Analyse théorique des réseaux de neurones Transformers

Valérie Castin

Introduction au domaine de recherche, mai 2024

1 Introduction

L'apprentissage automatique (machine learning) est un domaine à l'intersection des mathématiques et de l'informatique, qui vise à développer des modèles capables d'exécuter une grande variété de tâches en "apprenant" à partir d'un ensemble de données. Les applications des modèles d'apprentissage, ou réseaux de neurones, incluent la vision par ordinateur (reconnaissance et classification d'images ou vidéos [39, 25], segmentation [68]), le traitement du langage naturel (traduction de textes [63, 22], analyse syntaxique et classification [24, 41]), le traitement audio [16], la prise de décision d'un agent artificiel dans un environnement [36, 11], et toutes les tâches d'intelligence artificielle générative, du développement d'agents conversationnels comme ChatGPT [51] à la génération d'images [30, 54] ou de vidéos hyper-réalistes [42].

L'apprentissage automatique connaît un essor remarquable à partir des années 2010, avec l'émergence des données massives et l'augmentation des capacités de calcul, qui permettent une amélioration spectaculaire de la performance des modèles [23, 39, 34, 59]. De nouvelles architectures de modèles se développent, le perceptron multicouche [3, 35] laissant la place aux réseaux de neurones convolutionnels [39] puis aux Transformers, introduits en 2017 par Vaswani et al. [63], sur lesquels nous allons nous concentrer dans ce mémoire. Avant de présenter en détail l'architecture des Transformers et les problèmes mathématiques que le fonctionnement de ces réseaux de neurones soulève, rappelons brièvement le cadre général de l'apprentissage supervisé.

Le paradigme de l'apprentissage supervisé. L'apprentissage supervisé consiste à optimiser les paramètres d'un modèle mathématique paramétrique pour mener à bien une tâche fixée. Le modèle prend en entrée des données (images, texte, audio) et leur associe en général un réel, qui donne une information sur ces données (par exemple, l'image contient-elle un visage ?). L'optimisation des paramètres se fait à l'aide d'un jeu de données d'entraînement, pour lesquelles on a associé manuellement à chaque donnée la réponse attendue, appelée *étiquette*. Il s'agit alors d'ajuster les paramètres du modèle pour que la fonction qu'il encode coïncide de façon satisfaisante sur les données d'entraînement avec la valeur de l'étiquette. Présentons le cas d'une tâche de classification binaire : on veut trier des données en deux catégories, par exemple distinguer des images de chien et de chat. On dispose de données d'entraînement étiquetées $(x_j, \varepsilon_j)_{1\leq j\leq N}$, avec $x_j \in \mathbb{R}^d$ et $\varepsilon_j \in \{0,1\}$. Les x_j correspondent à différents points de données, par exemple des images représentées par le vecteur de leurs pixels, et les ϵ_j donnent la classe de l'image x_j (donc chien ou chat). On choisit alors un modèle d'apprentissage particulier, c'est-à-dire une famille paramétrique de fonctions $f_\theta \colon \mathbb{R}^d \to \{0,1\}$ paramétrées par un vecteur de paramètres $\theta \in \mathbb{R}^p$, et l'on utilise un algorithme d'optimisation pour trouver des paramètres θ^* minimisant une *fonction de coût* $\mathcal{L}$ qui mesure l'écart entre les prédictions $(f_{\theta^*}(x_j))_{1\leq j\leq N}$ et les étiquettes

$(\varepsilon_j)_{1\leq j\leq N}$. Une fonction de coût classique est l'écart quadratique régularisé

$$\mathcal{L}(f_\theta, (x_j, \varepsilon_j)_{1\leq j\leq N}) \coloneqq \frac{1}{N}\sum_{j=1}^{N}(f_\theta(x_j) - \varepsilon_j)^2 + |\theta|^2\,,$$

où $|\theta|$ désigne la norme euclidienne de θ. La minimisation en θ de $\mathcal{L}(f_\theta, (x_j, \varepsilon_j)_{1\leq j\leq N})$ se fait par un algorithme dérivé de la descente de gradient. Nous renvoyons vers le livre de Bach et Chizat [6] pour plus de précisions sur les questions d'optimisation en apprentissage automatique. La phase d'optimisation de la fonction de coût est appelée *entraînement* du réseau. Les succès empiriques des dix dernières années [39, 33, 63] ont montré que lorsque l'on dispose d'une très grande quantité de données et que le modèle a une quantité encore plus importante de paramètres, les paramètres appris θ^* ont de bonnes propriétés de *généralisation*, c'est-à-dire que si $(x_{N+1}, \varepsilon_{N+1})$ est une nouvelle donnée qui n'appartient pas au jeu de données d'entraînement, la prédiction $f_{\theta^*}(x_{N+1})$ sera proche de ε_{N+1} en moyenne. L'étude statistique de la performance de f_{θ^*} sur de nouvelles données fait l'objet de tout un pan de la théorie de l'apprentissage, appelée apprentissage statistique. Nous renvoyons au cours de Sylvain Arlot [5] pour une introduction complète aux principaux enjeux de ce domaine.

Les Transformers et l'auto-attention. Les Transformers sont des modèles d'apprentissage très performants, introduits en 2017 par Vaswani et al. [63], et qui constituent aujourd'hui l'état de l'art dans la plupart des tâches de traitement du langage naturel [18, 65, 1] comme de vision par ordinateur [25] ou de traitement audio [16, 19]. L'architecture des Transformers a renouvelé le paradigme traditionnel d'apprentissage en représentant chaque donnée (image, phrase, enregistrement audio) non plus par un seul vecteur $x \in \mathbb{R}^d$ mais par une suite de vecteurs $(x_1, \ldots, x_n) \in (\mathbb{R}^d)^n$ de longueur n fixe (pour le traitement d'images) ou variable (pour le traitement de texte ou d'audio), appelés *tokens*. Les x_i sont typiquement des fragments de l'image x initiale, ou des fragments de mots de la phrase initiale – nous renvoyons à la Section 2 pour plus de détails. Cette représentation multivectorielle est exploitée par une partie de l'architecture des Transformers appelée *auto-attention*, qui permet de modéliser les dépendances entre les tokens, donc entre les différentes parties d'une image ou les différents mots d'une phrase. Le fonction d'auto-attention a pour paramètres deux matrices $A, V \in \mathbb{R}^{d\times d}$ et est définie de la façon suivante :

$$f\colon \cup_{n\geq 1}(\mathbb{R}^d)^n \to \cup_{n\geq 1}(\mathbb{R}^d)^n, \quad X = (x_1, \ldots, x_n) \mapsto (f(X)_1, \ldots, f(X)_n) \tag{1}$$

où

$$f(X)_i \coloneqq V\sum_{j=1}^{n} P_{ij}x_j \quad \text{avec} \quad P_{ij} \coloneqq e^{x_i^\top A^\top x_j}/\sum_{k=1}^{n} e^{x_i^\top A^\top x_k}.$$

