Logistics

- HW3 is out
- We have a feedback form (due Friday, March 8th)
- Project proposal due Thursday, March 7th
Project

- Aim: to get hands on experience with implementing modern deep learning methods
- To be completed in groups of 2-3
- Find a recent deep learning research paper
- Reproduce a specific result from the paper
  - Need to implement yourself!
- It's ok if open-source implementations exist
  - But you can’t use them!
A page long project proposal due March 7. It should contain the following:

- **Paper selection:**
  - Title, authors, and publication venue of the chosen paper
  - Brief summary of the chosen paper
  - Brief justification of why you choose this paper

- **Result Selection**
  - Tell us which result you want to replicate

- **Re-implementation Plan**
  - Describe architecture, method, and metrics
  - Details about how much compute and time is required to replicate results

- **Detailed instructions on canvas**
The best animals are puppies. I enjoy petting baby dogs. The tastiest fruits are oranges.
$f(x) = \text{word embedding}$

Vector Space

“puppies” $\textbf{x}_1$

“orange” $\textbf{x}_2$

“purple” $\textbf{x}_4$

“apple” $\textbf{x}_5$

“dogs” $\textbf{x}_3$
Vector Space

$f(x) = \text{raw pixels}$

**Semantically different:**
puppy vs. cow

**Structurally similar:**
black and white animal, grass

**Structurally different:**
hands, different backgrounds

**Semantically similar:**
Bernese puppies
Pixel-Space: Nearest Neighbors

- Dominated by shallow similarities
  - Background, etc.
- Poor semantic alignment

Cifar-10 Example
Vector Space

\[ f(x) = \text{classification network} \]

Semantically different:
- puppy vs. cow

Structurally similar:
- black and white animal, grass
- hands, different backgrounds
- Bernese puppies

\[ f(x_1) \quad f(x_2) \quad f(x_3) \]
How does the network know that these should be mapped to similar space?

$$f(x) = \text{classification network}$$
How does the network know that these should be mapped to similar space?
Image Classification

feature extraction

classification

Image features!

0.9 “dog”

0.1 “cat”
Neural Net Features: Nearest Neighbors

- Image classification features work really well!
- Strong semantic alignment
- More robust to shallow variations
Pretraining: Train a general purpose model on lots of data, and later customize it for more specific tasks.

NLP: BERT

CV: Imagenet

Already have a very well-defined vector space.
Image Pretraining

First, train on a large, diverse dataset so that our model learns to extract robust image features.
Fine-tuning

Then, finetune for a specific task
Pre-train then Fine-tune

- Use image classification backbone as a feature extractor for other vision tasks
  - E.g. Instance segmentation
- Significantly accelerates training
  - Random init requires much longer training


Few Shot Learning

Adapt to variations within known classes, with LIMITED labeled training data

- We’ve only seen a few puppies and a few kittens, but a lot of other pretrained data
Potential Problems?

Figure 1: Example images from the IMAGENET, the *retinal fundus photographs*, and the CHEXPERT datasets, respectively. The fundus photographs and chest x-rays have much higher resolution than the IMAGENET images, and are classified by looking for small local variations in tissue.

Transfer Learning

Images may be out-of-distribution from the training data

[x_1]  [x_2]  [x_3]

\[ f(x_1) \bullet \quad f(x_2) \bullet \quad f(x_3) \bullet \]
Potential Problems?

