Vision Transformers I

An Image is Worth 16x16 Words

Katie Awty-Carroll, Ryan Chan

07 August 2023

Outline

Overview of Deep Learning for Computer Vision
- Background
- Applications

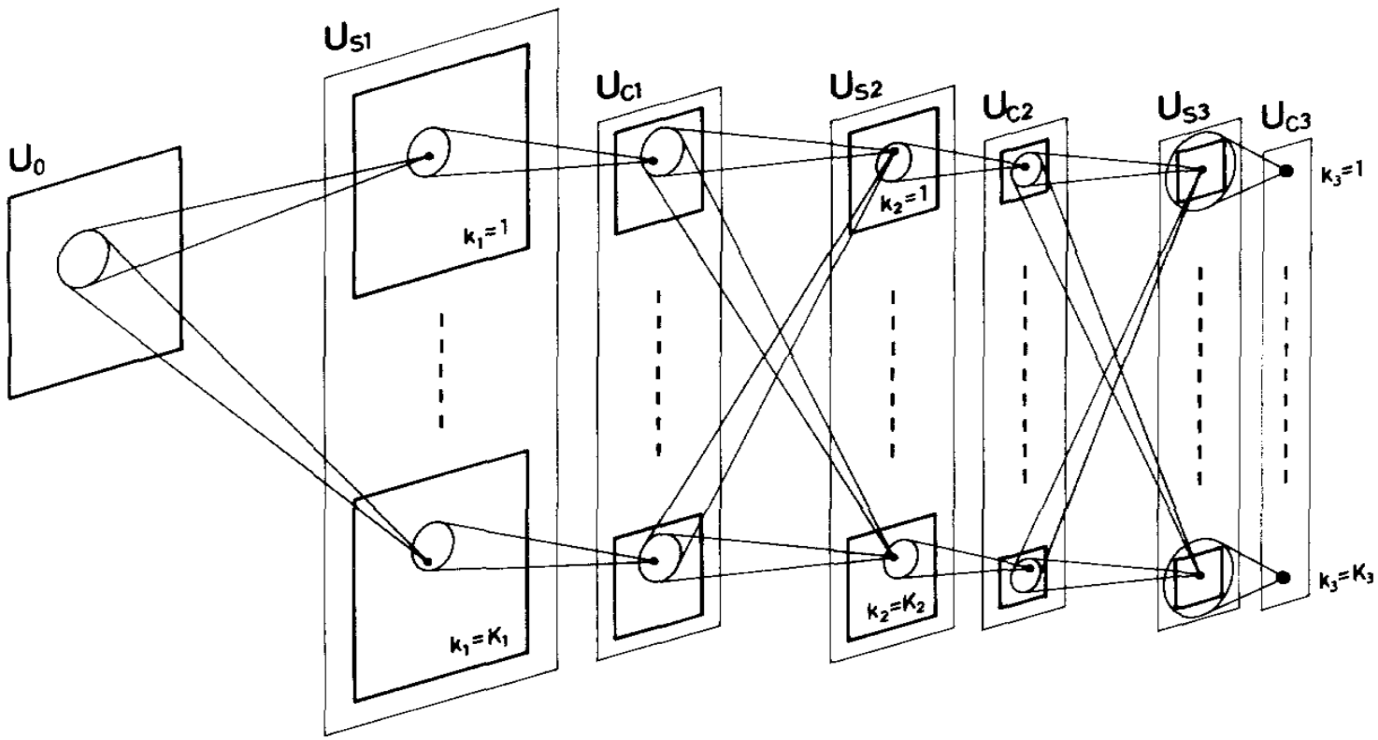
Vision Transformer
- Motivation
- Using Transformer encoder for image classification
- Summary of results

The dawn of the CNN

- Fukushima [1980] proposed the first NN architecture capable of processing images in a manner unaffected by position or distortion
  - Strongly inspired by biology
  - "If we could make a neural network model which has the same capability for pattern recognition as a human being, it would give us a powerful clue to the understanding of the neural mechanism in the brain."
- This architecture was named the Neocognitron
- The Neocognitron introduced two major processes we use in modern CNNs - feature extraction and pooling
  - These are termed S-cells and C-cells to correspond with simple and complex cells within the visual cortex

The Neocognitron

- Multilayered architecture with layered S-cells and C-cells

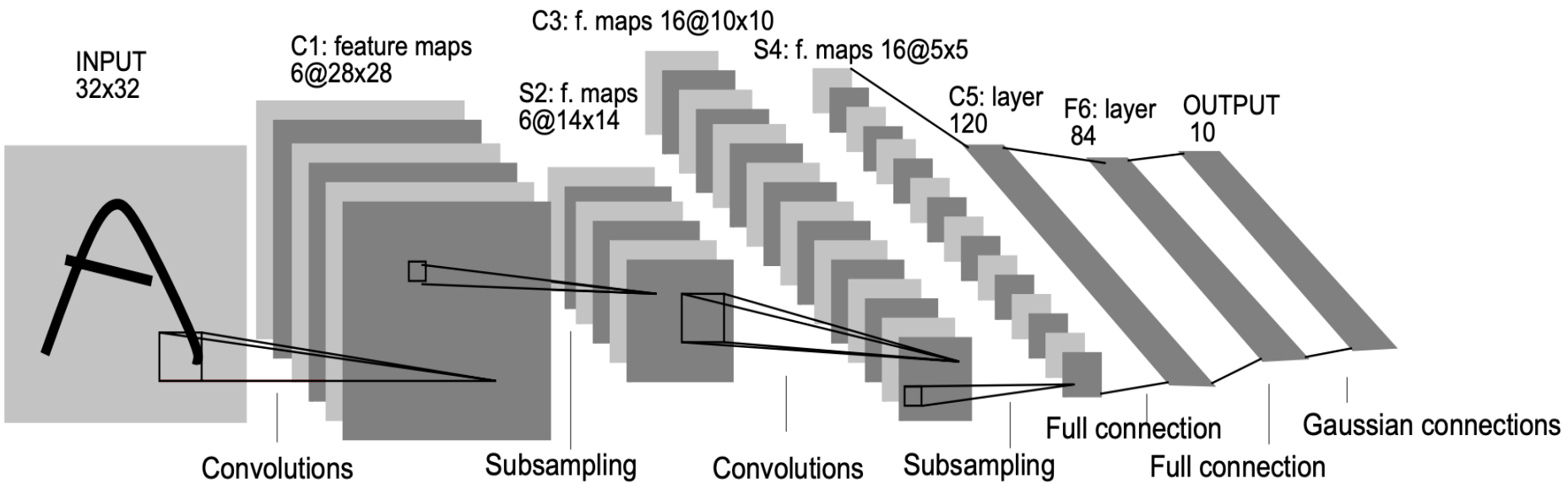


[1]

[1] Fukushima [1980]

LeNet-5

- The first modern CNN was proposed by LeCun et al. [1998]
- LeNet is a 7-layer architecture with three convolutional layers, two average pooling layers, and two fully connected layers
  - Designed for character recognition with grayscale (2D) inputs
  - Approx. 60k parameters



[2]LeCun et al. [1998]

2D convolutions

- 2D convolutions are fundamental to feature extraction in CNNs
- Given an input image $\boldsymbol{f}$ and a kernel $\boldsymbol{h}$ we calculate the output matrix $\boldsymbol{g}$ where $\boldsymbol{g}[m, n] = (\boldsymbol{f} * \boldsymbol{h})[m, n] = \sum_j \sum_k \boldsymbol{h}[j, k]\boldsymbol{f}[m - j, n - k]$

