Rapport de Projet : Exploration des Modèles Mixture of Experts (MoE) pour la Classification de Texte et d'Images

Iliescu Eduard Nathan Fitger

17 janvier 2026

https://github.com/0xNatgan/Arna/

Table des matières

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 :

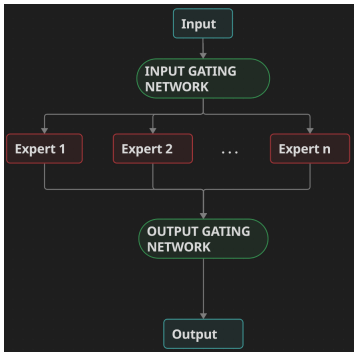

FIGURE 1 – Architecture MoE

- **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 fonction 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 Load Balancing Loss

L'un des problèmes classiques des architectures MoE est le **load balancing** entre les experts. 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*.

1.3 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 NLP 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.

2.2 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

2.2.1 Transformer MoE

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.

2.2.2 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.

2.3 Résultats Expérimentaux et Interprétation

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.

Nous avons construit 3 modeles differents :

— Un modele avec Bert comme le Backbone
— Un modele Transformer classique avec une couche MoE
— Un modele Transformer

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.

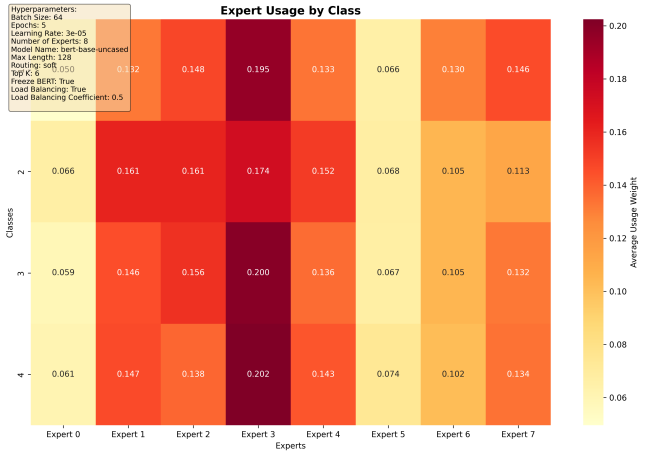

FIGURE 2 – Soft routing avec load balancing disponible dans metrics/model 2.

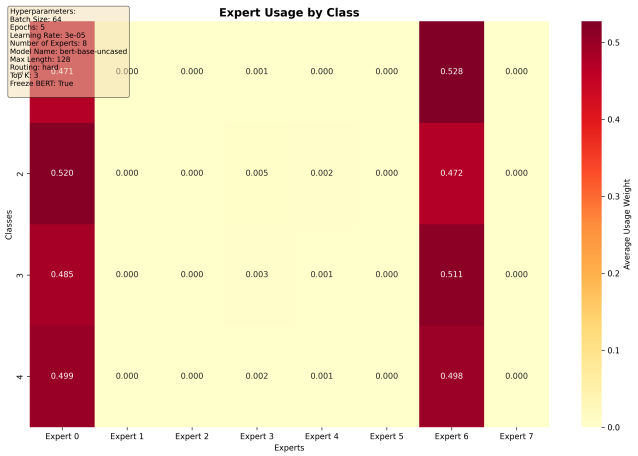

FIGURE 3 – Hard routing disponible dans metrics/model 1.

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.

Lors de la comparaison de l'*accuracy* entre notre modèle MoE avec BERT en *backbone* et un modèle BERT classique simulant des experts, nous observons que le MoE est légèrement supérieur. Cette différence peut s'expliquer par le **nombre plus élevé de paramètres** dans le modèle MoE, comme le montre le graphique *Accuracy vs Model Size*.

