Deep Learning for Computer Vision

Evolution of CNN Architectures: InceptionNet, ResNet

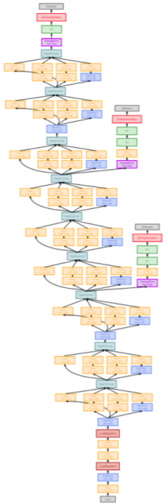
Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

InceptionNet (GoogleNet)

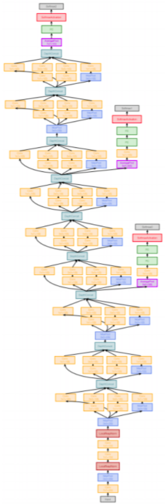
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

InceptionNet (GoogleNet)

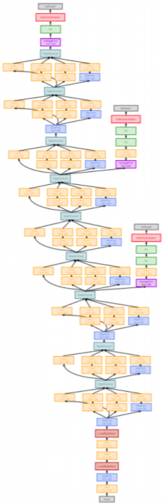
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation
- 22 layers



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

InceptionNet (GoogleNet)

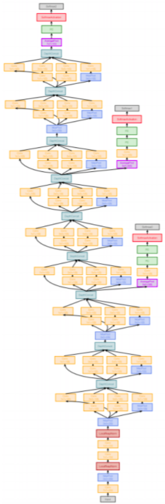
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation
- 22 layers
- No FC layers



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

InceptionNet (GoogleNet)

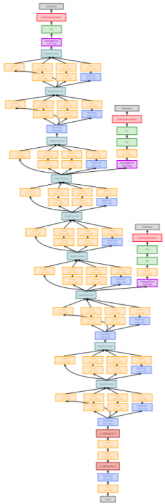
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation
- 22 layers
- No FC layers
- Efficient "**Inception**" module



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

InceptionNet (GoogleNet)

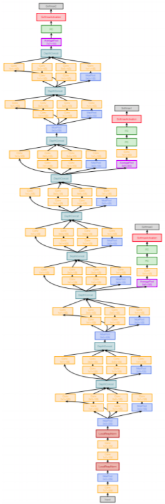
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation
- 22 layers
- No FC layers
- Efficient "**Inception**" module
- Only 5 million parameters! (12x less than AlexNet)



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

InceptionNet (GoogleNet)

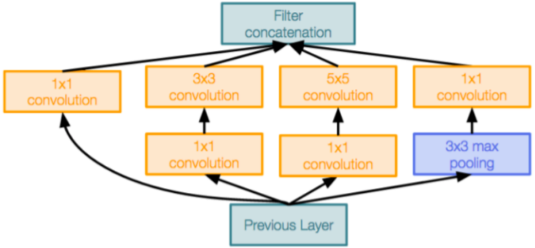
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation
- 22 layers
- No FC layers
- Efficient "**Inception**" module
- Only 5 million parameters! (12x less than AlexNet)
- **ILSVRC'14 classification winner** (6.7% top-5 error)



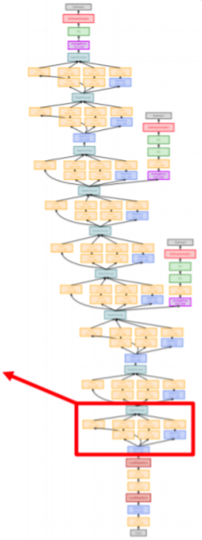
Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

InceptionNet (GoogleNet)

- **Inception module:** Local unit with parallel branches
- Local structure repeated many times throughout the network



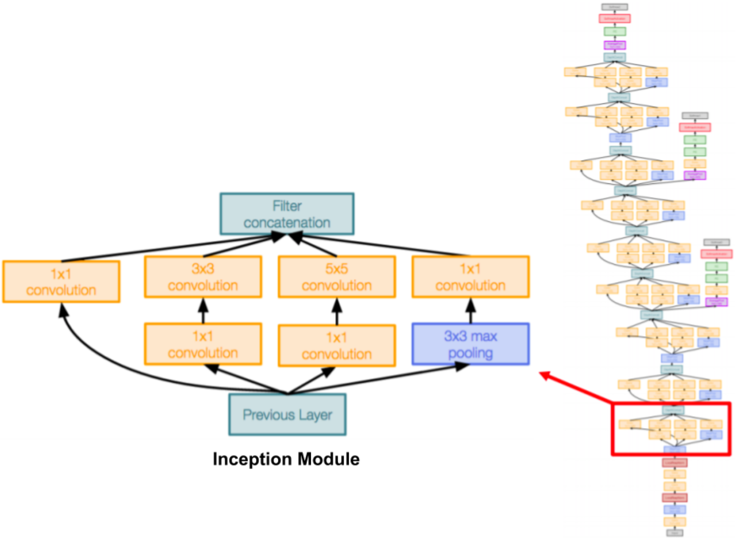
Inception Module



Credit: Justin Johnson, Univ of Michigan

InceptionNet (GoogleNet)

