3	LReLU	512 × 8 × 8	2,4 M	Conv 3 × 3	LReLU	32 × 512 × 512	9,2 k
Suréchantillonnage	—	512 × 16 × 16	—	Conv 3 × 3	LReLU	64 × 512 × 512	18 k
Conv 3 × 3	LReLU	512 × 16 × 16	2,4 M	Sous-échantillonnage	—	64 × 256 × 256	—
Conv 3 × 3	LReLU	512 × 16 × 16	2,4 M	Conv 3 × 3	LReLU	64 × 256 × 256	37 k
Suréchantillonnage	—	512 × 32 × 32	—	Conv 3 × 3	LReLU	128 × 256 × 256	74 k
Conv 3 × 3	LReLU	512 × 32 × 32	2,4 M	Sous-échantillonnage	—	128 × 128 × 128	—
Conv 3 × 3	LReLU	512 × 32 × 32	2,4 M	Conv 3 × 3	LReLU	128 × 128 × 128	148 k
Suréchantillonnage	—	512 × 64 × 64	—	Conv 3 × 3	LReLU	256 × 128 × 128	295 k
Conv 3 × 3	LReLU	256 × 64 × 64	1,2 M	Sous-échantillonnage	—	256 × 64 × 64	—
Conv 3 × 3	LReLU	256 × 64 × 64	590 k	Conv 3 × 3	LReLU	256 × 64 × 64	590 k
Suréchantillonnage	—	256 × 128 × 128	—	Conv 3 × 3	LReLU	512 × 64 × 64	1,2 M
Conv 3 × 3	LReLU	128 × 128 × 128	295 k	Sous-échantillonnage	—	512 × 32 × 32	—
Conv 3 × 3	LReLU	128 × 128 × 128	148 k	Conv 3 × 3	LReLU	512 × 32 × 32	2,4 M
Suréchantillonnage	—	128 × 256 × 256	—	Conv 3 × 3	LReLU	512 × 32 × 32	2,4 M
Conv 3 × 3	LReLU	64 × 256 × 256	74 k	Sous-échantillonnage	—	512 × 16 × 16	—
Conv 3 × 3	LReLU	64 × 256 × 256	37 k	Conv 3 × 3	LReLU	512 × 16 × 16	2,4 M
Suréchantillonnage	—	64 × 512 × 512	—	Conv 3 × 3	LReLU	512 × 16 × 16	2,4 M
Conv 3 × 3	LReLU	32 × 512 × 512	18 k	Sous-échantillonnage	—	512 × 8 × 8	—
Conv 3 × 3	LReLU	32 × 512 × 512	9,2 k	Conv 3 × 3	LReLU	512 × 8 × 8	2,4 M
Suréchantillonnage	—	32 × 1 024 × 1 024	—	Conv 3 × 3	LReLU	512 × 8 × 8	2,4 M
Conv 3 × 3	LReLU	16 × 1 024 × 1 024	4,6 k	Sous-échantillonnage	—	512 × 4 × 4	—
Conv 3 × 3	LReLU	16 × 1 024 × 1 024	2,3 k	Écart type des mini lots	—	512 × 4 × 4	—
Conv 1 × 1	Linéaire	3 × 1 024 × 1 024	51	Conv 3 × 3	LReLU	512 × 4 × 4	2,4 M
Nombre total de paramètres entraînaibles			23,1 M	Conv 4 × 4	LReLU	512 × 1 × 1	4,2 M
				Complètement connecté	Linéaire	1 × 1 × 1	513
				Nombre total de paramètres entraînaibles			23,1 M

Figure 10.6 : Générateur et discriminateur ProGAN utilisés pour générer des visages en $1\,024 \times 1\,024$ pixels (source : Karras et al., 2018 ; <https://arxiv.org/abs/1812.04948>).



Figure 10.7 : Exemples générés à partir d'un ProGAN entraîné progressivement sur le jeu de données LSUN à une résolution de 256×256 (source : Karras et al., 2017).

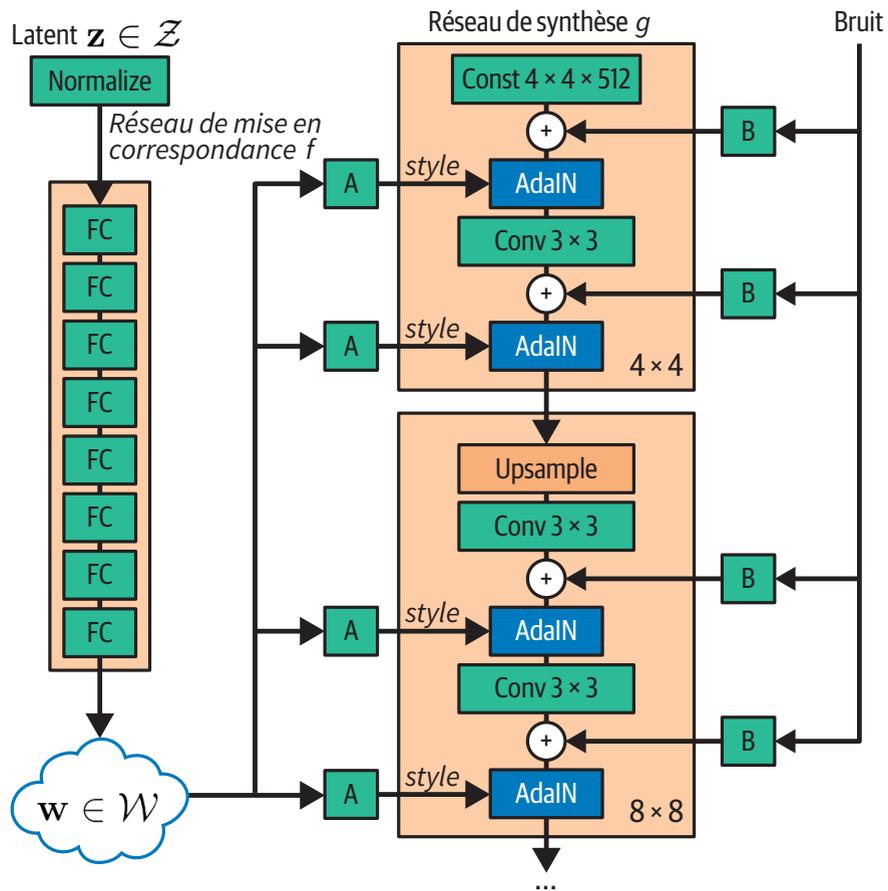


Figure 10.8 : Architecture du générateur du modèle StyleGAN (source : Karras et al., 2018).

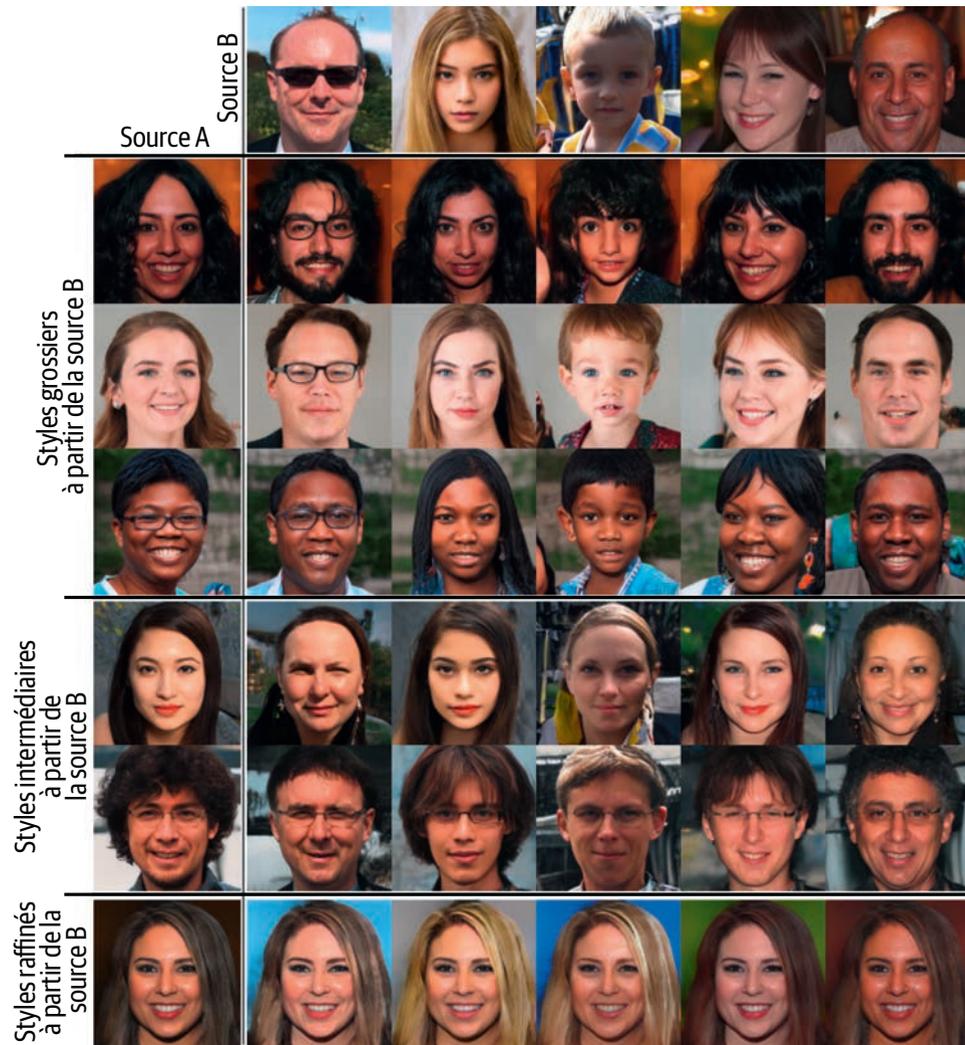


Figure 10.9 : Fusion de styles entre deux images générées à différents niveaux de détail (source : Karras et al., 2018).

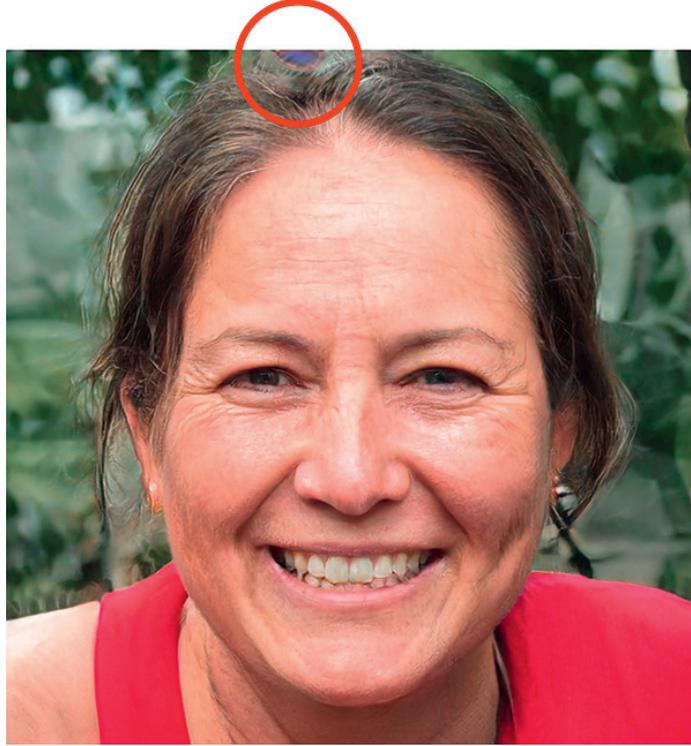


Figure 10.10 : Artefact dans une image de visage générée par StyleGAN (source : Karras et al., 2019).

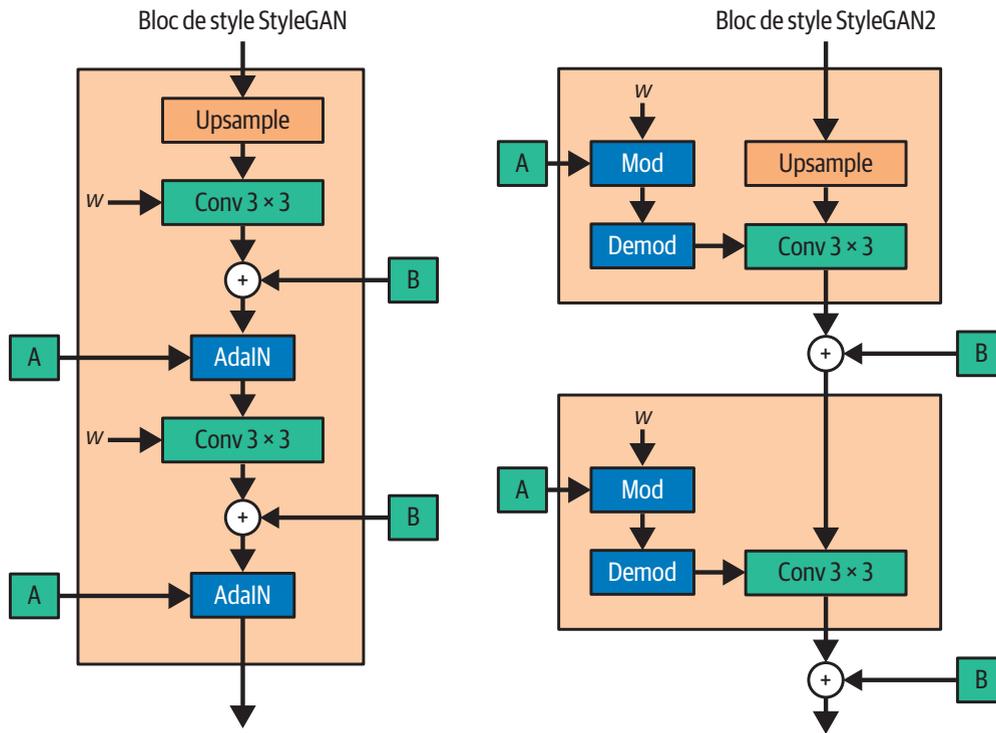


Figure 10.11 : Comparaison entre les blocs de style StyleGAN et StyleGAN2.

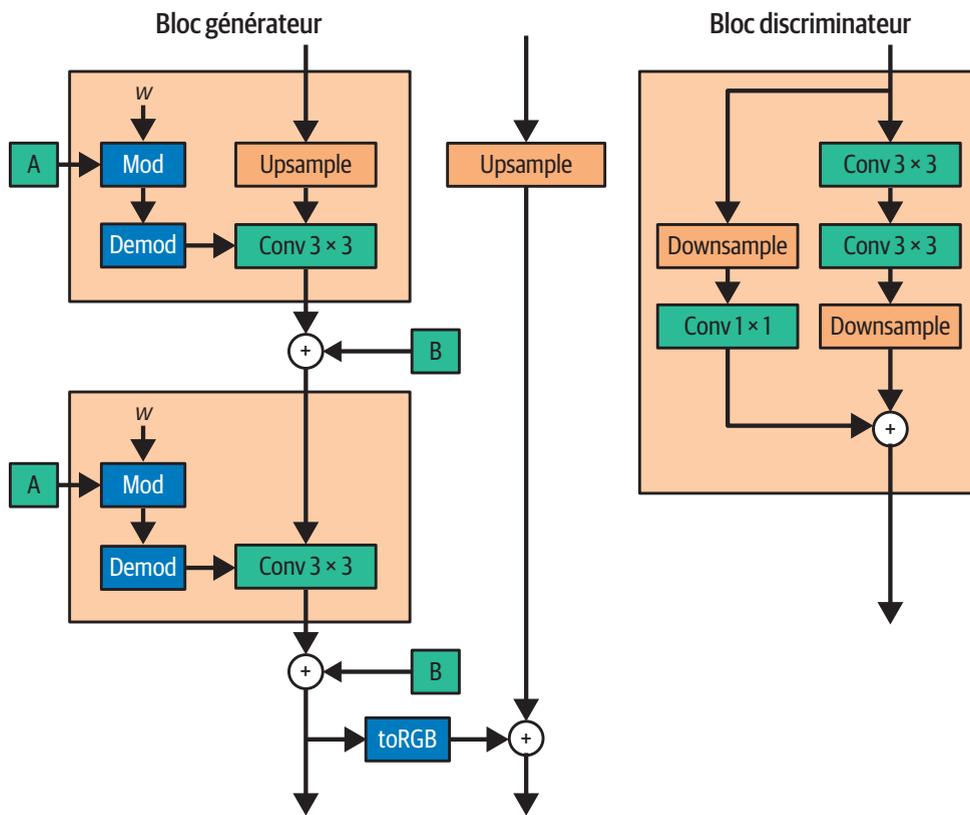


Figure 10.12 : Blocs générateur et discriminateur dans StyleGAN2.

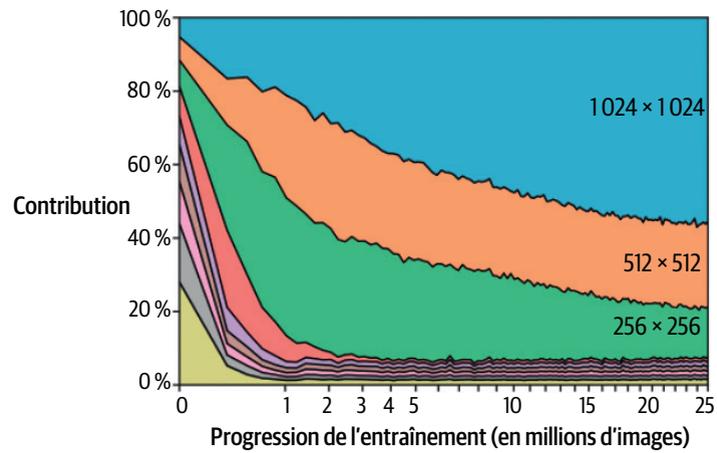


Figure 10.13 : La contribution de chaque couche de résolution à la sortie du générateur, par période d'entraînement (adapté de Karras et al., 2019).

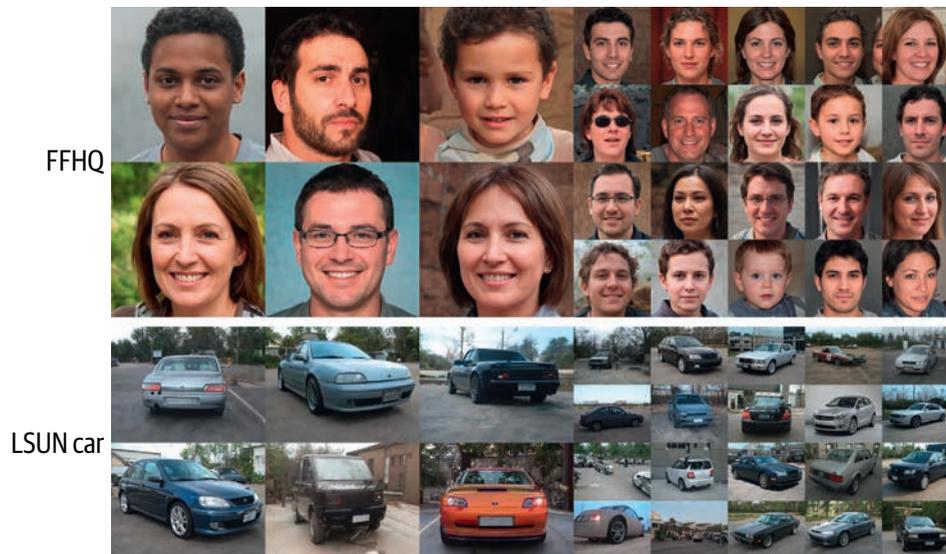


Figure 10.14 : Productions de StyleGAN2 pour le jeu de données FFHQ et LSUN (source : Karras et al., 2019).

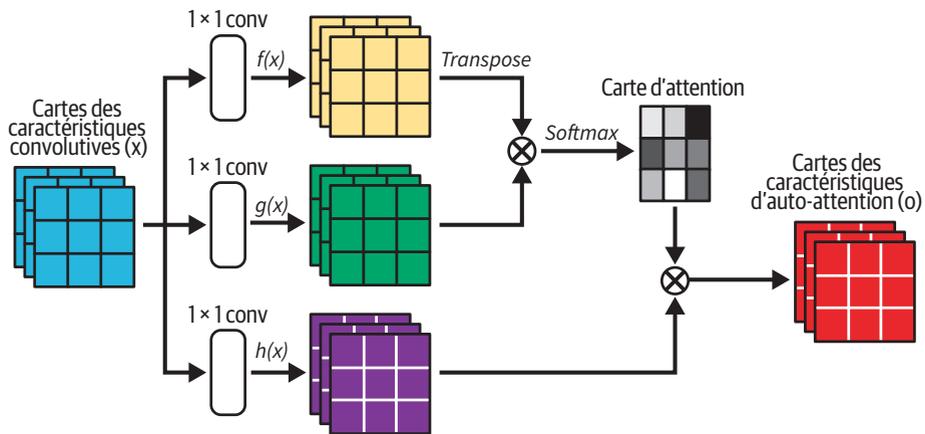


Figure 10.15 : Mécanisme d'auto-attention dans le modèle SAGAN (source : Zhang et al., 2018).