- Classify diabetic retinopathy in retinal photographs
- Introduce classification simple architecture
  - Sequence of: Convolution, Batchnorm, ReLU (CBR)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model Architecture</th>
<th>Random Init</th>
<th>Transfer</th>
<th>Parameters</th>
<th>IMAGENET Top5</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETINA</td>
<td>Resnet-50</td>
<td>96.4% ± 0.05</td>
<td>96.7% ± 0.04</td>
<td>23570408</td>
<td>92.0% ± 0.06</td>
</tr>
<tr>
<td>RETINA</td>
<td>Inception-v3</td>
<td>96.6% ± 0.13</td>
<td>96.7% ± 0.05</td>
<td>22881424</td>
<td>93.9%</td>
</tr>
<tr>
<td>RETINA</td>
<td>CBR-LargeT</td>
<td>96.2% ± 0.04</td>
<td>96.2% ± 0.04</td>
<td>8532480</td>
<td>77.5% ± 0.03</td>
</tr>
<tr>
<td>RETINA</td>
<td>CBR-LargeW</td>
<td>95.8% ± 0.04</td>
<td>95.8% ± 0.05</td>
<td>8432128</td>
<td>75.1% ± 0.3</td>
</tr>
<tr>
<td>RETINA</td>
<td>CBR-Small</td>
<td>95.7% ± 0.04</td>
<td>95.8% ± 0.01</td>
<td>2108672</td>
<td>67.6% ± 0.3</td>
</tr>
<tr>
<td>RETINA</td>
<td>CBR-Tiny</td>
<td>95.8% ± 0.03</td>
<td>95.8% ± 0.01</td>
<td>1076480</td>
<td>73.5% ± 0.05</td>
</tr>
</tbody>
</table>

Table 1: Transfer learning and random initialization perform comparably across both standard IMAGENET architectures and simple, lightweight CNNs for AUCs from diagnosing moderate DR. Both sets of models also have similar AUCs, despite significant differences in size and complexity. Model performance on DR diagnosis is also not closely correlated with IMAGENET performance, with the small models performing poorly on IMAGENET but very comparably on the medical task.

Potential Problems?

- Classify pathologies in chest x-rays

<table>
<thead>
<tr>
<th>Model Architecture</th>
<th>Atelectasis</th>
<th>Cardiomegaly</th>
<th>Consolidation</th>
<th>Edema</th>
<th>Pleural Effusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resnet-50</td>
<td>79.52±0.31</td>
<td>75.23±0.35</td>
<td>85.49±1.32</td>
<td>88.34±1.17</td>
<td>88.70±0.13</td>
</tr>
<tr>
<td>Resnet-50 (trans)</td>
<td>79.76±0.47</td>
<td>74.93±1.41</td>
<td>84.42±0.65</td>
<td>88.89±1.66</td>
<td>88.07±1.23</td>
</tr>
<tr>
<td>CBR-LargeT</td>
<td>81.52±0.25</td>
<td>74.83±1.66</td>
<td>88.12±0.25</td>
<td>87.97±1.40</td>
<td>88.37±0.01</td>
</tr>
<tr>
<td>CBR-LargeT (trans)</td>
<td>80.89±1.68</td>
<td>76.84±0.87</td>
<td>86.15±0.71</td>
<td>89.03±0.74</td>
<td>88.44±0.84</td>
</tr>
<tr>
<td>CBR-LargeW</td>
<td>79.79±0.79</td>
<td>74.63±0.69</td>
<td>86.71±1.45</td>
<td>84.80±0.77</td>
<td>86.53±0.54</td>
</tr>
<tr>
<td>CBR-LargeW (trans)</td>
<td>80.70±0.31</td>
<td>77.23±0.84</td>
<td>86.87±0.33</td>
<td>89.57±0.34</td>
<td>87.29±0.69</td>
</tr>
<tr>
<td>CBR-Small</td>
<td>80.43±0.72</td>
<td>74.36±1.06</td>
<td>88.07±0.60</td>
<td>86.20±1.35</td>
<td>86.14±1.78</td>
</tr>
<tr>
<td>CBR-Small (trans)</td>
<td>80.18±0.85</td>
<td>75.24±1.43</td>
<td>86.48±1.13</td>
<td>89.09±1.04</td>
<td>87.88±1.01</td>
</tr>
<tr>
<td>CBR-Tiny</td>
<td>80.81±0.55</td>
<td>75.17±0.73</td>
<td>85.31±0.82</td>
<td>84.87±1.13</td>
<td>85.56±0.89</td>
</tr>
<tr>
<td>CBR-Tiny (trans)</td>
<td>80.02±1.06</td>
<td>75.74±0.71</td>
<td>84.28±0.82</td>
<td>89.81±1.08</td>
<td>87.69±0.75</td>
</tr>
</tbody>
</table>

Table 2: Transfer learning provides mixed performance gains on chest x-rays. Performances (AUC%) of diagnosing different pathologies on the CHEXPERT dataset. Again we see that transfer learning does not help significantly, and much smaller models performing comparably.

