Vision Transformers : ViT, LeViT and Swim

Computer Vision

Alexandre Abela
Vincent Lébé
Théo Saulais

Table of Contents

Table of Contents

Transformers Architecture

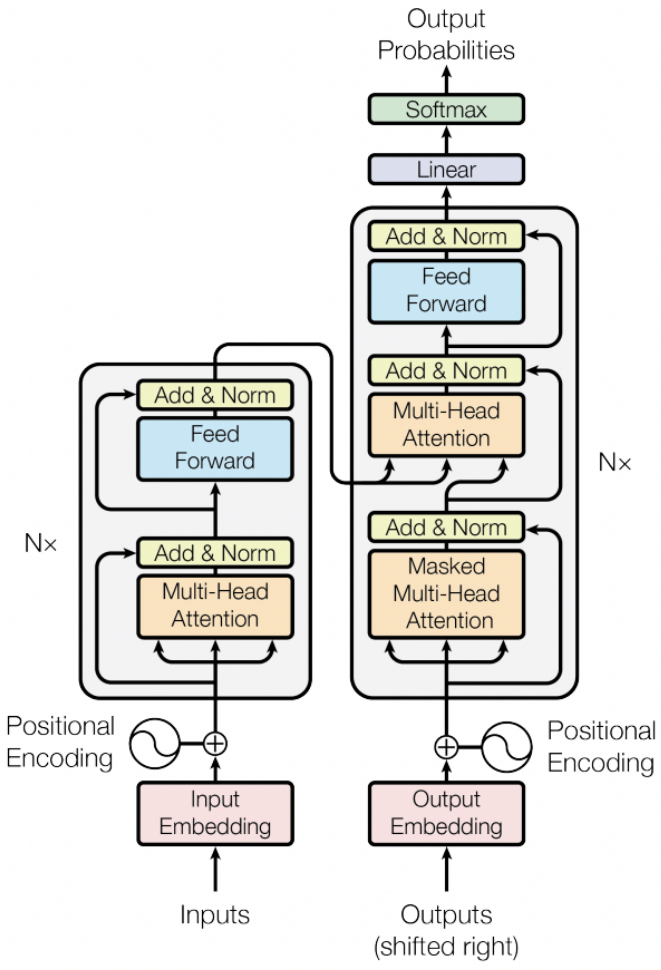


Figure 1: The Transformer - model architecture.

Transformers Architecture

Encodeur →

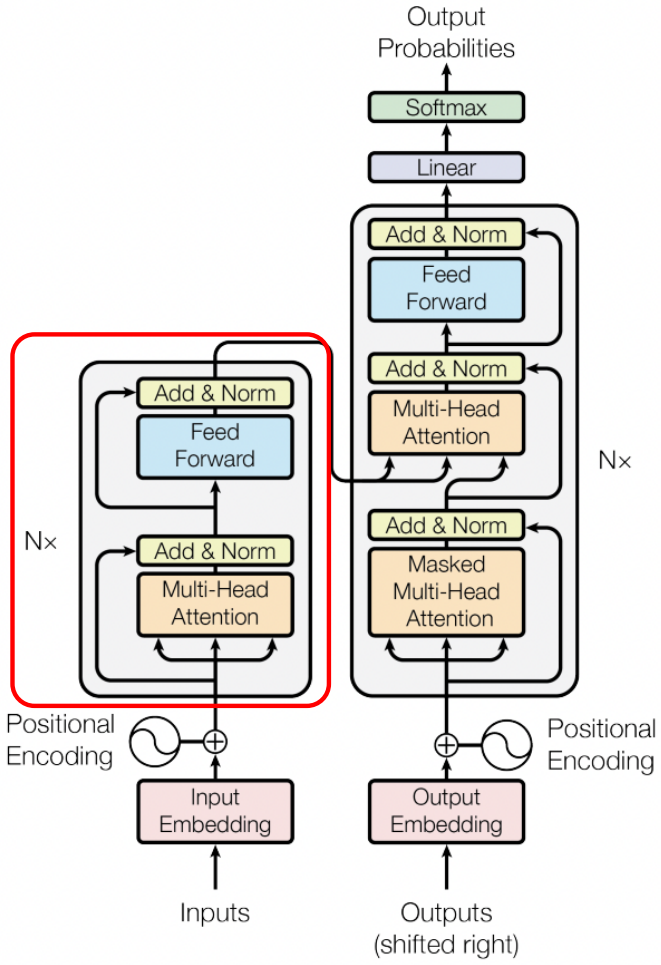


Figure 1: The Transformer - model architecture.

Transformers Architecture

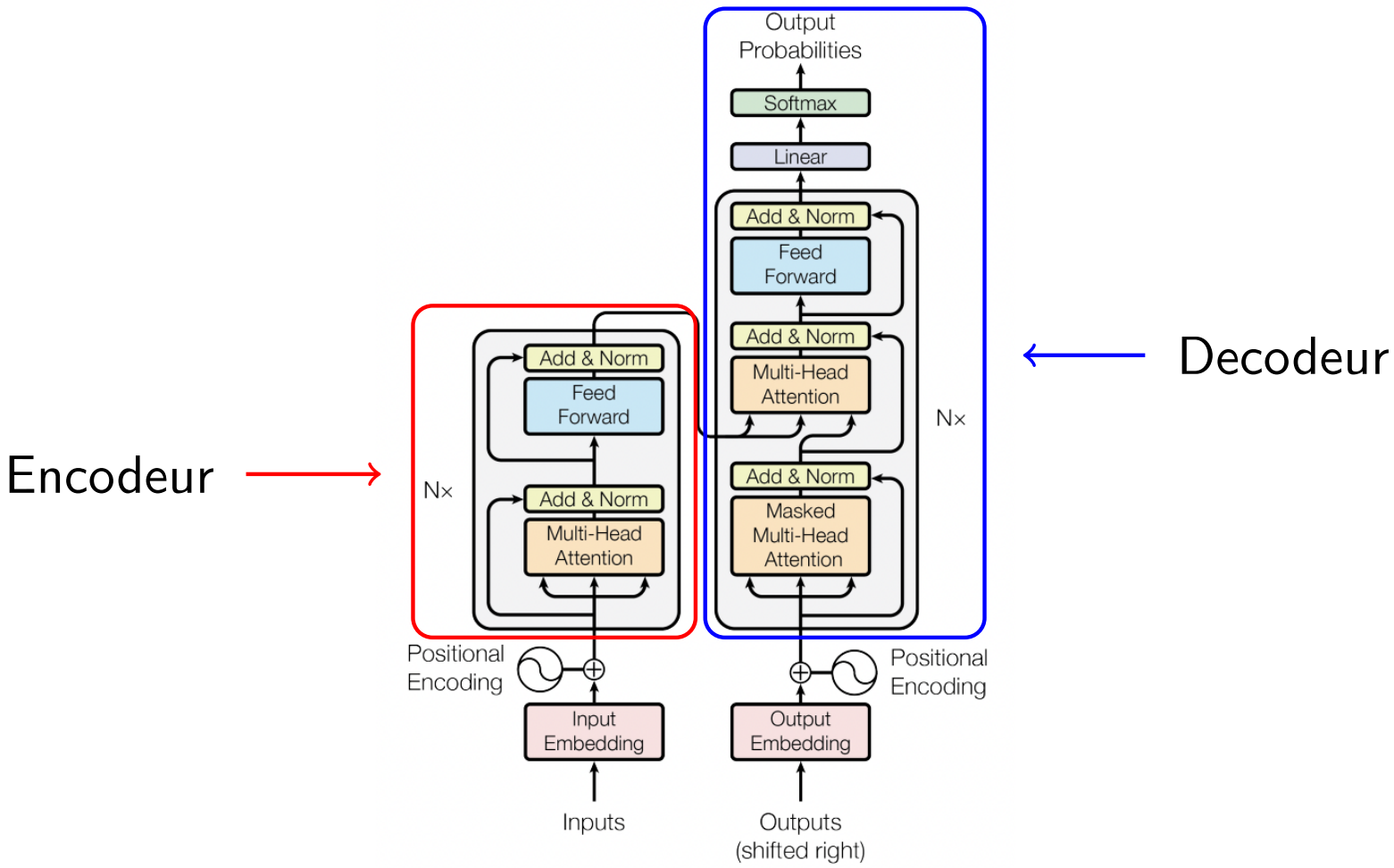


Figure 1: The Transformer - model architecture.

Attention

Multi-head attention

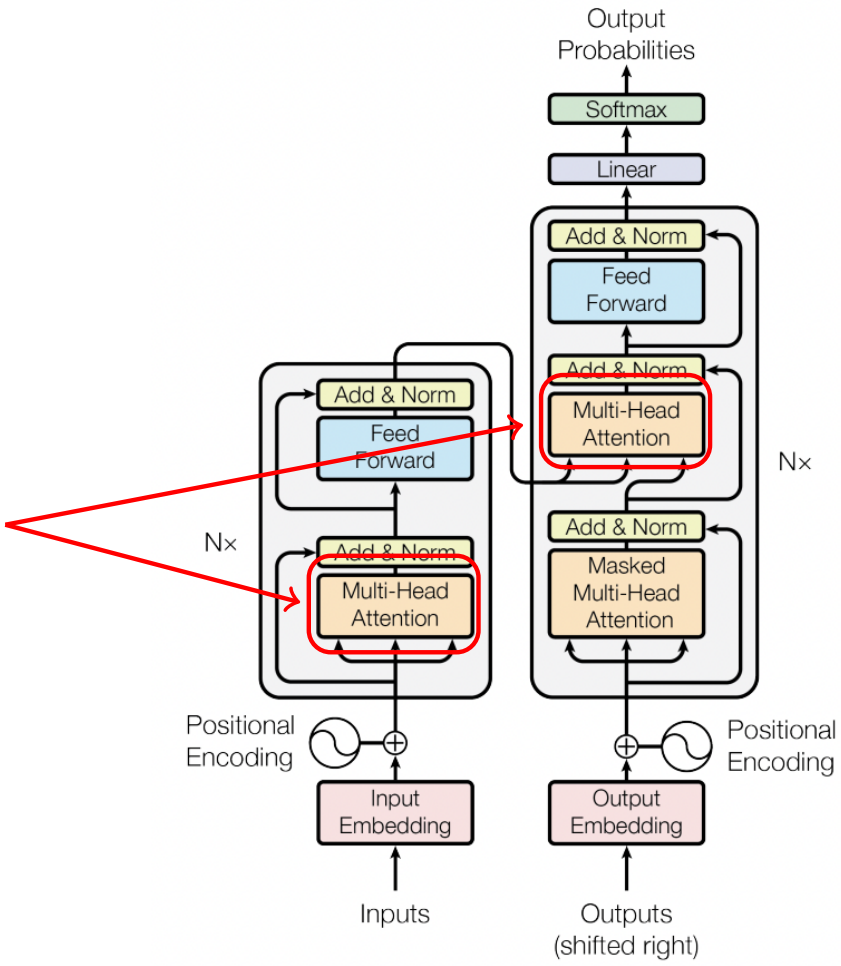
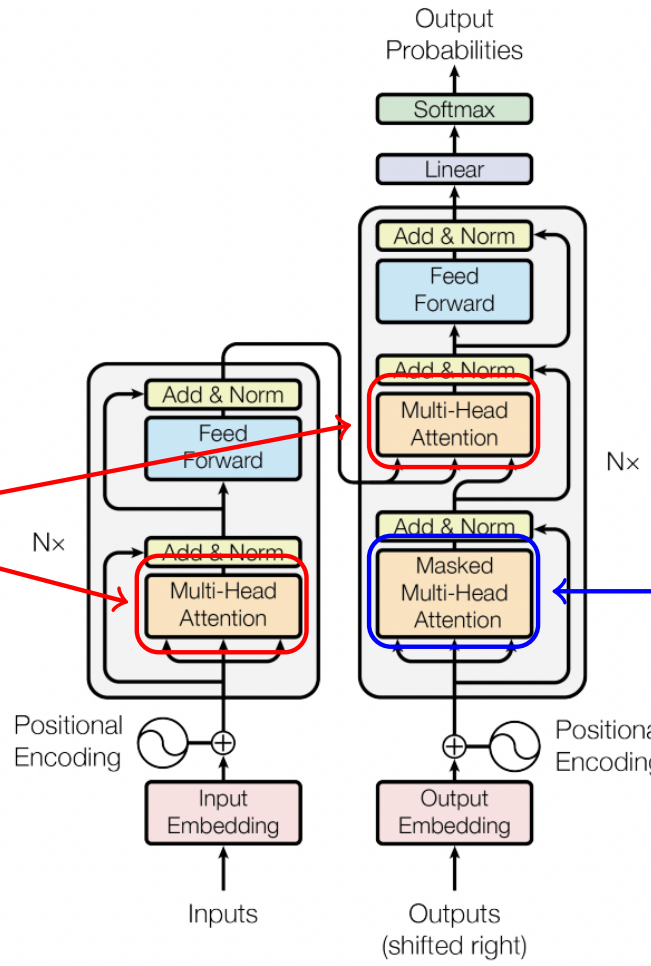


Figure 1: The Transformer - model architecture.

