Vision Transformers

An image is worth 16x16 words

Motivation

- Revolutionary paper which found a way to adapt conventional transformer architecture to computer vision tasks
- Previously transformers were predominantly used for NLP related tasks
- ‘Pure transformer’ → no convolutions

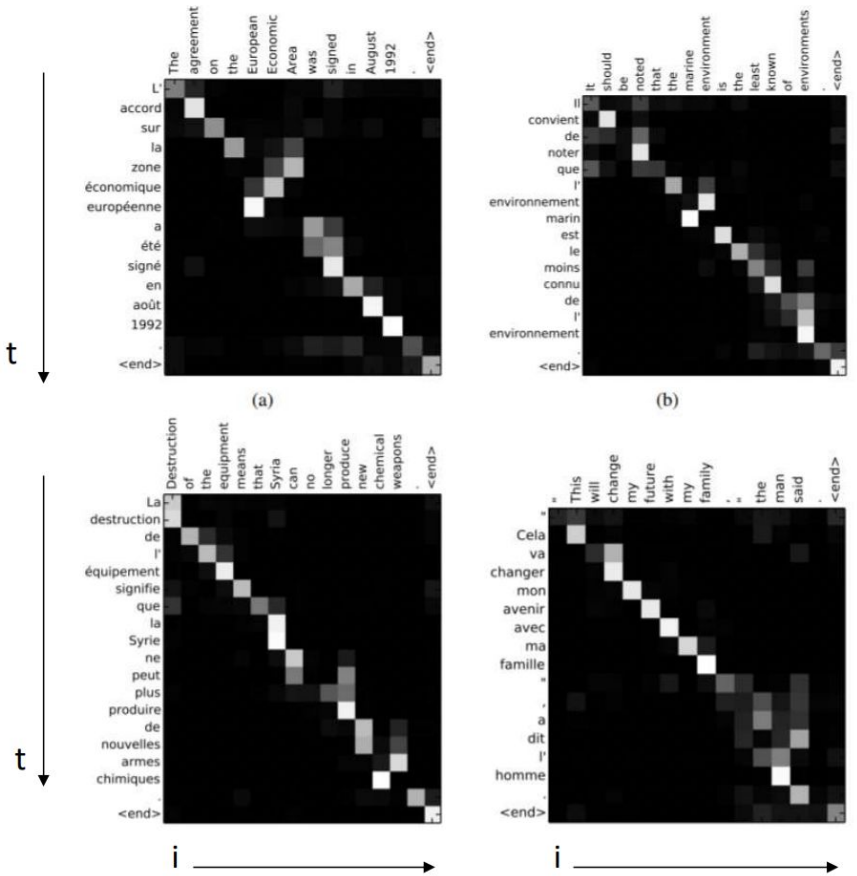
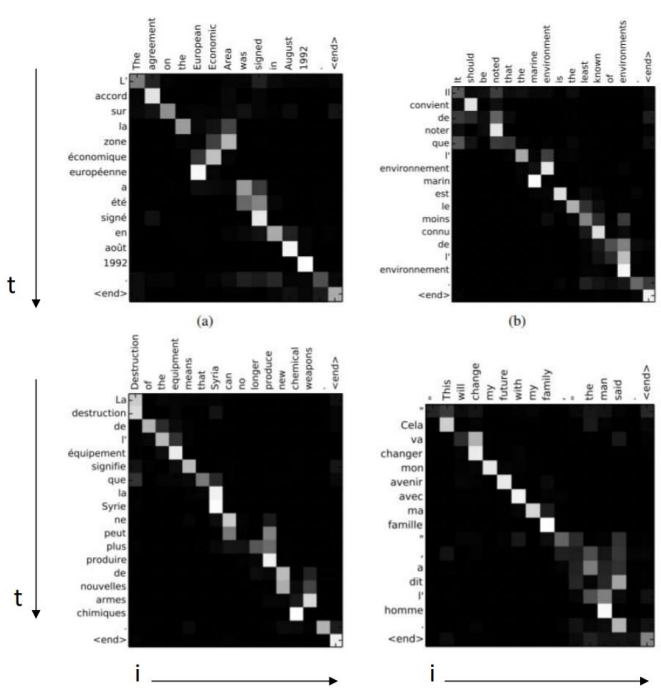


Image source: https://doi.org/10.48550/arXiv.1409.0473

Significance



How?

