Fonctionnement de ConvNeXt V2 — Étapes clés :

L'architecture **ConvNeXt V2** s'inspire des Vision Transformers(notre premier en terme modèle) mais tout en conservant une structure entièrement convolutive. L'objectif principal de ConvNeXt V2 est de combiner les avantages des ConvNets avec ceux des **autoencodeurs masqués (Masked Autoencoders - MAE) qui** masque aléatoirement une partie des patchs d'image, encode les patchs visibles à l'aide de convolutions profondes (sans Transformers), c'est une technique d'apprentissage auto-supervisé populaire dans les Transformers. Cette combinaison vise à améliorer les performances des ConvNets sur diverses tâches de vision par ordinateur, telles que la classification d'images, la détection d'objets et la segmentation sémantique.

Voici les étapes clés de son pipeline de traitement :

1. Patchification (Stem Layer)

- **Objectif** : Convertir l'image d'entrée en patchs.
- **Méthode** : Une **convolution 4×4 avec stride 4** est utilisée au lieu d'un découpage manuel des patchs comme dans ViT.
- **Résultat** : une **feature map** de plus petite résolution (**H/4 × W/4**) avec `C` canaux. Cela sert de point de départ pour l'extraction hiérarchique.

2. Apprentissage auto-supervisé avec FCMAE

- **Masquage aléatoire** des patchs d'entrée.
- **Encodeur convolutif profond** (pas de Transformer) traite les patchs visibles.
- **Décodeur léger** reconstruit les patchs manquants.
- **But** : Forcer le réseau à apprendre les relations spatiales globales à partir d'observations partielles (comme MAE mais avec uniquement des convolutions).

3. Extraction hiérarchique via 4 Stages

Chaque stage contient plusieurs **ConvNeXt V2 Blocks** :

➤ a. ConvNeXt V2 Block :

- `Depthwise Conv 7×7` → large réceptive field.
- `LayerNorm (NHWC)` → meilleure stabilité.
- `GELU` → non-linéarité douce.
- `Pointwise Conv 1×1` (MLP-like).
- `GRN` (Global Response Normalization) → régularise la réponse globale des canaux.
- `Skip connection` → facilite l'optimisation.

➤ b. Downsampling entre les stages :

- `Conv 2×2, stride 2`
- **Réduction de résolution** (H, W divisés par 2).

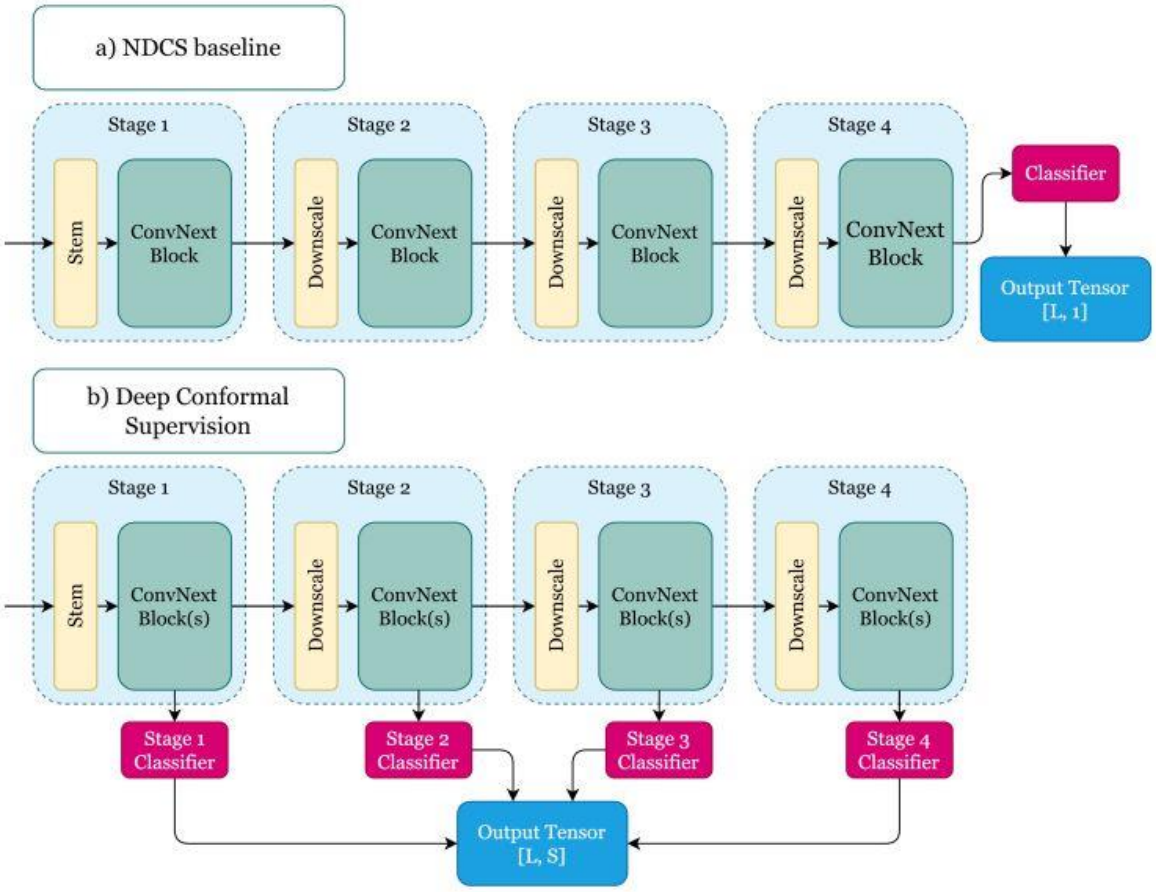
- **Augmentation des canaux** : C → 2C → 4C → 8C.
- Cela forme une **pyramide de caractéristiques** à différentes échelles.

