Une introduction aux sciences des données

Gabriel Peyré
CNRS & DMA, École Normale Supérieure, PSL

CHAPITRE 5

Réseaux de neurones

9 septembre 2026

Les réseaux de neurones apprennent des transformations à partir d'exemples. Ce chapitre explique comment composer et entraîner leurs couches, puis comment un générateur peut apprendre à reproduire une distribution de données. Le lien avec le transport optimal éclaire les objectifs de cet apprentissage et ses limites.

5.1 Réseaux discriminatifs

5.1.1 Une fonction construite par couches

Un réseau de neurones est une fonction paramétrée f_w. À partir d'une entrée x, par exemple une image, il calcule une sortie $\hat{y} = f_w(x)$. Pour la classification, la sortie peut représenter des probabilités estimées de plusieurs classes. Une probabilité élevée exprime la confiance du modèle ; elle ne garantit ni que cette confiance soit bien calibrée, ni que la prédiction soit correcte.

Supposons d'abord qu'une image appartienne à exactement l'une de deux classes : chat ou chien. Un classifieur peut renvoyer deux nombres positifs ou nuls de somme un. Cela diffère d'une tâche à étiquettes multiples, où une image peut contenir à la fois un chat et un chien : dans ce cas, les probabilités séparées n'ont pas à sommer à un.

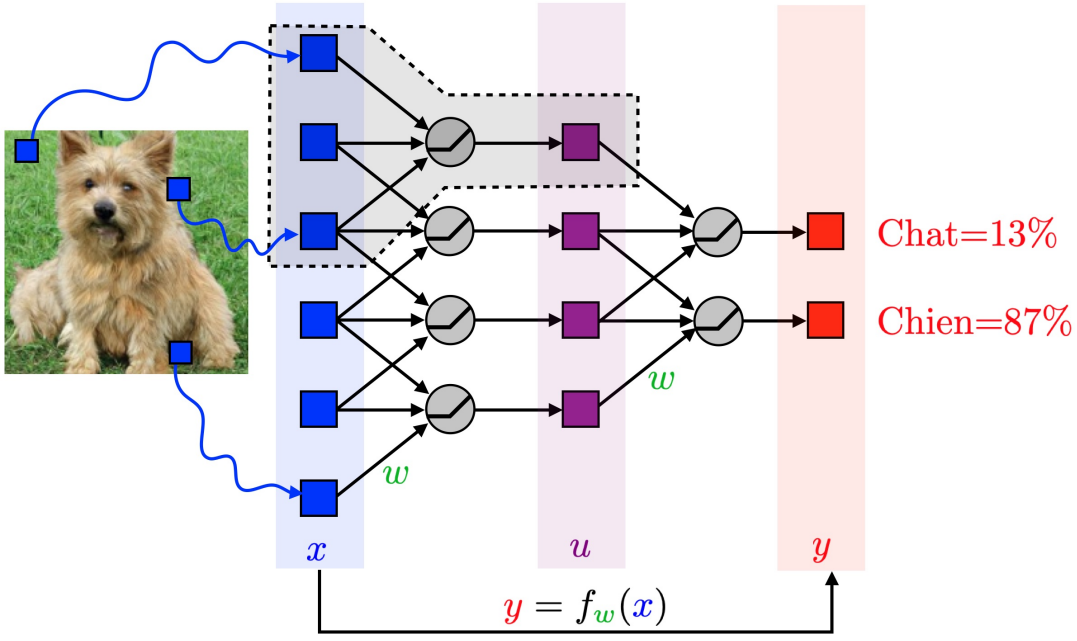

FIGURE 5.1. Schéma d'un classifieur avec une couche cachée et une couche de sortie. Seuls quelques pixels d'entrée et quelques connexions sont dessinés. Les pourcentages sont des prédictions du modèle ; une normalisation telle que softmax est nécessaire pour obtenir des probabilités de classes.

Chaque couche associe des opérations affines à une fonction d'activation. Une unité linéaire rectifiée, ou ReLU, calcule

$$u_1 = \mathrm{ReLU}(w_1 x_1 + w_2 x_2 + w_3 x_3 + b), \qquad \mathrm{ReLU}(t) = \max(t, 0). \tag{5.1}$$

Les poids déterminent la contribution de chaque entrée ; le biais b déplace le seuil d'activation. Une ReLU renvoie zéro pour une entrée négative ou nulle et conserve les entrées positives. Elle diffère d'un neurone à seuil binaire, qui n'a que deux valeurs de sortie possibles.

