CS502: Deep Learning in Biomedicine
Convolutional Neural Networks

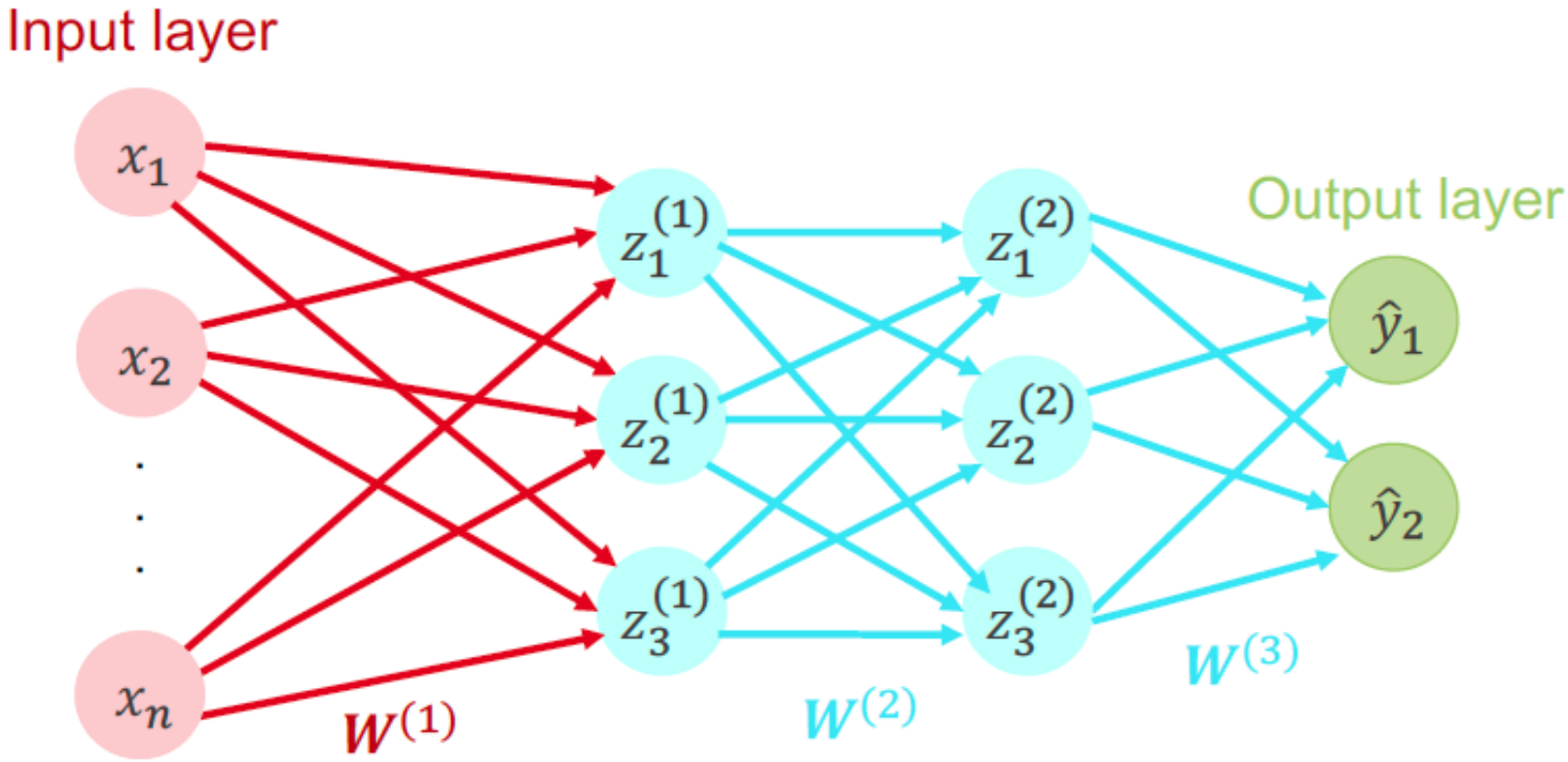
Maria Brbić

Fall 2023

École
polytechnique
fédérale
de Lausanne

Last Week: Recap

- What we covered last time: Fully connected neural networks



Last Week: Recap

What we covered last time

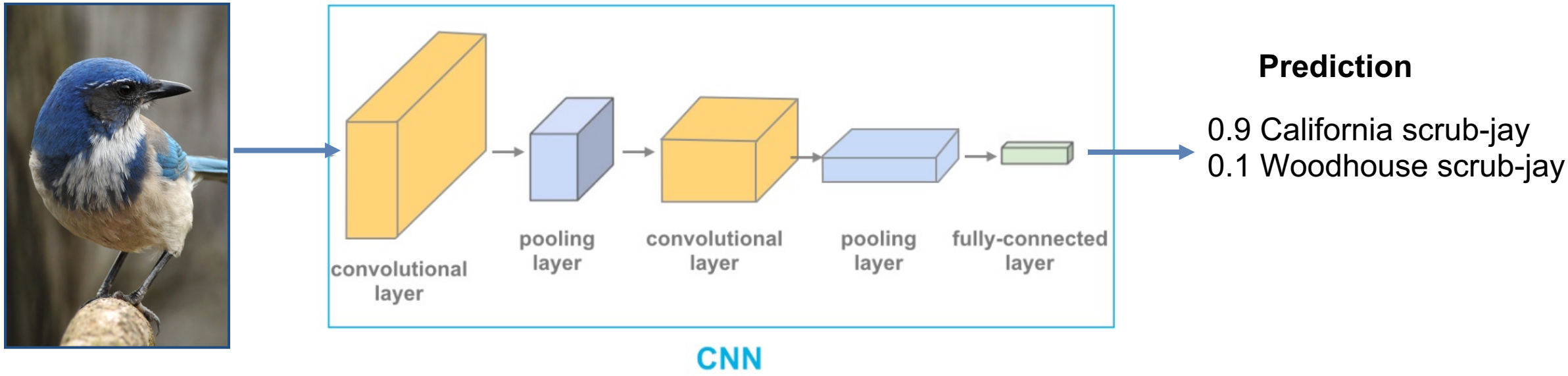
- **Objective function:**

$$\min_{\Theta} \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}))$$

- Sample a minibatch of input $\boldsymbol{x}$
- **Forward propagation:** compute $\mathcal{L}$ given $\boldsymbol{x}$
- **Back-propagation:** obtain gradient $\nabla_{\Theta}\mathcal{L}$ using a chain rule
- Use **stochastic gradient descent (SGD)** to optimize for Θ over many iterations
- f can be any neural network

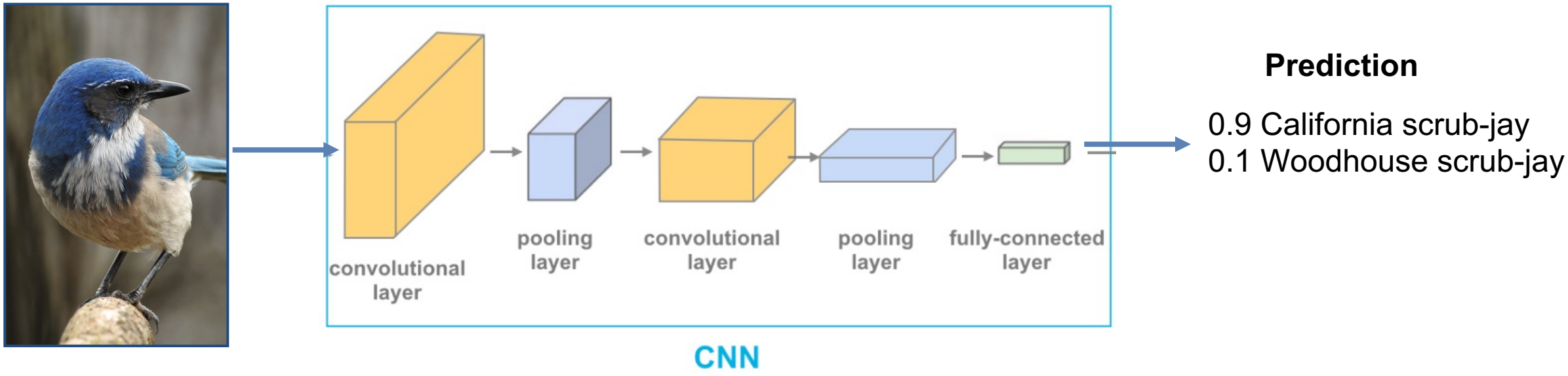
Today: Convolutional Neural Networks (CNNs)

