Architectures de réseaux profonds

Introduction à l'apprentissage automatique – GIF-4101 / GIF-7005

Professeur : Christian Gagné

Semaine 10

UNIVERSITÉ LAVAL

10.1 Convolution et traitement des images

Convolution

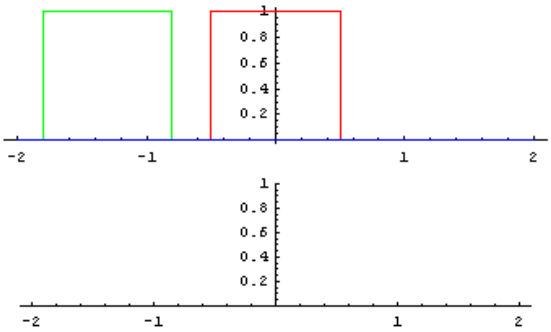
- Convolution : produit de deux fonctions sur un même domaine

$$f(x) * g(x) \equiv \int_{t=-\infty}^{\infty} f(x-t)\, g(t)\, \mathrm{d}t$$

- Formulation discrète

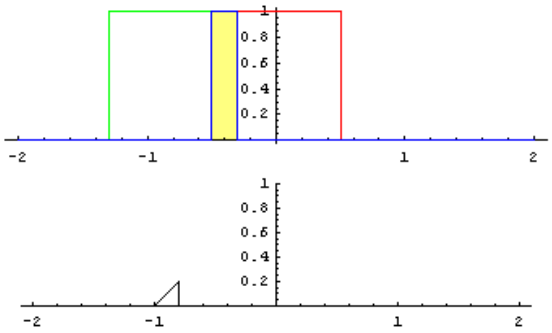
$$f(x) * g(x) \equiv \sum_{t=-\infty}^{\infty} f(x-t)\, g(t)$$

Exemple de convolution



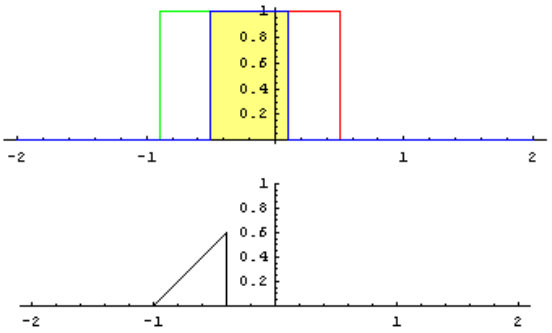
Par Lautaro Carmona, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif.

Exemple de convolution



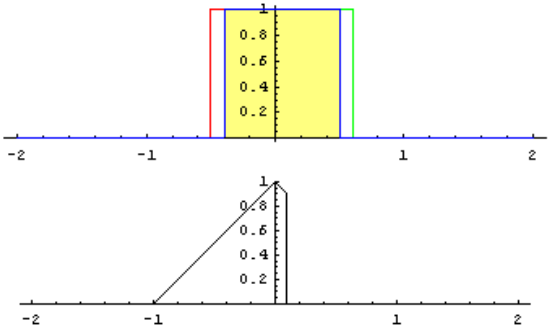
Par Lautaro Carmona, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif.

Exemple de convolution



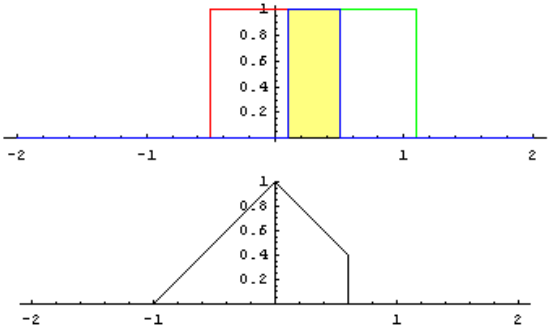
Par Lautaro Carmona, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif.

Exemple de convolution



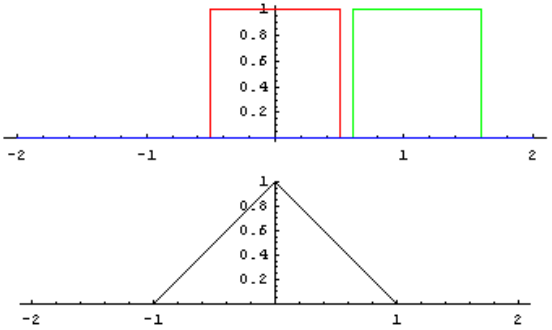
Par Lautaro Carmona, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif.

Exemple de convolution



Par Lautaro Carmona, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif.

Exemple de convolution



Par Lautaro Carmona, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif.

Convolution et estimation de densité

- Distribution de Dirac décentrée

$$\delta(x-t) = \begin{cases} \infty & \text{si } x = t \\ 0 & \text{autrement} \end{cases}, \quad \int_{x=-\infty}^{\infty} \delta(x-t)\,\mathrm{d}x = 1.$$

  - Convolution sur Dirac décentrés

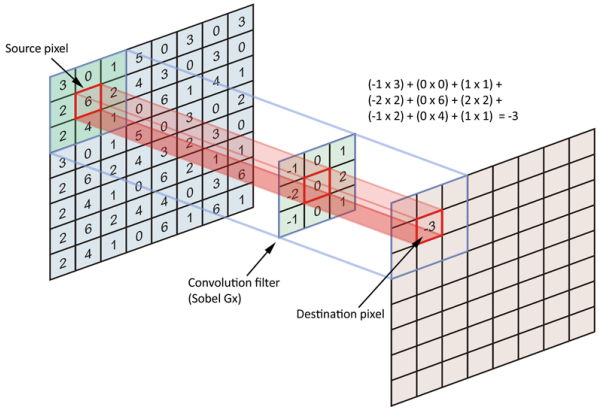
$$f(x) * \delta(x-u) = f(x-u)$$

- Estimation de densité à noyau : convolution du noyau avec plusieurs Diracs centrés sur les données

$$\hat{p}(x) = \frac{1}{Nh}\sum_{t=1}^{N} K\left(\frac{x-x^t}{h}\right) = \frac{1}{Nh}\sum_{t=1}^{N} K\left(\frac{x}{h}\right) * \delta(x-x^t)$$

Traitement d'images

- Convolution 2D est un élément de base du traitement d'images

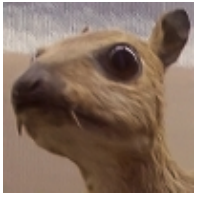


Source : https://thigiacmaytinh.com/wp-content/uploads/2018/05/kernel.png, accédé le 13 novembre 2018.

