Université de Bordeaux

Master 2 Informatique - Intelligence Artificielle

Architectures de Réseaux de neurones avancées

Implémentation et Analyse d'un Mixture of Experts (MoE) sur CIFAR-10

Auteurs :
Arthur Macdonald
Niels Roudeau
Repository Github :
https://github.com/Roirtur/RoinisMoEAI

Année Universitaire 2025-2026

Table des matières

1 Présentation du projet

L'objectif de ce projet est d'implémenter, d'entraîner et d'analyser une architecture neuronale de type **Mixture of Experts (MoE)** appliquée au jeu de données CIFAR-10.

Les modèles modernes d'apprentissage profond cherchent constamment à augmenter leur capacité (nombre de paramètres) pour capturer des nuances plus fines dans les données. Cependant, cette augmentation de taille s'accompagne traditionnellement d'une explosion des coûts de calcul. L'approche MoE, popularisée par Shazeer et al. (2017) et récemment réactualisée par Mixtral, propose une solution via le **Calcul Conditionnel** (*Conditional Computation*).

Dans ce rapport, nous détaillerons :

— Nos choix d'architecture (Gating Network, Experts Convolutionnels).

— L'implémentation modulaire en PyTorch.

— Les résultats expérimentaux comparant notre MoE à une *Baseline* Dense, en analysant spécifiquement la spécialisation des experts.

2 Architecture du Projet

Le code complet et la structure du projet sont organisés de manière modulaire pour séparer la logique de modélisation, l'entraînement et l'analyse.

Les composants principaux sont :

— **Le noyau du modèle** dans le dossier `models/` :
  — `gating.py` : Définit le réseau de routage (Router).
  — `experts.py` : Définit l'architecture des sous-réseaux experts.
  — `moe_model.py` : Orchestre le tout (Manager) et gère le routage Top-k.
  — `dense_baseline.py` : Une architecture CNN classique servant de point de comparaison.

— **Les scripts d'exécution** :
  — `train.py` : Script CLI pour lancer les entraînements avec gestion des hyperparamètres.
  — `experiments.ipynb` : Notebook Jupyter contenant le protocole expérimental complet, les boucles d'entraînement et la génération des visualisations.

— **Utilitaires** :
  — `data_loader.py` : Gestion du dataset CIFAR-10 et augmentations.
  — `visualization.py` : Génération des courbes de perte et des heatmaps d'experts.

3 Définition du Modèle MoE

3.1 Représentation et Routage

Contrairement à un modèle dense où chaque couche traite l'intégralité de l'entrée, notre modèle MoE sélectionne dynamiquement quels paramètres utiliser.

L'entrée est une image x de dimension $(3, 32, 32)$.

1. **Le Gating Network ($g(x)$)** : Un petit CNN analyse l'image et produit un vecteur de probabilité sur N experts. Nous utilisons un *Noisy Top-k Gating* (bruit gaussien ajouté à l'entraînement, déterministe à l'inférence).
2. **Sélection (Sparse)** : Seuls les indices k ayant les plus hautes probabilités sont retenus.
3. **Calcul Expert** : L'image est passée uniquement à travers les experts sélectionnés $E_i(x)$.
4. **Agrégation** : La sortie est la somme pondérée des sorties des experts :

$$y = \sum_{i \in \text{Top-k}} g(x)_i \cdot E_i(x)$$

3.2 Architecture des Experts

Chaque expert est un CNN indépendant conçu pour être plus spécialisé qu'un réseau généraliste.

— **Structure :** 3 blocs de convolution ($32 \rightarrow 64 \rightarrow 128$ filtres) avec MaxPool, suivis d'une tête linéaire.

— **Intuition :** Certains experts peuvent se spécialiser sur les textures (animaux), d'autres sur les formes rigides (véhicules).