Attention

Multi-head attention



Masked attention

Figure 1: The Transformer - model architecture.

Self attention

Self Attention

L'idée derrière **self attention** est de déterminer quelle partie de la phrase est importante pour le mot étudié et de stocker cette information sous forme de vector.

- Le $\rightarrow$ Le gentil chien noir$\rightarrow$ 0.71

Self attention

Self Attention

L'idée derrière **self attention** est de déterminer quelle partie de la phrase est importante pour le mot étudié et de stocker cette information sous forme de vector.

- Le $\rightarrow$ Le gentil chien noir$\rightarrow$ 0.71
- Le $\rightarrow$ Le gentil chien noir$\rightarrow$ 0.09

Self attention

Self Attention

L'idée derrière **self attention** est de déterminer quelle partie de la phrase est importante pour le mot étudié et de stocker cette information sous forme de vector.

- Le $\rightarrow$ Le gentil chien noir$\rightarrow$ 0.71
- Le $\rightarrow$ Le gentil chien noir$\rightarrow$ 0.09
- Le $\rightarrow$ Le gentil chien noir$\rightarrow$ 0.38

Self attention

Self Attention

L'idée derrière **self attention** est de déterminer quelle partie de la phrase est importante pour le mot étudié et de stocker cette information sous forme de vector.

- Le $\rightarrow$ Le gentil chien noir$\rightarrow$ 0.71
- Le $\rightarrow$ Le gentil chien noir$\rightarrow$ 0.09
- Le $\rightarrow$ Le gentil chien noir$\rightarrow$ 0.38
- Le $\rightarrow$ Le gentil chien noir$\rightarrow$ 0.04

Self attention

Attention

L'attention peut être décrite comme la correspondance d'une requête (query) et d'un ensemble clé-valeur (key-value) d'une base de données vers une sortie où les requête, les clés, les valeurs et la sortie sont tous des vecteurs :

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{Q.K^T}{\sqrt{d_k}}).V$$

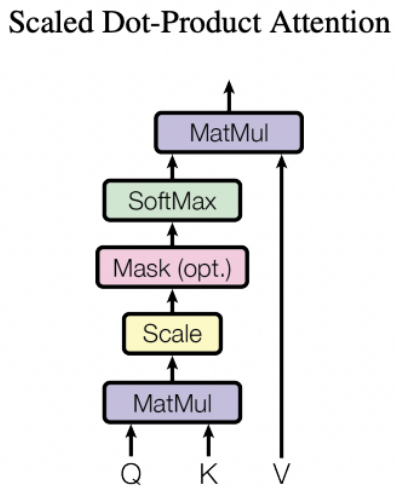


Figure: Scaled Dot-Product Attention pris de [1]

Self attention

Multi-head attention

Multi-head Attention

L'idée derrière **multi-head attention** est la même que le self-attention mais en omettant cette fois-ci l'importance du mot pour lui-même et de recommencer ce processus autant de fois qu'il y a de têtes mais pour des matrices W^q, W^k et W^v.

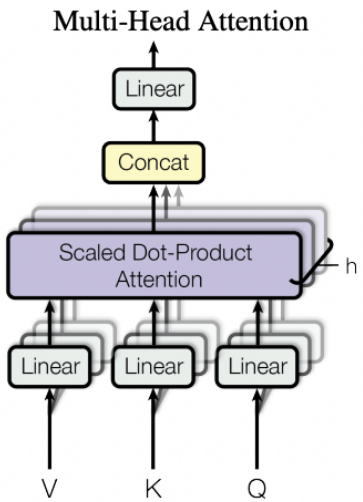


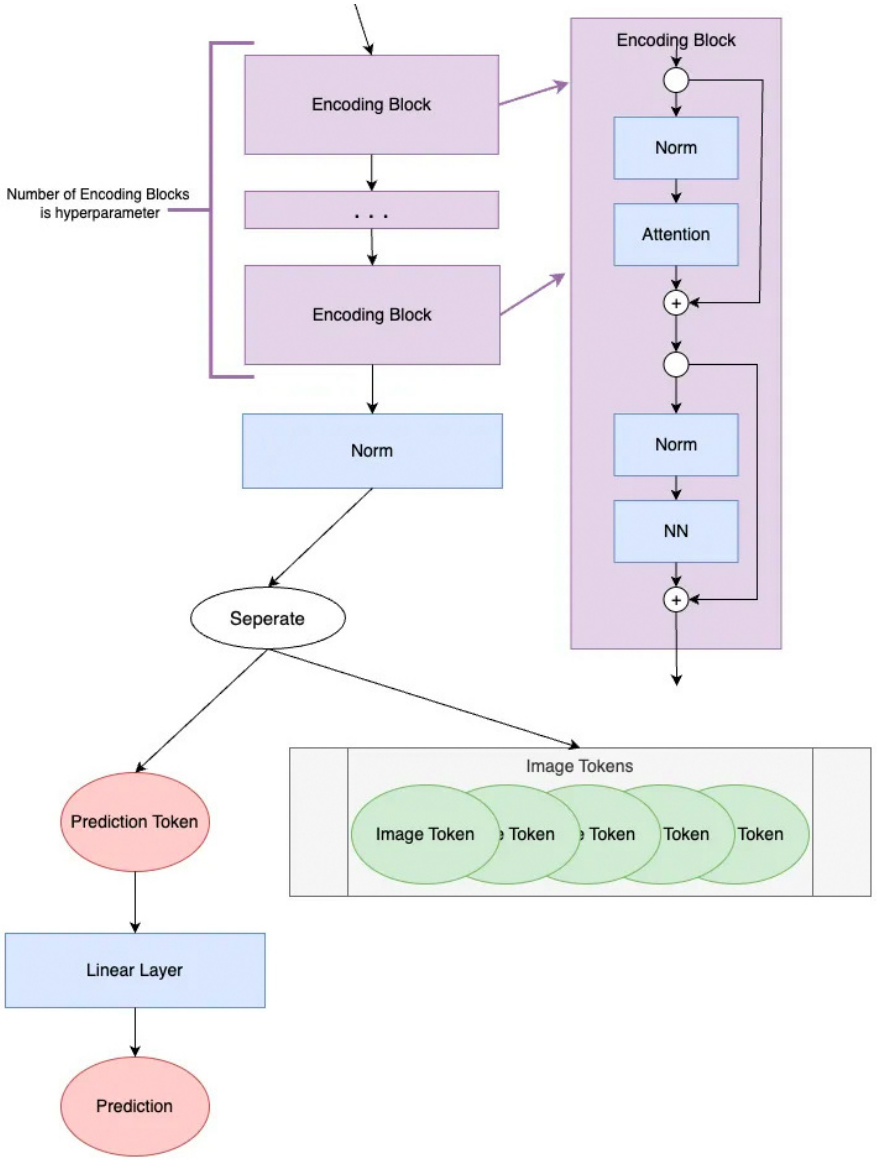
Figure: Multi-head Attention pris de [1]

Table of Contents

Introduction aux Vision Transformers

- **Transformer Architecture**
  - Utilisation d'un modèle transformer pour capturer les relations entre les patchs d'image.
  - Patchs devenant des "mots" dans la séquence du modèle.
- **Applications**
  - Adaptation aux tâches de vision par ordinateur.
  - Amélioration des performances dans la classification et la détection d'objets.
- **"Une image vaut 16x16 mots"**
  - Concept central dans l'architecture ViT.
  - Chaque patch est traité comme un "mot" dans la séquence du modèle.
  - Cela permet au transformer de saisir des relations complexes et à longue portée entre les différentes parties de l'image.

Architecture ViT



Tokenization de l'image d'entrée

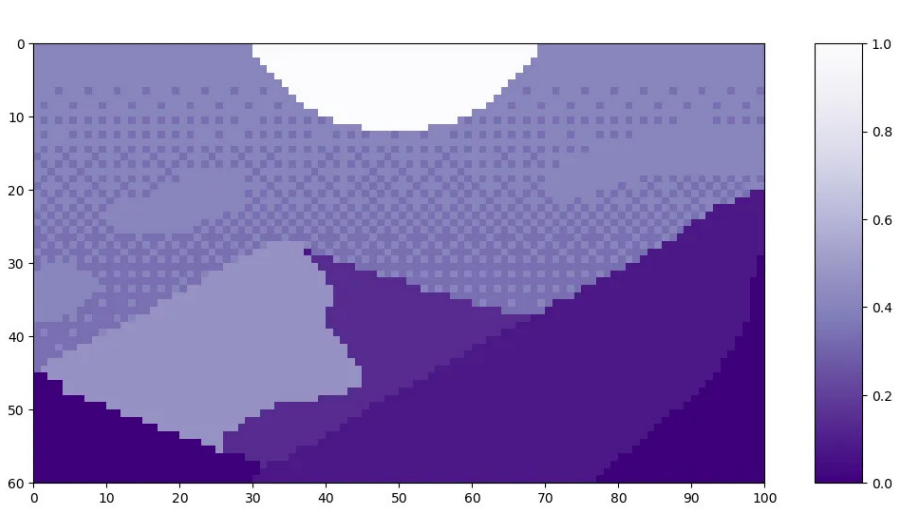
Conversion de l'image en N patch (ou token) de taille P :

$$N = \frac{HW}{P^2}$$

Chaque patch est de longueur

$$C \cdot P^2$$

Tokenization de l'image d'entrée



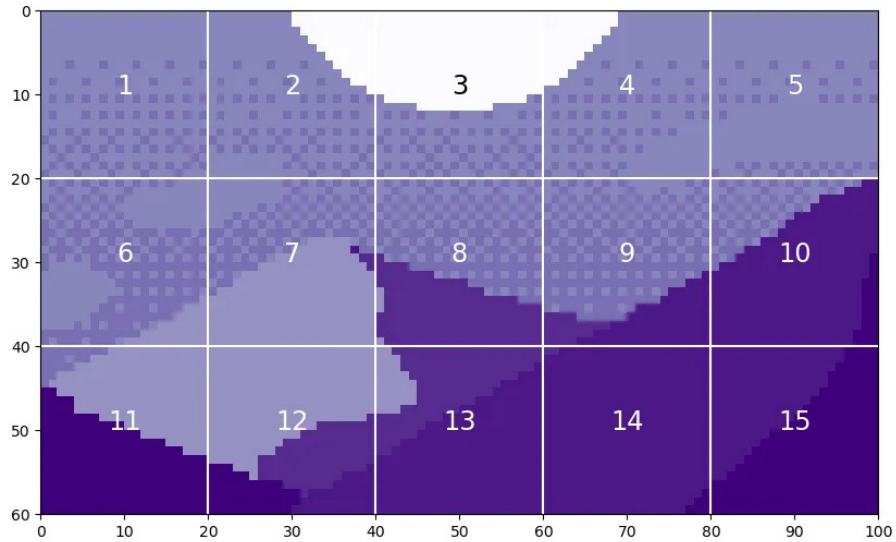
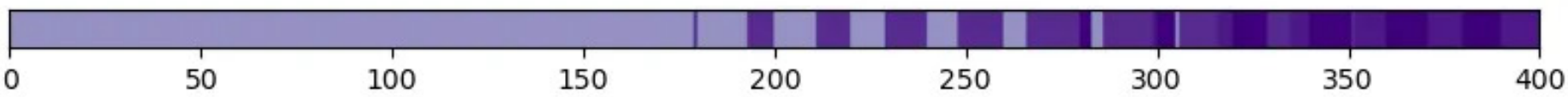