Exemples de filtres

Noyau identité (3×3) :

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Flou de Gauss :

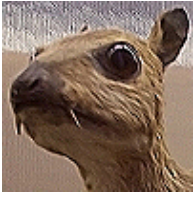
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Détection de contours :

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Amélioration de la netteté :

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

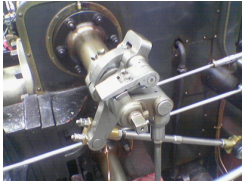


Opérateur de Sobel

- Filtre classique pour la détection d'arêtes
  - Calcul les gradients locaux de l'intensité de l'image
  - S'appuie sur deux convolutions pour obtenir le gradient vertical $\mathbf{G}_x$ et le gradient horizontal $\mathbf{G}_y$ d'une image $\mathbf{A}$, le résultat est une image $\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$

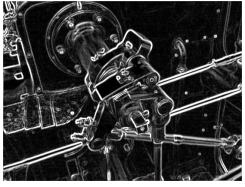
$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}, \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Image d'origine :



Par Simpsons contributor, CC-BY-SA 3.0, https://commons.wikimedia.org/wiki/File:Valve_original_(1).PNG

Application de Sobel :



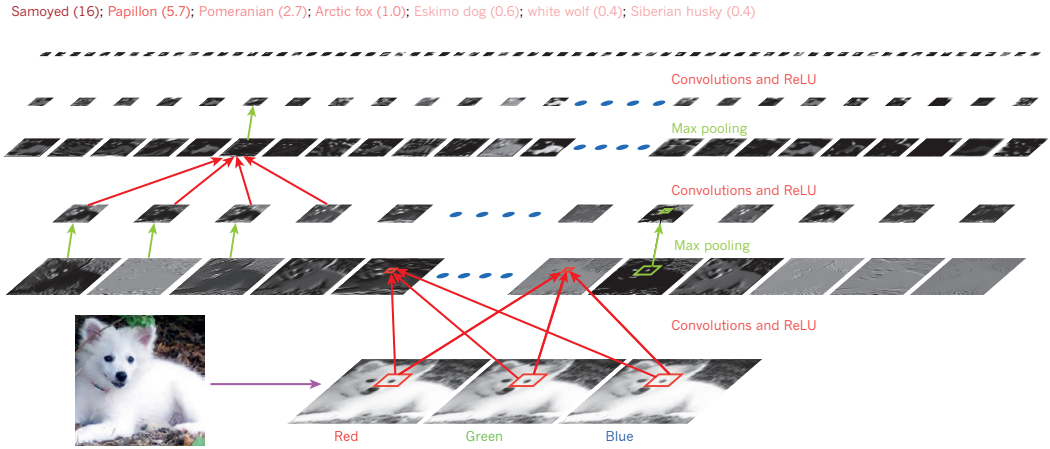
Par Simpsons contributor, CC-BY-SA 3.0, https://commons.wikimedia.org/wiki/File:Valve_sobel_(3).PNG

10.2 Réseaux de neurones à convolution

Réseaux de neurones à convolution

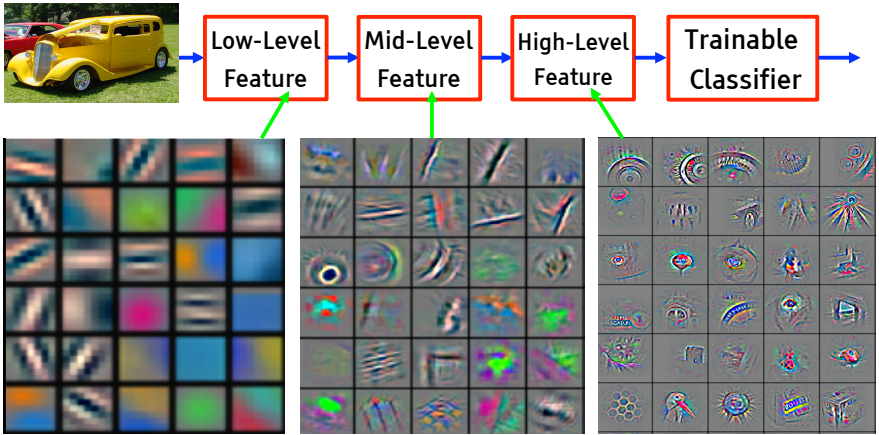
- Idée : définir des réseaux de neurones comportant des opérations de convolution
  - Apprendre les valeurs numériques des filtres convolués
  - Définir un réseau exploitant des éléments de la structure des données
    - Son ou parole : données temporelles (convolutions 1D)
    - Image : données spatiales (convolutions 2D)
    - Vidéo : données spatio-temporelles (convolutions 3D)
  - Enchaînement d'étages de convolutions, filtrant sortie de couche précédente
  - Permet une modèlisation plus compacte que réseaux pleinement connectés et invariante en translation
- Quelques composants d'un réseaux à convolution
  - Couche de filtres convolués sur les différents canaux
  - Pooling : valeur maximale (max pool) ou moyenne (avg pool) dans une certaine fenêtre convoluée
  - Fonctions de transfert : ReLU, etc.
  - Près de la sortie, couches pleinement connectées (comme avec perceptron multicouche)

Réseau à convolution



Tiré de *Y. LeCun, Y. Bengio et G. Hinton, Deep Learning, Nature, vol. 521, 28 mai 2015*. Accédé en ligne le 6 novembre 2020 au `https://www.nature.com/articles/nature14539`.

Composition de filtres

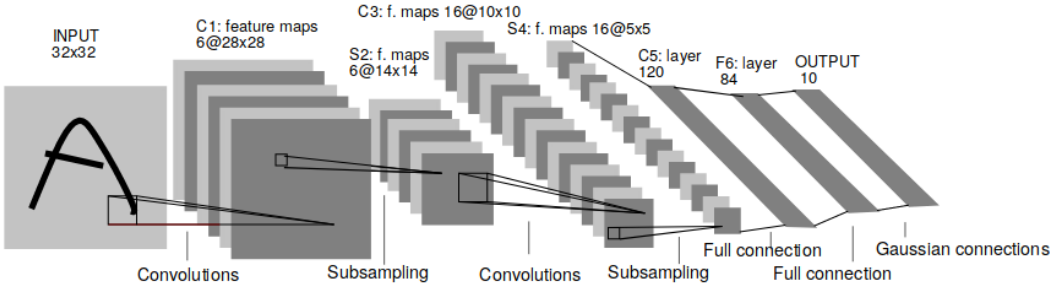


Tiré de *G. Hinton, Y. Bengio et Y. LeCun, Deep Learning NIPS'15 Tutorial, 2015*. Accédé en ligne le 6 novembre 2020 au `https://nips.cc/Conferences/2015/Schedule?showEvent=4891`.