- **Inception module:** Local unit with parallel branches
- Local structure repeated many times throughout the network

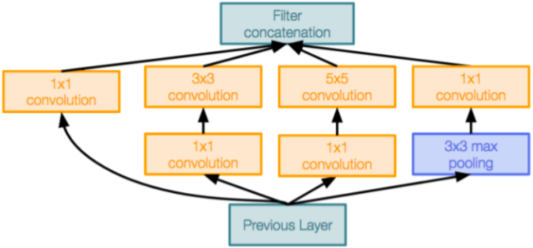


Inception Module

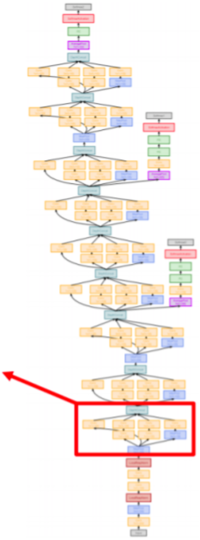
Credit: Justin Johnson, Univ of Michigan

InceptionNet (GoogleNet)

- **Inception module:** Local unit with parallel branches
- Local structure repeated many times throughout the network

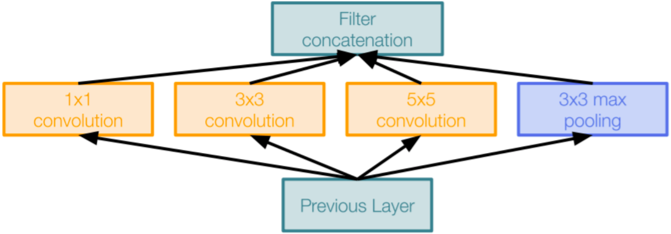


Inception Module



Credit: Justin Johnson, Univ of Michigan

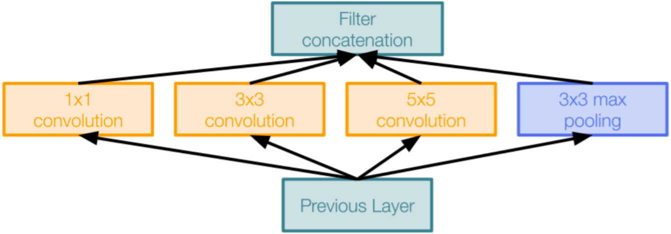
InceptionNet (GoogleNet)



Naive Inception module

- Apply parallel filter operations on the input from previous layer:
  - Multiple receptive field sizes for convolution (1×1, 3×3, 5×5)
  - Pooling operation (3×3 max pooling)
- Concatenate all filter outputs together depth-wise

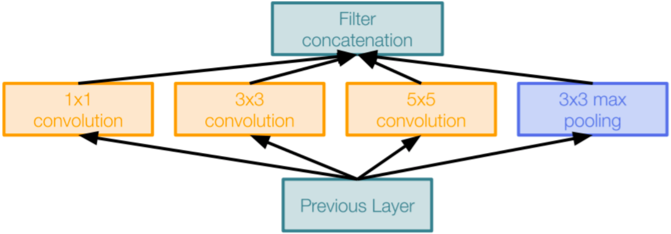
InceptionNet (GoogleNet)



Naive Inception module

- Apply parallel filter operations on the input from previous layer:
  - Multiple receptive field sizes for convolution (1×1, 3×3, 5×5)
  - Pooling operation (3×3 max pooling)
- Concatenate all filter outputs together depth-wise
- **What's the problem with this?**

InceptionNet (GoogleNet)



Naive Inception module

- Apply parallel filter operations on the input from previous layer:
  - Multiple receptive field sizes for convolution (1×1, 3×3, 5×5)
  - Pooling operation (3×3 max pooling)
- Concatenate all filter outputs together depth-wise
- **What's the problem with this?** Computationally very expensive

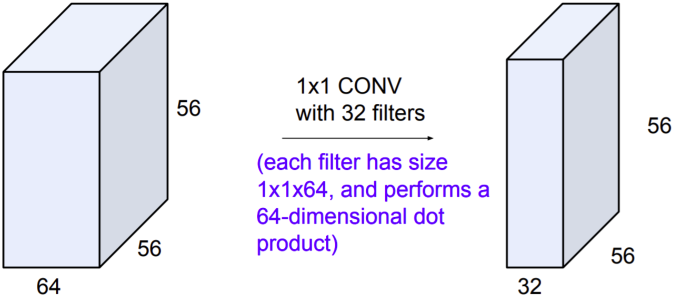
Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