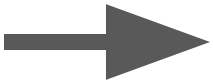


Image source: https://doi.org/10.48550/arXiv.1409.0473

Image source: https://doi.org/10.48550/arXiv.2010.11929

Adapting conventional transformers

- Transformers were traditionally used where Input is a sequence of tokens.
- The attention mechanism is $O(n^2)$ in the number of tokens
- If you were to naively apply this to images treating each pixel as a token the computational cost makes attention infeasible.
- Therefore the naive approach of each pixel attending to every other pixel wouldn't work

Image source: https://doi.org/10.48550/arXiv.1706.03762

Solution

1. Split an image into fixed size patches
2. Linearly embed each of them
3. Add position embeddings
4. Feed the resulting sequence into a standard transformer encoder
5. Extra learnable ‘classification token’

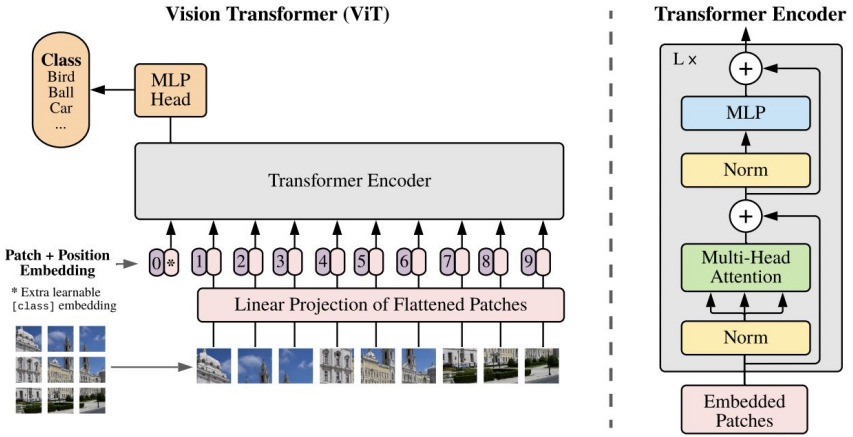


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable “classification token” to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

Image source: https://doi.org/10.48550/arXiv.2010.11929

Solution

- Split each image into 16x16 patches
- Flatten each patch into a vector, pass through a Linear layer to generate an embedding
- Steps 1 & 2 of the overall process.

