# CSE 465
# Lecture 15

Vision Transformer

# Vision Transformer

- The paper “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale” received a lot of “attention” from the community
- This was one of the pioneering works that implemented Transformers for image data and got acceptable results
- While attempts have been made in the past to apply Transformers for image processing, this paper was one of the first to apply Transformers to full-sized images.

# Abstract from the paper

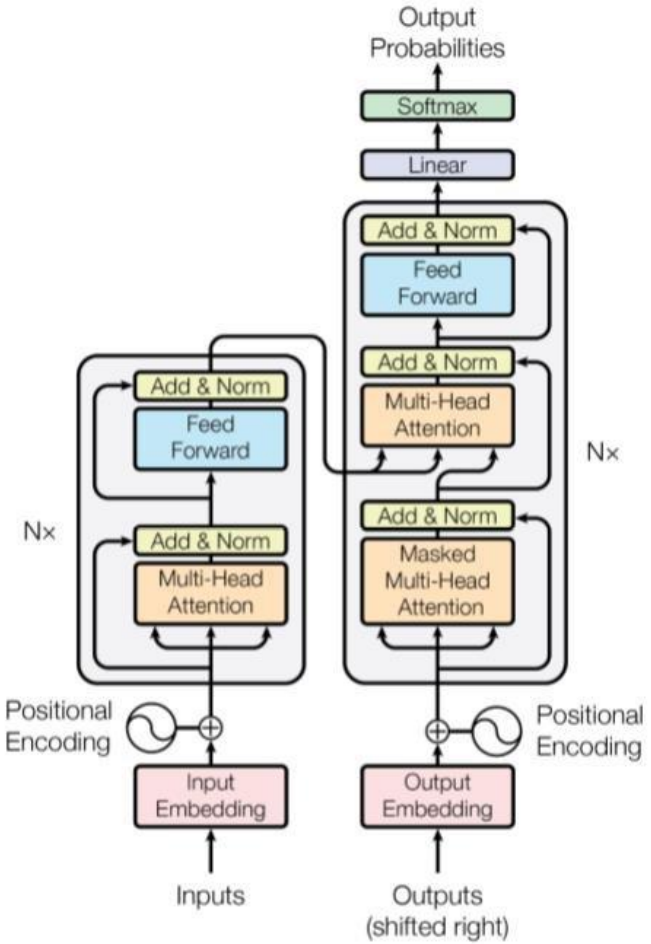
- “While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks or used to replace specific components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is unnecessary, 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.”

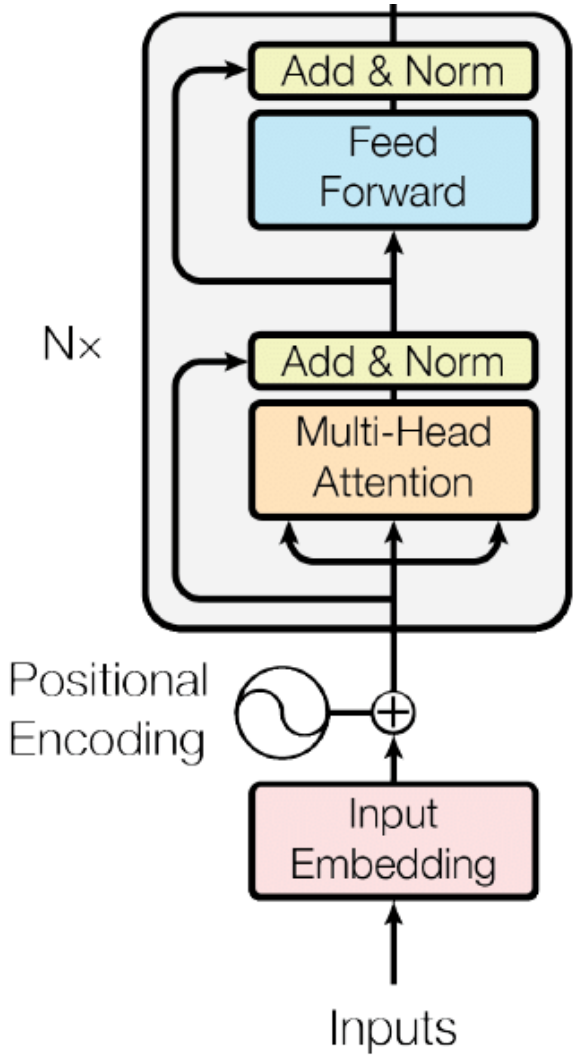
# What Does it do?

