Transformers #2: Transformers and Vision

9 March 2021
Federico Baldassarre
Announcements

- 23 March
  - Transformers for NLP
  - Youssef

- 6 April
  - Transformers application, other domains and alternatives
  - Yonk and Sofia
Outline

● Main paper
  ○ An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (Vision Transformer)
  ○ Discussion

● Secondary papers
  ○ End-to-End Object Detection with Transformers (DETR)
  ○ Discussion
  ○ Generative Pretraining from Pixels (iGPT)
  ○ Discussion

● General comments and discussion
An Image is Worth 16x16 Words:
Transformers for Image Recognition at Scale

Dosovitskiy, Beyer, Kolesnikov, Weissenborn, Zhai, Unterthiner, Dehghani, Minderer, Heigold, Gelly, Uszkoreit, Houlsby (Google Research)

Published at ICLR 2021 (Oral)
Motivation

- From NLP: large-scale pre-training of Transformers
- Convolution is an established inductive bias. Can attention replace it?
- Related works using attention require specialized architectures
Approach: the Vision Transformer

\[ y = \text{LN}(z^0_L) \]

\[ z'_\ell = \text{MSA}(\text{LN}(z_{\ell-1})) + z_{\ell-1} \]

\[ z_\ell = \text{MLP}(\text{LN}(z'_\ell)) + z'_\ell, \]

\[ z_0 = [x_{\text{class}}; x^1_p E; x^2_p E; \ldots; x^N_p E] + E_{\text{pos}} \]
Image as 16x16 patches

- 16x16 patches → sequence of tokens
- Weak 2D locality prior (bias #1)
- Convolution hiding in plain sight?

[\[x_1^p E; x_2^p E; \cdots; x_N^p E\]]
Reintroducing spatial information

- Learned embedding for each patch position
- For inference at higher resolutions, embeddings are interpolated in a 2D grid (bias #2)
**Embedding interpolation**

Pre-training embeddings 3x4 = 12

Fine-tuning embeddings 4x5 = 20
Almost a standard Transformer

- Sequence: image patches + classification token
- Output class is predicted from the classification token
- During fine-tuning, the MLP head is replaced
Experiments

General setting:
- Pre-train on a large-scale supervised classification task (low resolution)
- Fine-tune on a specific classification task (high resolution)

Research questions:
- Comparison with CNNs (training cost, transfer accuracy)
- Interplay between convolutional bias and compute budget
- Inspection of learned weights and typical attention maps
Datasets

Pre-training

- ImageNet
  - 1k classes
  - 1.3M images

- ImageNet-21k
  - 21k classes
  - 14M images

- JFT
  - 18k classes
  - 303M images
  - Private Google dataset

Fine-tuning

- CIFAR 10/100
- ImageNet
- Oxford Pets/Flowers
- VTAB 19-task suite

Pre-training uses a lower resolution than fine-tuning. Both are supervised.
Comparison with CNNs

“Small” pre-train datasets
ResNet $>$ Transformer

Larger pre-train datasets
Transformer $\approx$ ResNet
Saturation for bigger Vision Transformers not yet observed
<table>
<thead>
<tr>
<th></th>
<th>Ours-JFT (ViT-H/14)</th>
<th>Ours-JFT (ViT-L/16)</th>
<th>Ours-I21K (ViT-L/16)</th>
<th>BiT-L(^{\wedge}) (ResNet152x4)</th>
<th>Noisy Student (EfficientNet-L2)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ImageNet</strong></td>
<td>88.55 ± 0.04</td>
<td>87.76 ± 0.03</td>
<td>85.30 ± 0.02</td>
<td>87.54 ± 0.02</td>
<td>88.4/88.5*</td>
</tr>
<tr>
<td><strong>TPUv3-core-days</strong></td>
<td>2.5k</td>
<td>0.68k</td>
<td>0.23k</td>
<td>9.9k</td>
<td>12.3k</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td>632M</td>
<td>307M</td>
<td>307M</td>
<td>928M</td>
<td></td>
</tr>
</tbody>
</table>

\(^{\wedge}\) BiT-L is trained on JFT: [Big Transfer (BiT): General Visual Representation Learning](https://arxiv.org/abs/2103.02032), Kolesnikov et al.
Finally, the ViT-L/16 model pre-trained on the public ImageNet-21k dataset performs well on most datasets too, while taking fewer resources to pre-train: it could be trained using a standard cloud TPUv3 with 8 cores in approximately 30 days.

