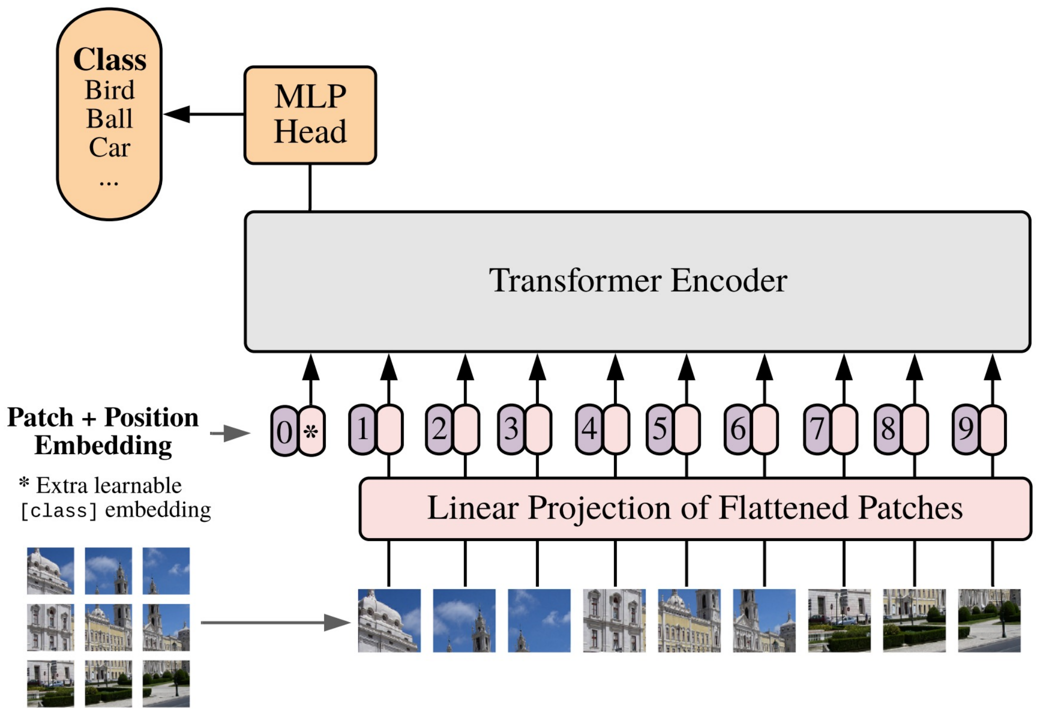
CS5670: Computer Vision

Vision Transformers



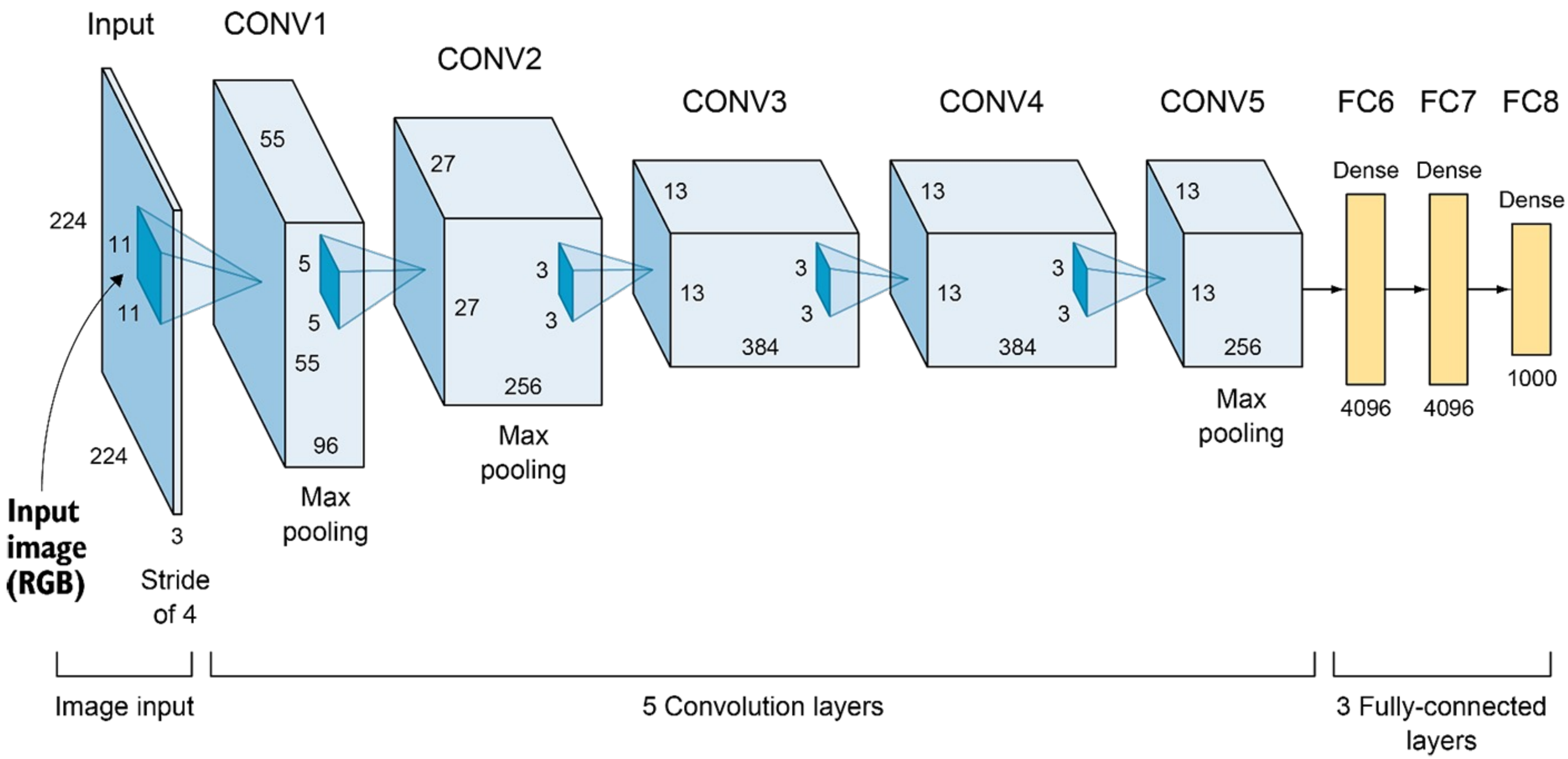
Readings

- Szeliski 2nd Edition, Chapter 5.5.3

Announcements

- Project 5 (Neural Radiance Fields) due Weds, May 1 by 8pm
- In class final on May 7
  - Allowed two sheets of notes (front and back sides)
- Course evaluations are open starting Monday, April 29
  - We would love your feedback!
  - Small amount of extra credit for filling out
    - What you write is still anonymous, instructors only see whether students filled it out
  - Link coming soon

Recall: ConvNets



Elgendy, *Deep Learning for Vision Systems,* https://livebook.manning.com/book/grokking-deep-learning-for-computer-vision/chapter-5/v-3/

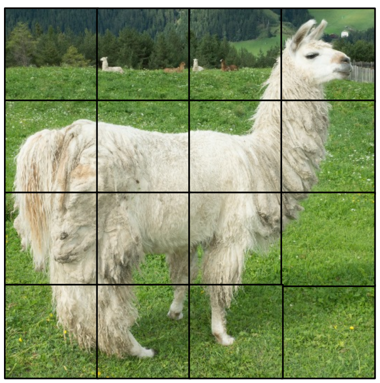
ConvNets assume spatial locality

- Assume nearby pixels are more important to making decisions than far away pixels (an example of an “inductive bias”)
- Only after stacking together several convolutional layers with spatial downsampling can distant pixels “talk” to each other
- As image datasets grow, we can do better by removing the spatial locality assumption and *learning how to process images from scratch*

