Deep learning

Modèles, optimisation et architectures

Redha Moulla

Plan

1. Réseaux de neurones et calcul différentiable
2. Optimisation et généralisation
3. Réseaux convolutifs
4. Transformers
5. NLP
6. Graph neural networks
7. Autoencodeurs variationnels et diffusion

1. Réseaux de neurones

Des fonctions paramétrées entraînées par différentiation

Apprentissage supervisé

On dispose d'un échantillon $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ et d'un modèle $f_{\boldsymbol{\theta}}$.

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \ell\big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i\big) + \lambda\, \Omega(\boldsymbol{\theta}).$$

Entraînement

Ajuster $\boldsymbol{\theta}$ sur les données d'apprentissage.

Généralisation

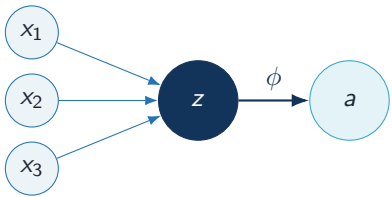
Obtenir une faible erreur sur des données jamais observées.

La fonction de perte, l'architecture, l'optimiseur et le protocole d'évaluation forment un même système.

Neurone artificiel

$$z = \boldsymbol{w}^\top \boldsymbol{x} + b, \qquad a = \phi(z).$$

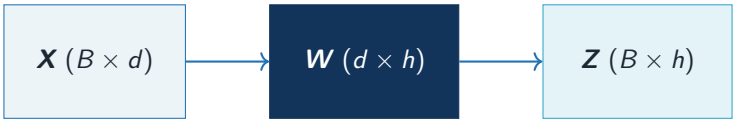
- z : **préactivation**
- ϕ : fonction d'activation
- a : sortie du neurone



Une couche dense

Pour un batch de B observations :

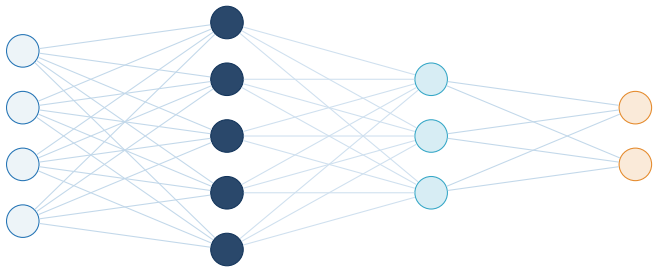
$$\boldsymbol{X} \in \mathbb{R}^{B \times d}, \quad \boldsymbol{W} \in \mathbb{R}^{d \times h}, \quad \boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W} + \mathbf{1}\boldsymbol{b}^{\top} \in \mathbb{R}^{B \times h}.$$



Le suivi explicite des dimensions évite une grande partie des erreurs d'implémentation.

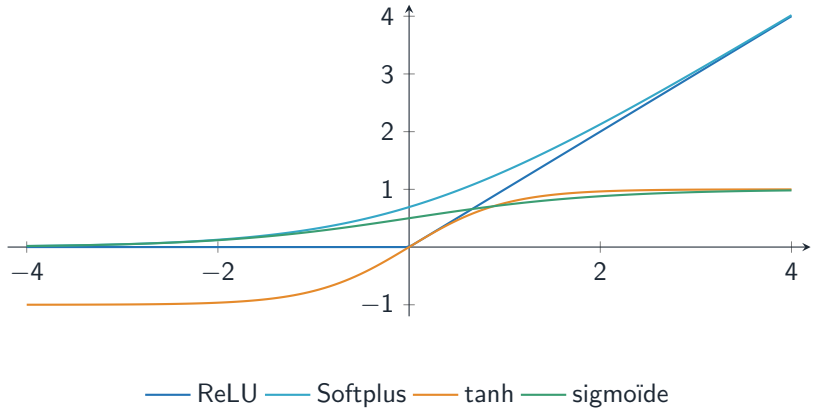
Réseau multicouche

$$\boldsymbol{h}^{(\ell)} = \phi\Big(\boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}\Big), \qquad \boldsymbol{h}^{(0)} = \boldsymbol{x}.$$



La profondeur permet de composer des transformations simples en représentations hiérarchiques.

Fonctions d'activation



ReLU et ses variantes dominent les couches cachées. Sigmoïde et tanh restent utiles lorsque leur intervalle de sortie a un sens fonctionnel.

Choisir l'activation

Fonction	Usage fréquent	Remarque
ReLU	couches cachées	simple, gradient nul pour $z < 0$
Leaky ReLU / GELU	couches cachées	gradient non nul ou transition lisse
Sigmoïde	sortie binaire, portes	saturation pour $\|z\|$ grand
tanh	états bornés, RNN	centrée, mais saturante
Softmax	sortie multiclasse	normalise un vecteur de scores

Une activation ne « résout » pas à elle seule la disparition du gradient : initialisation, normalisation et connexions résiduelles comptent aussi.

Classification binaire

Le modèle produit un logit $z \in \mathbb{R}$, puis une probabilité $p = \sigma(z)$.

$$\ell_{\mathrm{BCE}}(z, y) = -y \log \sigma(z) - (1 - y) \log \big(1 - \sigma(z)\big).$$

Calcul numérique

En pratique, on utilise directement une perte « BCE avec logits ». Elle combine sigmoïde et logarithmes de manière stable.

Le seuil de décision n'est pas nécessairement 0,5 : il dépend des coûts d'erreur et de la distribution d'usage.

Classification multiclasse

Pour K classes, le réseau produit des logits $\boldsymbol{z} \in \mathbb{R}^K$.

$$p_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}, \qquad \ell_{\mathrm{CE}}(\boldsymbol{z}, y) = -\log p_y.$$

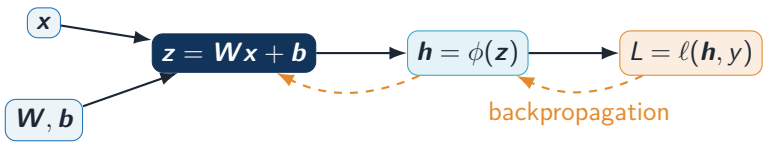
Pendant l'entraînement

La perte utilise les logits ; il n'est pas nécessaire de calculer explicitement softmax avant une implémentation stable de l'entropie croisée.

Pendant l'inférence

$\arg\max_k z_k = \arg\max_k p_k$. Softmax sert si des probabilités sont requises.

Graphe de calcul



La différentiation automatique applique localement la chain rule sur ce graphe ; elle ne réalise pas une différentiation symbolique globale.

Chain rule

Si $L = L(\boldsymbol{h})$, $\boldsymbol{h} = \phi(\boldsymbol{z})$ et $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, avec des gradients colonnes :

$$\nabla_{\boldsymbol{z}} L = J_\phi(\boldsymbol{z})^\top \nabla_{\boldsymbol{h}} L = \nabla_{\boldsymbol{h}} L \odot \phi'(\boldsymbol{z}),$$

pour une activation appliquée composante par composante. On obtient ensuite

$$\nabla_{\boldsymbol{W}} L = (\nabla_{\boldsymbol{z}} L)\boldsymbol{x}^\top, \qquad \nabla_{\boldsymbol{x}} L = \boldsymbol{W}^\top \nabla_{\boldsymbol{z}} L, \qquad \nabla_{\boldsymbol{b}} L = \nabla_{\boldsymbol{z}} L.$$

J_ϕ est la Jacobienne de ϕ ; elle est diagonale uniquement dans le cas composante par composante.