10.3 Exemples de réseaux à convolution

LeNet5

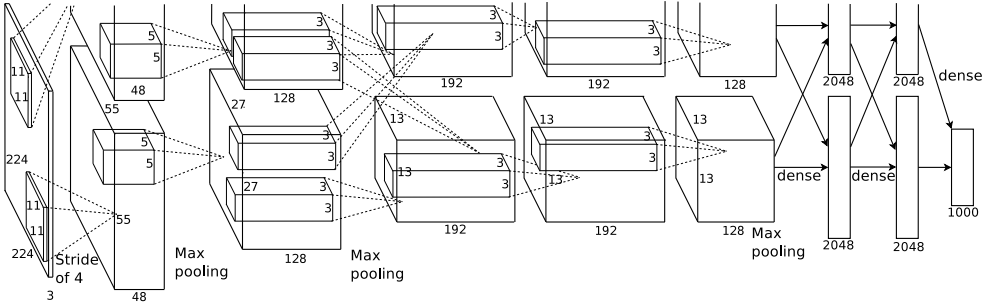
- LeNet5 : réseau à convolution classique, proposé dans les années 1990
  - 3 couches de convolutions, 2 couches de average pooling, 2 couches pleinement connectées
  - 60k paramètres (de 10M à 100M avec réseaux modernes)



Tiré de *Y. LeCun, L. Bottou, Y. Bengio et P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86(11), 1998.* Accédé en ligne le 6 novembre 2020 au `http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf`.

AlexNet

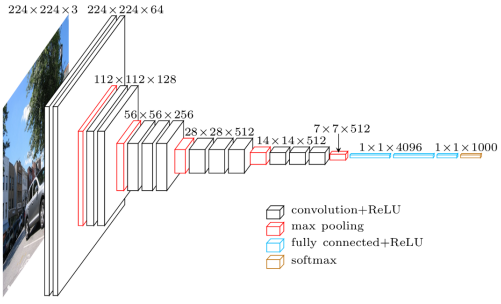
- AlexNet : réseau pour la reconnaissance d'objets
  - Gagnant du concours ImageNet 2012
  - Implémenté pour calculs sur GPU
  - Souvent utilisé comme modèle de base pour transfert de représentations
  - 8 couches de convolution, quelques couches de max pooling, 3 couches pleinement connectées



Tiré de *A. Krizhevsky, I. Sutskever, et G. Hinton, Imagenet classification with deep convolutional neural networks. NIPS, 2012.* Accédé en ligne le 6 novembre 2020 au `https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

VGG

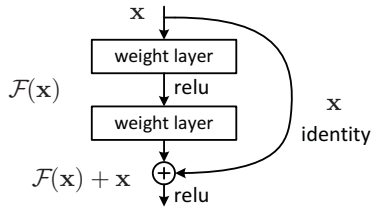
- VGGNet : plus grande profondeur avec topologie simplifiée
  - Gagnant du concours ImageNet 2013
  - Profondeur est un élément critique pour de bonnes performances
  - Similaire à AlexNet, mais avec seulement convolutions 3×3, max pooling 2×2, 3 couches pleinement connectées et 16 couches au total (VGG-16)



Source : https://heuritech.files.wordpress.com/2016/02/vgg16.png, accédé le 13 novembre 2018.

ResNet

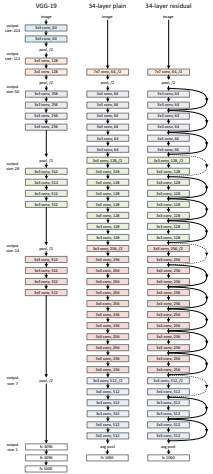
- Réseaux résiduels : permettre des connexions directes entre couches non adjacentes (*skip links*)



Tiré de *K. He, X. Zhang, S. Ren, et J. Sun, Deep residual learning for image recognition. CVPR, 2016.* Accédé en ligne le 6 novembre 2020 au `https://arxiv.org/abs/1512.03385`.

- Permet des réseaux beaucoup plus profonds et performants
  - Gagnant de compétition ImageNet 2015 (3,57 % d'erreur top 5)
  - Facilite l'optimisation et la propagation du signal à travers le réseau
  - Bloc résiduel doit faire mieux qu'un traitement directement sur le bloc précédent

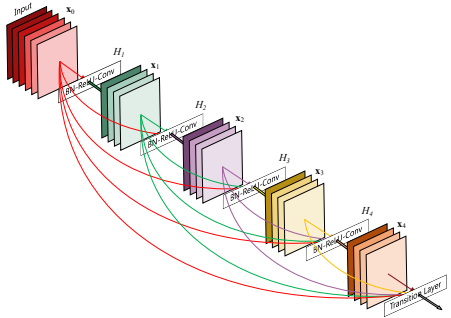
ResNet



Tiré de *K. He, X. Zhang, S. Ren, et J. Sun, Deep residual learning for image recognition. CVPR, 2016.* Accédé en ligne le 6 novembre 2020 au `https://arxiv.org/abs/1512.03385`.

DenseNet

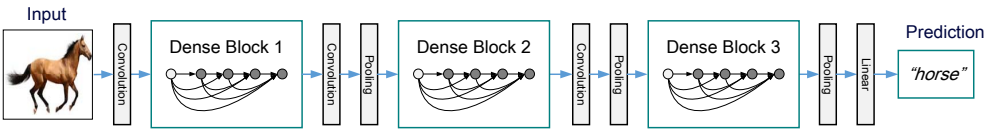
- Observation : réseaux à convolution peuvent être plus profonds et obtenir de meilleures performances avec connections proches dans tout le réseau à son entrée
- DenseNet : connecter chaque couche à toutes les couches qui précèdent
  - Réseau à L couches aura $L(L+1)/2$ connections directes entre les couches



Tiré de *G. Huang, Z. Liu, L. Van Der Maaten et K.Q. Weinberger, Densely Connected Convolutional Networks. CVPR, 2017.* Accédé en ligne le 6 novembre 2020 au `https://arxiv.org/abs/1608.06993`.

