

lec9

Creation Date: 13/01/2020 23:10

Last Modified Date: 14/01/2020 15:57

Lec 9 : CNN Architectures

The screenshot shows a Google Slides presentation window. The title slide has a dark background with white text. The title 'Last time: Deep learning frameworks' is centered at the top. Below it is a large empty space for notes or diagrams. At the bottom of the slide, there is a dark red footer bar containing the names 'Fei-Fei Li & Justin Johnson & Serena Yeung' on the left, 'Lecture 9 - 4' in the center, and 'University of California Berkeley May 2, 2017' on the right.

Last time: Deep learning frameworks

- (1) Easily build big computational graphs
- (2) Easily compute gradients in computational graphs
- (3) Run it all efficiently on GPU (wrap cuDNN, cuBLAS, etc)

Fei-Fei Li & Justin Johnson & Serena Yeung Lecture 9 - 4 University of California Berkeley
May 2, 2017

The screenshot shows a Google Slides presentation window. The title slide has a dark background with white text. The title 'Today: CNN Architectures' is centered at the top. Below it is a section titled 'Case Studies' with a list of neural network architectures. Further down is a section titled 'Also....' with another list of network types. At the bottom of the slide, there is a dark red footer bar containing the names 'Fei-Fei Li & Justin Johnson & Serena Yeung' on the left, 'Lecture 9 - 7' in the center, and 'University of California Berkeley May 2, 2017' on the right.

Today: CNN Architectures

Case Studies

- AlexNet
- VGG
- GoogLeNet
- ResNet

Also....

- NiN (Network in Network)
- Wide ResNet
- ResNeXT
- Stochastic Depth
- DenseNet
- FractalNet
- SqueezeNet

Fei-Fei Li & Justin Johnson & Serena Yeung Lecture 9 - 7 University of California Berkeley
May 2, 2017

Lecture 9 - Google Slides Serena

Secure https://docs.google.com/presentation/d/18XFVwCdtxApm6D3LNSbQ1zdnuk87JD8_L5O4Zz6BQQ/edit#slide=id.p

Review: LeNet-5

[LeCun et al., 1998]

The diagram illustrates the LeNet-5 architecture. It starts with an **Input** image of a handwritten digit 'K'. This is processed by **Convolutions**, which produce **Image Maps** (green and orange layers). Arrows point from the input to the first convolution layer and from the first convolution layer to the second. The second convolution layer is highlighted in orange. Subsampling (Pooling) layers are shown as layers where each unit in the next layer is connected to a specific subset of units in the previous layer. These are labeled **Subsampling**. Finally, the output is produced by **Fully Connected** layers, represented by layers of triangles.

Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

Fei-Fei Li & Justin Johnson & Serena Yeung Lecture 9 - 8 University of California, Berkeley, May 2, 2017

- LeNet is one of the oldest CNN paper.

AlexNet

Case Study: AlexNet

[Krizhevsky et al. 2012]

Architecture:

CONV1

MAX POOL1

NORM1

CONV2

MAX POOL2

NORM2

CONV3

CONV4

CONV5

Max POOL3

FC6

FC7

FC8

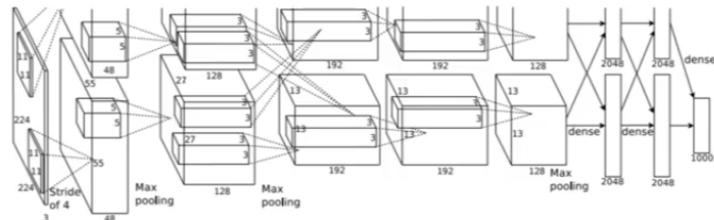


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

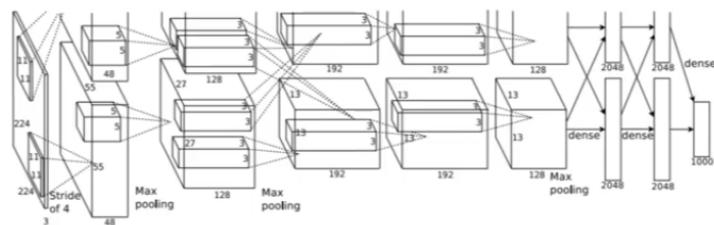
Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 9

University
May 2, 2017

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.

Fei-Fei Li & Justin Johnson & Serena Yeung

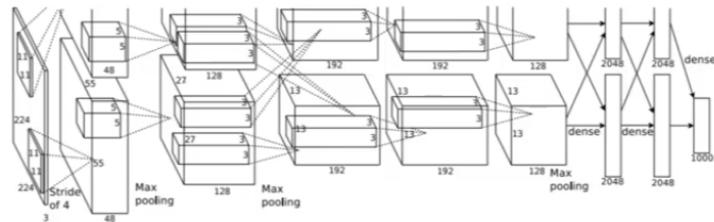
Lecture 9 - 10

University
May 2, 2017

- Output size: 55x55x96

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

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

=>

Output volume **[55x55x96]**

Parameters: $(11 \times 11 \times 3) \times 96 = 35\text{K}$

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, University of Toronto, 2012. All rights reserved.

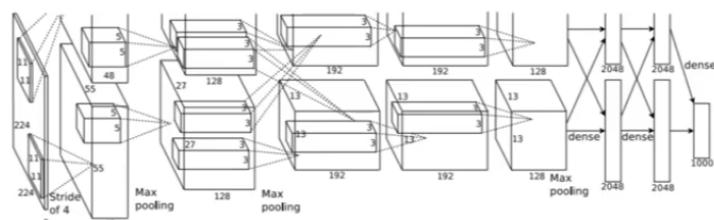
Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 12

University
May 2, 2017

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: $(55-3)/2+1 = 27$

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, University of Toronto, 2012. All rights reserved.

Fei-Fei Li & Justin Johnson & Serena Yeung

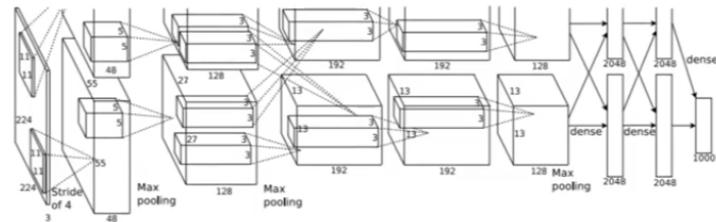
Lecture 9 - 13

University
May 2, 2017

- The output volume will be 27x27x96

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!

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, all rights reserved.

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 15

University
May 2, 2017

- Pooling layers has no parameters.

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

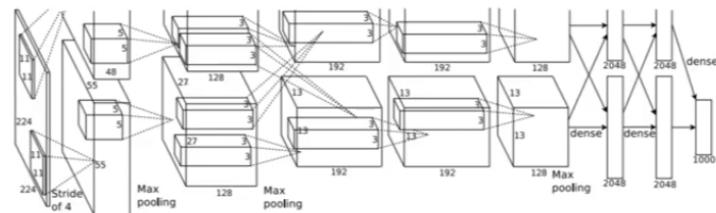
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[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, all rights reserved.

