

Rapport de Projet : Classification de News via Mixture of Experts (MoE)

Votre Nom

16 janvier 2026

1 Introduction et Revue du Concept de MoE

L'objectif de ce projet est d'explorer l'architecture *Mixture of Experts* (MoE) pour la classification de textes. Les modèles MoE reposent sur le principe de "diviser pour régner" : au lieu d'utiliser un seul réseau dense, on utilise plusieurs sous-réseaux spécialisés (**Experts**) et un mécanisme de routage (**Gating Network**) qui décide quel expert doit traiter quel échantillon.

1.1 Modèles MoE

Les modèles MoE se composent de deux composants principaux :

- **Couches MoE éparses** : Elles remplacent les couches de réseau *feed-forward* (FFN) denses. Les couches MoE contiennent un certain nombre d'*experts* (par exemple 8), où chaque expert est un réseau de neurones. En pratique, les experts sont des FFN, mais ils peuvent également être des réseaux plus complexes, voire un MoE lui-même, conduisant à des **MoE hiérarchiques**.
- **Réseau de routage (Gate/Router)** : Il détermine quels tokens sont envoyés à quels experts. Par exemple, le token "More" peut être envoyé au deuxième expert, tandis que le token "Parameters" est envoyé au premier réseau. Comme nous l'explorerons plus tard, il est possible d'envoyer un token à plusieurs experts simultanément. La décision de **comment router un token vers un expert** est l'une des décisions majeures lors de la conception d'un MoE, le routeur est composé de paramètres appris et est pré-entraîné en même temps que le reste du réseau.

1.2 Load Balancing Loss

Les modèles MoE classiques souffrent de problèmes de *load balancing* : certains experts sont consultés fréquemment, tandis que d'autres sont rarement ou jamais sollicités. Pour encourager le routeur à sélectionner chaque expert avec une fréquence égale au sein de chaque batch, chaque couche MoE utilise des fonctions de perte auxiliaires.

Le **Switch Transformer** simplifie cela en une seule fonction de perte auxiliaire. Soit n le nombre d'experts, et pour un batch de requêtes $\{x_1, x_2, \dots, x_T\}$, la perte auxiliaire est définie par :

$$\mathcal{L}_{\text{aux}} = n \sum_{i=1}^n f_i \cdot P_i \quad (1)$$

où :

— $f_i = \frac{1}{T} \sum_{t=1}^T \mathbb{1}\{\text{argmax}(g(x_t)) = i\}$ est la fraction des tokens routés vers l'expert i

— $P_i = \frac{1}{T} \sum_{t=1}^T p_i(x_t)$ est la probabilité moyenne assignée à l'expert i par le routeur

Cette perte encourage une distribution uniforme de la charge entre les experts, minimisant ainsi le risque d'*expert collapse*.

1.3 Avantages des MoE

- **Spécialisation** : Chaque expert peut se spécialiser dans un aspect particulier des données.
- **Scalabilité** : Permet d'augmenter la capacité du modèle sans augmenter proportionnellement le coût computationnel.
- **Efficacité** : En n'activant qu'un sous-ensemble d'experts, on réduit le coût de calcul.

2 Description des Modèles Implémentés

2.1 NLP Model MoE

2.1.1 Architecture

Notre modèle, **MoEBertModel**, utilise **BERT** (*bert-base-uncased*) comme extracteur de caractéristiques (backbone). La sortie du *pooler* de BERT alimente :

- Un réseau de routage (couche linéaire) qui génère des poids pour chaque expert.
- Une liste de N experts (**TextExpert**), chacun étant un réseau de neurones *feed-forward* avec normalisation et dropout.

2.1.2 Choix de Conception

- **Soft Routing** : Nous utilisons un routage "doux" via une fonction Softmax pour permettre un apprentissage stable de tous les experts simultanément.
- **Exploration** : Ajout d'un bruit gaussien sur les scores du router durant l'entraînement pour éviter l'effondrement précoce sur un seul expert.
- **GELU** (Gaussian Error Linear Unit) : en raison de ses performances supérieures dans les tâches NLP par rapport à ReLU.

Nous avons effectué des expériences avec les deux types de routage : **Soft** et **Hard**. Le routage dur sélectionne les k meilleurs experts, réduisant ainsi le coût computationnel à l'inférence, mais peut être plus difficile à entraîner.

3 Protocole Expérimental et Hyperparamètres

Le meilleur entraînement avec BERT unfreeze a été effectué sur le dataset AG News. Le

Hyperparamètre	Valeur
Batch Size	64
Nombre d'Époques	3
Learning Rate	2×10^{-5}
Nombre d'Experts	8
Architecture Backbone	BERT Base
Max Length Texte	128
Routing	Soft
Top K Experts	6
Freeze Bert	False
Load Balancing Coef	0.01
Load Balancing	True

TABLE 1 – Configuration des hyperparamètres

meilleur entraînement avec BERT freeze a été effectué sur le dataset AG News.