DenseNet

- En pratique, on créé des blocs denses séparés par des couches de convolutions et de pooling



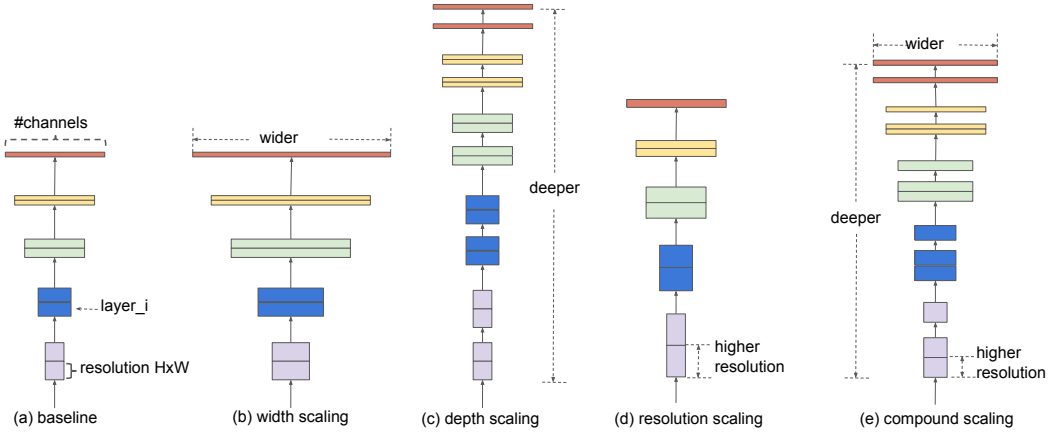
Tiré de *G. Huang, Z. Liu, L. Van Der Maaten et K.Q. Weinberger, Densely Connected Convolutional Networks. CVPR, 2017.* Accédé en ligne le 6 novembre 2020 au `https://arxiv.org/abs/1608.06993`.

- Chaque couche dans un bloc dense peut être relativement « étroite », c'est-à-dire peut comprendre peu de neurones

EfficientNet

- EfficientNet : ajustement optimal de la taille de réseaux à convolution
  - Comment faire l'ajustement de l'architecture de réseaux selon les ressources disponibles ?
- Idée : si la résolution de l'image est plus élevée, les performances seront meilleures, mais les ressources requises (profondeur et largeur) sont plus importants pour bien capturer les détails des images
- Ajustement proportionnel de la profondeur, largeur et résolution selon facteur ϕ
  - Profondeur : nombre de couches du réseau, selon α^ϕ
  - Largeur : nombre de canaux dans chaque couche, selon β^ϕ
  - Résolution : ajustement de la résolution de l'image d'entrée, selon γ^ϕ
  - Valeurs de α, β et γ déterminée expérimentalement (recherche en grille) pour un réseau ayant ses ressources doublées ($\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$)
- Architecture basée sur MobileNet V2, avec goulot d'étranglement inversé des connexions résiduelles

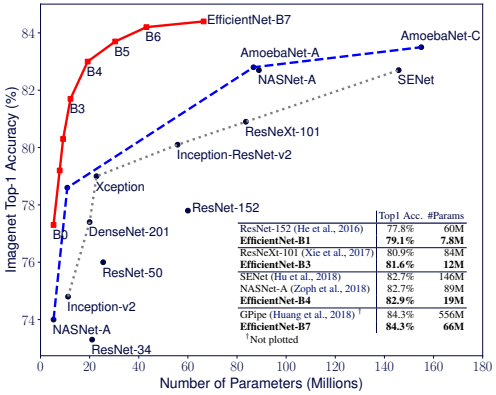
Ajustement de la taille dans EfficientNet



Tiré de *M. Tan, Q.V. Le, EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks. ICML, 2019.* Accédé en ligne le 29 octobre 2023 au `https://arxiv.org/abs/1905.11946`.

Performances avec EfficientNet

- À ressources égales, EfficientNet offre des performances supérieures
- Huit versions (EfficientNet-B0 à B7) ont été proposées pour différents compromis ressources requises / performance
- Modèles idéaux pour utilisation dans dispositifs mobiles et l'informatique en périphérie (*edge computing*)



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

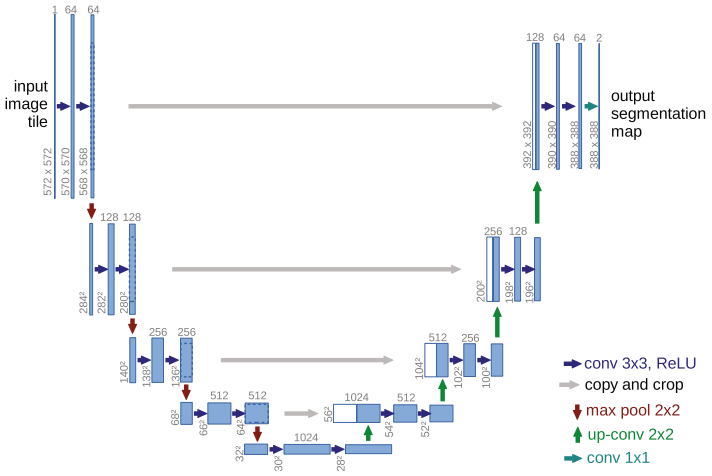
†Not plotted

Tiré de *M. Tan, Q.V. Le, EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks. ICML, 2019.* Accédé en ligne le 29 octobre 2023 au `https://arxiv.org/abs/1905.11946`.

U-Net

- Réseaux présentés jusqu'à présent d'abord proposés et testés pour reconnaissance d'objets (classement)
  - D'autres tâches possibles en vision : détection, suivi, etc.
- Segmentation : identifier régions cohérentes de l'image
  - Séparer les différentes régions
  - Donner une étiquette à chaque région
- U-Net : réseau proposé en imagerie biomédicale
  - Réseau pleinement convolutionnel, donne une image en sortie
  - Compression de l'information en milieu de réseau, similaire à un auto-encodeur
  - Skip links permettent de conserver structure spatiale

U-Net



Tiré de *O. Ronneberger, P. Fischer, et T. Brox, U-net : Convolutional networks for biomedical image segmentation. MICCAI, 2015.* Accédé en ligne le 6 novembre 2020 au `https://arxiv.org/abs/1505.04597`.

10.4 Génération d'images

Génération d'exemples

- Idée : générer des données d'entrées à partir d'une sortie désirée
  - Générer donc un modèle de la donnée pouvant produire la sortie selon le réseau de neurones
- Approche : descendre le gradient sur la donnée d'entrée

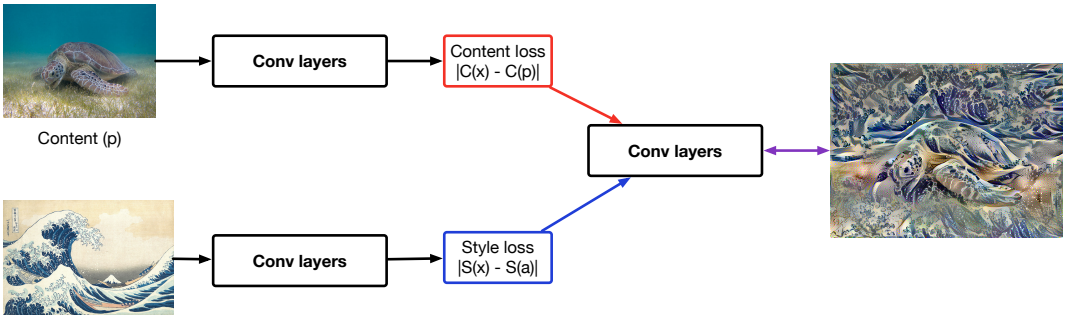
$$\Delta \mathbf{x} = -\eta \frac{\partial E(\mathbf{x}|\theta)}{\partial \mathbf{x}}$$

