Combiner convolution et transformateurs pour la vision par ordinateur : étude du modèle CoAtNet

Matthieu Basset, Sébastien Goll, Floriane Ronzon

Abstract

Les transformateurs ont suscité un intérêt croissant dans le domaine de la vision par ordinateur, mais ils restent encore en deçà des réseaux convolutionnels. Le modèle Vision Transformer (ViT) a permis de montrer que les transformateurs ont d'excellents résultats lorsqu'ils sont pré-entrainés sur de grosses bases de données, mais ils sont dépassés par les modèles convolutionnels dans le cas de plus petites bases de données. C'est pourquoi l'équipe « Brain Team » de Google a mis en place un modèle combinant CNN et transformateurs : CoAtNet. Dans ce papier, nous étudions les performances du CoAtNet et les comparer à un modèle ViT. Nous montrerons que les performances du CoAtNet est nettement supérieur a celle des ViT pour un même nombre de paramètres.

1. Introduction

Les architectures basées sur l'auto-attention, en particulier les transformateurs, sont devenues le modèle de choix dans le traitement du langage naturel. L'approche dominante consiste à effectuer un pré-entraînement sur un grand corpus de textes, puis à effectuer un réglage fin sur un plus petit ensemble de données spécifiques à une tâche. Grâce à l'efficacité et à l'évolutivité des Transformers, il est devenu possible de former des modèles d'une taille sans précédent, avec plus de 100 milliards de paramètres.

Avec ce succès, de nombreux travaux antérieurs ont tenté d'intégrer la puissance de l'attention dans la vision par ordinateur. En 2021, le modèle (ViT) a montré qu'avec presque uniquement des couches de Transformer vanille, on pouvait obtenir des performances raisonnables sur ImageNet-1K uniquement. Lorsqu'il est pré-entraîné sur l'ensemble de données JFT-300M à grande échelle faiblement étiqueté, ViT obtient des résultats comparables à ceux des ConvNets comme ResNet, ce qui indique que les modèles Transformer ont potentiellement une plus grande capacité à l'échelle que les ConvNets.

L'équipe de Google a ainsi montré que les couches convolutionnelles tendent à mieux généraliser avec une vitesse de convergence plus rapide grâce à leur forte priorité au biais inductif, tandis que les couches d'attention ont une plus grande capacité de modèle qui peut bénéficier de plus grands ensembles de données. Par conséquent, la combinaison des couches de convolution et d'attention permet d'obtenir une meilleure généralisation et une meilleure capacité.

Coatnet parvient ainsi à atteindre des résultats comparables à l'état de l'art pour différentes tailles de données. Lorsque CoAtNet est entrainé sur ImageNet-1K atteint une précision de 86,0 %, ce qui correspond à l'art antérieur NFNet dans des conditions de ressources de calcul et d'entraînement similaires. Lorsqu'il est pré-entrainé sur ImageNet-21K avec environ 10 millions d'images, CoAtNet atteint une précision de 88,56 % lorsqu'il est ajusté sur ImageNet-1K, ce qui correspond à ViT-Huge pré-entrainé sur JFT-300M, un ensemble de données 23 fois plus grand. Finalement, lorsque JFT-3B est utilisé pour le pré-entraînement, CoAtNet fait preuve d'une meilleure efficacité par rapport à ViT, et pousse la précision top-1 d'ImageNet-1K à 90,88 % tout en utilisant 1,5 fois moins de calculs que ViT-G/14[1]

2. 2Modèles du ViT et du CoAtNet

Avant de présenter nos résultats, nous souhaitons faire un rappel sur le principe de fonctionnement de chacun des modèles.

2.1 ViT

Le ViT suit de très près l'architecture d'un transformateur classique. La figure 1 montre un aperçu du modèle.

