ARCHITECTURE · CALCUL CONDITIONNEL

Mélange d’experts et routage top-k

Un mélange d’experts confie chaque entrée à quelques sous-réseaux choisis par un routeur. Le nombre de paramètres augmente avec le nombre d’experts, mais le calcul par jeton ne dépend que de k.

Avancé · 11 min · MoE · routage · top-k · parcimonie · Exemple — non révisé par l’auteur

Sheet by Simon-Pierre Boucher

sheet.spboucher.ai/s/melange-experts-routage-top-k
17 novembre 2026

Dans un réseau classique, **tous** les paramètres travaillent pour **chaque** entrée. Un mélange d’experts (*Mixture of Experts*, MoE) rompt avec cette règle : un petit réseau de routage choisit, pour chaque entrée, quelques sous-réseaux spécialisés et laisse les autres au repos [3]. L’idée remonte aux années 1990 [1] ; sa version parcimonieuse est celle qu’on retrouve dans les grands modèles de langage.

Des experts

N sous-réseaux de même forme mais de paramètres différents. Dans un Transformer, chacun est un réseau à propagation avant.

Un routeur

Un réseau très léger qui attribue un score à chaque expert à partir de l’entrée, puis n’en retient que k.

Un équilibrage

Une perte auxiliaire qui pousse le routeur à répartir le travail, sans quoi quelques experts reçoivent tout.

Noter

Une multiplication matricielle donne N scores. Son coût est négligeable devant celui d’un expert.

Sélectionner

On retient k experts, avec $k \ll N$. Certains modèles routent chaque jeton vers un seul expert [5].

Calculer

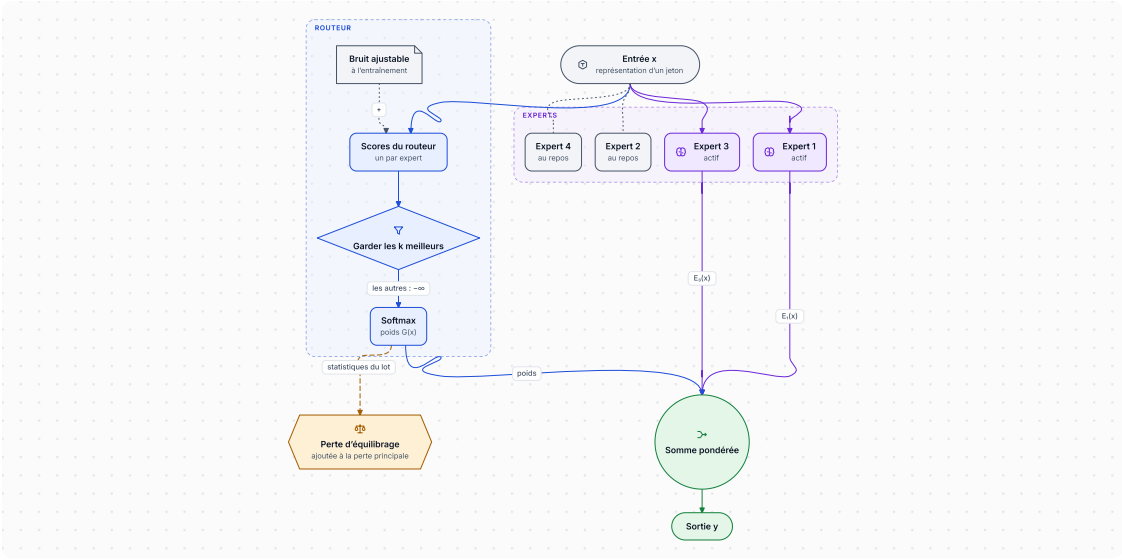
Seuls les experts retenus sont exécutés, chacun sur les jetons qui lui ont été attribués.

Recombiner

Les sorties sont pondérées, additionnées, puis la couche suivante reprend comme dans un Transformer ordinaire.