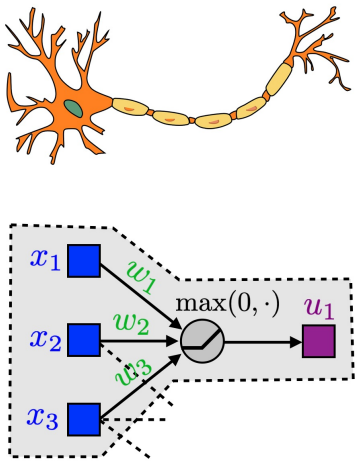

Figure 5.2. Un neurone biologique et un neurone artificiel schématique. Le modèle mathématique ne conserve qu'une somme pondérée et une activation ; le biais de (5.1) est omis du dessin.

Pour une entrée $x \in \mathbb{R}^d$, une couche cachée de largeur m et K classes, on peut définir le réseau

$$u = \mathrm{ReLU}(Wx + b), \qquad a = Vu + c, \qquad \hat{y}_k = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}},$$

où $W \in \mathbb{R}^{m \times d}$, $b \in \mathbb{R}^m$, $V \in \mathbb{R}^{K \times m}$ et $c \in \mathbb{R}^K$. ReLU agit coordonnée par coordonnée. L'opération finale est softmax ; les valeurs intermédiaires a_k sont appelées *logits*. En calcul numérique, remplacer chaque a_k par $a_k - \max_j a_j$ donne les mêmes probabilités et évite les débordements. Les entrées de W, b, V, c forment le vecteur de paramètres w.

Composer davantage de couches donne un réseau profond. Les activations non linéaires sont essentielles : une composition d'applications affines reste affine. La profondeur permet de représenter certaines fonctions plus économiquement qu'une seule couche cachée très large, mais l'architecture appropriée dépend de la tâche.

Le vocabulaire biologique exprime une analogie, et non un modèle fidèle du cerveau. McCulloch et Pitts ont étudié des neurones logiques à seuil [12], tandis que le perceptron de Rosenblatt a introduit un premier modèle d'apprentissage [14]. Ces modèles se distinguent des réseaux ReLU décrits ici. Les études biologiques du cortex visuel et de la dynamique membranaire portent sur des mécanismes bien plus riches [6, 7].

5.1.2 Apprendre à partir d'exemples étiquetés

L'apprentissage supervisé part de couples $(x_i, y_i)_{i=1}^n$. Pour une étiquette de classe unique k_i, la cible est souvent un vecteur indicateur : $(y_i)_{k_i} = 1$, et toutes les autres composantes sont nulles. Ces étiquettes représentent des annotations, plutôt que des probabilités conditionnelles connues. Les données peuvent provenir d'annotations humaines ou d'une autre source de supervision, et contenir des erreurs.

L'objectif d'apprentissage mesure le désaccord entre prédictions et étiquettes. L'erreur quadratique moyenne et l'entropie croisée en sont deux exemples :

$$E_{\mathrm{sq}}(w) = \frac{1}{n} \sum_{i=1}^{n} \| f_w(x_i) - y_i \|_2^2, \tag{5.2}$$

$$E_{\mathrm{CE}}(w) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} (y_i)_k \log(f_w(x_i)_k). \tag{5.3}$$

Figure 5.3. Exemples d'images étiquetées de chiens et de chats issues d'ImageNet [4]. L'apprentissage utilise de nombreux exemples ; l'évaluation demande des données séparées.

Pour une cible indicatrice, le terme d'entropie croisée correspondant vaut $-\log(f_w(x_i)_{k_i})$. Le logarithme est naturel, et softmax donne des probabilités strictement positives pour des logits finis. Une implémentation stable évalue directement le logarithme de softmax. On peut ajouter à l'objectif une pénalité sur les poids.

L'entraînement cherche à réduire cette erreur empirique. Comme le réseau dépend non linéairement de nombreux paramètres, l'objectif est généralement non convexe. Les méthodes d'optimisation peuvent être efficaces sans fournir de garantie générale d'atteindre son minimum global.

