Self-Supervised Learning (Continued)
Large Vision and Language Models
Pretext tasks from image transformations

- Image completion
- Rotation prediction
- “Jigsaw puzzle”
- Colorization

Learned representations may be tied to a specific pretext task!
Can we come up with a more general pretext task?
A more general pretext task?

same object
A more general pretext task?

same object

different object
Contrastive Representation Learning
Today’s Agenda

Pretext tasks from image transformations
- Rotation, inpainting, rearrangement, coloring

Contrastive representation learning
- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC
Contrastive Representation Learning
Contrastive Representation Learning

\[ x^+ \]

\[ \mathbf{x} \]

\[ x^- \] reference
\[ x^+ \] positive
\[ x^- \] negative
A formulation of contrastive learning

What we want:

\[ \text{score}(f(x), f(x^+)) >> \text{score}(f(x), f(x^-)) \]

\( x \): reference sample; \( x^+ \) positive sample; \( x^- \) negative sample

Given a chosen score function, we aim to learn an encoder function \( f \) that yields high score for positive pairs \( (x, x^+) \) and low scores for negative pairs \( (x, x^-) \).
A formulation of contrastive learning

Loss function given 1 positive sample and $N - 1$ negative samples:

$$L = -\mathbb{E}_X \left[ \log \left( \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right) \right]$$
A formulation of contrastive learning

Loss function given 1 positive sample and \( N - 1 \) negative samples:

\[
L = -\mathbb{E}_x \left[ \log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]
\]
A formulation of contrastive learning

Loss function given 1 positive sample and N - 1 negative samples:

\[
L = -\mathbb{E}_x \left[ \log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]
\]

This seems familiar ...
A formulation of contrastive learning

Loss function given 1 positive sample and N - 1 negative samples:

\[
L = -\mathbb{E}_x \left[ \log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]
\]

This seems familiar ...
Cross entropy loss for a N-way softmax classifier!
I.e., learn to find the positive sample from the N samples
A formulation of contrastive learning

Loss function given 1 positive sample and N - 1 negative samples:

\[
L = -\mathbb{E}_x \left[ \log \frac{\exp(s(f(x), f(x^+))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]
\]

Commonly known as the InfoNCE loss (van den Oord et al., 2018)

A lower bound on the mutual information between \( f(x) \) and \( f(x^+) \)

\[
MI[f(x), f(x^+)] - \log(N) \geq -L
\]

The larger the negative sample size (N), the tighter the bound

Detailed derivation: Poole et al., 2019
SimCLR: A Simple Framework for Contrastive Learning

Cosine similarity as the score function:
\[ s(u, v) = \frac{u^T v}{||u||||v||} \]

Use a projection network \( h(\cdot) \) to project features to a space where contrastive learning is applied

Generate positive samples through data augmentation:
- random cropping, random color distortion, and random blur.

Source: Chen et al., 2020
SimCLR: generating positive samples from data augmentation

Source: Chen et al., 2020
SimCLR

Generate a positive pair by sampling data augmentation functions

Algorithm 1 SimCLR’s main learning algorithm.

```
input: batch size $N$, constant $\tau$, structure of $f$, $g$, $\mathcal{T}$.
for sampled minibatch $\{x_k\}_{k=1}^{N}$ do
  for all $k \in \{1, \ldots, N\}$ do
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{x}_{2k-1} = t(x_k)$
    $h_{2k-1} = f(\tilde{x}_{2k-1})$ # representation
    $z_{2k-1} = g(h_{2k-1})$ # projection
    # the second augmentation
    $\tilde{x}_{2k} = t'(x_k)$
    $h_{2k} = f(\tilde{x}_{2k})$ # representation
    $z_{2k} = g(h_{2k})$ # projection
  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 $g(\cdot)$
```

Source: Chen et al., 2020
SimCLR

Algorithm 1 SimCLR’s main learning algorithm.

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

Source: Chen et al., 2020

Generate a positive pair by sampling data augmentation functions

InfoNCE loss: Use all non-positive samples in the batch as $x^-$
SimCLR

Source: Chen et al., 2020

**Algorithm 1** SimCLR’s main learning algorithm.

**input:** batch size $N$, constant $\tau$, structure of $f$, $g$, $\mathcal{T}$.

for sampled minibatch $\{x_k\}_{k=1}^{N}$ do
  for all $k \in \{1, \ldots, N\}$ do
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{x}_{2k-1} = t(x_k)$
    $h_{2k-1} = f(\tilde{x}_{2k-1})$  # representation
    $z_{2k-1} = g(h_{2k-1})$  # projection
    # the second augmentation
    $\tilde{x}_{2k} = t'(x_k)$
    $h_{2k} = f(\tilde{x}_{2k})$  # representation
    $z_{2k} = g(h_{2k})$  # projection
  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
  \[
  \ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbbm{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}
  \]
  $L = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks $f$ and $g$ to minimize $L$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$
SimCLR: mini-batch training

Each 2k and 2k + 1 element is a positive pair

\[ s_{i,j} = \frac{z_i^T z_j}{\|z_i\| \|z_j\|} \]

"Affinity matrix"
SimCLR: mini-batch training

Each 2k and 2k + 1 element is a positive pair.