01 Anatomie d’une couche MoE

Routage top-k avec N = 4 experts et k = 2



Décision de routage · Calcul effectué par un expert actif · Expert au repos : chemin non emprunté, aucun calcul · Signal d’entraînement auxiliaire · Sortie de la couche

1 **Noter chaque expert**
— Le routeur multiplie l’entrée par une matrice apprise W_g : un score par expert. À l’entraînement, un bruit dont l’ampleur est elle-même apprise s’y ajoute.

2 **Ne garder que k experts**
— Les k meilleurs scores sont conservés, les autres sont mis à $-\infty$. Après le softmax, leur poids vaut exactement 0.

3 **N’évaluer que les experts retenus**
— L’entrée n’est envoyée qu’aux experts 1 et 3. Les experts 2 et 4 ne calculent rien : c’est là que se fait l’économie.

4 **Combiner les sorties**
— La sortie est la somme des sorties des experts actifs, pondérée par les poids du routeur.

5 **Surveiller la répartition**
— Sur l’ensemble du lot, on mesure la charge reçue par chaque expert ; une perte auxiliaire pénalise les déséquilibres.

Seuls les experts retenus par le routeur sont évalués ; leurs sorties sont combinées avec les poids du routeur [3].

Sources : 3

02 Le routage en équations

ƒ SORTIE DE LA COUCHE

$$y = \sum_{i=1}^{N} G(x)_i \, E_i(x)$$

Quand $G(x)_i = 0$, il est inutile de calculer $E_i(x)$: la parcimonie du vecteur $G(x)$ se traduit directement en calcul évité [3].

x	Entrée de la couche (représentation d'un jeton)
E_i	Expert i, un réseau de neurones avec ses propres paramètres
$G(x)_i$	Poids que le routeur accorde à l'expert i ; les poids somment à 1
N	Nombre d'experts

Sources : 3

ƒ ROUTAGE TOP-K BRUITÉ

$$G(x) = \mathrm{Softmax}\big(\mathrm{KeepTopK}(H(x), k)\big)$$

$$H(x)_i = (x \cdot W_g)_i + \varepsilon_i \cdot \mathrm{Softplus}\big((x \cdot W_{\text{noise}})_i\big), \quad \varepsilon_i \sim \mathcal{N}(0,1)$$

$$\mathrm{KeepTopK}(v,k)_i = \begin{cases} v_i & \text{if } v_i \text{ is among the top } k \text{ elements of } v \\ -\infty & \text{otherwise} \end{cases}$$

Le bruit, dont l'ampleur par expert est apprise, aide à répartir la charge pendant l'entraînement [3].

W_g	Matrice de poids du routeur
W_{noise}	Matrice apprise qui règle l'ampleur du bruit par expert
k	Nombre d'experts activés par entrée
$\mathrm{Softplus}$	Fonction $z \mapsto \ln(1 + e^z)$, toujours positive

Sources : 3

DÉFINITION

Où se place la couche MoE ?

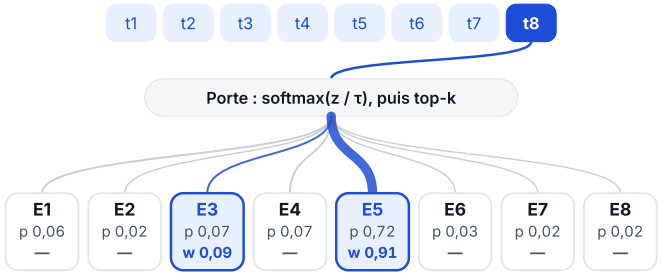
Dans les Transformers à experts, la couche MoE **remplace le réseau à propagation avant** de certains blocs ; l'attention reste dense. Le routage se fait jeton par jeton, de sorte que deux jetons d'une même phrase peuvent emprunter des experts différents [4] [5].

Router des jetons vers des experts

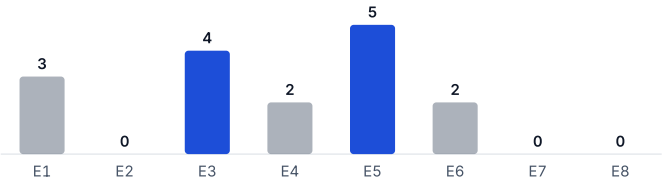
Données synthétiques — graine 11

Pour chaque jeton, la porte calcule un softmax sur N experts, ne garde que les k plus probables et renormalise leurs poids ; la sortie est la somme pondérée de ces k experts. Simplification : les logits de la porte sont tirés au hasard (préférence fixe par expert + terme propre au jeton) au lieu d'être produits par un routeur appris.

