Deep Learning
Week 05: ViTs and CLIP

Logistics

- HW3 is due next week
- We have a feedback form (due Friday, March 8th)
- Project proposal due today
  - You can use slip days
  - Please think about compute when you choose a paper for your project
    - Look for reported training hardware and training times
  - Choose papers from ICML, ICLR, NeuRIPS, CVPR, ECCV, ACL, EMNLP, NAACL

Use self-supervised learning to learn embeddings for images

Vector Space

How to use Attention for Vision Tasks?

Idea #1: Add attention to existing CNNs

How to use Attention for Vision Tasks?

Idea #2: Adapt standard transformers to image data

Can we extend this idea to images?

Use pixels as input

R x R image needs R^4 elements per attention matrix. R=128, 48 layers, 16 heads per layer takes 768GB of memory for attention matrices for a single example...
Image patches as “words”

CNNs can be used to preprocess the image before converting it to patches!

Using image patches like words in a sentence!

Adding positional embeddings

Positional embeddings for ViTs are usually learned d-dimensional vectors per position

Classification

Transformer Encoder
**Vision Transformer**


**ViT Results**


**Training time:**
- ViT-H/14: 2.5k TPU days
- Largest BiT: 9.9k TPU-v3 days

**ViT Summary**

**Model:**
- Model is almost identical to BERT
- Replace words with PxP pixel image patches, P ∈ {14, 16, 32} (no overlap)
- Each patch is embedded linearly into a vector of size 1024
- 1D positional embeddings

**Training:**
- For pre-training, optimize for image classification on large supervised dataset (e.g. ImageNet 21K, JFT-300M)
- For fine-tuning, learn a new classification head on a small dataset (e.g. CIFAR-100)

**ACTIVITY:** When do ViTs outperform CNNs, and vice versa?

Think about what you know about transformers - what are some of their drawbacks?

When is it “worth it” to use transformers instead of just CNNs?
Swin Transformers

Hierarchical architecture that has the flexibility to model at various scales


Shifted Window attention

Linear computational complexity with respect to image size

Performance

<table>
<thead>
<tr>
<th>method</th>
<th>image size</th>
<th>#param.</th>
<th>FLOPs (G)</th>
<th>throughput (image / s)</th>
<th>ImageNet top-1 acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-101x3 [38]</td>
<td>384x384</td>
<td>388M</td>
<td>204.6G</td>
<td>-</td>
<td>84.4</td>
</tr>
<tr>
<td>R-152x4 [38]</td>
<td>480x480</td>
<td>937M</td>
<td>840.5G</td>
<td>-</td>
<td>85.4</td>
</tr>
<tr>
<td>ViT-B/16 [20]</td>
<td>384x384</td>
<td>86M</td>
<td>55.4G</td>
<td>85.9</td>
<td>84.0</td>
</tr>
<tr>
<td>ViT-L/16 [20]</td>
<td>384x384</td>
<td>307M</td>
<td>190.7G</td>
<td>27.3</td>
<td>85.2</td>
</tr>
<tr>
<td>Swin-B</td>
<td>224x224</td>
<td>88M</td>
<td>15.4G</td>
<td>278.1</td>
<td>85.2</td>
</tr>
<tr>
<td>Swin-B</td>
<td>384x384</td>
<td>88M</td>
<td>47.0G</td>
<td>84.7</td>
<td>86.4</td>
</tr>
<tr>
<td>Swin-L</td>
<td>384x384</td>
<td>197M</td>
<td>103.9G</td>
<td>42.1</td>
<td>87.3</td>
</tr>
</tbody>
</table>
Self-Supervised Vision Transformers (DiNO)

Centering and sharpening

- Centering prevents one dimension from dominating
- Sharpening prevents learning a uniform distribution


