

CS181: Computer Vision

Popular CNNs (part#2)

(VGG, ResNet)

Dissecting CNNs

Tuesday, September 15th, 2026



Md Alimoor Reza

Associate Professor of Computer Science

Review

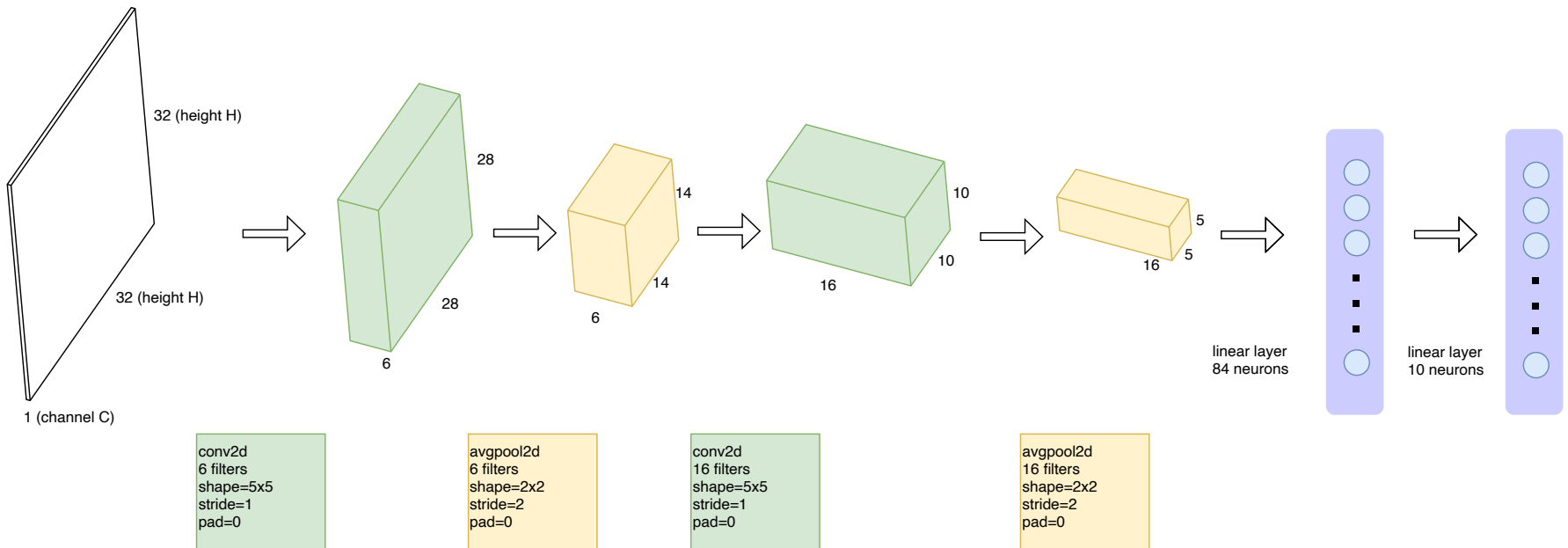
- Popular CNNs
 - LeNet
 - AlexNet

Review: LeNet implementation

- Go to Blackboard and let's work on the notebook titled "CNN with activation and pooling layers". Link shown below:

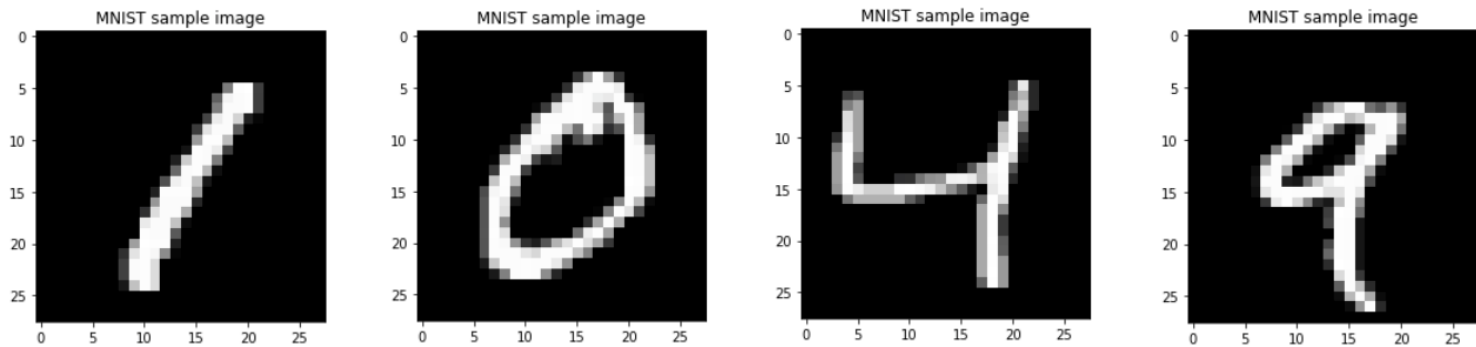
https://github.com/alimoorreza/CS181-fall26-notes/blob/main/day07_popular_cnn1_lenet_implementation.ipynb

LeNet Architecture



Review: LeNet insufficient for real-world images

- LeNet is a simple CNN architecture suitable for well-structured image
 - e.g., 28x28 pixels image of digits from 0 to 9 in MNIST or our Fashion-MNIST dataset



- Real-world images are much more complicated; pose challenges in classification
 - e.g., high resolution images 600x480 pixels image and contents have a lot more diversity



Review: ImageNet Challenge

- **ImageNet dataset**

- 14 million labeled images
- 20k classes

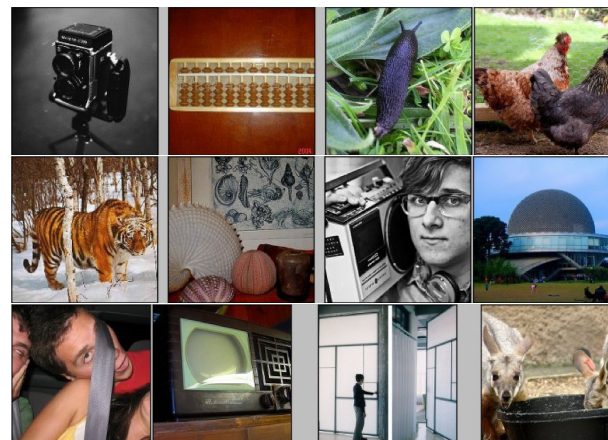


- **Data source**

- Images gathered from Internet

- **Image annotators**

- Human labels via Amazon Turk



Project led by Fei-Fei Li at Stanford
CVPR 2009

- **What is ImageNet challenge**

- Train your network with [a subset of 1.2 million training images](#) from ImageNet
- Test your network using another testing subset where you need to [classify each image to one of the 1000 classes](#)

Review: Popular CNN: AlexNet

- AlexNet's important features. Similar framework to LeNet with important distinction such as:

- Bigger model compared to LeNet
 - 5 convolution layers + 3 linear layers
 - 60,000,000 params
- Trained on more data
 - 10^6 images (ImageNet) vs. 10^3 images (digit images)
- GPU implementation (50x speedup over CPU)
 - trained on two GPUs for a week
- Better regularization for training
 - introduced a new technique called **Dropout**
 - **Dropout**: randomly turning off neurons from the network

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

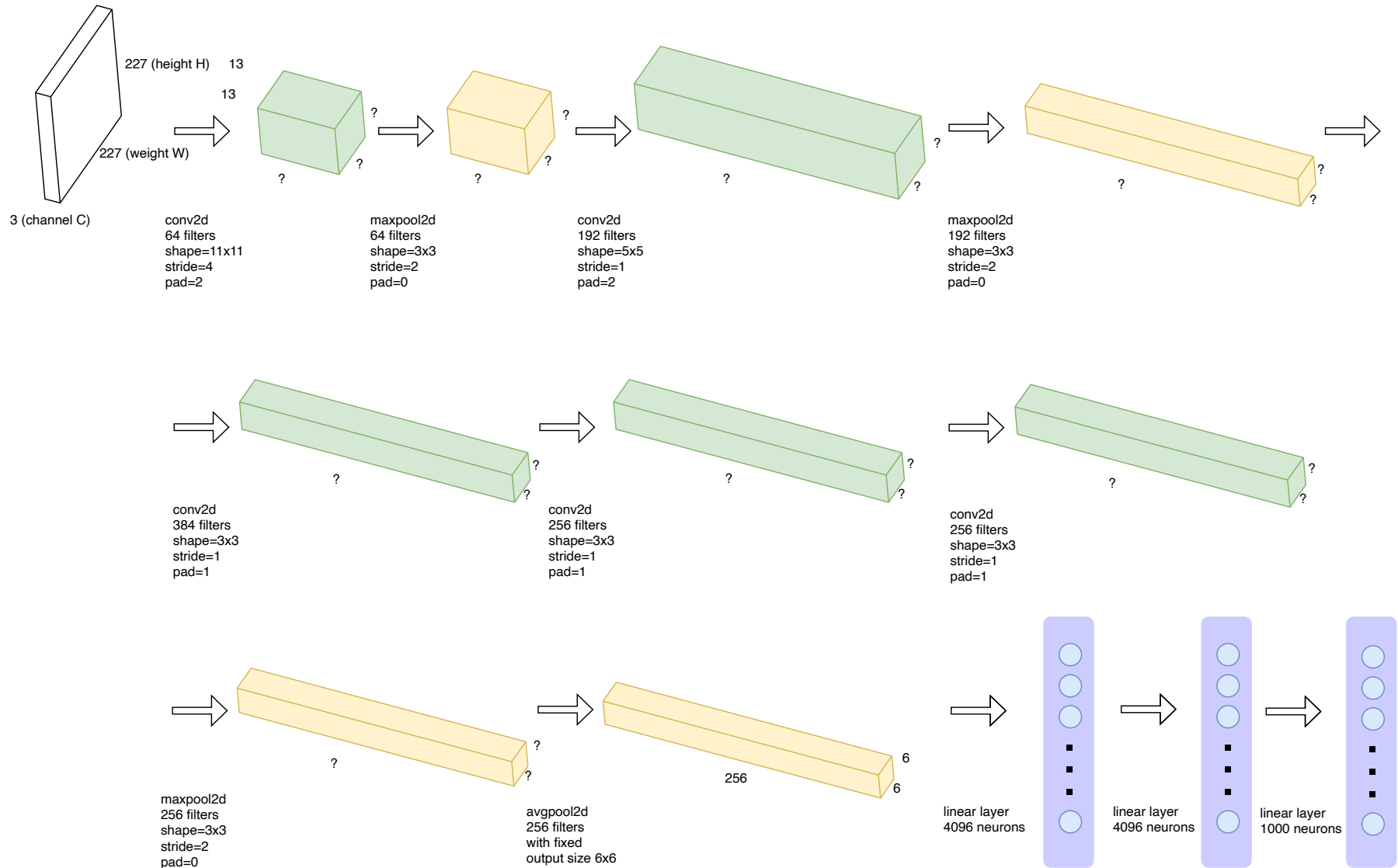
Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet ILSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