  - On va donc générer une nouvelle donnée à partir de la valeur initiale de **x** et la sortie désirée **r**
  - Poids du réseau ne changent pas

Deep dream

Transfert de style

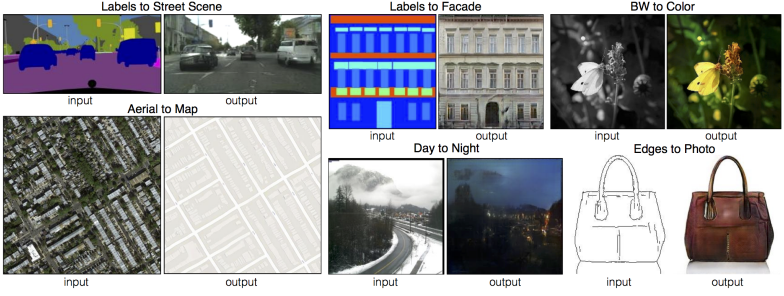
- Idée : transférer le style d'une image dans une nouvelle image
  - Comparer le contenu dans les couches de convolution (ex. VGG19) et le style (matrice de Gram)



Green Sea Turtle grazing seagrass, CC-BY-SA-3.0, https://commons.wikimedia.org/wiki/File:Green_Sea_Turtle_grazing_seagrass.jpg
The Great Wave off Kanagawa, public domain, https://commons.wikimedia.org/wiki/File:Tsunami_by_hokusai_19th_century.jpg

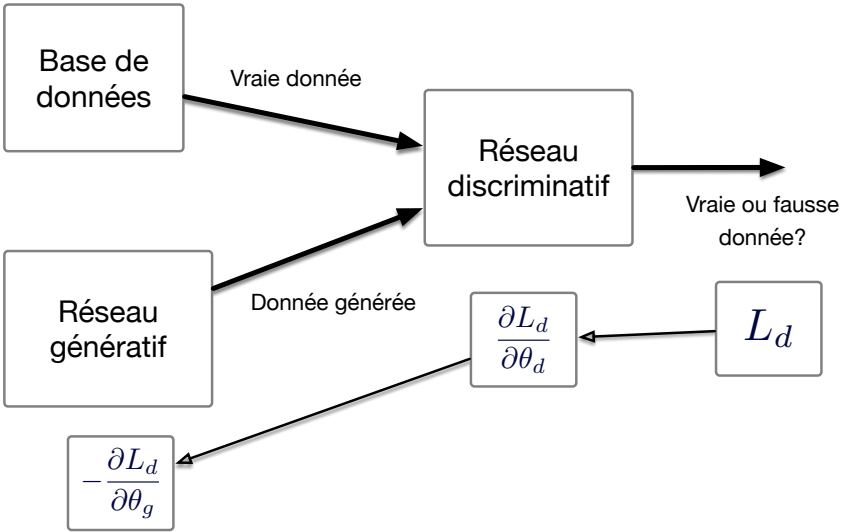
Generative Adversarial Networks (GAN)

- Modèle GAN : mettre en compétition deux réseaux de neurones
  - Réseau discriminatif : distinguer données véritables du problème des données générées
  - Réseau génératif : produire des données ayant l'air authentiques
  - Permet des traitements variés basés sur un apprentissage non supervisé
- Exemple : traduction image à image avec GANs conditionnels



Tiré de *Isola, Zhu, Zhou et Efros, Image-to-Image Translation with Conditional Adversarial Networks, CVPR, 2017.* Accédé en ligne le 19 octobre 2020 au `https://arxiv.org/pdf/1611.07004v3.pdf`.

Generative Adversarial Networks (GAN)



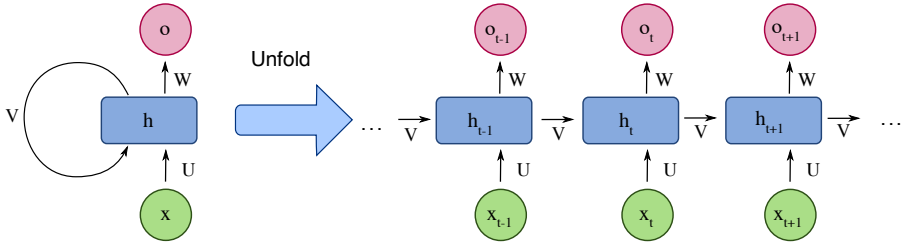
Caractéristiques des GAN

- Méthode clé dans le développement de modèles génératifs
  - La plupart des modèles génératifs historiques capables de résultats réalistes sont basés sur des GANs
  - Ex. *This person does not exist* basé sur StyleGAN
- Entraînement autosupervisé, sans requérir de données étiquetées ou de mesure de qualité explicite
  - Déclencheur d'avancées dans l'utilisation d'approches autosupervisées pour entraîner des réseaux profonds
  - Pas de garantie du réalisme et de la qualité des données produites
- Modèle complexe à entraîner
  - Équilibre dans l'entraînement des modèles génératifs et discriminatifs difficile à maintenir, tâche discriminative plus facile que la tâche générative
  - Perte de couverture dans la génération par effondrement sur un mode (*mode collapse*)
  - Entraînement peut être assez lourd en calculs

10.5 Traitement de séquences

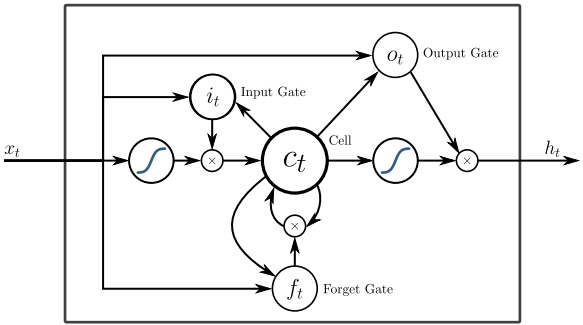
Réseau récurrent

- Réseaux usuels (*feedforward*) : données propagées dans le réseau, indépendant des données suivantes / précédentes
  - Traitement de donnés séquentielles important dans nombreux contextes
- Réseaux récurrents : connexions avec valeurs précédentes
  - Traitement avec algorithmes habituels en déroulant le réseau



Par fdeloche, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg

Long Short-Term Memory (LSTM)



