STEVENS
INSTITUTE OF TECHNOLOGY
1870

Convolutional Neural Networks

FA690 Machine Learning in Finance

Dr. Zonghao Yang

2025 Spring

Learning Objectives

- Understand the representation of digital images in computer systems and explain why traditional neural networks are suboptimal for image processing tasks
- Describe the core principles of Convolutional Neural Networks (CNNs), including local pattern recognition, parameter sharing, and translation invariance
- Explain the key components of CNN architecture including convolutional layers, pooling layers, and filters
- Apply concepts of transfer learning to leverage pre-trained CNN models for new image classification tasks

Convolutional Neural Network

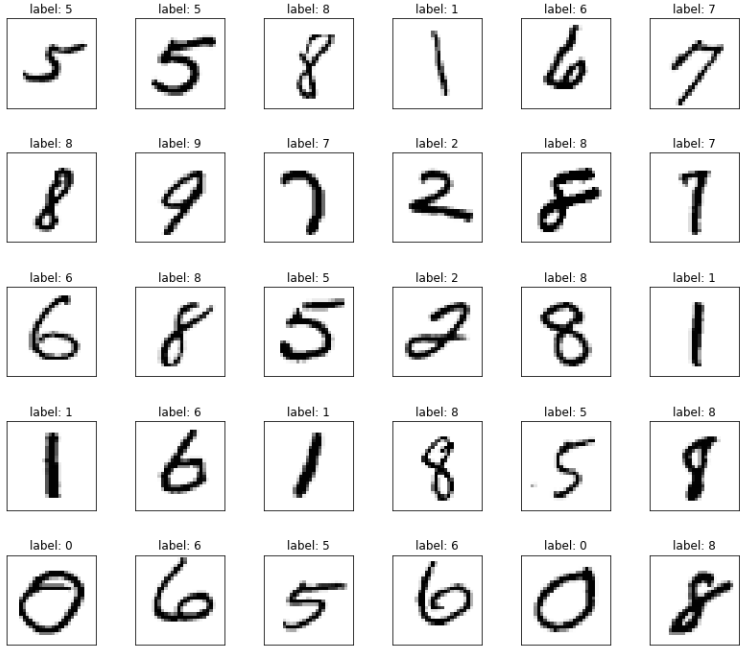
Neural Network Architecture Designed for Images

Image Representation

Grayscale Images

MNIST dataset: Collection of well-centered handwritten digits

- Size: 70,000 images
- Image: 28×28 pixels with grayscale, i.e., with pixel values ranging from 0 (white) to 255 (black)
- Label ranges: 0 to 9 (10 classes)
- Benchmark for machine learning models, especially in image classification tasks

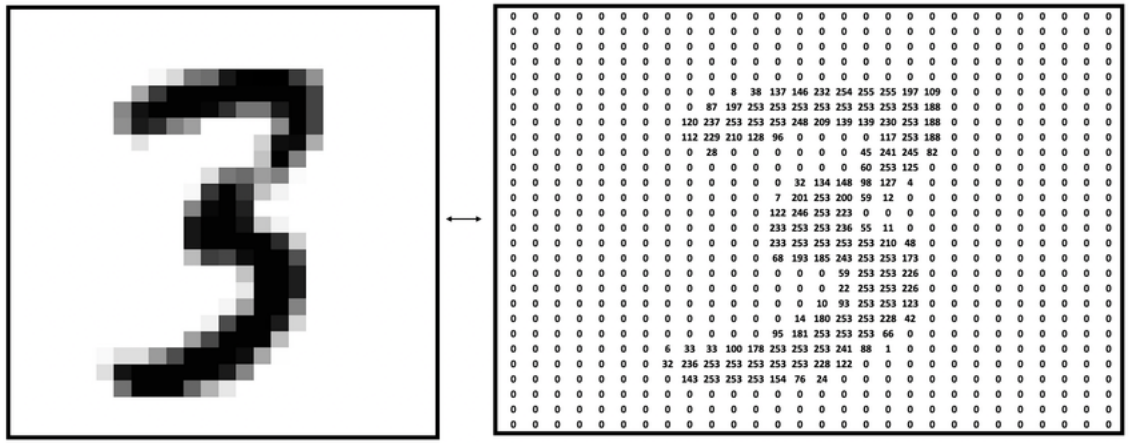


Sample digits from the MNIST dataset

Image Representation

Grayscale Images

- A grayscale image is a rectangular array of pixels
- The light intensity of each pixel is a number between 0 and 255; As the number increases from 0 to 255, the pixel goes from black through gray to white
- Each cell of the matrix shows the light intensity of the pixel at that location
- What the computer sees: a 28×28 matrix of pixel values ranging from 0 (white) to 255 (black)

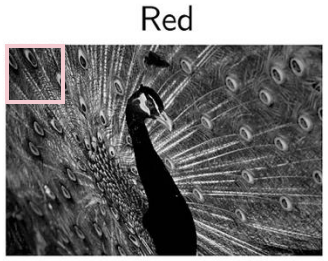


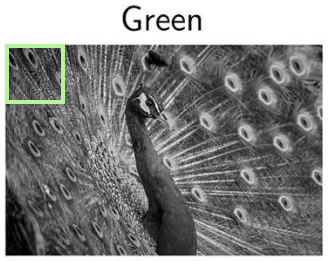
Representation of Grayscale Images

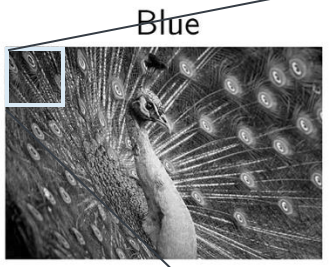
Image Representation

Color Images

- Each pixel of a color image is represented by three intensities, corresponding to the pixel's "redness,", "greenness", and "blueness"
- Each light intensity is still a number between 0 and 255
- Color images are represented as 3 matrices of numbers corresponding to the Red, Green, and Blue (RGB) "channels" respectively







128	41	151	219	52	70
176	20	137	165	167	18
120	144	249	133	0	250
52	233	174	102	232	40
218	131	104	244	95	9
166	34	121	251	172	56

Image Classification

100 x 100

Model

$$\begin{bmatrix} \vdots \\ 0.2 \\ 0.7 \\ 0.1 \\ \vdots \end{bmatrix} \quad y'$$

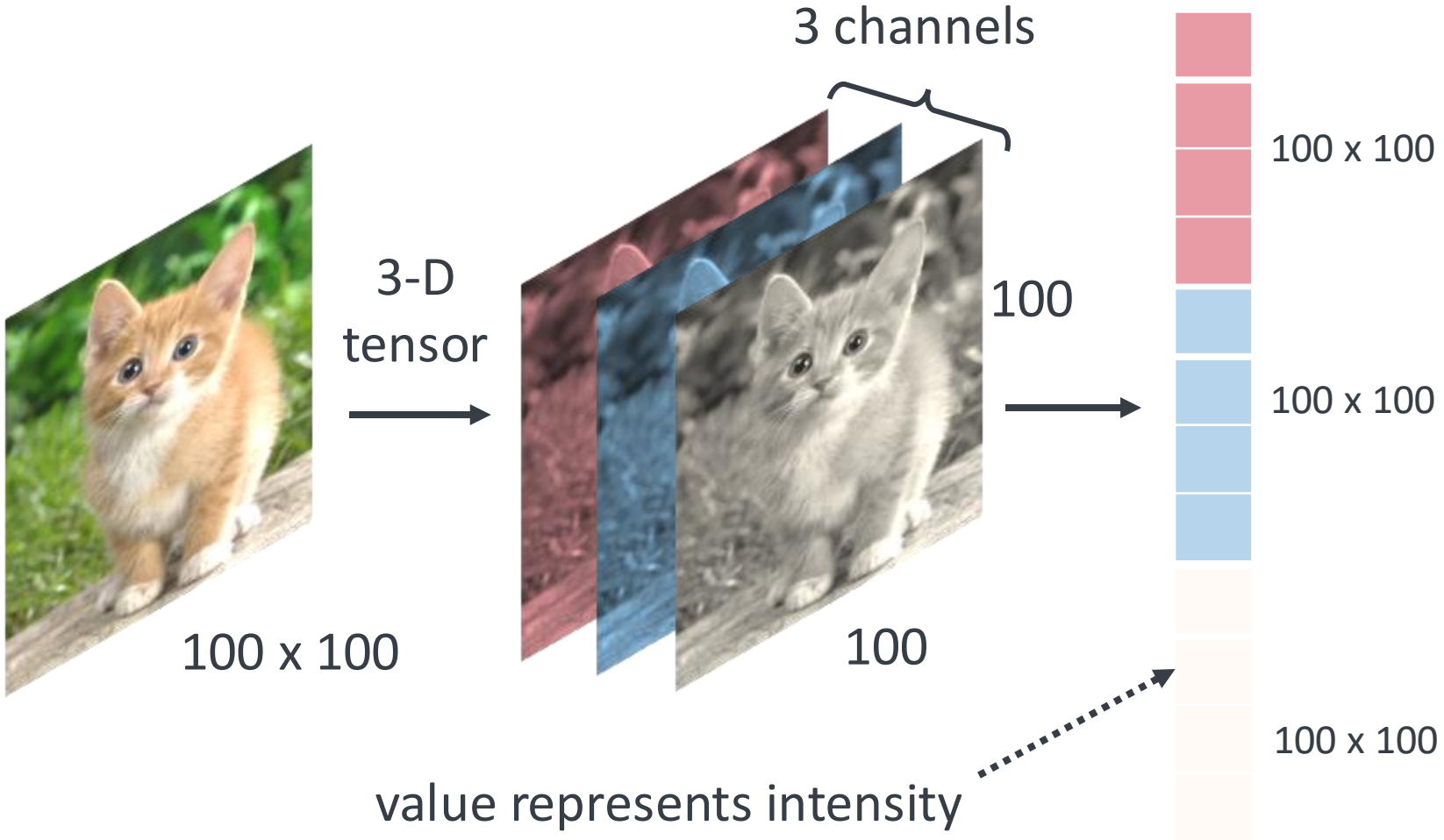
$$\begin{matrix} \\ \text{dog} \\ \text{cat} \\ \text{tree} \\ \\ \end{matrix} \begin{bmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} \quad \hat{y}$$

Cross entropy

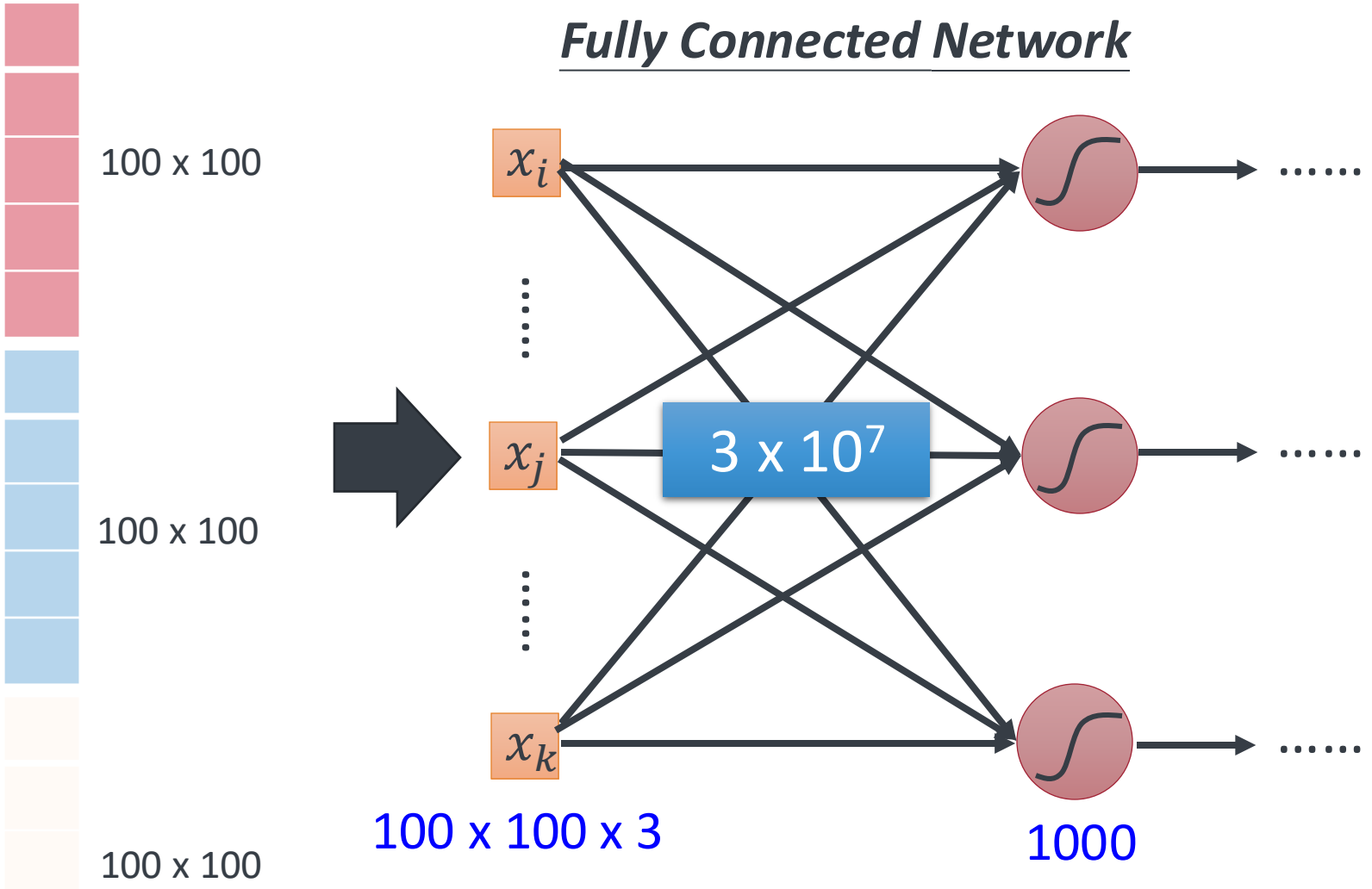
(All the images to be preprocessed to have the same size.)

From Perceptron to Convolution

Idea: Flatten the image and feed it into a fully-connected neural network



Fully Connected Network



Do we really need "*fully connected*" in image processing?

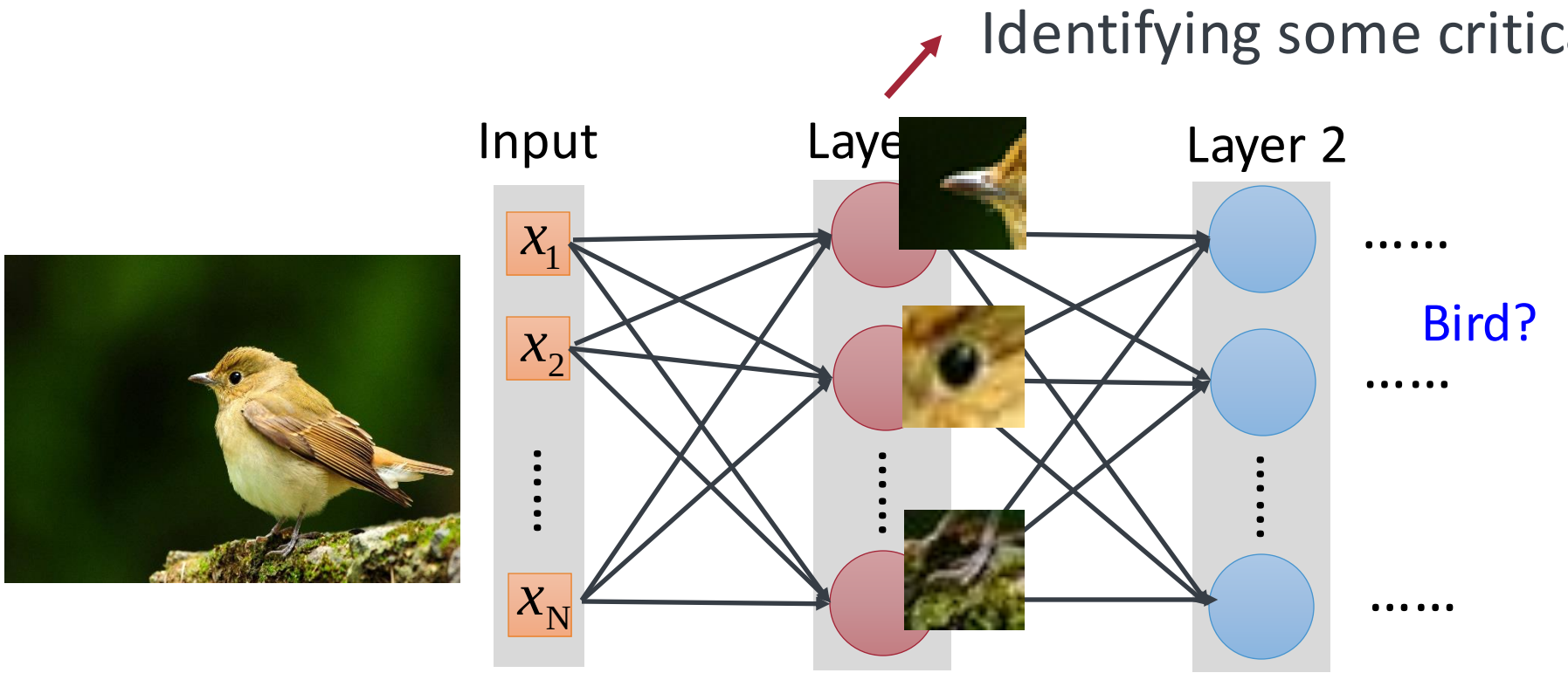
From Perceptron to Convolution

By flattening the image matrix into a long vector, the representation loses the spatial aspect

- **Parameter Overload:** Flattening a 100 x 100 x 3 – pixel color image and connecting it to a single 1000-neuron dense layer generates approximately 30 million parameters.
- **Computational Intensity and Overfitting:** The architecture becomes computationally intensive to train, data-hungry, and more prone to overfitting.
- **Loss of Spatial Relationships:** Important local spatial relationships among pixels, which are crucial for recognizing features, are lost when the image is flattened.
- **Inefficiency in Feature Learning:** The model fails to learn features that can be reused across different parts of an image. Ideally, if a feature (like a vertical line or a circle) appears in various places within an image, the model should learn this feature just once and recognize it wherever it appears, rather than learning it anew each time it's encountered.

Observation 1

Some patterns are much smaller than the whole image.