InceptionNet (GoogleNet)

Solution: Use 1×1 "Bottleneck" layers to reduce channel dimension before expensive conv layers

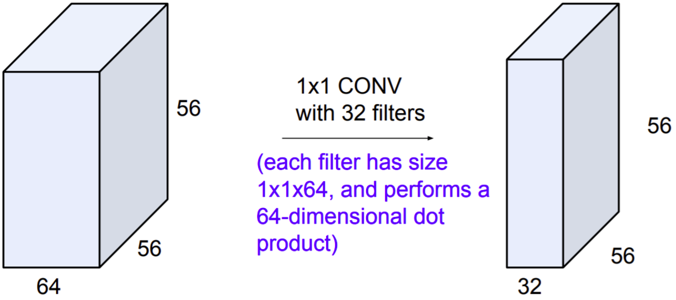
InceptionNet (GoogleNet)

Solution: Use 1×1 “Bottleneck” layers to reduce channel dimension before expensive conv layers



InceptionNet (GoogleNet)

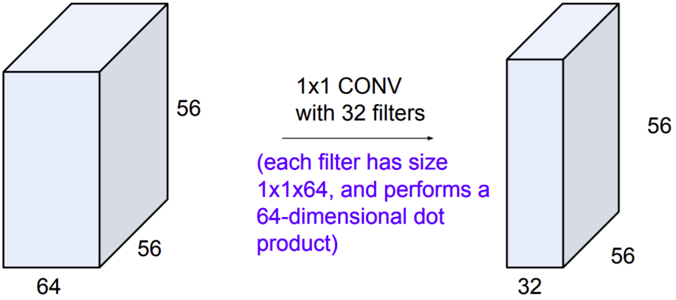
Solution: Use 1×1 "Bottleneck" layers to reduce channel dimension before expensive conv layers



- Preserves spatial dimensions, reduces depth!

InceptionNet (GoogleNet)

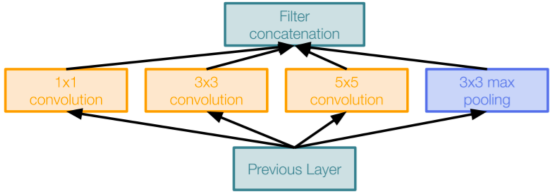
Solution: Use 1×1 “Bottleneck” layers to reduce channel dimension before expensive conv layers



- Preserves spatial dimensions, reduces depth!
- Projects depth to lower dimension (combination of feature maps)

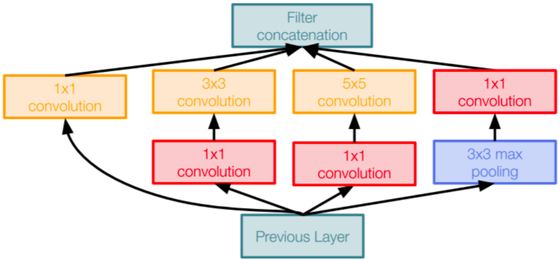
Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

InceptionNet (GoogleNet)



