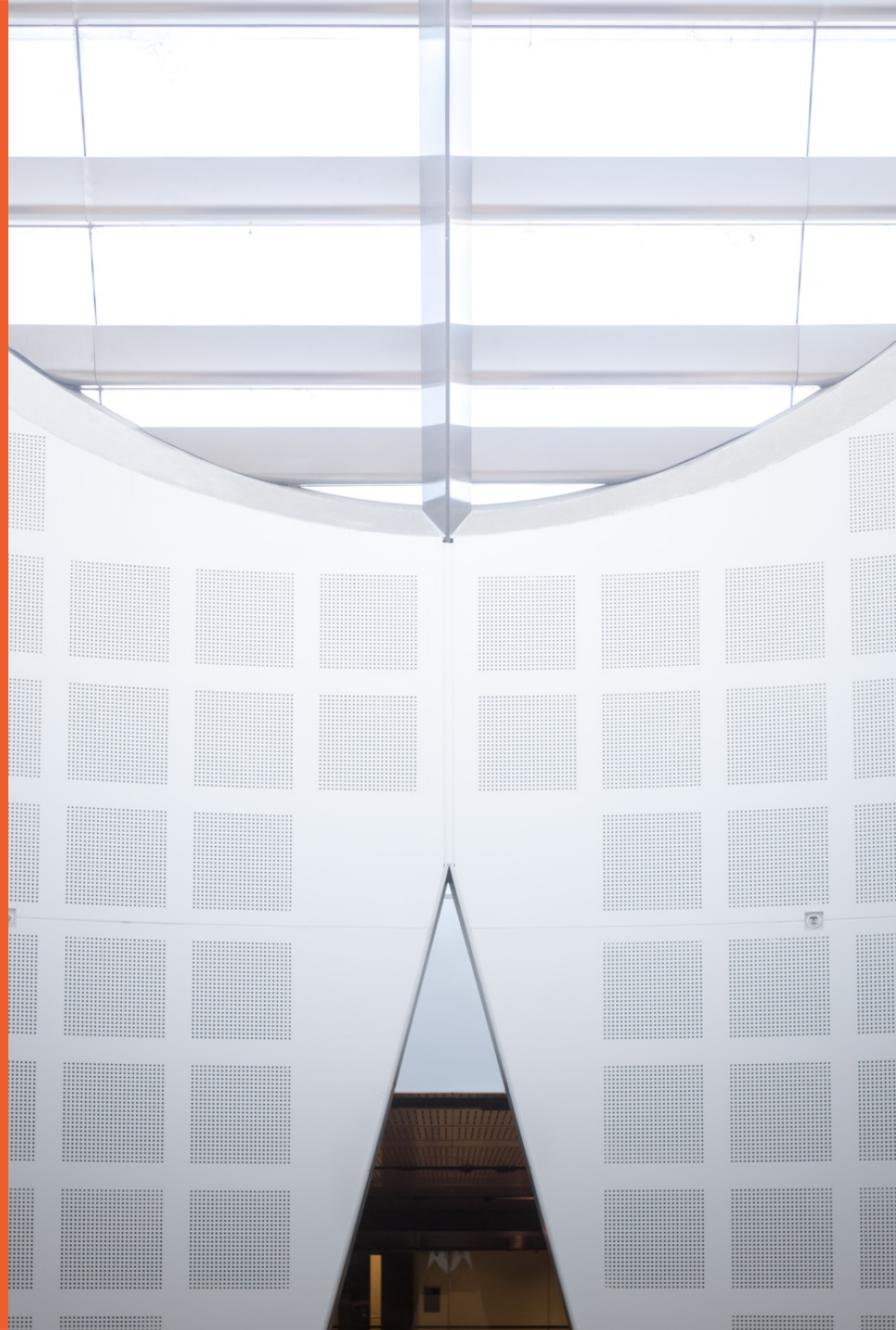
CNN Architectures

Dr Chang Xu

School of Computer Science



THE UNIVERSITY OF SYDNEY

ILSVRC

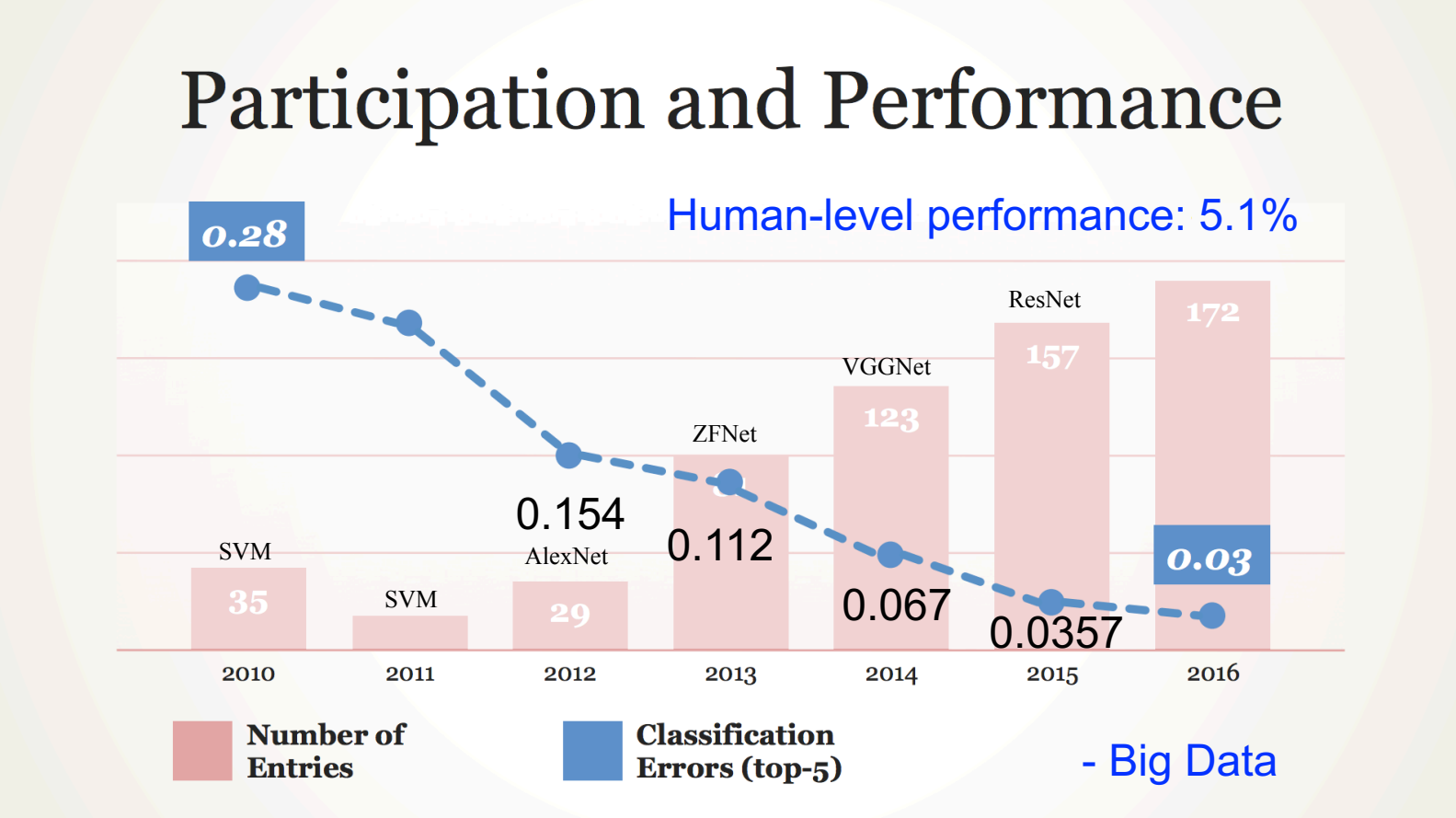
- **Image Classification**
  - one of the core problems in computer vision
  - many other tasks (such as object detection, segmentation) can be reduced to image classification

- **ImageNet Large Scale Visual Recognition Challenge**
  - 2010 - 2017
  - main tasks: classification and detection
  - a large-scale benchmark for image classification methods
    - 1000 classes
    - 1.2 million training images
    - 50 thousand verification images
    - 150 thousand test images

* *Some slides are borrowed from cs231n.stanford.edu*

ILSVRC

ILSVRC



- Big Data

- GPU

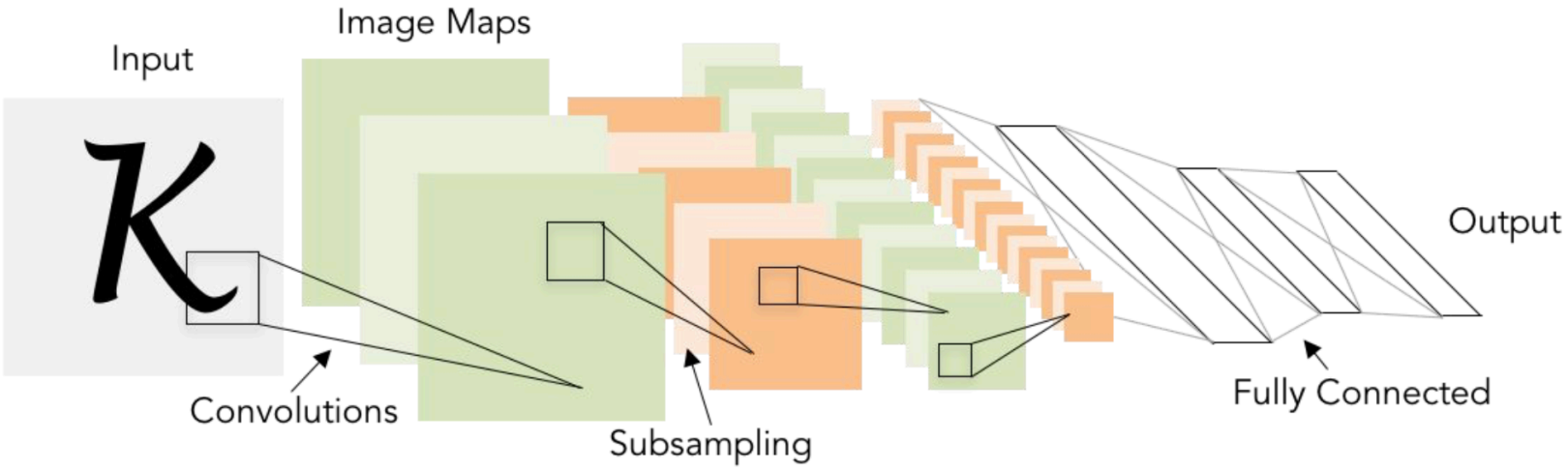
- Algorithm improvement

Winning CNN Architectures

Model	AlexNet	ZF Net	GoogLeNet	Resnet
Year	2012	2013	2014	2015
#Layer	8	8	22	152
Top 5 Acc	15.4%	11.2%	6.7%	3.57%
Data augmentation	√	√	√	√
Dropout	√	√		
Batch normalization				√

CNN Architecture: Part I

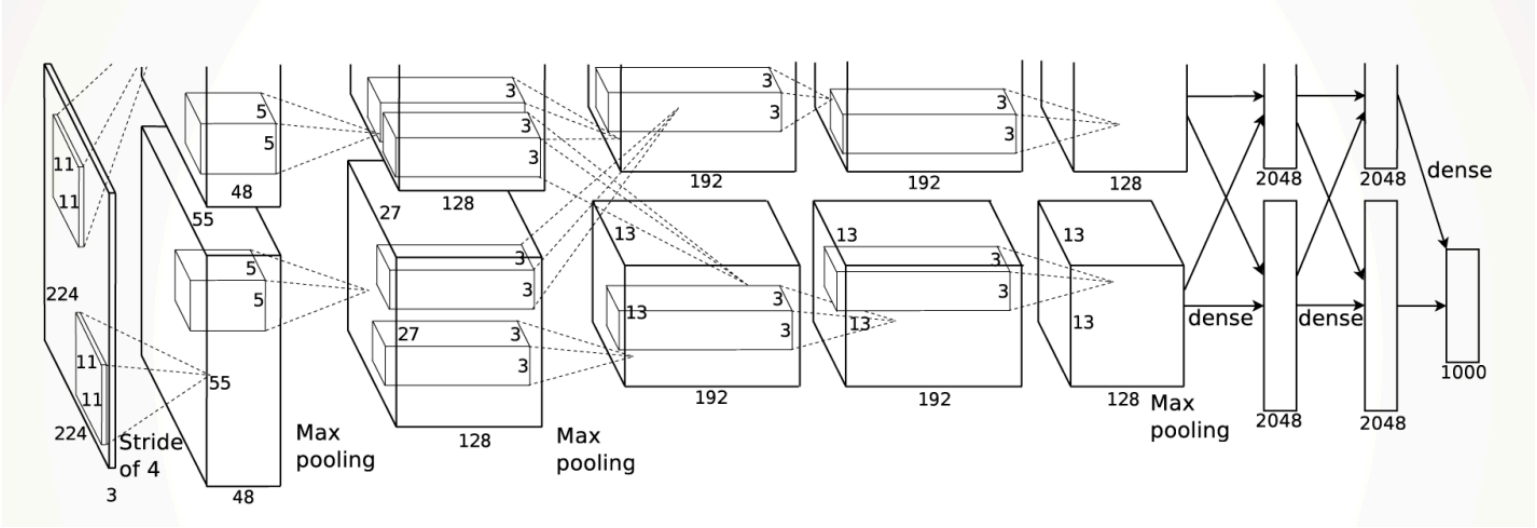
❑ **LeNet** [LeCun et al., 1998]



- Architecture is [CONV-POOL-CONV-POOL-FC-FC]
- CONV: 5x5 filter, stride=1
- POOL: 2x2 filter, stride=2

CNN Architecture: Part I

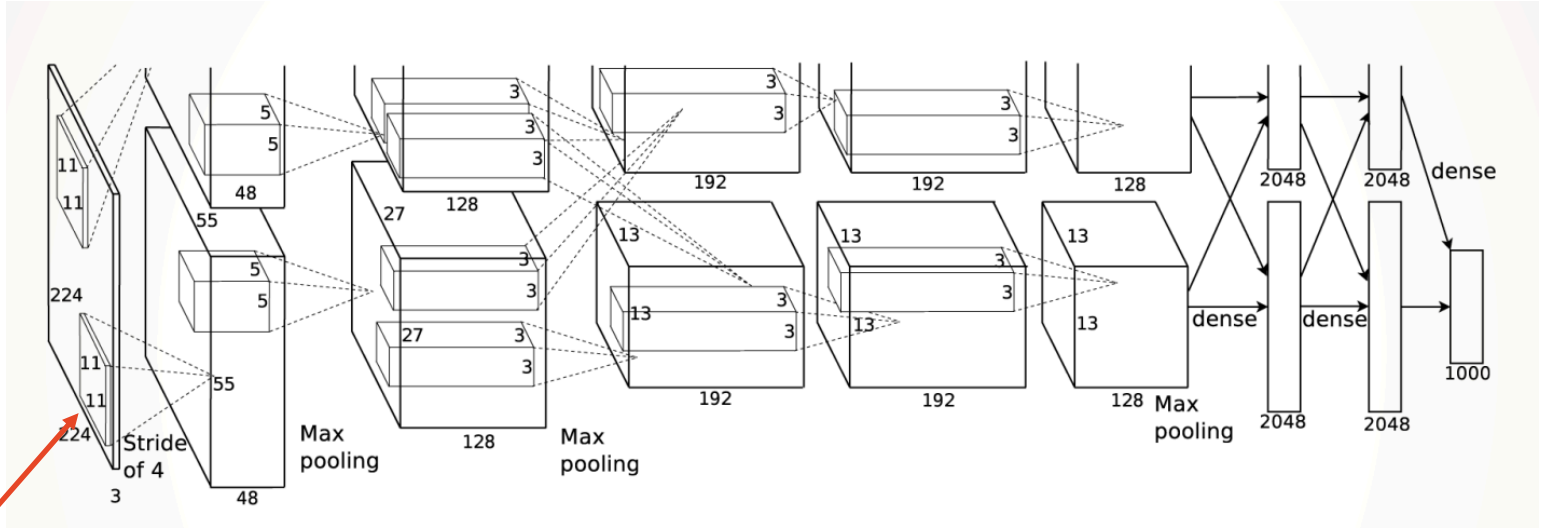
- **AlexNet** [Krizhevsky et al. 2012]



ILSVRC2010: 28.2%
ILSVRC2011: 25.8%
ILSVRC2012: 16.4% (second winner 26.2%)

CNN Architecture: Part I

❑ **AlexNet** [Krizhevsky et al. 2012] -5 conv layers, 3 fully connected layers.



