Topics in AI (CPSC 532L): Multimodal Learning with Vision, Language and Sound

Lecture 4: Convolutional Neural Networks (Part 1)
Course Logistics

- **Azure credits** have been distributed
- Instructions for using Azure are on Piazza
- **Office hours** (Shikib) on Wednesday (January 17th) 1-2pm
- Assignment 1 grading
- Assignment 2 will be out today (on CNNs)

- Start thinking about **papers**
- Start thinking about **project** and forming groups (proposal is in ~month)
Robert Herjavec
(Canadian businessman and investor)
Shark Tank
“A good entrepreneur can take a mediocre idea and make it great, a bad entrepreneur will take the best idea and run it into the ground.”

Robert Herjavec
(Canadian businessman and investor)
Shark Tank
Research Projects

Business

“A good entrepreneur can take a mediocre idea and make it great, a bad entrepreneur will take the best idea and run it into the ground.”

Robert Herjavec
(Canadian businessman and investor)
Shark Tank

Research

“A good idea can make a mediocre project great, a bad idea will take the best project and run it into the ground.” — Me
Research Projects Ideas Levels of Abstraction

I want to **solve vision** / language / etc.

Marvin Minsky
I want to solve vision / language / etc.

I want to do X (e.g., image captioning)

— This is excellent if X is something no one has done or thought about and is important (guaranteed success)
— Requires forward thinking, knowledge of the field
— A sure way to get tenure
— Difficult to do as the field matures
I want to solve vision / language / etc.

I want to do X (e.g., image captioning)

I think the right way to solve (or improve) X is Y
Research Projects Ideas Levels of Abstraction

I want to solve vision / language / etc.

I want to do X (e.g., image captioning)

I think the right way to solve (or improve) X is Y

   — More incremental; a lot of science is incremental ("standing on the shoulders of giants")
Research Projects Ideas Levels of Abstraction

I want to solve vision / language / etc.

I want to do X (e.g., image captioning)

I think the right way to solve (or improve) X is Y

- More incremental; a lot of science is incremental ("standing on the shoulders of giants")
- Retrospective: compare existing approaches see why they work what is missing (guaranteed success)
- Perspective: come up with an idea or the insight that you truly believe and test it
Research Projects Ideas Levels of Abstraction

I want to solve vision / language / etc.

I want to do X (e.g., image captioning)

I think the right way to solve (or improve) X is Y

- More incremental; a lot of science is incremental (“standing on the shoulders of giants”)
- Retrospective: compare existing approaches see why they work what is missing (guaranteed success)
- Perspective: come up with an idea or the insight that you truly believe and test it
- Requires through knowledge of the sub-field (lots of reading)
Research Projects Ideas Levels of Abstraction

I want to **solve vision** / language / etc.

I want to do X (e.g., **image captioning**)

I think the **right way to solve** (or improve) X is Y

- More incremental; a lot of science is incremental (“standing on the shoulders of giants”)
- **Retrospective:** compare existing approaches see why they work what is missing (**guaranteed success**)
- **Perspective:** come up with an idea or the insight that you truly believe and test it
- Requires through knowledge of the sub-field (lots of reading)
- Requires strong intuition and high level (intuitive) thinking
Research Projects Ideas Levels of Abstraction

I want to solve vision / language / etc.

I want to do X (e.g., image captioning)

I think the right way to solve (or improve) X is Y

— More incremental; a lot of science is incremental (“standing on the shoulders of giants”)
— **Retrospective:** compare existing approaches see why they work what is missing *(guaranteed success)*
— **Perspective:** come up with an idea or the insight that you truly believe and test it
— Requires through knowledge of the sub-field (lots of reading)
— Requires strong intuition and high level (intuitive) thinking
— Requires understanding of the mathematical tools and formulations to know what maybe possible
I want to solve vision / language / etc.

