**Vision Transformer**

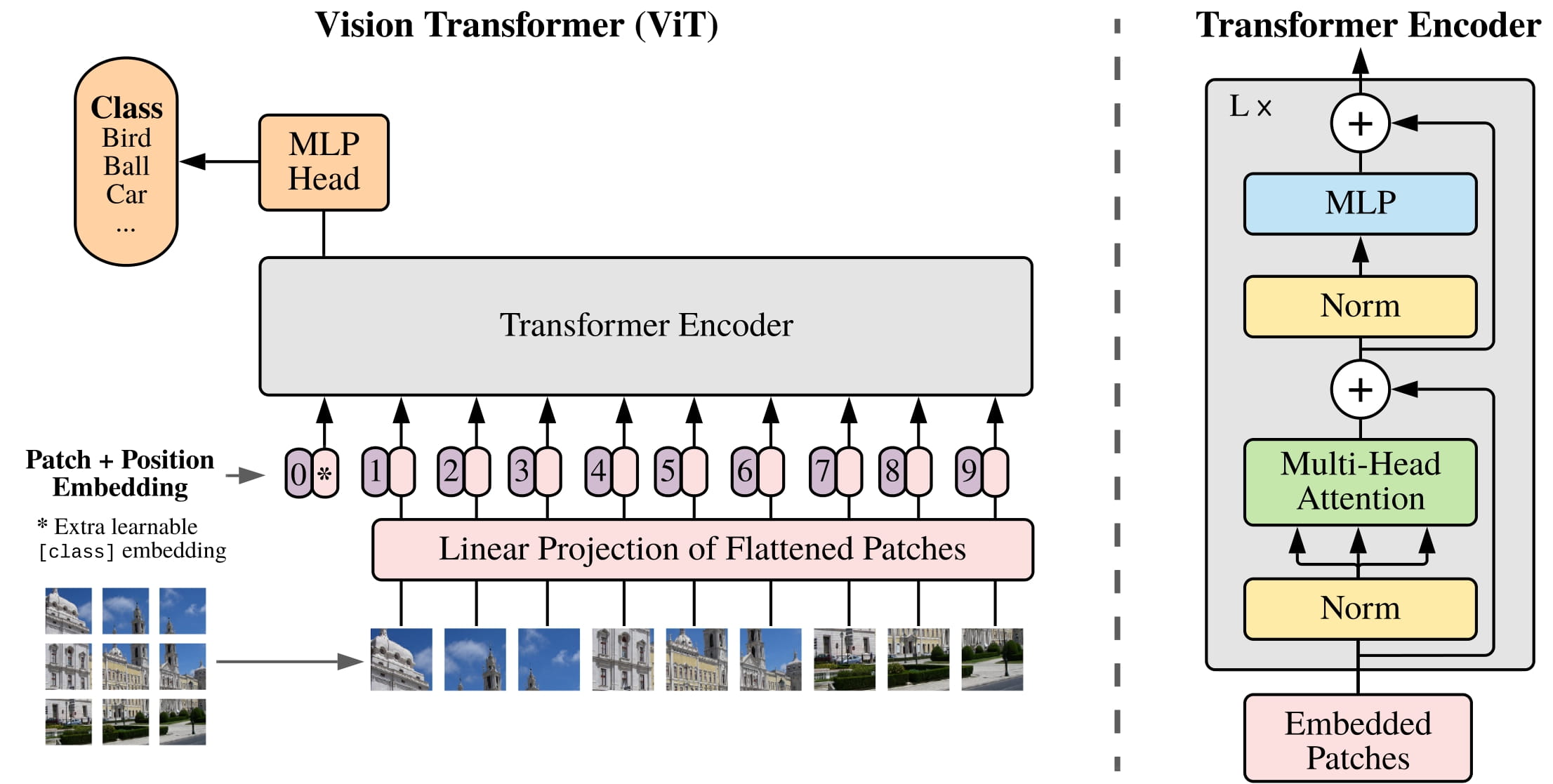
**1.Introduction**

Self-attention-based architectures, in particular Transformers (Vaswani et al., 2017), have become the model of choice in natural language processing (NLP). The dominant approach is to pre-train on a large text corpus and then fine-tune on a smaller task-specific dataset (Devlin et al., 2019). Thanks to Transformers’ computational efficiency and scalability, it has become possible to train models of unprecedented size, with over 100B parameters (Brown et al., 2020; Lepikhin et al., 2020). With the models and datasets growing, there is still no sign of saturating performance.