Backpropagation

1. Calcul avant : stocker les valeurs nécessaires au gradient.
2. Calculer la perte scalaire.
3. Parcourir le graphe en ordre topologique inverse.
4. Accumuler les dérivées lorsque plusieurs chemins rejoignent une variable.
5. Mettre à jour les paramètres avec l'optimiseur.

La backpropagation est un algorithme efficace de calcul de gradients ; l'optimiseur décide ensuite comment utiliser ces gradients.

Mini-batches

Pour un mini-batch $\mathcal{B}_t$ de taille B :

$$\boldsymbol{g}_t = \frac{1}{B} \sum_{i \in \mathcal{B}_t} \nabla_{\boldsymbol{\theta}} \ell_i(\boldsymbol{\theta}_t), \qquad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \boldsymbol{g}_t.$$

- calcul vectorisé efficace ;
- gradient bruité utile à l'exploration ;
- mémoire proportionnelle aux activations du batch.

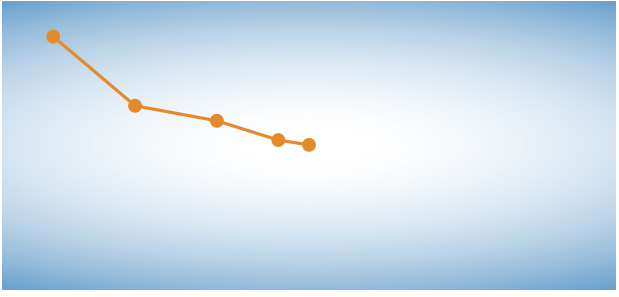
- un **step** traite un mini-batch ;
- une **epoch** parcourt environ toutes les observations ;
- le dernier batch peut être plus petit.

2. Optimisation

Stabiliser l'entraînement et contrôler la généralisation

Descente de gradient stochastique

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \boldsymbol{g}_t.$$



Le learning rate η_t est souvent l'hyperparamètre le plus déterminant : trop grand, l'optimisation diverge ; trop petit, elle progresse lentement.

Momentum

$$\boldsymbol{v}_t = \mu \boldsymbol{v}_{t-1} + \boldsymbol{g}_t, \qquad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{v}_t.$$

Effet

Accumule une direction persistante et atténue les oscillations dans les vallées mal conditionnées.

Paramètres

η contrôle la taille des pas ; μ contrôle la mémoire, souvent proche de 0,9.

Le « Nesterov momentum » évalue le gradient après une avance approximative dans la direction de l'impulsion.

Adam et AdamW

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t, \quad \boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^{\odot 2}.$$

Après correction des biais :

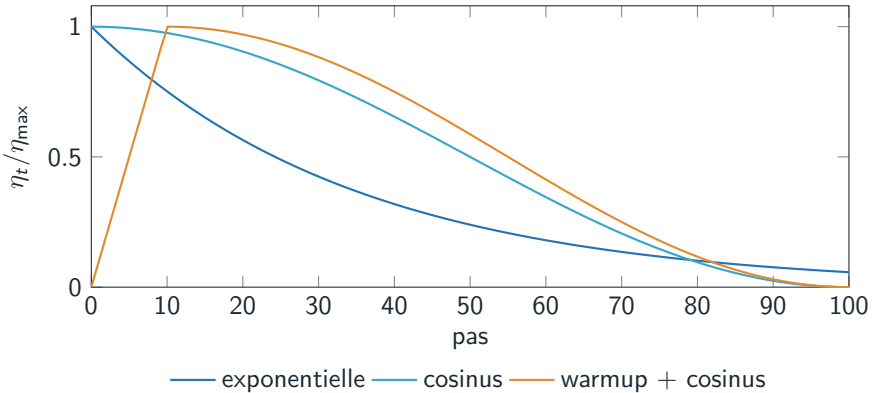
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \varepsilon}.$$

AdamW

Le weight decay est découplé de la mise à jour adaptative : $\boldsymbol{\theta} \leftarrow (1-\eta\lambda)\boldsymbol{\theta} - \eta\,\text{AdamStep}$. Elle n'est donc pas équivalente à ajouter naïvement une pénalité L_2 dans Adam.

Source : Loshchilov et Hutter, 2019.

Learning rate schedule



Un warmup limite les mises à jour initiales instables ; une décroissance permet d'affiner la solution en fin d'entraînement.

Initialisation

Pour conserver l'échelle des activations et des gradients :

Initialisation	Variance typique	Usage
Xavier / Glorot	$\mathrm{Var}(W_{ij}) \approx 2/(n_{in} + n_{out})$	activations symétriques, par ex. tanh
He / Kaiming	$\mathrm{Var}(W_{ij}) \approx 2/n_{in}$	ReLU et variantes
Orthogonale	matrice semi-orthogonale, puis gain	réseaux récurrents ou profonds

Initialiser tous les poids à zéro empêcherait les neurones d'une même couche de se différencier.

Batch normalization

Pour une composante d'activation sur un mini-batch :

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \varepsilon}}, \qquad y_i = \gamma \hat{x}_i + \beta.$$

- statistiques du mini-batch à l'entraînement ;
- running mean and variance à l'inférence ;
- comportement sensible aux très petits batches.

- lisse et reparamètre l'optimisation ;
- autorise souvent des learning rates plus élevés ;
- l'explication par le seul « covariate shift interne » est insuffisante.

Source : Ioffe et Szegedy, 2015 ; Santurkar et al., 2018.

Layer normalization

LayerNorm normalise les composantes d'une représentation pour chaque observation :

$$\mu = \frac{1}{d}\sum_{j=1}^{d} x_j, \qquad \hat{x}_j = \frac{x_j - \mu}{\sqrt{\frac{1}{d}\sum_k (x_k - \mu)^2 + \varepsilon}}, \qquad y_j = \gamma_j \hat{x}_j + \beta_j.$$

- indépendante de la taille du mini-batch ;
- adaptée aux séquences de longueur variable ;
- standard dans les Transformers ;
- placement pré-norme ou post-norme selon l'architecture.

Régularisation

Méthode	Mécanisme	Usage
Weight decay	contracte les poids à chaque pas	optimisateurs, notamment AdamW
Dropout	annule aléatoirement des activations	couches denses, attention, MLP
Augmentation	transforme les exemples sans changer la cible	images, audio, texte
Label smoothing	cible moins concentrée	classification, calibration à vérifier
Early stopping	arrête selon une métrique de validation	contrôle du surapprentissage

Dropout

À l'entraînement, pour un taux p :

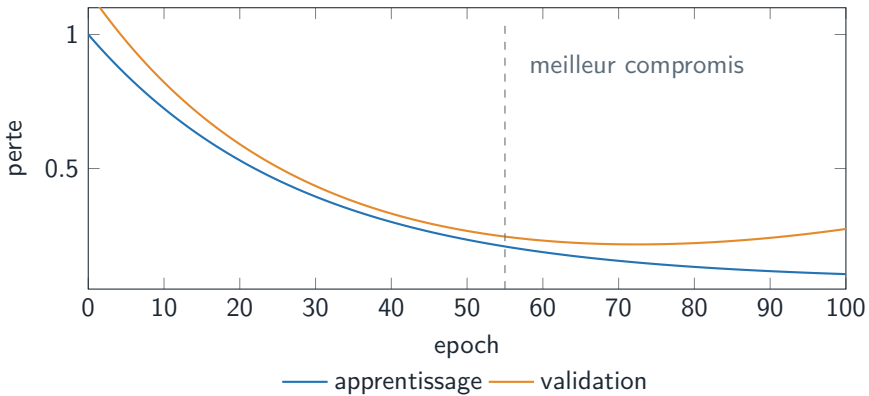
$$\tilde{\boldsymbol{h}} = \frac{\boldsymbol{m} \odot \boldsymbol{h}}{1-p}, \qquad m_j \sim \text{Bernoulli}(1-p).$$

À l'inférence, le masque est désactivé : l'échelle est déjà corrigée par le facteur $1/(1-p)$.