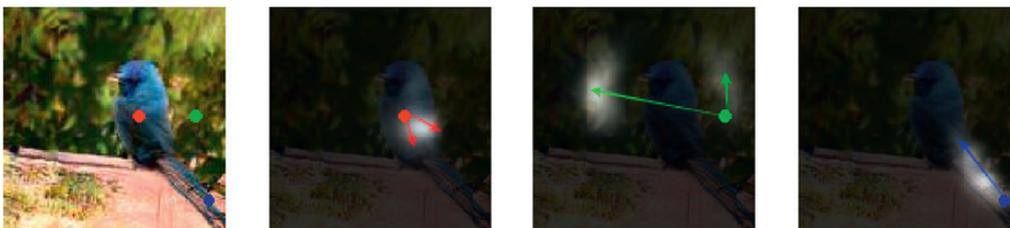


Figure 10.16 : Image d'un oiseau générée par SAGAN (case la plus à gauche) et cartes d'attention de la couche génératrice finale basée sur l'attention pour les pixels couverts par les trois points colorés (cases les plus à droite ; source : Zhang et al., 2018).

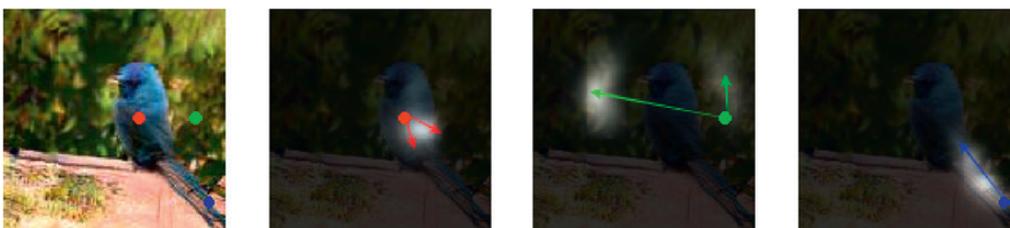


Figure 10.17 : Exemples d'images générées par BigGAN (source : Brock et al., 2018).



Figure 10.18 : Astuce de la troncature : de gauche à droite, le seuil est fixé à 2, 1, 0,5 et 0,04 (source : Brock et al., 2018).

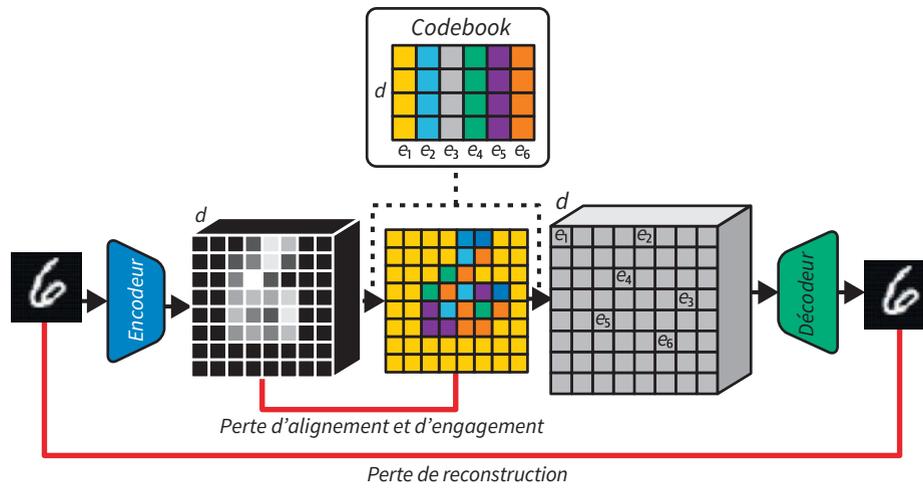


Figure 10.19 : Schéma d'un VQ-VAE.

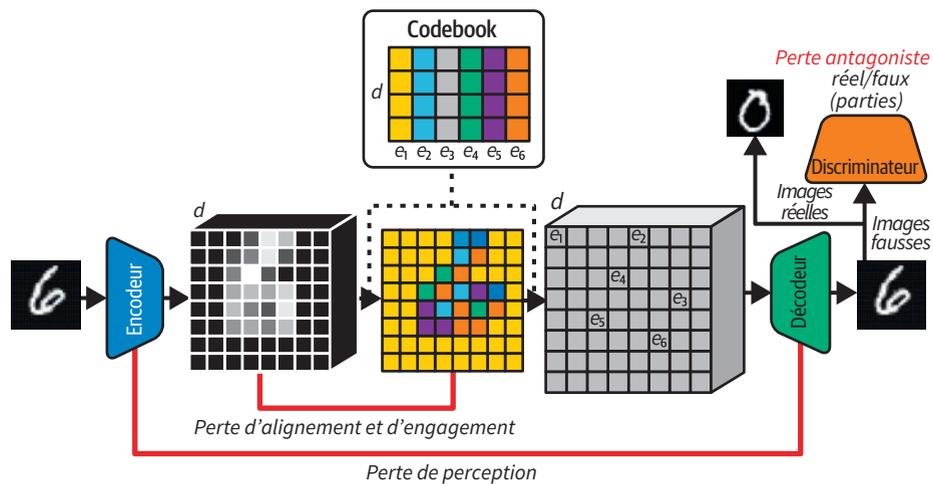


Figure 10.20 : Schéma d'un VQ-GAN : le discriminateur GAN encourage le VAE à générer des images moins floues par le biais d'un terme de perte antagoniste supplémentaire.

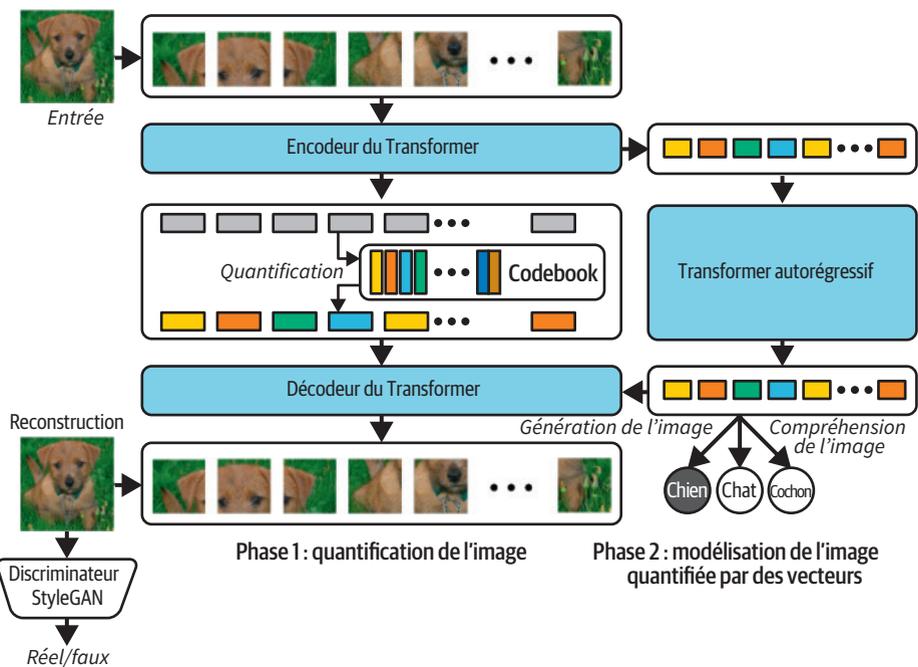


Figure 10.21 : Schéma d'un ViT VQ-GAN : le discriminateur GAN encourage le VAE à générer des images moins floues grâce à un terme de perte antagoniste supplémentaire (source : Yu et Koh, 2022).



Figure 10.22 : Exemples d'images générées par un ViT VQ-GAN entraîné sur ImageNet (source : Yu et al., 2021 ; <https://arxiv.org/abs/2110.04627>).

Chapitre 11. Génération de musique



Figure 11.1 : Ouverture de la Suite pour violoncelle n° 1 de Jean-Sébastien Bach (prélude).

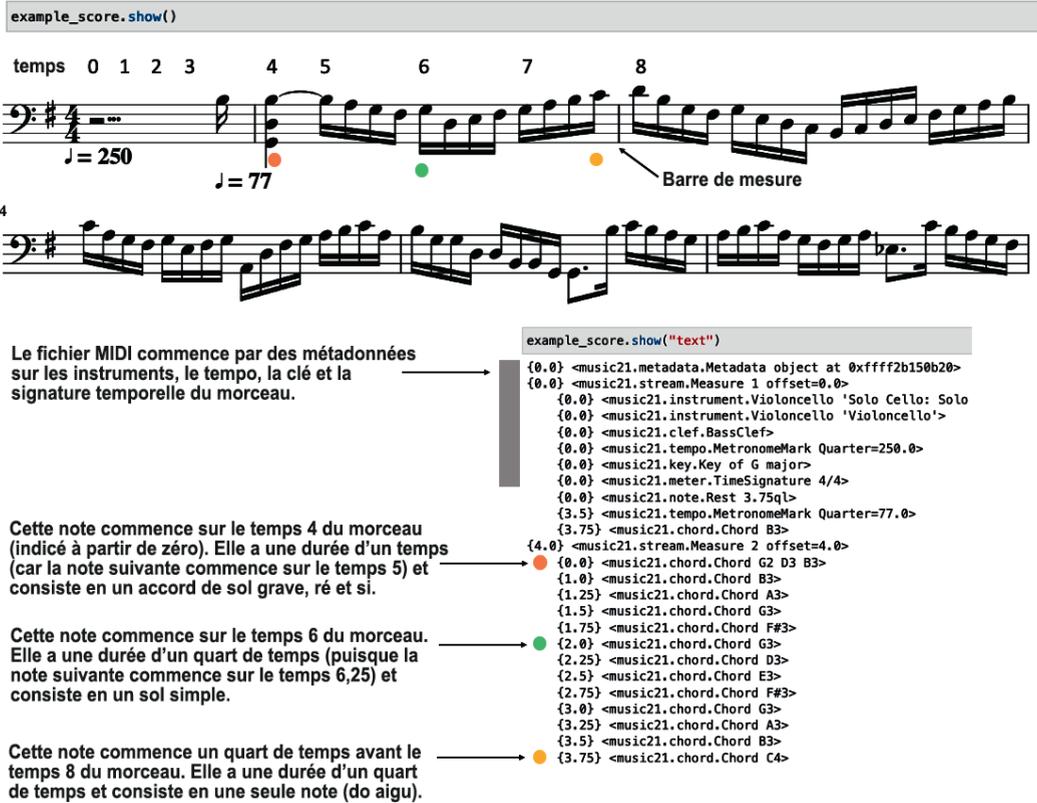


Figure 11.2 : Notation musicale.

```
Chaîne des notes
START G:major 4/4TS rest B3 B3 B3 A3 G3 F#3 G3 D3 E3 F#3 G
3 A3 B3 C4 D4 B3 G3 F#3 G3 E3 D3 C3 B2 C3 D3 E3 F#3 G3 A3 B
3 C4 A3 G3 F#3 G3 E3 F#3 G3 A2 D3 F#3 G3 A3 B3 C4 A3 B3 ...

Chaîne de la durée
0.0 0.0 0.0 3.75 0.25 1.0 0.25 0.25 0.25 0.25 0.25 0.25 0.
25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25
0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25
0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25
0.25 0.25 0.25 ...
```

Figure 11.3 : Échantillons de la chaîne des notes et de la chaîne de la durée, correspondant aux données de la Figure 11.2.

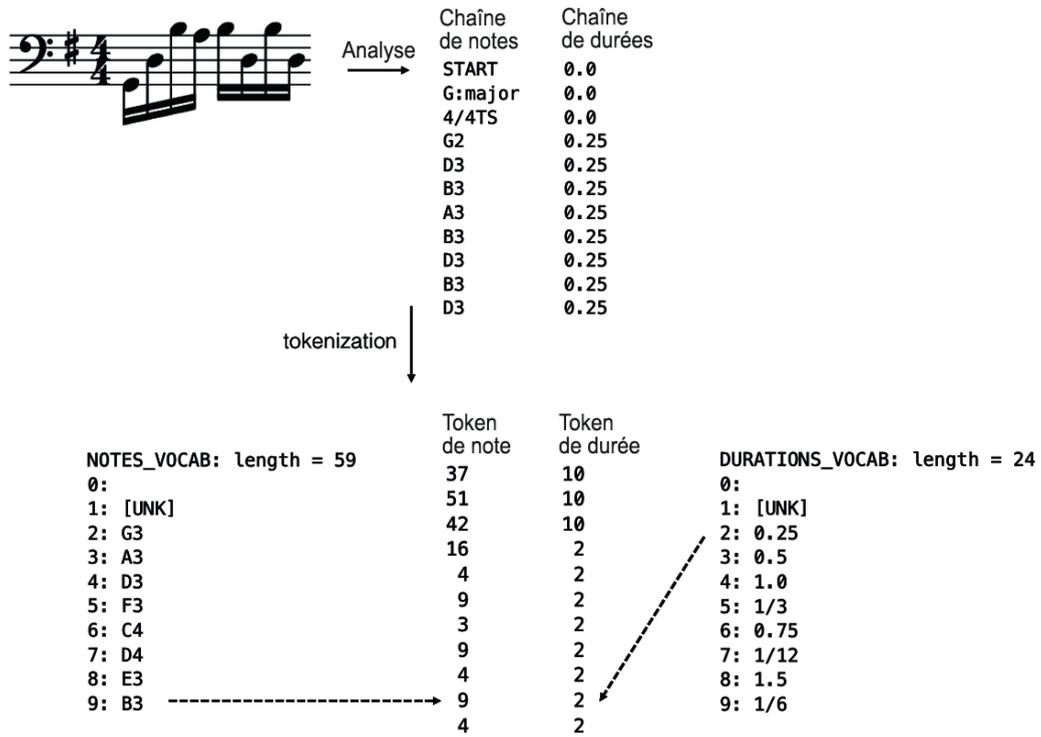


Figure 11.4 : Analyse des fichiers MIDI et tokenisation des notes et des durées.

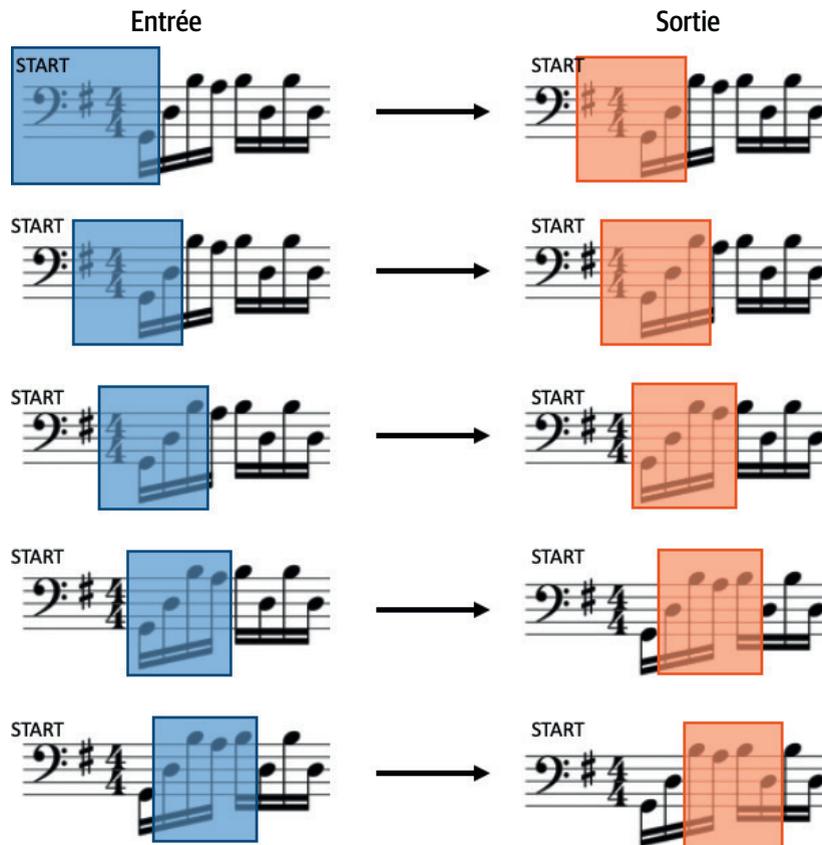


Figure 11.5 : Entrées et sorties du modèle musical Transformer ; dans cet exemple, une fenêtre glissante de largeur 4 est utilisée pour créer des fragments d'entrée, qui sont ensuite décalés d'un élément pour créer la sortie cible.

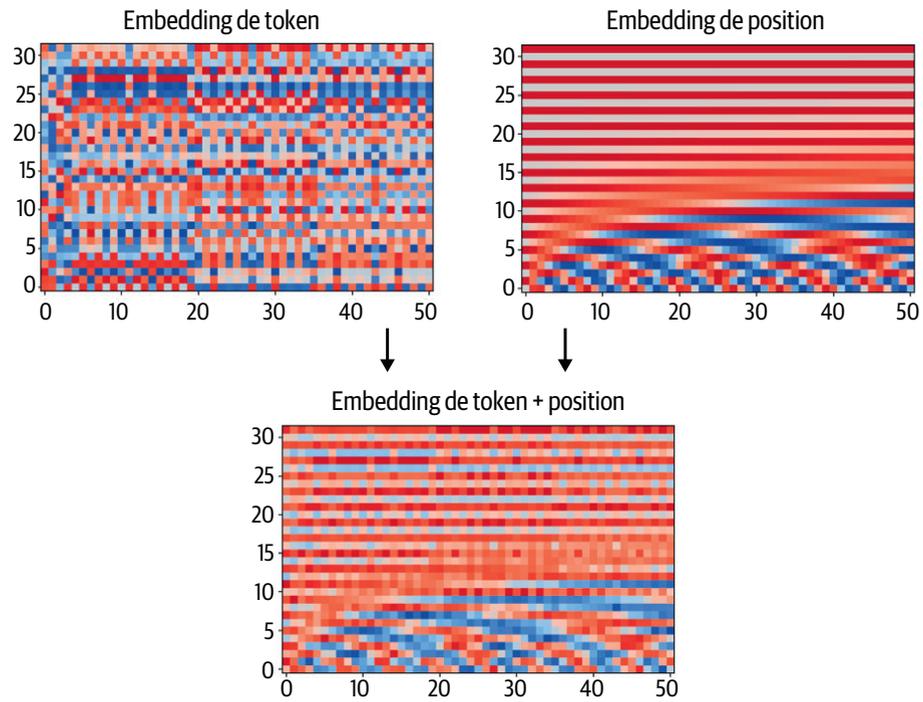


Figure 11.6 : La couche `TokenAndPositionEmbedding` ajoute les embeddings de tokens aux embeddings de positions avec un sinus afin de produire l'embedding global de la séquence.

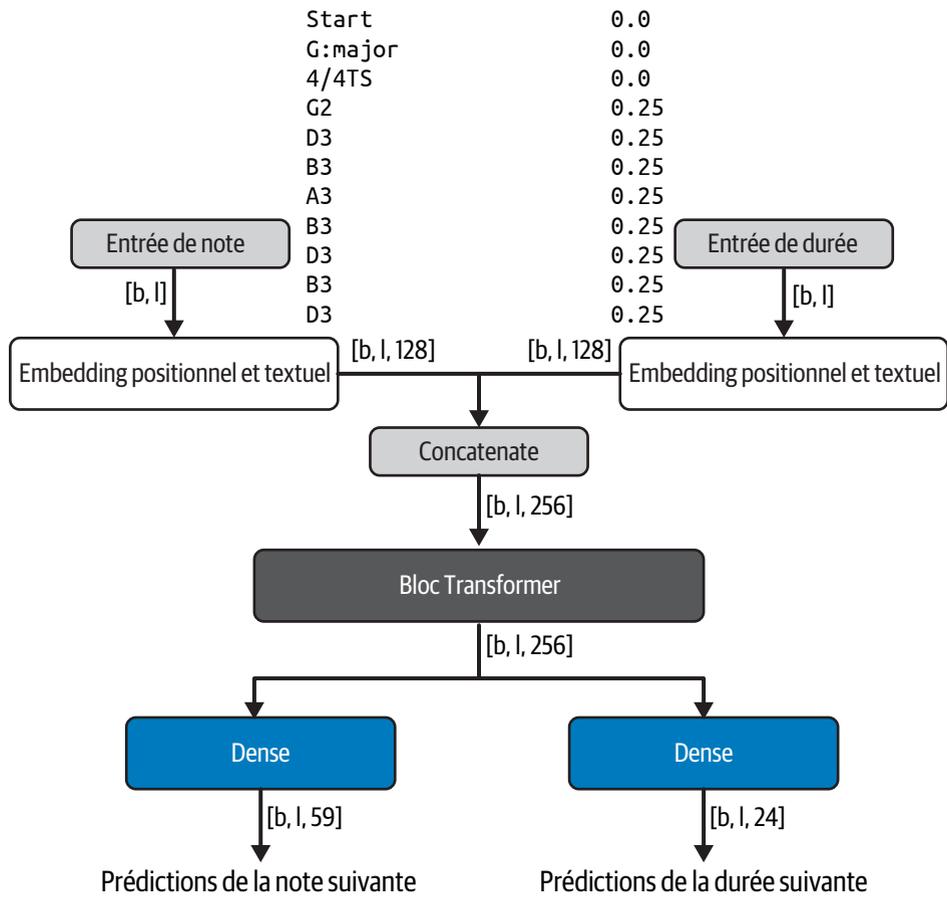


Figure 11.7 : Architecture du Transformer générateur de musique.