Les paramètres A et V sont appris lors de l'entraînement, de sorte que les produits scalaires $x_i^\top A^\top x_j$ reflètent les liens existant entre les tokens (liens sémantiques par exemple, pour des données textuelles). Les vecteurs de produits scalaires $(x_i^\top A^\top x_j)_j$ pour $1 \leq i \leq n$ sont ensuite transformés en vecteurs du simplexe $\Sigma_n \coloneqq \{a \in [0,1]^n : \sum_{i=1}^n a_i = 1\}$ par la fonction softmax, définie comme suit :

$$\text{softmax}\colon \begin{cases} \mathbb{R}^n \to \mathbb{R}^n \\ (w_1, \ldots, w_n) \mapsto \left(\frac{\exp(w_i)}{\sum_{j=1}^d \exp(w_j)}\right)_{1\leq i\leq n} \end{cases}. \tag{2}$$

Nous reviendrons sur la définition de l'attention dans la Section 2. L'attention est véritablement centrale dans l'architecture des Transformers, car c'est la seule composante du modèle qui fait

interagir les tokens entre eux. Les autres transformations appliquées par le modèle se font en parallèle sur les différents tokens, et sont constituées d'une succession d'applications linéaires et d'une non-linéarité fixée appliquée coordonnée par coordonnée. Notons enfin que la structure d'un Transformer est organisée par couches : on compose la même fonction (qui constitue une couche) un grand nombre de fois, avec des paramètres différents, ce qui augmente l'*expressivité* du modèle, c'est-à-dire sa capacité à représenter des fonctions suffisamment complexes pour répondre à la tâche demandée. Par exemple, le Transformer GPT-4, qui est utilisé dans l'agent conversationnel ChatGPT [51], a 120 couches pour un total de 10^{12} paramètres.

Les principales questions ouvertes. L'architecture des Transformers, et en particulier la définition de la self-attention, sont le résultat d'avancées empiriques successives, à l'image de ce qui se fait en apprentissage profond. Cependant, de nombreuses questions restent ouvertes pour avoir une meilleure compréhension théorique de ces modèles, et sont l'objet d'un champ de recherche très actif.

- **Optimisation :** Peut-on justifier que l'entraînement des Transformers converge vers un "bon" choix de paramètres ? Cette question est difficile car le paysage d'optimisation de la fonction de coût d'un Transformer profond est fortement non convexe. Dans le cas des Perceptrons à une couche cachée, qui comptent parmi les premiers modèles à avoir été utilisés en apprentissage automatique [3, 35], des résultats ont été obtenus par Chizat et Bach [15]. Le cas des Transformers est compliqué par la présence de la fonction d'attention, et par son caractère de réseau profond, qui demande une analyse spécifique. Chen et al. [14] font une étude de convergence de l'optimisation dans un cadre simplifié, mettant en évidence différentes phases pendant l'entraînement.
- **Robustesse :** Les Transformers (entraînés) sont-ils robustes aux petites perturbations des données (bruit, attaques dites adversariales pour tromper le réseau) ? Autrement dit, peut-on faire varier beaucoup la sortie d'un réseau entraîné en perturbant très légèrement l'entrée, de façon aléatoire ou au contraire, bien choisie ? Des études empiriques suggèrent que les Transformers sont robustes au bruit naturel [52, 9] et aux perturbations adversariales dans une certaine mesure [45]. Les analyses théoriques sont en revanche encore incomplètes, se heurtant notamment à la difficulté d'analyser la robustesse de la composition d'un grand nombre de couches (de l'ordre de 10^2 pour GPT-4). Des progrès récents ont cependant été faits en ce qui concerne la robustesse du composant principal de l'architecture des Transformers, l'auto-attention [37, 10] (voir Section 3).
- **Parcimonie :** Peut-on alléger l'architecture des Transformers tout en préservant leur performance ? Les réseaux de neurones profonds sont en effet particulièrement énergivores à entraîner, stocker et évaluer, et l'algorithme d'optimisation utilisé, appelé Adam [38], est nettement plus complexe et coûteux computationnellement qu'une simple descente de gradient stochastique. De nombreuses modifications de l'architecture des Transformers ont été proposées pour pallier ces problèmes, par exemple une fonction d'auto-attention plus efficace à évaluer [66], ou des Transformers dont les matrices de paramètres sont parcimonieuses (i.e. ont seulement un petit nombre de coefficients non nuls) [58, 20].
- **Explicabilité :** Que "fait" un Transformer entraîné ? Cette question très générale consiste à expliquer théoriquement des phénomènes observés empiriquement sur les Transformers. On peut donner l'exemple des travaux de Geshkovski et al. [29, 28], qui expliquent théoriquement le regroupement des tokens en amas lorsqu'ils passent à travers un Transformer profond (voir Section 3). La branche d'étude de l'apprentissage contextuel (in-context learning) rentre également dans ce cadre, en s'intéressant aux mécanismes qui permettent au

FIGURE 1 – Visualisation des tokens avec la tokenisation de GPT-3 [50]. Les identifiants des tokens correspondants sont : $[1212, 318, 703, 402, 11571, 12, 18, 11241, 4340, 428, 6827, 13]$. On constate que le mot « tokenizes » ne figure pas dans l'alphabet des tokens, et qu'il est donc subdivisé en deux tokens.

FIGURE 2 – Tokenisation d'une image suivant Dosovitskiy et al. [25].

Transformer de compléter des suites logiques de paires entrée-sortie associées à une tâche donnée sur laquelle il n'a pas été entraîné [26, 56].

L'expressivité des Transformers, c'est-à-dire leur capacité à approcher des fonctions quelconques $(\mathbb{R}^d)^n \to (\mathbb{R}^d)^n$ à support compact, est en revanche bien comprise [70], même si leur capacité d'approximation vient essentiellement des perceptrons intégrés dans l'architecture (voir Section 2), plus que de l'auto-attention.

Penchons-nous à présent plus en détail sur l'architecture des Transformers.

2 L'architecture des Transformers

Il existe plusieurs variantes d'architectures Transformer, réparties en trois types : encodeur-décodeur, encodeur seul et décodeur seul. Concentrons-nous sur l'architecture de Transformer traditionnelle (encodeur-décodeur) telle qu'introduite dans l'article précurseur de Vaswani et al. [63]. Le Transformer convertit chaque donnée d'entrée en une suite finie de vecteurs, que nous appellerons *séquence d'entrée*, et renvoie une autre suite de vecteurs de même longueur, la *séquence de sortie*. En fonction de la tâche effectuée, cette séquence de sortie est traitée ultérieurement ; nous ne nous attarderons pas sur cette partie finale de l'architecture. L'architecture du Transformer se compose des blocs suivants (voir la Figure 4 pour une vue d'ensemble).

Tokenisation. Les données brutes sont d'abord converties en une séquence d'unités gérables $(s_1, \ldots, s_n)$, appelées tokens, appartenant à un *alphabet* de tokens adapté à la tâche à résoudre. Pour les tâches de traitement du langage, les tokens correspondent à des mots ou fractions de mots, ainsi qu'aux signes de ponctuation. Les débuts de phrase sont marqués par un token particulier, noté <bos> (beginning of sentence). La figure 1 montre un exemple de tokenisation d'une phrase par GPT-3 [50]. Dans les tâches de traitement d'images, l'image est divisée en morceaux de taille fixe (voir la figure 2), et chaque morceau est traité comme un token.