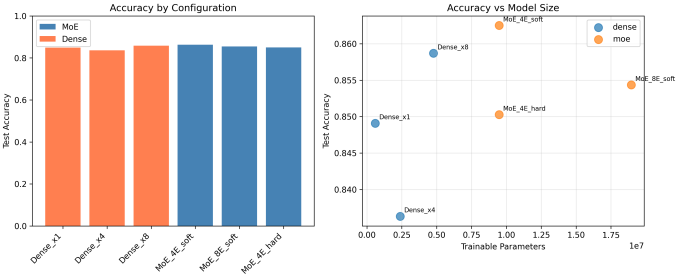

FIGURE 4 – Comparaison.

Le modèle `Dense_x1` atteint une *accuracy* de **85,4%** avec seulement **3 millions de paramètres**, ce qui en fait une solution très efficace. En revanche, notre **meilleur modèle MoE**, avec **18 millions de paramètres**, n'améliore cette performance que de **1,5%**, tout en étant plus coûteux en termes de ressources.

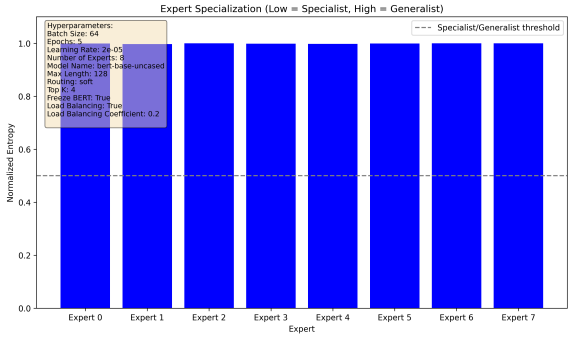

FIGURE 5 – Model Bert Specialization disponible dans metrics/model bert.

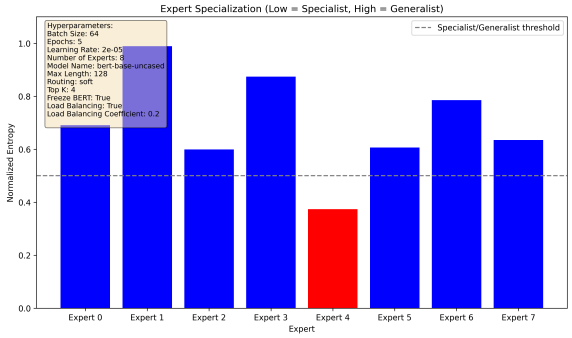

FIGURE 6 – Model Transformer with MoE Layer Specialization.

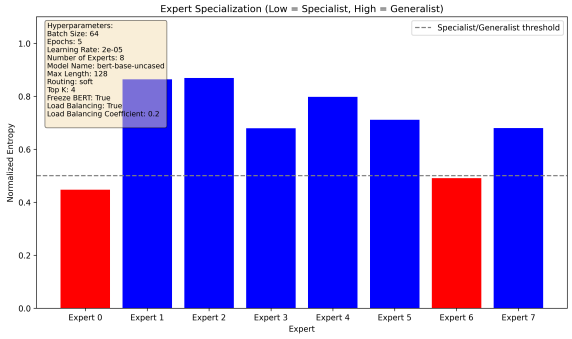

FIGURE 7 – Model Transformer Specialization.

En comparant les métriques des modèles `model_bert`, `model_transformer` et `model_transformerv2`, nous constatons que `model_transformerv2` qui utilise une couche d'experts sur un *transformer* simple, ainsi que le modèle utilisant un *transformer* en *backbone*, permettent aux experts de **se spécialiser davantage**. À l'inverse, le modèle utilisant **BERT en *backbone*** tend à **généraliser les experts**.

Cette différence peut s'expliquer par la nature des représentations apprises :

— Dans un *transformer simple*, les représentations initiales sont moins riches, ce qui favorise la spécialisation des experts.

— Avec **BERT**, les représentations sont déjà **riches, sémantiques et discriminantes**, limitant ainsi la marge de spécialisation des experts.