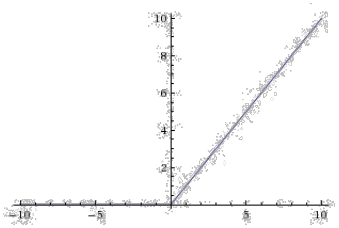
11x11 filters

- Activation function: ReLU
- Data Augmentation
- Dropout (drop rate=0.5)
- Local Response Normalization
- Overlapping Pooling

CNN Architecture: Part I

❑ **AlexNet** [Krizhevsky et al. 2012] -5 conv layers, 3 fully connected layers.

- Activation function: ReLU
- Data Augmentation
- Dropout
- Local Response Normalization
- Overlapping Pooling



$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

$$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

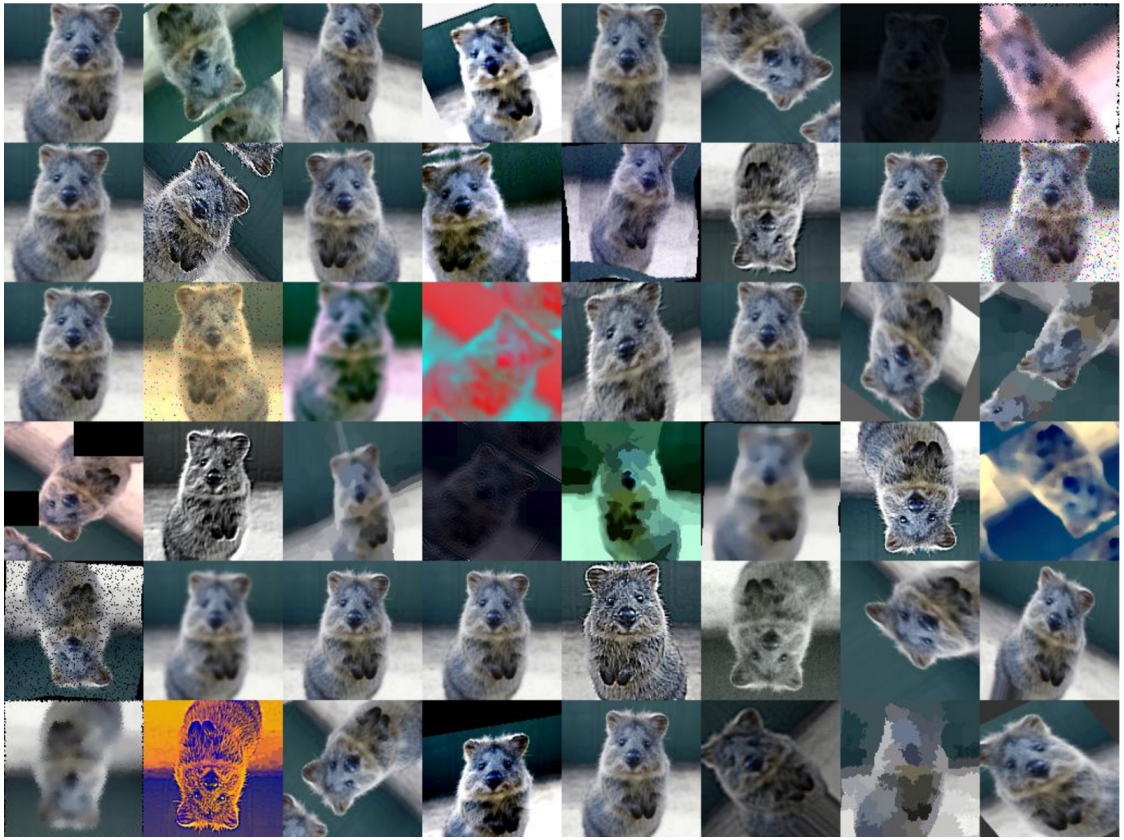
Pros:
- Easy to feed forward
- Easy to back propagate
- Help prevent saturation
- Sparse

Cons:
- Dead ReLU

CNN Architecture: Part I

❑ AlexNet [Krizhevsky et al. 2012]

- Activation function: ReLU
- Data Augmentation
- Dropout
- Local Response Normalization
- Overlapping Pooling

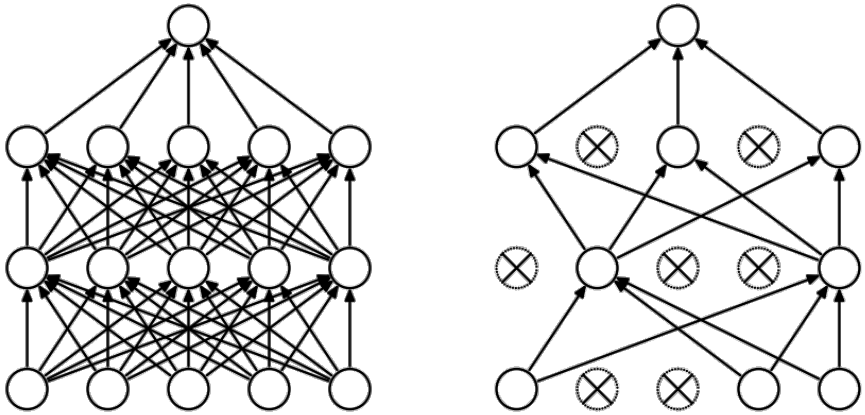


- Randomly Rotate Images
- Flip Images
- Shift Images
- Contrast Stretching
- Adaptive Equalization
- etc.

CNN Architecture: Part I

❑ **AlexNet** [Krizhevsky et al. 2012]

- Activation function: ReLU
- Data Augmentation
- Dropout
- Local Response Normalization
- Overlapping Pooling



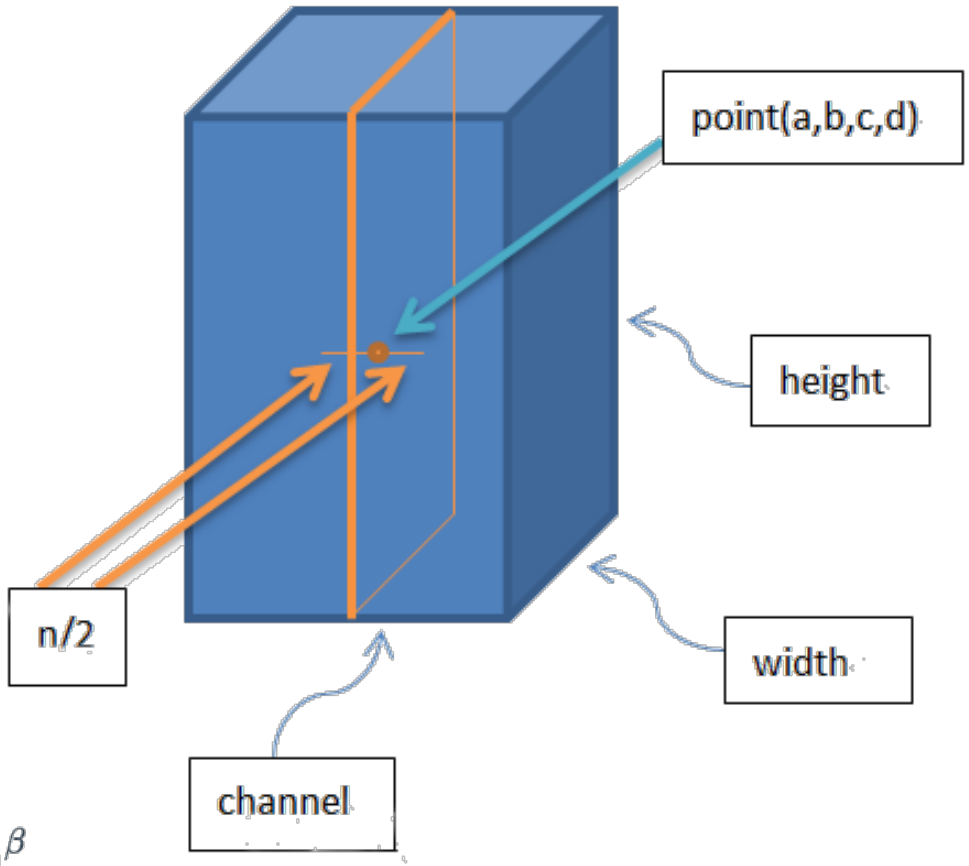
In practice, dropout trains 2^n networks (n – number of units).

Drop is a technique which deal with overfitting by combining the predictions of many different large neural nets at test time.

CNN Architecture: Part I

❑ AlexNet [Krizhevsky et al. 2012]

- Activation function: ReLU
- Data Augmentation
- Dropout
- Local Response Normalization
- Overlapping Pooling



$$b^i_{x,y} = a^i_{x,y} / (k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^i_{x,y})^2)^\beta$$

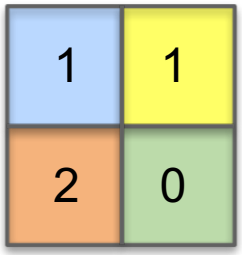
CNN Architecture: Part I

❑ AlexNet [Krizhevsky et al. 2012]

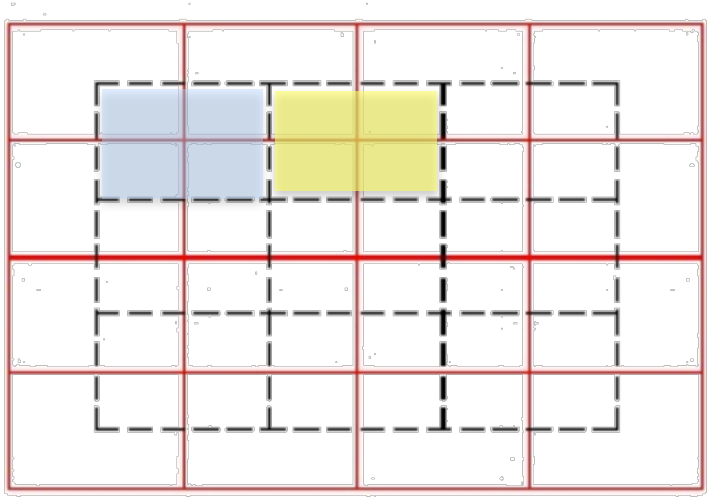
- Activation function: ReLU
- Data Augmentation
- Dropout
- Local Response Normalization
- Overlapping Pooling

Feature map

Max pooling



Average pooling



Pooling stride < Pooling kernel size

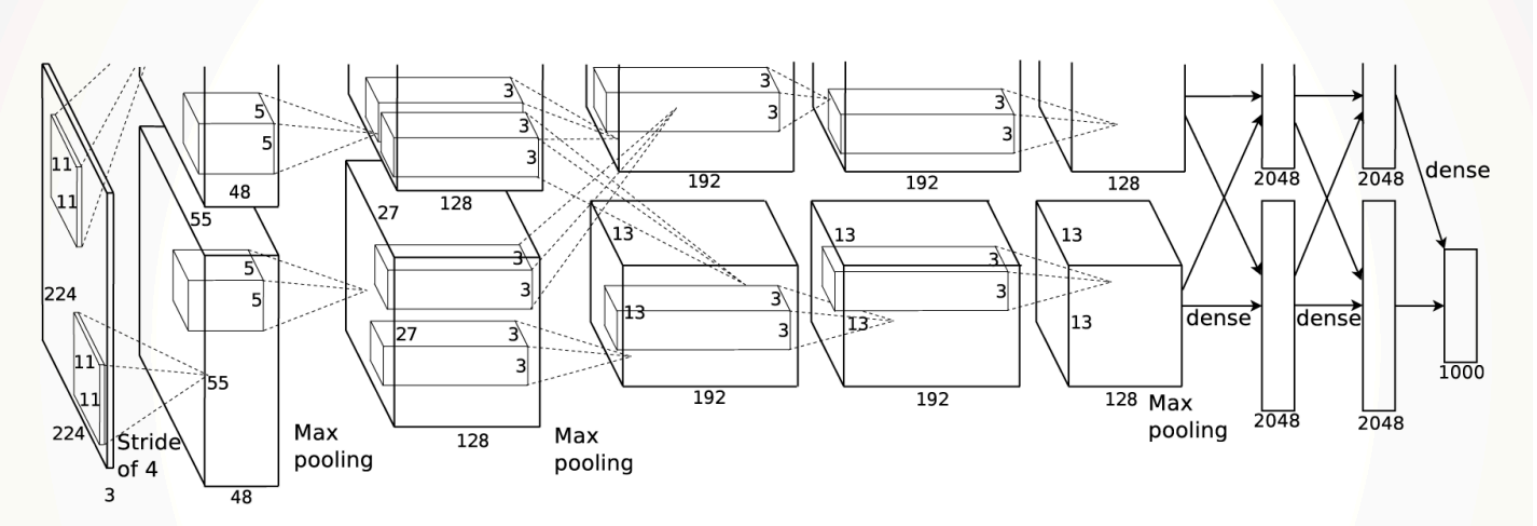
Overlapping Pooling

Standard Pooling

Pooling stride = Pooling kernel size

CNN Architecture: Part I

- **AlexNet** [Krizhevsky et al. 2012]

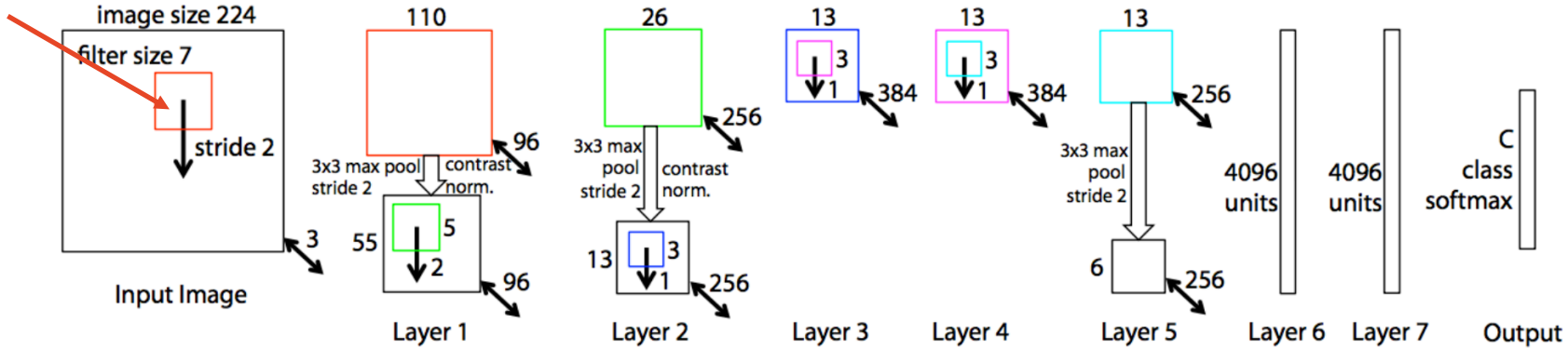


The problem set is divided into 2 parts, half executing on GPU 1 & another half on GPU 2.