Plongement et encodage positionnel. Une fois créée, la séquence de tokens $(s_1, \ldots, s_n)$ est transformée en une séquence de vecteurs $(\tau_1, \ldots, \tau_n) \in (\mathbb{R}^d)^n$, via un encodage one-hot [1]

1. Le principe de l'encodage *one-hot* est le suivant. On choisit un ordre sur l'alphabet des tokens, puis chaque token est représenté par un vecteur de la forme $(0, 0, \ldots, 1, \ldots, 0)$ de dimension égale à la taille de l'alphabet, où

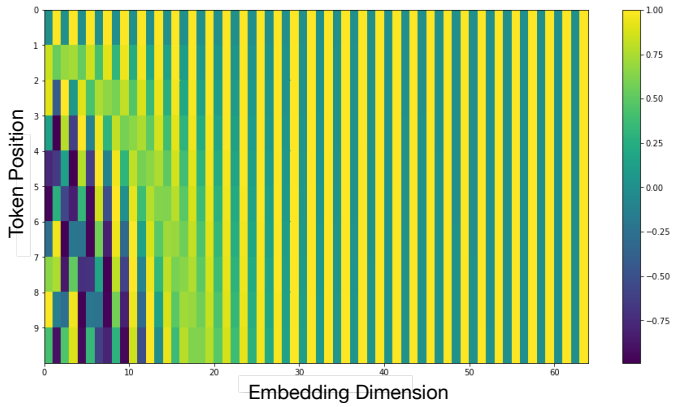

FIGURE 3 – Un exemple d'encodage positionnel pour 10 mots (lignes), avec une dimension de plongement de 64 (colonnes), tel que défini dans Vaswani et al. [63]. Figure tirée de Alammar [2].

si les tokens ne sont pas des vecteurs (par exemple en traitement du langage), puis avec un plongement linéaire. Ensuite, un n-uplet $(\pi_1, \ldots, \pi_n)$, appelé encodage positionnel, qui peut être appris ou fixé, est ajouté à $(\tau_1, \ldots, \tau_n)$, pour incorporer l'information de l'ordre des vecteurs τ_i dans la séquence d'entrée elle-même. Lorsqu'il n'est pas appris, l'encodage positionnel π est généralement défini à l'aide de fonctions sinusoïdales, par exemple :

$$\pi_i = \begin{pmatrix} \sin(\omega_1 i) \\ \cos(\omega_1 i) \\ \sin(\omega_2 i) \\ \cos(\omega_2 i) \\ \vdots \\ \sin(\omega_{d/2} i) \\ \cos(\omega_{d/2} i) \end{pmatrix},$$

avec $\omega_k = 10^{-8k/d}$ pour $1 \leq k \leq d/2$ dans l'article de Vaswani et al. [63] (voir Figure 3). L'idée de l'encodage positionnel est d'incorporer l'information de l'ordre des τ_i directement dans la séquence d'entrée, de sorte que le reste du modèle (en tout cas l'encodeur) puisse être *équivariant par permutation*. Une fonction $f\colon (\mathbb{R}^d)^n \to (\mathbb{R}^d)^n$ est dite équivariante par permutation si, pour toute permutation $\sigma \in \mathfrak{S}_n$, on a $f(x_{\sigma(1)}, \ldots, x_{\sigma(n)}) = (f(x_1, \ldots, x_n)_{\sigma(i)})_{1 \leq i \leq n}$. Le fait que l'architecture de l'encodeur soit équivariante par permutation accélère considérablement l'apprentissage.

Encodeur. Le résultat $(x_1, \ldots, x_n) := (\tau_1, \ldots, \tau_n) + (\pi_1, \ldots, \pi_n)$ de l'étape précédente passe ensuite à travers L couches avec la même structure, chacune composée de deux sous-couches : un bloc d'auto-attention multi-tête et un perceptron multi-couches (MLP) à une couche cachée. L'attention multi-tête a pour paramètres le nombre de têtes H (typiquement égal à 12), qui doit être un diviseur de la dimension d'entrée d, et des matrices $(A^{(h)}, V^{(h)}, W^{(h)})_{1 \leq h \leq H}$ avec

le 1 est situé à la position du token dans l'alphabet.

$A^{(h)} \in \mathbb{R}^{d\times d}$, $V^{(h)} \in \mathbb{R}^{k\times d}$ et $W^{(h)} \in \mathbb{R}^{d\times k}$, où $k \coloneqq d/H$. Elle est définie de la façon suivante :

$$f^{\mathrm{MH}} \coloneqq \sum_{h=1}^{H} W^{(h)} f^{(h)},$$

où $f^{(h)}$ est la fonction d'attention associée aux paramètres $A^{(h)}, V^{(h)}$ (voir Equation (1), et va donc de $(\mathbb{R}^d)^n$ dans $(\mathbb{R}^d)^n$. Le MLP à une couche cachée a pour paramètres $(U_1, U_2, b) \in \mathbb{R}^{\nu\times d} \times \mathbb{R}^{d'\times\nu} \times \mathbb{R}^{\nu}$, et est défini par

$$g\colon x \in \mathbb{R}^d \mapsto U_2\sigma\left(U_1 x + b\right)$$

où $\sigma\colon \mathbb{R} \to \mathbb{R}$ est une fonction non linéaire à choisir, par exemple la partie positive $x \mapsto \max(0, x)$. Il est appliqué en parallèle à chaque élément de la séquence.

Les deux sous-couches précédentes sont *résiduelles*, c'est-à-dire de la forme $X \mapsto X +$ module(X) pour $X = (x_1, \ldots, x_n)$, où module indique f^{MH} ou g, et sont suivies d'une opératio de *normalisation*, de la forme

$$\mathrm{LayerNorm}_{\alpha,\beta}\colon z \in \mathbb{R}^d \mapsto \frac{\alpha}{\sigma_z}(z - \mu_z) + \beta,$$

avec $\mu_z = \frac{1}{d}\sum_{j=1}^{d} z_j$ la moyenne des coordonnées de z, et $\sigma_z = \frac{1}{d}\sum_{j=1}^{d}(z_j - \mu_z)^2$ leur écart-type, et $\alpha \in \mathbb{R}$ et $\beta \in \mathbb{R}^d$, la normalisation LN étant appliquée en parallèle à chaque élément de la séquence. [2]

Remarque 2.1. *Les couches résiduelles, introduites dans les réseaux appelés ResNets [32], facilitent grandement l'optimisation des réseaux profonds, en évitant que le gradient tende vers 0 trop vite au cours de l'optimisation. Ce sujet est hors du champ de nos recherches. Notons cependant que les réseaux résiduels (i.e. constitués de couches résiduelles) peuvent être modélisés de façon naturelle par une équation différentielle ou une équation aux dérivées partielles dans la limite d'un nombre de couches infini, ce qui donne accès à des outils puissants pour leur analyse théorique (voir sous-section 3.2).*

Décodeur. Soit $(z_1, \ldots, z_n)$ la sortie de l'encodeur. Le décodeur se compose de L couches identiques. Sa structure est similaire à celle de l'encodeur, avec les différences suivantes. Premièrement, au lieu de traiter la séquence d'entrée comme le fait l'encodeur, le décodeur traite itérativement ses propres tokens de sortie décalés vers la droite, en commençant par un token de début de phrase (<bos>). Dans le cas d'une tâche de génération de texte, on peut voir ça comme la construction mot à mot de la réponse du modèle, chaque nouveau mot dépendant des précédents déjà renvoyés. En outre, l'auto-attention est appliquée de façon séquentielle : pour chaque token traité par le décodeur, l'auto-attention est réalisée sur ce token par rapport à la séquence d'entrée tronquée juste après le token en question. Ce processus est modélisé par ce que l'on appelle l'auto-attention masquée, définie comme suit :

$$f^m\colon X = (x_1, \ldots, x_n) \in (\mathbb{R}^d)^n \mapsto (f^m(X)_1, \ldots, f^m(X)_n) \in (\mathbb{R}^d)^n$$

où

$$f^m(X)_i \coloneqq f(x_1, \ldots, x_i)_i$$

avec f l'auto-attention classique [3]. Enfin, une couche d'attention dite *croisée* est appliquée. Nous n'en reparlerons pas dans ce qui suit ; nous renvoyons le lecteur ou la lectrice à l'article de Alammar [2] pour une définition illustrée.