An alternative to convolution: Attention

- Goal: consider long-range relationships between pixels

An alternative to convolution: Attention



Step 1: Break image into patches

An alternative to convolution: Attention

Step 1: Break image into patches

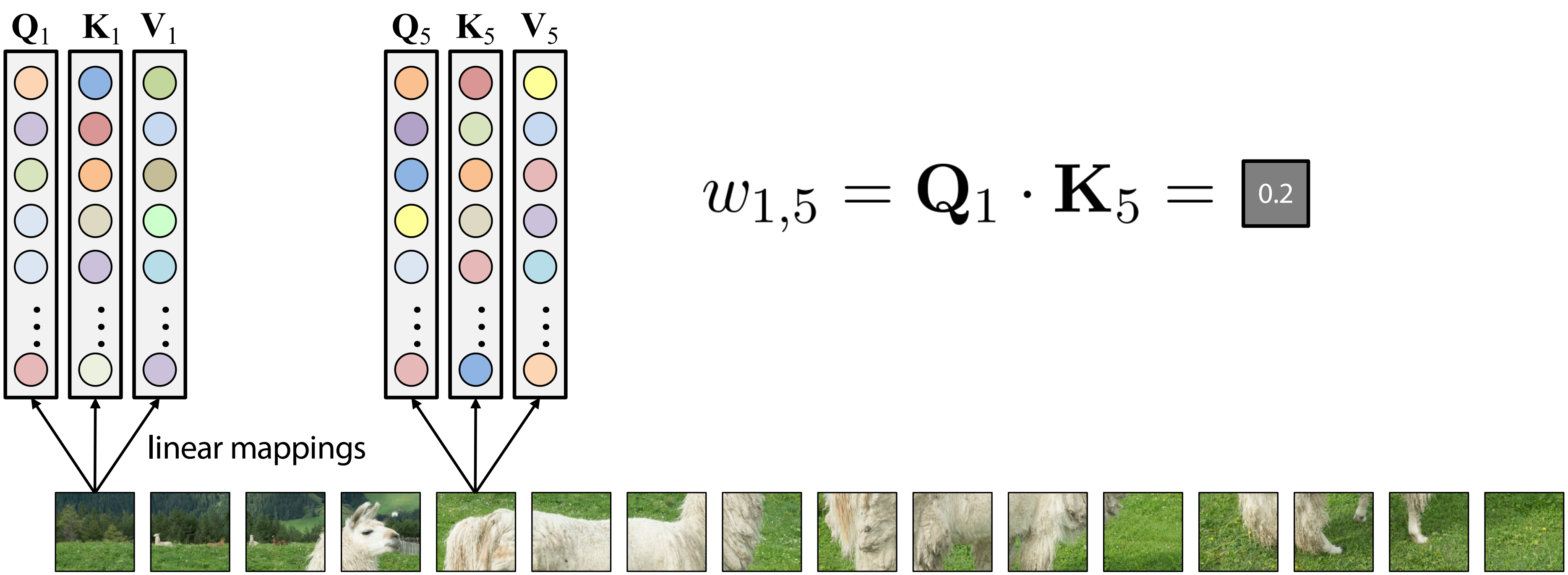
An alternative to convolution: Attention

Step 2: Map each patch to three vectors:

Query (Q), Key (K), and Value (V)

An alternative to convolution: Attention

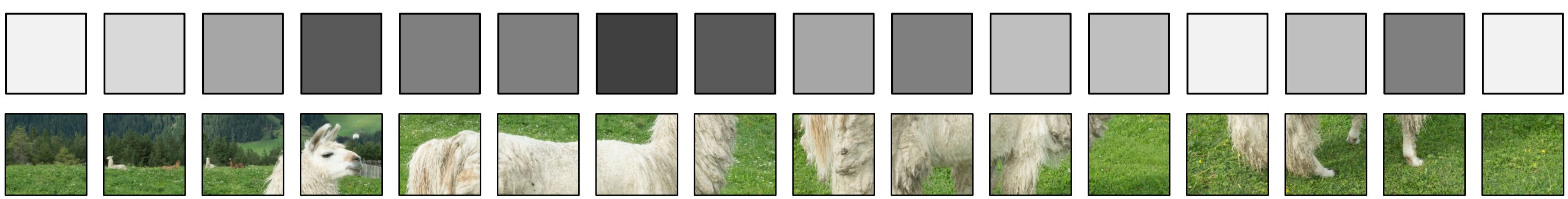
Step 3: For each patch, compare its query vector to all key vectors



$$w_{1,5} = \mathbf{Q}_1 \cdot \mathbf{K}_5 = \boxed{0.2}$$

An alternative to convolution: Attention

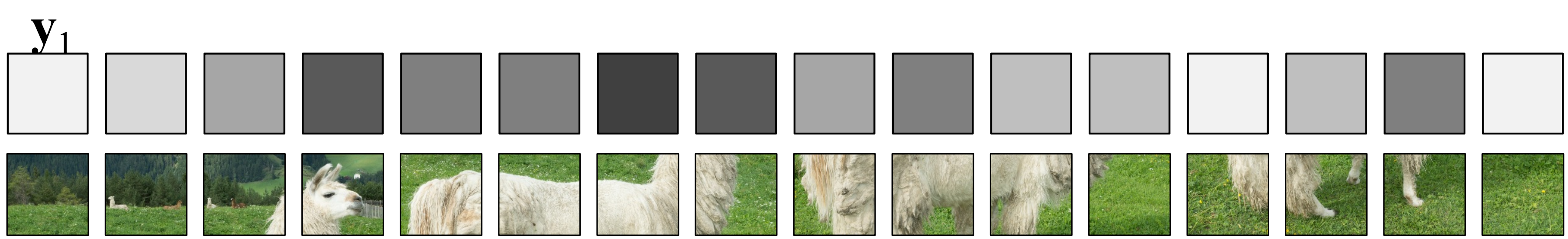
Step 3: For each patch, compare its query vector to all key vectors



