CS 540 Introduction to Artificial Intelligence
Deep Learning III

University of Wisconsin–Madison
Fall 2025, Section 3
October 27, 2025

CS540

- Home
- About
- Announcements
- Schedule
- Teaching Team
- Office Hours
- Important Links
- Piazza
- Canvas

Academic Integrity

You are encouraged to discuss with your peers, the TA or the instructors ideas, approaches and techniques broadly. However, all examinations, programming assignments, and written homeworks must be written up individually. For example, code for programming assignments must not be developed in groups, nor should code be shared. Make sure you work through all problems yourself, and that your final write-up is your own. If you feel your peer discussions are too deep for comfort, declare it in the homework solution: “I discussed with X,Y,Z the following specific ideas: A, B, C; therefore our solutions may have similarities on D, E, F…”.

You may use books or legit online resources to help solve homework problems, but you must always credit all such sources in your writeup and you must never copy material verbatim.

Use of AI Tools: All submitted work must be your own. You may use artificial intelligence tools (like ChatGPT, Claude, or Cursor) in this class only as you might consult a peer for help, as outlined in the guidelines above. You may consult an AI tool to brainstorm approaches, clarify instructions, review concepts. You may ask for help with language or package syntax. You may use an AI tool for debugging help as long as you remain the primary problem-solver. You may **not** use AI to generate and/or copy solutions, code, or written work, even partially. When in doubt, ask: “Would it be okay if a friend did this for me?” If the answer is no, it’s not okay to have an AI do it either.

We are aware that certain websites host previous years’ CS540 homework assignments and solutions against the wish of instructors. Do not be tempted to use them: the solutions may contain “poisonous berries” previous instructors planted intentionally to

All submitted work must be your own.

Outline

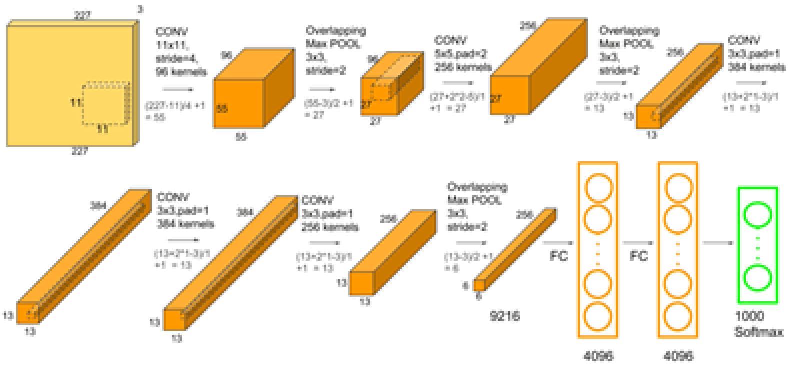
- ResNets
  - Layer problems, residual connections, identity maps
  - Build CNNs with more layers!
- Data Augmentation & Regularization
  - Expanding the dataset, avoiding overfitting
- More Signal From our Data
  - Graph-structured data, graph neural networks

Last Time: CNNs

We talked about CNN components & architectures

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

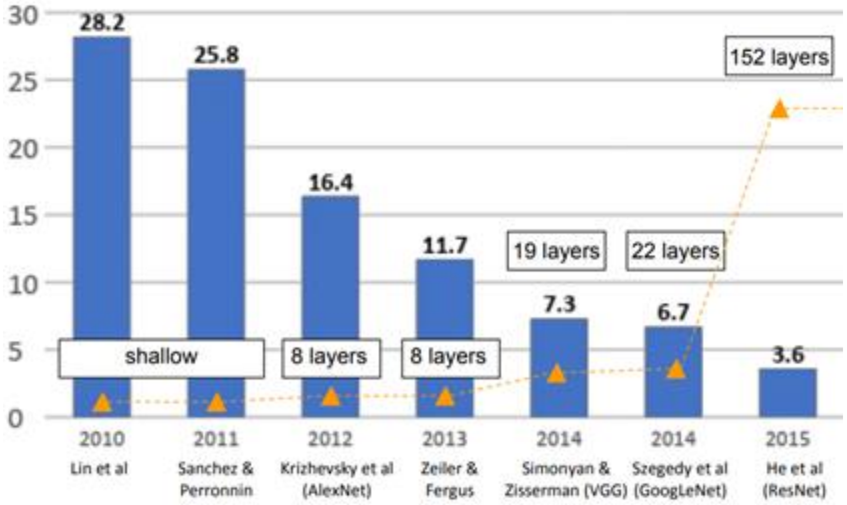
- Trend: bigger, deeper.



