

Introduction to Vision Transformers

Konstantin A. Maslov
k.a.maslov@utwente.nl

University of Twente
Faculty of Geo-Information Science and Earth Observation

31 Oct 2022



Outline

Essential basics

- Layer normalization

- Multi-head self-attention

ViT

Other architectures

- DeiT

- SETR

- Segmenter

- Swin transformer

- SegFormer

- MLP-Mixer

Common practices

Summary & discussion

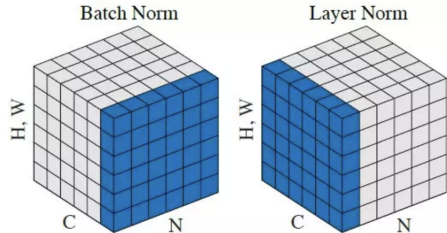


Layer normalization

- ▶ Somehow similar to batch normalization, but estimates normalization statistics from **all** 'features' for **one** sample in a batch
- ▶ Thus, there is no dependencies between different samples in a batch
- ▶ It works well with RNNs and is (always) used in transformers

Batch normalization:

- ▶ All samples in a batch
- ▶ All 'pixels'
- ▶ One 'feature'



Source: <https://paperswithcode.com/method/layer-normalization>

Layer normalization:

- ▶ One sample in a batch
- ▶ All 'pixels'
- ▶ All 'features'



Layer normalization

- ▶ Layer statistics are calculated as

$$\mu = \frac{1}{H} \sum_i^H x_i, \quad (1)$$

$$\sigma = \sqrt{\frac{1}{H} \sum_i^H (x_i - \mu)^2}, \quad (2)$$

where μ and σ are the mean and the standard deviation, H is the number of hidden units, and \mathbf{x} is the input tensor

- ▶ Layer normalization is then defined as

$$LN(\mathbf{x}) = \frac{\bar{\gamma}}{\sigma} \cdot (\mathbf{x} - \mu) + \bar{\beta}, \quad (3)$$

where $\bar{\gamma}$ and $\bar{\beta}$ are learnable parameters. Note that $\bar{\gamma}$ and $\bar{\beta}$ are vectors (not scalars!)



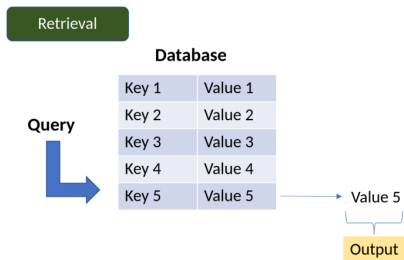
Attention

- ▶ Scaled dot-product attention can be defined as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right) \mathbf{V}, \quad (4)$$

where \mathbf{Q} , \mathbf{K} and \mathbf{V} are the query, key and value, respectively, and d_k is the number of the key 'features'

- ▶ Attention can be understood as searching for a value (\mathbf{V}) in a database based on how the search query (\mathbf{Q}) is similar to a table key (\mathbf{K})



The product \mathbf{QK}^T can be seen as a cross-correlation between queries and values (a similarity measure), $\text{Softmax}(\dots)$ further 'chooses' the row with the highest similarity

Source: <https://towardsdatascience.com/>



Self-attention

- Self-attention implies that \mathbf{Q} , \mathbf{K} and \mathbf{V} are calculated from the same input \mathbf{X} and learnt

$$\text{Self-Attention}(\mathbf{X}) = \text{Softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}} \right) \mathbf{V}, \quad (5)$$

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}, \quad \mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}, \quad \mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$$

Let's investigate in detail how tensor shapes are changing within self-attention:

Tensor	Shape
\mathbf{X}	(n_tokens, d_x)
$\mathbf{W}^{(q)}, \mathbf{W}^{(k)}, \mathbf{W}^{(v)}$	(d_x, d)
$\mathbf{Q}, \mathbf{K}, \mathbf{V}$	(n_tokens, d)
$\mathbf{Q}\mathbf{K}^T$	(n_tokens, n_tokens)
$\text{Self-Attention}(\mathbf{X})$	(n_tokens, d)



