Topics:
• Convolutional Neural Networks Architectures
Recap: Image features are spatially localized!

- Relevant features repeated across the image
  - Edges
  - Color
  - Motifs (corners, etc.)

- No reason to believe one feature tends to appear in a fixed location. Need to search in entire image.

Can we enforce a structure in the design of a neural network layer to reflect this?
Recap: Convolution

1-D Convolution is defined as the integral of the product of two functions after one is reflected about the y-axis and shifted.

Cross-correlation is convolution without the y-axis reflection.

**Intuitively**: given function $f$ and filter $g$. How similar is $g(-x)$ with the part of $f(x)$ that it’s operating on.

For ConvNets, we don’t flip filters, so we are really using Cross-Correlation Nets!

From https://en.wikipedia.org/wiki/Convolution
Convolution Layer

32x32x3 image
5x5x3 filter $w$

$w^T x + b$
Recap: Convolution Layer
Recap: Convolution Layer

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation map
For example, if we had 6 5x5 filters, we’ll get 6 separate activation maps:
In practice: Common to zero pad the border
e.g. input 7x7
3x3 filter, applied with \textit{stride 1}
\textit{pad with 1 pixel} border => what is the output?
7x7 output!

\begin{align*}
N &= \text{input dimension} \\
P &= \text{padding size} \\
F &= \text{filter size} \\
\text{Output size} &= \frac{N - F + 2P}{\text{stride}} + 1 \\
&= \frac{7 - 3 + 2 \times 1}{1} + 1 = 7
\end{align*}
Remember back to…
With padding, we can keep the same spatial feature dimension throughout the convolution layers.
Pooling layer (down-sampling)
- makes the representations spatially smaller
- saves computation (GPU mem & speed), allows go deeper
- operates over each activation map independently:

![Diagram](image.png)
A canonical (shallow) convolutional neural net
The ImageNet dataset contains 14,197,122 annotated images according to the WordNet hierarchy. ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a benchmark for image classification and object detection based on the dataset.
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

<table>
<thead>
<tr>
<th>Year</th>
<th>Winner</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>Lin et al</td>
<td>(AlexNet)</td>
</tr>
<tr>
<td>2011</td>
<td>Sanchez &amp; Perronnin</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Krizhevsky et al</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Zeiler &amp; Fergus</td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>Simonyan &amp; Zisserman</td>
<td>(VGG)</td>
</tr>
<tr>
<td>2014</td>
<td>Szegedy et al</td>
<td>(GoogLeNet)</td>
</tr>
<tr>
<td>2015</td>
<td>He et al</td>
<td>(ResNet)</td>
</tr>
<tr>
<td>2016</td>
<td>Shao et al</td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td>Hu et al</td>
<td>(SENet)</td>
</tr>
<tr>
<td>Human</td>
<td>Russakovsky et al</td>
<td></td>
</tr>
</tbody>
</table>

“Pre-Deep Learning”
ConvNets: Where are we today?

https://paperswithcode.com/sota/image-classification-on-imagenet
CNN Architectures

Case Studies
- AlexNet
- VGG
- GoogLeNet
- ResNet

Also....
- SENet
- Wide ResNet
- ResNeXt
- DenseNet
- MobileNets
- NASNet
- EfficientNet
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

16.4

First CNN-based winner

28.2

25.8

2010

2011

2012

2013

2014

2014

2015

2016

2017

Human

Lin et al

Sanchez & Perronnin

Krizhevsky et al (AlexNet)

Zeiler & Fergus

Simonyan & Zisserman (VGG)

Szegedy et al (GoogLeNet)

He et al (ResNet)

Shao et al

Hu et al (SENet)

Russakovsky et al

shallow

8 layers

8 layers

19 layers

22 layers

152 layers

152 layers

152 layers
Case Study: AlexNet

[Krizhevsky et al. 2012]

Architecture:
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4

=>

Q: what is the output volume size? Hint: (227-11)/4+1 = 55

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume [55x55x96]

\[ W' = \frac{(W - F + 2P)}{S} + 1 \]
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images

**First layer** (CONV1): 96 11x11 filters applied at stride 4

=>

Output volume **[55x55x96]**

Q: What is the total number of parameters in this layer?
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images