Credit: Mathworks

Evolution of CNNs

ImageNet competition (error rate)



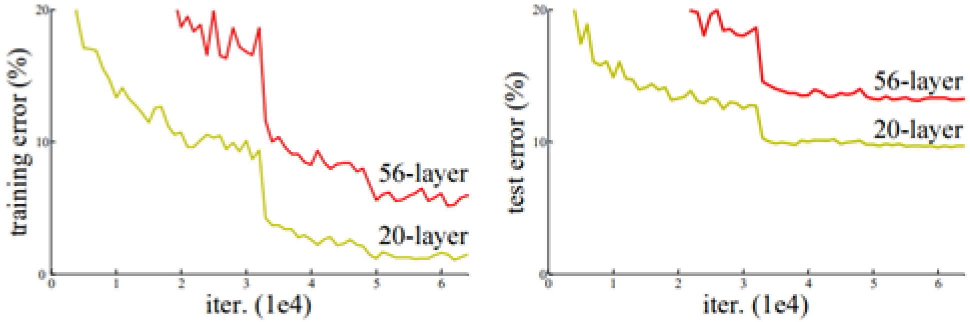
Credit: Stanford CS 231n

Simple Idea: Add More Layers

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

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

Reflected in training error:

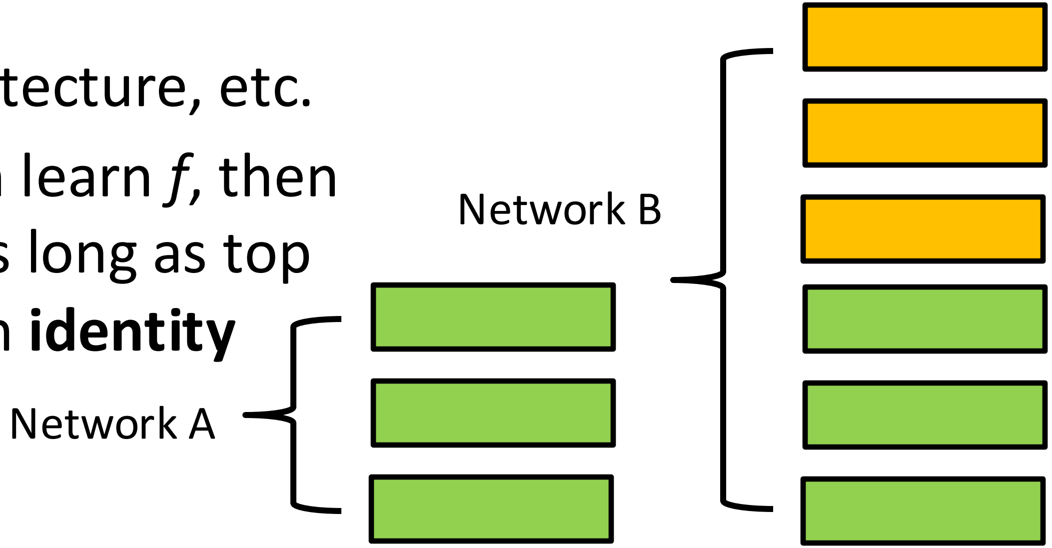


He et al: "Deep Residual Learning for Image Recognition"

Depth Issues & Learning Identity

Why would more layers result in **worse** performance?