Review: Group Activity

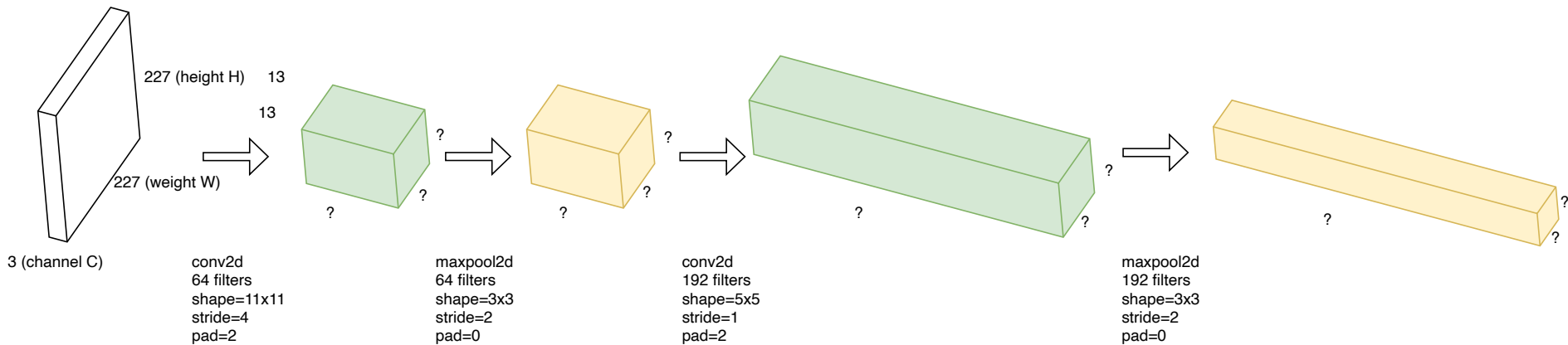
What are the dimensions of different volumes?



A. Krizhevsky, I. Sutskever, and G. Hinton, [ImageNet Classification with Deep Convolutional Neural Networks](#), NIPS 2012

Review: Group Activity

What are the dimensions of different volumes?



Convolution output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = K \text{ ***number of filters***}$$

Pooling output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1 \text{ ***input volume's channel***}$$

CONV1
 $W = \frac{(227 - 11 + 4)}{4} + 1 = 55 + 1 = 56$

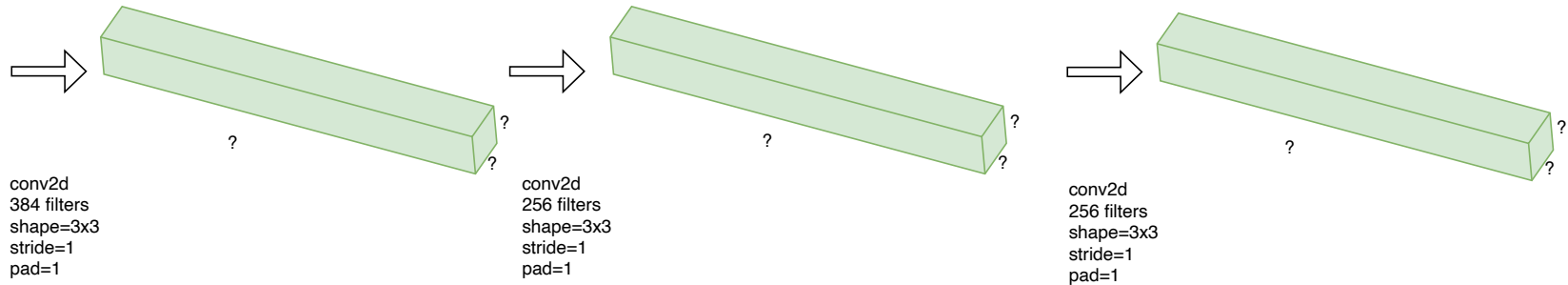
POOL1(max)
 $W = \frac{(56 - 3 + 0)}{2} + 1 = 26 + 1 = 27$

CONV2
 $W = \frac{(27 - 5 + 4)}{1} + 1 = 26 + 1 = 27$

POOL2(max)
 $W = \frac{(27 - 3 + 0)}{2} + 1 = 12 + 1 = 13$

Review: Group Activity

What are the dimensions of different volumes?



Convolution output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = K \text{ ***number of filters***}$$

Pooling output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1 \text{ ***input volume's channel***}$$

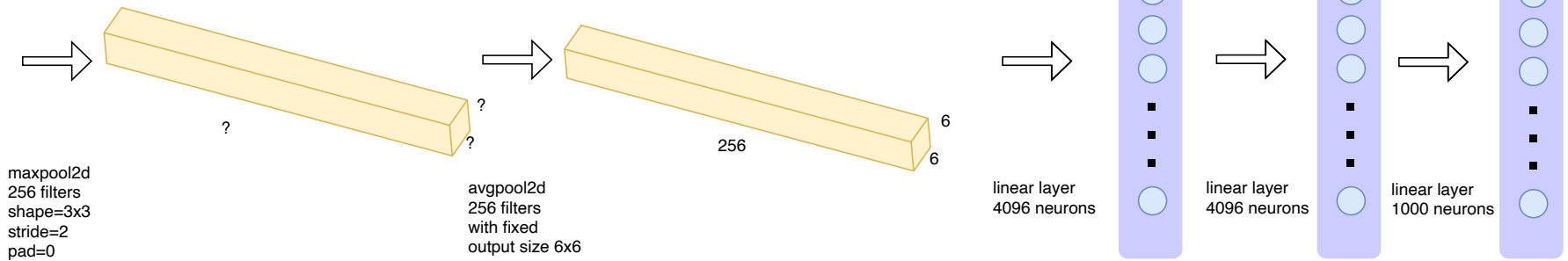
conv3
 $W = \frac{(13 - 3 + 2)}{1} + 1 = 12 + 1 = 13$

conv4
 $W = \frac{(13 - 3 + 2)}{1} + 1 = 12 + 1 = 13$

conv5
 $W = \frac{(13 - 3 + 2)}{1} + 1 = 12 + 1 = 13$

Review: Group Activity

What are the dimensions of different volumes?



Convolution output volume ($C_2 \times H_2 \times W_2$)	Pooling output volume ($C_2 \times H_2 \times W_2$)
$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$	$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$
$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$	$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$
$C_2 = K \text{ ***number of filters***}$	$C_2 = C_1 \text{ ***input volume's channel***}$

pool 3 (max)
 $W = \frac{(13 - 3 + 0)}{2} + 1 = 5 + 1 = 6$

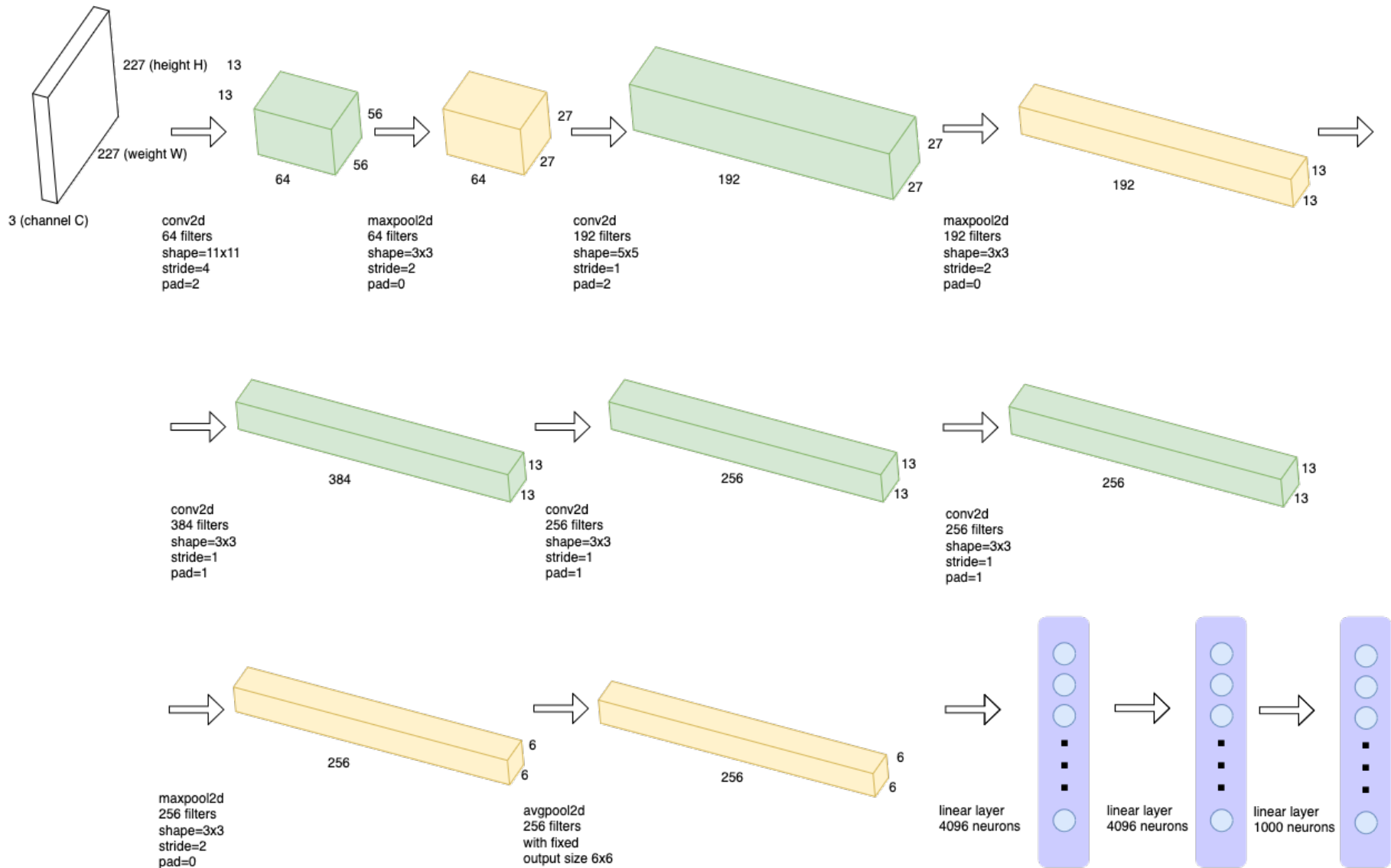
pool 4 (avg)
 $W = \text{FIXED OUTPUT DIMENSION OF } 6$

linear 1
 $(256 * 6 * 6 * 4096)$

linear 2
 $(4096 * 4096)$

linear 3
 $(4096 * 1000)$

Review: Full solution (dimensions of different volumes)



