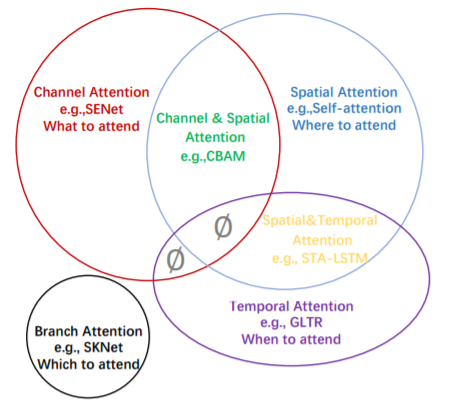
Self-Attention Mechanism

**Vision Transformer is currently SOTA in CV**

1. Self-attention mechanism has a wide array of usage in the field of computer vision and NLP. This widely adapted methodology describes a dynamic selection process of features or regions of an input that add more value to the output and disregard the irrelevant regions.
2. Vision Transformer introduced by Google is a Deep Learning Model that employs the self-attention methodology, differentially weighting each region of the input data to grasp the most significant part.
3. [Good article](https://towardsdatascience.com/self-attention-in-computer-vision-2782727021f6) with image related tasks.
4. [Article](https://viso.ai/deep-learning/vision-transformer-vit/) discussing vision transformers
5. List of papers on self-attention collected in [github](https://github.com/MenghaoGuo/Awesome-Vision-Attentions).
6. According to [this paper](https://arxiv.org/pdf/2111.07624.pdf), there are 4 categories.



1. **Attention is All you need** - [paper](https://arxiv.org/pdf/1706.03762v5.pdf), [code](https://github.com/jadore801120/attention-is-all-you-need-pytorch)

“Our model achieves 28.4 BLEU on the WMT 2014 Englishto-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. “

**a**. Previously,

In encoder-decoder models, Input sequence of X produces a continuous layer Z. The decoder uses Z to output a final sequence Y.

**b**. “An **attention** function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key”

**In other words**, we have a key vector and a query vector (similar to 3x3 filters in cnn). we find dot product of the two to get the ‘values’ vector. This resultant vector is passed through softmax function and their weighted sum gives the output vector.

**c.** two types: **additive attention and dot-product (multiplicative) attention function**. Dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

**d. Multi-Head Attention** employs scaled dot product attention function in bulk to many dimensions at once in the form of a matrix. This is then concatenated to get a single linear output.

**e. Position-wise Feed-Forward Networks** - r contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a **ReLU** activation in between.

**f.** multi-Head attention and point wise feed forward network together forms a single module. These modules stacked sequentially with a softmax layer in the end forms the self-attention model.

1. Soft vs Hard attention:
   1. attention reweighs certain features of the network according to some externally or internally (self-attention) supplied weights. Hereby, soft attention allows these weights to be continuous while hard attention requires them to be binary, i.e., 0 or 1.
2. Stand-alone self-attention
   1. Idea of using self-attention as a replacement of convolution blocks in cnn and not as an added feature
   2. Convolutions better capture low-level features while stand-alone attention layers may better integrate global information
3. **Vision Transformer (**[**code**](https://github.com/google-research/vision_transformer)**)**
   1. The ViT models were pre-trained on the ImageNet and ImageNet-21k datasets.
   2. Transformers in machine learning are composed of multiple self-attention layers
   3. ViT shows less bias and more reliance on regularization and augmentation for small datasets
   4. The ViT is a visual model based on the architecture of a transformer originally designed for text-based tasks. The ViT model represents an input image as a series of image patches, like the series of word embeddings used when using transformers to text, and directly predicts class labels for the image.
   5. CNN uses pixel arrays, whereas ViT splits the images into visual tokens
4. The transformer encoder includes:
   1. Multi-Head Self Attention Layer (MSP): This layer concatenates all the attention outputs linearly to the right dimensions. The many attention heads help train local and global dependencies in an image.
   2. Multi-Layer Perceptrons (MLP) Layer: This layer contains a two-layer with Gaussian Error Linear Unit (GELU).
   3. Layer Norm (LN): This is added prior to each block as it does not include any new dependencies between the training images. This thereby helps improve the training time and overall performance.
5. ViT architecture
   1. Split an image into patches (fixed sizes)
   2. Flatten the image patches
   3. Create lower-dimensional linear embeddings from these flattened image patches
   4. Include positional embeddings
   5. Feed the sequence as an input to a state-of-the-art transformer encoder
   6. Pre-train the ViT model with image labels, which is then fully supervised on a big dataset
   7. Fine-tune on the downstream dataset for image classification
6. **ViT workflow explained**: CNN turns basic pixels into a feature map. Later, the feature map is translated by a tokenizer into a sequence of tokens that are then inputted into the transformer. The transformer then applies the attention technique to create a sequence of output tokens. Eventually, a projector reconnects the output tokens to the feature map. The latter allows the examination to navigate potentially crucial pixel-level details. This thereby lowers the number of tokens that need to be studied, lowering costs significantly.
7. ViT is more limited in terms of finetuning. ViT model is trained on a massive dataset giving its fantastic performance. BUT for smaller datasets its best to stick to SOTA CNNS like RestNet or EfficientNet.

<https://giters.com/kobiso/vision-transformer-pytorch>

<https://github.com/lukemelas/PyTorch-Pretrained-ViT>

<https://github.com/RizhaoCai/Awesome-FAS/issues/1>

https://www.idiap.ch/en/dataset/hq-wmca