Convolutional Neural Net Architectures

Materials from

- Intel Deep Learning https://www.intel.com/content/www/us/en/developer/learn/course-deep-learning.html
- Introduction to Neural Networks https://www.deeplearning.ai

Legal Notices and Disclaimers

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

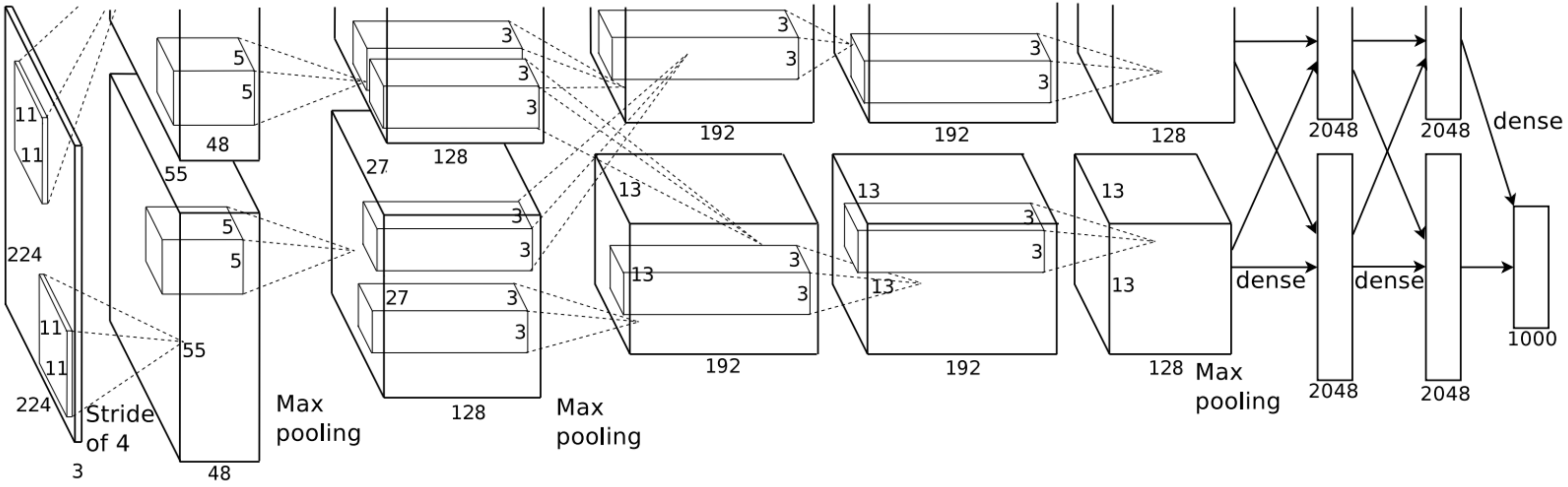
Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

This sample source code is released under the Intel Sample Source Code License Agreement.

AlexNet

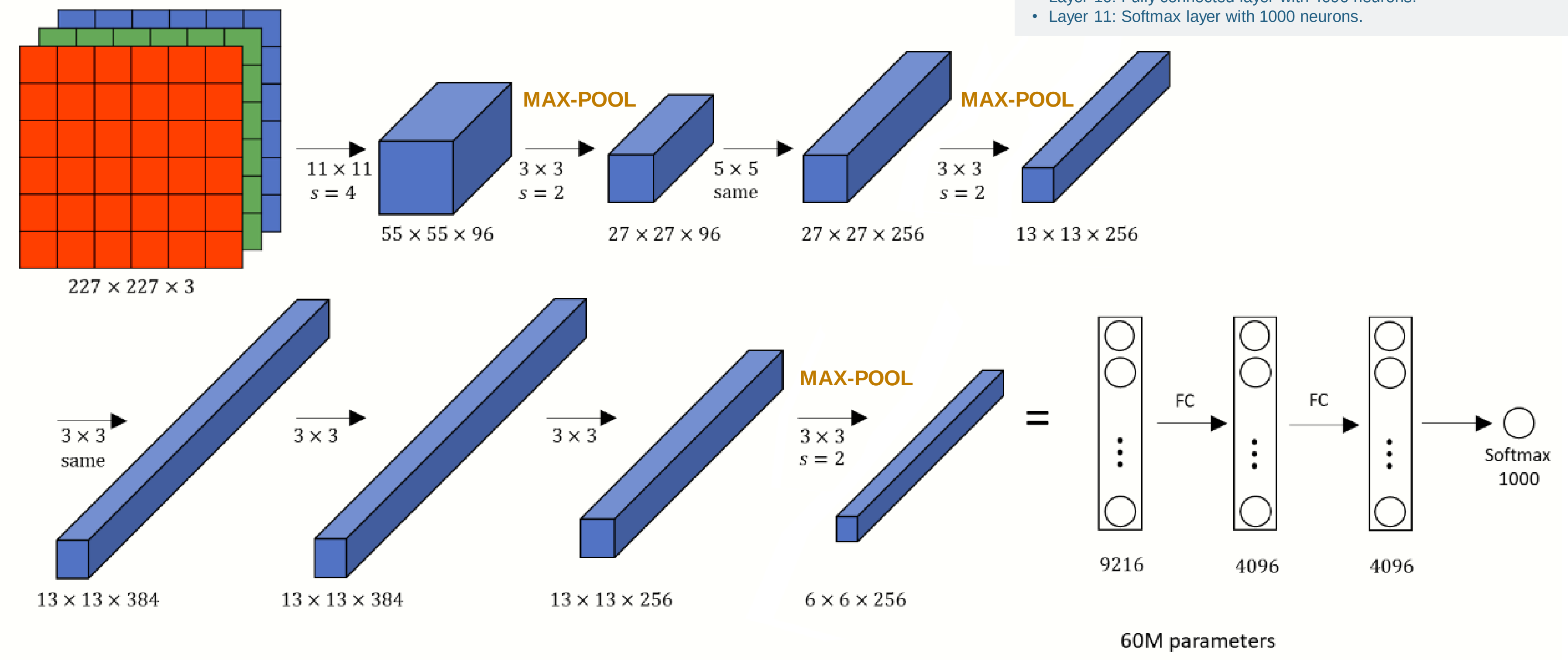
- Created in 2012 for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- Task: predict the correct label from among 1000 classes
- Dataset: around 1.2 million images
- Considered the “flash point” for modern deep learning
- Demolished the competition.
- Top 5 error rate of 15.3%
- Next best: 26.2%

AlexNet - Model Diagram



AlexNet - Model Diagram

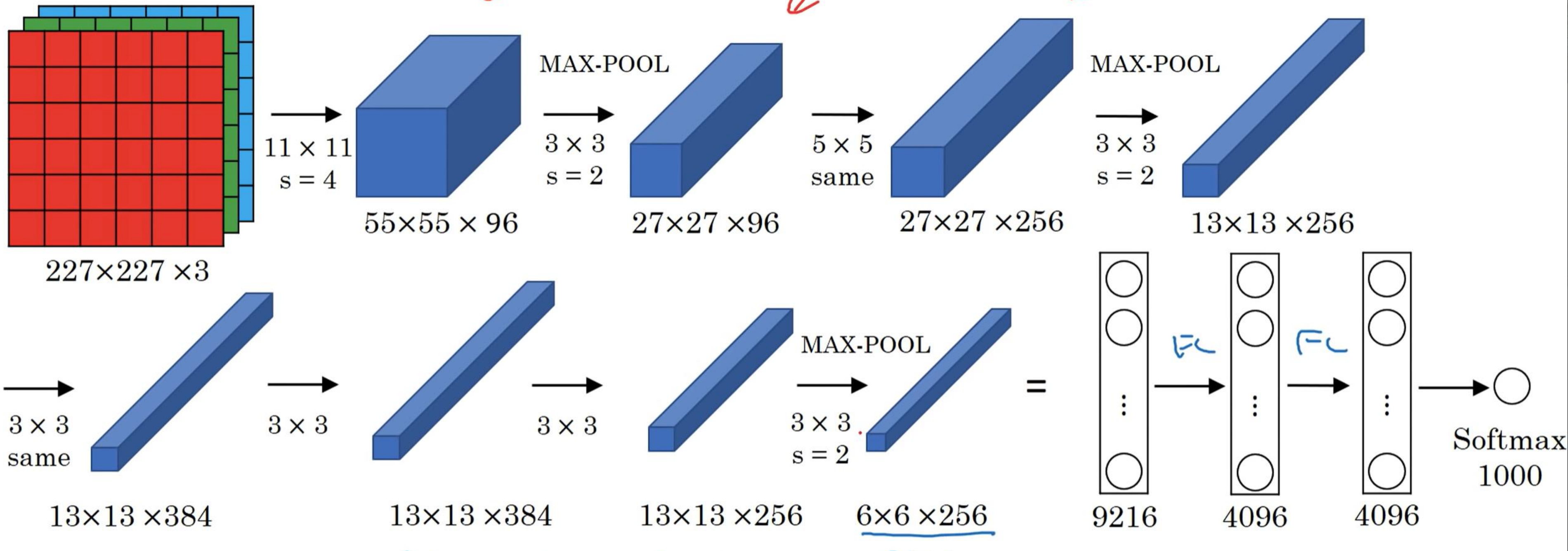
- Layer 1: Convolutional layer with 96 filters of size 11x11 and a stride of 4.
- Layer 2: Max-pooling layer with a pooling size of 3x3 and a stride of 2.
- Layer 3: Convolutional layer with 256 filters of size 5x5 and a stride of 1.
- Layer 4: Max-pooling layer with a pooling size of 3x3 and a stride of 2.
- Layer 5: Convolutional layer with 384 filters of size 3x3 and a stride of 1.
- Layer 6: Convolutional layer with 384 filters of size 3x3 and a stride of 1.
- Layer 7: Convolutional layer with 256 filters of size 3x3 and a stride of 1.
- Layer 8: Max-pooling layer with a pooling size of 3x3 and a stride of 2.
- Layer 9: Fully connected layer with 4096 neurons.
- Layer 10: Fully connected layer with 4096 neurons.
- Layer 11: Softmax layer with 1000 neurons.



[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Andrew Ng

AlexNet



- Similar to LeNet, but much bigger. ~60M parameters
- ReLU
- Multiple GPUs.

[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Andrew Ng

AlexNet - Details

- They performed *data augmentation* for training
  - Includes Cropping, horizontal flipping, and other manipulations

- Basic Template:
  - Convolutions with ReLUs
  - Sometimes add maxpool after convolutional layer
  - Fully connected layers at the end before a softmax classifier

VGG

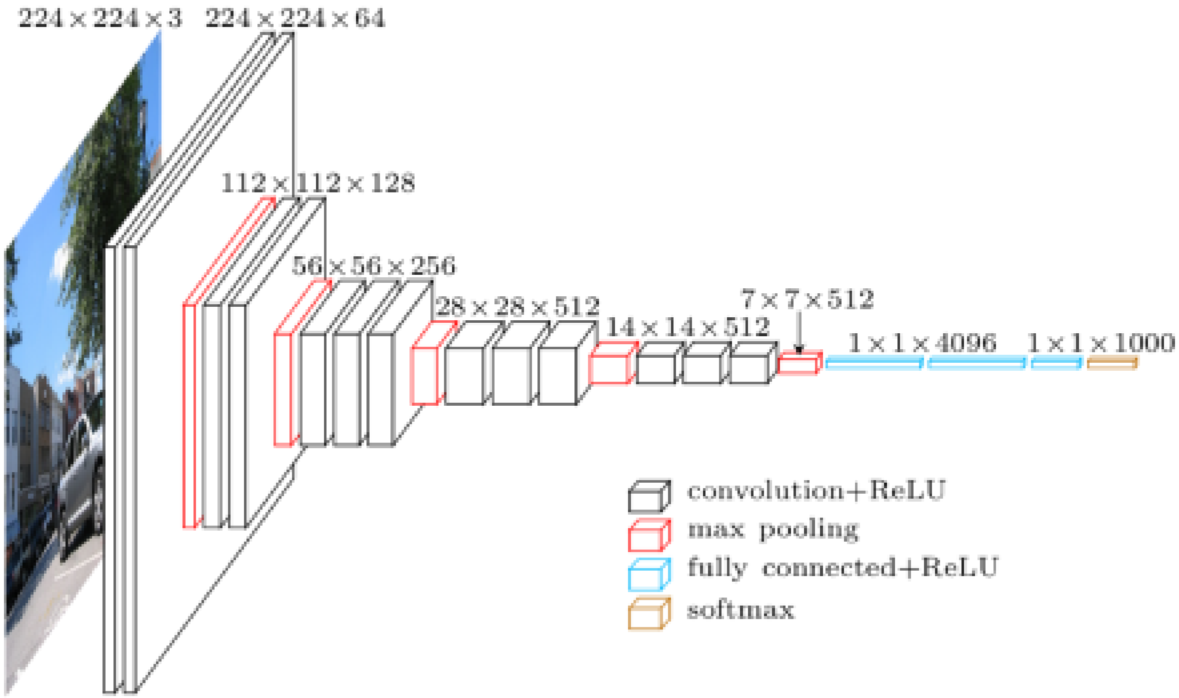
- Simplify Network Structure
- Achieved at error rate of 7.32%, Top 5 Recall in 2014
- Avoid Manual Choices of Convolution Size
- Very Deep Network with 3x3 Convolutions
- These “effectively” give rise to larger convolutions