A. Krizhevsky, I. Sutskever, and G. Hinton, [ImageNet Classification with Deep Convolutional Neural Networks](#), NIPS 2012

Group Activity: Inference with AlexNet

https://github.com/alimoorreza/CS181-fall26-notes/blob/main/day08_popular_cnn2_alexnet_dissection.ipynb

Review: Popular CNN: VGG-16

- VGG-16 was the winner of ImageNet (1000-class image classification) challenge in 2014
 - proposed by Oxford University (Andrew Zisserman's group)

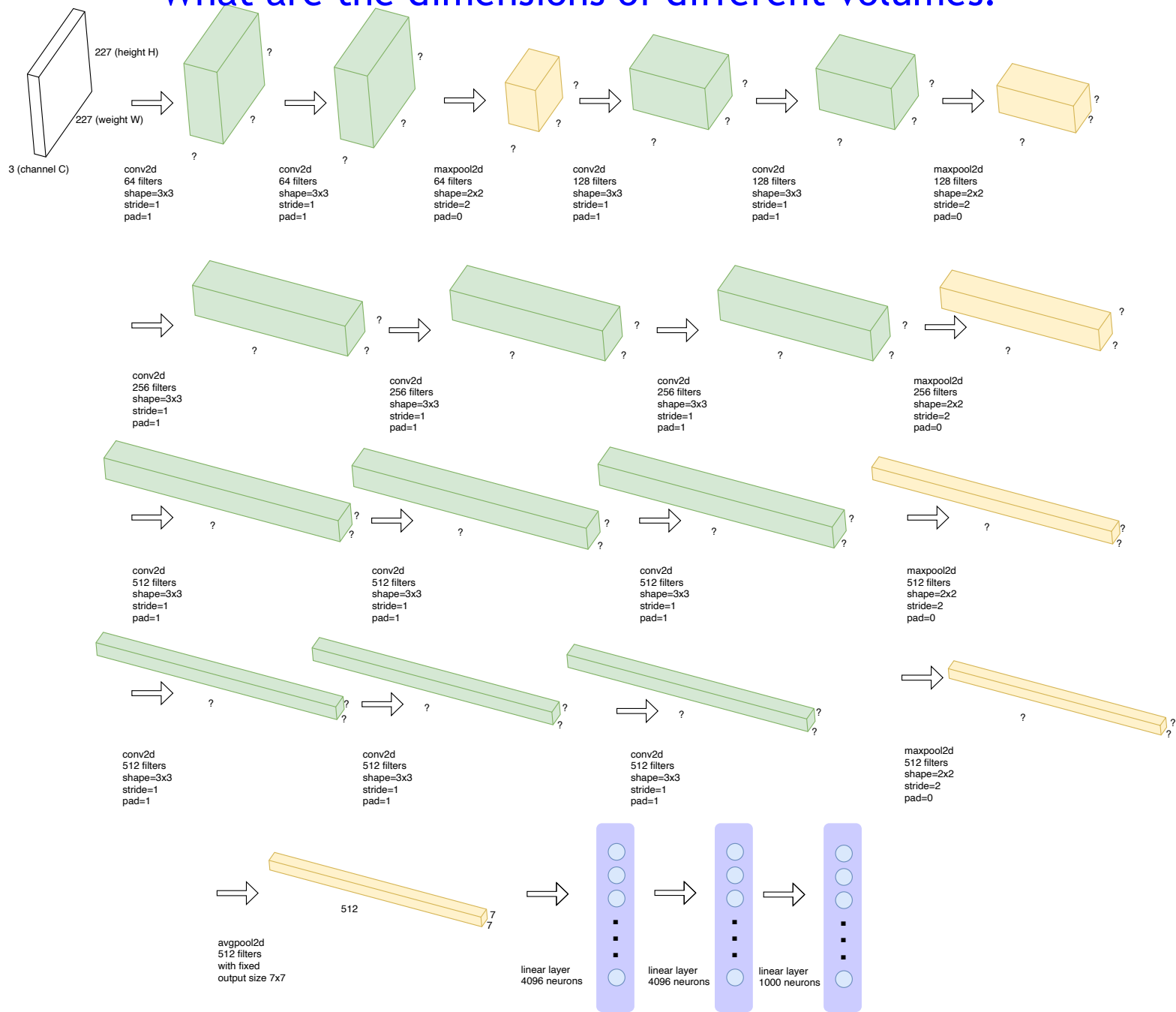


Input image



Review: Full solution: Group Activity

What are the dimensions of different volumes?



Topics

- Popular CNNs
 - VGG dissection
 - ResNet dissection

Popular CNN: VGG-16

- VGG-16's important features.
- VGG was the winner of ImageNet challenge in 2014:

Published as a conference paper at ICLR 2015

- Bigger model compared to AlexNet
 - vgg16: 13 convolutional layers + 3 linear layers
 - vgg19: 16 convolutional layers + 3 linear layers
 - last 3 linear layers are the same as AlexNet
 - approximately 140,000,000 params

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

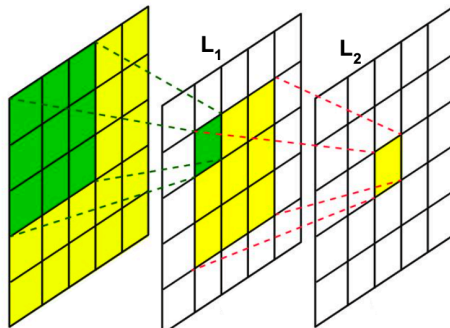
Karen Simonyan* & Andrew Zisserman*

Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

ABSTRACT

In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively. We also show that our representations generalise well to other datasets, where they achieve state-of-the-art results. We have made our two best-performing ConvNet models publicly available to facilitate further research on the use of deep visual representations in computer vision.

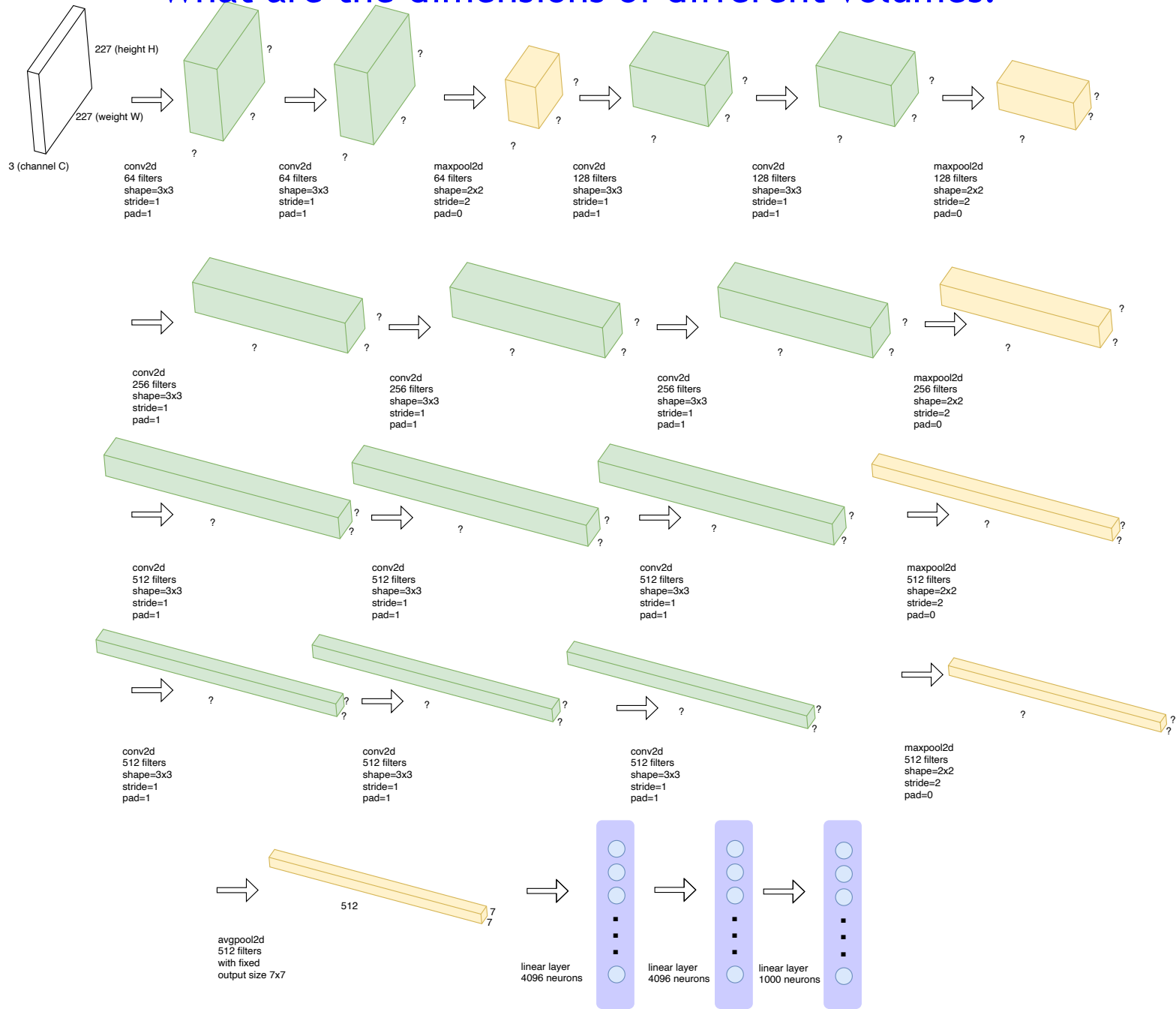
- Stacking 3x3 convolution filters:
 - fixed 3x3 convolution filters but many of them
 - within a sublayer, multiple 3x3 convolution filters are stacked together
 - stacking them together reduces the number of parameters needed to cover similar fields of view