\( s_{i,j} = \frac{z_i^T z_j}{||z_i|| \cdot ||z_j||} \)

"Affinity matrix"

\[ z \in \mathbb{R}^{2N \times D} \]

\[ 2N \]

\[ 2N \]

\[ \boxed{\text{classification label for each row}} \]
Training linear classifier on SimCLR features

Train feature encoder on ImageNet (entire training set) using SimCLR.

Freeze feature encoder, train a linear classifier on top with labeled data.

Source: Chen et al., 2020
Semi-supervised learning on SimCLR features

<table>
<thead>
<tr>
<th>Method</th>
<th>Architecture</th>
<th>Label fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Top 5</td>
</tr>
<tr>
<td>Supervised baseline</td>
<td>ResNet-50</td>
<td>48.4</td>
</tr>
<tr>
<td>Methods using other label-propagation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pseudo-label</td>
<td>ResNet-50</td>
<td>51.6</td>
</tr>
<tr>
<td>VAT+Entropy Min.</td>
<td>ResNet-50</td>
<td>47.0</td>
</tr>
<tr>
<td>UDA (w. RandAug)</td>
<td>ResNet-50</td>
<td>-</td>
</tr>
<tr>
<td>FixMatch (w. RandAug)</td>
<td>ResNet-50</td>
<td>-</td>
</tr>
<tr>
<td>S4L (Rot+VAT+En. M.)</td>
<td>ResNet-50 (4×)</td>
<td>-</td>
</tr>
<tr>
<td>Methods using representation learning only:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InstDisc</td>
<td>ResNet-50</td>
<td>39.2</td>
</tr>
<tr>
<td>BigBiGAN</td>
<td>RevNet-50 (4×)</td>
<td>55.2</td>
</tr>
<tr>
<td>PIRL</td>
<td>ResNet-50</td>
<td>57.2</td>
</tr>
<tr>
<td>CPC v2</td>
<td>ResNet-161(*)</td>
<td>77.9</td>
</tr>
<tr>
<td>SimCLR (ours)</td>
<td>ResNet-50</td>
<td>75.5</td>
</tr>
<tr>
<td>SimCLR (ours)</td>
<td>ResNet-50 (2×)</td>
<td>83.0</td>
</tr>
<tr>
<td>SimCLR (ours)</td>
<td>ResNet-50 (4×)</td>
<td><strong>85.8</strong></td>
</tr>
</tbody>
</table>

Train feature encoder on **ImageNet** (entire training set) using SimCLR.

**Finetune** the encoder with 1% / 10% of labeled data on ImageNet.

Source: [Chen et al., 2020](#)
SimCLR design choices: projection head

Linear / non-linear projection heads improve representation learning.

A possible explanation:
- contrastive learning objective may discard useful information for downstream tasks
- representation space $z$ is trained to be invariant to data transformation.
- by leveraging the projection head $g(\cdot)$, more information can be preserved in the $h$ representation space

Source: Chen et al., 2020
SimCLR design choices: large batch size

Large training batch size is crucial for SimCLR!

Large batch size causes large memory footprint during backpropagation: requires distributed training on TPUs (ImageNet experiments)

Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.\textsuperscript{10}

Source: Chen et al., 2020
Momentum Contrastive Learning (MoCo)

Key differences to SimCLR:

- Keep a running queue of keys (negative samples).
- Compute gradients and update the encoder only through the queries.
- Decouple min-batch size with the number of keys: can support a large number of negative samples.

Source: He et al., 2020
Momentum Contrastive Learning (MoCo)

Key differences to SimCLR:

- Keep a running queue of keys (negative samples).
- Compute gradients and update the encoder only through the queries.
- Decouple min-batch size with the number of keys: can support a large number of negative samples.
- The key encoder is slowly progressing through the momentum update rules:

  \[ \theta_k \leftarrow m\theta_k + (1 - m)\theta_q \]

Source: He et al., 2020
MoCo

Generate a positive pair by sampling data augmentation functions

No gradient through the positive sample

Update the FIFO negative sample queue

Use the running queue of keys as the negative samples

InfoNCE loss

Update f_k through momentum

Source: He et al., 2020
“MoCo V2”

Improved Baselines with Momentum Contrastive Learning

Xinlei Chen  Haoqi Fan  Ross Girshick  Kaiming He
Facebook AI Research (FAIR)

A hybrid of ideas from SimCLR and MoCo:

- **From SimCLR**: non-linear projection head and strong data augmentation.
- **From MoCo**: momentum-updated queues that allow training on a large number of negative samples (no TPU required!).

Source: Chen et al., 2020
MoCo vs. SimCLR vs. MoCo V2

Key takeaways:
- Non-linear projection head and strong data augmentation are crucial for contrastive learning.