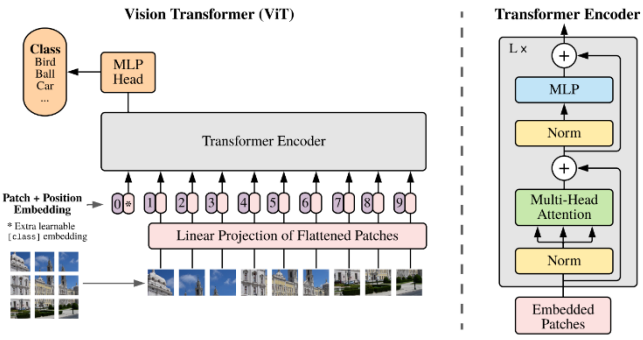

Figure 1: Architecture des ViT

[1] Données issues de l'article *CoAtNet: Marrying Convolution and Attention for All Data Sizes*

Plongements

Nous devons transformer l'image dans un format utilisable par le transformateur : une séquence de vecteurs à D dimensions. Pour cela, on commence par séparer l'image d'origine (HxW) en petit patchs de taille PxP. Ces patchs sont ensuite aplatis puis passés dans une couche de plongement. Puis on ajoute un jeton de classe pour permettre la prédiction de la classe. On obtient ainsi le résultat suivant pour l'input :

$$z0 = [xclass;\ x1pE;\ x2pE; \cdots;\ xNpE]$$

Jeton de classe

Pour prédire la classe de l'image, nous utilisons une technique inspirée de BERT, un autre modèle de langage. Nous donnons en entrée un jeton au début de l'image. A la sortie du ViT, ce jeton passe par un MLP simple pour donner la prédiction de la classe de l'image.

Positional embeddings

Plusieurs approches sont disponibles pour l'apprentissage du positional embeddings :

- Aucun plongement
- Plongement 1D : plongement pour la séquence de patch
- Plongement 2D : Plongement par colonne et par ligne
- Plongements relatifs : Encodage de la distance entre deux patchs

Ainsi, la formule finale de l'input est :

$$z0 = [xclass;\ x1pE;\ x2pE; \cdots;\ xNpE] + Epos$$

2.2 CoAtNet

Deux questions sont à se poser pour combiner de manière "optimale" la convolution et le transformateur :

1. Comment combiner la convolution et l'auto-attention au sein d'un même bloc de calcul
2. Comment empiler verticalement différents types de blocs de calcul pour former un réseau complet ?

Combiner convolution et auto-attention

L'idée consiste à utiliser le bloc MBConv, qui utilise la convolution en profondeur avec un goulot d'étranglement résiduel inversé. Ce schéma d'expansion-compression est commun avec le module FFN du transformateur. De plus, la convolution en profondeur et l'auto-attention peuvent toutes deux être exprimées sous la forme d'une somme pondérée par dimension des valeurs dans un champ réceptif prédéfini. Ainsi, la convolution en profondeur peut être exprimée comme suit:

$$y_i = \sum_{j \in \mathcal{L}(i)} w_{i-j} \odot x_j$$

où x_i et y_i sont les entrées et sorties à la position i, w_{i-j} est la matrice de poids à la position $(i-j)$ et $L(i)$ est le voisinage local de i.

En comparaison, l'auto-attention permet au champ réceptif de ne pas être un voisinage local et calcule les poids sur la base de la similarité par paire suivie d'une fonction softmax :

$$y_i = \sum_{j \in \mathcal{G}} \underbrace{\frac{\exp\left(x_i^\top x_j\right)}{\sum_{k \in \mathcal{G}} \exp\left(x_i^\top x_k\right)}}_{A_{i,j}} x_j$$

où G indique l'espace global et x_i, x_j sont deux paires (par exemple deux patchs d'une image)

Cette approche propose plusieurs avantages :

- **Pondération adaptée à l'entrée** : la matrice w_{i-j} est une valeur statique indépendante de l'entrée, tandis que les poids d'attention A_{ij} dépendent de la représentation de l'entrée. L'auto-attention est donc plus encline à capturer les relations entre les différents éléments de l'entrée, au prix d'un risque de sur-apprentissage lorsque les données sont limitées.
- **Équivariance de traduction** : le poids de convolution w_{i-j} s'intéresse au décalage relatif entre i et j plutôt qu'aux valeurs spécifiques de i et j. Cette équivariance de traduction améliore la généralisation dans le cadre d'un ensemble de données de taille limitée.
- **Champ réceptif global** : le champ réceptif plus large utilisé dans l'auto-attention fournit davantage d'informations contextuelles que le champ réceptif local du CNN.