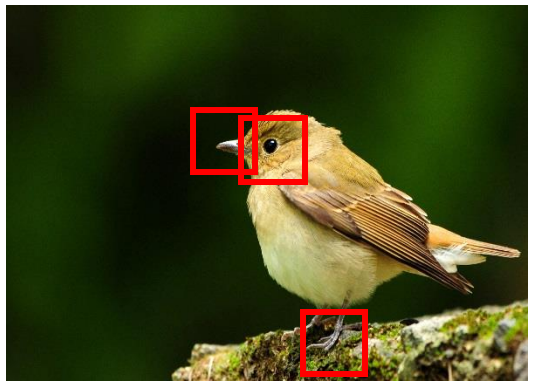
Perhaps human also identify birds in a similar way

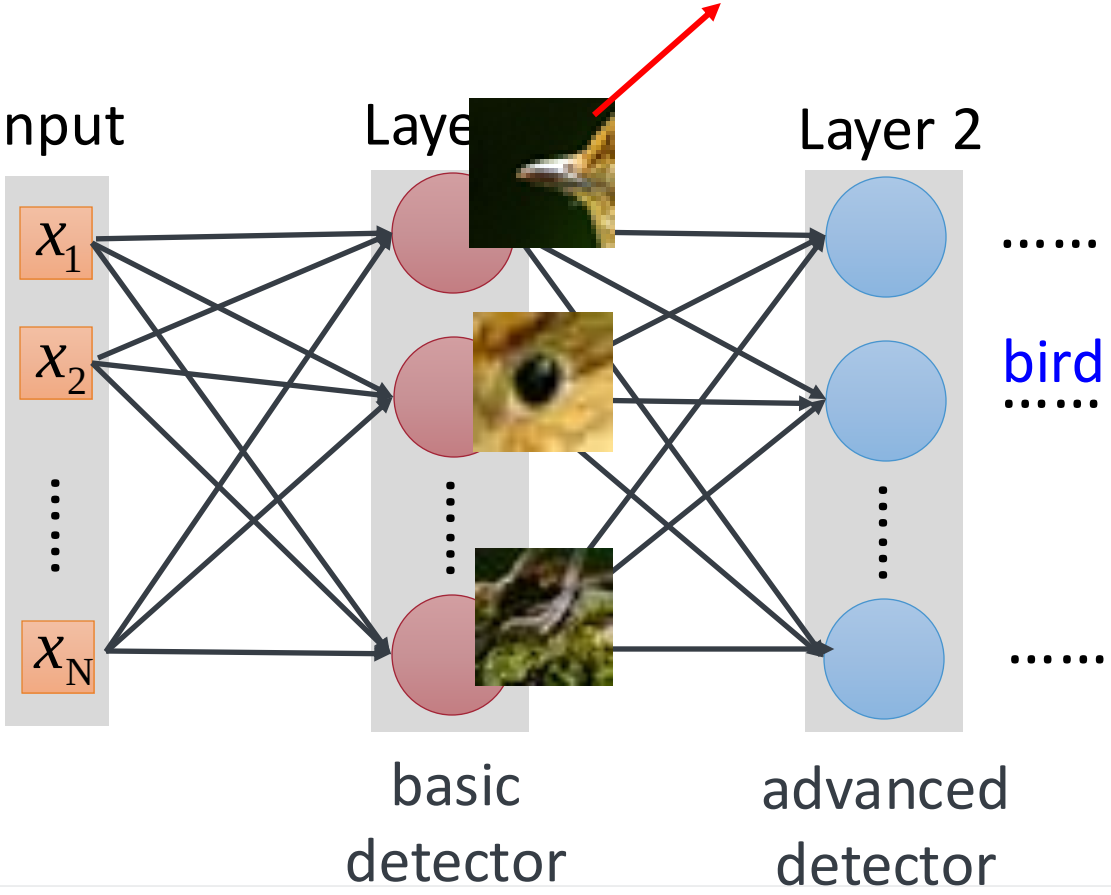
Observation 1

Some patterns are much smaller than the whole image.

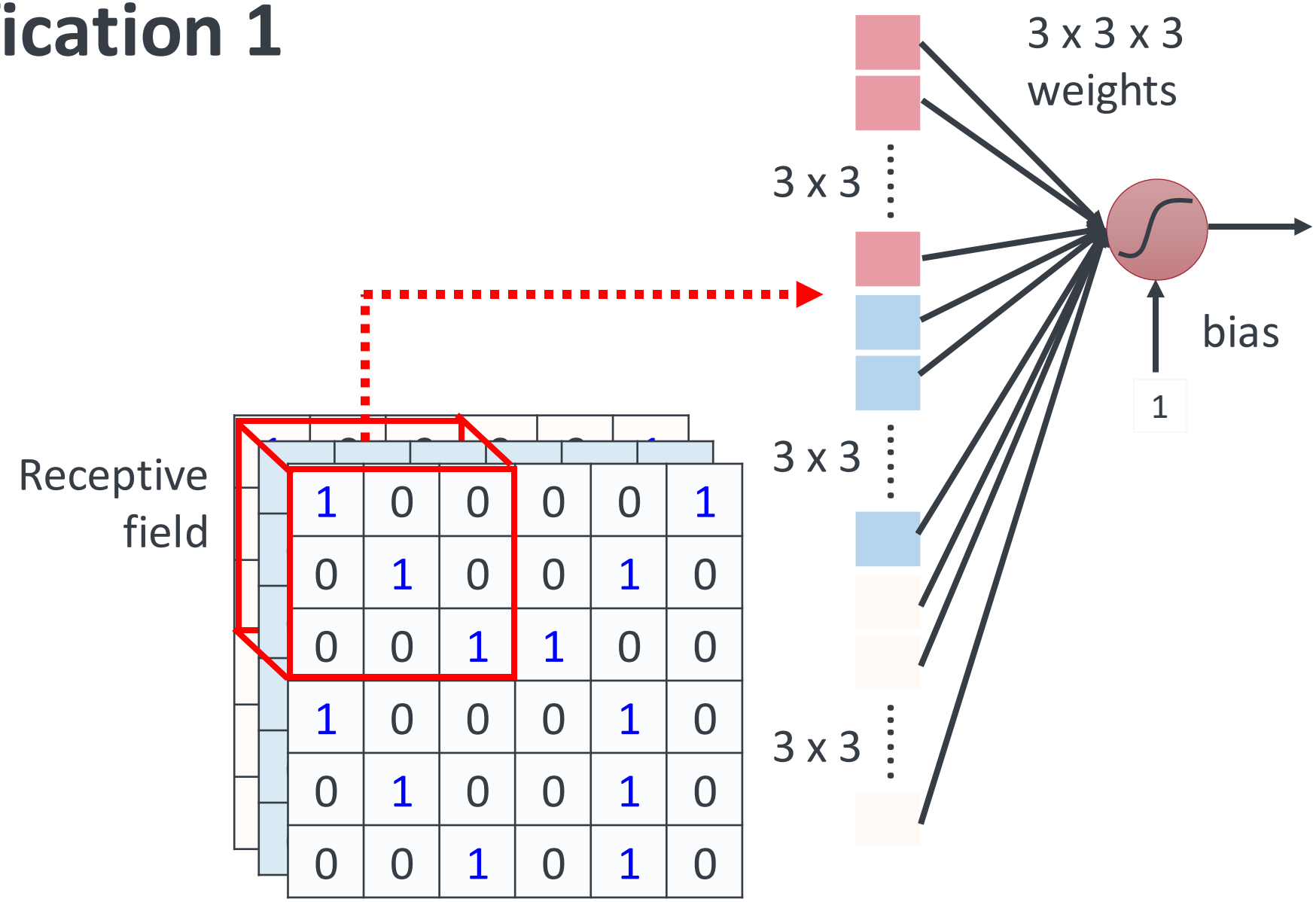
A neuron does not have to see the whole image.

Need to see the whole image?



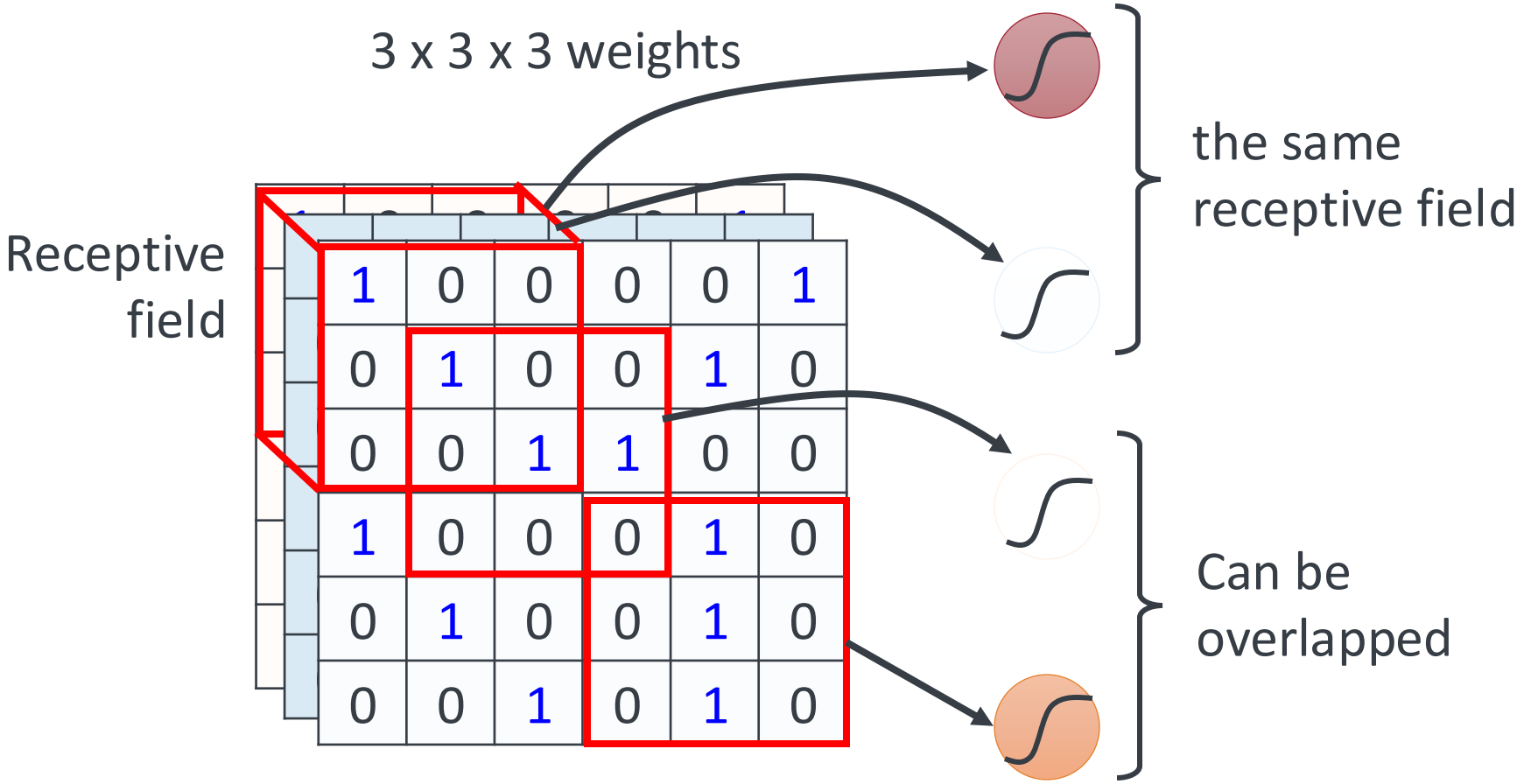


Specification 1



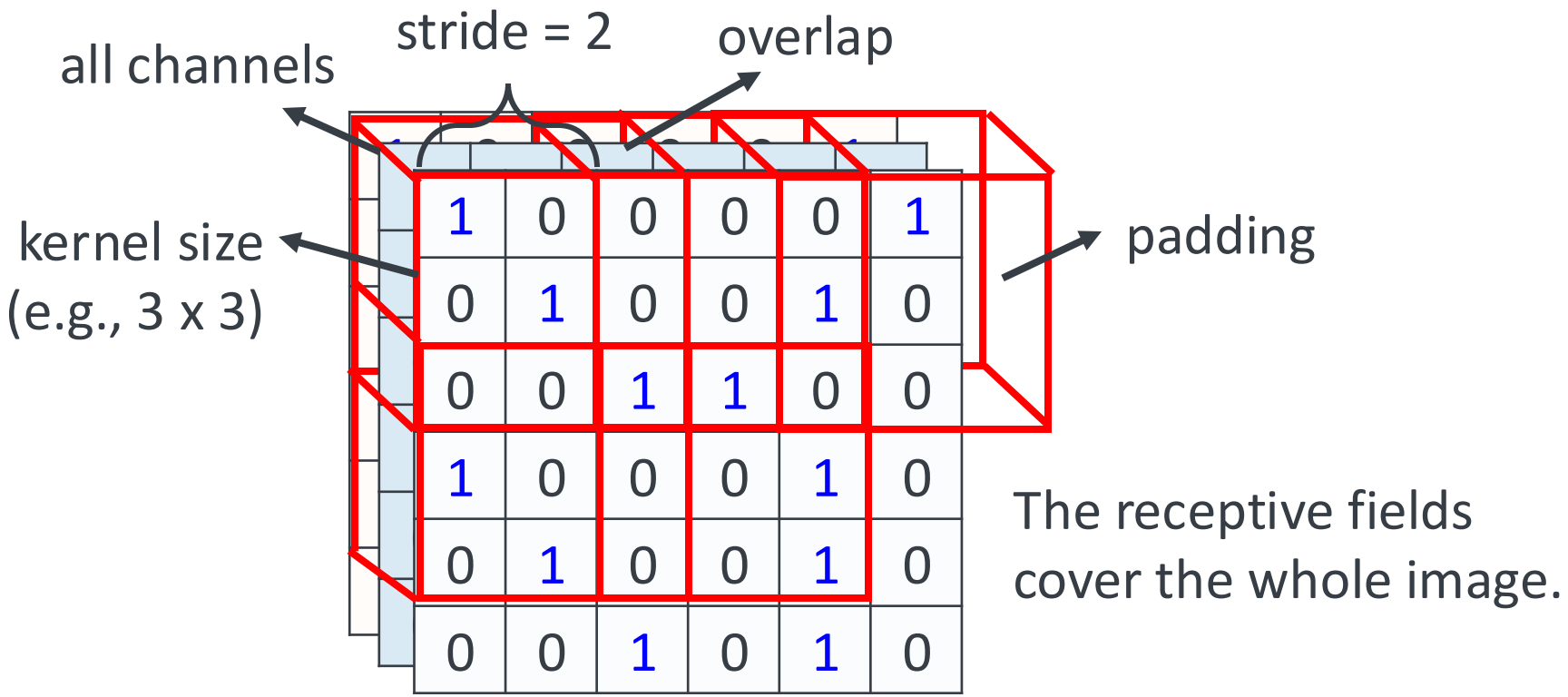
Specification 1

- Can different neurons have different sizes of receptive field?
- Cover only some channels?
- Not square receptive field?



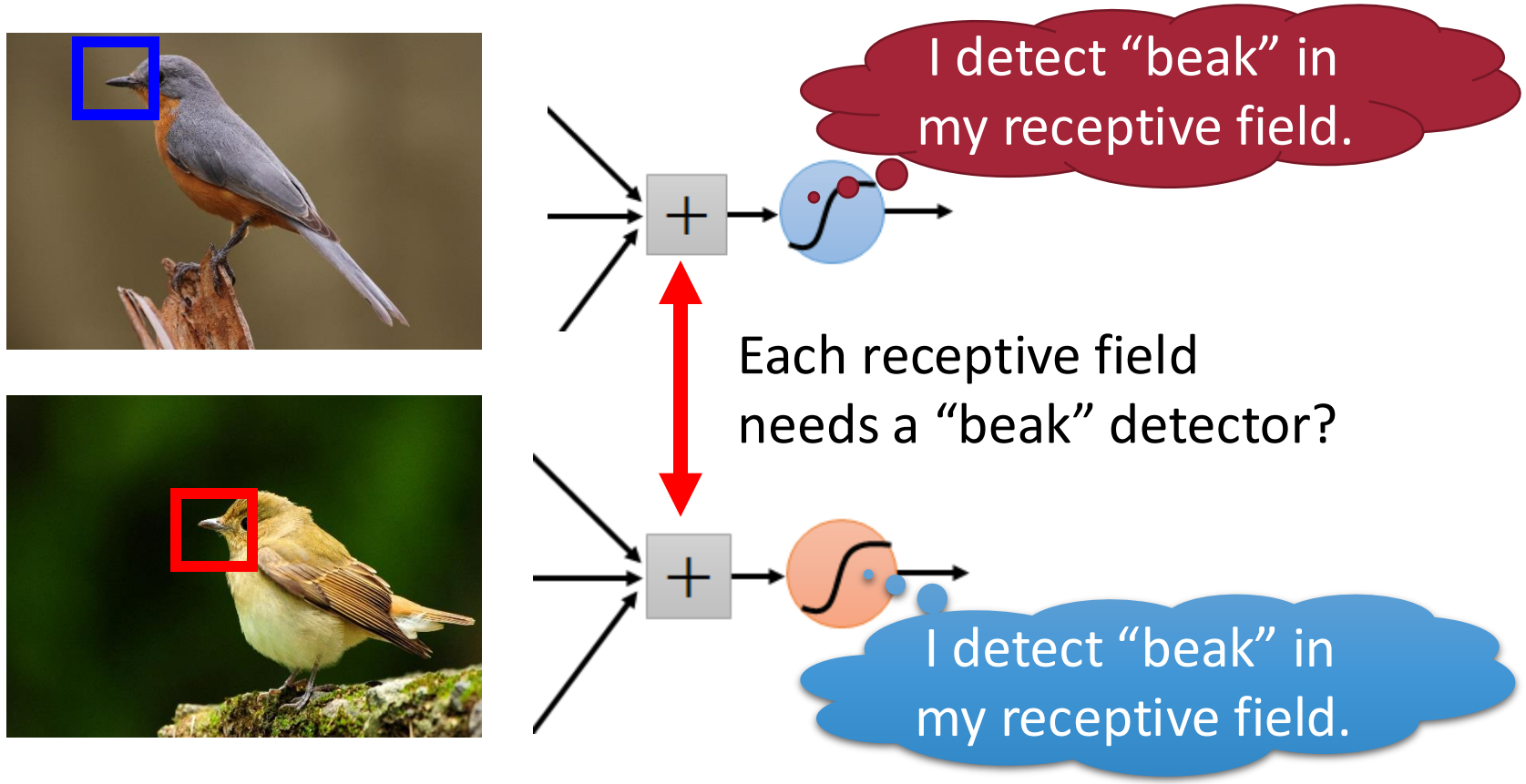
Specification 1 – Typical Setting

Each receptive field has a set of neurons (e.g., 64 neurons).