Not all images are labeled

- Particular problem for specialized domains (e.g. medicine)
  - Annotation is expensive!
- Much easier to collect unlabeled data
  - Similar to text!
- Can we still learn good image representations?
The exact same image, rotated, maps to a completely different location in vector space.
How do we learn structure so that these map to similar points in vector space?

\[ f(x) = \text{???} \]
$f(x) = \text{classification network}$

**Vector Space**

Class “puppy”

Class “puppy”

$x_1$  

$x_2$

$f(x_1)$  

$f(x_2)$
And what if they are unlabeled?

\[ f(x) = \text{classification network} \]

Vector Space

\[ f(x_1), f(x_2) \]
Self-Supervised Learning

- Aim to learn from data without manual label annotation
  - Useful for specialized domains (e.g., medicine) with limited annotated data
- Self-supervised learning methods solve “pretext” tasks that produce good features for downstream tasks.
  - Learn with supervised learning objectives (e.g., classification, regression)
  - Labels of these pretext tasks are generated automatically

Figure 1: Images rotated by random multiples of 90 degrees (e.g., 0, 90, 180, or 270 degrees). The core intuition of our self-supervised feature learning approach is that if someone is not aware of the concepts of the objects depicted in the images, he cannot recognize the rotation that was applied to them.
Rotation Prediction

- Self-supervised learning by rotating the input image
- Predict which rotation is applied
  - 4-way classification
Rotation Prediction

- Self-supervised learning by rotating the input image
- Predict which rotation is applied
  - 4-way classification

How to evaluate a self-supervised learning method?

- Don’t care about the performance of the self-supervised learning task
  - E.g. Image rotation prediction
- Evaluate the learned feature encoder on downstream target tasks
How to evaluate a self-supervised learning method?

1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations
How to evaluate a self-supervised learning method?

1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

2. Attach a shallow network on the feature extractor; train the shallow network on the target task with small amount of labeled data
Self-Supervised Evaluation

- Downstream performance correlates with prefix task: rotation prediction

Cifar-10 Image Classification

Self-Supervised Evaluation

- Self-supervised learning outperforms supervised learning with limited data
  - Can use large volumes of unlabeled data!

Discuss

We are provided this image without labels: what are some other tasks we can do with it?

How can we perform self-supervised learning with images?
Can we learn this directly?

\[ f(x) = ??? \]
Review: Image Augmentation

- Horizontal flips
- Rotate image
- Zoom/crop image
- Brighten/darken image
- Shift colors

$f(x) =$ contrastive learning

Vector Space

$x_1$

$x_2$

$x_3$

All positive pairs are augmentations of the original image
Any other image is a negative pair
Basic Model for Contrastive Learning

- Push positive pairs close together in feature space
- Pull negative pairs far apart in feature space

Loss function
Triplet loss function

\[ \ell = \max(0, \| f(x_i) - f(x^+) \|^2 - \| f(x_i) - f(x^-) \|^2 + c) \]

Anchor example

Model

Positive pair

Negative pair
Triplet loss function

\[ \ell = \max(0, \| f(x_i) - f(x^+) \|^2 - \| f(x_i) - f(x^-) \|^2 + c) \]

- Ensures loss is not negative
- Model should map positive examples close together
- Model should map negative examples far apart
- Margin
Discuss

- Any potential problems with the triplet loss?
- Any ideas to remedy those problems

\[ \ell = \max(0, \|f(x_i) - f(x^+)\|^2 - \|f(x_i) - f(x^-)\|^2 + c) \]

