Transformers Revisited: Architecture, Attention, and Limitations

1 Introduction

Les transformers constituent aujourd'hui l'une des architectures fondamentales de l'intelligence artificielle moderne, en particulier en traitement automatique du langage naturel. Ils ont profondément transformé la manière dont les machines comprennent, représentent et génèrent des séquences.

Ce document propose une explication progressive et conceptuelle des transformers, en allant au-delà des formules mathématiques afin d'en saisir la logique profonde, l'architecture et les limites.

2 Philosophie des transformers

Avant les transformers, les modèles séquentiels tels que les réseaux de neurones récurrents (RNN) et les LSTM traitaient les données étape par étape. Cette approche présentait deux limitations majeures :

- la difficulté à capturer des dépendances longues,
- l'impossibilité de paralléliser efficacement les calculs.

La philosophie des transformers repose sur une idée clé : **toute la séquence peut être traitée simultanément**, sans dépendre d'un ordre strictement temporel.

Au cœur de cette approche se trouve le mécanisme d'attention. Au lieu de stocker l'information dans un état caché qui se propage, le modèle apprend à identifier quelles parties de l'entrée sont pertinentes pour comprendre une autre partie. Chaque élément de la séquence peut ainsi « regarder » tous les autres.

Les transformers privilégient donc :

- une vision globale de la séquence,
- une meilleure capture du contexte,
- une forte capacité de parallélisation,
- une modularité facilitant l'extension à grande échelle.

Cette approche marque un changement de paradigme : la séquence n'est plus vue comme une chaîne linéaire, mais comme un ensemble interconnecté.

3 Architecture des transformers

L'architecture standard d'un transformer repose sur deux grands blocs :

- l'encodeur,
- le décodeur.

Selon les modèles, on peut utiliser uniquement l'encodeur (BERT), uniquement le décodeur (GPT) ou les deux (transformer original).

3.1 Les embeddings

Les tokens textuels sont projetés dans un espace vectoriel continu à l'aide d'embeddings. Chaque token est représenté par un vecteur dense de dimension fixe, capturant des similarités sémantiques.

3.2 Encodage positionnel

Les transformers ne possèdent pas de notion intrinsèque d'ordre. Pour y remédier, un encodage positionnel est ajouté aux embeddings. Celui-ci peut être :

- sinusoïdal (déterministe),
- appris durant l'entraînement.

3.3 Mécanisme d'attention

Chaque token est projeté dans trois espaces vectoriels :

- **Query (Q)** : ce que le token recherche,
- **Key (K)** : ce que le token propose,
- **Value (V)** : l'information transmise.

La formule de l'attention est :

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

où d_k est la dimension des clés. Le facteur $\sqrt{d_k}$ stabilise les gradients.

Intuitivement, chaque token calcule à quel point il doit prêter attention aux autres pour construire une représentation contextualisée.

3.4 Multi-Head Attention

Plutôt qu'une seule attention, plusieurs têtes d'attention sont utilisées en parallèle. Chaque tête se spécialise dans un type de relation différent. Les sorties sont concaténées puis projetées dans un espace commun.

3.5 Réseaux feed-forward

Après l'attention, chaque token est transformé indépendamment via un réseau feed-forward :

$$\text{FFN}(x) = \text{Linear}_2(\text{ReLU}(\text{Linear}_1(x)))$$

3.6 Connexions résiduelles et normalisation

Chaque sous-bloc est entouré de connexions résiduelles et d'une normalisation de couche (LayerNorm), ce qui facilite la convergence et la propagation du gradient.

3.7 Encodeur et décodeur

Encodeur : empilement de blocs attention + feed-forward.

Décodeur : similaire à l'encodeur, avec une attention masquée et une attention supplémentaire sur la sortie de l'encodeur.

4 Modèles basés sur les transformers

Les transformers ont donné naissance à de nombreux modèles :

- BERT (encodeur),
- GPT (décodeur),
- T5 (encodeur-décodeur),
- Vision Transformer (ViT),
- Transformers multimodaux.

5 Méthodes d'entraînement

L'entraînement repose sur :

- de grandes quantités de données,
- la fonction de perte cross-entropy,
- des optimiseurs comme Adam ou AdamW,
- des techniques de régularisation.

Deux phases sont généralement distinguées :

1. pré-entraînement auto-supervisé,
2. fine-tuning sur des tâches spécifiques.

6 Complexité et limites

6.1 Complexité computationnelle

Le coût de l'attention est quadratique :

$$O(n^2)$$

où n est la longueur de la séquence.

6.2 Limites mémoire et énergétiques

La matrice d'attention $n \times n$ impose des contraintes fortes en mémoire, données et consommation énergétique.

6.3 Pistes d'amélioration

Des solutions incluent :

- attention clairsemée,
- attention linéaire,
- modèles hiérarchiques,
- distillation.

7 Comparaison RNN, CNN et Transformers

Architecture	Dépendances longues	Parallélisation	Complexité
RNN	Faible	Faible	$O(n)$
CNN	Moyenne	Élevée	$O(nk)$
Transformer	Élevée	Très élevée	$O(n^2)$

8 Pseudo-code d'un transformer

1. Embedding + encodage positionnel
2. Pour chaque couche :
   - Self-attention multi-têtes
   - Connexion résiduelle + LayerNorm
   - Feed-forward
   - Connexion résiduelle + LayerNorm
3. Projection finale et softmax

9 Conclusion

Les transformers ont profondément modifié le paysage de l'intelligence artificielle en introduisant une modélisation directe des dépendances longues, une parallélisation efficace et une grande flexibilité architecturale.

Au-delà de leurs performances, ils représentent un véritable changement de paradigme dans la conception des modèles d'apprentissage profond. Malgré leurs limites computationnelles, les recherches actuelles visent à améliorer leur efficacité, garantissant un rôle central et durable des transformers dans l'IA moderne.

Références

- Vaswani et al., *Attention Is All You Need*, NeurIPS 2017.
- Devlin et al., *BERT*, NAACL 2019.
- Radford et al., *GPT*, OpenAI 2019.
- Raffel et al., *T5*, JMLR 2020.
- Dosovitskiy et al., *ViT*, ICLR 2021.
- Choromanski et al., *Performer*, ICLR 2021.
- Beltagy et al., *Longformer*, 2020.