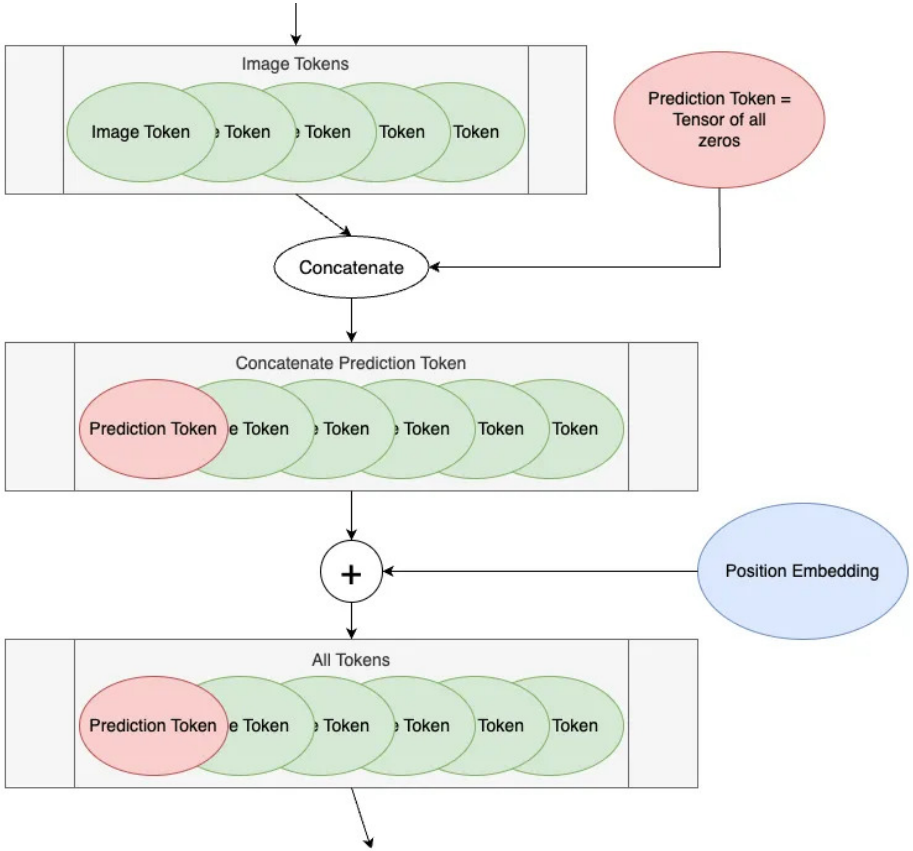
Illustration de la slide précédente à l'aide du pixel art "Mountain at dusk" par Luis ZUNO.

On représente ici le tokenization du patch n°12 après sa "flattenisation"



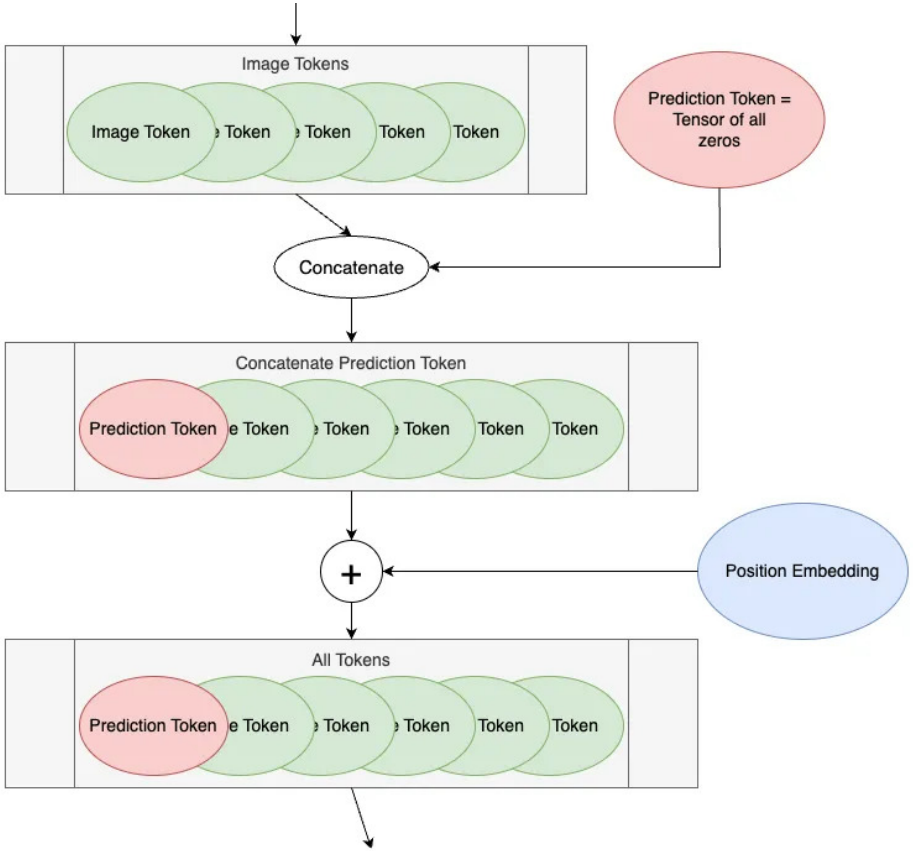
Concaténation du token de prédiction

- Token utilisé à la sortie des blocs d'encodage pour effectuer la prédiction finale.
- Initialisé vide (équivalent à 0).
- Recueille les informations des tokens de l'image.

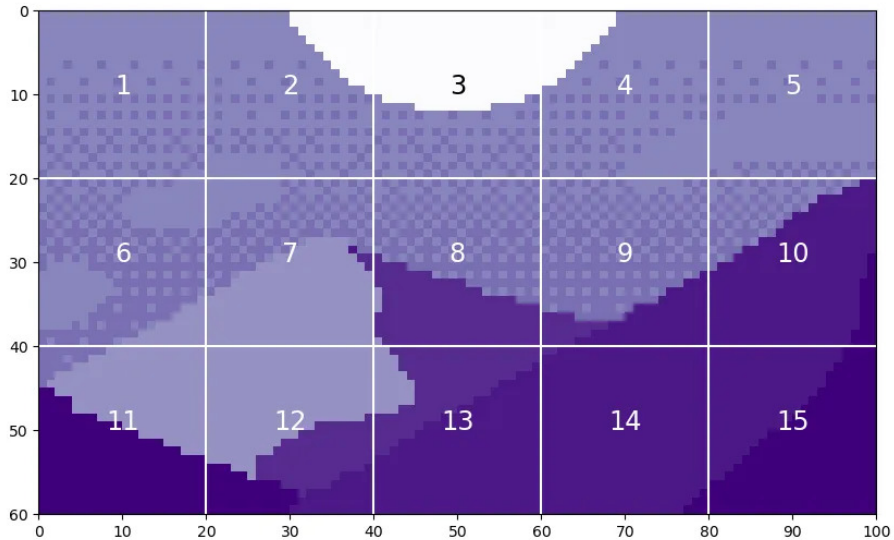


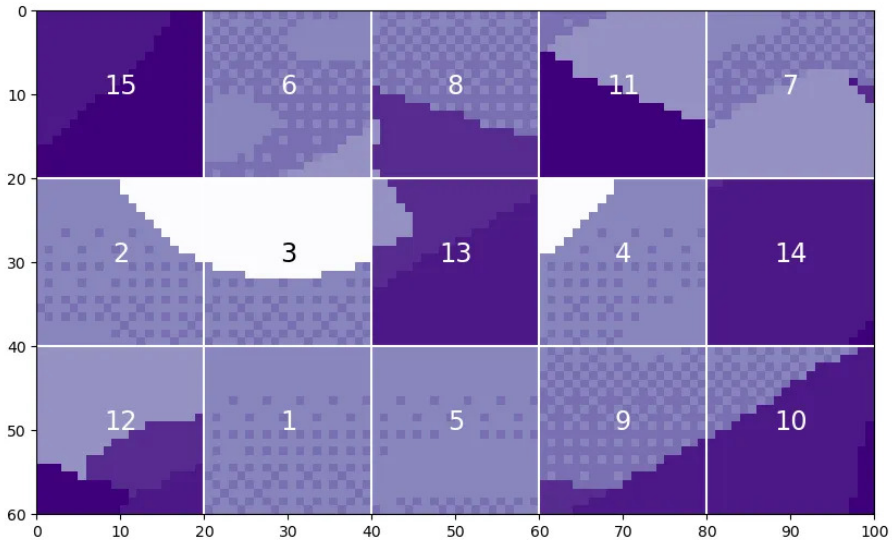
Ajout du positional embedding

- Ajout d'un positional embedding pour les tokens.
- Permet au transformer de comprendre l'ordre des tokens.
- Important : c'est une addition, pas une concaténation !



Importance du positional embedding

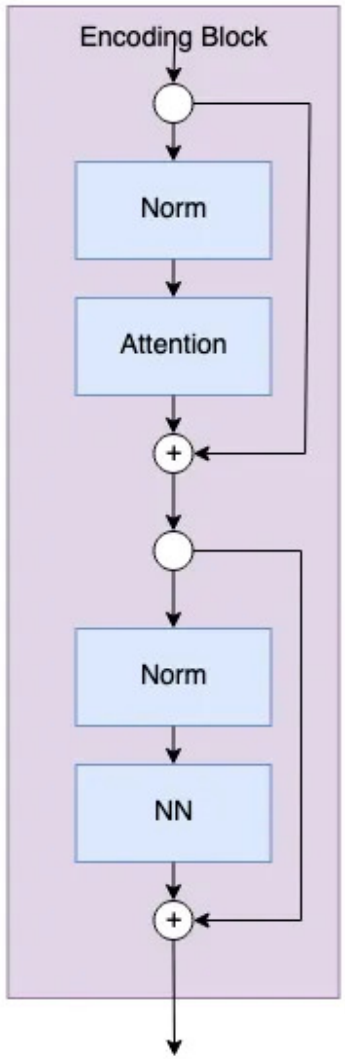




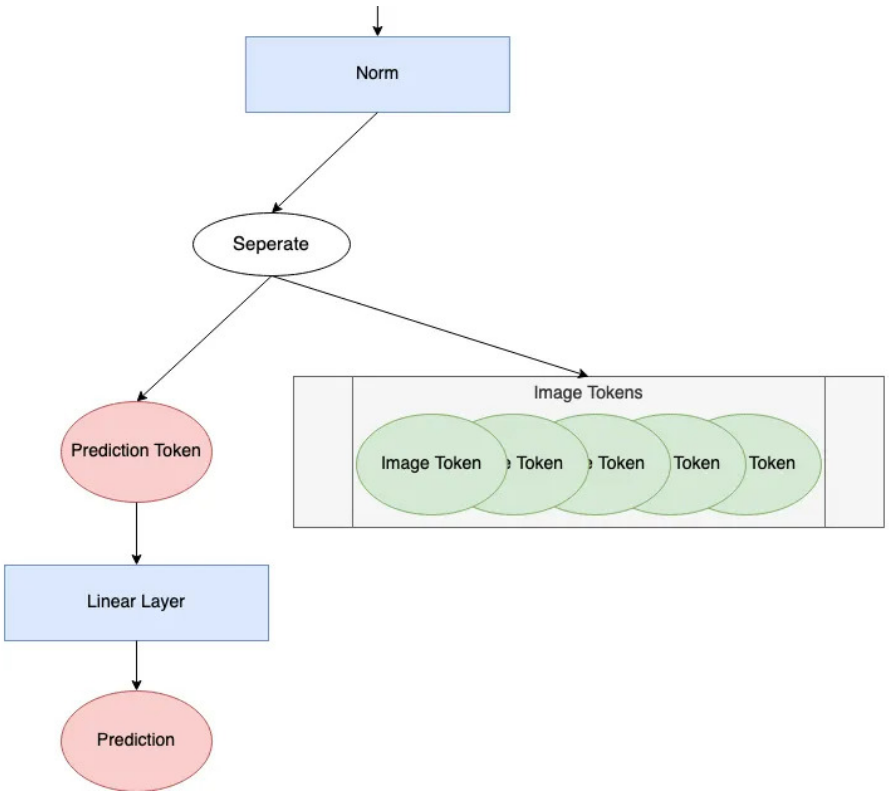
Sans le positionnal embedding, ces deux images seraient interprétées de la même façon, ce qui est problématique.

Encodeur

- La partie encodeur du ViT est semblable à celle d'un transformers en NLP.
- Le nombre de blocs encodeurs est un hyperparamètre à définir.
- Le réseau de neuronne à la fin est composé d'une couche dense, une couche d'activation et une autre couche dense.



Prédiction



À l'inverse du transformers en NLP, les modèles ViT n'ont pas de décodeur. En effet la prédiction est réalisé par un réseau de neurones.

Le réseau dépend du modèle. Par exemple,

- Une image vaut 16x16 mots utilise **un Multi Layer Perceptron**.
- Les Tokens-to-Token utilisent une **couche linéaire**.

Limites du ViT

Le gros problème des Vision Transformers réside dans leurs datasets d'entrainement.