I want to do X (e.g., image captioning)

I think the right way to solve (or improve) X is Y

- More incremental; a lot of science is incremental (“standing on the shoulders of giants”)
- Retrospective: compare existing approaches see why they work what is missing (guaranteed success)
- Perspective: come up with an idea or the insight that you truly believe and test it
- Requires through knowledge of the sub-field (lots of reading)
- Requires strong intuition and high level (intuitive) thinking
- Requires understanding of the mathematical tools and formulations to know what maybe possible
- Helps to bringing knowledge from other fields (field cross pollination)
Research Projects Ideas Levels of Abstraction

I want to solve vision / language / etc.

I want to do X (e.g., image captioning)

I think the right way to solve (or improve) X is Y

Mathematical formulation

Implementation / engineering

Experimental testing
On to *today's* lecture ...
Fully Connected Layer

Example: 200 x 200 image (small) x 40K hidden units

* slide from Marc’Aurelio Renzato
**Fully Connected Layer**

**Example:** 200 x 200 image (small) x 40K hidden units = ~ 2 Billion parameters (for one layer!)

*slide from Marc'Aurelio Renzato*
Fully Connected Layer

Example: 200 x 200 image (small) x 40K hidden units = \sim 2 \text{ Billion} \text{ parameters (for one layer!)}

Spatial correlations are generally local

Waste of resources + we don’t have enough data to train networks this large

* slide from Marc’Aurelio Renzato
Locally Connected Layer

Example: 200 x 200 image (small)
  x 40K hidden units

Filter size: 10 x 10

= \sim 4 \text{ Million} \text{ parameters}

* slide from Marc’Aurelio Renzato
Locally Connected Layer

**Example:** 200 x 200 image (small) x 40K hidden units

**Filter size:** 10 x 10

= ~ 4 Million parameters

**Stationarity** — statistics is similar at different locations

* slide from Marc'Aurelio Renzato
Convolutional Layer

**Example:** 200 x 200 image (small) x 40K hidden units

**Filter size:** 10 x 10

= ~ **4 Million** parameters

Share the same parameters across the locations (assuming input is stationary)

* slide adopted from Marc’Aurelio Renzato*
Convolutional Layer

Example: 200 x 200 image (small) x 40K hidden units

Filter size: 10 x 10

\[= \sim 4 \text{ Million} \times \text{parameters}\]
\[= 100 \text{ parameters}\]

Share the same parameters across the locations (assuming input is stationary)

* slide adopted from Marc’Aurelio Renzato
Convolutional Layer

* slide from Marc’Aurelio Renzato
Convolutional Layer

* slide from Marc'Aurelio Renzato
Convolutional Layer

* slide from Marc’Aurelio Renzato
Convolutional Layer

* slide from Marc’Aurelio Renzato
Convolutional Layer

* slide from Marc’Aurelio Renzato
Convolutional Layer

* slide from Marc’Aurelio Renzato
Convolutional Layer

* slide from Marc’Aurelio Renzato
Convolutional Layer

*slide from Marc’Aurelio Renzato*
Convolutional Layer

* slide from Marc’Aurelio Renzato
Convolutional Layer

* slide from Marc’Aurelio Renzato
Convolutional Layer
Convolutional Layer

* slide from Marc’Aurelio Renzato
Convolutional Layer

*slide from Marc’Aurelio Renzato*
Convolutional Layer

* slide from Marc’Aurelio Renzato
Convolutional Layer

* slide from Marc’Aurelio Renzato
Convolution Layer

\[ \begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1
\end{bmatrix} \]
Convolution Layer

\[
\begin{bmatrix}
0.11 & 0.11 & 0.11 \\
0.11 & 0.11 & 0.11 \\
0.11 & 0.11 & 0.11
\end{bmatrix}
\]
Convolutional Layer

Example: 200 x 200 image (small) x 40K hidden units

Filter size: 10 x 10

# of filters: 20

Learn multiple filters

* slide from Marc’Aurelio Renzato
Convolutional Layer

**Example:** 200 x 200 image (small) x 40K hidden units

**Filter size:** 10 x 10

**# of filters:** 20

= 2000 parameters

---

* slide from Marc’Aurelio Renzato
Convolutional Layer

32 x 32 x 3 image (note the image preserves spatial structure)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
**Convolutional Layer**

32 x 32 x 3 image

5 x 5 x 3 filter

Convolve the filter with the image (i.e., “slide over the image spatially, computing dot products”)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer

32 x 32 x 3 image

Filters always extend the full depth of the input volume

32 height

5 x 5 x 3 filter

Convolve the filter with the image (i.e., “slide over the image spatially, computing dot products”)

3 depth

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer

32 x 32 x 3 image

5 x 5 x 3 filter \( (W) \)

1 number: the result of taking a dot product between the filter and a small 5 x 5 x 3 part of the image

\[ W^T x + b, \text{ where } W, x \in \mathbb{R}^{75} \]
Convolutional Layer

32 x 32 x 3 image

5 x 5 x 3 filter (W)

1 number: the result of taking a dot product between the filter and a small 5 x 5 x 3 part of the image

\[ W^T x + b, \text{ where } W, x \in \mathbb{R}^{75} \]

How many parameters does the layer have?

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer

32 x 32 x 3 image

5 x 5 x 3 filter ($W$)

32 width

3 depth

1 number: the result of taking a dot product between the filter and a small 5 x 5 x 3 part of the image

$$W^T x + b, \text{ where } W, x \in \mathbb{R}^{75}$$

How many parameters does the layer have? 76

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer

32 x 32 x 3 image

5 x 5 x 3 filter (W)

convolve (slide) over all spatial locations

activation map

28 width

28 height

1 depth

32 width

3 depth

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer

32 x 32 x 3 image

5 x 5 x 3 filter \( (W) \)

convolve (slide) over all spatial locations

consider another green filter

activation map

32 width

1 depth

5 x 5 x 3 filter

28 height

28 width

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer

If we have 6 5x5 filter, we’ll get 6 separate activation maps: *activation map*

32 width

28 height

6 depth

3 depth

this results in the “new image” of size 28 x 28 x 6!

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Neural Network (ConvNet)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
What **filters** do networks learn?

[Zeiler and Fergus, 2013]
What **filters** do networks learn?

[ Zeiler and Fergus, 2013 ]
Convolutional Layer: Closer Look at Spatial Dimensions

32 x 32 x 3 image

5 x 5 x 3 filter (W)

convolve (slide) over all spatial locations

activation map

28 height

32 width

1 depth

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer: Closer Look at **Spatial Dimensions**

7 width

7 x 7 input image (spatially)
3 x 3 filter

7 height

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer: Closer Look at **Spatial Dimensions**

- 7 width
- 7 x 7 input image (spatially)
- 3 x 3 filter
- 7 height

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer: Closer Look at Spatial Dimensions

7 width

7 x 7 input image (spatially)
3 x 3 filter

7 height

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer: Closer Look at **Spatial Dimensions**

- **7 width**
- **7 x 7 input image (spatially)**
- **3 x 3 filter**
- **7 height**

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer: Closer Look at **Spatial Dimensions**

7 width

7 x 7 input image (spatially)
3 x 3 filter

7 height

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer: Closer Look at **Spatial Dimensions**

7 width

7 x 7 input image (spatially)
3 x 3 filter

=> 5 x 5 output

7 height

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, *cs231n Stanford*
Convolutional Layer: Closer Look at **Spatial Dimensions**

7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 2**)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer: Closer Look at **Spatial Dimensions**

7 width

7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 2**)

7 height

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs213 Stanford
Convolutional Layer: Closer Look at **Spatial Dimensions**

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford

- 7 width
- 7 x 7 input image (spatially)
- 3 x 3 filter
- (applied with **stride 2**)

7 height
Convolutional Layer: Closer Look at **Spatial Dimensions**

7 width

7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 2**)

=> 3 x 3 output

7 height

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer: Closer Look at **Spatial Dimensions**

- **7 width**
- **7 x 7 input image (spatially)**
- **3 x 3 filter**
- (applied with **stride 3**)

- **7 height**

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer: Closer Look at Spatial Dimensions

7 width

7 x 7 input image (spatially)
3 x 3 filter
(applied with stride 3)

7 height

Does not fit! **Cannot apply** 3 x 3 filter on 7 x 7 image with stride 3

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer: Closer Look at **Spatial Dimensions**