**First layer** (CONV1): 96 11x11 filters applied at stride 4 =>

Output volume **[55x55x96]**

Parameters: \((11 \times 11 \times 3 + 1) \times 96 = 35K\)

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2

Q: what is the output volume size? Hint: (55-3)/2+1 = 27

\[ W' = \frac{(W - F + 2P)}{S} + 1 \]
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images
After CONV1: 55x55x96

**Second layer** (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96

Q: what is the number of parameters in this layer?

\[
W' = (W - F + 2P) / S + 1
\]

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96
Parameters: 0!
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images
After CONV1: 55x55x96
After POOL1: 27x27x96
...

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Next two lectures!

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

- INPUT
- [227x227x3] INPUT
- [55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
- [27x27x96] MAX POOL1: 3x3 filters at stride 2
- [27x27x96] NORM1: Normalization layer
- [27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
- [13x13x256] MAX POOL2: 3x3 filters at stride 2
- [13x13x256] NORM2: Normalization layer
- [13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
- [13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
- [13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
- [6x6x256] MAX POOL3: 3x3 filters at stride 2
- [4096] FC6: 4096 neurons
- [4096] FC7: 4096 neurons
- [1000] FC8: 1000 neurons (class scores)

Historical note: Trained on GTX 580 GPU with only 3 GB of memory. Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

- [227x227x3] INPUT
- [55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
- [27x27x96] MAX POOL1: 3x3 filters at stride 2
- [27x27x96] NORM1: Normalization layer
- [27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
- [13x13x256] MAX POOL2: 3x3 filters at stride 2
- [13x13x256] NORM2: Normalization layer
- [13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
- [13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
- [13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
- [6x6x256] MAX POOL3: 3x3 filters at stride 2
- [4096] FC6: 4096 neurons
- [4096] FC7: 4096 neurons
- [1000] FC8: 1000 neurons (class scores)

CONV3, FC6, FC7, FC8: Connections with all feature maps in preceding layer, communication across GPUs

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

- **INPUT**
- **CONV1**: 96 11x11 filters at stride 4, pad 0
- **MAX POOL1**: 3x3 filters at stride 2
- **NORM1**: Normalization layer
- **CONV2**: 256 5x5 filters at stride 1, pad 2
- **MAX POOL2**: 3x3 filters at stride 2
- **NORM2**: Normalization layer
- **CONV3**: 384 3x3 filters at stride 1, pad 1
- **CONV4**: 384 3x3 filters at stride 1, pad 1
- **CONV5**: 256 3x3 filters at stride 1, pad 1
- **MAX POOL3**: 3x3 filters at stride 2
- **FC6**: 4096 neurons
- **FC7**: 4096 neurons
- **FC8**: 1000 neurons (class scores)

How to choose these hyperparameters?
1. Trial and error 🤔
2. Computational cost (memory and tflops)

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.
Case Study: AlexNet

[Krizhevsky et al. 2012]

- High memory (feature volume) in earlier convs
- More parameters in FC than in conv
- Most FLO occurs in conv layers

Figure credit: Umich EECS 498.008 / 598.008
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

- Lin et al
- Sanchez & Perronnin
- Krizhevsky et al (AlexNet)
- Zeiler & Fergus
- Simonyan & Zisserman (VGG)
- Szegedy et al (GoogLeNet)
- He et al (ResNet)
- Shao et al
- Hu et al (SENet)
- Russakovsky et al

First CNN-based winner

- Shallow: 8 layers
- 19 layers
- 22 layers
- 152 layers
- 152 layers
- 152 layers
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

- Lin et al (2010, 28.2%)
- Sanchez & Perronnin (2011, 25.8%)
- Krizhevsky et al (AlexNet) (2012, 16.4%)
- Zeiler & Fergus (2013, 11.7%)
- Simonyan & Zisserman (VGG) (2014, 7.3%)
- Szegedy et al (GoogLeNet) (2014, 6.7%)
- He et al (ResNet) (2015, 3.6%)
- Shao et al (2016, 3%)
- Hu et al (SENet) (2017, 2.3%)
- Human (2018, 5.1%)