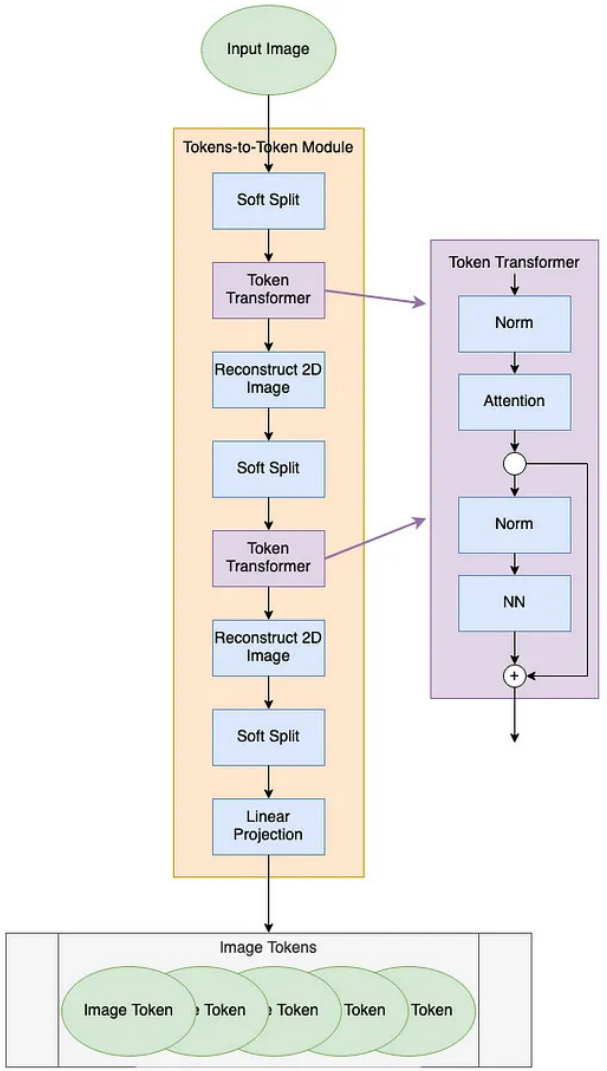
- Il s'agit d'immenses datasets.
- Dans le cas du 16x16 mots, le datasets n'est pas public.
- Pour des taches autres que la classification d'images, les datasets ne sont pas toujours disponibles.

$\Rightarrow$ Apparition des Tokens-to-Token ViT

Tokens-to-Tokens (T2T) Module

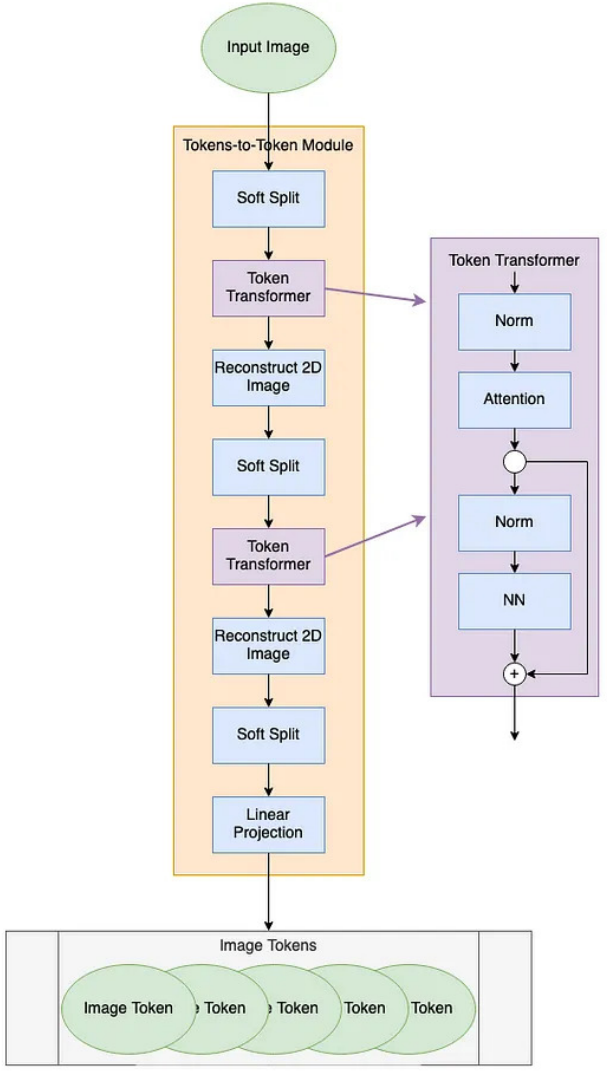
L'architecture du T2T ViT diffère légèrement de celle du ViT. Dans cette partie, on se concentre uniquement sur le module T2T qui vient se placer en entrée du transformers.

Au lieu de simplement diviser l'image en patches, le module T2T calcule sequentiellement l'attention entre les jetons et les regroupe afin de capurer une structure supplémentaire dans l'image.

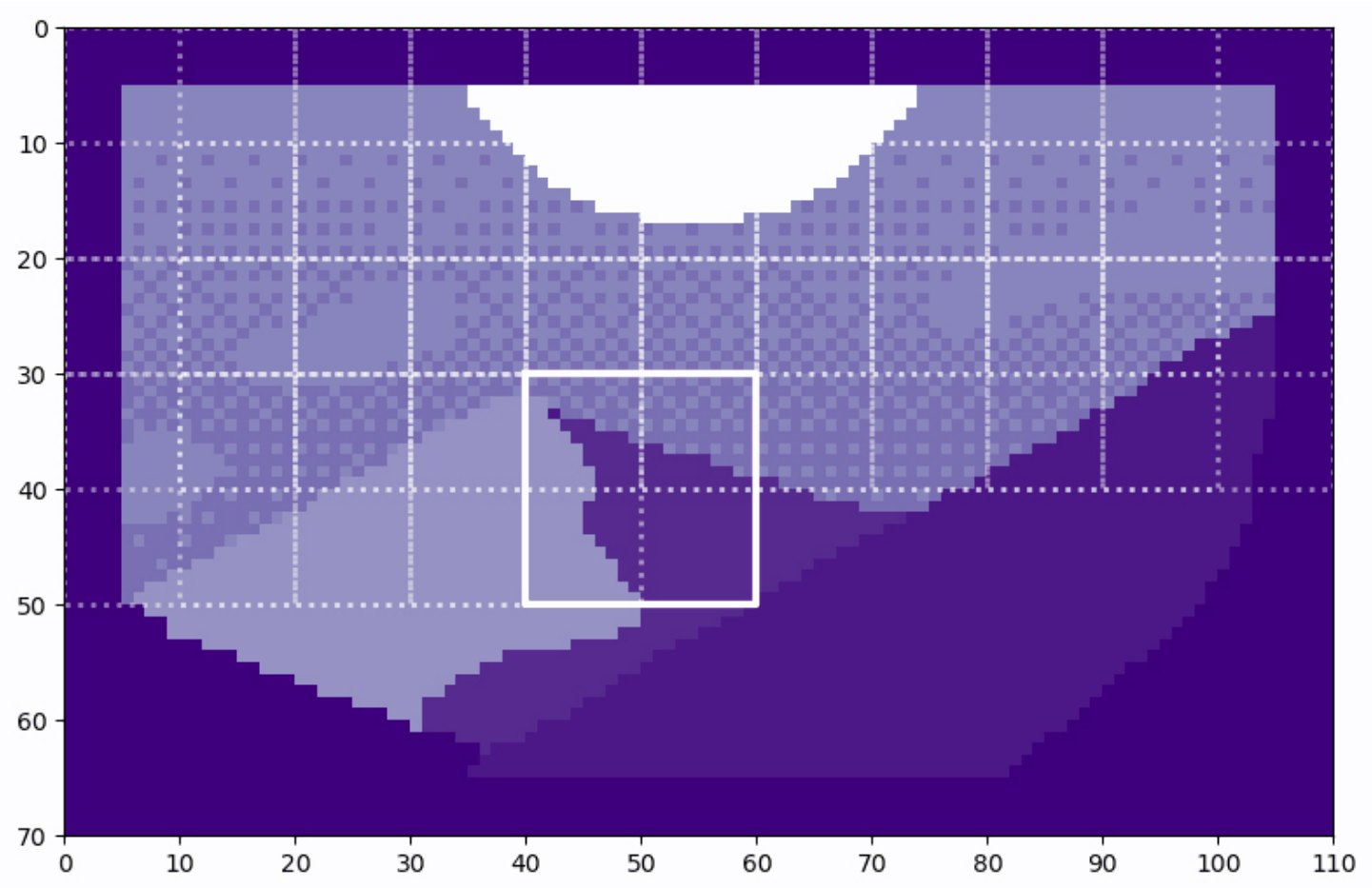


Soft split

- Visible en bleu dans le module T2T.
- Assure la séparation de l'image en patches.
- À l'inverse du modèle ViT, le soft-split génère des patches qui se chevauchent.

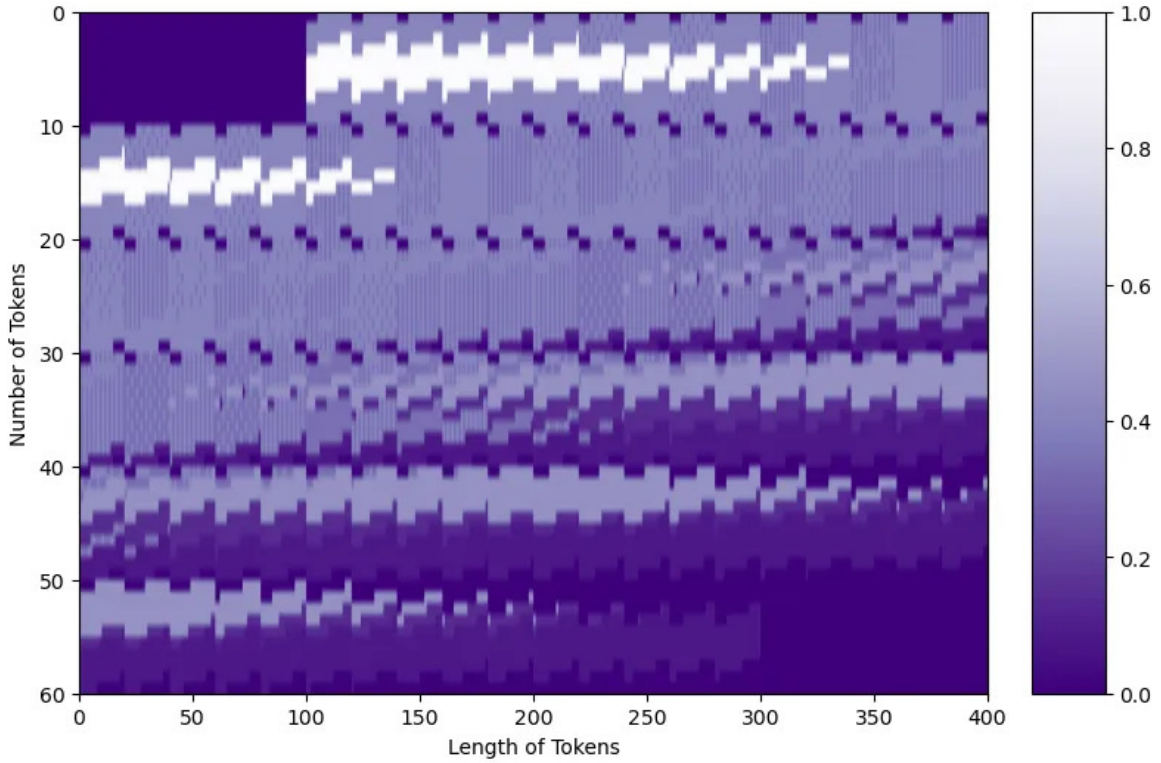


Soft-split



$$Ntok = \left\lfloor \frac{(h + 2 \cdot p - (k-1) - 1)}{s} + 1 \right\rfloor \cdot \left\lfloor \frac{(w + 2 \cdot p - (k-1) - 1)}{s} + 1 \right\rfloor$$

Soft-split



En appliquant la tokenization puis la flattenization de chaque patches, on obtient ainsi cette image que l'on va passer dans le bloc encodeur.