Karen Simonyan and Andrew Zisserman, [Very Deep Convolutional Networks for Large-Scale Image Recognition](#), ICLR 2015

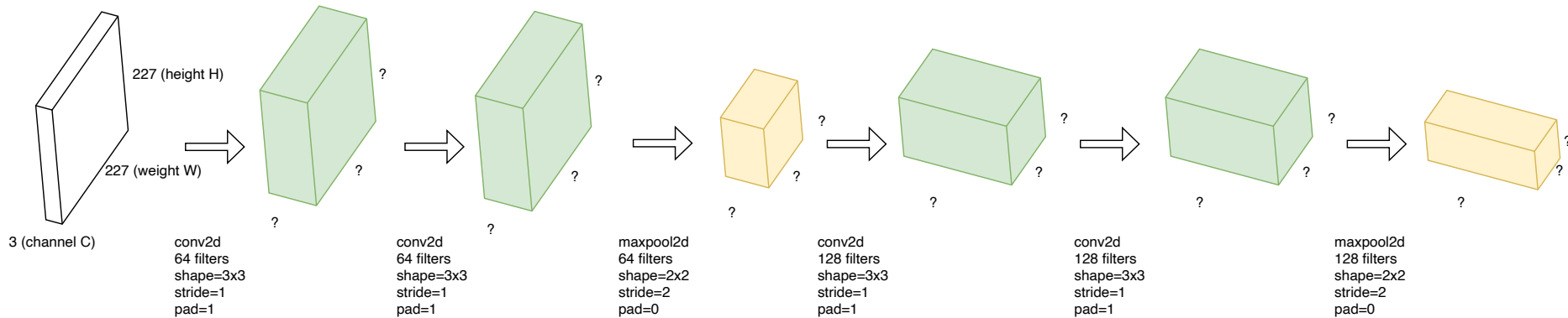
Group Activity

What are the dimensions of different volumes?



Solution: Group Activity

What are the dimensions of different volumes?



Convolution output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = K \text{ ***number of filters***}$$

Pooling output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1 \text{ ***input volume's channel***}$$

Conv#1

$$W = (227 - 3 + 2 \cdot 1) / 1 + 1 = 227$$

Conv#2

$$W = (227 - 3 + 2 \cdot 1) / 1 + 1 = 227$$

Pooling#1

$$W = (227 - 2 + 2 \cdot 0) / 2 + 1 = 113$$

Conv#3

$$W = (113 - 3 + 2 \cdot 1) / 1 + 1 = 113$$

Conv#4

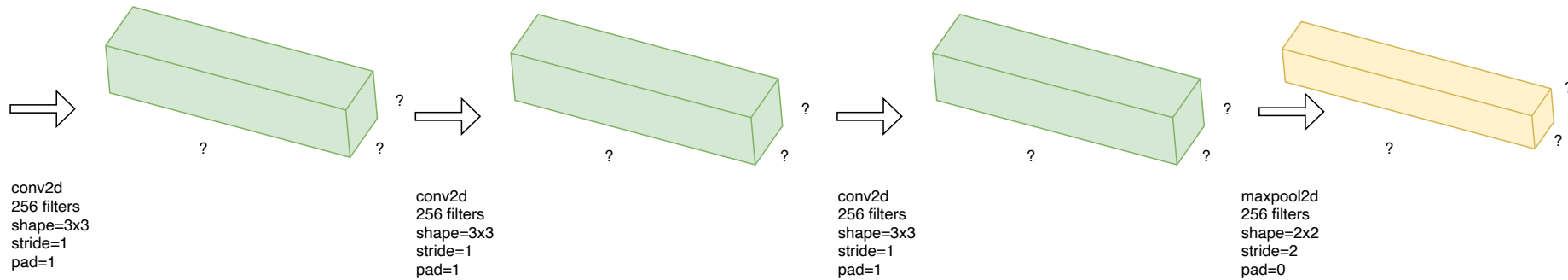
$$W = (113 - 3 + 2 \cdot 1) / 1 + 1 = 113$$

Pooling#2

$$W = (113 - 2 + 2 \cdot 0) / 2 + 1 = 56$$

Solution: Group Activity

What are the dimensions of different volumes?



Convolution output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = K \text{ ***number of filters***}$$

Pooling output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1 \text{ ***input volume's channel***}$$

Conv#5

$$W = (56 - 3 + 2 \cdot 1) / 1 + 1 = 56$$

Conv#6

$$W = (56 - 3 + 2 \cdot 1) / 1 + 1 = 56$$

Conv#7

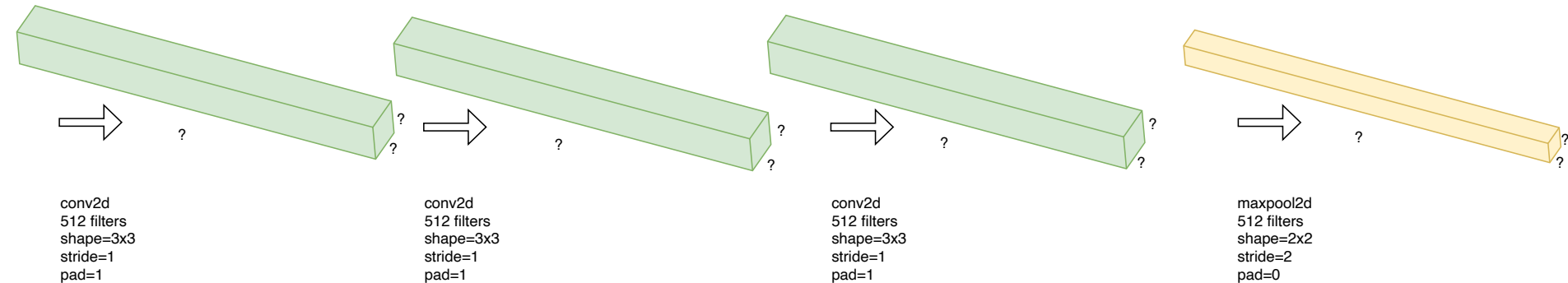
$$W = (56 - 3 + 2 \cdot 1) / 1 + 1 = 56$$

Pooling#3

$$W = (56 - 2 + 2 \cdot 0) / 2 + 1 = 28$$

Solution: Group Activity

What are the dimensions of different volumes?



Convolution output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = K \text{ ***number of filters***}$$

Pooling output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1 \text{ ***input volume's channel***}$$

Conv#5

$$W = (28 - 3 + 2 \cdot 1) / 1 + 1 = 28$$

Conv#6

$$W = (28 - 3 + 2 \cdot 1) / 1 + 1 = 28$$

Conv#7

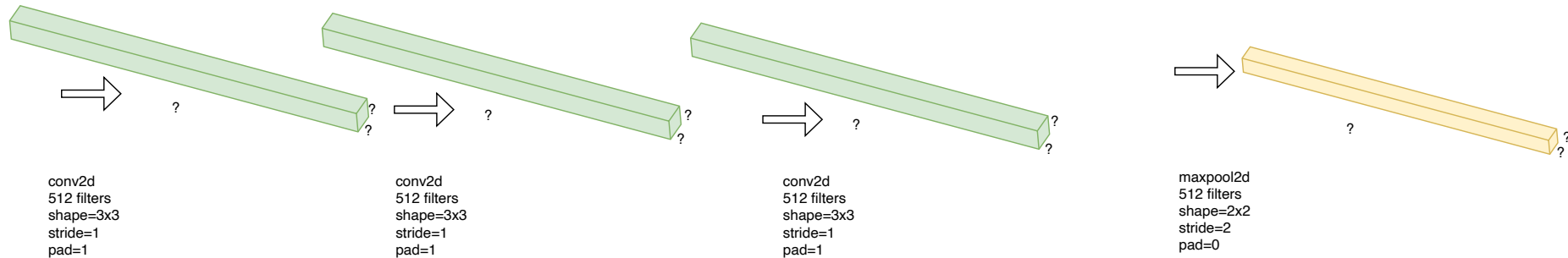
$$W = (28 - 3 + 2 \cdot 1) / 1 + 1 = 28$$

Pooling#3

$$W = (28 - 2 + 2 \cdot 0) / 2 + 1 = 14$$

Solution: Group Activity

What are the dimensions of different volumes?



Convolution output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = K \text{ ***number of filters***}$$

Pooling output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = C_1 \text{ ***input volume's channel***}$$

Conv#11

$$W = (14 - 3 + 2 \cdot 1) / 1 + 1 = 14$$

Conv#12

$$W = (14 - 3 + 2 \cdot 1) / 1 + 1 = 14$$

Conv#13

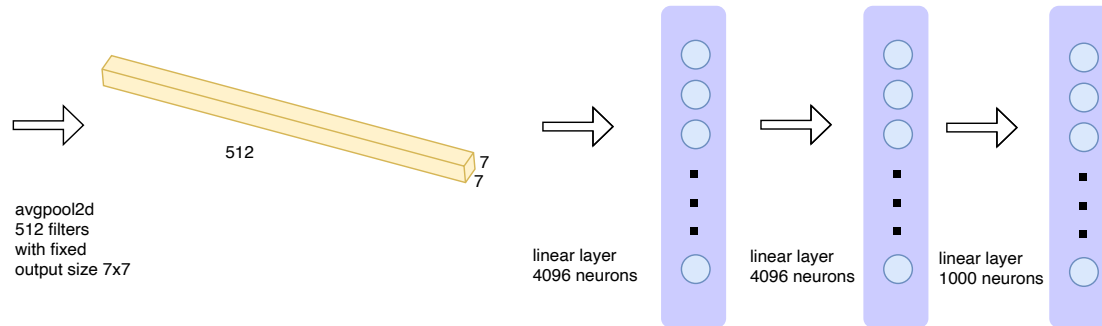
$$W = (14 - 3 + 2 \cdot 1) / 1 + 1 = 14$$

Pooling#5

$$W = (14 - 2 + 2 \cdot 0) / 2 + 1 = 7$$

Solution: Group Activity

What are the dimensions of different volumes?