Table 1. Ablation of MoCo baselines, evaluated by ResNet-50 for (i) ImageNet linear classification, and (ii) fine-tuning VOC object detection (mean of 5 trials). “MLP”: with an MLP head; “aug+”: with extra blur augmentation; “cos”: cosine learning rate schedule.

<table>
<thead>
<tr>
<th>case</th>
<th>unsup. pre-train</th>
<th>ImageNet acc.</th>
<th>VOC detection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MLP</td>
<td>aug+</td>
<td>cos</td>
</tr>
<tr>
<td>supervised</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MoCo v1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(c)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(d)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(e)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Source: Chen et al., 2020
MoCo vs. SimCLR vs. MoCo V2

<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 [2]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>200</td>
<td>8192</td>
<td>66.6</td>
</tr>
<tr>
<td>MoCo v2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>200</td>
<td>256</td>
<td><strong>67.5</strong></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).

**Key takeaways:**

- Non-linear projection head and strong data augmentation are crucial for contrastive learning.
- Decoupling mini-batch size with negative sample size allows MoCo-V2 to outperform SimCLR with smaller batch size (256 vs. 8192).

Source: Chen et al., 2020
MoCo vs. SimCLR vs. MoCo V2

Key takeaways:

- Non-linear projection head and strong data augmentation are crucial for contrastive learning.

- Decoupling mini-batch size with negative sample size allows MoCo-V2 to outperform SimCLR with smaller batch size (256 vs. 8192).

- ... all with much smaller memory footprint! (“end-to-end” means SimCLR here)

<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.

Source: Chen et al., 2020
Instance vs. Sequence Contrastive Learning

Instance-level contrastive learning: contrastive learning based on positive & negative instances. Examples: SimCLR, MoCo

Sequence-level contrastive learning: contrastive learning based on sequential / temporal orders. Example: Contrastive Predictive Coding (CPC)

Source: van den Oord et al., 2018
Contrastive Predictive Coding (CPC)

Contrastive: contrast between “right” and “wrong” sequences using contrastive learning.

Predictive: the model has to predict future patterns given the current context.

Coding: the model learns useful feature vectors, or “code”, for downstream tasks, similar to other self-supervised methods.

Source: van den Oord et al., 2018,

Figure source
Contrastive Predictive Coding (CPC)

1. Encode all samples in a sequence into vectors $z_t = g_{\text{enc}}(x_t)$

Source: van den Oord et al., 2018,
Contrastive Predictive Coding (CPC)

1. Encode all samples in a sequence into vectors $z_t = g_{enc}(x_t)$

2. Summarize context (e.g., half of a sequence) into a context code $c_t$ using an auto-regressive model ($g_{ar}$).

Figure source: van den Oord et al., 2018,
Contrastive Predictive Coding (CPC)

1. Encode all samples in a sequence into vectors $z_t = g_{enc}(x_t)$

2. Summarize context (e.g., half of a sequence) into a context code $c_t$ using an auto-regressive model ($g_{ar}$).

3. Compute InfoNCE loss between the context $c_t$ and future code $z_{t+k}$ using the following time-dependent score function:

$$s_k(z_{t+k}, c_t) = z_{t+k}^T W_k c_t$$

, where $W_k$ is a trainable matrix.

Source: van den Oord et al., 2018,
CPC example: modeling audio sequences

Source: van den Oord et al., 2018,
CPC example: modeling audio sequences

Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

<table>
<thead>
<tr>
<th>Method</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phone classification</strong></td>
<td></td>
</tr>
<tr>
<td>Random initialization</td>
<td>27.6</td>
</tr>
<tr>
<td>MFCC features</td>
<td>39.7</td>
</tr>
<tr>
<td>CPC</td>
<td>64.6</td>
</tr>
<tr>
<td>Supervised</td>
<td>74.6</td>
</tr>
<tr>
<td><strong>Speaker classification</strong></td>
<td></td>
</tr>
<tr>
<td>Random initialization</td>
<td>1.87</td>
</tr>
<tr>
<td>MFCC features</td>
<td>17.6</td>
</tr>
<tr>
<td>CPC</td>
<td>97.4</td>
</tr>
<tr>
<td>Supervised</td>
<td>98.5</td>
</tr>
</tbody>
</table>

Linear classification on trained representations (LibriSpeech dataset)

Source: van den Oord et al., 2018,
CPC example: modeling visual context

Idea: split image into patches, model rows of patches from top to bottom as a sequence. I.e., use top rows as context to predict bottom rows.

Source: van den Oord et al., 2018,
CPC example: modeling visual context

- Compares favorably with other pretext task-based self-supervised learning method.
- Doesn’t do as well compared to newer instance-based contrastive learning methods on image feature learning.

