Funnel Vision Transformer for image classification

yujingz@stanford.edu
Background

Prior work

Transformer
- Dominant in NLP
- Good at capturing distant dependencies
- Parallel self-attention computation over different token pairs
- Easy to scale up

Vision Transformer
- Transformer architecture on the image classification tasks.
- Entirely replace the convolutions with self-attentions
- Outperform the state-of-the-art convolutional networks

Funnel-Transformer
- Efficient Transformer architecture for NLP pre-training in terms of computation FLOPs and memory
- Remove redundant information in sequence dimension

Problem

- Model performance improves with a large model size and more training data, but it’s computational more expensive.

Motivation

- Build an efficient Vision Transformer by removing redundant spatial information in deep Transformer layers

Funnel Vision Transformer
= Funnel Transformer + Vision Transformer
Problem statement

- **Task**: image classification
  - input: an image and a pre-labeled class
  - output: predict a class
  - architecture: Base-Transformer

- **Goal**: improve the training **efficiency** of Vision Transformer
  - Reach **comparable accuracy** to Vision Transformer with **less computation resource** (faster speed and less memory)
  - improve accuracy by re-investing the saved resources to a larger model capacity

- **Eval metrics**
  - quality: top-1 accuracy
  - resource: memory usage, steps/sec
Dataset

ImageNet

- 1000 classes
- 1,281,167 training images, 50,000 validation images and 100,000 test images.
- input resolution: 224 for pre-training, 384 for fine-tuning
- preprocessing: convert to tfrecords to re-use the ViT implementation provided by tensorflow official models.
Methods (baseline)

Vision Transformer (ViT)

Image pixels: $H \times W$

Patch size: $P \times P$

\[
Z_0 = [X_{\text{class}}; X_p^1 E; X_p^2 E; \ldots; X_p^N E] + E_{\text{pos}}
\]

\[
Z'_1 = \text{Self-Attention}(\text{LayerNorm}(z_{l-1}))+z_{l-1}, l = 1 \ldots L
\]

\[
Z_l = \text{MLP}(\text{LayerNorm}(z_l)) + z'_l, l = 1 \ldots L
\]

\[
y = \text{LayerNorm}(Z^L_l)
\]

Fine-tuning with different image resolution
- different patch sequence lengths
- apply 2D interpolation of the pre-trained position embeddings

Copied from 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. An image is worth 16x16 words: Transformers for image recognition at scale. ArXiv, abs/2010.11929, 2021
Methods

Funnel Vision Transformer (Funnel-ViT)

- Compress the patch sequence dimension by half through a pooling layer
  - patch dim: $N \rightarrow (N - 1) / 2 + 1$
  - (no pooling on class token)
  - weights shapes are not changed
- "pool-query-only" for self-attentions of the first layer after pooling
  \[
  \text{Attn}(Q = h', KV = h)
  \]
- Different re-invest strategies
  - different Transformer configurations across blocks
Experiments and analysis

Pre-training:

- with a small pre-training accuracy compromise (< 1%)
  - 25.8% less memory, 23% speedup with two funnel blocks
  - 40% less memory, 37.5% speedup with three funnel blocks
- Mean pooling performs better than max pooling

<table>
<thead>
<tr>
<th>block layout</th>
<th>train top1</th>
<th>val top1</th>
<th>train top5</th>
<th>val top5</th>
<th>memory usage</th>
<th>steps/sec time</th>
</tr>
</thead>
<tbody>
<tr>
<td>B12(197) (ViT)</td>
<td>92.4%</td>
<td>71.41%</td>
<td>97.28%</td>
<td>89%</td>
<td>12.38G</td>
<td>2.0 (12 hours to converge)</td>
</tr>
<tr>
<td>B6(197)-6(99), max</td>
<td>92.05%</td>
<td>70.42%</td>
<td>97.15%</td>
<td>88.41%</td>
<td>9.18G</td>
<td>2.6</td>
</tr>
<tr>
<td><strong>B6(197)-6(99), mean</strong></td>
<td>92.12%</td>
<td>71.19%</td>
<td>97.17%</td>
<td>88.9%</td>
<td>9.18G</td>
<td>2.6</td>
</tr>
<tr>
<td>B4(197)-4(99)-4(50), max</td>
<td>91.66%</td>
<td>69.71%</td>
<td>97.01%</td>
<td>87.8%</td>
<td>7.38G</td>
<td>3.2</td>
</tr>
<tr>
<td>B6(197)-6(50), max</td>
<td>92.02%</td>
<td>70.42%</td>
<td>97.14%</td>
<td>88.26%</td>
<td>8.04G</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Table 2. Different block layouts of pre-training on ImageNet.

Bn(t) means that there are n Transformer layers in a block with patch length t.
Experiments and analysis

- Converge at the same number of steps.
- If compress the spatial dimensions too much each time (e.g. stride = 4), it would make the training more unstable.

