An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

By: Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby

Kishore Kumar Alajangi, James Dwyer, Michael Fielder, Joseph Rivera
CAP 6412
Spring 2022
Outline

1. Introduction
2. Objectives
3. Related Work
4. Method
5. Experiments
6. Conclusion
Introduction

- Using transformers for Natural Language Processing (NLP) involves pre-training and fine-tuning
- Previous use of transformers for computer vision lacked scalability and generalization
- Pure transformer applied to image patches – similar to NLP, but with patches instead of words
- Transformer pre-trained on huge (14 million to 300 million images) datasets
Objectives

- Apply transformers used for Natural Language Processing to computer vision problems - Vision Transformer (ViT)
- Utilize transformers without relying on CNNs
- Demonstrate results with a number of models and datasets
- Achieve comparable or improved results with transformers compared to CNNs
- Lower the computational complexity
Related Work

- Recurrent Neural Networks first handled sequence data, then LSTMs
  - Slow training
  - Limited memory i.e. small window size
- Transformers in NLP Tasks
  - *Attention Is All You Need* introduced the transformer in 2017
  - Transformers allow parallel training of longer sequences
  - BERT and GPT serve as the inspiration from NLP tasks
Comparison to State of the Art on ImageNet

- Convolutional Neural Networks dominated classification tasks until recently
- Transformers have recently overtaken state of the art with caveats like extra data or hybrid models

<table>
<thead>
<tr>
<th>Model Category</th>
<th>Model Name</th>
<th>ImageNet Top-1 Accuracy</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td>ViT-H/14</td>
<td>88.55%*</td>
<td>&quot;AN IMAGE IS WORTH 16X16 WORDS...&quot; (ICLR, 2021)</td>
</tr>
<tr>
<td>CNN</td>
<td>EfficientNet-L2</td>
<td>90.2%*</td>
<td>&quot;Meta Psuedo Labels&quot; (EfficientNet-L2) (CVPR, 2021)</td>
</tr>
<tr>
<td>Transformer</td>
<td>ViT-G/14</td>
<td>90.45%*</td>
<td>&quot;Scaling Vision Transformers (ArXiv, 2021)</td>
</tr>
<tr>
<td>Hybrid CNN-Transformer</td>
<td>CoAtNet-7</td>
<td>90.88%*</td>
<td>&quot;CoAtNet: Marrying Convolution and Attention for All Data Sizes&quot; (NeurIPS, 2021)</td>
</tr>
</tbody>
</table>

*trained with additional data from Google’s Internal Datasets (JFT-300M & JFT-3B)
Method

Vision Transformer (ViT)

Transformer Encoder

Class
Bird
Ball
Car...

MLP Head

Transformer Encoder

Linear Projection of Flattened Patches

Patch + Position Embedding

* Extra learnable [class] embedding

Embedded Patches

MLP

Norm

Multi-Head Attention

Norm
Method
\[ x_p \in \mathbb{R}^{N \times (P^2 \cdot C)} \]

\[ z_0 = [x_{\text{class}}; x_p^1 \mathbf{E}; x_p^2 \mathbf{E}; \ldots; x_p^N \mathbf{E}] + \mathbf{E}_{\text{pos}}, \quad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}, \quad \mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1) \times D} \]
Method
Hybrid Architecture

- Found that input sequence can be formed using feature maps of CNN instead of raw image patches.
- Combines splitting into patches and performing linear projection.
- Patches can have 1x1 spatial size meaning input is obtained by flattening spatial dimensions and projecting to the Transformers dimension.
- Using a Conv Layer with kernel size and stride equal to the patch size.

Source: https://amaarora.github.io/2021/01/18/ViT.html
Method: Fine-Tuning and Higher Resolution

Source: https://www.youtube.com/watch?v=HZ4j_U3FC94
Experiments: Setup: Explanation of Datasets and Model Variants - Datasets

- **Pre-training Datasets**
  - ILSVRC-2012 ImageNet (i.e., ImageNet), ~1000 classes, ~1.3 million images
  - ImageNet-21k, ~21000 classes, ~14 million images
  - JFT, ~18000 classes, ~303 million images

- **Fine-tuning Datasets**
  - ImageNet - both original validation labels and Reassessed Labels (ReaL)
  - CIFAR-10/100, ~60000 images each
  - Oxford-IIIT Pets, ~7400 images
  - Oxford Flowers-102, ~7100 images
  - VTAB: natural, specialized, structured

- **Preprocessing**
  - Pre-training: image cropped, random horizontal mirroring, resize to 224x224
  - Fine-tuning: Images resized to 448x448, random crop of 384x384
  - Random horizontal flips
Experiments: Setup: Explanation of Datasets and Model Variants

- **ViT Model Variants**
  - Base - 32x32 and 16x16
  - Large - 32x32 and 16x16
  - Huge - 14x14

- **CNNs (ResNet (BiT))**
  - ResNet(50, 101, 152, 200)
  - Batch Normalization replaced with Group Normalization

- **Hybrids**
  - Use ResNet50 output from stage 4
  - Output (feature maps) fed to ViT with patch size of 1 pixel

---

Table 1: Details of Vision Transformer model variants.

