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) et son application à différentes tâches dans ce projet nous nous sommes intréssés à la classification de texte et d'images.
Le concept de MoE repose sur le constat qu'un modèle unique peut avoir du mal à exceller dans l'ensemble des tâches ou les domaines qui lui sont confiés sans devenir excessivement complexe et coûteux en termes de calcul. L'idée ppour surmonter la dificulté de modélisation de tâches complexes avec un seul modèle dense est de diviser le modèle en un ensemple de sous-modèle plus petits et spécialisés appelés **experts** et d'un mécanisme de routage appelé **gating network** qui décide quel expert doit traiter chaque entrée. Ainsi au on cherche à achever des performances similaires voir supérieures à celles d'un modèle dense tout en réduisant le coût calcul global.

1.1 Architecture Générale d'un MoE

L'architecture générale d'un modèle MoE peut être représentée comme suit :

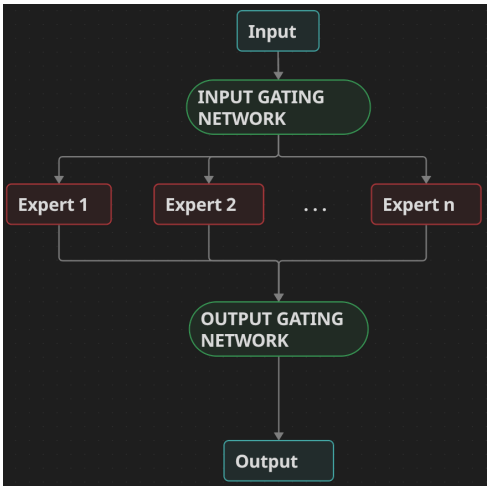


FIGURE 1 – Architecture MoE

— **Le routeur (Gating Network)** : C'est un modèle qui décide quel expert doit traiter chaque entrée. Le routeur prend l'entrée et produit un ensemble de poids ou de probabilités qui indiquent l'importance relative de chaque expert pour cette entrée. Le routeur peut être un réseau de neurones simple ou un modèle plus complexe. Le routeur peut utiliser différentes stratégies de routage, comme le routage dur (hard routing) où un seul expert est sélectionné, ou le routage doux (soft routing) où plusieurs experts sont combinés. Il

existe aussi des variantes plus hybrides avec un une selection d'experts en focntion d'un seuil.

— **Les experts (Experts)** : Ce sont des sous-modèles spécialisés qui se concentrent sur des aspects spécifiques de la tâche principale. Chaque expert peut être un réseau de neurones, un arbre de décision, ou tout autre type de modèle. Certains sont concus spécialement pour une tâche particulière à l'avance d'autres se spécialisent durant l'entrainement. Dans certains modèles MoE les experts font tous partie d'un même réseau de neurones ou seulement une partie s'active pour chaque experts. ce qui permet de réduire le cout mémoire du modèle.

— **Combinaison des sorties** : Une fois que les experts ont traité l'entrée, leurs sorties sont combinées pour produire la sortie finale du modèle MoE. La combinaison peut être une simple moyenne pondérée, une somme, ou une opération plus complexe en fonction des poids fournis par le routeur.

1.2 Architecture Générale d'un MoE

L'architecture générale d'un modèle MoE peut être représentée comme suit :

— **Les couches Mixture of Experts (MoE Layers)** :

Une couche MoE remplace une couche dense classique par un ensemble d'expertes en parallèle et Chaque expert est un sous modèle spécialisé, la forme et lacomplexité des experts peuvent varier en fonction de la tâche principale et de la conception du MoE mais leur rôle est le même : traiter une partie spécifique du problème. Dans le cas ou tout les experts sont de même type, on parle de MoE homogène, sinon on parle de MoE hétérogène. Les MoEhomogènesp Les couches MoE 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 (Gating Network)** : 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.

1.3 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 $\{x_1, x_2, \ldots, x_T\}$, la perte auxiliaire est définie par :

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

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*.

1.4 Avantages des MoE

— **Efficacité** : Seuls les experts compétents pour une partie spécifique du problème sont utilisés, ce qui permet d'économiser du temps et de la puissance de calcul.
— **Flexibilité** : Il est facile d'ajouter de nouveaux experts ou de modifier leurs spécialités, rendant le système adaptable à différents problèmes.
— **Meilleurs résultats** : Étant donné que chaque expert se concentre sur ce qu'il maîtrise le mieux, la solution globale est généralement plus précise et fiable.
— **Scalabilité** : Permet d'augmenter la capacité du modèle sans augmenter proportionnellement le coût computationnel.
— **Spécialisation** : Chaque expert peut se spécialiser dans un aspect particulier des données.