An alternative to convolution: Attention

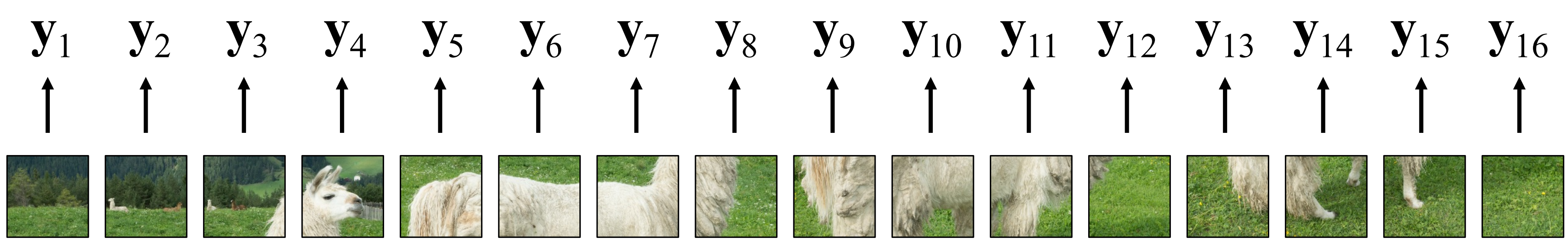
Step 4: Compute weighted sum of value vectors

$$\text{New vector } \mathbf{y}_1 = \sum_{i=1}^{n} \mathrm{softmax}\left(\frac{\mathbf{Q}_1 \cdot \mathbf{K}_i}{D}\right) \mathbf{V}_i$$



An alternative to convolution: Attention

Step 5: Repeat for all patches



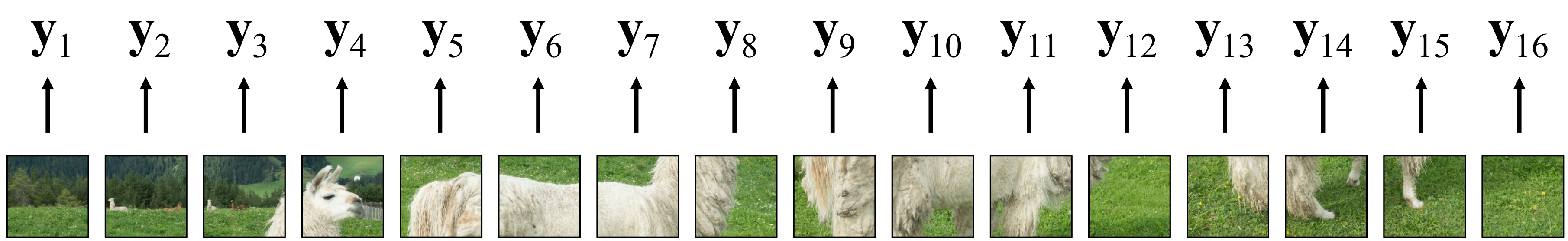
An alternative to convolution: Attention

Result: we've transformed all of the input patches into new vectors, by comparing vectors derived from all pairs of patches

This operation is called **attention** – the network can choose, for each patch, which other patches to attend to (i.e., give high weight to)

Unlike convolution, a patch is allowed to talk to the entire image

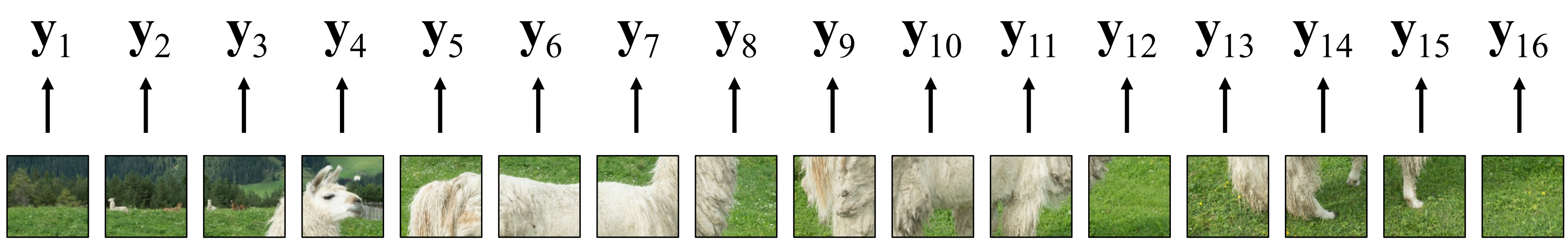
Attention is a *set-to-set* operation – it is equivariant to permuting the patches



An alternative to convolution: Attention

Parameters: weight matrices $\mathbf{W_q}$, $\mathbf{W_k}$, $\mathbf{W_v}$ that map input patches to query, key, and value vectors

$$\mathbf{Q}_i = \mathbf{W_q}\mathbf{x}_i, \mathbf{K}_i = \mathbf{W_k}\mathbf{x}_i, \mathbf{V}_i = \mathbf{W_v}\mathbf{x}_i$$