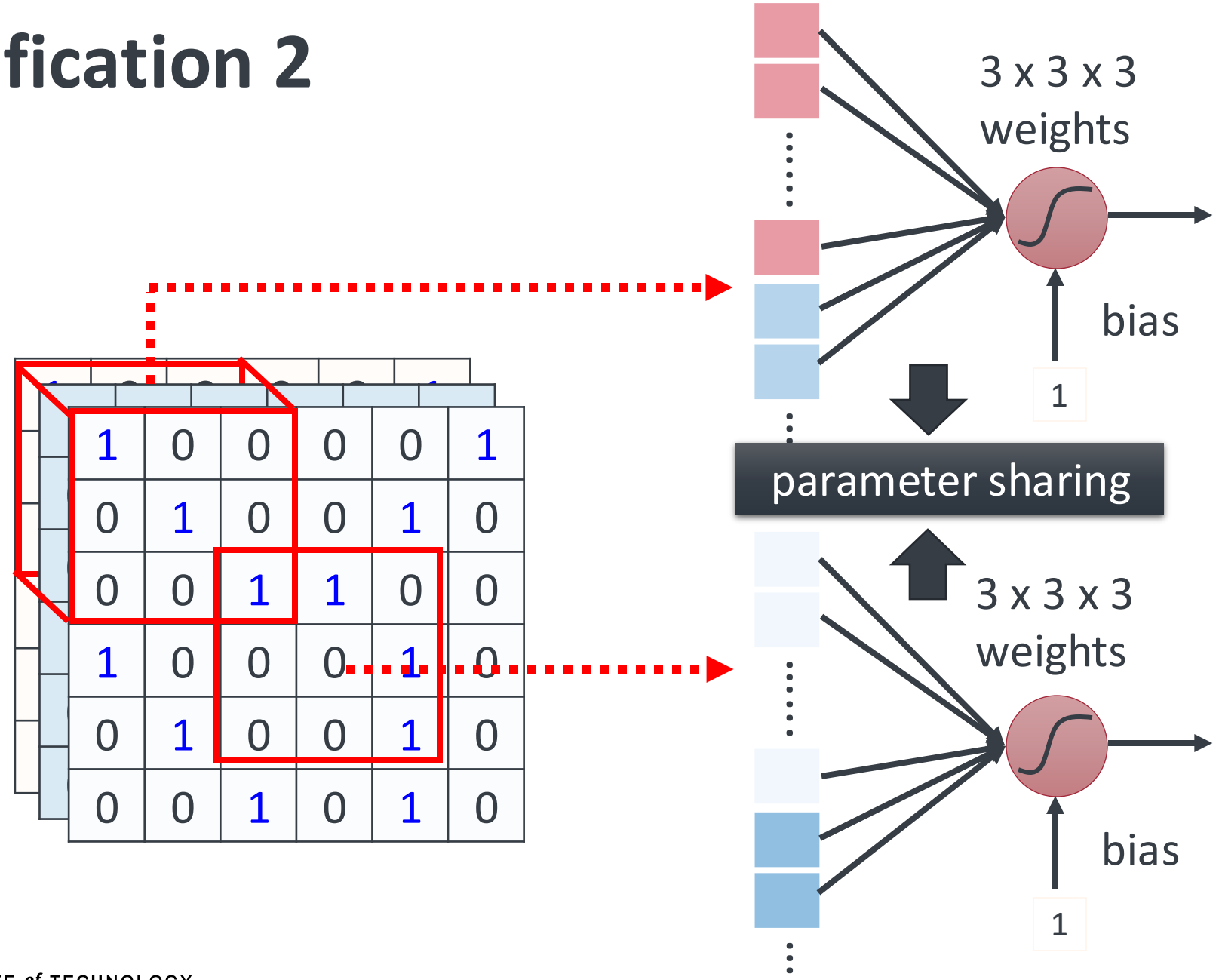


Observation 2

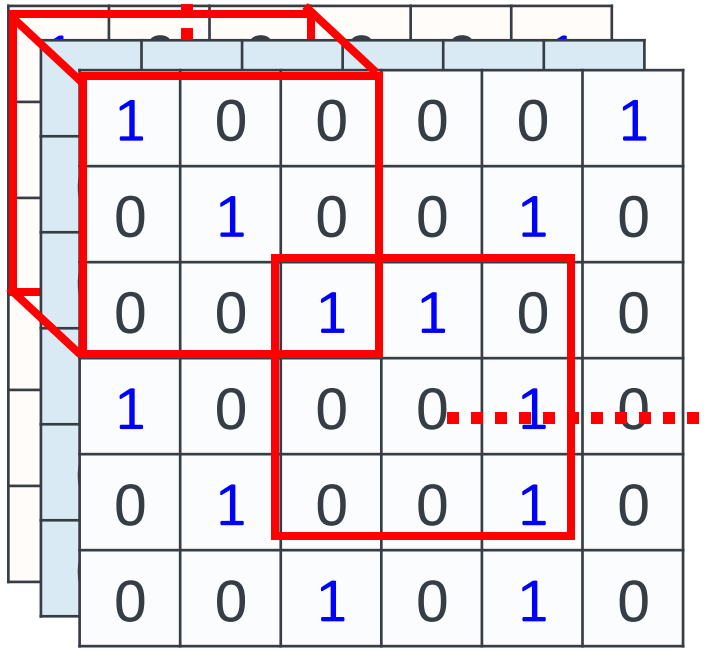
The same patterns appear in different regions

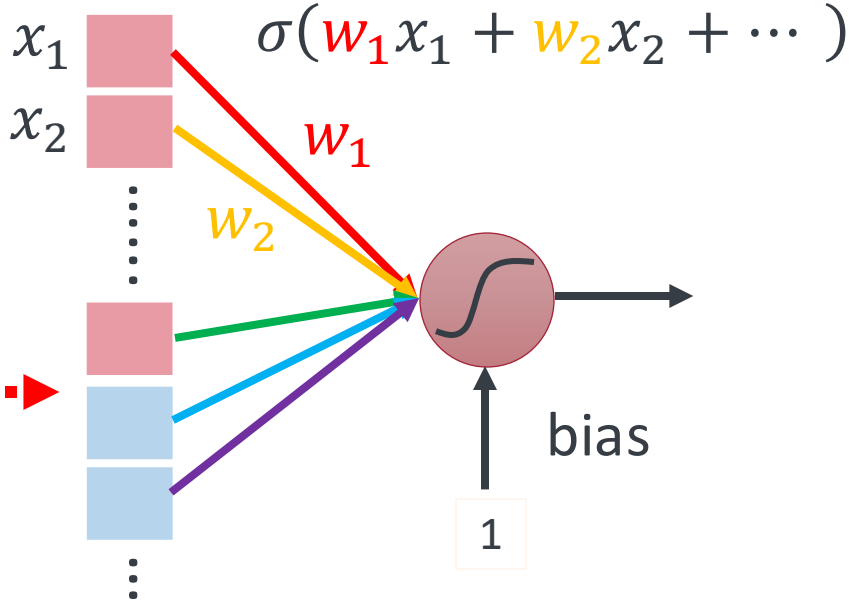


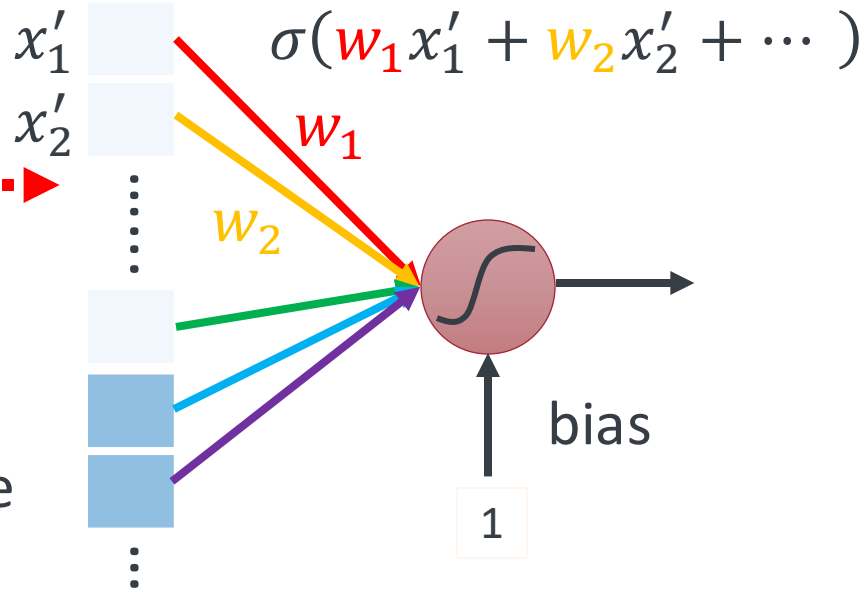
Specification 2



Specification 2



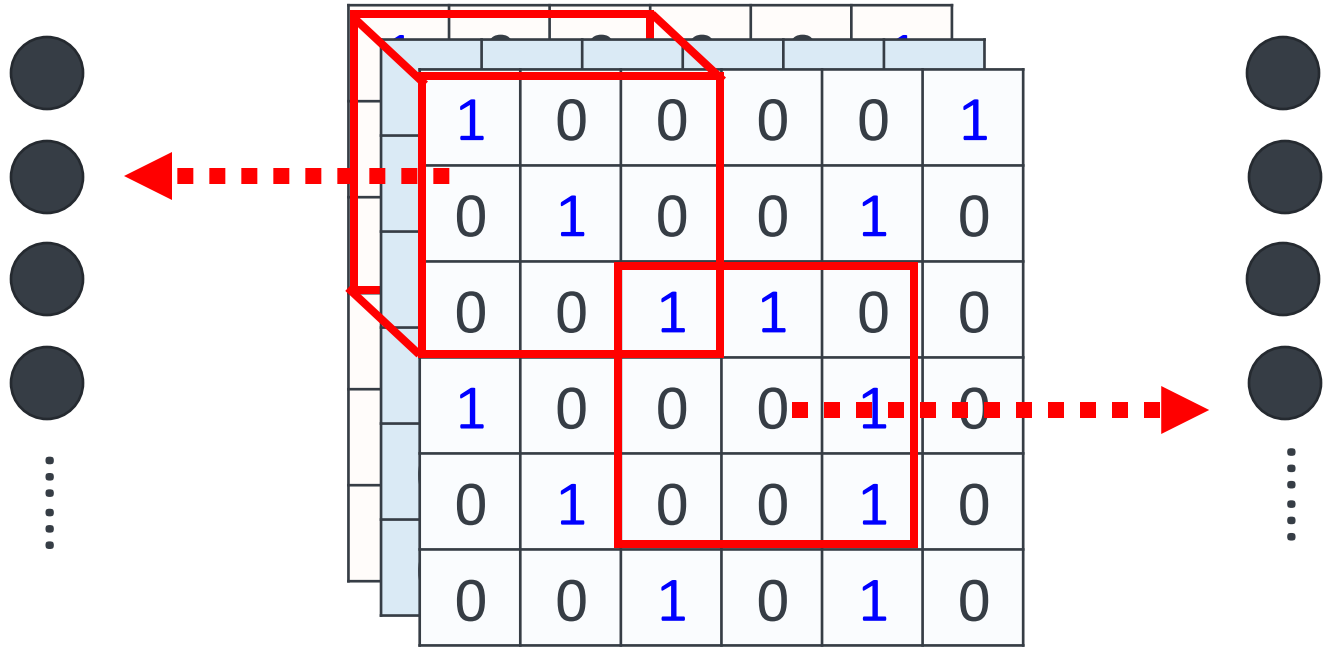




Two neurons with the same receptive field would not share parameters.

Specification 2– Typical Setting

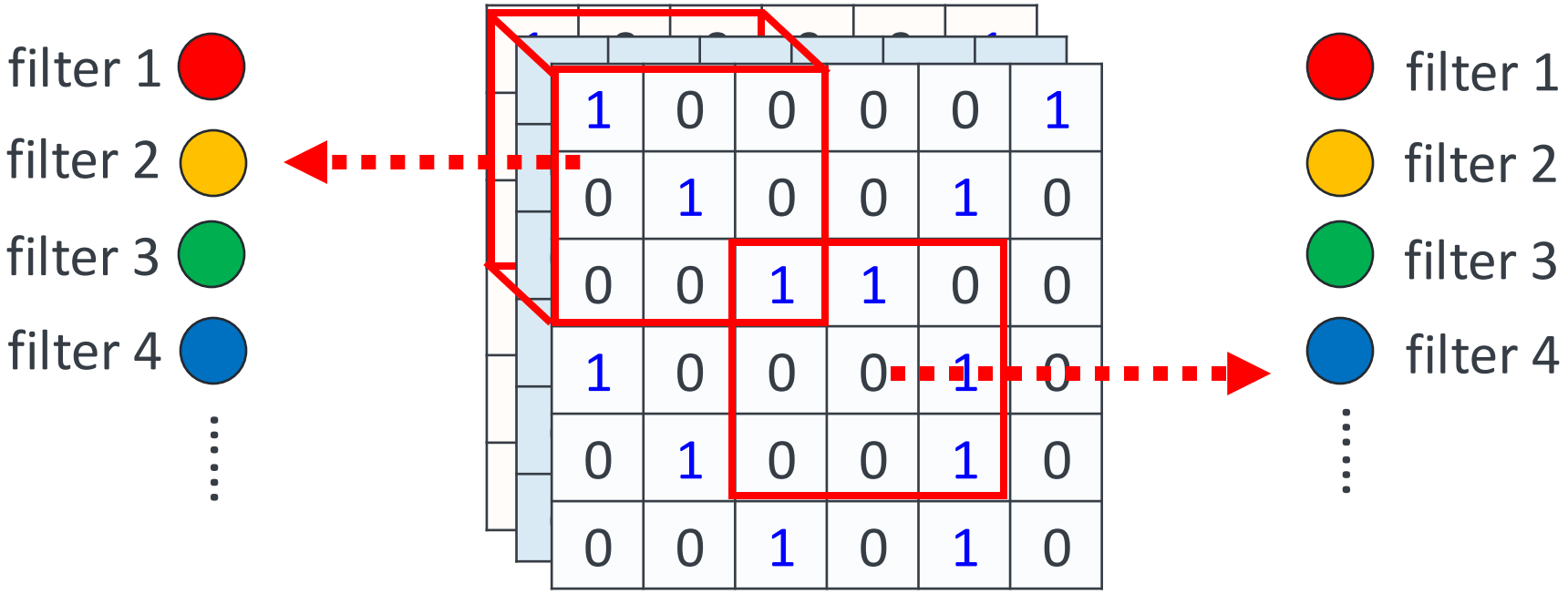
Each receptive field has a set of neurons (e.g., 64 neurons).



Specification 2 – Typical Setting

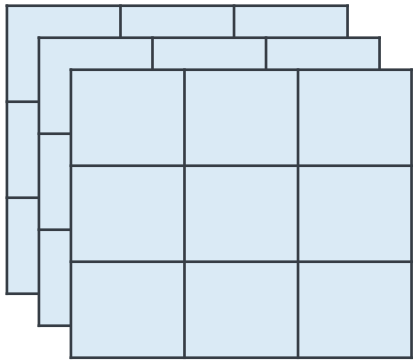
Each receptive field has a set of neurons (e.g., 64 neurons).

Each receptive field has the neurons with the same set of parameters.

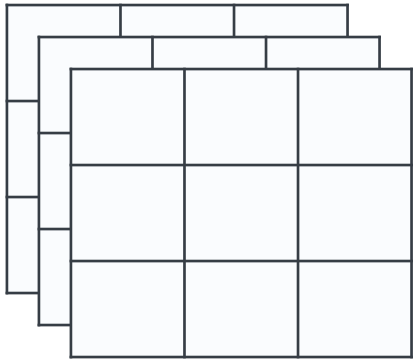


Convolutional Layer

Convolution



Filter 1
3 x 3 x channel
tensor



Filter 2
3 x 3 x channel
tensor

Each filter detects a small pattern (3 x 3 x channel).

channel = 3 (colorful)

channel = 1 (black and white)

Convolutional Layer

Consider channel = 1
(black and white image)

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

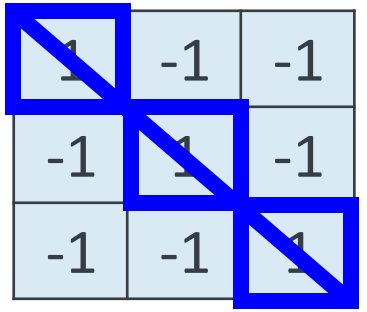
Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

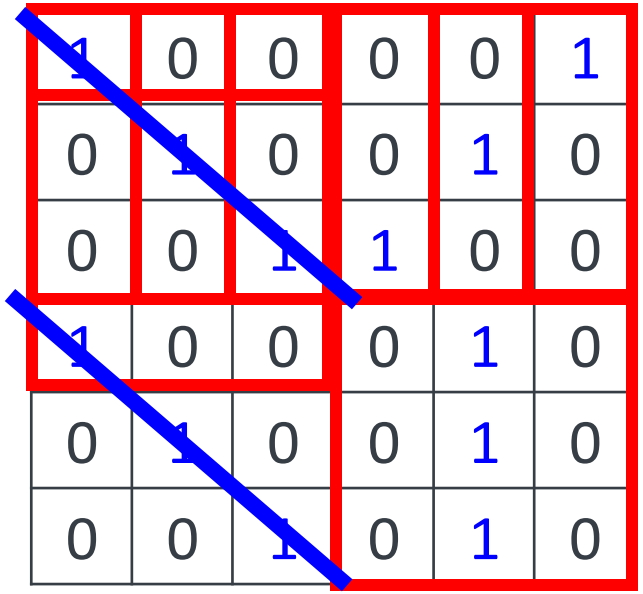
(The values in the filters are unknown parameters.)