- Ensures loss is not negative
- Model should map positive examples close together
- Model should map negative examples far apart
SimCLR: A Simple Contrastive Learning Framework for Images

- Sample two different augmentations of an image

SimCLR: A Simple Contrastive Learning Framework for Images

- Sample two different augmentations of an image
- Apply a base encoder to each view of the image to extract an image feature
  - e.g. ResNet

SimCLR: A Simple Contrastive Learning Framework for Images

- Sample two different augmentations of an image
- Apply a base encoder to each view of the image to extract an image feature
  - e.g. ResNet
- Apply an MLP projection head to generate final representations
  - Throw away projection head after training

SimCLR: A Simple Contrastive Learning Framework for Images

- Sample two different augmentations of an image
- Apply a base encoder to each view of the image to extract an image feature
  - e.g. ResNet
- Apply an MLP projection head to generate final representations
  - Throw away projection head after training

SimCLR Augmentations

- (a) Original
- (b) Crop and resize
- (c) Crop, resize (and flip)
- (d) Color distort. (drop)
- (e) Color distort. (jitter)
- (f) Rotate {90°, 180°, 270°}
- (g) Cutout
- (h) Gaussian noise
- (i) Gaussian blur
- (j) Sobel filtering

SimCLR Loss

- Temperature-scaled cross-entropy loss

\[ \mathcal{L}_{\text{SimCLR}} = -\log \left( \frac{\exp(d(x_i, x_i^+)/\tau)}{\exp(d(x_i, x_i^+)/\tau) + \exp(d(x_i, x_i^-)/\tau)} \right) \]

Model should map positive examples close together

Model should map negative examples far apart

SimCLR Algorithm

- Use other images in the mini-batch as negatives
- L2 normalize representations
  - Use cosine similarity as the distance metric
- Compute temperature-scaled cross-entropy for all positive pairs

Algorithm 1 SimCLR’s main learning algorithm.

```
input: batch size $N$, constant $\tau$, structure of $f, g, T$
for sampled minibatch $\{x_k\}_{k=1}^N$ do
  for all $k \in \{1, \ldots, N\}$ do
    draw two augmentation functions $t \sim T$, $t' \sim T$
    # the first augmentation
    $\tilde{x}_{2k-1} = t(x_k)$
    $h_{2k-1} = f(\tilde{x}_{2k-1})$
    $z_{2k-1} = g(h_{2k-1})$
    # the second augmentation
    $\tilde{x}_{2k} = t'(x_k)$
    $h_{2k} = f(\tilde{x}_{2k})$
    $z_{2k} = g(h_{2k})$
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ do
    $s_{i,j} = z_i^\top z_j / (\|z_i\| \|z_j\|)$
    # pairwise similarity
  end for
  define $\ell(i,j)$ as $\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1,2k) + \ell(2k,2k-1)]$
  update networks $f$ and $g$ to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$
```

Comparison of Loss Functions

- Temperature-scaled cross entropy places more weight on hard negatives
  - Don’t need to mine hard negatives

<table>
<thead>
<tr>
<th>Name</th>
<th>Negative loss function</th>
<th>Gradient w.r.t. $u$</th>
</tr>
</thead>
<tbody>
<tr>
<td>NT-Xent</td>
<td>$u^T v^+ / \tau - \log \sum_{v \in {v^+, v^-}} \exp(u^T v / \tau)$</td>
<td>$(1 - \frac{\exp(u^T v^+ / \tau)}{Z(u)})/\tau v^+ - \sum v^- \frac{\exp(u^T v^- / \tau)}{Z(u)}/\tau v^-$</td>
</tr>
<tr>
<td>NT-Logistic</td>
<td>$\log \sigma(u^T v^+ / \tau) + \log \sigma(-u^T v^- / \tau)$</td>
<td>$(\sigma(-u^T v^+ / \tau))/\tau v^+ - \sigma(u^T v^- / \tau)/\tau v^-$</td>
</tr>
<tr>
<td>Margin Triplet</td>
<td>$- \max(u^T v^- - u^T v^+ + m, 0)$</td>
<td>$v^+ - v^-$ if $u^T v^+ - u^T v^- &lt; m$ else 0</td>
</tr>
</tbody>
</table>