Reference:

Very Deep Convolutional Networks for Large-Scale Image Recognition

Karen Simonyan and Andrew Zisserman, 2014

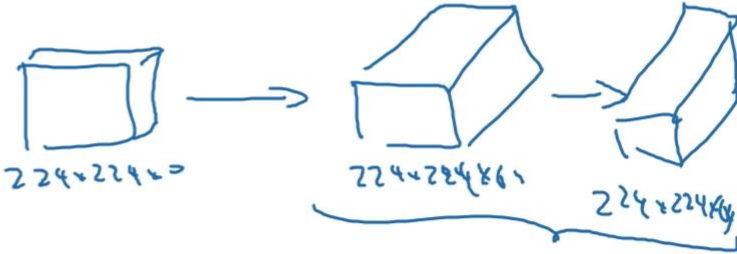
VGG-16 Diagram

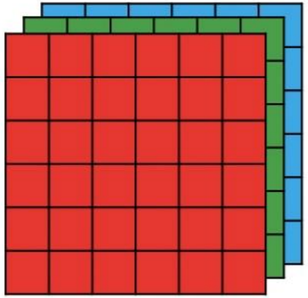


VGG - 16

VGG-19

CONV = 3×3 filter, s = 1, same   MAX-POOL = 2×2 , s = 2





224×224 ×3 $\xrightarrow[\times 2]{\text{[CONV 64]}}$ 224×224×64 $\xrightarrow{\text{POOL}}$ 112×112 ×64 $\xrightarrow[\times 2]{\text{[CONV 128]}}$ 112×112 ×128 $\xrightarrow{\text{POOL}}$ 56×56 ×128

$\xrightarrow[\times 3]{\text{[CONV 256]}}$ 56×56 ×256 $\xrightarrow{\text{POOL}}$ 28×28 ×256 $\xrightarrow[\times 3]{\text{[CONV 512]}}$ 28×28 ×512 $\xrightarrow{\text{POOL}}$ 14×14×512

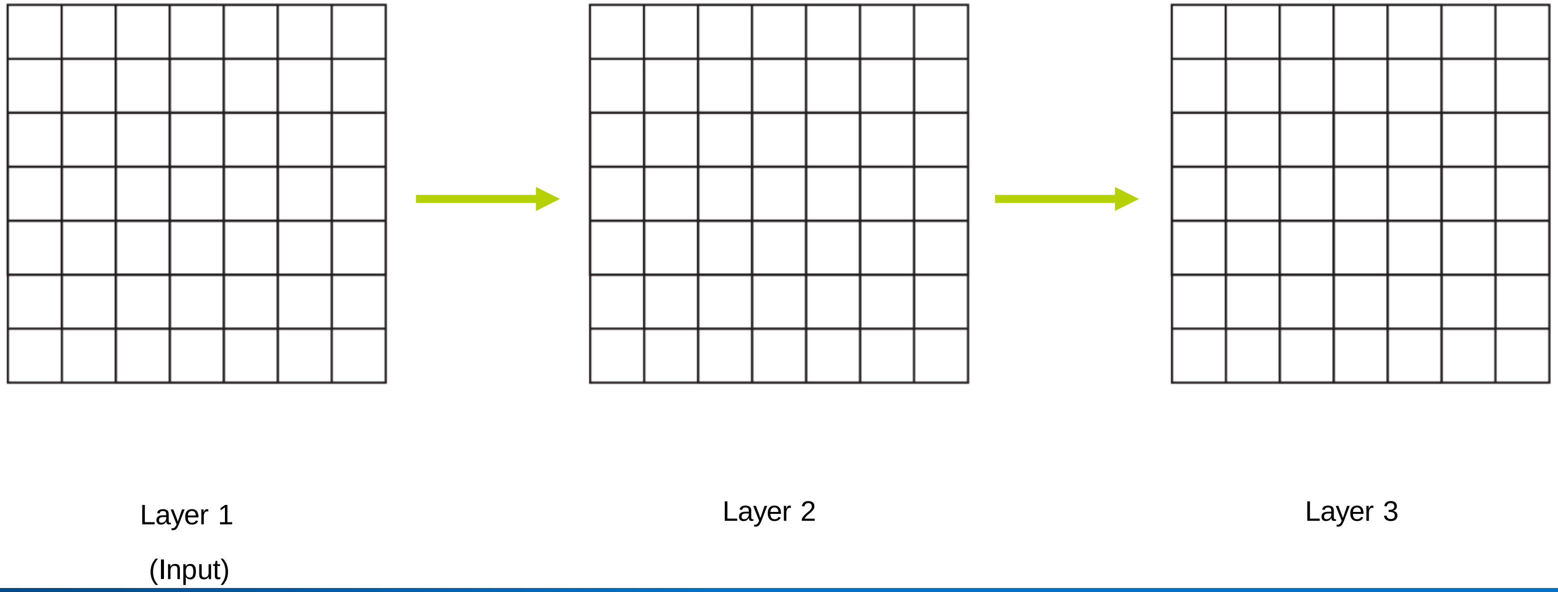
$\xrightarrow[\times 3]{\text{[CONV 512]}}$ 14×14 ×512 $\xrightarrow{\text{POOL}}$ 7×7×512 $\longrightarrow$ FC 4096 $\longrightarrow$ FC 4096 $\longrightarrow$ Softmax 1000

$n_H, n_W \downarrow$ $n_c \uparrow$ ~138M

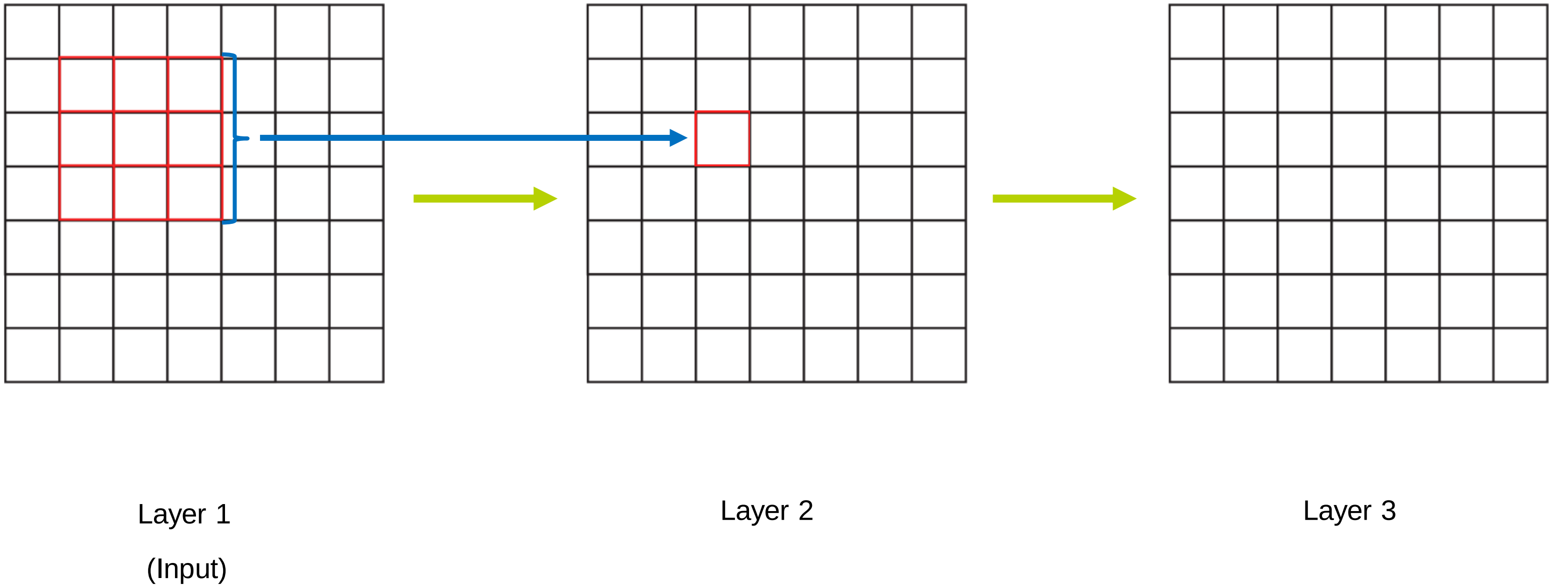
[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

Andrew Ng

VGG



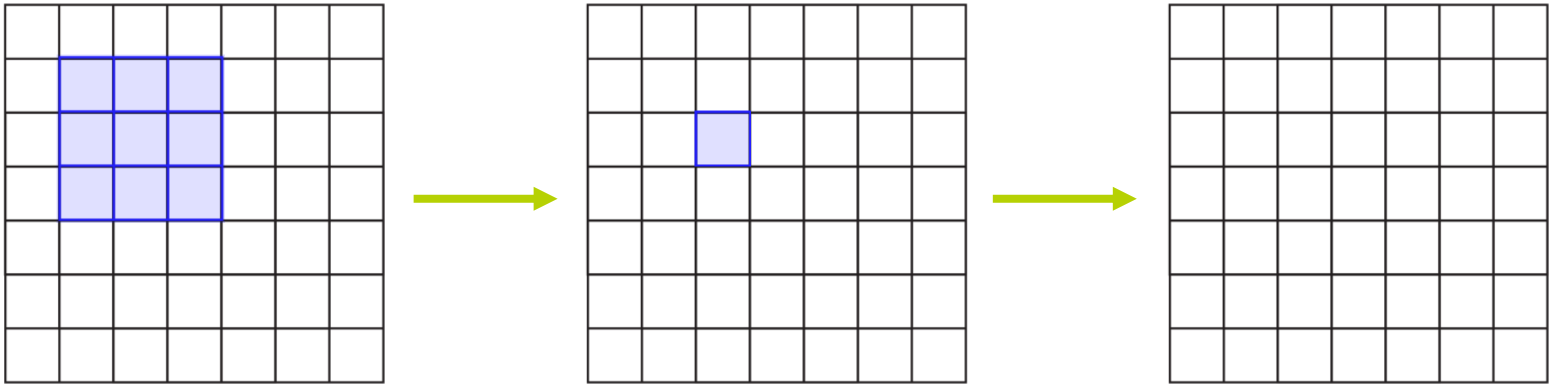
VGG



VGG

We can say that the “receptive field” of Layer 2 is 3x3

Each output has been influenced by a 3x3 patch of inputs



Layer 1

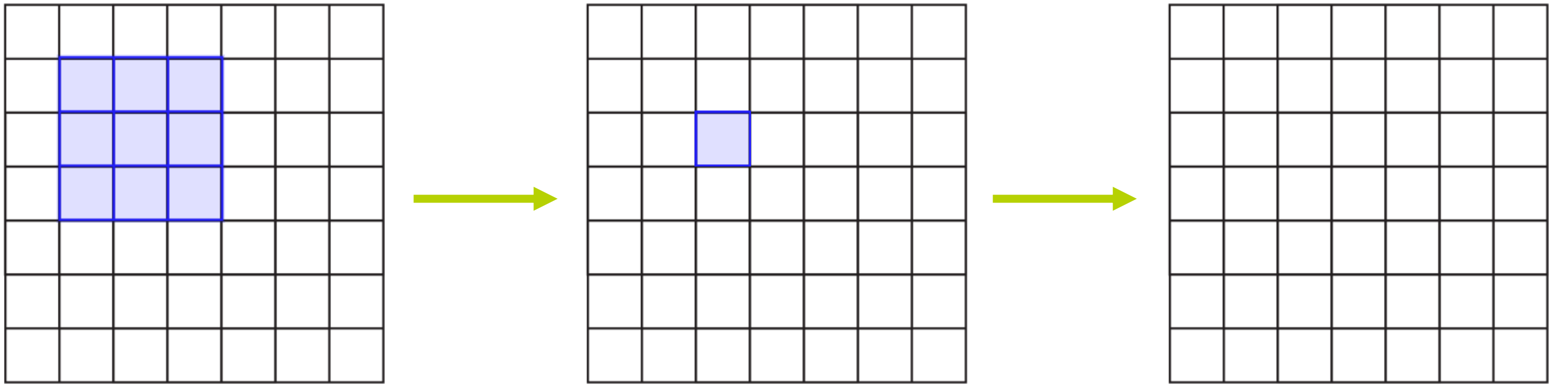
(Input)

Layer 2

Layer 3

VGG

What about on Layer 3?