	Epoch 2
	Epoch 10
	Epoch 20

Figure 11.8 : Exemples de passages générés par le modèle lorsqu'il n'est alimenté que par un token de note *START* et un token de durée θ .

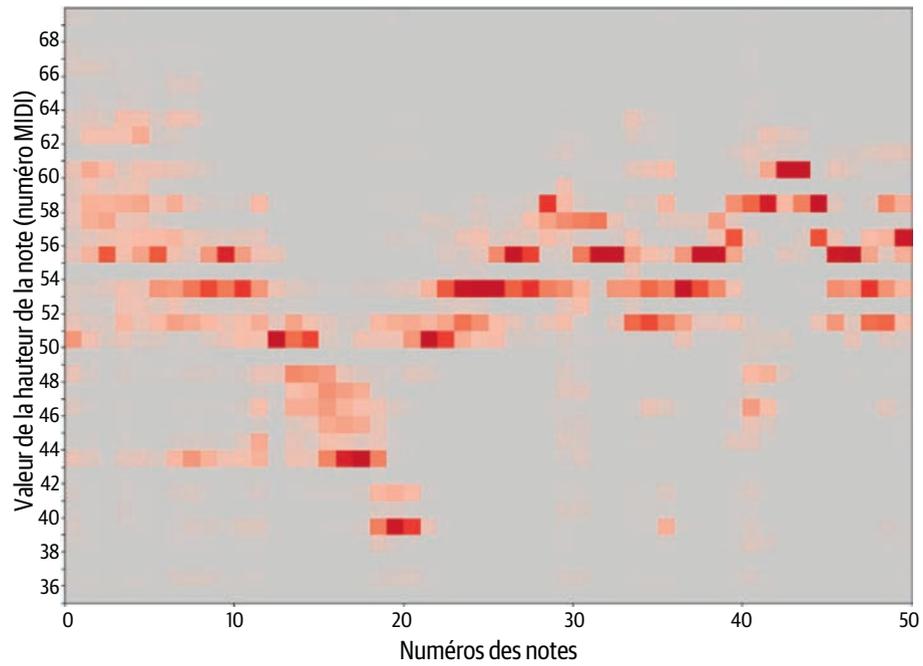
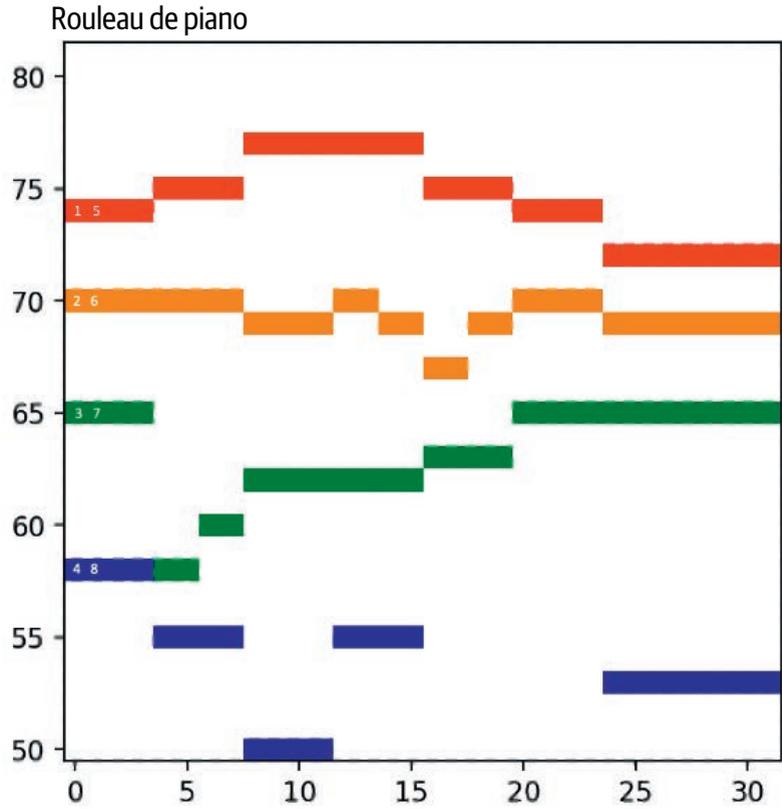


Figure 11.9 : Distribution des notes suivantes possibles dans le temps (à l'époch 20) : plus le carré est foncé, plus le modèle est certain que la note suivante est à cette hauteur.

The image displays a four-part vocal setting in G minor, 4/4 time, by J.S. Bach. The parts are labeled S (Soprano), A (Alto), T (Tenor), and B (Bass). The key signature has two flats (Bb and Eb), and the time signature is 4/4. The Soprano part begins with a half note G4, followed by quarter notes A4 and Bb4. The Alto part starts with a half note G3, followed by quarter notes A3 and Bb3. The Tenor part begins with a half note G3, followed by quarter notes A3 and Bb3. The Bass part starts with a half note G2, followed by quarter notes A2 and Bb2. The first measure of each part is a half note, and the second measure consists of two quarter notes.

Figure 11.11 : Les deux premières mesures d'un choral de J.-S. Bach.



Tokenisation de la grille

1	2	3	4	5	6	7	8	...
[74.	70.	65.	58.	74.	70.	65.	58.	74.
75.	70.	58.	55.	75.	70.	58.	55.	75.
77.	69.	62.	50.	77.	69.	62.	50.	77.
77.	70.	62.	55.	77.	70.	62.	55.	77.
75.	67.	63.	48.	75.	67.	63.	48.	75.
74.	70.	65.	46.	74.	70.	65.	46.	74.
72.	69.	65.	53.	72.	69.	65.	53.	72.
72.	69.	65.	53.	72.	69.	65.	53.	72.

Figure 11.12 : Création de la tokenisation de la grille pour les deux premières mesures d'un choral de Bach.

Tokenisation des événements



```
[NOTE_ON<74>, NOTE_ON<70>, NOTE_ON<65>, NOTE_ON<58>  
, TIME_SHIFT<1.0>  
, NOTE_OFF<74>, NOTE_OFF<65>, NOTE_OFF<58>  
, NOTE_ON<75>, NOTE_ON<58>, NOTE_ON<55>  
, TIME_SHIFT<0.5>  
, NOTE_OFF<58>, NOTE_ON<60>  
, TIME_SHIFT<0.5>  
, NOTE_OFF<75>, NOTE_OFF<70>, NOTE_OFF<60>, NOTE_OFF<55>  
, NOTE_ON<77>, NOTE_ON<69>, NOTE_ON<62>, NOTE_ON<50>  
, TIME_SHIFT<1.0>  
, NOTE_OFF<69>, NOTE_OFF<50>, NOTE_ON<70>, NOTE_ON<55>  
, TIME_SHIFT<0.5>  
, NOTE_OFF<70>, NOTE_ON<69>  
, TIME_SHIFT<0.5>  
, NOTE_OFF<77>, NOTE_OFF<69>, NOTE_OFF<62>, NOTE_OFF<55>  
, ...
```

Figure 11.13 : Tokenisation des événements pour la première mesure d'un choral de Bach.

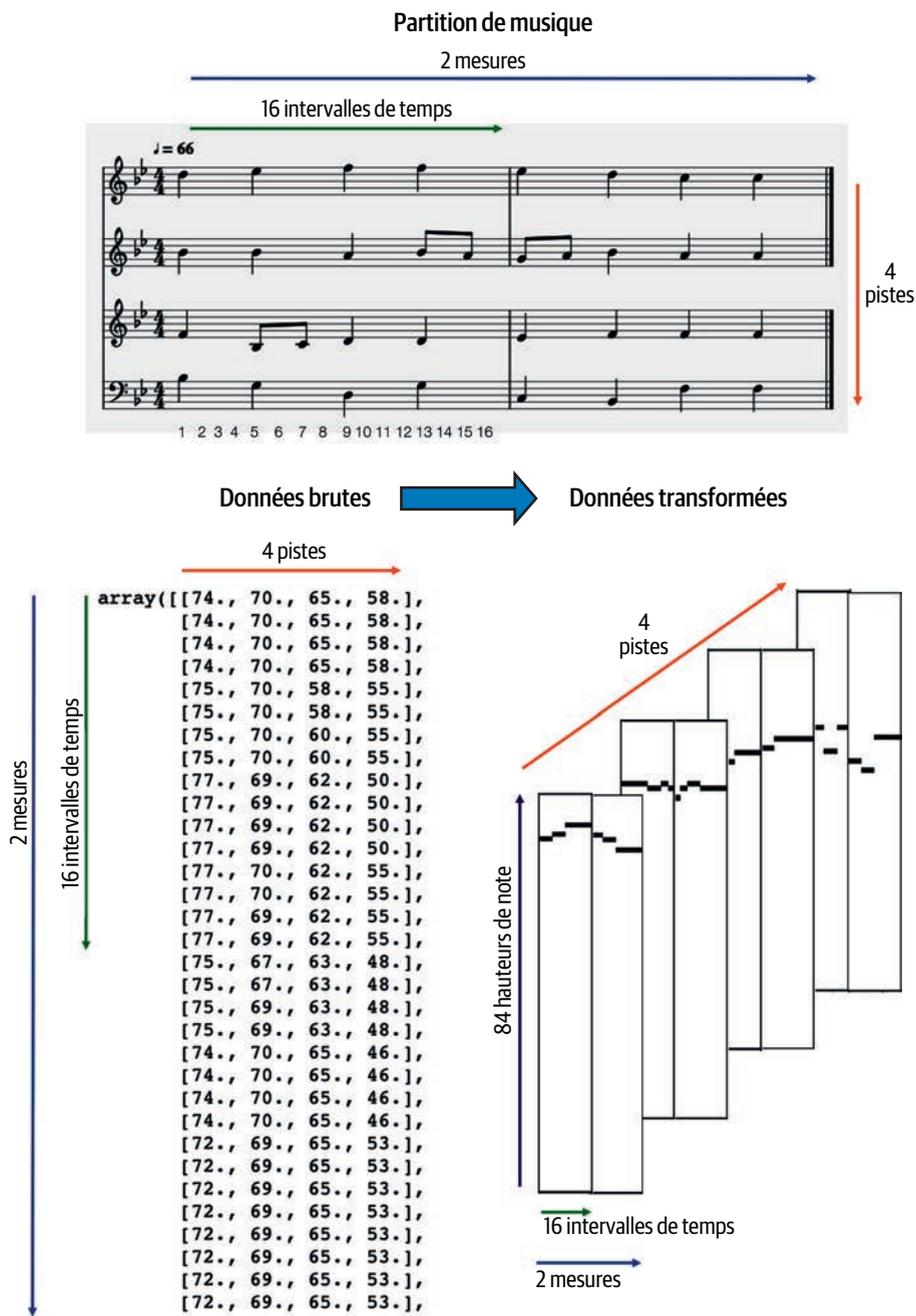


Figure 11.14 : Traitement de deux mesures de données brutes en données de rouleau de piano que nous pouvons utiliser pour entraîner le GAN.

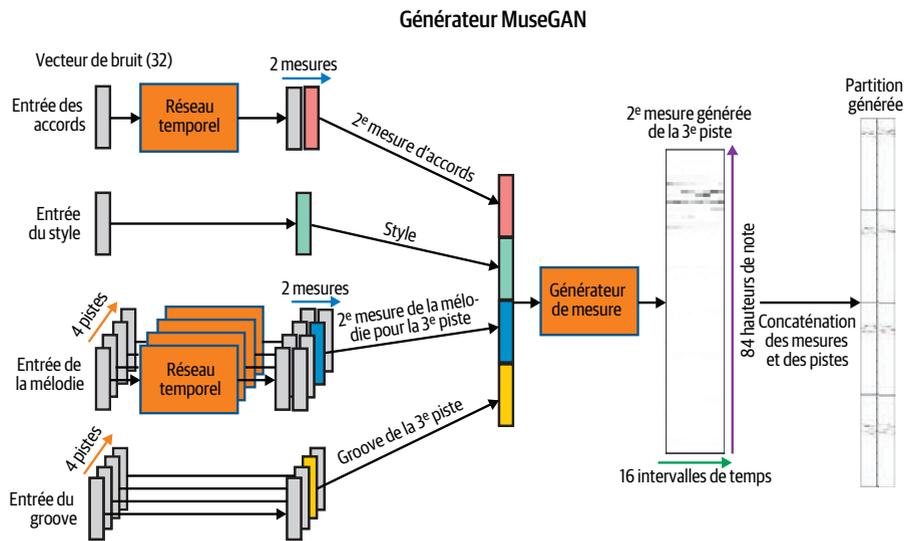


Figure 11.15 : Aperçu général du générateur MuseGAN.

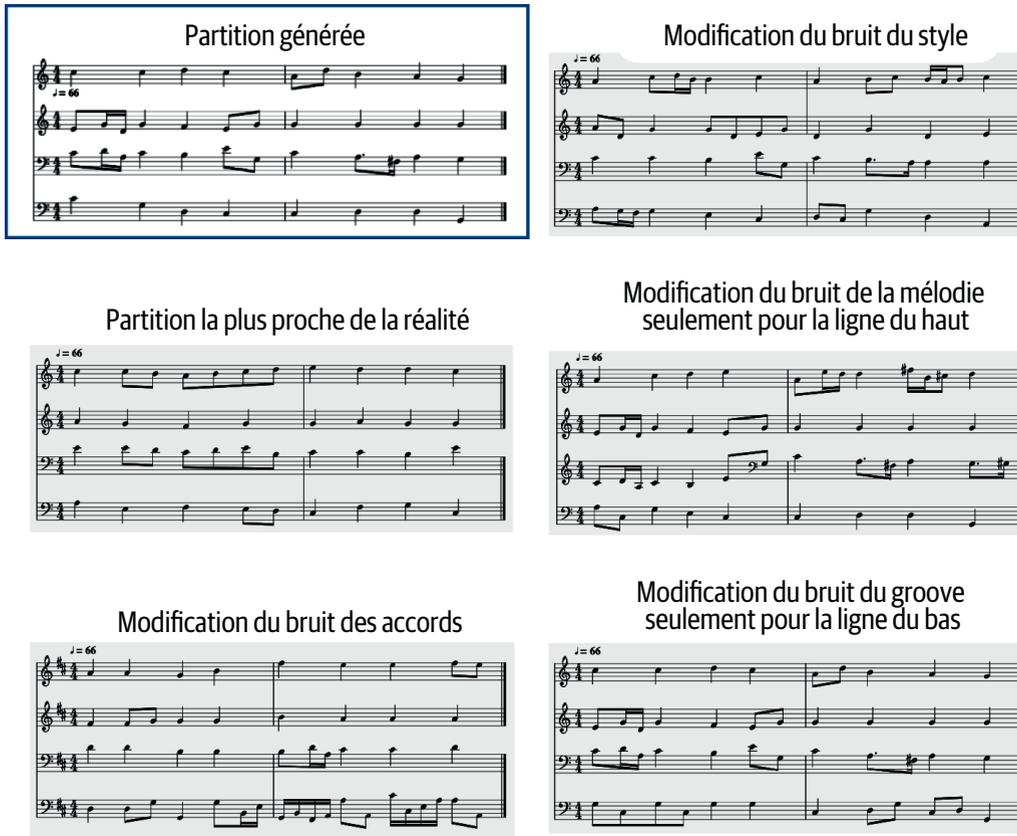


Figure 11.16 : Exemple de partition prédite par MuseGAN, montrant la partition réelle la plus proche dans les données d'entraînement et comment la partition générée est affectée par le changement du bruit d'entrée.

Chapitre 12. Modèles du monde

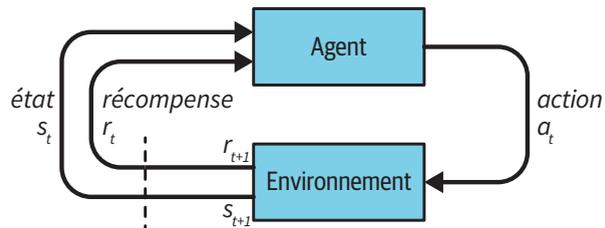


Figure 12.1 : Schéma de l'apprentissage par renforcement.

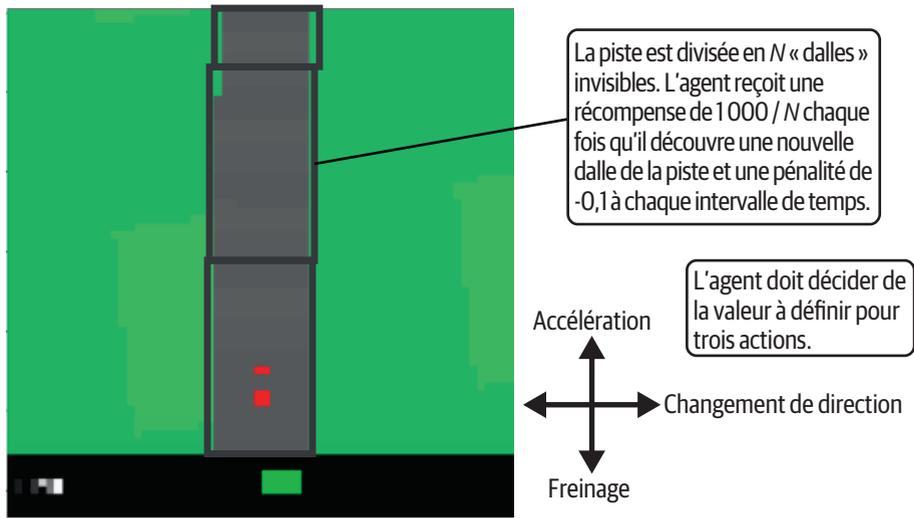


Figure 12.2 : Représentation graphique d'un état de jeu dans l'environnement *CarRacing*.

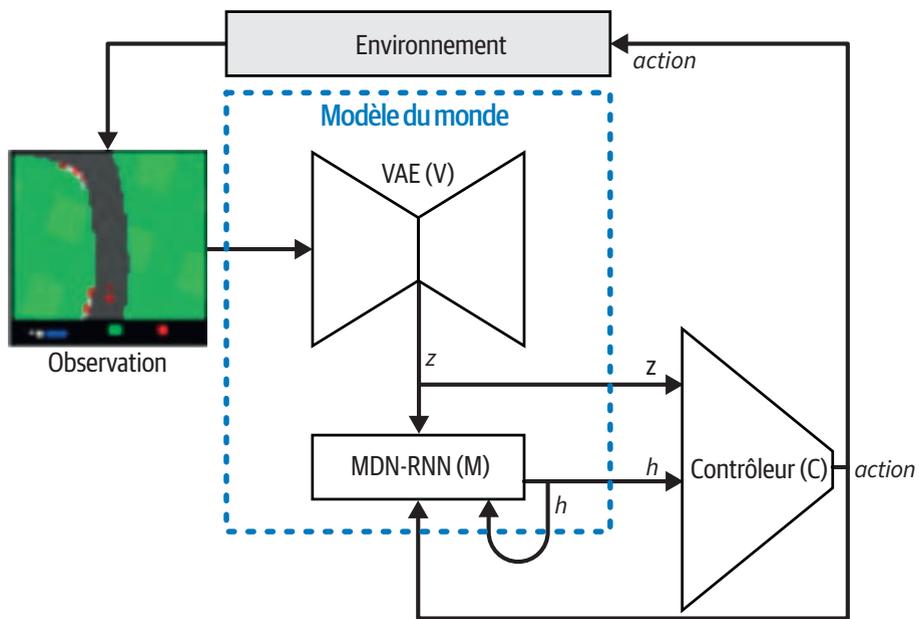


Figure 12.3 : Schéma de l'architecture d'un modèle du monde.