Remarque

Dropout perturbe les coadaptations, mais il ne remplace ni un jeu de validation, ni une augmentation adaptée, ni une capacité de modèle cohérente avec les données.

Diagnostiquer l'entraînement



Une divergence persistante entre apprentissage et validation signale un problème de généralisation, pas nécessairement un simple manque de régularisation.

Stabilité numérique

- Utiliser log-softmax et les pertes « with logits ».
- Soustraire $\max_k z_k$ avant une exponentielle : softmax reste inchangé.
- Ajouter ε dans les divisions et normalisations.
- Surveiller NaN, Inf, normes des gradients et amplitude des activations.
- En précision mixte, employer une mise à l'échelle dynamique de la perte.
- Le gradient clipping peut contrôler leur explosion ; cela ne résout pas leur disparition.

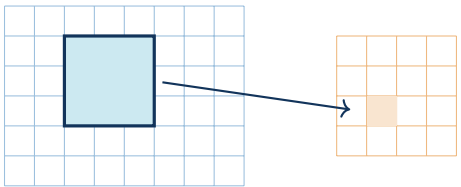
3. Réseaux convolutifs

Exploiter la structure spatiale des données

Convolution discrète

Pour une image $\boldsymbol{X} \in \mathbb{R}^{H \times W \times C_{in}}$ et C_{out} filtres :

$$Y_{i,j,k} = b_k + \sum_{u,v,c} K_{u,v,c,k}\, X_{i+u,j+v,c}.$$



Les bibliothèques implémentent généralement une corrélation croisée, sans retournement du noyau ; le noyau appris absorbe cette convention.

Dimensions de sortie

Pour une dimension d'entrée H, noyau K, padding P, stride S et dilatation D :

$$H_{out} = \left\lfloor \frac{H + 2P - D(K-1) - 1}{S} + 1 \right\rfloor .$$

Exemple

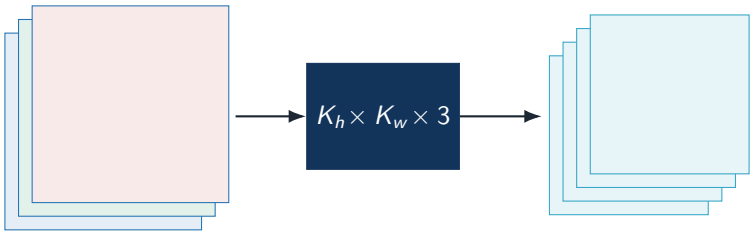
$H = 32$, $K = 3$, $P = 1$, $S = 2$, $D = 1$ donnent $H_{out} = 16$.

Le nombre de paramètres vaut

$$K_h K_w C_{in} C_{out} + C_{out},$$

indépendamment de H et W grâce au partage des poids.

Canaux et filtres



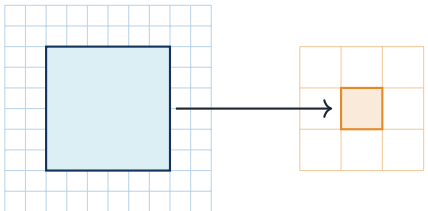
Un filtre s'étend sur tous les canaux d'entrée et produit une carte. Augmenter le nombre de filtres augmente la capacité *et* le nombre de paramètres.

Champ réceptif

Pour des couches successives de noyaux k_ℓ et strides s_ℓ :

$$j_\ell = j_{\ell-1} s_\ell, \qquad r_\ell = r_{\ell-1} + (k_\ell - 1) j_{\ell-1},$$

avec $j_0 = r_0 = 1$.



zone d'entrée influente un neurone profond

Le champ réceptif théorique croît avec la profondeur ; la contribution effective est souvent plus concentrée.

Downsampling

Pooling

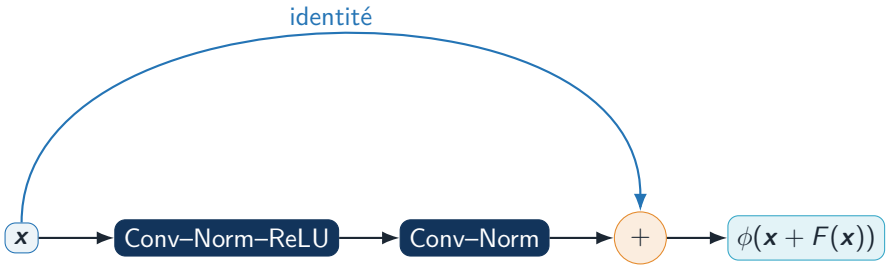
Max ou moyenne locale, sans paramètres appris. Réduit la résolution et apporte une certaine robustesse aux petites translations.

Convolution avec stride

Downsampling appris. Le filtrage et la réduction de résolution sont optimisés ensemble.

Le pooling ne réduit pas directement le nombre de paramètres des convolutions précédentes et ne garantit pas, à lui seul, une meilleure généralisation.

Bloc résiduel



Les connexions résiduelles facilitent l'optimisation de réseaux très profonds. Le « problème de dégradation » désigne une hausse de l'erreur d'entraînement lorsque la profondeur augmente dans un réseau non résiduel.

Source : He et al., 2016.

ResNet : deux blocs

Bloc	Chemin résiduel	Usage
Basic block	$3 \times 3 \to 3 \times 3$	ResNet-18, ResNet-34
Bottleneck	$1 \times 1 \to 3 \times 3 \to 1 \times 1$	ResNet-50 et au-delà
Projection	convolution 1×1 sur le raccourci	changement de canaux ou de résolution

L'addition impose des dimensions identiques ; une projection aligne les tenseurs lorsque nécessaire.

Depthwise separable convolution

Convolution standard

$K^2 C_{in} C_{out}$ paramètres (hors biais).

Depthwise + pointwise

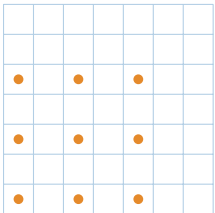
$K^2 C_{in} + C_{in} C_{out}$ paramètres (hors biais).

Ces formules supposent un depth multiplier égal à 1.

Cette factorisation réduit fortement le coût, au prix d'une contrainte structurelle sur le mélange spatial et intercanal.