CNN Architecture: Part I

❑ ZFNet [Zeiler and Fergus, 2013]

7x7 filters



- An improved version of AlexNet: top-5 error from 16.4% to 11.7%
- First convolutional layer: 11x11 filter, stride=4 -> 7x7 filter, stride=2
- The number of filters increase as we go deeper.

CNN Architecture: Part I

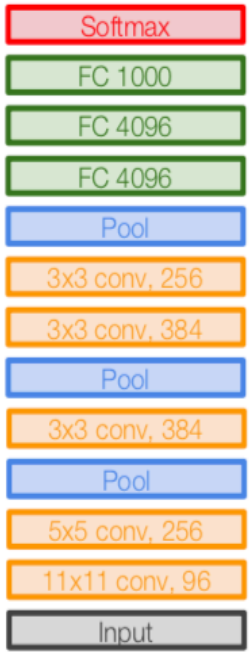
❑ **VGGNet** [Simonyan and Zisserman, 2014]

Small filters:

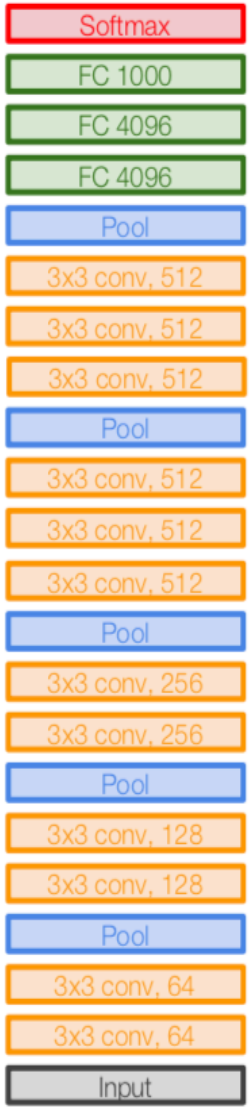
- 3x3 convolutional layers (stride 1, pad 1)
- 2x2 max-pooling, stride 2

Deeper networks:

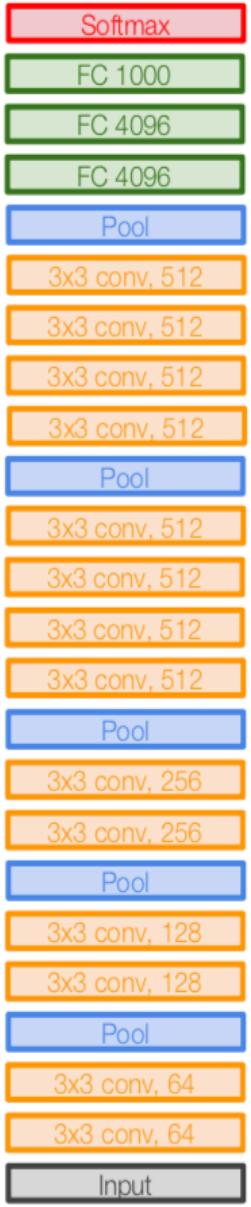
- AlexNet: 8 layers
- VGGNet: 16 or 19 layers



AlexNet



VGG16



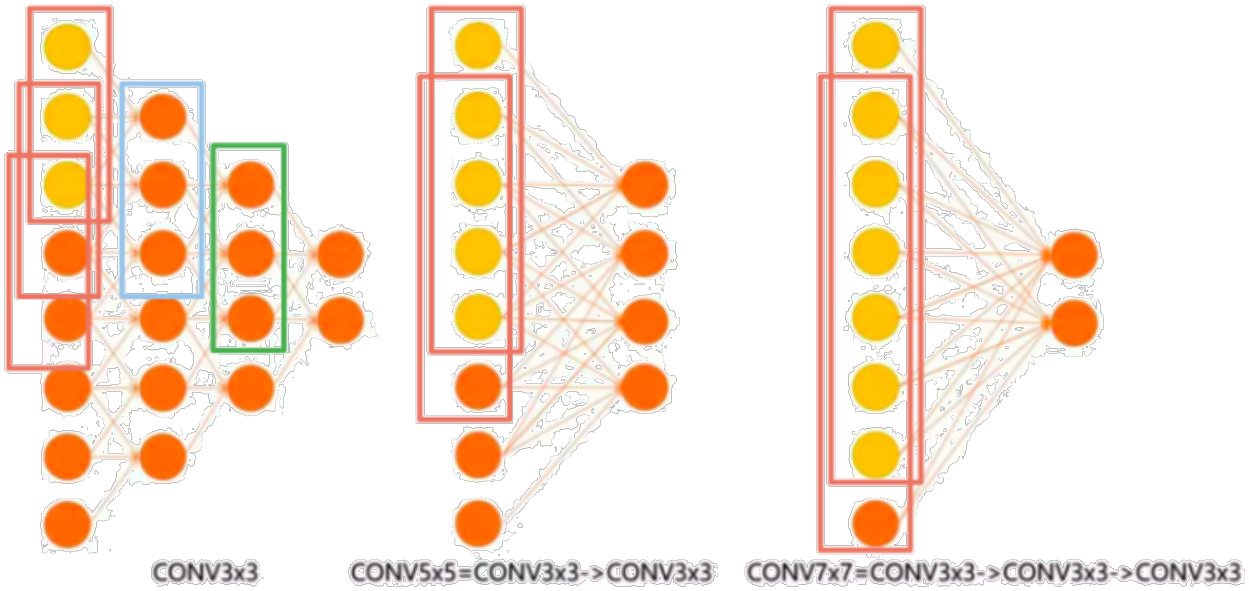
VGG19

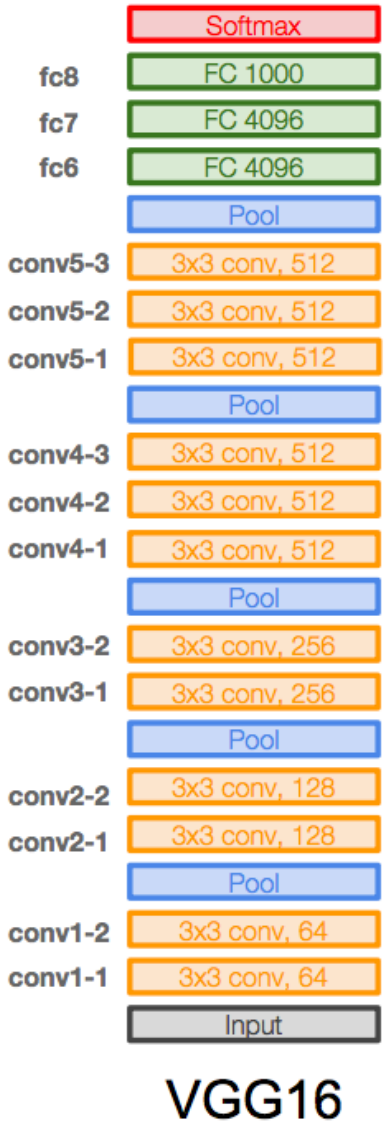
CNN Architecture: Part I

Why small filters?

- Two 3x3 layer = 5x5 layer
  Three 3x3 layer = 7x7 layer
- Deeper, more non-linearity
- Less parameters

Receptive Field (RF): the region in the input space that a particular CNN's feature is looking at (i.e. be affected by).





VGG16

Image credit to mc.ai

CNN Architecture: Part I

Why small filters?

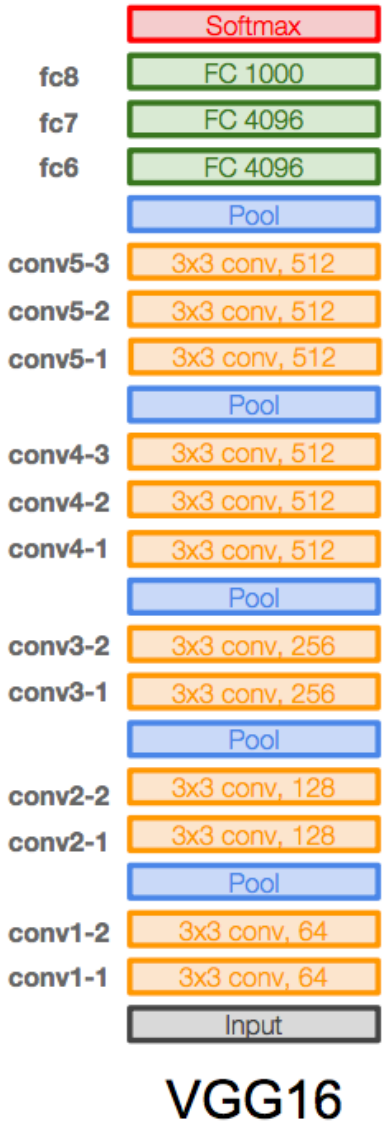
- Two 3x3 layer = 5x5 layer
  Three 3x3 layer = 7x7 layer
- Deeper, more non-linearity
- Less parameters

Why popular?

- FC7 features generalize well to other tasks
- Plain network structure

The rule for network design

- Prefer a stack of small filters to a large filter



VGG16

CNN Architecture: Part I

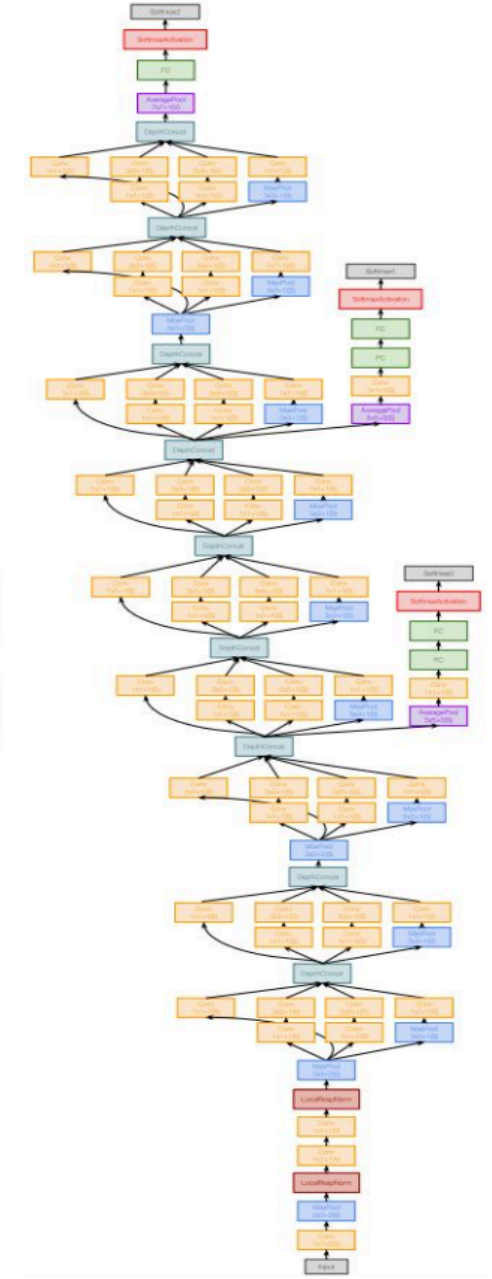
❑ GoogLeNet [Szegedy et al., 2014]

Deeper networks

- 22 layers.
- Auxiliary loss

Computational efficiency

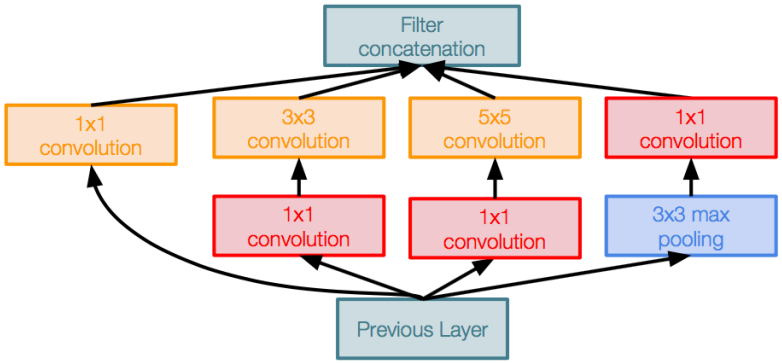
- Inception module
- Remove FC layer, use global average pooling
- 12x less parameters than AlexNet



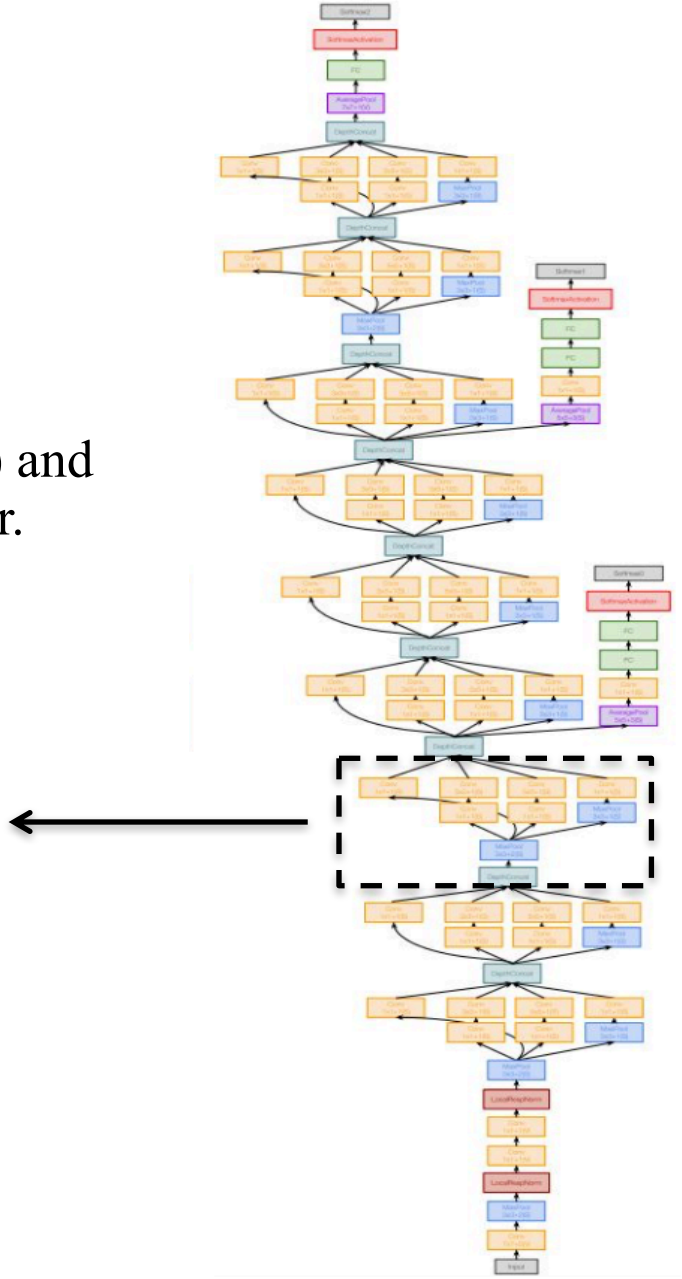
CNN Architecture: Part I

❑ GoogLeNet [Szegedy et al., 2014]

Design a good local network topology (NIN) and then stack these modules on top of each other.