Nous avons effectué une analyse de 6 modeles différents avec diverses configurations de routage (hard, soft, top-k), nombre d'experts ainsi que dropout. Ces tests ont été réalisés sur le dataset AG News avec un Transformer avec une couche MoE.

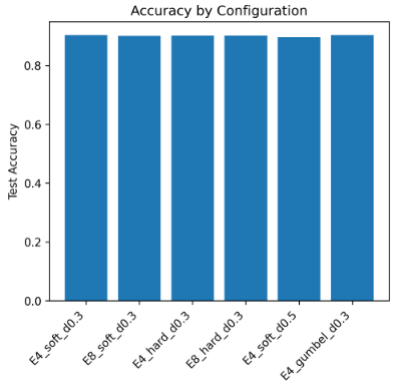

FIGURE 8 – Comparaison accuracy.

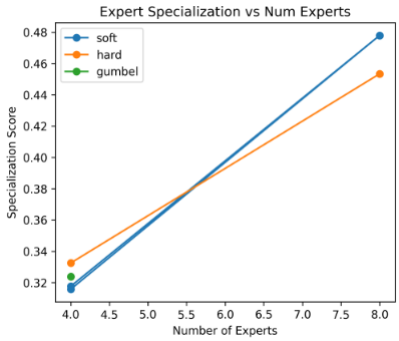

FIGURE 9 – Comparaison expert number.

Nous pouvons conclure que le routing ne change pas significativement l'accuracy du modèle ni la spécialisation des experts. Cela est explique par le fait d'utiliser un Transformer simple avec une couche MoE, de meme pour un transformer simple. Lorsqu'on utilise un MOE avec Bert en backbone, le routing a un impact plus significatif sur la spécialisation des experts.

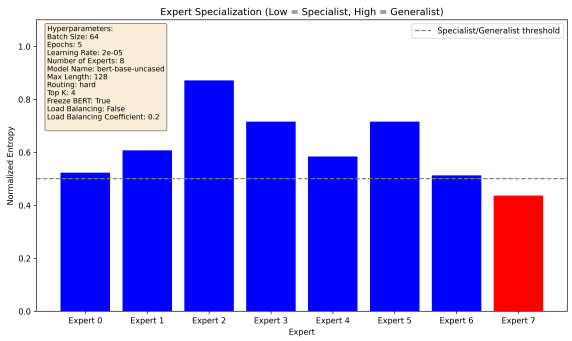

FIGURE 10 – No load balance, bert backbone.

Lors d'une experience avec bert backbone et sans load balancing, on remarque que le routage hard permet une meilleure spécialisation des experts.

D'autres expériences avec des configurations variées sont disponibles dans le dossier **NLP_MoE/metrics**, la configuration du modèle est indiquée sur chaque images

2.4 Discussion Critique

2.4.1 Forces

Capacité de spécialisation et maintien d'une haute précision sur des classes complexes grâce à la modularité des experts. Flexibilité du routage permettant d'adapter le modèle à différentes tâches et distributions de données, par exemple soft, hard ou top-k. La modularité architecturale permettant d'ajouter/retirer des experts et d'adapter à différents backbones (Transformer, Bert). Il est aussi possible d'etendre le modèle à d'autres tâches comme la détection de sentiment.

2.4.2 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 router n'est pas bien régularisé.

Les principales limites de nos expériences résident dans le **nombre d'*epochs*** utilisé pour l'entraînement. En raison du **coût computationnel élevé**, certaines expériences ont été limitées à **2 ou 3 *epochs***.

Cependant, tous les modèles entraînés de manière isolée ont bénéficié d'un entraînement complet sur **5 *epochs***, malgré le coût élevé. Nous avons jugé cette approche plus pertinente pour obtenir des résultats significatifs.

2.4.3 Perspectives

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.

Amélioration du mécanisme de routage.