- Default architecture to handle image datasets



Two Main Building Blocks

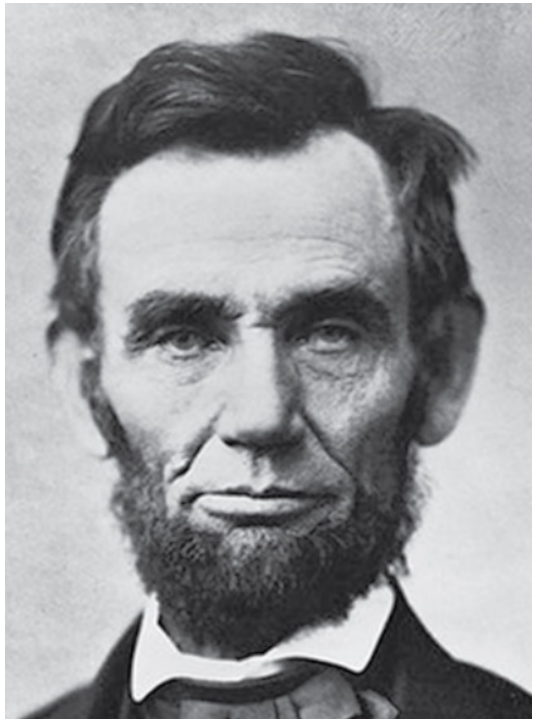
1. Convolutional layer
2. Pooling layer



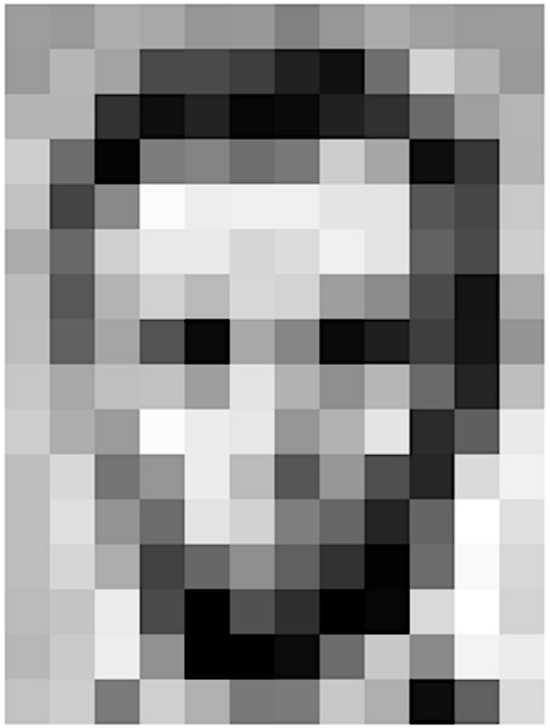
Images Are Matrices

- An image is just a matrix of numbers
- Range: [0,255]

Original image



Low-resolution image



Pixel intensity

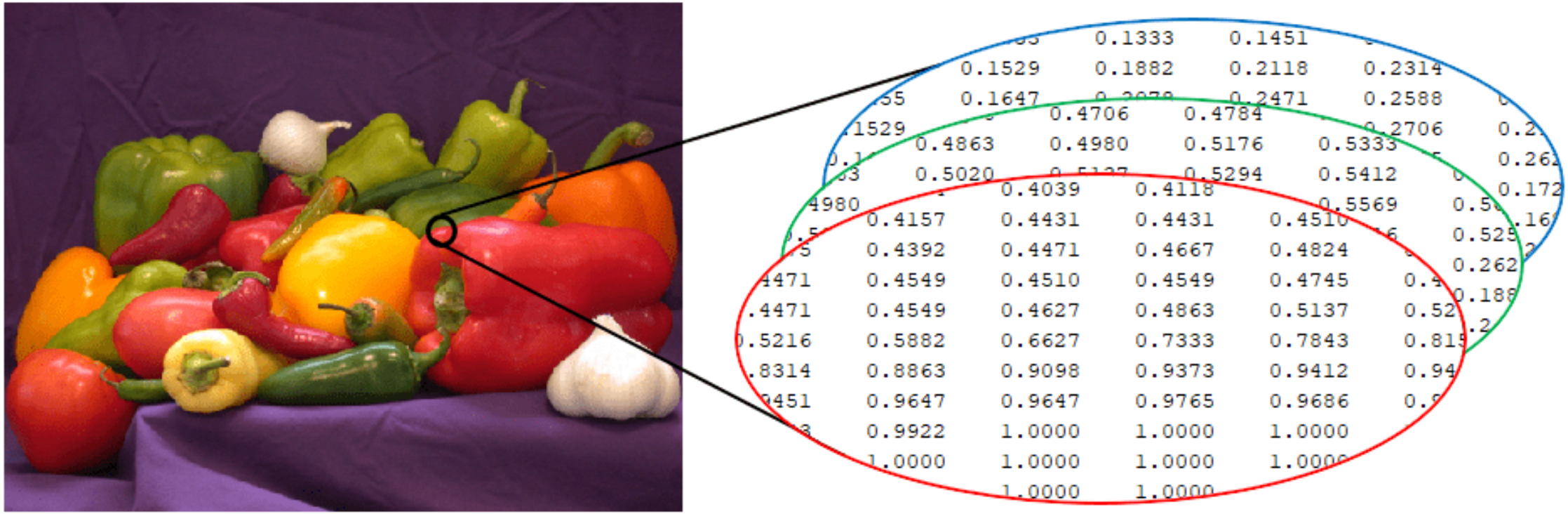
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

Stored as matrix

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

Images Are Matrices

- RGB image, e.g., 550 x 400 x **3**

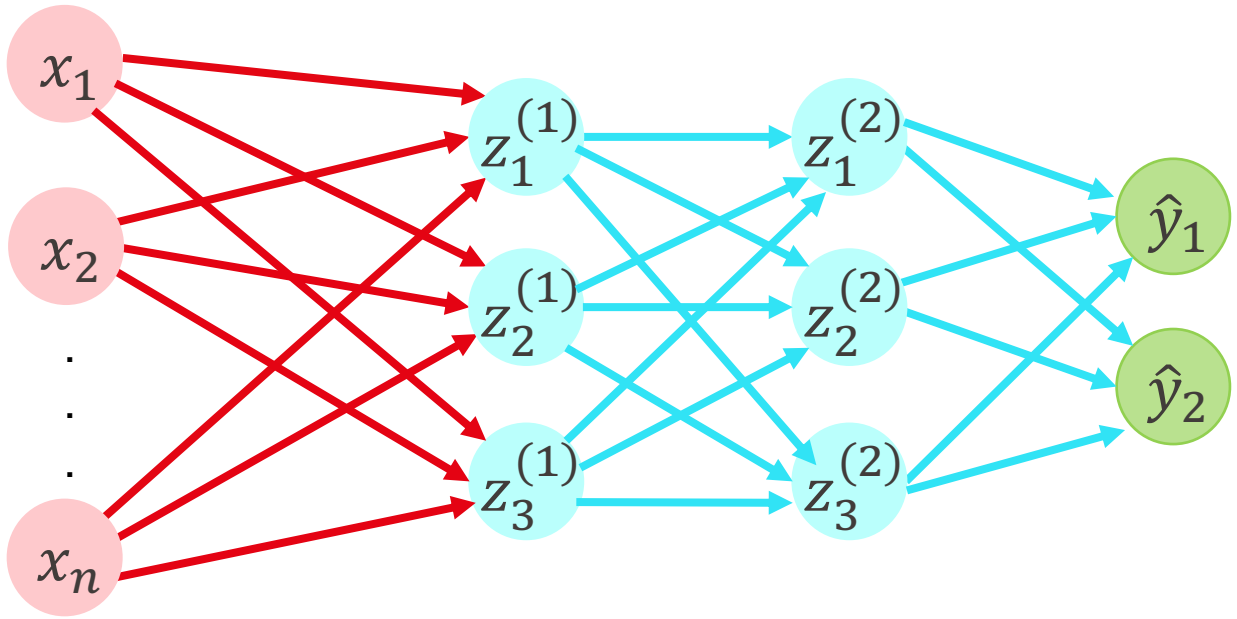


Why Do We Need CNNs?