5.1.3 Rétropropagation et mise à jour des poids

La rétropropagation calcule le gradient d'une perte scalaire en appliquant la règle de dérivation des fonctions composées à travers le réseau, dans l'ordre inverse [11, 15]. C'est la différentiation automatique en mode inverse. Un algorithme d'optimisation utilise ensuite ces dérivées pour mettre à jour les poids ; par exemple, la descente de gradient effectue le pas

$$w^{(t+1)} = w^{(t)} - \eta_t \nabla E(w^{(t)}), \qquad \eta_t > 0.$$

Le choix du pas est important : un pas trop grand peut augmenter l'erreur. La descente de gradient stochastique remplace le gradient complet par une estimation calculée sur un petit lot choisi aléatoirement, ce qui réduit le coût de chaque mise à jour. Un pas stochastique individuel ne diminue pas nécessairement l'erreur totale d'apprentissage.

Pour le classifieur à deux couches ci-dessus et un exemple de perte ℓ donnée par l'entropie croisée, la dérivation se lit directement. Posons $r = Wx + b$, $u = \mathrm{ReLU}(r)$ et $\delta = \hat{y} - y$. En dehors des points où $r_j = 0$,

$$\frac{\partial \ell}{\partial V} = \delta u^{\mathsf{T}}, \qquad \frac{\partial \ell}{\partial c} = \delta, \qquad q = (V^{\mathsf{T}}\delta) \odot \mathbf{1}_{r>0}, \qquad \frac{\partial \ell}{\partial W} = q x^{\mathsf{T}}, \quad \frac{\partial \ell}{\partial b} = q.$$

Le symbole $\odot$ désigne le produit coordonnée par coordonnée. Au point anguleux de ReLU, les logiciels choisissent une convention, souvent zéro, pour la dérivée de l'activation. Cette convention ne rend pas le réseau différentiable au sens classique en ce point.

De grandes bases étiquetées, le calcul efficace sur processeurs graphiques et les progrès des architectures et de l'entraînement ont contribué au succès des classifieurs profonds, notamment au résultat ImageNet de 2012 [8]. Des techniques de régularisation comme le *dropout* peuvent limiter le surapprentissage [16]. Les travaux de Yoshua Bengio, Geoffrey Hinton et Yann LeCun ont été distingués par le prix Turing 2018 de l'ACM, annoncé en 2019. [1]

5.1.4 Approximation, optimisation et généralisation

Un théorème d'approximation universelle porte sur ce qu'un réseau peut représenter. Un résultat classique de Cybenko [3] affirme que, pour une fonction continue $F : [0,1]^d \to \mathbb{R}$ et tout $\varepsilon > 0$, il existe un entier fini m et des coefficients tels que

$$G(x) = \sum_{j=1}^{m} \alpha_j \sigma(v_j \cdot x + \beta_j), \qquad \sup_{x \in [0,1]^d} |F(x) - G(x)| < \varepsilon,$$

1. https://awards.acm.org/about/2018-turing

où σ est une activation sigmoïde continue fixée, par exemple $\sigma(t) = 1/(1 + e^{-t})$. La couche de sortie est linéaire, et la largeur et les poids ne sont pas contraints. Il s'agit du théorème pour les sigmoïdes ; les résultats d'approximation pour les réseaux ReLU demandent leurs propres énoncés.

Ce théorème ne fait intervenir aucune donnée d'apprentissage. Il n'exige pas une infinité d'exemples et ne garantit pas qu'un algorithme puisse trouver le réseau approchant à partir d'un nombre fini d'observations. Trois questions doivent être distinguées : l'existence d'une bonne représentation, la capacité de l'optimisation à la trouver et la qualité des prédictions sur de nouvelles données.

La généralisation s'évalue sur des données exclues de l'entraînement. Les données de validation guident des choix comme l'architecture et le moment d'arrêt ; un jeu de test distinct évalue la procédure finale. La qualité des résultats dépend aussi de la représentativité de ces données par rapport aux conditions d'utilisation du réseau.

Les réseaux convolutifs intègrent des hypothèses sur la structure spatiale [9]. Les connexions locales limitent le champ réceptif, et le partage des poids applique le même filtre en différentes positions. Une convolution de pas un commute avec les translations sur une grille infinie ou périodique ; les conditions au bord et le sous-échantillonnage peuvent modifier cette propriété. L'équivariance par translation d'une carte de caractéristiques se distingue également de l'invariance par translation d'une classification finale.