N x N input image (spatially)
F x F filter

**Output size:** \((N-F) / \text{stride} + 1\)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs213n Stanford
Convolutional Layer: Closer Look at Spatial Dimensions

N width

N x N input image (spatially)
F x F filter

Output size: \((N-F) / \text{stride} + 1\)

Example: \(N = 7, F = 3\)

\[
\begin{align*}
\text{stride 1} & \Rightarrow (7-3)/1+1 = 5 \\
\text{stride 2} & \Rightarrow (7-3)/2+1 = 3 \\
\text{stride 3} & \Rightarrow (7-3)/3+1 = 2.33
\end{align*}
\]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
### Convolutional Layer: **Border padding**

**Input Image**: 5 x 5 (spatially)

**Filter**: 3 x 3

*applied with stride 1*

**Output Size**: 7 x 7

---

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Padding**: pad with 1 pixel border

---

*slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford*
Convolutional Layer: **Border padding**

- **7 width**
- **5 x 5 input image (spatially)**
- **3 x 3 filter**
- (applied with **stride 3**)
- **pad** with 1 pixel border

- **7 height**
Convolutional Layer: **Border padding**

- 5 x 5 input image (spatially)
- 3 x 3 filter
  - (applied with *stride 3*)
- **pad** with 1 pixel border

**Example:** \( N = 7, F = 3 \)

- Stride 1 \( \Rightarrow \frac{(9-3)}{1}+1 = 7 \)
- Stride 2 \( \Rightarrow \frac{(9-3)}{2}+1 = 4 \)
- Stride 3 \( \Rightarrow \frac{(9-3)}{3}+1 = 3 \)
**Convolutional Neural Network (ConvNet)**

- **32 height**
- **32 width**
- **3 depth**

**CONV, ReLU**

E.g. **6 5x5x3 filters**

*slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford*
Convolutional Neural Network (ConvNet)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Neural Network (ConvNet)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Neural Network (ConvNet)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Neural Network (ConvNet)

32 width, 32 height, 3 depth

CONV, ReLU
e.g. 6 5x5x3 filters

28 width, 28 height, 6 depth

CONV, ReLU
e.g. 10 5x5x6 filters

24 width, 24 height, 10 depth

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Neural Network (ConvNet)

With padding we can achieve no shrinking (32 -> 28 -> 24); shrinking quickly (which happens with larger filters) doesn’t work well in practice.

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Layer: **1x1 convolutions**

56 x 56 x 64 image → 56 x 56 x 32 image

- 32 filters of size, 1 x 1 x 64

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Neural Networks

VGG-16 Network
**CNNs: Reminder Fully Connected Layers**

Input

3072

(32 x 32 x 3 image -> stretches to 3072 x 1)

\[ W^T x + b, \text{ where } W \in \mathbb{R}^{10 \times 3072} \]

each neuron looks at the full input volume

Activation

10

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Neural Networks

VGG-16 Network
Convolutional Neural Networks

VGG-16 Network
CNNs: Reminder Fully Connected Layers

\[ w^T x + b, \text{ where } W \in \mathbb{R}^{25,088 \times 4,096} \]

Input: 25,088
(7 x 7 x 512 image -> stretches to 25,088 x 1)

Activation: 4,096

each neuron looks at the full input volume

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
**CNNs: Reminder Fully Connected Layers**

$$W^T x + b, \text{ where } W \in \mathbb{R}^{25,088 \times 4,096}$$

Each neuron looks at the full input volume.

Input: 25,088

(7 x 7 x 512 image -> stretches to 25,088 x 1)

Activation: 4,096

102,760,448 parameters!

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Convolutional Neural Networks

VGG-16 Network
Convolutional Neural Networks

VGG-16 Network
Let us assume the filter is an “eye” detector.

How can we make detection spatially invariant (insensitive to position of the eye in the image)?

* slide from Marc’Aurelio Renzato
Let us assume the filter is an “eye” detector

How can we make detection spatially invariant (insensitive to position of the eye in the image)

By “pooling” (e.g., taking a max) response over a spatial locations we gain robustness to position variations

* slide from Marc’Aurelio Renzato
Pooling Layer

- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Pooling Layer

- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently

*slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford*
Pooling Layer

- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently

How many parameters? None!