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1 ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using AlexNet conv5</td>
<td></td>
</tr>
<tr>
<td>Video [28]</td>
<td>29.8</td>
</tr>
<tr>
<td>BiGan [35]</td>
<td>34.8</td>
</tr>
<tr>
<td>Colorization [10]</td>
<td>35.2</td>
</tr>
<tr>
<td>Jigsaw [29] *</td>
<td>38.1</td>
</tr>
<tr>
<td>Using ResNet-V2</td>
<td></td>
</tr>
<tr>
<td>Motion Segmentation [36]</td>
<td>27.6</td>
</tr>
<tr>
<td>Exemplar [36]</td>
<td>31.5</td>
</tr>
<tr>
<td>Relative Position [36]</td>
<td>36.2</td>
</tr>
<tr>
<td>Colorization [36]</td>
<td>39.6</td>
</tr>
<tr>
<td>CPC</td>
<td>48.7</td>
</tr>
</tbody>
</table>

Table 3: ImageNet top-1 unsupervised classification results. *Jigsaw is not directly comparable to the other AlexNet results because of architectural differences.

Source: van den Oord et al., 2018
Summary: Contrastive Representation Learning

A general formulation for contrastive learning:

\[
\text{score}(f(x), f(x^+)) \gg \text{score}(f(x), f(x^-))
\]

InfoNCE loss: N-way classification among positive and negative samples

\[
L = -\mathbb{E}_x \left[ \log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]
\]

Commonly known as the InfoNCE loss \(\text{van den Oord et al., 2018}\)

A lower bound on the mutual information between \(f(x)\) and \(f(x^+)\)

\[
MI[f(x), f(x^+)] - \log(N) \geq -L
\]
**Summary: Contrastive Representation Learning**

**SimCLR**: a simple framework for contrastive representation learning

- **Key ideas**: non-linear projection head to allow flexible representation learning
- Simple to implement, effective in learning visual representation
- Requires large training batch size to be effective; large memory footprint
**Summary: Contrastive Representation Learning**

**MoCo (v1, v2):** contrastive learning using momentum sample encoder
- Decouples negative sample size from minibatch size; allows large batch training without TPU
- MoCo-v2 combines the key ideas from SimCLR, i.e., nonlinear projection head, strong data augmentation, with momentum contrastive learning
Summary: Contrastive Representation Learning

**CPC**: sequence-level contrastive learning
- Contrast “right” sequence with “wrong” sequence.
- InfoNCE loss with a time-dependent score function.
- Can be applied to a variety of learning problems, but not as effective in learning image representations compared to instance-level methods.
Other examples

Contrastive learning between image and natural language sentences

1. Contrastive pre-training

Contrastive Language–Image Pre-training (CLIP) Radford et al., 2021

2. Create dataset classifier from label text

3. Use for zero-shot prediction
Other examples

Contrastive learning on pixel-wise feature descriptors

Dense Object Net, Florence et al., 2018
Other examples

Dense Object Net, Florence et al., 2018
Vision and Language Models: Connecting the Pixel and Semantic Worlds at Scale
Why Vision-Language Models?

- Language is the most intuitive interface for an unstructured data space (e.g., natural images)
- Important to ground sensory information to semantic concepts
- Complementary information sources for a given task
- Claim: you cannot learn language without grounding
History: the first captioning model (Ordonez, 2011)

Im2Text: Describing Images Using 1 Million Captioned Photographs

Vicente Ordonez  Girish Kulkarni  Tamara L Berg
Stony Brook University
Stony Brook, NY 11794
{vordonezroma or tlberg}@cs.stonybrook.edu

Abstract
History: the first captioning model (Ordonez, 2011)

Query image

Gist + Tiny images ranking

Extract High Level Information

Top re-ranked images

Top associated captions
Across the street from Yannicks apartment. At night the headlight on the handlebars above the door lights up.
The building in which I live. My window is on the right on the 4th floor.
This is the car I was in after they had removed the roof and successfully removed me to the ambulance.
I really like doors. I took this photo out of the car window while driving by a church in Pennsylvania.
History: the first deep captioning model (Vinyals, 2015)

Show and Tell: A Neural Image Caption Generator

Oriol Vinyals
Google
vinyals@google.com

Alexander Toshev
Google
toshev@google.com

Samy Bengio
Google
bengio@google.com

Dumitru Erhan
Google
dumitru@google.com
History: the first deep captioning model (Vinyals, 2015)
History: the first VQA model (Agrawal, 2015)

VQA: Visual Question Answering
www.visualqa.org


Abstract—We propose the task of free-form and open-ended Visual Question Answering (VQA). Given an image and a natural language question about the image, the task is to provide an accurate natural language answer. Mirroring real-world scenarios, such as helping the visually impaired, both the questions and answers are open-ended. Visual questions selectivity target different areas of an image, including background details and underlying context. As a result, a system that succeeds at VQA typically needs a more detailed understanding of the image and complex reasoning than a system producing generic image captions. Moreover, VQA is amenable to automatic evaluation, since many open-ended answers contain only a few words or a closed set of answers that can be provided in a multiple-choice format. We provide a dataset containing ~0.25M images, ~0.76M questions, and ~10M answers (www.visualqa.org), and discuss the information it provides. Numerous baselines and methods for VQA are provided and compared with human performance. Our VQA demo is available on CloudCV (http://cloudcv.org/vqa).
History: the first VQA model (Agrawal, 2015)
Foundation VLM (2019-)