Connections between pooling and linear layer

$$(C_2 \times H_2 \times W_2) \times 4096 \\ = (512 \times 7 \times 7) \times 4096$$

input volume's channel

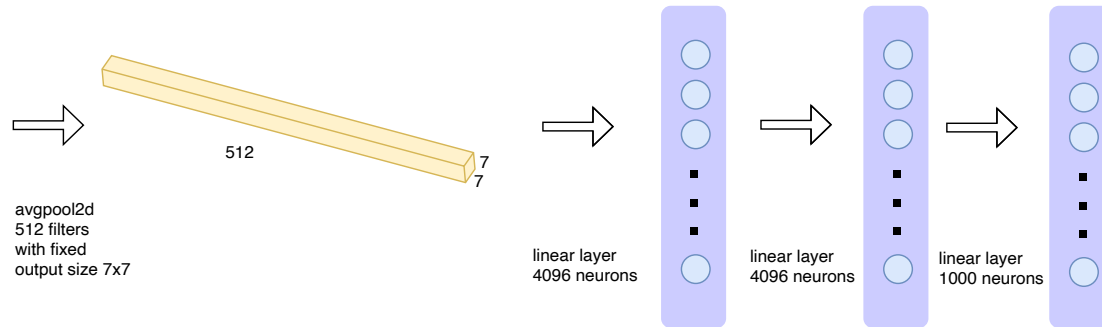
input volume's height

input volume's width

output number of neurons

Solution: Group Activity

What are the dimensions of different volumes?



Connections between linear
and linear layers

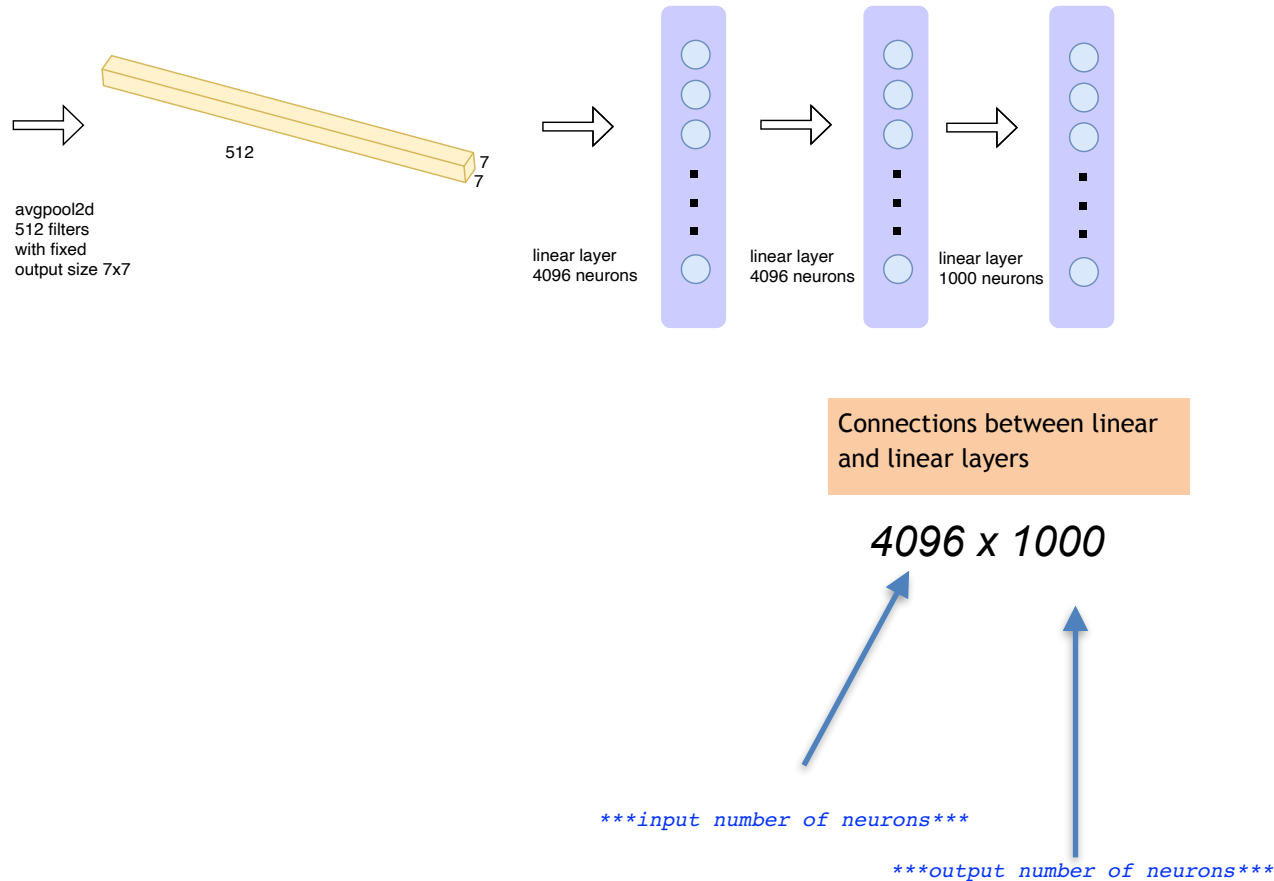
4096×4096

input number of neurons

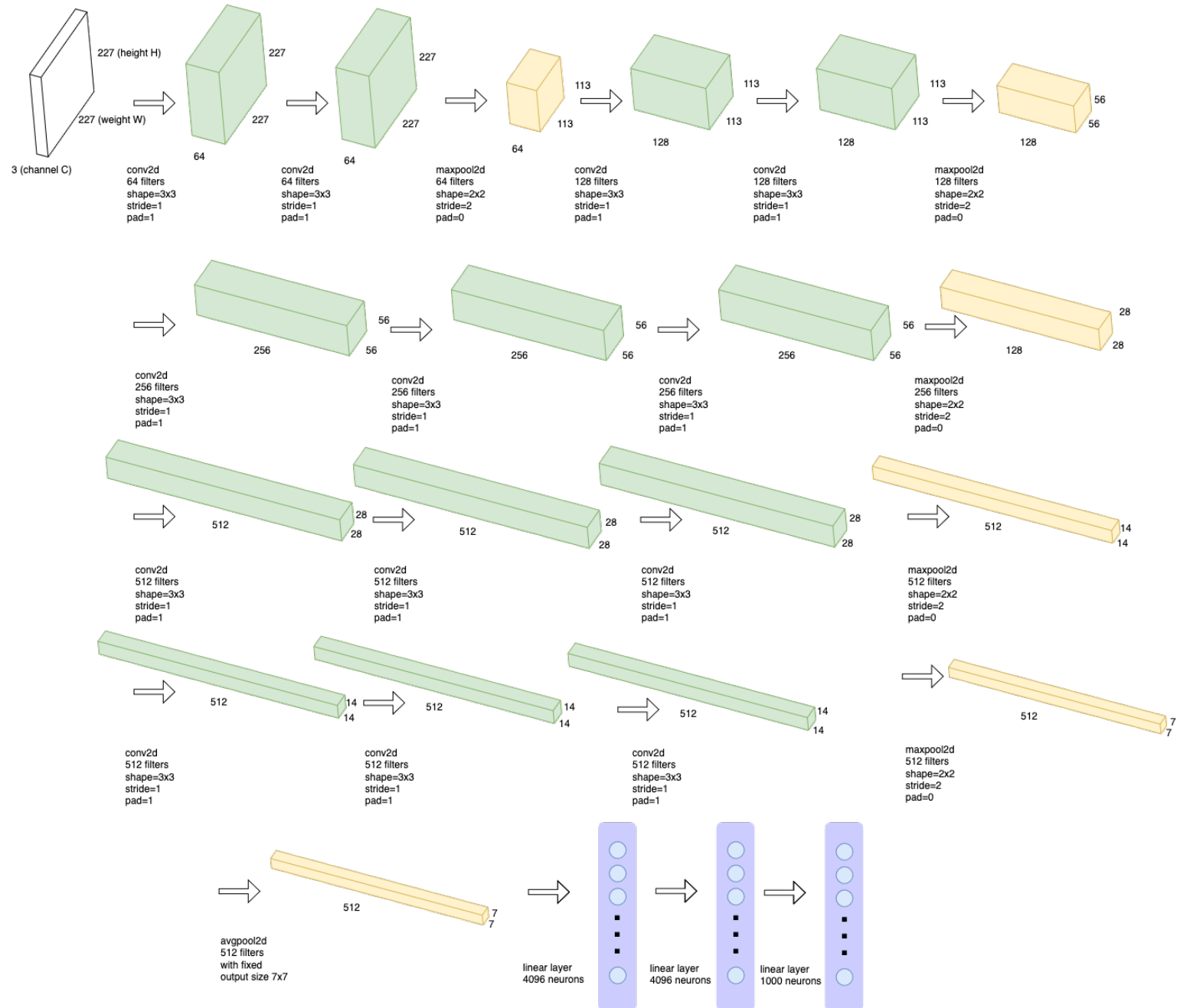
output number of neurons

Solution: Group Activity

What are the dimensions of different volumes?



Full solution (the dimensions of different volumes)



Popular CNN: VGG-16

```
[torch.Size([3, 227, 227])] INPUT
[torch.Size([64, 227, 227])] CONV1_1: 64 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([64, 227, 227])] CONV1_2: 64 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([64, 113, 113])] MAX POOL1: 2x2 max pooling filters with stride=2, pad=0
[torch.Size([128, 113, 113])] CONV2_1: 128 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([128, 113, 113])] CONV2_2: 128 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([128, 56, 56])] MAX POOL2: 2x2 max pooling filters with stride=2, pad=0
[torch.Size([256, 56, 56])] CONV3_1: 256 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([256, 56, 56])] CONV3_2: 256 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([256, 56, 56])] CONV3_3: 256 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([256, 28, 28])] MAX POOL3: 2x2 max pooling filters with stride=2, pad=0
[torch.Size([512, 28, 28])] CONV4_1: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 28, 28])] CONV4_2: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 28, 28])] CONV4_3: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 14, 14])] MAX POOL4: 2x2 max pooling filters with stride=2, pad=0
[torch.Size([512, 14, 14])] CONV5_1: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 14, 14])] CONV5_2: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 14, 14])] CONV5_3: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 7, 7])] MAX POOL5: 2x2 max pooling filters with stride=2, pad=0
[torch.Size([512, 7, 7])] AVG POOL: fixed output size [width x height]
[torch.Size([4096])] FC6: neurons
[torch.Size([4096])] FC7: neurons
[torch.Size([1000])] FC8: neurons
```