- Same architecture, etc.
- If the A can learn *f*, then so can B, as long as top layers learn **identity**



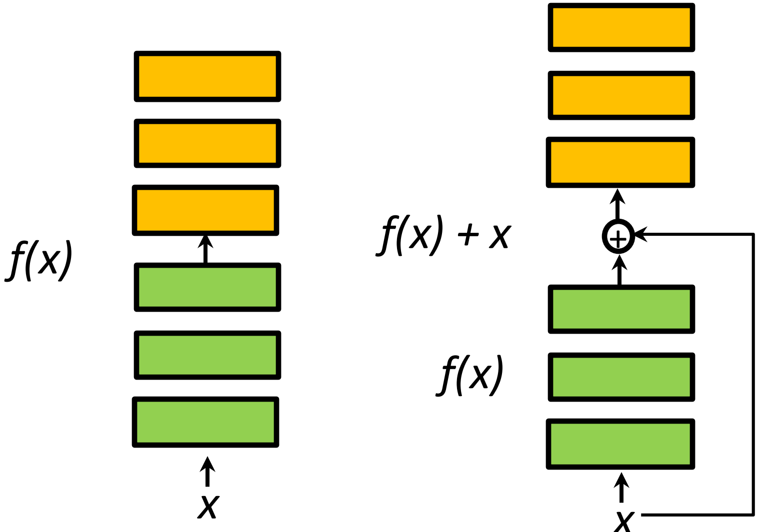
Q: can we learn identity here?

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

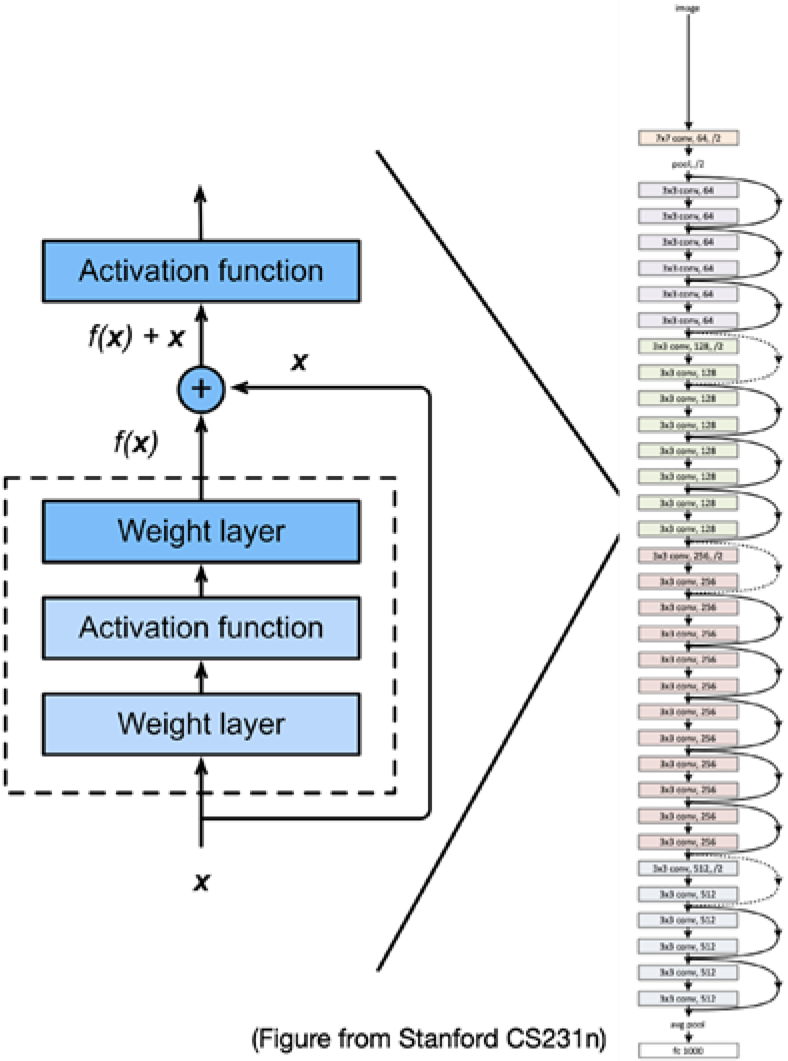
Right: **Residual** layer block

To learn identity $f(x) = x$, layers now need to learn $f(x) = 0$ ➔ 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)