3.3 Load Balancing (Équilibrage de charge)

Un défi majeur des MoE est d'éviter un scénario de déséquilibre de la spécialisation des experts (Expert Specialization Collapse), où le routeur envoie toutes les images vers le même expert, perdant ainsi son aspect de spécialisation.
Pour contrer cela, nous avons implémenté une **perte auxiliaire** ($\mathcal{L}_{aux}$) ajoutée à la fonction de coût principale :

$$\mathcal{L}_{total} = \mathcal{L}_{CrossEntropy} + \lambda \cdot \mathcal{L}_{aux}$$

Cette perte pénalise une distribution inégale de l'utilisation des experts au sein d'un batch (basée sur l'Erreur Quadratique Moyenne - MSE - par rapport à une distribution uniforme).

4 Résultats expérimentaux

L'ensemble des expériences et résultats détaillés sont présentés dans le notebook **experiments.ipynb**. Nous résumons ici les protocoles choisis et conclusions essentielles.

4.1 Protocole

— **Données :** CIFAR-10 (45k train / 5k val / 10k test).

— ***Baseline* :** Un CNN dense de largeur ajustable (`width_multiplier`) pour correspondre soit au nombre de paramètres total du MoE, soit à son coût d'inférence (FLOPs).

— **MoE Configuration :** Différents modèles de MoE avec des hyperparamètres variables.

4.2 Comparaison de Performance

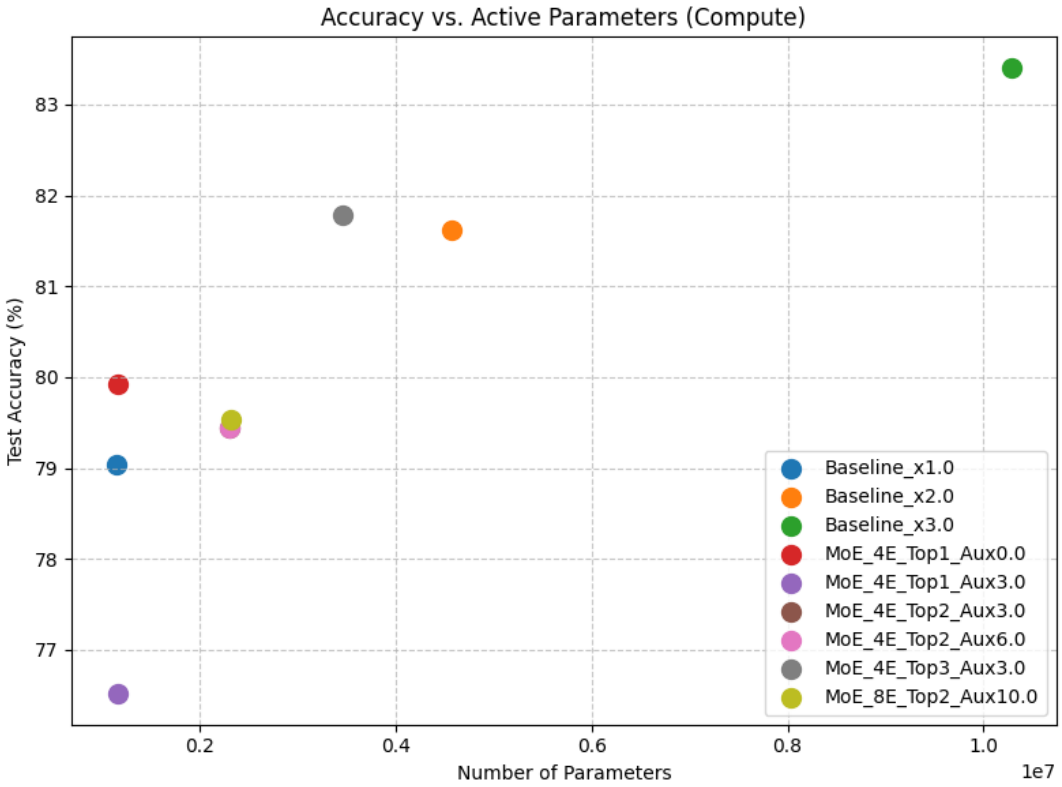

FIGURE 1 – Evolution de la précision en fonction du nombre de paramètres des modèles

— **Evolution de la *baseline* :**
Le modèle *baseline* semble suivre une courbe logarithmique : augmenter le nombre de paramètre rend le modèle plus robuste mais est de plus en plus gourmand.