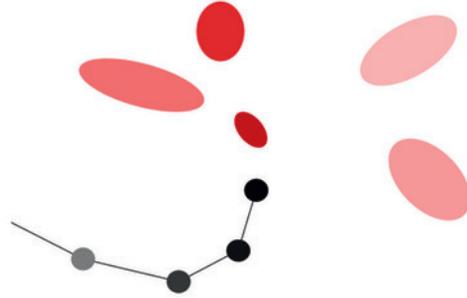


Figure 12.4 : MDN pour la génération d'écriture manuscrite (voir la figure en couleur en ligne).

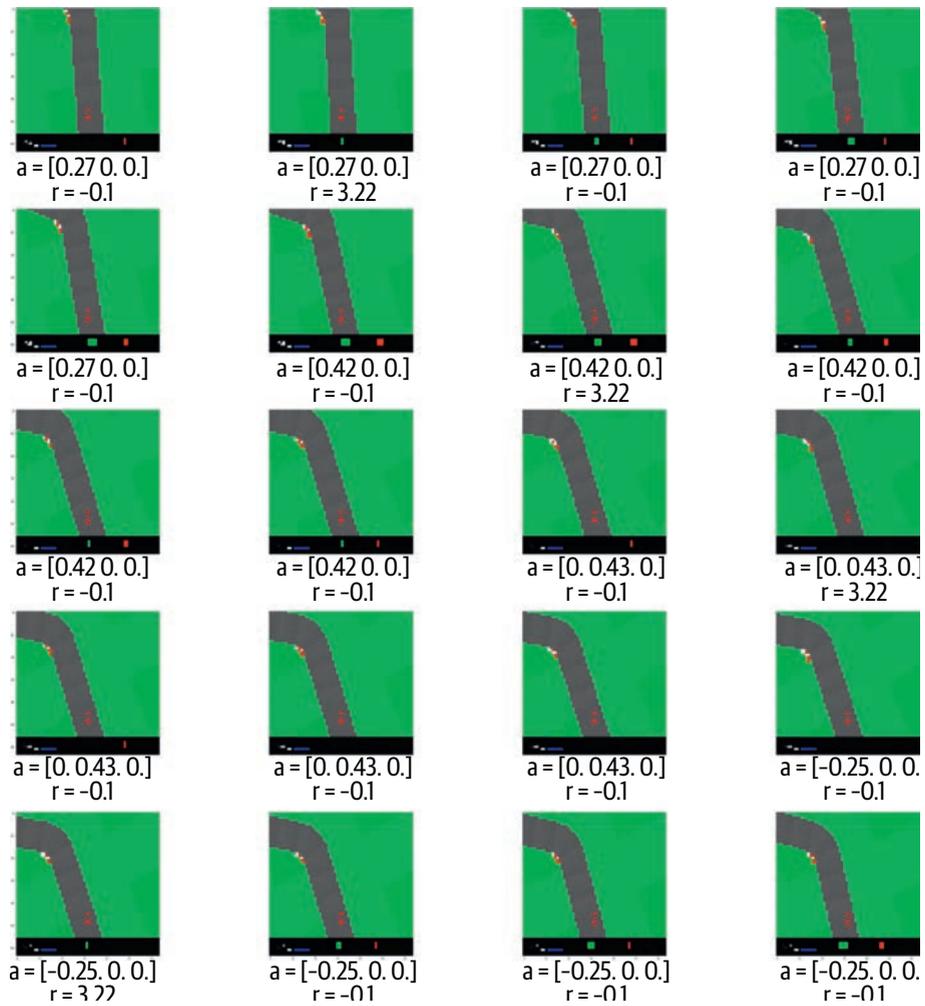


Figure 12.5 : Images 40 à 59 d'un épisode.

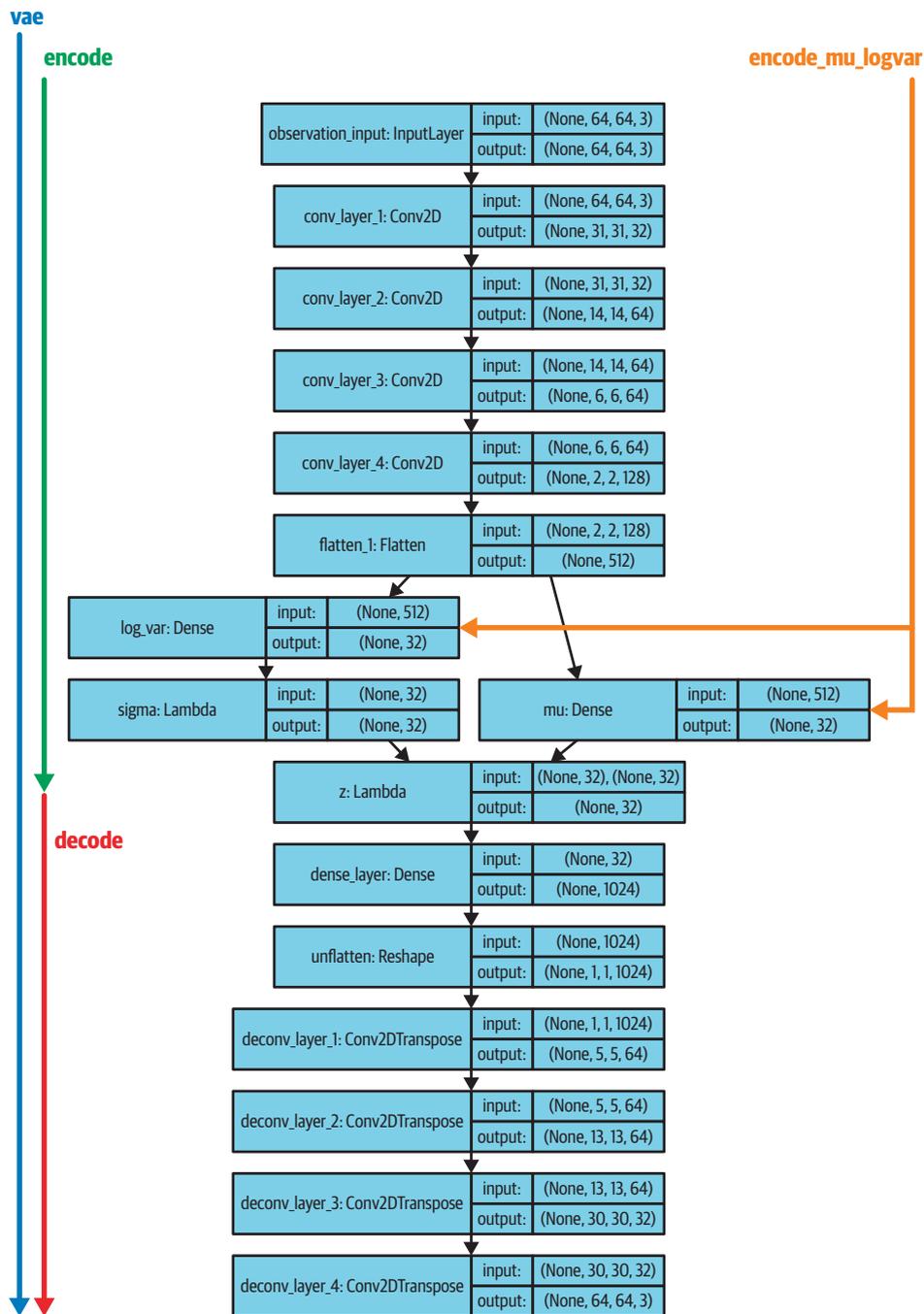


Figure 12.6 : Architecture du VAE telle qu'elle est décrite dans l'article « World Models ».

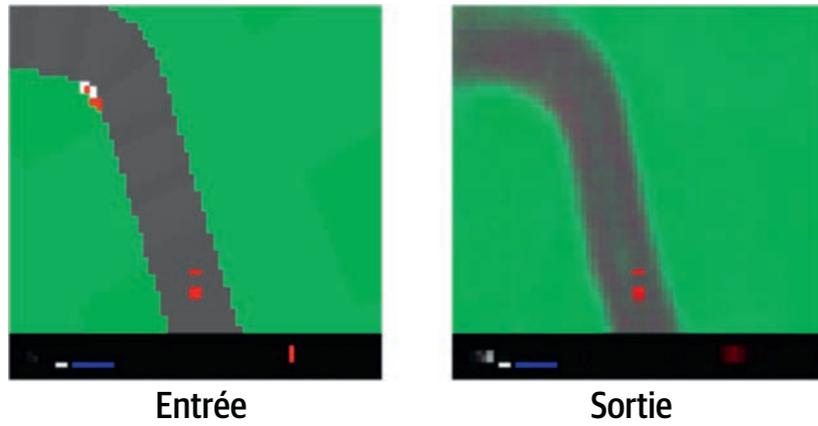


Figure 12.7 : Données d'entrée (input) et de sortie (output) du modèle VAE.

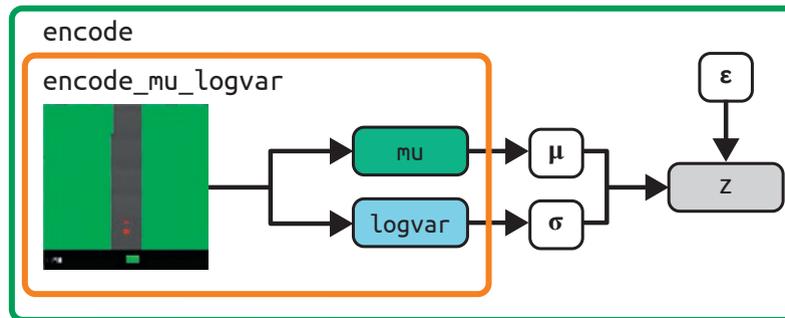


Figure 12.8 : Sortie à partir des modèles d'encodeurs.

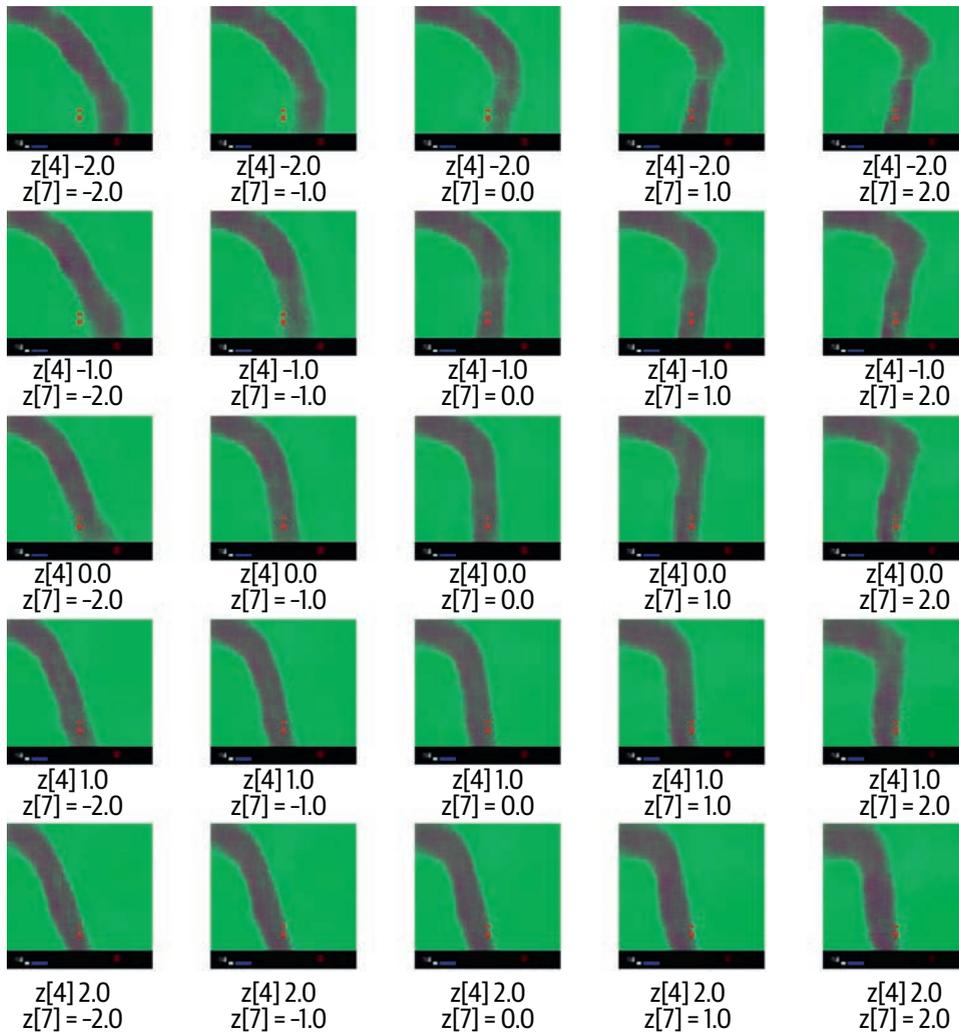


Figure 12.9 : Interpolation linéaire des deux dimensions de z .

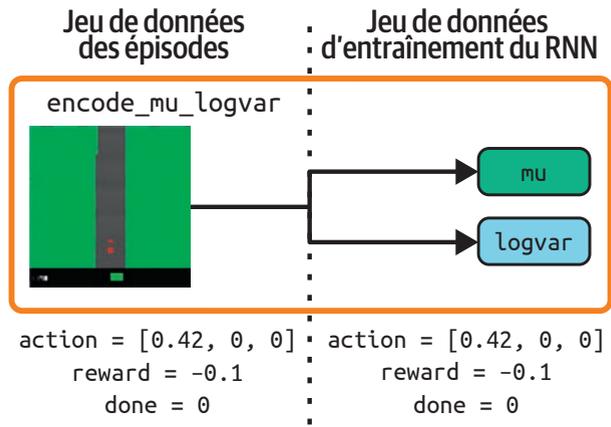


Figure 12.10 : Création du jeu de données d'entraînement pour le MDN-RNN.

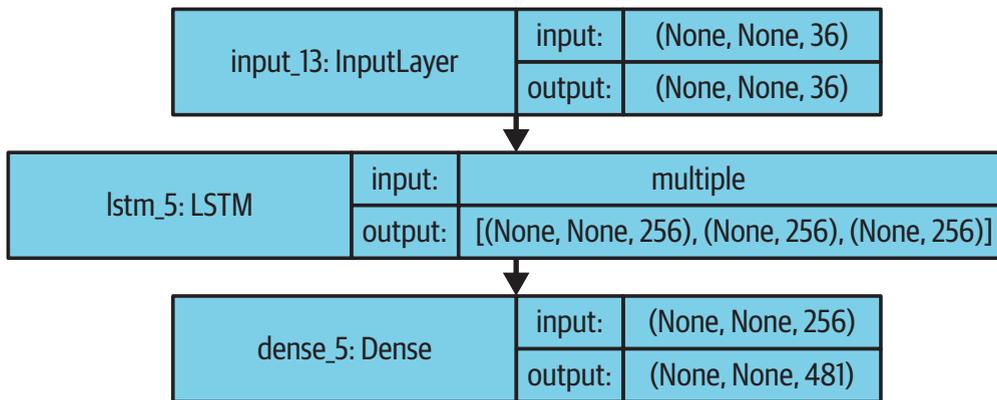


Figure 12.11 : Architecture MDN-RNN.

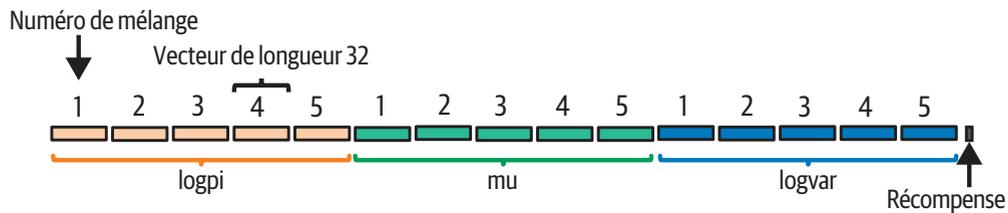


Figure 12.12 : Sortie du réseau à densité de mélange.

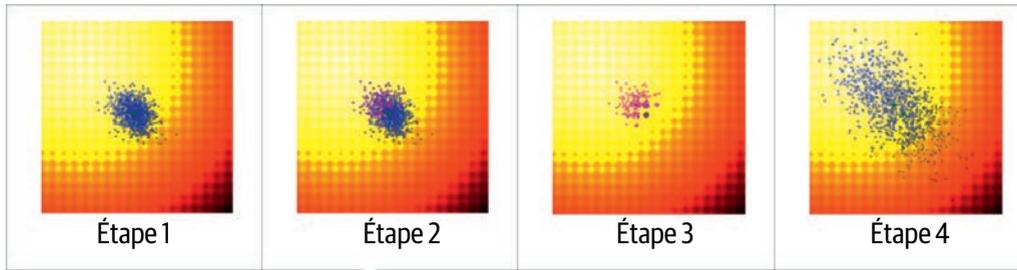


Figure 12.13 : Une étape de mise à jour de l'algorithme CMA-ES (source : Ha, 2017).

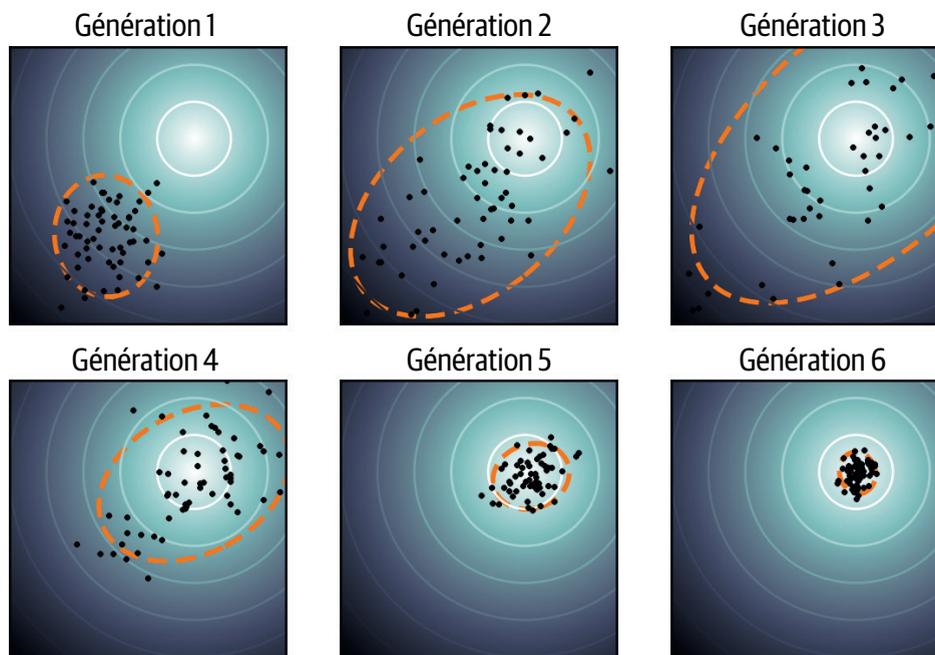


Figure 12.14 : CMA-ES (source : Wikipédia ; [https://fr.wikipedia.org/wiki/Stratégie_d'évolution#CMA-ES](https://fr.wikipedia.org/wiki/Strat%C3%A9gie_d%27%C3%A9volution#CMA-ES)).