2. Les architectures utilisées aujourd'hui appliquent plutôt la normalisation juste avant d'appliquer f^{MH} et g.

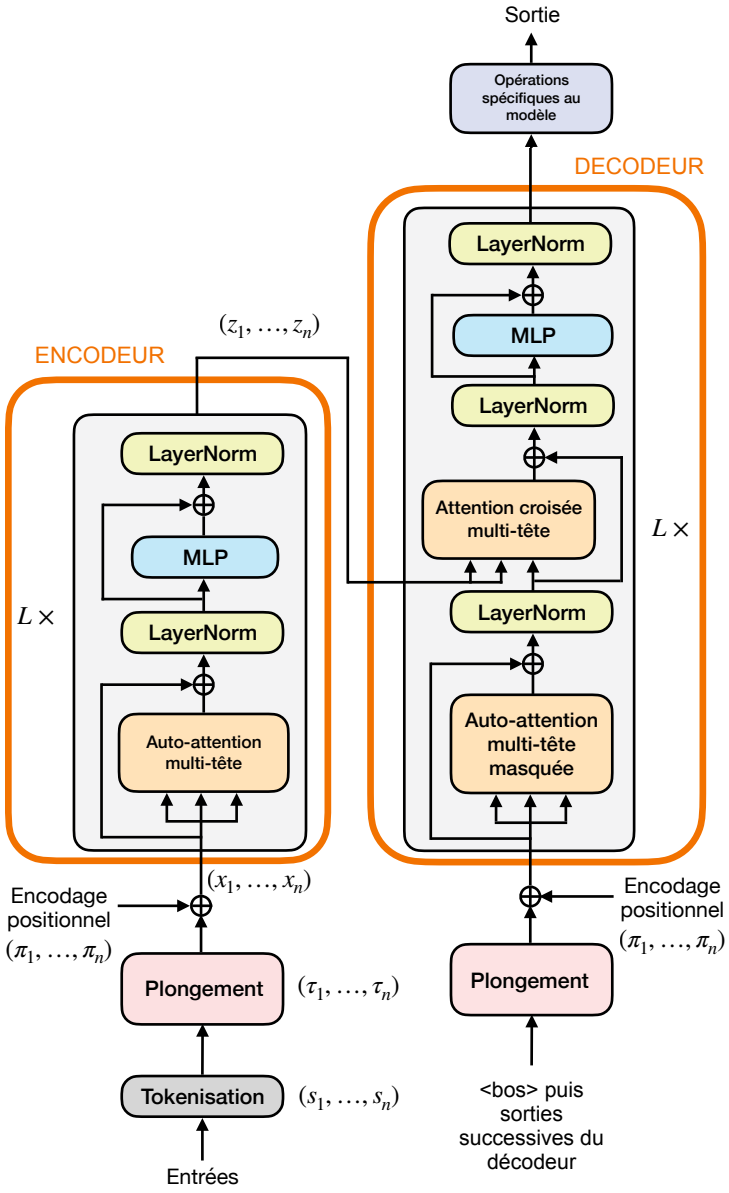

FIGURE 4 – Vue d'ensemble de l'architecture Transformer. Figure inspirée de Vaswani et al. [63].

3 Directions de recherche

Mon travail en stage de M2 et préthèse s'est concentré sur deux aspects de la théorie des Transformers.

3.1 Sur la régularité de l'auto-attention

Notre premier projet porte sur la régularité de l'auto-attention, et plus précisément sur sa constante de Lipschitz

$$\mathrm{Lip}(f) \coloneqq \sup_{X \neq Y} \frac{\|f(X) - f(Y)\|}{\|X - Y\|}$$

où

$$\|X\| \coloneqq \sqrt{\sum_{1 \leq i \leq n} \sum_{1 \leq j \leq d} X_{ij}^2}$$

pour $X = (x_1, \ldots, x_n) \in (\mathbb{R}^d)^n$. En fait, l'auto-attention n'est pas globalement Lipschitzienne [37], nous allons donc nous concentrer sur sa constante de Lipschitz restreinte au compact B_R^n, où $B_R \subset \mathbb{R}^d$ est la boule fermée de rayon R centrée en 0.

Pourquoi étudier la constante de Lipschitz ? L'étude de la Lipschitzianité des réseaux de neurones est intéressante pour différents problèmes en apprentissage automatique [55]. Contrôler la constante de Lipschitz d'un modèle d'apprentissage fournit des garanties de robustesse au bruit et aux attaques adversariales [61, 17, 62, 4, 67], ce qui se comprend intuitivement car la constante de Lipschitz détermine à quel point la sortie d'une fonction peut varier sous l'effet d'une perturbation de l'entrée de norme fixée. Dans le même esprit, déterminer les petites perturbations induisant des variations particulièrement importantes de la sortie permet de développer des stratégies de robustification, par exemple avec la méthode d'apprentissage par exemples contradictoires [31, 46, 48, 40]. La constante de Lipschitz intervient également dans des bornes de généralisation [60, 49, 7, 44], assurant que les réseaux très réguliers ont tendance à bien généraliser (voir le paragraphe 1 pour une définition de la généralisation). Des réseaux de neurones dont la constante de Lipschitz est contrainte lors de l'entraînement peuvent aussi être utilisés pour faire de l'estimation de distance de Wasserstein [53], améliorer l'expressivité et la performance des modèles d'apprentissage profond [47, 21], et construire des réseaux de neurones inversibles [8, 13]. Enfin, déterminer la constante de Lipschitz d'un réseau de neurones est une étape importante dans l'étude de l'équation différentielle neuronale associée [12], en particulier de son caractère bien posé [43, 29, 27] (voir sous-section 3.2).

Deux formalismes différents de l'auto-attention. Nous avons déjà vu la définition de l'auto-attention classique.

Définition 3.1 (Auto-attention)**.** *Soit $A, V \in \mathbb{R}^{d \times d}$. L'auto-attention de paramètres (A, V) est définie comme*

$$f \colon \cup_{n \geq 1} (\mathbb{R}^d)^n \to \cup_{n \geq 1} (\mathbb{R}^d)^n, \quad X = (x_1, \ldots, x_n) \mapsto (f(X)_1, \ldots, f(X)_n) \tag{3}$$

C'est ce qu'on appelle l'architecture pre-LayerNorm [69].

3. En pratique, on utilise l'attention masquée multi-tête, définie exactement sur le modèle de l'attention multi-tête classique.

où

$$f(X)_i \coloneqq V \sum_{j=1}^{n} P_{ij} x_j \qquad \textit{avec} \qquad P_{ij} \coloneqq e^{x_i^\top A^\top x_j} / \sum_{k=1}^{n} e^{x_i^\top A^\top x_k}.$$

Cette définition peut être généralisée à l'espace $\mathcal{P}(B_R)$ des mesures de probabilité à support dans B_R, du fait du caractère équivariant par permutation de l'auto-attention.