- The overall architecture can be described easily in five simple steps below:
  - Split an input image into patches.
  - Get linear embeddings (representation) from each patch, called Patch Embeddings.
  - Add position embeddings and a [cls] token to each Patch Embeddings.
  - Pass through a Transformer Encoder and get the output values for each [cls] token.
  - Pass the representations of [cls] tokens through an MLP Head to get final class predictions.
    - There is an MLP inside the Transformer Encoder, and an MLP Head that gives the class predictions, these two are different

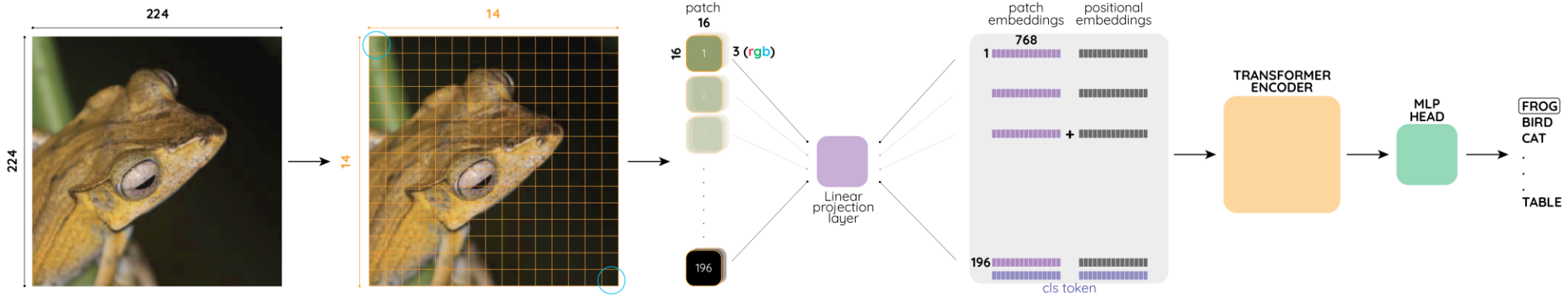
# Vision Transformer Overview

# The Transformer Architecture



# ViT: Everything Put Together



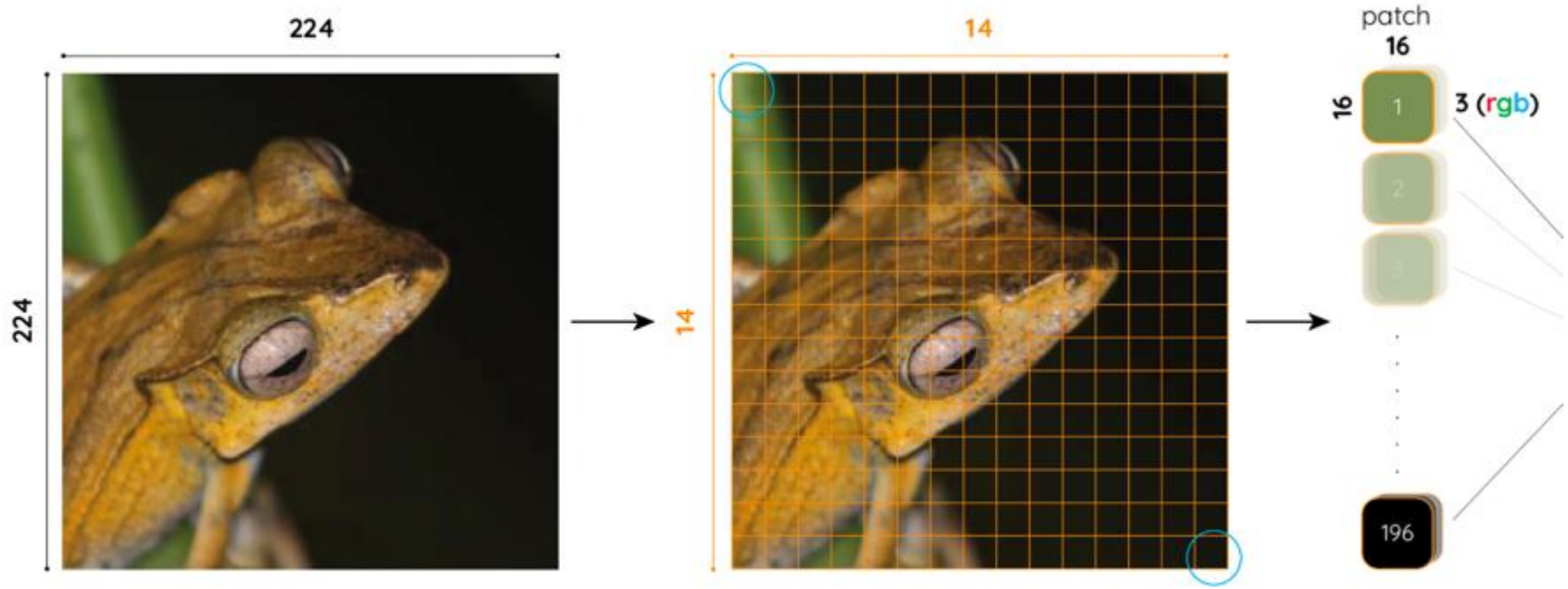

