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

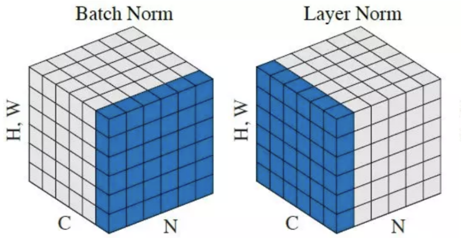
Summary & discussion

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

$$\mu = \frac{1}{H}\sum_{i}^{H} x_i, \qquad (1)$$

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

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

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

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

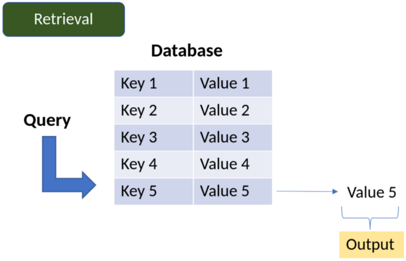
Attention

- Scaled dot-product attention can be defined as

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

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 ($\mathbf{K}$)



Source: https://towardsdatascience.com/all-you-need-to-know-about-attention-and-transformers-in-depth-understanding-part-1-552f0b41d021

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

Self-attention

- Self-attention implies that **Q**, **K** and **V** are calculated from the same input **X** and learnt

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

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

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

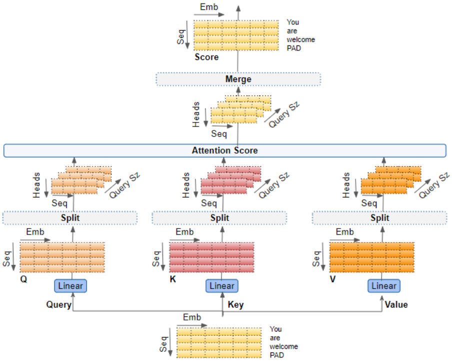
Tensor	Shape
$\mathbf{X}$	(n_tokens, d_x)
$\mathbf{W}^{(q)}$, $\mathbf{W}^{(k)}$, $\mathbf{W}^{(v)}$	(d_x, d)
$\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$	(n_tokens, d)
$\mathbf{Q}\mathbf{K}^T$	(n_tokens, n_tokens)
$\mathit{Self\text{-}Attention}(\mathbf{X})$	(n_tokens, d)

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

$$MHSA(\mathbf{X}) = [head_1, head_2, ..., head_h]\mathbf{W}_0,$$

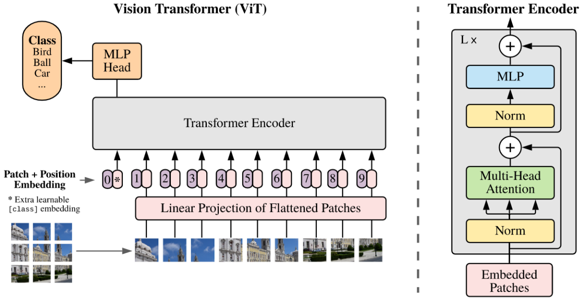
$$head_i = Attention\left(\mathbf{XW}_i^{(q)}, \mathbf{XW}_i^{(v)}, \mathbf{XW}_i^{(v)}\right)$$



Source: https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1f

- It ensures learning richer data representations

ViT



- 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

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. https://doi.org/10.48550/arxiv.2010.11929