2D convolutions

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0

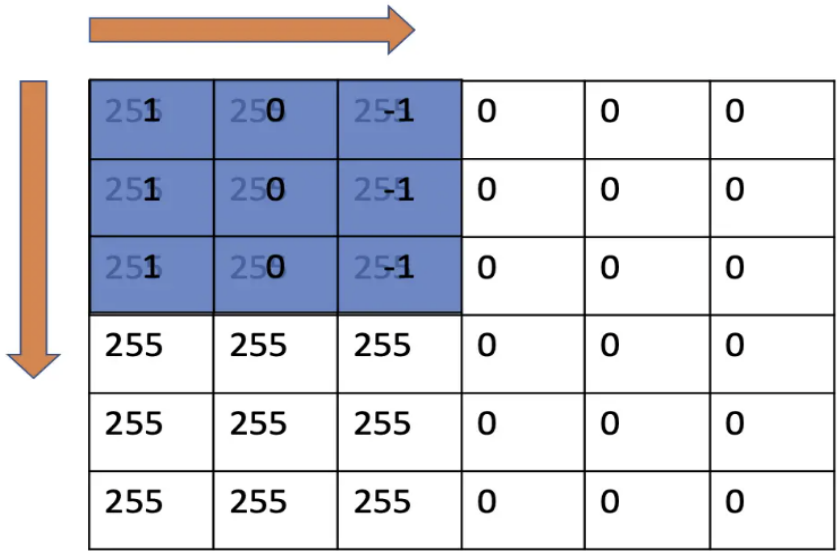
$*$

1	0	-1
1	0	-1
1	0	-1

3

[3] programmathically.com

2D convolutions



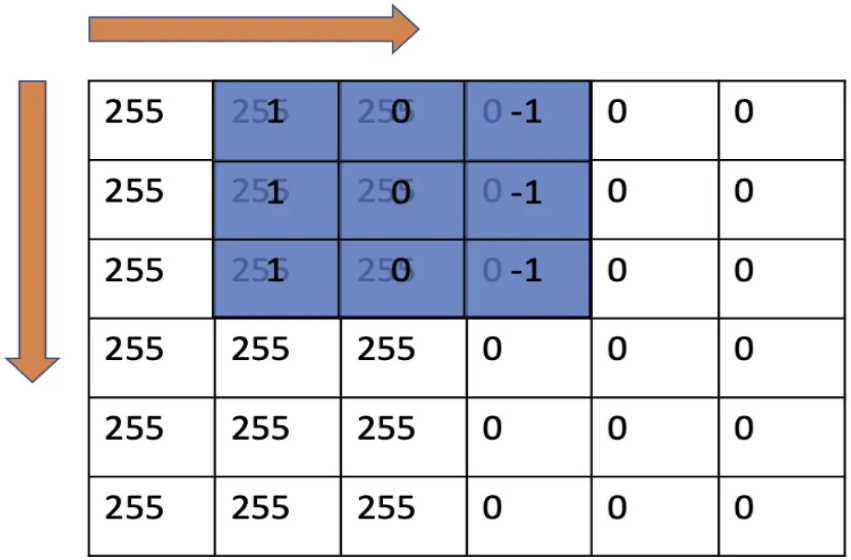
1	0	-1	0	0	0
1	0	-1	0	0	0
1	0	-1	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0

4

$1x255 + 0x255 + (-1)x255$
$+ 1x255 + 0x255 + (-1)x255$
$+1x255 + 0x255 + (-1)x\ 255$
$= 0$

[4] programmathically.com

2D convolutions



255	1	0	-1	0	0
255	1	0	-1	0	0
255	1	0	-1	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0

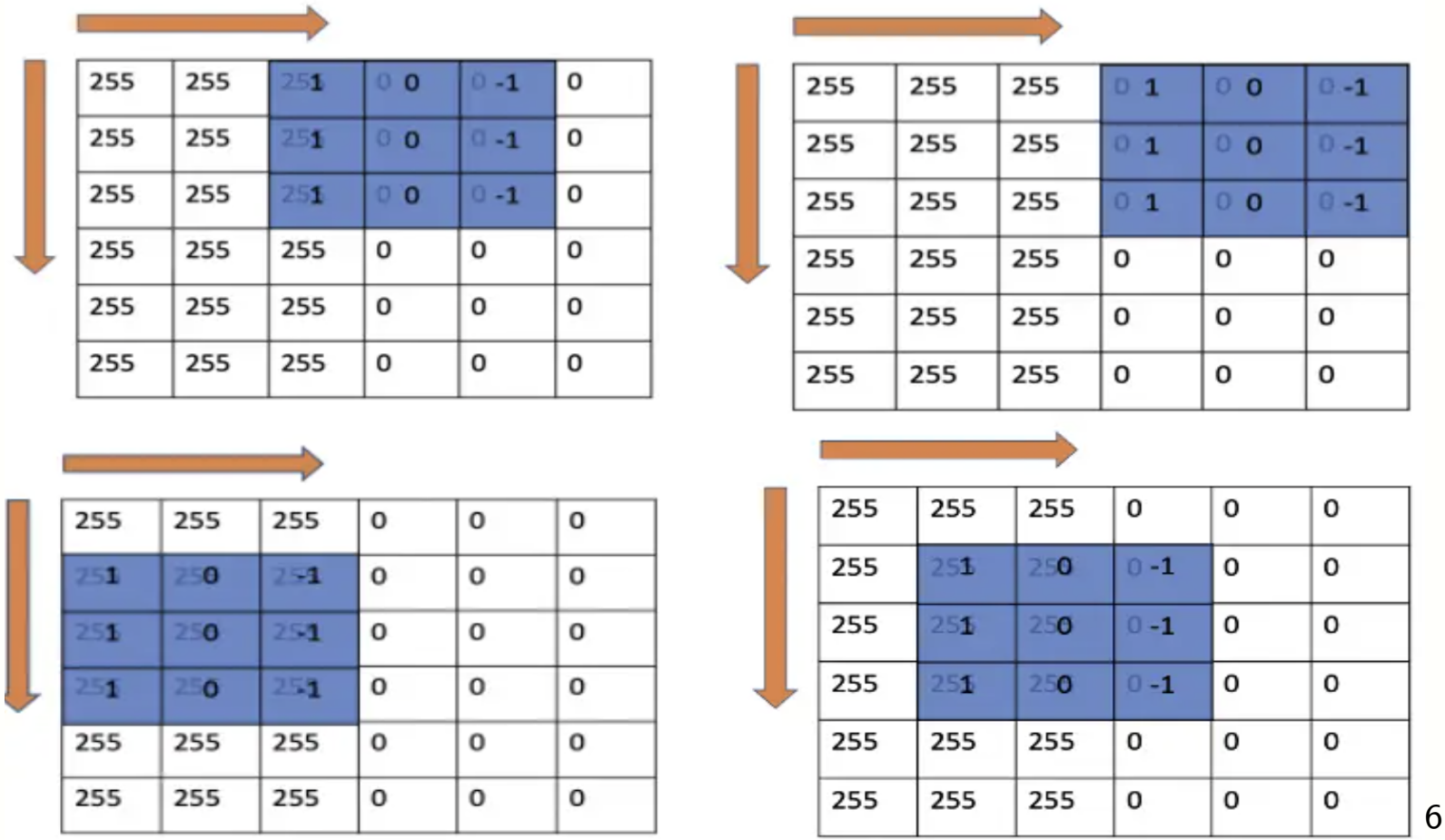
[5]

$1x255 + 0x255 + (-1)x0$
$+ 1x255 + 0x255 + (-1)x0$
$+ 1x255 + 0x255 + (-1)x0$
$= 765$

[5] programmathically.com

2D convolutions

etc.

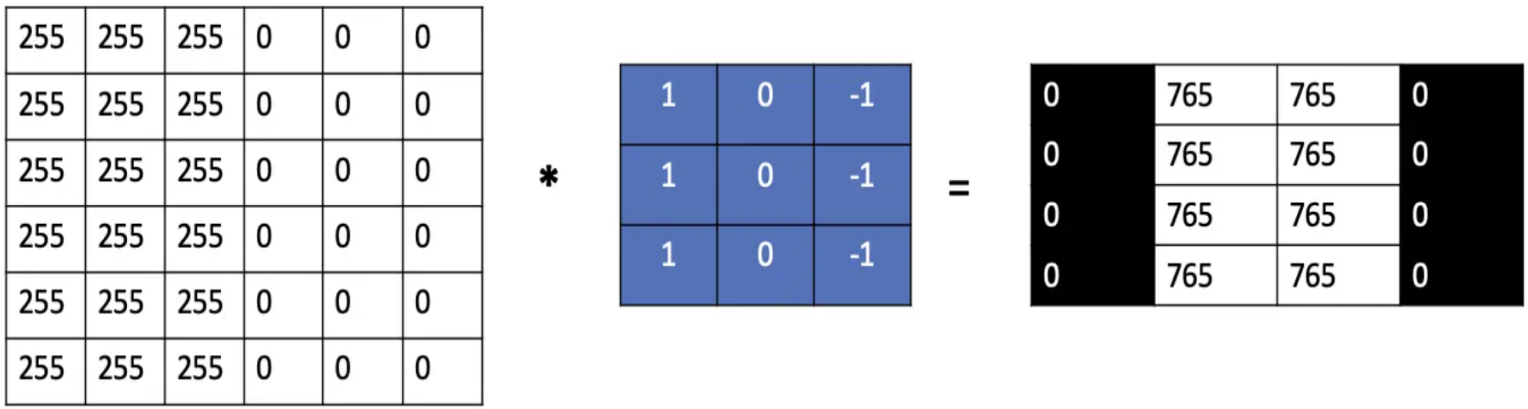


[6]

