Representation Learning
Example in image representation

Train a neural network (ResNet) on ImageNet (1M data, 1000 classes)

**Representation (feature extractor):**
The mapping from image to the second-to-the-last layer.

Fix the representation, just re-train the last linear layer.
Example in image representation

Source tasks (for training representation): ImageNet

Target task:
Few-shot Learning on VOC07 dataset (20 classes, 1-8 examples per class)

- Without representation learning: 5% - 10% (random guess = 5%)
- With representation learning: 50% - 80%
Example in image representation

Source tasks (for training representation): ImageNet

Target task:
Few-shot Learning on VOC07 dataset (20 classes, 1-8 examples per class)

- Without representation learning: 5% - 10% (random guess = 5%)
- With representation learning: 50% - 80%
Examples

Natural Language Processing

Graph Representation Learning

Final hidden state: Sentence representation

Feature representation, embedding

node $u \rightarrow \mathbb{R}^d$

vector

$f: u \rightarrow \mathbb{R}^d$
Representation learning

• A function that maps the raw input to a compact representation (feature vector). Learn an embedding / feature / representation from labeled/unlabeled data.

• Supervised:
  • Multi-task learning
  • Meta-learning
  • Multi-modal learning
  • …

• Unsupervised:
  • PCA
  • ICA
  • Dictionary learning
  • Sparse coding
  • Boltzmann machine
  • Autoencoder
  • Contrastive learning
  • Self-supervised learning
  • …
Desiderata for representations

Many possible answers here.

- **Downstream usability**: the learned features are “useful” for downstream tasks:
  - Example: a linear (or simple) classifier applied on the learned features only requires a small number of labeled samples. A classifier on raw inputs requires a large mount of data.

- **Interpretability**: the learned features are semantically meaningful, interpretable by a human, can be easily evaluated.
  - Not well-defined mathematically.
  - **Sparsity** is an important subcase.
Desiderata for representations

From Bengio, Courville, Vincent ’14:

• **Hierarchy / compositionality:** video/image/text are expected to have hierarchial structure: need *deep* learning.

• **Semantic clusterability:** features of the same “semantic class” (e.g. images in the same class) are clustered together.

• **Linear interpolation:** in the representation space, linear interpolations produce meaningful data points (latent space is convex). Also called *manifold flattening*.

• **Disentanglement:** features capture “independent factors of variation” of data. A popular principle in modern unsupervised learning.
Semantic clustering

**Semantic clusterability:** features of the same “semantic class” (e.g. images in the same class) are clustered together.

![t-SNE projection](image)

**Intuition:** If semantic classes are linearly separable, and labels on downstreams tasks depend linearly on semantic classes: we only need to learn a simple classifier.

t-SNE projection (a data visualization method) of VAE-learned features of 10 MNIST classes.
Linear interpolation

**Linear interpolation**: in the representation space, linear interpolations produce meaningful data points (latent space is convex).

**Intuition**: the data lies on a manifold which is complicated/curved.

The latent variable manifold is a convex set: moving in straight lies is still on it.

Interpolations for a VAE trained feature on MNIST.
**Linear interpolation**

*Linear interpolation:* in the representation space, linear interpolations produce meaningful data points (latent space is convex).

Interpolations for a BigGAN image.
**Disentanglement**

**Disentanglement:** features capture “independent factors of variation” of data (Bengio, Courville, Vincent ’14).

- Very popular in modern unsupervised learning.
- Strong connections with generative models: \( p_\theta(z) = \Pi_i p_\theta(z_i) \).

*Figure 4: Latent factors learnt by β-VAE on celebA: traversal of individual latents demonstrates that β-VAE discovered in an unsupervised manner factors that encode skin colour, transition from an elderly male to younger female, and image saturation.*
Representation Learning Methods
Can we embed words into a latent space?

This embedding came from directly querying for relationships.