Ainsi, le modèle CoAtNet possède à la fois la pondération adaptée à l'entrée et l'équivariance de traduction grâce à la partie auto-attention et le champ réceptif global grâce au CNN. L'idée proposée par les auteurs est d'additionner un noyau de convolution statique global avec la matrice d'attention adaptative, après ou avant l'initialisation de la softmax :

$$y_i^{\text{post}} = \sum_{j \in \mathcal{G}} \left(\frac{\exp\left(x_i^\top x_j\right)}{\sum_{k \in \mathcal{G}} \exp\left(x_i^\top x_k\right)} + w_{i-j} \right) x_j \quad \text{or} \quad y_i^{\text{pre}} = \sum_{j \in \mathcal{G}} \frac{\exp\left(x_i^\top x_j + w_{i-j}\right)}{\sum_{k \in \mathcal{G}} \exp\left(x_i^\top x_k + w_{i-k}\right)} x_j.$$

Design vertical

Les auteurs ont décidé d'utiliser la convolution pour effectuer le sous-échantillonnage et l'attention relative globale uniquement lorsque la taille des cartes de caractéristiques était suffisamment petite pour être traitée. Le sous-échantillonnage est effectué de la façon suivante, qui consiste à utiliser un schéma à plusieurs étapes avec une mise en commun progressive. Cette approche est divisée en 5 étapes. Cependant la première couche sera toujours une couche convolutive (nommée S0), et la seconde sera toujours un bloc MBConv, utilisé pour réduire la dimensionnalité. Les trois dernières étapes peuvent être soit un bloc de convolution, soit un bloc transformateur. Néanmoins, on doit tenir compte du fait que les blocs de convolution sont nécessairement placés avant les

blocs transformateurs. Ainsi, en notant C pour convolution et T pour transformateur, on obtient les quatre variantes possibles suivantes : C-C-C-C, C-C-C-T, C-C-T-T et C-T-T-T.

Il en résulte 5 modèles qui ont été comparés en termes de généralisation (l'écart entre la perte d'entraînement et la précision de l'évaluation) en utilisant 1,3 millions d'images et la capacité du modèle (la capacité à s'adapter à un grand ensemble de données d'entraînement) en utilisant plus de 3 milliards d'images. En termes de généralisation, les auteurs ont obtenu les résultats suivants :

$$\text{C-C-C-C} \approx \text{C-C-C-T} \geq \text{C-C-T-T} > \text{C-T-T-T} \gg \text{ViT}$$

Et pour la capacité du modèle :

$$\text{C-C-T-T} \approx \text{C-T-T-T} > \text{ViT} > \text{C-C-C-T} > \text{C-C-C-C}$$

Ces comparaisons suggèrent que l'ajout de bloc transformateur n'entraine pas automatiquement une amélioration de la généralisation. Dans notre cas, nous avons choisi d'utiliser l'architecture C-C-T-T pour faire nos expériences.

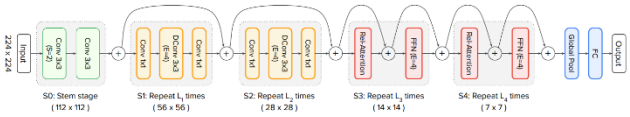

Figure 2: Exemple d'architecture CoAtNet

3. Modèles étudiés

Les modèles que nous avons décidé d'étudier son de tailles inferieur aux modèles présentés dans leurs articles respectifs. Ce choix est principalement dû à deux aspects :

- **Images moins grandes :** Dans nos expériences, nous testons nos modèles sur CIFAR10 et CIFAR100, composés d'images 32x32. Les articles originaux utilisent principalement ImageNet, composé d'images de 224x224. Afin de s'assurer que les modèles n'overfittent pas les données, nous avons donc réduit le nombre de paramètres les constituant.
- **Temps d'entrainement :** Le modèle ViT-L/16 de 307M de paramètre nécessite 680 jours d'entrainement sur un core TPUv3 et CoAtNet-3 de 168M de paramètre, 580 jours. Ces valeurs ne sont pas du tout réalisables compte tenu des machines et du temps disponible pour notre étude. Des modèles de tailles plus petites seront donc mieux adaptés à nos moyens.

