

# CS 540 Introduction to Artificial Intelligence **Deep Learning III**

University of Wisconsin-Madison Spring 2024

#### Announcements

- Homework:
  - HW7 is due on Thursday Apr. 4 at 11 AM
- Exam results published

• Class roadmap:

Thursday, Mar. 21	Machine Learning: Deep Learning III
Tuesday, Apr. 2	Deep Learning and Neural Network's Summary
Thursday, Apr. 4	Search I: Un-Informed search

# Outline

- CNNs with more layers: ResNets
  - Layer problems, residual connections, identity maps
- Data Augmentation & Regularization
  - Expanding the dataset, avoiding overfitting
- More Signal From our Data
  - Graph-structured data, graph neural networks

### Last Time: CNNs

We talked about CNN components & architectures

- Components: convolutional layers, pooling layers (recall kernels, channels, strides, padding)
- Architectures: LeNet, AlexNet, VGG

• Trend: bigger, deeper.



### **Evolution of CNNs**

#### ImageNet competition (error rate)



Credit: Stanford CS 231n

### Simple Idea: Add More Layers

VGG: 19 layers. ResNet: 152 layers. **Add more layers**... sufficient?

- No! Some problems:
  - i) Vanishing gradients: more layers  $\rightarrow$  more likely
  - ii) Instability: deeper models are harder to optimize

**Reflected in training error:** 



He et al: "Deep Residual Learning for Image Recognition"

**Depth Issues & Learning Identity** 

Why would more layers result in worse performance?



Idea: if layers can learn identity, can't get worse.

### **Residual Connections**

Idea: Identity might be hard to learn, but zero is easy!

- Make all the weights tiny, produces zero for output
- Can easily transform learning identity to learning zero:



Left: Conventional layers block

Right: Residual layer block

To learn identity f(x) = x, layers now need to learn  $f(x) = 0 \rightarrow$  easier

# Full ResNet Architecture

[He et al. 2015]

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride of 2 (/2 in each dimension)





Idea: Residual (skip) connections help make learning easier

- Example architecture:
- Note: residual connections
  - Every two layers for ResNet34
- Vastly better performance
  - No additional parameters!
  - Records on many benchmarks



He et al: "Deep Residual Learning for Image Recognition"

#### Various depth

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer						
conv1	112×112	7×7, 64, stride 2										
		$3 \times 3$ max pool, stride 2										
conv2_x	56×56	$\left[\begin{array}{c} 3\times3,64\\ 3\times3,64\end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3, 64\\ 3\times3, 64\end{array}\right]\times3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$						
conv3_x	28×28	$\left[\begin{array}{c} 3\times3,128\\3\times3,128\end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3,128\\3\times3,128\end{array}\right]\times4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$						
conv4_x	14×14	$\left[\begin{array}{c} 3\times3,256\\ 3\times3,256\end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3,256\\ 3\times3,256\end{array}\right]\times6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$						
conv5_x	7×7	$\left[\begin{array}{c} 3\times3,512\\ 3\times3,512\end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3,512\\ 3\times3,512\end{array}\right]\times3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$						
	1×1	average pool, 1000-d fc, softmax										
FLOPs		$1.8 \times 10^{9}$	$3.6 \times 10^{9}$	$3.8 \times 10^{9}$	$7.6 \times 10^9$	$11.3 \times 10^{9}$						

#### Various depth

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer						
conv1	112×112	7×7, 64, stride 2										
				3×3 max pool, strid	le 2							
conv2_x	56×56	$\left[\begin{array}{c} 3\times3,64\\ 3\times3,64 \end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3, 64\\ 3\times3, 64 \end{array}\right]\times3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$						
conv3_x	28×28	$\left[\begin{array}{c} 3\times3,128\\3\times3,128\end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3,128\\ 3\times3,128\end{array}\right]\times4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$						
conv4_x	14×14	$\left[\begin{array}{c} 3\times3,256\\ 3\times3,256\end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3,256\\ 3\times3,256\end{array}\right]\times6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$						
conv5_x	7×7	$\left[\begin{array}{c} 3\times3,512\\ 3\times3,512\end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3,512\\ 3\times3,512\end{array}\right]\times3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$						
	1×1	average pool, 1000-d fc, softmax										
FLOPs		1.8×10 <sup>9</sup>	$3.6 \times 10^{9}$	3.8×10 <sup>9</sup>	7.6×10 <sup>9</sup>	$11.3 \times 10^{9}$						

Various depth

				/									
layer name	output size	18-layer 34-layer 50-layer 101-layer 152-layer											
conv1	112×112	7×7, 64, stride 2											
				3×3 max pool, stric	le 2	×							
conv2_x	56×56	$\left[\begin{array}{c} 3\times3,64\\ 3\times3,64 \end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3, 64\\ 3\times3, 64 \end{array}\right]\times3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$							
conv3_x	28×28	$\left[\begin{array}{c} 3\times3,128\\3\times3,128\end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3,128\\3\times3,128\end{array}\right]\times4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$							
conv4_x	14×14	$\begin{bmatrix} 3\times3,256\\3\times3,256\end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,256\\3\times3,256\end{bmatrix}\times6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$							
conv5_x	7×7	$\left[\begin{array}{c} 3\times3,512\\ 3\times3,512\end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3,512\\ 3\times3,512\end{array}\right]\times3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$							
	1×1	average pool, 1000-d fc, softmax											
FL	OPs	$1.8 \times 10^{9}$	$3.6 \times 10^{9}$	$3.8 \times 10^{9}$	$7.6 \times 10^{9}$	$11.3 \times 10^{9}$							

