Object Detection

Transformer-based

- **An Image is Worth 16x16 Words Transformers for Image Recognition at Scale [ICLR 2021]** Alexey Dosovitskiy (arXiv) (pdf)

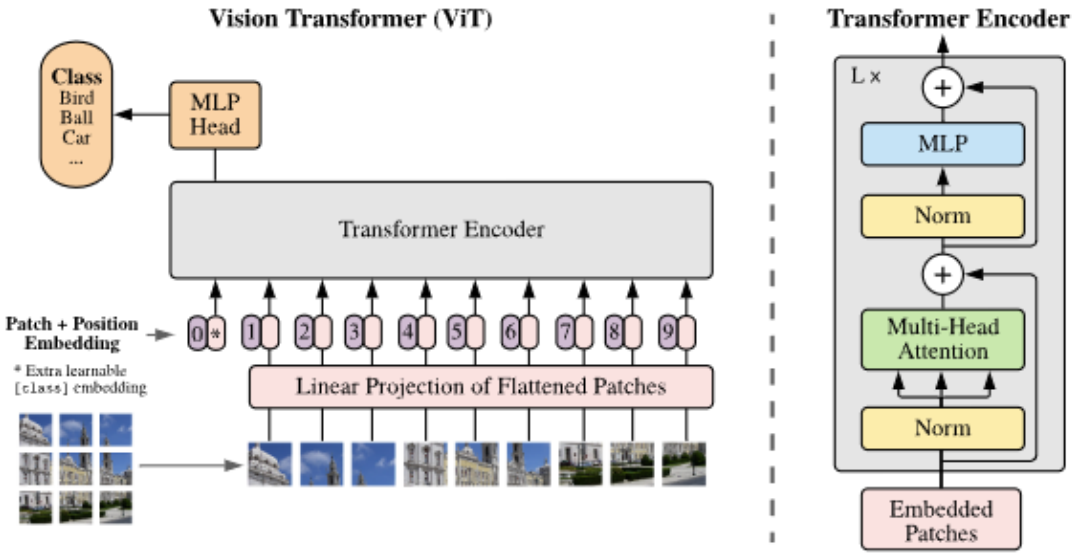

Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

- Tokenization:
    - Input image x has dimension $H\times W\times C$, suchs as $224 \times 224 \times 3$.
    - Divide x into N image patches, each patch is a 2D matrix with image resolution of (P, P) (such as $(16, 16)$).
    - Thus, there are $N= (HW)/P^2$ patches. $N=196=(224 \times 224)/16^2$.
    - Therefore, there are N tokens are transformer inputs, and each token is a $16\times 16 \times 3=768$ vector.
    - Each token is **added** with position embeddings to retain positional information.
        - Position embedding: a learnable 1D vector with the length of 768, i.e., identical to the length of a token.
    - Add a learnable embedding to the sequence of embedded patches (1×768), whose state at the output of the Transformer encoder serves as the image representation y.
    - Overall size of the input: image patches 196×768 + class embedding 1×768 = 197×768.

- **Masked Autoencoders Are Scalable Vision Learners [arXiv 2021]** *Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, Ross Girshick* (arXiv) (pdf) (Citation: 2861)

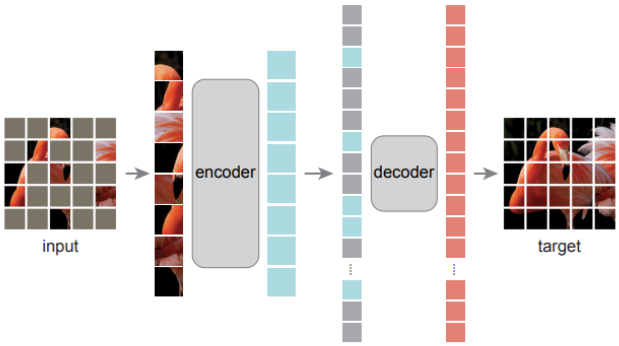

Figure 1. **Our MAE architecture**. During pre-training, a large random subset of image patches (*e.g.*, 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

- MAE contains an encoder and a decoder. Encoder maps the observed signals to a latent representation, and a decoder that reconstructs the original signal from the latent representation.
    - Part 1 -> Masking (divide an image into regular non-overlapping patches, then randomly sample a subset of patches and mask the remaining ones. High masking ratio largely eliminates **redundancy**.)
    - Part 2 -> MAE encoder is a ViT, but applied only on visible, unmasked patches (MAE follows the standard ViT operation, divide an image into patches, added with positional embedding, and process through Transformer layers).
    - Part 3 -> MAE decoder has a series of Transformer blocks. Inputs to MAE decoder include 1. encoded visible patches + 2. mask tokens (a shared, learned vector that indicates the presence of a missing patch to be predicted).
    - Part 4 -> Reconstruction Target. Each element in the decoder's output is a vector of pixel values representing a patch. The loss function computes the MSE between the reconstructed and original images in the pixel space.
- Simple Implementation
    - Step 1 -> Generate a token for every input patch (with positional embedding)
    - Step 2 -> Encoding. Randomly shuffle the list of tokens and remove the last portion of the list, based on the masking ratio. (This process produces a small subset of tokens for the encoder)
    - Step 3 -> Decoding. Append a list of mask tokens to the list of encoded patches, and unshuffle this full list to align all tokens with their targets. The decoder is applied to the full list.

YOLO Series

- **CSPNet A New Backbone that can Enhance Learning Capability of CNN [`arXiv 2019`]** *Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh* (arXiv)(pdf)
- **YOLOv3 An Incremental Improvement [`arXiv 2018`]** Joseph Redmon (arXiv) (pdf)
- **YOLOv4 Optimal Speed and Accuracy of Object Detection [`arXiv 2020`]** Alexey Bochkovskiy (arXiv) (pdf)