Jetons



Charge cumulée : jetons routés vers chaque expert



Jeton courant	Experts retenus	Masse de la porte conservée	Entropie de la porte (normalisée)
t8 (8 / 8)	E5, E3	0,788	0,525

Déséquilibre final (max / moyenne)
2,5

Paramètres : Experts N = 8 ; Experts actifs k = 2 ; Jetons = 8 ; Température τ = 1

Simulation à but pédagogique sur des scores tirés au hasard (graine fixe) : ce n'est pas le routeur d'un modèle entraîné.

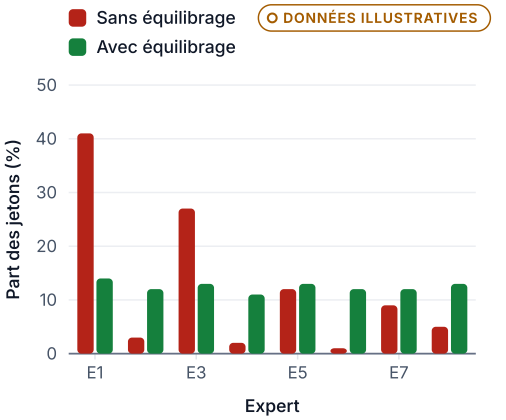
03 Équilibrer la charge

ATTENTION

Le piège : l'effondrement du routage

Sans contrainte, le routeur tend à envoyer presque tout vers les mêmes experts. Le déséquilibre se renforce de lui-même : les experts favorisés s'entraînent davantage, donc sont encore plus souvent choisis [3]. Les autres ne servent plus à rien, et les experts surchargés créent un goulot d'étranglement matériel.

Part des jetons reçue par chaque expert



Valeurs illustratives choisies à la main pour montrer la forme du phénomène ; ce ne sont pas des mesures.

ƒ PERTE D'IMPORTANCE (SHAZEER ET COLL.)

$$\mathrm{Importance}(X) = \sum_{x \in X} G(x)$$

$$L_{\text{importance}}(X) = w_{\text{importance}} \cdot \mathrm{CV}\big(\mathrm{Importance}(X)\big)^2$$

La perte est nulle quand tous les experts reçoivent la même importance totale sur le lot, et croît avec la dispersion [3].

X	Lot d'exemples d'entraînement
CV	Coefficient de variation : écart-type divisé par la moyenne
$w_{\text{importance}}$	Poids de la perte auxiliaire, réglé à la main

Sources : 3

ƒ PERTE AUXILIAIRE (SWITCH TRANSFORMER)

$$L_{\text{aux}} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$$

Sous une répartition uniforme, $f_i = P_i = 1/N$ et la somme atteint son minimum $1/N$. Comme f_i n'est pas dérivable, le gradient passe par P_i [5].

f_i	Fraction des jetons du lot envoyés à l'expert i
P_i	Probabilité moyenne que le routeur attribue à l'expert i sur le lot
α	Coefficient de la perte auxiliaire

Sources : 5

04 Pourquoi cela passe à l'échelle

f PARAMÈTRES ET CALCUL SE DÉCOUPLENT

$$\underbrace{P_{\text{MoE}} \approx N \cdot P_E}_{\text{parameters}} \qquad \underbrace{C_{\text{token}} \approx k \cdot C_E + C_{\text{router}}}_{\text{compute per token}}$$

f_x Résultat dérivé dans cette fiche

Relation dérivée de la définition de la couche : ajouter des experts augmente N, donc la capacité, sans toucher à k, donc au calcul par jeton. Elle néglige le coût de communication entre machines.

P_E	Nombre de paramètres d'un expert
C_E	Coût de calcul d'un expert pour un jeton
C_{router}	Coût du routeur, petit devant C_E

Trois façons d'utiliser N sous-réseaux.