Let’s imagine that we want to classify a 3 channel (RGB) input image of a frog of size 224 x 224.

# First Step: Create Patches

- The first step is to create patches all over the image of patch size 16 x 16. Thus, we create 14 x 14 or 196 such patches.
- We can have these patches in a straight line where the first patch comes from the top-left of the input image and the last patch comes from the bottom-right.
- As can be seen from the figure, the patch size is 3 x 16 x 16 where 3 represents the number of channels (RGB).

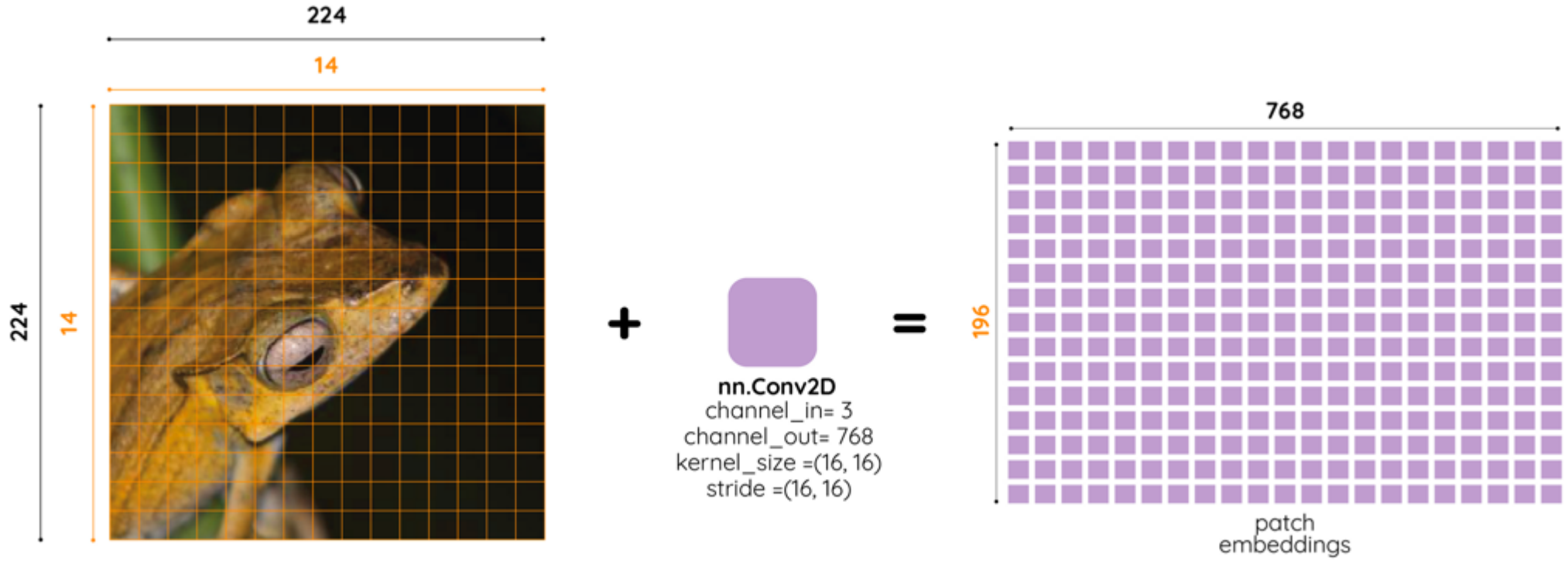
# Raw Patch Embeddings

- Image of size (3,224,224)
  - Divided into 196 (14×14) patches of size 3×16×16
    - 16*14 = 224 (original image size)