[6] programmathically.com

2D convolutions

- This particular kernel detects edges, as we can see by the result

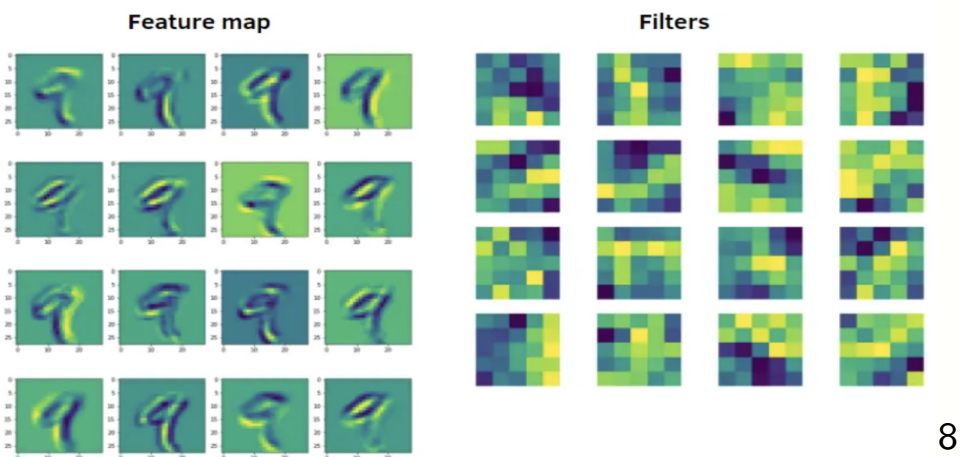


7

[7] programmathically.com

The convolutional layer

- A given convolutional layer in a CNN is made up of multiple kernels where the value of each element in the kernel is a learnable parameter
- For example, Layer 1 of LeNet-5 is made up of 6 5x5 kernels
  - For a given input of size 32x32, this results in 6 feature maps each of size 28x28 (also called an activation map)
- The conv layer is typically followed by a nonlinearity (tanh activation in the case of LeNet-5)



[8]

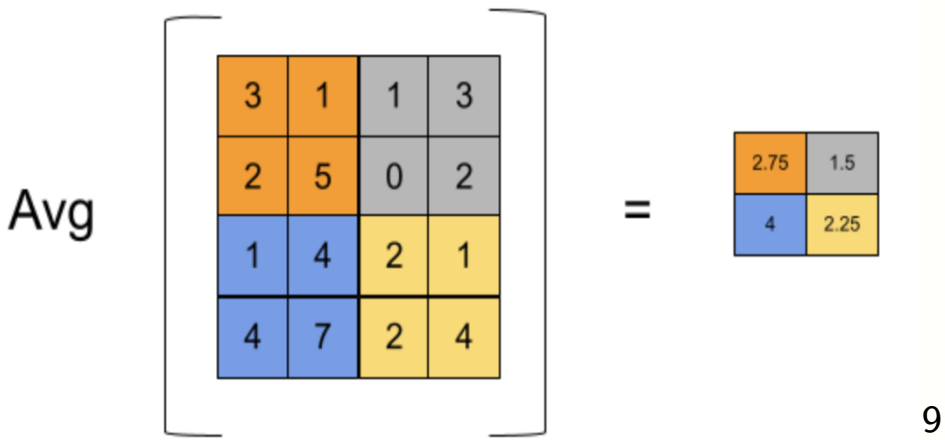
[8] Eugenia Anello

A note about padding

- There are a couple of issues with performing convolutions as I've represented it so far
  - Edge pixels are less influential
  - Our output image is small than our input image
- These effects can be mitigated through image padding
- Typically when implementing convolution there is a choice between same or valid padding
- Same padding pads the image by p pixels where $p = \frac{f-1}{2}$ (where f is the size of the filter)
- Valid padding uses the original image only, resulting in the output size being reduced by $f - 1$ in both dimensions

The pooling layer

- LeNet-5 also implements average pooling
  - Max pooling is now more common
- Pooling downsamples the feature maps and makes the network invariant to local shifts and translations

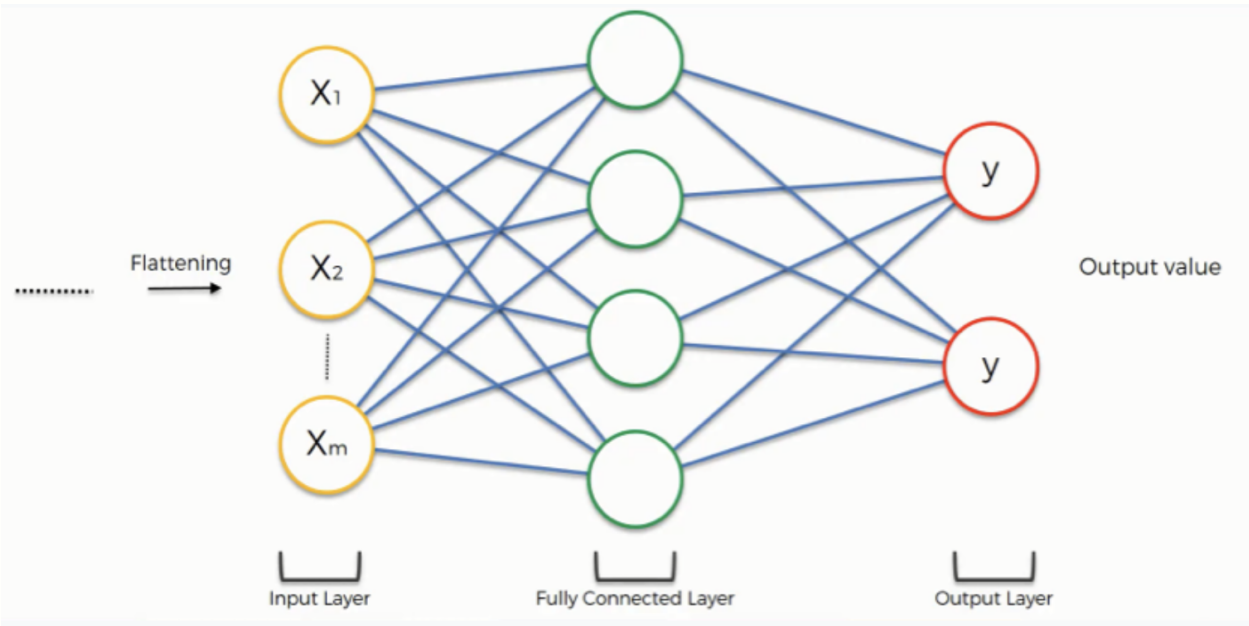


9

[9] For Machine Learning

FC layers

- Fully connected layers serve to flatten the output and produce the final result



10

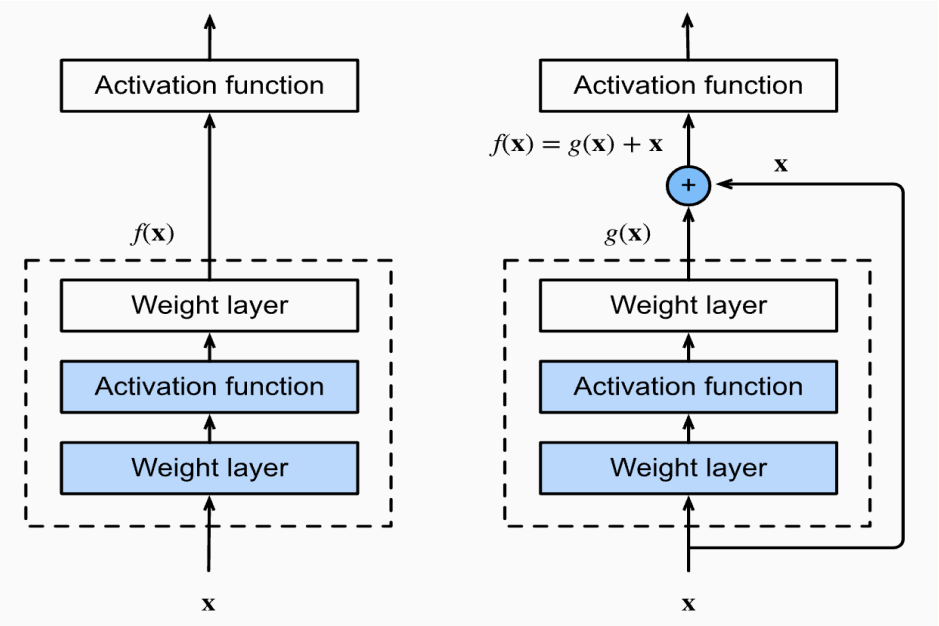
10 SuperDataScience

Further developments

- LeNet gave us the classic CNN architecture
- AlexNet [Krizhevsky et al., 2012] (∼60M params)
  - Introduced normalisation and ReLu
  - Reduced error on ImageNet from 25.8% to 16.4%
- VGGNet [Simonyan and Zisserman, 2014] (∼135M params (VGG-16))
  - Increased network depth - VGG-16 has 16 convolutional layers
- ResNet [He et al., 2015] (∼24M params (ResNet-50))
  - Introduced residual blocks through identity (or skip) connections, solving the vanishing gradient problem in CNNs

