A Vision Transformer Without Attention

Abstract:

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We assume that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.

Existing System:

Transformers were proposed by Vaswani for machine translation, and have since become the state-of-the-art method in many NLP tasks. Large Transformer-based models are often pre-trained on large corpora and then fine-tuned for the test at hand: BERT uses a denoising self-supervised pre-training task. Naïve application of self-attention to images would require that each pixel attends to every other pixel. With quadratic cost in the number of pixels, this does not scale to realistic input sizes. Thus, we are proposing a vision transformer without attention to be done.

Proposed System:

The standard Transformer receives as input a 1D sequence of token embeddings. To handle 2D images, we reshape the image into a sequence of flattened 2D patches. The Transformer uses constant latent vector size D through all of its layers, so we flatten the patches and map to D dimensions with a trainable linear projection. We refer to the output of this projection as the patch embeddings. Similar to BERT’s [class] token, we prepend a learnable embedding to the sequence of embedded patches whose state at the output of the Transformer encoder serves as the image representation y. Both during pre-training and fine-tuning, a classification head is attached. The classification head is implemented by a MLP with one hidden layer at pre-training time and by a single linear layer at fine-tuning time.

Software Tools:

1. TensorFlow
2. Keras
3. Jupyter Notebook
4. Colab
5. Python3
6. Anaconda
7. VS Code

Hardware Tools:

1. Laptop
2. Operating System: Windows 11
3. RAM: 16GB
4. ROM: 4GB
5. GPU
6. Fast Internet Connectivity