<table>
<thead>
<tr>
<th>TPU type (v2)</th>
<th>v2 cores</th>
<th>Total memory</th>
<th>On-demand price (USD)</th>
<th>Preemptible price (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>v2-8</td>
<td>8</td>
<td>64 GiB</td>
<td>$3,285 / month</td>
<td>$986 / month</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TPU type (v3)</th>
<th>v3 cores</th>
<th>Total memory</th>
<th>On-demand price (USD)</th>
<th>Preemptible price (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3-8</td>
<td>8</td>
<td>128 GiB</td>
<td>$5,840 / month</td>
<td>$1,752 / month</td>
</tr>
</tbody>
</table>

TPU prices only, virtual machine billed separately: [Cloud TPU pricing](#)
Convolutional bias vs. compute budget

Vision Transformer vs. Big Transfer
- Given similar budget, Transformers perform generally better than CNNs

Hybrid vs. pure transformer
- In low-compute regime, the convolutional bias helps
- With enough budget, the bias becomes unnecessary.

All models are pre-trained on JFT
Discussion: just add data?

Our paper is an attempt to put the focus back on the data. The models seem to be plateauing but when it comes to the performance with respect to data – but modest performance improvements are still possible for exponential increases of the data. Another major finding of our paper is that having better models is not leading to substantial gains because ImageNet is no more sufficient to use all the parameters or their representational power.
Model inspection

RGB embedding filters (first 28 principal components)

Learned linear projection filters resemble low-level filters in CNNs

Learned position embeddings encode 2D information
Learned attention patterns match the typical CNN receptive fields.

Attention maps of the classification head have (seemingly) learned to localize objects.
Conclusions

With large amounts of data:

- The convolutional bias can be dropped
- Weak 2D biases remain necessary
- Transformers are more efficient than CNNs

Future work:

- Other computer vision tasks, e.g. DETR
- ViT remains fully-supervised, see iGPT
Vision Transformer

Discussion
End-to-End Object Detection with Transformers

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, Sergey Zagoruyko (Facebook AI)

Published at ECCV 2020
Ideas

- Object detection can be seen as a set-to-set problem
- Transformers are efficient set-to-set architectures
Goal: simplicity

Faster R-CNN:
- Several components
- Detectron2 codebase is well-written but complex

DETR:
- Straightforward set-to-set prediction
- Off-the-shelf Transformer layers

Feature extraction

- Transformer encoder: a stack of self-attention layers
- Features from a CNN backbone
- 2D positional encoding
Set-to-set prediction

- Transformer decoder: object queries attend to image patches
- Parallel decoding (all at once, not autoregressive)
Matching loss

- Bipartite matching between predictions and ground-truth boxes
- Loss is a combination of object classification and box regression
## Results

<table>
<thead>
<tr>
<th>Model</th>
<th>GFLOPS/FPS</th>
<th>#params</th>
<th>AP</th>
<th>AP$_{50}$</th>
<th>AP$_{75}$</th>
<th>AP$_{S}$</th>
<th>AP$_{M}$</th>
<th>AP$_{L}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster RCNN-DC5</td>
<td>320/16</td>
<td>166M</td>
<td>39.0</td>
<td>60.5</td>
<td>42.3</td>
<td>21.4</td>
<td>43.5</td>
<td>52.5</td>
</tr>
<tr>
<td>Faster RCNN-FPN</td>
<td>180/26</td>
<td>42M</td>
<td>40.2</td>
<td>61.0</td>
<td>43.8</td>
<td>24.2</td>
<td>43.5</td>
<td>52.0</td>
</tr>
<tr>
<td>Faster RCNN-R101-FPN</td>
<td>246/20</td>
<td>60M</td>
<td>42.0</td>
<td>62.5</td>
<td>45.9</td>
<td>25.2</td>
<td>45.6</td>
<td>54.6</td>
</tr>
<tr>
<td>Faster RCNN-DC5+</td>
<td>320/16</td>
<td>166M</td>
<td>41.1</td>
<td>61.4</td>
<td>44.3</td>
<td>22.9</td>
<td>45.9</td>
<td>55.0</td>
</tr>
<tr>
<td>Faster RCNN-FPN+</td>
<td>180/26</td>
<td>42M</td>
<td>42.0</td>
<td>62.1</td>
<td>45.5</td>
<td>26.6</td>
<td>45.4</td>
<td>53.4</td>
</tr>
<tr>
<td>Faster RCNN-R101-FPN+</td>
<td>246/20</td>
<td>60M</td>
<td>44.0</td>
<td>63.9</td>
<td><strong>47.8</strong></td>
<td><strong>27.2</strong></td>
<td>48.1</td>
<td>56.0</td>
</tr>
<tr>
<td>DETR</td>
<td>86/28</td>
<td>41M</td>
<td>42.0</td>
<td>62.4</td>
<td>44.2</td>
<td>20.5</td>
<td>45.8</td>
<td>61.1</td>
</tr>
<tr>
<td>DETR-DC5</td>
<td>187/12</td>
<td>41M</td>
<td>43.3</td>
<td>63.1</td>
<td>45.9</td>
<td>22.5</td>
<td>47.3</td>
<td>61.1</td>
</tr>
<tr>
<td>DETR-R101</td>
<td>152/20</td>
<td>60M</td>
<td>43.5</td>
<td>63.8</td>
<td>46.4</td>
<td>21.9</td>
<td>48.0</td>
<td>61.8</td>
</tr>
<tr>
<td>DETR-DC5-R101</td>
<td>253/10</td>
<td>60M</td>
<td><strong>44.9</strong></td>
<td><strong>64.7</strong></td>
<td>47.7</td>
<td>23.7</td>
<td><strong>49.5</strong></td>
<td><strong>62.3</strong></td>
</tr>
</tbody>
</table>