Par Graves, Mohamed et Hinton, CC-SA 4.0, https://en.wikipedia.org/wiki/File:Peephole_Long_Short-Term_Memory.svg

- Modèle LSTM : ajouter de la mémoire au réseau
- Cellule de mémoire (état), avec quatre neurones
  - Entrée
  - Activation de l'entrée
  - Activation de l'oubli
  - Activation de la sortie
- Cellules utilisées comme neurones avec une mémoire dans des réseaux multicouches

Variants des LSTM

- LSTM bidirectionnel (BiLSTM) : traiter la séquence dans les deux directions
  - Cellules additionelles pour traiter données en sens inverse
  - Permet de mieux exploiter le contenu de la séquence
  - Particulièrement utile pour traitement de la langue naturelle
- GRU (*Gated Recurrent Unit*) : simplification du modèle LSTM
  - Simplification du modèle de cellule LSTM en combinant activation de l'entrée et de l'oubli
  - Fait un compromis entre complexité et performance

Forces et faiblesses des LSTM

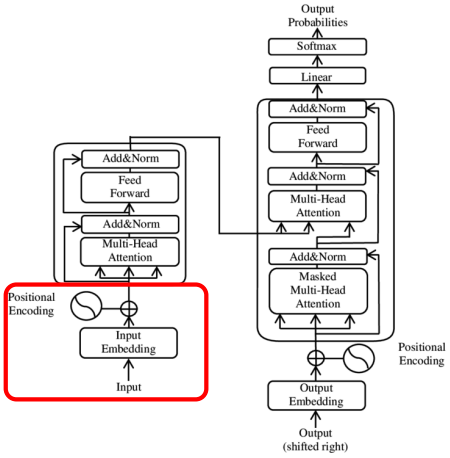
- Forces des LSTM
  - Apte à capturer des relations éloignées dans les séquences
  - A démontré une grande versatilité dans son application au traitement de séquences (ex. traduction automatisée, reconnaissance de la parole)
  - Offre un meilleur contrôle sur la dillution du gradient, qui est un enjeu avec les réseaux récurrents classiques
- Faiblesses des LSTM
  - Modèles complexes, avec nombre élevé de paramètres, requérant de longs temps d'entraînement et de gros jeux de données
  - A tendance à faire sur surapprentissage, en particulier sur de petits jeux de données

10.6 Réseaux autoattentifs

Réseaux autoattentifs (transformers)

- Réseaux autoattentifs (en anglais : *transformer networks* ou bien *self-attentive network*)
  - Utilise un mécanisme d'attention pour établir les relations les éléments d'une séquence (ex. mots d'une phrase)
  - Conçus pour permettre un traitement parallèle avec têtes multiples, permet utilisation efficace de GPU
  - Comportent une composante encodeur et une composante décodeur
  - N'utilise pas de récurrence, le mécanisme d'attention donne une capacité d'utiliser l'ensemble du contexte (mémoire à long terme)
- Modèles centraux dans les grands modèles de langue (GPT, BERT)
  - Aussi utilisé avec les images (*vision transformers* (ViT)), reconnaissance de la parole, etc.

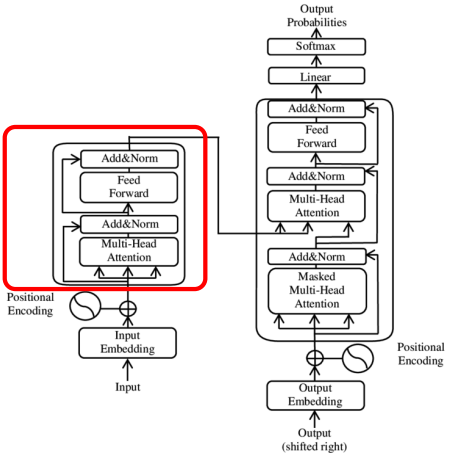
Fonctionnement des réseaux autoattentifs



Par Yuening Jia, CC BY-SA 3.0 DEED, https://commons.wikimedia.org/wiki/File:The-Transformer-model-architecture.png.

- Entrée : transformation de la séquence d'entrée en vecteur
  - Pour texte, plongement lexical + encodage positionnel de chaque mot

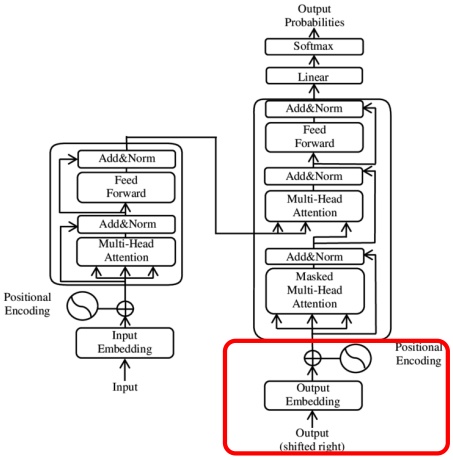
Fonctionnement des réseaux autoattentifs



Par Yuening Jia, CC BY-SA 3.0 DEED, https://commons.wikimedia.org/wiki/File:The-Transformer-model-architecture.png.

- Entrée : transformation de la séquence d'entrée en vecteur
  - Pour texte, plongement lexical + encodage positionnel de chaque mot
- Encodeur : attention sur plusieurs têtes + renormalisation
  - Attention calculée entre tous les éléments
  - Normalisation par couche pleinement connectées

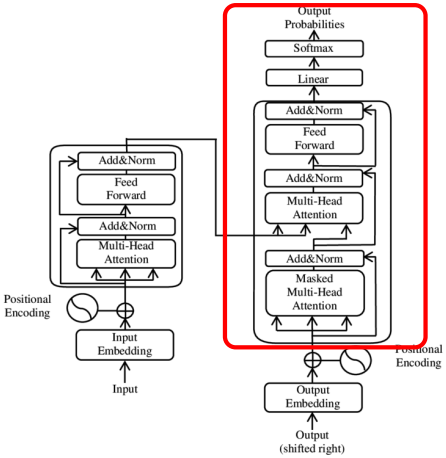
Fonctionnement des réseaux autoattentifs



Par Yuening Jia, CC BY-SA 3.0 DEED, https://commons.wikimedia.org/wiki/File:The-Transformer-model-architecture.png.

- Entrée : transformation de la séquence d'entrée en vecteur
  - Pour texte, plongement lexical + encodage positionnel de chaque mot
- Encodeur : attention sur plusieurs têtes + renormalisation
  - Attention calculée entre tous les éléments
  - Normalisation par couche pleinement connectées
- Sortie : transformation de la séquence de sortie en vecteur

Fonctionnement des réseaux autoattentifs



Par Yuening Jia, CC BY-SA 3.0 DEED, https://commons.wikimedia.org/wiki/File:The-Transformer-model-architecture.png.