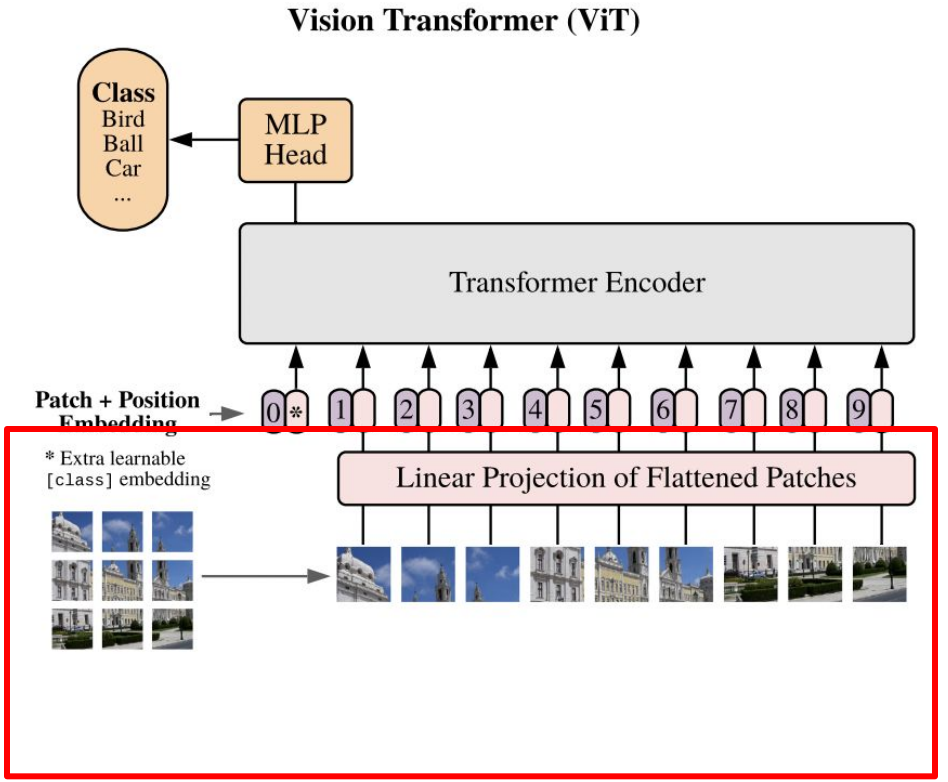


Image source: https://doi.org/10.48550/arXiv.2010.11929

Solution

- Combine linear embedding with position embedding

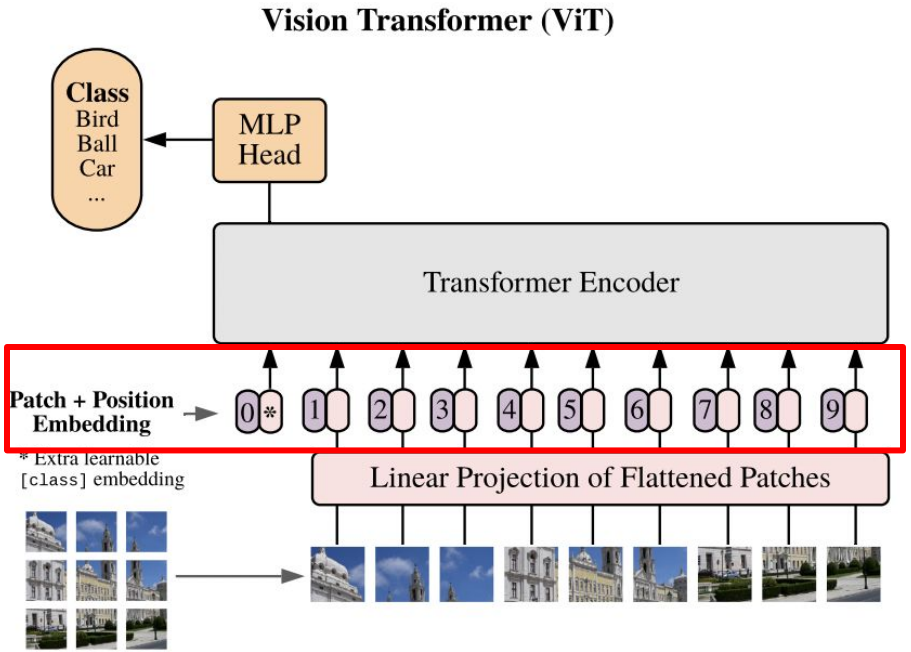


Image source: https://doi.org/10.48550/arXiv.2010.11929

Solution

- Standard transformer encoder (BERT)