Group Activity: Inference with VGG-16

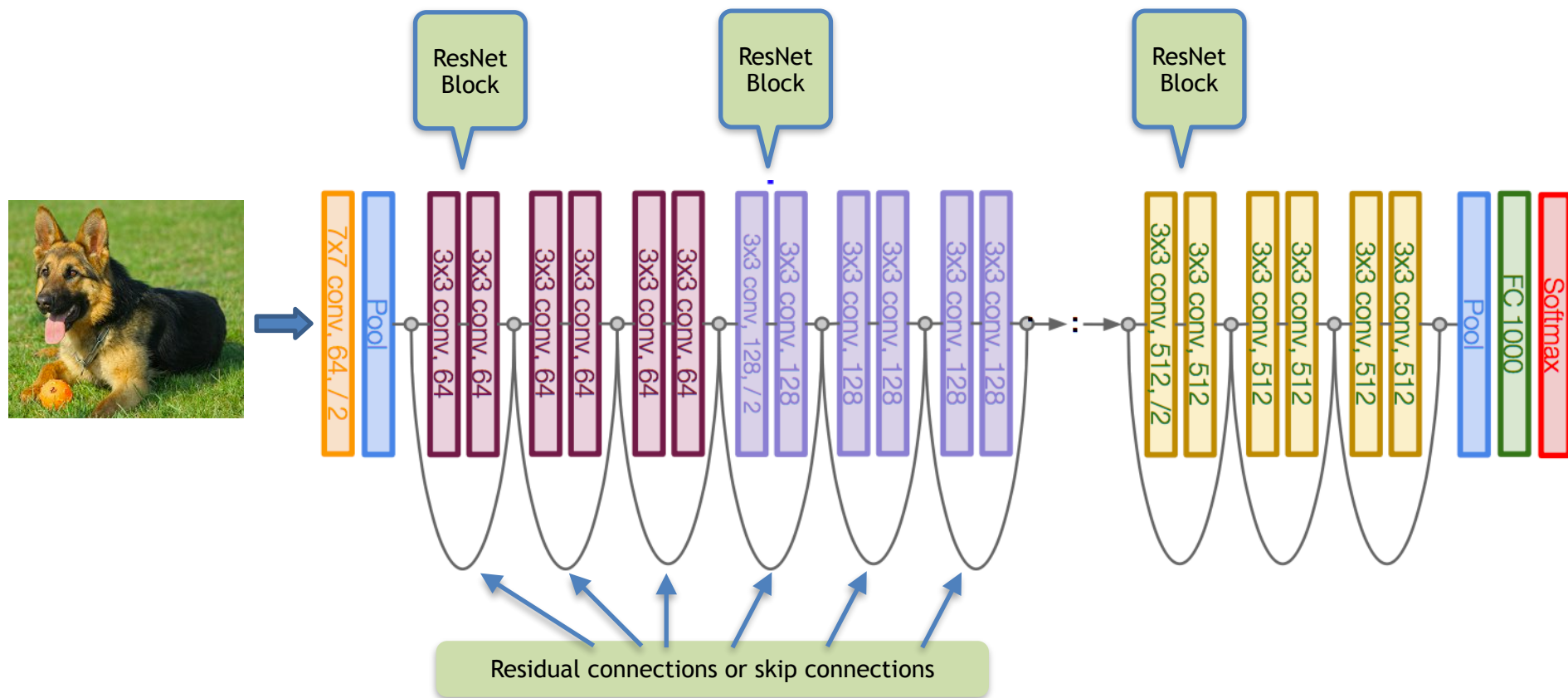
https://github.com/alimoorreza/CS181-fall26-notes/blob/main/day08_popular_cnn3_vgg16_dissection.ipynb

Topics

- Popular CNNs
 - VGG dissection
 - ResNet dissection

Popular CNN: ResNet

- ResNet was the winner of the ImageNet Challenge in 2015. It is a CNN composed of several ResNet blocks connected sequentially as shown below.

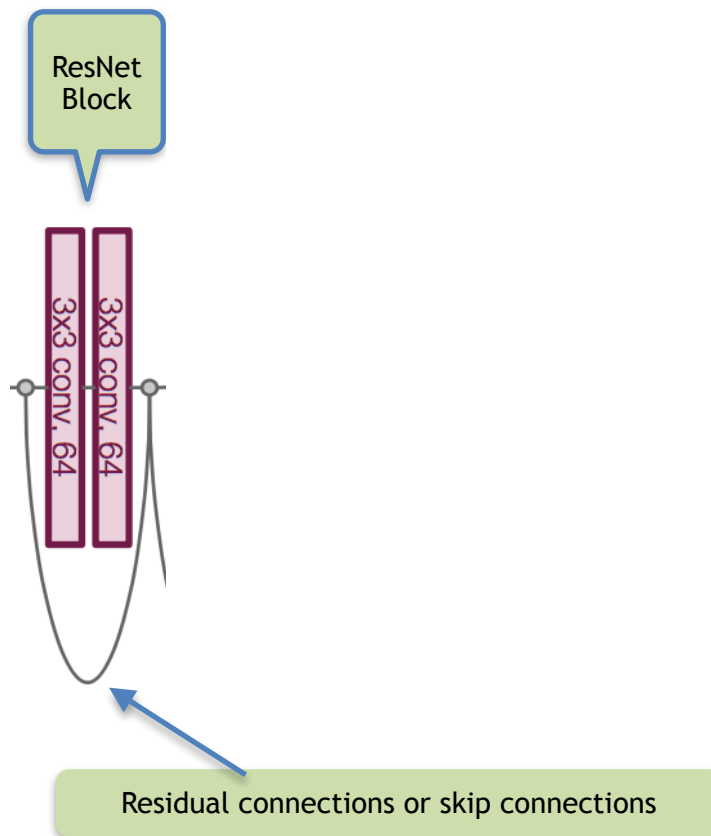


Many names but they refer to the same thing

Kaiming He et al., [Deep Residual Learning for Image Recognition](#), CVPR 2015

What is a ResNet block?

- Let's consider the first ResNet block in the network. It has 2 convolutional layers connected sequentially. There is also a skip connection that connects the input directly to the output.

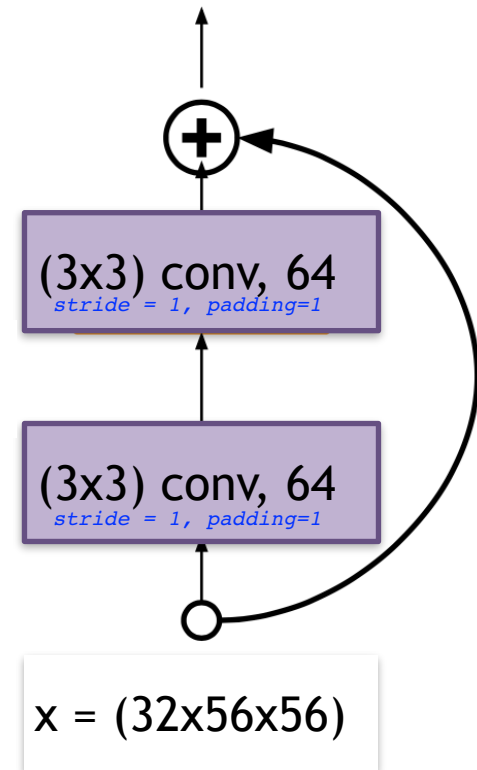


Many names but they refer to the same thing

Kaiming He et al., [Deep Residual Learning for Image Recognition](#), CVPR 2015

ResNet's main innovation residual block

- The ResNet block shown here, let's say, gets an input of size (32x56x56)



Convolution output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = K \text{ ***number of filters***}$$

Kaiming He et al., [Deep Residual Learning for Image Recognition](#), CVPR 2015

ResNet's main innovation residual block

- The ResNet block shown here, let's say, gets an input of size (32x56x56)
- After applying the first convolution inside the ResNet block, the output tensor's size will be (64x56x56)

Conv#1

$$W_2 = (56 - 3 + 2 \cdot 1) / 1 + 1 = 56$$

$$C_2 = 64$$

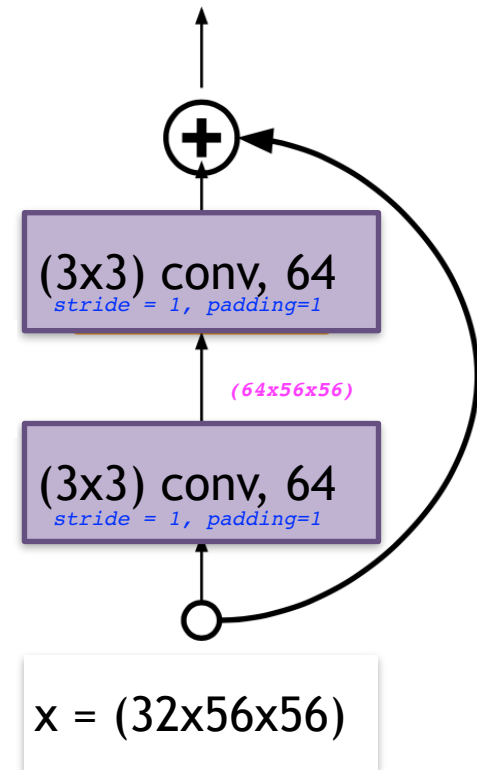
(64x56x56)

Convolution output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = K \text{ ***number of filters***}$$



ResNet's main innovation residual block

- The ResNet block shown here, let's say, gets an input of size (32x56x56)
- After applying the first convolution inside the ResNet block, the output tensor's size will be (64x56x56)

Conv#1

$$W_2 = (56 - 3 + 2 \cdot 1) / 1 + 1 = 56$$

$$C_2 = 64$$

- After applying the second convolution inside the ResNet block, the output tensor's size will be (64x56x56)

Conv#1

$$W_2 = (56 - 3 + 2 \cdot 1) / 1 + 1 = 56$$

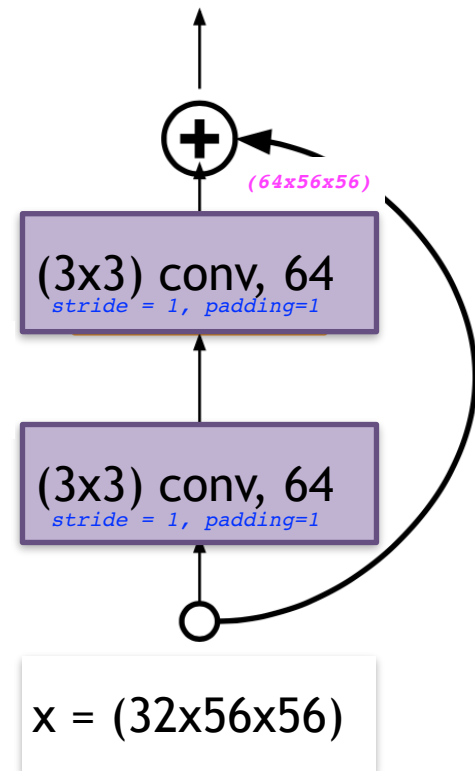
$$C_2 = 64$$

Convolution output volume
($C_2 \times H_2 \times W_2$)