Convolution dilatée

Une dilatation $D > 1$ espace les points du noyau sans ajouter de paramètres.



noyau 3×3, dilatation 2

Elle agrandit le champ réceptif et préserve la résolution, mais des motifs de grille peuvent apparaître si les dilatations sont mal combinées.

Transfer learning

1. Charger un réseau pretrained sur une source large.
2. Remplacer la tête de prédiction.
3. Entraîner d'abord la tête avec le backbone gelé.
4. Dégeler progressivement et affiner avec un learning rate plus faible.
5. Valider les transformations d'entrée et le décalage de domaine.

Remarque

Le pretraining n'assure pas un gain : une source trop éloignée, des biais hérités ou une normalisation inadéquate peuvent dégrader les résultats.

Classification, détection, segmentation

Tâche	Sortie	Mesures fréquentes
Classification	une distribution par image	exactitude, F1, log-loss
Détection	boîtes, classes, scores	mAP selon seuils d'IoU
Segmentation	une classe par pixel	IoU, Dice

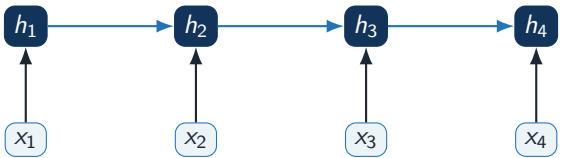
Les sorties, les pertes et les métriques doivent correspondre à la structure de la tâche ; une forte exactitude ne résume pas la qualité d'une segmentation ou d'une détection.

4. Transformers

Séquences, self-attention et calcul parallèle

Recurrent neural network

$$\boldsymbol{h}_t = \phi(\boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}), \qquad \boldsymbol{y}_t = g(\boldsymbol{h}_t).$$



Les mêmes paramètres sont partagés dans le temps. Le calcul est séquentiel, ce qui limite le parallélisme.

Backpropagation through time

La contribution d'un état ancien contient un produit ordonné de Jacobiennes. En posant $J_k = \partial \boldsymbol{h}_k / \partial \boldsymbol{h}_{k-1}$:

$$\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{h}_t} = J_T J_{T-1} \cdots J_{t+1}.$$

Vanishing gradients

Si $\|J_k\|_2 \leq \rho < 1$ sur de nombreuses étapes, la norme du produit décroît exponentiellement.

Exploding gradients

Si le produit amplifie durablement certaines directions, sa norme peut croître exponentiellement.

Le gradient clipping traite l'explosion ; les portes, les chemins additifs et les connexions résiduelles facilitent les dépendances longues.

Cellule LSTM

$$\begin{aligned}
\boldsymbol{f}_t &= \sigma(W_f[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_f), & \boldsymbol{i}_t &= \sigma(W_i[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_i),\\
\tilde{\boldsymbol{c}}_t &= \tanh(W_c[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_c), & \boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t,\\
\boldsymbol{o}_t &= \sigma(W_o[\boldsymbol{h}_{t-1}; \boldsymbol{x}_t] + \boldsymbol{b}_o), & \boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t).
\end{aligned}$$

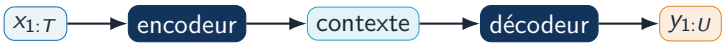
Les portes sont des coefficients de contrôle ; les états sont $\boldsymbol{c}_t$ et $\boldsymbol{h}_t$. Le chemin additif de $\boldsymbol{c}_t$ facilite la propagation du gradient.

GRU

$$
\begin{aligned}
\boldsymbol{z}_t &= \sigma(W_z \boldsymbol{x}_t + U_z \boldsymbol{h}_{t-1}), \\
\boldsymbol{r}_t &= \sigma(W_r \boldsymbol{x}_t + U_r \boldsymbol{h}_{t-1}), \\
\tilde{\boldsymbol{h}}_t &= \tanh(W_h \boldsymbol{x}_t + U_h(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})), \\
\boldsymbol{h}_t &= \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1} + (1 - \boldsymbol{z}_t) \odot \tilde{\boldsymbol{h}}_t.
\end{aligned}
$$

La GRU fusionne mémoire et état caché. Elle utilise moins de portes qu'une LSTM ; la meilleure cellule dépend de la tâche et du budget.

Encoder-decoder

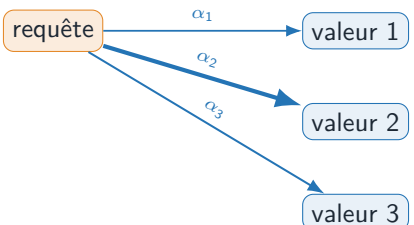


Pendant l'entraînement, le **teacher forcing** fournit souvent le vrai token précédent. À l'inférence, le décodeur utilise ses propres sorties : ce décalage peut accumuler les erreurs.

Attention

Pour une requête $\boldsymbol{q}$, des clés $\boldsymbol{k}_j$ et des valeurs $\boldsymbol{v}_j$:

$$e_j = \frac{\boldsymbol{q}^\top \boldsymbol{k}_j}{\sqrt{d_k}}, \qquad \alpha_j = \frac{e^{e_j}}{\sum_m e^{e_m}}, \qquad \boldsymbol{c} = \sum_j \alpha_j \boldsymbol{v}_j.$$



Le facteur $1/\sqrt{d_k}$ évite que les logits deviennent trop dispersés lorsque la dimension augmente.

Scaled dot-product attention

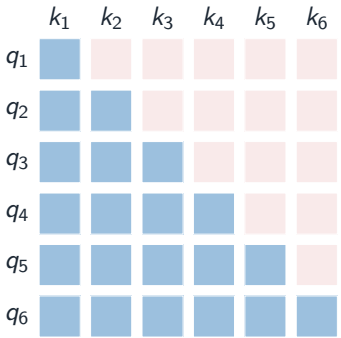
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right) V.$$

Si une séquence contient T éléments :

$$Q \in \mathbb{R}^{T_q \times d_k}, \quad K \in \mathbb{R}^{T_k \times d_k}, \quad V \in \mathbb{R}^{T_k \times d_v}, \quad A \in \mathbb{R}^{T_q \times T_k}.$$

Le masque M vaut typiquement 0 pour une position autorisée et une très grande valeur négative pour une position interdite.

Causal mask



Une requête en position t ne peut lire que les clés de positions $\leq t$. Le décodage reste autorégressif.

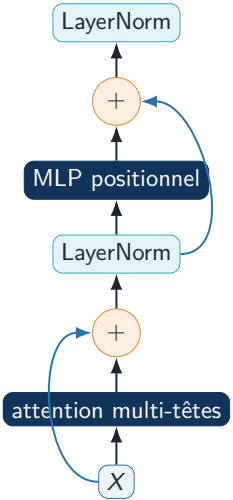
Le padding mask est distinct : il exclut les positions ajoutées uniquement pour former des mini-batches rectangulaires.

Multi-head attention

$$\text{head}_h = \text{Attention}(QW_h^Q, KW_h^K, VW_h^V),$$
$$\text{MHA}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)W^O.$$

- Chaque tête possède ses projections apprises.
- Les têtes peuvent représenter des relations différentes, sans garantie d'interprétabilité.
- Si la dimension totale reste d_{model}, multiplier les têtes ne multiplie pas nécessairement le coût des projections par H.