Convolutional Layer

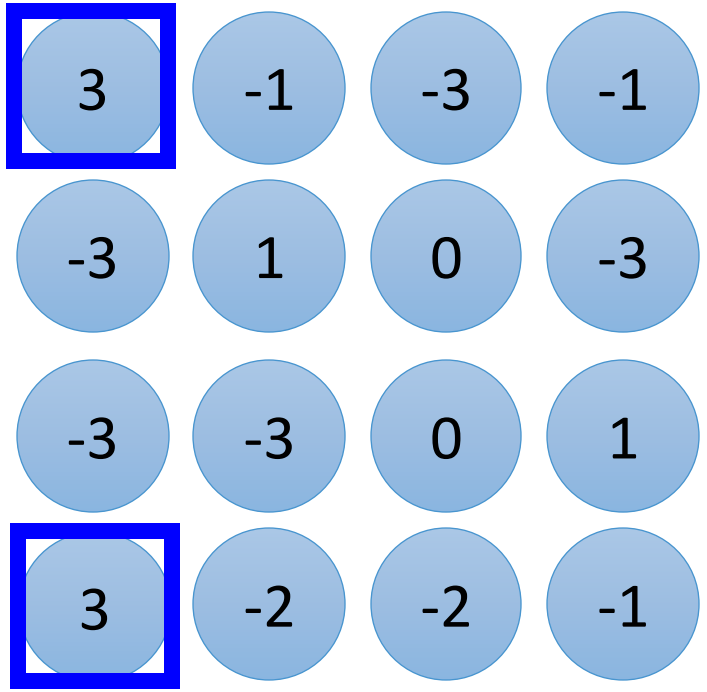


Filter 1

stride=1



6 x 6 image



Convolutional Layer

-1	1	-1
-1	1	-1
-1	1	-1

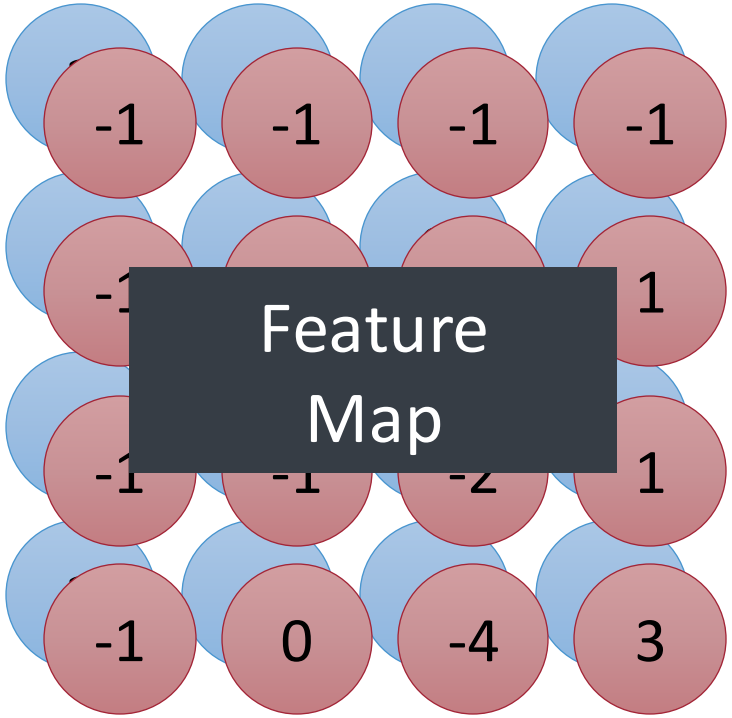
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Do the same process for every filter

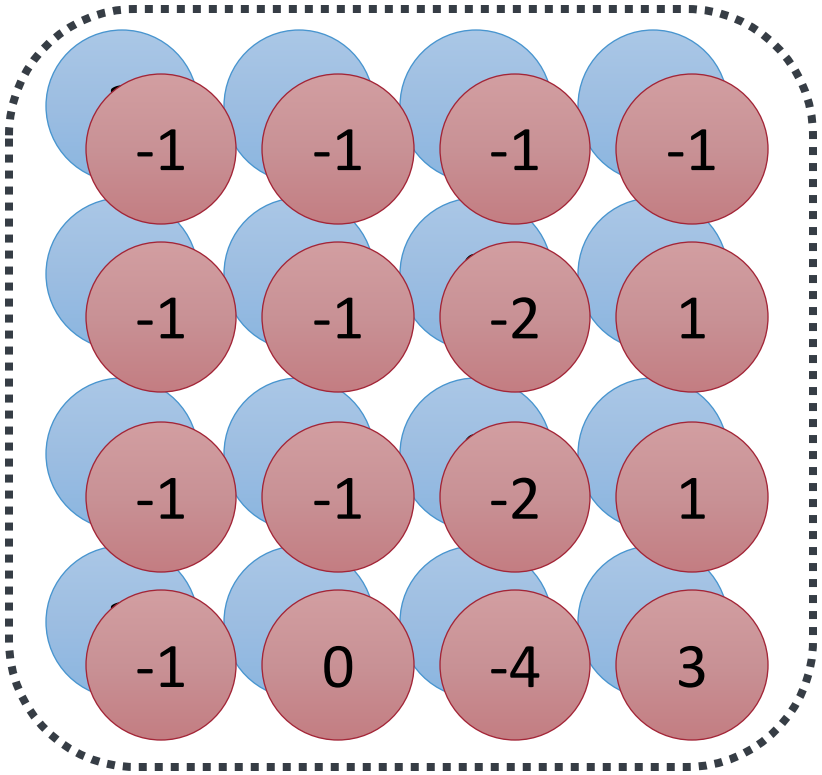


Convolutional Layer

64
filters

Convolution

Convolution



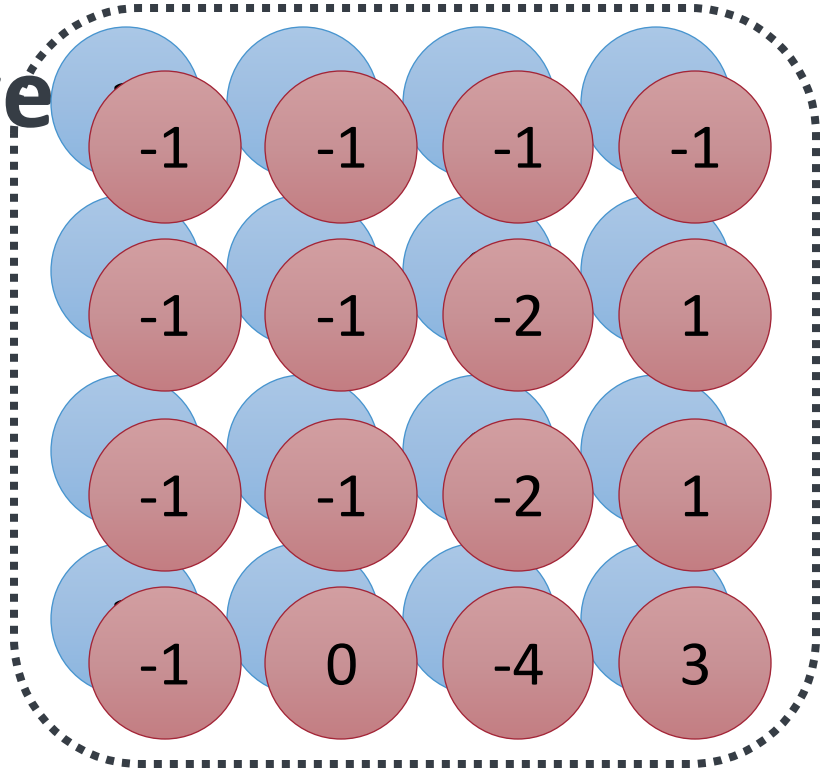
"Image" with 64 channels

Multiple Convolutional Laye

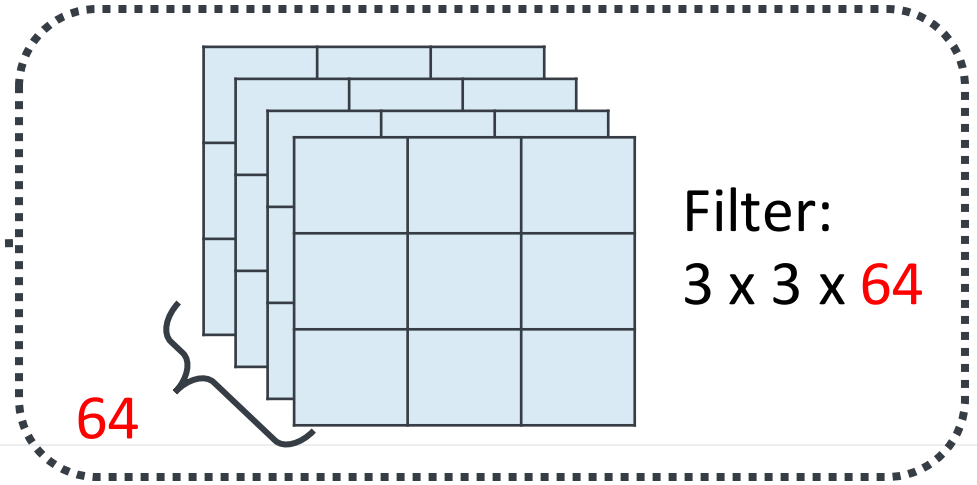
64
filters

Convolution

Convolution



"Image" with 64 channels

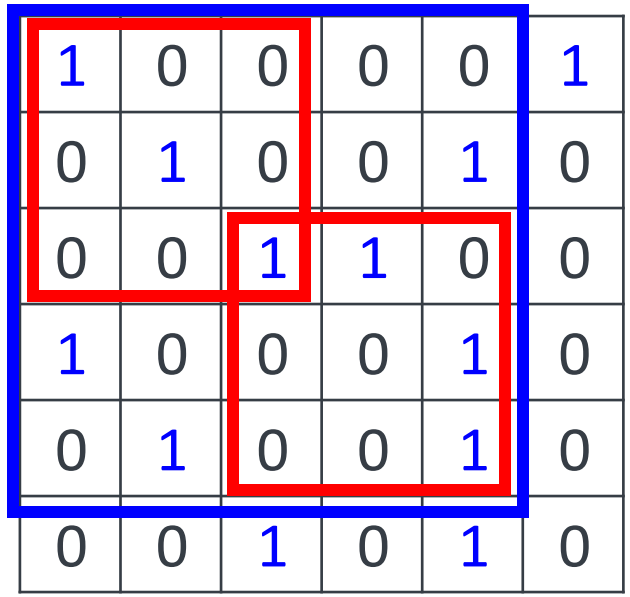


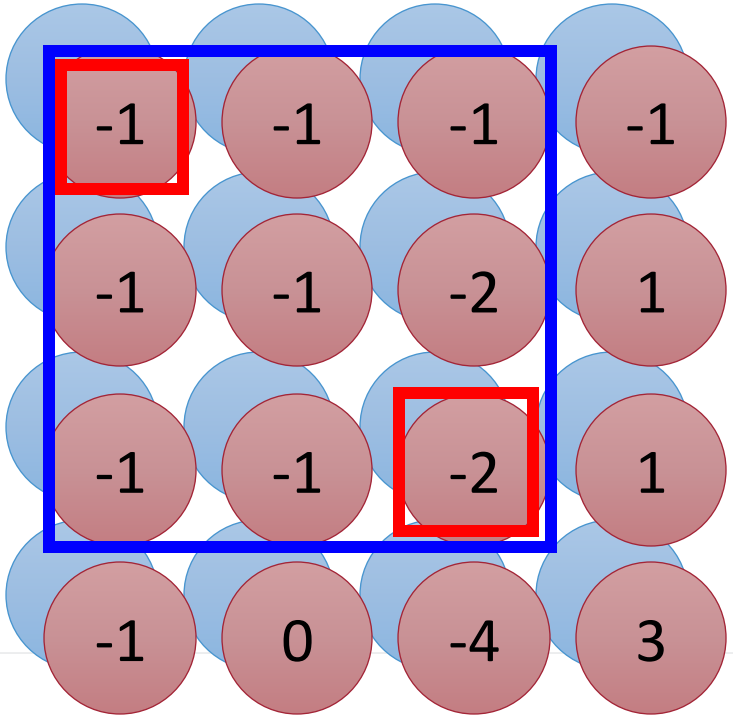
Multiple Convolutional Layers

64 filters

Convolution

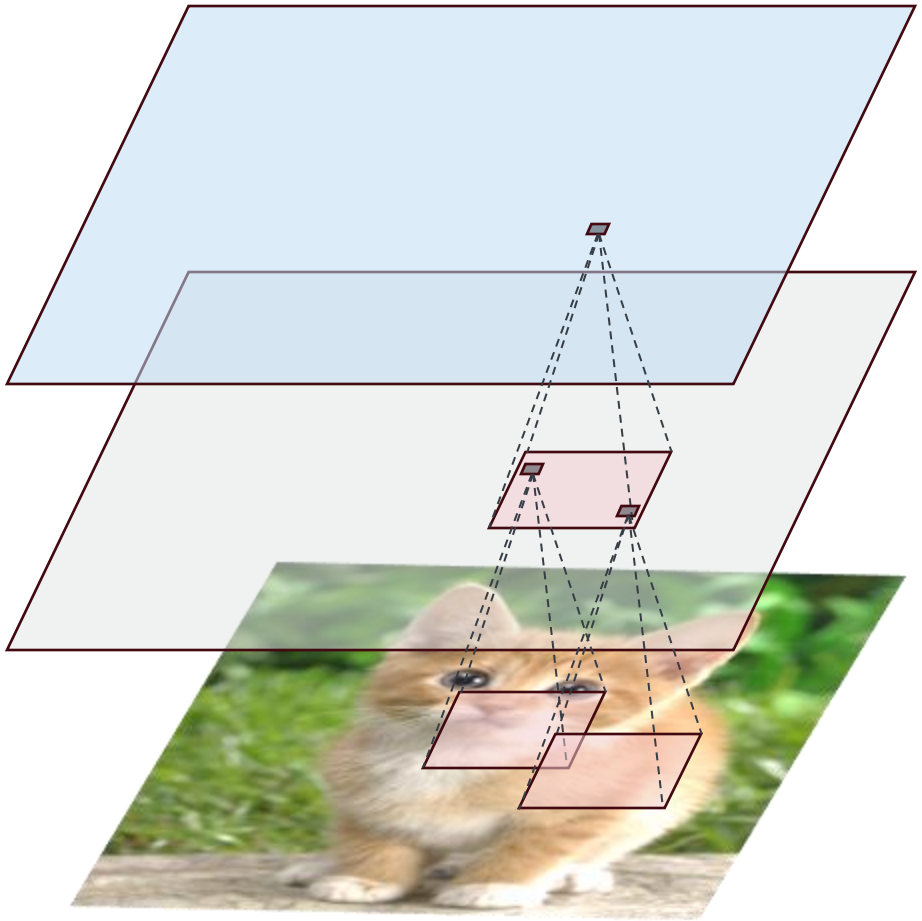
Convolution





Intuition about Convolutional Power

Later convolutional layers “see” more of the original input than the earlier layers



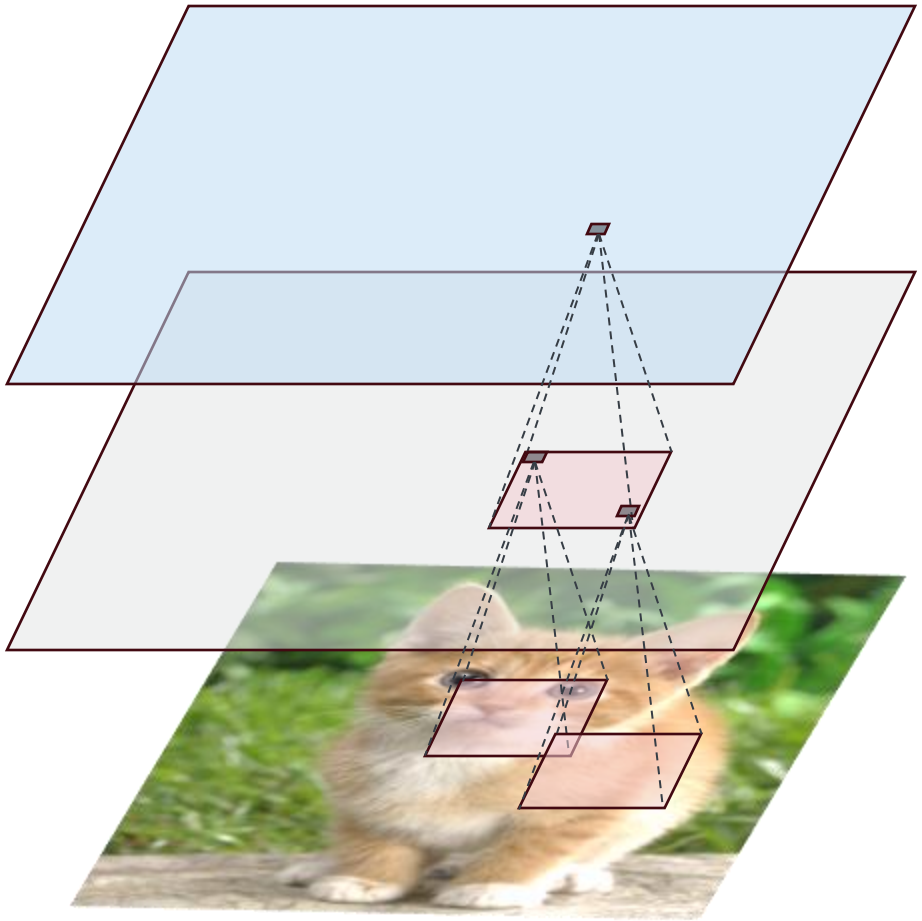
Convolutional layer 2

Convolutional layer 1

Input layer 2

Intuition about Convolutional Power

Later convolutional layers “see” more of the original input than the earlier layers



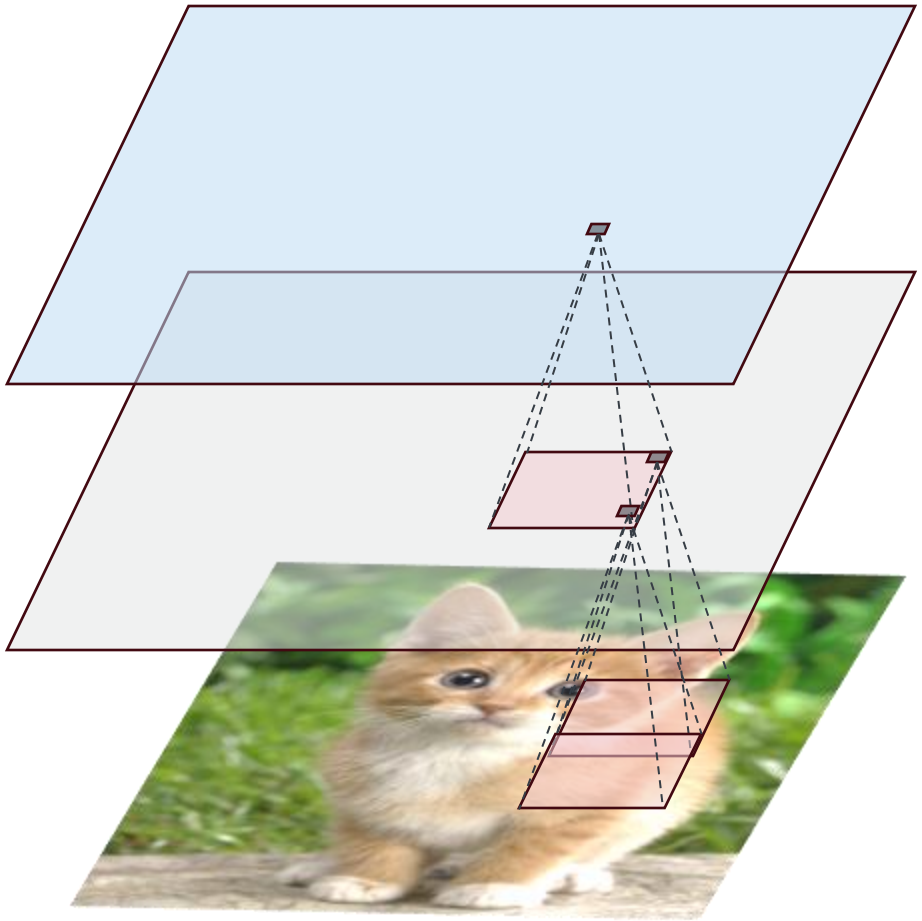
Convolutional layer 2

Convolutional layer 1

Input layer 2

Intuition about Convolutional Power

Later convolutional layers “see” more of the original input than the earlier layers