So far, fully connected neural networks:

What if input is an image?

157	153	174	168	150	152
155	182	163	74	75	62
180	180	50	14	34	6
206	109	5	124	131	111
194	68	137	251	237	239
172	105	207	233	233	214
188	88	179	209	209	215
189	97	165	84	84	168

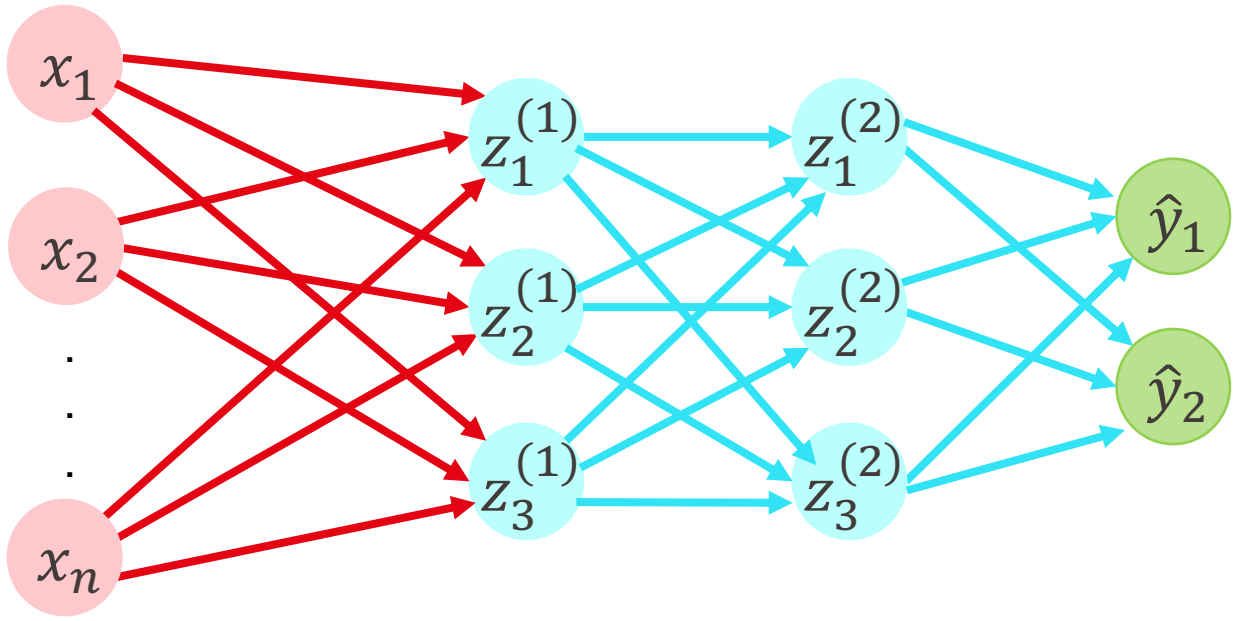


Why Do We Need CNNs?

So far, fully connected neural networks:

Flatten image?

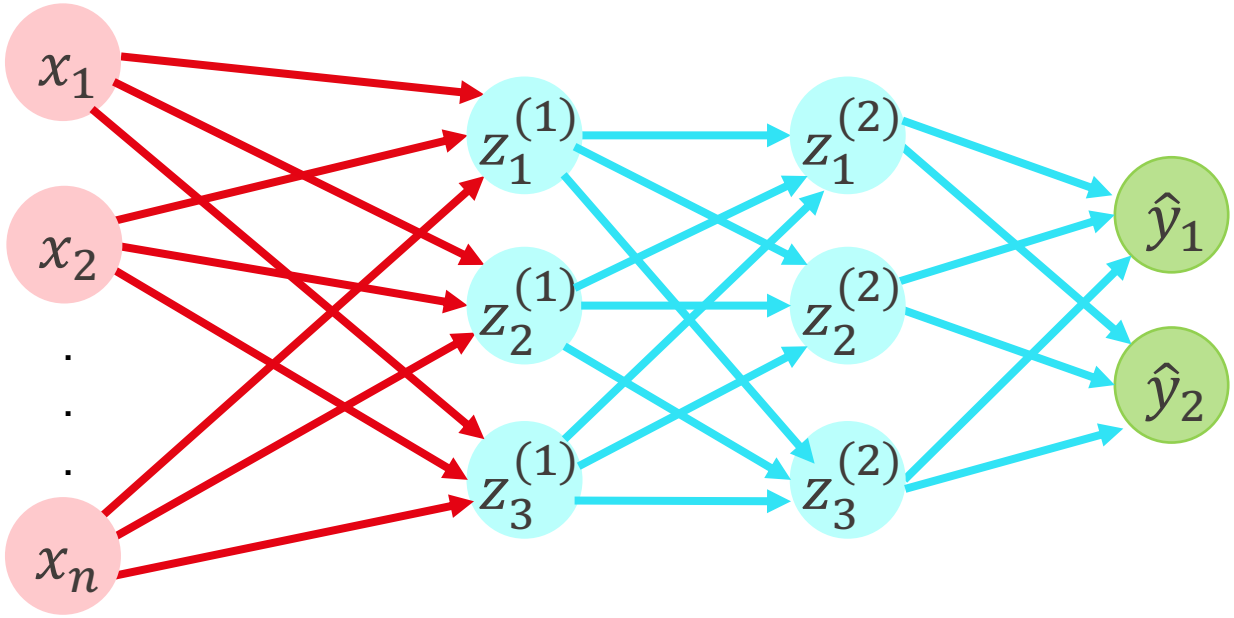
157
155
180
206
194
172
188
189
…
168
215



Why Do We Need CNNs?

Flatten image?

157
155
180
206
194
172
188
189
…
168
215



Problems:

1) Does not scale to large images
2) Spatial information is not preserved

Example:

200 x 200 x 3 image would have 120,000 weights in the first layer!