Details

- Rather than working with raw RGB image patches, the patches can themselves be features (e.g., produced by a linear mapping from RGB patches, or the output of a CNN)
- The feature vectors produced by the attention layer are often passed through an MLP (adding more parameters to the system)
- Each patch can be combined with a *positional encoding* indicating the spatial location of the patch, enabling spatial reasoning
- Instead of single $\mathbf{W_q}$, $\mathbf{W_k}$, $\mathbf{W_v}$ weight matrices, multiple linear mappings can be learned for an attention layer, and the resulting features concatenated (*multi-headed attention)*

Transformers

- Just like any network layer, we can stack attention layers – the output of one becomes the input to the next – to form a bigger network, called a **transformer**
- Transformers are very large, powerful learners that transcend convolutional networks by representing a larger class of functions

Vision Transformer (ViT)

- The network defined so far is designed for image classification, and roughly follows:

AN IMAGE IS WORTH 16X16 WORDS:
TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], Lucas Beyer[*], Alexander Kolesnikov[*], Dirk Weissenborn[*], Xiaohua Zhai[*], Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby[*,†]

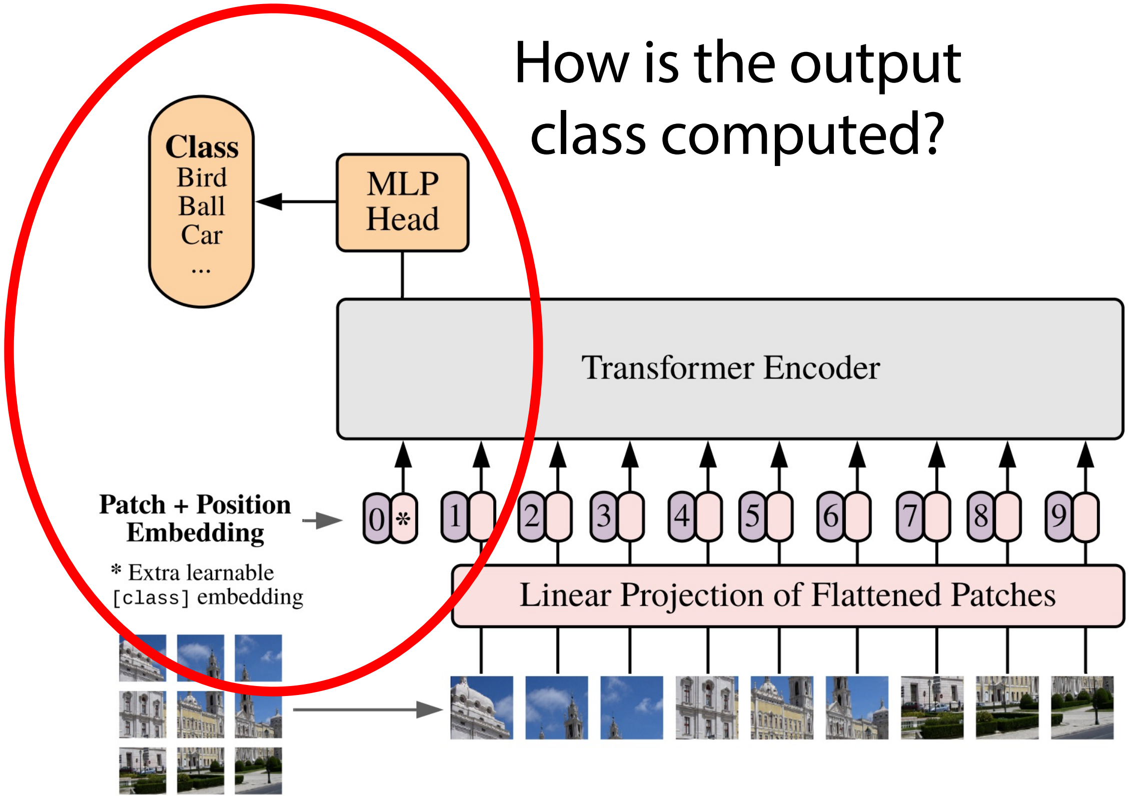
[*]equal technical contribution, [†]equal advising

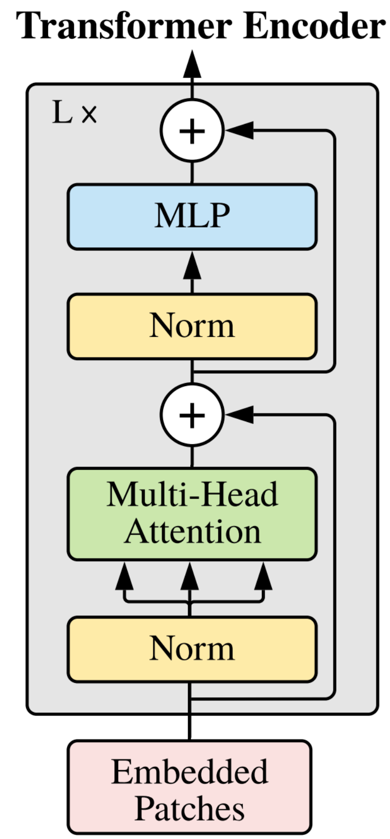
Google Research, Brain Team

ICLR 2021

Vision Transformer (ViT)