Hand-drawn sketch to website source code
GPT 4v(ision) (OpenAI, 2023)
Major Areas

- **Representation**: how to convert raw data into meaningful features
- **Translation**: transform one modality to another
- **Alignment**: discover relationships between elements across modalities
- **Fusion**: join features from modalities to support prediction
- **Co-learning**: transferring knowledge from one modality to another
Language->Vision: Language-guided Image Gen

**TEXT DESCRIPTION**

An astronaut  Teddy bears  A bowl of soup

riding a horse  lounging in a tropical resort in space  playing basketball with cats in space

in a photorealistic style  in the style of Andy Warhol  as a pencil drawing

---

https://openai.com/dall-e-2/
A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track
Image – Language Association

Contrastive learning between image and natural language sentences

1. Contrastive pre-training

2. Create dataset classifier from label text

3. Use for zero-shot prediction

CLIP (Contrastive Language–Image Pre-training) Radford et al., 2021
Image – language encoding architectures

**Associative**

- NCE
- $f_\phi$
- $f_\theta$
- image
- text

**Joint**

- $f_\theta$
- image
- text

- text
CLIP: Associative Encoding

CLIP (Contrastive Language–Image Pre-training) Radford et al., 2021
CLIP: Training

```python
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_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_i]
T_f = text_encoder(T) # [n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# 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: Zero-shot Classification

CLIP (Contrastive Language–Image Pre-training) Radford et al., 2021
CLIP: Zero-shot Classification

```python
# Load the model
device = "cuda" if torch.cuda.is_available() else "cpu"
model, preprocess = clip.load('ViT-B/32', device)

# Download the dataset
cifar100 = CIFAR100(root=os.path.expanduser("~/.cache"), download=True, train=False)

# Prepare the inputs
image, class_id = cifar100[3637]
image_input = preprocess(image).unsqueeze(0).to(device)
text_inputs = torch.cat([clip.tokenize(f"a photo of a {c}" for c in cifar100.classes)].to(device)

# Calculate features
with torch.no_grad():
    image_features = model.encode_image(image_input)
text_features = model.encode_text(text_inputs)