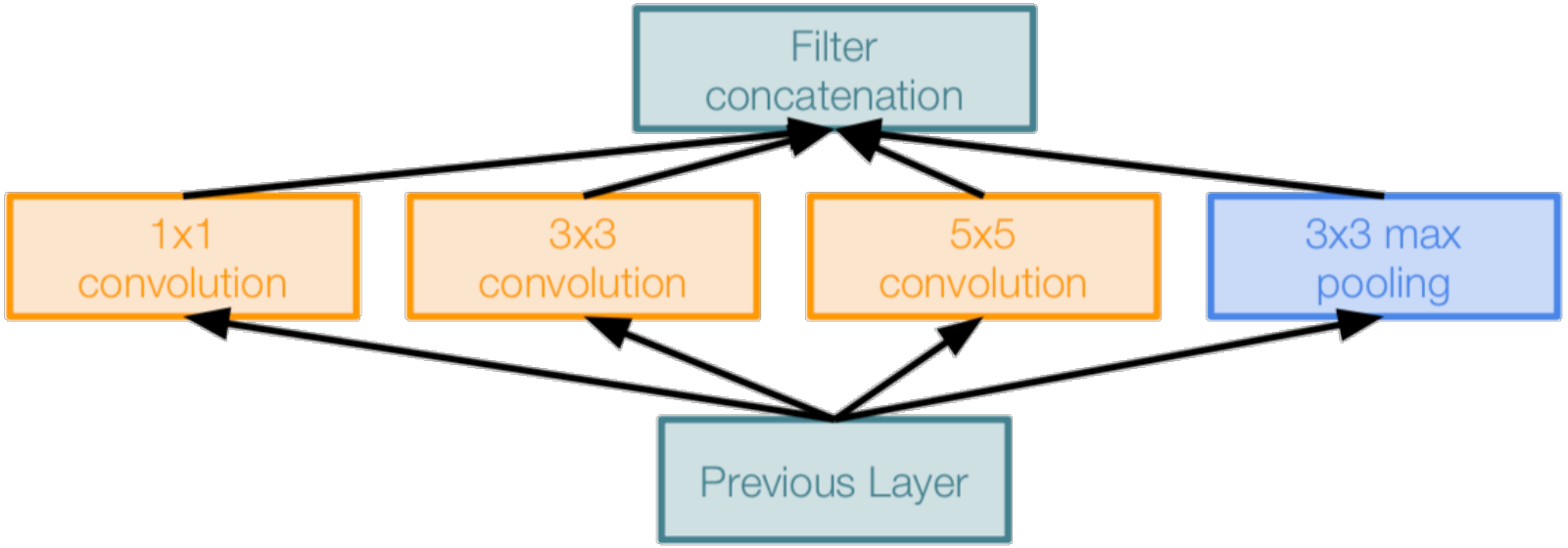
Inception module



CNN Architecture: Part I

- Different receptive fields
- The same feature dimension
- The effectives of pooling

❑ **GoogLeNet** [Szegedy et al., 2014]



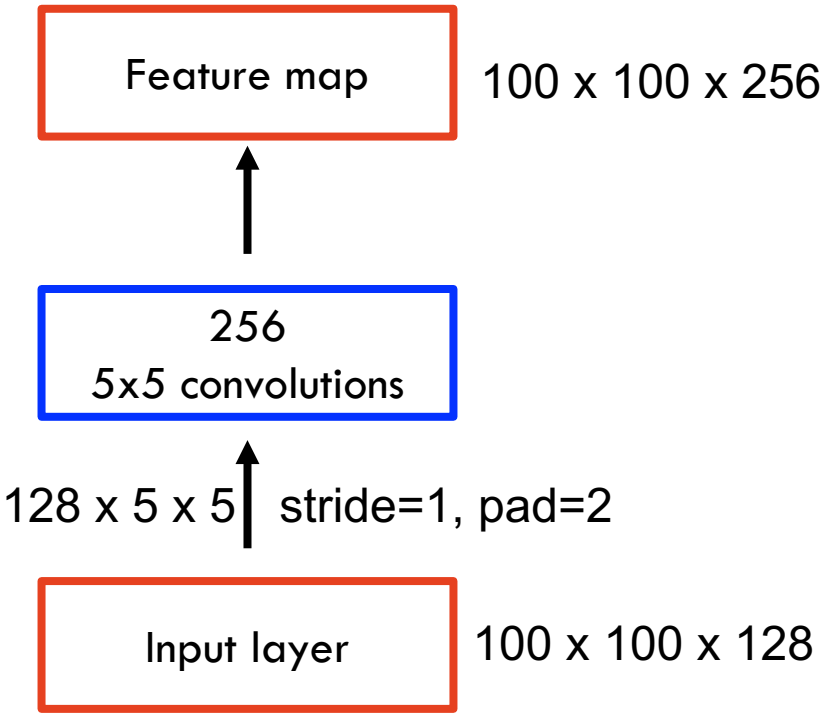
Naive Inception module

$$\text{Output Size} = \frac{N+2P-K}{S} + 1$$

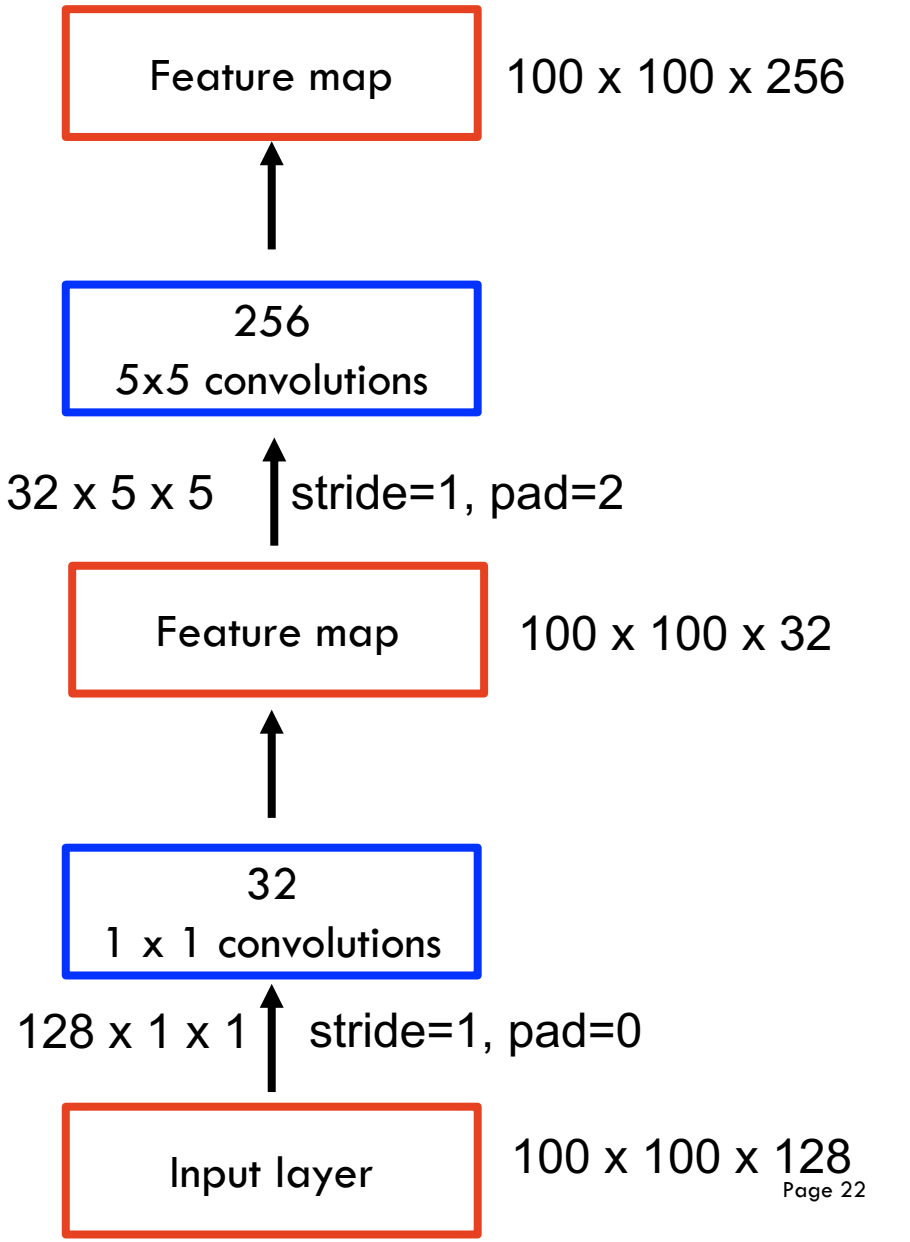
CNN Architecture: Part I

❑ **GoogLeNet** [Szegedy et al., 2014]

#parameter: 128x5x5x256

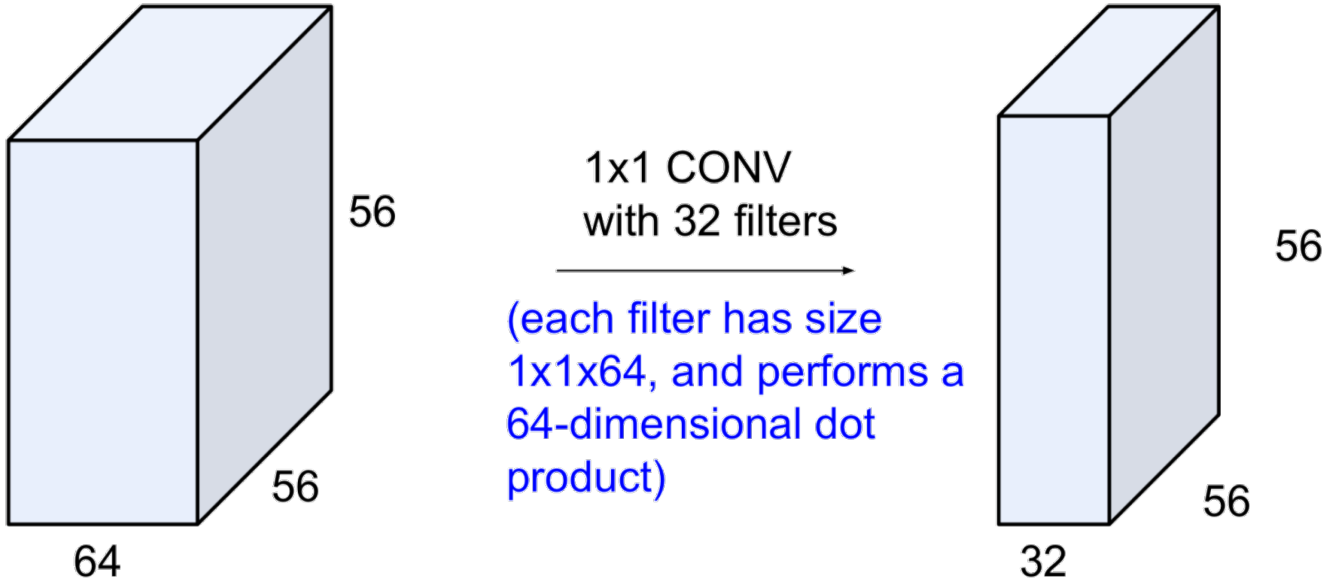


#parameter: 128x1x1x32 + 32x5x5x256



CNN Architecture: Part I

❑ **GoogLeNet** [Szegedy et al., 2014]