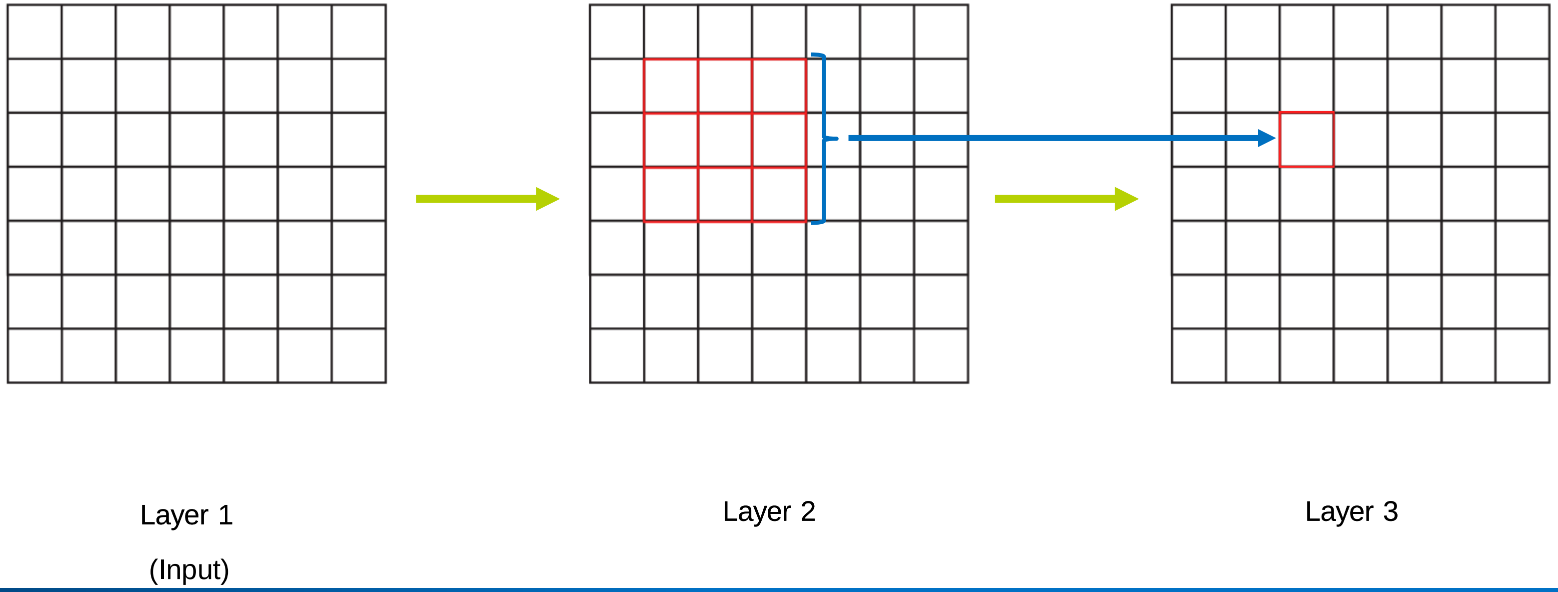


Layer 1 (Input) | Layer 2 | Layer 3

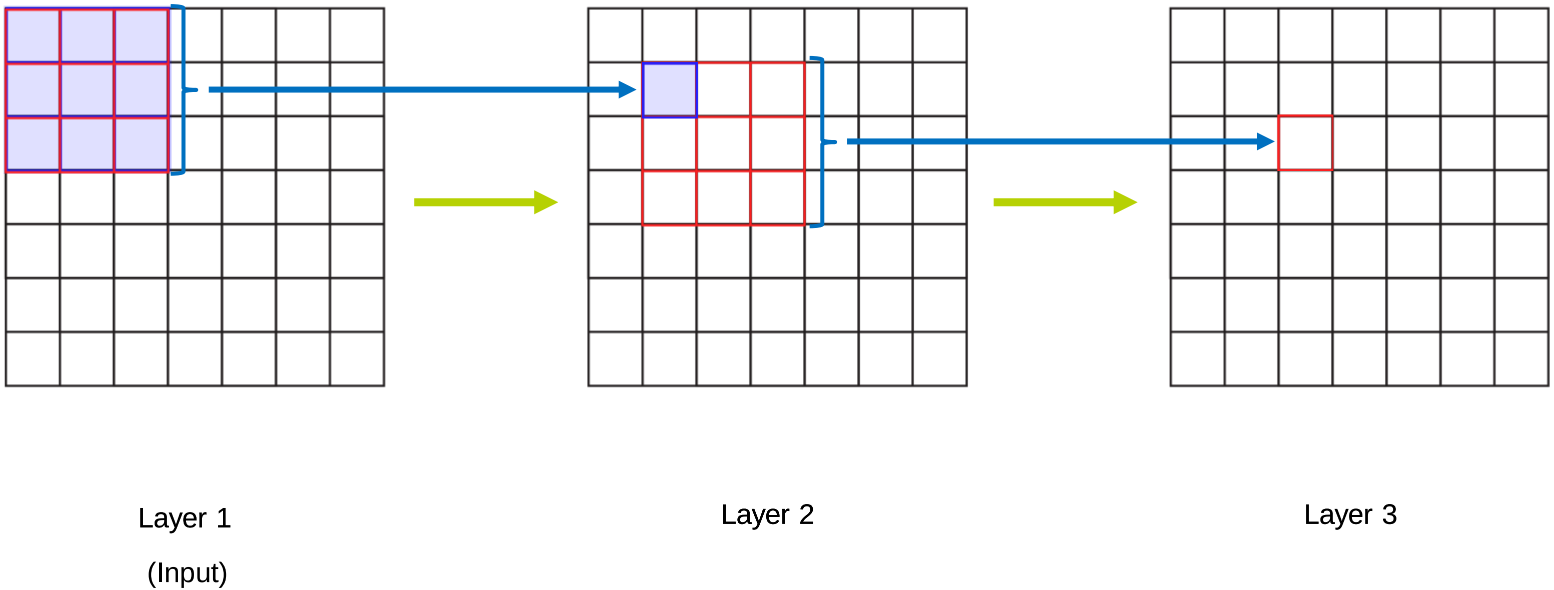
VGG

This output on Layer 3 uses a 3x3 patch from Layer 2

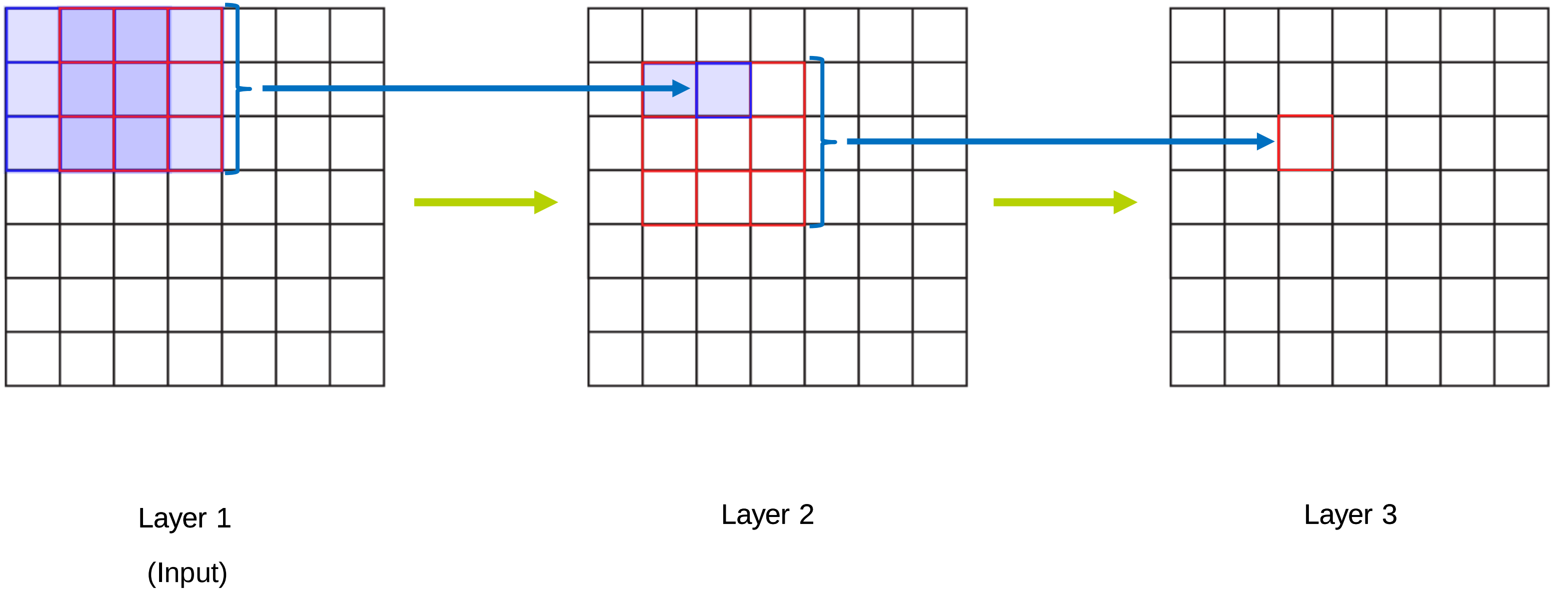
How much from Layer 1 does it use?