- **Expert Choice Routing** : Inverser la logique du routage en permettant aux experts de choisir les tokens plutôt que l'inverse, comme proposé dans *Expert Choice*. Cette approche garantit un équilibrage naturel de la charge.
- **Routage hiérarchique** : Implémenter un routage à deux niveaux où un premier routeur sélectionne un groupe d'experts, puis un second affine la sélection au sein du groupe.
- **Routage appris par token** : Au lieu de router au niveau de la phrase (via [CLS]), router chaque token individuellement pour capturer des spécialisations plus fines.

Intégration plus profonde avec le backbone.

- **MoE dans les couches intermédiaires de BERT** : Remplacer les FFN de certaines couches BERT par des couches MoE (comme dans Switch Transformer) plutôt que d'ajouter le MoE uniquement en sortie.
- **Adapter Layers** : Ajouter des *adapter layers* entraînables entre BERT et le MoE pour mieux adapter les représentations.
- **Dégel progressif** : Dégeler progressivement les dernières couches de BERT durant l'entraînement pour permettre une co-adaptation avec les experts.

Généralisation à d'autres tâches NLP.

- **Multi-task Learning** : Entraîner le MoE sur plusieurs tâches simultanément (classification, NER, sentiment) avec des experts partagés et spécialisés.
- **Few-shot Learning** : Évaluer la capacité du MoE à généraliser sur de nouvelles classes avec peu d'exemples.
- **Domain Adaptation** : Tester la transférabilité des experts sur des domaines non vus durant l'entraînement.

3 CNN MoE pour la Classification d'Images du dataset Mnist

L'implémentation de ce projet est disponible dans le dossier **Vision_Moe/Mnist_MoE**.

3.1 Architecture

Notre Gating Network est un CNN simple ou les sorties de la derniere couche convolutive sont aplaties et passées à une couche linéaire pour produire les scores des experts. Chaque expert est un CNN un peu plus profond que le routeur, avec plusieurs couches convolutives suivies de couches fully connected. en fonction du type de routage choisi (soft, hard, hybrid), les experts sont sélectionnés et leurs sorties combinées pour produire la sortie finale du modèle.

3.2 Choix de Conception

- **Entrainement** : L'entrainement des experts et du routeur se fait simultanément pour permettre au routeur d'apprendre à sélectionner les experts les plus pertinents en fonction des données d'entrée.
- **Parametres d'entrainement** : Durant l'entrainement le routage est effectué en mode soft pour permettre une exploration complète et un entraînement complet des experts.
- **Load balancing** : Nous avons implémenté une perte de load balancing pour encourager une utilisation uniforme des experts et éviter un déséquilibrage dans le routage des experts.
- **Routage Hybride** : Le routage hybride consiste à selectionner les experts dont les scores depassent un certain seuil, permettant ainsi une flexibilité entre les deux extremes il permet d'avoir un score légerement plus élévé lors des tests mais peut utiliser plus d'experts que le routage hard.

3.3 Protocole Expérimental et Hyperparamètres

3.3.1 Première Version : Experts Préentrainés

Dans un premier temps, nous n'avions pas compris exactement le concept attendu et crée un modèle MoE avec des experts préentrainés chacun avec des altérations différentes du dataset MNIST (rotation, bruit, etc.). Cette approche bien que relativement performante et créative ne correspondait pas à l'objectif principal de l'architecture MoE. Le routeur lui était bien entrainé à appairer les différents experts en fonction des altérations mais celà n'avait aucun interet et revenait à faire de la sélection de modèle que l'on aurait pu faire à la main plutôt que de l'apprentissage collaboratif entre experts. Ce projet est trouvable dans le dossier **Vision_Moe/Mnist_MoE/Pretrained** et dans le notebook **Vision_MoE/Mnist_pretrained.ipynb**.