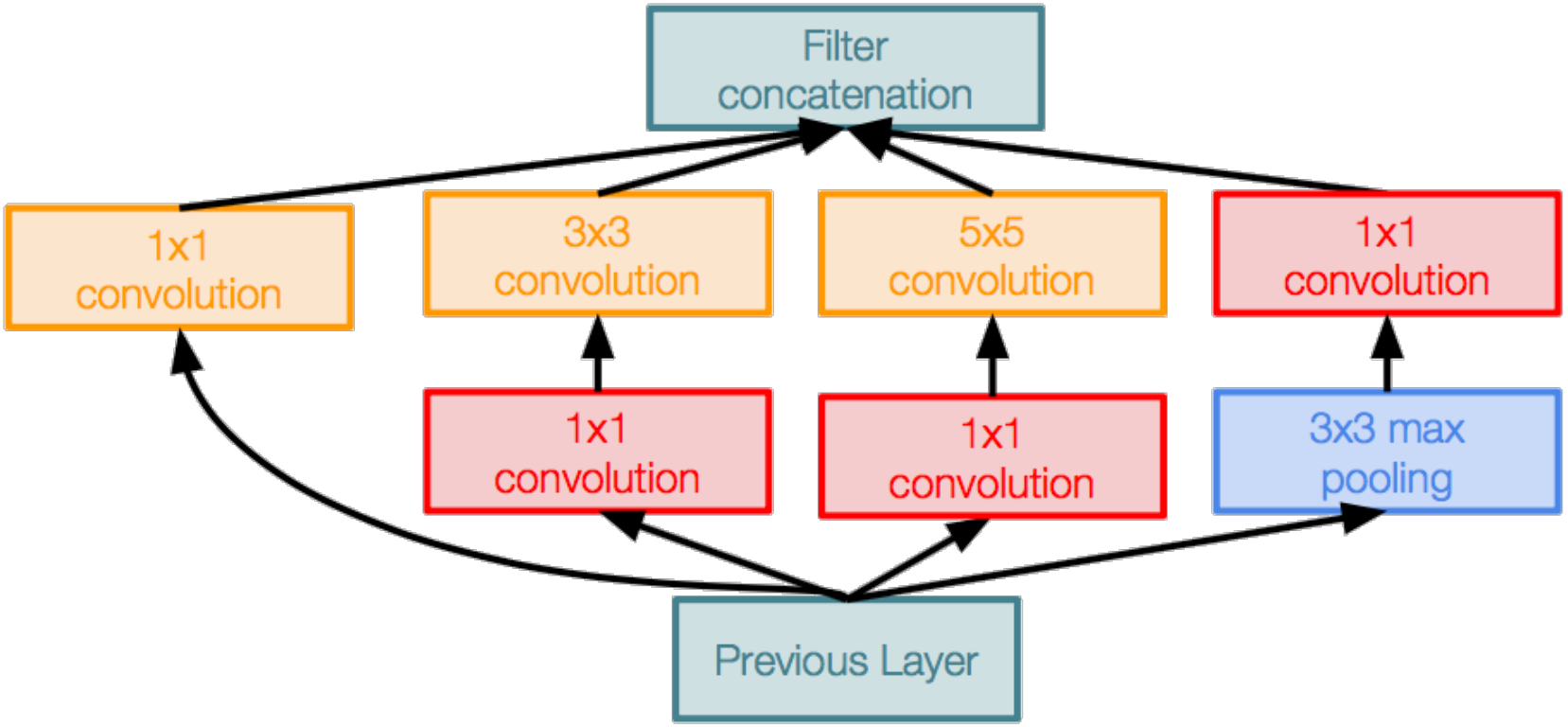


1x1 convolutional layer

- preserve spatial dimensions (56x56)
- reduces depth (64 -> 32)

CNN Architecture: Part I

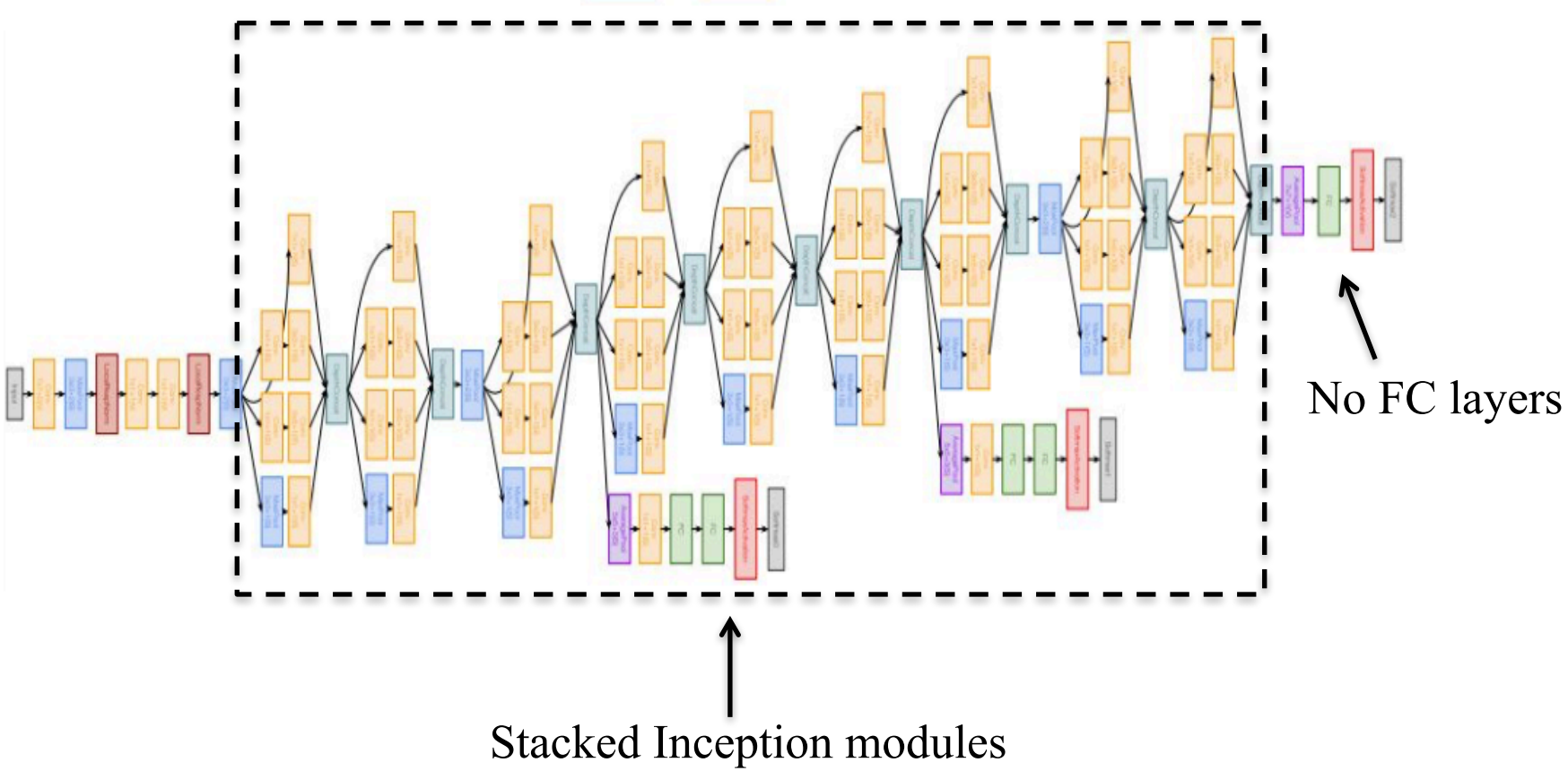
❑ **GoogLeNet** [Szegedy et al., 2014]



Inception module

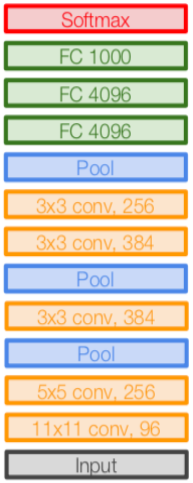
CNN Architecture: Part I

❑ **GoogLeNet** [Szegedy et al., 2014]



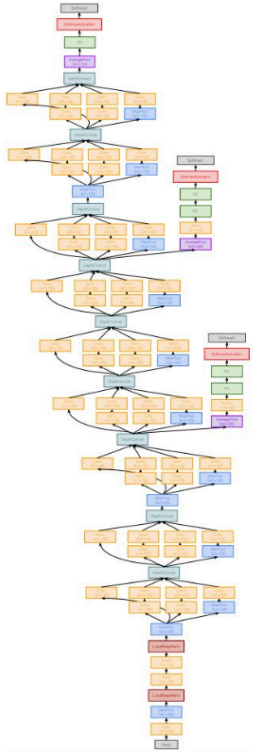
CNN Architecture: Part I

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



AlexNet (9 layers)

VGG16 (16 layers)

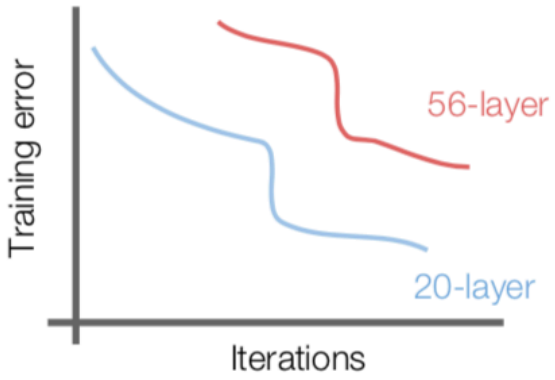


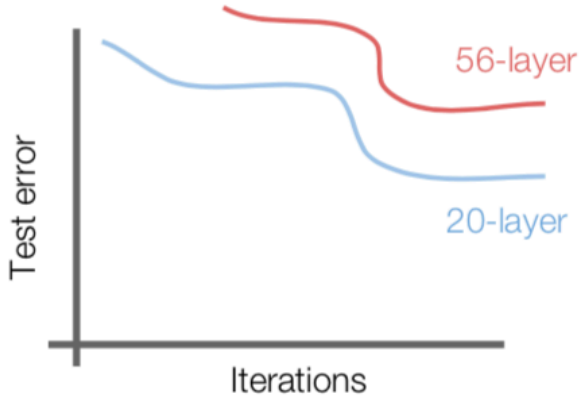
GoogLeNet (22 layers)

CNN Architecture: Part I

❑ ResNet [He et al., 2015]

Stacking deeper layers on a "plain" convolutional neural network
=> the deeper model performs worse, but not overfitting.



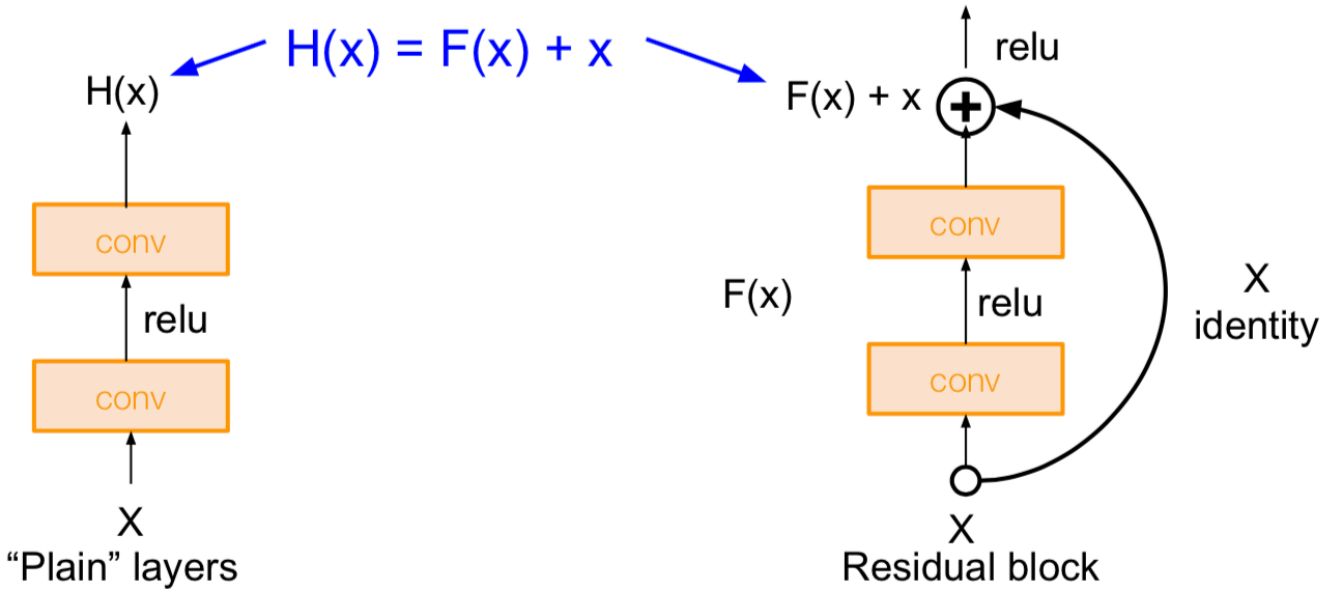


Hypothesis: deeper models are harder to optimize.

CNN Architecture: Part I

❑ ResNet [He et al., 2015]

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



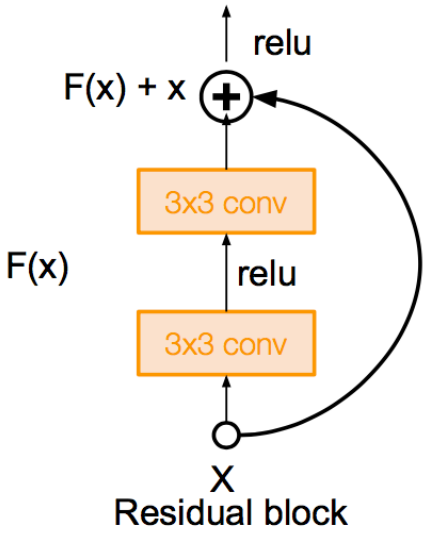
CNN Architecture: Part I

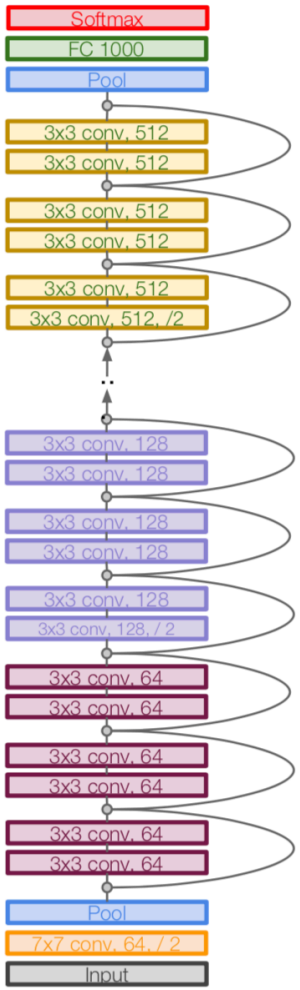
❑ ResNet [He et al., 2015]

Very deep networks using residual connections

Basic design:

- all 3x3 conv (almost)
- spatial size /2 => #filter x2
- just deep
- average pooling (remove FC)
- no dropout



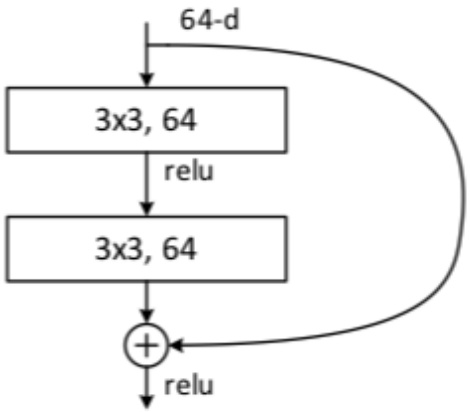


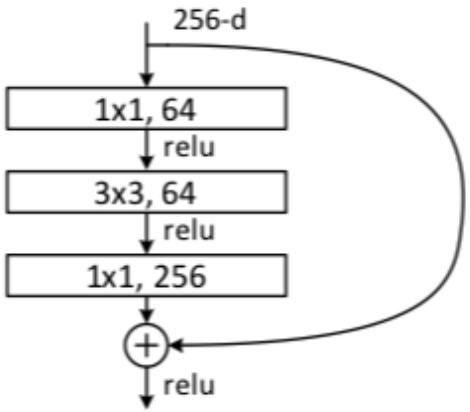
ResNet

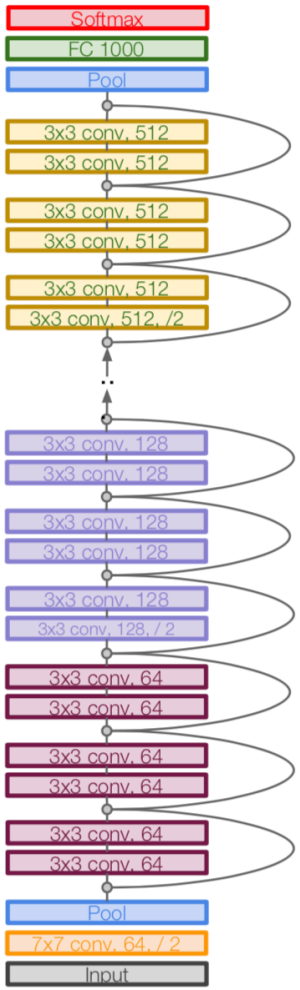
CNN Architecture: Part I

❑ ResNet [He et al., 2015]

Use "**bottleneck**" layer to improve efficiency (similar to GoogLeNet).