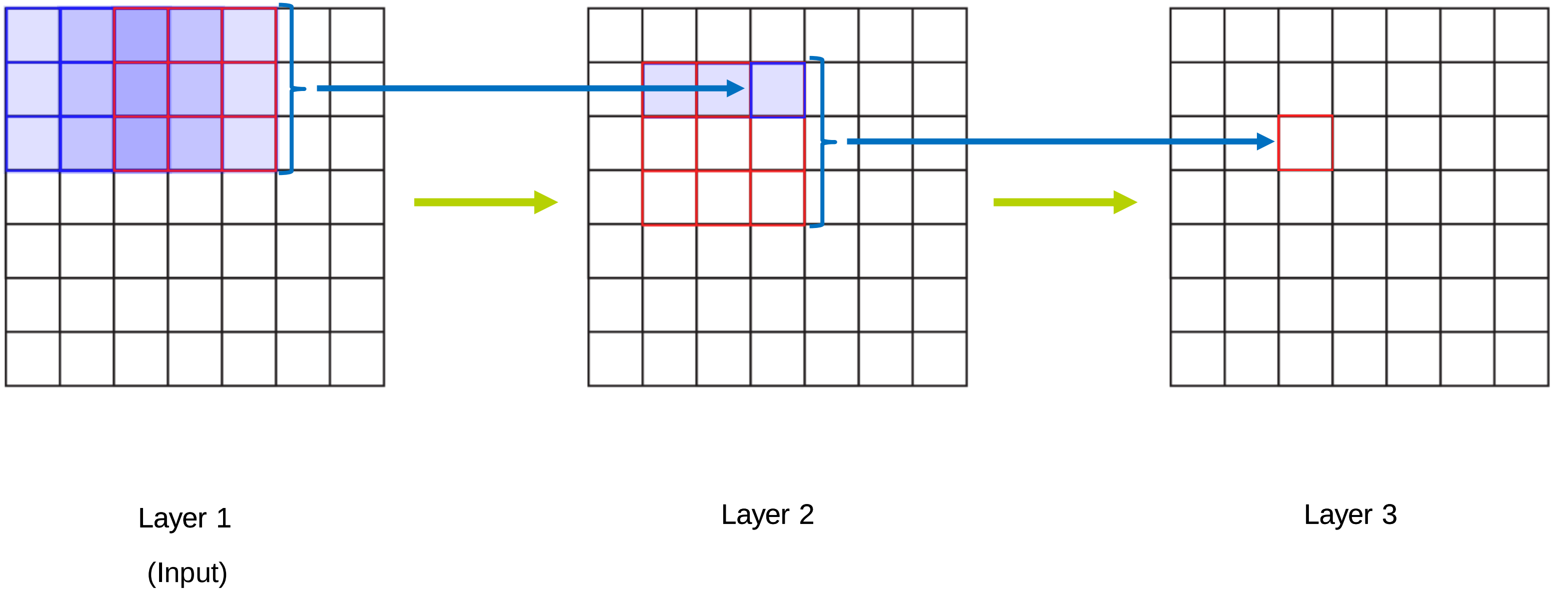
VGG



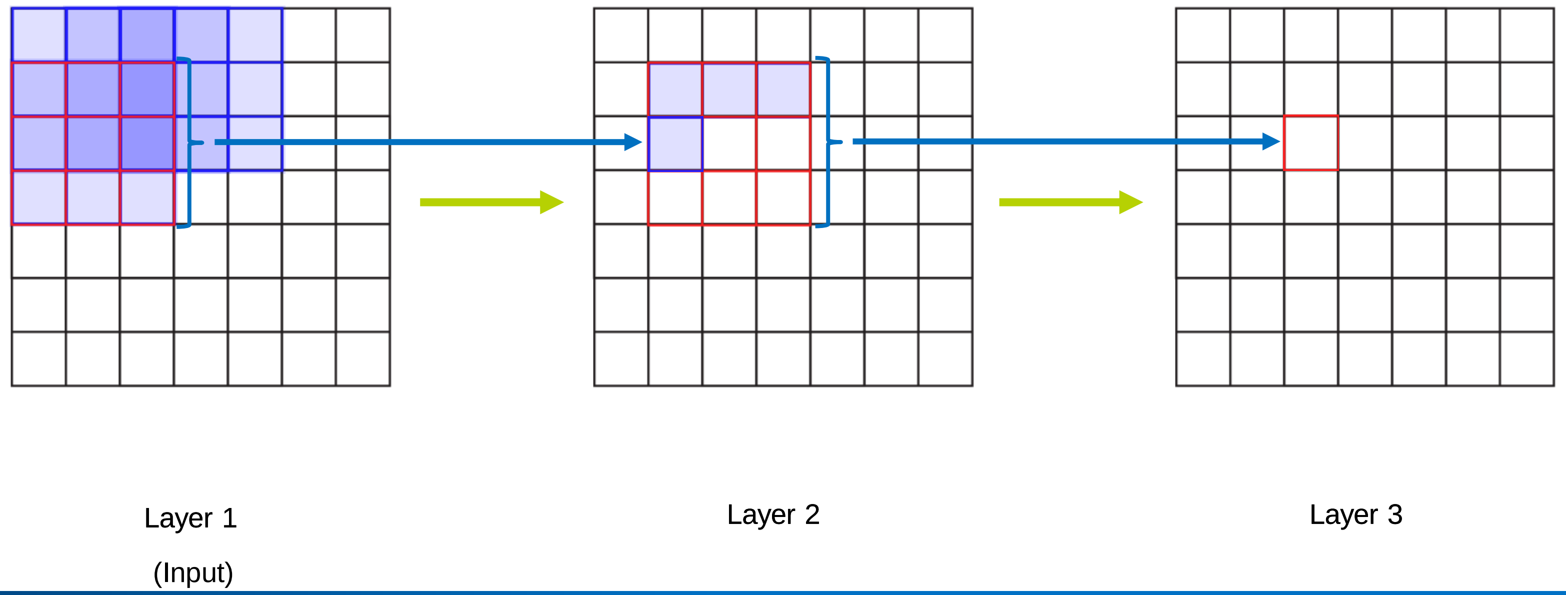
VGG



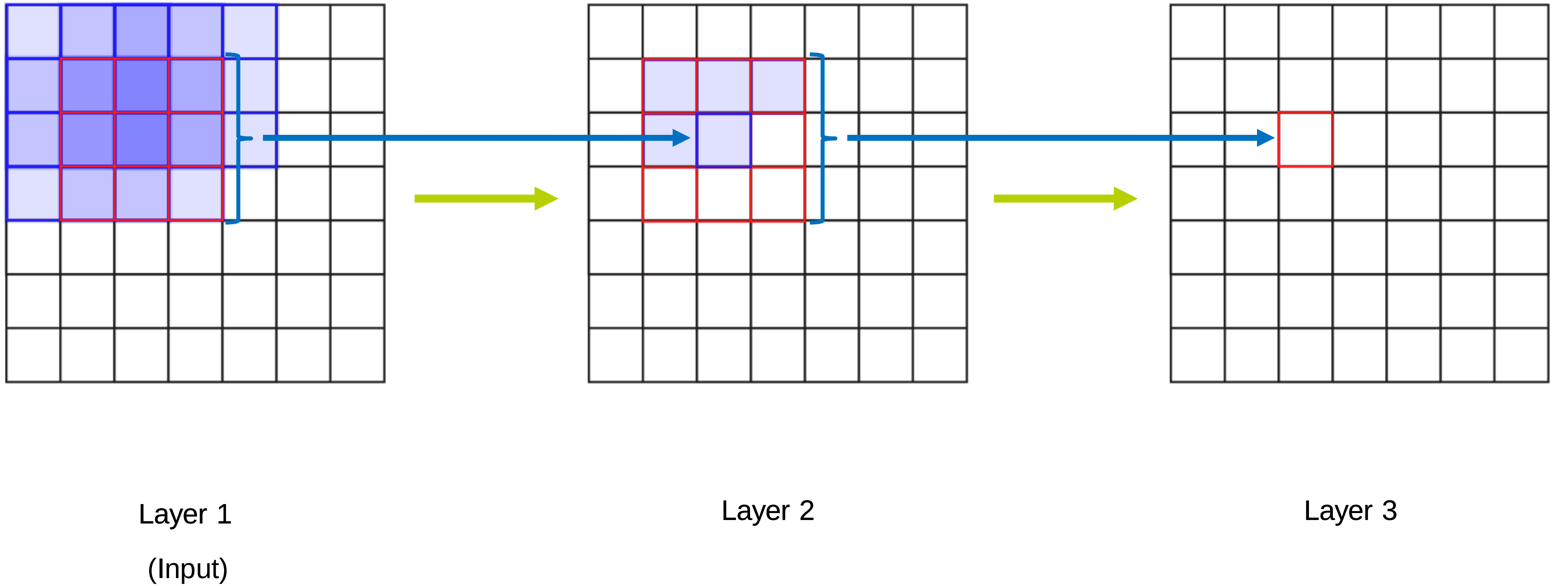
VGG



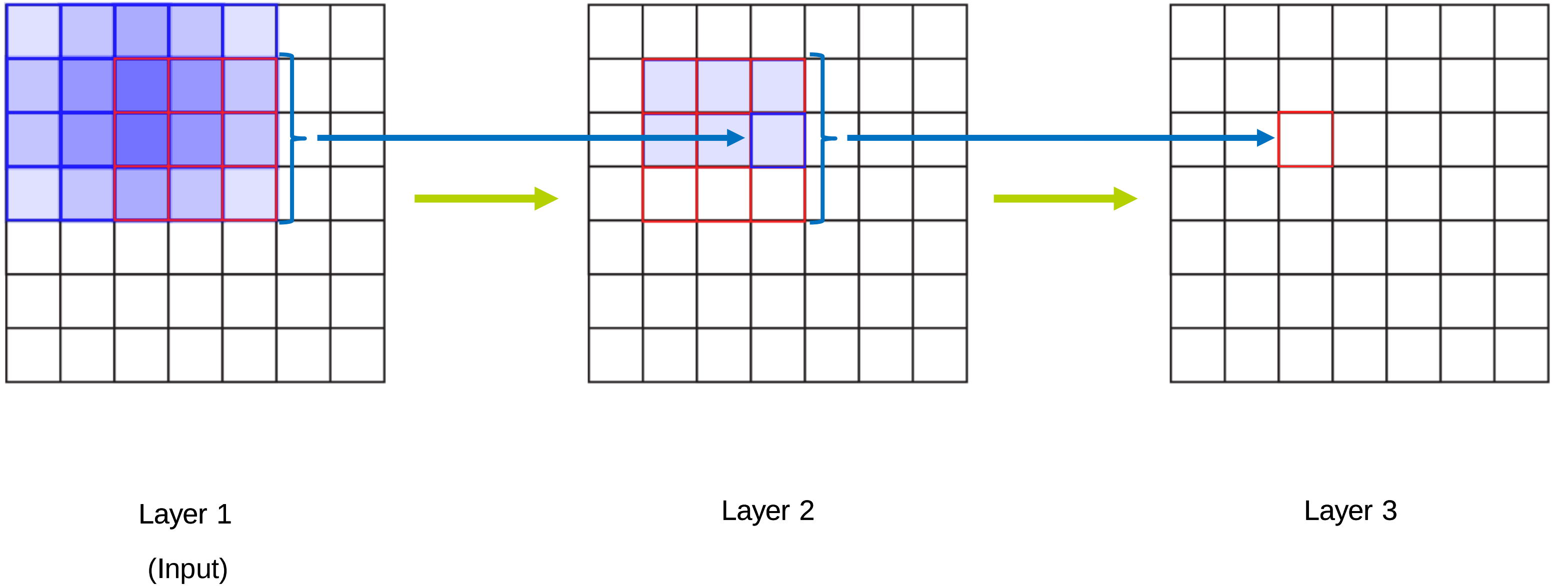
VGG



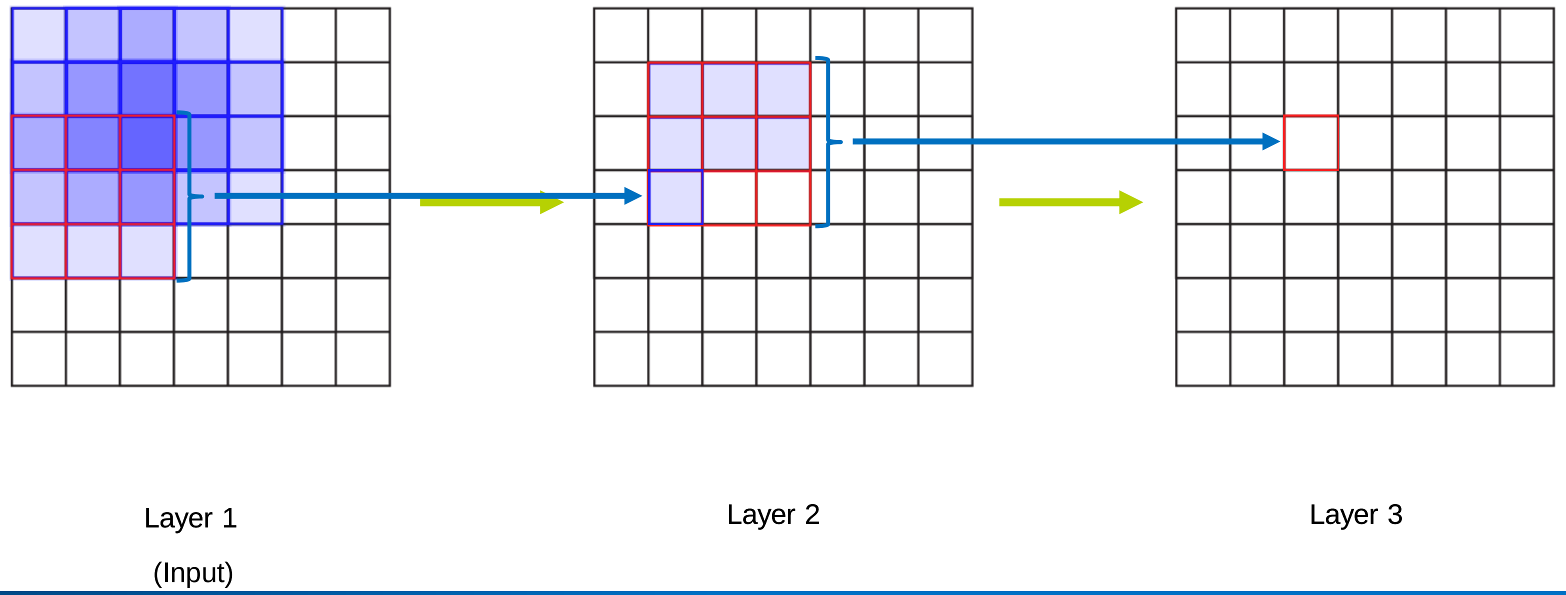
VGG



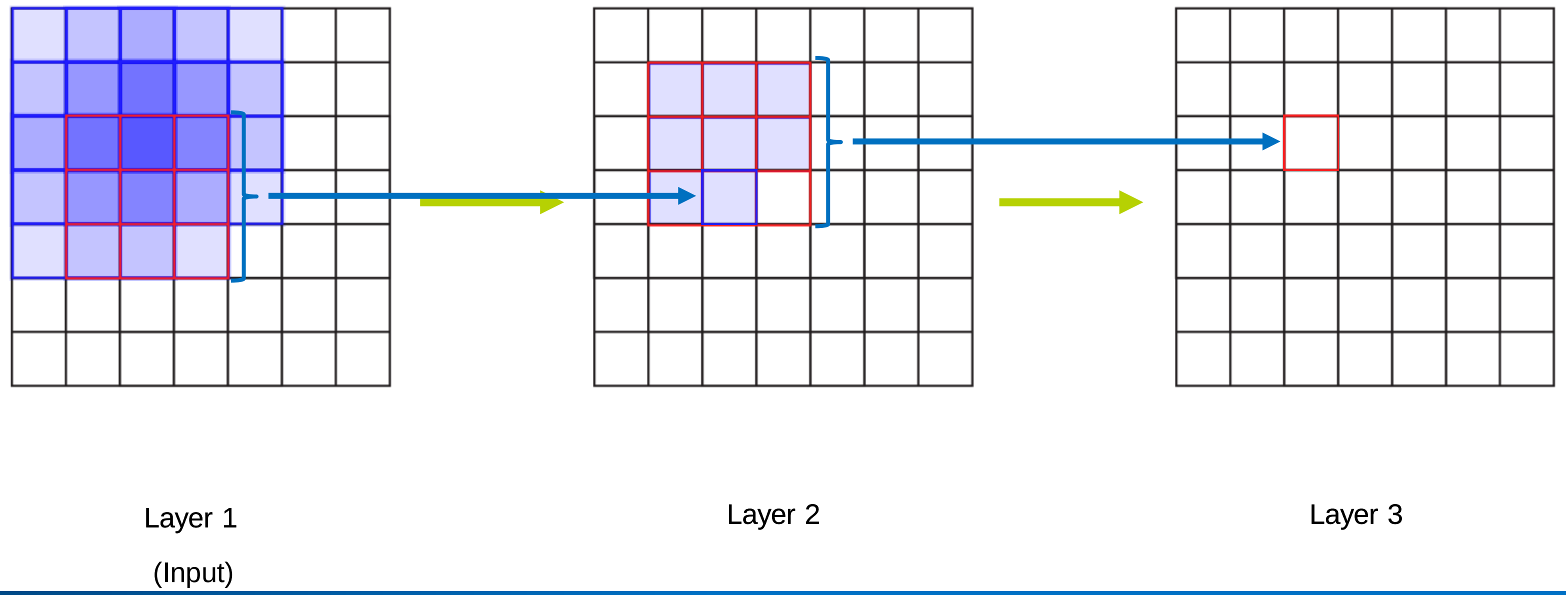
VGG



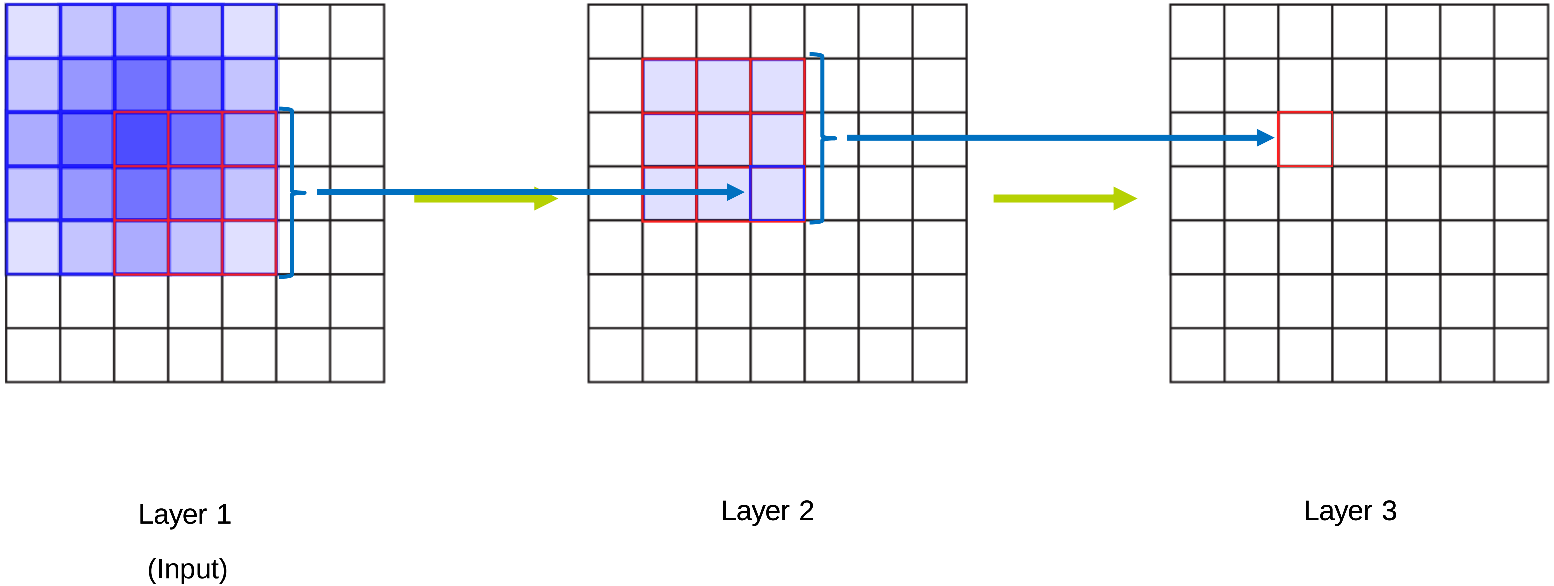
VGG



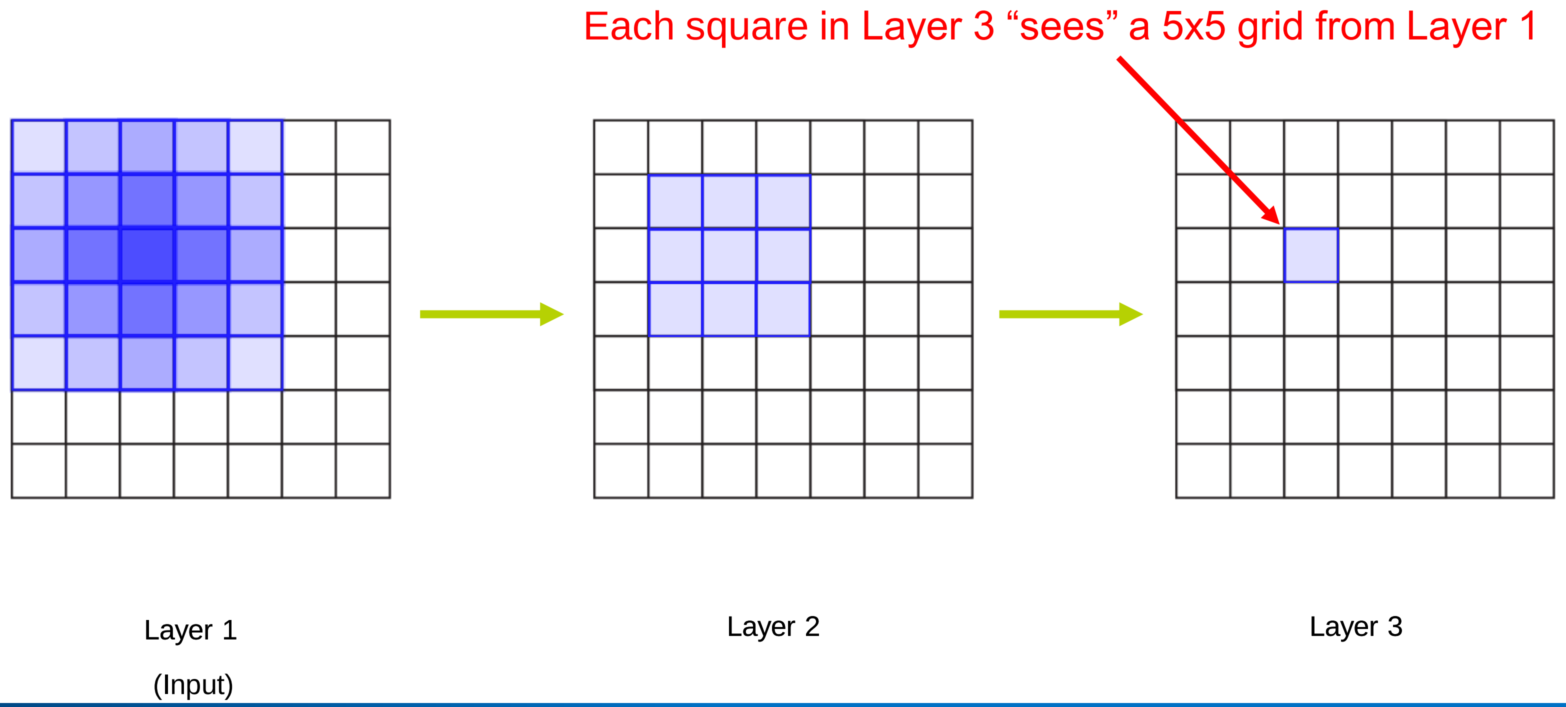
VGG



VGG



VGG


Each square in Layer 3 “sees” a 5x5 grid from Layer 1
Layer 1
(Input)
Layer 2
Layer 3

VGG

Two 3x3, stride 1 convolutions in a row → one 5x5

Three 3x3 convolutions → one 7x7 convolution

Benefit: fewer parameters