3.3.2 Version Finale : MoE CNN Classique

Pour la version fidèle à l'architecture MoE classique nous avons entrepris de créer un modèle CNN MoE où le routeur et les experts sont entrainés simultanément sur le dataset MNIST. Dans un premier temps nous voulions comparer différentesarchitectures d'expertsCNN simples avec différentes profondeurs et nombres de filtres. Cependant après de nombreux essais le routing convergeait toujours vers le même expert. Nous avons donc décidé d'utiliser des experts identiques pour garantir une spécialisation équitable entre les experts.

Notre implémentation finale du modèle MoE CNN est disponible dans le dossier **Vision_Moe/Mnist_MoE/Final_MoE** et dans le notebook **Mnist_MoE.ipynb**.

Nous avons entrainé notre modèle sur le dataset MNIST avec les hyperparamètres suivants :

Hyperparamètre	Valeur
Batch Size	256
Nombre d'Époques	10
Nombre d'Experts	7
Routing	soft(training)
Load Balancing Coef	0.05
τ (softmax + gumbell)	1
gumbell-softmax	True

TABLE 4 – Configuration des hyperparamètres pour le modèle CNN MoE sur MNIST

Les paramètres de test durantl'inférence sont les suivants :

Hyperparamètre	Valeur
Batch Size	256
Routing	hybrid
softmax	True

3.4 Résultats Expérimentaux et Interprétation

Durant nos expérimentations nous avons eu plusieurs résultats non concluants mais qui nous on mieux permis de comprendre certaines parties de l'architecture MoE.

Parmis nos expériences et observations les plus importantes nous retenons les suivantes :

- **Architecture des experts** : L'utilisation d'experts avec une architecture identique nous à permis d'éviter l'effondrement de certains experts et de garantir une spécialisation équitable. Des experts trop différentspeuvent conduire à une domination de certains experts.

— **Impact du Load Balancing** : L'ajout de la perte de load balancing a significativement amélioré la répartition de l'utilisation des experts, évitant ainsi l'effoondrement dès le début de la répartition des experts.
— **Routage Hybride** : Le routage hybride a permis un compromis intéressant entre le routage hard et soft, offrant une flexibilité accrue lors de l'inférence dont la difficulté réside dans la définition du threshold.
— **Gating Network** : Le choix d'architecture du gating network est important pour bien capturer les caractéristiques des différents experts. Dans un premier temps nous avions utilisé un réseau linéaire qui ne permettait pas de router correctement les entrées vers les experts puis nous nous sommes tournés vers un réseau avec deux couches de convolution.
— **Performance** : Notre modèle MoE CNN a atteint une accuracy de environ 99.2% sur le dataset MNIST, ce qui est compétitif par rapport aux modèles CNN classiques tout en offrant la flexibilité et la modularité des architectures MoE.
— **Dimensions des batchs d'entrainement** : Nous avons remarqué que des batchs plus grands permettaient une meilleure estimation des statistiques de load balancing, conduisant à une répartition plus uniforme de l'utilisation des experts. Des batchs trop petits (64 mènents à un routing inégal et/ou peu stable entre les epochs.)

L'impact de certains hyperparamètres sur la performance et la spécialisation des experts est assez notable :

— **Load Balancing Coefficient** : Un coefficient trop faible conduit à un effondrement des experts, tandis qu'un coefficient trop élevé peut nuire à la performance globale en forçant une répartition trop uniforme et des performances plus faibles.
— **Type de Routage** : Le routage soft favorise une exploration complète des experts, tandis que le routage hard peut conduire à une spécialisation plus marquée mais au risque d'effondrement. Pour cette raison nous avons opté pour un routage hybride durant l'inférence.
— **Nombre d'Experts** : Un nombre trop élevé d'experts peut compliquer l'apprentissage du routeur et mettre de coté quelques experts, tandis qu'un nombre trop faible limite la capacité de spécialisation et augmente l'effondrement de la répartition.
— **Nombre d'epochs** :Un trop grand nombre d'epochs (au delà de 8) peut conduire à un surapprentissage de certains experts et à un éffondrement de la répartition. A l'inverse un nombre trop faible(en dessous de 3-4) donne des performances plus faibles.