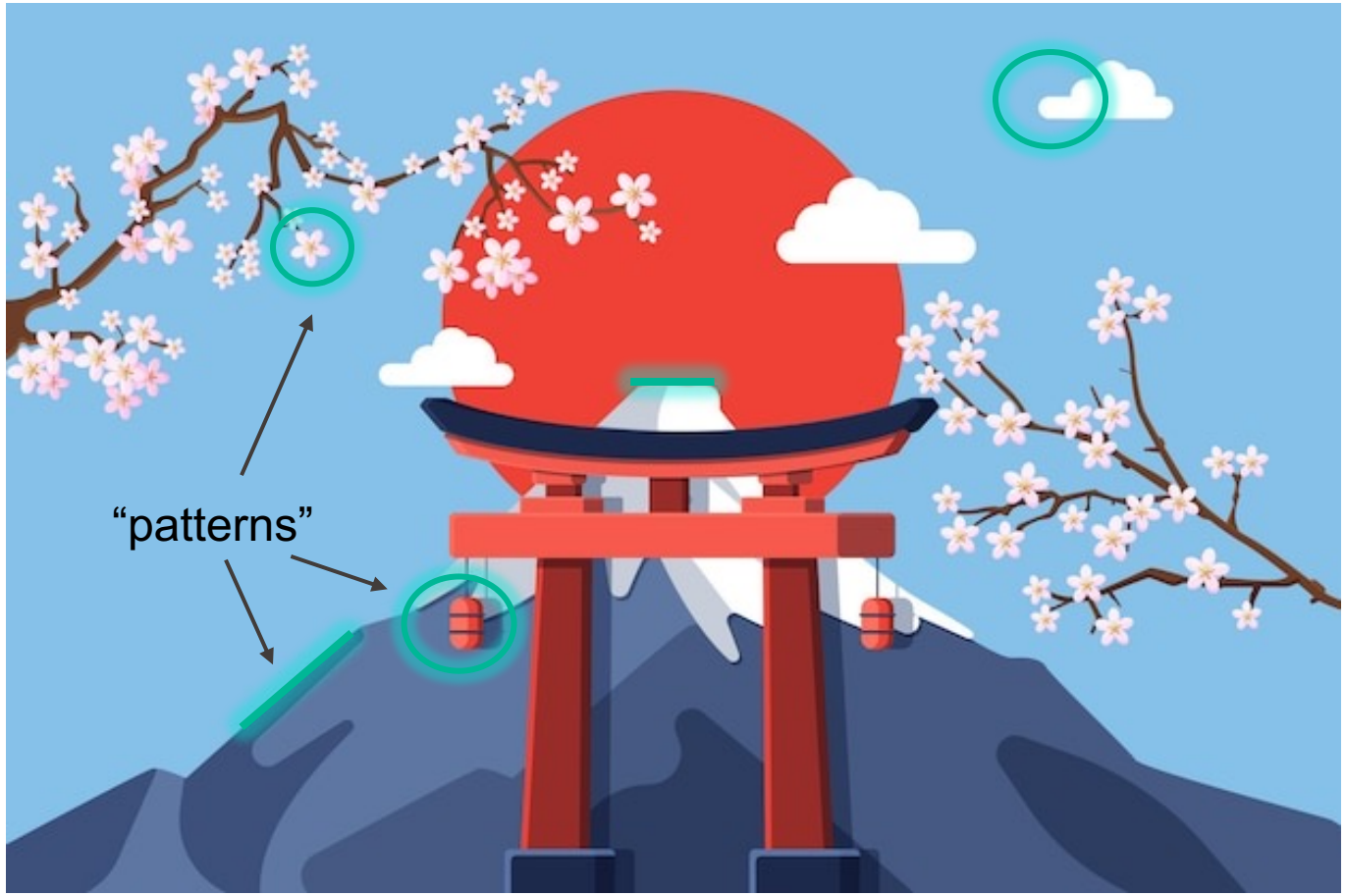
Solution: Convolutional Layer

Three main ideas:

1. **Sparse connections**
   - Use filters/kernels of smaller size than the image that detect meaningful features/patterns in the image
2. **Parameter sharing**
   - Reuse same parameters across locations instead of learning a separate set of parameters for every location
3. **Equivariant representations**
   - A function $f(x)$ is equivariant to a function g, if $f\big(g(x)\big) = g(f(x))$
   - Convolution is equivariant to translation: if we move the object in the input, its representation will move the same amount in the output

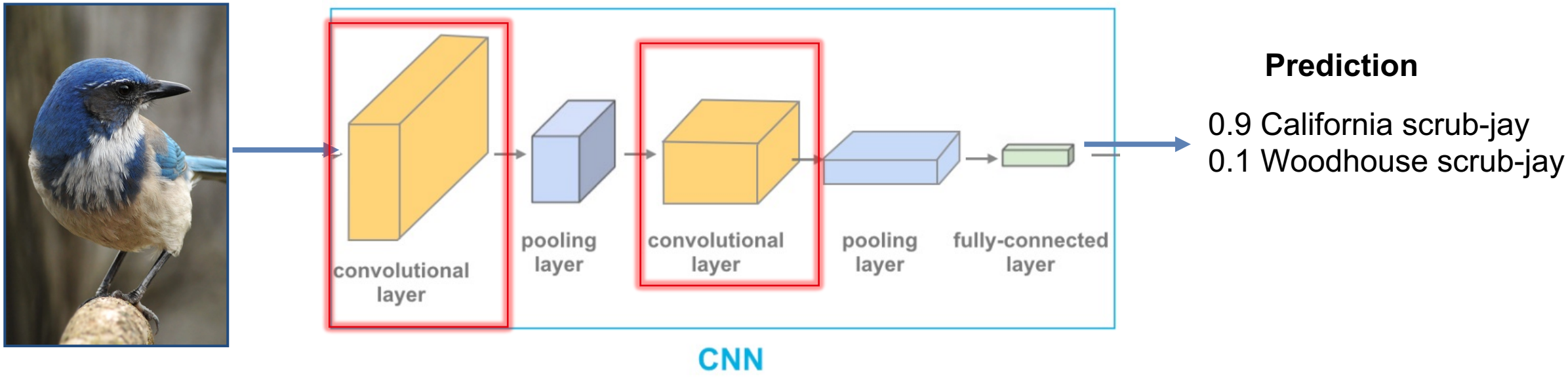
Image: Many Patterns

- Edges, shapes, objects



Two Main Building Blocks

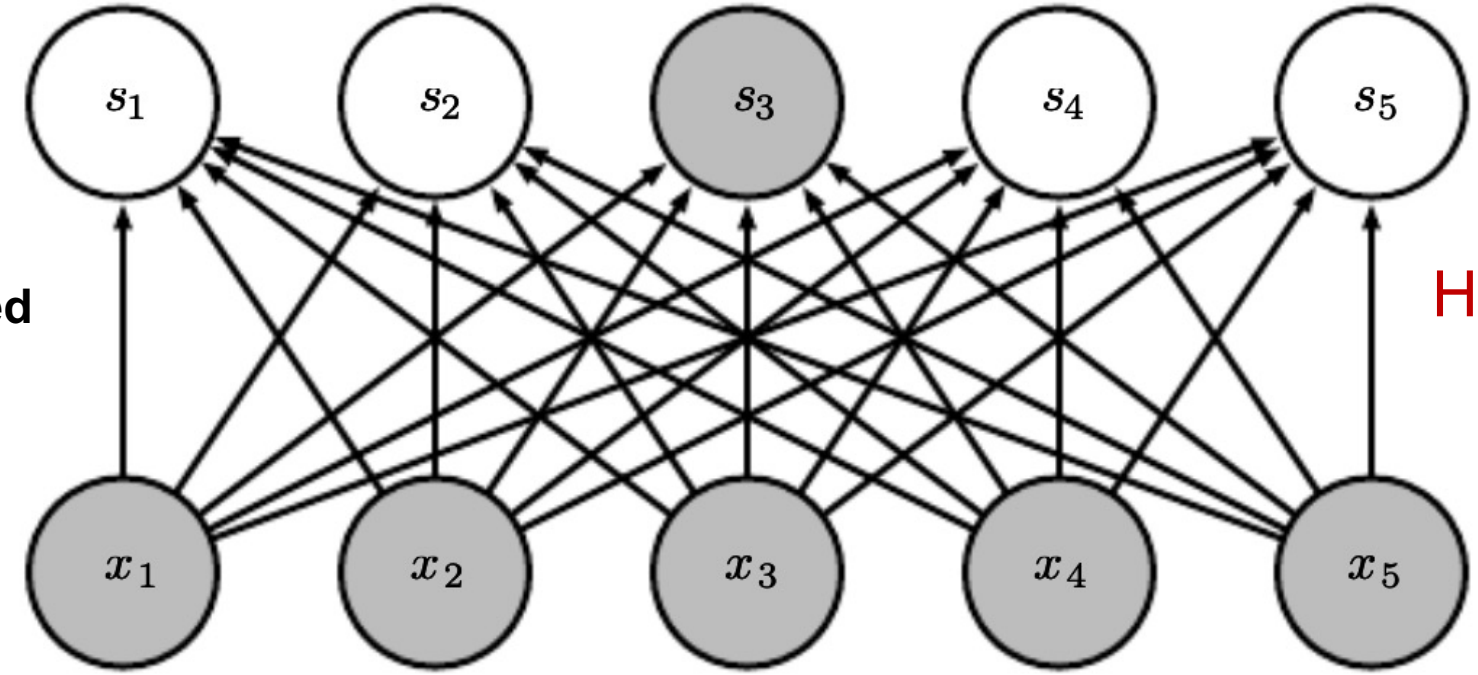
1. **Convolutional layer**
2. Pooling layer