In computer vision, however, convolutional architectures remain dominant (LeCun et al., 1989; Krizhevsky et al., 2012; He et al., 2016). Inspired by NLP successes, multiple works try combining CNN-like architectures with self-attention (Wang et al., 2018; Carion et al., 2020), some replacing the convolutions entirely (Ramachandran et al., 2019; Wang et al., 2020a). The latter models, while theoretically efficient, have not yet been scaled effectively on modern hardware accelerators due to the use of specialized attention patterns. Therefore, in large-scale image recognition, classic ResNetlike architectures are still state of the art (Mahajan et al., 2018; Xie et al., 2020; Kolesnikov et al., 2020).

Inspired by the Transformer scaling successes in NLP, we experiment with applying a standard Transformer directly to images, with the fewest possible modifications. To do so, we split an image into patches and provide the sequence of linear embeddings of these patches as an input to a Transformer. Image patches are treated the same way as tokens (words) in an NLP application. We train the model on image classification in supervised fashion. When trained on mid-sized datasets such as ImageNet without strong regularization, these models yield modest accuracies of a few percentage points below ResNets of comparable size.

This seemingly discouraging outcome may be expected: Transformers lack some of the inductive biasesinherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data. However, the picture changes if the models are trained on larger datasets (14M-300M images). We find that large scale training trumps inductive bias.



**2. Model Architecture**

An overview of the model is depicted in Figure 1. The standard Transformer receives as input a 1D sequence of token embeddings. To handle 2D images, we reshape the image x ∈ R H×W×C into a sequence of flattened 2D patches xp ∈ R N×(P 2 ·C) , where (H, W) is the resolution of the original image, C is the number of channels, (P, P) is the resolution of each image patch, and N = HW/P2 is the resulting number of patches, which also serves as the effective input sequence length for the Transformer.

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 (Eq. 1). 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 (z 0 0 = xclass), whose state at the output of the Transformer encoder (z 0 L ) serves as the image representation y (Eq. 4). Both during pre-training and fine-tuning, a classification head is attached to z 0 L .

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. Position embeddings are added to the patch embeddings to retain positional information. We use standard learnable 1D position embeddings, since we have not observed significant performance gains from using more advanced 2D-aware position embeddings (Appendix D.4). The resulting sequence of embedding vectors serves as input to the encoder.

The Transformer encoder (Vaswani et al., 2017) consists of alternating layers of multiheaded self attention (MSA, see Appendix A) and MLP blocks (Eq. 2, 3). Layernorm (LN) is applied before every block, and residual connections after every block (Wang et al., 2019; Baevski & Auli, 2019).

The MLP contains two layers with a GELU non-linearity.

z0 = [xclass; x 1 pE; x 2 pE; · · · ; x N p E] + Epos, E ∈ R (P 2 ·C)×D, Epos ∈ R (N+1)×D

(1) z 0 ` = MSA(LN(z`−1)) + z`−1, ` = 1 . . . L

(2) z` = MLP(LN(z 0 `)) + z 0 `, ` = 1 . . . L

(3) y = LN(z 0 L)

**3.Dataset**

Plant Village Dataset is being used for vi Transformer model with the labels of 15 that contains disease type three plants Pepper, Potato, Tomato. To feed images to the Transformer encoder, each image is split into a sequence of fixed-size non-overlapping patches, which are then linearly embedded. A [CLS] token is added to serve as representation of an entire image, which can be used for classification. The authors also add absolute position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder

**4. Training**

The ViT is pre-trained on larger datasets (such as ImageNet, ImageNet-21k and JFT-300M) and fine-tuned to a smaller number of classes.

During pre-training, the classification head in use that is attached to the encoder output, is implemented by a MLP with one hidden layer and GELU non-linearity, as has been described earlier.

During fine-tuning, the MLP is replaced by a single (zero-initialized) feedforward layer of size, D×K, with K denoting the number of classes corresponding to the task at hand.

Fine-tuning is carried out on images that are of higher resolution than those used during pre-training, but the patch size into which the input images are cut is kept the same at all stages of training. This results in an input sequence of larger length at the fine-tuning stage, in comparison to that used during pre-training.

The implication of having a lengthier input sequence is that fine-tuning requires more position embeddings than pre-training. To circumvent this problem, Dosovitskiy et al. interpolate, in two-dimensions, the pre-training position embeddings according to their location in the original image, to obtain a longer sequence that matches the number of image patches in use during fine-tuning.