Afin de déterminer les tailles des modèles, nous allons étudier les relations entre les grandeurs des modèles proposés dans leurs articles et leur nombre de paramètres afin de s'assurer un choix cohérent avec le fonctionnement de l'architecture.

3.1 ViT

Pour le ViT, 4 paramètres du modèle influencent la taille du modèle :

- Le nombre de couche (*L*)
- La dimension cachée (*D)*
- La taille du MLP (*M*)
- Le nombre de tète d'attention (*H)*

Le papier nous présente ces valeurs :

Tableau 1: Nombre de paramètres et valeurs des paramètres pour les modèles ViT

L	D	M	H	#Params
12	768	3072	12	**86M**
24	1024	4096	16	**307M**
32	1280	5120	16	**632M**

On peut alors définir des courbes de tendance afin de prédire les valeurs de ces paramètres en fonction du nombre de paramètre *x* :

$$L = 1{,}3275x^{0{,}4978}$$
$$D = 246{,}98x^{0{,}2527}$$
$$M = 987{,}91x^{0{,}2527}$$
$$H = 6{,}2041x^{0{,}1536}$$

Cela nous permet de définir les paramètres de notre modèle en fonction du nombre de paramètre souhaité.

3.2 CoAtNet

Pour CoAtNet, nous avons 10 paramètres qui influencent le nombre de paramètre du modèle, les nombre de passage dans chaque couche, de L_0 a L_4 et les tailles de ces couches, de D_0 a D_4.

Tableau 2: Nombre de paramètres et valeurs des paramètres pour les modèles CoAtNet

L_0	2	2	2	2	2
L_1	2	2	2	2	2
L_2	3	6	6	6	12
L_3	5	14	14	14	28
L_4	2	2	2	2	2
D_0	64	64	128	192	192
D_1	96	96	128	192	192
D_2	192	192	256	384	384
D_3	384	384	512	768	768
D_4	768	768	1024	1536	1536
#Params	**25M**	**42M**	**75M**	**168M**	**275M**

Comme les paramètres L_0, L_1 et L_4 ne changent jamais selon les modèles, nous avons décider de garder ces valeurs pour nos modèles aussi.

3.3 Nos modèles

Nous avons décidé de crée des grands modèles (Our_ViT_L et Our_CoAtNet_L) proche de 8M de paramètres ainsi qu'une petite version (Our_ViT_S et Our_CoAtNet_S) proches de 2 M de paramètres dont voici les tailles :

Tableau 3: Valeur de paramètres de nos modèles ViT

Modèle	L	D	M	H	#Param
Our_ViT_S	2	256	1024	6	**2.0M**
Our_ViT_L	4	384	1536	8	**8.2M**

Tableau 4: Valeur des paramètres de nos modèles CoAtnet

Model	Our_CoAtNet_S	Our_CoAtNet_L
$\mathbf{L_0}$	2	2
$\mathbf{L_1}$	2	2
$\mathbf{L_2}$	2	3
$\mathbf{L_3}$	3	4
$\mathbf{L_4}$	2	2
$\mathbf{D_0}$	32	64
$\mathbf{D_1}$	64	92
$\mathbf{D_2}$	92	192
$\mathbf{D_3}$	128	256
$\mathbf{D_4}$	256	512
#Params	**2.8M**	**8.1M**