One 3x3 layer

$$3 \times 3 \times C = 9C$$

One 7x7 layer

$$7 \times 7 \times C = 49C$$

Three 3x3 layers

$$3 \times (9C) = 27C$$

$49C$ → $27C$ → ≈45% reduction!

VGG

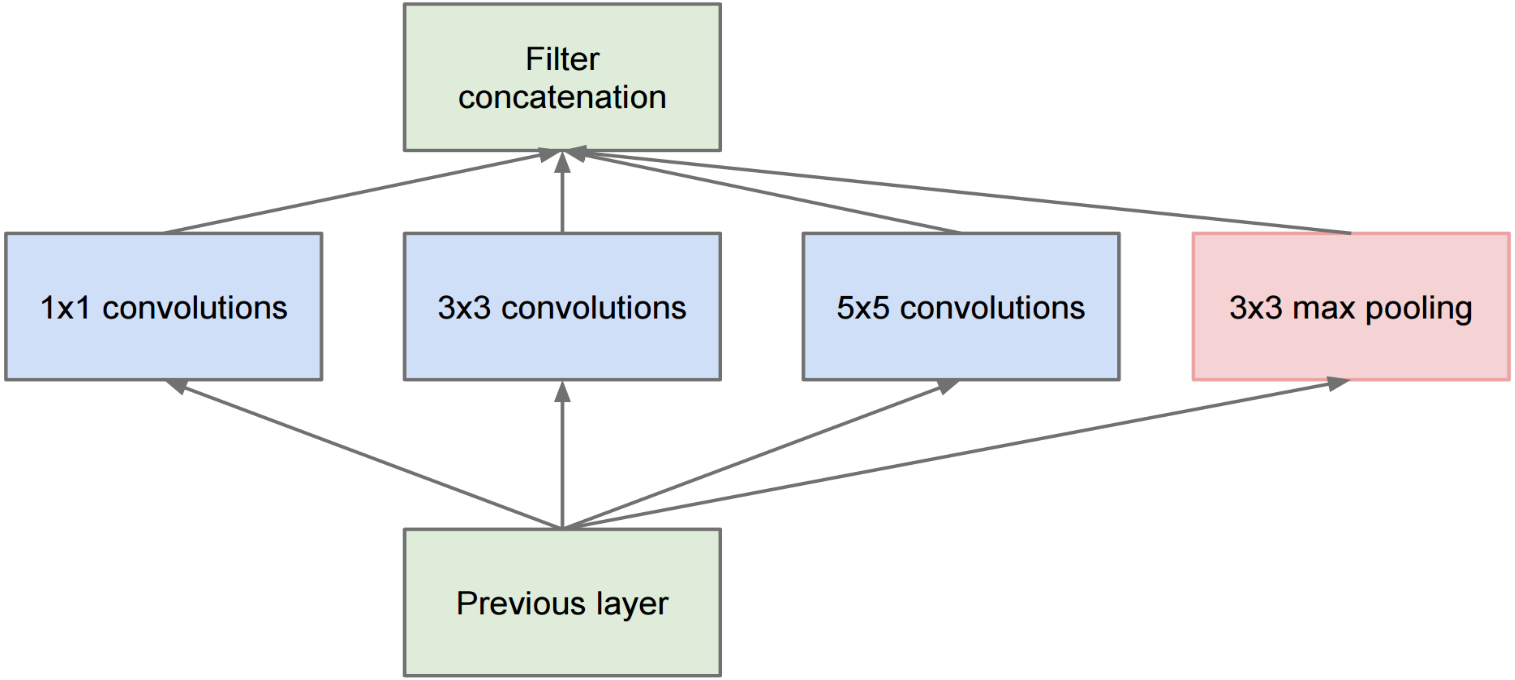
- One of the first architectures to experiment with many layers (More is better!)
- Can use multiple 3x3 convolutions to simulate larger kernels with fewer parameters
- VGGNet has a total of 138 million parameters, which is more than twice that of AlexNet
- Often used as a starting point for transfer learning because of its simple architecture
- Served as ”base model” for future works