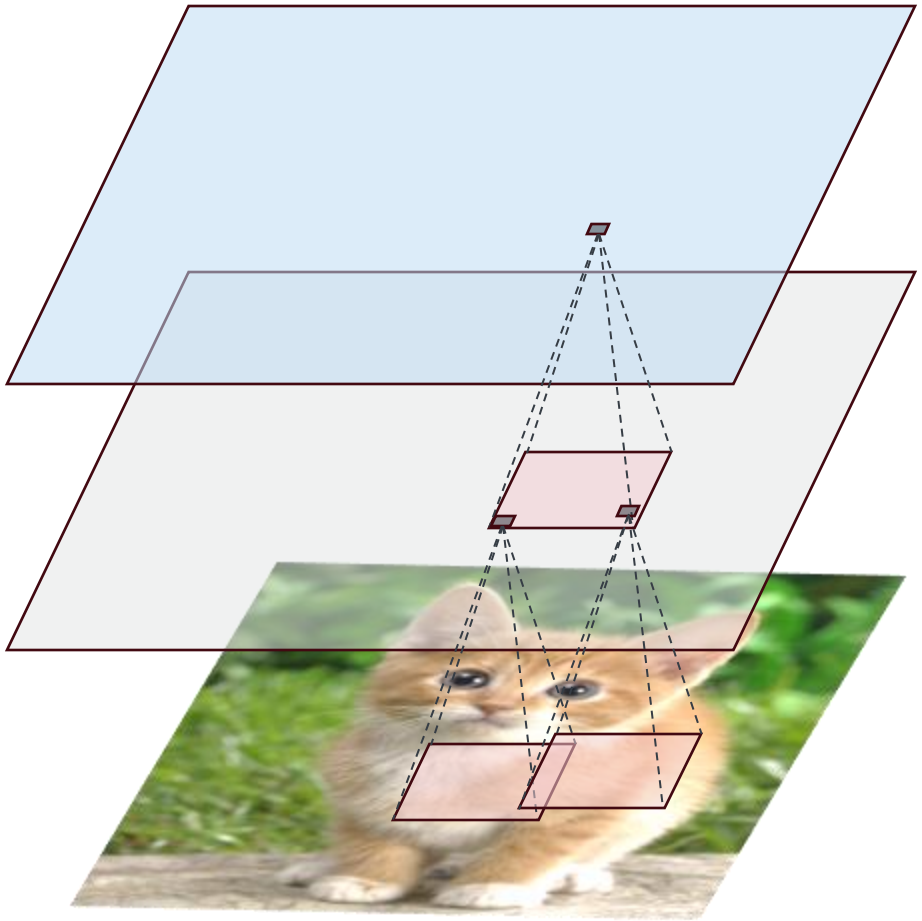
Convolutional layer 2

Convolutional layer 1

Input layer 2

Intuition about Convolutional Power

Later convolutional layers “see” more of the original input than the earlier layers



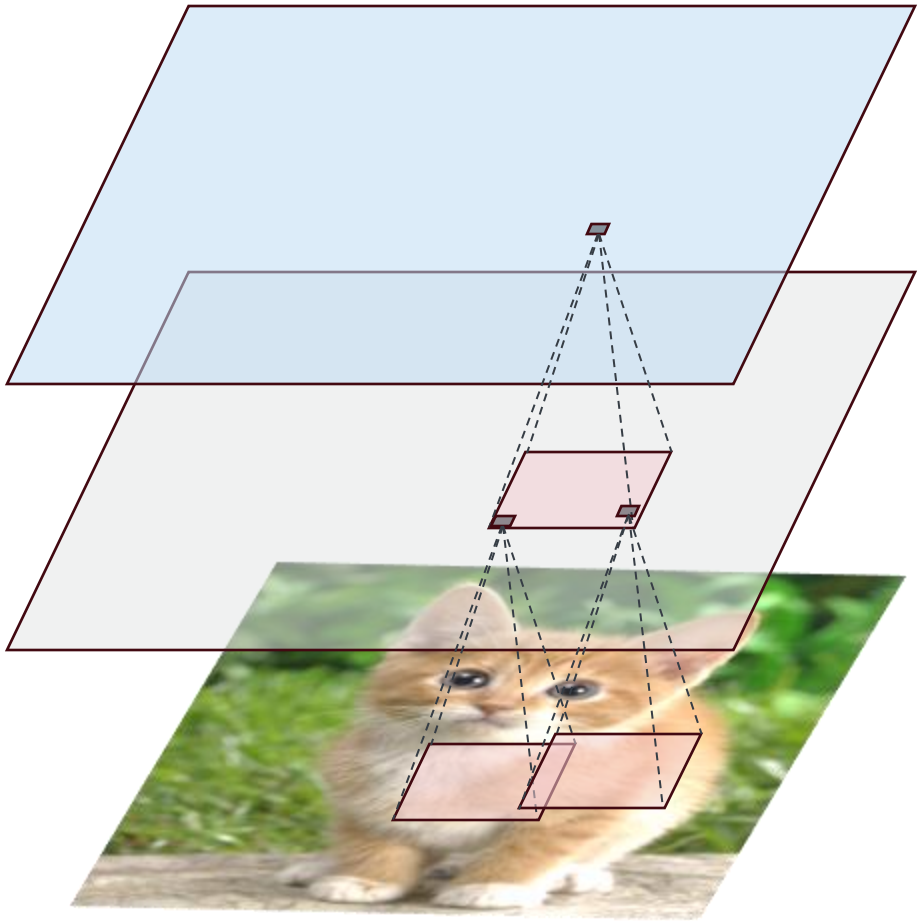
Convolutional layer 2

Convolutional layer 1

Input layer 2

Intuition about Convolutional Power

Later convolutional layers “see” more of the original input than the earlier layers



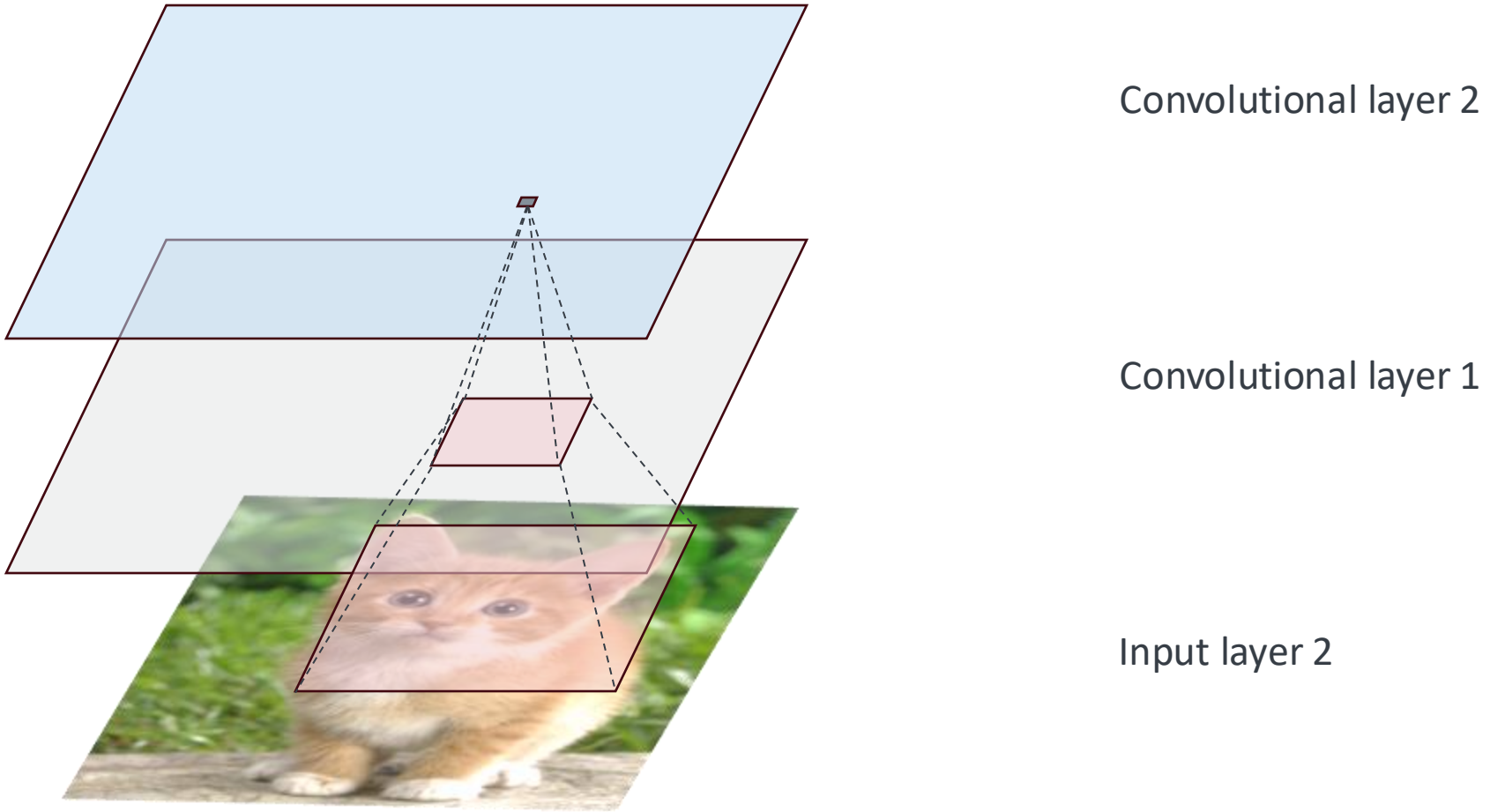
Convolutional layer 2

Convolutional layer 1

Input layer 2

Intuition about Convolutional Power

Later convolutional layers “see” more of the original input than the earlier layers

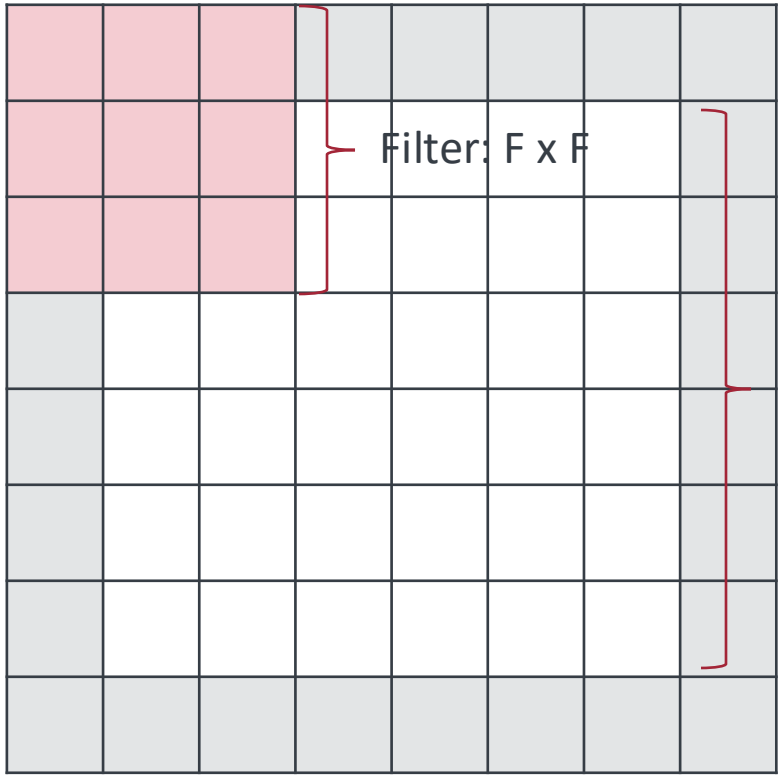


The Convolution Operation

Summary of Traditional Neurons vs. Convolutional Filters

- Filtering Application: Moving the filter across the image is equivalent to applying a filter to each specific local region, and crucially, this is done with shared weights, promoting feature detection consistency across the image
- Far fewer parameters: Utilizing shared weights across spatial locations means convolutional neural networks have significantly fewer parameters compared to fully connected layers, reducing the model's complexity
- Preserve local adjacency: Convolutional layers maintain the spatial hierarchy of the input data, ensuring that the local adjacency of pixels—crucial for detecting patterns and features—is preserved
- Translation invariance: They can recognize patterns irrespective of their position in the input image. This means that once the network learns a feature, it can detect it anywhere, making the model more efficient and robust to variations in object placement

Calculating the Size of the Output Layer



- Output size:

$$\frac{N+2P-F}{stride}+1$$

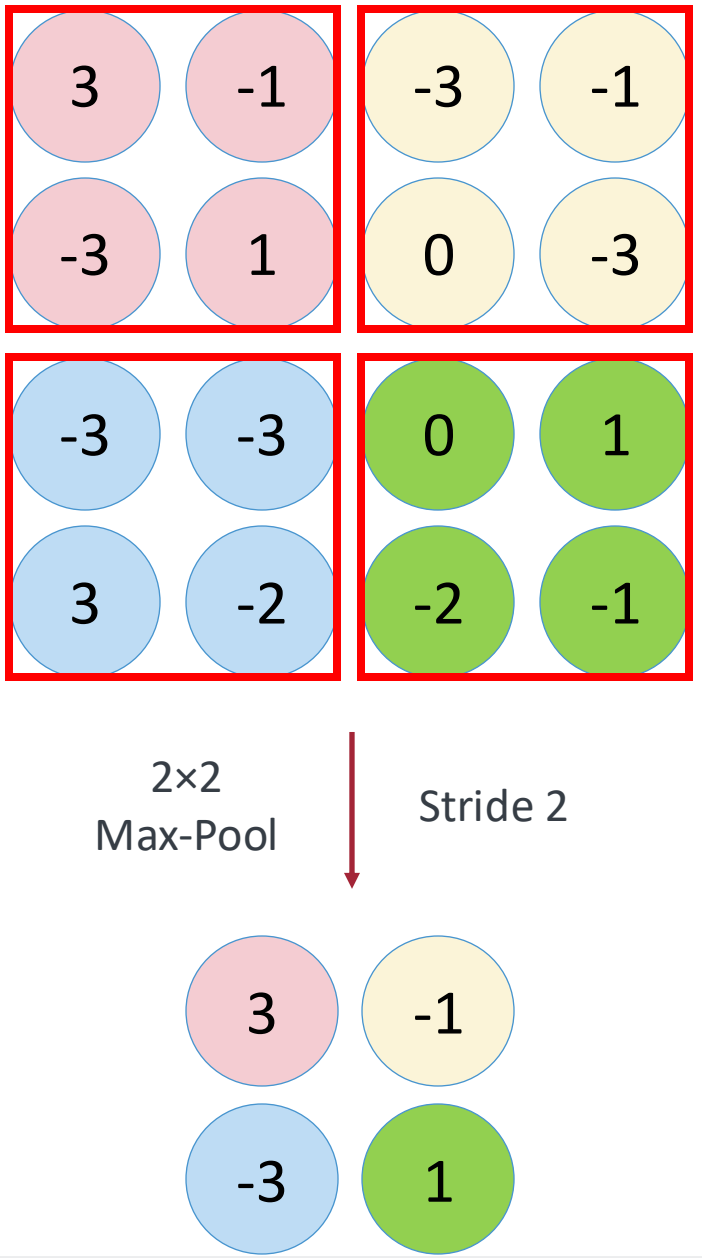
- Example: N = 6, F = 3, P = 1
  - stride 1 ⇒ (6+2-3)/1 + 1 = 6
  - stride 2 ⇒ (6+2-3)/2 + 1 = 3.5:\
  - stride 3 ⇒ (6+2-3)/3 + 1 = 2.67 :\
- Example: N = 5, F = 3, P = 1
  - stride 1 ⇒ (5+2-3)/1 + 1 = 5
  - stride 2 ⇒ (5+2-3)/2 + 1 = 3
  - stride 3 ⇒ (5+2-3)/3 + 1 = 2.33 :\

Observation 3

Subsampling the pixels will not change the object

Pooling - Max Pooling

- Pooling layers reduce the size of the tensor coming out of a convolutional layer
- The output from the pooling layer is fed to the next layer as usual
- Pooling increases robustness to noise and variations
- Max pooling propagates only the strongest signals from a cluster of neurons in the feature map

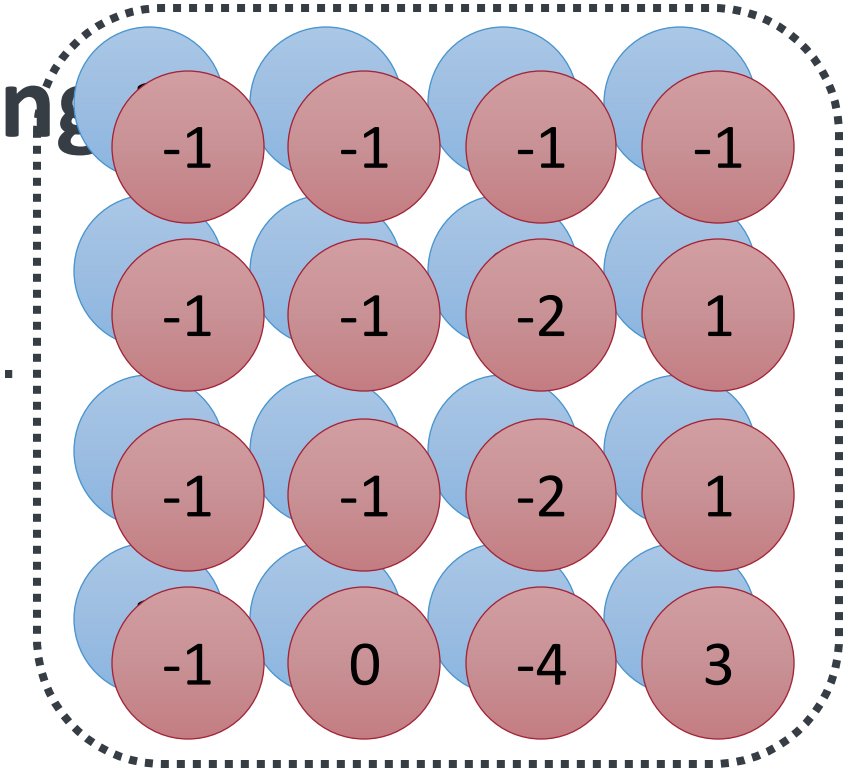


Convolutional Layers + Pooling

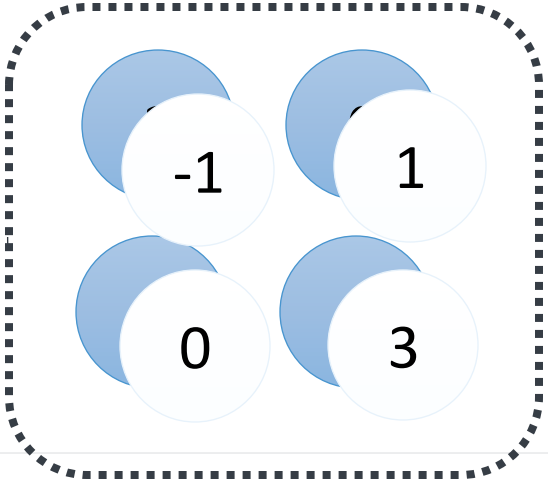
Repeat

Convolution

Pooling



"Image" with 64 channels



The Overall CNN Architecture

Convolution

Pooling

Convolution

Pooling

Flatten

Fully Connected Layers

softmax

cat dog ……

The Overall CNN Architecture

- Convolutional block
  - A convolutional block has 1-3 convolutional layers optionally followed by a max-pooling layer
  - Each block typically has more depth than the previous block but has a lower height & width
- Key components of convolutional neural networks
  - Convolutional operation: Learn features in input images through convolution
  - Activation function: Introduce non-linearity through activation function
  - Pooling: Reduce dimensionality and preserve invariance
- Animation: CNN Explainer [Link]