Utilisations usuelles

Les ViT et T2T ViT sont complémentaires et peuvent être utilisés dans différents cas de figures :

Vision Transformers "classique"

- Classification d'images
- Détection d'objet
- Reconnaissance faciale
- Segmentation d'images

Tokens-to-token ViT

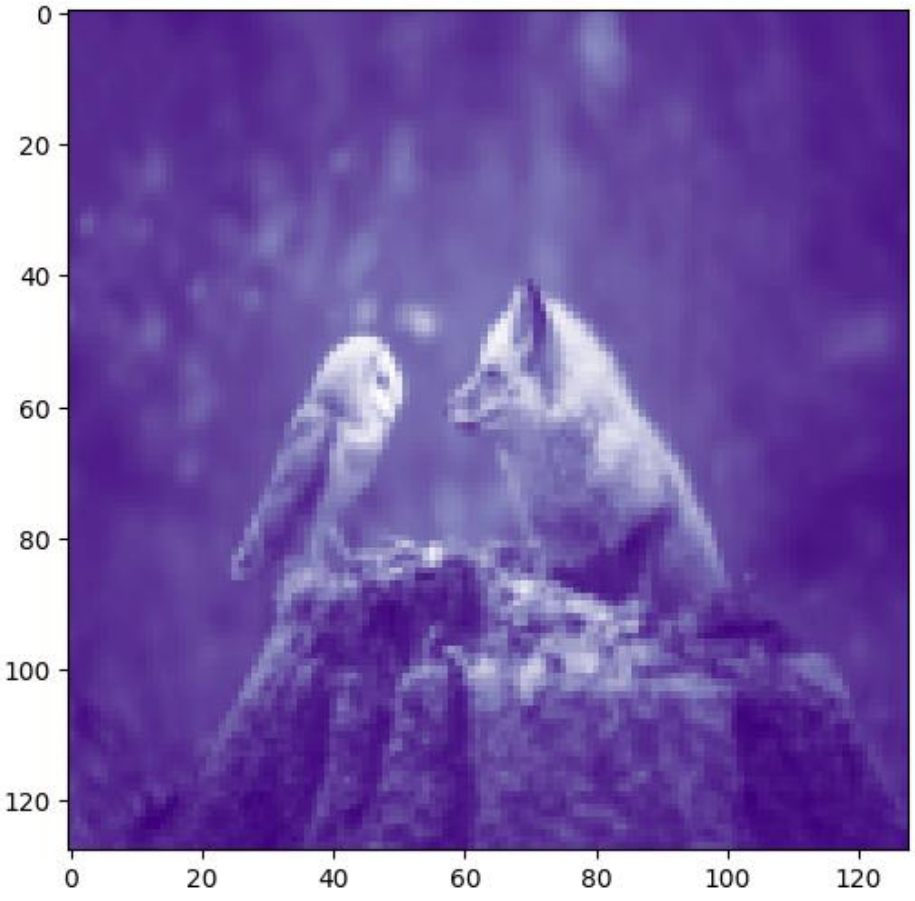
- Segmentation d'images
- Génération d'images
- Traduction d'images
- Super résolution d'images