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$C_2 = K \text{ ***number of filters***}$$



ResNet's main innovation is residual block

PyTorch
implementation calls it
downsampling with
1x1 convolution

- The ResNet block shown here, let's say, gets an input of size (32x56x56)

- After applying the first convolution inside the ResNet block, the output tensor's size will be (64x56x56)

Conv#1

$$W_2 = (56 - 3 + 2 \cdot 1) / 1 + 1 = 56$$

$$C_2 = 64$$

- After applying the second convolution inside the ResNet block, the output tensor's size will be (64x56x56)

Conv#2

$$W_2 = (56 - 3 + 2 \cdot 1) / 1 + 1 = 56$$

$$C_2 = 64$$

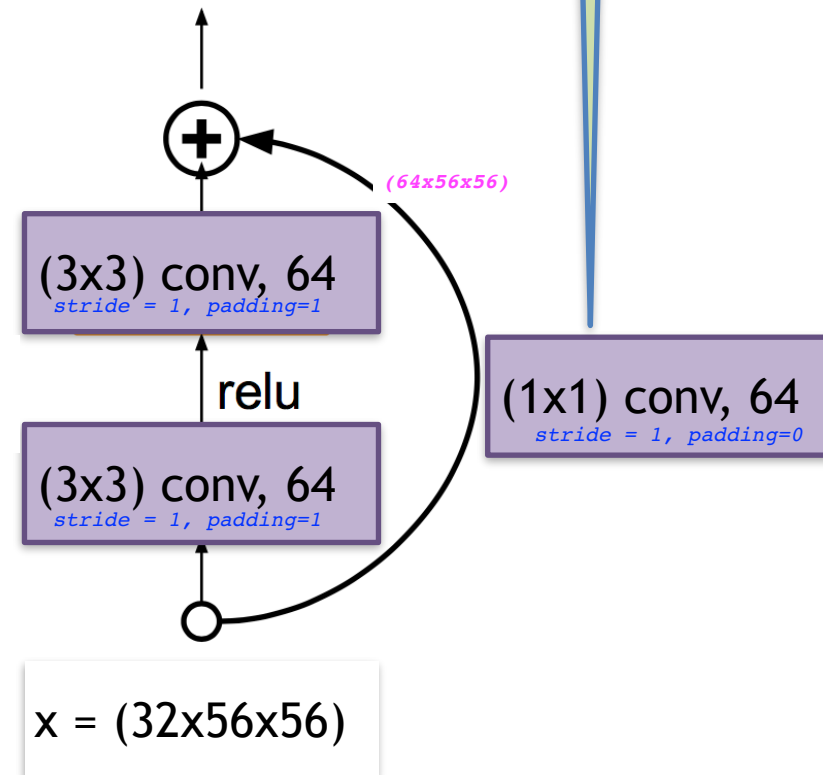
- Now, the ResNet block applies a **convolution operation** with 64 filters, each with a size of (1x1), to the input. The output of this operation produces a tensor of size (64x56x56).

Downsampling with conv

$$W_2 = (56 - 1 + 2 \cdot 0) / 1 + 1 = 56$$

$$C_2 = 64$$

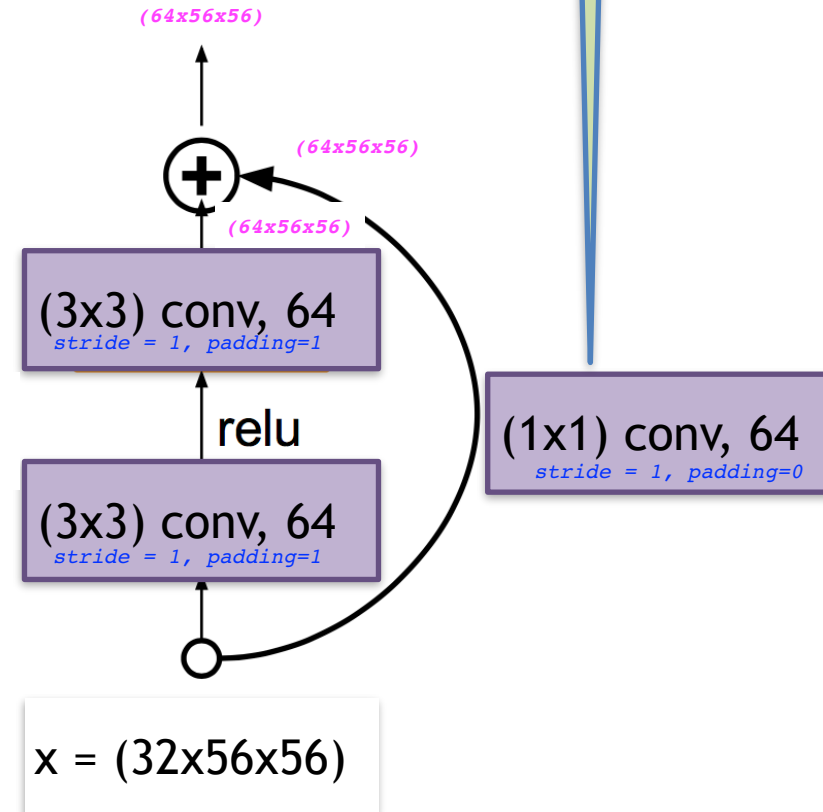
(64x56x56)



Kaiming He et al., [Deep Residual Learning for Image Recognition](#), CVPR 2015

ResNet's main innovation is residual block

- Finally, the ResNet block adds two tensors, each with a size of $(64 \times 56 \times 56)$ with an element-wise matrix operation. The final output of ResNet block is a tensor of size $(64 \times 56 \times 56)$. This will be applied as an input to the next ResNet block



ResNet-18's Full Architecture



Check out the notebook for details

layer name	output size	18-layer
conv1	112×112	
conv_stage1	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv_stage2	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv_stage3	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv_stage4	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
	1×1	
FLOPs		1.8×10^9

1 convolutional layer

Stage#1: 2 [convolutions](#) x 2 [resnet-block](#) = 4 [convolutions](#)
 each resnet block has [2 convolution layers](#) with 64 filters of size (3x3)

Stage#2: 2 [convolutions](#) x 2 [resnet-block](#) = 4 [convolutions](#)
 each resnet block has [2 convolution layers](#) with 128 filters of size (3x3)

Stage#3: 2 [convolutions](#) x 2 [resnet-block](#) = 4 [convolutions](#)
 each resnet block has [2 convolution layers](#) with 256 filters of size (3x3)

Stage#4: 2 [convolutions](#) x 2 [resnet-block](#) = 4 [convolutions](#)
 each resnet block has [2 convolution layers](#) with 512 filters of size (3x3)

1 linear layer

In total, there are $(1 + 4 + 4 + 4 + 4) = 17$ [convolutional layers](#) In addition, there is 1 [linear layer](#) at the end

In total, the number of learnable components: $17+1 = 18$

That's why it is called ResNet18

Kaiming He et al., [Deep Residual Learning for Image Recognition](#), CVPR 2015

ResNet-34's Full Architecture



Check out the notebook for details

layer name	output size	34-layer
conv1	112×112	
conv_stage1	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$
conv_stage2	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$
conv_stage3	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$
conv_stage4	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$
	1×1	avg
FLOPs		3.6×10^9

1 convolutional layer

Stage#1: 2 [convolutions](#) x 3 [resnet-block](#) = 6 [convolutions](#)

each resnet block has [2 convolution layers](#) with 64 filters of size (3x3)

Stage#2: 2 [convolutions](#) x 4 [resnet-block](#) = 8 [convolutions](#)

each resnet block has [2 convolution layers](#) with 128 filters of size (3x3)

Stage#3: 2 [convolutions](#) x 6 [resnet-block](#) = 12 [convolutions](#)

each resnet block has [2 convolution layers](#) with 256 filters of size (3x3)

Stage#4: 2 [convolutions](#) x 3 [resnet-block](#) = 6 [convolutions](#)

each resnet block has [2 convolution layers](#) with 512 filters of size (3x3)

1 linear layer

In total, there are $(1 + 6 + 8 + 12 + 6) = 33$ [convolutional layers](#)

In addition, there is 1 [linear layer](#) at the end

In total, the number of learnable components: $33+1 = 34$

That's why it is called ResNet34

ResNet-50's Full Architecture



Check out the notebook for details

layer name	output size	50-layer
conv1	112×112	7×7, 64, stride 2
		3×3 max pool, stride 2
conv_stage1	56×56	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv_stage2	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv_stage3	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
conv_stage4	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc,
FLOPs		3.8×10^9

1 convolutional layer

Stage#1: 3 convolutions x 3 resnet-block = 9 convolutions

each resnet block has 3 convolution layers with varying filter of sizes

Stage#2: 3 convolutions x 4 resnet-block = 12 convolutions

each resnet block has 3 convolution layers with varying filter of sizes

Stage#3: 3 convolutions x 6 resnet-block = 18 convolutions

each resnet block has 3 convolution layers with varying filter of sizes

Stage#4: 3 convolutions x 3 resnet-block = 9 convolutions

each resnet block has 3 convolution layers with varying filter of sizes

1 linear layer

In total, there are $(1 + 9 + 12 + 18 + 9) = 49$ convolutional layers

In addition, there is 1 linear layer at the end

In total, the number of learnable components: $49+1 = 50$

That's why it is called ResNet50

Kaiming He et al., [Deep Residual Learning for Image Recognition](#), CVPR

ResNet-101's Full Architecture



Check out the notebook for details

layer name	output size	101-layer
conv1	112×112	
		le 2
conv_stage1	56×56	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv_stage2	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv_stage3	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
conv_stage4	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	softmax
FLOPs		7.6×10^9