— **Variations de l'accuracy des MoE :**
La performance d'un MoE ne dépend pas uniquement de sa taille, mais surtout de l'équilibre entre les experts(aux) et le routage. Sans contrainte (Aux 0.0), le MoE se limite à un seul expert, c'est ce qu'on veut éviter avec la perte auxiliaire.
L'introduction d'une perte auxiliaire permet de forcer l'usage de plus d'experts, mais cela ne devient réellement bénéfique que si l'on active plusieurs experts simultanément (Top-k à 2 ou 3).
Cela crée un effet d'ensemble efficace, permettant par exemple au modèle Top-3 de rivaliser avec un modèle dense deux fois plus gros tout en étant moins coûteux.
Enfin, augmenter aveuglément le nombre total d'experts n'est pas automatiquement bénéfique : si la tâche n'est pas assez complexe, diviser le modèle en trop de sous-parties devient inutile.

4.3 Analyse de la Spécialisation

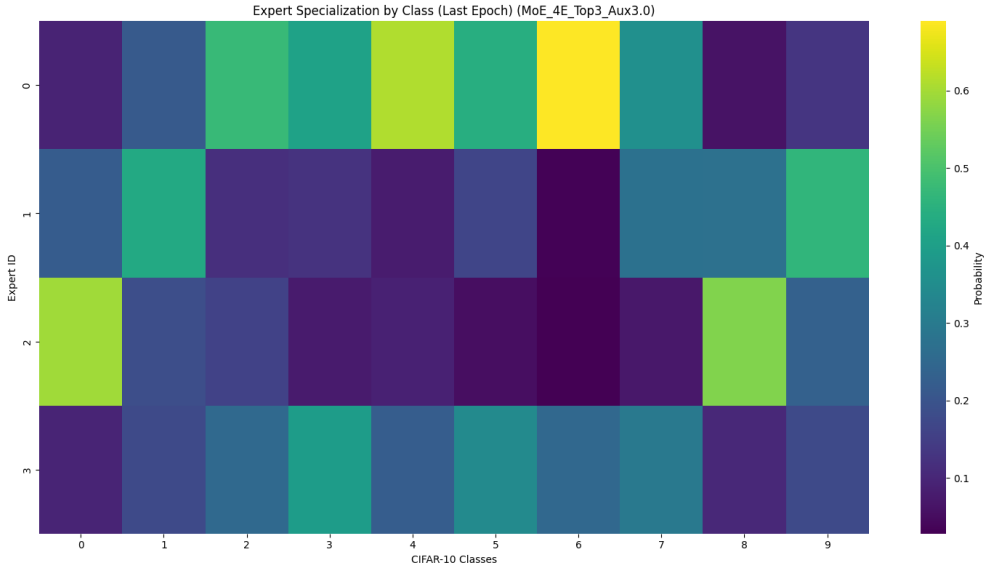

FIGURE 2 – Heatmap de routage : Fréquence d'activation de chaque expert pour chaque classe.