- Entrée : transformation de la séquence d'entrée en vecteur
  - Pour texte, plongement lexical + encodage positionnel de chaque mot
- Encodeur : attention sur plusieurs têtes + renormalisation
  - Attention calculée entre tous les éléments
  - Normalisation par couche pleinement connectées
- Sortie : transformation de la séquence de sortie en vecteur
- Décodeur : mécanisme d'attention sur sortie et entrée
  - Premières étapes seulement sur sortie **masquée**
  - Étapes suivantes en combinant représentation de la sortie et de l'entrée
  - Normalisation par couche pleinement connectées
  - Probabilités du prochain mot en sortie

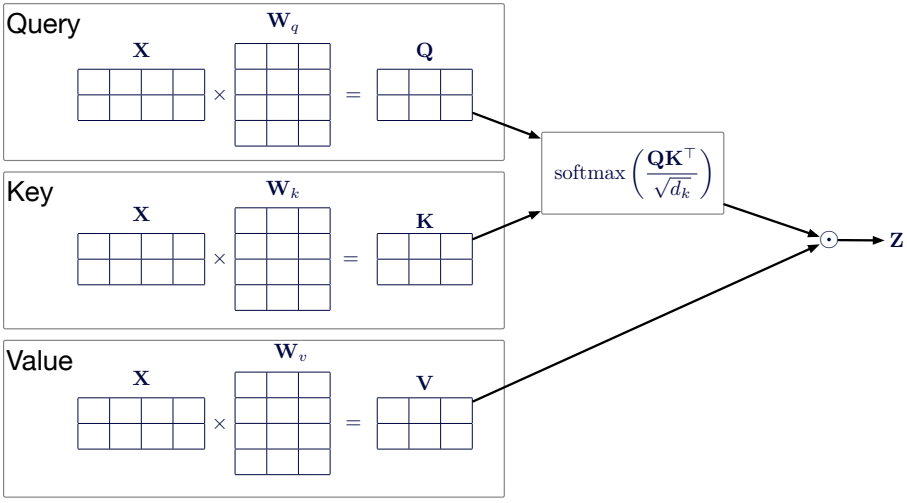
Mécanisme d'attention

- Calcul de l'attention entre la requête $\mathbf{Q}$, la clé $\mathbf{K}$ et la valeur $\mathbf{V}$ selon :

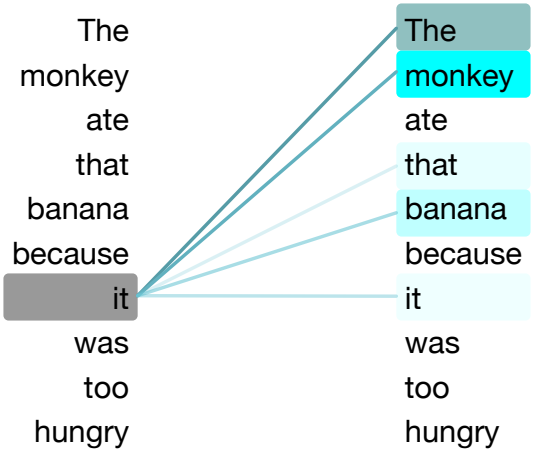
$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

  - Les valeurs de $\mathbf{Q}$, $\mathbf{K}$ et $\mathbf{V}$ découlent de l'application de poids $\mathbf{W}_q$, $\mathbf{W}_k$ et $\mathbf{W}_v$ sur les données $\mathbf{X}$
  - Division par $\sqrt{d_k}$ pour stabiliser le gradient (d_k : taille des clés $\mathbf{K}$)
- Chaque tête travaille en parallèle avec ses propres poids $\mathbf{W}_q$, $\mathbf{W}_k$ et $\mathbf{W}_v$

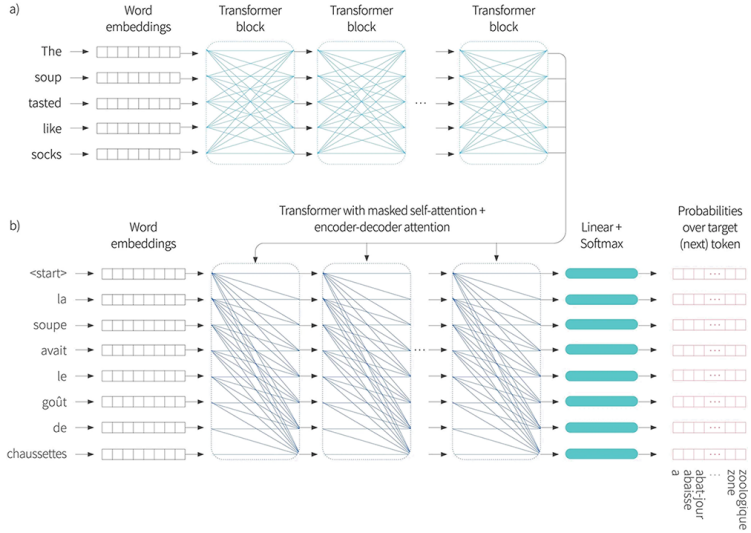
Mécanisme d'attention



Mécanisme d'attention



Exemple d'application en traduction



Source : `https://www.borealisai.com/wp-content/uploads/2021/06/T14_10.png`, accédé le 1er novembre 2023.