- ZFNet: Improved hyperparameters over AlexNet
- 152 layers
- 152 layers
- 152 layers
- 19 layers
- 22 layers

- Shallow: 8 layers
ZFNet

[Zeiler and Fergus, 2013]

AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 16.4% -> 11.7%
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

- Lin et al
- Sanchez & Perronnin
- Krizhevsky et al (AlexNet)
- Zeiler & Fergus
- Simonyan & Zisserman (VGG)
- Szegedy et al (GoogLeNet)
- He et al (ResNet)
- Shao et al
- Hu et al (SENet)
- Russakovsky et al

- 2010: 28.2
- 2011: 25.8
- 2012: 16.4
- 2013: 11.7
- 2014: 7.3
- 2014: 6.7
- 2015: 3.6
- 2016: 3
- 2017: 2.3
- Human: 5.1

Deeper Networks: 152 layers, 152 layers, 152 layers

Letters:
- shallow: 8 layers
- 8 layers
- 19 layers
- 22 layers

- LIN
- Sanchez
- Krizhevsky
- Zeiler
- Simonyan
- Szegedy
- He
- Shao
- Hu
- Russakovsky
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC’13
(ZFNet)
-> 7.3% top 5 error in ILSVRC’14
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?
Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

[7x7]
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

But deeper, more non-linearities

And fewer parameters: 3 \times (3^2C^2) vs. 7^2C^2 for C channels per layer
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Details:
- ILSVRC’14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky 2012
- No Local Response Normalisation (LRN)
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks

Still very expensive!
TOTAL memory: 24M * 4 bytes ~= 96MB / image
(only forward! ~*2 for bwd)
TOTAL params: 138M parameters
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

- Lin et al
- Sanchez & Perronnin
- Krizhevsky et al (AlexNet)
- Zeiler & Fergus
- Simonyan & Zisserman (VGG)
- Szegedy et al (GoogLeNet)
- He et al (ResNet)
- Shao et al
- Hu et al (SENet)
- Russakovsky et al

Shallow: 8 layers
19 layers
22 layers
152 layers
152 layers
152 layers

Deeper Networks
Case Study: GoogLeNet

[Szegedy et al., 2014]

Deeper networks, focus on computational efficiency

- ILSVRC’14 classification winner (6.7% top 5 error)
- 22 layers
- Only 5 million parameters!
  12x less than AlexNet
  27x less than VGG-16
- Efficient “Inception” module
- No FC layers
Case Study: GoogLeNet

[Szegedy et al., 2014]

Deeper networks, focus on computational efficiency

- ILSVRC’14 classification winner (6.7% top 5 error)
- 22 layers
- Only 5 million parameters! 12x less than AlexNet 27x less than VGG-16
- Efficient “Inception” module
- No FC layers

**Stem Network**: aggressively reduce the input feature volume
- Conv 7 x 7 x 64 with stride 2
- MaxPool
- Conv 1 x 1 x 64
- Conv 3 x 3 x 192
- MaxPool

Reduce 224 x 224 spatial solution to 28 x 28 with just 418 MFLOP! (Comparing to 7485 MFLOP of VGG)
Case Study: GoogLeNet

[Szegedy et al., 2014]

“Inception module”: design a good local network topology (network within a network) and then stack these modules on top of each other
Case Study: GoogLeNet

[Szegedy et al., 2014]

“Inception module”: design a good local network topology (network within a network) and then stack these modules on top of each other

Multiple conv filter size diversifies learned features
Case Study: GoogLeNet

[ Szegedy et al., 2014 ]

Inception module
Review: 1x1 convolutions

1x1 CONV with 32 filters

(each filter has size 1x1x64, and performs a 64-dimensional dot product)
Review: 1x1 convolutions

1x1 CONV with 32 filters

(Each filter has size 1x1x64, and performs a 64-dimensional dot product)

Alternatively, interpret it as applying the same FC layer on each input pixel.
Review: 1x1 convolutions

1x1 CONV with 32 filters preserves spatial dimensions, reduces depth!