	Couche dense	Mélange doux (1991)	MoE top-k (PARCIMONIEUX)
EXPERTS ÉVALUÉS PAR ENTRÉE	un seul réseau, toujours	tous les N	seulement k
CALCUL PAR JETON	C_E	$N \cdot C_E$	$k \cdot C_E$
PARAMÈTRES	P_E	$N \cdot P_E$	$N \cdot P_E$
DIFFICULTÉ PROPRE	aucune	coût qui croît avec N	équilibrage, sélection non dérivable

VERDICT

Le MoE top-k garde la capacité du mélange doux pour un coût de calcul proche de celui d'une couche dense. C'est ce découplage qui a permis d'entraîner des modèles de plusieurs centaines de milliards de paramètres [4].

HYPOTHÈSE

Ce que le calcul parcimonieux n'économise pas

Tous les experts doivent rester **chargés en mémoire**, puisque n'importe lequel peut être appelé au jeton suivant. Un modèle MoE économise du calcul par jeton, pas de la mémoire : à nombre de paramètres égal, il occupe autant de place qu'un modèle dense.

05 Vérifier sa compréhension

QUESTION

Une couche MoE passe de 8 à 64 experts, en gardant $k = 2$. Que se passe-t-il, à peu de chose près ?

A. Les paramètres de la couche sont multipliés par 8 ; le calcul par jeton change à peine.

B. Les paramètres et le calcul par jeton sont tous deux multipliés par 8.

C. Rien ne change, puisque seuls 2 experts sont utilisés.

VOIR LA RÉPONSE

A — Bonne réponse. Exact : les paramètres suivent N, le calcul suit k. Seul le routeur, très léger, calcule 64 scores au lieu de 8.

B — Mauvaise réponse. Ce serait le cas d'un mélange doux, où tous les experts sont évalués.

C — Mauvaise réponse. Deux experts par jeton, mais pas toujours les mêmes : les 64 existent et doivent être stockés et entraînés.

Les experts sont-ils spécialisés par sujet (médecine, droit, code) ?

Pas par construction. Personne n'attribue de rôle aux experts : la répartition émerge de l'entraînement, et le routage se décide jeton par jeton à partir de la représentation interne. Le mot « expert » désigne un sous-réseau, pas une compétence lisible par un humain.

Quelle est la différence avec un ensemble de modèles ?

Un ensemble évalue **tous** les modèles et moyenne leurs réponses. Un MoE est un seul modèle, entraîné de bout en bout, dans lequel le routeur choisit **quelques** sous-réseaux à l'intérieur d'une couche ; le reste du réseau est partagé.

Comment le routeur apprend-il si la sélection top-k n'est pas dérivable ?

Le gradient passe par les poids des experts **retenus** : ces poids sont une fonction dérivable des scores. Les experts écartés ne reçoivent aucun gradient pour cet exemple, d'où l'utilité du bruit et de la perte d'équilibrage pour que chacun soit essayé [3].

D'où vient l'idée ?

Jacobs, Jordan, Nowlan et Hinton proposent en 1991 un système d'experts mis en concurrence par un réseau de routage, qui apprend à découper une tâche en sous-tâches [1]. Jordan et Jacobs l'étendent en 1994 à une hiérarchie d'experts [2]. La version parcimonieuse à grande échelle date de 2017 [3].

Références

[1] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, Geoffrey E. Hinton (1991). *Adaptive Mixtures of Local Experts.* Neural Computation, 3(1), 79–87. doi:10.1162/neco.1991.3.1.79

[2] Michael I. Jordan, Robert A. Jacobs (1994). *Hierarchical Mixtures of Experts and the EM Algorithm.* Neural Computation, 6(2), 181–214. doi:10.1162/neco.1994.6.2.181

[3] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, Jeff Dean (2017). *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer.* arXiv:1701.06538. arxiv.org/abs/1701.06538

[4] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, Zhifeng Chen (2020). *GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding.* arXiv:2006.16668. arxiv.org/abs/2006.16668

[5] William Fedus, Barret Zoph, Noam Shazeer (2021). *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity.* arXiv:2101.03961. arxiv.org/abs/2101.03961