- Competitive with highly-optimized Faster R-CNN models
- Detection of small objects needs to be improved (likely with a FPN)
Encoder self-attention inspection

- Self-attention weights resemble object masks
- Encoder already builds a representation for object detection
Decoder object queries inspection

Where are the object boxes that are predicted by each object query?

- Specialised for certain sizes at certain locations
- “Is there an object here?”

Green: small boxes
Red: large horizontal boxes
Blue: large vertical boxes
Conclusions

Novel approach for object detection

- Set-to-set transformers with matching loss
- Drop hand-engineered components
- Attention maps might be useful for inspecting the model and panoptic segmentation
DETR

Discussion
Generative Pretraining from Pixels

Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, Ilya Sutskever (OpenAI)

Published at ICML 2020
Ideas

- Pixel-by-pixel image generation using an autoregressive transformer
- Self-supervised pre-training, no labels needed
- Evaluate learned representation using linear probes
Next-pixel prediction

- Each pixel is a token
- Resolution limitations
- Reduced color palette

- Autoregressive architecture
- Standard sequence modeling objective

\[ p(x) = \prod_{i=1}^{n} p(x_{\pi_i} | x_{\pi_1}, \ldots, x_{\pi_{i-1}}, \theta) \]

\[ L_{AR} = \mathbb{E}_{x \sim X} [-\log p(x)] \]
Linear probe

- Average over the latent codes of each pixel in one layer

\[ f^l = \langle n^l_i \rangle_i \]

- Train a linear classifier over the layer representation
Datasets

Self-supervised pre-training

- ImageNet
  - 1.2M images
  - Low resolution
  - 9-bit color palette
  - No labels

- Web images
  - 100M images
  - Low resolution
  - 9-bit color palette
  - No labels

Supervised linear probing

- CIFAR-10
- CIFAR-100
- STL-10
- ImageNet

Pre-training and fine-tuning use the same resolution. For most experiments 32x32, otherwise 48x48, or 64x64.
Representation quality by layer

- Best representations in the middle
- Authors’ hypothesis: iGPT behaves similarly to an autoencoder, but without the bottleneck
Representation quality by model size

- Horizontal axis: larger models are better generators
- Correlation: generation performance and probe accuracy
- Generation performance being equal, larger models learn more discriminative features

* Note the different sizes of dim model. In iGPT-L, the linear probe has 3x more values to work with.
# State-of-the-art accuracies

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
<th>STL-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-152</td>
<td>94</td>
<td>78.0</td>
<td>94.2</td>
</tr>
<tr>
<td>SimCLR</td>
<td>95.3</td>
<td>80.2</td>
<td></td>
</tr>
<tr>
<td>iGPT-L</td>
<td>96.3</td>
<td>82.8</td>
<td>95.5</td>
</tr>
</tbody>
</table>