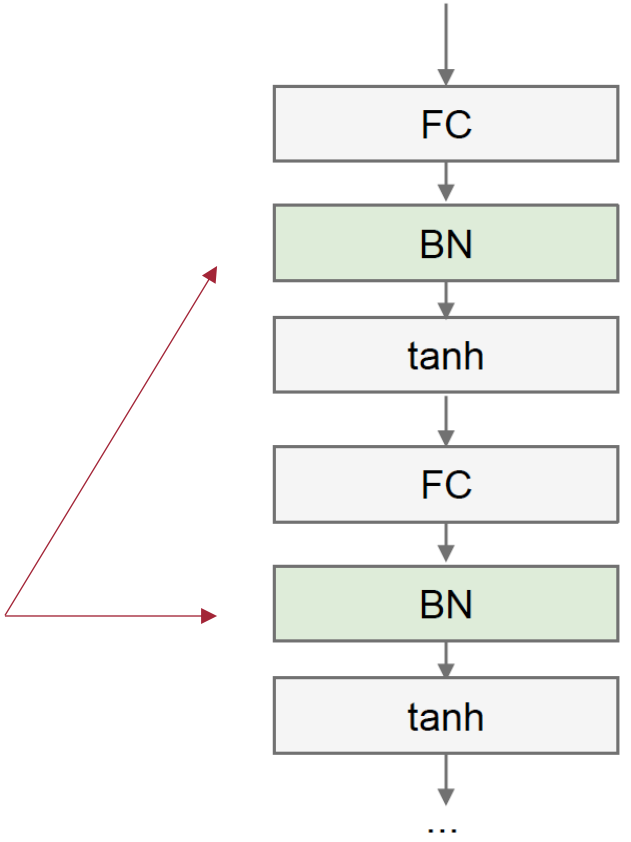
Training

Batch Normalization

- Denote by B a minibatch and let $x \in B$ be an input to batch normalization (BN)
- Batch normalization is defined as

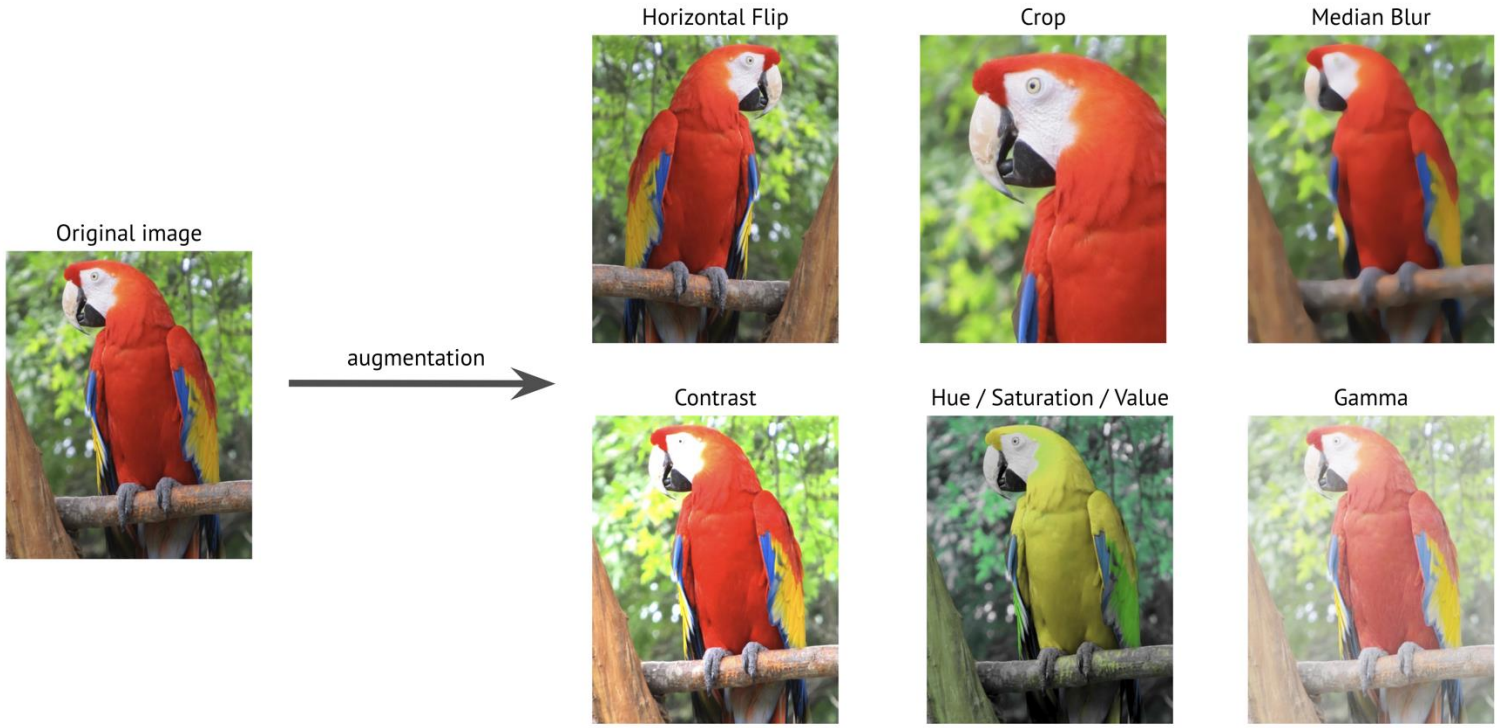
$$BN(x) = \frac{x - \hat{\mu}_B}{\hat{\sigma}_B}$$

  - $\hat{\mu}_B$ is the sample mean and $\hat{\sigma}_B$ is the sample standard deviation of the minibatch B
- Batch normalization is an effective technique that consistently accelerates the convergence of deep networks
- Batch normalization is usually inserted after Fully Connected or Convolutional layers, and before nonlinearity



Data Augmentation

Data Augmentation is used to create additional training data to make training more robust. The idea is to randomly transform/generate data while training.



Data Augmentation

The transformation does not change the key properties of an image for the purpose of image classification

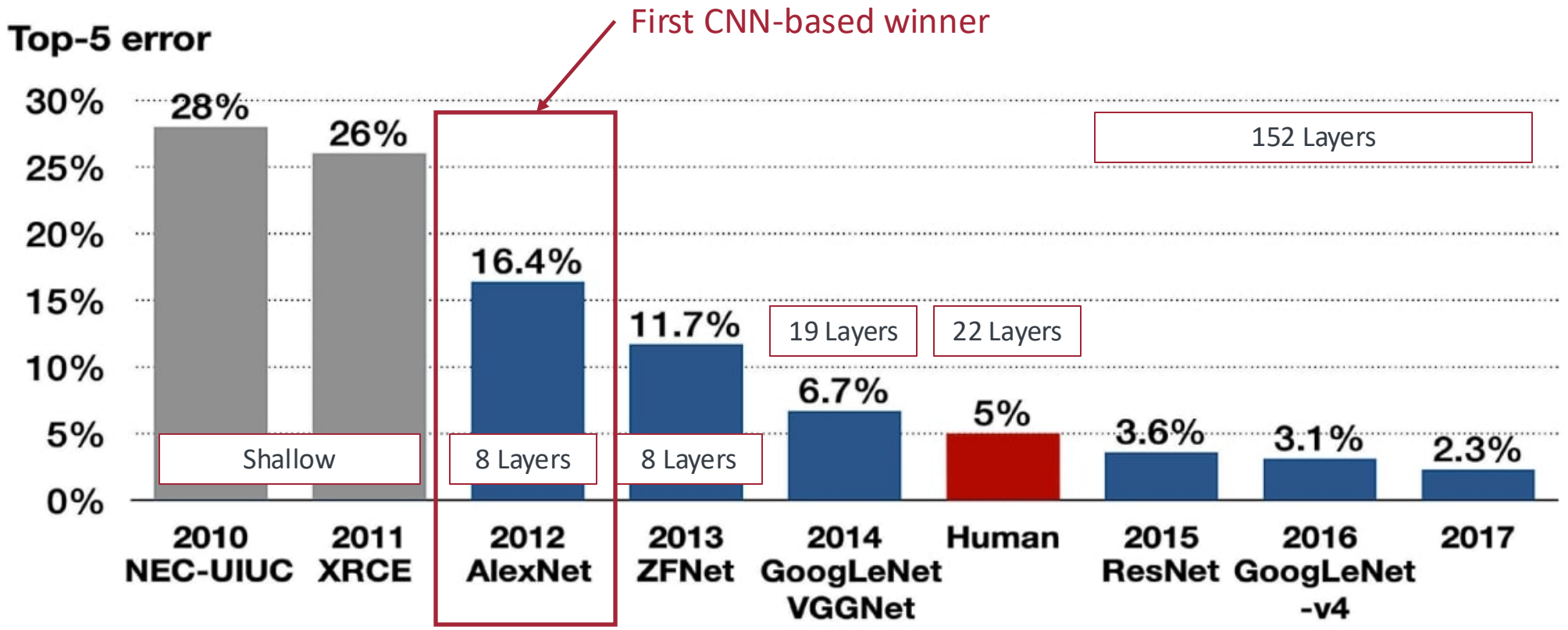
- Recall the three observations we made for images and image classification
  - Local patterns (much smaller than the whole image)
  - The same patterns appear in different regions
  - Subsampling the pixels will not change the object

CNN Architectures

CNN Architectures

- Components of CNNs
  - Convolution layers
  - Pooling layers
  - Fully-connected layers
  - Activation function
  - Normalization
- CNN Architectures: Combination of different components
  - AlexNet
  - VGG
  - ResNet

ImageNet Winners



AlexNet (2012)

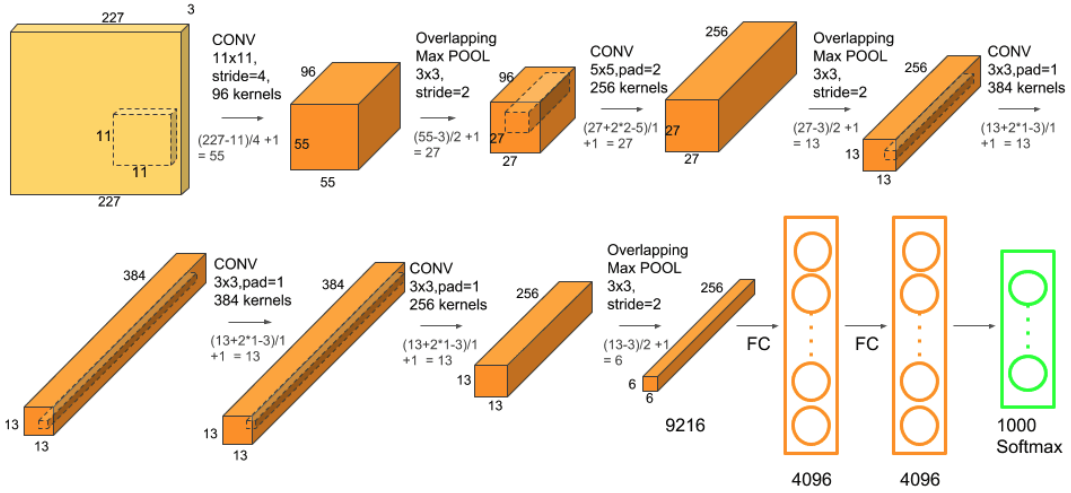
Imagenet classification with **deep convolutional** neural **networks**

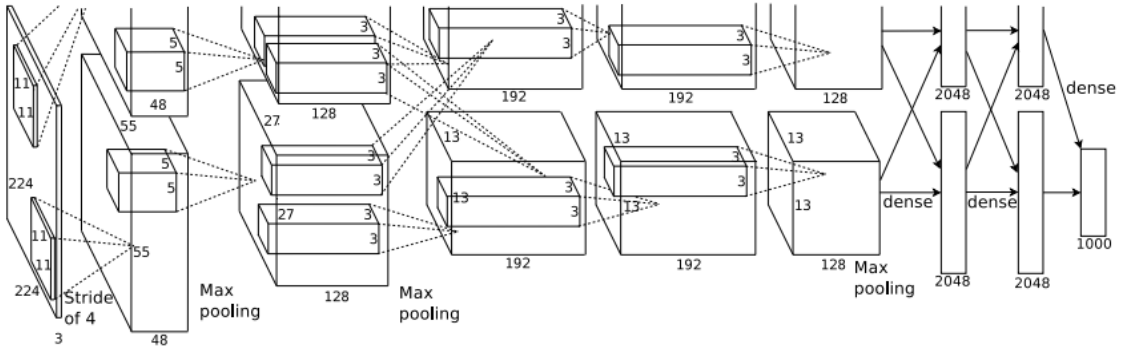
A Krizhevsky, I Sutskever… - Advances in neural …, 2012 - proceedings.neurips.cc

… We trained a large, **deep convolutional** neural **network** to **classify** the 1.2 million high-resolution images in the **ImageNet** LSVRC-2010 contest into the 1000 different classes. On the test …

☆ Save Cite Cited by 138265 Related articles All 98 versions »

AlexNet demonstrated a substantial improvement in ImageNet classification error over the best non-CNN method (25.8% -> 16.4%, human error = 5.1%)

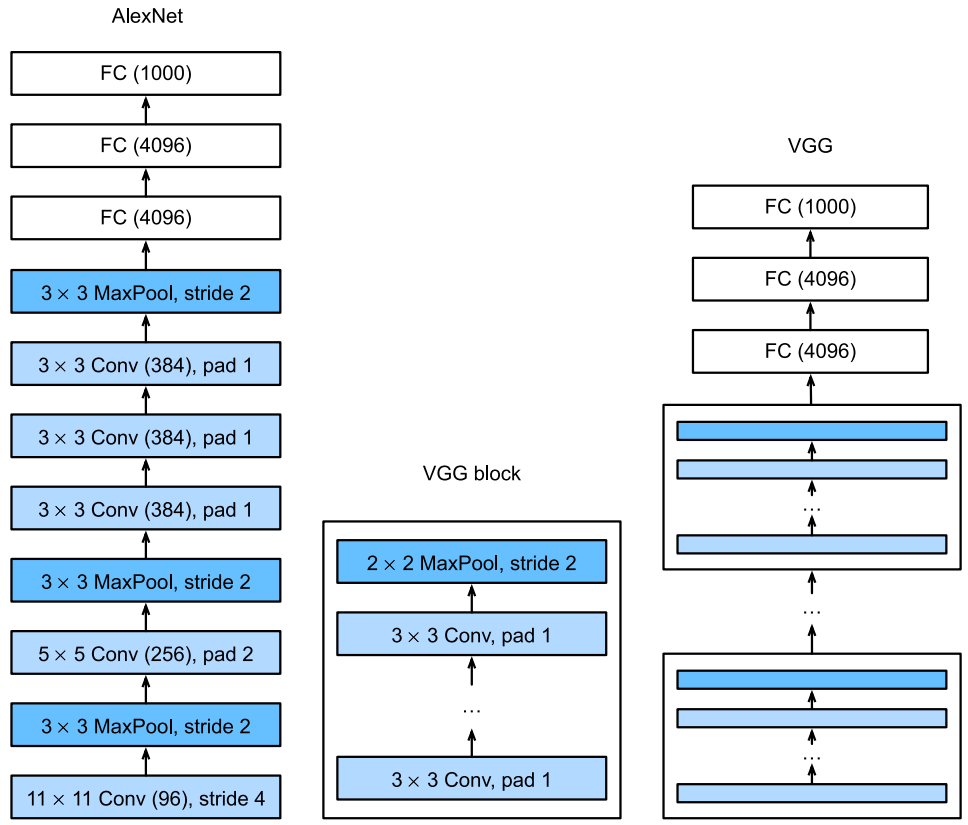




VGGNet (2014)

Very deep convolutional networks for large-scale image recognition
K Simonyan, A Zisserman - arXiv preprint arXiv:1409.1556, 2014 - arxiv.org
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 …
☆ Save Cite Cited by 137497 Related articles All 41 versions »

- AlexNet vs. VGGNet
  - VGG has small filters and deeper networks: 8 layers (AlexNet) to 19 layers (VGGNet)
  - VGG consists of blocks of layers, whereas AlexNet's layers are all designed individually

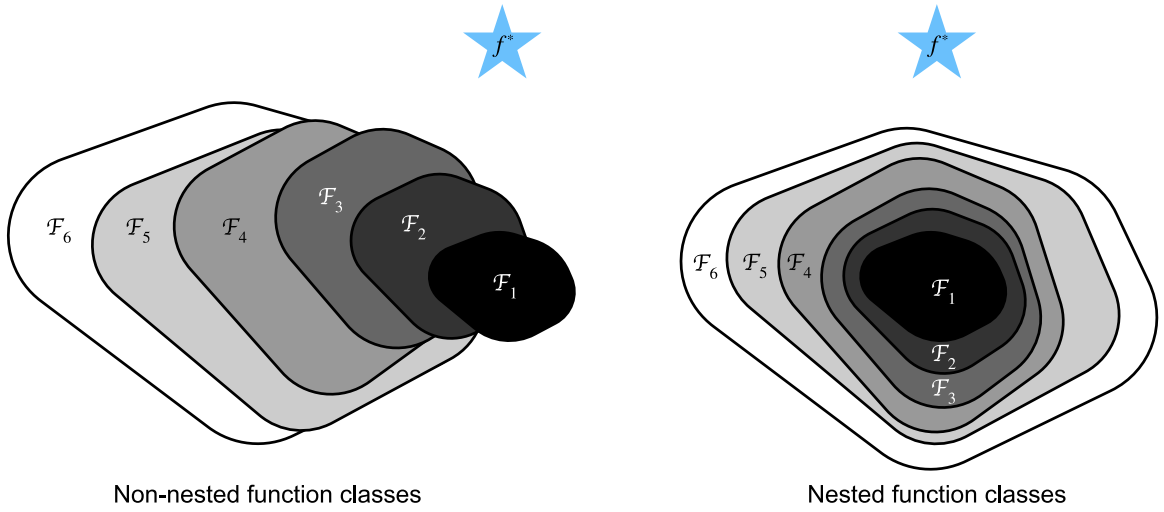


ResNet (2015)

Deep residual learning for image recognition
K He, X Zhang, S Ren, J Sun - Proceedings of the IEEE …, 2016 - openaccess.thecvf.com
… This result won the 1st place on the ILSVRC **2015** classification task. We also present …
Deep residual nets are foundations of our submissions to ILSVRC & COCO **2015** competitions1, …
☆ Save 99 Cite Cited by 255149 Related articles All 65 versions »

Motivation

- Very deep networks using residual connections: 152-layer model for ImageNet
- What happens when we continue stacking deeper layers on a "plain" convolutional neural network?
  - Solving the optimization problem becomes an issue
- A larger size function class (indicated by the area) does not guarantee we will get closer to the "truth" function (f^*). This issue can be avoided by designing the network to be nested function classes.