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 18

University
May 2, 2017

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

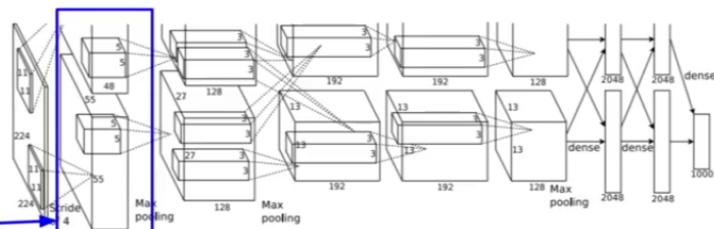
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

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

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)



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, University of Toronto, released under a Creative Commons license.

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 19

University
May 2, 2017

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

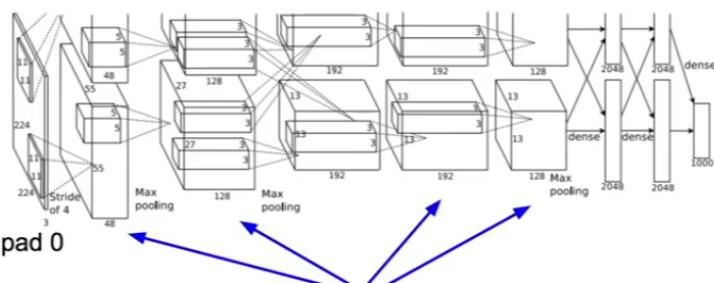
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

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

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)



CONV1, CONV2, CONV4, CONV5:
Connections only with feature maps
on same GPU

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, University of Toronto, released under a Creative Commons license.

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 20

University
May 2, 2017

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

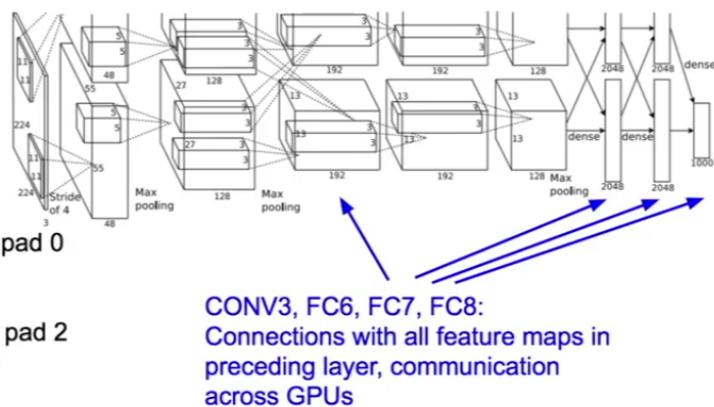
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[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, University of Toronto, 2012. Reproduced with permission.

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 21

University
May 2, 2017

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

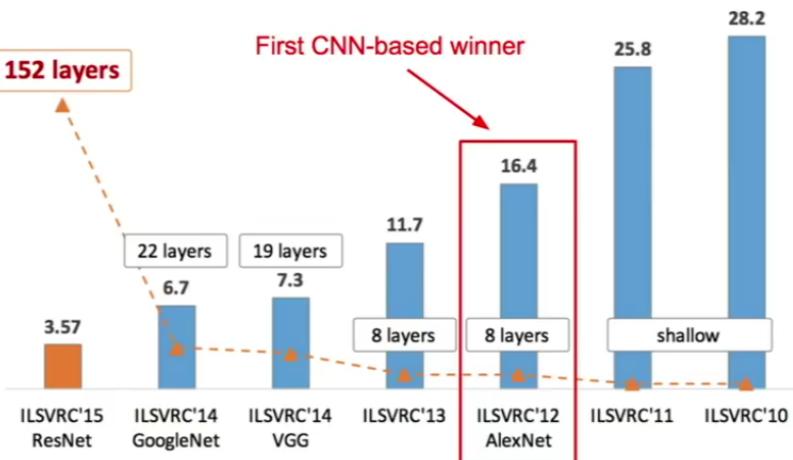


Figure copyright Kaiming He, J. Fei-Fei Li, and R. Girshick, 2015. Reproduced with permission.

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 22

University
May 2, 2017

VGGNet

Case Study: VGGNet

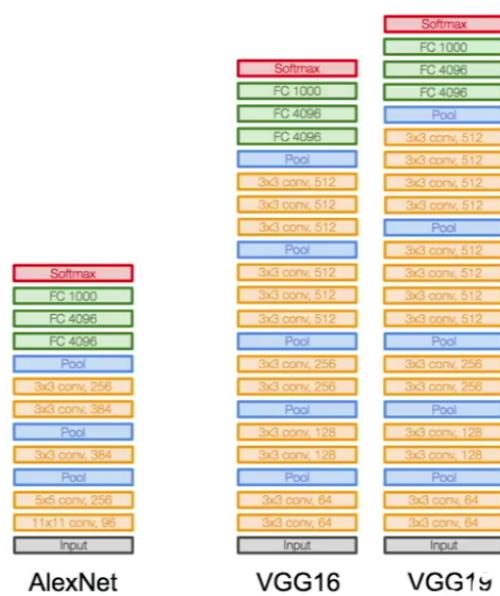
[Simonyan and Zisserman, 2014]

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



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 26

University
May 2, 2017

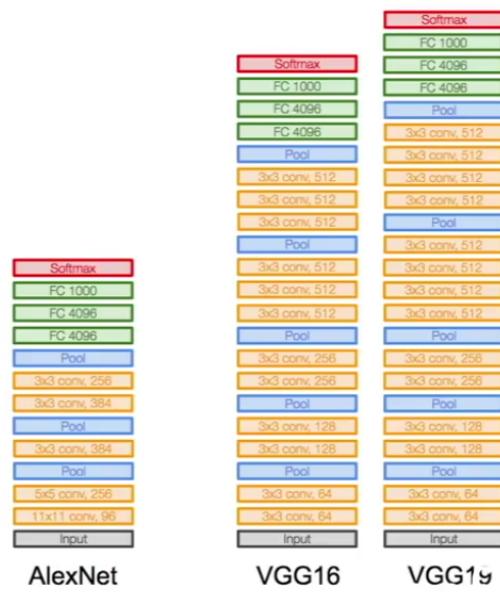
Case Study: VGGNet

[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?