1 convolutional layer

Stage#1: 3 convolutions x 3 resnet-block = 9 convolutions

each resnet block has 3 convolution layers with varying filter of sizes

Stage#2: 3 convolutions x 4 resnet-block = 12 convolutions

each resnet block has 3 convolution layers with varying filter of sizes

Stage#3: 3 convolutions x 23 resnet-block = 69 convolutions

each resnet block has 3 convolution layers with varying filter of sizes

Stage#4: 3 convolutions x 3 resnet-block = 9 convolutions

each resnet block has 3 convolution layers with varying filter of sizes

1 linear layer

In total, there are $(1 + 9 + 12 + 69 + 9) = 100$ convolutional layers
In addition, there is 1 linear layer at the end

In total, the number of learnable components: $100+1 = 101$

That's why it is called ResNet101

Kaiming He et al., [Deep Residual Learning for Image Recognition](#), CVPR

ResNet-152's Full Architecture



Check out the notebook for details

layer name	output size	152-layer
conv1	112×112	
conv_stage1	56×56	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv_stage2	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv_stage3	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv_stage4	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	
FLOPs		11.3×10^9

1 convolutional layer

Stage#1: 3 [convolutions](#) x 3 [resent-block](#) = 9 [convolutions](#)

each resent block has 3 [convolution layers](#) with varying filter of sizes

Stage#2: 3 [convolutions](#) x 8 [resent-block](#) = 24 [convolutions](#)

each resent block has 3 [convolution layers](#) with varying filter of sizes

Stage#3: 3 [convolutions](#) x 36 [resent-block](#) = 108 [convolutions](#)

each resent block has 3 [convolution layers](#) with varying filter of sizes

Stage#4: 3 [convolutions](#) x 3 [resent-block](#) = 9 [convolutions](#)

each resent block has 3 [convolution layers](#) with varying filter of sizes

1 linear layer

In total, there are $(1 + 9 + 24 + 108 + 9) = 151$ [convolutional layers](#)
In addition, there is 1 [linear layer](#) at the end

In total, the number of learnable components: $151 + 1 = 152$

That's why it is called ResNet152

Kaiming He et al., [Deep Residual Learning for Image Recognition](#), CVPR

All ResNet versions



Check out the notebook
for details

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv_stage1	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times 3, 64 \\ 3\times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times 3, 64 \\ 3\times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 1, 256 \end{bmatrix} \times 3$
conv_stage2	28×28	$\begin{bmatrix} 3\times 3, 128 \\ 3\times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times 3, 128 \\ 3\times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128 \\ 1\times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128 \\ 1\times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128 \\ 1\times 1, 512 \end{bmatrix} \times 8$
conv_stage3	14×14	$\begin{bmatrix} 3\times 3, 256 \\ 3\times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times 3, 256 \\ 3\times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256 \\ 1\times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256 \\ 1\times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256 \\ 1\times 1, 1024 \end{bmatrix} \times 36$
conv_stage4	7×7	$\begin{bmatrix} 3\times 3, 512 \\ 3\times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times 3, 512 \\ 3\times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512 \\ 1\times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512 \\ 1\times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512 \\ 1\times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Kaiming He et al., [Deep Residual Learning for Image Recognition](#), CVPR 2015

ResNet: Network Architecture

```
class ResNet(nn.Module):

    def __init__(self, num_classes, pretrained=True):
        super(ResNet, self).__init__()
        # download PyTorch's own implementation of VGG model trained on ImageNet dataset
        net = models.resnet18(pretrained=True)

        # retained weightes for convolutional, maxpooling, and linear layers from ResNet
        self.conv1 = net.conv1
        self.bn1 = net.bn1
        self.relu = net.relu
        self.maxpool = net.maxpool

        self.layer1 = net.layer1
        self.layer2 = net.layer2
        self.layer3 = net.layer3
        self.layer4 = net.layer4

        self.avgpool = net.avgpool
        self.fc = net.fc

        # IMPORTANT: "If you need to fine-tune this network for your own dataset,
        # the simplest modification is to replace the last layer in self.classifier with
        # the updated ResNet has the desired number of output classes: 'num_classes'
        # self.classifier[-1] = nn.Linear(4096, num_classes) # only this last layer's weights will be trained from scratch

    def forward(self, x):

        print("shape of input: ", x.shape)

        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)
        x = self.maxpool(x)

        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)
        x = self.layer4(x)

        x = self.avgpool(x)
        print("output shape (self.avgpool): ", x.shape)
        x = torch.flatten(x, 1)
        x = self.fc(x)
        print("output shape (self.fc): ", x.shape)
```

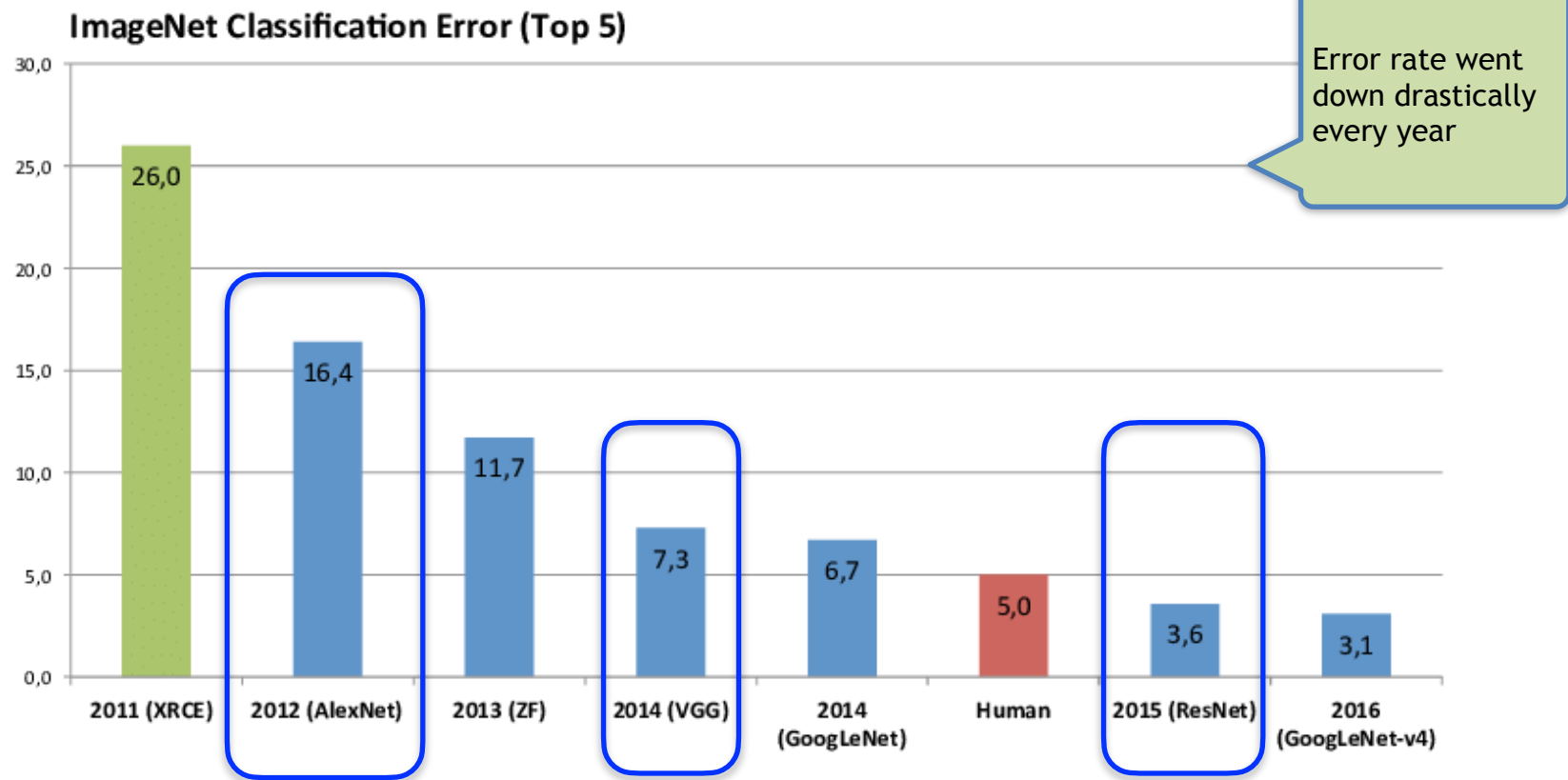
Group Activity: Inference with ResNet

https://github.com/alimoorreza/CS181-fall26-notes/blob/main/day08_popular_cnn4_resnet_dissection.ipynb

Kaiming He et al., [Deep Residual Learning for Image Recognition](#), CVPR 2015

ImageNet Winners by the Popular CNNs

- AlexNet (2012) → VGG (2014) → ResNet (2015)



Popular Model Implementation on Huggingface

List of models from huggingface

Models

All model architecture families include variants with pretrained weights. There are specific model variants without any weights, it is NOT a bug. Help training new or better weights is always appreciated.

- Aggregating Nested Transformers - <https://arxiv.org/abs/2105.12723>
- BEiT - <https://arxiv.org/abs/2106.08254>
- Big Transfer ResNetV2 (BiT) - <https://arxiv.org/abs/1912.11370>
- Bottleneck Transformers - <https://arxiv.org/abs/2101.11605>
- CaiT (Class-Attention in Image Transformers) - <https://arxiv.org/abs/2103.17239>
- CoaT (Co-Scale Conv-Attentional Image Transformers) - <https://arxiv.org/abs/2104.06399>
- CoAtNet (Convolution and Attention) - <https://arxiv.org/abs/2106.04803>
- ConvNeXt - <https://arxiv.org/abs/2201.03545>
- ConvNeXt-V2 - <http://arxiv.org/abs/2301.00808>
- ConViT (Soft Convolutional Inductive Biases Vision Transformers)- <https://arxiv.org/abs/2103.10697>
- CspNet (Cross-Stage Partial Networks) - <https://arxiv.org/abs/1911.11929>
- DeiT - <https://arxiv.org/abs/2012.12877>
- DeiT-III - <https://arxiv.org/pdf/2204.07118.pdf>
- DenseNet - <https://arxiv.org/abs/1608.06993>
- DLA - <https://arxiv.org/abs/1707.06484>
- DPN (Dual-Path Network) - <https://arxiv.org/abs/1707.01629>
- EdgeNeXt - <https://arxiv.org/abs/2206.10589>
- EfficientFormer - <https://arxiv.org/abs/2206.01191>
- EfficientNet (MBConvNet Family)