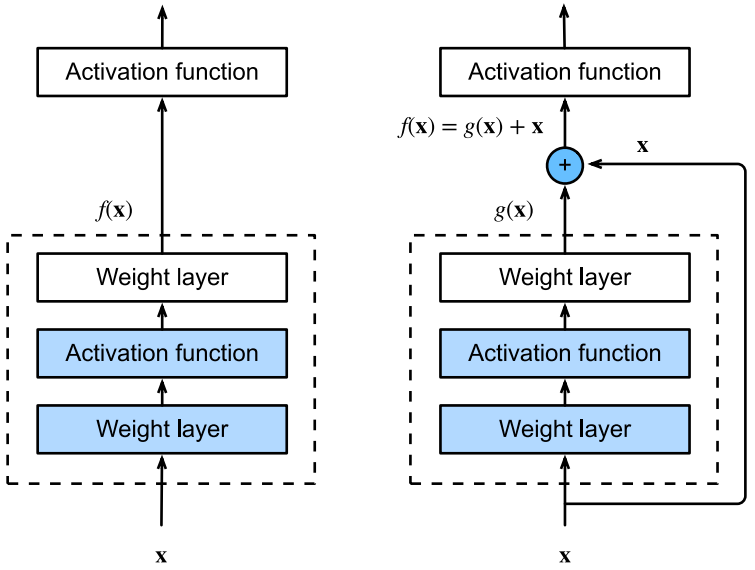
Non-nested function classes

Nested function classes

ResNet (2015)

Deep residual learning for image recognition
K He, X Zhang, S Ren, J Sun - Proceedings of the IEEE …, 2016 - openaccess.thecvf.com
… This result won the 1st place on the ILSVRC **2015** classification task. We also present …
Deep residual nets are foundations of our submissions to ILSVRC & COCO **2015** competitions1, …
☆ Save 99 Cite Cited by 255149 Related articles All 65 versions »

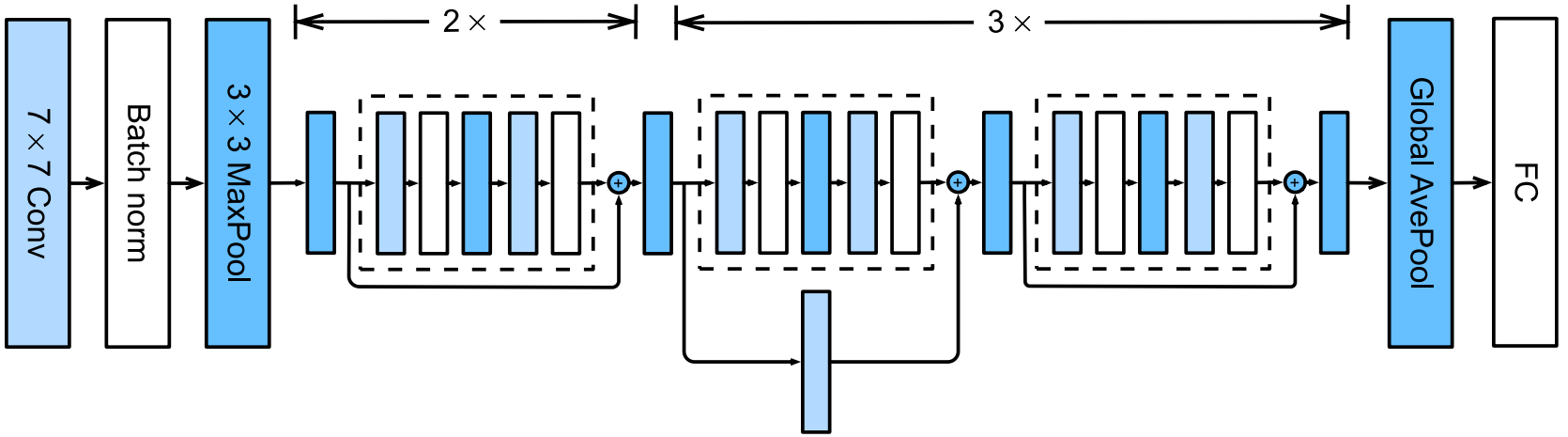
Residual Blocks: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



ResNet (2015)

Deep residual learning for image recognition
K He, X Zhang, S Ren, J Sun - Proceedings of the IEEE …, 2016 - openaccess.thecvf.com
… This result won the 1st place on the ILSVRC **2015** classification task. We also present …
Deep residual nets are foundations of our submissions to ILSVRC & COCO **2015** competitions1, …
☆ Save 99 Cite Cited by 255149 Related articles All 65 versions »

ResNet-18; The residual blocks can be stacked to create CNNs with deeper size, such 34, 50, 101, and 152.



Main Takeaways

- **AlexNet** shows that we can use CNNs to train Computer Vision models
- **VGG** shows that bigger networks work better
- **ResNet** shows us how to train extremely deep networks
  - Limited only by GPU & memory!
  - Diminishing returns as networks got bigger
- ResNets are good defaults to use (still true today in February 2025)

Transfer Learning

Do you need a lot of data to train/use CNNs?

Feature Learning

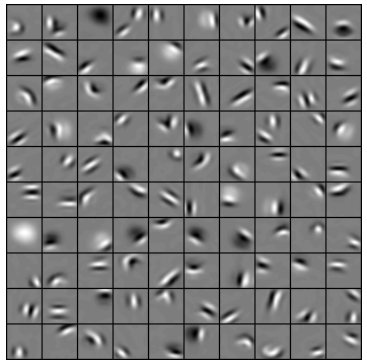
Standard CNN trained on big dataset

Predictions

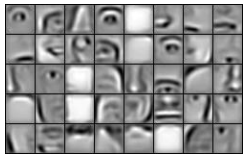
0.03	dog
0.8	cat
...	...
0.02	car
0.1	table

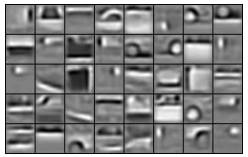
1000 classes

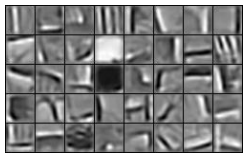
Combination of pixels



Combination of edges



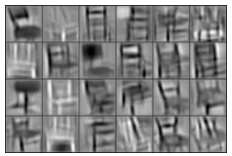




Combination of patterns, edges

face

car



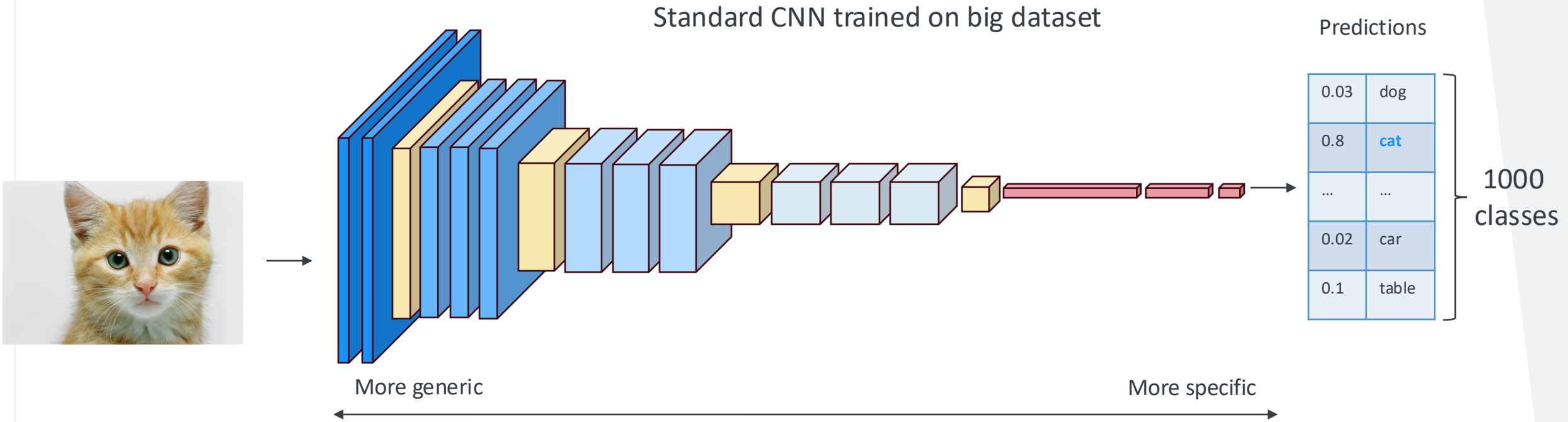
chair

Combination of parts

Take the high-level features extracted by the convolutional layers and combine them to make a final prediction

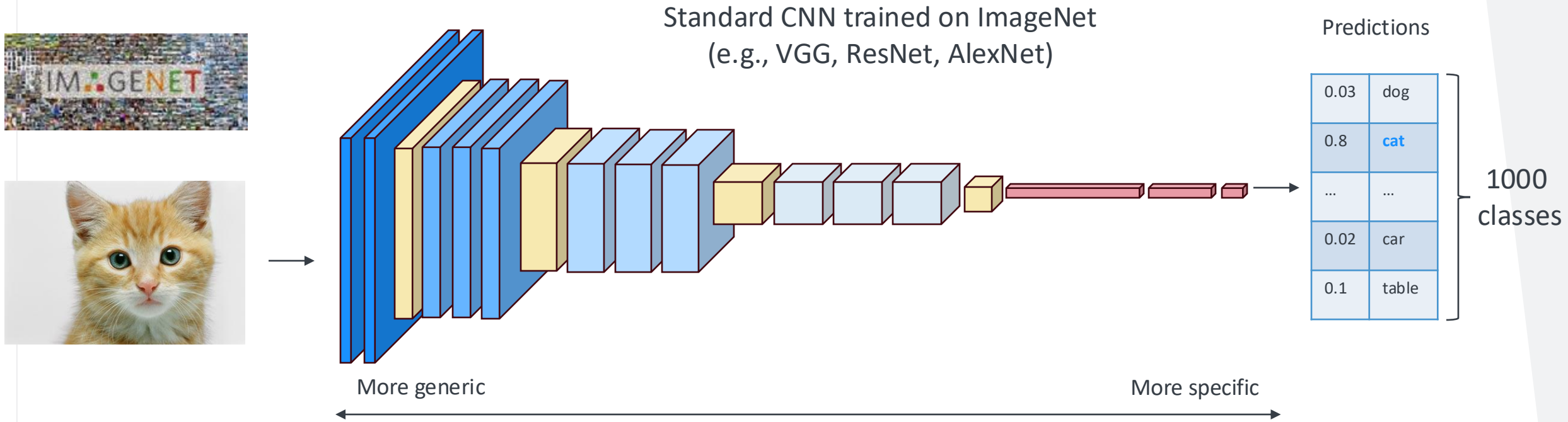
Reference: Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML* (pp. 609-616).

Feature Learning



- We expect the learned weights from a trained CNN to embody knowledge about the characteristics of the millions of everyday images the network was trained on

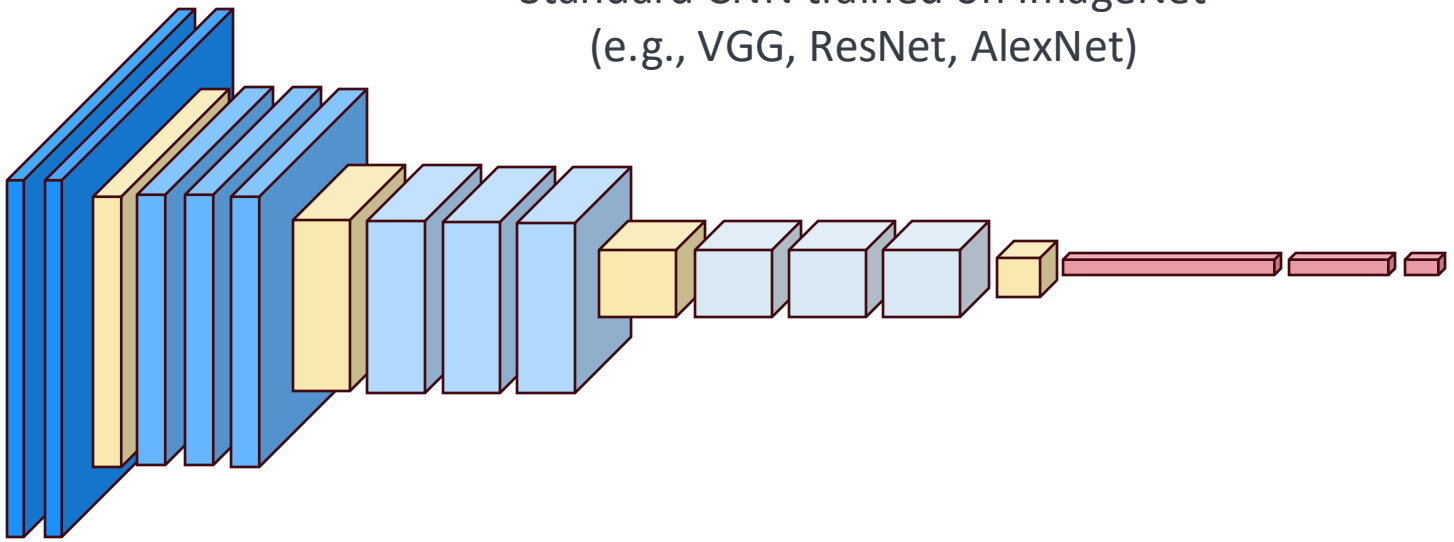
Transfer Learning



- We expect the learned weights from a trained CNN to embody knowledge about the characteristics of the millions of everyday images the network was trained on

Transfer Learning

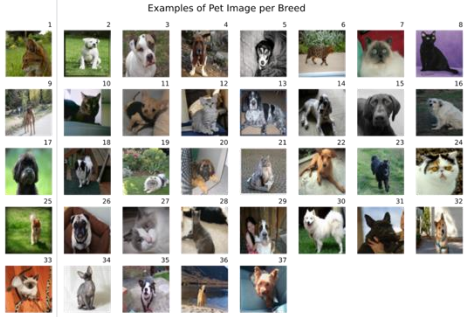
Standard CNN trained on ImageNet
(e.g., VGG, ResNet, AlexNet)



Predictions

0.03	dog
0.8	cat
...	...
0.02	car
0.1	table

1000 classes



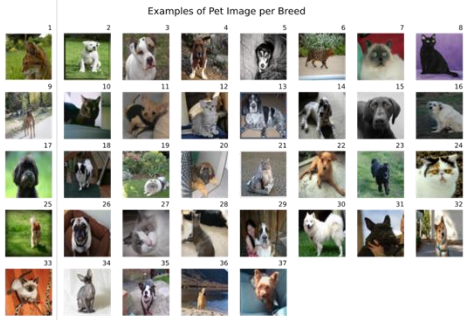
Cat and dog dataset

Predictions

...	dog
...	cat

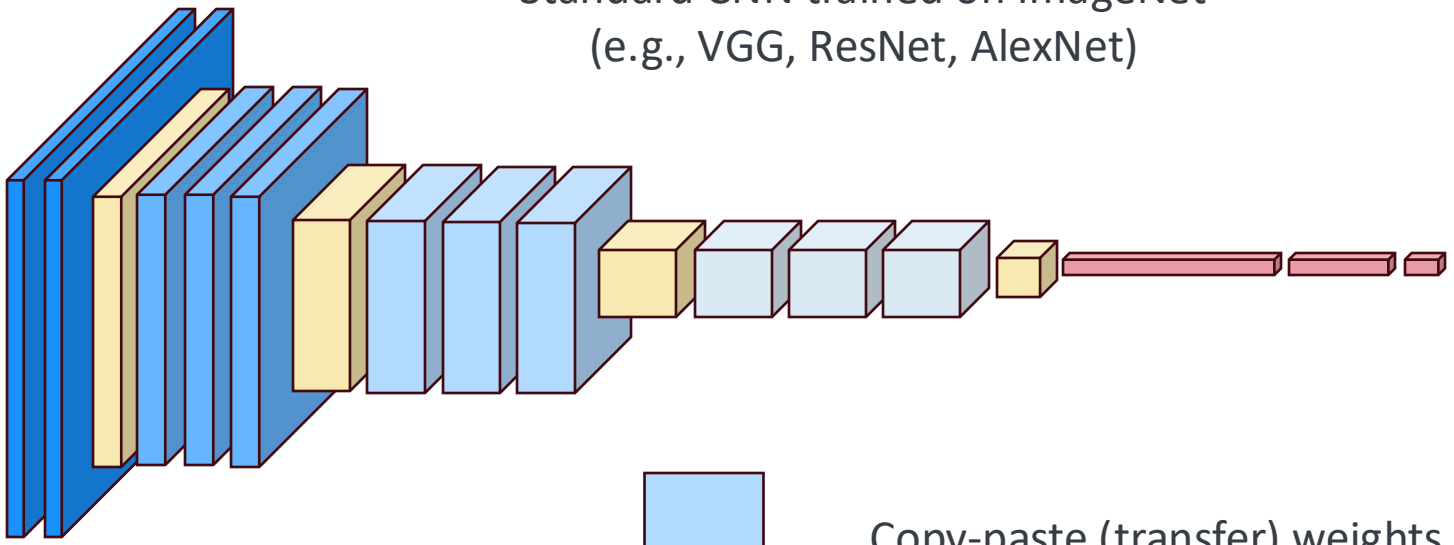
2 classes

Transfer Learning



Cat and dog dataset

Standard CNN trained on ImageNet
(e.g., VGG, ResNet, AlexNet)