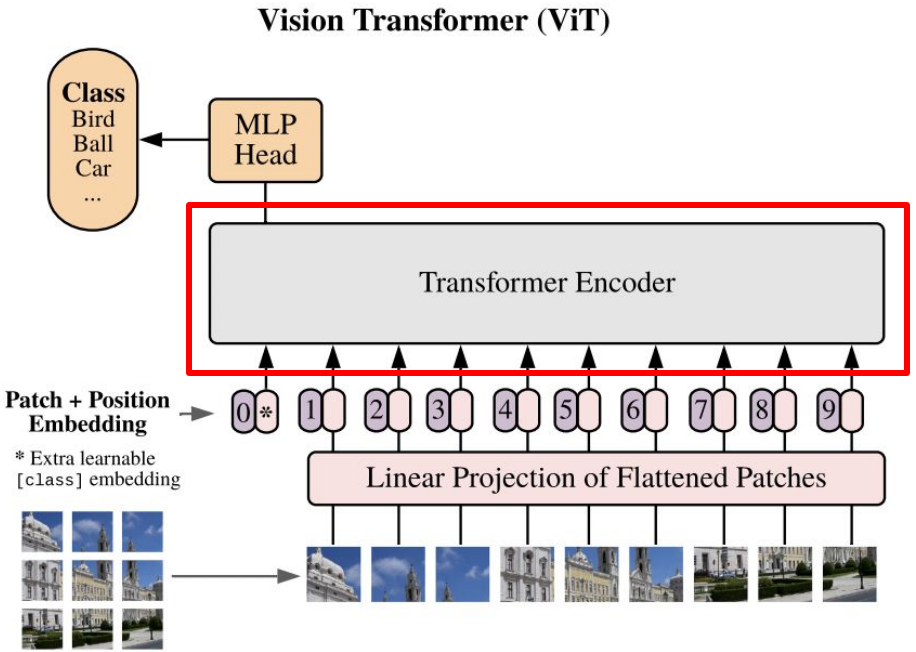


Image source: https://doi.org/10.48550/arXiv.2010.11929

Solution

- Use Extra learnable ‘classification token’ to perform classification task
- Learns a general embedding of the whole image which is then used in classification

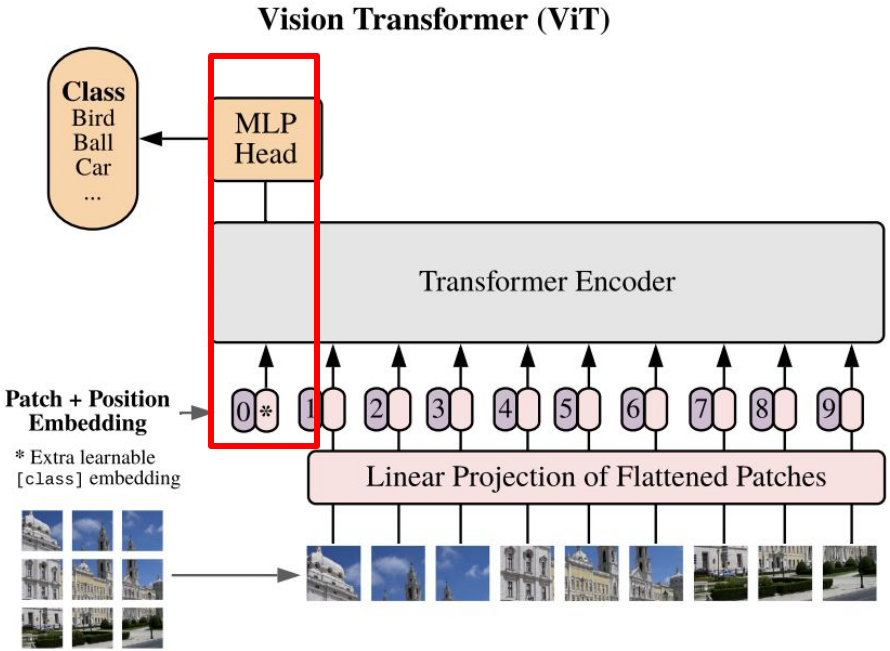


Image source: https://doi.org/10.48550/arXiv.2010.11929

Did it work?

	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

Image source: https://doi.org/10.48550/arXiv.2010.11929

Transformers or CNNs?

model	image size	FLOPs	throughput (image / s)	IN-1K / 22K trained, 1K acc.
○ Swin-T	224^2	4.5G	1325.6	81.3 / –
● ConvNeXt-T	224^2	4.5G	**1943.5** (+47%)	**82.1** / –
○ Swin-S	224^2	8.7G	857.3	83.0 / –
● ConvNeXt-S	224^2	8.7G	**1275.3** (+49%)	**83.1** / –
○ Swin-B	224^2	15.4G	662.8	83.5 / 85.2
● ConvNeXt-B	224^2	15.4G	**969.0** (+46%)	**83.8 / 85.8**
○ Swin-B	384^2	47.1G	242.5	84.5 / 86.4
● ConvNeXt-B	384^2	45.0G	**336.6** (+39%)	**85.1 / 86.8**
○ Swin-L	224^2	34.5G	435.9	– / 86.3
● ConvNeXt-L	224^2	34.4G	**611.5** (+40%)	**84.3 / 86.6**
○ Swin-L	384^2	103.9G	157.9	– / 87.3
● ConvNeXt-L	384^2	101.0G	**211.4** (+34%)	85.5 / **87.5**
● ConvNeXt-XL	224^2	60.9G	**424.4**	– / **87.0**
● ConvNeXt-XL	384^2	179.0G	**147.4**	– / **87.8**

Image source: https://doi.org/10.48550/arXiv.2201.03545