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 28

University
May 2, 2017

Case Study: VGGNet

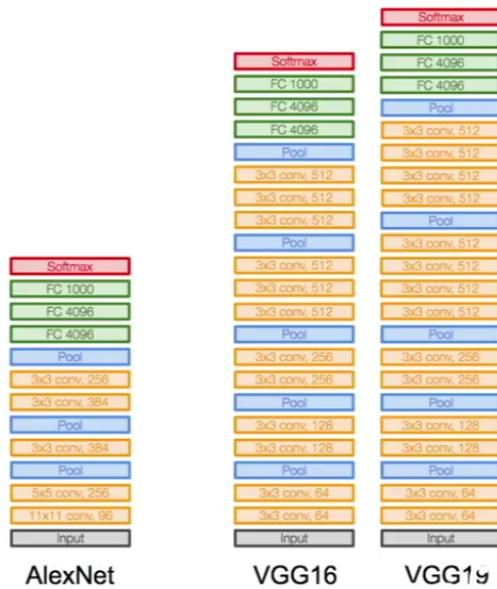
[Simonyan and Zisserman, 2014]

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: $3 * (3^2 C^2)$ vs. $7^2 C^2$ for C channels per layer

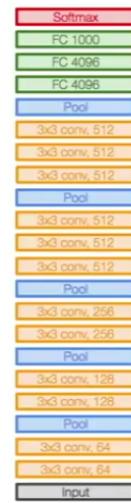


Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 30

University
May 2, 2017

INPUT: [224x224x3] memory: $224 \times 224 \times 3 = 150K$ params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2M$ params: $(3 \times 3 \times 3) \times 64 = 1,728$
CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2M$ params: $(3 \times 3 \times 64) \times 64 = 36,864$
POOL2: [112x112x64] memory: $112 \times 112 \times 64 = 800K$ params: 0
CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6M$ params: $(3 \times 3 \times 64) \times 128 = 73,728$
CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6M$ params: $(3 \times 3 \times 128) \times 128 = 147,456$
POOL2: [56x56x128] memory: $56 \times 56 \times 128 = 400K$ params: 0
CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 128) \times 256 = 294,912$
CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 256) \times 256 = 589,824$
CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 256) \times 256 = 589,824$
POOL2: [28x28x256] memory: $28 \times 28 \times 256 = 200K$ params: 0
CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 256) \times 512 = 1,179,648$
CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
POOL2: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: 0
CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
POOL2: [7x7x512] memory: $7 \times 7 \times 512 = 25K$ params: 0
FC: [1x1x4096] memory: 4096 params: $7 \times 7 \times 512 \times 4096 = 102,760,448$
FC: [1x1x4096] memory: 4096 params: $4096 \times 4096 = 16,777,216$
FC: [1x1x1000] memory: 1000 params: $4096 \times 1000 = 4,096,000$



TOTAL memory: 24M * 4 bytes \approx 96MB / image (only forward! \sim^2 for bwd)
TOTAL params: 138M parameters

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 32

University
May 2, 2017

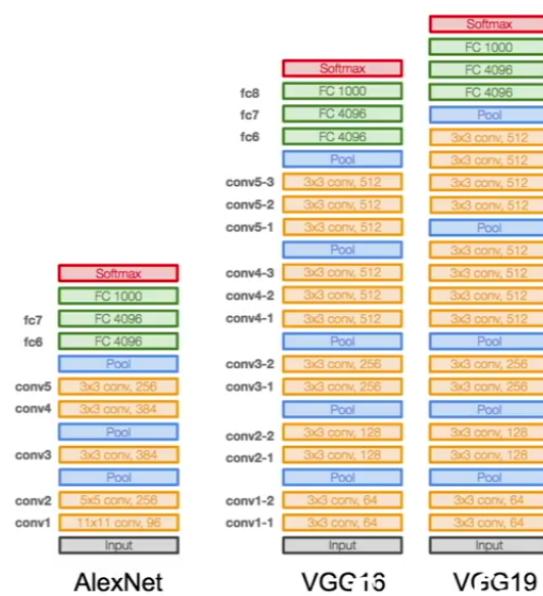
- MOST memory is taken by the initial CONV layers
 - MOST parameters are in the end FC layers

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



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 35

University
May 2, 2017

- The FC7 layer of VGG16 is commonly used for various other tasks during Transfer Learning.
- After FC7, usually we connect our required layer. For example, If we are supposed to classify 10 classes then we can use a FC layer of 10 neurons after FC7, and then fine-tune the model.

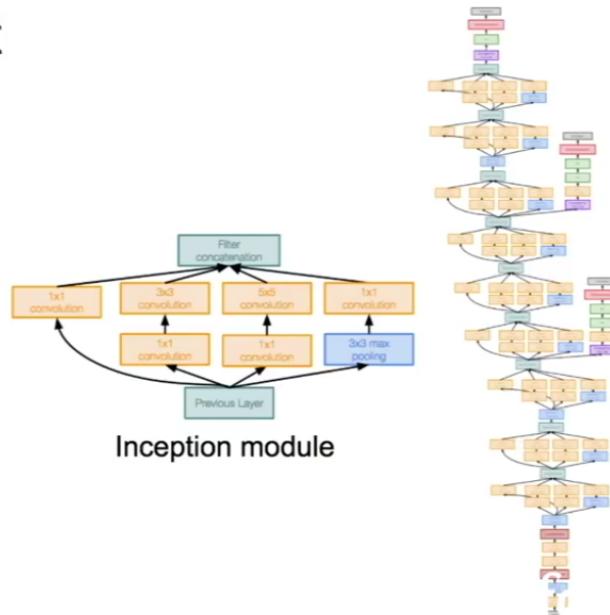
GoogleNet

Case Study: GoogLeNet

[Szegedy et al., 2014]

Deeper networks, with computational efficiency

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



Fei-Fei Li & Justin Johnson & Serena Yeung

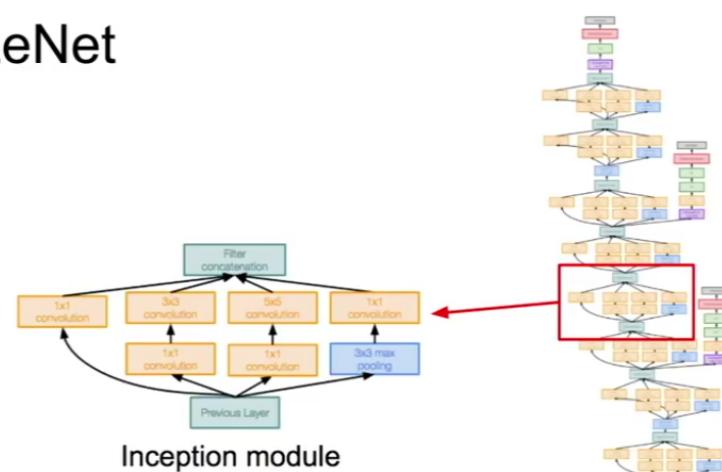
Lecture 9 - 37

University
May 2, 2017

Case Study: GoogLeNet

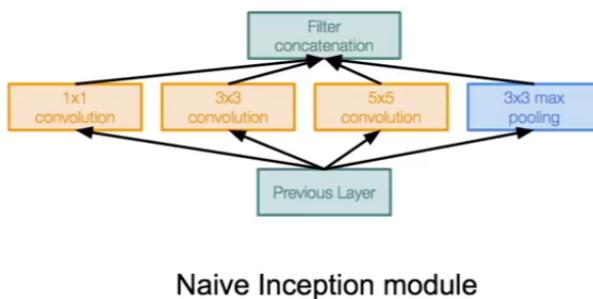
[Szegedy et al., 2014]

“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]



Apply parallel filter operations on the input from previous layer:

- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

Q: What is the problem with this?
[Hint: Computational complexity]

- We apply different size of filters (1X1, 3X3, 5X5) and also a pooling of 3X3 and then concatenate(stack) them to create the processed layer.