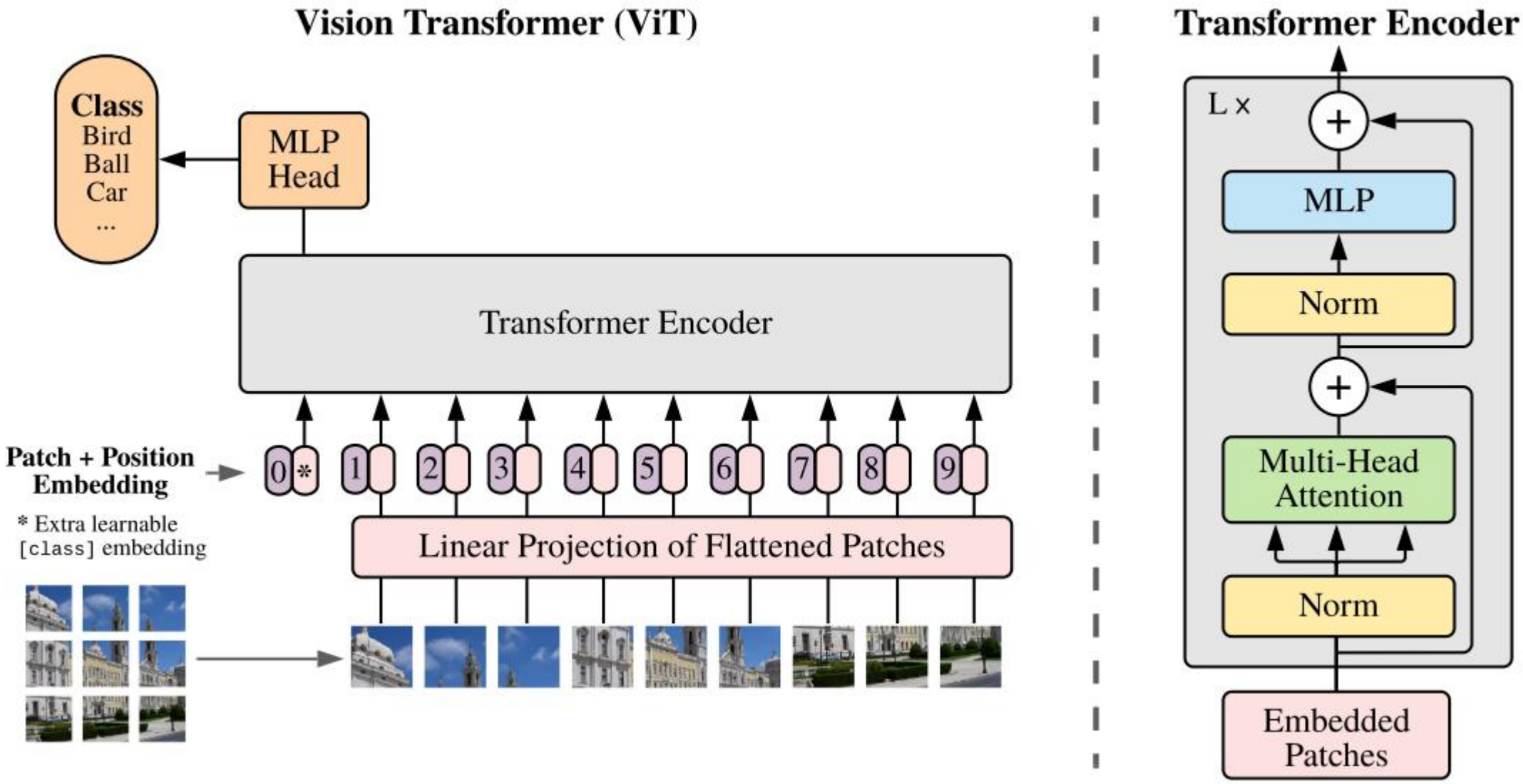


# Patch using CNN

- Each patch is converted into a vector of size 3×16×16=768
  - The input to the transformer encoder: 196 vectors of size 768 - a matrix of size (196,768)