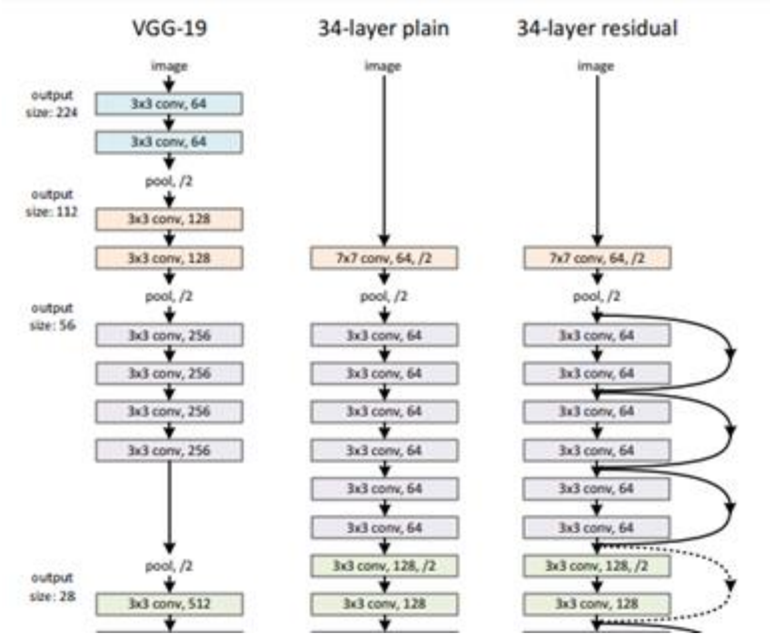


(Figure from Stanford CS231n)

ResNet Architecture

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”

ResNet Architecture

Various depth

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet Architecture

Various depth

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\ 3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\ 3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\ 3\times3, 64\\ 1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\ 3\times3, 64\\ 1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\ 3\times3, 64\\ 1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\ 3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\ 3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\ 3\times3, 128\\ 1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\ 3\times3, 128\\ 1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\ 3\times3, 128\\ 1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\ 3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\ 3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\ 3\times3, 256\\ 1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\ 3\times3, 256\\ 1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\ 3\times3, 256\\ 1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\ 3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\ 3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\ 3\times3, 512\\ 1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\ 3\times3, 512\\ 1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\ 3\times3, 512\\ 1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet Architecture