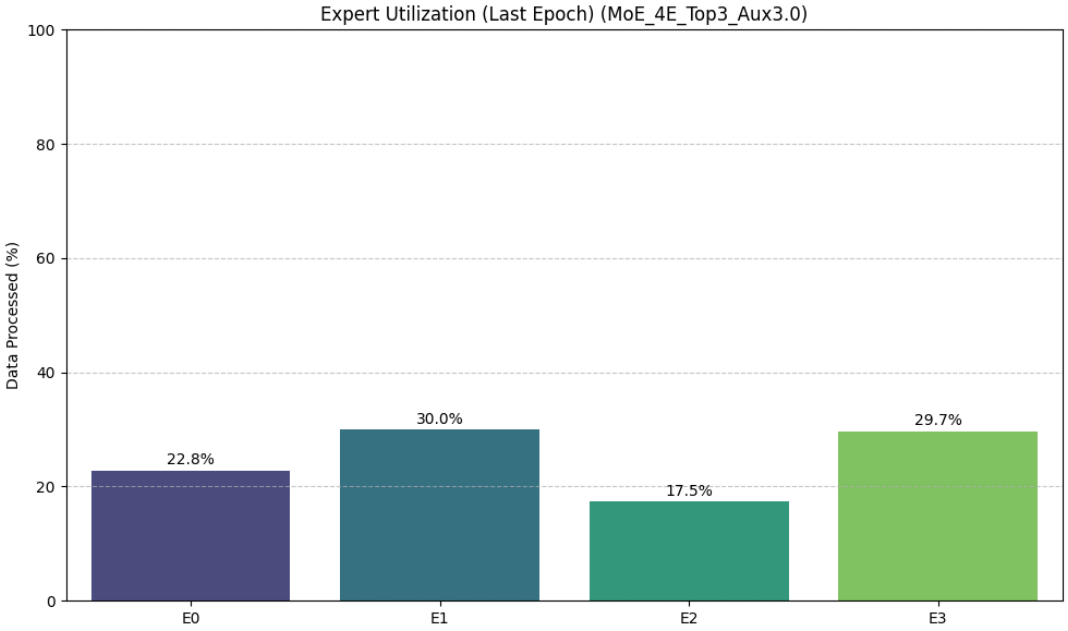

FIGURE 3 – Répartition de l'utilisation des experts sur l'ensemble des données.

Analyse :

L'histogramme d'utilisation (Fig. 3) montre une répartition relativement équilibrée de la charge, bien que non uniforme. Tous les experts sont sollicités, avec une charge variant de 17,5% (Expert 2) à 30,0% (Expert 1). On distingue deux groupes : les experts à forte charge (E1 et E3, $\approx 30\%$ chacun) et ceux à charge modérée (E0 et E2).

Cependant, c'est la matrice de spécialisation (Fig. 2) qui révèle la dynamique réelle du modèle. Contrairement à une simple polyvalence, on observe une **segmentation sémantique** claire entre les classes "Véhicules" et "Animaux" du dataset CIFAR-10.

— **Groupe "Animaux" (Experts 0 et 3) :**

- **L'Expert 0** est un "spécialiste pointu". Il présente des pics d'activation très intenses (couleur jaune) pour la classe 6 (Grenouille) et la classe 4 (Cerf), et une activité modérée pour les autres animaux.
- **L'Expert 3** agit comme un "généraliste de catégorie". Bien qu'il n'ait pas de pic jaune vif, il maintient une activation constante (bleu/vert) sur l'ensemble des classes animales (2 à 7), tout en étant très peu actif sur les véhicules.

— **Groupe "Véhicules" (Experts 1 et 2) :**

- **L'Expert 2**, bien que le moins utilisé globalement (17,5%), est hautement spécialisé. Il gère prioritairement la classe 0 (Avion) et la classe 8 (Bateau).
- **L'Expert 1** est le pendant de l'Expert 3 pour les véhicules. Il montre une activation diffuse mais présente sur les classes mécaniques (1, 8, 9), expliquant sa forte utilisation globale (30%) sans pour autant avoir de "pic" unique.

5 Discussion Critique

Bien que les résultats obtenus confirment le potentiel des architectures *Mixture of Experts* sur CIFAR-10, la mise en œuvre de ce genre de modèle nous a permis de mettre le doigt sur quelques limitations.

5.1 Le défi de l'effondrement des experts (Expert Collapse)

La difficulté majeure que nous avons rencontrée lors de l'entraînement est le phénomène d'effondrement des experts (*Expert Collapse*). Sans mécanisme de régulation, le réseau de routage tend naturellement vers une solution triviale où il attribue systématiquement la probabilité maximale aux mêmes experts (souvent un ou deux), laissant les autres inactifs.