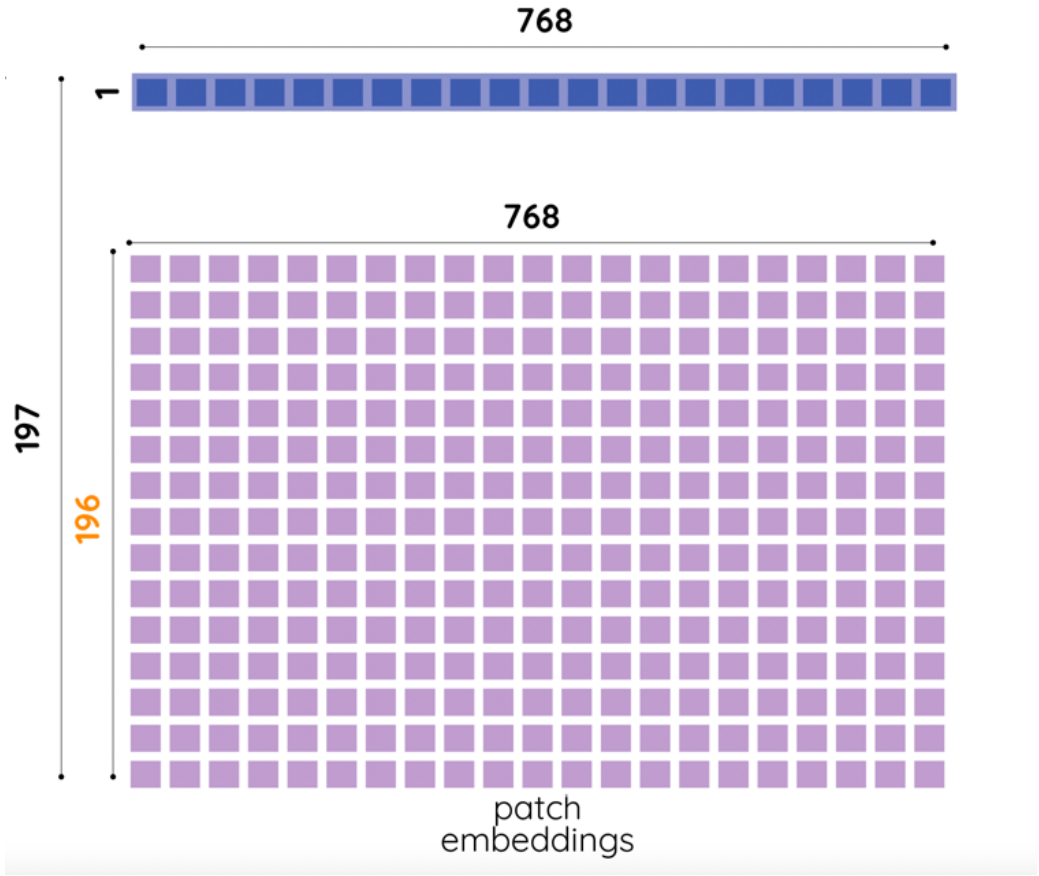


# Architecture



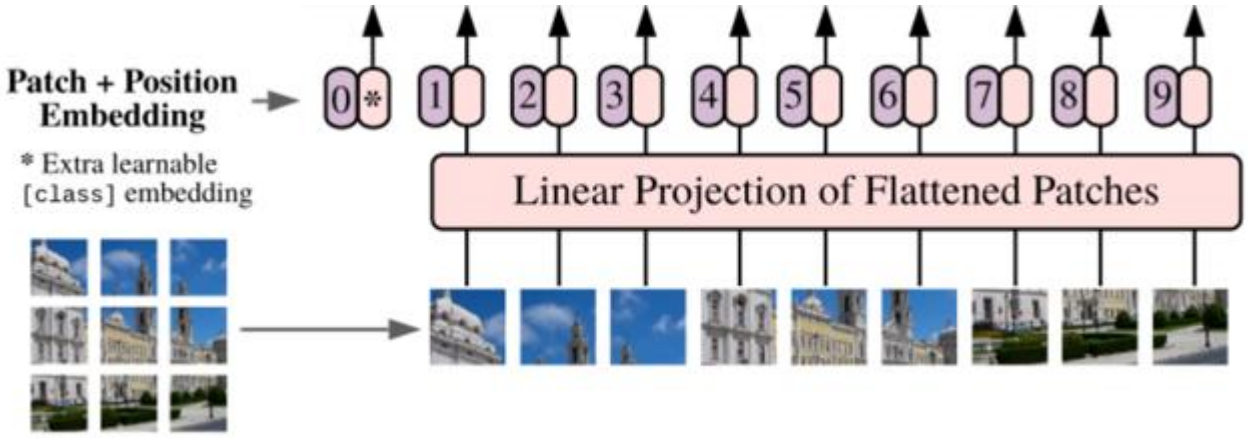
# [CLS] Token

- Similarly to the situation in BERT we need to add a [CLS] token
  - [CLS] token is a vector of size (1,768)
- The final patch matrix has size (197,768), 196 from patches and 1 [CLS] token