4. Résultats

4.1 Comparaison des ViT

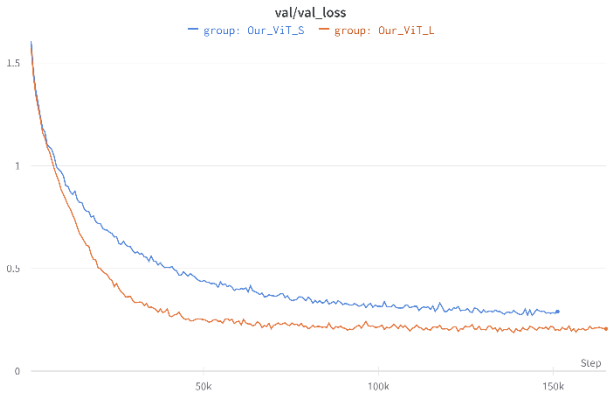

Figure 3: Comparaison de la loss pour nos deux modèles ViT

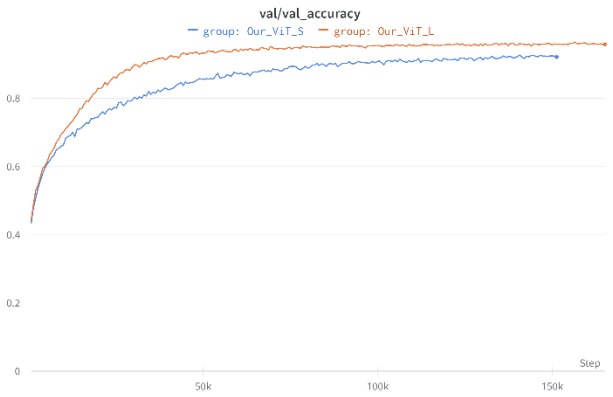

Figure 4: Comparaison de l'accuracy pour nos deux modèles ViT

Nos deux tailles de modèles produisent des résultats satisfaisant sur le jeu de donnée CIFAR10, en atteignant 96% de précision pour le modèle large, et 92% pour le petit modèle.

Comme attendu, les performances du modèle large sont supérieures à celles du petit modèle, indiquant que le choix de la taille des modèles (8M et 2M de paramètre) sont adaptées à la tache de classification que nous avons évaluée.

Pour atteindre ces résultats, nous avons entrainé Our_ViT_L pendant 470 epochs et Our_ViT_S pendant 430 epochs.

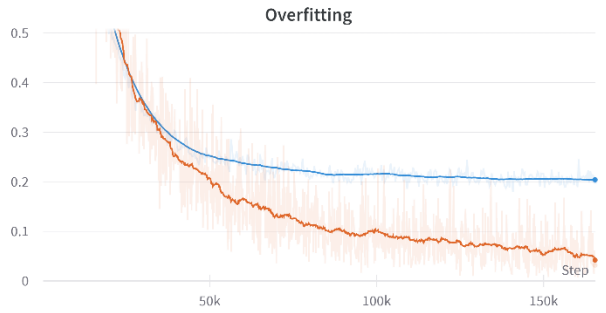

Figure 5: Representation du surapprentissage pour Our_ViT_L

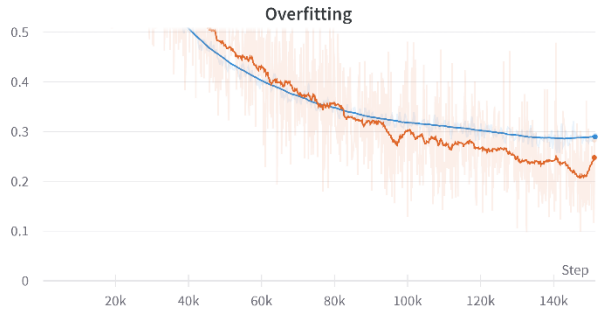

Figure 6: Représentation du surapprentissage pour Our_ViT_S

On peut remarquer une grande différence concernant ces deux modèles en étudiant dans quelle mesure ils se trouvent en surapprentissage. Pour cella nous comparons sur Figure 5 et Figure 6 la valeur de loss sur le dataset d'entrainement (en orange) et la valeur sur le dataset de validation (en bleu).

On remarque donc que pour le plus grand modèle, l'écart entre ces deux valeurs est grand, indiquant que le modèle est en surapprentissage. Le petit modèle entre aussi en surapprentissage mais dans de moindre mesure et plus tard que sa version large.

4.1 Comparaison des CoA

Figure 7: Comparaison de la loss pour nos deux modèles CoAtNet

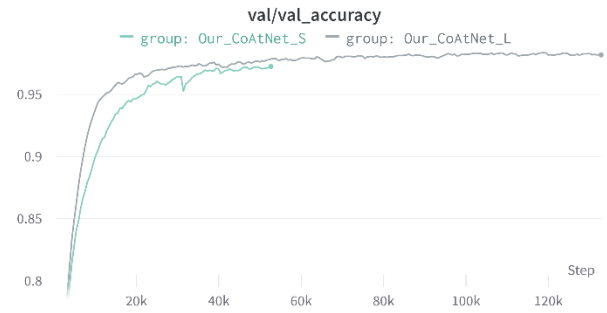

Figure 8: Comparaison de l'accuracy pour nos deux modèles CoAtNet

Entre les deux modèles CoAtNet, l'ecart de performance est moins important puisque Our_CoAtNet_L atteins de 98% la ou Our_CoAtNet_S atteins les 97% de précision.