L'architecture, la régularisation, les données et l'optimisation influencent toutes les performances [10]. Des résultats mathématiques expliquent certains mécanismes et certains régimes, mais aucun théorème unique ne rend compte de toutes les architectures efficaces.

5.2 Réseaux génératifs

5.2.1 Des entrées aléatoires à une distribution

Un générateur g_w transforme une entrée aléatoire $Y \in \mathbb{R}^r$ en une image $X = g_w(Y) \in \mathbb{R}^d$. La loi d'entrée η, par exemple une gaussienne standard, est choisie à l'avance. La loi de sortie est la mesure image $\mu_w = (g_w)_{\#}\eta$. Cette notation décrit la redistribution de la masse de probabilité par le réseau. Un générateur est une transformation apprise, qui n'est généralement pas l'inverse d'un classifieur.

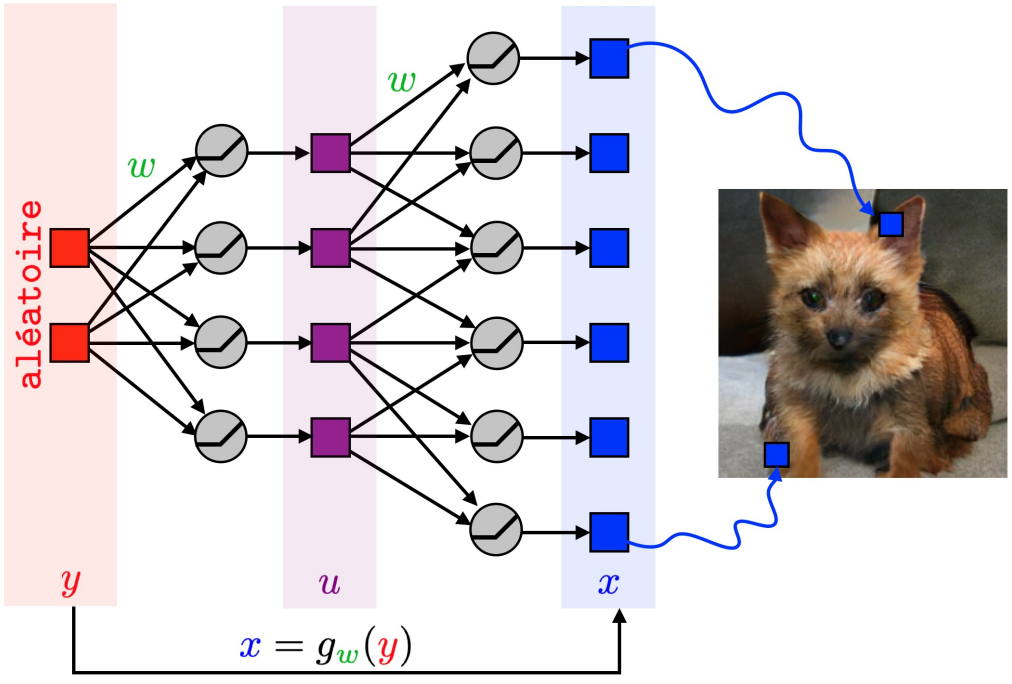

Figure 5.4. Un générateur schématique transforme une entrée aléatoire y, en passant par une couche cachée, en une image $x = g_w(y)$. Les générateurs d'images utilisés en pratique ont davantage de couches et une dimension de sortie bien plus grande.

Pour la génération non conditionnelle, l'apprentissage peut utiliser des images sans étiquette de classe. Leur collecte, leur sélection et leur prétraitement demandent toujours de l'attention. La génération conditionnelle

fournit en outre une information telle qu'une étiquette de classe, donnant un modèle $g_w(y, c)$. Tous les modèles génératifs ne sont donc pas entraînés sans supervision.

Le but est de reproduire une distribution, y compris sa diversité. Si la distribution de référence attribue une probabilité $1/2$ aux chats et $1/2$ aux chiens, un générateur idéal possède aussi ces probabilités. Un lot aléatoire fini ne contient pas nécessairement exactement la moitié de chaque catégorie. Un modèle produisant toujours la même image plausible n'a pas appris toute la distribution ; c'est une forme extrême d'effondrement des modes.