# Pick the top 5 most similar labels for the image
image_features /= image_features.norm(dim=-1, keepdim=True)
text_features /= text_features.norm(dim=-1, keepdim=True)
similarity = (100.0 * image_features @ text_features.T).softmax(dim=-1)
values, indices = similarity[0].topk(5)
```

https://github.com/openai/CLIP
CLIP: Zero-shot Classification

PatchCamelyon (PCam)
healthy lymph node tissue (77.2%) Ranked 2 out of 2 labels

X: this is a photo of lymph node tumor tissue
✓: this is a photo of healthy lymph node tissue

ImageNet-A (Adversarial)
lyne (47.9%) Ranked 5 out of 200 labels

X: a photo of a fox equalled
X: a photo of a mongoose.
X: a photo of a shark.
X: a photo of a red fox.
✓: a photo of a lynx.

CIFAR-10
bird (40.3%) Ranked 1 out of 10 labels

✓: a photo of a bird.
X: a photo of a cali.
X: a photo of a deer.
X: a photo of a frog.
X: a photo of a dog.

CLEVR Count
4 (75.0%) Ranked 2 out of 8 labels

X: a photo of 3 objects.
✓: a photo of 4 objects.
X: a photo of 5 objects.
X: a photo of 6 objects.
X: a photo of 10 objects.

CLIP (Contrastive Language–Image Pre-training) Radford et al., 2021
Generating Images from CLIP Latents (DALL-E 2)

- Train image diffusion with classifier-free guidance using CLIP image embedding
- Train another diffusion model to predict CLIP image embedding from the CLIP embedding of the input text.

Hierarchical Text-Conditional Image Generation with CLIP Latents (Ramesh, Dhariwal, Nichol, Chu, Chen, 2022)
Generating Images from CLIP Latents (DALL-E 2)

Learning objective for the text to image CLIP embedding diffusion model:

$$L_{\text{prior}} = \mathbb{E}_{t \sim [1, T], z^{(t)}_i \sim q_t} \left[ \| f_{\theta}(z^{(t)}_i, t, y) - z^*_i \|^2 \right]$$

Hierarchical Text-Conditional Image Generation with CLIP Latents (Ramesh, Dhariwal, Nichol, Chu, Chen, 2022)
Joint Encodings: ViLBERT (2019)

Vision and Language Joint Pretraining

ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks (Lu et al., 2019)
Joint Encodings: ViLBERT (2019)

ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks (Lu et al., 2019)
Joint Encodings: ViLBERT (2019)

(a) Masked multi-modal learning
(b) Multi-modal alignment prediction

Vision and Language Joint Pretraining

ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks (Lu et al., 2019)
Joint Encodings: ViLT (2021)

Categories of vision-language model in terms of model complexity / capacity

ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision (Kim and Son, 2021)
Joint Encodings: ViLT (2021)

ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision (Kim and Son, 2021)
Data matters
Scaling Up Foundation Vision and Language Models
Pre-foundation model era (2015 – 2020)

Who is wearing glasses?
- man
- woman

Where is the child sitting?
- fridge
- arms

Is the umbrella upside down?
- yes
- no

How many children are in the bed?
- 2
- 1

Visual Question Answering
(Goyal and Knot, 2017)

Image Captioning
(MS-COCO)
Pre-foundation model era (2015 – 2020)

Q: Are there an equal number of large things and metal spheres?  
Q: What size is the cylinder that is left of the brown metal thing that is left of the big sphere?  
Q: There is a sphere with the same size as the metal cube; is it made of the same material as the small red sphere?  
Q: How many objects are either small cylinders or metal things?

Diagnostic Language and Visual Reasoning  
(CLEVR, Johnson et al., 2016)
The “Foundation Model Era” (2020-now)

- **LAION-400M**: 400 million image-text pairs
- Built using Common Crawl datasets,
- Extracting image-text pairs from HTML data.
- Post-processing filters unsuitable pairs using OpenAI's CLIP model.
- A10TB webdataset with CLIP embeddings and kNN indices.
The “Foundation Model Era” (2020-now)

- **LAION-5B**: Significantly larger than LAION-400M
- Crawled using 50 billion webpages + CLIP filtering
- 2.3 billion pairs in English + 2.2 billions in other languages + 1 billion unassignable languages (e.g., names).
The “Foundation Model Era” (2020-now)

Stable Diffusion

Stable Diffusion was made possible thanks to a collaboration with Stability AI and Runway and builds upon our previous work:

High-Resolution Image Synthesis with Latent Diffusion Models
Robin Rombach*, Andreas Blattmann*, Dominik Lorenz, Patrick Esser, Björn Ommer
CVPR ’22 Oral | GitHub | arXiv | Project page

Stable Diffusion is a latent text-to-image diffusion model. Thanks to a generous compute donation from Stability AI and support from LAION, we were able to train a Latent Diffusion Model on 512x512 images from a subset of the LAION-5B database. Similar to Google’s Imagen, this model uses a frozen CLIP ViT-L/14 text encoder to condition the model on text prompts. With its 860M UNet and 123M text encoder, the model is relatively lightweight and runs on a GPU with at least 10GB VRAM. See this section below and the model card.
A snapshot of vision-language dataset
Automatic data crawling is great but …

<table>
<thead>
<tr>
<th>tomclancysthedivision2_gc18images_0001</th>
<th>Enchantments-JUN16-13.jpg</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>They Shall Not Grow Old</strong>*. Watching Peter Jackson tinker with WW1 is like watching George Lucas tinker with <strong>Star Wars</strong>*. Only way more offensive. pic.twitter.com/PkteSrh9tR***</td>
<td></td>
</tr>
<tr>
<td>The International Code Council (ICC) has ratified a change to the 2021 International Building Code (IBC) to allow the use of shipping containers in commercial construction. Photo © <a href="http://www.bigstockphoto.com">www.bigstockphoto.com</a></td>
<td></td>
</tr>
</tbody>
</table>

https://laion-aesthetic.datasette.io/laion-aesthetic-6pls/images?_next=300
Composing Vision and Language Models
How to compose trained L and V models?
How to compose *trained* L and V models?

Fast finetuning

```
\[ f_\theta \rightarrow f_\phi \rightarrow \text{answer} \]
```

Language as interface