Bloc Transformer



- attention : mélange entre positions ;
- MLP : transformation de chaque position ;
- résidus : chemins courts pour l'optimisation ;

Positional encoding

Sans information de position, la self-attention est équivariante aux permutations.

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d}\right), \quad PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d}\right).$$

Type	Principe
Absolu appris	un vecteur appris par position
Sinusoïdal	fonctions fixes à plusieurs fréquences
Relatif / rotatif	encode les décalages ou rotations dépendant de la position

Source : Vaswani et al., 2017.

Trois familles de Transformers

Architecture	Masque et objectif	Exemples d'usage
Encodeur seul	contexte bidirectionnel, masquage	classification, représentation
Décodeur seul	masque causal, prédiction du prochain token	génération autorégressive
Encodeur-décodeur	encodage complet + décodage causal	traduction, transduction

Ces catégories décrivent l'architecture et l'objectif ; elles ne déterminent pas à elles seules la qualité d'un modèle.

Coût de la self-attention

Pour une longueur T et une dimension d :

$$QK^\top : O(T^2 d), \qquad A \in \mathbb{R}^{T \times T}.$$

Conséquence

La mémoire des poids d'attention croît quadratiquement avec la longueur.

Réponses possibles

Fenêtres locales, attention creuse, approximation linéaire, compression ou traitement par blocs.

Le coût exact dépend aussi des projections, du MLP, du nombre de têtes, du batch et de l'implémentation mémoire.

Parameter-efficient fine-tuning

Fine-tuning complet

Met à jour tous les paramètres. Flexible, mais coûteux en mémoire et en stockage.

LoRA

Apprend une mise à jour de faible rang $\Delta W = BA$ avec $r \ll d$, en gelant généralement W.

$$W' = W + \frac{\alpha}{r} BA.$$

L'économie porte surtout sur les paramètres entraînables et les états de l'optimiseur ; l'inférence n'est pas automatiquement moins coûteuse.

Source : Hu et al., 2022.

Contrastive learning

Dans un batch de paires positives $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, une perte de type InfoNCE rapproche les paires appariées et éloigne les autres :

$$\ell_i = -\log \frac{\exp(s(\boldsymbol{x}_i, \boldsymbol{y}_i)/\tau)}{\sum_j \exp(s(\boldsymbol{x}_i, \boldsymbol{y}_j)/\tau)}.$$

- s est souvent une similarité cosinus entre représentations normalisées ;
- τ contrôle la concentration des scores ;
- CLIP combine les pertes image-vers-texte et texte-vers-image.

Source : Radford et al., 2021.

5. NLP

Des symboles discrets aux représentations contextuelles

Fréquences et vocabulaire

Dans de nombreux corpus, la fréquence du mot de rang r suit approximativement une loi de Zipf :

$$f(r) \propto r^{-s}, \qquad s \approx 1.$$

- Quelques formes concentrent une grande partie des occurrences.
- La longue traîne contient de nombreuses formes rares.
- Normalisation, vocabulaire et tokenization déterminent la part de cette traîne conservée.
- La loi est empirique et approximative ; son exposant dépend du corpus et du preprocessing.

Bag-of-words et TF-IDF

Pour un document d et un terme t :

$$\mathrm{tfidf}(t, d) = \mathrm{tf}(t, d) \log \frac{N}{\mathrm{df}(t)}.$$

- Représentation sparse, simple et souvent très compétitive sur petits corpus.
- Les n-grams ajoutent un contexte local limité.
- L'ordre global et la polysémie restent mal représentés.
- Le vocabulaire doit être appris uniquement sur le train split pour éviter la fuite d'information.

TF-IDF constitue une baseline importante avant de conclure à l'intérêt d'un modèle profond.

Topic modeling

Latent Dirichlet Allocation suppose que chaque document mélange des topics et que chaque topic définit une distribution sur les mots.

$$\boldsymbol{\theta}_d \sim \text{Dirichlet}(\boldsymbol{\alpha}), \qquad z_{dn} \sim \text{Categorical}(\boldsymbol{\theta}_d), \qquad w_{dn} \sim \text{Categorical}(\boldsymbol{\beta}_{z_{dn}}).$$

Les topics sont des variables latentes statistiques, pas des thèmes garantis sémantiquement. Leur nombre, le preprocessing et les priors modifient fortement l'interprétation.

Représenter des symboles

Un modèle ne manipule pas directement des mots ou des tokens : il reçoit des indices, puis apprend une table d'embeddings.

$$E \in \mathbb{R}^{|\mathcal{V}| \times d}, \qquad \boldsymbol{e}_i = E[i, :].$$

One-hot

Vecteur de dimension $|\mathcal{V}|$, orthogonal aux autres symboles et sans notion de proximité.

Embedding dense

Vecteur appris de dimension $d \ll |\mathcal{V}|$; la géométrie dépend de l'objectif d'entraînement.

La proximité dans l'espace d'embedding n'a de sens qu'au regard des données, de l'objectif et de la métrique utilisés.

Tokenization

La tokenization définit le vocabulaire et la séquence effectivement vue par le modèle.

Unité	Avantage	Limite
Mot	interprétation directe	vocabulaire très large, mots inconnus
Caractère	vocabulaire compact	séquences longues, caractères inconnus possibles
Byte	couverture complète avec 256 symboles	séquences souvent très longues
Subword	compromis taille-longueur	segmentation dépendante du corpus

La tokenization affecte le coût, la longueur de contexte, le traitement des langues et la représentation des nombres ou du code.

BPE

Byte Pair Encoding construit un vocabulaire de subwords par fusions successives.

1. Initialiser les mots comme séquences de caractères ou de bytes.
2. Compter les paires adjacentes dans le corpus.
3. Fusionner la paire la plus fréquente.
4. Répéter jusqu'à la taille de vocabulaire souhaitée.

Exemple simplifié

Si la paire `e s` est fréquente, elle devient un nouveau symbole `es`. Les fusions apprises sont ensuite appliquées dans le même ordre au texte à encoder.

WordPiece choisit les unités selon un critère probabiliste ; SentencePiece peut apprendre directement sur le texte brut.

Matrice de cooccurrence

On définit M_{ij} à partir du nombre de fois où le mot j apparaît dans une fenêtre autour du mot i.

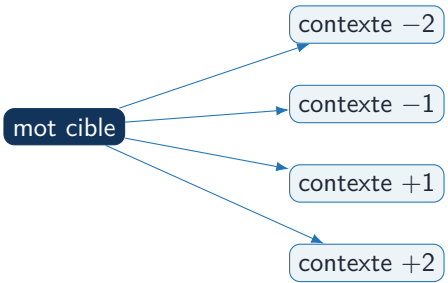
$$M \approx U_r \Sigma_r V_r^\top.$$

Une représentation classique utilise $U_r \Sigma_r^\alpha$, avec $\alpha = 1$ ou $1/2$ selon la construction.

- Une pondération de type PMI réduit l'effet des mots très fréquents.
- La factorisation capture des régularités globales du corpus.
- Les embeddings obtenus restent statiques : un mot garde le même vecteur dans tous ses contextes.

Hypothèse distributionnelle

Des mots employés dans des contextes similaires tendent à avoir des usages proches.



Cette hypothèse relie cooccurrences et géométrie, mais la similarité distributionnelle ne se confond pas toujours avec la synonymie.