Table of Contents

Problèmes avec ViT : Images haute résolution

128px

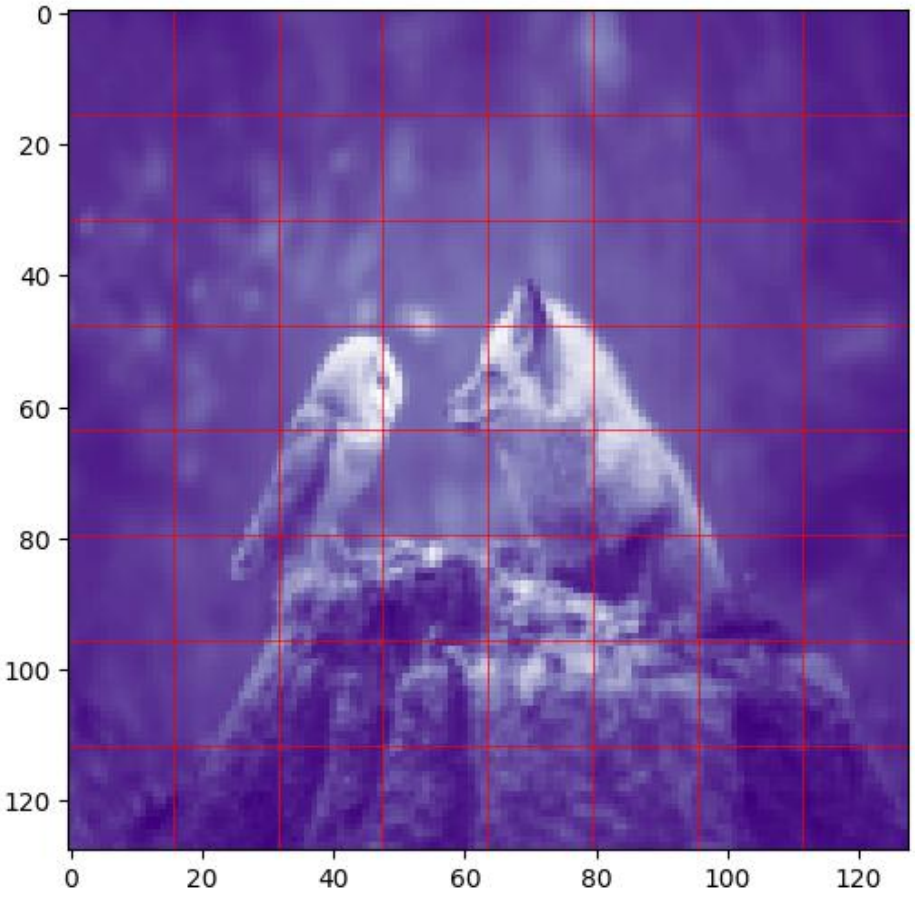
128px



Problèmes avec ViT : Images haute résolution

128px

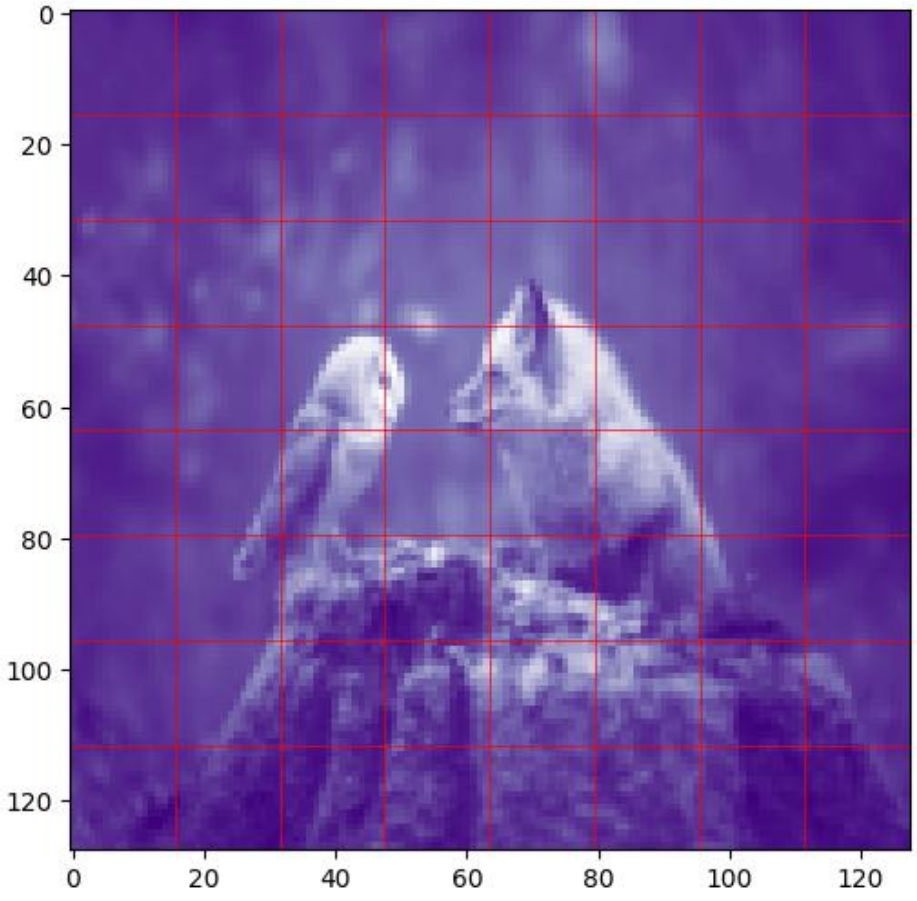
128px



Problèmes avec ViT : Images haute résolution

128px

128px

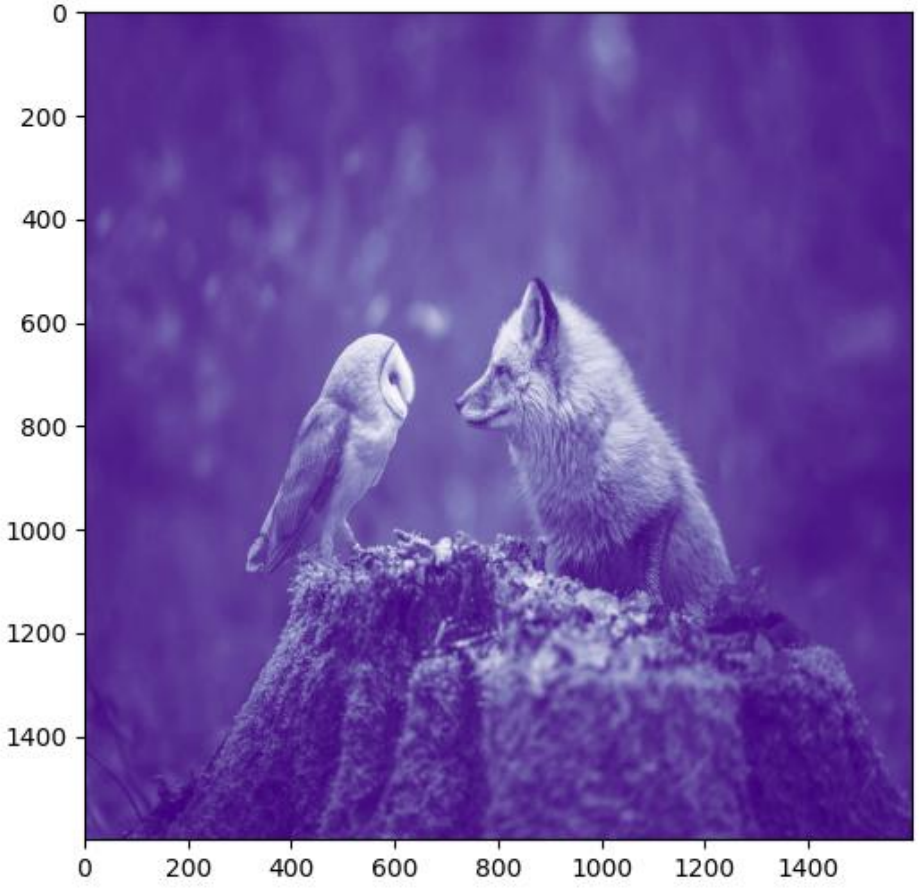


$\longrightarrow$ 64 tokens

Problèmes avec ViT : Images haute résolution

1600px

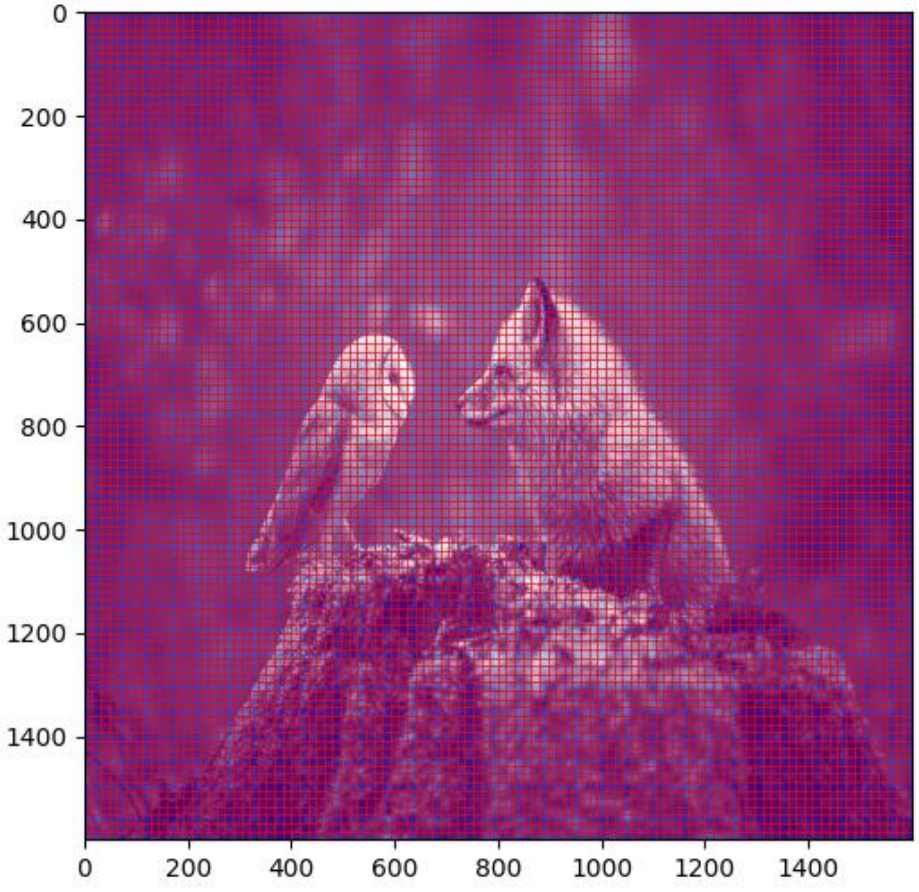
1600px



Problèmes avec ViT : Images haute résolution

1600px

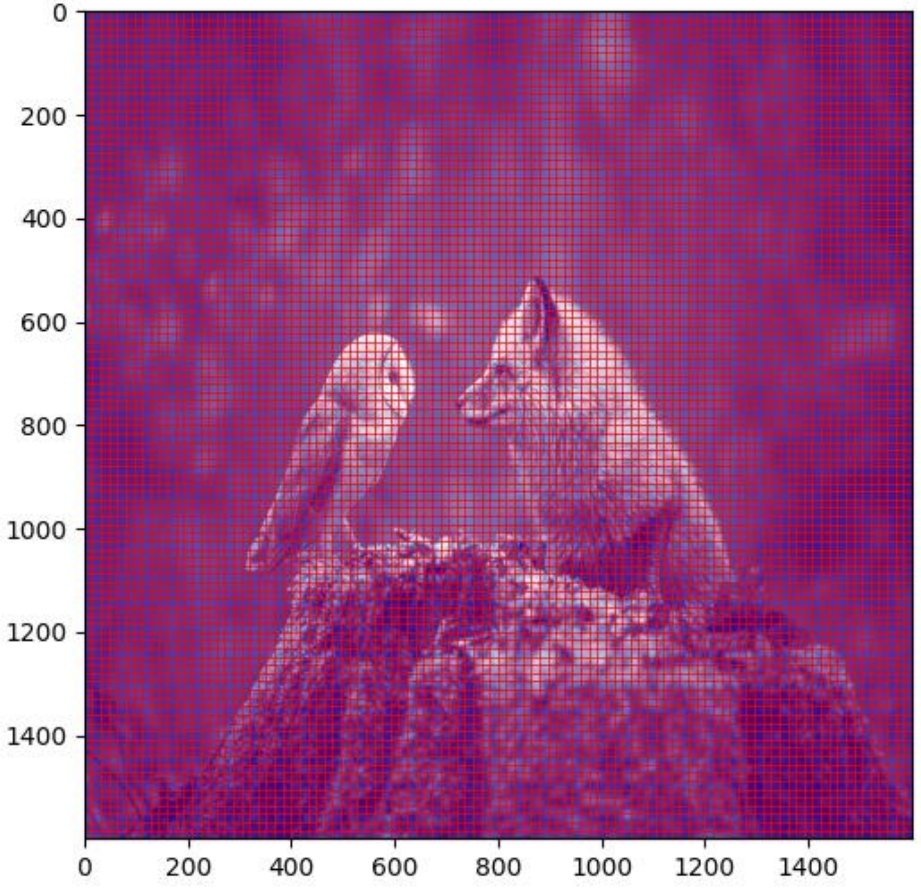
1600px



Problèmes avec ViT : Images haute résolution

1600px

1600px

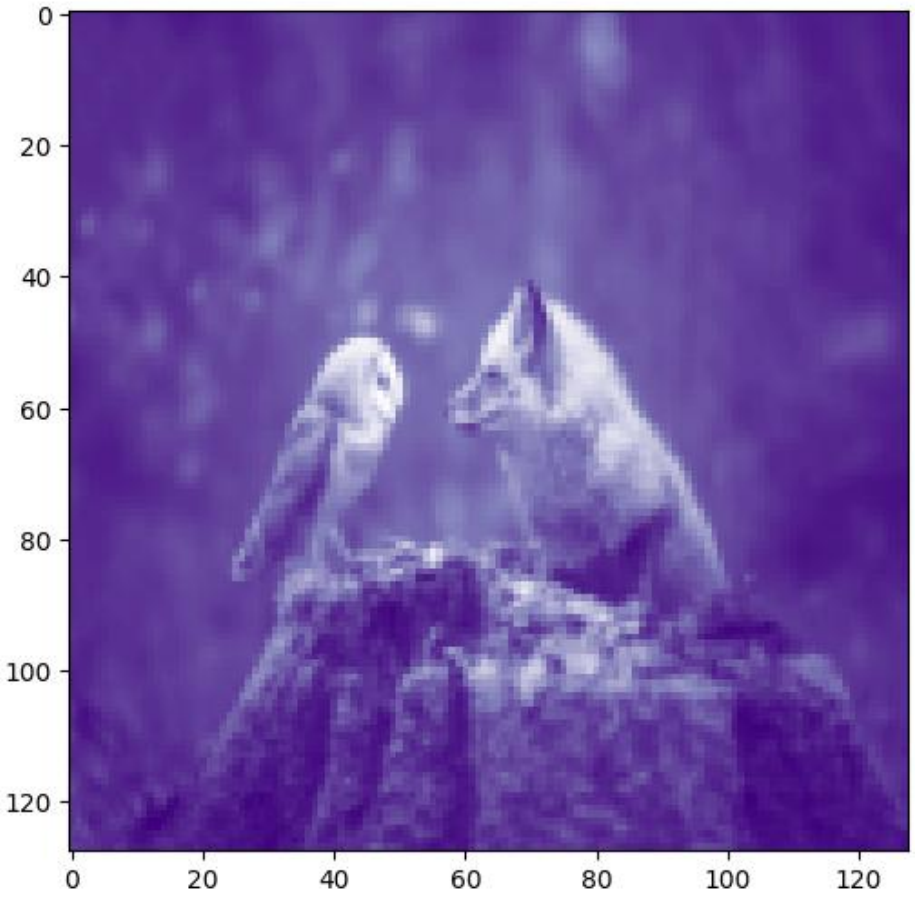


$\longrightarrow$ 10 000 tokens

Problèmes avec ViT : Tâches pixel-level

128px

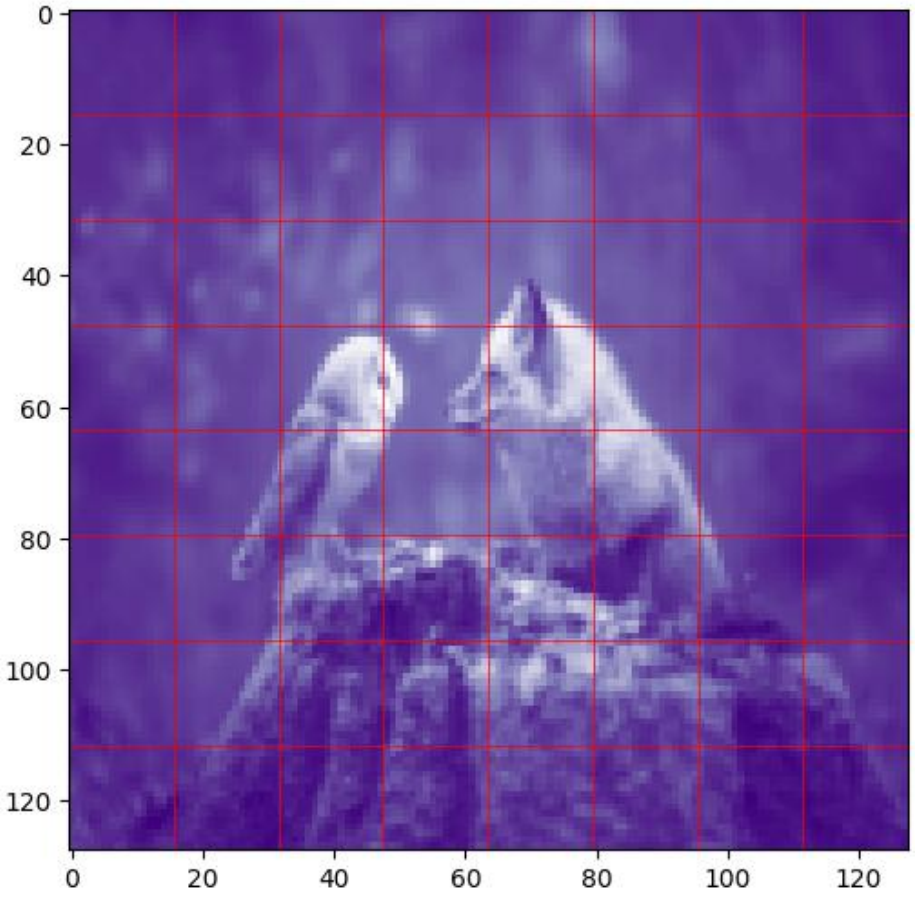
128px



Problèmes avec ViT : Tâches pixel-level

128px

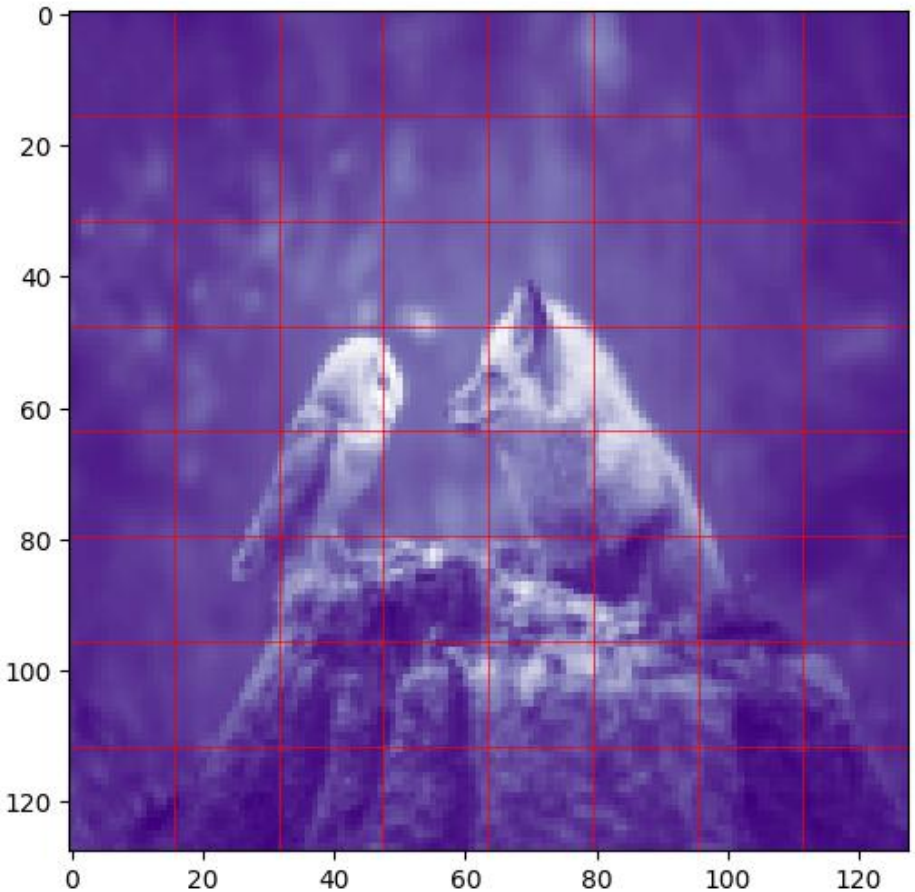
128px

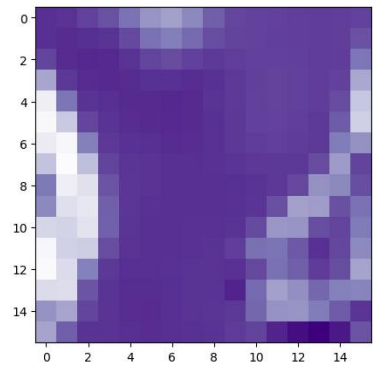


Problèmes avec ViT : Tâches pixel-level

128px

128px





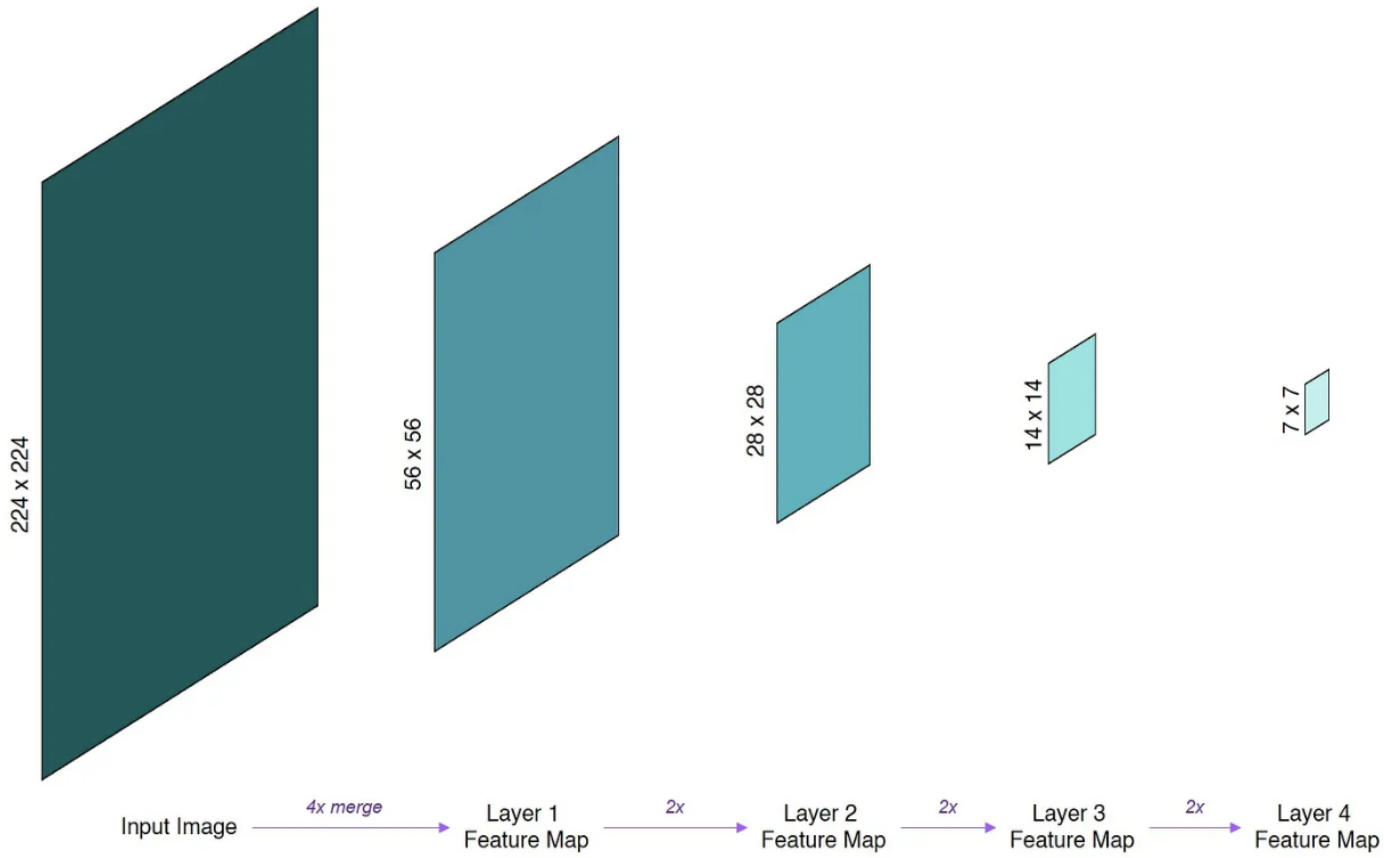
Problèmes avec ViT

En conclusion, les deux problèmes de ViT que souhaite résoudre l'architecture :

- Scalabilité en fonction de la résolution de l'image
- Attributs et détails de l'image plus fins

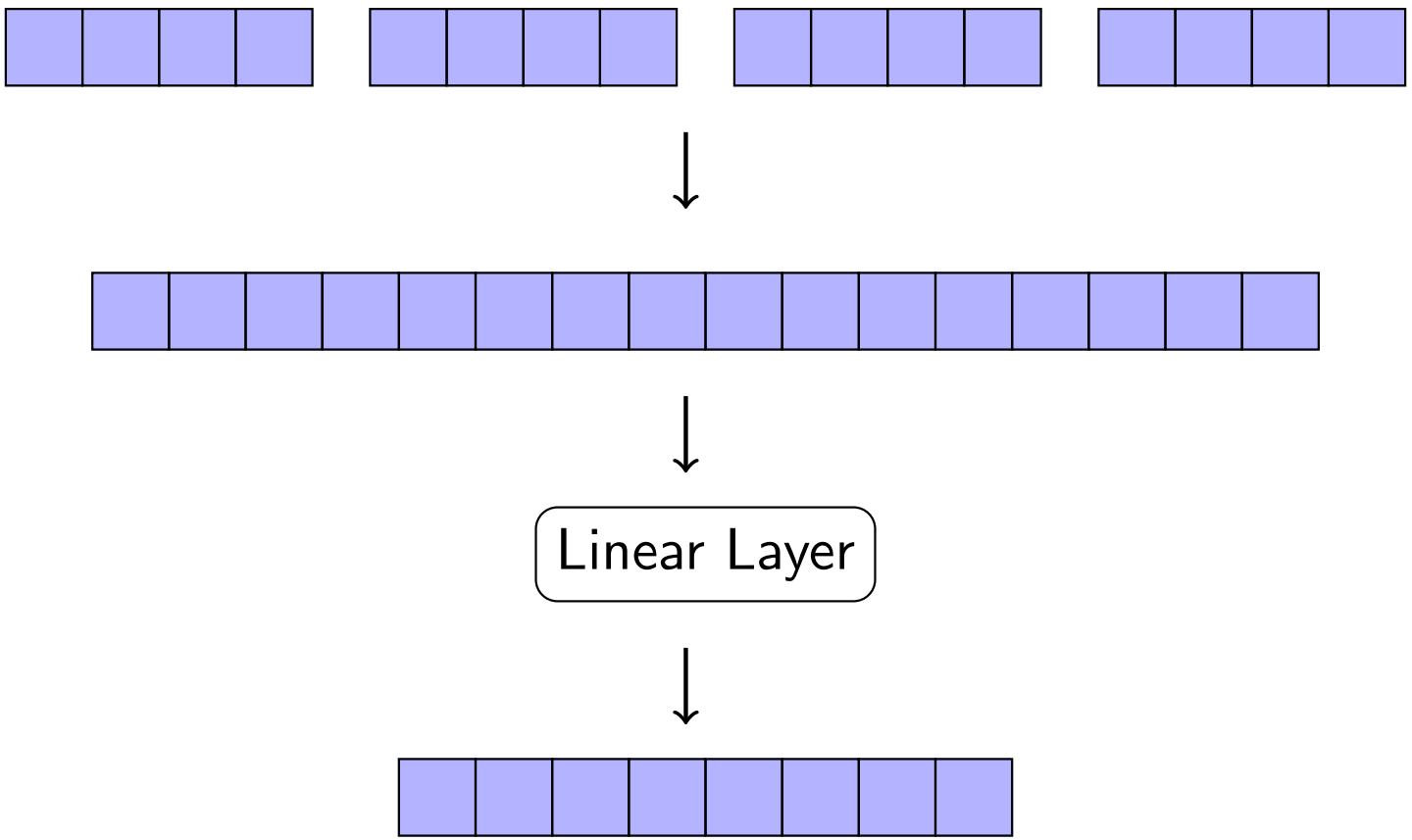
Swin Transformers : Hierarchical Feature Maps