*word2vec* is a popular unsupervised learning approach that just uses a text corpus (e.g. nytimes.com)
Word embeddings, word2vec

Source Text

| The quick brown fox jumps over the lazy dog. → |
| The quick brown fox jumps over the lazy dog. → |
| The quick brown fox jumps over the lazy dog. → |
| The quick brown fox jumps over the lazy dog. → |

Training Samples

- (the, quick)
- (the, brown)
- (quick, the)
- (quick, brown)
- (quick, fox)
- (brown, the)
- (brown, quick)
- (brown, fox)
- (brown, jumps)
- (fox, quick)
- (fox, brown)
- (fox, jumps)
- (fox, over)
Training neural network to predict co-occurring words. Use first layer weights as embedding, throw out output layer.

slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
Word embeddings, word2vec

Training neural network to predict co-occurring words. Use first layer weights as embedding, throw out output layer

slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
Self-supervised learning

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible
- Pretend there is a part of the input you don’t know and predict that.
**Transformer Pretraining**

- Collect a large amount of corpus (wiki) and pretrain a large transformer.

- For down-stream tasks, fine-tune the pretrained model:
  - Or use the pretrained model to extract features.

- How to pretrain a transformer on texts?
  - Pretrain an encoder
    - bi-directional
  - Pretrain a decoder
    - auto-regressive
Pre-training Transformer Encoder

- Pre-training a bi-directional encoder
  - Cannot directly adopt language modeling
  - **Idea**: word prediction given contexts (similar to word2vec)

- Masked language model
  - Randomly “masked out” some words
  - Run full transformer encoder
  - Predict the words at masked positions

- Designed for feature extraction
  - Suitable for down-stream tasks
Pre-training Transformer Encoder

- **BERT**: Pre-training of Deep Bidirectional Transformers
  - Devlin et al., Google, 2018
    - BERT-base: 12 layers, 110M params
    - BERT-large: 24 layers, 340M params
    - Training on 64 TPUs in 4 days
    - Fine-tuning can be down in a single GPU
  - Masked language model
    - Masked out input words 80% of the time
    - Replace 10% words with random tokens
    - 10% words remain unchanged
    - Predict 15% of word tokens

![Diagram showing masked language model example](image)
Pre-training Transformer Encoder

- **BERT**: Pre-training of Deep Bidirectional Transformers
  - Devlin et al., Google, 2018
    - BERT-base: 12 layers, 110M params
    - BERT-large: 24 layers, 340M params
    - Training on 64 TPUs in 4 days
    - Fine-tuning can be done in a single GPU
  - Masked language model
    - Masked out input words 80% of the time
    - Replace 10% words with random tokens
    - 10% words remain unchanged

<table>
<thead>
<tr>
<th>System</th>
<th>MNLI-(m/mm)</th>
<th>QQP</th>
<th>QNLI</th>
<th>SST-2</th>
<th>CoLA</th>
<th>STS-B</th>
<th>MRPC</th>
<th>RTE</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-OpenAI SOTA</td>
<td>80.6/80.1</td>
<td>66.1</td>
<td>82.3</td>
<td>93.2</td>
<td>35.0</td>
<td>81.0</td>
<td>86.0</td>
<td>61.7</td>
<td>74.0</td>
</tr>
<tr>
<td>BiLSTM+ELMo+Attn</td>
<td>76.4/76.1</td>
<td>64.8</td>
<td>79.8</td>
<td>90.4</td>
<td>36.0</td>
<td>73.3</td>
<td>84.9</td>
<td>56.8</td>
<td>71.0</td>
</tr>
<tr>
<td>OpenAI GPT</td>
<td>82.1/81.4</td>
<td>70.3</td>
<td>87.4</td>
<td>91.3</td>
<td>45.4</td>
<td>80.0</td>
<td>82.3</td>
<td>56.0</td>
<td>75.1</td>
</tr>
<tr>
<td>BERT_{BASE}</td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>90.5</td>
<td>93.5</td>
<td>52.1</td>
<td>85.8</td>
<td>88.9</td>
<td>66.4</td>
<td>79.6</td>
</tr>
<tr>
<td>BERT_{LARGE}</td>
<td><strong>86.7/85.9</strong></td>
<td><strong>72.1</strong></td>
<td><strong>92.7</strong></td>
<td><strong>94.9</strong></td>
<td><strong>60.5</strong></td>
<td><strong>86.5</strong></td>
<td><strong>89.3</strong></td>
<td><strong>70.1</strong></td>
<td><strong>82.1</strong></td>
</tr>
</tbody>
</table>