Naive Inception module

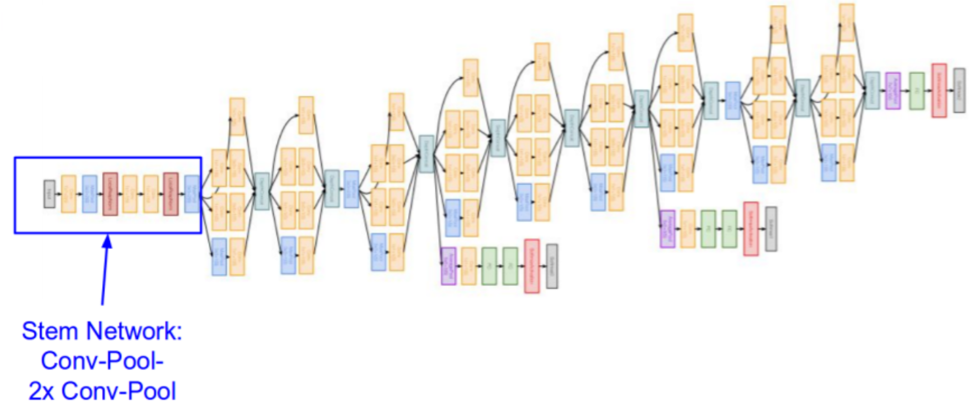
1x1 conv “bottleneck” layers



Inception module with dimension reduction

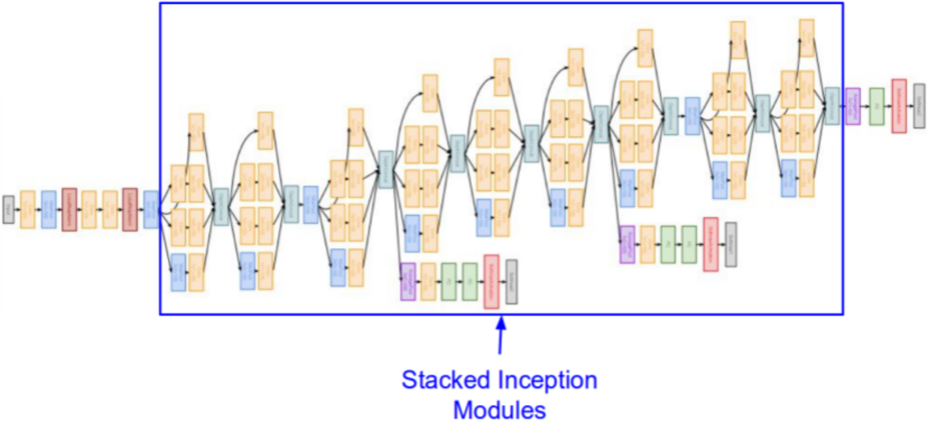
Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

Full Architecture:



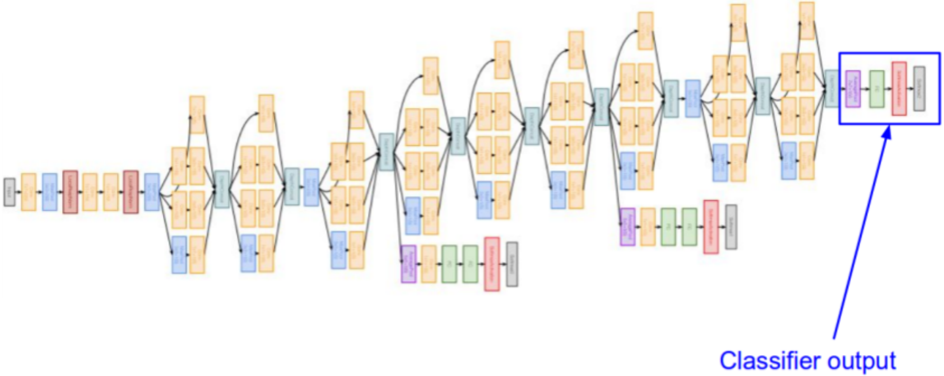
InceptionNet (GoogleNet)

Full Architecture:



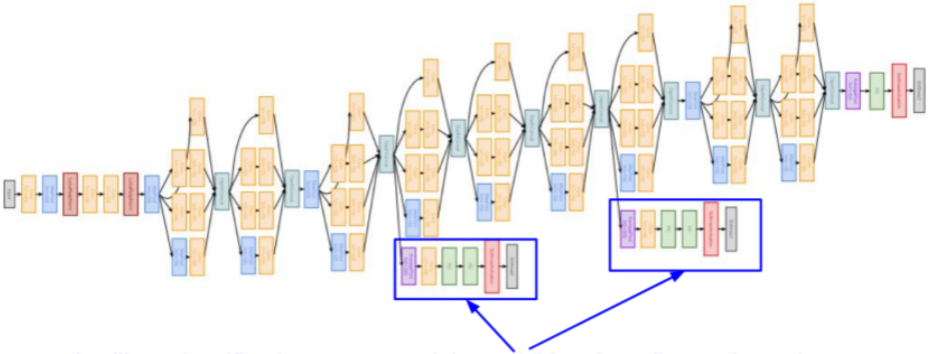
InceptionNet (GoogleNet)

Full Architecture:



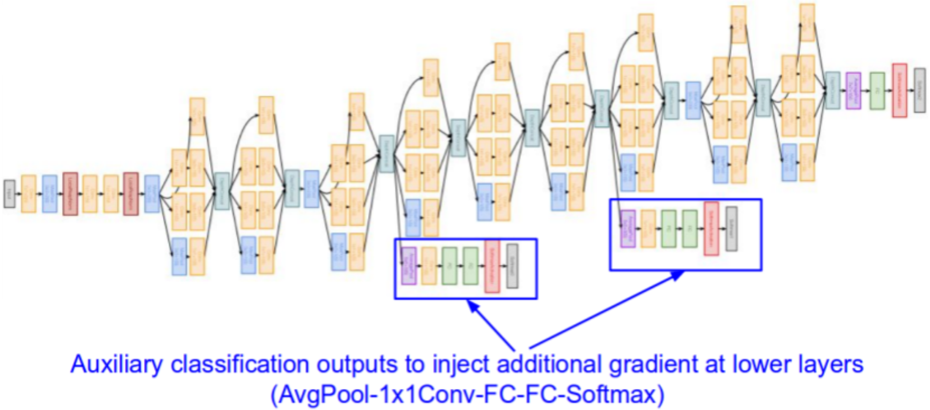
InceptionNet (GoogleNet)

Full Architecture:



Auxiliary classification outputs to inject additional gradient at lower layers
(AvgPool-1x1Conv-FC-FC-Softmax)

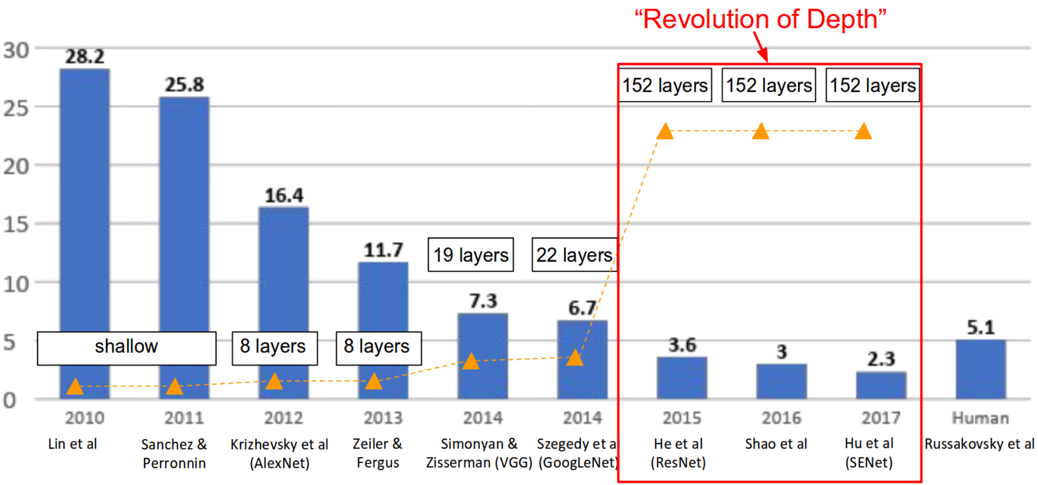
Full Architecture:



22 total layers (parallel layers count as 1 layer. Auxiliary output layers not counted)

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

Deeper the Merrier



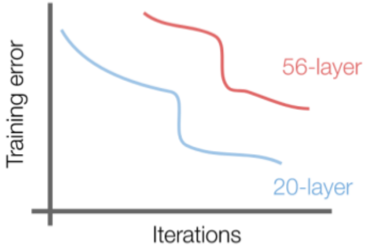
Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

How deep can we go?

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

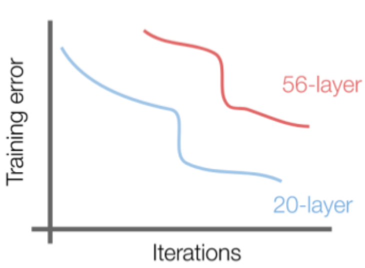
How deep can we go?

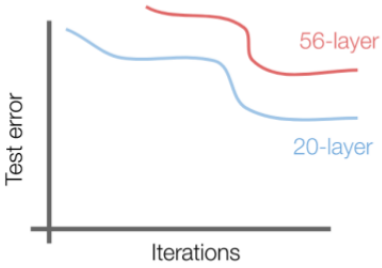
What happens when we continue stacking deeper layers on a “plain” convolutional neural network?



How deep can we go?

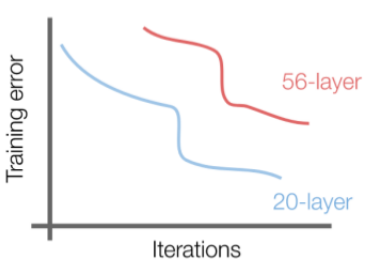
What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

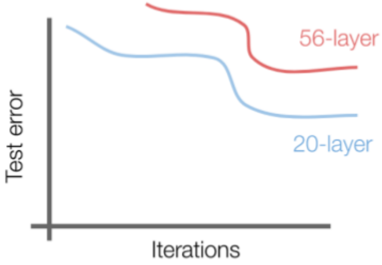




How deep can we go?