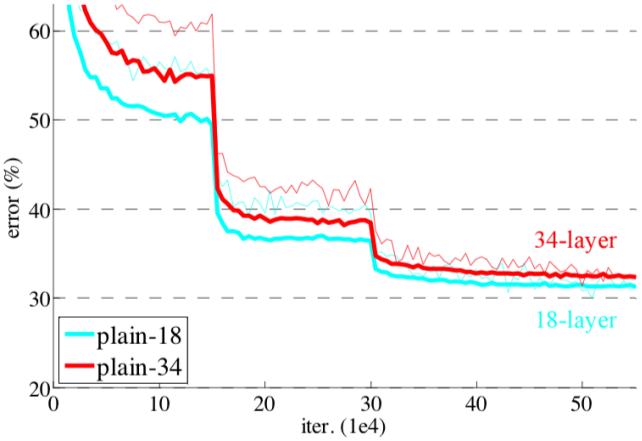


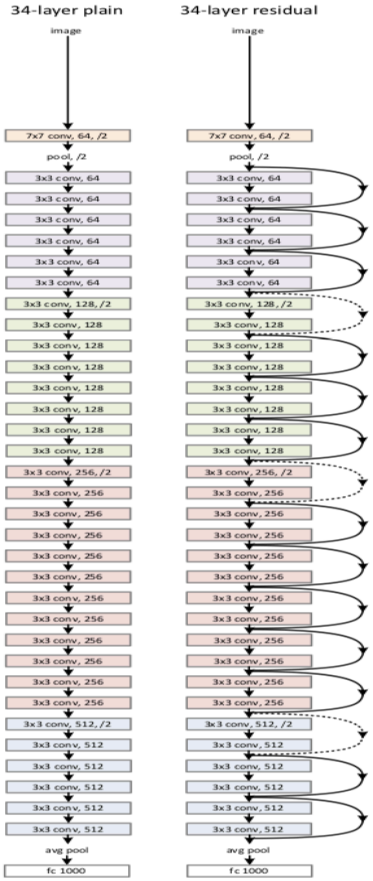


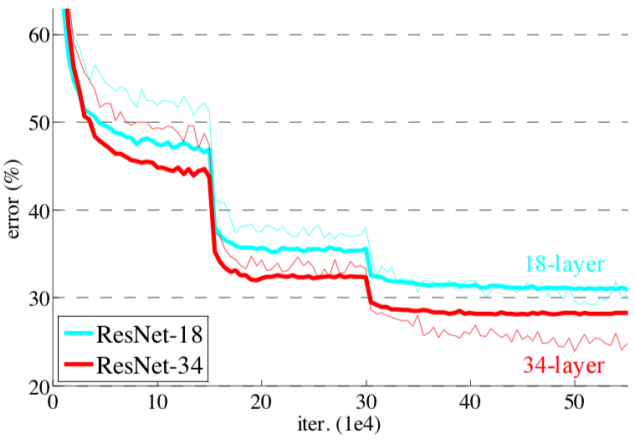
ResNet

CNN Architecture: Part I

- **ResNet** [He et al., 2015]







CNN Architecture: Part I

❑ ResNet [He et al., 2015]

- **1st places in all five main tracks**
  - ImageNet Classification: "*Ultra-deep*" 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

Features matter: deeper features are well transferrable

CNN Architecture: Part II

CNN Architecture: Part II

Inception: Inception v1,v2,v3,v4

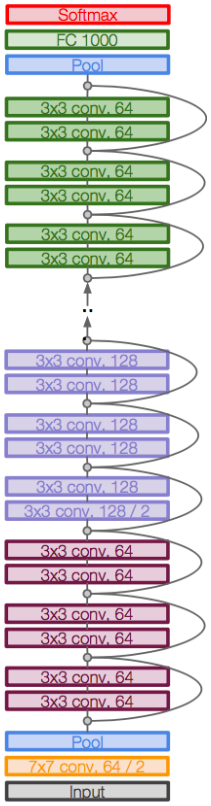
Inception-res-v1,v2

Xception

ResNet: Wide-ResNet

ResNeXt

DenseNet



CNN Architecture: Part II

❑ Inception v2

- batch normalization
- remove LRN and dropout
- small kernels (inspired by VGG)

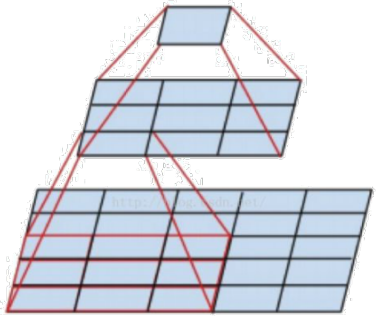
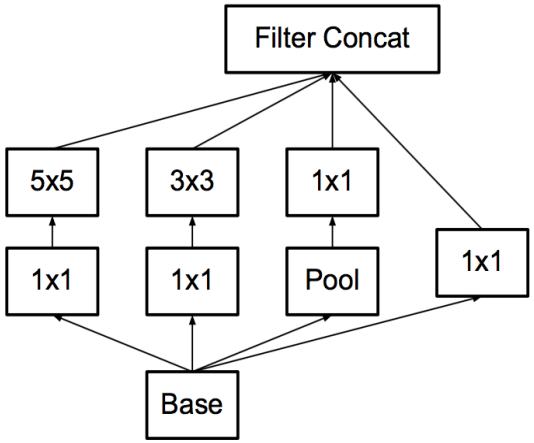
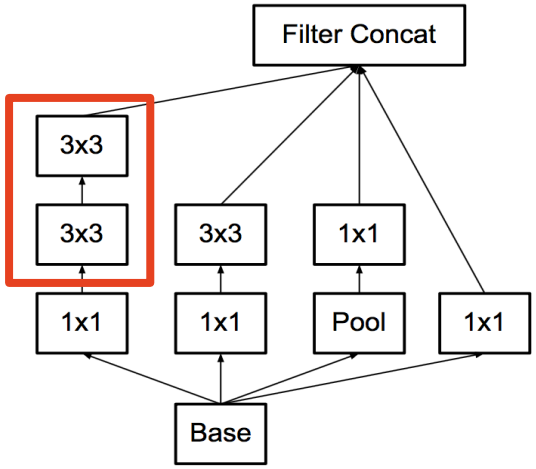


Figure 1. Mini-network replacing the 5 × 5 convolutions.



Inception module v1
(In GoogLeNet)



CNN Architecture: Part II

❑ Inception v2

Factorization
- 7x7 filter -> 7x1 and 1x7 filters
- 3x3 filter -> 3x1 and 1x3 filters

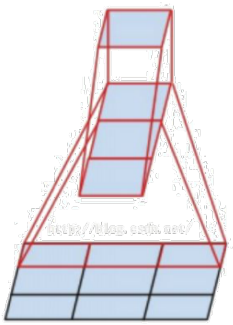
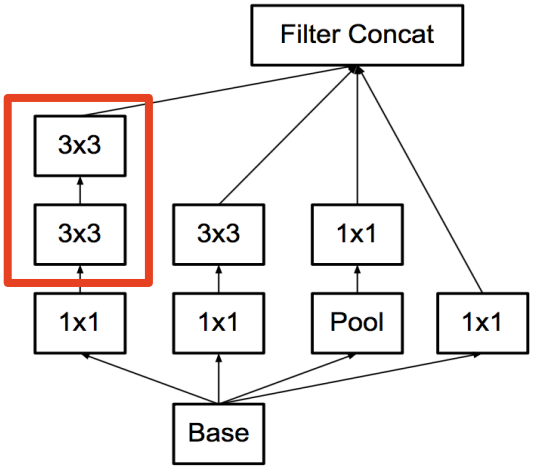
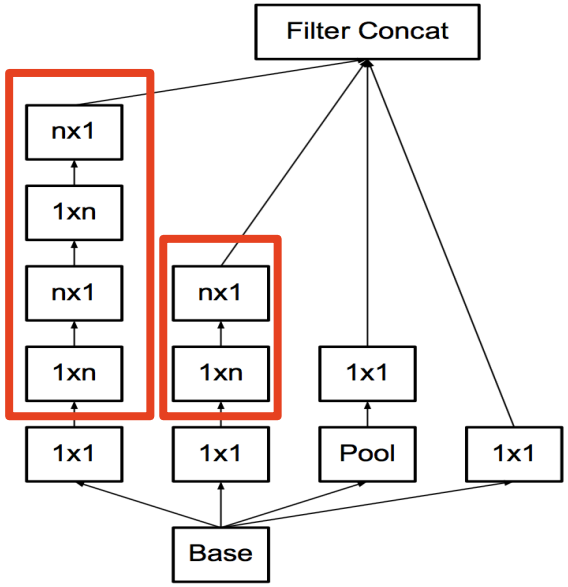


Figure 3. Mini-network replacing the 3 × 3 convolutions. The lower layer of this network consists of a 3 × 1 convolution with 3 output units.





CNN Architecture: Part II

❑ Inception v3

Label Smoothing

For a training example with ground-truth label y, we replace the label distribution with

$$q'(k|x) = (1-\epsilon)q(k|x) + \epsilon u(k)$$

Network	Top-1 Error	Top-5 Error	Cost Bn Ops
GoogLeNet [20]	29%	9.2%	1.5
BN-GoogLeNet	26.8%	-	**1.5**
BN-Inception [7]	25.2%	7.8	2.0
Inception-v2	23.4%	-	3.8
Inception-v2 RMSProp	23.1%	6.3	3.8
Inception-v2 Label Smoothing	22.8%	6.1	3.8
Inception-v2 Factorized 7×7	21.6%	5.8	4.8
Inception-v2 BN-auxiliary	**21.2%**	**5.6%**	4.8

CNN Architecture: Part II

❑ Inception v4

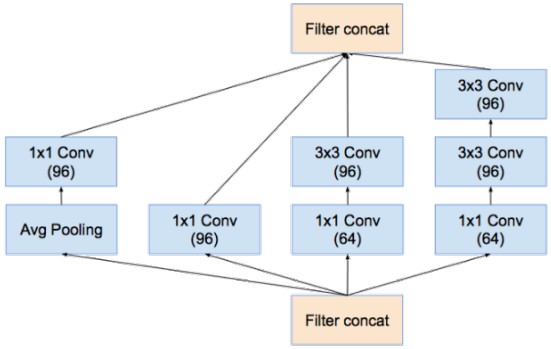


Figure 4. The schema for 35 × 35 grid modules of the pure Inception-v4 network. This is the Inception-A block of Figure 9.

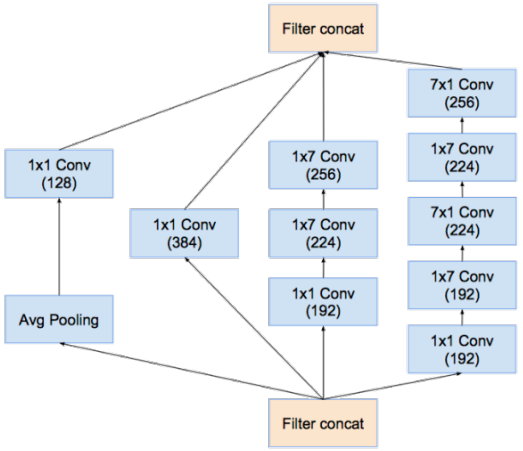


Figure 5. The schema for 17 × 17 grid modules of the pure Inception-v4 network. This is the Inception-B block of Figure 9.

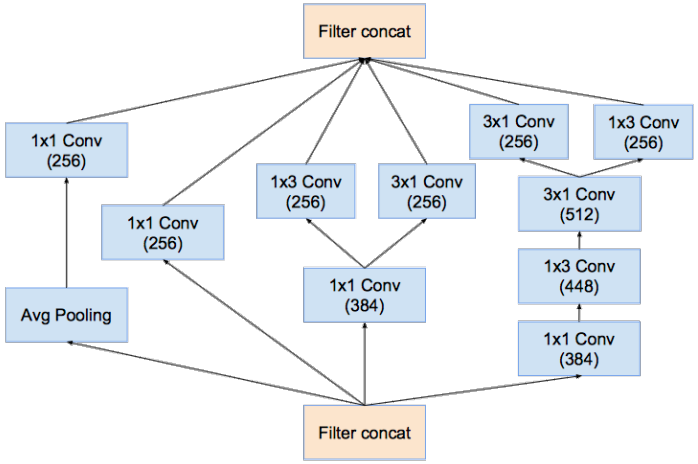
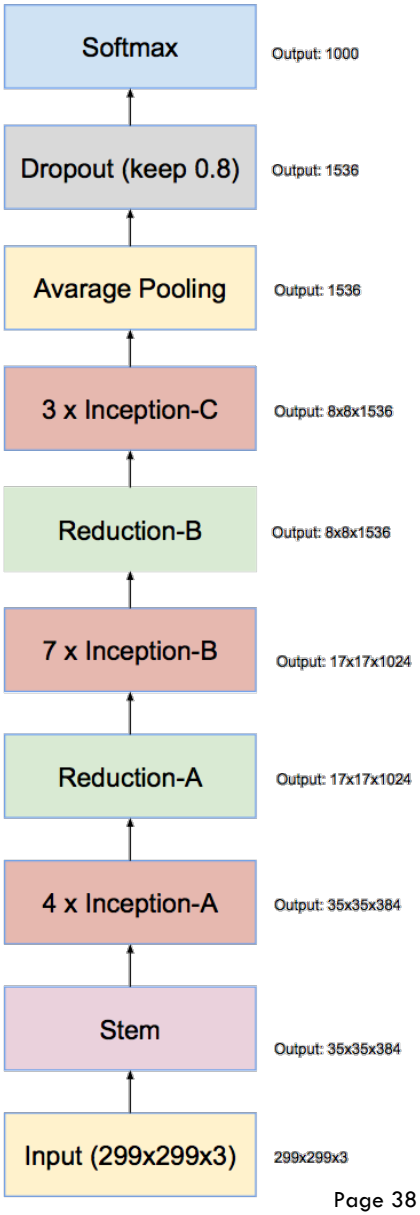


Figure 6. The schema for 8 × 8 grid modules of the pure Inception-v4 network. This is the Inception-C block of Figure 9.