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Max Pooling

activation map

max pool with 2 x 2 filter and stride of 2

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
### Average Pooling

**activation map**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

avg pool with 2 x 2 filter and stride of 2

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.25</td>
<td>5.25</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Pooling Layer **Receptive Field**

If convolutional filters have size $K \times K$ and stride 1, and pooling layer has pools of size $P \times P$, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: $(P+K-1) \times (P+K-1)$

* slide from Marc’Aurelio Renzato
Pooling Layer **Receptive Field**

If convolutional filters have size $K \times K$ and stride 1, and pooling layer has pools of size $P \times P$, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: $(P+K-1) \times (P+K-1)$

* slide from Marc’Aurelio Renzato
Convolutional Neural Networks

VGG-16 Network
Local **Contrast Normalization** Layer

ensures response is the same in both case (details omitted, no longer popular)
Improving Single Model

Regularization

- L2, L1
- Dropout / Inverted Dropout
- Data augmentation
Improving Single Model

Regularization

- L2, L1
- Dropout / Inverted Dropout
- Data augmentation

**L2 Regularization:** Learn a more (dense) distributed representation

\[ R(W) = \| W \|_2 = \sum_i \sum_j W_{i,j}^2 \]

**L1 Regularization:** Learn a sparse representation (few non-zero weight elements)

\[ R(W) = \| W \|_1 = \sum_i \sum_j |W_{i,j}| \]
Improving **Single Model**

**Regularization**
- L2, L1
- Dropout / Inverted Dropout
- Data augmentation

**L2 Regularization:** Learn a more (dense) distributed representation

\[ R(W) = \|W\|_2 = \sum_i \sum_j W_{i,j}^2 \]

**L1 Regularization:** Learn a sparse representation (few non-zero weight elements)

\[ R(W) = \|W\|_1 = \sum_i \sum_j |W_{i,j}| \]
Regularization: Data Augmentation

Load image and label

CNN

Compute Loss

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
**Regularization: Data Augmentation**

Load image and label

Transform image

CNN

Compute Loss

Load image and label

**Attributes**

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**
**Regularization:** Data Augmentation

<table>
<thead>
<tr>
<th>Horizontal flips</th>
<th>Random crops &amp; scales</th>
<th>Color Jitter</th>
</tr>
</thead>
</table>

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, [cs231n Stanford](https://cs231n.stanford.edu/)
Regularization: Data Augmentation

Horizontal flips
Random crops & scales
Color Jitter

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
**Regularization:** Data Augmentation

**Horizontal flips**

**Random crops & scales**

**Color Jitter**

**Training:** sample random crops and scales

e.g., ResNet:

1. Pick random $L$ in range $[256, 480]$
2. Resize training image, short size $= L$
3. Sample random 224x224 patch

**Testing:** average a fix set of crops

e.g., ResNet:

1. Resize image to 5 scales (224, 256, 384, 480, 640)
2. For each image use 10 224x224 crops: 4 corners + center, + flips

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Regularization: Data Augmentation

- Horizontal flips
- Random crops & scales
- Color Jitter

Random perturbations in contrast and brightness

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Regularization: Stochastic Depth

Effectively “dropout” but for layers

Stochastically with some probability **turn off** some layer (for each batch)

Effectively trains a collection of neural networks

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford

[ Huang et al., ECCV 2016 ]
Transfer Learning with CNNs

Common “Wisdom”: You need a lot of data to train a CNN

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Transfer Learning with CNNs

Common “Wisdom”: You need a lot of data to train a CNN

Solution: Transfer learning — taking a model trained on the task that has lots of data and adopting it to the task that may not

* adapted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Transfer Learning with CNNs

Common “Wisdom”: You need a lot of data to train a CNN

Solution: Transfer learning — taking a model trained on the task that has lots of data and adopting it to the task that may not

This strategy is PERVASIVE.

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Transfer Learning with CNNs

Train on ImageNet

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Transfer Learning with CNNs

Train on ImageNet

Why on ImageNet?