Convolutional Layer: Sparse Connections

Connect each neuron to only a local region of the input

Instead of



Fully connected layer

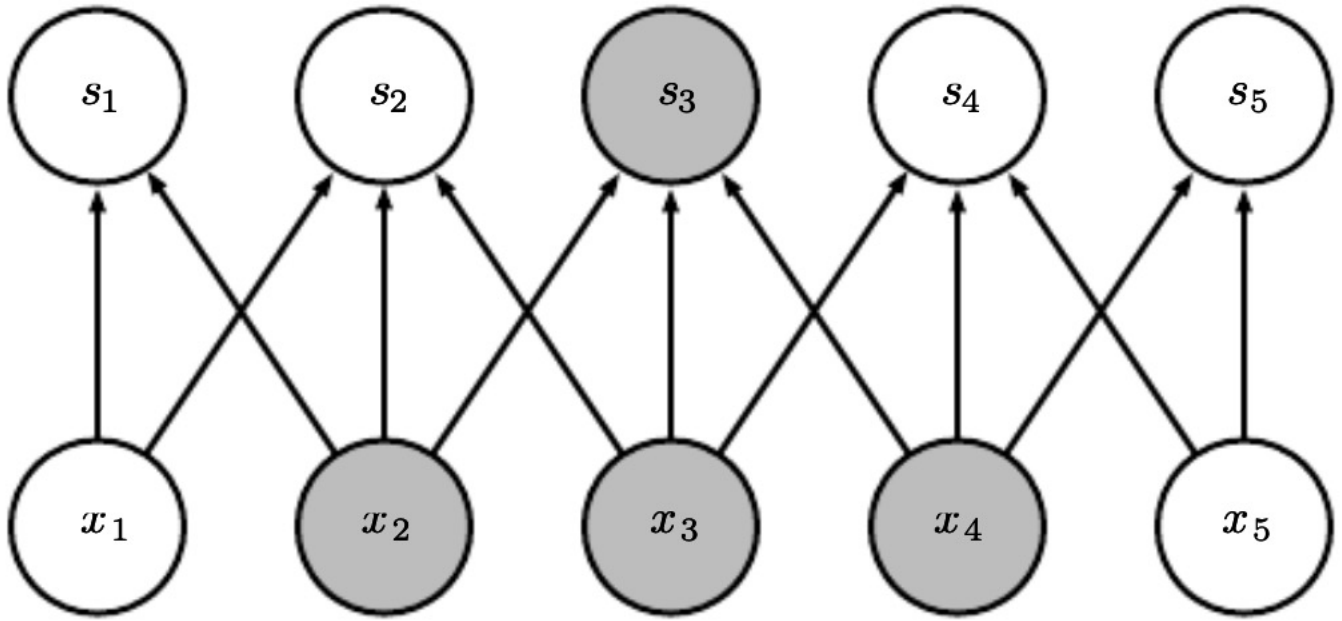
How many parameters do we have?

Convolutional Layer: Sparse Connections

Connect each neuron to only a local region of the input

Sparse connectivity

Only x_2, x_3, x_4 affect s_3 (receptive field of s_3)



Convolutional layer

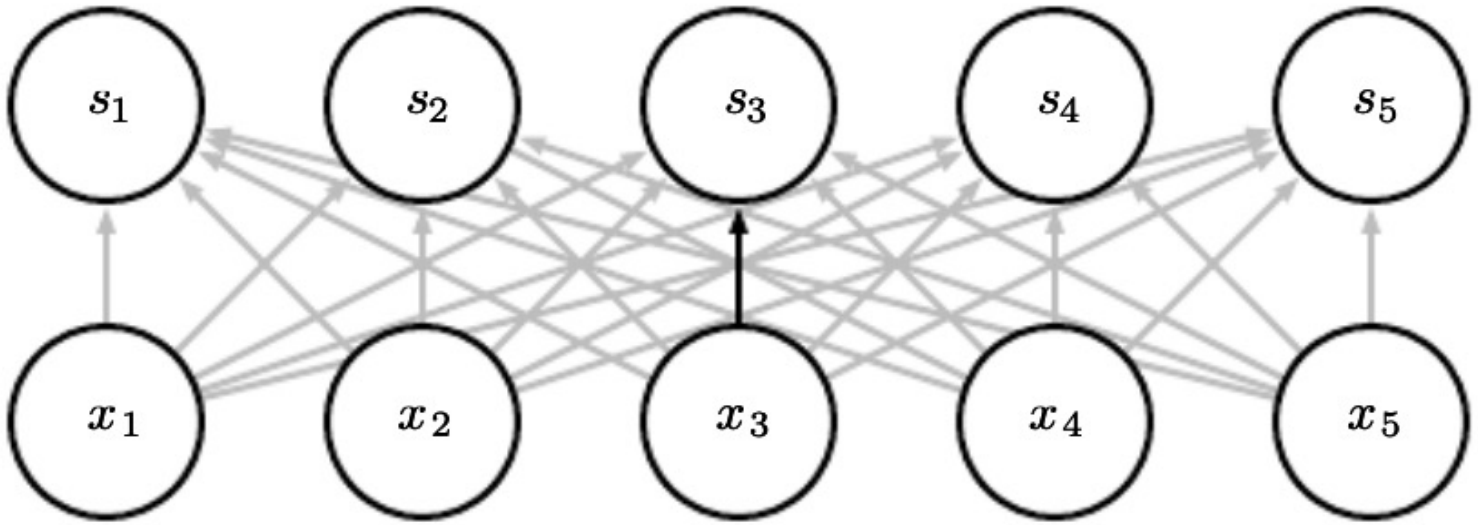
Convolutional Layer: Parameter Sharing

Connect each neuron to only a local region of the input

Instead of

No parameter sharing, so the parameter is used only once

Fully connected layer

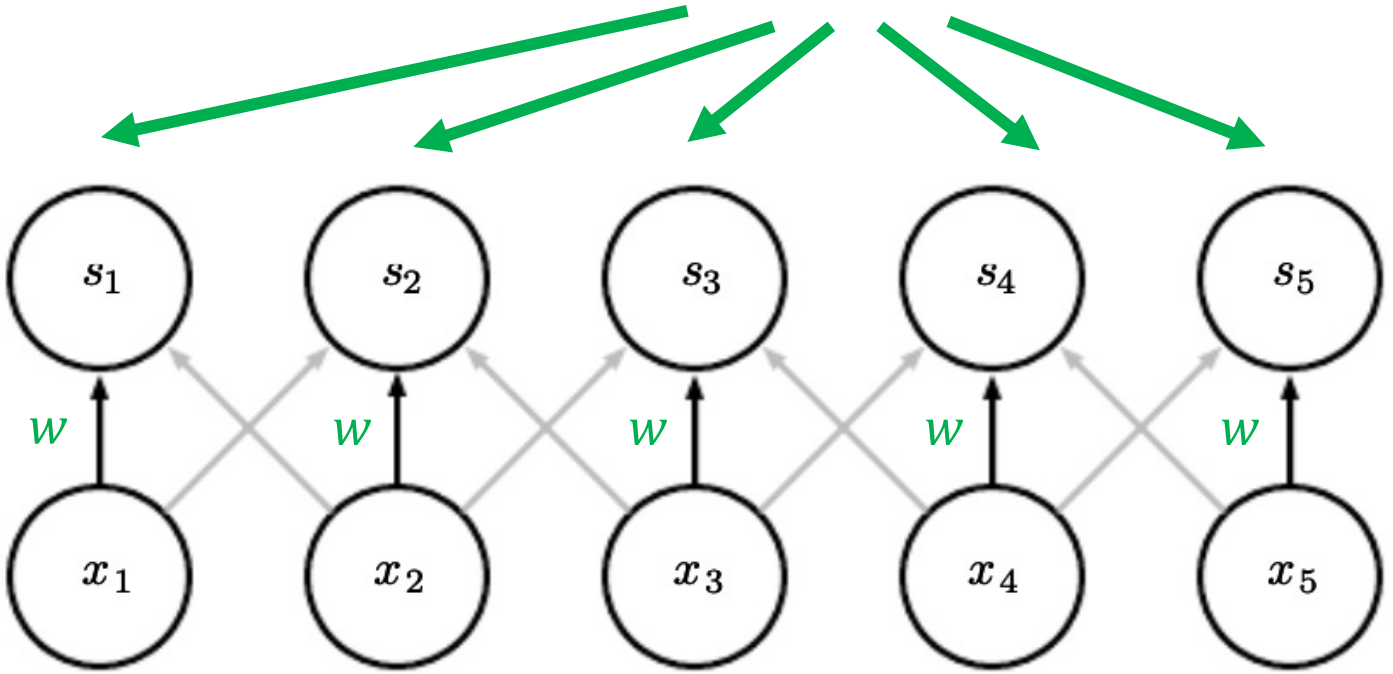


Convolutional Layer: Parameter Sharing

Connect each neuron to only a local region of the input

Parameter sharing

Parameter sharing with a 3-element kernel: single parameter is used at all input locations



Convolutional Layer

- Consists of a set of learnable filters or kernels
  - Kernel size defines receptive field
- Convolution: in the forward pass we slide each filter across the image
  - Produce response of a filter at every spatial position
- The network learns filters that activate when they see some type of visual feature (e.g., edge or orientiation)

Let's see next how does this work in practice!