**Table 2.** Negative loss functions and their gradients. All input vectors, i.e. $u$, $v^+$, $v^-$, are $\ell_2$ normalized. NT-Xent is an abbreviation for “Normalized Temperature-scaled Cross Entropy”. Different loss functions impose different weightings of positive and negative examples.
SimCLR Results

- Train a linear classifier on features from SimCLR
- Approaches supervised performance!

SimCLR Results

- Self-supervised vs. supervised ImageNet pre-training
- Evaluate transfer performance across 12 downstream classification datasets
  - Often outperforms supervised pre-training!

<table>
<thead>
<tr>
<th>Linear evaluation:</th>
<th>Food</th>
<th>CIFAR10</th>
<th>CIFAR100</th>
<th>Birdsnap</th>
<th>SUN397</th>
<th>Cars</th>
<th>Aircraft</th>
<th>VOC2007</th>
<th>DTD</th>
<th>Pets</th>
<th>Caltech-101</th>
<th>Flowers</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimCLR (ours)</td>
<td>76.9</td>
<td>95.3</td>
<td>80.2</td>
<td>48.4</td>
<td>65.9</td>
<td>60.0</td>
<td>61.2</td>
<td>84.2</td>
<td>78.9</td>
<td>89.2</td>
<td>93.9</td>
<td>95.0</td>
</tr>
<tr>
<td>Supervised</td>
<td>75.2</td>
<td>95.7</td>
<td>81.2</td>
<td>56.4</td>
<td>64.9</td>
<td>68.8</td>
<td>63.8</td>
<td>83.8</td>
<td>78.7</td>
<td>92.3</td>
<td>94.1</td>
<td>94.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fine-tuned:</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SimCLR (ours)</td>
<td>89.4</td>
<td>98.6</td>
<td>89.0</td>
<td>78.2</td>
<td>68.1</td>
<td>92.1</td>
<td>87.0</td>
<td>86.6</td>
<td>77.8</td>
<td>92.1</td>
<td>94.1</td>
<td>97.6</td>
</tr>
<tr>
<td>Supervised</td>
<td>88.7</td>
<td>98.3</td>
<td>88.7</td>
<td>77.8</td>
<td>67.0</td>
<td>91.4</td>
<td>88.0</td>
<td>86.5</td>
<td>78.8</td>
<td>93.2</td>
<td>94.2</td>
<td>98.0</td>
</tr>
<tr>
<td>Random init</td>
<td>88.3</td>
<td>96.0</td>
<td>81.9</td>
<td>77.0</td>
<td>53.7</td>
<td>91.3</td>
<td>84.8</td>
<td>69.4</td>
<td>64.1</td>
<td>82.7</td>
<td>72.5</td>
<td>92.5</td>
</tr>
</tbody>
</table>

Effect Of Projection Head

- Projects data to “augmentation-invariant” representation
  - Less useful features for downstream tasks

*Figure 8.* Linear evaluation of representations with different projection heads $g(\cdot)$ and various dimensions of $z = g(h)$. The representation $h$ (before projection) is 2048-dimensional here.
Effect Of Projection Head

- Projects data to “augmentation-invariant” representation
  - Features less useful for downstream tasks

---

**Figure B.4.** t-SNE visualizations of hidden vectors of images from a randomly selected 10 classes in the validation set.