CNN Architecture: Part II

❑ Inception-ResNet v1, v2

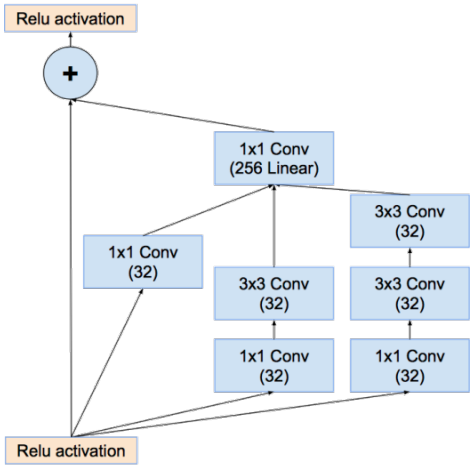


Figure 10. The schema for 35 × 35 grid (Inception-ResNet-A) module of Inception-ResNet-v1 network.

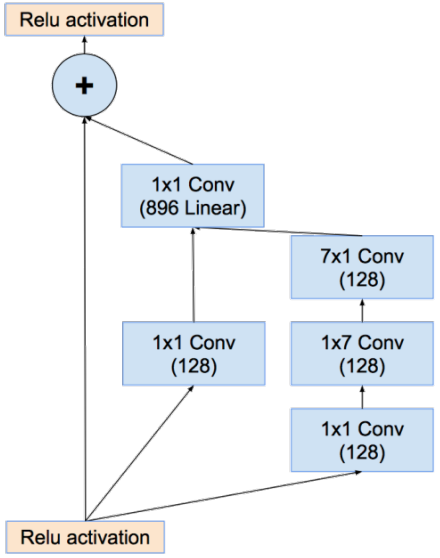


Figure 11. The schema for 17 × 17 grid (Inception-ResNet-B) module of Inception-ResNet-v1 network.

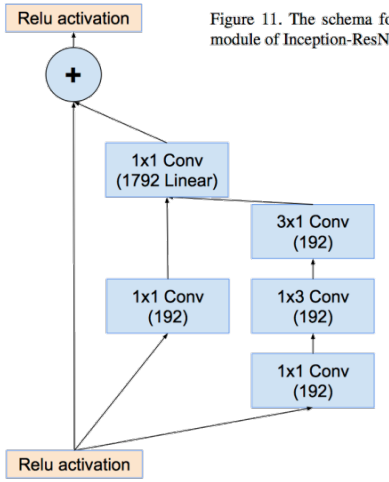
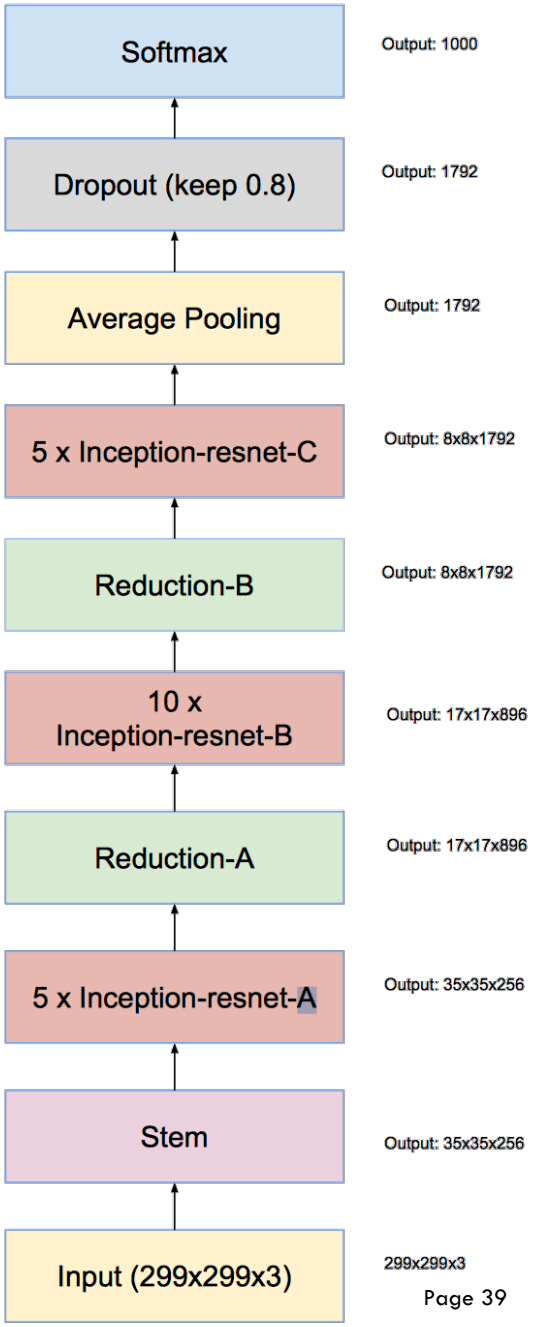


Figure 13. The schema for 8×8 grid (Inception-ResNet-C) module of Inception-ResNet-v1 network.



CNN Architecture: Part II

❑ Error on ImageNet-1000 classification

Network	Top-1 Error	Top-5 Error
BN-Inception [6]	25.2%	7.8%
Inception-v3 [15]	21.2%	5.6%
Inception-ResNet-v1	21.3%	5.5%
Inception-v4	20.0%	5.0%
Inception-ResNet-v2	19.9%	4.9%

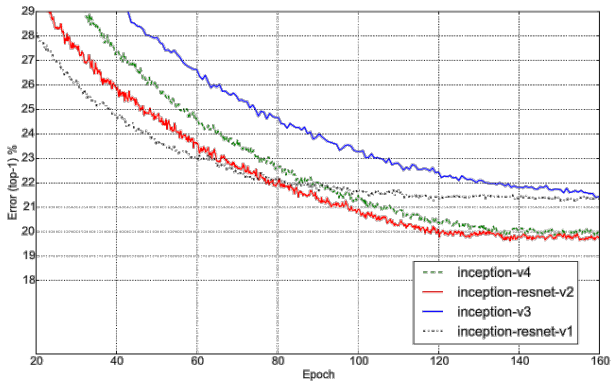


Figure 26. Top-1 error evolution of all four models (single model, single crop). This paints a similar picture as the top-5 evaluation.

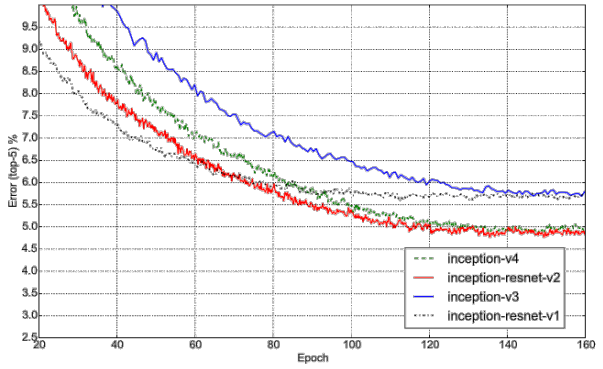


Figure 25. Top-5 error evolution of all four models (single model, single crop). Showing the improvement due to larger model size. Although the residual version converges faster, the final accuracy seems to mainly depend on the model size.

CNN Architecture: Part II

❑ Xception [Franc ,ois Chollet ,2017]

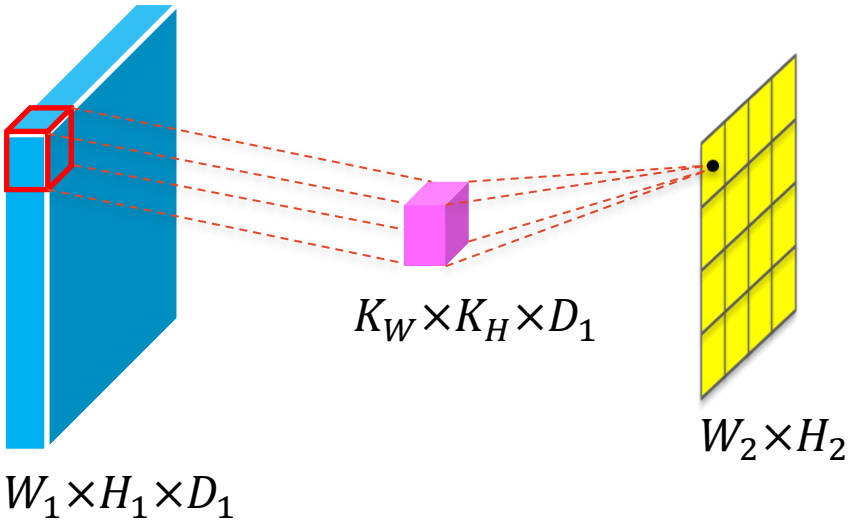


Figure 1. A canonical Inception module (Inception V3).

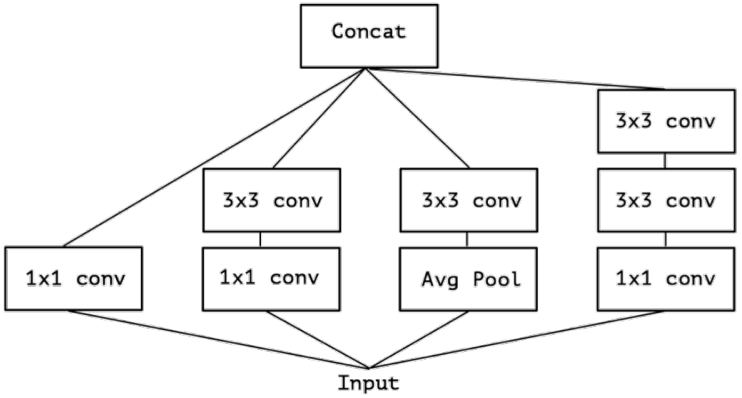
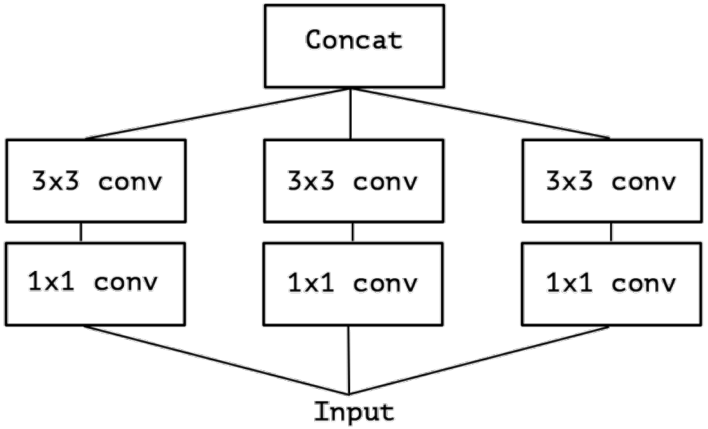


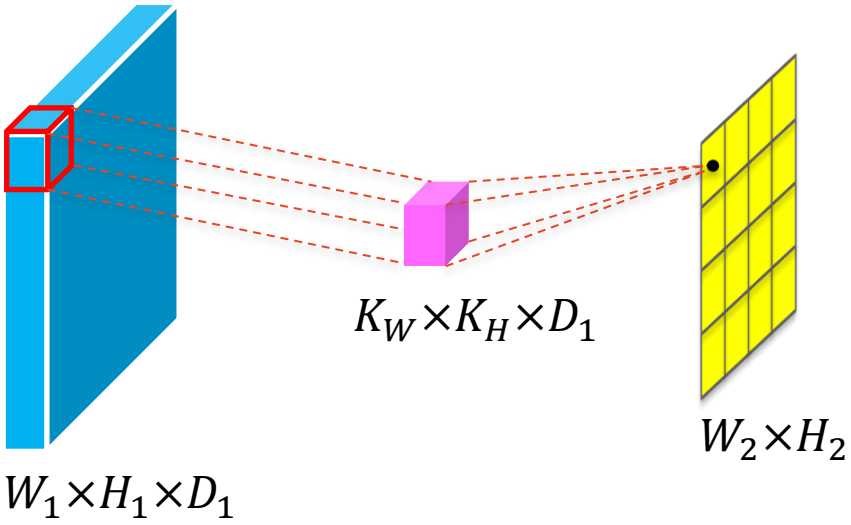
Figure 2. A simplified Inception module.



Inception module is to explicitly factorize it into a series of operations that would independently look at cross-channel correlations and at spatial correlations.

CNN Architecture: Part II

❑ Xception [Franc ,ois Chollet ,2017]



Assume that cross-channel correlations and spatial correlations can be mapped completely separately.

Figure 4. An "extreme" version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.

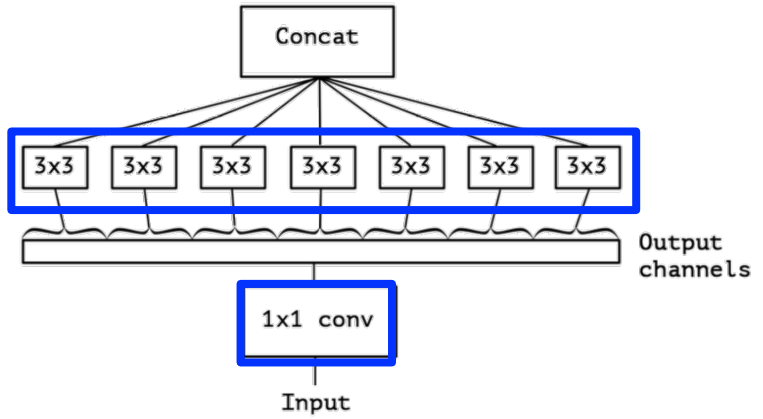
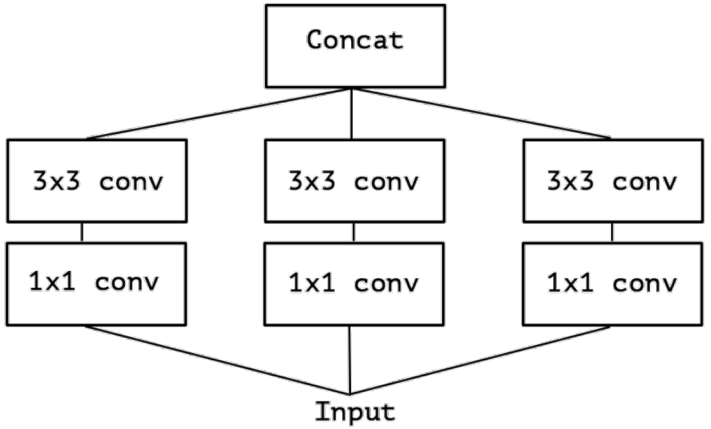


Figure 2. A simplified Inception module.