Projects depth to lower dimension (combination of feature maps)

Alternatively, interpret it as applying the same FC layer on each input pixel.
Case Study: GoogLeNet

[Szegedy et al., 2014]

Inception module

Uses 1x1 “Bottleneck” layers to reduce channel dimension before expensive conv (we will revisit this with ResNet!)
Case Study: GoogLeNet

[Szegedy et al., 2014]

Stack Inception modules with dimension reduction on top of each other
Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

Stem Network: Conv-Pool-2x Conv-Pool
Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

Stacked Inception Modules
Case Study: GoogLeNet

[Szegedy et al., 2014]
Case Study: GoogLeNet
[Szegedy et al., 2014]

Full GoogLeNet architecture

Note: after the last convolutional layer, a global average pooling layer is used that spatially averages across each feature map, before final FC layer. No longer multiple expensive FC layers!
(Also used in ResNet)
Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

Auxiliary classification outputs to inject additional gradient at lower layers (AvgPool-1x1Conv-FC-FC-Softmax)
Related to **vanishing gradient**, will discuss further in Lecture 10
Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

22 total layers with weights
(parallel layers count as 1 layer => 2 layers per Inception module. Don’t count auxiliary output layers)
Case Study: GoogLeNet

[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient “Inception” module
- Avoids expensive FC layers
- 12x less params than AlexNet
- 27x less params than VGG-16
- ILSVRC’14 classification winner
  (6.7% top 5 error)
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

Lin et al, Sanchez & Perronnin, Krizhevsky et al (AlexNet), Zeiler & Fergus, Simonyan & Zisserman (VGG), Szegedy et al (GoogLeNet), He et al (ResNet), Shao et al, Hu et al (SENet), Russakovsky et al

“Revolution of Depth”

- 28.2 layers (2010: Lin et al)
- 25.8 layers (2011: Sanchez & Perronnin)
- 16.4 layers (2012: Krizhevsky et al)
- 11.7 layers (2013: Zeiler & Fergus)
- 7.3 layers (2014: Simonyan & Zisserman)
- 6.7 layers (2014: Szegedy et al)
- 3.6 layers (2015: He et al)
- 3 layers (2016: Shao et al)
- 2.3 layers (2017: Hu et al)
- 5.1 layers (Human)

- 8 layers (shallow)
- 19 layers
- 22 layers
- 152 layers
Case Study: ResNet

[He et al., 2015]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC’15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC’15 and COCO’15!
Case Study: ResNet

[He et al., 2015]

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?
Case Study: ResNet

[He et al., 2015]

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?
Case Study: ResNet

[He et al., 2015]

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

56-layer model performs worse on both test and training error

-> The deeper model performs worse, but it’s not caused by overfitting!
Case Study: ResNet

[He et al., 2015]

Fact: Deep models have more representation power (more parameters) than shallower models.

Hypothesis: the problem is an optimization problem, deeper models are harder to optimize
Case Study: ResNet

[He et al., 2015]

A deeper model can **emulate** a shallower model: copy layers from shallower model, set extra layers to identity

Thus deeper models should do at least **as good as** shallow models
Case Study: ResNet

[He et al., 2015]

A deeper model can **emulate** a shallower model: copy layers from shallower model, set extra layers to identity

Thus deeper models should do at least **as good as** shallow models.

Deeper models are harder to optimize. They don’t learn identity functions to emulate shallow models.

**Solution**: Change the network so learning identity functions as extra layers is easy.
Case Study: ResNet

[He et al., 2015]

Solution: Change the network so learning identity functions as extra layers is easy
Case Study: ResNet

[He et al., 2015]

Solution: Change the network so learning identity functions as extra layers is easy

H(x) = F(x) + x

Identity mapping: H(x) = x if F(x) = 0
Case Study: ResNet

[He et al., 2015]

Solution: Change the network so learning identity functions as extra layers is easy.