Lecture 9 - Google Slides Serena

[Secure https://docs.google.com/presentation/d/18XFVwCdxApMs6D3LNSbQlzdnu87JD8_L5O4Zz6BQQ/edit#slide=id.p](https://docs.google.com/presentation/d/18XFVwCdxApMs6D3LNSbQlzdnu87JD8_L5O4Zz6BQQ/edit#slide=id.p)

Case Study: GoogLeNet

[Szegedy et al., 2014]

Example: Q1: What is the output size of the 1x1 conv, with 128 filters?

```

graph TD
    Input[Input: 28x28x256] --> 1x1[1x1 conv, 128]
    Input --> 3x3_1[3x3 conv, 192]
    Input --> 5x5[5x5 conv, 96]
    Input --> Pool[3x3 pool]
    1x1 --> Conc[Filter concatenation]
    3x3_1 --> Conc
    5x5 --> Conc
    Pool --> Conc
    Conc --> Output[Output: 28x28x128]
  
```

Naive Inception module

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 42

University May 2, 2017

- The output of 1x1 conv will be 28X28X128

Lecture 9 - Google Slides Serena

[Secure https://docs.google.com/presentation/d/18XFVwCdxApMs6D3LNSbQlzdnu87JD8_L5O4Zz6BQQ/edit#slide=id.p](https://docs.google.com/presentation/d/18XFVwCdxApMs6D3LNSbQlzdnu87JD8_L5O4Zz6BQQ/edit#slide=id.p)

Case Study: GoogLeNet

[Szegedy et al., 2014]

Example: Q2: What are the output sizes of all different filter operations?

```

graph TD
    Input[Input: 28x28x256] --> 1x1[1x1 conv, 128]
    Input --> 3x3_1[3x3 conv, 192]
    Input --> 5x5[5x5 conv, 96]
    Input --> Pool[3x3 pool]
    1x1 --> Conc[Filter concatenation]
    3x3_1 --> Conc
    5x5 --> Conc
    Pool --> Conc
    Conc --> Output[Output: 28x28x128]
    1x1 -- "28x28x128" --> 1x1
    3x3_1 -- "28x28x192" --> 3x3_1
    5x5 -- "28x28x96" --> 5x5
    Pool -- "28x28x256" --> Pool
  
```

Naive Inception module

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 45

University May 2, 2017

- The output of 3x3 conv will be 28x28x192.
- The output of 5x5 conv will be 28x28x96.
- The output of 3x3 conv will be 28x28x256.
- Note: **POOLING LAYER PRESERVES THE DEPTH**
- Note: **We apply padding to preserve the spatial dimension of 28x28**

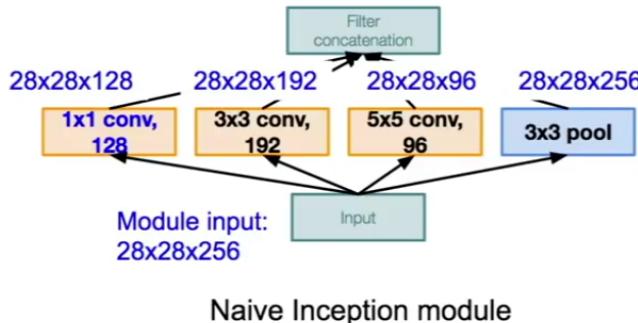
Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q3: What is output size after filter concatenation?

$$28 \times 28 \times (128 + 192 + 96 + 256) = 28 \times 28 \times 672$$



Q: What is the problem with this?
[Hint: Computational complexity]

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 47

University
May 2, 2017

- The output size after concatenation will be $28 \times 28 \times (128 + 192 + 96 + 256) = 28 \times 28 \times 672$
- So in Inception module, we maintain the spatial dimension but increase the depth

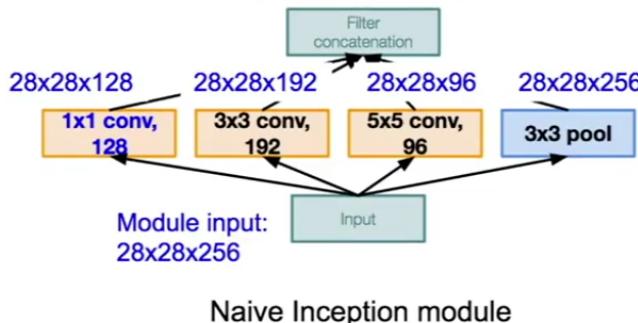
Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q3: What is output size after filter concatenation?

$$28 \times 28 \times (128 + 192 + 96 + 256) = 28 \times 28 \times 672$$



Q: What is the problem with this?
[Hint: Computational complexity]

Conv Ops:

[1x1 conv, 128] $28 \times 28 \times 128 \times 1 \times 1 \times 256$
[3x3 conv, 192] $28 \times 28 \times 192 \times 3 \times 3 \times 256$
[5x5 conv, 96] $28 \times 28 \times 96 \times 5 \times 5 \times 256$

Total: 854M ops

Very expensive compute

Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer!

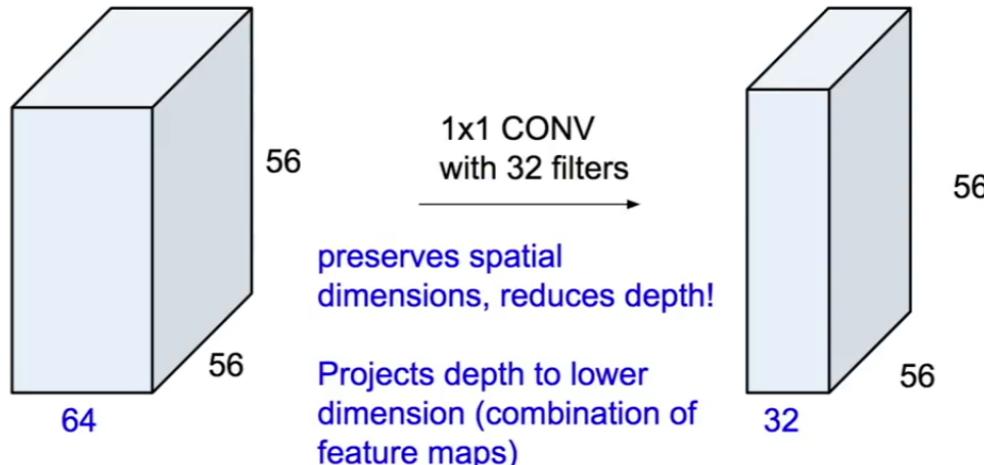
Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 49