<table>
<thead>
<tr>
<th>Model</th>
<th>Layers</th>
<th>Hidden size $D$</th>
<th>MLP size</th>
<th>Heads</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT-Base</td>
<td>12</td>
<td>768</td>
<td>3072</td>
<td>12</td>
<td>86M</td>
</tr>
<tr>
<td>ViT-Large</td>
<td>24</td>
<td>1024</td>
<td>4096</td>
<td>16</td>
<td>307M</td>
</tr>
<tr>
<td>ViT-Huge</td>
<td>32</td>
<td>1280</td>
<td>5120</td>
<td>16</td>
<td>632M</td>
</tr>
</tbody>
</table>


Experiments: Setup: Explanation of Datasets and Model Variants - Training and Fine Tuning

- **Training**
  - Uses Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$
  - Batch size 4096
  - Weight decay 0.1
  - Linear learning rate warmup and decay

- **Fine-tuning**
  - SGD with momentum
  - Batch size 512
  - Cosine learning rate decay
  - No weight decay

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ResNet50 Adam</th>
<th>ResNet50 SGD</th>
<th>ResNet152x2 Adam</th>
<th>ResNet152x2 SGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet</td>
<td>77.54</td>
<td>78.24</td>
<td>84.97</td>
<td>84.37</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>97.67</td>
<td>97.46</td>
<td>99.06</td>
<td>99.07</td>
</tr>
<tr>
<td>CIFAR100</td>
<td>86.07</td>
<td>85.17</td>
<td>92.05</td>
<td>91.06</td>
</tr>
<tr>
<td>Oxford-IIIT Pets</td>
<td>91.11</td>
<td>91.00</td>
<td>95.37</td>
<td>94.79</td>
</tr>
<tr>
<td>Oxford Flowers-102</td>
<td>94.26</td>
<td>92.06</td>
<td>98.62</td>
<td>99.32</td>
</tr>
<tr>
<td>Average</td>
<td>89.33</td>
<td>88.79</td>
<td>94.01</td>
<td>93.72</td>
</tr>
</tbody>
</table>
Experiments: Setup: Explanation of Datasets and Model Variants - Metrics

- Metrics
  - Fine-tuning accuracy: results after fine-tuning
  - Few-shot accuracy: results using a small subset of data (used for quick evaluations)
## Experiments: Pre-training Data Requirements

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of Images</th>
<th># of Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet (Small)</td>
<td>1.3 Million</td>
<td>1 Thousand</td>
</tr>
<tr>
<td>ImageNet-21K (Medium)</td>
<td>14 Million</td>
<td>21 Thousand</td>
</tr>
<tr>
<td>JFT (Big)</td>
<td>300 Million</td>
<td>18 Thousand</td>
</tr>
</tbody>
</table>
Experiments: Scaling Study

<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-IIIT 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>V TAB (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>

Source: https://www.youtube.com/watch?v=HZ4j_U3FC94
Inspecting Vision Transformers

Attention Rollout (Samira Abnar, 2020)
Visualization of Embedding Filters

What does the model learn?

- Calculate PCA on the embedding E i.e. create new features that summarize information from all feature vectors
- The learned filter look similar to low level CNN feature maps (left)
- The transformer is learning visual information in a familiar way to CNNs e.g. lines for edge detection in various orientations
What Does The Position Embedding Learn?

- Recall the Position Embedding is a matrix with \( N+1 \) Rows by \( D \) features

\[
E_{pos} \in \mathbb{R}^{(N+1) \times D}
\]

- For all patches \( N \)
  - Compute similarity between each row vector and itself

\[
similarity(A, B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}
\]

- Plotting the similarities show that the embedding is highly correlated between the rows and columns of its original 2-D position.

ViT-L/32
Attention Distance Over Layer Depth

ViT-L/16

Mean attention distance (pixels)

Network depth (layer)
Self-Supervision

Conclusion - Advantages

- Self-Attention
- Parallelized training over other sequence models
- Can learn spatial mapping of where patches were pulled
- Attention is widely spread across pixels even at early layers, revealing global understanding
- Diagrams are clear and well-made
Conclusion - Disadvantages

- Trained on additional data that is internal to Google
- Extremely large networks, not suited for deployment on edge computing
Computing $Z_0$ - Output of the Embedding

- Image $HxWxC$
- Image Patches $N \times (P \times P \times C)$
- Flatten Image Patches $N \times (P^2 \times C)$
- Class Token $1 \times D$
- Patch Embedding Filter $(P^2 \times C) \times D$
- Position Embedding $(N+1) \times D$