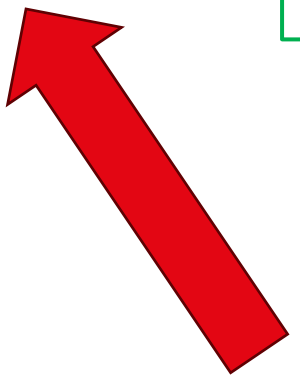
Convolution Operation

Image

157	153	174	168	150	152
155	182	163	74	75	62
180	180	50	14	34	6
206	109	5	124	131	111
194	68	137	251	237	239
172	105	207	233	233	214
188	88	179	209	209	215
189	97	165	84	84	168

Filter

0	-1	0
-1	4	-1
0	-1	0



Convolve

Slide the filer over the image and multiply filter values with pixel values

Filter (kernel) values are learnable!

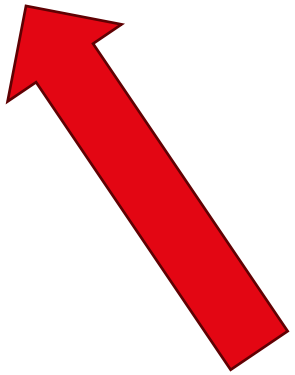
Convolution Operation

Image

157	153	174	168	150	152
155	182	163	74	75	62
180	180	50	14	34	6
206	109	5	124	131	111
194	68	137	251	237	239
172	105	207	233	233	214
188	88	179	209	209	215
189	97	165	84	84	168

Filter

0	-1	0
-1	4	-1
0	-1	0



77

$$0\times157 - 1\times153 + 0\times174$$
$$-1\times155 + 4\times182 - 1\times163$$
$$+0\times180 - 1\times180 + 0\times50$$

Filter (kernel) values are learnable!

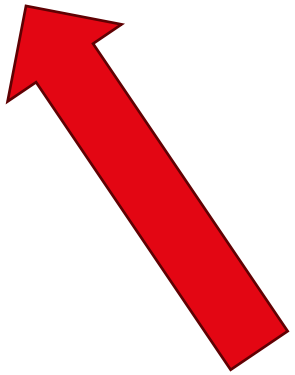
Convolution Operation

Image

157	153	174	168	150	152
155	182	163	74	75	62
180	180	50	14	34	6
206	109	5	124	131	111
194	68	137	251	237	239
172	105	207	233	233	214
188	88	179	209	209	215
189	97	165	84	84	168

Filter

0	-1	0
-1	4	-1
0	-1	0



77	172

$$0\times153 - 1\times174 + 0\times168$$
$$-1\times182 + 4\times163 - 1\times74$$
$$+0\times180 - 1\times50 + 0\times14$$

Filter (kernel) values are learnable!

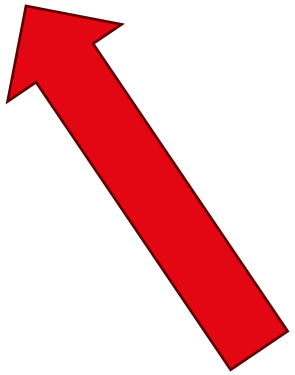
Convolution Operation

Image

157	153	174	168	150	152
155	182	163	74	75	62
180	180	50	14	34	6
206	109	5	124	131	111
194	68	137	251	237	239
172	105	207	233	233	214
188	88	179	209	209	215
189	97	165	84	84	168

Filter

0	-1	0
-1	4	-1
0	-1	0



77	172	-124

$$0\times174 - 1\times168 + 0\times150$$
$$-1\times163 + 4\times74 - 1\times75$$
$$+0\times50 - 1\times14 + 0\times34$$

Filter (kernel) values are learnable!

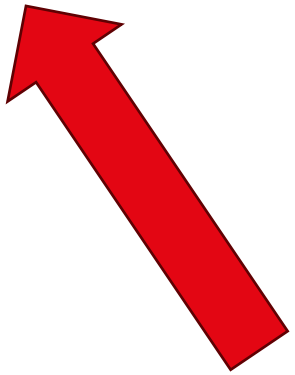
Convolution Operation

Image

157	153	174	168	150	152
155	182	163	74	75	62
180	180	50	14	34	6
206	109	5	124	131	111
194	68	137	251	237	239
172	105	207	233	233	214
188	88	179	209	209	215
189	97	165	84	84	168

Filter

0	-1	0
-1	4	-1
0	-1	0



77	172	-124	178

$$0\times168 - 1\times150 + 0\times152$$
$$-1\times74 + 4\times75 - 1\times62$$
$$+0\times14 - 1\times34 + 0\times6$$

Filter (kernel) values are learnable!

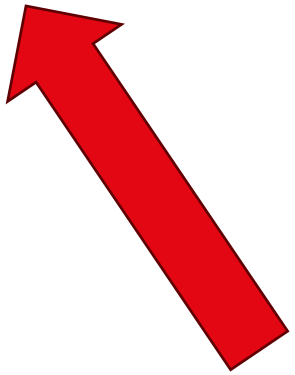
Convolution Operation

Image

157	153	174	168	150	152
155	182	163	74	75	62
180	180	50	14	34	6
206	109	5	124	131	111
194	68	137	251	237	239
172	105	207	233	233	214
188	88	179	209	209	215
189	97	165	84	84	168

Filter

0	-1	0
-1	4	-1
0	-1	0



77	172	-124	178
199			

$$0\times155 - 1\times182 + 0\times163$$
$$-1\times180 + 4\times180 - 1\times50$$
$$+0\times206 - 1\times109 + 0\times5$$

Filter (kernel) values are learnable!

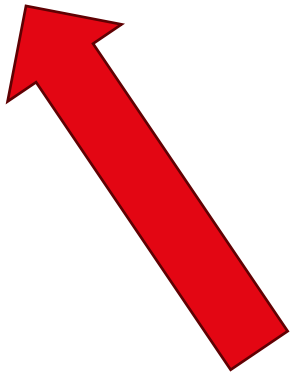
Convolution Operation

Image

157	153	174	168	150	152
155	182	163	74	75	62
180	180	50	14	34	6
206	109	5	124	131	111
194	68	137	251	237	239
172	105	207	233	233	214
188	88	179	209	209	215
189	97	165	84	84	168

Filter

0	-1	0
-1	4	-1
0	-1	0



77	172	-124	178
199	-162		

$$0\times182 - 1\times163 + 0\times74$$
$$-1\times180 + 4\times50 - 1\times14$$
$$+0\times109 - 1\times5 + 0\times124$$

Filter (kernel) values are learnable!

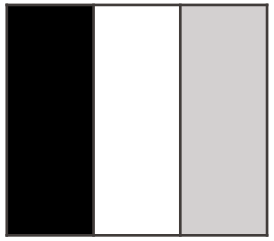
Different Filters Capture Different Local Features