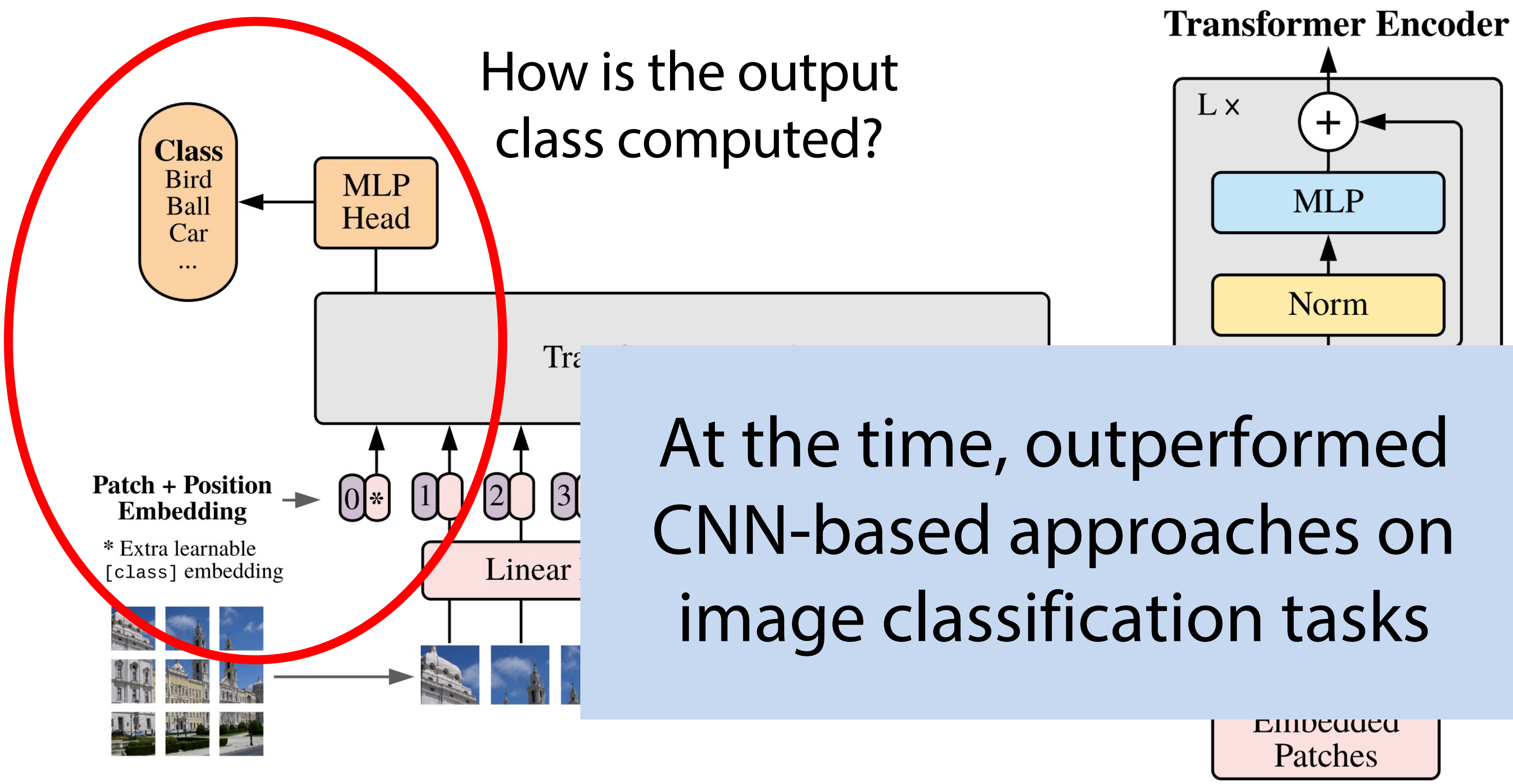
How is the output class computed?





Vision Transformer (ViT)

How is the output class computed?