Figure 4. Loss and accuracy pretraining curves of ViT and Funnel-Vit
(black: ViT; blue: Funnel-ViT B6(197)-6(99); red: Funnel-ViT B4(197)-4(99)-4(50))
Funnel-ViT is able to capture similar class features as ViT.
Funnel-ViT slightly enlarges regions with a large effect on the classification score. The noise introduced by the additional pixels would disturb the feature extraction.
Results and analysis

Fine-tuning:

- Funnel-ViT outperforms ViT
- Fine-tuning ViT with a pre-trained Funnel-ViT yields better accuracy than the baseline
- Fine-tuning Funnel-ViT with a pre-trained ViT yields worse accuracy than baseline

<table>
<thead>
<tr>
<th>pre-training</th>
<th>fine-tuning</th>
<th>train top1</th>
<th>val top1</th>
<th>train top5</th>
<th>val top5</th>
<th>memory usage</th>
<th>steps/sec</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT</td>
<td>ViT</td>
<td>73.84%</td>
<td>70.95%</td>
<td>90.06%</td>
<td>89.45%</td>
<td>7.32G</td>
<td>4.4</td>
<td></td>
</tr>
<tr>
<td>Funnel-ViT</td>
<td>Funnel-ViT</td>
<td>75.28%</td>
<td>71.53%</td>
<td>90.8%</td>
<td>89.97%</td>
<td>5.38G</td>
<td>5.8</td>
<td></td>
</tr>
<tr>
<td>Funnel-ViT</td>
<td>ViT</td>
<td>75.18%</td>
<td>71.35%</td>
<td>90.74%</td>
<td>89.8%</td>
<td>7.32G</td>
<td>4.4</td>
<td></td>
</tr>
<tr>
<td>ViT</td>
<td>Funnel-ViT</td>
<td>73.09%</td>
<td>70.44%</td>
<td>89.7%</td>
<td>89.14%</td>
<td>5.38G</td>
<td>5.8</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Fine-tuning performance on ImageNet.
Results and analysis

Re-investment:

- Accuracy can be improved slightly with a deeper and wider model
- Overfitting
  - Mitigate overfitting by tuning model width and depth
    - Slightly increase model depth
    - Increase width of shallow layers (MLP dim); decrease width of deep layers (MLP dim)
  - Avoid overfitting requires more training data (e.g. Imagenet-21k)

<table>
<thead>
<tr>
<th>block layout</th>
<th>train top1</th>
<th>val top1</th>
<th>train top5</th>
<th>val top5</th>
<th>memory usage</th>
<th>steps/sec time</th>
</tr>
</thead>
<tbody>
<tr>
<td>B12(197) (ViT)</td>
<td>92.4%</td>
<td>71.41%</td>
<td>97.28%</td>
<td>89%</td>
<td>12.38G</td>
<td>2.0</td>
</tr>
<tr>
<td>B6(197)-6(99)</td>
<td>92.12%</td>
<td>71.19%</td>
<td>97.17%</td>
<td>88.9%</td>
<td>9.18G</td>
<td>2.6</td>
</tr>
<tr>
<td>B6(197)-6(99)-4(50)</td>
<td>93.7%</td>
<td>70.88%</td>
<td>97.76%</td>
<td>88.32%</td>
<td>10.19G</td>
<td>2.3</td>
</tr>
<tr>
<td>B6(197, h1024)-6(99, h1024)</td>
<td>93.45%</td>
<td>70.7%</td>
<td>97.69%</td>
<td>88.42%</td>
<td>10.54G</td>
<td>1.76</td>
</tr>
<tr>
<td>B6(197, m4096)-6(99, m4096)</td>
<td>93.29%</td>
<td>70.27%</td>
<td>97.62%</td>
<td>88.11%</td>
<td>10.04G</td>
<td>2.35</td>
</tr>
<tr>
<td>B6(197)-6(99)-4(50, m1536)</td>
<td>93.04%</td>
<td>71.17%</td>
<td>97.53%</td>
<td>88.79%</td>
<td>9.78G</td>
<td>2.4</td>
</tr>
<tr>
<td>B6(197)-6(99)-2(50, m1536)</td>
<td>92.65%</td>
<td>71.13%</td>
<td>97.36%</td>
<td>88.69%</td>
<td>9.44G</td>
<td>2.5</td>
</tr>
<tr>
<td>B6(197, m4096)-6(99)-2(50, m1536)</td>
<td>93.05%</td>
<td>71.31%</td>
<td>97.51%</td>
<td>88.69%</td>
<td>10.97G</td>
<td>2.1</td>
</tr>
</tbody>
</table>

Table 4. Different re-investment configurations of pre-training on ImageNet.
(h: hidden dim; m: MLP dim)
Conclusions

● Demonstrate *spatial information redundancy* in deeper Transformer layers

● Compression
  ○ For pre-training, with a small accuracy loss
    ■ 25.8% less memory, 23% speedup with two funnel blocks (0.22% accuracy loss)
    ■ 40% less memory, 37.5% speedup with three funnel blocks (0.7% accuracy loss)
  ○ For fine-tuning, improves accuracy by 0.6%
  ○ For the whole pre-training and fine-tuning pipeline, *attain accuracy* but with much *less computation resource*

● Re-investment
  ○ *Improve accuracy* slightly by re-investing some saved resources to a deeper and wider model.

Future work

● Try Large-Transformer on a larger pre-training dataset to further explore different re-investment strategies (e.g. ImageNet-21k)
● Experiment with other datasets for fine-tuning to consolidate the conclusion (e.g. CIFAR-10, CIFAR100)