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

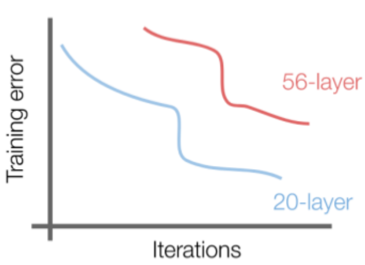


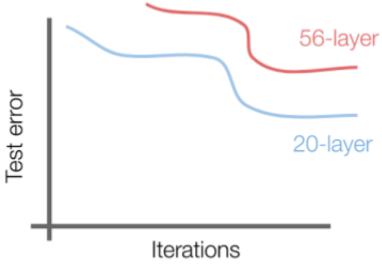


Deeper model does worse than shallow model!

How deep can we go?

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





Deeper model does worse than shallow model! **Why?**

How deep can we go?

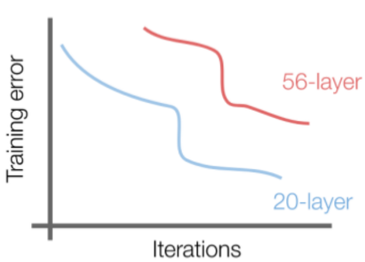
What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

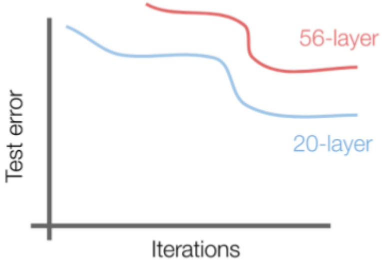
Deeper model does worse than shallow model! **Why?**

The initial guess is that the deep model is **overfitting** since it is much bigger than the shallow model

How deep can we go?

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





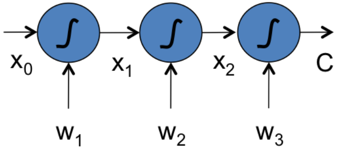
Deeper model does worse than shallow model! **Why?**

The deep model is actually **underfitting** since it also performs worse than the shallow model on the training set

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

How deep can we go? Vanishing/Exploding Gradient

Consider a simple network:

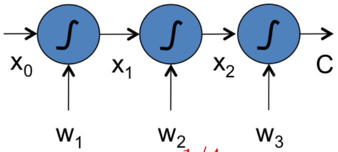


$$\frac{\partial L}{\partial x_0} = \overset{<1/4}{\boxed{\sigma'(w_3^T x_2)}} \times w_3 \times \overset{<1/4}{\boxed{\sigma'(w_2^T x_1)}} \times w_2 \times \overset{<1/4}{\boxed{\sigma'(w_1^T x_0)}} \times w_1$$

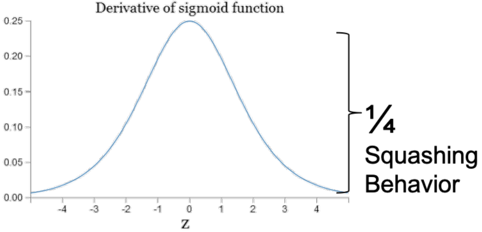
Why?

How deep can we go? Vanishing/Exploding Gradient

Consider a simple network:

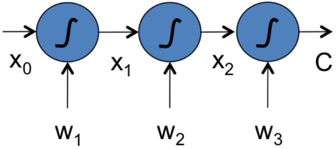


$$\frac{\partial L}{\partial x_0} = \underbrace{\sigma'(w_3^T x_2)}_{<1/4} \times w_3 \times \underbrace{\sigma'(w_2^T x_1)}_{<1/4} \times w_2 \times \underbrace{\sigma'(w_1^T x_0)}_{<1/4} \times w_1$$



How deep can we go? Vanishing/Exploding Gradient

Consider a simple network:



$$\frac{\partial L}{\partial x_0} = \overset{<1/4}{\boxed{\sigma'(w_3^T x_2)}} \times w_3 \times \overset{<1/4}{\boxed{\sigma'(w_2^T x_1)}} \times w_2 \times \overset{<1/4}{\boxed{\sigma'(w_1^T x_0)}} \times w_1$$

- **Vanishing gradients:** Deeper the network, gradients vanish quickly, thereby slowing the rate of change in initial layers
- **Exploding gradients:** Happen when the individual layer gradients are much higher than 1, for instance - can be overcome by **gradient clipping**

ResNet

The deeper model should be able to perform at least as well as the shallower model;how?