Définition 3.2 (Auto-attention généralisée)**.** *Soit $A, V \in \mathbb{R}^{d\times d}$. L'auto-attention généralisée de paramètres (A, V) est définie comme*

$$F \colon \mu \in \mathcal{P}(B_R) \to (\Gamma_\mu)_\sharp \mu \in \mathcal{P}(B_R)$$

avec

$$\Gamma_\mu \colon x \in \mathbb{R}^d \mapsto \frac{\int V y \, \exp(x^\top A^\top y) \mathrm{d}\mu(y)}{\int \exp(x^\top A^\top y) \mathrm{d}\mu(y)}$$

et où $(\Gamma_\mu)_\sharp \mu$ est la mesure image de μ par Γ_μ, défini ci-après. [4]

Introduisons la notion de mesure image, qui intervient dans la définition de l'auto-attention généralisée.

Définition 3.3 (Mesure image)**.** *Soit μ une mesure de probabilité sur B_R et $\Gamma \colon B_R \to B_R$ une fonction mesurable. La mesure image de μ par Γ, noté $\Gamma_\sharp \mu$, est la mesure de probabilité donnée par*

$$\left(\Gamma_\sharp \mu\right)(\mathcal{B}) = \mu(\Gamma^{-1}(\mathcal{B}))$$

pour tout borélien $\mathcal{B} \subset \mathcal{X}$, où $\Gamma^{-1}(\mathcal{B}) = \{x \in B_R : \Gamma(x) \in \mathcal{B}\}$.

On vérifie que l'auto-attention sur les mesures F coïncide avec l'auto-attention classique si l'on représente chaque entrée $(x_1, \dots, x_n)$ par la mesure de probabilité $\frac{1}{n}\sum_{i=1}^n \delta_{x_i}$, où δ_x est la masse de Dirac en x. Pour mesurer la constante de Lipschitz de l'auto-attention généralisée, on munit l'espace $\mathcal{P}(B_R)$ de la distance de Wasserstein 2.

Définition 3.4 (Distance de Wasserstein 2)**.** *Pour $\mu, \nu \in \mathcal{P}(B_R)$, on définit la distance de Wasserstein 2 entre μ et ν par*

$$W_2(\mu, \nu) = \left(\inf_{\pi \in \Pi(\mu,\nu)} \int |x - y|^2 \, \mathrm{d}\pi(x, y) \right)^{1/2}$$

où $\Pi(\mu, \nu)$ est l'ensemble des couplages entre μ et ν, i.e. des mesures de probabilité $\pi \in \mathcal{P}(B_R \times B_R)$ telles que $\int \pi(\cdot, y)\mathrm{d}y = \mu$ and $\int \pi(x, \cdot)\mathrm{d}x = \nu$.

Ce choix de distance assure la relation suivante entre la constante de Lipschitz euclidienne de l'attention classique et la constante de Lipschitz Wasserstein de l'attention généralisée [10] :

$$\mathrm{Lip}(f_{|B_R^n}) = \mathrm{Lip}(F_{|\mathcal{P}_n(B_R)}) \leq \mathrm{Lip}(F_{|\mathcal{P}(B_R)}),$$

où $\mathcal{P}_n(B_R)$ désigne l'ensemble des mesures de probabilité de la forme $\frac{1}{n}\sum_{i=1}^n x_i$ avec $x_1, \dots, x_n \in B_R$. Cette relation est cruciale pour pouvoir transposer à l'attention classique tout majorant obtenu sur la constante de Lipschitz de l'attention généralisée. Et justement, contrairement au cas discret classique, le formalisme de l'auto-attention généralisée donne accès à des outils de transport optimal qui permettent une majoration de la constante de Lipschitz de l'attention qui soit *indépendante* de la longueur n de la séquence d'entrée (voir paragraphe 3.1).

4. Intuitivement, la mesure image est obtenue en transportant chaque élément infinitésimal de masse de x vers $\Gamma_\mu(x)$.

État de l'art. Kim, Papamakarios et Mnih [37] montrent que l'auto-attention n'est pas globalement Lipschitzienne en établissant un minorant sur la constante de Lipschitz de f restreinte à B_R^n, minorant qui croît quadratiquement avec R. Pour gagner en régularité, ils définissent une nouvelle version de l'auto-attention, appelée auto-attention L2, qui est globalement Lipschitzienne à longueur de séquence n fixée. Geshkovski et al. [29] et Vuckovic, Baratin et Combes [64] montrent une majoration de la constante de Lipschitz de l'auto-attention généralisée sur $\mathcal{P}(B_R)$ pour la distance de Wasserstein 2, majoration qui croît exponentiellement avec R^2. Le minorant quadratique et le majorant exponentiel combinés fournissent donc une estimation très vague de la constante de Lipschitz de l'auto-attention sur les ensembles compacts.

Contributions. Dans un article accepté à la conférence ICML 2024, nous présentons les résultats nouveaux suivants. Tout d'abord, nous obtenons un majorant de la constante de Lipschitz de l'auto-attention à longueur de séquence n fixée, et montrons l'optimalité de la borne en n dans un certain régime de longueur de séquence.

Théorème 3.1. *Soit $A, V \in \mathbb{R}^{d\times d}$. Soit $R > 0$ et $n \in \mathbb{N}^*$. L'auto-attention f de paramètres (A, V) est Lipschitzienne sur B_R^n, et*

$$\mathrm{Lip}\left(f_{|B_R^n}\right) \leq \sqrt{3}\,\|V\|_2 \left(\|A\|_2^2\, R^4(4n+1)+n\right)^{1/2}.$$

De plus, soit $\gamma_1 \geq \cdots \geq \gamma_\delta$ les valeurs propres réelles de A. En notant $\gamma \coloneqq \max(-\gamma_\delta, \gamma_1/8)$, on a

$$\mathrm{Lip}(f_{|B_R^n}) \geq \frac{1}{1+(n-1)e^{-2R^2\gamma}}\sqrt{n-1}.$$

L'on voit que lorsque n est trop grand, le majorant donné par le Théorème 3.1 n'est plus optimal, car le minorant ne croît plus comme $\sqrt{n}$. Pire, le majorant tend vers $+\infty$ quand $n \to +\infty$, ce qui ne devrait pas être le cas car l'auto-attention généralisée est globalement Lipschitzienne sur $\mathcal{P}(B_R)$ [29]. Il convient alors de compléter le Théorème 3.1 par une borne sur la constante de Lipschitz de l'attention généralisée.