ViT

- 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

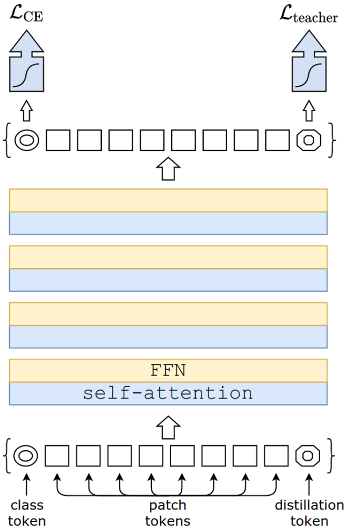
ViT DEMO

https://github.com/konstantin-a-maslov/transformers-seminar

```python
class ViT(tf.keras.models.Model):
    def __init__(
        self,
        n_classes,
        patch_size=16,
        embedding_size=768,
        mlp_size=3072,
        n_blocks=12,
        n_heads=12,
        dropout=0.1,
        name="ViT",
        **kwargs
    ):
        super(ViT, self).__init__(name=name, **kwargs)
        self.patch_extraction = PatchExtraction(patch_size)
        self.patch_embedding = PatchEmbedding(embedding_size)
        self.add_class_token = AddClassToken()
        self.add_positional_embedding = AddPositionalEmbedding()
        self.transformer_blocks = [
            TransformerBlock(embedding_size, mlp_size, n_heads, dropout)
            for _ in range(n_blocks)
        ]
        self.extract_class_token = ExtractClassToken()
        self.mlp = MLP(mlp_size, n_classes, dropout)

    def call(self, inputs):
        patches = self.patch_extraction(inputs)
        patches = self.patch_embedding(patches)
        patches = self.add_class_token(patches)
        patches = self.add_positional_embedding(patches)
        for block in self.transformer_blocks:
            patches = block(patches)
        class_token = self.extract_class_token(patches)
        outputs = self.mlp(class_token)
        return outputs
```

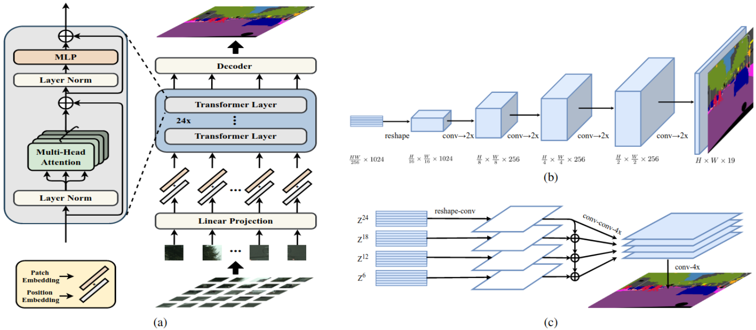
DeiT



- 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

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H., & Ai, F. (2020). Training data-efficient image transformers & distillation through attention. https://doi.org/10.48550/arxiv.2012.12877

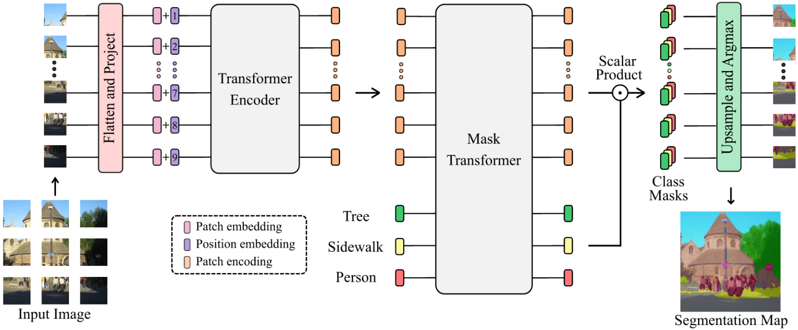
SETR



- 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

Strudel, R., Garcia, R., Laptev, I., & Schmid, C. (2021). Segmenter: Transformer for Semantic Segmentation. Proceedings of the IEEE International Conference on Computer Vision, 7242–7252. https://doi.org/10.48550/arxiv.2105.05633

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."