, Repeat x3 times

# of filters

#### Various depth

layer name	output size	18-layer	34-layer	50-layer	152-layer								
conv1	112×112	7×7, 64, stride 2											
		3×3 max pool, stride 2											
conv2_x	56×56	$\left[\begin{array}{c} 3\times3, 64\\ 3\times3, 64 \end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3,64\\ 3\times3,64\end{array}\right]\times3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$							
conv3_x	28×28	$\left[\begin{array}{c} 3\times3,128\\ 3\times3,128\end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3,128\\ 3\times3,128\end{array}\right]\times4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$							
conv4_x	14×14	$\begin{bmatrix} 3\times3,256\\3\times3,256\end{bmatrix}\times2$	$\begin{bmatrix} 3\times3,256\\3\times3,256\end{bmatrix}\times6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$							
conv5_x	7×7	$\left[\begin{array}{c} 3\times3,512\\ 3\times3,512\\ 3\times3,512\end{array}\right]\times2$	$\begin{bmatrix} 3\times3,512\\ 3\times3,512\end{bmatrix}\times3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$							
	1×1	average pool, 1000-d fc, softmax											
FLOPs		1.8×10 <sup>9</sup>	$3.6 \times 10^{9}$	3.8×10 <sup>9</sup>	$7.6 \times 10^9$	11.3×10 <sup>9</sup>							

#### 1 + 2x3 + 2x4 + 2x6 + 2x3 + 1 = 34

#### ResNet Training Curves on ImageNet [He et al., 2015]



# A Bit More on ResNets

Idea: Residual (skip) connections help make learning easier

- Note: Can also analyze from **backpropagation** p.o.v
  - Residual connections add paths to computation graph
- Also uses **batch normalization** 
  - Normalize the features at each layer to have same mean/variance
  - Common deep learning trick
- Highway networks: learn weights for residual connections

Ioffe and Szegedy: "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift"

**Q 1.1**: Which of the following is **not** true?

- A. Adding more layers can improve the performance of a neural network.
- B. Residual connections help deal with vanishing gradients.
- C. CNN architectures use no more than ~20 layers to avoid problems such as vanishing gradients.
- D. It is usually easier to learn a zero mapping than the identity mapping.

**Q 1.1**: Which of the following is **not** true?

- A. Adding more layers can improve the performance of a neural network.
- B. Residual connections help deal with vanishing gradients.
- C. CNN architectures use no more than ~20 layers to avoid problems such as vanishing gradients.
- D. It is usually easier to learn a zero mapping than the identity mapping.

**Q 1.1**: Which of the following is **not** true?

- A. Adding more layers can improve the performance of a neural network. (Yes, as long as we're careful, e.g., ResNets.)
- B. Residual connections help deal with vanishing gradients. (Yes, this is an explicit consideration for residual connections.)
- C. CNN architectures use no more than ~20 layers to avoid problems such as vanishing gradients. (No, much deeper networks.)
- D. It is usually easier to learn a zero mapping than the identity mapping. (Yes: simple way to learn zero is to make weights zero)

### Data Concerns

What if we don't have a lot of data?

- We risk overfitting
- Avoiding overfitting: **regularization** methods
- Data augmentation: a classic way to regularize



### Data Augmentation

Augmentation: transform + add new samples to dataset

- Transformations: based on domain
- Idea: build invariances into the model
  - Ex: if all images have same alignment, model learns to use it
- Keep the label the same!



# Transformations

- Examples of transformations for images
- Crop (and zoom)
- **Color** (change contrast/brightness)
- **Rotations+** (translate, stretch, shear, etc)

Many more possibilities. Combine as well!

Q: how to deal with this at **test time**?

• A: transform, test, average



# **Combining & Automating Transformations**

One way to automate the process:

- Apply every transformation and combinations
- **Downside:** most don't help...

Want a good policy, ie,  $\rightarrow \rightarrow \rightarrow \rightarrow \rightarrow$ 

- Active area of research: search for good policies
  - 1. Ratner et al: "Learning to Compose Domain-Specific Transformations for Data Augmentation"
  - 2. Cubuk et al: "AutoAugment: Learning Augmentation Strategies from Data"



# **Other Domains**

Not just for image data. For example, on text:

- Substitution
  - E.g., "It is a great day" → "It is a wonderful day"
  - Use a thesaurus for particular words
  - Or, use a model. Pre-trained word embeddings, language models
- Back-translation
  - "Given the low budget and production limitations, this movie is very good."
     → "There are few budget items and production limitations to make this film a really good one"

Xie et al: "Unsupervised Data Augmentation for Consistency Training"

## Importance of Augmentation

Data augmentation is critical for top performance!