Word2Vec

Word2Vec apprend deux tables : les embeddings d'entrée V et les embeddings de sortie U.

CBOW

Prédire le mot central à partir des mots du contexte.

$$p(w_t \mid C_t).$$

Skip-gram

Prédire les mots du contexte à partir du mot central.

$$\prod_{j \in C_t} p(w_j \mid w_t).$$

Les deux tables ne sont pas interchangeables pendant l'entraînement ; l'embedding final est souvent V, ou une combinaison de V et U.

Source : Mikolov et al., 2013.

Negative sampling

Pour une paire positive (c, o) et K négatifs $n_k \sim P_n$:

$$\log \sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c) + \sum_{k=1}^{K} \log \sigma(-\boldsymbol{u}_{n_k}^\top \boldsymbol{v}_c).$$

- $\boldsymbol{v}_c$ est l'embedding d'entrée du mot central.
- $\boldsymbol{u}_o$ et $\boldsymbol{u}_{n_k}$ sont des embeddings de sortie.
- P_n est souvent proportionnelle à la fréquence élevée à la puissance $3/4$.

Le negative sampling remplace une normalisation sur tout le vocabulaire par un problème de discrimination locale.

Limites des embeddings statiques

Le même vecteur représente un mot dans tous ses contextes.

Exemple

Dans « l'avocat plaide au tribunal » et « l'avocat est mûr », le sens dépend de la phrase alors que Word2Vec fournit le même embedding lexical.

Les embeddings contextuels calculent au contraire une représentation par occurrence :

$$\boldsymbol{h}_t = f_\theta(w_1, \ldots, w_T)_t.$$

Le contexte peut ainsi modifier la représentation, mais il ne garantit pas une désambiguïsation parfaite.

Language modeling

Un modèle autoregressif factorise la probabilité d'une séquence :

$$p(w_{1:T}) = \prod_{t=1}^{T} p(w_t \mid w_{<t}).$$

La cross-entropy moyenne est

$$\mathcal{L} = -\frac{1}{T} \sum_{t=1}^{T} \log p_\theta(w_t \mid w_{<t}),$$

et la perplexity vaut $\exp(\mathcal{L})$ lorsque les unités et la base logarithmique sont fixées.

Remarque

Une perplexity n'est comparable qu'avec la même tokenization, le même corpus et le même protocole d'évaluation.

Masked language modeling

L'encodeur reçoit une séquence partiellement masquée et prédit les tokens sélectionnés.

$$\mathcal{L}_{MLM} = -\sum_{t \in \mathcal{M}} \log p_\theta(w_t \mid w_{\setminus \mathcal{M}}).$$

Dans BERT, 15% des positions sont sélectionnées ; parmi elles, 80% sont remplacées par `[MASK]`, 10% par un token aléatoire et 10% restent inchangées.

- Le contexte est bidirectionnel.
- L'objectif ne définit pas directement une génération left-to-right.
- Des variantes modernes modifient ou remplacent ce protocole.

Source : Devlin et al., 2018.

BERT

BERT est un Transformer encoder pretrained par masked language modeling et, dans la version originale, par Next Sentence Prediction.

Modèle	Architecture	Taille
BERT Base	12 couches, $d = 768$, 12 têtes	110 M paramètres
BERT Large	24 couches, $d = 1024$, 16 têtes	340 M paramètres

Pour NSP, 50% des paires sont réellement consécutives et 50% associent une seconde phrase aléatoire. Des travaux ultérieurs ont montré que NSP n'est pas indispensable dans toutes les variantes.

Source : Devlin et al., 2018 ; Liu et al., 2019.

GPT

Un GPT est un Transformer decoder entraîné à prédire le prochain token avec un causal mask.

1. Pretraining autoregressif sur un grand corpus.
2. Adaptation par fine-tuning, prompting ou in-context learning.
3. Décodage token par token à l'inférence.

La température, top-k et top-p modifient la distribution d'échantillonnage ; ils ne changent pas les paramètres du modèle.

Remarque

La génération fluide ne constitue pas une garantie de factualité, de calibration ou de raisonnement correct.

Instruction tuning

Le pretraining apprend principalement la distribution du texte. L'instruction tuning utilise des paires instruction-réponse pour apprendre un comportement conditionné par la demande.

$$\mathcal{L}_{SFT} = - \sum_{t \in \text{réponse}} \log p_\theta(y_t \mid x, y_{<t}).$$

- Le prompt peut être masqué dans la loss afin d'optimiser seulement la réponse.
- La diversité et la qualité des tâches comptent davantage qu'une simple duplication des exemples.
- Le supervised fine-tuning ne garantit ni factualité ni alignement avec toutes les préférences.

Preference optimization

Des comparaisons de réponses permettent d'orienter le modèle vers des préférences humaines ou métier.

1. Collecter des paires (y^+, y^-) pour un même prompt x.
2. Apprendre un reward model puis optimiser par RL, par exemple PPO ; ou
3. optimiser directement une loss de préférence, par exemple DPO.

Ces méthodes optimisent les préférences représentées dans les données de comparaison. Elles n'établissent pas une notion universelle de qualité ou de vérité.

Source : Ouyang et al., 2022 ; Rafailov et al., 2023.

Encoder, decoder, encoder-decoder

Famille	Pretraining typique	Tâches
Encoder	masked prediction	classification, retrieval, tagging
Decoder	next-token prediction	génération, code, dialogue
Encoder-decoder	denoising ou seq2seq	traduction, résumé, transduction

La frontière n'est pas absolue : les objectifs, les données, l'échelle et le fine-tuning comptent autant que la famille architecturale.

Fine-tuning d'un language model

- Ajouter une classification head ou conserver la language-modeling head.
- Utiliser un learning rate inférieur à celui du pretraining.
- Contrôler le catastrophic forgetting et le surapprentissage.
- Comparer full fine-tuning, adapters et LoRA sous le même budget.
- Évaluer sur les sous-populations, les longueurs et les domaines pertinents.

Le train/validation/test split doit être construit au niveau des documents ou des utilisateurs lorsque des fragments proches risquent de se retrouver dans plusieurs ensembles.

6. Graph neural networks

Apprendre à partir des nœuds, des arêtes et de leur structure

Adjacency et features

Pour n nœuds, une matrice d'adjacence $A \in \mathbb{R}^{n \times n}$ encode les arêtes et $X \in \mathbb{R}^{n \times d}$ les node features.

- Graphe non orienté : $A = A^\top$.
- Graphe pondéré : A_{ij} porte un poids réel.
- Graphe dirigé : les voisinages entrant et sortant sont distincts.
- Multi-relations : une adjacency ou une edge list par type de relation.

Une permutation des nœuds transforme (A, X) en $(PAP^\top, PX)$ sans changer le graphe abstrait.

Données de graphe

Un graphe $G = (V, E)$ peut porter des features de nœuds X, des features d'arêtes et plusieurs types de relations.

Niveau	Exemples de tâches	Sortie
Nœud	classification, détection d'anomalies	une prédiction par nœud
Arête	link prediction, scoring	une prédiction par paire
Graphe	molécule, document, réseau	une prédiction globale

L'ordre des nœuds dans une matrice d'adjacence est arbitraire : une architecture de graphe doit respecter les permutations de cet ordre.