5.2.2 Comparer des distributions empiriques

Notons $z_1, \dots, z_n$ les images d'apprentissage et $x_i = g_w(y_i)$ les images générées à partir d'échantillons latents indépendants. Leurs distributions empiriques sont

$$\mu_{w,n} = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}, \qquad \nu_n = \frac{1}{n} \sum_{j=1}^{n} \delta_{z_j}.$$

L'emploi de mesures conserve les fréquences et les répétitions d'échantillons, que la notation ensembliste ordinaire effacerait. Un objectif possible est une distance de transport entre ces mesures.

Pour un coût euclidien, la distance de Wasserstein empirique d'ordre un vaut

$$W_1(\mu_{w,n}, \nu_n) = \frac{1}{n} \min_{s \in \Sigma_n} \sum_{i=1}^{n} \| g_w(y_i) - z_{s(i)} \|_2. \tag{5.4}$$

Le facteur $1/n$ normalise la masse totale de probabilité. Cette distance diffère du coût quadratique introduit au chapitre sur le transport. Pour des nombres d'échantillons ou des poids différents, on remplace la permutation par un couplage aux marginales prescrites.

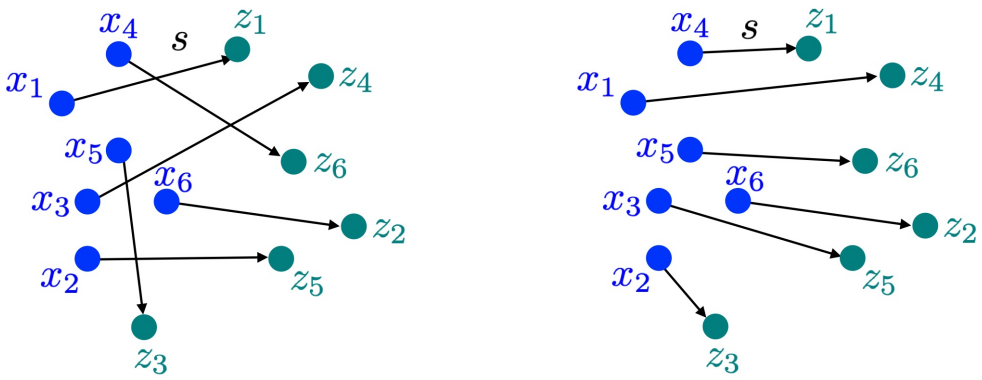

Figure 5.5. Deux affectations de six points sources à six points cibles. L'affectation de droite a un coût de transport euclidien plus faible et constitue l'appariement optimal de cet exemple.

À la figure 5.5, l'affectation de gauche est $(1, 5, 4, 6, 3, 2)$, tandis que celle de droite est $(4, 3, 5, 1, 6, 2)$. La notation donne la liste $(s(1), \dots, s(6))$. Bien qu'il y ait $6! = 720$ possibilités, des algorithmes d'affectation efficaces évitent l'énumération des $n!$ permutations et résolvent un problème dense en $O(n^3)$ opérations. Le chapitre sur le transport explique la relaxation convexe sous-jacente [13].

L'objectif d'apprentissage distributionnel est $\min_w W_1(\mu_w, \nu)$ pour une loi cible ν. Le transport entre échantillons finis en fournit une approximation numérique, mais soulève des difficultés statistiques et d'optimisation. Le transport entre petits lots est généralement un estimateur biaisé de la distance entre les lois. L'affectation optimale peut changer avec w, de sorte que l'objectif empirique n'est pas partout différentiable.

La distance de base joue aussi un rôle essentiel. La distance euclidienne entre tableaux de pixels est sensible aux petites translations et peut mal refléter la similarité sémantique. Une représentation par caractéristiques peut fournir un coût plus pertinent, mais son choix modifie la géométrie apprise.

5.2.3 Réseaux génératifs antagonistes

Les réseaux génératifs antagonistes, ou GAN, introduits par Goodfellow et ses collaborateurs en 2014 [5], entraînent un générateur face à un discriminateur. Le discriminateur $D_v(x) \in (0, 1)$ cherche à distinguer les

données des échantillons générés. L'objectif minimax original est

$$\min_{w} \max_{v} \left\{ \mathbb{E}_{Z\sim\nu}[\log D_v(Z)] + \mathbb{E}_{Y\sim\eta}[\log(1 - D_v(g_w(Y)))] \right\}. \tag{5.5}$$

Le discriminateur augmente l'objectif ; le générateur le diminue. Des mises à jour de gradient alternées approchent ce jeu, sans garantie générale de convergence pour des réseaux arbitraires.