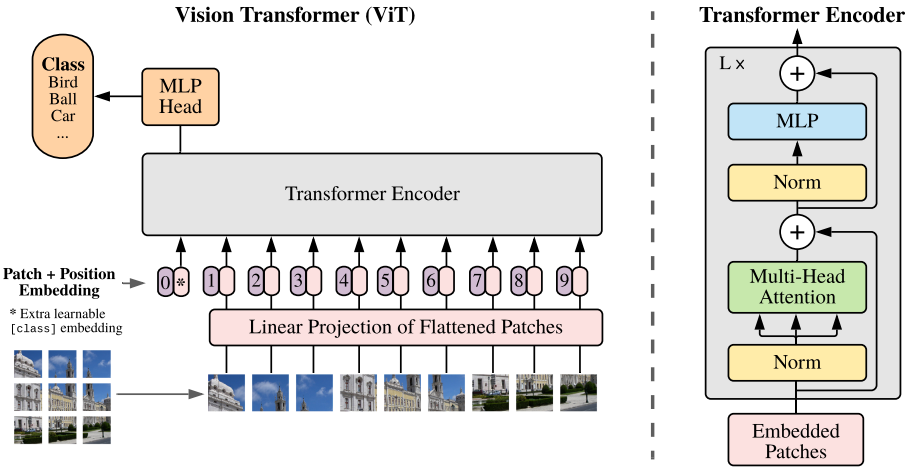
Grands modèles de langue

- BERT (*Bidirectional Encoder Representations from Transformers*)
  - Proposé en 2018 par une équipe de chercheurs de Google
  - Formé d'un module de plongement lexical, plusieurs couches d'encodeurs autoattentifs, et d'une conversion vers sortie probabiliste
  - Rapidement devenu un modèle central en traitement de la langue naturelle
  - Depuis 2020, pratiquement toutes les requêtes de recherche en anglais sur Google sont traitées avec BERT
- GPT (*Generative Pre-trained Transformer*)
  - Famille de modèles d'OpenAI basés sur des transformeurs, proposée en 2018 (GPT-1)
  - GPT-3 : GPT-1 + normalisation modifiée (GPT-2) + mise à l'échelle, proposé en 2020, 175G de paramètres entraîné sur 500G tokens
  - GPT-4 : architecture non dévoilée mais estimé à 1,7T (1700G) paramètres
  - ChatGPT : intégration de GPT-3.5 / GPT-4 et apprentissage par renforcement avec rétroaction humaine (RLHF)

Vision transformers (ViT)

- Vision transformers (ViT) : adaptation de l'architecture de réseaux autoattentifs à la vision numérique
  - Au lieu de traiter des séquences de mots, traite des tuiles de taille fixe, sans recoupement, de l'image
  - Chaque tuile est représentée par un vecteur 1D, avec information positionnelle ajoutée à la représentation
  - Représentation des tuiles fournie comme une séquence au réseau autoattentif
- Caractéristiques des ViT
  - En mesure de capturer des relations complexes et distantes dans les images, sans requérir des couches de convolution
  - Peut obtenir des performances au niveau de l'état de l'art, avec suffisamment de données et de ressources
  - Requiert de très gros jeux de données et entraînement intensif pour bien performer

Vision transformers (ViT)



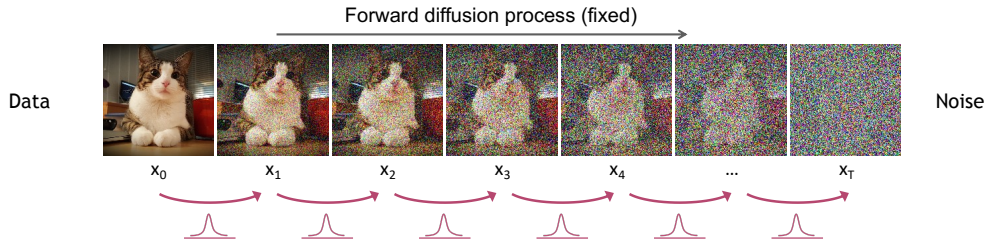
Tiré de *Dosovitskiy et coll., An Image is Worth 16x16 Words : Transformers for Image Recognition at Scale, ICLR, 2020.* Accédé en ligne le 2 novembre 2023 au `https://arxiv.org/pdf/2010.11929.pdf`.

10.7 Modèles de diffusion

Modèles de diffusion

- Modèles de diffusion : classe de modèles génératifs simulant un processus de diffusion aléatoire transformant une instance de données en instance de bruit
  - Inspirés par la physique, avec diffusion de particules d'un milieu de hautes concentrations vers de basses concentrations
  - Utilisés pour la génération, débruitage ou reconstruction d'images
- Processus de diffusion
  - Diffusion avant : démarrer par une image nette sur laquelle on ajout successivement un bruit léger jusqu'à ce que l'image ne soit que du bruit
  - Diffusion arrière : démarrer le processus d'une image de pur bruit sur laquelle ont applique successivement des opérations de débruitage pour obtenir une image nette
  - Chaque étape de diffusion avant ou arrière guidé par une fonction de transition, typiquement gaussienne, conditionnée sur l'état actuel
  - Une fois la mécanique de diffusion arrière (débruitage) bien apprise, elle peut être utilisée pour générer des nouvelles images

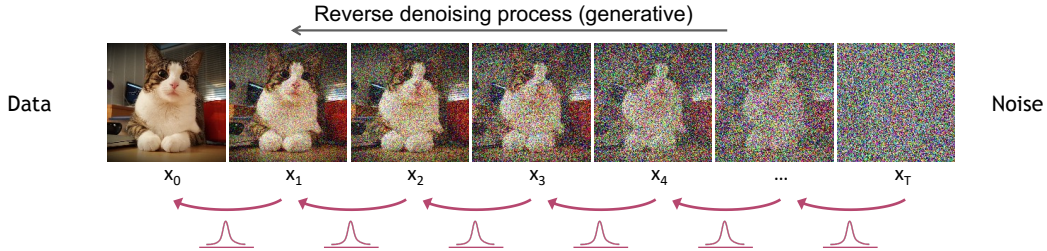
Processus de diffusion avant



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

Tiré de Song, Meng et Vahdat, *Denoising Diffusion Models : A Generative Learning Big Bang*, CVPR 2023 tutorial, `https://cvpr2023-tutorial-diffusion-models.github.io/`, accédé le 2 novembre 2023.

Processus de diffusion arrière



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}, \sigma_t^2\mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Tiré de Song, Meng et Vahdat, *Denoising Diffusion Models : A Generative Learning Big Bang*, CVPR 2023 tutorial, https://cvpr2023-tutorial-diffusion-models.github.io/, accédé le 2 novembre 2023.

Entraînement de modèles de diffusion

- Diffusion avant : typiquement consiste en l'application d'un bruit gaussien sur les pixels
  - Application répétée d'un faible bruit gaussien transforme l'ensemble des pixels en valeurs aléatoires de distribution normale
  - Niveau de bruit appliqué peut varier dans la séquence selon une cédule
- Diffusion arrière : réseau de neurones pour retirer le bruit
  - Utiliser données de la diffusion avant pour entraîner le réseau de débruitage
  - Réseau de débruitage reçoit niveau de bruit actuel
  - U-Net couramment utilisés comme réseaux de débruitage
- Le processus de diffusion arrière peut être conditionné
  - Classe spécifique visée
  - Requête texte, en utilisant représentation vectorielle de celle-ci (plongement lexical ou réseau autoattentif)

Forces et faiblesse des modèles de diffusion

- Forces des modèles de diffusion
  - Capables de générer des données de haute qualité
  - Flexibles et peut être adapté à différents types de données, fonctionne avec des distributions de données complexes
- Faiblesses des modèles de diffusion
  - Processus de génération peut être lourd en termes computationnels, avec les nombreuses itérations requises dans la diffusion arrière
  - Entraînement est très lourd en calculs
- Ces modèles sont à la base des modèles génératifs d'images comme DALL-E (OpenAI), Midjourney ou Stable Diffusion