Node embeddings

On cherche des vecteurs $\boldsymbol{z}_v \in \mathbb{R}^d$ qui préservent une notion de proximité définie par la structure ou les features.

$$s(u, v) = \boldsymbol{z}_u^\top \boldsymbol{z}_v \quad \text{ou} \quad s(u, v) = \mathrm{MLP}([\boldsymbol{z}_u; \boldsymbol{z}_v]).$$

- Méthodes shallow : une table de paramètres par nœud.
- GNN : une fonction partagée calcule l'embedding à partir du voisinage.
- Les méthodes transductives n'encodent pas automatiquement un nouveau nœud absent de l'entraînement.

DeepWalk

DeepWalk transforme le graphe en corpus de random walks, puis applique Skip-gram.

1. Lancer plusieurs random walks depuis chaque nœud.
2. Considérer les nœuds proches dans la marche comme un contexte.
3. Optimiser Skip-gram avec hierarchical softmax dans l'article original ; le negative sampling est une alternative courante.

La méthode préserve une proximité induite par les marches, pas toutes les propriétés du graphe. Elle est transductive et ne mobilise pas nécessairement les node features.

Source : Perozzi et al., 2014.

Node2Vec

Après une transition $t \to v$, la probabilité non normalisée d'aller vers un voisin x de v est

$$\pi_{vx} = \alpha_{pq}(t,x)\, w_{vx}, \qquad \alpha_{pq}(t,x) = \begin{cases} 1/p, & d(t,x) = 0, \\ 1, & d(t,x) = 1, \\ 1/q, & d(t,x) = 2. \end{cases}$$

p contrôle le retour immédiat. Pour $q > 1$, les candidats proches de t sont favorisés ; pour $q < 1$, les candidats à distance 2 le sont davantage. La distance est mesurée entre le nœud précédent t et le candidat x.

Source : Grover et Leskovec, 2016.

Message passing

Une couche de GNN agrège les messages des voisins puis met à jour chaque représentation :

$$\boldsymbol{m}_v^{(\ell)} = \mathrm{AGG}^{(\ell)}\left(\left\{\psi^{(\ell)}(\boldsymbol{h}_v^{(\ell)}, \boldsymbol{h}_u^{(\ell)}, \boldsymbol{e}_{uv}) : u \in \mathcal{N}(v)\right\}\right),$$

$$\boldsymbol{h}_v^{(\ell+1)} = \phi^{(\ell)}(\boldsymbol{h}_v^{(\ell)}, \boldsymbol{m}_v^{(\ell)}).$$

L'agrégation doit être invariante à l'ordre des voisins : somme, moyenne, maximum ou attention normalisée.

GCN

Avec $\tilde{A} = A + I$ et $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$:

$$H^{(\ell+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(\ell)}W^{(\ell)}\right).$$

- Les self-loops conservent l'information propre du nœud.
- La normalisation symétrique contrôle l'échelle des agrégations.
- Après L couches, un nœud dépend au plus de son voisinage à L hops.

Un GCN n'est pas, en général, une succession obligatoire « convolution, pooling, fully connected » : l'architecture dépend de la tâche.

Source : Kipf et Welling, 2017.

Laplacien de graphe

Pour $D_{ii} = \sum_j A_{ij}$:

$$L = D - A, \qquad L_{sym} = I - D^{-1/2} A D^{-1/2}.$$

Pour un graphe non orienté à poids non négatifs :

$$\boldsymbol{x}^\top L \boldsymbol{x} = \frac{1}{2} \sum_{i,j} A_{ij} (x_i - x_j)^2 \geq 0.$$

Le Laplacien pénalise les variations le long des arêtes. Les convolutions spectrales et le GCN peuvent être reliés à des filtres ou approximations construits à partir de cet opérateur.

GraphSAGE

GraphSAGE apprend une fonction d'agrégation réutilisable sur de nouveaux nœuds :

$$\boldsymbol{h}_v^{(\ell+1)} = \sigma\Big(W^{(\ell)}[\boldsymbol{h}_v^{(\ell)}; \mathrm{AGG}\{\boldsymbol{h}_u^{(\ell)} : u \in \mathcal{N}(v)\}]\Big).$$

- Sampling d'un nombre borné de voisins pour contrôler le coût.
- Agrégateurs mean, pooling ou LSTM dans l'article initial ; l'agrégateur LSTM n'est pas permutation-invariant et dépend de l'ordre choisi.
- Capacité inductive si les features permettent d'encoder les nouveaux nœuds.

Source : Hamilton et al., 2017.

Graph Attention Network

Pour une arête (i, j) :

$$e_{ij} = \mathrm{LeakyReLU}\left(\boldsymbol{a}^\top [W\boldsymbol{h}_i \,\|\, W\boldsymbol{h}_j]\right), \qquad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}(i)} \exp(e_{ik})},$$

$$\boldsymbol{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij} W\boldsymbol{h}_j\right).$$

Les coefficients sont appris localement sur le voisinage. Ils pondèrent les messages, mais ne constituent pas automatiquement une explication causale.

Source : Veličković et al., 2018.

Graph-level prediction

Pour prédire une propriété du graphe, il faut un readout invariant aux permutations :

$$\boldsymbol{h}_G = \mathrm{READOUT}\{\boldsymbol{h}_v^{(L)} : v \in V\}.$$

Readout	Principe	Remarque
Somme	addition des node embeddings	sensible à la taille
Moyenne	représentation moyenne	peut perdre le comptage
Attention	pondération apprise	coût et interprétation à contrôler
Pooling hiérarchique	coarsening du graphe	structure additionnelle apprise

Limites des GNN

- **Homophily bias :** agréger les voisins aide moins lorsque les labels ou features diffèrent fortement le long des arêtes.
- **Over-smoothing :** les représentations deviennent trop semblables avec la profondeur.
- **Over-squashing :** trop d'information distante est comprimée dans un vecteur de taille fixe.
- **Scalabilité :** la taille du voisinage peut croître rapidement.
- **Expressivité :** les MPNN standards ont des limites liées au test de Weisfeiler-Lehman.

Ces phénomènes sont distincts et appellent des diagnostics différents.

Graphes hétérogènes

Un graphe hétérogène contient plusieurs types de nœuds ou de relations.

$$\boldsymbol{m}_v = \sum_{r \in \mathcal{R}} \sum_{u \in \mathcal{N}_r(v)} \alpha_{uv}^{(r)} W_r \boldsymbol{h}_u.$$

- Paramètres ou transformations spécifiques aux relations.
- Schéma de métapaths ou attention entre types selon le modèle.
- Gestion explicite des directions et des types d'arêtes.

Un GAT vanilla ne traite pas automatiquement cette hétérogénéité ; il faut adapter l'architecture, par exemple avec R-GCN ou HGT.

7. Modèles génératifs

Apprendre une distribution et produire de nouveaux exemples

Modélisation générative

On cherche à approximer la distribution des données $p_{data}(\boldsymbol{x})$, éventuellement conditionnée par $\boldsymbol{c}$.

Densité explicite ou borne

Modèles autorégressifs, VAE, flows : objectif lié à la vraisemblance.

Objectif indirect

GAN : discrimination adversariale ; diffusion : débruitage ou estimation du score.

La fidélité, la diversité, la couverture des modes et le coût d'échantillonnage sont des critères distincts.