2 Description des Modèles Implémentés

2.1 Bert 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.

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

Le 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

3.1 Transformer MoE

3.2 Transformer MoE Layer

Nous avons également implémenté un modèle Transformer avec des couches MoE personnalisées. Le modèle, `MoETransformerModelV2`, utilise des couches d'attention multi-têtes suivies de couches MoE éparses. Chaque couche MoE contient plusieurs experts, chacun étant un réseau feed-forward.

3.2.1 Architecture MoE Transformer V2

Notre modèle `MoETransformerModelV2` suit l'architecture Switch Transformer en intégrant des couches Mixture-of-Experts (MoE) **à l'intérieur** de chaque bloc Transformer.

Embeddings. Les tokens d'entrée sont convertis en vecteurs via une couche d'embedding de dimension d_{model}, additionnés avec des embeddings positionnels apprenables.

Couches MoE Transformer. Le modèle empile N couches identiques, chacune composée de :

1. **Multi-Head Self-Attention** avec connexion résiduelle et normalisation de couche.

2. **MoE Feed-Forward** qui remplace le FFN dense standard :
   — Un *réseau de gating* $g(x) = \text{softmax}(W_g x)$ calcule les poids de routage pour E experts.
   — Seuls les *Top-K* experts sont activés pour chaque token.
   — La sortie est la somme pondérée : $y = \sum_{i \in \text{Top-K}} g_i(x) \cdot E_i(x)$

Classification. Après les N couches, une normalisation finale est appliquée, suivie d'un mean-pooling sur les tokens non-masqués et d'un classifieur linéaire.

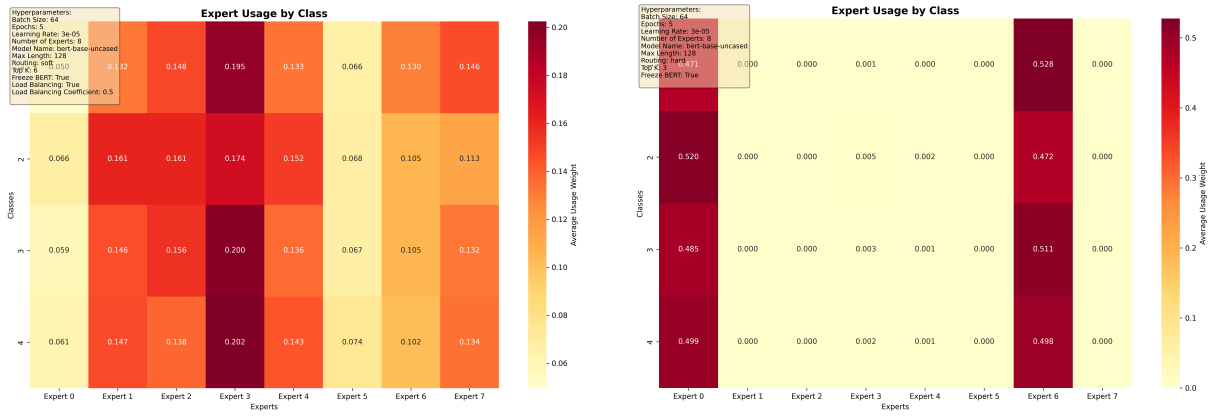
Paramètre	Valeur
Dimension embedding (d_{model})	256
Dimension FFN (d_{ff})	512
Nombre de têtes d'attention	4
Nombre de couches (N)	2
Nombre d'experts (E)	8
Top-K	4
Dropout	0.3

TABLE 3 – Hyperparamètres du MoE Transformer V2.

4 Résultats Expérimentaux et Interprétation

4.1 NLP

Pour le modèle NLP, nous avons entrainé plusieurs configurations MoE avec difficiles types de routage et comparé leurs performances à un modèle BERT standard. Comme modele de base, nous avons utilise BERT base uncased, et nous avons freeze les poids de BERT pendant l'entrainement des modèles MoE pour réduire le cout computationnel. Lorsque nous entrainons le modele MoE avec BERT freeze, nous obtenons une accuracy proche de 88%, ce qui est comparable au modele BERT standard. Nous avons egalement observe l'impact dur load balancing, qui a permis d'ameliorer la repartition de l'utilisation des experts et d'eviter l'effondrement de certains experts. Ce qui est visible par exemple dans la figure ci dessous



(a) Soft routing avec load balancing

(b) Hard routing

FIGURE 2 – Comparaison de l'utilisation des experts selon le type de routage.

Nous remarquons que l'usage des experts est assez bien repartis 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'usage des experts devient plus inégal, ce qui peut indiquer un effondrement de certains experts.

Cependant le routage Hard, peut permettre aux experts de se specialiser davantage, ce qui peut conduire à de meilleures performances dans certains cas. En comparant le routage hard et soft, on remarque que les experts se specialisent plus avec le routage hard, mais au prix d'une utilisation inégale des 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 genéralisation 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 router 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 ameliorer la specialisation des experts et la performance globale du modele. Nous pourrons aussi ajouter du noise au gating ou encore ajouter une couche d'adaptation avant le gating.