At the time, outperformed CNN-based approaches on image classification tasks

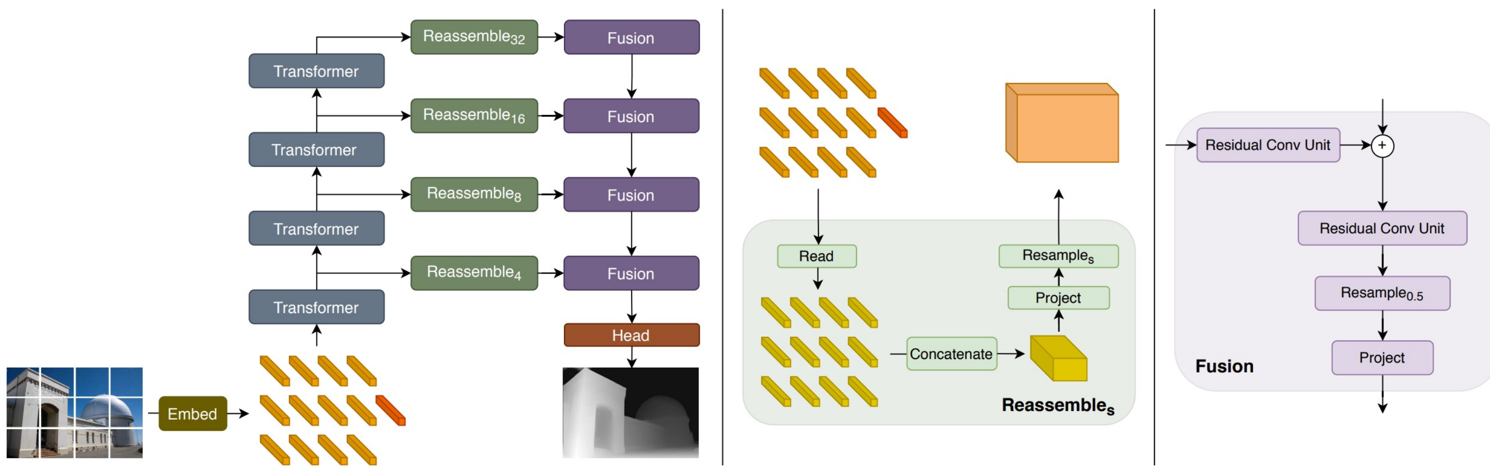
Vision Transformer (ViT)

- Note: this is just one possible approach – lots of others variants of transformers for vision task exist!
- (For instance, combinations of transformers and CNNs)

DPT: Dense Prediction Transformers

[Ranftl et al., 2021]

- Predicts an image-shaped output (e.g., segmentation map or depth map) from an image-shaped input