[Predict these!]
Pre-training Transformer Encoder

- **BERT**: Pre-training of Deep Bidirectional Transformers
- **RoBERTa**: A robustly optimized BERT Pretraining approach
  - Facebook AI and UW, ’19
  - More compute, data, and improved objective

<table>
<thead>
<tr>
<th>Model</th>
<th>data</th>
<th>bsz</th>
<th>steps</th>
<th>SQuAD (v1.1/2.0)</th>
<th>MNLI-m</th>
<th>SST-2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RoBERTa</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with BOOKS + WIKI</td>
<td>16GB</td>
<td>8K</td>
<td>100K</td>
<td>93.6/87.3</td>
<td>89.0</td>
<td>95.3</td>
</tr>
<tr>
<td>+ additional data (§3.2)</td>
<td>160GB</td>
<td>8K</td>
<td>100K</td>
<td>94.0/87.7</td>
<td>89.3</td>
<td>95.6</td>
</tr>
<tr>
<td>+ pretrain longer</td>
<td>160GB</td>
<td>8K</td>
<td>300K</td>
<td>94.4/88.7</td>
<td>90.0</td>
<td>96.1</td>
</tr>
<tr>
<td>+ pretrain even longer</td>
<td>160GB</td>
<td>8K</td>
<td>500K</td>
<td><strong>94.6/89.4</strong></td>
<td><strong>90.2</strong></td>
<td><strong>96.4</strong></td>
</tr>
<tr>
<td><strong>BERT</strong>&lt;sub&gt;LARGE&lt;/sub&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with BOOKS + WIKI</td>
<td>13GB</td>
<td>256</td>
<td>1M</td>
<td>90.9/81.8</td>
<td>86.6</td>
<td>93.7</td>
</tr>
</tbody>
</table>
Pre-training Decoder

• Decoder Pretraining
  • Just train a language model over corpus.
  • Good for generative task (e.g., text generation)

• Generative Pretrained Transformer (GPT, OpenAI ’18)
  • 120 layers transformer, 7680d hidden, 3072-d MLP
  • Data: BooksCropus (>7k books)

• GPT-2 (Radford et al., OpenAI ’19)
  • 1.5B parameters, 40GB internet texts

• GPT-3 (OpenAI ’20)
  • Language models are few-shot learners
  • 175B parameters

• Also Image GPT (OpenAI ’20)
Pre-training Decoder

- GPT-3 (OpenAI ’20)
  - You may not need to fine-tune the model parameters for downstream tasks.
  - New paradigm: prompt learning

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

<table>
<thead>
<tr>
<th></th>
<th>task description</th>
<th>examples</th>
<th>prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Translate English to French:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>sea otter =&gt; loutre de mer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>peppermint =&gt; menthe poivrée</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>plush giraffe =&gt; girafe peluche</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>cheese =&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Code: `px.line(df.query("continent == 'Europe' and country == 'France'"), x='year', y='gdpPerCap', color='country', log_y=False, log_x=False)`

Description: Actually, replace GDP with population

Code: `px.line(df.query("continent == 'Europe' and country == 'France'"), x='year', y='pop', color='country', log_y=False, log_x=False)`

Description: Put y-axis on log scale

Code: `px.line(df.query("continent == 'Europe' and country == 'France'"), x='year', y='pop', color='country', log_y=True, log_x=False)`
Pre-training Decoder