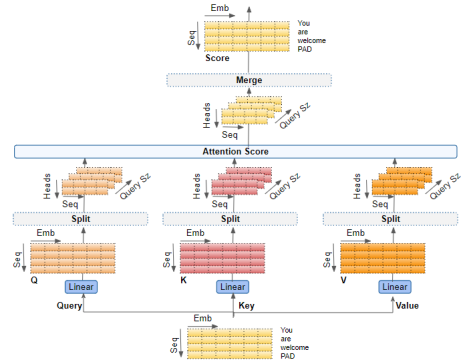
Multi-head self-attention

- In multi-head self-attention, we split **Q**, **K** and **V** into sections, process them simultaneously and then concatenate and linearly transform

$$MHSA(\mathbf{X}) = [\text{head}_1, \text{head}_2, \dots, \text{head}_h] \mathbf{W}_0,$$

$$\text{head}_i =$$

$$\text{Attention}(\mathbf{XW}_i^{(q)}, \mathbf{XW}_i^{(v)}, \mathbf{XW}_i^{(v)})$$

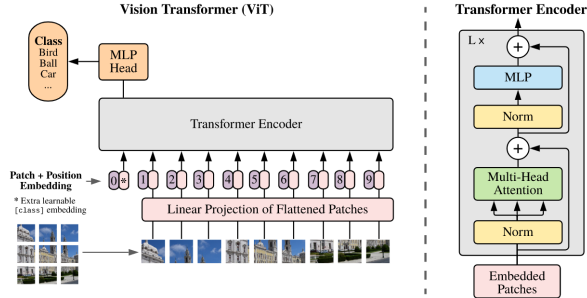


Source: <https://towardsdatascience.com/>

transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1f

- It ensures learning richer data representations





- ▶ Transformer design from NLP with minimal changes
- ▶ Achieves performance close to the state-of-the-art (CNNs) in classification tasks
- ▶ Can learn long-range relations in the very first layers due to the multi-head self-attention
- ▶ Requires pre-training on huge datasets to achieve good performance

- ▶ The authors tried to use a CNN for patch embedding instead of simple linear projection, it did not show significant differences
- ▶ The authors tried to remove the class token and feed the classification head with globally pooled features, it did not show a significant difference (but changed the requirements for the optimal learning rate)
- ▶ In addition to 1-D positional embedding, the authors considered no embedding, 2-D embedding and relative positional embedding, no embedding showed a performance drop, while for the rest there is no significant difference



ViT DEMO

[https://github.com/
konstantin-a-maslov/
transformers-seminar](https://github.com/konstantin-a-maslov/transformers-seminar)

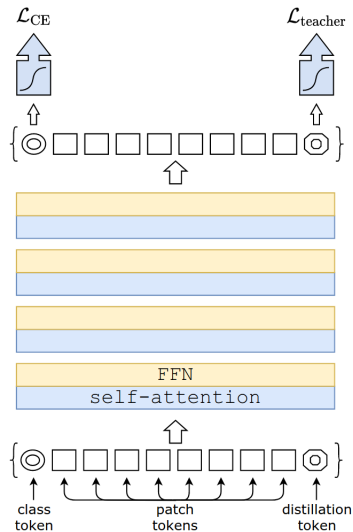


```
def ViT(
    input_shape, n_classes, patch_size=16, embedding_size=768, mlp_size=3072,
    n_blocks=12, n_heads=12, dropout=0.1, name="ViT", **kwargs
):
    inputs = tf.keras.layers.Input(input_shape)

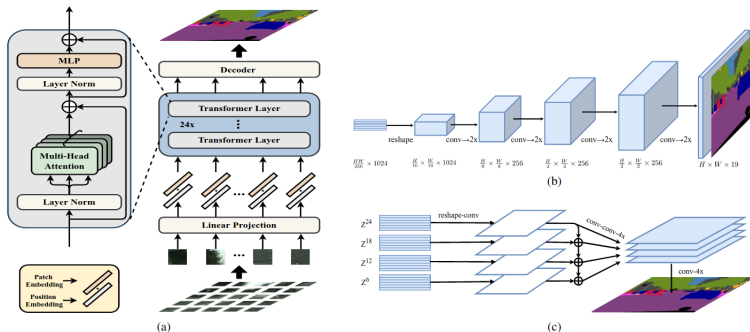
    patches = PatchExtraction(patch_size)(inputs)
    patches = PatchEmbedding(embedding_size)(patches)
    patches = AddClassToken()(patches)
    patches = AddPositionalEmbedding()(patches)
    for _ in range(n_blocks):
        patches = TransformerBlock(embedding_size, mlp_size, n_heads, dropout)(patches)
    class_token = ExtractClassToken()(patches)
    outputs = MLP(mlp_size, n_classes, dropout)(class_token)

    model = tf.keras.models.Model(inputs=inputs, outputs=outputs, name=name, **kwargs)
    return model
```