ResNet

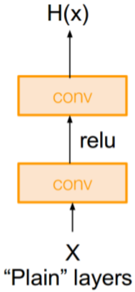
The deeper model should be able to perform at least as well as the shallower model;how?

Solution: Change the network with identity connections between layers:

ResNet

The deeper model should be able to perform at least as well as the shallower model;how?

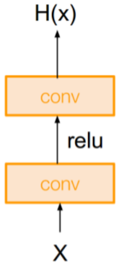
Solution: Change the network with identity connections between layers:



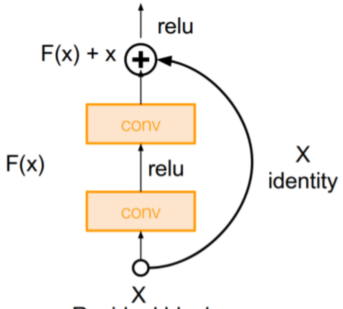
ResNet

The deeper model should be able to perform at least as well as the shallower model;how?

Solution: Change the network with identity connections between layers:



"Plain" layers



Residual block

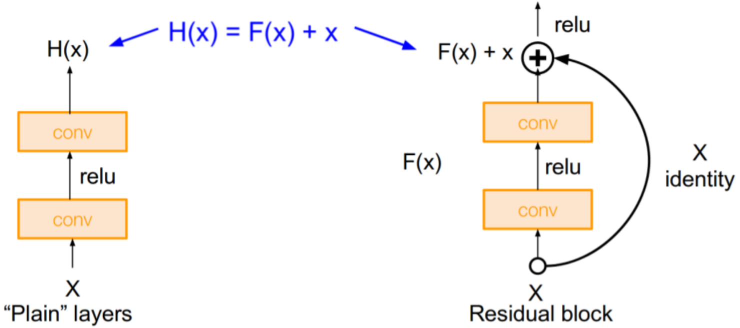
Use network layers to fit residual F(x) = H(x) - x instead of H(x) directly

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

ResNet

The deeper model should be able to perform at least as well as the shallower model;how?

Solution: Change the network with identity connections between layers:

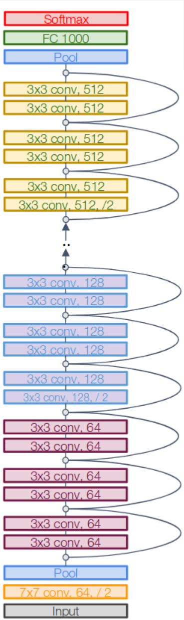


Use network layers to fit residual F(x) = H(x) - x instead of H(x) directly

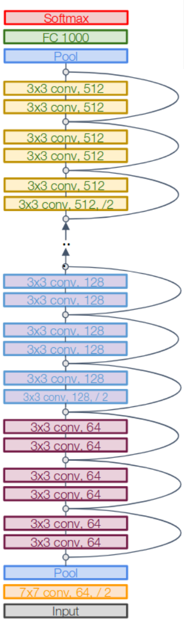
Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

ResNet

- A residual network is a stack of many residual blocks
- Each residual block has two 3×3 conv layers

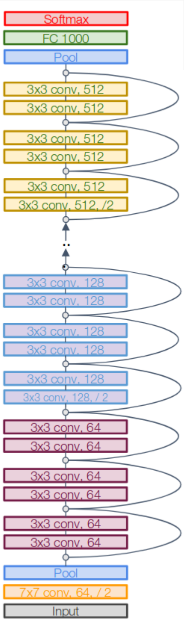


ResNet



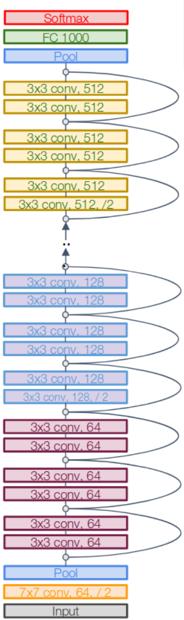
- A residual network is a stack of many residual blocks
- Each residual block has two 3×3 conv layers
- Periodically, double number of filters and downsample spatially using stride 2 ($/2$ in each dimension)

ResNet



- A residual network is a stack of many residual blocks
- Each residual block has two 3×3 conv layers
- Periodically, double number of filters and downsample spatially using stride 2 (/2 in each dimension)
- Use **global average pooling** and a single linear layer at the end (FC 1000 to output classes)