Théorème 3.2. *Soit $R > 0$ et $A, V \in \mathbb{R}^{k\times d}$. L'auto-attention généralisée F de paramètres (A, V) est W_2-Lipschitzienne sur $\mathcal{P}(B_R)$, et*

$$\mathrm{Lip}^{W_2}(F_{|\mathcal{P}(B_R)}) \leq \|V\|_2\,(1+3\,\|A\|_2\,R^2)e^{2\|A\|_2 R^2}.$$

De plus, en supposant que $V = I_d$, et $n \sim_{R\to+\infty} e^{2\gamma R^2}$, il existe une fonction $\theta\colon [0,+\infty[\to [0,+\infty[$ telle que $\theta(R) \to_{R\to+\infty} 1$ et

$$\mathrm{Lip}(f_{|B_R^n}) \geq \theta(R)\frac{\gamma}{2}R^2 e^{\gamma R^2}.$$

Le Théorème 3.2 montre que la dépendance de la constante de Lipschitz de l'attention en R est catastrophique pour de grandes longueurs de séquence. Heureusement, dans le régime pratique, c'est-à-dire avec de vraies données, c'est plutôt le Théorème 3.1 qui s'applique. Nous avons confirmé cette observation par des expériences numériques, qui montrent que la constante de Lipschitz locale de l'auto-attention [5] croît effectivement comme $\sqrt{n}$ dans le pire des cas (deuxième ligne de la Figure 5). Le numérique nous apprend également que la constante de Lipschitz locale de *vraies données* croît avec n, non pas comme une racine mais plutôt comme $n^{1/4}$ (première ligne de la Figure 5).

5. La constante de Lipschitz locale de l'auto-attention en $X \in (\mathbb{R}^d)^n$ est définie par la norme d'opérateur $\|D_X f\|_2$ de la différentielle de f en X.

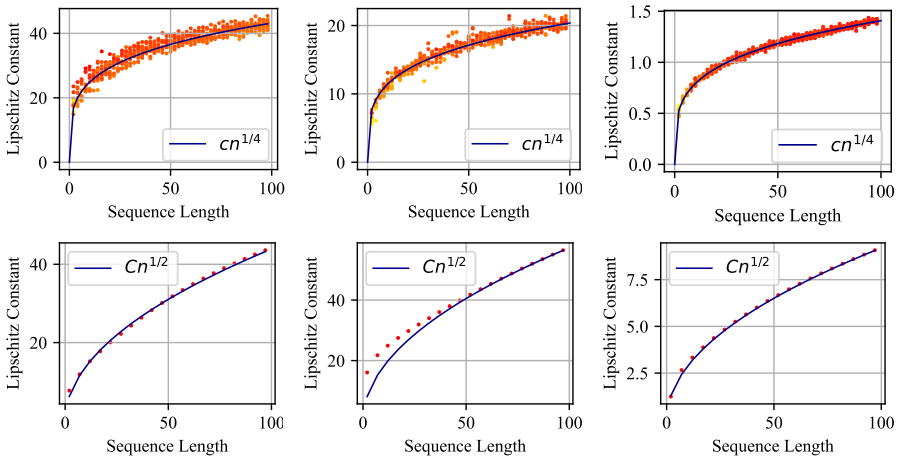

FIGURE 5 – Représentation de la constante de Lipschitz locale de l'auto-attention (colonne 1) et de l'auto-attention masquée (colonnes 2 et 3) sur de vraies données textuelles (première ligne) et des données adversariales (deuxième ligne) en fonction de la longueur de séquence n. Les deux premières colonnes correspondent à deux modèles BERT [24] pré-entraînés différents : un encodeur et un décodeur, sur le même jeu de données Alice in Wonderland, respectivement pour les couches d'attention 0 et 6. La troisième colonne est obtenue avec la sixième couche d'auto-attention masquée de GPT-2 initialisé aléatoirement, sur l'ensemble de données AG_NEWS.

Remarque 3.5. *Nos résultats mettent en évidence la difficulté de définir des couches d'attention qui soient 1-Lipschitziennes (ce qui serait souhaitable pour plusieurs applications, voir paragraphe 3.1). En effet, la méthode simple consistant à rajouter un préfacteur devant l'attention correspondant au majorant du Théorème 3.2, ou du Théorème 3.1 à condition de borner les longueurs de séquence autorisées, induirait une grande perte d'expressivité sur les entrées ayant une plus petite longueur de séquence.*

3.2 Modéliser le Transformer par une équation aux dérivées partielles

Nous avons commencé un deuxième projet lié à la modélisation de l'architecture Transformer par une équation aux dérivées partielles sur les mesures de probabilité. Cette approche est naturelle une fois que l'on a introduit l'auto-attention généralisée, puisque l'architecture du Transformer est résiduelle (voir Section 2). En se restreignant à l'encodeur et en supprimant les MLP intercalés entre les couches d'attention ainsi que l'opération de normalisation, on peut voir un Transformer infiniment profond comme l'action de l'équation aux dérivées partielles suivante :

$$\partial_t \mu + \operatorname{div}(\mu \Gamma_\mu) = 0, \tag{4}$$

avec les notations de la Définition 3.2. Cette idée a été introduite par Sander et al. [57] sur le modèle des ResNets [13], et approfondie par Geshkovski et al. [29]. En particulier, Geshkovski et al. [29] étudient la dynamique des solutions de l'équation (4) associées à des conditions initiales discrètes $\frac{1}{n}\sum_{i=1}^{n} \delta_{x_i(0)}$, et montrent que si l'on renormalise les $x_i(t)$ de la façon suivante :

$$z_i(t) \coloneqq e^{-tV} x_i(t),$$

on observe un regroupement en amas des z_i quand $t \to +\infty$. Cette renormalisation fait écho à l'opération de normalisation LayerNorm présente dans l'architecture des Transformers. Dans

un deuxième article, les auteurs ajoutent l'action de LayerNorm à leur équation aux dérivées partielles, obtenant une EDP sur la sphère dont ils analysent à nouveau la dynamique [27].

Contributions. Nous nous sommes concentrés sur la démonstration du caractère bien posé de l'équation (4) dans le cas d'une condition initiale à support compact puis d'une condition initiale gaussienne, pour différentes variantes de l'auto-attention : l'attention classique, l'attention L2 [37] et l'attention Sinkformer [57], associées aux champs de vitesse suivants.

Définition 3.6. *Soit $d \in \mathbb{N}^*$ et $Q, K, V \in \mathbb{R}^{d\times d}$. Soit aussi $\mu \in \mathcal{P}_c(\mathbb{R}^d)$ une mesure de probabilité à support compact. On considère les champs de vitesse suivants associés à μ, chacun correspondant à une variante différente de l'auto-attention.*

- *Le champ de vitesses associé à l'auto-attention classique est défini par*

$$\Gamma_\mu^{(\mathrm{trad})} \colon x \in \mathbb{R}^d \mapsto \frac{\int V y\, e^{\langle Ax, y\rangle} \mathrm{d}\mu(y)}{\int e^{\langle Ax, y\rangle} \mathrm{d}\mu(y)},$$

  où $A \coloneqq K^\top Q/\sqrt{d}$.

- *Le champ de vitesses associé à l'auto-attention L2 est défini par*

$$\Gamma_\mu^{(\mathrm{L2})} \colon x \in \mathbb{R}^d \mapsto \frac{\int V y\, e^{-|Qx-Ky|^2} \mathrm{d}\mu(y)}{\int e^{-|Qx-Ky|^2} \mathrm{d}\mu(y)}.$$

- *Le champ de vitesses associé à l'auto-attention Sinkformer est défini par*

$$\Gamma_\mu^{(\mathrm{sink})} \colon x \in \mathbb{R}^d \mapsto \int V y\, k^\infty(x, y) \mathrm{d}\mu(y)$$

  où k^∞ s'obtient en appliquant l'algorithme de Sinkhorn-Knopp à $k^0(x, y) \coloneqq e^{-|Qx-Ky|^2}$, i.e. $k^\infty(x, y)$ est la limite des itérations suivantes :

$$k^{\ell+1}(x, y) = \begin{cases} \frac{k^\ell(x,y)}{\int k^\ell(x,y')\mathrm{d}\mu(y')} & \text{si } \ell \text{ est pair,} \\ \frac{k^\ell(x,y)}{\int k^\ell(x',y)\mathrm{d}\mu(x')} & \text{si } \ell \text{ est impair.} \end{cases} \tag{5}$$

Lorsque la condition initiale est à support compact, nous montrons que les trois types d'attention induisent une EDP bien posée.