Various depth

Repeat x3 times

# of filters

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet Architecture

Various depth

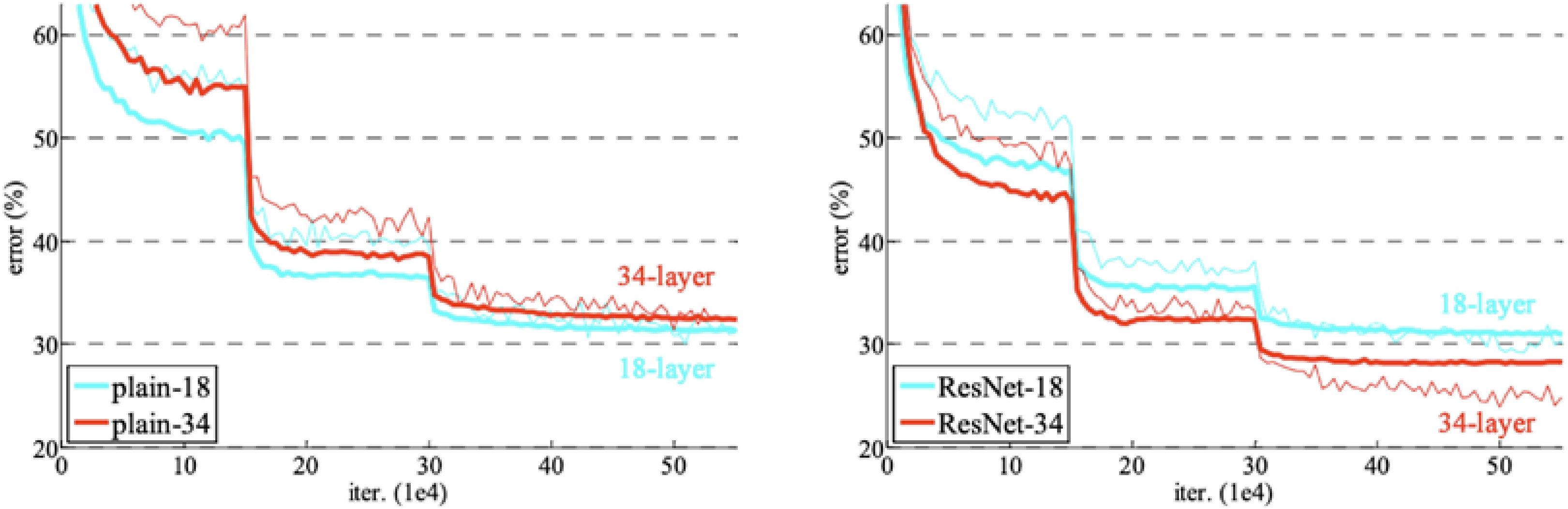
1+ 2x3 + 2x4 + 2x6 + 2x3 +1 = 34

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet Training Curves on ImageNet

[He et al., 2015]



A Bit More on ResNets

Idea: Residual (skip) connections help make learning easier

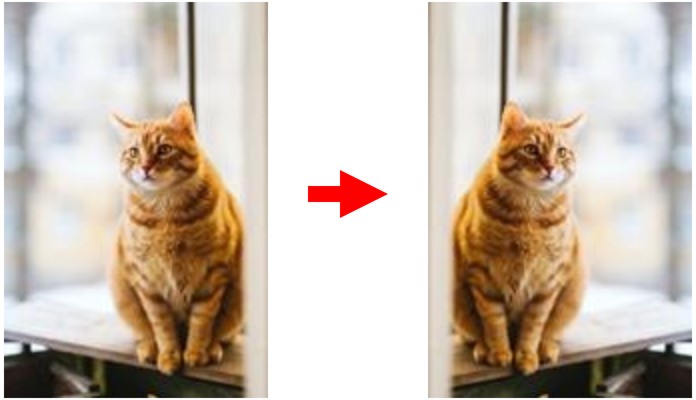
- Note: Can also analyze from **backpropagation** p.o.v
  - Residual connections add paths to computation graph
- Also uses **batch normalization**
  - Normalize the features at each layer to have same mean/variance
  - Common deep learning trick
- Highway networks: learn weights for residual connections

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

Data Concerns

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

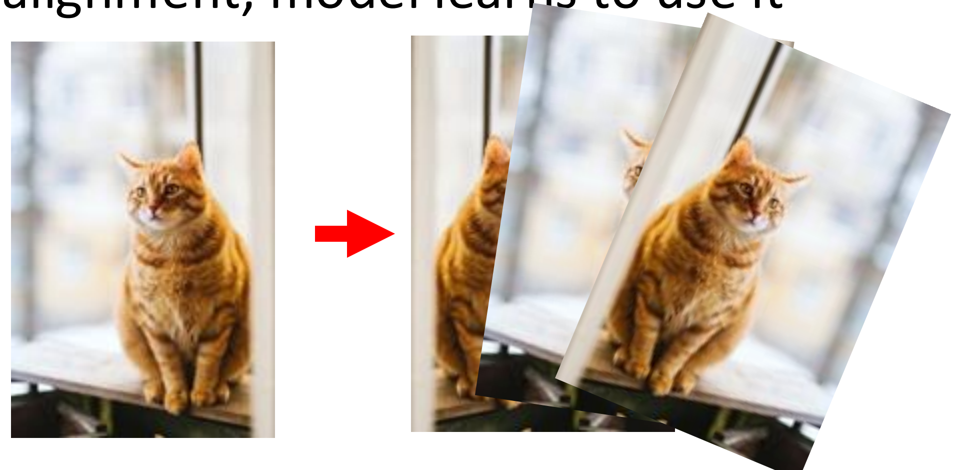
- We risk overfitting
- Avoiding overfitting: **regularization** methods
- Data augmentation: a classic way to regularize



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, → → → → →

- 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”