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Transfer Learning with CNNs

Train on ImageNet

Why on ImageNet?
- Convenience, lots of data
- We know how to train these well

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Transfer Learning with CNNs

Train on ImageNet

Why on ImageNet?
- Convenience, lots of data
- We know how to train these well

However, for some tasks we would need to start with something else (e.g., videos for optical flow)

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Transfer Learning with CNNs

Train on ImageNet

Small dataset with C classes

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Transfer Learning with CNNs

Train on ImageNet

- FC-1000
- FC-4096
- Conv-512
- MaxPool
- Conv-512
- MaxPool
- Conv-256
- MaxPool
- Conv-128
- MaxPool
- Conv-64
- Image

Small dataset with C classes

- FC-1000
- FC-4096
- Conv-512
- MaxPool
- Conv-512
- MaxPool
- Conv-256
- MaxPool
- Conv-128
- MaxPool
- Conv-64
- Image

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford

[ Yosinski et al., NIPS 2014 ]
[ Donahue et al., ICML 2014 ]
[ Razavian et al., CVPR Workshop 2014 ]
Transfer Learning with CNNs

Train on ImageNet

Small dataset with C classes

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Transfer Learning with CNNs

Train on ImageNet

Small dataset with C classes

Re-initialize and train

Freeze these layers

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Transfer Learning with CNNs

Train on ImageNet

Small dataset with C classes

- Freeze these layers
- Re-initialize and train

Lower levels of the CNN are at task independent anyways

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford

[ Yosinski et al., NIPS 2014 ]
[ Donahue et al., ICML 2014 ]
[ Razavian et al., CVPR Workshop 2014 ]
Transfer Learning with CNNs

- **Train on ImageNet**
  - FC-1000
  - FC-4096
  - FC-4096
  - MaxPool
  - Conv-512
  - Conv-512
  - MaxPool
  - Conv-512
  - Conv-512
  - MaxPool
  - Conv-256
  - Conv-256
  - MaxPool
  - Conv-128
  - Conv-128
  - MaxPool
  - Conv-64
  - Conv-64
  - Image

- **Small dataset with C classes**
  - FC-1000
  - FC-4096
  - FC-4096
  - MaxPool
  - Conv-512
  - Conv-512
  - MaxPool
  - Conv-256
  - Conv-256
  - MaxPool
  - Conv-128
  - Conv-128
  - MaxPool
  - Conv-64
  - Conv-64
  - Image

- **Larger dataset**
  - Freeze these layers
  - Re-initialize and train

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford

[ Yosinski et al., NIPS 2014 ]
[ Donahue et al., ICML 2014 ]
[ Razavian et al., CVPR Workshop 2014 ]
Transfer Learning with CNNs

Train on ImageNet

Small dataset with C classes

- FC-1000
- FC-4096
- FC-4096
- MaxPool
- Conv-512
- Conv-512
- MaxPool
- Conv-512
- Conv-512
- MaxPool
- Conv-256
- Conv-256
- MaxPool
- Conv-128
- Conv-128
- MaxPool
- Conv-64
- Conv-64
- Image

Re-initialize and train

- FC-1000
- FC-4096
- FC-4096
- MaxPool
- Conv-512
- Conv-512
- MaxPool
- Conv-256
- Conv-256
- MaxPool
- Conv-128
- Conv-128
- MaxPool
- Conv-64
- Conv-64
- Image

Freeze these layers

Larger dataset

- FC-1000
- FC-4096
- FC-4096
- MaxPool
- Conv-512
- Conv-512
- MaxPool
- Conv-256
- Conv-256
- MaxPool
- Conv-128
- Conv-128
- MaxPool
- Conv-64
- Conv-64
- Image

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford

[ Yosinski et al., NIPS 2014 ]
[ Donahue et al., ICML 2014 ]
[ Razavian et al., CVPR Workshop 2014 ]
Transfer Learning with CNNs

Train on ImageNet

Small dataset with C classes

Larger dataset

Re-initialize and train

Freeze these layers

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford

[ Dosnhaue et al., ICML 2014]
[ Razavian et al., CVPR Workshop 2014]
[ Yosinski et al., NIPS 2014]
Transfer Learning with CNNs