University
May 2, 2017

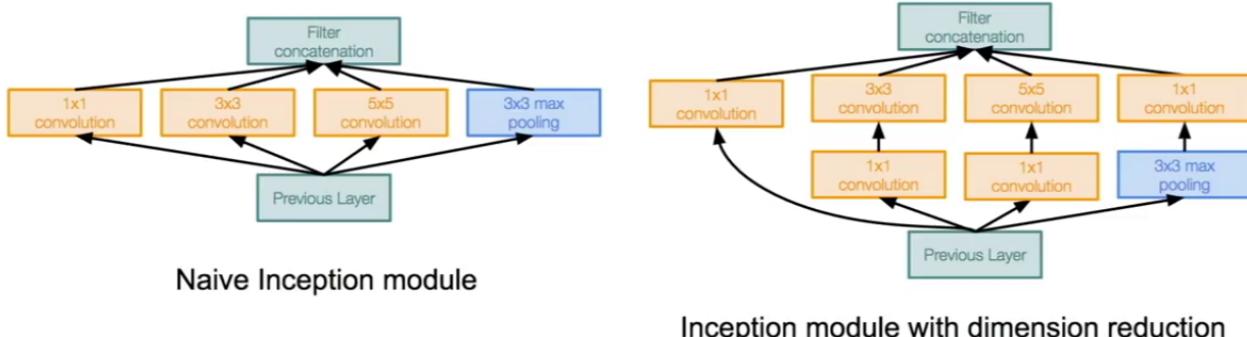
- Pooling layer preserves the depth, hence passing on a lot of parameters to further layers.
- Solution: Use Bottle-neck layers, i.e., use k 1X1 convolution filters to reduce the depth.

Reminder: 1x1 convolutions



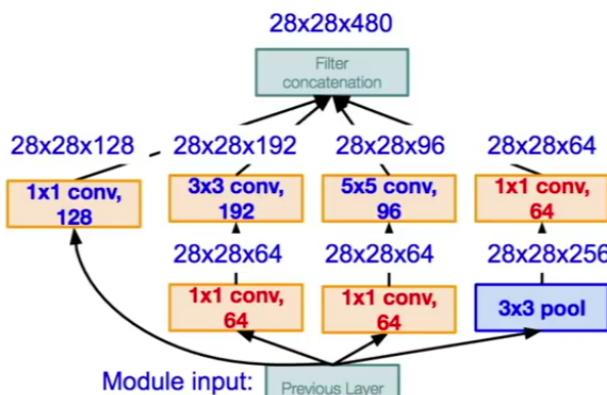
Case Study: GoogLeNet

[Szegedy et al., 2014]



Case Study: GoogLeNet

[Szegedy et al., 2014]



Inception module with dimension reduction

Using same parallel layers as naive example, and adding “1x1 conv, 64 filter” bottlenecks:

Conv Ops:

- [1x1 conv, 64] 28x28x64x1x1x256
- [1x1 conv, 64] 28x28x64x1x1x256
- [1x1 conv, 128] 28x28x128x1x1x256
- [3x3 conv, 192] 28x28x192x3x3x64
- [5x5 conv, 96] 28x28x96x5x5x64
- [1x1 conv, 64] 28x28x64x1x1x256

Total: 358M ops

Compared to 854M ops for naive version

Bottleneck can also reduce depth after pooling layer

Fei-Fei Li & Justin Johnson & Serena Yeung

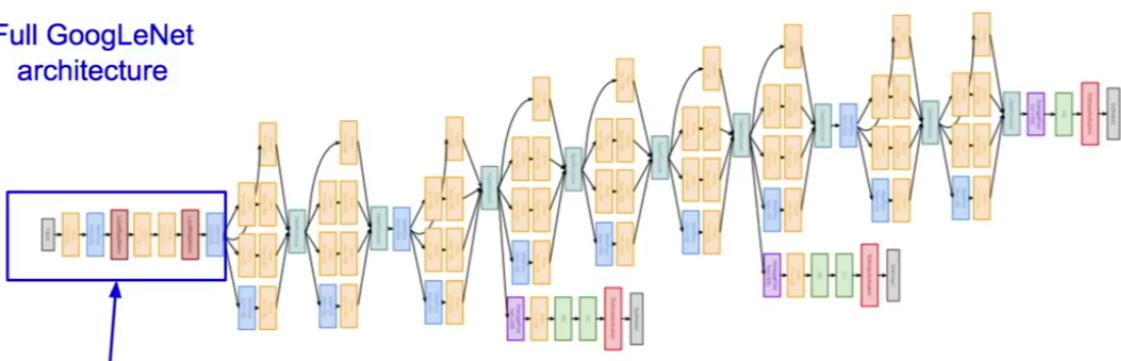
Lecture 9 - 55

University
May 2, 2017

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



Stem Network:
Conv-Pool-
2x Conv-Pool

Fei-Fei Li & Justin Johnson & Serena Yeung

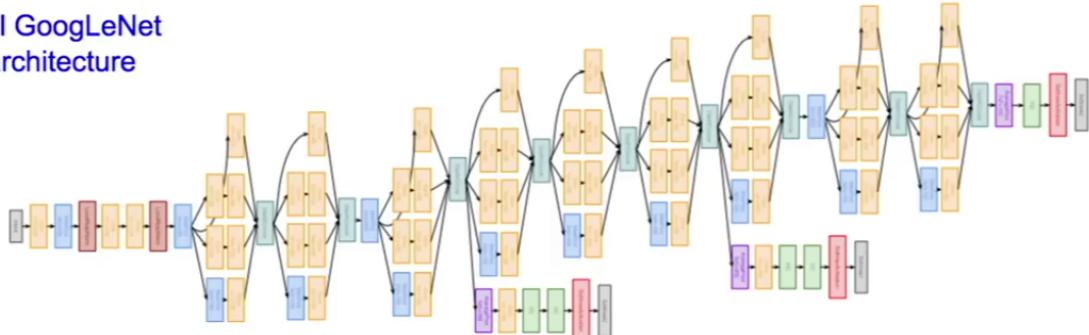
Lecture 9 - 57

University
May 2, 2017

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



22 total layers with weights (including each parallel layer in an Inception module)

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 62

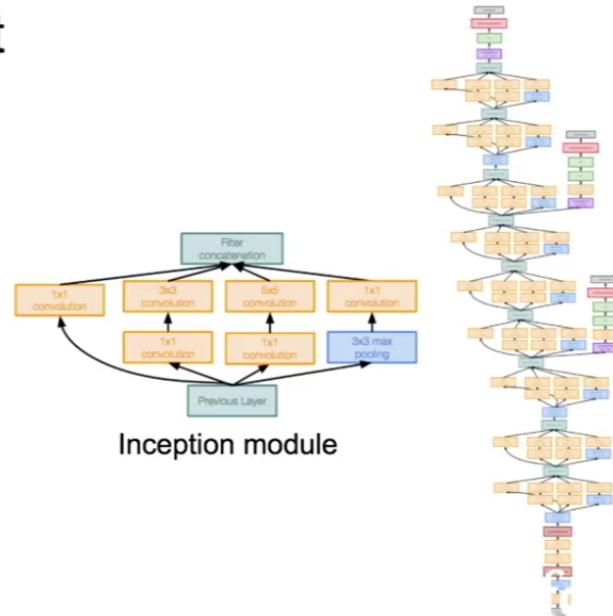
University
May 2, 2017

Case Study: GoogLeNet

[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient “Inception” module
- No FC layers
- 12x less params than AlexNet
- ILSVRC’14 classification winner (6.7% top 5 error)



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 63

University
May 2, 2017

- NO FC layers.
- Less params compared to AlexNet.