Inception

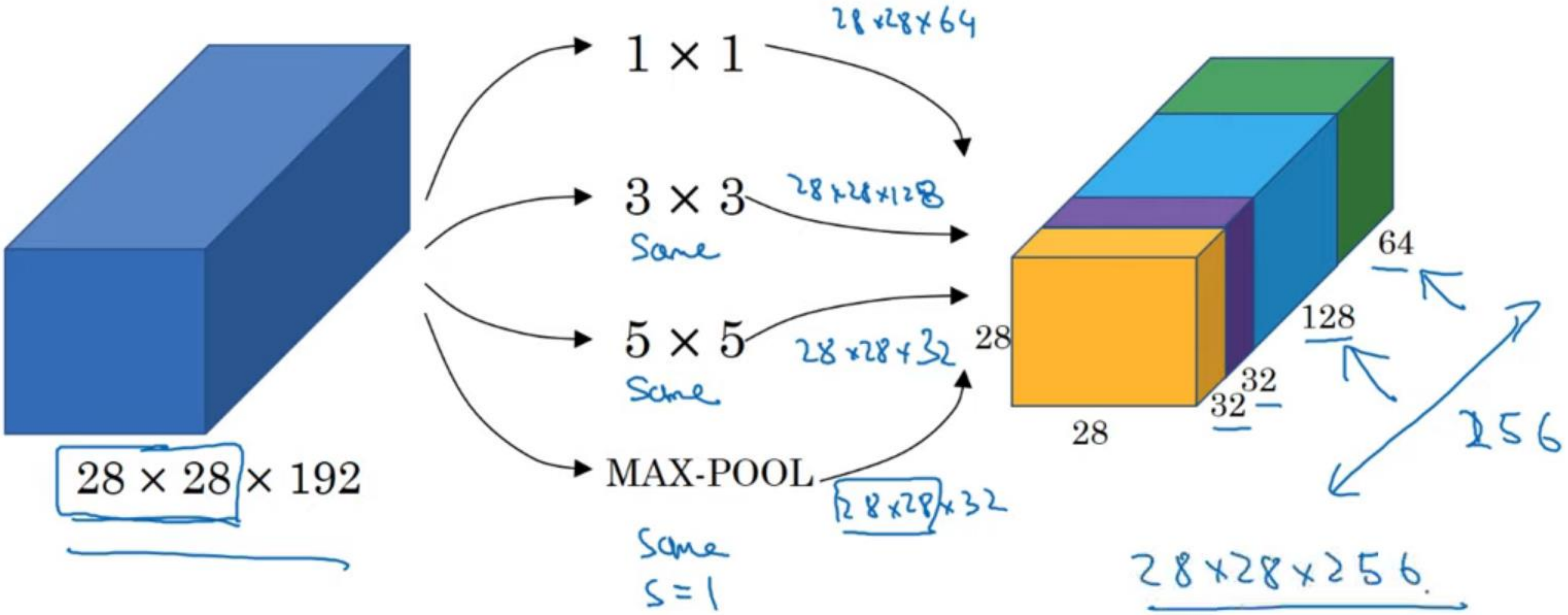
- Szegedy et al 2014
- Inception architecture-based GoogLeNet, was the winner of the 2014 ImageNet challenge, with a 6.7% top five error rate
- Idea: network would want to use different receptive fields
- Want computational efficiency
- Solution: Turn each layer into branches of convolutions
- Each branch handles smaller portion of workload
- Concatenate different branches at the end

Inception

Basic idea: replace single 3x3 convolutions with module

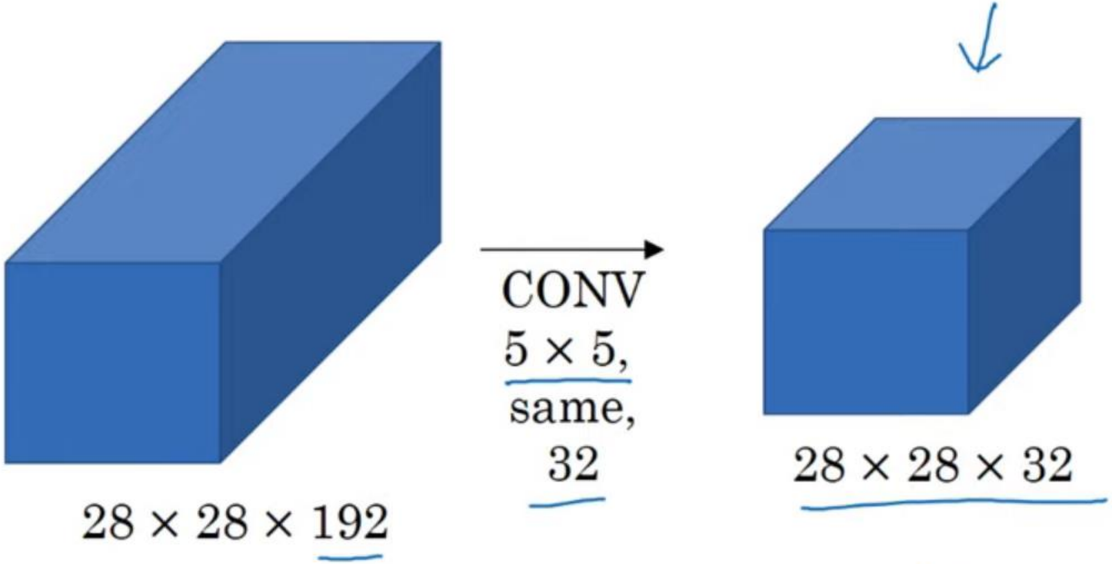


Motivation for inception network



[Szegedy et al. 2014. Going deeper with convolutions]

The problem of computational cost

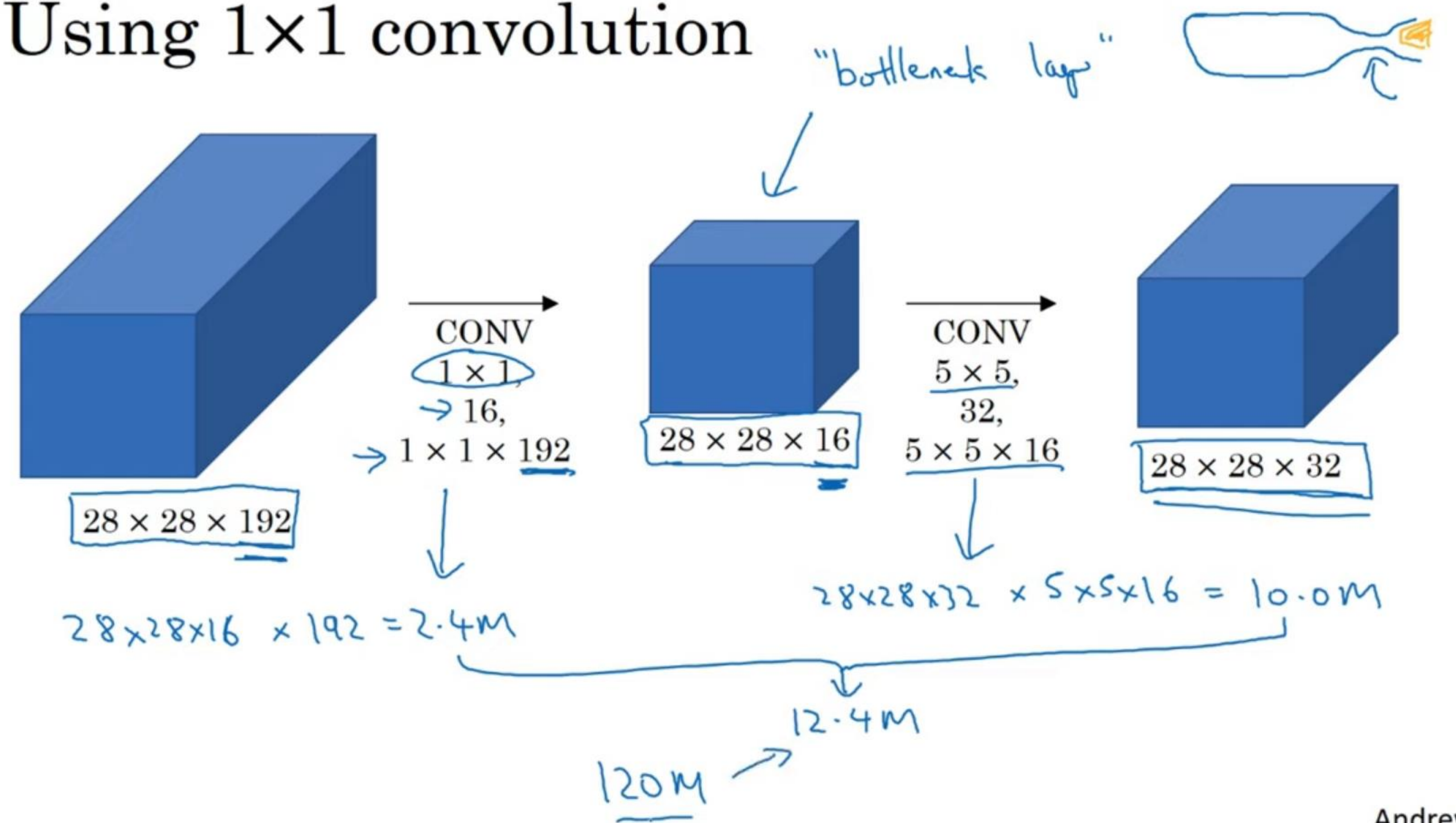


32 filters.