Cette faible différence peut s'expliqué par des modèles trop performants pour les taches testées.

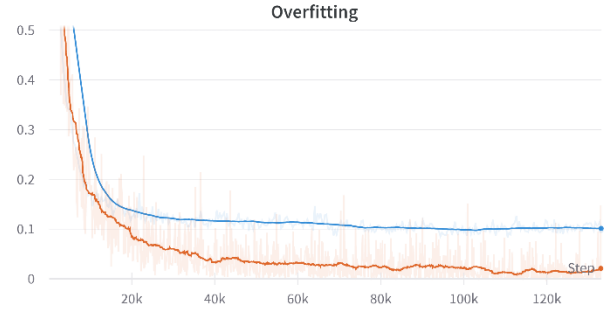

Figure 9: Representation de l'overfitting pour Our_CoAtNet_L

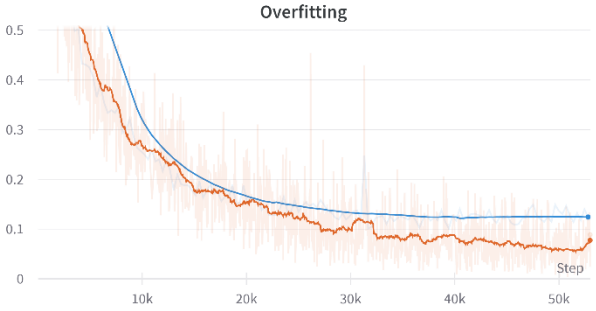

Figure 10: Representation de l'overfitting pour Our_CoAtNet_S

Pour nos CoAtNet, les deux modèles présentent un cas fort d'overfitting. Comme pour nos ViT, on observe que le surapprentissage est plus important sur le grand modèle que sur le petit, ce qui était prévisible. De plus, ces surapprentissage élevés confirment l'hypothèse que nos modèles CoAtNet étaient trop complexe pour la tâche.

4.1 Comparaison ViT/CoAtNet

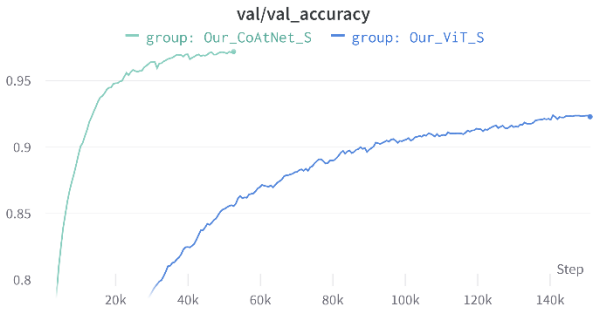

Figure 11: Comparaison de précision entre nos petits modèles (2M paramètres)

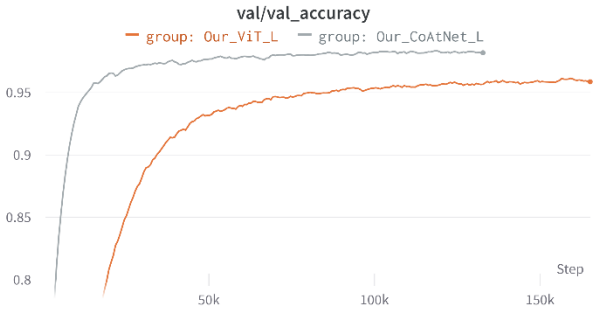

Figure 12: Comparaison de précision entre nos grand modèles (8M paramètres)

Dans les deux cas de taille de modèles, on observe que les CoAtNet offrent de bien meilleures performances que les ViT. De plus selon Figure 11 et Figure 12, on peut relever qu'en plus d'être plus précis, on remarque aussi que les CoAtNet ont besoin de moins d'entrainement pour atteindre ces performances.