Filter 1

-1	-1	-1
1	1	1
0	0	0

Filter 2

-1	1	0
-1	1	0
-1	1	0

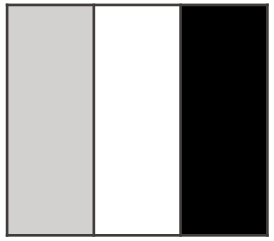


Filter 3

0	0	0
1	1	1
-1	-1	-1

Filter 4

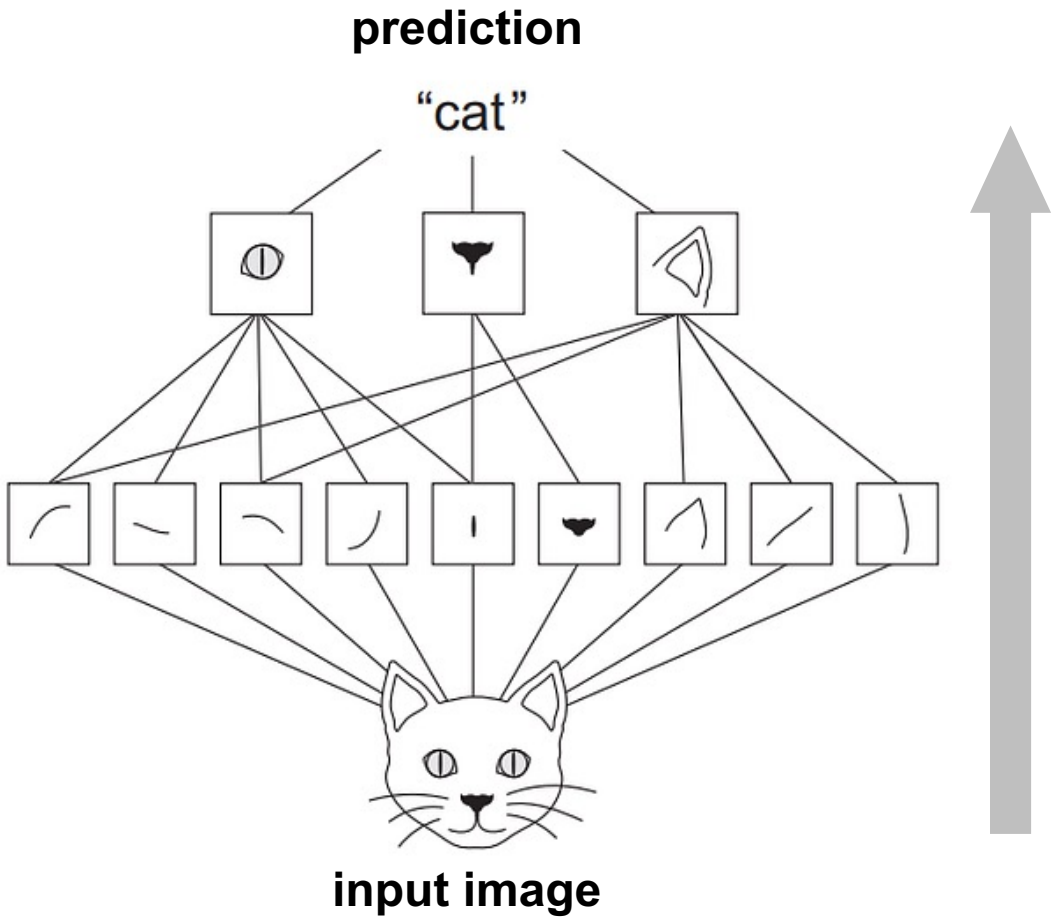
0	1	-1
0	1	-1
0	1	-1



Detecting different types of edges

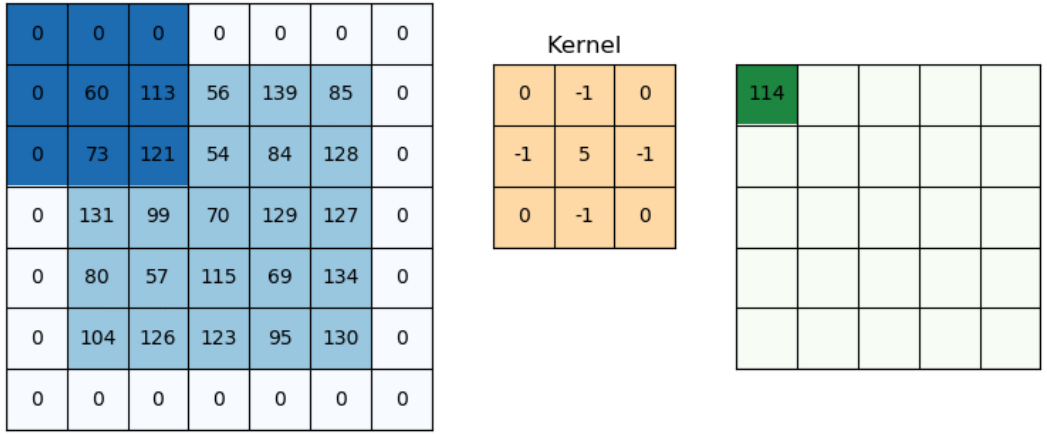
Hierarchical Learning

- CNNs learn feature hierarchies
  - Filters in the lower convolutional layers focus on detecting small local patterns such as edges
  - Features from higher levels of the hierarchy are formed by the composition of lower level features and detect more sophisticated patterns such as objects



Zero Padding

- Symmetrically adding zeros around the boundary of our input image to increase the effective size of the image
- Helps in reducing the loss of information at the borders of the input feature map
  - For any given convolution, we might only lose a few pixels, but this can add up as we apply many successive convolutional layers

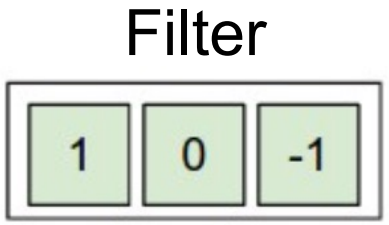


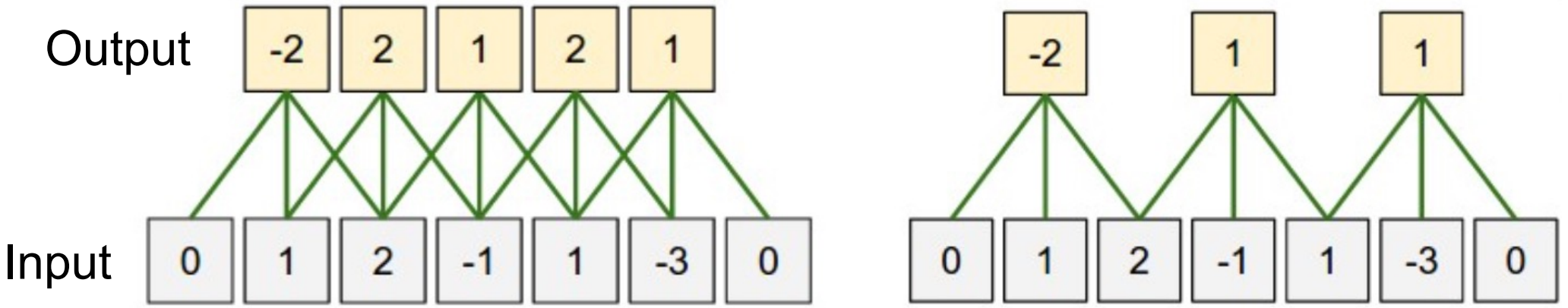
Concepts

- Hyperparameters that control the size of the output:
  - Depth (D): number of filters
  - Stride (S): how many pixels filters slide at a time
    - E.g., stride=2 means that the filters jumps 2 pixels at a time
    - Larger size will results in smaller output volumes
  - Zero-padding (P): how many zeros to pad around the border
  - Receptive field size (F): the spatial extent of this connectivity (filter size)

- Output size: $(W - F + 2P)/S + 1$, W is input size
  - Why? https://cs231n.github.io/convolutional-networks/