Residual blocks

- Residual blocks implement skip connections - meaning the outout of a layer is not only connected to the next layer, but to another layer n hops away
- For a given model with an input $\boldsymbol{x}$ we want to learn the mapping function $f(\boldsymbol{x})$
- Rather than directly learning $f(\boldsymbol{x})$, the network only has to learn the residual mapping $g(\boldsymbol{x}) = f(\boldsymbol{x}) - \boldsymbol{x}$

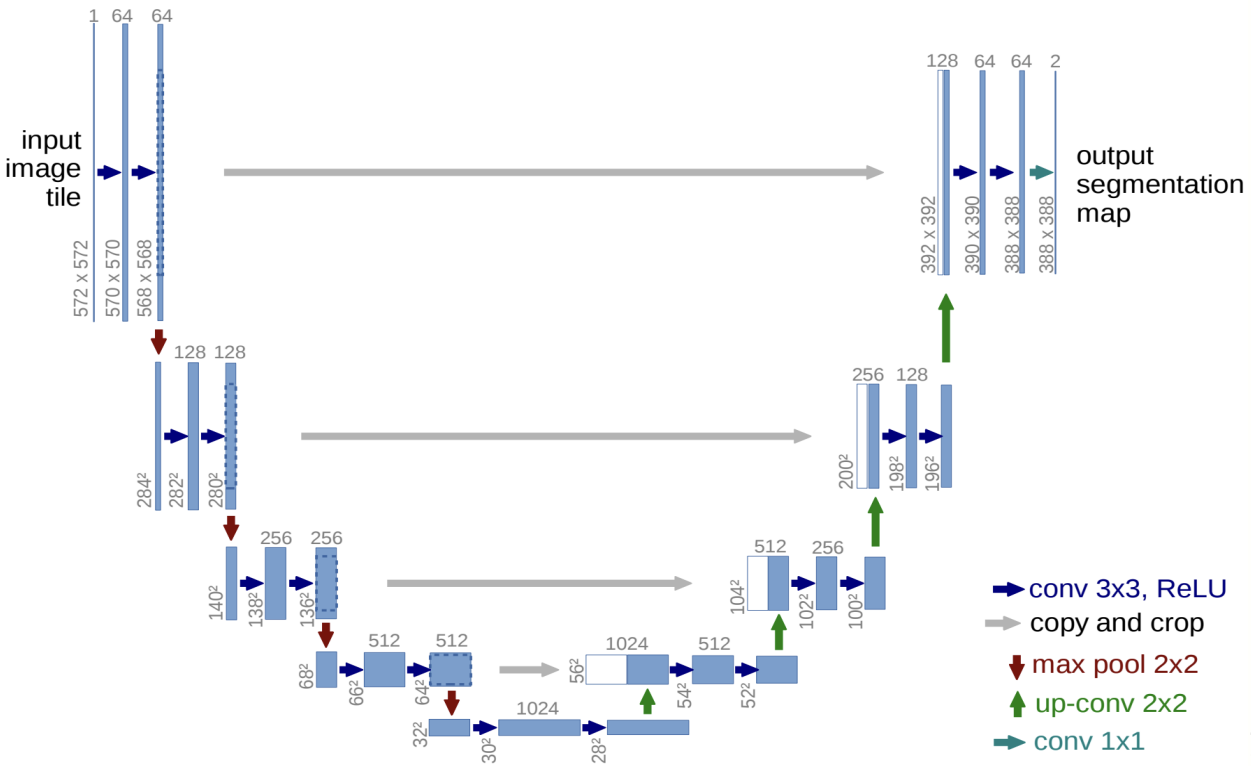


[11]

[11] Dive Into Deep Learning

U-Net

- Developed by Ronneberger et al. [2015] for medical image segmentation



12

[12] [Ronneberger et al., 2015]

Conclusion

- CNNs have been the SOTA for image classification, object detection, semantic segmentation, and instance segmentation for >10 years
  - Hierarchical structure results in strong inductive bias
- However, they are computationally expensive to train and require large training datasets to achieve high accuracy

The Transformer

- The Transformers architecture [Vaswani et al., 2017] has become the de-facto standard model for NLP tasks, but its application within computer vision has been limited
- With Transformers, the dominant paradigm within NLP is:
  - Pre-train on a large (unlabelled) text corpus
  - Fine-tune the (pre-trained) model on a (smaller) task-specific dataset
- Transformers are computationally efficient and scalable (since many of its operations can be easily parallelised) - it is now possible to train models of unprecedented size with over 100B parameters
  - GPT4 is rumoured to have over 1T parameters!

The Vision Transformer

- In computer vision, convolution neural networks (CNNs) [LeCun et al., 1989; He et al., 2016] are the state-of-the-art (SOTA)
  - There has been work to incorporate attention mechanisms within CNNs - promising but required complex engineering to implement efficiently on hardware accelerators
- A naive application of self-attention would require every pixel attends to every pixel
  - Results in quadratic cost in the number of pixels and does not scale to realistic input sizes
- Dosovitskiy et al. [2020] propose the Vision Transformer (ViT) which is a pure transformer encoder (as in Vaswani et al. [2017]) applied directly to sequence of image patches
  - They pre-train and fine-tune on the image classification task
  - Attains similar results to state-of-the-art convolution networks at a lower cost
  - For very large datasets, ViT outperforms current SOTA

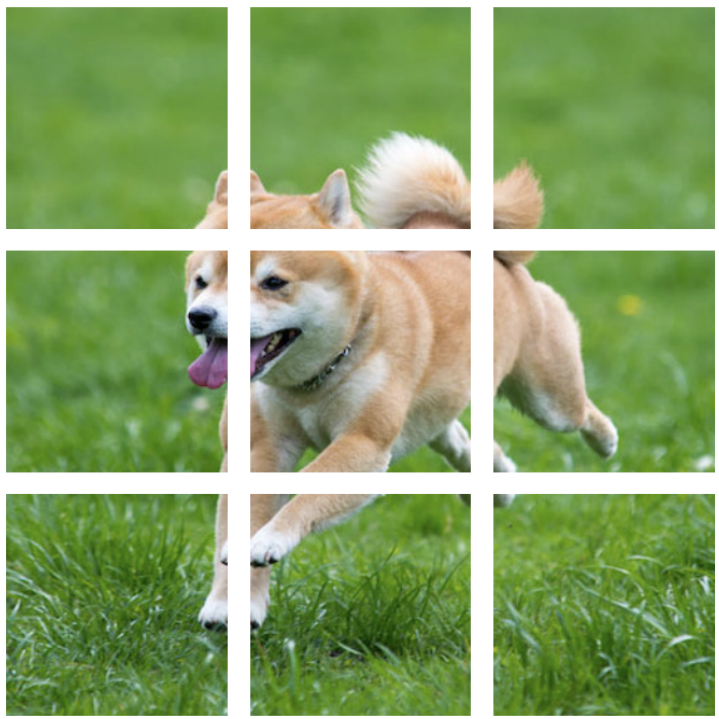
Image representation

- Problem: the standard Transformer typically receives as input sequence of n d-dimensional token/word embeddings, $\boldsymbol{x}^{(1:n)} \in \mathbb{R}^{n \times d}$, how can we handle inputs which are 2D images?