Inception module is to explicitly factorize it into a series of operations that would independently look at cross-channel correlations and at spatial correlations.



CNN Architecture: Part II

❑ Wide ResNet (WRNs) [Zagoruyko et al. 2016]

A percent of improved accuracy costs nearly doubling the number of layers.

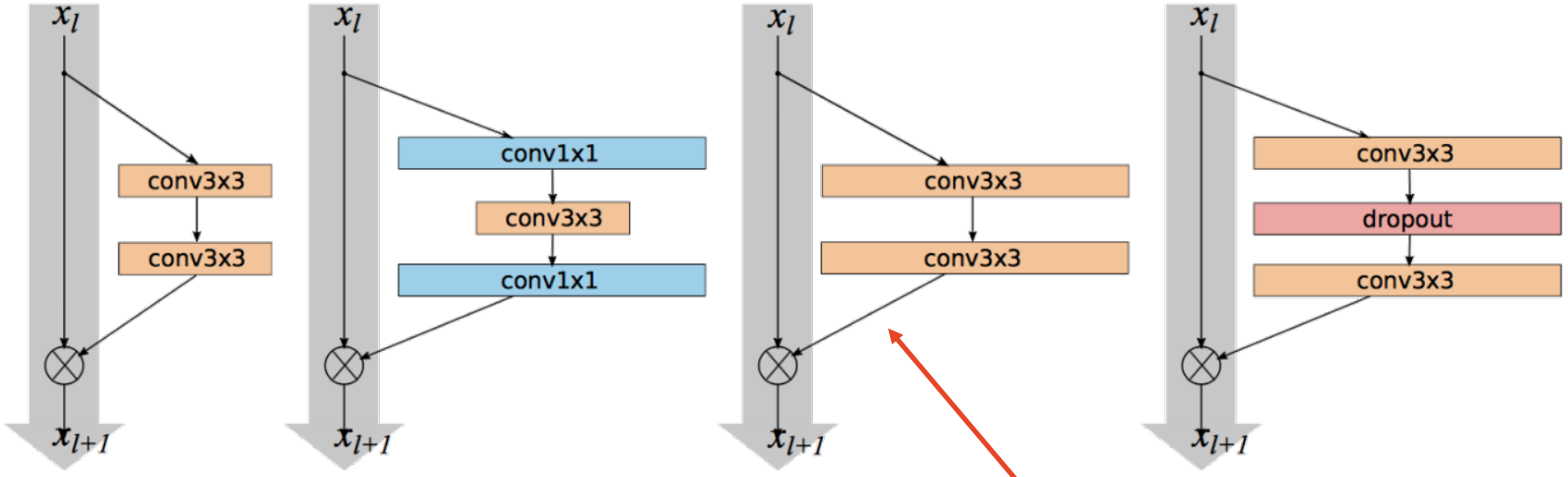
model	top-1	top-5
ResNet-50	22.9%	6.7%
ResNet-101	21.8%	6.1%
ResNet-152	21.4%	5.7%

trade-off: decrease depth and increase width of residual networks

CNN Architecture: Part II

❑ Wide ResNet (WRNs) [Zagoruyko et al. 2016]

- Residuals are important, not depth
- Increasing width instead of depth more computationally efficient



- more convolutional layers per block
- more feature planes

CNN Architecture: Part II

❑ Wide ResNet (WRNs) [Zagoruyko et al. 2016]

Model	top-1 err, %	top-5 err, %	#params	time/batch 16
ResNet-50	24.01	7.02	25.6M	49
ResNet-101	22.44	6.21	44.5M	82
ResNet-152	22.16	6.16	60.2M	115
WRN-50-2-bottleneck	21.9	6.03	68.9M	93

50-layer wide ResNet outperforms 152-layer original ResNet

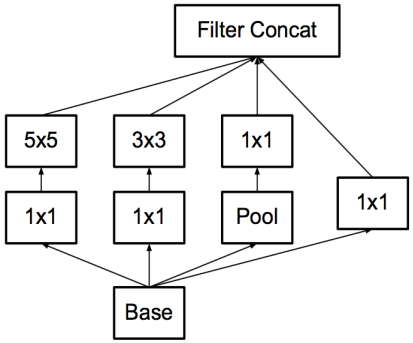
CNN Architecture: Part II

❑ ResNeXt [Xie et al. 2016]

Repeating a building block that aggregates a set of transformations with the same topology.

- Stack building blocks of the same shape, e.g. VGG and ResNets.

- Split-transform-merge strategy. In an Inception module, the input is split into a few lower-dimensional embeddings (by 1×1 convolutions), transformed by a set of specialized filters (3×3, 5×5, etc.), and merged by concatenation.



Inception model

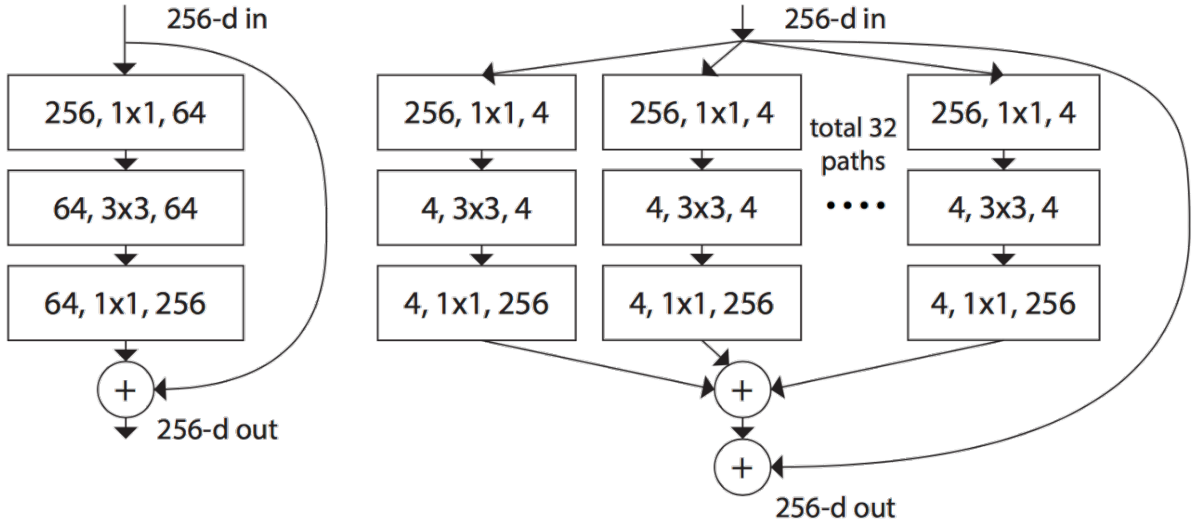


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

CNN Architecture: Part II

- ResNeXt [Xie et al. 2016]

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

Similar complexity

ResNet = ResNeXt

CNN Architecture: Part II

- ResNeXt [Xie et al. 2016]

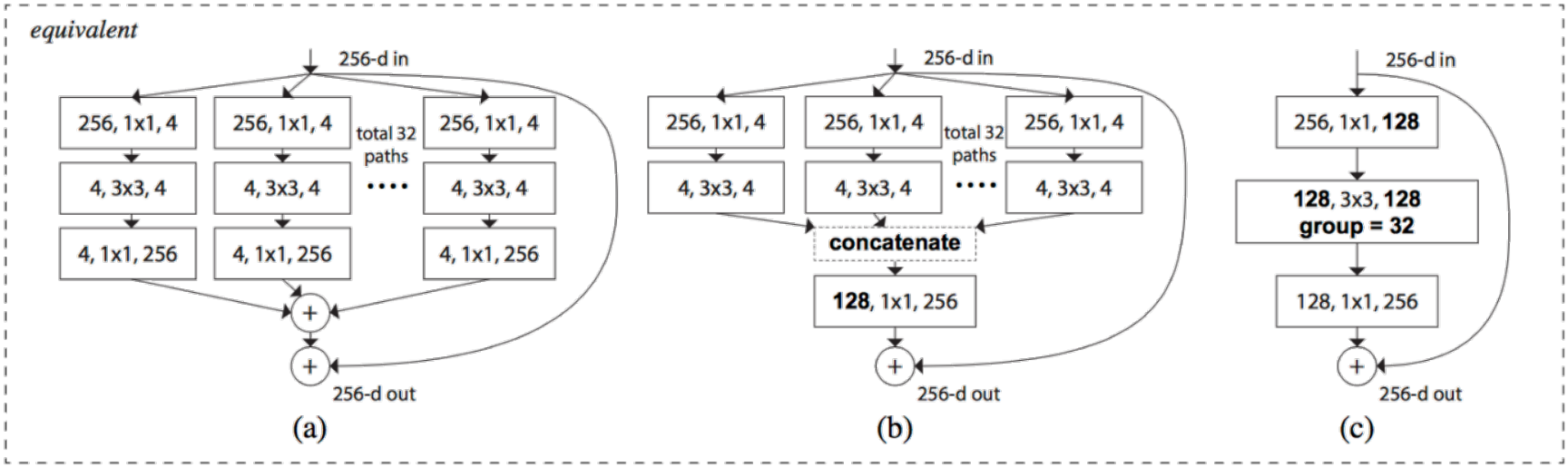
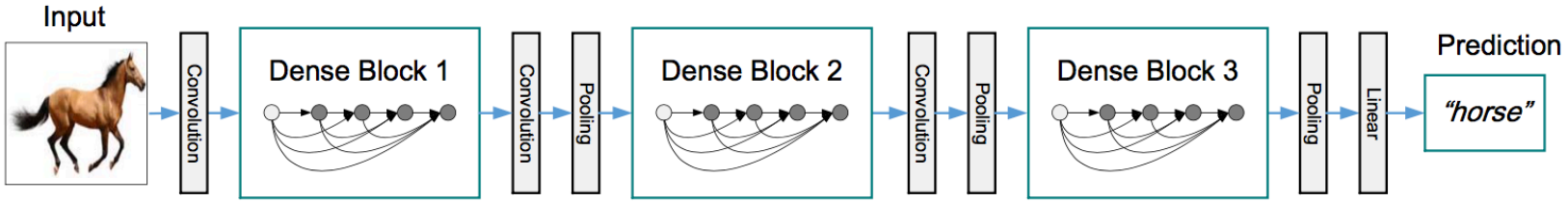


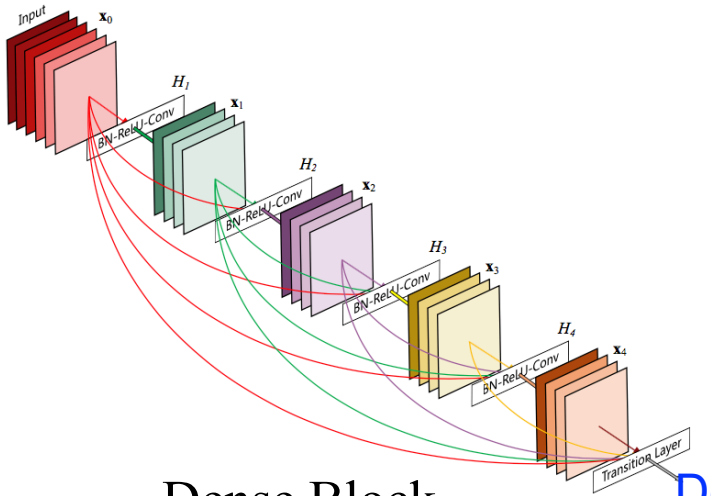
Figure 3. Equivalent building blocks of ResNeXt. **(a)**: Aggregated residual transformations, the same as Fig. 1 right. **(b)**: A block equivalent to (a), implemented as early concatenation. **(c)**: A block equivalent to (a,b), implemented as grouped convolutions [24]. Notations in **bold** text highlight the reformulation changes. A layer is denoted as (# input channels, filter size, # output channels).

CNN Architecture: Part II

❑ DenseNet [Huang et al. 2017]

Each layer has direct access to the gradients from the loss function and the original input signal, leading to an implicit deep supervision.





Dense Block

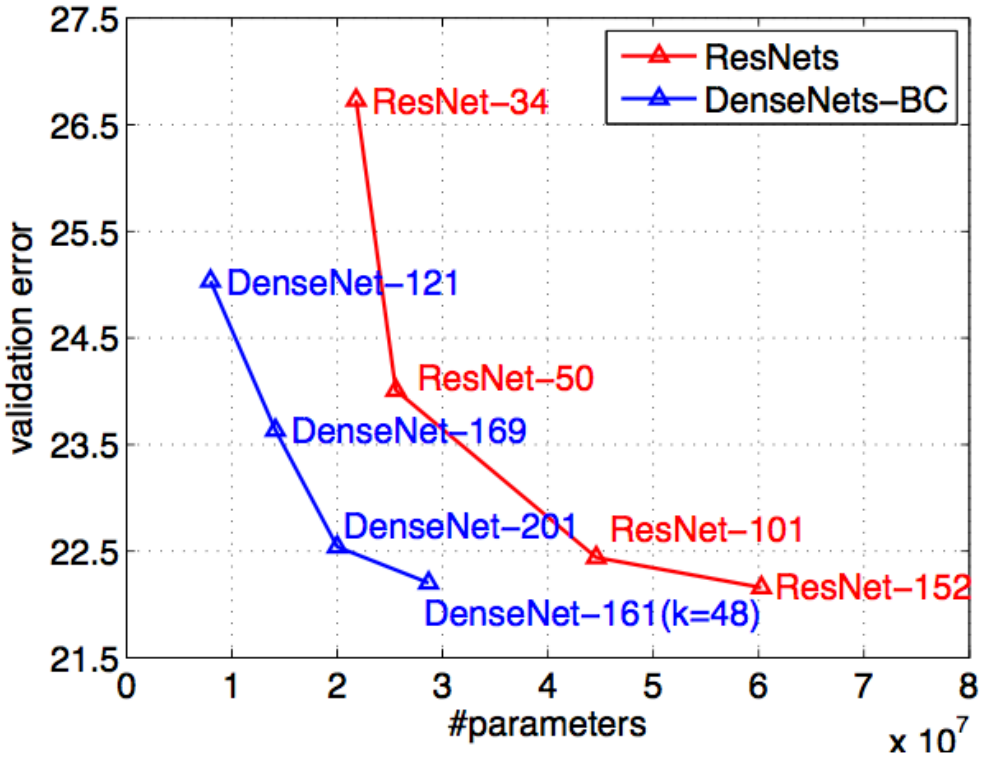
- alleviate the vanishing-gradient problem
- strengthen feature propagation
- encourage feature reuse
- reduce the number of parameters

ResNets: $\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1}$

DenseNets: $\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}])$

CNN Architecture: Part II

- **DenseNet** [Huang et al. 2017]



Comparison of the DenseNet and ResNet on model size