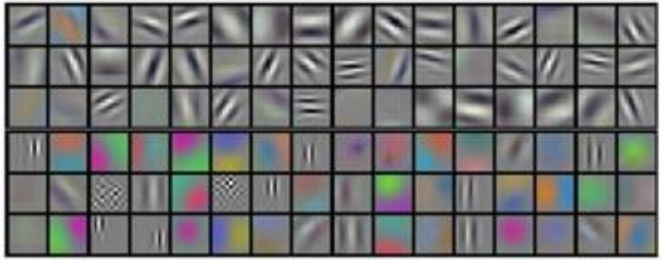
Xie **et al**: “Unsupervised Data Augmentation for Consistency Training”

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.

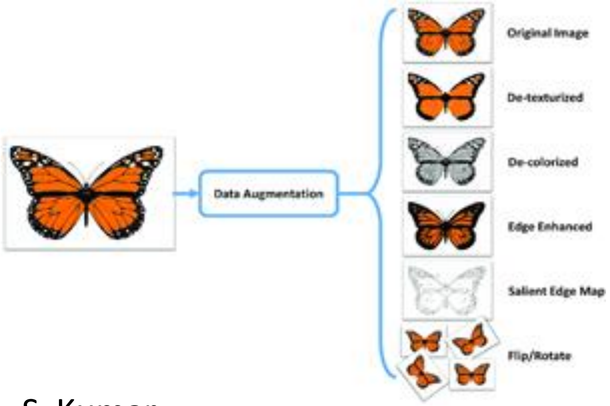
Krizhevsky et al: “ImageNet Classification with Deep Convolutional Neural Networks”



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



S. Kumar

Other Forms of Regularization

Classic regularizations

1. Modify loss functions

Ex: regularized least squares LR

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

$$\min_\theta \frac{1}{n}\sum_{i=1}^{n}\underbrace{\ell(f_\theta(x_i), y_i)}_{\text{Standard loss}} + \underbrace{\lambda}_{\text{Regularization parameter}}\overbrace{R(f_\theta)}^{\text{Regularizer}}$$

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

Break & Quiz

Break & Quiz

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.

Break & Quiz

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.

Break & Quiz

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)

Break & Quiz

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

Break & Quiz

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**

Break & Quiz

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**

Break & Quiz

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

Break & Quiz

Q 2.2: What are some consequences of data augmentation?

(i) We have to store a much bigger dataset in memory

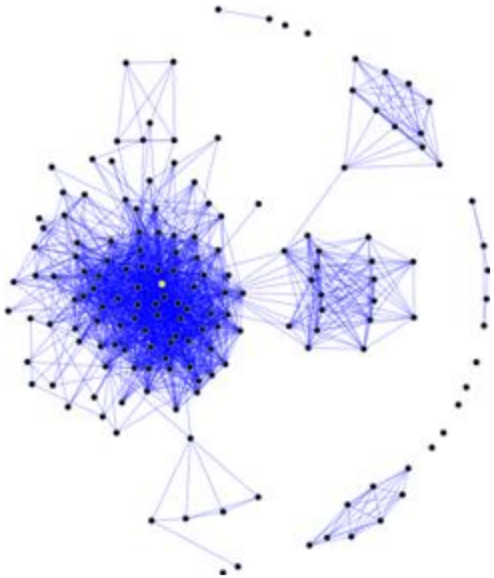
(ii) For a fixed batch size, there will be more batches per epoch

- 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 $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$
- 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

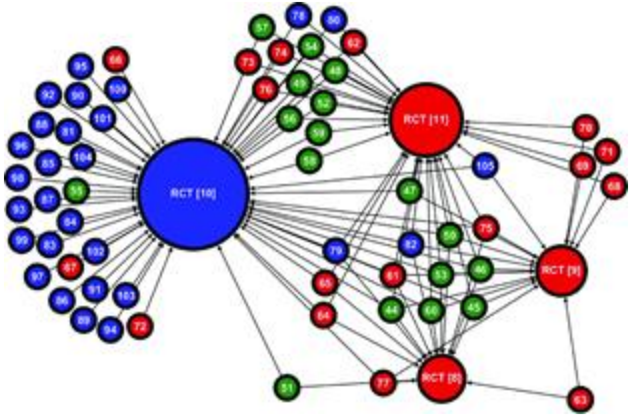


Wiki

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



Leng

Graph Neural Networks

Have: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n), G = (V, E)$

How should our new architecture look?