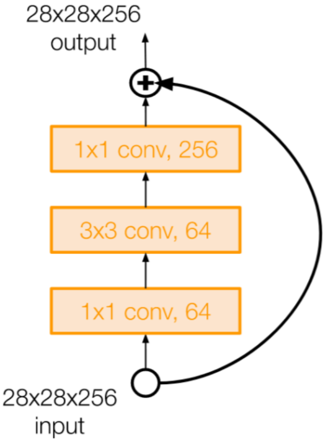
ResNet



- A residual network is a stack of many residual blocks
- Each residual block has two 3×3 conv layers
- Periodically, double number of filters and downsample spatially using stride 2 ($/2$ in each dimension)
- Use **global average pooling** and a single linear layer at the end (FC 1000 to output classes)
- Total depths of 34, 50, 101, or 152 layers for ImageNet dataset

ResNet

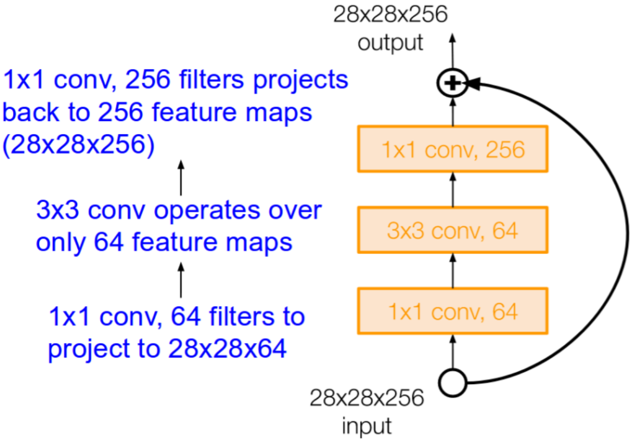
For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

ResNet

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



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

ResNet

- Able to train very deep networks
- Deeper networks performs better than shallow networks now (as expected); residual blocks help avoid vanishing gradient
- 1st place in all ILSVRC and COCO 2015 competitions
- We will discuss detection, localization, segmentation and the COCO dataset a bit later

ResNet

- Able to train very deep networks
- Deeper networks performs better than shallow networks now (as expected); residual blocks help avoid vanishing gradient
- 1st place in all ILSVRC and COCO 2015 competitions
- We will discuss detection, localization, segmentation and the COCO dataset a bit later

ResNet

- Able to train very deep networks
- Deeper networks performs better than shallow networks now (as expected); residual blocks help avoid vanishing gradient
- 1st place in all ILSVRC and COCO 2015 competitions
- We will discuss detection, localization, segmentation and the COCO dataset a bit later

ResNet

- Able to train very deep networks
- Deeper networks performs better than shallow networks now (as expected); residual blocks help avoid vanishing gradient
- 1st place in all ILSVRC and COCO 2015 competitions
- We will discuss detection, localization, segmentation and the COCO dataset a bit later

ResNet

- Able to train very deep networks
- Deeper networks performs better than shallow networks now (as expected); residual blocks help avoid vanishing gradient
- 1st place in all ILSVRC and COCO 2015 competitions
- We will discuss detection, localization, segmentation and the COCO dataset a bit later

ResNet @ ILSVRC & COCO 2015 Competitions

1st place in all five major challenges

- ImageNet Classification: "Ultra-deep" 152-layer nets
- ImageNet Detection: 16% better than the 2nd best
- ImageNet Localization: 27% better than the 2nd best
- COCO Detection: 11% better than the 2nd best
- COCO Segmentation: 12% better than the 2nd best

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

Homework

Readings

- Tutorial: Illustrated: 10 CNN Architectures
  - Read 4-6: Inception-v1, Inception-v3, ResNet-50
- (Optional) For more details, skim through the following papers:
  - ImageNet Classification with Deep Convolutional Neural Networks
  - Very Deep Convolutional Networks for Large-Scale Image Recognition
  - Going Deeper with Convolutions
  - Deep Residual Learning for Image Recognition

Exercise

- Show that minimizing negative log likelihood in a neural network with a softmax activation function in the last layer is equivalent to minimizing cross-entropy error function (*Hint:* Read Chapter 3 of Nielsen's online book on basics of NNs)

References

[1] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: 1998.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*. 2012.

[3] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2015).

[4] Christian Szegedy et al. "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 1–9.

[5] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

[6] *Johnson, Justin, EECS 498-007 / 598-005 - Deep Learning for Computer Vision (Fall 2019)*. URL: https://web.eecs.umich.edu/~justincj/teaching/eecs498/ (visited on 06/29/2020).

[7] *Li, Fei-Fei; Johnson, Justin; Serena, Yeung; CS 231n - Convolutional Neural Networks for Visual Recognition (Spring 2019)*. URL: http://cs231n.stanford.edu/2019/ (visited on 06/29/2020).