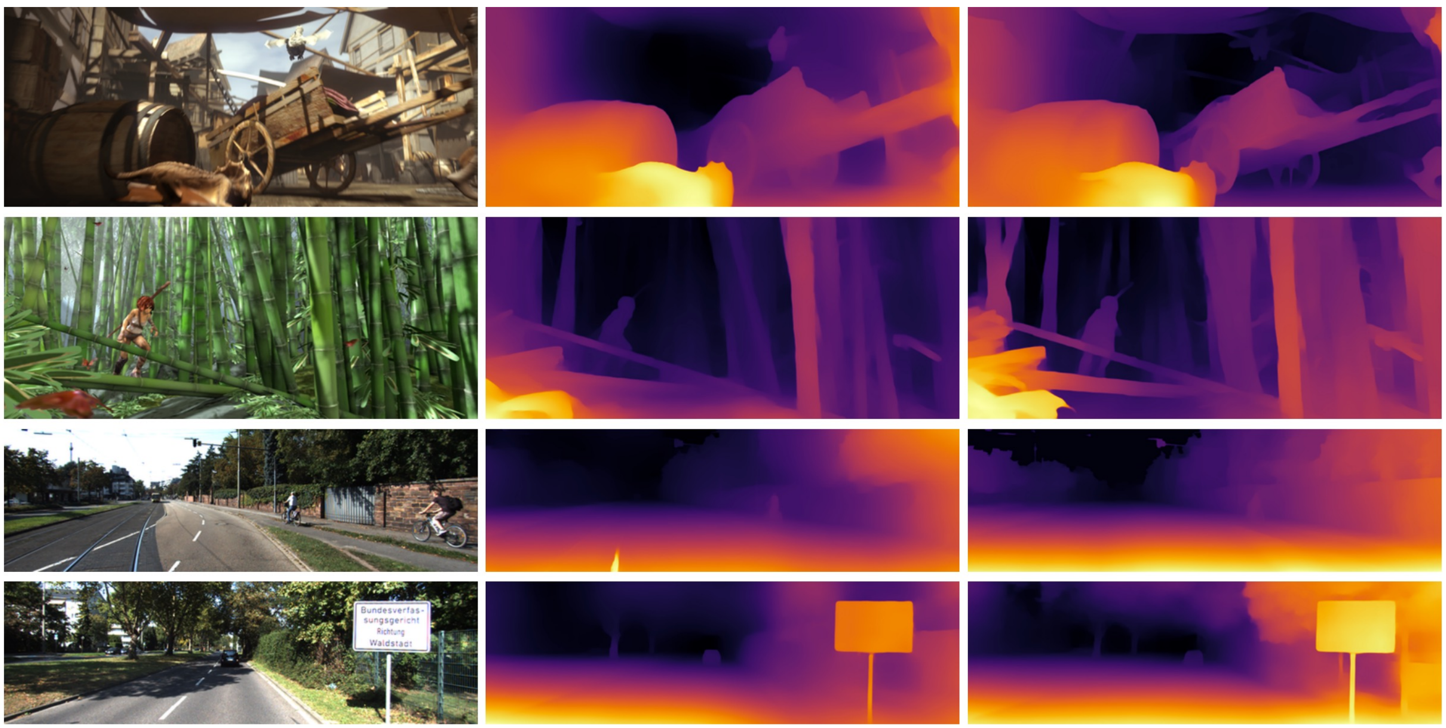
DPT architecture

Reassemble operation

Fusion operation

DPT: Depth prediction results

Input | MiDaS (CNN-based) | DPT (Transformer)



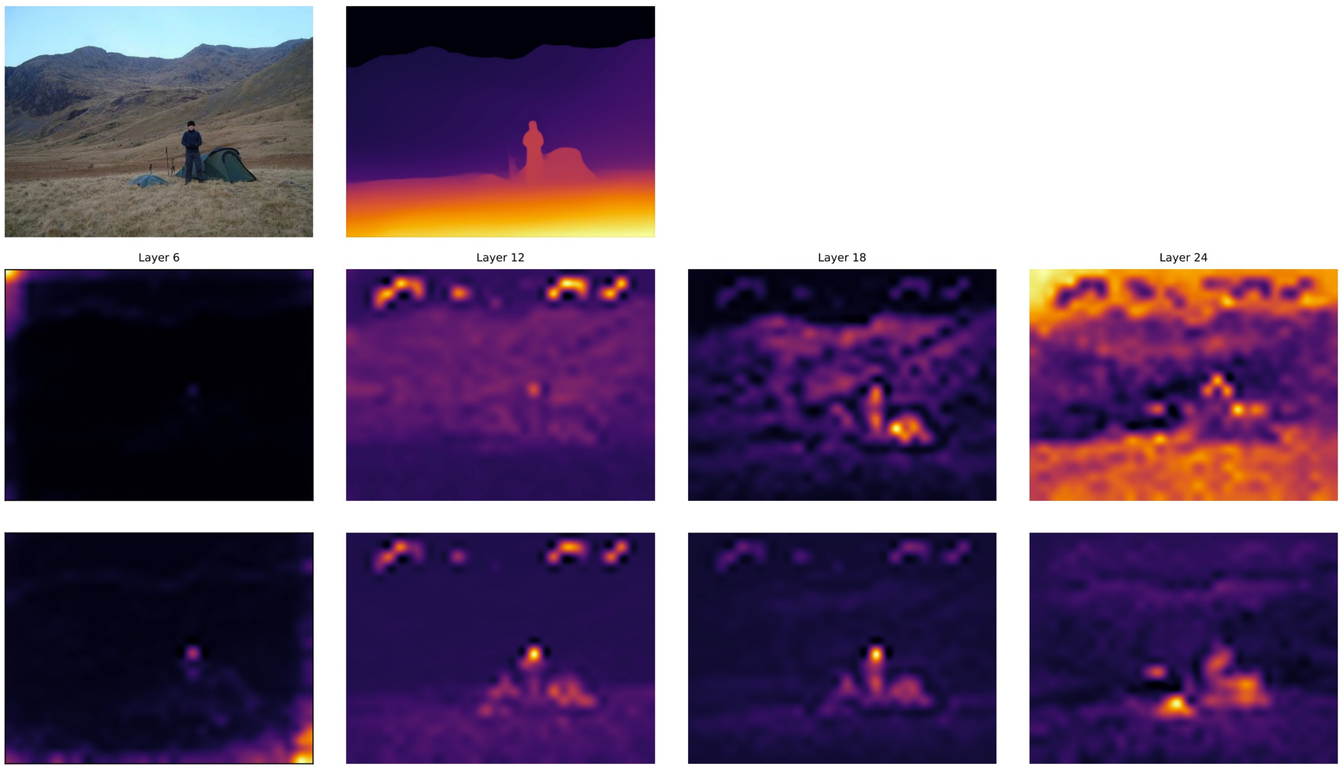
DPT: Attention maps

Input

Depth prediction

Attention maps for upper right corner

Attention maps for lower right corner



Questions?