- ▶ ViT trained with distillation
- ▶ The teacher model is a CNN
- ▶ The authors claim that it reduces the amount of data required to train a transformer

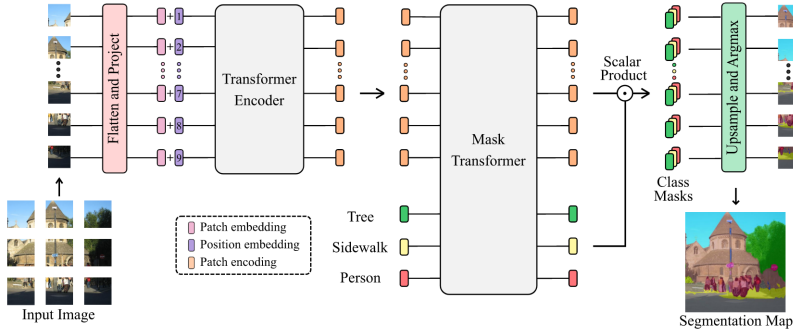


- ▶ ViT with a typical upsampling decoder as in FCNs
- ▶ The model has shown state-of-the-art performance in some tasks

Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P. H. S., & Zhang, L. (2020). Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 6877–6886. <https://doi.org/10.48550/arxiv.2012.15840>



Segmenter



- ▶ Transformer-based decoder
- ▶ Introduced class embeddings
- ▶ The authors emphasized that transformer-based models are not so good at generating sharp object boundaries
- ▶ Does not seem to be a popular choice nowadays

Segmenter

- ▶ “... the performance is better for large models and small patch sizes.”
- ▶ “We observe that for a patch size of 32, the model learns a globally meaningful segmentation but produces poor boundaries...”
- ▶ “However DeepLab performs similarly to Seg-B/16 on small and medium instances while having a similar number of parameters.”

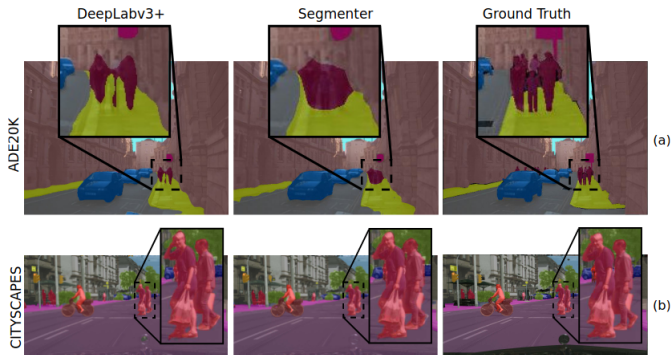
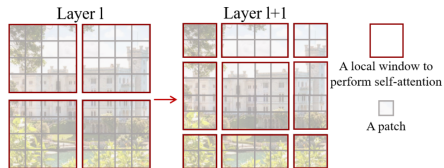
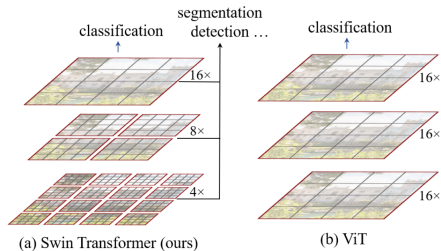


Figure 11: Comparison of Seg-L-Mask/16 with DeepLabV3+ ResNeSt-101 for images with near-by persons. We can observe that DeepLabV3+ localizes boundaries better.