```
\[ f_\theta \rightarrow \text{text} \rightarrow f_\phi \rightarrow \text{answer} \]
```
Finetuning VLM: Frozen LM, finetune VM

• Train image encoder with frozen language model.
• At test time, can do 0-shot VQA or few-shot classification through in-context learning

Multimodal Few-Shot Learning with Frozen Language Models (Tsimpoukelli et al., 2021)
Finetuning VLM: Frozen LM, finetune VM

- Train image encoder with frozen language model.
- At test time, can do 0-shot VQA or few-shot classification through in-context learning

Multimodal Few-Shot Learning with Frozen Language Models (Tsimpoukelli et al., 2021)
### Finetuning VLM: Frozen LM, finetune VM

<table>
<thead>
<tr>
<th>n-shot Acc.</th>
<th>n=0</th>
<th>n=1</th>
<th>n=4</th>
<th>(\tau)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frozen</td>
<td>29.5</td>
<td>35.7</td>
<td>38.2</td>
<td>(\times)</td>
</tr>
<tr>
<td>Frozen (_{\text{scratch}})</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>(\times)</td>
</tr>
<tr>
<td>Frozen (_{\text{finetuned}})</td>
<td>24.0</td>
<td>28.2</td>
<td>29.2</td>
<td>(\times)</td>
</tr>
<tr>
<td>Frozen (_{\text{train-blind}})</td>
<td>26.2</td>
<td>33.5</td>
<td>33.3</td>
<td>(\times)</td>
</tr>
<tr>
<td>Frozen (_{\text{VQA}})</td>
<td>48.4</td>
<td>–</td>
<td>–</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>Frozen (_{\text{VQA-blind}})</td>
<td>39.1</td>
<td>–</td>
<td>–</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>Oscar [23]</td>
<td>73.8</td>
<td>–</td>
<td>–</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>(\tau) Frozen</td>
<td>5.9</td>
<td>9.7</td>
<td>12.6</td>
<td>(\times)</td>
</tr>
<tr>
<td>Frozen (_{400mLM})</td>
<td>4.0</td>
<td>5.9</td>
<td>6.6</td>
<td>(\times)</td>
</tr>
<tr>
<td>Frozen (_{\text{finetuned}})</td>
<td>4.2</td>
<td>4.1</td>
<td>4.6</td>
<td>(\times)</td>
</tr>
<tr>
<td>Frozen (_{\text{train-blind}})</td>
<td>3.3</td>
<td>7.2</td>
<td>0.0</td>
<td>(\times)</td>
</tr>
<tr>
<td>Frozen (_{\text{VQA}})</td>
<td>19.6</td>
<td>–</td>
<td>–</td>
<td>(\times)</td>
</tr>
<tr>
<td>Frozen (_{\text{VQA-blind}})</td>
<td>12.5</td>
<td>–</td>
<td>–</td>
<td>(\times)</td>
</tr>
<tr>
<td>MAVEX [42]</td>
<td>39.4</td>
<td>–</td>
<td>–</td>
<td>(\checkmark)</td>
</tr>
</tbody>
</table>

- Training large VLM from scratch does not work at all
- Finetuning LM degrades performance
- “Blind” baselines till works, showing the innate power of LM

Multimodal Few-Shot Learning with Frozen Language Models (Tsimpoukelli et al., 2021)
Finetuning VLM: freeze both LM and VM

- Interleaved text-image input
- Only finetune the cross attention (XATTN-DENSE) layers

Flamingo: a Visual Language Model for Few-Shot Learning (Alayrac et al., 2022)
Finetuning VLM: freeze both LM and VM

- Largely outperforms previous zero/few shot SotA
- More in-context learning examples do help
- Larger model gives better results

Flamingo: a Visual Language Model for Few-Shot Learning (Alayrac et al., 2022)
Finetuning VLM: freeze both LM and VM

Freeze VM and LM. Train the linear layer and LORA finetune Llama 2

MiniGPT-v2: large language model as a unified interface for vision-language multi-task learning (Chen et al., 2023)
Low-rank finetuning (LORA) quickly finetune a billion-parameter model

**Problem**: finetuning still takes a lot of data, especially if the model is huge and/or the domain gap is large.

**Fact**: finetuning is just adding a $W_\delta$ to the existing weight matrix $W$, i.e., $W^* = W + W_\delta$

**Hypothesis**: $W_\delta$ is low-rank, meaning that $W_\delta$ can be decomposed into two smaller matrices $A$ and $B$, i.e., $W_\delta = A^T B$.

**So what?**: $A$ and $B$ have a lot fewer parameters than the full $W$. Requires less data and faster to train.

Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models.”, 2021
Low-rank finetuning (LORA)
quickly finetune a billion-parameter model

Parameter-Efficient Fine-Tuning (PEFT) methods enable efficient adaptation of pre-trained language models (PLMs) to various downstream applications without fine-tuning all the model’s parameters. Fine-tuning large-scale PLMs is often prohibitively costly. In this regard, PEFT methods only fine-tune a small number of (extra) model parameters, thereby greatly decreasing the computational and storage costs. Recent State-of-the-Art PEFT techniques achieve performance comparable to that of full fine-tuning.

Seamlessly integrated with 📦 Accelerate for large scale models leveraging DeepSpeed and Big Model Inference.

Supported methods:
1. LoRA: LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS
2. Prefix Tuning: Prefix-Tuning: Optimizing Continuous Prompts for Generation, P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks
3. P-Tuning: GPT Understands, Too
4. Prompt Tuning: The Power of Scale for Parameter-Efficient Prompt Tuning
5. AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning
6. (J.A): Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning
7. MultiTask Prompt Tuning: Multitask Prompt Tuning Enables Parameter-Efficient Transfer Learning
9. LoKr: Krona: Parameter Efficient Tuning with Kronecker Adapter based on Navigating Text-To-Image Customization From sCyCoris Fine-Tuning to Model Evaluation implementation