Pour un discriminateur sans restriction et optimisé exactement, le supremum intérieur vaut $-\log 4 + 2\,\mathrm{JS}(\nu, \mu_w)$, où la divergence de Jensen–Shannon est

$$\mathrm{JS}(\nu, \mu) = \tfrac{1}{2}\,\mathrm{KL}(\nu \mid m) + \tfrac{1}{2}\,\mathrm{KL}(\mu \mid m), \qquad m = \tfrac{1}{2}(\nu + \mu).$$

Ici, KL utilise les logarithmes naturels et se définit entre mesures à l'aide de leurs densités de Radon–Nikodym. Les deux mesures sont absolument continues par rapport à m. L'identité concerne l'optimum idéal du discriminateur ; des réseaux finis et une optimisation incomplète modifient l'objectif effectivement atteint. En pratique, on utilise souvent pour le générateur la perte $-\mathbb{E}\log D_v(g_w(Y))$, qui vise la même distribution optimale mais donne des gradients différents.

5.2.4 Le lien avec Wasserstein

Les GAN de Wasserstein [1] emploient un autre objectif, fondé sur la formule duale de Kantorovitch–Rubinstein. Pour des mesures de probabilité sur $\mathbb{R}^d$ à moments d'ordre un finis,

$$W_1(\nu, \mu_w) = \sup_{\mathrm{Lip}(h)\le 1} \left\{ \mathbb{E}_{Z\sim\nu} h(Z) - \mathbb{E}_{Y\sim\eta} h(g_w(Y)) \right\}. \tag{5.6}$$

La contrainte signifie $|h(x) - h(x')| \le \|x - x'\|_2$ pour tous x, x'. Un réseau critique approche cette classe de fonctions ; ses valeurs sont des scores réels, et non des probabilités de classes. Le générateur minimise l'écart estimé par le critique.

Le critique est un potentiel dual pour une distance de base *fixée* ; il n'apprend pas automatiquement un nouveau coût de transport entre paires d'images. Imposer ou approcher la contrainte lipschitzienne est essentiel. La restriction de la classe des critiques et leur optimisation approchée signifient aussi que le score calculé n'est pas nécessairement égal à la véritable distance de Wasserstein.

Les GAN ordinaires et les GAN de Wasserstein expriment ainsi des principes antagonistes proches, avec des objectifs mathématiques différents. Aucune des deux procédures ne garantit que toute image générée soit pertinente ou que tous les modes soient représentés.

Figure 5.6. Deux suites d'images synthétiques interpolant entre chats et chiens, obtenues avec BigGAN [2]. BigGAN est conditionné par les classes : cette information fait partie du conditionnement. Ces exemples illustrent la génération et l'interpolation, sans affirmer qu'il s'agit de chemins de transport optimal.

Interpoler les variables latentes ou de conditionnement peut produire des transitions visuelles régulières, comme à la figure 5.6. Ces chemins ne sont pas nécessairement des géodésiques de transport et ne restent pas toujours typiques de la distribution d'apprentissage. La plausibilité visuelle, la diversité, la fidélité à la loi cible et les résultats sur des données réservées sont des aspects distincts de la qualité d'un modèle génératif.

Remerciements. Je remercie Gwenn Guichaoua pour sa relecture attentive, ainsi que Sébastien Racanière et Vincent Barra pour leurs corrections.

Références

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, 2017. URL https://proceedings.mlr.press/v70/arjovsky17a.html.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1809.11096.

[3] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4) :303–314, 1989.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet : A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[6] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4) :500–544, 1952.

[7] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1) :106–154, 1962.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553) :436–444, 2015.

[11] Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16 (2) :146–160, 1976.

[12] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4) :115–133, 1943.

[13] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5–6) :355–607, 2019. URL https://arxiv.org/abs/1803.00567.

[14] Frank Rosenblatt. The perceptron : A perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, 1957.

[15] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088) :533–536, 1986.

[16] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1) : 1929–1958, 2014.