Les hierarchical feature maps, à l'instar des architectures CNN (ResNet), permettent à Swin Transformer de gérer les problèmes où des prédictions plus fines sont nécessaires comme dans la sgmentation.



Swin Transformers : Patch Merging

Patch Merging = Concatenation des tokens voisins + Projection linéaire d'un facteur C

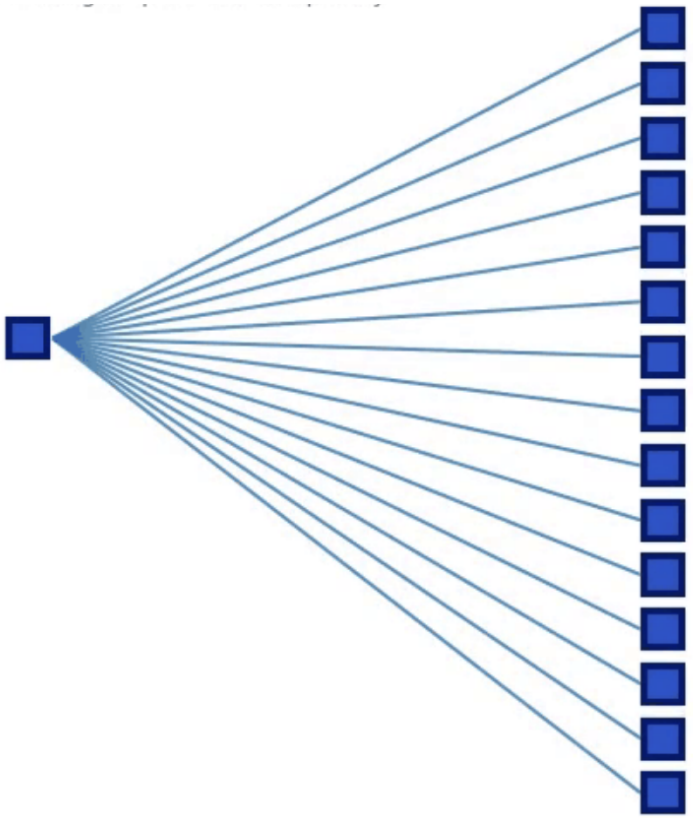


Swin Transformers : Limited attention

Pour des images haute résolution, le nombre de tokens augmente fortement donc la **complexité de l'attention explose $\mathbf{O(N^2)}$**.

Swin Transformers : Limited attention

Pour des images haute résolution, le nombre de tokens augmente fortement donc la **complexité de l'attention explose $\mathbf{O(N^2)}$**.

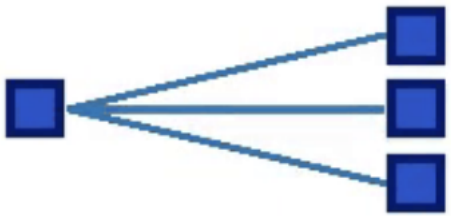


Swin Transformers : Limited attention

Pour palier ce problème, Swin utilise une version limitée de l'attention et ne calcule l'attention que pour les M voisins du token étudié. La **complexité devient alors linéaire O(M*N)** pour M assez petit.

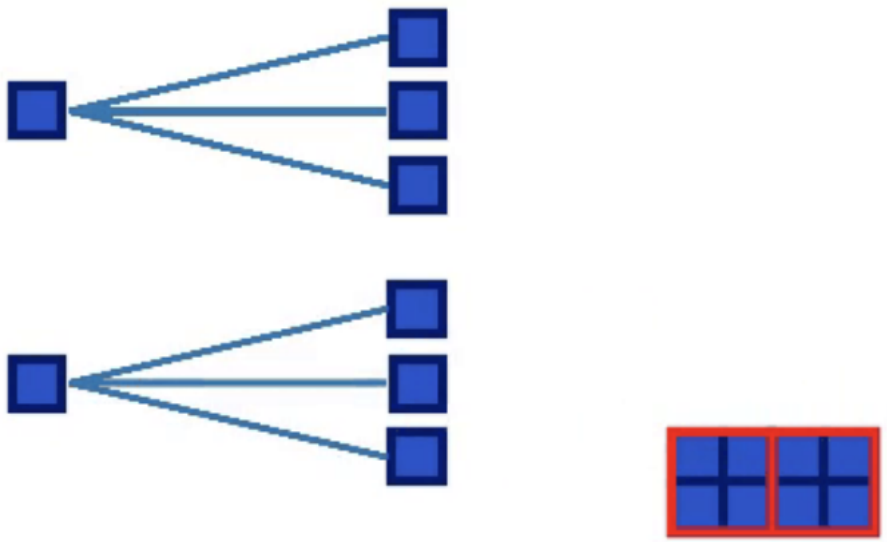
Swin Transformers : Limited attention

Pour palier ce problème, Swin utilise une version limitée de l'attention et ne calcule l'attention que pour les M voisins du token étudié. La **complexité devient alors linéaire O(M*N)** pour M assez petit.