- A big ongoing race on training large language models
  - Megatron-Turing NLG (530B, Microsoft, ’22)
  - Pathways Language Model (540B, Google, ’22)
Autoencoders

Find a low dimensional representation for your data by predicting your data

Input: $x \in \mathbb{R}^d$

Encoder

Code: $f(x) \in \mathbb{R}^r$

Decoder

Output: $\hat{x} = g(f(x)) \in \mathbb{R}^d$

$$\min_{f,g} \sum_{i=1}^{n} \|x_i - g(f(x_i))\|_2^2$$
Autoencoders

**Input:** $x \in \mathbb{R}^d$

**Code:** $f(x) \in \mathbb{R}^r$

**Output:** $\hat{x} = g(f(x)) \in \mathbb{R}^d$

minimize $\sum_{i=1}^{n} \|x_i - g(f(x_i))\|_2^2$

What if $f(X) = Ax$ and $g(y) = By$?

\[ \cap A \]
Autoencoders

Input: $x \in \mathbb{R}^d$

Code: $f(x) \in \mathbb{R}^r$

Output: $\hat{x} = g(f(x)) \in \mathbb{R}^d$

Minimize:

$$\min_{f,g} \sum_{i=1}^{n} \| x_i - g(f(x_i)) \|_2^2$$

What if $f(X) = Ax$ and $g(y) = By$?
Self-supervised learning in computer vision

Context Prediction (Pathak et al., ‘15)

Figure 1. Our task for learning patch representations involves randomly sampling a patch (blue) and then one of eight possible neighbors (red). Can you guess the spatial configuration for the two pairs of patches? Note that the task is much easier once you have recognized the object! Answer Key: Q1: Bottom right. Q2: Top center.

\[ X = (\text{cat}), \quad Y = 3 \]
Self-supervised learning in computer vision

• **Feature learning by Inpainting** (Pathak et al., ’16)
  • The most obvious analogue to word embeddings: predict parts of image from the remainder of image

Architectures:
An encoder takes a part of an image, constructs a representation.
A decoder takes the representation, tries to reconstruct the missing part.

Trickier than NLP:
1. Meaningful losses for vision are more difficult to design.
2. Choice of region to mask out is important
Self-supervised learning in computer vision

- Feature learning by Inpainting (Pathak et al., '16)

Figure 3: An example of image $x$ with our different region masks $\hat{M}$ applied, as described in Section 3.3.

Fixed region vs. random square block vs. random region
Self-supervised learning in computer vision

- **Image Colorization** (Zhang et al. ’16)

![Diagram showing self-supervised learning process](image)

Input Image $X$ → $F_1$ → Predicted Image $\hat{X}$
Self-supervised learning in computer vision

- **Rotation Prediction** (Gidaris et al., ’18)
Contrastive learning

**Idea:** if features are “semantically” relevant, a “distortion” of an image should produce similar features.

**Framework:**
- For every training sample, produce multiple augmented samples by applying various transformations.
- Train an encoder $E$ to predict whether two samples are augmentations of the same base sample.
- A common way is train $\langle E(x), E(x') \rangle$ big if $x, x'$ are two augmentations of the same sample:

$$
\ell_{x,x'} = - \log \left( \frac{\exp(\tau \langle E(x), E(x') \rangle)}{\sum_{\tilde{x}} \exp(\tau \langle E(x), E(\tilde{x}) \rangle)} \right)
$$

$$
\min_{x,x'} \sum_{x,x'} \ell_{x,x'}
$$

augments of each other
Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., ’18)
- SimCLR (Chen et al. ’20)
  - A simple framework for contrastive learning of visual representations
    - Predefine a set of transformations
    - For a data, sample two transformations
    - Maximum agreement on representations
- No negative pairs explicitly
  - Non-paired data in the batch are negative