Théorème 3.3. *Soit $d \in \mathbb{N}^*$. Soit $Q, K, V \colon [0, +\infty) \to \mathbb{R}^{d\times d}$ continues, modélisant l'évolution des paramètres Q, K, V à travers les couches du Transformer. Soit $\mu_0 \in \mathcal{P}_c(\mathbb{R}^d)$ une condition initiale à support compact. Alors, pour tout choix de $\Gamma \in \{\Gamma^{(\mathrm{trad})}, \Gamma^{(\mathrm{L2})}, \Gamma^{(\mathrm{sink})}\}$, et en notant $\Gamma_\mu(t) \coloneqq \Gamma_{\mu(t)}$ le champ de vitesses associé aux paramètres $Q(t), K(t), V(t)$, l'équation aux dérivées partielles*

$$\partial_t \mu + \mathrm{div}(\mu \Gamma_\mu) = 0 \tag{6}$$

associée à la condition initiale $\mu(0) = \mu_0$ a une unique solution globale $\mu \in \mathcal{C}([0, +\infty), \mathcal{P}(\mathbb{R}^d))$, où $\mathcal{P}(\mathbb{R}^d)$ est équipé de la distance de Wasserstein W_2. De plus, soit R_0 le plus petit rayon tel que $\mathrm{Supp}\,\mu_0 \subset B_{R_0}$, et notons

$$R(t) \coloneqq e^{\int_0^t \|V(s)\|_2 \mathrm{d}s} R_0$$

pour $t \geq 0$. Alors, la solution μ vérifie

$$\mathrm{Supp}\,\mu(t) \subset B_{R(t)}$$

pour tout $t \geq 0$. Enfin, on a une estimée de stabilité de la forme suivante. Pour tout $R_0 > 0$ et $T > 0$, il existe une constante $C(T, R_0)$ ne dépendant que de T et R_0 (et du choix de Γ_μ), telle que pour toutes conditions initiales μ_0 et ν_0 supportées dans B_{R_0}, et en notant μ et ν les solutions gloables associées de l'équation (6), on a

$$W_p(\mu(t), \nu(t)) \leq C(T, R_0) W_p(\mu_0, \nu_0).$$

Le cas d'une condition initiale gaussienne, bien qu'assez éloigné de tout questionnement pratique, offre un cadre théorique intéressant car pour les trois types d'attention ci-dessus, la solution de l'équation (4) reste gaussienne pour tout temps, et induit une équation différentielle matricielle sur la covariance qui peut être analysée théoriquement et numériquement. Nous ne détaillons pas cette partie du projet, qui est encore en cours.

Soulignons enfin que la modélisation de l'architecture Transformer par une équation aux dérivées partielles offre des perspectives intéressantes pour mieux comprendre l'optimisation des Transformers, suivant une approche semblable à celle de Chizat et Bach [15]. Cette piste pourrait être l'objectif à long terme de ma thèse.

Références

[1] Imtiaz Ahmed et al. « ChatGPT vs. Bard : A Comparative Study ». In : *UMBC Student Collection* (2023).

[2] Jay Alammar. *The Illustrated Transformer*. 2018. URL : http://jalammar.github.io/illustrated-transformer/.

[3] Shunichi Amari. « A theory of adaptive pattern classifiers ». In : *IEEE Transactions on Electronic Computers* 3 (1967), p. 299-307.

[4] Cem Anil, James Lucas et Roger Grosse. « Sorting out Lipschitz function approximation ». In : *International Conference on Machine Learning*. PMLR. 2019, p. 291-301.

[5] Sylvain Arlot. « Fondamentaux de l'apprentissage statistique ». In : *Apprentissage statistique et donn\'ees massives* (2018).

[6] Francis Bach et Lenaic Chizat. « Gradient descent on infinitely wide neural networks : Global convergence and generalization ». In : *arXiv preprint arXiv :2110.08084* (2021).

[7] Peter L Bartlett, Dylan J Foster et Matus J Telgarsky. « Spectrally-normalized margin bounds for neural networks ». In : *Advances in neural information processing systems* 30 (2017).

[8] Jens Behrmann et al. « Invertible residual networks ». In : *International conference on machine learning*. PMLR. 2019, p. 573-582.

[9] Srinadh Bhojanapalli et al. « Understanding robustness of transformers for image classification ». In : *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, p. 10231-10241.

[10] Valérie Castin, Pierre Ablin et Gabriel Peyré. « Understanding the Regularity of Self-Attention with Optimal Transport ». In : *arXiv preprint arXiv :2312.14820* (2023).

[11] Lili Chen et al. « Decision transformer : Reinforcement learning via sequence modeling ». In : *Advances in neural information processing systems* 34 (2021), p. 15084-15097.

[12] Ricky TQ Chen et al. « Neural ordinary differential equations ». In : *Advances in neural information processing systems* 31 (2018).

[13] Ricky TQ Chen et al. « Residual flows for invertible generative modeling ». In : *Advances in Neural Information Processing Systems* 32 (2019).

[14] Siyu Chen et al. « Training Dynamics of Multi-Head Softmax Attention for In-Context Learning : Emergence, Convergence, and Optimality ». In : *arXiv preprint arXiv :2402.19442* (2024).

[15] Lenaic Chizat et Francis Bach. « On the global convergence of gradient descent for over-parameterized models using optimal transport ». In : *Advances in neural information processing systems* 31 (2018).

[16] Yu-An Chung et al. *W2v-BERT : Combining Contrastive Learning and Masked Language Modeling for Self-Supervised Speech Pre-Training*. 2021. arXiv : 2108.06209 [cs.LG].

[17] Moustapha Cisse et al. « Parseval networks : Improving robustness to adversarial examples ». In : *International conference on machine learning*. PMLR. 2017, p. 854-863.

[18] Papers With Code. *Machine Translation Leaderboards*. 2023. URL : https://paperswithcode.com/sota/machine-translation-on-wmt2014-english-german.

[19] Papers With Code. *Speech Recognition Leaderboard on LibriSpeech*. 2023. URL : https://paperswithcode.com/sota/speech-recognition-on-librispeech-test-other.

[20] Gonçalo M Correia, Vlad Niculae et André FT Martins. « Adaptively sparse transformers ». In : *arXiv preprint arXiv :1909.00015* (2019).

[21] George Dasoulas, Kevin Scaman et Aladin Virmaux. « Lipschitz normalization for self-attention layers with application to graph neural networks ». In : *International Conference on Machine Learning*. PMLR. 2021, p. 2456-2466.

[22] *DeepL Press Release*. URL : https://www.deepl.com/press.html.

[23] Jia Deng et al. « Imagenet : A large-scale hierarchical image database ». In : *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, p. 248-255.