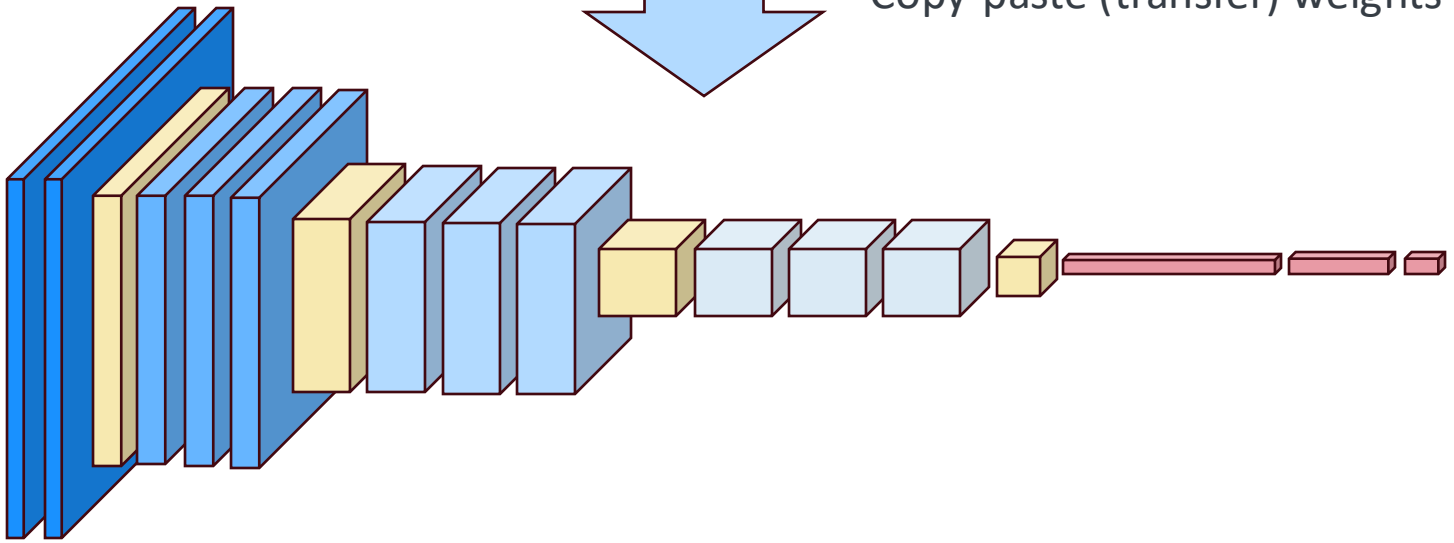


Predictions

0.03	dog
0.08	cat
...	...
0.02	car
0.1	table

1000 classes

Copy-paste (transfer) weights

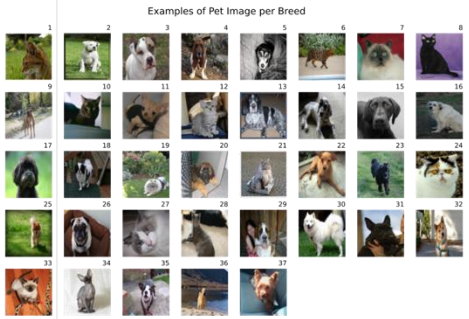


Predictions

...	dog
...	cat
...	...
...	car
...	table

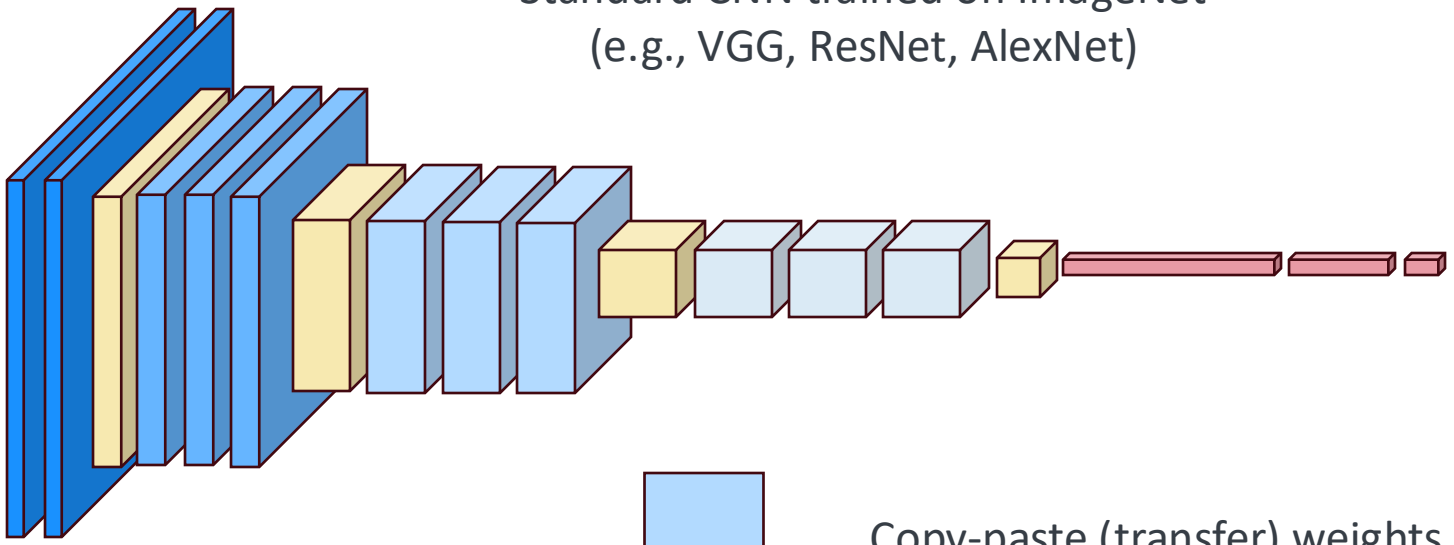
1000 classes

Transfer Learning



Cat and dog dataset

Standard CNN trained on ImageNet
(e.g., VGG, ResNet, AlexNet)

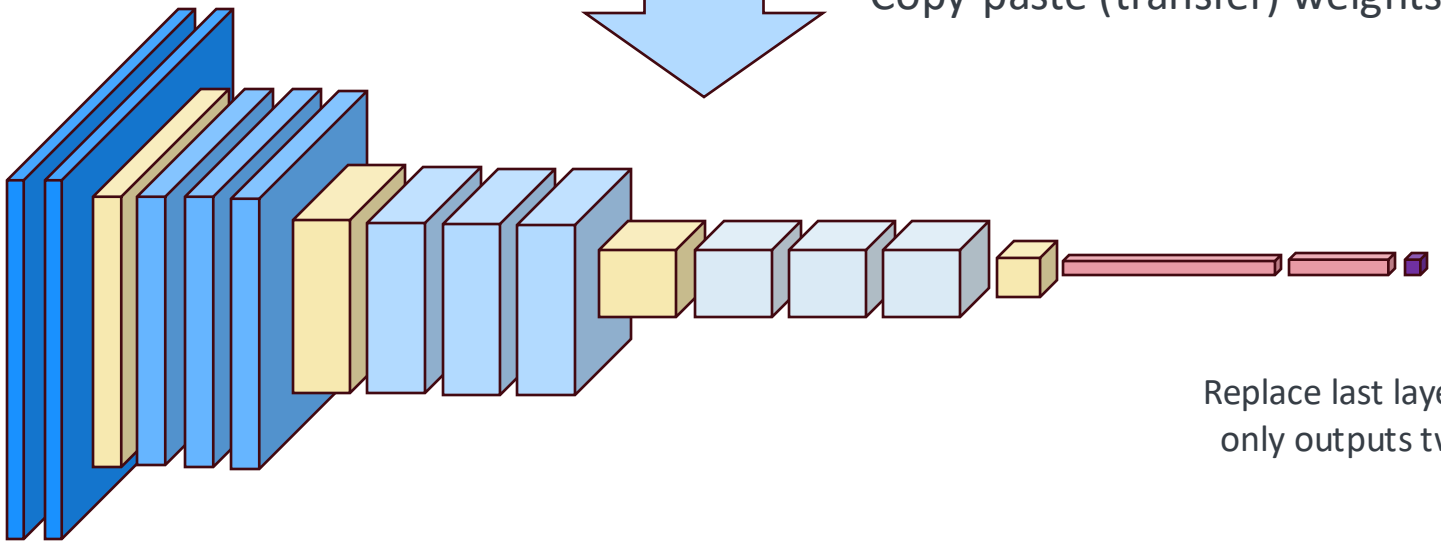


Predictions

0.03	dog
0.08	cat
...	...
0.02	car
0.1	table

1000 classes

Copy-paste (transfer) weights



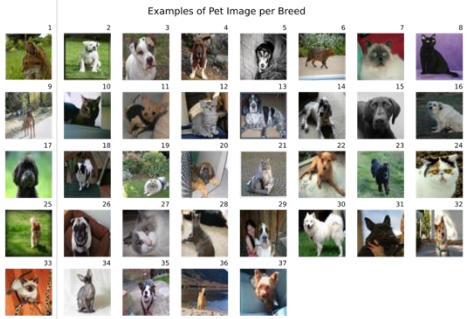
Predictions

...	dog
...	cat

2 classes

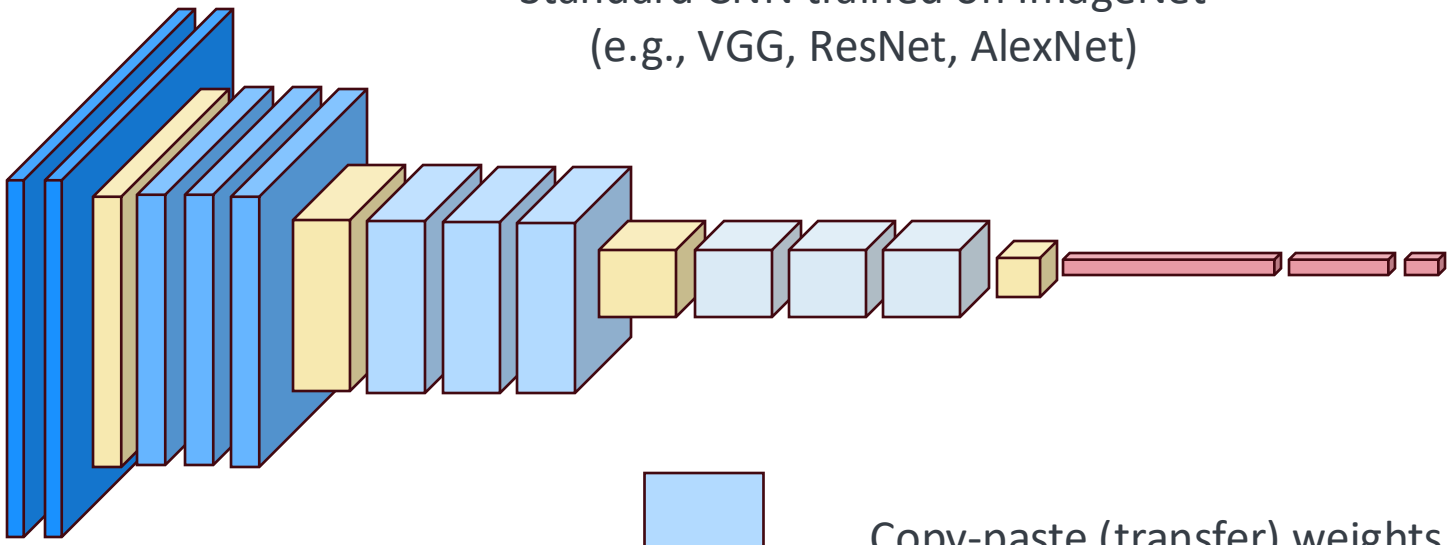
Replace last layer so that it
only outputs two classes

Transfer Learning



Cat and dog dataset

Standard CNN trained on ImageNet
(e.g., VGG, ResNet, AlexNet)

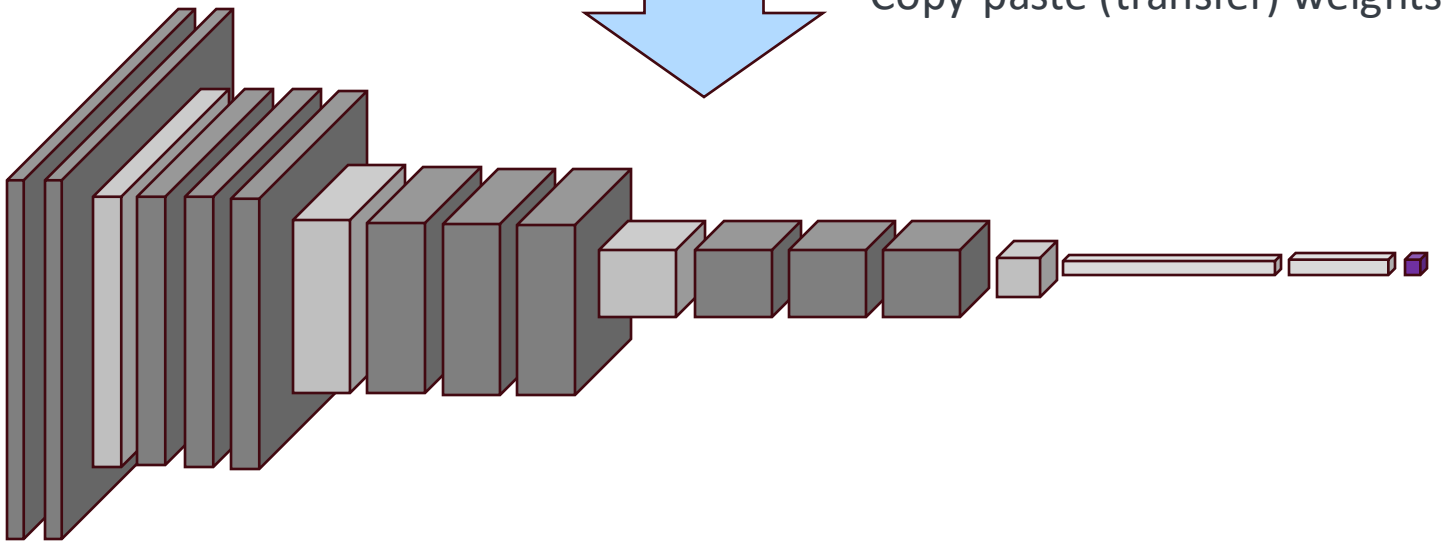


Predictions

0.03	dog
0.08	cat
...	...
0.02	car
0.1	table

1000 classes

Copy-paste (transfer) weights



Predictions

...	dog
...	cat

2 classes

Freeze

Train

Transfer Learning

Standard CNN trained on ImageNet
(e.g., VGG, ResNet, AlexNet)

Predictions

0.03	dog
0.08	cat
...	...
0.02	car
0.1	table

1000 classes

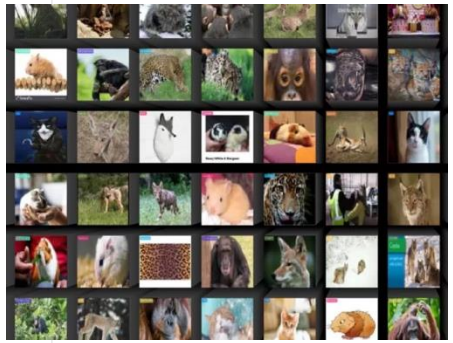
Copy-paste (transfer) weights

Predictions

...	dog
...	cat
...	...
...	lion
...	fox

1000 classes

Custom fully-connected layers



Animal dataset

Freeze

Train

Transfer Learning

Standard CNN trained on ImageNet
(e.g., VGG, ResNet, AlexNet)

Predictions

0.03	dog
0.08	cat
...	...
0.02	car
0.1	table

1000 classes

Copy-paste (transfer) weights

Predictions

...	...
...	...
...	...
...	...
...	...

4244 classes

Custom fully-connected layers

Fine-grained bird dataset

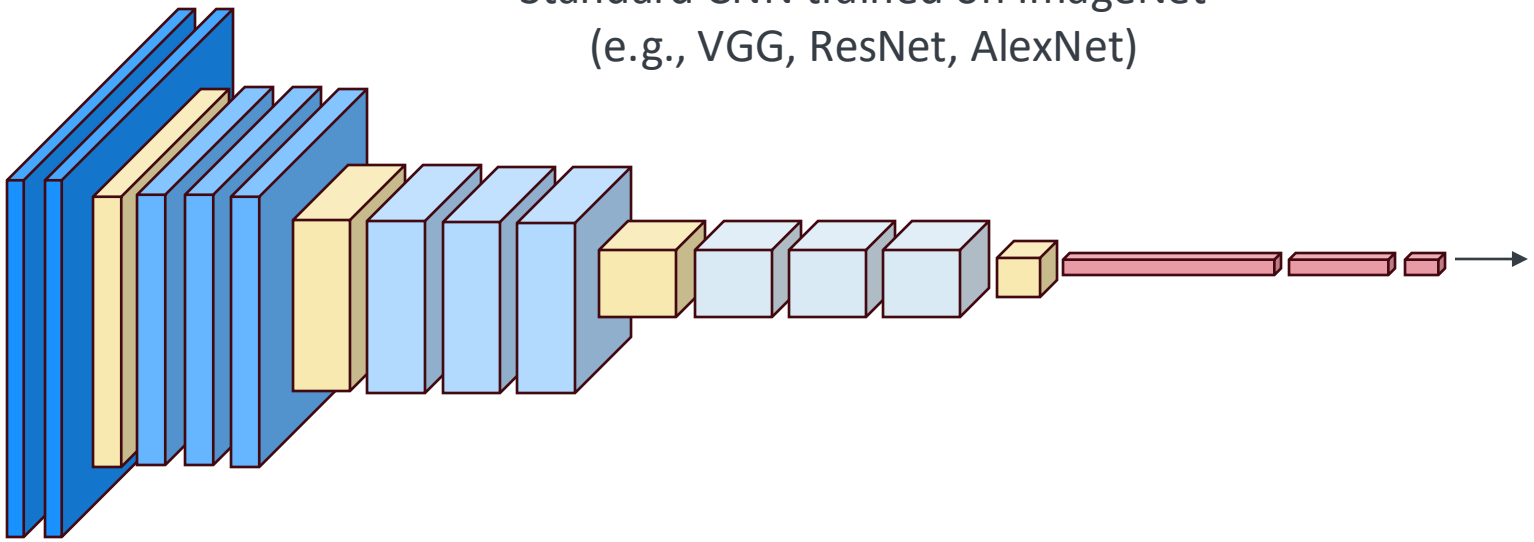
Freeze

Fine-tune (retrain with small learning rate)

Train

Transfer Learning

Standard CNN trained on ImageNet
(e.g., VGG, ResNet, AlexNet)



Predictions

0.03	dog
0.08	cat
...	...
0.02	car
0.1	table

1000 classes

More generic ⟷ More specific

Transfer learning strategies	Very similar dataset	Very different dataset
Very little data	Replace linear classifier on top layer	Challenging… Try linear classifier from different stages
Quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Acknowledgement

The lecture note has benefited from various resources, including those listed below. Please contact Zonghao Yang (zyang99@stevens.edu) with any questions or concerns about the use of these materials.

- Lecture Notes on Convolutional Neural Networks and Transfer Learning by Léonard Boussioux at University of Washington
- Lecture Notes on Convolutional Neural Networks from ML 2021 Spring by Hung-Yi Lee at National Taiwan University
- Lecture Notes on Image Classification with CNNs from CS231n 2024 Spring at Stanford University
- Lecture Notes on Convolutional Neural Networks from DSME6635 2024 Spring by Renyu Zhang at Chinese University of Hong Kong

STEVENS
INSTITUTE OF TECHNOLOGY
1870

THANK **YOU**

Stevens Institute of Technology
1 Castle Point Terrace, Hoboken, NJ 07030