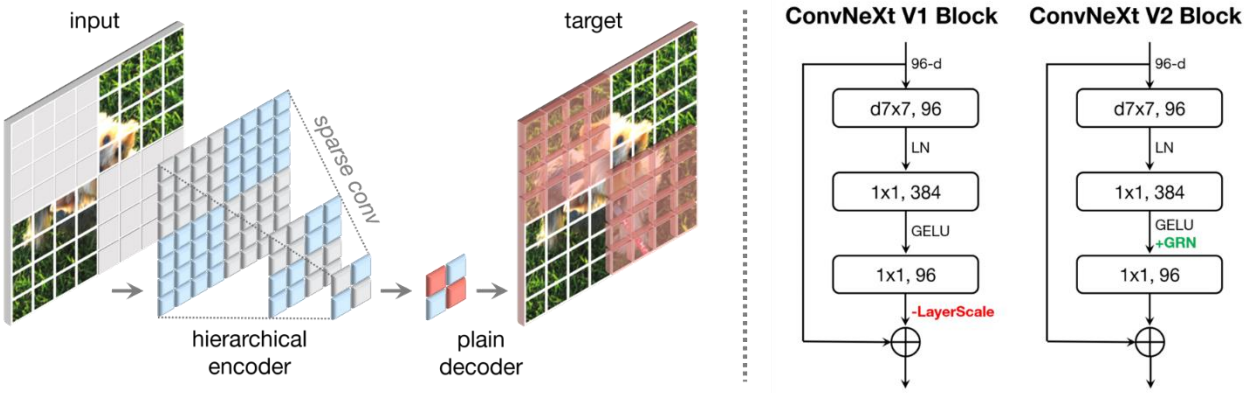
4. Global Feature Aggregation

- À la fin du dernier stage : la feature map a une taille **H/32 × W/32 × 8C**.
- On applique un **Global Average Pooling** pour résumer spatialement → vecteur **1D** de taille 8C.

5. Tête de classification

- Une **LayerNorm** est appliquée.
- Suivie d'une **Fully Connected Layer**.
- Le modèle sort un **vecteur de logits**, un score par classe cible.





Résumé Visuel Rapide :

Image (H × W × 3)

↓

Patchification (Conv 4×4, stride 4)

↓

Stage 1 : N1 × ConvNeXt V2 Blocks

↓ Downsampling (Conv 2×2)

Stage 2 : N2 × Blocks

↓ Downsampling

Stage 3 : N3 × Blocks

↓ Downsampling

Stage 4 : N4 × Blocks

↓

Global Average Pooling

↓

LayerNorm + Linear

↓

Logits de classification

Source :

- viblo.asia
- github convNext v2