![Diagram](image_url)
Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., ’18)
- SimCLR (Chen et al. ’20)

Algorithm 1 SimCLR’s main learning algorithm.

\[
\text{input: batch size } N, \text{ constant } \tau, \text{ structure of } f, g, \mathcal{T}.
\]

\[
\begin{align*}
\text{for sampled minibatch } \{x_k\}_{k=1}^N \text{ do} \\
\text{for all } k \in \{1, \ldots, N\} \text{ do} \\
\quad \text{draw two augmentation functions } t \sim \mathcal{T}, t' \sim \mathcal{T} \\
\quad \# \text{ the first augmentation} \\
\quad \tilde{x}_{2k-1} = t(x_k) \\
\quad h_{2k-1} = f(\tilde{x}_{2k-1}) \\
\quad z_{2k-1} = g(h_{2k-1}) \\
\quad \# \text{ representation} \\
\end{align*}
\]

\[
\begin{align*}
\tilde{x}_{2k} = t'(x_k) \\
\quad \# \text{ the second augmentation} \\
\quad h_{2k} = f(\tilde{x}_{2k}) \\
\quad z_{2k} = g(h_{2k}) \\
\quad \# \text{ projection} \\
\end{align*}
\]

\[
\begin{align*}
\text{for all } i \in \{1, \ldots, 2N\} \text{ and } j \in \{1, \ldots, 2N\} \text{ do} \\
\quad s_{i,j} = z_i^T z_j / (\|z_i\| \cdot \|z_j\|) \\
\quad \# \text{ pairwise similarity} \\
\end{align*}
\]

\[
\begin{align*}
\text{end for} \\
\text{define } \ell(i, j) \text{ as } \ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbf{1}_{[k \neq i]} \exp(s_{i,k}/\tau)} \\
\end{align*}
\]

\[
\begin{align*}
\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)] \\
\text{update networks } f \text{ and } g \text{ to minimize } \mathcal{L} \\
\text{end for} \\
\text{return encoder network } f(\cdot), \text{ and throw away } g(\cdot)
\end{align*}
\]
Contrastive learning

Contrastive Predictive Coding (Van den Oord et al., ’18)
- SimCLR (Chen et al. ’20)

<table>
<thead>
<tr>
<th>Method</th>
<th>Architecture</th>
<th>Label fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised baseline</td>
<td>ResNet-50</td>
<td>48.4 80.4</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 82.4</td>
</tr>
<tr>
<td>VAT+Entropy Min.</td>
<td>ResNet-50</td>
<td>47.0 83.4</td>
</tr>
<tr>
<td>UDA (w. RandAug)</td>
<td>ResNet-50</td>
<td>- 88.5</td>
</tr>
<tr>
<td>FixMatch (w. RandAug)</td>
<td>ResNet-50</td>
<td>- 89.1</td>
</tr>
<tr>
<td>S4L (Rot+VAT+En. M.)</td>
<td>ResNet-50 (4x)</td>
<td>- 91.2</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 77.4</td>
</tr>
<tr>
<td>BigBiGAN</td>
<td>RevNet-50 (4x)</td>
<td>55.2 78.8</td>
</tr>
<tr>
<td>PIRL</td>
<td>ResNet-50</td>
<td>57.2 83.8</td>
</tr>
<tr>
<td>CPC v2</td>
<td>ResNet-161(*)</td>
<td>77.9 91.2</td>
</tr>
<tr>
<td>SimCLR (ours)</td>
<td>ResNet-50</td>
<td>75.5 87.8</td>
</tr>
<tr>
<td>SimCLR (ours)</td>
<td>ResNet-50 (2x)</td>
<td>83.0 91.2</td>
</tr>
<tr>
<td>SimCLR (ours)</td>
<td>ResNet-50 (4x)</td>
<td><strong>85.8</strong> 92.6</td>
</tr>
</tbody>
</table>

*Table 7. ImageNet accuracy of models trained with few labels.*