```python
import torch
from peft import inject_adapter_in_model, LoraConfig

class DummyModel(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.embedding = torch.nn.Embedding(10, 10)
        self.linear = torch.nn.Linear(10, 10)
        self.lm_head = torch.nn.Linear(10, 10)

    def forward(self, input_ids):
        x = self.embedding(input_ids)
        x = self.linear(x)
        x = self.lm_head(x)
        return x

lora_config = LoraConfig(
    lora_alpha=16,
    lora_dropout=0.1,
    r=64,
    bias="none",
    target_modules=["linear"],
)

model = DummyModel()
model = inject_adapter_in_model(lora_config, model)
dummy_inputs = torch.LongTensor([[0, 1, 2, 3, 4, 5, 6, 7]])
dummy_outputs = model(dummy_inputs)
```

https://github.com/huggingface/peft
How to compose trained L and V models?

Fast finetuning

Language as interface
Neural Module Networks (Andreas et al., 2015)

Idea: train modular networks (attend, classify). Use a controller network to decide how to compose the modules together to solve a task.
## Neural Module Networks (Andreas et al., 2015)

<table>
<thead>
<tr>
<th>How many different lights in various different shapes and sizes?</th>
<th>What is the color of the horse?</th>
<th>What color is the vase?</th>
<th>Is the bus full of passengers?</th>
<th>Is there a red shape above a circle?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>measure[is](attend[red], re-attend[above](attend[circle]))</code></td>
<td>classify[is](attend[bus], attend[full])</td>
<td>classify<a href="attend%5Bvase%5D">color</a></td>
<td>classify<a href="attend%5Bhorse%5D">color</a></td>
<td>measure<a href="attend%5Blight%5D">count</a></td>
</tr>
<tr>
<td>four (four)</td>
<td>green (green)</td>
<td>brown (brown)</td>
<td>measure<a href="attend%5Blight%5D">is</a></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>What is stuffed with toothbrushes wrapped in plastic?</th>
<th>Where does the shabby cat watch a horse eating hay?</th>
<th>What material are the boxes made of?</th>
<th>Is this a clock?</th>
<th>Is a red shape blue?</th>
</tr>
</thead>
<tbody>
<tr>
<td>classify<a href="attend%5Bstuff%5D">what</a></td>
<td>classify<a href="attend%5Bwatch%5D">where</a></td>
<td>classify<a href="attend%5Bbox%5D">material</a></td>
<td>measure<a href="attend%5Bclock%5D">is</a></td>
<td>measure[is](combine[and](attend[red], attend[blue]))</td>
</tr>
<tr>
<td>container (cup)</td>
<td>pen (harness)</td>
<td>leather (cardboard)</td>
<td>yes (no)</td>
<td>yes (no)</td>
</tr>
</tbody>
</table>
Inferring and Executing Programs for Visual Reasoning (Johnson et al., 2017)

Similar to NMN, but train a program generator using REINFORCE. Reward comes from whether the answer is correct.
Visual Programming: Compositional visual reasoning without training (Gupta et al., 2023)
**Visual Programming: Compositional visual reasoning without training** (Gupta et al., 2023)

**Instruction:** Replace the ground with white snow and the bear with a white polar bear

**Prediction:**

<table>
<thead>
<tr>
<th>Action</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJ1=Seg</td>
<td><img src="image" alt="Image" /></td>
</tr>
<tr>
<td>OBJ1=Select</td>
<td><img src="image" alt="Image" /></td>
</tr>
<tr>
<td>OBJ2=Seg</td>
<td><img src="image" alt="Image" /></td>
</tr>
<tr>
<td>OBJ3=Select</td>
<td><img src="image" alt="Image" /></td>
</tr>
<tr>
<td>IMAGE=Replace</td>
<td><img src="image" alt="Image" /></td>
</tr>
</tbody>
</table>

```latex
\text{OBJ1}=\text{Seg}(\text{image}=\text{IMAGE})
\text{OBJ1}=\text{Select}(\text{image}=\text{IMAGE}, \text{object}=\text{OBJ1}, \text{query}=\text{ground}')
\text{OBJ2}=\text{Seg}(\text{image}=\text{IMAGE})
\text{OBJ3}=\text{Select}(\text{image}=\text{IMAGE}, \text{object}=\text{OBJ3}, \text{query}=\text{bear}')
\text{IMAGE}=\text{Replace}(\text{image}=\text{IMAGE}, \text{object}=\text{OBJ1}, \text{prompt}=\text{white snow}')
```
Use large language models (LLMs) to generate program-like semantic plans from natural language command.
VoxPoser (Huang et al., 2023): Program to Grounded Actions

Use LLMs to guide VMs to find where to act next in a 3D scene.
VoxPoser (Huang et al., 2023): Program to Grounded Actions

“Sort the paper trash into the blue tray.”
Summary: Large Vision and Language Models

- Very active field of research, with a history as long as modern deep learning (2011 -)
- Foundation vision and language models have revolutionized the research paradigm post 2019.
- Trending towards larger model and dataset.
- Many active research on how to finetune / adapt VLMs with small amount of compute / data.
- The future is going to be multimodal.