Image representation example

- Must specify a patch size and stride
  - Stride: how much the sliding window moves each time (smaller stride → larger number of patches → larger sequence length for Transformer)
  - In Dosovitskiy et al. [2020], the patches are non-overlapping with patch size 16x16 and stride 16x16

→

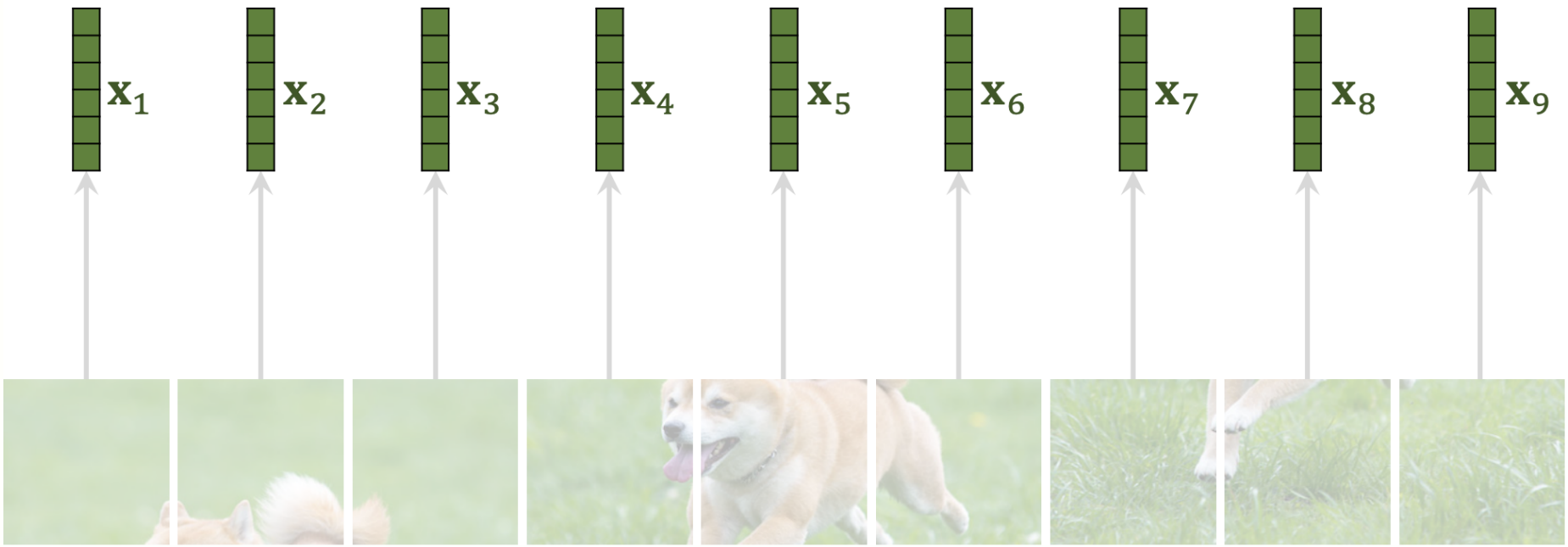


[13]

[13] Wang [2021] (wangshusen/DeepLearning)

Image representation example

- If the patches are $P_1 \times P_2 \times C$ tensors, then these flattened vectors are $(P_1 \cdot P_2 \cdot C) \times 1$ tensors



[14]

[14] Wang [2021] (wangshusen/DeepLearning)

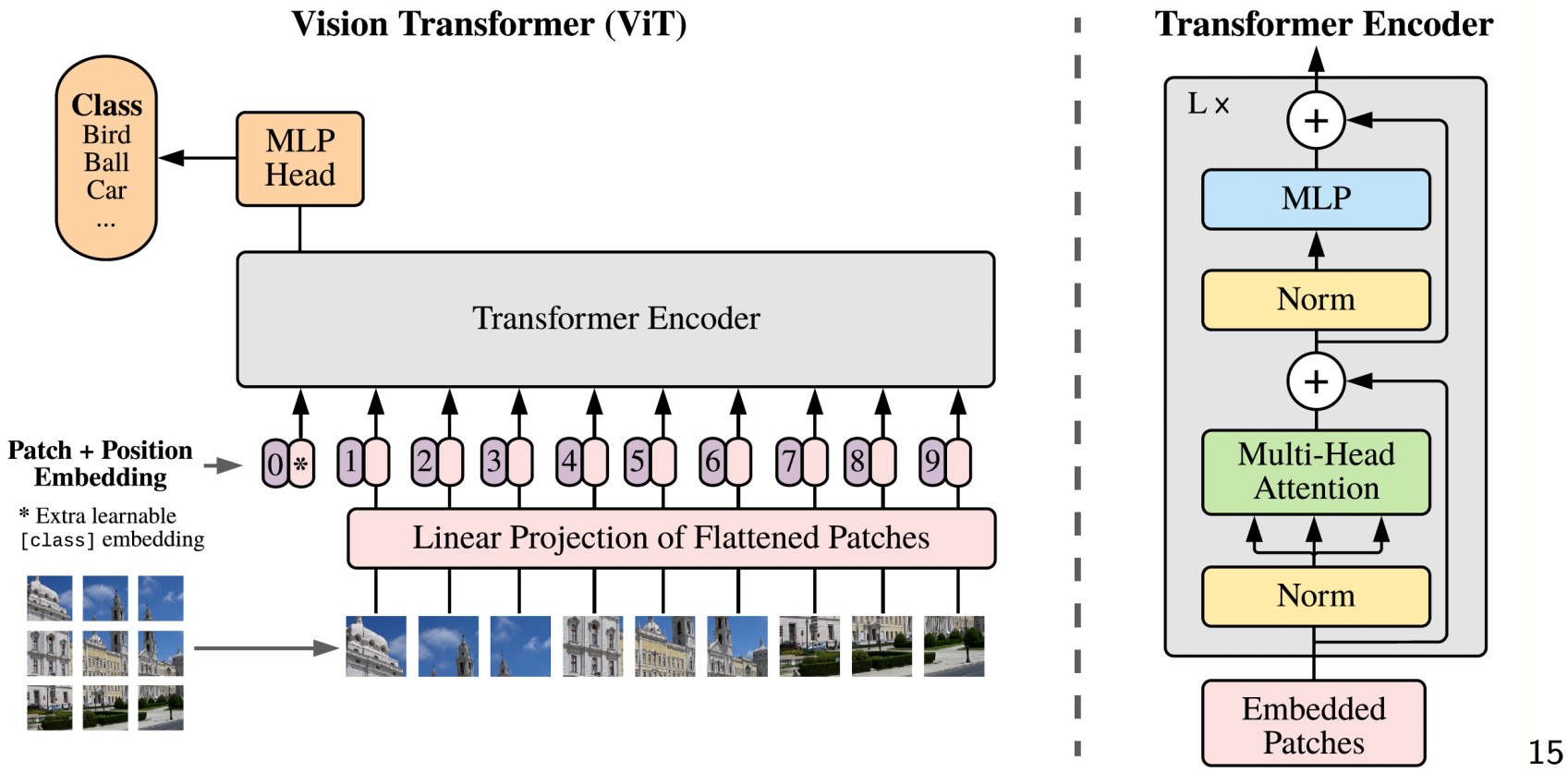
Image representation

- Problem: the standard Transformer typically receives as input sequence of n d-dimensional token/word embeddings, $\boldsymbol{x}^{(1:n)} \in \mathbb{R}^{n \times d}$, how can we handle inputs which are 2D images?
- Solution: we reshape an image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$ into a sequence of n flattened 2D patches $\boldsymbol{x}_p \in \mathbb{R}^{n \times (P_1 \cdot P_2 \cdot C)}$, where
  - (H, W) is the *resolution* of the 2D image
  - C is the *number of channels*
  - (P_1, P_2) is the *resolution of the image patch*
    - In Dosovitskiy et al. [2020], $P_1 = P_2 = P$
  - $n = \frac{H \cdot W}{P_1 \cdot P_2}$ (if we have non-overlapping patches)
    - n serves as the effective input sequence length for the transformer
- If d is the hidden dimension size of the Transformer network, we project the flattened patches into d dimensions using a trainable linear projection

Vision Transformer (ViT) architecture

- To use Transformers for images:
  1. Split an image into image patches
     - Image patches are treated the same way as tokens/words in an NLP application
  2. Use image patches as inputs to the Transformer encoder described in Vaswani et al. [2017]; Devlin et al. [2019] with fewest possible modifications:
- As with BERT [Devlin et al., 2019], ViT also prepends the sequence with a classification token, $\boldsymbol{z}_0^0 = \boldsymbol{x}_{\text{class}}$
  - In pre-training and fine-tuning, a classification head is attached to $\boldsymbol{y} = \boldsymbol{z}_L^0$ using a FFN
- Position encodings are added to the patch embeddings to retain positional information (standard learnable position embeddings)

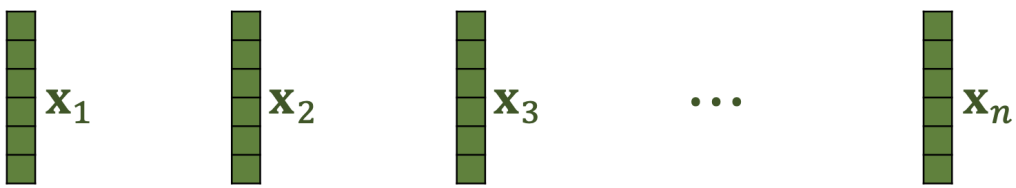
Vision Transformer (ViT) architecture



15

[15] Dosovitskiy et al. [2020, Figure 1]