Resultats de notreMoE :

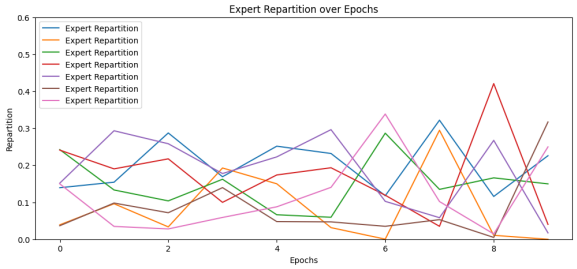

FIGURE 11 – ACtivation des experts au court de l'entrainement

La répartition des expert la plus intéressante est situé dans l'époque 4ensuite il semble y avoir un éfondrement de la répartition des experts(epoques 6-8). A l'époque 4 la répartition desexperts est entre9 et 20%soit une répartition pour 7 experts assez satisfaisante

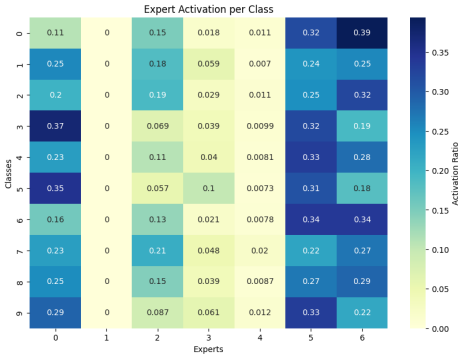

FIGURE 12 – Activation des experts en fonction des classes du dataset

La répartition de l'activation des modèles en fonction des classes dans le cas ou le MoE est entrainé sur trop d'époques montreun éfondrement de la répartition

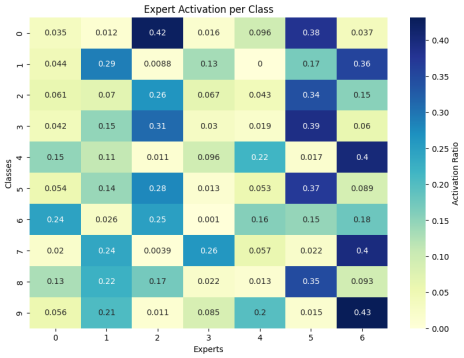

FIGURE 13 – Activation des experts en fonction des classes du dataset (entrainé sur 4 époques)

Répartition des l'activation desexpertsen fonction des classes. La répartition est homogène avec aucun expert laissé en dehors.

3.5 forces, limites, perspectives

L'implémentation de ceMoEpour le dataset Mnistn'est pas forcément la plusaboutie. Il manqueplus de robustessepour que la répartition des expertsrestestable tout au long des époques. Il serait intéressant d'avoir plus d'informations et de données en comparaison avec un modèle dense ainsi que une évolution des performances en fonction de chaquehyperparamètres. Pour desraisons de puissance et detemps nous n'avons pas pu fournir toutes ces comparaisons. Toutefoisle modèle estcompletet modulaire et des amélioration comme celles citées ne seraient pas très compliqué à implémenter.

Références

[1] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers : Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961, 2021.

[2] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

[3] Ka Man Lo, Zeyu Huang, Zihan Qiu, Zili Wang, and Jie Fu. A closer look into mixture-of-experts in large language models, 2025.

[4] Siyuan Mu and Sen Lin. A comprehensive survey of mixture-of-experts : Algorithms, theory, and applications, 2025.

[5] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks : The sparsely-gated mixture-of-experts layer. *CoRR*, abs/1701.06538, 2017.

[6] Alexandre TL. Qu'est-ce que le mixture of experts (moe) ?, 2024. YouTube video.