Train on ImageNet

Small dataset with C classes

Larger dataset

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Transfer Learning with CNNs

5: Transfer + fine-tuning improves generalization
3: Fine-tuning recovers co-adapted interactions
2: Performance drops due to fragile co-adaptation
4: Performance drops due to representation specificity

[ Yosinski et al., NIPS 2014 ]
Model **Ensemble**

**Training:** Train multiple independent models

**Test:** Average their results

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Model **Ensemble**

**Training:** Train multiple independent models

**Test:** Average their results

~ 2% improved performance in practice

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, [cs231n Stanford](https://cs231n.stanford.edu/)*
Model Ensemble

Training: Train multiple independent models

Test: Average their results

~ 2% improved performance in practice

Alternative: Multiple snapshots of the single model during training!

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Model **Ensemble**

**Training:** Train multiple independent models

**Test:** Average their results

~ 2% improved performance in practice

**Alternative:** Multiple snapshots of the single model during training!

**Improvement:** Instead of using the actual parameter vector, keep a moving average of the parameter vector and use that at test time (Polyak averaging)

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
CPU vs. GPU (Why do we need Azure?)

Data from https://github.com/jcjohnson/cnn-benchmarks

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Frameworks: Super quick overview

1. Easily **build computational graphs**

2. Easily **compute gradients** in computational graphs

3. **Run it all efficiently** on a GPU (weap cuDNN, cuBLAS, etc.)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
<table>
<thead>
<tr>
<th>Framework</th>
<th>Core DNN Frameworks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caffe</td>
<td>UC Berkeley</td>
</tr>
<tr>
<td>Caffe 2</td>
<td>Facebook</td>
</tr>
<tr>
<td>Puddle</td>
<td>Baidu</td>
</tr>
<tr>
<td>Torch</td>
<td>NYU/Facebook</td>
</tr>
<tr>
<td>PyTorch</td>
<td>Facebook</td>
</tr>
<tr>
<td>CNTK</td>
<td>Microsoft</td>
</tr>
<tr>
<td>Theano</td>
<td>U Montreal</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>Google</td>
</tr>
<tr>
<td>MXNet</td>
<td>Amazon</td>
</tr>
</tbody>
</table>

* *slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford*
<table>
<thead>
<tr>
<th>Core DNN Frameworks</th>
<th>Wrapper Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Caffe</strong> (UC Berkeley)</td>
<td><strong>Keras</strong></td>
</tr>
<tr>
<td><strong>Caffe 2</strong> (Facebook)</td>
<td><strong>TFLearn</strong></td>
</tr>
<tr>
<td><strong>Puddle</strong> (Baidu)</td>
<td><strong>TensorLayer</strong></td>
</tr>
<tr>
<td><strong>Torch</strong> (NYU/Facebook)</td>
<td><strong>tf.layers</strong></td>
</tr>
<tr>
<td><strong>PyTorch</strong> (Facebook)</td>
<td><strong>TF-Slim</strong></td>
</tr>
<tr>
<td><strong>Theano</strong> (U Montreal)</td>
<td><strong>tf.contrib.learn</strong></td>
</tr>
<tr>
<td><strong>TensorFlow</strong> (Google)</td>
<td><strong>Pretty Tensor</strong></td>
</tr>
<tr>
<td><strong>MXNet</strong> (Amazon)</td>
<td></td>
</tr>
</tbody>
</table>

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
Frameworks: PyTorch vs. TensorFlow

Dynamic vs. Static computational graphs
Frameworks: PyTorch vs. TensorFlow

Dynamic vs. Static computational graphs

With static graphs, framework can optimize the graph for you before it runs!

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
**Frameworks:** PyTorch vs. TensorFlow

**Dynamic** vs. Static computational graphs

Graph building and execution is intertwined. Graph can be different for every sample.

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford
**PyTorch:** Three levels of abstraction

**Tensor:** Imperative ndarray, but runs on GPU

**Variable:** Node in a computational graph; stores data and gradients

**Module:** A neural network layer; may store state or learnable weights

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, cs231n Stanford