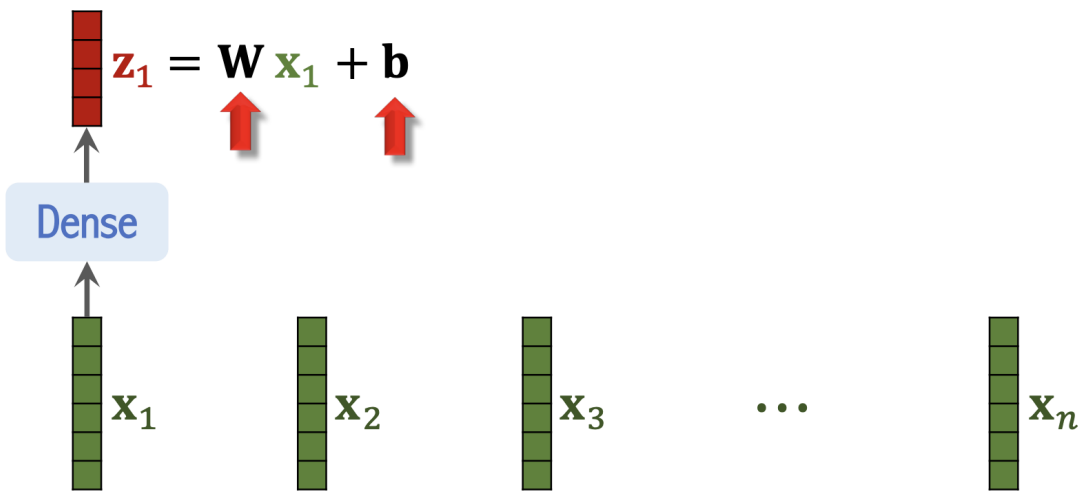
Vision Transformer illustration



16

[16] Wang [2021] (wangshusen/DeepLearning)

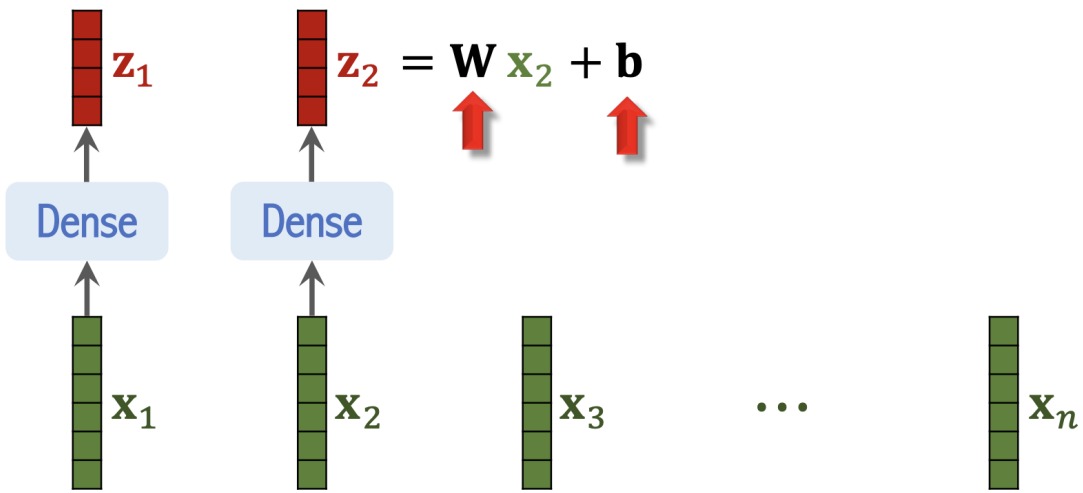
Vision Transformer illustration



17

[17] Wang [2021] (wangshusen/DeepLearning)

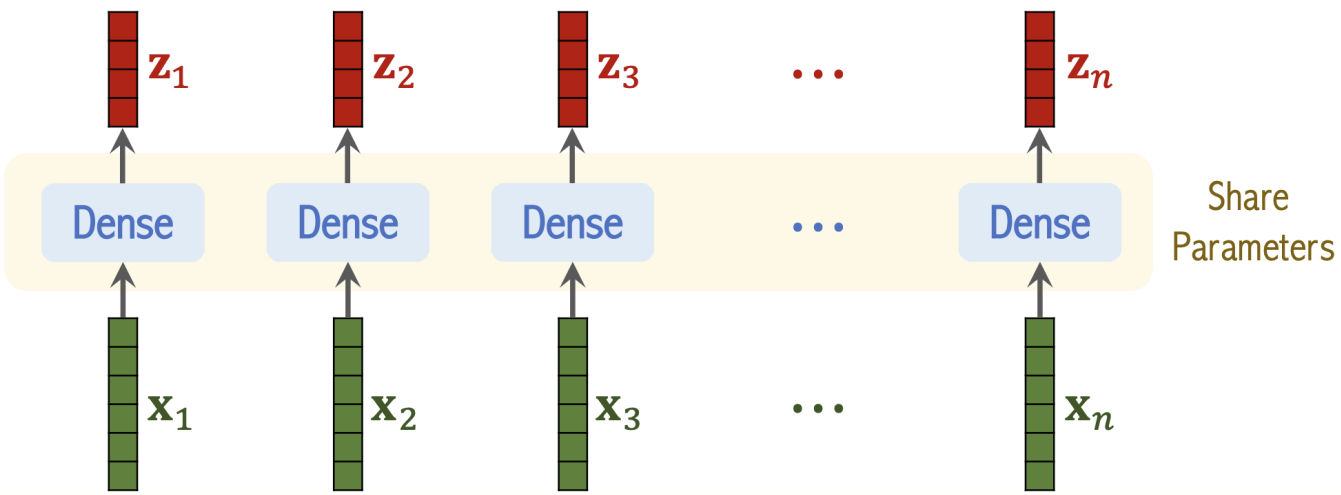
Vision Transformer illustration



18

[18] Wang [2021] (wangshusen/DeepLearning)

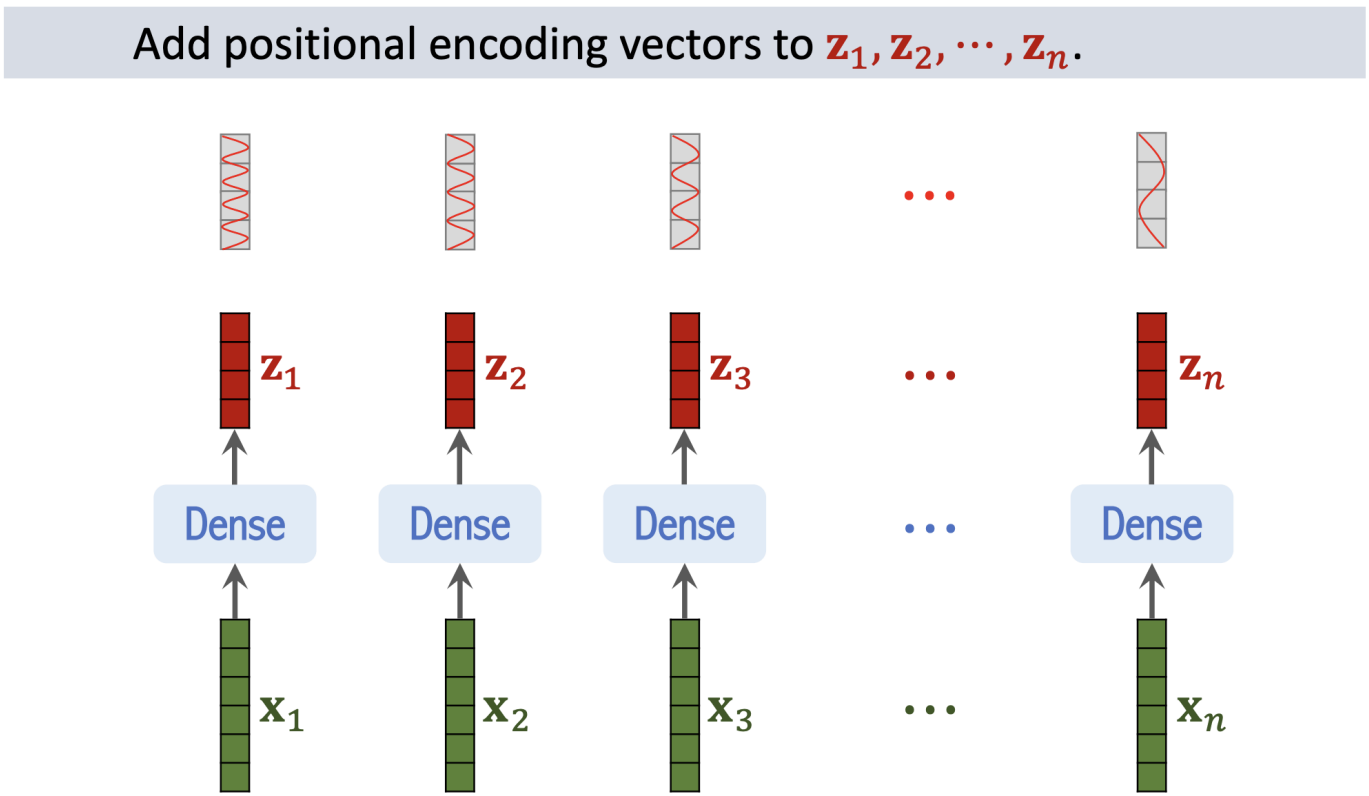
Vision Transformer illustration



19

[19] Wang [2021] (wangshusen/DeepLearning)