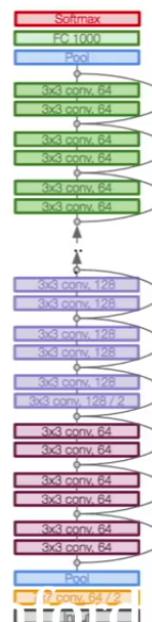
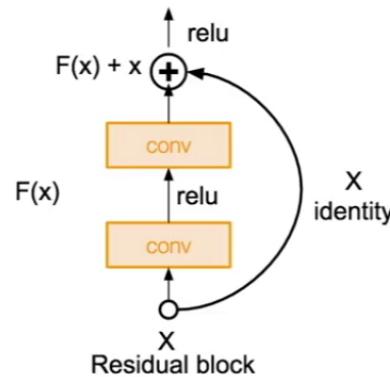
ResNet

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!



Fei-Fei Li & Justin Johnson & Serena Yeung

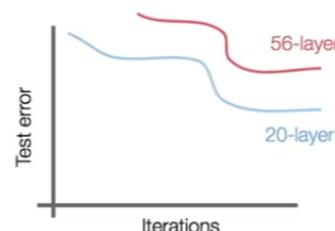
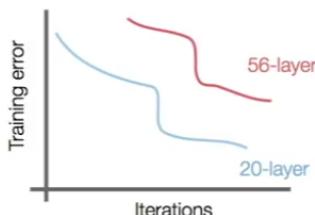
Lecture 9 - 65

University
May 2, 2017

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 training and test error
-> The deeper model performs worse, but it's not caused by overfitting!

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 68

University
May 2, 2017

- We can notice that the Training and Test error for the 56-layer network is higher compared to the 20-layer network.
- This shouldn't be the case since we are having more parameters, the error rate should be low for training at the least.
- The problem is of optimization and not of overfitting. Deeper networks are harder to optimize compared to shallower networks.

Case Study: ResNet

[He et al., 2015]

Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize

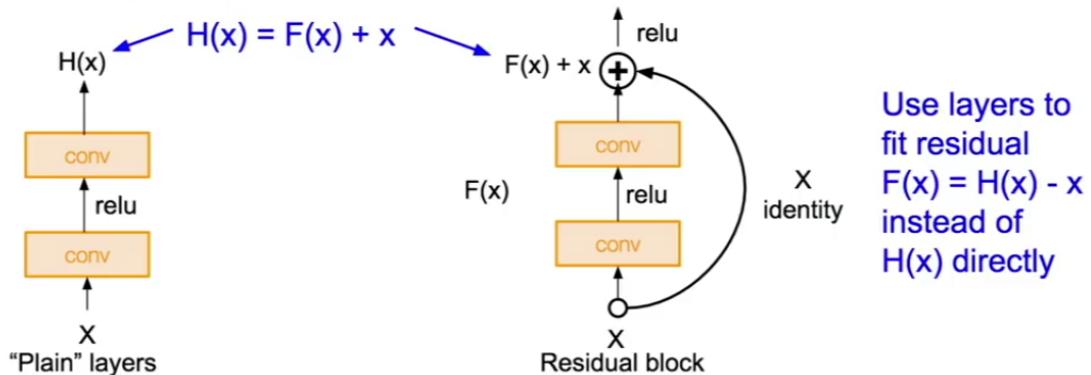
The deeper model should be able to perform at least as well as the shallower model.

A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping.

Case Study: ResNet

[He et al., 2015]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



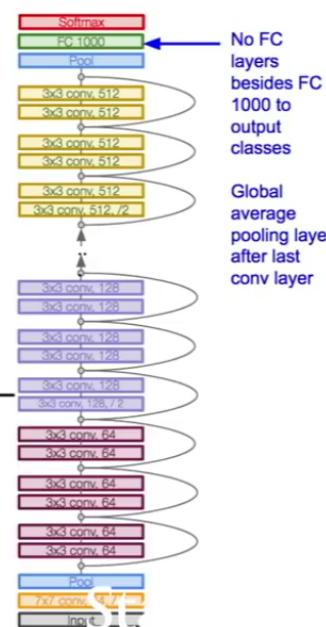
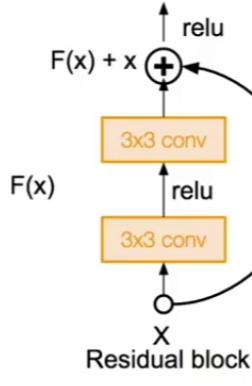
- We know it's hard to learn $H(x)$ because in this case, the network is deeper in nature.
- Here $F(x)$ is the residual.
- **Why should learning the residual [$F(x)$] be easier than learning the hypothesis [$H(x)$]?**
- Ans: This is just the hypothesis of the author, the hypothesis is that if we are learning the residual then we just have to learn the delta to X . i.e., $F(x)$, if we have some number of these shallow layers that were learned and we had all these identity mappings at the top, which implies that these layers are somewhat close to identity, would be a good layer.
- We pad the layers to fit the sizes.

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
- No FC layers at the end (only FC 1000 to output classes)

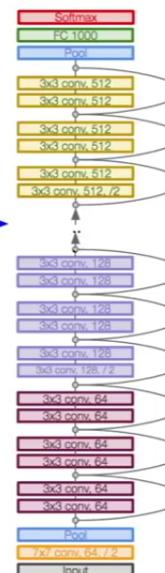


- Variants of ResNet:

Case Study: ResNet

[He et al., 2015]

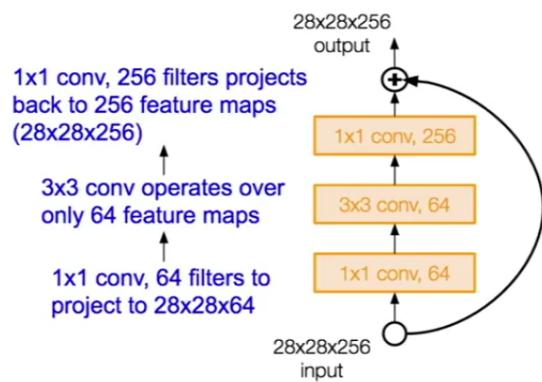
Total depths of 34, 50, 101, or 152 layers for ImageNet