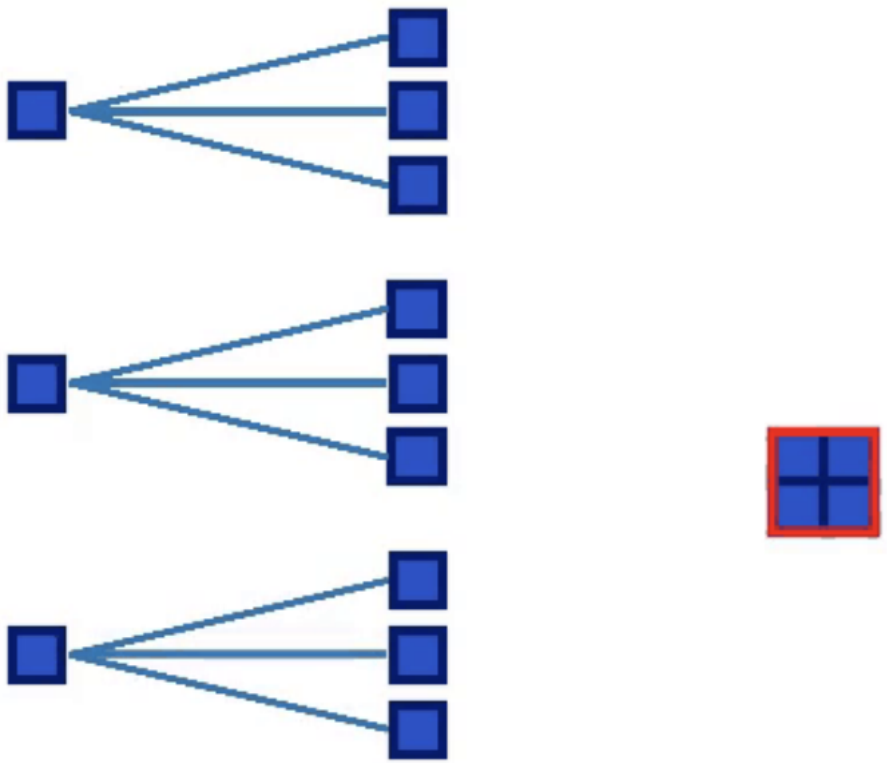
Swin Transformers : Limited attention

Pour palier ce problème, Swin utilise une version limitée de l'attention et ne calcule l'attention que pour les M voisins du token étudié. La **complexité devient alors linéaire O(M*N)** pour M assez petit.



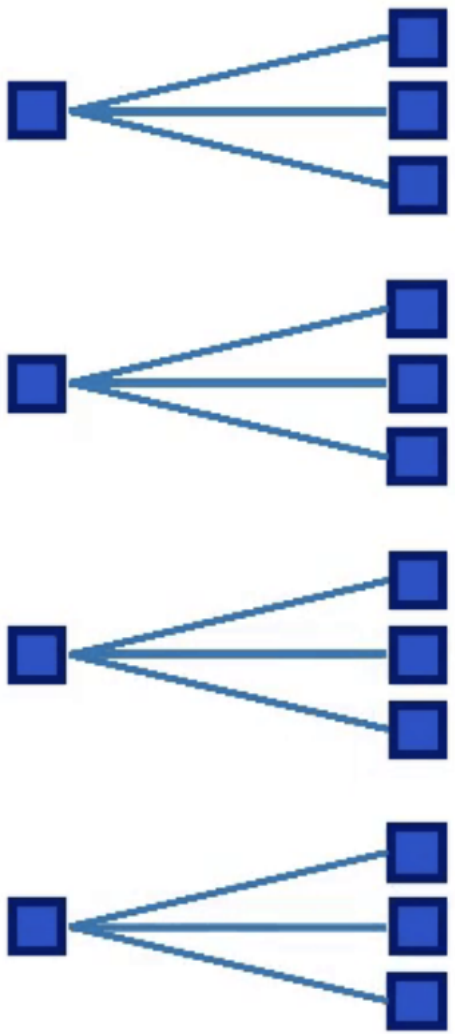
Swin Transformers : Limited attention

Pour palier ce problème, Swin utilise une version limitée de l'attention et ne calcule l'attention que pour les M voisins du token étudié. La **complexité devient alors linéaire O(M*N)** pour M assez petit.



Swin Transformers : Limited attention

Pour palier ce problème, Swin utilise une version limitée de l'attention et ne calcule l'attention que pour les M voisins du token étudié. La **complexité devient alors linéaire O(M*N)** pour M assez petit.



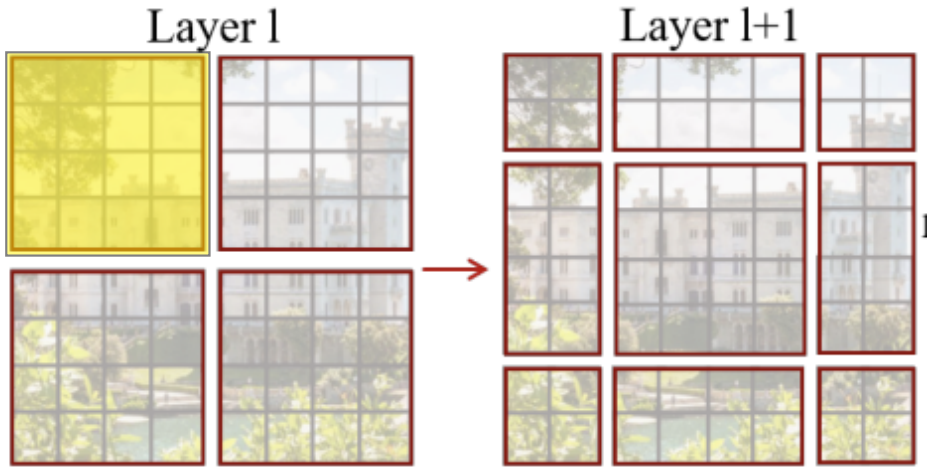
Swin Transformers : Shifted Windows Mechanism

Le problème de la limited attention est le fait de **restreindre l'attention à une fenêtre de voisins**.

Cela permet de réduire les complexités de calculs mais réduit aussi la capacité du modèle à capturer les relations entre deux objets de l'image qui ne seraient pas voisins.

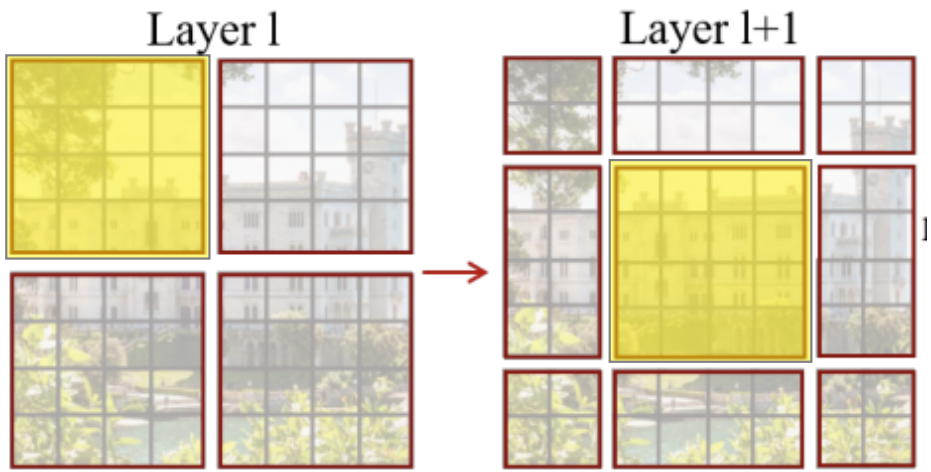
Swin Transformers : Shifted Windows Mechanism

Pour palier ce problème, Swin introduit la notion de fenêtre glissante entre les différents layer du modèle :



Swin Transformers : Shifted Windows Mechanism

Pour palier ce problème, Swin introduit la notion de fenêtre glissante entre les différents layer du modèle :



Swin Transformers : Shifted Windows Mechanism

Pour palier ce problème, Swin introduit la notion de fenêtre glissante entre les différents layer du modèle :

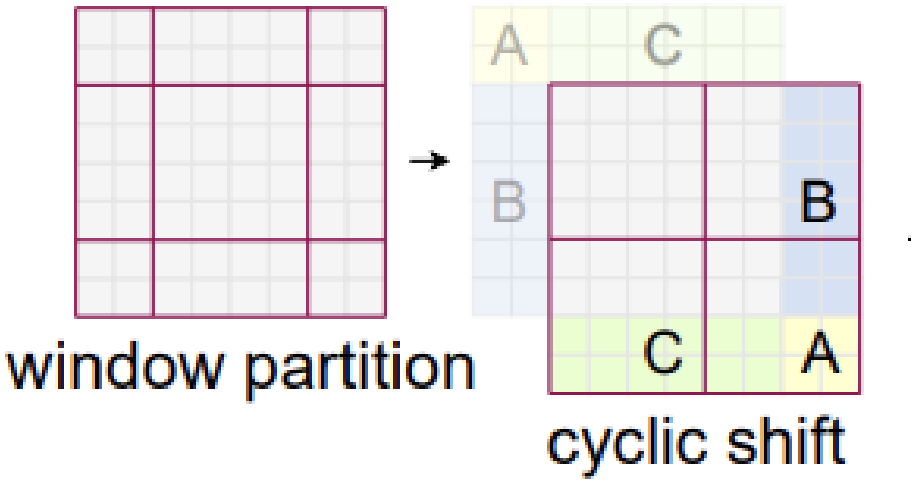


Table of Contents

Architecture hybrides Transformers + CNN

Origine

Les architectures hybrides combinent les forces des Transformers et des CNN pour la vision artificielle.

- **Transformers** : Capture des dépendances à longue distance et la globalité des images.
- **CNN** : Extraction des caractéristiques locales et spatiales précises.

Quelques modèles

- **TransUNet**
  - Cette architecture utilise un Transformer comme encodeur et un CNN comme décodeur.
  - Le CNN reconstruit l'image pixel par pixel.
  - Utilisé pour la segmentation et la génération d'images.
- **DeiT-CNN**
  - Combinaison d'un Transformer DeiT (Data-Efficient Image Transformer) et d'un CNN ResNet.
  - DeiT-CNN a atteint des performances de pointe sur la classification d'images.
- **Hybrid Transformer-CNN for Medical Image Segmentation**
  - Utilisé pour la segmentation d'images médicales
  - Cette architecture a obtenu des résultats prometteurs pour la segmentation de tumeurs cérébrales et de lésions cardiaques.

Avantages des architectures hybrides

- **Meilleures performances** : Les architectures hybrides peuvent souvent atteindre des performances supérieures aux Transformers ou CNN purs.
- **Complémentarité** : Les Transformers et CNN se complètent en capturant différents types d'informations dans les images.
- **Flexibilité** : Les architectures hybrides peuvent être adaptées à différentes tâches de vision artificielle en ajustant les composants Transformer et CNN.

Inconvénients des architectures hybrides

- **Complexité accrue** : Les architectures hybrides peuvent être plus complexes à concevoir et à mettre en œuvre que les Transformers ou CNN purs.
- **Coût de calcul** : Les architectures hybrides peuvent être plus coûteuses en termes de calcul que les Transformers ou CNN purs.

References

[1] : Vaswani, Ashish, et al. **"Attention is all you need."** Advances in neural information processing systems 30 (2017).

[2] : Dosovitskiy, Alexey, et al. **"An image is worth 16x16 words: Transformers for image recognition at scale."** arXiv preprint arXiv:2010.11929 (2020).

[3] : Jean Callis, Skylar. **Vision Transformers Explained**. 27/02/2024, https://towardsdatascience.com/vision-transformers-explained-a9d07147e4c8

[4] : Yuan, Li, et al. **"Tokens-to-token vit: Training vision transformers from scratch on imagenet."** Proceedings of the IEEE/CVF international conference on computer vision. 2021.

[5] : Jean Callis, Skylar. **Tokens-to-Token Vision Transformers, Explained**. 27/02/2024, https://towardsdatascience.com/tokens-to-token-vision-transformers-explained-2fa4e2002daa

[6] : Liu, Ze, et al. **"Swin transformer: Hierarchical vision transformer using shifted windows."** Proceedings of the IEEE/CVF international conference on computer vision. 2021.

[7] : Loy, James. **A Comprehensive Guide to Microsoft's Swin Transformer**. 11/09/2023, https://towardsdatascience.com/a-comprehensive-guide-to-swin-transformer-64965f89d14c

[8] : Sik-Ho, Tsang. **Review — CMT: Convolutional Neural Networks Meet Vision Transformers**. 5/09/2023, https://sh-tsang.medium.com/review-cmt-convolutional-neural-networks-meet-vision-transformers-1538c84a332a