filters are 5×5×192 multiplications

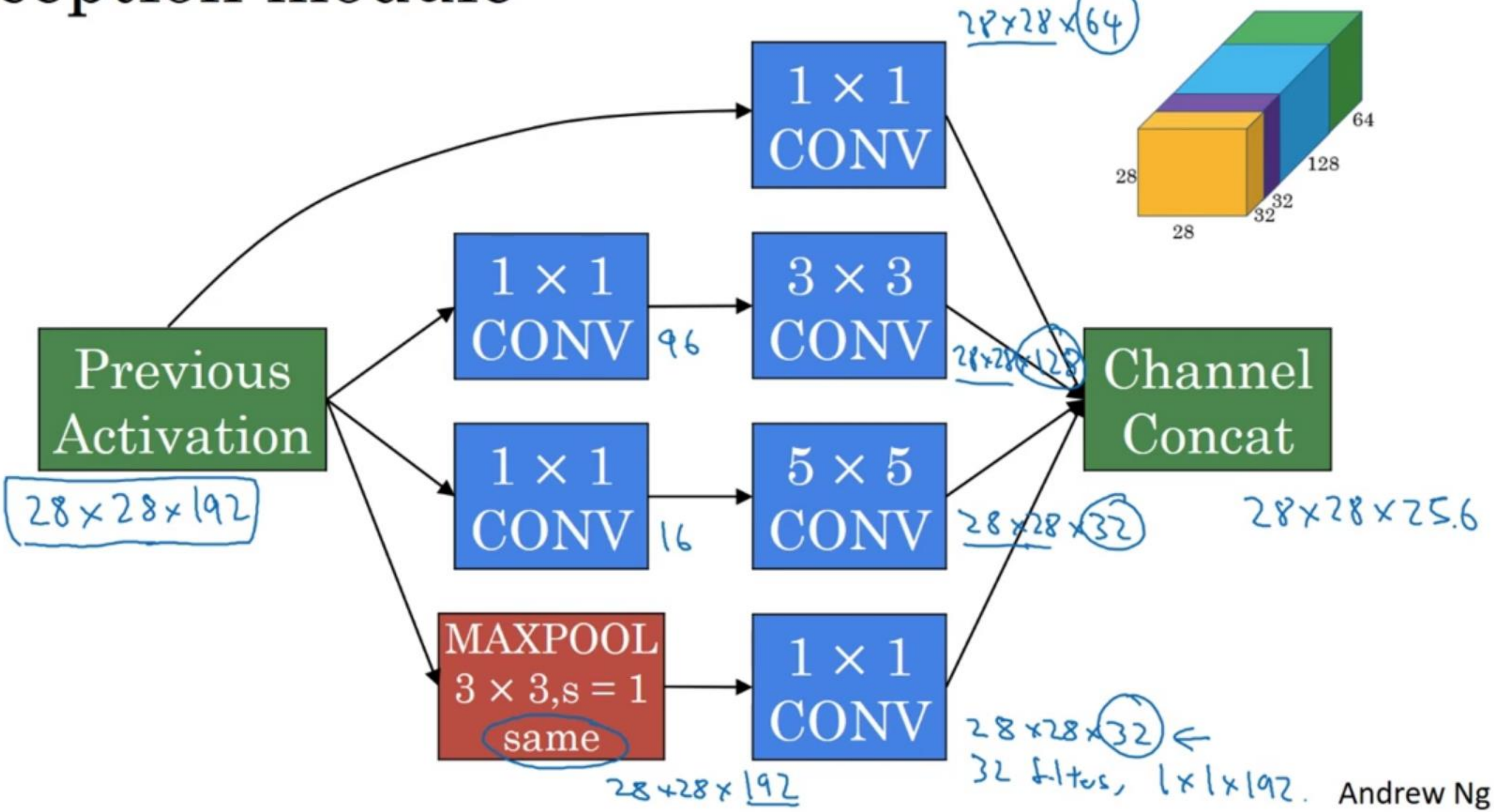
28×28×32 × 5×5×192 = 120M

Using 1×1 convolution



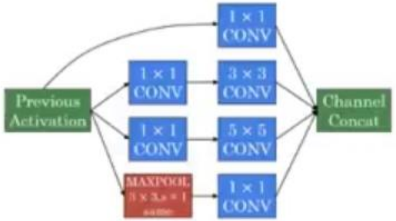
Andrew Ng

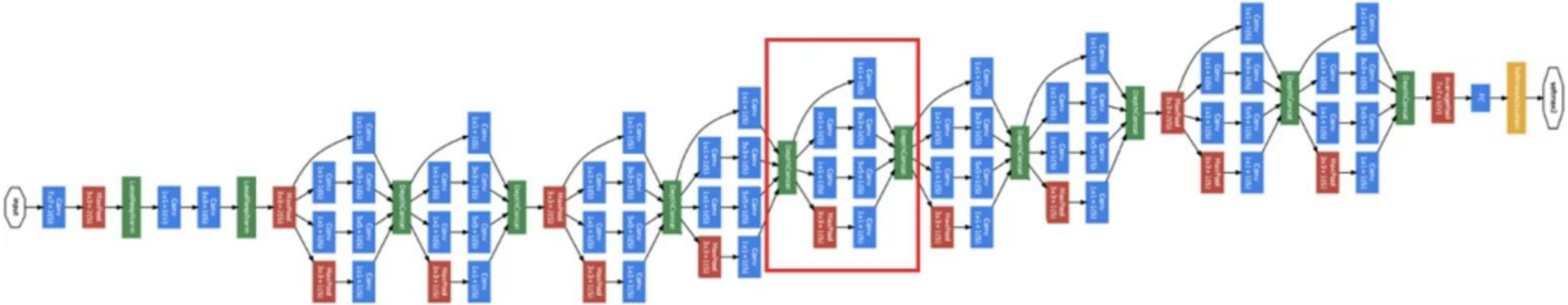
Inception module



Andrew Ng

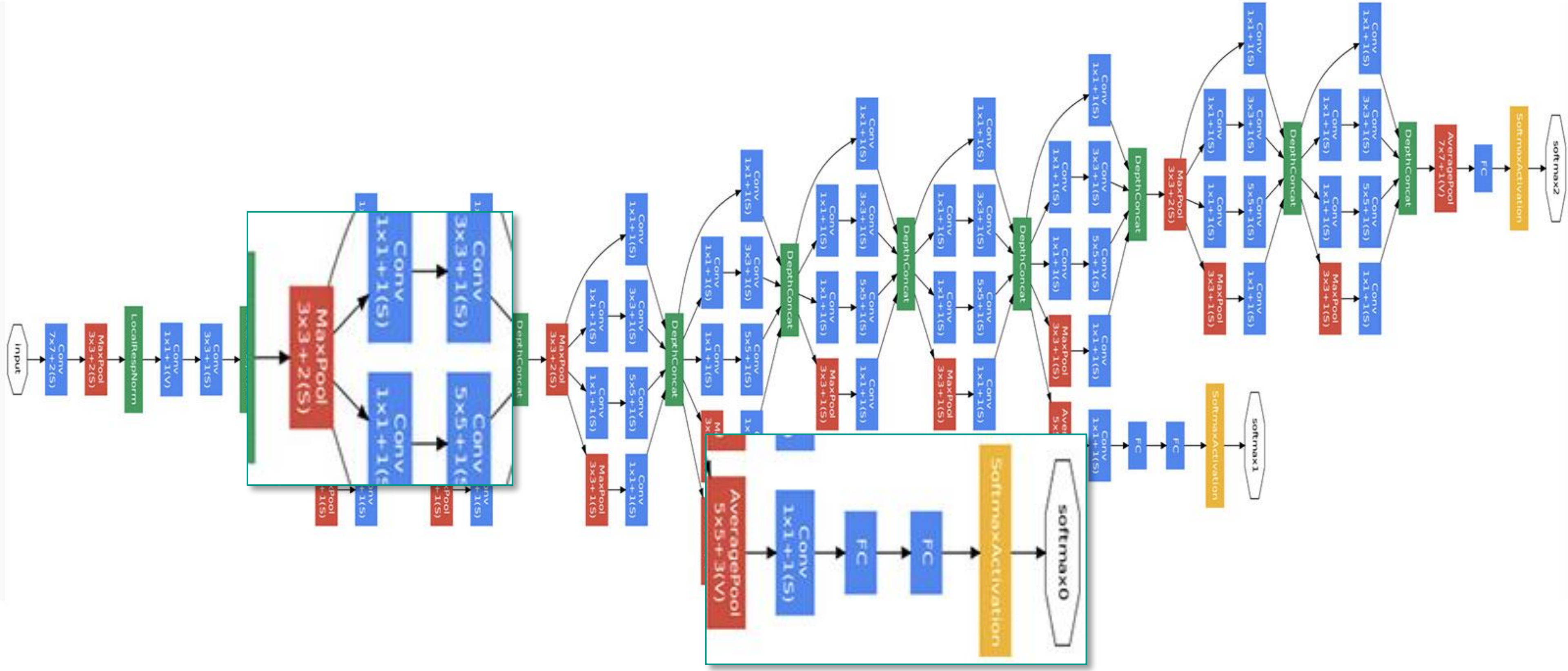
Inception network





[Szegedy et al., 2014, Going Deeper with Convolutions]

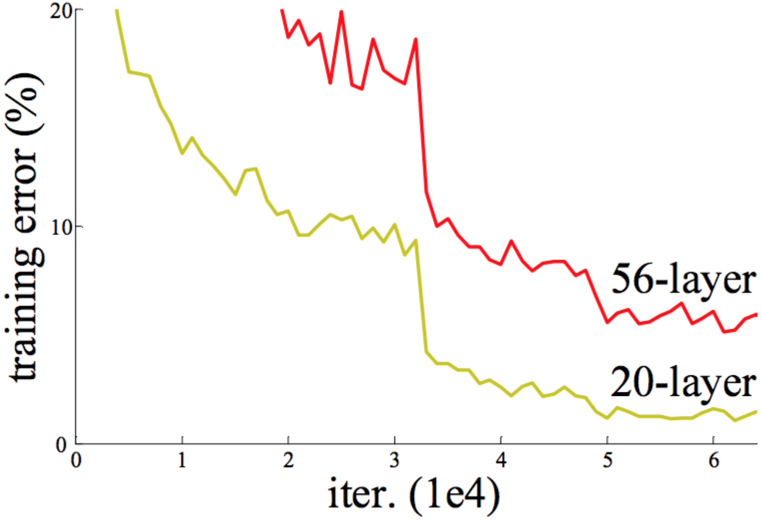
Andrew Ng

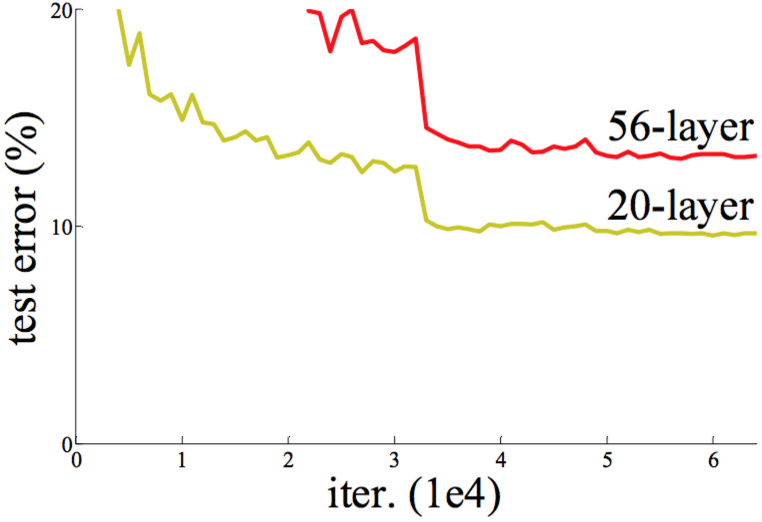


input
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
DepthConcat
AveragePool 5x5+3(V)
FC
SoftmaxActivation
softmax0
softmax1
softmax2

ResNet - Motivation

Issue: Deeper Networks performing worse on **training** data!
(as well as test data)



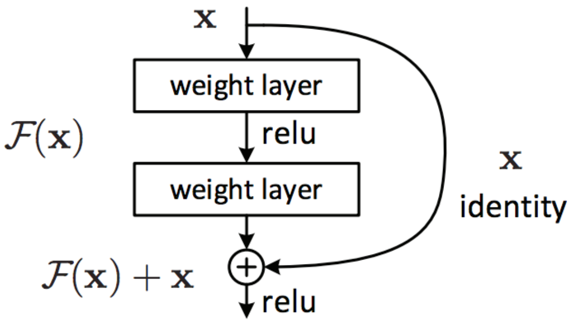


ResNet

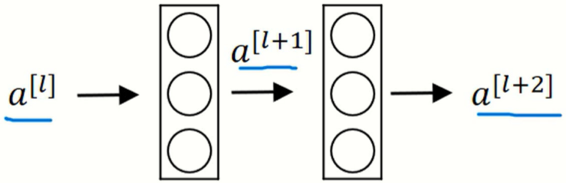
- Surprising because deeper networks should overfit more.
- So what's happening?
- Early layers of Deep Networks are very slow to adjust.
- Analogous to "Vanishing Gradient" issue.
- In theory, should be able to just have an "identity" transformation that makes the deeper network behave like a shallower one

ResNet

- Assumption: best transformation over multiple layers is close to $\mathcal{F}(x) + x$
- x → input to series of layers
- $\mathcal{F}(x)$ → function represented by several layers (such as convs)
- Enforce this by adding "shortcut connections"
- Add the inputs from an earlier layer to the output of current layer



Residual block



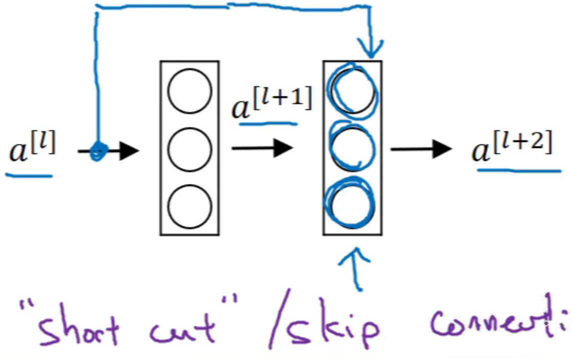
$a^{[l]} \rightarrow$ Linear $\rightarrow$ ReLU $\xrightarrow{a^{[l+1]}}$ Linear $\rightarrow$ ReLU $\rightarrow a^{[l+2]}$

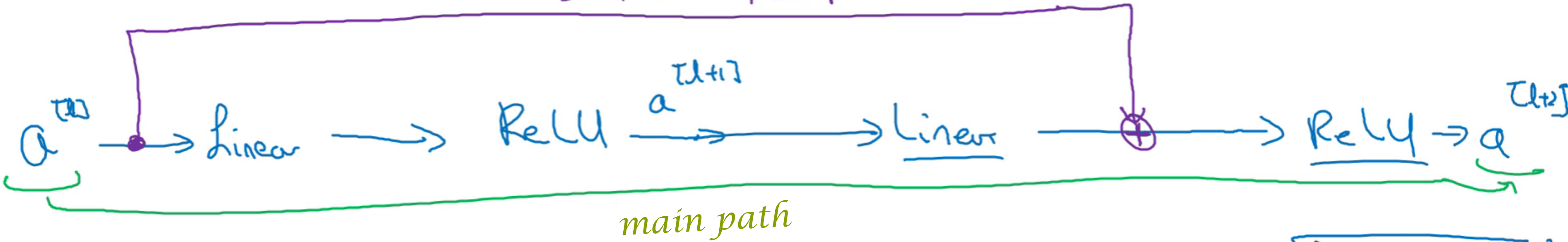
$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$ $\quad a^{[l+1]} = g(z^{[l+1]})$ $\quad z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$ $\quad a^{[l+2]} = g(z^{[l+2]})$

[He et al., 2015. Deep residual networks for image recognition]

Andrew Ng

Residual block





$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$ $\quad a^{[l+1]} = g(z^{[l+1]})$ $\quad z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$ $\quad \boxed{\cancel{a^{[l+2]} = g(z^{[l+2]})}}$

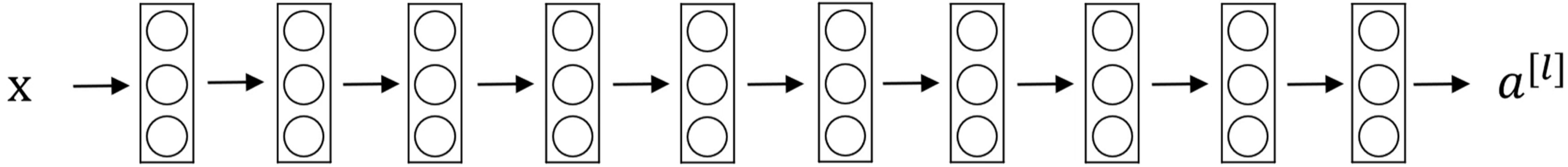
$$a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right)$$

[He et al., 2015. Deep residual networks for image recognition]