# Positional Embedding

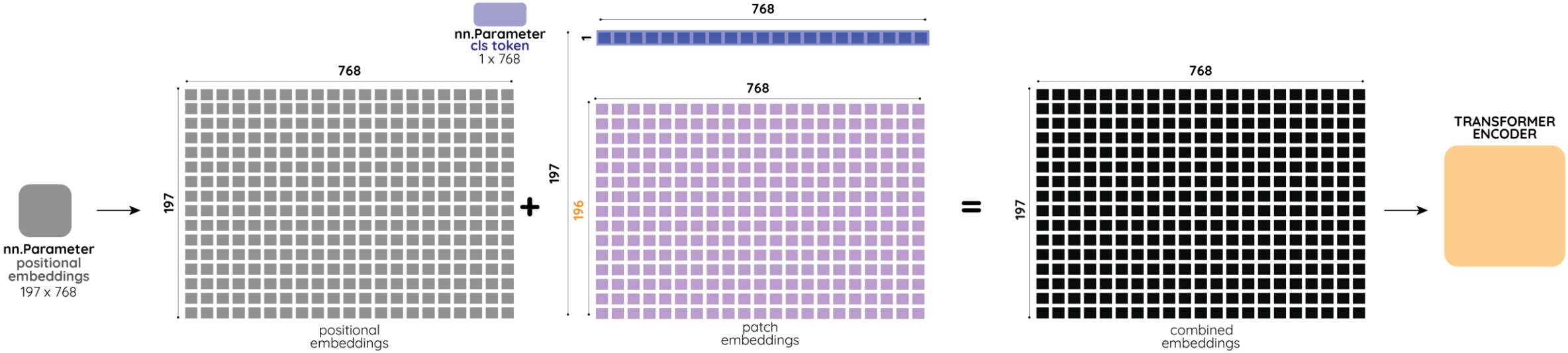
- Original implementation used standard learnable 1D position embeddings and the resulting sequence of embedding vectors serves as input to the encoder



# Make Patches: Second Step

- In the second step, the patches are passed through a linear projection layer to get a 1 x 768 long vector representation for each of the image patches.
- In the paper, the authors refer to these representations of the patches as Patch Embeddings.
  - The size of this patch embedding matrix is 196 x 768.
- A total of 196 patches and each patch has been represented as a 1 x 768 long vector.
  - Therefore, the total size of the patch embedding matrix is 196 x 768
  - Including the CLS the input should be 197X768

# Position Embeddings



# Positional Embedding

- Patch embeddings are augmented with positional information
- And a [cls] token (which is the only non-variable input)
  - Transformer Encoder learns the representations of the [cls] token based on the image
- Thus, the output from the Transformer Encoder would be of size 1 x 768, which is then fed to the MLP Head (which is nothing but a Linear Layer) as part of the final fifth step to get class predictions.

# Encoder

- Transformer Encoder consists of alternating layers of Multi-Head Attention and MLP blocks.
- Layer Norm is used before every block and residual connections after every block.
- The first layer of the Transformer Encoder accepts combined embeddings of shape197 x 768 as input.
- For all subsequent layers, the inputs are the outputs Out matrix of shape 197 x 768 from the previous layer of the Transformer Encoder.
- There are a total of 12 such layers in the Transformer Encoder of the ViT-Base architecture.
- Inside the layer, the inputs are first passed through a Layer Norm, and then fed to Multi-Head Attention block

# Encoder

- Inside each head of the Multi-Head Attention, the inputs are first converted using Q, K, and V matrices.
  - These (Q, K, V) have the dimension of 768X64 each
  - These matrices project the patches into 64X12 (768)
  - Thus, 12 different heads process each patch in parallel
  - Final outputs are concatenated into 768 again (12X64)
  - So, the patches are like words
- Output from all heads is for one layer only
- Transformer uses multiple such layers