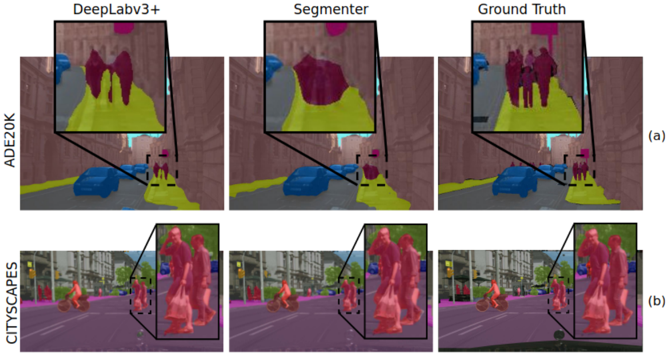
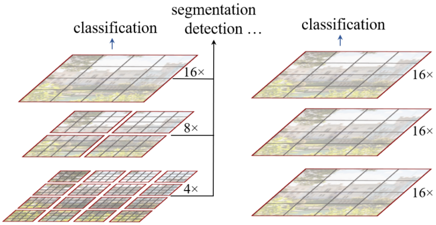


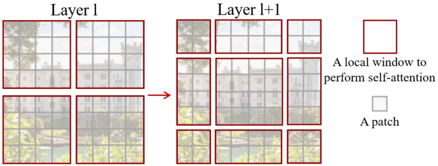
Figure 11: Comparison of Seg-L-Mask/16 with DeepLabV3+ ResNeSt-101 for images with near-by persons. We can observe that DeepLabV3+ localizes boundaries better.

Strudel, R., Garcia, R., Laptev, I., & Schmid, C. (2021). Segmenter: Transformer for Semantic Segmentation. Proceedings of the IEEE International Conference on Computer Vision, 7242–7252. https://doi.org/10.48550/arxiv.2105.05633

Swin transformer



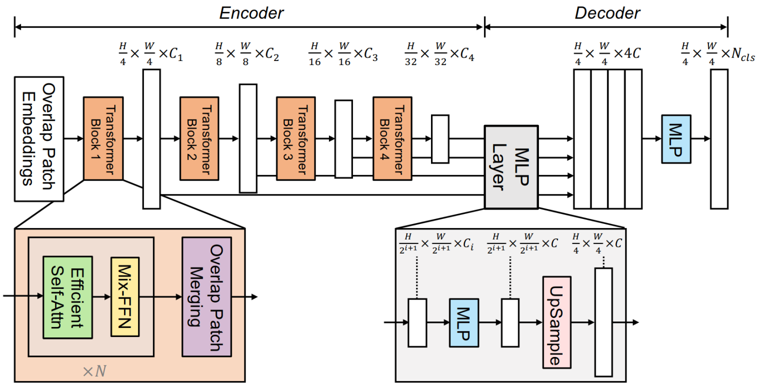
(a) Swin Transformer (ours)
(b) ViT



- 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

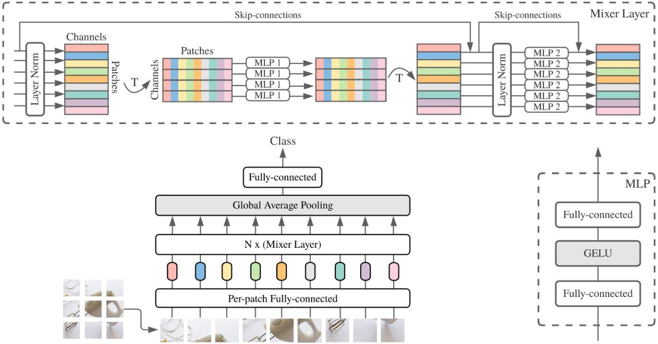
SegFormer



- 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

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., & Luo, P. (2021). SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. Advances in Neural Information Processing Systems, 15, 12077–12090. https://doi.org/10.48550/arxiv.2105.15203

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

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., & Dosovitskiy, A. (2021). MLP-Mixer: An all-MLP Architecture for Vision. Advances in Neural Information Processing Systems, 29, 24261–24272. https://doi.org/10.48550/arxiv.2105.01601

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

Wightman, R., Touvron, H., Jégou, H., & Ai, F. (2021). ResNet strikes back: An improved training procedure in timm. https://doi.org/10.48550/arxiv.2110.00476