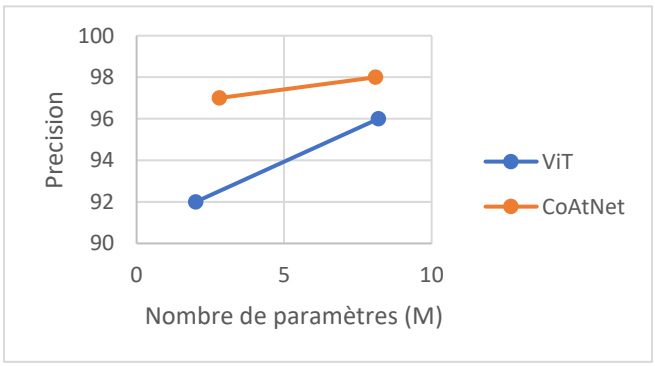

Figure 13: Comparaison des précisions des types de modèles en fonction du nombre de paramètre

Dans la Figure 13, nous comparons directement les performances des types de modèles en fonction du nombre de paramètre, mettant en avant la conclusion énoncée précédemment.

4.1 Comparaison des grands modèles sur CIFAR100

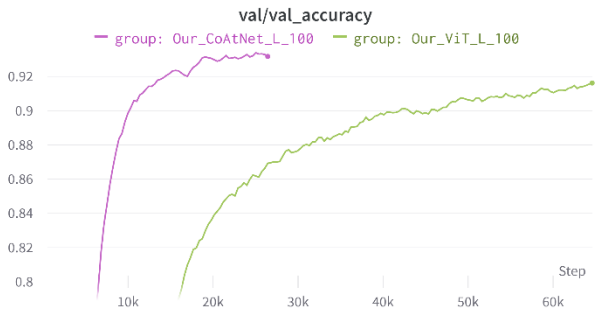

Figure 14: Comparaison de la précision des grands modèles sur CIFAR100

CIFAR100 est un jeu de donnée similaire a CIFAR10, mais qui comporte 100 classes différentes au lieu de 10.

Nous avons essayé d'utiliser nos plus grands modèles sur CIFAR100 afin de les comparer sur une tache plus compliquée afin de voir leurs limites.

La Figure 14 donne des résultats similaires a ceux obtenus avec CIFAR10, avec le CoAtNet restant nettement supérieur au modèle ViT.

5. Conclusion

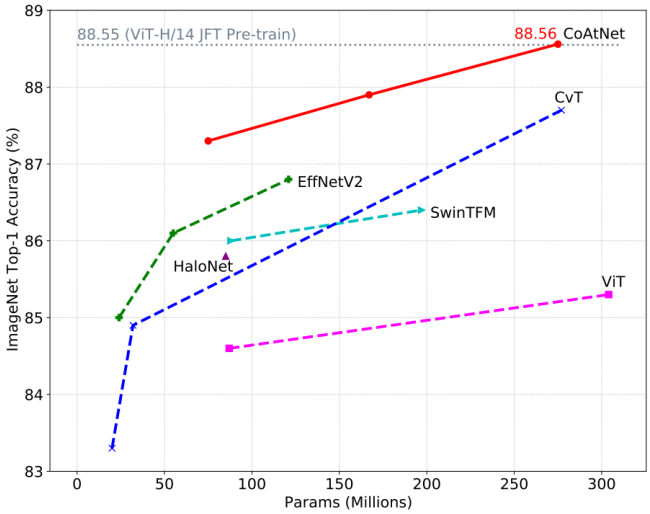

Figure 15: Résultat attendu présent dans l'article du CoAtNet

Nous avons montré que même à petite échelle, les modèles de type CoAtNet sont plus performants que les modèles ViT, reproduisant a plus petite échelle les résultats présentés dans Dai, Zihang, et al. (Figure 15). Cela met en évidence l'efficacité de mêler a la fois des idées des modèles convolutionnels et des modèles basés sur l'attention comme les ViT.

Références

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Dai, Z., Liu, H., Le, Q. V., & Tan, M. (2021). Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, *34*, 3965-3977.

Lien du projet Weight and Biases