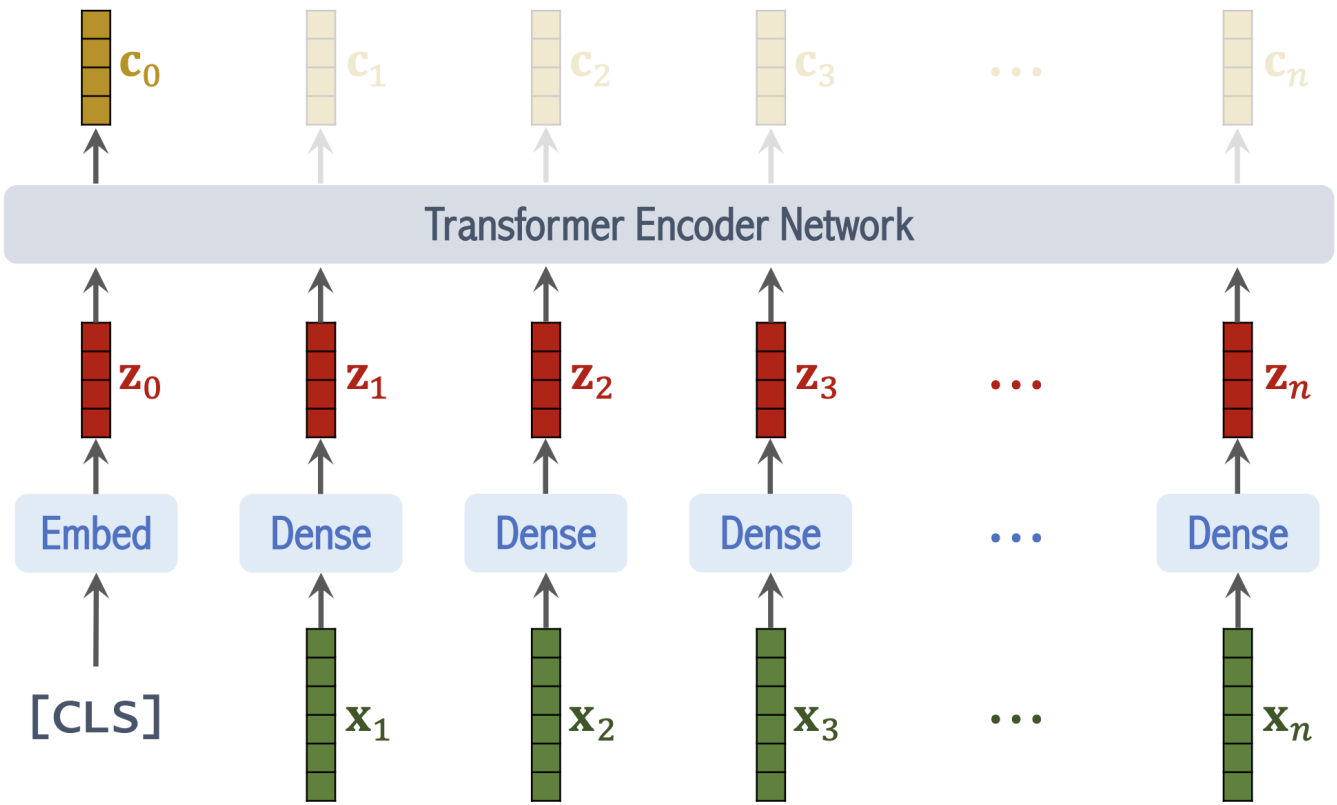
Vision Transformer illustration



20

[20] Wang [2021] (wangshusen/DeepLearning)

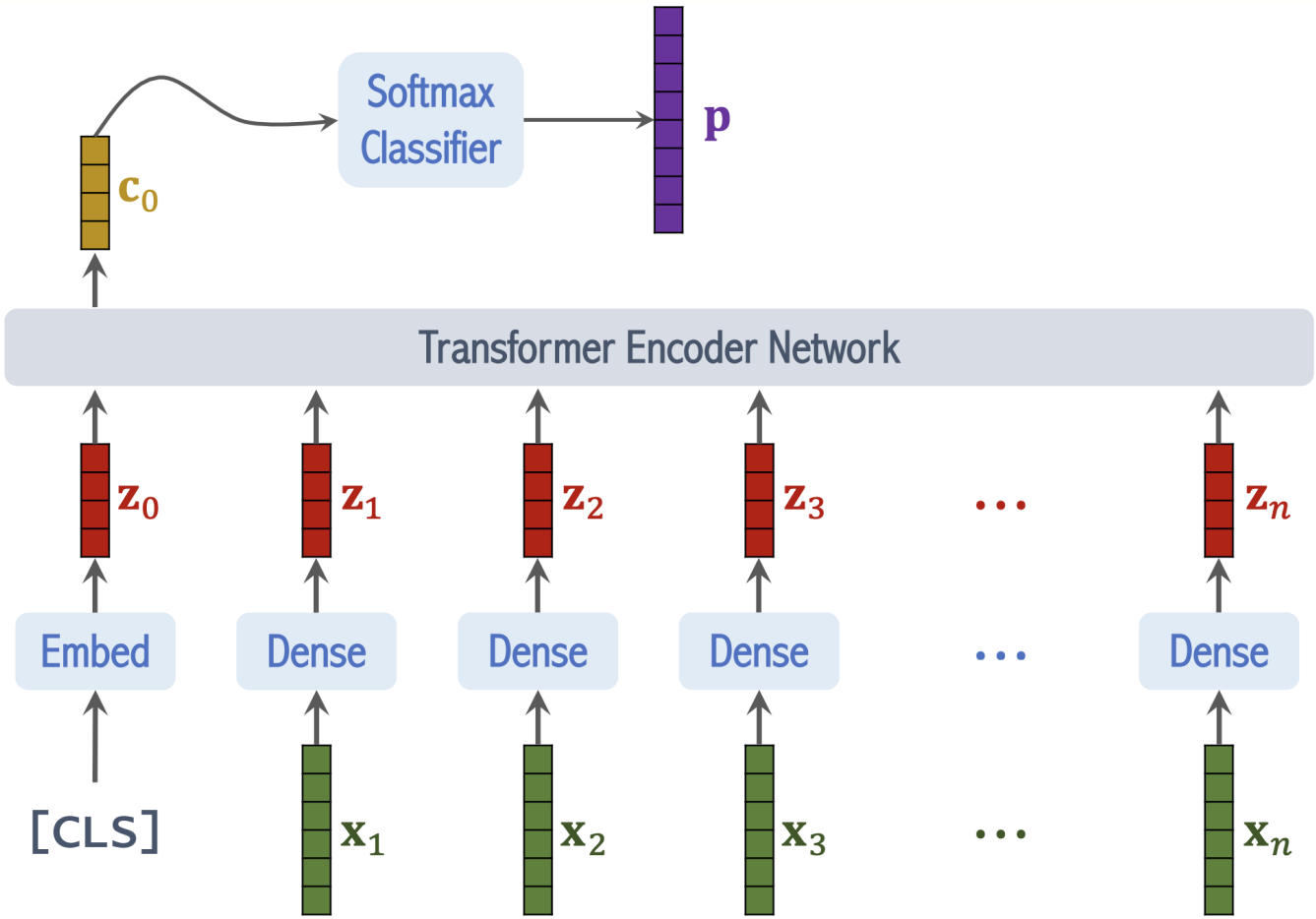
Vision Transformer illustration



21

[21] Wang [2021] (wangshusen/DeepLearning)