DINO v2

<table>
<thead>
<tr>
<th></th>
<th>INet-1k k-NN</th>
<th>INet-1k linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>iBOT</td>
<td>72.9</td>
<td>82.3</td>
</tr>
<tr>
<td>+(our reproduction)</td>
<td>74.5 ± 1.6</td>
<td>83.2 ± 0.9</td>
</tr>
<tr>
<td>+LayerScale, Stochastic Depth</td>
<td>78.4 ± 0.9</td>
<td>82.0 ± 1.2</td>
</tr>
<tr>
<td>+128k prototypes</td>
<td>76.6 ± 1.2</td>
<td>81.9 ± 0.1</td>
</tr>
<tr>
<td>+KoLeo</td>
<td>75.9 ± 2.3</td>
<td>82.5 ± 0.6</td>
</tr>
<tr>
<td>+SwiGLU FFN</td>
<td>78.7 ± 0.2</td>
<td>83.1 ± 0.6</td>
</tr>
<tr>
<td>+Patch size 14</td>
<td>78.9 ± 0.2</td>
<td>83.5 ± 0.4</td>
</tr>
<tr>
<td>+Teacher momentum 0.994</td>
<td>79.4 ± 0.5</td>
<td>83.6 ± 0.1</td>
</tr>
<tr>
<td>+Tweak warmup schedules</td>
<td>80.5 ± 1.1</td>
<td>83.8 ± 0.2</td>
</tr>
<tr>
<td>+Batch size 3k</td>
<td>81.7 ± 1.2</td>
<td>84.7 ± 0.9</td>
</tr>
<tr>
<td>+Sinkhorn-Knopp</td>
<td>81.7 =</td>
<td>84.7 =</td>
</tr>
<tr>
<td>+Unifying heads = DINOv2</td>
<td>82.0 ± 0.3</td>
<td>84.5 ± 0.2</td>
</tr>
</tbody>
</table>

Discuss: BERT is trained with cross entropy loss. Can you do the same with MaE or should you use a different loss?
MaE Results

- Compared to supervised ViTs
  - Requires minimal data augmentation
  - Transfers better to downstream vision tasks
    - Object detection, segmentation

<table>
<thead>
<tr>
<th>Case</th>
<th>r</th>
<th>lin</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>84.0</td>
<td>65.7</td>
</tr>
<tr>
<td>crop, fixed size</td>
<td>84.7</td>
<td>73.1</td>
</tr>
<tr>
<td>crop, rand size</td>
<td>84.9</td>
<td>73.8</td>
</tr>
<tr>
<td>crop + color jitter</td>
<td>84.3</td>
<td>71.9</td>
</tr>
</tbody>
</table>

(e) Data augmentation. Our MAE works with minimal or no augmentation.

Table 4: COCO object detection and segmentation using a ViT Mask R-CNN baseline. All entries are based on our implementation. Self-supervised entries use INIK data without labels. Mask AP follows a similar trend as box AP.

CLIP (Contrastive Language–Image Pre-training)

Discuss: How can you train this model?
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# T[n, h, w, c] - minibatch of aligned images
# T[n, 1] - minibatch of aligned texts
# N_i[d_i, d_e] - learned proj of image to embed
# N_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) @ [n, d_e]

- Trained on 256 V100 GPUs for two weeks on 400 million (image, text pairs)
- On AWS, this would cost at least 200k dollars

```python
# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T.e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```

Clip demo

[Link to Hugging Face Spaces](https://huggingface.co/spaces/vivien/clip)
Application of CLIP

Vector Space

**f(x) = audio spectrogram transformer**
Audio Processing

Audio File

Spectrogram:
- Energy, pitch, fundamental frequency
- Decomposes signal into frequencies and their corresponding amplitudes

Spectrogram source: Dumpala 2017

Audio as a vision problem

Most likely word sequence

Vision Transformer

<(0,0)> <(0,1)> <(0,2)> <(1,1)>

Review

- Transformers can be used for vision tasks
- Swin transformers can be used for learning features at different scales
- Self-supervised learning is also helpful for transformer backbone vision models
  - Dino and MAE both learn very good embeddings
- Using transformer models for images and text helps build multi-modal models like CLIP