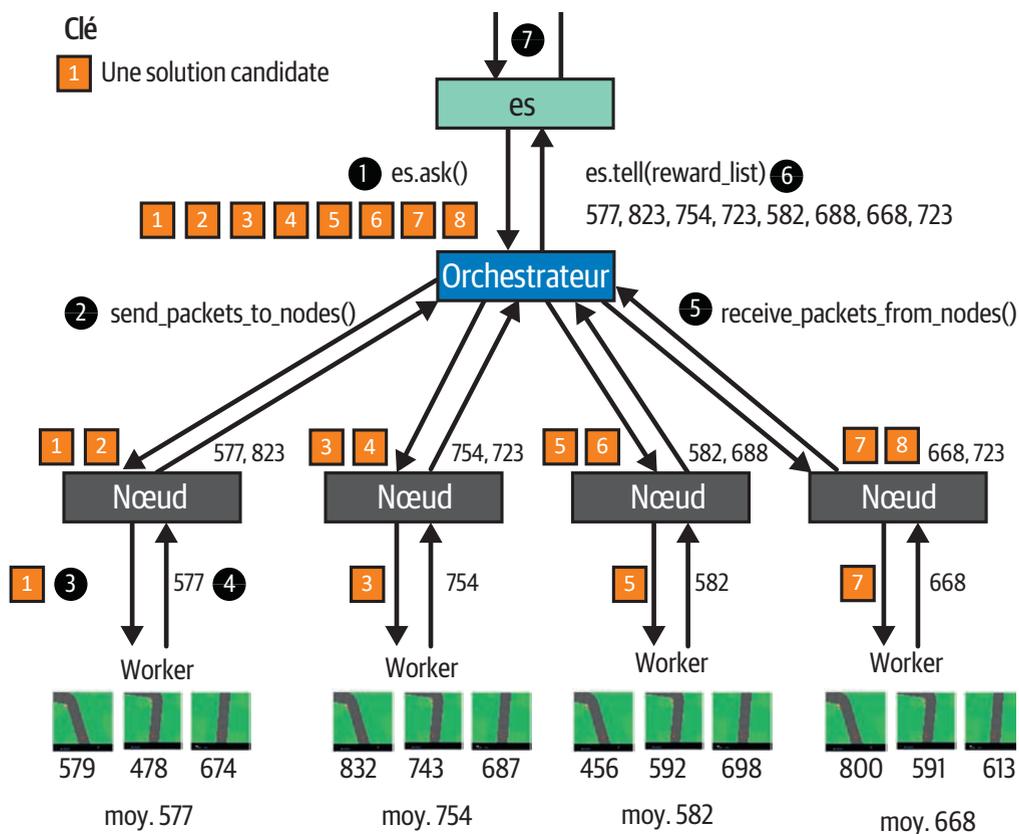


Figure 12.15 : Parallélisation de CMA-ES ; ici, la taille de la population est de huit et il y a quatre nœuds (si bien que $t = 2$, le nombre d'essais dont chaque nœud est responsable).

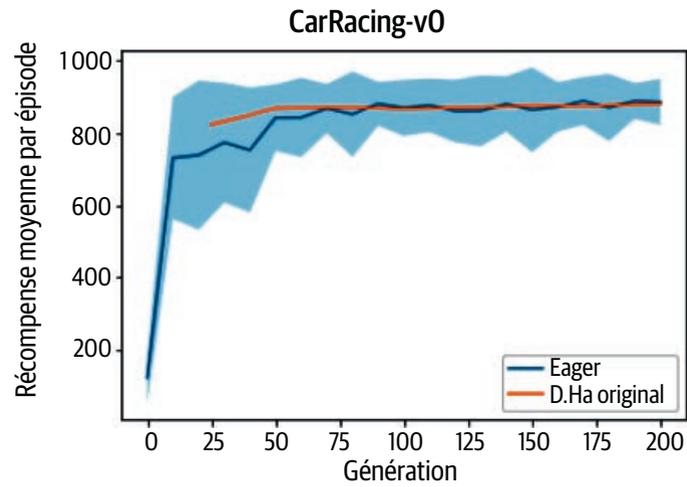


Figure 12.16 : Récompense moyenne par épisode du processus d'entraînement du contrôleur, par génération (source : Zac Wellmer, « World Models »).

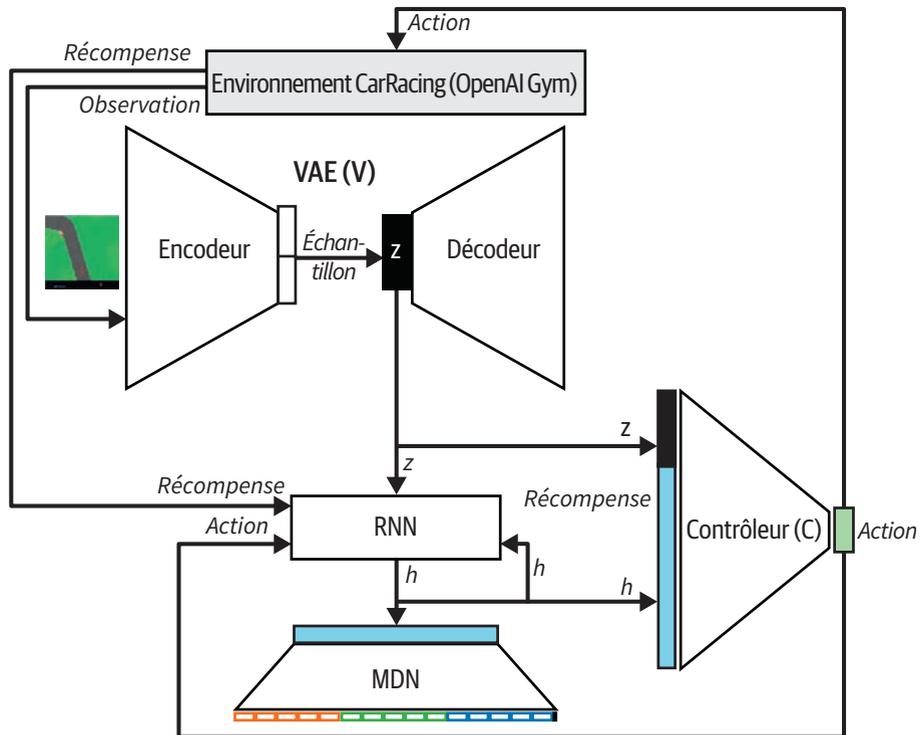


Figure 12.17 : Entraînement du contrôleur dans l'environnement Gym.

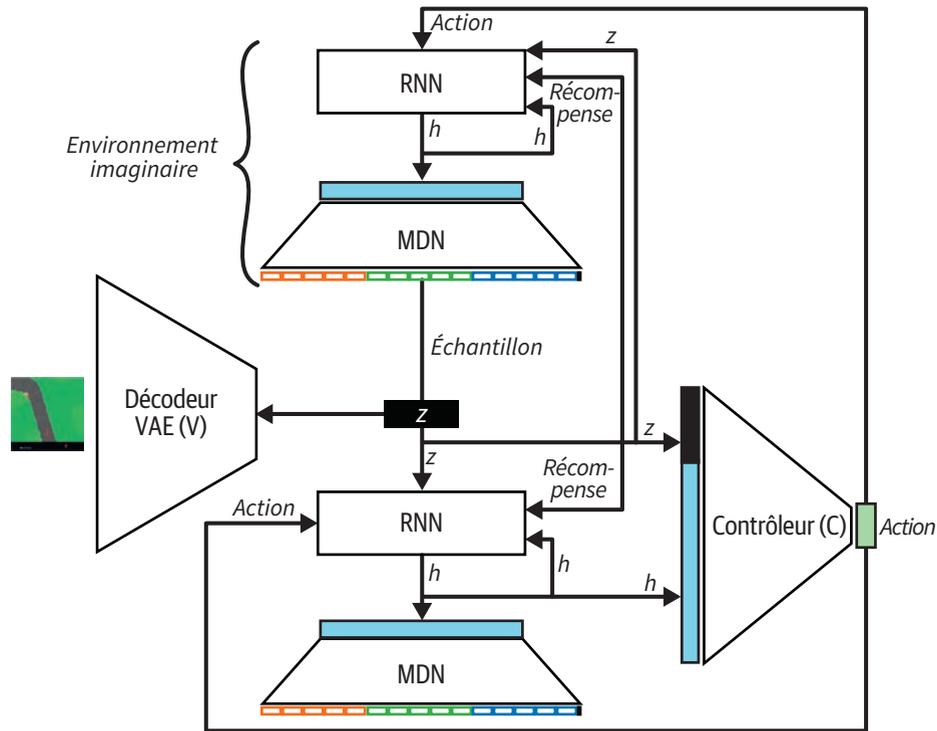


Figure 12.18 : Entraînement du contrôleur dans l'environnement imaginaire MDN-RNN.

TEMPÉRATURE τ	SCORE VIRTUEL	SCORE RÉEL
0,10	2086 \pm 140	193 \pm 58
0,50	2060 \pm 277	196 \pm 50
1,00	1145 \pm 690	868 \pm 511
1,15	918 \pm 546	1092 \pm 556
1,30	732 \pm 269	753 \pm 139
STRATÉGIE ALÉATOIRE	N/D	210 \pm 108
MEILLEUR SCORE GYM	N/D	820 \pm 58

Figure 12.19 : Utilisation de la température pour contrôler la volatilité de l'environnement imaginaire (source : Ha et Schmidhuber, 2018).

Chapitre 13. Modèles multimodaux

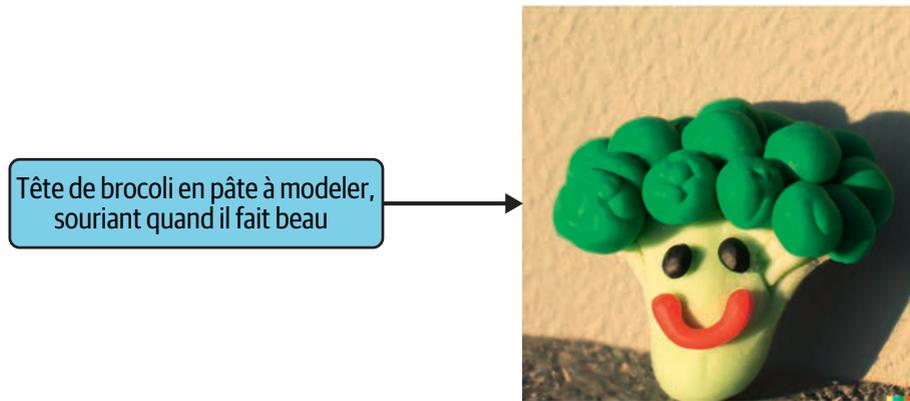


Figure 13.1 : Exemple de génération d'une image à partir d'un texte par DALL.E 2.

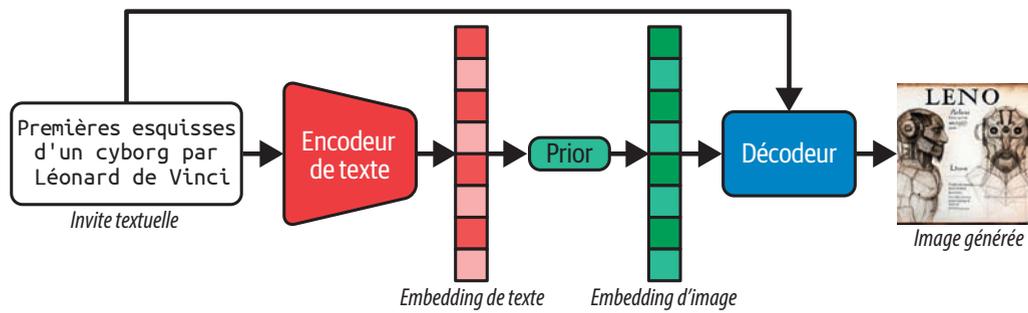


Figure 13.2 : Architecture de DALL.E 2.

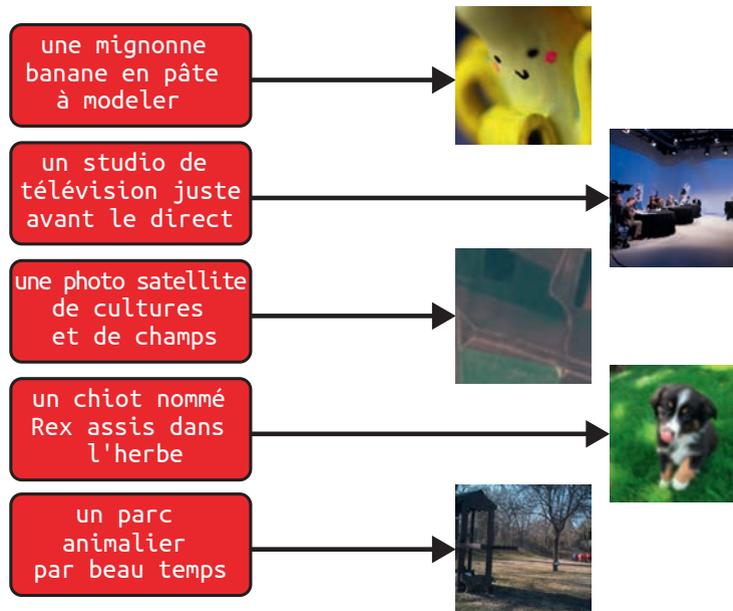


Figure 13.3 : Exemples de paires texte-image.

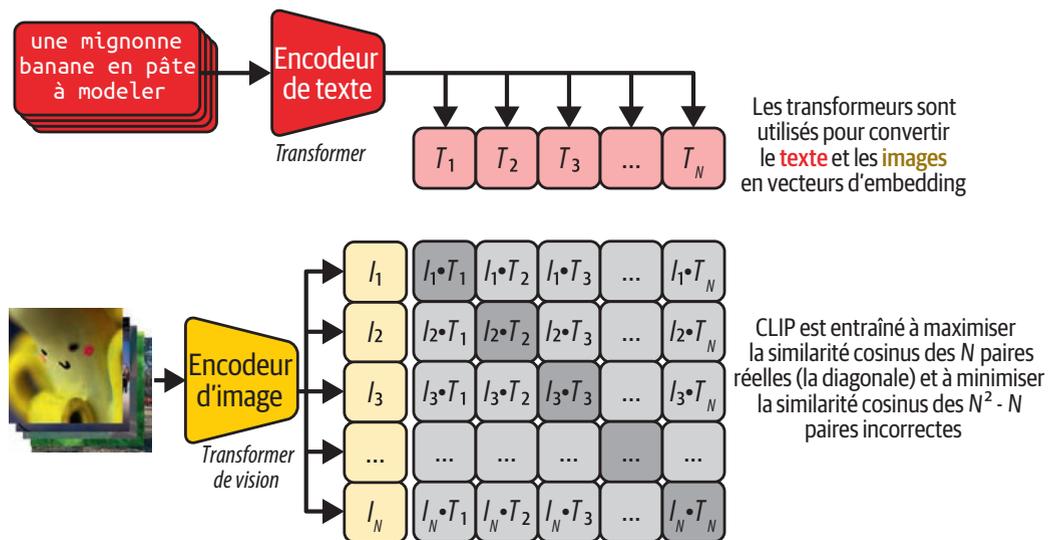


Figure 13.4 : Processus d'entraînement CLIP.

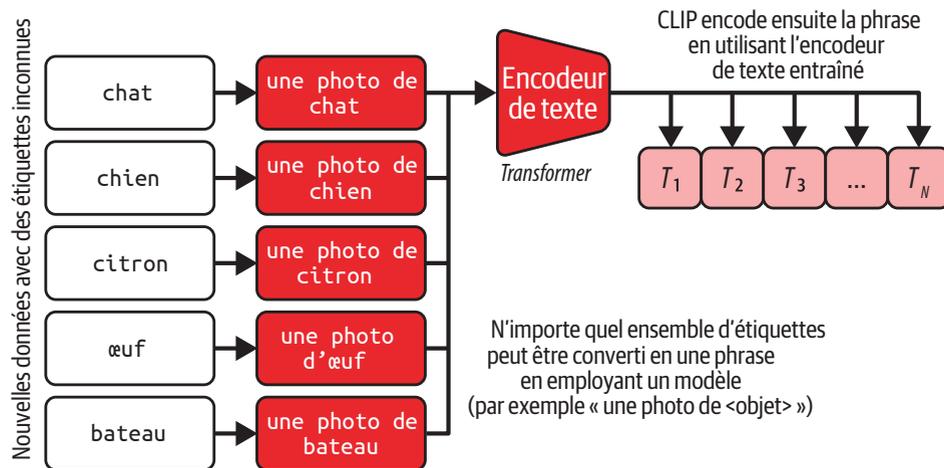


Figure 13.5 : Conversion d'étiquettes en un nouveau jeu de données de légendes, afin de produire des embeddings de texte CLIP.

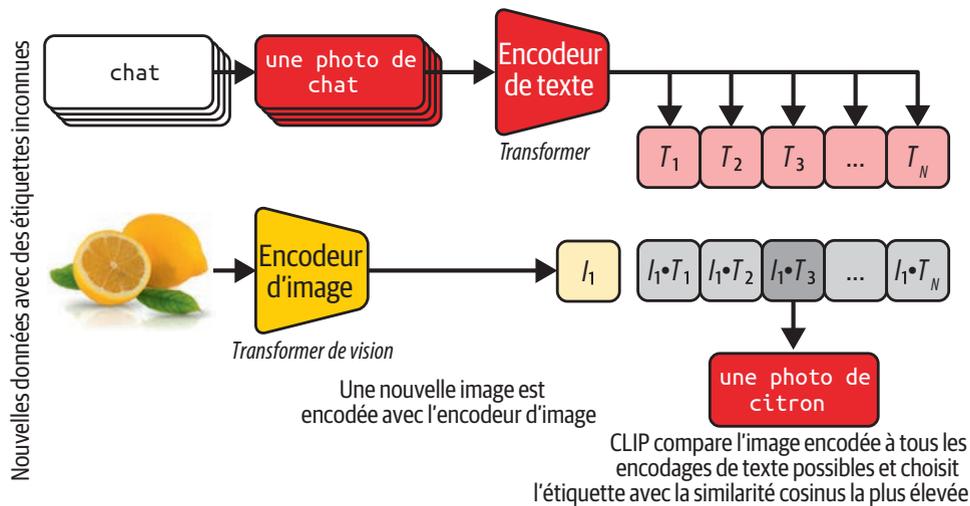


Figure 13.6 : Utilisation de CLIP pour prédire le contenu d'une image.

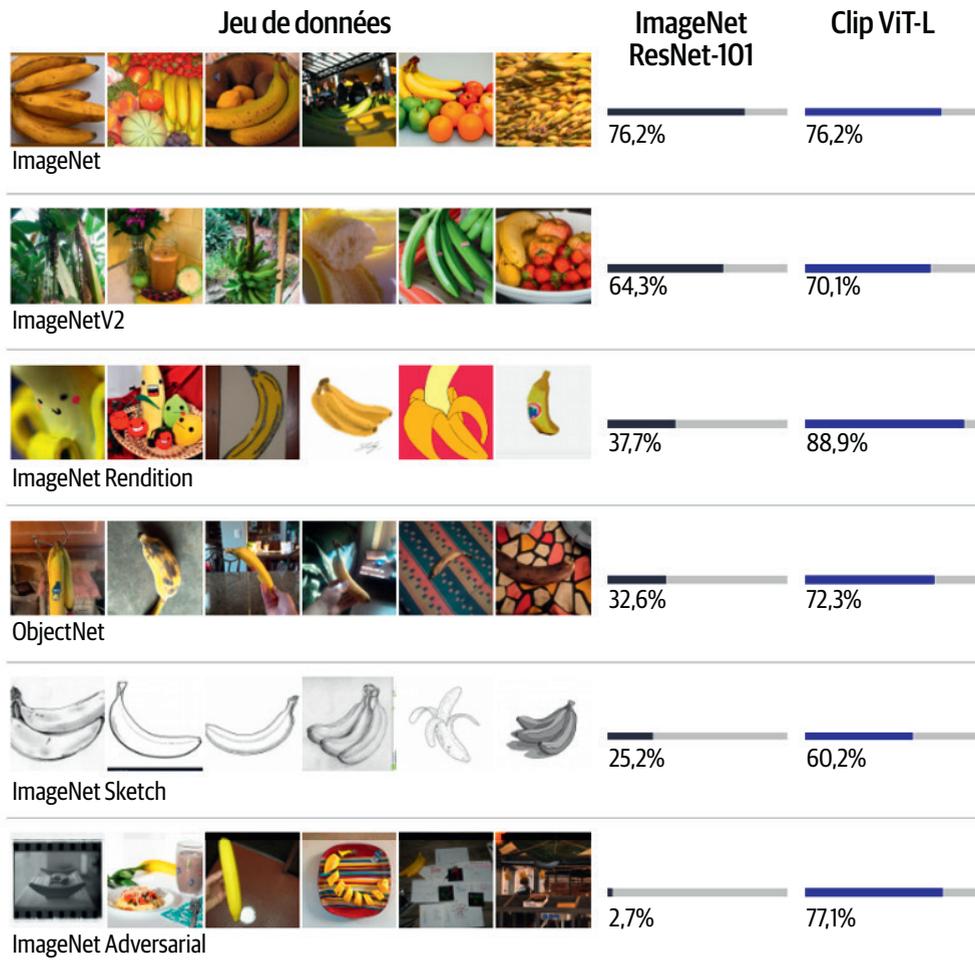


Figure 13.7 : CLIP donne de bons résultats sur un large éventail de jeux de données d'étiquetage d'images (source : Radford et al., 2021 ; <https://arxiv.org/abs/2103.00020>).