Andrew Ng

Residual Network

plain network

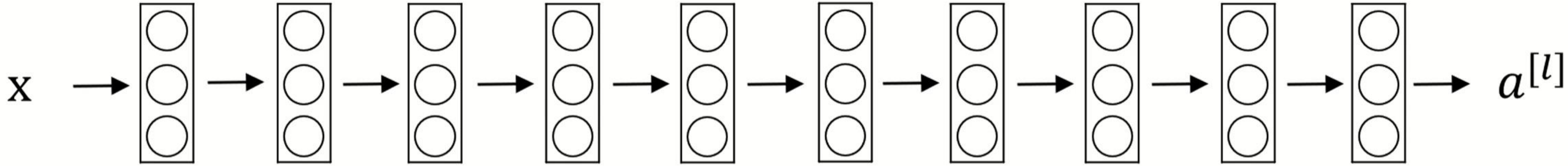


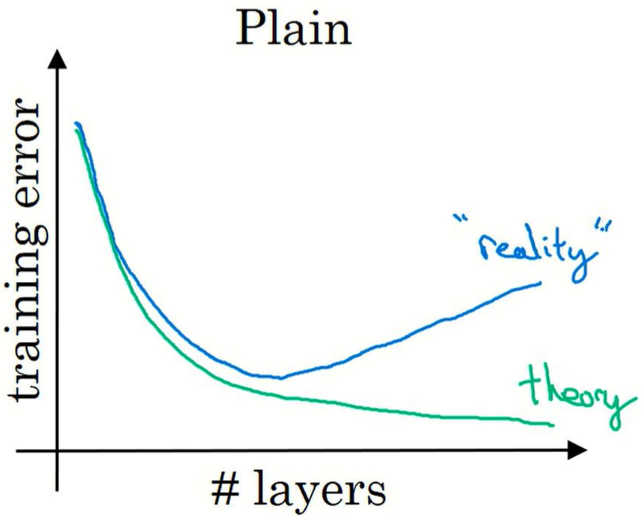
[He et al., 2015. Deep residual networks for image recognition]

Andrew Ng

Residual Network

plain network

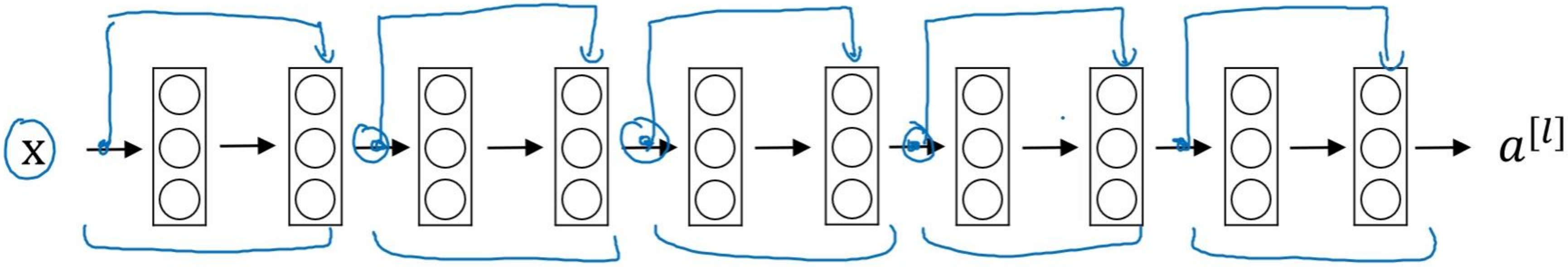


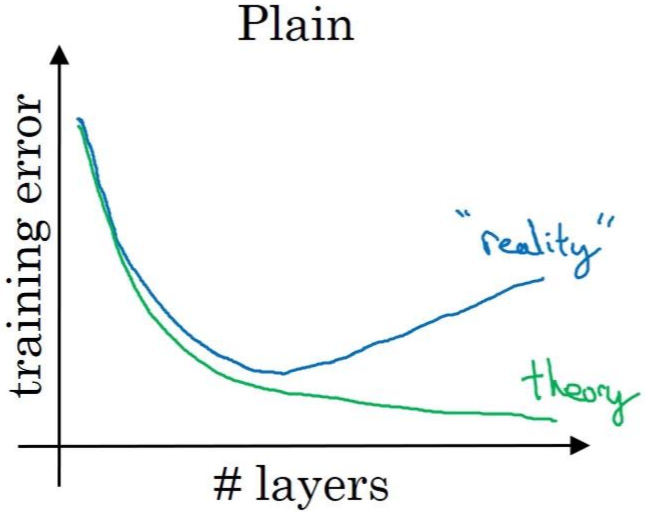


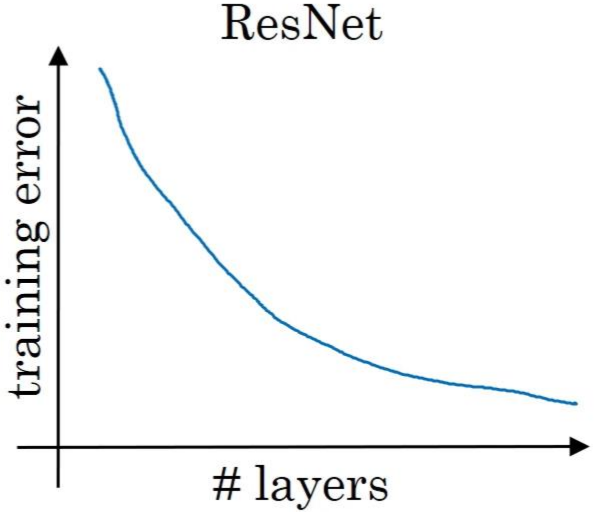
[He et al., 2015. Deep residual networks for image recognition]

Residual Network

with residual blocks

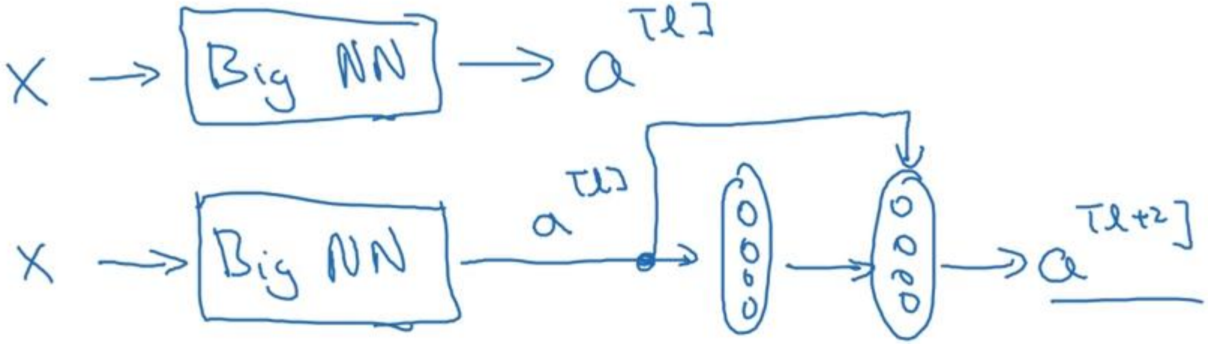






[He et al., 2015. Deep residual networks for image recognition]

Why do residual networks work?



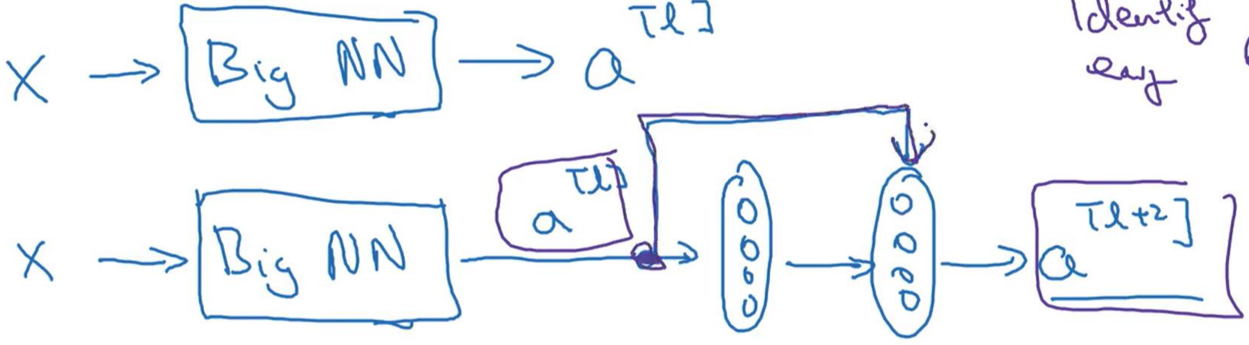
ReLU. $a \geq 0$

$$a^{[l+2]} = g(\underline{z^{[l+2]}} + \underline{a^{[l]}})$$

$$= g(\underline{W^{[l+2]} a^{[l+1]} + b^{[l+2]}} + a^{[l]}$$

Andrew Ng

Why do residual networks work?



Identity function is easy for Residual block to learn!

ReLU. $a \geq 0$

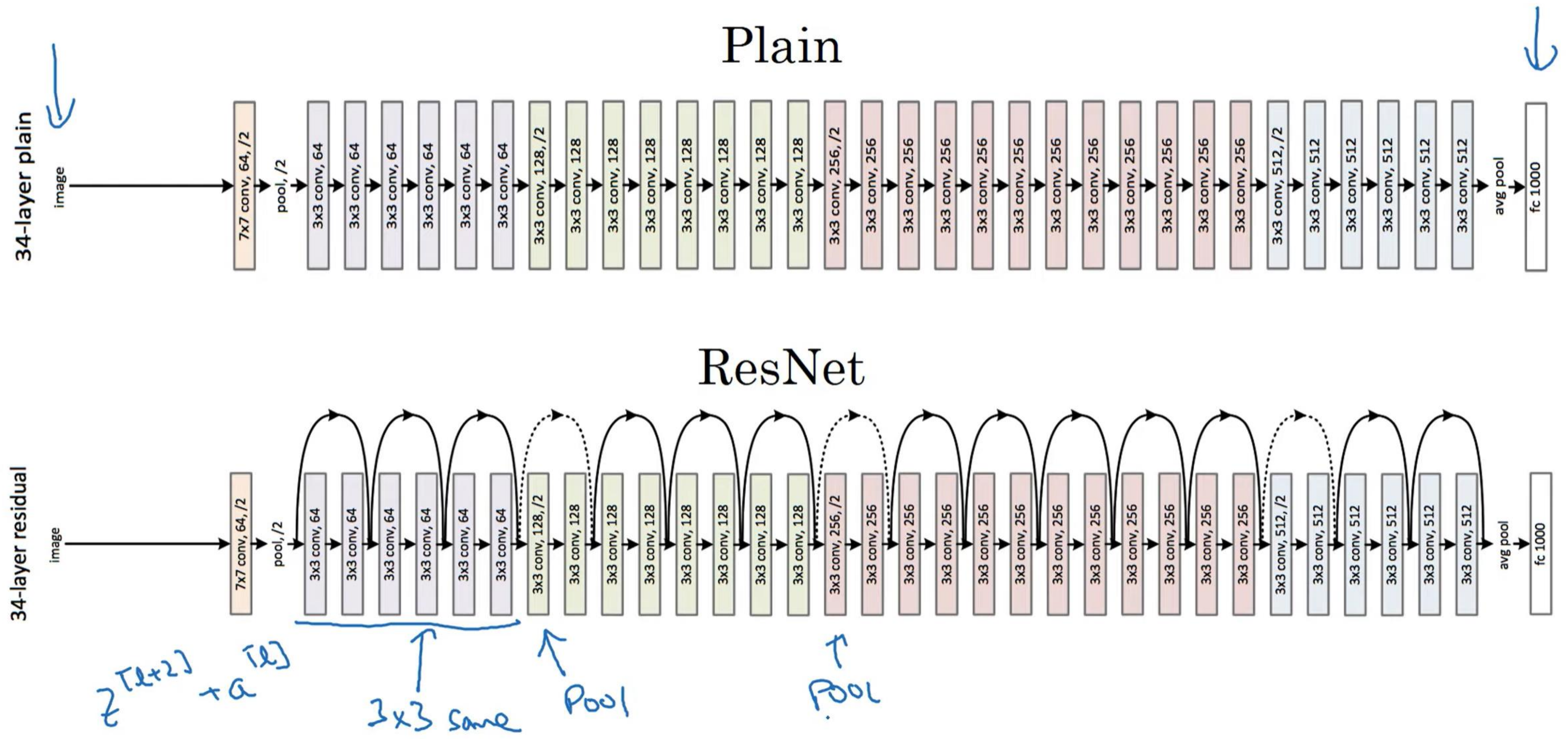
$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

$$= g(\cancel{W^{[l+2]} a^{[l+1]} + b^{[l+2]}} + a^{[l]}) = g(a^{[l]})$$

$$= a^{[l]}$$

If $W^{[l+2]} = 0$, $b^{[l+2]} = 0$

ResNet



[He et al., 2015. Deep residual networks for image recognition]

Andrew Ng

ResNet

- ResNet uses 152 layers and achieved 3.6% top five error to win the 2015 ImageNet competition.
- This exceeds average human ability.
- ResNet was faster to train than VGGNet, which has only 20 layers.

Software