- **iGPT linear probe accuracy**

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
<th>STL-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoAugment</td>
<td>98.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SimCLR</td>
<td>98.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPipe</td>
<td>99.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iGPT-L</td>
<td>99.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **iGPT fine-tuning accuracy**

iGPT is pre-trained on ImageNet (unlabeled and downsampled). Sup transfer means pre-trained on ImageNet with labels. iGPT-L is bigger and more expensive than the other models.
Conclusions

Confirm that

● Self-supervised pixel-wise pre-training can learn good representations
● 2D inductive biases can be abandoned (no patches, no conv layer at the input)

However:

● Abandoning 2D priors causes scaling issues at higher resolutions
● Humans do not “see” the world one pixel at the time, row-by-row. Is this the best kind of self-supervision to learn meaningful representations?
iGPT

Discussion
General discussion
General discussion

Adapting Transformers for vision, approaches:

- Features from a CNN become tokens (DETR)
- Pixel patches become tokens (ViT)
- Quantized pixels become tokens (iGPT)

Fewer biases, More compute
General discussion

Comparison with CNNs:

- Can attention layers generalize convolutional layers?
  - Translational equivariance
  - Sparsity of connections
  - Locality
  - Positional embedding

- Are Transformers more computationally efficient than ConvNets? Why? Is it related to how GPUs kernels work?
General discussion

Research trends and future directions:

● Making architectures more general
● Removing inductive biases
● Training on huge datasets
● Is this the way to go?

Related read: The Bitter Lesson, Sutton, 2019
Thanks for the attention!
Extra slides
## ViT state-of-the-art comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ours-JFT (ViT-H/14)</th>
<th>Ours-JFT (ViT-L/16)</th>
<th>Ours-I21K (ViT-L/16)</th>
<th>BiT-L (ResNet152x4)</th>
<th>Noisy Student (EfficientNet-L2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet</td>
<td>88.55 ± 0.04</td>
<td>87.76 ± 0.03</td>
<td>85.30 ± 0.02</td>
<td>87.54 ± 0.02</td>
<td>88.4/88.5*</td>
</tr>
<tr>
<td>ImageNet ReaL</td>
<td>90.72 ± 0.05</td>
<td>90.54 ± 0.03</td>
<td>88.62 ± 0.05</td>
<td>90.54</td>
<td>90.55</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>99.50 ± 0.06</td>
<td>99.42 ± 0.03</td>
<td>99.15 ± 0.03</td>
<td>99.37 ± 0.06</td>
<td>-</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>94.55 ± 0.04</td>
<td>93.90 ± 0.05</td>
<td>93.25 ± 0.05</td>
<td>93.51 ± 0.08</td>
<td>-</td>
</tr>
<tr>
<td>Oxford-IIT Pets</td>
<td>97.56 ± 0.03</td>
<td>97.32 ± 0.11</td>
<td>94.67 ± 0.15</td>
<td>96.62 ± 0.23</td>
<td>-</td>
</tr>
<tr>
<td>Oxford Flowers-102</td>
<td>99.68 ± 0.02</td>
<td>99.74 ± 0.00</td>
<td>99.61 ± 0.02</td>
<td>99.63 ± 0.03</td>
<td>-</td>
</tr>
<tr>
<td>VTAB (19 tasks)</td>
<td>77.63 ± 0.23</td>
<td>76.28 ± 0.46</td>
<td>72.72 ± 0.21</td>
<td>76.29 ± 1.70</td>
<td>-</td>
</tr>
<tr>
<td>TPUv3-core-days</td>
<td>2.5k</td>
<td>0.68k</td>
<td>0.23k</td>
<td>9.9k</td>
<td>12.3k</td>
</tr>
</tbody>
</table>

### Accuracy [%]

- **ViT-H/14**
- **BiT-L (R152x4)**
- **VIVI-Ex-100% (R50x3)**
- **S4L (R50x1)**

- VTAB (19 tasks)
- Natural (7 tasks)
- Specialized (4 tasks)
- Structured (8 tasks)
ViT: model inspection

RGB embedding filters
(first 28 principal components)

Position embedding similarity

ViT-L/16

Mean attention distance (pixels)

Network depth (layer)
ViT: small architectural difference

“Attention Is All You Need”, Vaswani et al.

“Vision Transformer”, Dosovitskiy et al.
DETR: panoptic segmentation

- For each detected object, the attention map of the corresponding object query can be used as the input to a segmentation model
- Can be trained jointly with the object detector or later
- Performs well on COCO (53 stuff classes, 80 object classes)
iGPT: autoregressive (GPT) vs masking (BERT)