- Still want layers
  - linear transformation + non-linearity

Parameters

Non-Linearity

Hidden Layer Representation

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

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

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

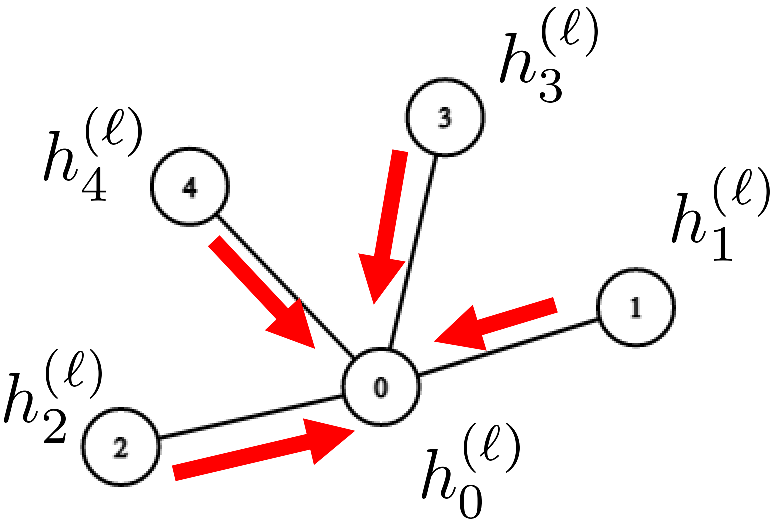
Kipf and Welling: "Semi-Supervised Classification with Graph Convolutional Networks"

Graph Mixing

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

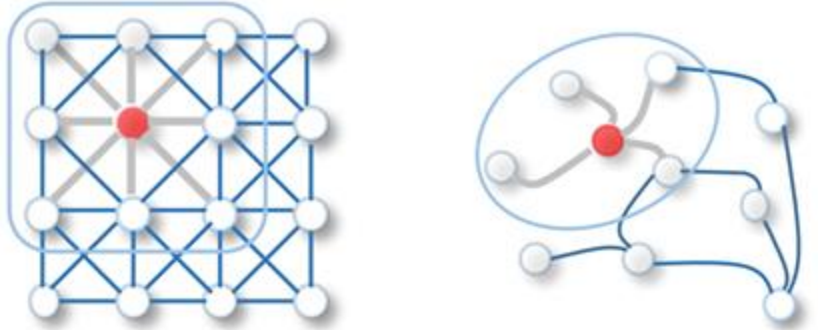


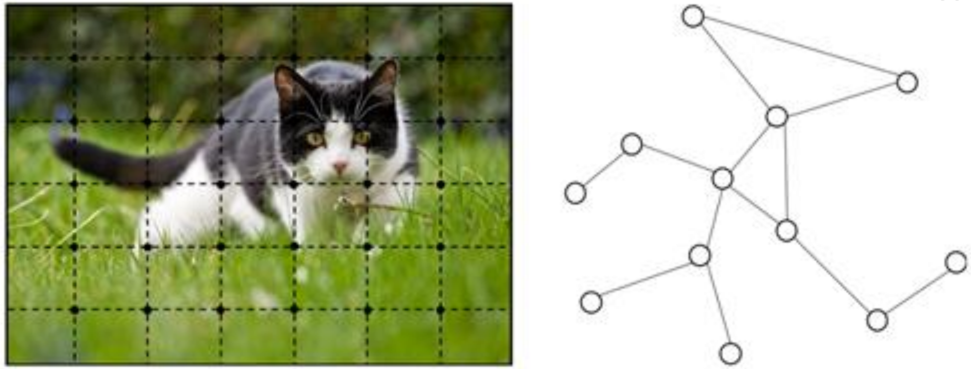
Graph Convolutional Networks

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
- More on regularization
  - Data augmentation + other regularizers
- Basic graph neural networks

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