Hyperparamètre	Valeur
Batch Size	64
Nombre d'Époques	5
Learning Rate	3×10^{-5}
Nombre d'Experts	8
Architecture Backbone	BERT Base
Max Length Texte	128
Routing	Soft
Top K Experts	6
Freeze Bert	True
Load Balancing Coef	0.5
Load Balancing	True

TABLE 2 – Configuration des hyperparamètres

4 Résultats Expérimentaux et Interprétation

4.1 NLP

Pour le modèle NLP, nous avons entraîné plusieurs configurations MoE avec différents types de routage et comparé leurs performances à un modèle BERT standard. Comme modèle de base, nous avons utilisé BERT base uncased, et nous avons gelé les poids de BERT pendant l'entraînement des modèles MoE pour réduire le coût computationnel. Lorsque nous entraînons le modèle MoE avec BERT freeze, nous obtenons une accuracy proche de 88%, ce qui est comparable au modèle BERT standard. Nous avons également observé l'impact du load balancing, qui a permis d'améliorer la répartition de l'utilisation des experts et d'éviter l'effondrement de certains experts. Ce qui est visible par exemple dans la figure ci-dessous

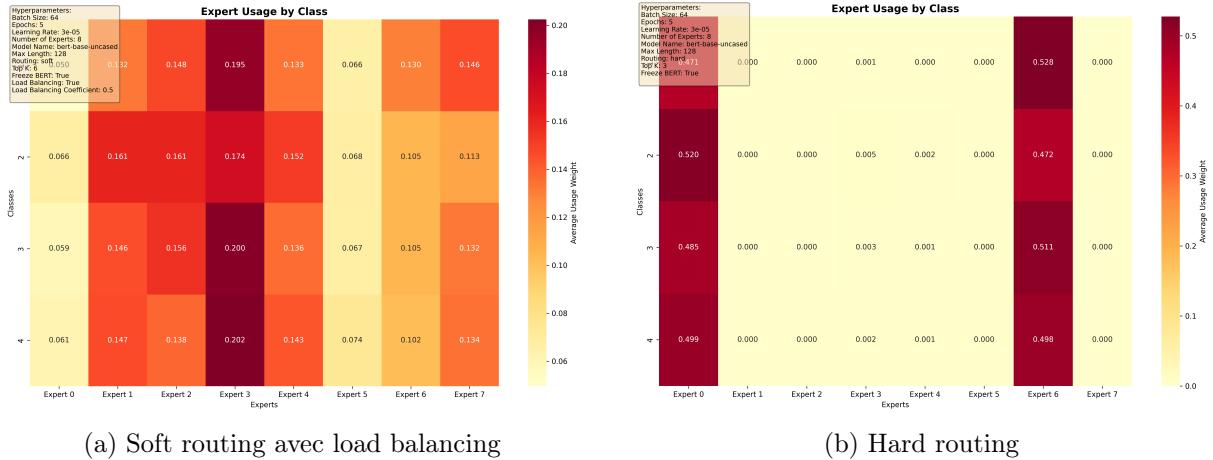


FIGURE 1 – Comparaison de l'utilisation des experts selon le type de routage.

Nous remarquons que l'utilisation des experts est assez bien repartie lorsqu'on utilise un routage soft et un load-balancing de 0.5. Cependant lorsqu'on passe à un routage hard, on remarque que l'utilisation des experts devient plus inégale, ce qui peut indiquer un effondrement de certains experts. Toutes les figures sont disponibles dans le dossier metrics.

5 Discussion Critique

5.1 Forces

Capacité de spécialisation et maintien d'une haute précision sur des classes complexes grâce à la modularité des experts.

5.2 Limites

Risque de généralisation des experts sur des sous-ensembles spécifiques du dataset, potentiellement limitant la robustesse globale du modèle. Risque d'**Expert Collapse** (effondrement) où seuls quelques experts sont activés, rendant le reste de la capacité du modèle inutile si le routage n'est pas bien régularisé.

5.3 Perspectives

5.3.1 NLP

Utiliser des fonctions de perte de *Load Balancing* plus agressives ou explorer le **Top-k routing** pour réduire le coût computationnel à l'inférence. Mixer des experts de différentes architectures (par exemple, CNN, RNN) pour capturer diverses caractéristiques des données textuelles. Cela pourrait améliorer la spécialisation des experts et la performance globale du modèle. Nous pourrions aussi ajouter du noise au gating ou encore ajouter une couche d'adaptation avant le gating.