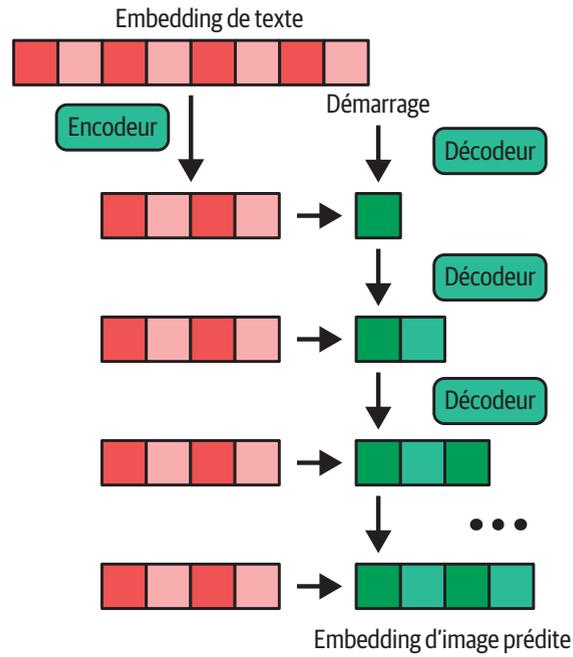


Figure 13.8 : Schéma simplifié du prior autorégressif de DALL.E 2.

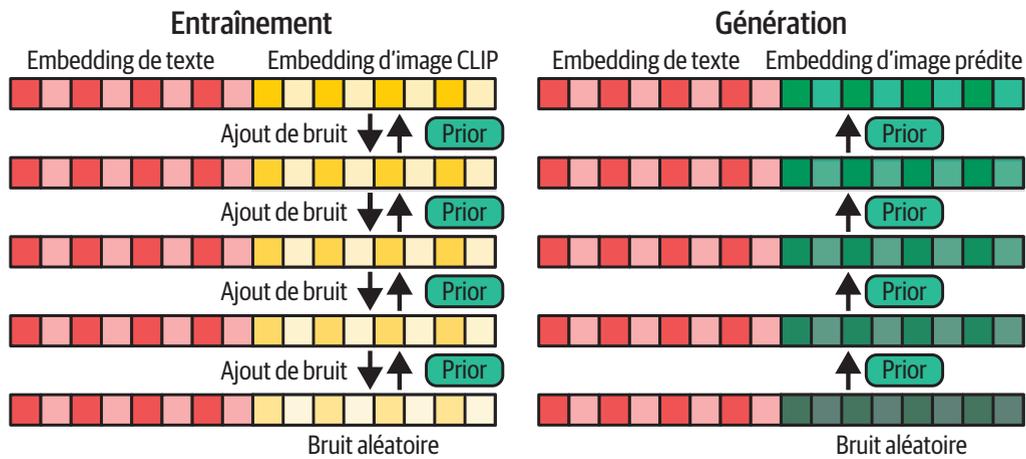


Figure 13.9 : Schéma simplifié du processus de génération et d'entraînement du prior de diffusion de DALL.E 2.

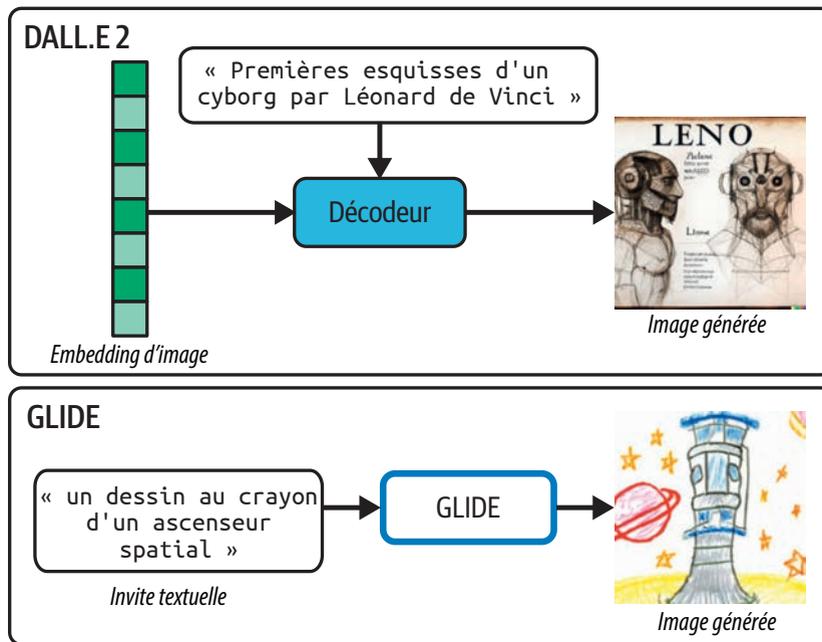


Figure 13.10 : Comparaison entre DALL.E 2 et GLIDE ; GLIDE entraîne le modèle génératif complet en partant de zéro, tandis que DALL.E 2 utilise les embeddings CLIP pour reporter l'information de l'invite textuelle initiale.

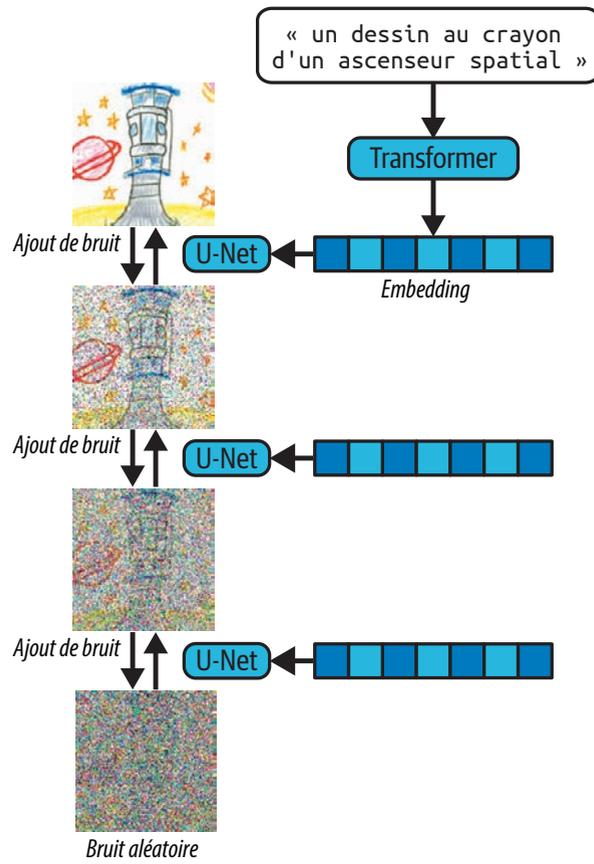


Figure 13.11 : Processus de diffusion GLIDE.

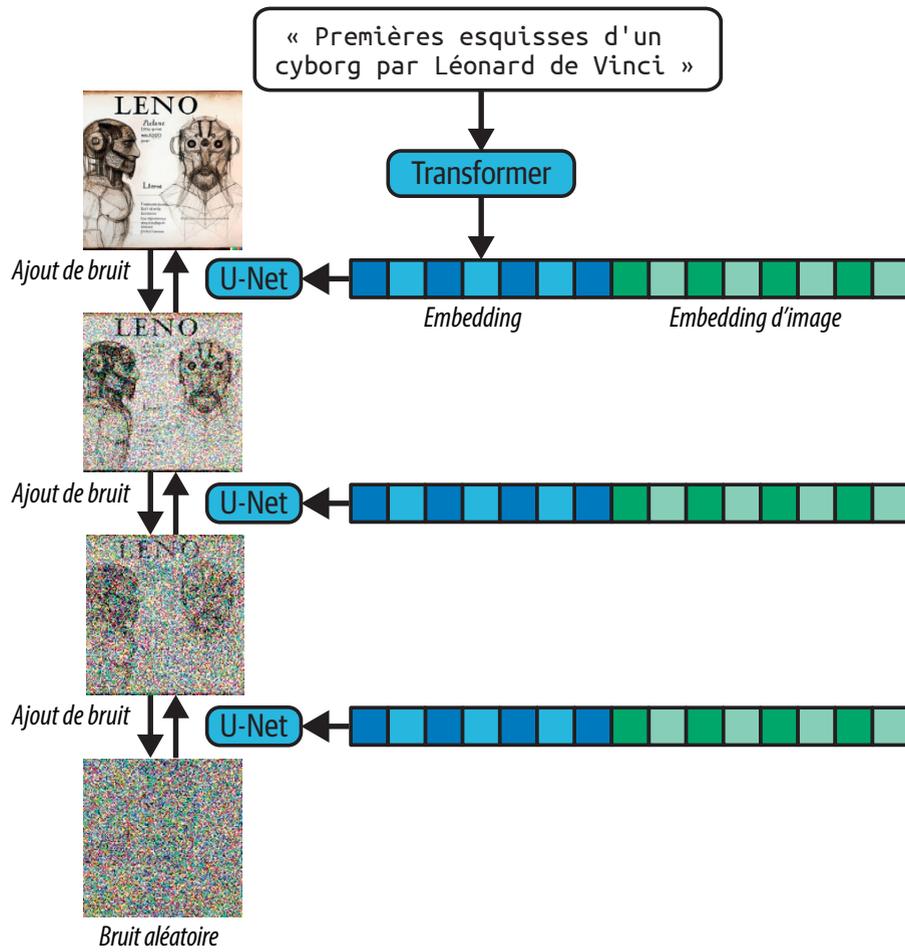


Figure 13.12 : Le décodeur DALL.E 2 s'appuie en outre sur l'embedding d'image produite par le prior.

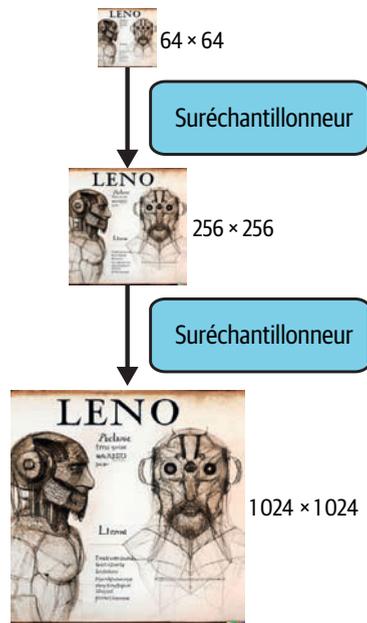


Figure 13.13 : Le premier modèle de diffusion avec un suréchantillonneur convertit l'image 64×64 pixels en 256×256 pixels, tandis que le second convertit l'image 256×256 pixels en $1\,024 \times 1\,024$ pixels.

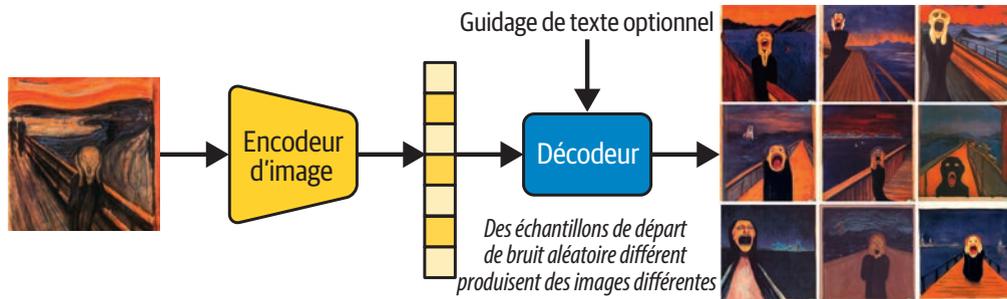


Figure 13.14 : DALL.E 2 peut être utilisé pour générer des variantes d'une image donnée.

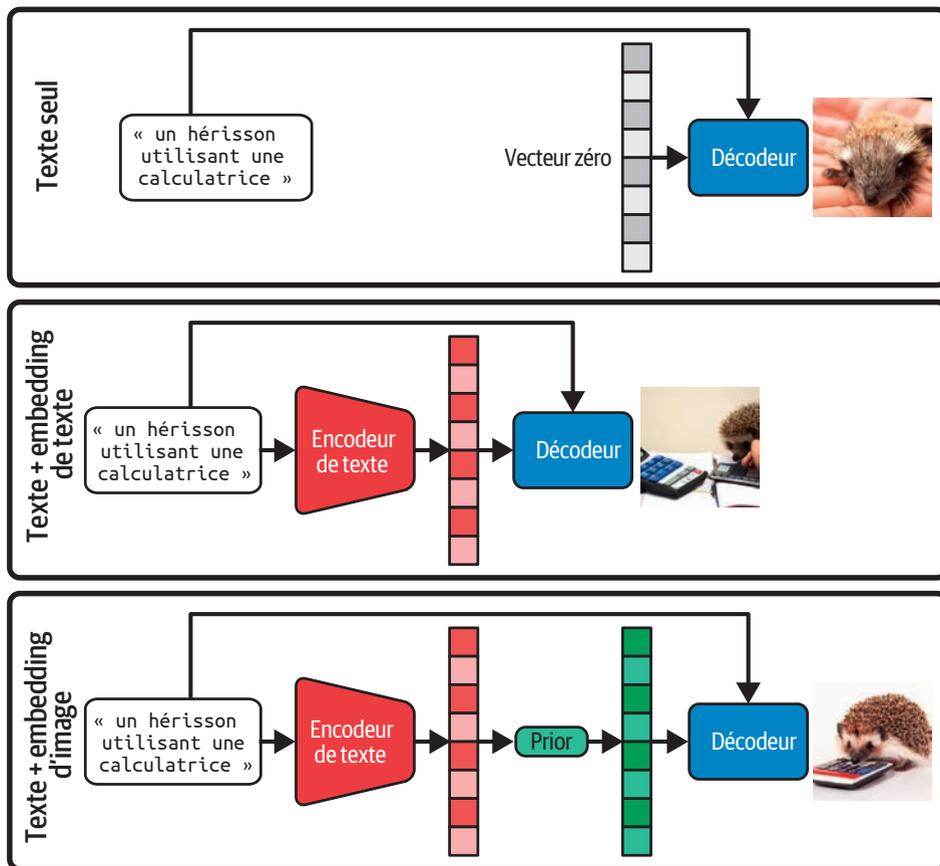


Figure 13.15 : Le prior fournit au modèle un contexte supplémentaire qui aide le décodeur à produire des générations plus précises (source : Ramesh et al., 2022 ; <https://arxiv.org/pdf/2204.06125.pdf>).

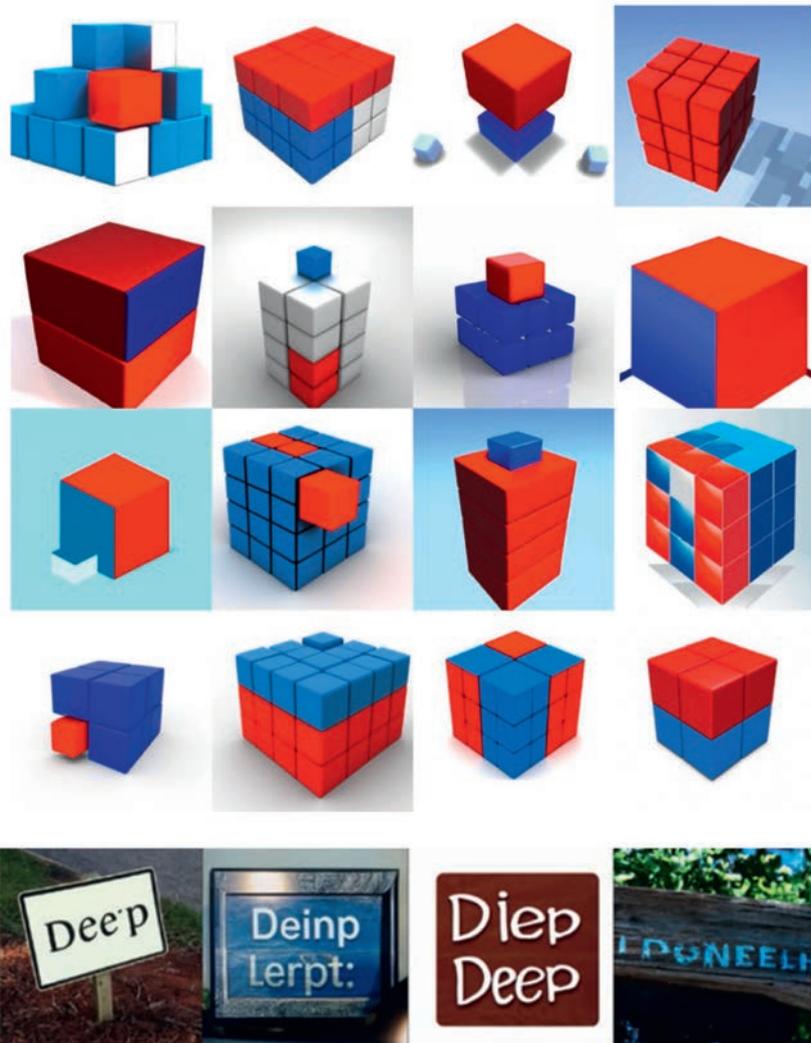


Figure 13.16 : Deux limites de DALL.E 2 résident dans sa capacité à lier des attributs à des objets et à reproduire des informations textuelles. Les cubes ont été générés par l'invite suivante : « Un cube rouge au-dessus d'un cube bleu » ; les panneaux ont été générés par l'invite suivante : « Un panneau indiquant deep learning » (source : Ramesh et al., 2022 ; <https://arxiv.org/pdf/2204.06125.pdf>).

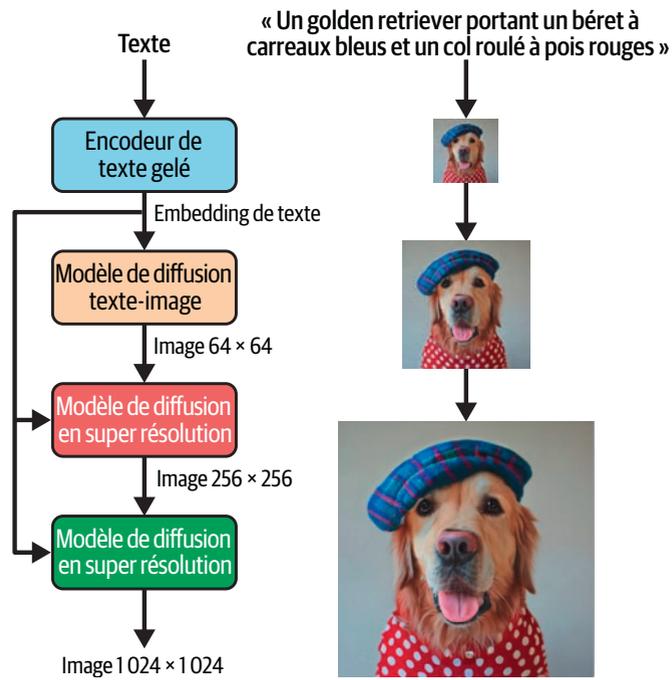


Figure 13.17 : Architecture d'Imagen (source : Saharia et al., 2022 ; <https://arxiv.org/abs/2205.11487>).

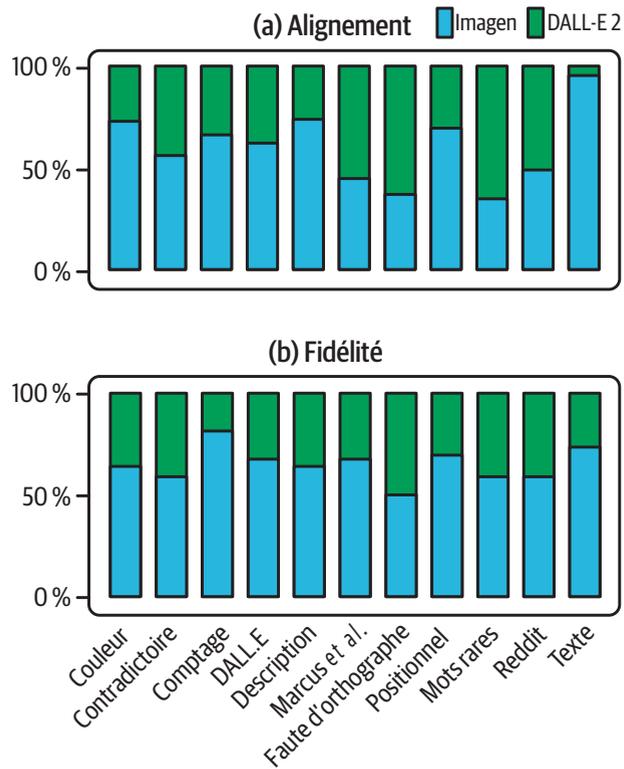


Figure 13.18 : Comparaison d'Imagen et de DALL.E 2 avec DrawBench. L'alignement et la fidélité de l'image sont mesurés (source : Saharia et al., 2022 ; <https://arxiv.org/pdf/2205.11487.pdf>). L'Annexe C de l'article décrit les 11 catégories de DrawBench.



Figure 13.19 : Exemple de générations d'images (source : Saharia et al., 2022).