Case Study: ResNet

[He et al., 2015]

For deeper networks
(ResNet-50+), use “bottleneck”
layer to improve efficiency
(similar to GoogLeNet)



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 79

University
May 2, 2017

Case Study: ResNet

[He et al., 2015]

Training ResNet in practice:

- Batch Normalization after every CONV layer
- Xavier/2 initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 80

University
May 2, 2017

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

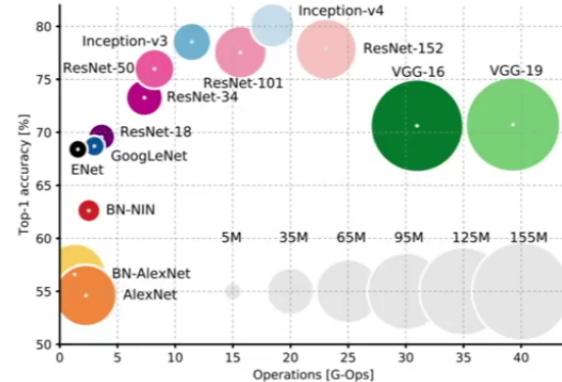
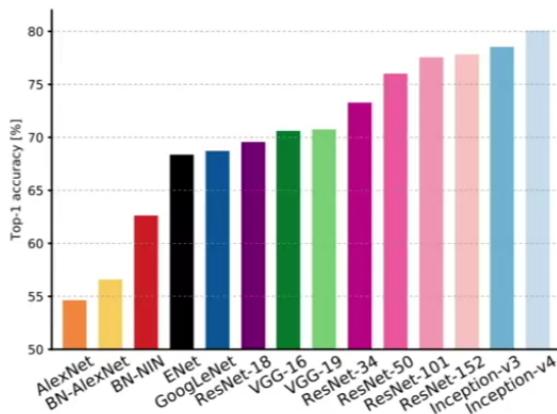
- ImageNet Classification: "Ultra-deep" (quote Yann) 152-layer nets
- ImageNet Detection: 16% better than 2nd
- ImageNet Localization: 27% better than 2nd
- COCO Detection: 11% better than 2nd
- COCO Segmentation: 12% better than 2nd

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 81

University
May 2, 2017

Comparing complexity...



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.

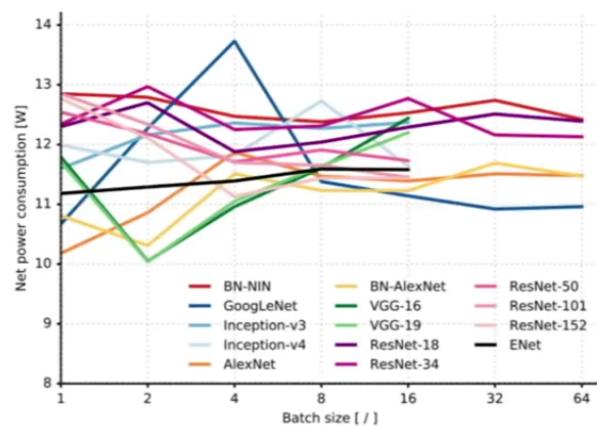
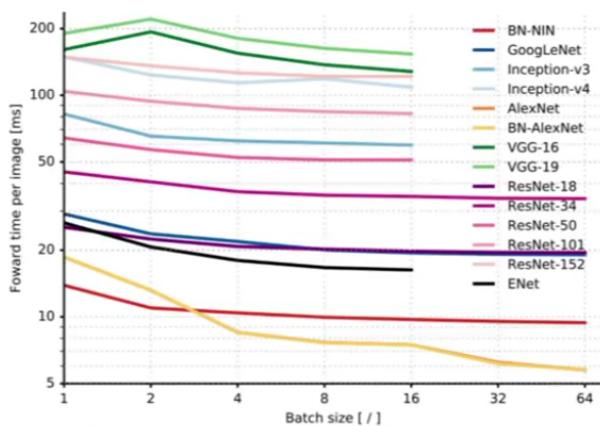
Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 84

University
May 2, 2017

- The size of the circle shows memory consumption
- VGG: Highest memory, most operations.
- GoogLeNet: most efficient
- AlexNet: Smaller Compute, Low Accuracy, still memory heavy.
- ResNet: Moderate efficiency, depending on model, highest accuracy.

Forward pass time and power consumption



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Figures copyright Alfredo Canziani, Adam Paszke, Eugenio Culurciello, 2017. Reproduced with permission.

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 90

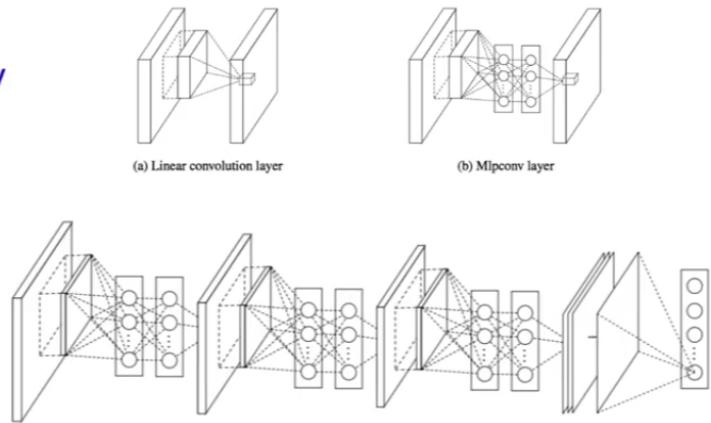
University
May 2, 2017

Other Architectures

Network in Network (NiN)

[Lin et al. 2014]

- Mlpconv layer with “micronetwork” within each conv layer to compute more abstract features for local patches
- Micronetwork uses multilayer perceptron (FC, i.e. 1x1 conv layers)
- Precursor to GoogLeNet and ResNet “bottleneck” layers
- Philosophical inspiration for GoogLeNet



Figures copyright Lin et al., 2014. Reproduced with permission.

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 92

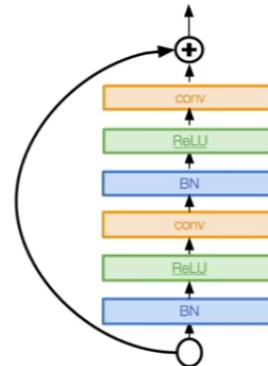
University
May 2, 2017

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 (moves activation to residual mapping pathway)
- Gives better performance

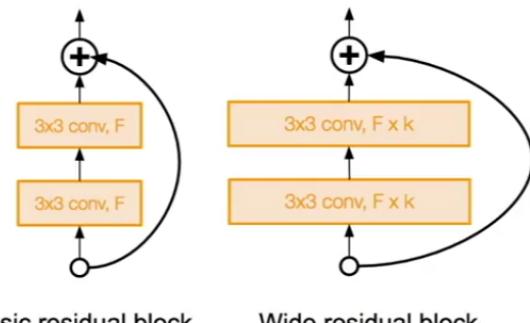


Improving ResNets...