[24] Jacob Devlin et al. « Bert : Pre-training of deep bidirectional transformers for language understanding ». In : *arXiv :1810.04805* (2018).

[25] Alexey Dosovitskiy et al. « An image is worth 16x16 words : Transformers for image recognition at scale ». In : *arXiv :2010.11929* (2020).

[26] Shivam Garg et al. « What can transformers learn in-context ? a case study of simple function classes ». In : *Advances in Neural Information Processing Systems* 35 (2022), p. 30583-30598.

[27] Borjan Geshkovski et al. « A Mathematical Perspective on Transformers ». 2023.

[28] Borjan Geshkovski et al. « A mathematical perspective on Transformers ». In : *arXiv preprint arXiv :2312.10794* (2023).

[29] Borjan Geshkovski et al. « The emergence of clusters in self-attention dynamics ». In : *arXiv preprint arXiv :2305.05465* (2023).

[30] Ian Goodfellow et al. « Generative adversarial nets ». In : *Advances in neural information processing systems* 27 (2014).

[31] Ian J Goodfellow, Jonathon Shlens et Christian Szegedy. « Explaining and harnessing adversarial examples ». In : *arXiv preprint arXiv :1412.6572* (2014).

[32] Kaiming He et al. « Deep residual learning for image recognition ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 770-778.

[33] Kaiming He et al. *Deep residual learning for image recognition. CoRR abs/1512.03385 (2015)*. 2015.

[34] Geoffrey Hinton et al. « Deep neural networks for acoustic modeling in speech recognition : The shared views of four research groups ». In : *IEEE Signal processing magazine* 29.6 (2012), p. 82-97.

[35] Aleksei Grigorevich Ivakhnenko, Valentin Grigorévich Lapa et al. « Cybernetic predicting devices ». In : (1966).

[36] Leslie Pack Kaelbling, Michael L Littman et Andrew W Moore. « Reinforcement learning : A survey ». In : *Journal of artificial intelligence research* 4 (1996), p. 237-285.

[37] Hyunjik Kim, George Papamakarios et Andriy Mnih. *The Lipschitz Constant of Self-Attention*. 2021. arXiv : 2006.04710 [stat.ML].

[38] Diederik P Kingma et Jimmy Ba. « Adam : A method for stochastic optimization ». In : *arXiv preprint arXiv :1412.6980* (2014).

[39] Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton. « Imagenet classification with deep convolutional neural networks ». In : *Advances in neural information processing systems* 25 (2012).

[40] Alexey Kurakin, Ian Goodfellow et Samy Bengio. « Adversarial machine learning at scale ». In : *arXiv preprint arXiv :1611.01236* (2016).

[41] Yinhan Liu et al. « Roberta : A robustly optimized bert pretraining approach ». In : *arXiv :1907.11692* (2019).

[42] Yixin Liu et al. « Sora : A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models ». In : *arXiv preprint arXiv :2402.17177* (2024).

[43] Yiping Lu et al. « Understanding and improving transformer from a multi-particle dynamic system point of view ». In : *arXiv preprint arXiv :1906.02762* (2019).

[44] Ulrike von Luxburg et Olivier Bousquet. « Distance-Based Classification with Lipschitz Functions. » In : *J. Mach. Learn. Res.* 5.Jun (2004), p. 669-695.

[45] Kaleel Mahmood, Rigel Mahmood et Marten Van Dijk. « On the robustness of vision transformers to adversarial examples ». In : *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, p. 7838-7847.

[46] Takeru Miyato et al. « Distributional smoothing with virtual adversarial training ». In : *arXiv preprint arXiv :1507.00677* (2015).

[47] Takeru Miyato et al. « Spectral normalization for generative adversarial networks ». In : *arXiv preprint arXiv :1802.05957* (2018).

[48] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi et Pascal Frossard. « Deepfool : a simple and accurate method to fool deep neural networks ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 2574-2582.

[49] Behnam Neyshabur et al. « Exploring generalization in deep learning ». In : *Advances in neural information processing systems* 30 (2017).

[50] OpenAI. *GPT-3 Tokenizer*. 2023. URL : https://platform.openai.com/tokenizer.

[51] OpenAI. *GPT-4 Technical Report*. 2023. arXiv : 2303.08774 [cs.CL].

[52] Sayak Paul et Pin-Yu Chen. « Vision transformers are robust learners ». In : *Proceedings of the AAAI conference on Artificial Intelligence*. T. 36. 2. 2022, p. 2071-2081.

[53] Gabriel Peyré, Marco Cuturi et al. « Computational optimal transport : With applications to data science ». In : *Foundations and Trends® in Machine Learning* 11.5-6 (2019), p. 355-607.

[54] Aditya Ramesh et al. « Zero-shot text-to-image generation ». In : *International conference on machine learning*. Pmlr. 2021, p. 8821-8831.

[55] Mihaela Rosca et al. « A case for new neural network smoothness constraints ». In : *PMLR* (2020).

[56] Michael E Sander et al. « How do Transformers perform In-Context Autoregressive Learning ? » In : *arXiv preprint arXiv :2402.05787* (2024).

[57] Michael E Sander et al. « Sinkformers : Transformers with doubly stochastic attention ». In : *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, p. 3515-3530.

[58] Hamed Shirzad et al. « Exphormer : Sparse transformers for graphs ». In : *International Conference on Machine Learning*. PMLR. 2023, p. 31613-31632.

[59] David Silver et al. « Mastering the game of Go with deep neural networks and tree search ». In : *nature* 529.7587 (2016), p. 484-489.

[60] Jure Sokolić et al. « Robust large margin deep neural networks ». In : *IEEE Transactions on Signal Processing* 65.16 (2017), p. 4265-4280.

[61] Christian Szegedy et al. « Intriguing properties of neural networks ». In : *arXiv preprint arXiv :1312.6199* (2013).

[62] Yusuke Tsuzuku, Issei Sato et Masashi Sugiyama. « Lipschitz-margin training : Scalable certification of perturbation invariance for deep neural networks ». In : *Advances in neural information processing systems* 31 (2018).

[63] Ashish Vaswani et al. « Attention is all you need ». In : *Advances in neural information processing systems* 30 (2017).

[64] James Vuckovic, Aristide Baratin et Rémi Tachet des Combes. « A Mathematical Theory of Attention ». In : *ArXiv* abs/2007.02876 (2020).

[65] Alex Wang et al. *SuperGlue Leaderboard*. 2023. URL : https://super.gluebenchmark.com/leaderboard/.

[66] Sinong Wang et al. « Linformer : Self-attention with linear complexity ». In : *arXiv preprint arXiv :2006.04768* (2020).

[67] Tsui-Wei Weng et al. « Evaluating the robustness of neural networks : An extreme value theory approach ». In : *arXiv preprint arXiv :1801.10578* (2018).

[68] Enze Xie et al. « SegFormer : Simple and efficient design for semantic segmentation with transformers ». In : *Advances in Neural Information Processing Systems* 34 (2021), p. 12077-12090.

[69] Ruibin Xiong et al. « On layer normalization in the transformer architecture ». In : *International Conference on Machine Learning*. PMLR. 2020, p. 10524-10533.

[70] Chulhee Yun et al. « Are transformers universal approximators of sequence-to-sequence functions ? » In : *arXiv preprint arXiv :1912.10077* (2019).