- **Conséquence :** Le modèle redevient un réseau dense standard mais avec beaucoup de paramètres "morts", perdant tout l'intérêt de la capacité conditionnelle.
- **Contre-mesure :** L'introduction de la perte auxiliaire (ou d'autres stratégies similaires) est indispensable pour forcer une distribution uniforme. Cependant, nous avons observé que cette méthode introduit un conflit d'objectifs : si son poids (λ) est trop élevé, le routeur privilégie l'équilibre au détriment de la précision de classification (les experts reçoivent des données qui ne leur correspondent pas).

5.2 Sensibilité aux Hyperparamètres

Contrairement aux CNN classiques où augmenter la taille améliore souvent la performance de manière monotone, les MoE sont extrêmement sensibles au réglage des hyperparamètres structurels :

- **Le Top-k :** Un $k = 1$ (Hard Routing) est très rapide mais instable car le gradient ne se propage que vers un seul expert. Un $k = 2$ ou $k = 3$ permet un effet d'ensemble bénéfique et une meilleure propagation de l'erreur, comme observé dans nos résultats, mais augmente le coût de calcul.
- **Nombre d'experts :** Augmenter le nombre d'experts n'est pas linéairement corrélé à la performance. Sur un dataset simple comme CIFAR-10, nous avons constaté qu'un nombre excessif d'experts dilue l'information : chaque expert voit trop peu d'échantillons pour apprendre une spécialisation robuste (comme la distinction animaux/véhicules).

5.3 Coût d'Entraînement et Convergence

Une distinction importante doit être faite entre l'efficacité à l'inférence et le coût de l'apprentissage. Si le MoE est efficace lors de son utilisation (inférence *sparse*), son entraînement est paradoxalement plus lourd qu'un modèle dense équivalent :

— **Convergence lente :** Le modèle possédant un très grand nombre de paramètres totaux, et chaque expert ne voyant qu'une fraction des données (routage conditionnel), la convergence est plus lente. Il faut plus d'époques pour que chaque sous-réseau apprenne efficacement sa spécialisation.

— **Impact énergétique :** Cette lenteur à l'entraînement pose un problème écologique et économique. L'entraînement étant l'étape la plus énergivore du cycle de vie d'un modèle (bien plus coûteuse que l'inférence), le surcoût énergétique nécessaire pour entraîner l'immense quantité de paramètres du MoE peut réduire, voire annuler, les gains d'efficacité obtenus lors du déploiement, à moins que le modèle ne soit utilisé à très grande échelle par la suite.

5.4 Limites Matérielles et Perspectives

Bien que le MoE réduise les FLOPs théoriques (coût d'inférence), il n'allège pas la charge mémoire (VRAM). Tous les paramètres de tous les experts doivent être chargés en mémoire, même si seulement une fraction est utilisée par itération. Une piste d'amélioration pour des travaux futurs serait de remplacer nos experts convolutifs simples par des blocs Transformer complets (type *Switch Transformer*), où le gain de capacité des MoE est plus documenté, notamment pour la modélisation de séquences complexes.

6 Bibliographie

Références

[1] Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, Damai Dai. *Auxiliary-Loss-Free Load Balancing Strategy for Mixture-of-Experts*. arXiv preprint arXiv :2408.15664, 2024. https://arxiv.org/abs/2408.15664

[2] DataCamp. *What is Mixture of Experts (MoE) ?* Blog technique. https://www.datacamp.com/blog/mixture-of-experts-moe

[3] Chris Hughes. *How MoE Models Actually Learn : A Guide to Auxiliary Losses and Expert Balancing.* Medium, 2025. https://medium.com/@chris.p.hughes10/how-moe-models-actually-learn-a-guide-to-auxiliary-losses-and-expert-balancing-293084e3f

[4] Ressource Vidéo. *Mixture of Experts Explained.* YouTube. https://www.youtube.com/watch?v=_P2T4Oi0VxE