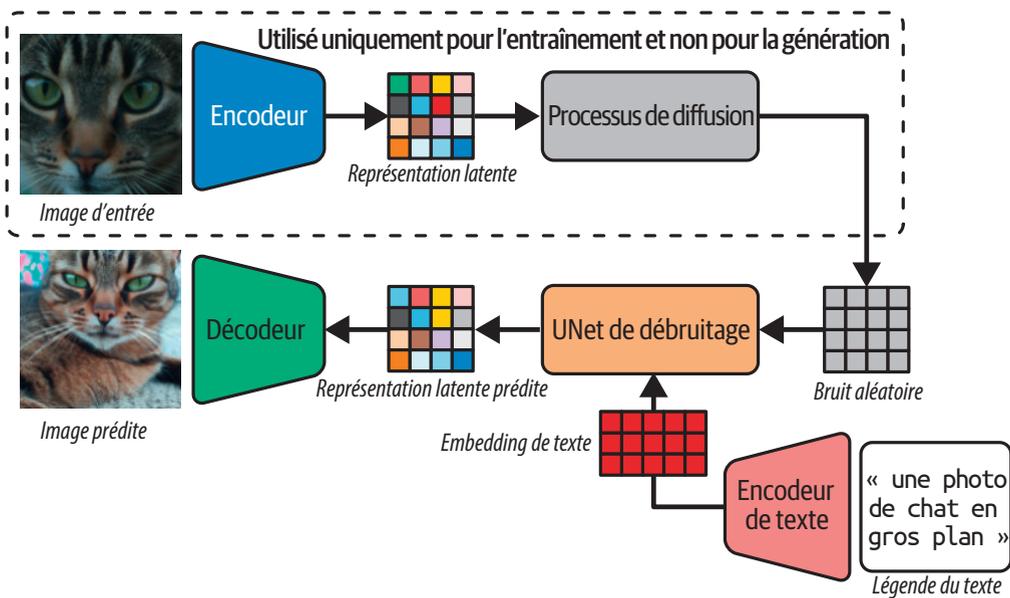


Figure 13.20 : Architecture de Stable Diffusion.



« un robot insecte préparant un repas délicieux »



« une cité utopique solaire high-tech dans la forêt amazonienne »



« un petit chalet au sommet d'une montagne enneigée dans le style des studios Disney »

Figure 13.21 : Exemples de sorties de Stable Diffusion 2.1.

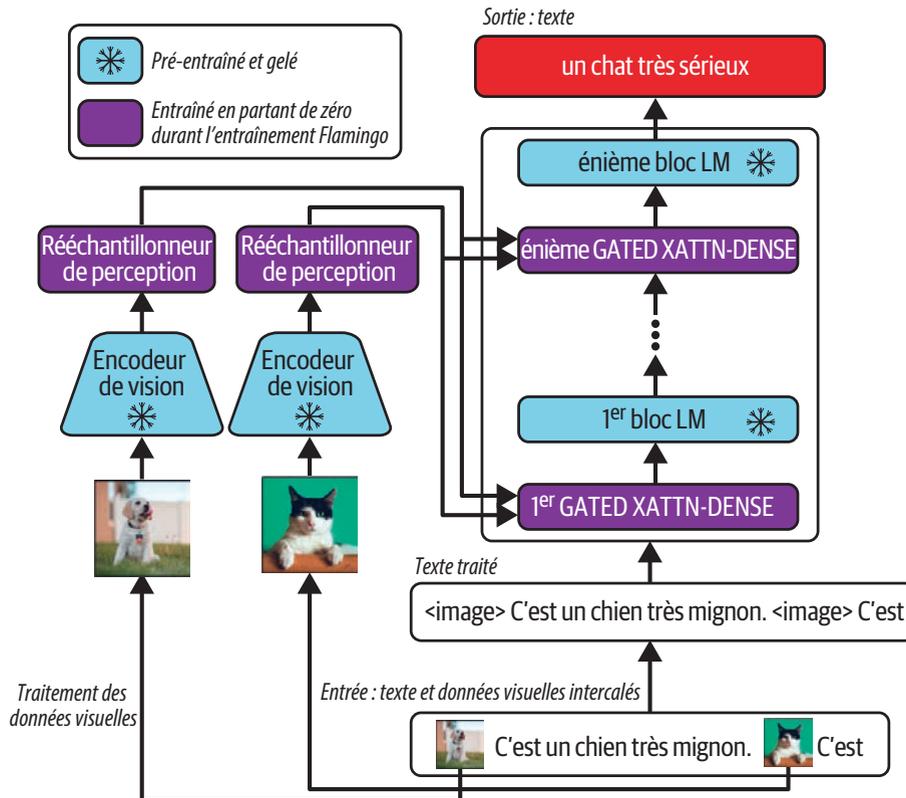
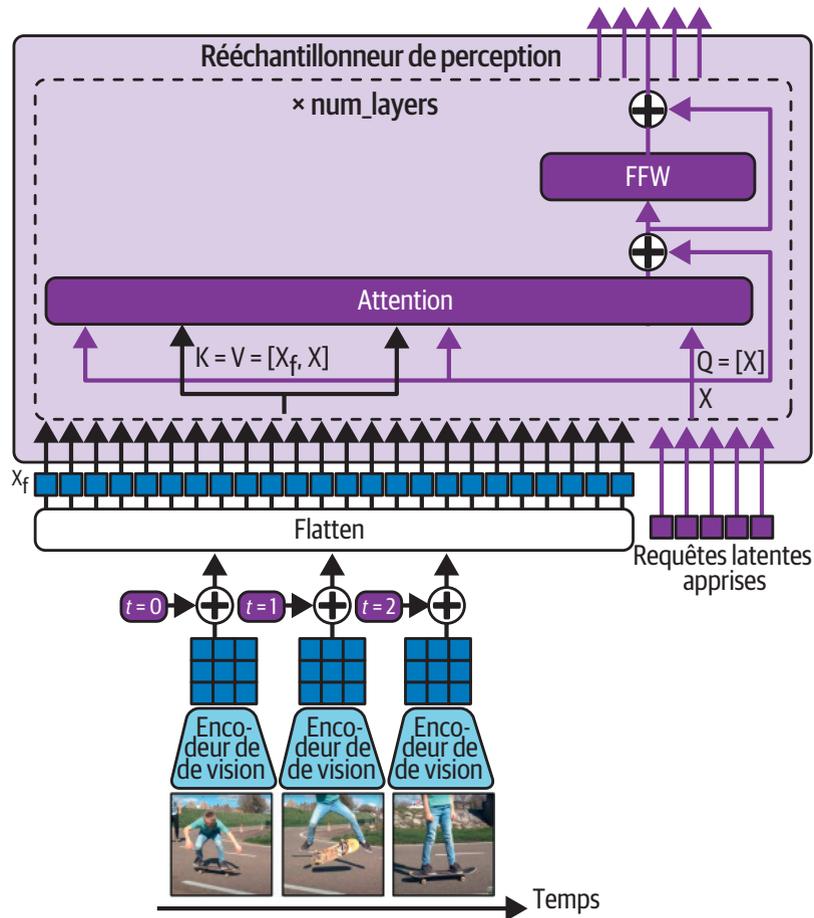


Figure 13.22 : Architecture de Flamingo (source : Alayrac et al., 2022 ; <https://arxiv.org/pdf/2204.14198.pdf>).



```
def perceiver_resampler(
    x_f, # Caractéristique visuelles [T, S, d] (T=temps, S=espace)
    time_embeddings, # Embeddings de position temporelle [T, 1, d]
    x, # R latences de forme apprises [R, d]
    num_layers, # Nombre de couches
):
    """ Modèle de rééchantillonneur de perception """

    # Ajoute les embeddings de position temporelle et la couche Flatten
    x_f = x_f + time_embeddings
    x_f = flatten(x_f) # [T, S, d] -> [T * S, d]
    # Applique les couches du rééchantillonneur de perception
    for i in range(num_layers):
        #Attention
        x = x + attention_i(q=x, kv=concat([x_f, x]))
        # Propagation avant
        x = x + ffw_i(x)
    return x
```

Figure 13.23 : Rééchantillonneur de perception appliqué à l'entrée vidéo (source : Alayrac et al., 2022 ; <https://arxiv.org/pdf/2204.14198.pdf>).

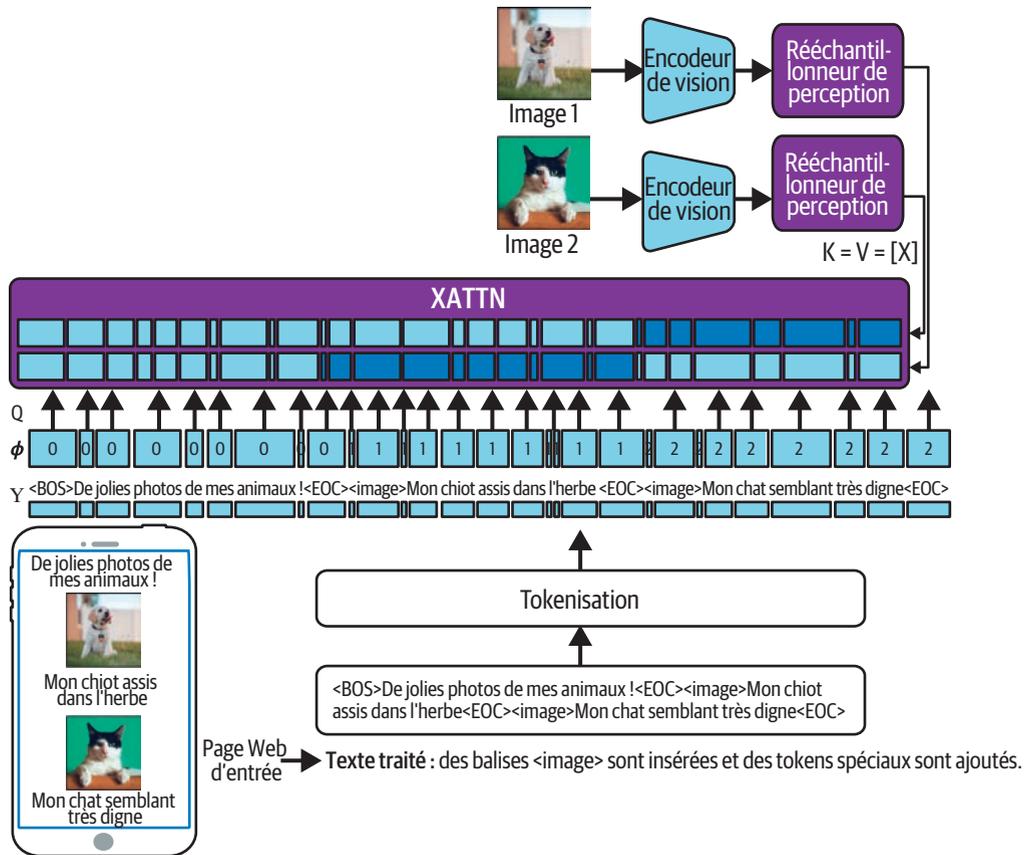
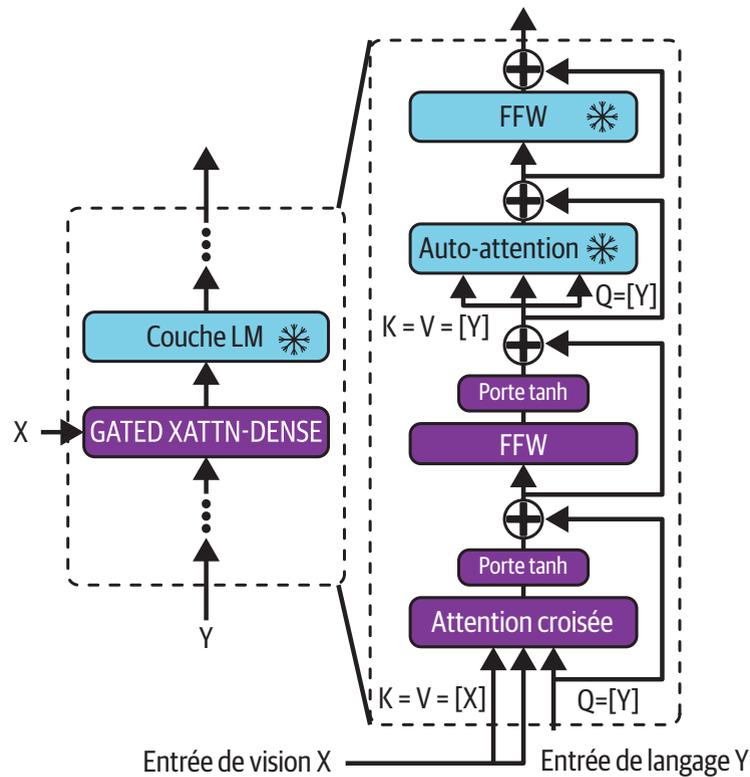


Figure 13.24 : Attention croisée masquée (XATTN), combinant des données visuelles et textuelles ; les entrées en bleu clair sont masquées et les entrées en bleu foncé ne le sont pas (source : Alayrac et al., 2022).



```

def gated_xattn_dense(
    y, # caractéristiques d'entrée de langage
    x, # caractéristiques d'entrée de vision
    alpha_xattn, # paramètres de porte xattn ; initialisés à 0
    alpha_dense, # paramètres de porte ffw ; initialisés à 0
):
    """ Applique une couche GATED XATTN-DENSE. """

    # 1. Porte d'attention croisée
    y = y + tanh(alpha_xattn) * attention(q=y, kv=x)
    # 2. Couche (dense) de propagation avant
    y = y + tanh(alpha_dense) * ffw(y)

    # Auto-attention normale + FFW sur le langage
    y = y + frozen_attention(q=y, kv=y)
    y = y + frozen_ffw(y)
    return y # renvoie des caractéristiques linguistiques visuellement informées

```

Figure 13.25 : Bloc de modèle de langage (LM) Flamingo, comprenant une couche de modèle de langage gelé de Chinchilla et une couche XATTN-DENSE GATED (source : Alayrac et al., 2022).

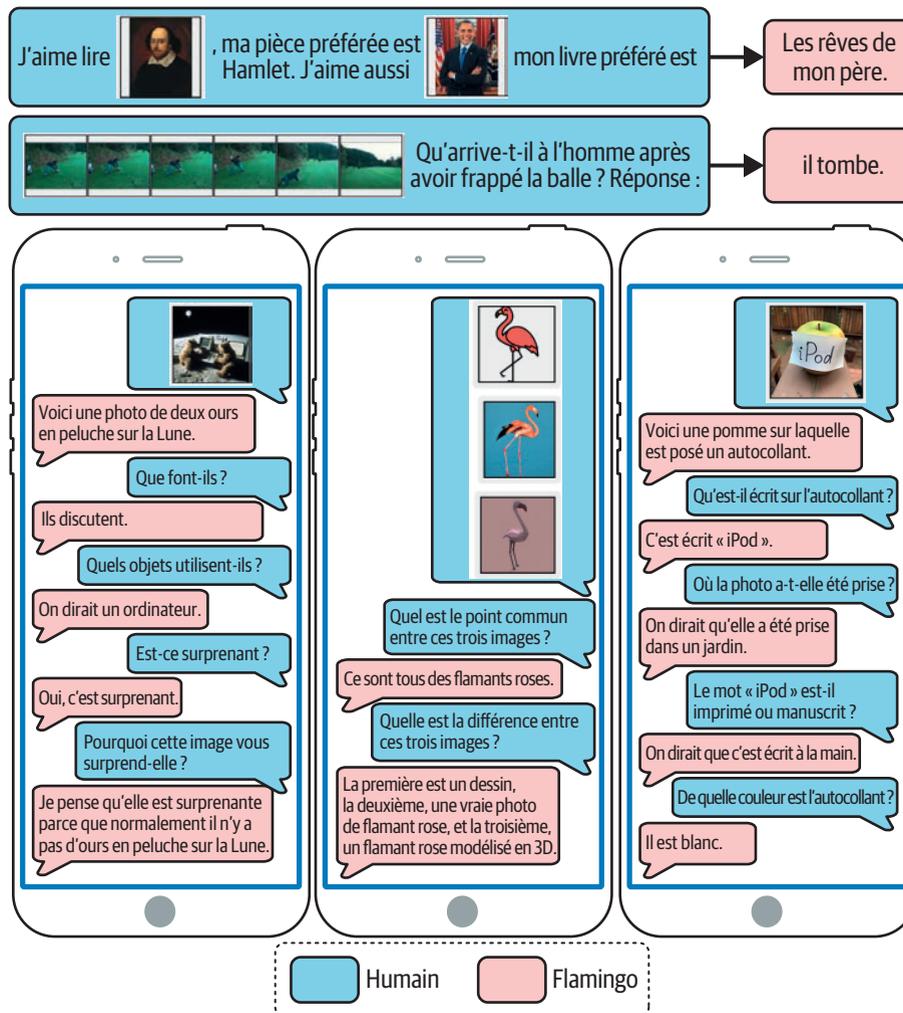


Figure 13.26 : Exemples d'entrées et de sorties obtenues à partir du modèle Flamingo doté de 80 milliards de paramètres (source : Alayrac et al., 2022).

Chapitre 14. Conclusion

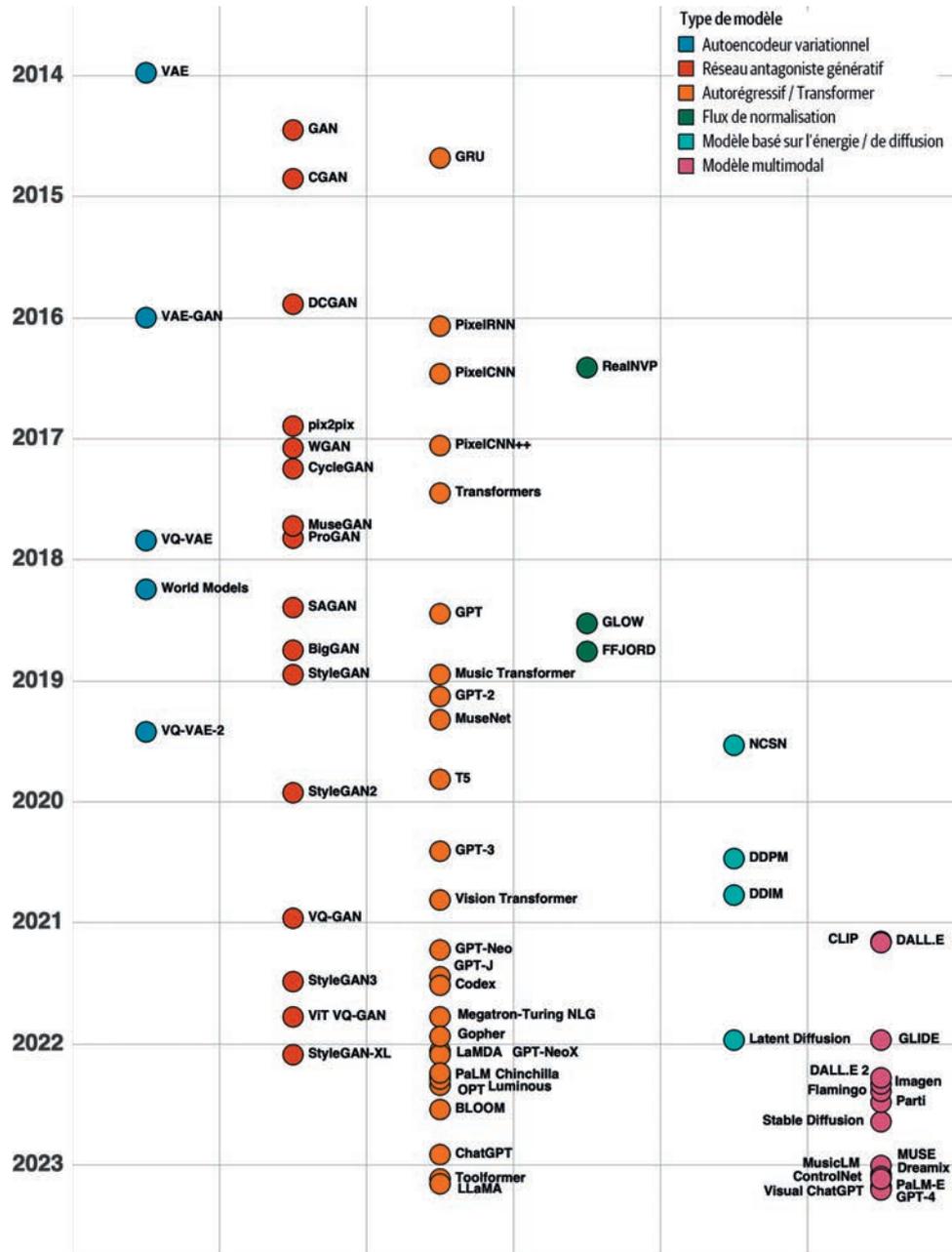


Figure 14.1 : Bref historique de l'IA générative de 2014 à 2023 (note : certains développements importants tels que les LSTM et les premiers modèles basés sur l'énergie [par exemple, les machines de Boltzmann] sont antérieurs à cette chronologie).