- You should use it!
- **AlexNet**: used (many papers re-used as well)
  - Random crops, rotations, flips.

Krizhevsky et al: "ImageNet Classification with Deep Convolutional Neural Networks"

-		*	11	1	5	3	Ľ	3	*	I	1	1			4
	1	1	1	1	-	1	4	11	1	1		1	11	W.	11
	No.	11		1	11	-	11				11			F	1
- 10	-			-			11	*	+			1		11	
	1		11		*	11		K.			1	-			
Lake		W.	1				1	11			11			-	-

# **Other Forms of Regularization**

Regularization has many interpretations

- **Goodfellow**: "any modification... to a learning algorithm that is intended to reduce its generalization error but not its training error."
- A way of adding knowledge / side information to model
- Enforcing parsimony/simplicity



# **Other Forms of Regularization**

#### Classic regularizations

Modify loss functions
 Ex: regularized least squares LR

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + x_i^T \theta - y_i)^2 + \lambda \|\theta\|_2^2$$

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f_{\theta}(x_i), y_i) + \lambda R(f_{\theta})$$

$$\|_{2}^{2} \qquad \qquad \text{Standard} \\ \text{loss} \qquad \qquad \text{Regularization} \\ \text{parameter} \\ \end{tabular}$$

- 1. Modify architecture/training/data
  - a) Dropout, batch normalization, augmentation

Q 2.1: If we apply data augmentation blindly, we might

(i) Change the label of the data point

(ii) Produce a useless training point

- A. (i) but not (ii)
- B. (ii) but not (i)
- C. Neither
- D. Both

Q 2.1: If we apply data augmentation blindly, we might

(i) Change the label of the data point

(ii) Produce a useless training point

- A. (i) but not (ii)
- B. (ii) but not (i)
- C. Neither
- D. Both

**Q 2.1**: If we apply data augmentation blindly, we might

(i) Change the label of the data point

(ii) Produce a useless training point

- A. (i) but not (ii) (Can do (ii): imagine turning up the contrast till the image is completely black and is unusable).
- B. (ii) but not (i) (Can change label: rotate a 6 into a 9).
- C. Neither (Can do either).
- D. Both

**Q 2.2**: What are some consequences of data augmentation?

- (i) We have to store a much bigger dataset in memory
- (ii) For a fixed batch size, there will be more batches per epoch
- A. (i) but not (ii)
- B. (ii) but not (i)
- C. Neither
- D. Both

**Q 2.2**: What are some consequences of data augmentation?

- (i) We have to store a much bigger dataset in memory
- (ii) For a fixed batch size, there will be more batches per epoch
- A. (i) but not (ii)
- B. (ii) but not (i)
- C. Neither
- D. Both

# **Relationships in Data**

- So far, all of our data consists of points
- Assume all are independent, "unrelated" in a sense (x<sub>1</sub>, y<sub>1</sub>), (x<sub>2</sub>, y<sub>2</sub>), ..., (x<sub>n</sub>, y<sub>n</sub>)
- Pretty common to have relationships between points
  - Social networks: individuals related by friendship
  - Biology/chemistry: bonds between compounds, molecules
  - Citation networks: Scientific papers cite each other



# Signal from Relationships

Suppose we are classifying scientific papers

- Features: title, abstract, authors. Labels: math/science/eng.
- Could build a reasonable classifier with the above data
- More signal from relationships
  - Cite each other, more likely from the same field
  - Note: citations are not features; they're links
  - Need a new type of network to handle



### **Graph Neural Networks**

- **Have:**  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n), G = (V, E)$
- How should our new architecture look?
- Still want layers
  - linear transformation + non-linearity

Hidden Layer Representation

- Now want to integrate neighbors
- Bottom: graph convolutional network

Kipf and Welling: "Semi-Supervised Classification with Graph Convolutional Networks"

**Graph Mixing** 

Parameters

**Non-Linearity** 

 $H^{(\ell+1)} = \sigma(H^{(\ell)}W^{(\ell)})$ 

 $H^{(\ell+1)} = \sigma(A_G H^{(\ell)} W^{(\ell)})$ 

# **Graph Convolutional Networks**

Let's examine the GCN architecture in more detail

- Difference: "graph mixing" component
- At each layer, get representation at each node
- Combine node's representation with neighboring nodes
- "Aggregate" and "Update" rules



# Graph Convolutional Networks

Note the resemblance to CNNs:

- Pixels: arranged as a very regular graph
- Want: more general configurations (less regular)





Wu et al, A Comprehensive Survey on Graph Neural Networks





Zhou et al, Graph Neural Networks: A Review of Methods and Applications

# Summary

- Intro to deeper networks (resnets)
  - Dealing with problems by adding skip connections
- Intro to regularization
  - Data augmentation + other regularizers
- Basic graph neural networks



#### **Acknowledgements**: Inspired by materials by Fei-Fei Li, Ranjay Krishna, Danfei Xu (Stanford CS231n)