Vision Transformer illustration



22

[22] Wang [2021] (wangshusen/DeepLearning)

Fine-tuning and higher resolution

- ViT is pre-trained on large datasets and then fine-tuned to (smaller) application-specific datasets
- For fine-tuning, we replace the pre-trained prediction head with a $d \times k$ linear layer, where k is the number of classes in the fine-tuning task
- When dealing with images of higher resolution, we keep the patch size used in pre-training - resulting in larger effective sequence length
  - ViT can handle arbitrary sequence lengths (up to memory constraints), but pre-trained position embeddings may no longer be meaningful if we need to extrapolate
  - Perform 2D interpolation of pre-trained position embeddings if needed

Inductive bias note

- ViT has much less image-specific inductive bias (or learning bias) than CNNs $\implies$ ViT performs poorly (compared to CNNs) when there is insufficient pre-training data
- CNNs are famously equivariant to image translation, that is if the input changes, the output changes as well - equivalently to the input:
  - If we move an object or feature in the input image, the corresponding feature maps or activations at different layers in the CNN will also shift in the same manner
  - This property allows CNNs to recognise patterns and features irrespective of their specific location in the input image
  - CNNs can efficiently learn and generalise from spatially shifted instances of patterns or objects, making them well-suited for various computer vision tasks
- "With CNNs, locality, two-dimensional neighbourhood structure and translation equivariance are baked into each layer throughout the model" [Dosovitskiy et al., 2020]

Inductive bias note

- In ViT, the two-dimensional neighbourhood structure is used sparingly:
  1. Beginning when cutting image into patches
  2. Fine-tuning when adjusting for position embeddings for images of different resolution
- Position embeddings at initialisation time carry no information about the 2D positions of the patches
  - Spatial relations between patches have to be learned from scratch during pre-training
  - There is no in-built assumption about how these patches are related spatially
  - Makes ViT cheaper to train at large scale, but theoretically are less likely to generalise well with small data - higher chance of overfitting on small datasets
- Dosovitskiy et al. [2020] showed that with sufficient data, ViTs can overcome this lack of inductive bias and can actually outpeform SOTA

Experimental set up

- Pre-train on large image classification datasets (to image classification task):
  - ImageNet (small): 1.3M images, 1k classes
  - ImageNet-21k (medium): 14M images, 21k classes
  - JFT (large): 300M images, 18k classes
- Fine-tune and evaluate on a number of benchmark tasks
- Model variants:

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

[23]

- Explored using 14x14, 16x16 and 32x32 patch sizes
- Train on TPUv3 (tensor processing units) hardware - report the number of TPU-core-days taken to train them (i.e. number of cores used multiplied by training time in days)

[23] Dosovitskiy et al. [2020, Table 1]

Comparison to state-of-the-art

- Compare with SOTA ResNet but replace Batch Normalisation layers with Group Normalisation layers $\rightarrow$ ResNet (BiT) (Big transfer) [Kolesnikov et al., 2020]
- Also looked at "hybrid" approach: feed CNN feature maps into ViT with patch size of 1x1 (apply patch embedding projection to patches extracted from CNN rather than raw image)

Comparison to state-of-the-art

- When pre-trained on ImageNet-21k (medium), ViT has similar performance to ResNet, but at a significantly lower computational cost
- When pre-trained on JFT (large), ViT outperforms ResNet

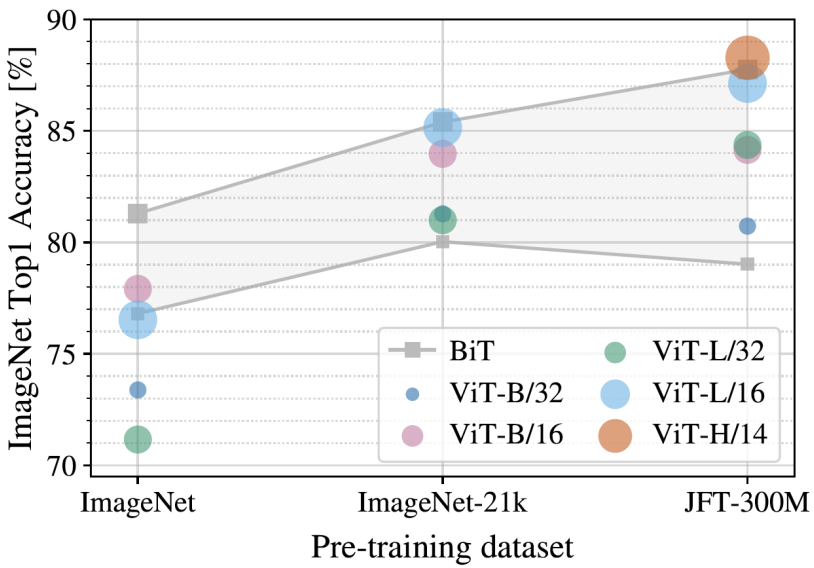
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

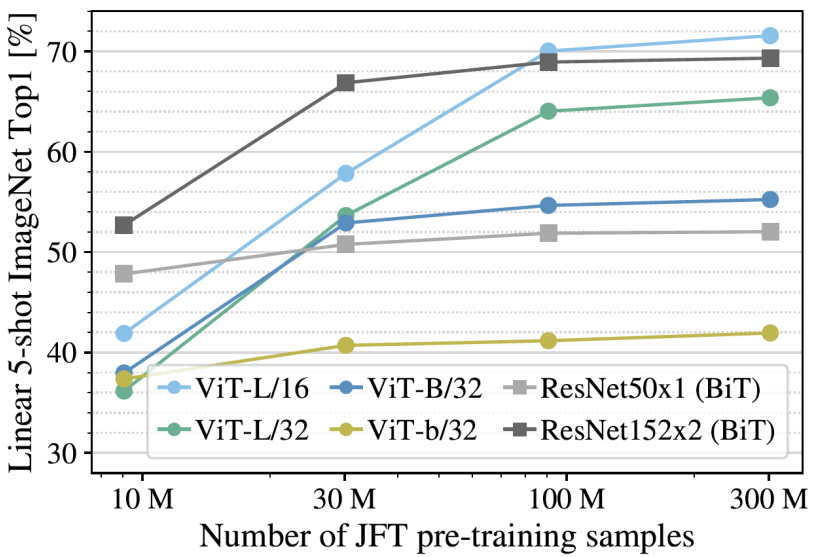
[24]

[24] Dosovitskiy et al. [2020, Table 2]

How much data do ViTs need?

- Pre-train ViT and ResNet on datasets of varying size and fine-tune & evaluate on ImageNet
  - ViTs tend to overfit more than ResNets when pre-training on smaller datasets - reinforces intuition that convolutional inductive bias is useful for small datasets
  - For larger datasets, learning patterns directly from data is sufficient
  - ResNet performance plateaus sooner than ViT





25

[25] Dosovitskiy et al. [2020, Figure 3]

Self-supervision learning for ViT

- Transformers success in NLP stems from their excellent scalability and being able to harness huge amounts of unlabelled text corpora
- In Dosovitskiy et al. [2020] pre-trained and fine-tuned ViT for image classification
  - But also gave preliminary results on pre-training on masked patch prediction for self-supervision (mimicking the masked language modelling task in BERT [Devlin et al., 2019])
  - Achieved 79.9% accuracy in ImageNet
    - 2% better than training from scratch $\rightarrow$ positive benefit from self-supervised pre-training
    - 4% worse than supervised pre-training $\rightarrow$ room for improvement
- See Dosovitskiy et al. [2020, Appendix B.1.2.] for full experimental details

ViT Summary

- Dosovitskiy et al. [2020] explores the direct application of Transformer encoders to image classification
- Interprets images as a sequence of patches and processes them using a standard Transformer encoder as in Vaswani et al. [2017]
- Simple strategy works well when pre-training with large datasets
  - Outpeforms SOTA in many image classification benchmarks whilst being relatively cheap to pre-train (see Dosovitskiy et al. [2020, Section 4.4, Figure 5] for a scaling study comparing transfer learning performance vs total pre-training compute)
- Next steps:
  - Further exploration of self-supervised pre-training methods (e.g. masked patch prediction)
  - Explore use of ViT in other computer vision tasks, e.g. segmentation, detection

References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*.

Fukushima, K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 36.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2020). Big transfer (BiT): General visual representation learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 491–507. Springer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). GradientBased Learning Applied to Document Recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation . *arXiv preprint arXiv:1505.04597*.

Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition . *arXiv preprint arXiv.1409.1556*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30.

Wang, S. (2021). Vision Transformer for Image Classification. https://youtu.be/HZ4j˙U3FC94.