- D Vous**
Rédige un paragraphe d'introduction passionnant et percutant pour un manuel sur l'IA générative.
- ChatGPT**
Entrez dans le monde dynamique et en constante évolution de l'intelligence artificielle générative avec ce manuel captivant. Là où autrefois la technologie se bornait à imiter, elle se métamorphose désormais en une force créatrice capable d'insuffler la vie à l'inimaginable. Dans les lignes qui suivent, vous découvrirez les intrications fascinantes de l'IA générative, une révolution qui transcende les limites du possible. Des algorithmes qui composent des symphonies à ceux qui esquissent des œuvres d'art inédites, cette exploration vous emmènera au cœur de l'extraordinaire fusion entre la logique binaire et la créativité infinie. Préparez-vous à dévoiler les mystères de cette frontière technologique, à comprendre les fondements de son fonctionnement, et à embrasser un avenir où la machine devient l'alliée créative de l'esprit humain. Bienvenue dans le manuel qui ouvre les portes d'une ère où l'impossible cède sa place à l'imprévisible.
- D Vous**
Écris un sonnet sur le paragraphe suivant :
Jusqu'à un passé récent, la modélisation discriminante a été la force motrice de la plupart des progrès dans la méthodologie de l'apprentissage automatique. En effet, pour tout problème de discrimination, le problème de modélisation générative correspondant est généralement beaucoup plus difficile à résoudre. Par exemple, il est beaucoup plus facile d'entraîner un modèle à prédire si une peinture est de Van Gogh que d'entraîner un modèle pour générer une peinture de Van Gogh en partant de zéro. De même, il est beaucoup plus facile d'entraîner un modèle pour prédire si une page de texte a été écrite par Charles Dickens que de construire un modèle pour générer un ensemble de paragraphes dans le style de Dickens. Encore récemment, la plupart des défis de l'IA générative étaient tout simplement hors de portée et beaucoup doutaient qu'ils puissent un jour être résolus. La créativité était considérée comme une capacité purement humaine qui ne pouvait être égalée par l'IA.
- ChatGPT**
Dans le rétroviseur du progrès, discriminante,
L'IA s'est dressée, grande avancée persistante.
Van Gogh prédire, tâche aisée et confiante,
Mais créer son œuvre, art générateur, émérite.
- Charles Dickens, ses mots, modèle repérable,
Prédire son écran, tâche tangible, fiable.
Mais forger un style, défi insurmontable,
Là où l'IA butait, frontière intangible.
- La créativité, domaine humain jadis,
Au fil du temps, l'IA tente et grandit.
Des doutes planaient, défis inaccessibles,
Mais l'avenir s'éclaire, horizon indicible.
- Les pages du progrès, des feuillets dévoilés,
Là où l'IA, créatrice, a désormais osé.

Figure 14.2 : Sortie de GPT-3.5.

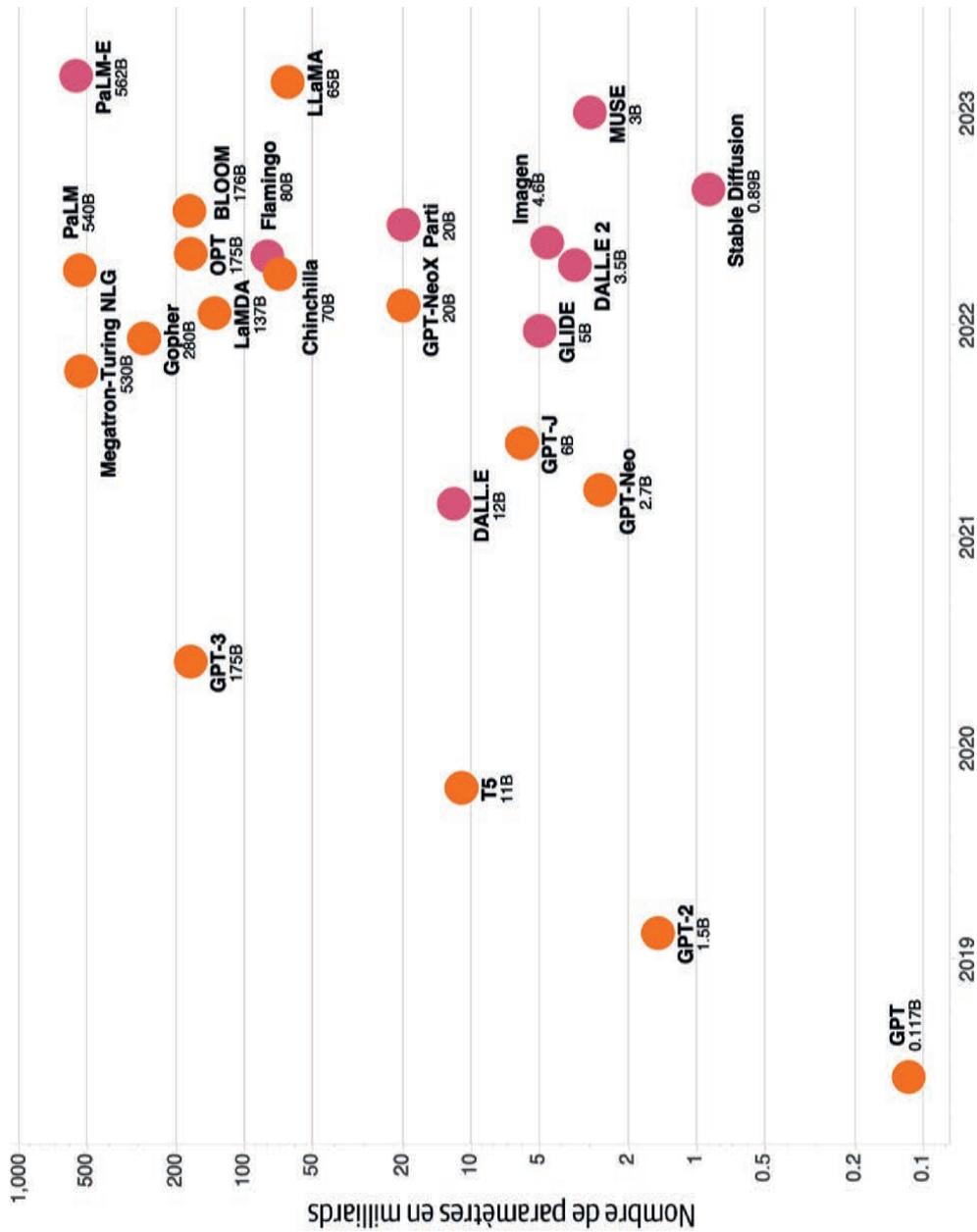


Figure 14.3 : Taille en milliards (B est l'abréviation anglaise de milliard) des grands modèles de langage (en orange) et des modèles multimodaux (en rose) en nombre de paramètres au fil du temps.

D

Vous

Quel est le record de vitesse pour la traversée de l'Atlantique à pied ?



ChatGPT

À ma connaissance en date de janvier 2022, il n'y a pas de record officiel ou reconnu pour la traversée de l'Atlantique à pied. La traversée de l'Atlantique à pied serait une entreprise extrêmement difficile, voire impossible, étant donné les vastes océans et les défis physiques et logistiques associés. Les traversées océaniques à pied ne sont généralement pas réalisables en raison des distances, des conditions météorologiques imprévisibles, et des dangers inhérents à la navigation en haute mer. Les records de vitesse sont plus couramment établis pour des traversées en bateau ou en avion. Si des tentatives de traversée de l'Atlantique à pied ont eu lieu récemment et ont établi des records, je ne dispose pas de ces informations après ma date de connaissance en janvier 2022.

D

Vous

Si j'avais le double de l'âge de ma sœur quand j'avais 8 ans, quel est mon âge aujourd'hui alors que ma sœur a 70 ans ?



ChatGPT

Lorsque vous aviez 8 ans, votre sœur avait probablement 0 ans (puisque'elle n'est pas mentionnée comme ayant un âge à ce moment-là). Si à cet âge vous aviez le double de l'âge de votre sœur, cela signifie que vous aviez 16 ans. Maintenant, si votre sœur a 70 ans, il s'est écoulé un certain nombre d'années depuis que vous aviez 16 ans. Pour trouver votre âge actuel, soustrayez votre âge initial de votre âge actuel. Donc, si vous aviez 16 ans lorsque vous aviez 8 ans et votre sœur a maintenant 70 ans, vous avez actuellement $16 + (70 - 8) = 78$ ans.

Figure 14.4 : Si les grands modèles de langage excellent dans certaines tâches, ils sont également sujets à des erreurs liées au raisonnement factuel ou logique (exemples de réponses de GPT-3.5).

```

1 import tweepy, os # secrets in environment variables
2
3 def fetch_tweets_from_user(user_name):
4     # authentication
5     auth = tweepy.OAuthHandler(os.environ['TWITTER_KEY'], os.environ['TWITTER_SECRET'])
6     auth.set_access_token(os.environ['TWITTER_TOKEN'], os.environ['TWITTER_TOKEN_SECRET'])
7     api = tweepy.API(auth)
8
9     # fetch tweets
10    tweets = api.user_timeline(screen_name=user, count=200, include_rts=False)
11    return tweets

```

Copilot

```

1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, value, currency).
5     Ignore lines starting with #.
6     Parse the date using datetime.
7     Example expenses_string:
8         2016-01-02 -34.01 USD
9         2016-01-03 2.59 DKK
10        2016-01-03 -2.72 EUR
11    """
12    expenses = []
13    for line in expenses_string.splitlines():
14        if line.startswith("#"):
15            continue
16        date, value, currency = line.split(" ")
17        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
18                        float(value),
19                        currency))
20    return expenses

```

Copilot

Figure 14.5 : Deux exemples des capacités de GitHub Copilot (source : GitHub Copilot).

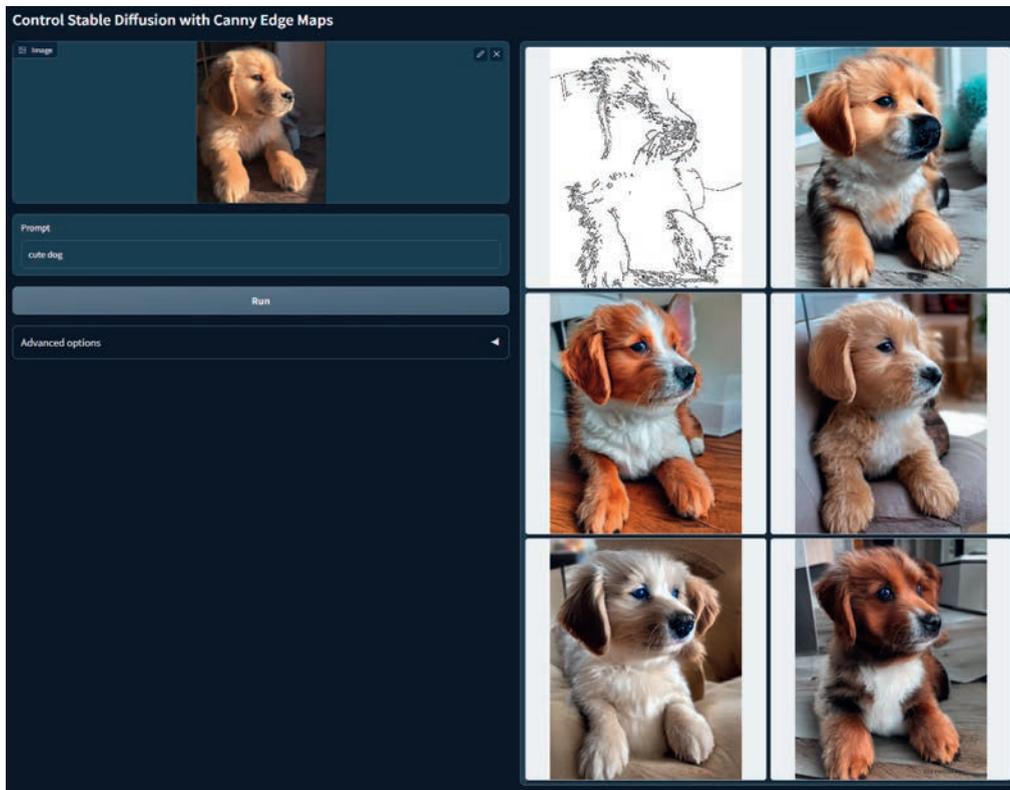


Figure 14.6 : Traitement de la sortie de Stable Diffusion à l'aide d'un filtre de Canny et de ControlNet (source : Lymin Zhang, ControlNet ; <https://github.com/llyasviel/ControlNet>).

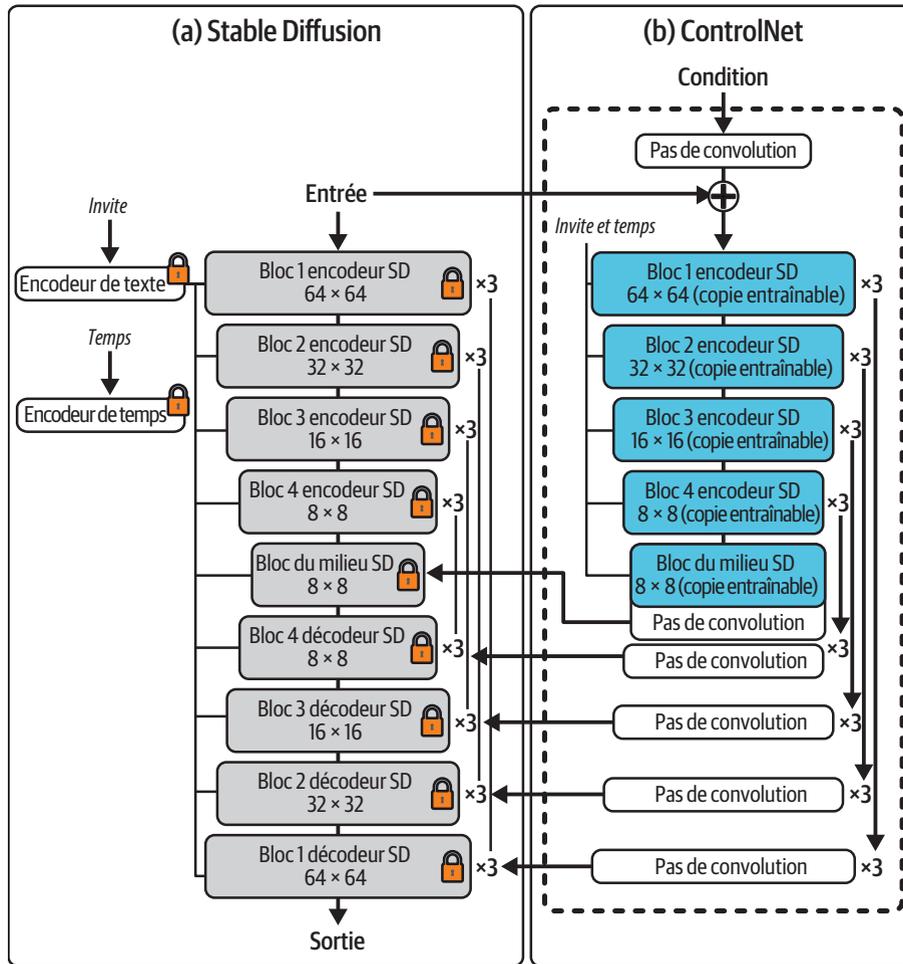
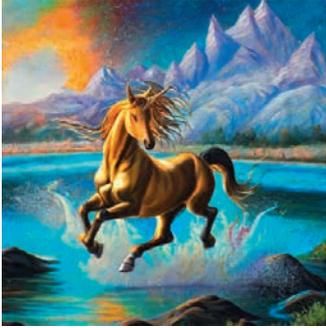


Figure 14.7 : Architecture de ControlNet, avec les copies entraînaibles des blocs de l'encodeur de Stable Diffusion (source : Lvmin Zhang, ControlNet).

Licorne sautant au-dessus d'une magnifique rivière, belle peinture à l'huile, paysage naturel époustouflant, lumière dorée, détails complexes.



Stable Diffusion v2.1



Midjourney



DALL.E 2

Figure 14.8 : Résultats de Stable Diffusion v2.1, Midjourney, et DALL.E 2 pour la même invite.

D Vous

Propose-moi trois destinations de vacances en Europe qui sortent des sentiers battus et qui proposent à la fois des activités sportives et de détente, pour une famille avec des adolescents. Donne des exemples expliquant pourquoi chaque destination est idéale.

 **ChatGPT**

1. Les Cinque Terre, Italie:

Activités Sportives: Les Cinque Terre offrent des sentiers de randonnée spectaculaires le long de la côte méditerranéenne, permettant à la famille de s'engager dans des excursions pittoresques avec des vues à couper le souffle. Les amateurs de sports nautiques peuvent également profiter du kayak ou de la plongée dans les eaux cristallines.

Détente: Les charmants villages colorés et la cuisine italienne délicieuse créent une atmosphère détendue. Les plages paisibles sont parfaites pour se relaxer et profiter du soleil méditerranéen.

2. L'Algarve, Portugal:

Activités Sportives: L'Algarve offre une variété d'activités nautiques passionnantes, telles que le surf et la planche à voile le long de ses côtes. Les falaises et les grottes offrent également d'excellentes opportunités pour l'escalade et l'exploration.

Détente: Les plages de sable doré, les villages de pêcheurs pittoresques et les marchés locaux animés permettent à la famille de se détendre après une journée active. La cuisine portugaise diversifiée ravira les papilles de tous.

3. La Croatie (îles de Hvar et Korčula):

Activités Sportives: Les îles de Hvar et Korčula en Croatie sont idéales pour les amateurs de voile. Les familles peuvent louer un bateau pour explorer les criques cachées, tandis que la plongée sous-marine offre une vue fascinante sur la vie marine adriatique.

Détente: Ces îles offrent des plages tranquilles, des ruelles médiévales charmantes et une atmosphère méditerranéenne décontractée. Les marchés locaux proposent des produits frais pour des pique-niques relaxants au bord de la mer.

Figure 14.9 : Production de GPT-3.5 donnant des conseils de voyage.



Vous

Explique à un étudiant la différence entre l'IA générative et l'IA discriminante.



ChatGPT

L'intelligence artificielle (IA) générative et discriminante sont deux approches différentes dans le domaine de l'apprentissage machine, et elles servent des objectifs distincts.

1. Intelligence Artificielle Générative (IA Générative):

- **Objectif Principal:** L'IA générative vise à créer de nouvelles données à partir d'un ensemble existant. En d'autres termes, elle cherche à générer des données qui ressemblent à celles qu'elle a apprises pendant son entraînement.
- **Fonctionnement:** Les modèles d'IA générative sont formés pour comprendre la structure et les motifs des données fournies. Ils peuvent ensuite créer de nouvelles données en extrapolant à partir de ces motifs. Des exemples incluent la génération d'images, de textes, de musique, etc.
- **Exemple concret:** Les réseaux génératifs adversaires (GANs) sont un type courant de modèle d'IA générative, utilisés pour créer des images réalistes qui semblent avoir été prises par des humains.

2. Intelligence Artificielle Discriminante (IA Discriminante):

- **Objectif Principal:** L'IA discriminante vise à classifier ou à catégoriser des données en fonction des informations qu'elle a apprises pendant son entraînement. Son but est de prendre des décisions basées sur des caractéristiques spécifiques des données.
- **Fonctionnement:** Les modèles d'IA discriminante analysent les données pour les classer dans différentes catégories prédéfinies. Ils sont souvent utilisés dans des tâches telles que la reconnaissance d'images, la détection de spam, la classification de texte, etc.
- **Exemple concret:** Les réseaux neuronaux convolutifs (CNN) sont un exemple populaire d'IA discriminante utilisée pour la classification d'images.

En résumé, l'IA générative crée de nouvelles données, tandis que l'IA discriminante classe ou catégorise des données existantes. Les deux types d'IA peuvent souvent être utilisés de manière complémentaire dans des projets plus vastes, mais ils ont des objectifs fondamentalement différents.

Figure 14.10 : Production de GPT-3.5 ; exemple d'utilisation de modèles de langage de grande taille pour l'apprentissage.

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

Figure 14.11 : Exemple de la façon dont Toolformer est capable d'appeler de manière autonome différentes API afin d'obtenir des informations précises si nécessaire (source : Schick et al., 2023).