Autoencodeur

$$\min_{\phi,\theta} \mathbb{E}_x \, \ell\big(x, g_\theta(f_\phi(x))\big).$$

La dimension latente n'est pas nécessairement inférieure à d : la contrainte peut venir de la parcimonie, du bruit, de la régularisation ou de l'architecture.

Autoencodeur variationnel

Le modèle génératif est $p(\boldsymbol{z})p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$; l'encodeur approxime le posterior par $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$.

$$\log p_\theta(\boldsymbol{x}) \geq \underbrace{\mathbb{E}_{q_\phi}[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})]}_{\text{reconstruction}} - \underbrace{D_{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p(\boldsymbol{z}))}_{\text{régularisation}}.$$

Cette borne inférieure, l'ELBO, est maximisée conjointement par l'encodeur et le décodeur.

Reparamétrisation

Pour un posterior gaussien diagonal :

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\sigma}^2_\phi(\boldsymbol{x}))),$$

$$\varepsilon \sim \mathcal{N}(0, I), \qquad \boldsymbol{z} = \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \boldsymbol{\sigma}_\phi(\boldsymbol{x}) \odot \varepsilon.$$

Le bruit est ainsi isolé dans ε ; $\boldsymbol{z}$ devient une fonction différentiable des sorties de l'encodeur.

Remarque

Un coefficient β devant le terme KL modifie le compromis reconstruction-structure latente ; il ne garantit pas une représentation « désenchevêtrée » sans hypothèses supplémentaires.

Processus de diffusion

Le processus avant ajoute progressivement du bruit gaussien :

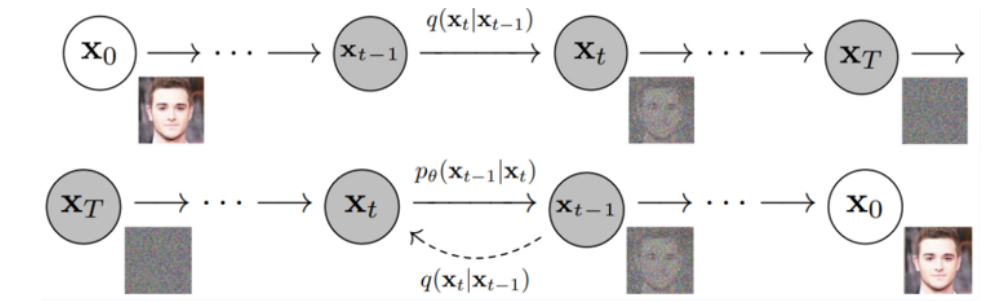
$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)I), \quad \alpha_t = 1 - \beta_t.$$

Avec $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, I).$$

Cette forme fermée permet d'échantillonner directement un niveau de bruit t pendant l'entraînement.

Forward et reverse process



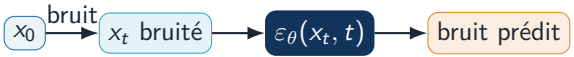
Le forward process q ajoute progressivement du bruit avec des transitions fixées. Le reverse process p_θ apprend les transitions de débruitage à partir d'un bruit gaussien.

Source : Schéma adapté du formalisme de Ho, Jain et Abbeel, 2020.

Apprendre le débruitage

Une paramétrisation courante prédit le bruit ajouté :

$$\mathcal{L}_{simple} = \mathbb{E}_{\boldsymbol{x}_0, t, \boldsymbol{\varepsilon}} \left[\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)\|_2^2 \right].$$



La fonction de débruitage est souvent un U-Net conditionné par un encodage du temps.

Source : Ho et al., 2020.

Échantillonnage

1. Initialiser $\boldsymbol{x}_T \sim \mathcal{N}(0, I)$.
2. Pour $t = T, \ldots, 1$, estimer le bruit ou le score.
3. Construire la moyenne de la transition inverse.
4. Ajouter le bruit prévu par la variance de la transition, sauf au dernier pas.

Le processus inverse n'est pas une inversion déterministe du bruitage observé. Des échantillonneurs déterministes ou d'ordre supérieur réduisent le nombre de pas, avec un compromis qualité-coût.

Classifier-free guidance

Le classifier-free guidance combine une prédiction conditionnelle et une prédiction non conditionnelle :

$$\hat{\varepsilon} = \varepsilon_\theta(\boldsymbol{x}_t, t, \varnothing) + s\left[\varepsilon_\theta(\boldsymbol{x}_t, t, \boldsymbol{c}) - \varepsilon_\theta(\boldsymbol{x}_t, t, \varnothing)\right].$$

$s = 0$

Prédiction non conditionnelle.

$s = 1$

Prédiction conditionnelle du modèle.

Pour $s > 1$, le guidance renforce généralement la condition, au prix d'une diversité souvent plus faible. Le modèle apprend le cas non conditionnel en supprimant aléatoirement la condition pendant l'entraînement.

Source : Ho et Salimans, 2022.

Latent diffusion

La diffusion latente déplace le processus dans une représentation comprimée. Elle réduit le coût spatial, mais la qualité dépend aussi de l'autoencodeur et de son compromis de compression.

Source : Rombach et al., 2022.

Synthèse

1. Un réseau profond est un graphe de calcul paramétré et différentiable.
2. Initialisation, optimisation et régularisation déterminent la dynamique d'entraînement.
3. Les CNN exploitent la localité ; l'attention apprend des interactions globales.
4. Les embeddings dépendent de leur objectif ; les language models les contextualisent.
5. Les GNN construisent des représentations par message passing ; leurs agrégateurs doivent respecter les symétries du graphe.
6. Les VAE optimisent une borne de vraisemblance ; la diffusion apprend un processus de débruitage.

Références (1/2)

- Goodfellow, Bengio, Courville. *Deep Learning*, 2016.
- He et al. Deep Residual Learning, 2016.
- Ioffe et Szegedy. Batch Normalization, 2015.
- Santurkar et al. How Does Batch Normalization Help Optimization?, 2018.
- Vaswani et al. Attention Is All You Need, 2017.
- Hu et al. LoRA, 2022.
- Radford et al. Learning Transferable Visual Models From Natural Language Supervision, 2021.
- Mikolov et al. Efficient Estimation of Word Representations, 2013.
- Devlin et al. BERT, 2018.
- Ouyang et al. Training Language Models to Follow Instructions, 2022.
- Rafailov et al. Direct Preference Optimization, 2023.

Références (2/2)

- Perozzi et al. DeepWalk, 2014.
- Grover et Leskovec. Node2Vec, 2016.
- Kipf et Welling. Semi-Supervised Classification with GCNs, 2017.
- Hamilton et al. Inductive Representation Learning on Large Graphs, 2017.
- Veličković et al. Graph Attention Networks, 2018.
- Kingma et Welling. Auto-Encoding Variational Bayes, 2014.
- Ho et al. Denoising Diffusion Probabilistic Models, 2020.
- Ho et Salimans. Classifier-Free Diffusion Guidance, 2022.
- Rombach et al. High-Resolution Image Synthesis with Latent Diffusion Models, 2022.
- Loshchilov et Hutter. Decoupled Weight Decay Regularization, 2019.

Deep learning

Redha Moulla

redha.moulla@axia-conseil.com