Identity mapping:

\[ H(x) = x \text{ if } F(x) = 0 \]

Use layers to fit residual

\[ F(x) = H(x) - x \]

instead of

\[ H(x) \] directly.
Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
Reduce the activation volume by half.
Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning (stem)
Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning (stem)
- No FC layers at the end (only FC 1000 to output classes)
- (In theory, you can train a ResNet with input image of variable sizes)
Case Study: ResNet

[He et al., 2015]

Total depths of 18, 34, 50, 101, or 152 layers for ImageNet
For deeper networks (ResNet-50+), use “bottleneck” layer to improve efficiency (similar to GoogLeNet)
For deeper networks (ResNet-50+), use “bottleneck” layer to improve efficiency (similar to GoogLeNet)

1x1 conv, 256 filters projects back to 256 feature maps (28x28x256)

3x3 conv operates over only 64 feature maps

1x1 conv, 64 filters to project to 28x28x64
Case Study: ResNet

[He et al., 2015]

Training ResNet in practice:

- Batch Normalization after every CONV layer (next lecture)
- Xavier initialization from He et al. (next lecture)
- SGD + Momentum (next lecture)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used
Case Study: ResNet

[He et al., 2015]

Experimental Results

- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lower training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: “Ultra-deep” (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd
Case Study: ResNet

[He et al., 2015]

Experimental Results
- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lower training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

MSRA @ ILSVRC & COCO 2015 Competitions

• 1st places in all five main tracks
  • ImageNet Classification: “Ultra-deep” (quote Yann) 152-layer nets
  • ImageNet Detection: 16% better than 2nd
  • ImageNet Localization: 27% better than 2nd
  • COCO Detection: 11% better than 2nd
  • COCO Segmentation: 12% better than 2nd

ILSVRC 2015 classification winner (3.6% top 5 error) -- better than “human performance”! (Russakovsky 2014)
Comparing complexity...


Comparing complexity...

Inception-v4: Resnet + Inception!


Comparing complexity...


Comparing complexity...

GoogLeNet: most efficient


Comparing complexity...

AlexNet:
Smaller compute, still memory heavy, lower accuracy


Comparing complexity...


ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

- Sanchez & Perronnin (2011)
- Zeiler & Fergus (2013)
- Simonyan & Zisserman (VGG) (2014)
- Szegedy et al (GoogLeNet) (2014)
- Shao et al (2016)
- Russakovsky et al (Human)

Network ensembling: 152 layers, 152 layers, 152 layers

- Shallow: 8 layers
- 19 layers
- 22 layers
- 152 layers
- 152 layers
- 152 layers

Layer counts:
- 2010: 28.2
- 2011: 25.8
- 2012: 16.4
- 2013: 11.7
- 2014: 7.3
- 2014: 6.7
- 2015: 3.6
- 2016: 3
- 2017: 2.3
- Human: 5.1
Improving ResNets...

“Good Practices for Deep Feature Fusion”

[Shao et al. 2016]

- Multi-scale ensembling of Inception, Inception-Resnet, Resnet, Wide Resnet models
- ILSVRC’16 classification winner

<table>
<thead>
<tr>
<th></th>
<th>Inception-v3</th>
<th>Inception-v4</th>
<th>Inception-Resnet-v2</th>
<th>Resnet-200</th>
<th>Wrn-68-3</th>
<th>Fusion (Val.)</th>
<th>Fusion (Test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Err. (%)</td>
<td>4.20</td>
<td>4.01</td>
<td>3.52</td>
<td>4.26</td>
<td>4.65</td>
<td>2.92 (-0.6)</td>
<td>2.99</td>
</tr>
</tbody>
</table>
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

- Lin et al
- Sanchez & Perronnin
- Krizhevsky et al (AlexNet)
- Zeiler & Fergus
- Simonyan & Zisserman (VGG)
- Szegedy et al (GoogLeNet)
- He et al (ResNet)
- Shao et al
- Hu et al (SENet)
- Russakovsky et al

Shallow: 8 layers
8 layers
19 layers
22 layers
152 layers
152 layers
152 layers

Adaptive feature map reweighting
Improving ResNets...