Swin transformer

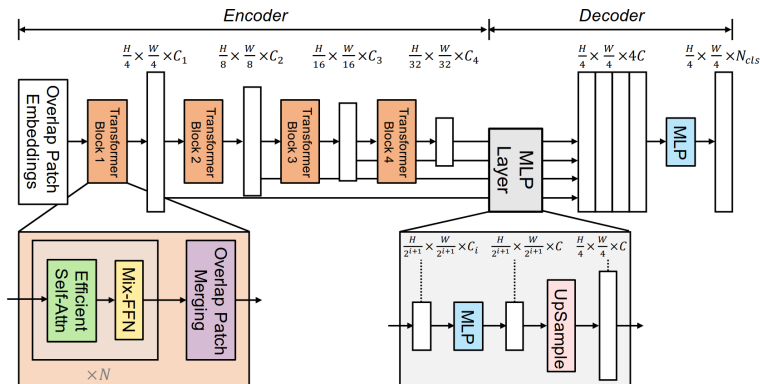


- ▶ Has linear time complexity due to the hierarchical design
- ▶ Introduced shifting windows

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. Proceedings of the IEEE International Conference on Computer Vision, 9992–10002. <https://doi.org/10.48550/arxiv.2103.14030>

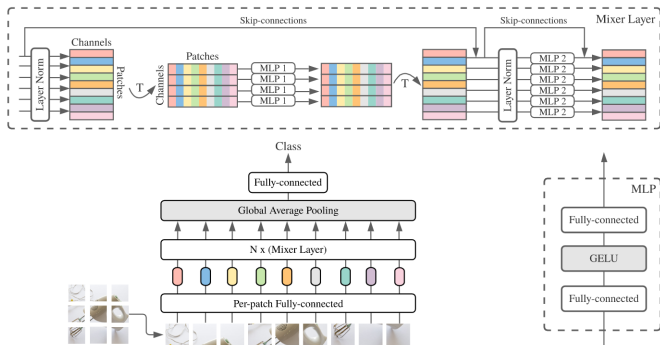


SegFormer



- ▶ The authors focused on an efficient design
- ▶ Seems to be the state-of-the-art among transformers for semantic image segmentation (general tasks) nowadays
- ▶ Has no positional embedding

MLP-Mixer



- Not a transformer!
- Replaces multi-head self-attentions with simple MLPs
- The authors have shown that it still possible to obtain good results with this design

Common practices

- ▶ Pre-training on huge (hundreds of millions of images) datasets
- ▶ Always employing very deep transformers (with tens of millions of parameters)
- ▶ Training a model on smaller images, fine-tuning on images with higher resolution
- ▶ Not using dropout, but using stochastic depth
- ▶ Training with AdamW, fine-tuning with SGD

Loshchilov, I., & Hutter, F. (2017). Decoupled Weight Decay Regularization. 7th International Conference on Learning Representations, ICLR 2019.
<https://doi.org/10.48550/arxiv.1711.05101>



Summary & discussion

- ▶ Transformers require a lot of data to train
 - ▶ Which can be not the case for remote sensing application
 - ▶ Pre-training is complicated due to the absence of huge datasets for multispectral data
 - ▶ Using the weights from more common datasets (ImageNet, JFT, ...) is still an option though, but it requires studies on how to better 'generalise' them for non-RGB images
- ▶ Seems like transformers are bad at restoring sharp boundaries in segmentation maps
 - ▶ Can be crucial as the spatial resolution of the satellite imagery we use is very different
 - ▶ Smaller patch sizes or overlapping patches improve the situation (if one has enough memory...)
 - ▶ Perhaps, there is a space to explore hybrid CNN-transformer models and shallow transformers
- ▶ There are works that emphasize that CNNs still outperform transformers if one focuses on the training procedure