Wide Residual Networks

[Zagoruyko et al. 2016]

- Argues that residuals are the important factor, not depth
- User wider residual blocks ($F \times k$ 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)

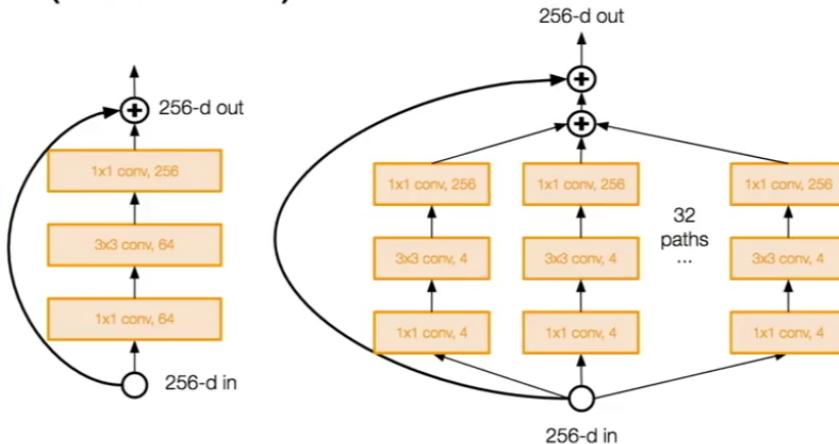


Improving ResNets...

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



And this other paper around the same time,

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 95

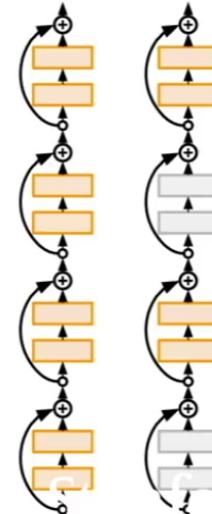
University May 2, 2017

Improving ResNets...

Deep Networks with Stochastic Depth

[Huang et al. 2016]

- Motivation: reduce vanishing gradients and training time through short networks during training
- Randomly drop a subset of layers during each training pass
- Bypass with identity function
- Use full deep network at test time



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 9 - 96

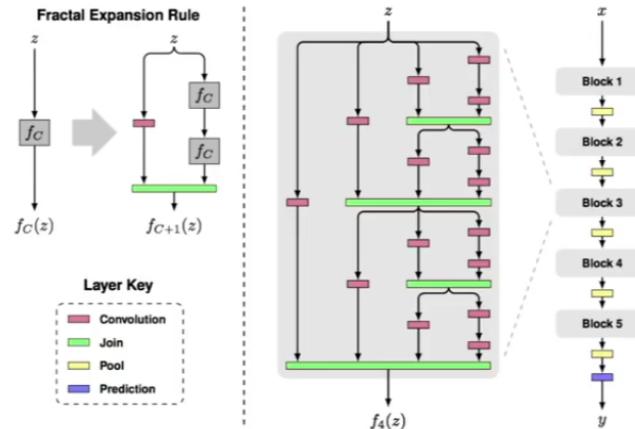
University May 2, 2017

Beyond ResNets...

FractalNet: Ultra-Deep Neural Networks without Residuals

[Larsson et al. 2017]

- Argues that key is transitioning effectively from shallow to deep and residual representations are not necessary
- Fractal architecture with both shallow and deep paths to output
- Trained with dropping out sub-paths
- Full network at test time



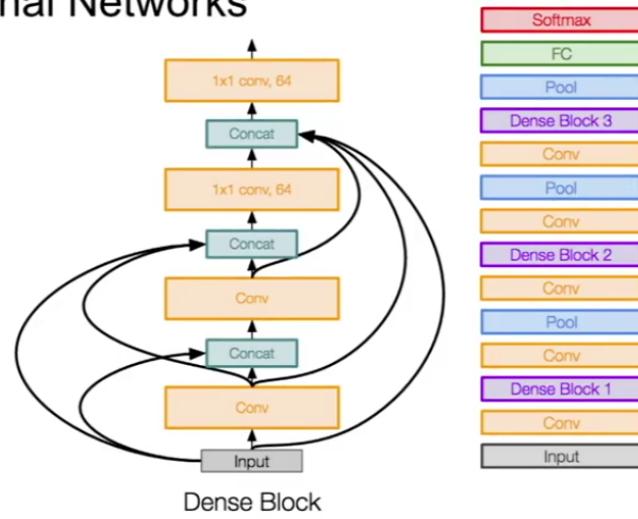
Figures copyright Larsson et al., 2017. Reproduced with permission.

Beyond ResNets...

Densely Connected Convolutional Networks

[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



Efficient networks...

SqueezeNet: AlexNet-level Accuracy With 50x Fewer Parameters and <0.5Mb Model Size

[Iandola et al. 2017]

- Fire modules consisting of a 'squeeze' layer with 1x1 filters feeding an 'expand' layer with 1x1 and 3x3 filters
- AlexNet level accuracy on ImageNet with 50x fewer parameters
- Can compress to 510x smaller than AlexNet (0.5Mb)

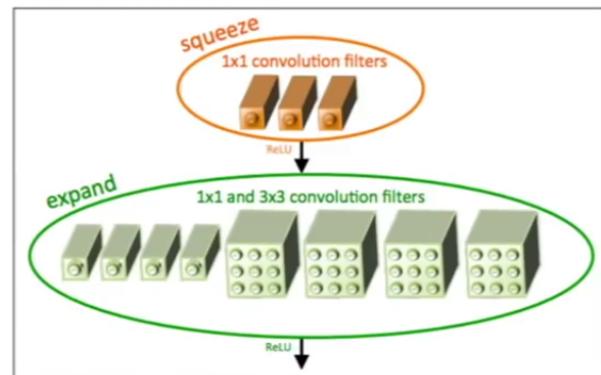


Figure copyright Iandola, Han, Moskewicz, Ashraf, Dally, Keutzer, 2017. Reproduced with permission.

Summary: CNN Architectures

Case Studies

- AlexNet
- VGG
- GoogLeNet
- ResNet

Also....

- NiN (Network in Network)
- Wide ResNet
- ResNeXT
- Stochastic Depth
- DenseNet
- FractalNet
- SqueezeNet

Lecture 9 - Google Slides

Secure https://docs.google.com/presentation/d/18XFVwCdtxApm6D3LNSbQ1zdnuk87JD8_L5O4Zz6BQQ/edit#slide=id.p

✓ Serena

Summary: CNN Architectures

- VGG, GoogLeNet, ResNet all in wide use, available in model zoos
- ResNet current best default
- Trend towards extremely deep networks
- Significant research centers around design of layer / skip connections and improving gradient flow
- Even more recent trend towards examining necessity of depth vs. width and residual connections
- Next time: Recurrent neural networks