Squeeze-and-Excitation Networks (SENet)

[Hu et al. 2017]

- Add a “feature recalibration” module that learns to adaptively reweight feature maps
- Global information (global avg. pooling layer) + 2 FC layers used to determine feature map weights
- ILSVRC’17 classification winner (using ResNeXt-152 as a base architecture)
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

- 2010: Lin et al
- 2011: Sanchez & Perronnin
- 2012: Krizhevsky et al (AlexNet)
- 2013: Zeiler & Fergus
- 2014: Simonyan & Zisserman (VGG)
- 2014: Szegedy et al (GoogLeNet)
- 2015: He et al (ResNet)
- 2016: Shao et al
- 2017: Hu et al (SENet)
- 2015-2017: Russakovsky et al

- Shallow: 8 layers
- 19 layers: 2012
- 22 layers: 2014
- 152 layers: 2015, 2016, 2017
Completion of the challenge: Annual ImageNet competition no longer held after 2017 -> now moved to Kaggle.
But research into CNN architectures is still flourishing
Improving ResNets...

Identity Mappings in Deep Residual Networks

[He et al. 2016]

- Improved ResNet block design from creators of ResNet
- Creates a more direct path for propagating information throughout network
- Gives better performance

Diagram:
- conv
- ReLU
- BN
- conv
- ReLU
- BN
Improving ResNets...

Wide Residual Networks

[Zagoruyko et al. 2016]

- Argues that residuals are the important factor, not depth
- User wider residual blocks (F x k filters instead of F filters in each layer)
- 50-layer wide ResNet outperforms 152-layer original ResNet
- Increasing width instead of depth more computationally efficient (parallelizable)
Improving ResNets...

Aggregated Residual Transformations for Deep Neural Networks (ResNeXt)

[Xie et al. 2016]

- Also from creators of ResNet
- Increases width of residual block through multiple parallel pathways ("cardinality")
- Parallel pathways similar in spirit to Inception module

![Diagram of ResNeXt architecture]

```
1x1 conv, 256
3x3 conv, 64
1x1 conv, 64
```

```
1x1 conv, 256
3x3 conv, 4
1x1 conv, 4
```

```
1x1 conv, 256
3x3 conv, 4
1x1 conv, 4
```

... 32 paths ...

```
1x1 conv, 256
3x3 conv, 4
1x1 conv, 4
```

```
1x1 conv, 256
3x3 conv, 4
1x1 conv, 4
```

```
1x1 conv, 256
```
Other ideas...

Densely Connected Convolutional Networks (DenseNet)

[Huang et al. 2017]

- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse
- Showed that shallow 50-layer network can outperform deeper 152 layer ResNet
Learning to search for network architectures...

Neural Architecture Search with Reinforcement Learning (NAS)

[Zoph et al. 2016]

- “Controller” network that learns to design a good network architecture (output a string corresponding to network design)
- Iterate:
  1) Sample an architecture from search space
  2) Train the architecture to get a “reward” R corresponding to accuracy
  3) Compute gradient of sample probability, and scale by R to perform controller parameter update (i.e. increase likelihood of good architecture being sampled, decrease likelihood of bad architecture)
Learning to search for network architectures...

Learning Transferable Architectures for Scalable Image Recognition

[Zoph et al. 2017]

- Applying neural architecture search (NAS) to a large dataset like ImageNet is expensive
- Design a search space of building blocks (“cells”) that can be flexibly stacked
- NASNet: Use NAS to find best cell structure on smaller CIFAR-10 dataset, then transfer architecture to ImageNet
- Many follow-up works in this space e.g. AmoebaNet (Real et al. 2019) and ENAS (Pham, Guan et al. 2018)
But sometimes smart heuristic is better than NAS ...

**EfficientNet: Smart Compound Scaling**

[Tan and Le. 2019]

- Increase network capacity by scaling width, depth, and resolution, while balancing accuracy and efficiency.
- Search for optimal set of compound scaling factors given a compute budget (target memory & flops).
- Scale up using smart heuristic rules

depth: $d = \alpha^\phi$

width: $w = \beta^\phi$

resolution: $r = \gamma^\phi$

s.t. $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$

$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$
Efficient networks...

https://openai.com/blog/ai-and-efficiency/
Today’s Lecture

Transformer

https://paperswithcode.com/sota/image-classification-on-imagenet
Next Time: Training Deep NNs (Part 1)

- Activation Functions
- Data Preprocessing
- Weight Initialization
- Batch Normalization
- Transfer learning