Impact of Loss Function

- Proposed loss outperforms the margin loss
  - Even with negative mining
- L2 normalization is useful
- Sensitive to cross-entropy temperature

<table>
<thead>
<tr>
<th>Margin</th>
<th>NT-Logi.</th>
<th>Margin (sh)</th>
<th>NT-Logi.(sh)</th>
<th>NT-Xent</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.9</td>
<td>51.6</td>
<td>57.5</td>
<td>57.9</td>
<td>63.9</td>
</tr>
</tbody>
</table>

*Table 4. Linear evaluation (top-1) for models trained with different loss functions. “sh” means using semi-hard negative mining.*

<table>
<thead>
<tr>
<th>$\ell_2$ norm?</th>
<th>$\tau$</th>
<th>Entropy</th>
<th>Contrastive acc.</th>
<th>Top 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>0.05</td>
<td>1.0</td>
<td>90.5</td>
<td>59.7</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>4.5</td>
<td>87.8</td>
<td>64.4</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>8.2</td>
<td>68.2</td>
<td>60.7</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>8.3</td>
<td>59.1</td>
<td>58.0</td>
</tr>
<tr>
<td>No</td>
<td>10</td>
<td>0.5</td>
<td>91.7</td>
<td>57.2</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.5</td>
<td>92.1</td>
<td>57.0</td>
</tr>
</tbody>
</table>

*Table 5. Linear evaluation for models trained with different choices of $\ell_2$ norm and temperature $\tau$ for NT-Xent loss. The contrastive distribution is over 4096 examples.*
Impact of Batch Size

- Requires large batches
  - Harder negatives!

Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.
Momentum Contrast (MoCo)

- Cache negative samples from earlier batches as you train

- Replace one encoder with an exponential moving average (EMA) of the model
  - Makes queued representations more stable
  \[ \theta_k \leftarrow m\theta_k + (1 - m)\theta_q \]

MoCo v2

- MoCo v2: MoCo with some tricks from SimCLR
  - Stronger augmentations
  - MLP projection head
- Outperform SimCLR with modest batch sizes
  - Large numbers of negatives available from the queue

<table>
<thead>
<tr>
<th>case</th>
<th>MLP</th>
<th>aug+</th>
<th>cos</th>
<th>epochs</th>
<th>batch</th>
<th>ImageNet acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCo v1 [6]</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>200</td>
<td>256</td>
<td>60.6</td>
</tr>
<tr>
<td>SimCLR [2]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>200</td>
<td>256</td>
<td>61.9</td>
</tr>
<tr>
<td>SimCLR v2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>8192</td>
<td>256</td>
<td>66.6</td>
</tr>
<tr>
<td>MoCo v2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>200</td>
<td>256</td>
<td>67.5</td>
</tr>
</tbody>
</table>

Results of longer unsupervised training follow:

<table>
<thead>
<tr>
<th>case</th>
<th>MLP</th>
<th>aug+</th>
<th>cos</th>
<th>epochs</th>
<th>batch</th>
<th>ImageNet acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimCLR [2]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1000</td>
<td>4096</td>
<td>69.3</td>
</tr>
<tr>
<td>MoCo v2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>800</td>
<td>256</td>
<td>71.1</td>
</tr>
</tbody>
</table>

Table 2. **MoCo vs. SimCLR**: ImageNet linear classifier accuracy (ResNet-50, 1-crop 224×224), trained on features from unsupervised pre-training. “aug+” in SimCLR includes blur and stronger color distortion. SimCLR ablations are from Fig. 9 in [2] (we thank the authors for providing the numerical results).

<table>
<thead>
<tr>
<th>mechanism</th>
<th>batch</th>
<th>memory / GPU</th>
<th>time / 200-ep.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCo</td>
<td>256</td>
<td>5.0G</td>
<td>53 hrs</td>
</tr>
<tr>
<td>end-to-end</td>
<td>256</td>
<td>7.4G</td>
<td>65 hrs</td>
</tr>
<tr>
<td>end-to-end</td>
<td>4096</td>
<td>93.0G†</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. †: based on our estimation.

Recap

- Supervised image classification pre-training produces strong image representations
  - Can efficiently transfer to other tasks
- Can apply self-supervised learning to images
  - Prefix tasks: rotation prediction, masked-image modeling, etc.
- Contrastive learning explicitly enforces similarity in representation space
  - Requires defining image augmentations