Concepts: Example





Receptive field $F = 3$

Input size $W = 5$

Zero-padding $P = 1$

Stride $S = 1$

$F = 3$

$W = 5$

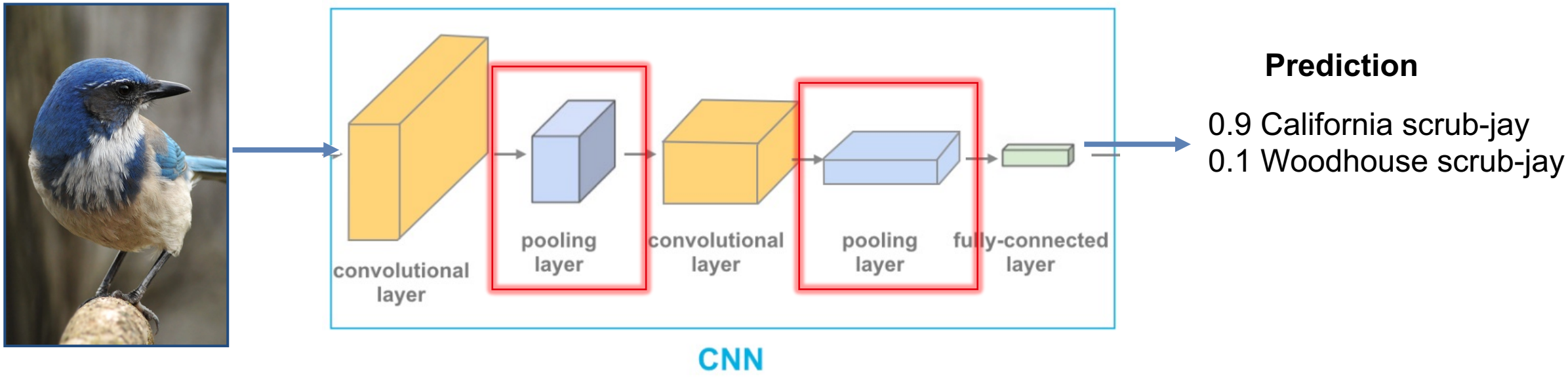
$P = 1$

$S = 2$

Figure from https://cs231n.github.io/convolutional-networks/

Two Main Building Blocks

1. Convolutional layer
2. **Pooling layer**

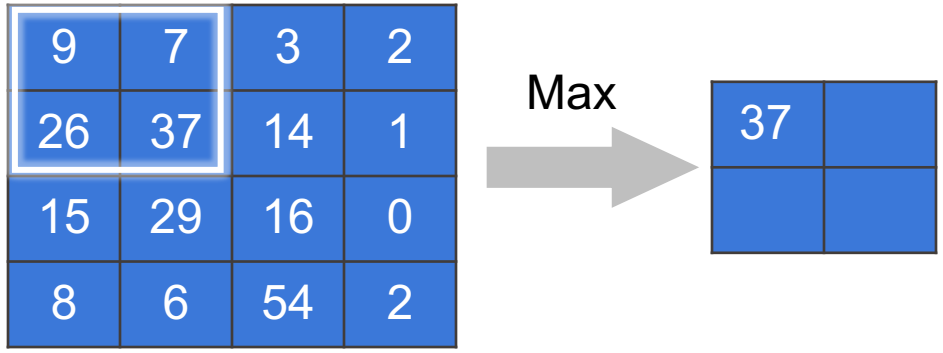


Pooling Layer

- Reduces spatial dimensions of the input
- Benefits:
  - The decrease in size leads to less computational overhead for the upcoming layers of the network
  - Prevents overfitting
- Typical pooling layers:
  - Max pooling: take the maximum value in the window
  - Average pooling: take the average value in the window
- Given activation map of size $W \times W \times D$ and a pooling filter of size F and stride S, the size of the output of the pooling layer is $W_{out} \times W_{out} \times D$ where $W_{out} = (W - F)/S + 1$
  - Pooling does not affect the depth volume D

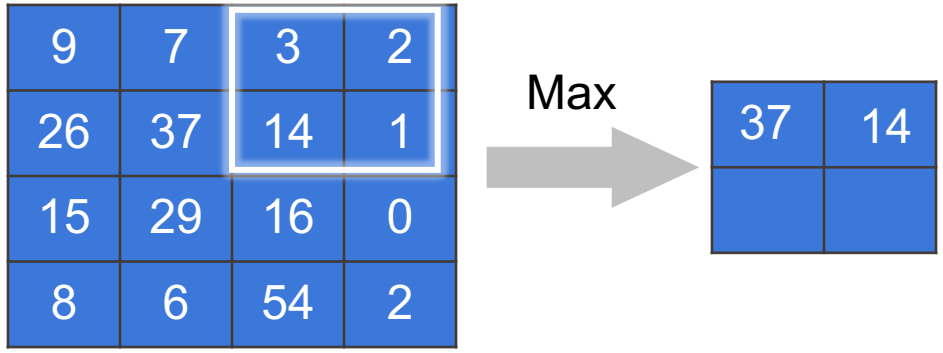
Max Pooling: Example

Filter size: $F = 2$

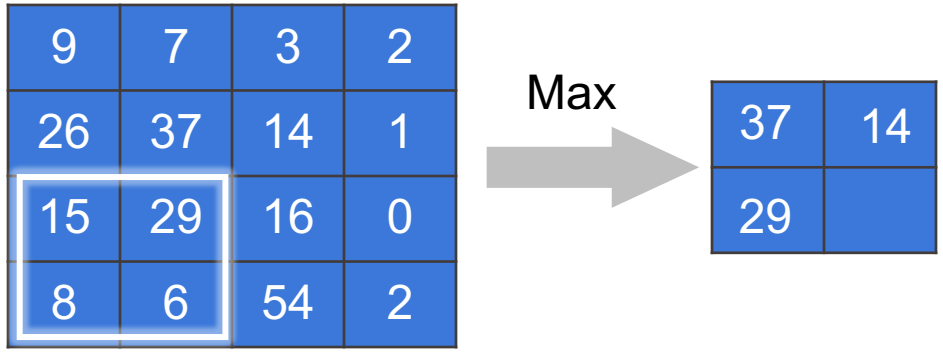


Max Pooling: Example

Stride S: $F = 2$

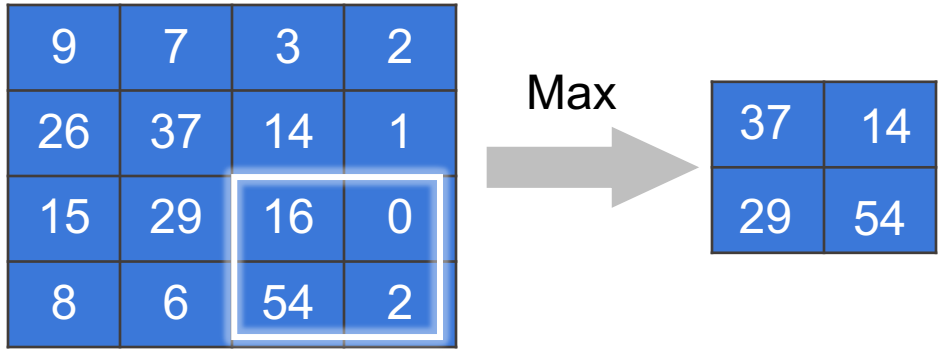


Max Pooling: Example



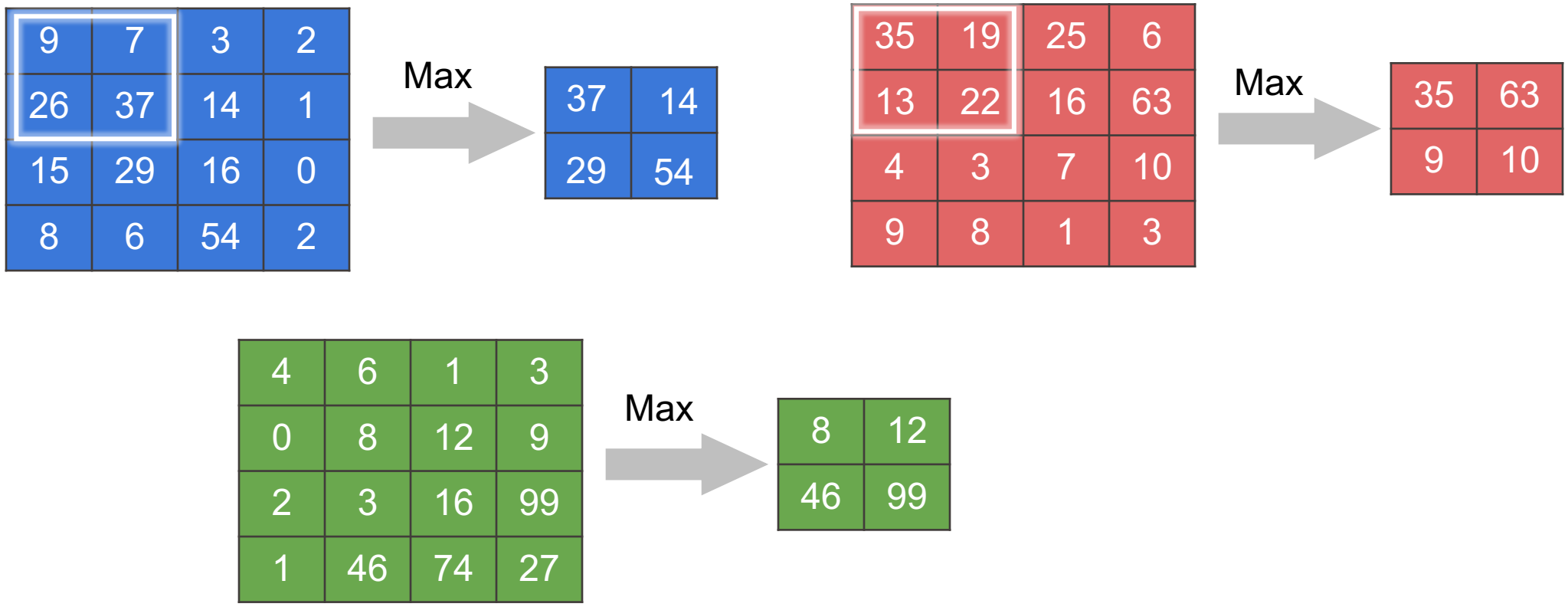
Max Pooling: Example

- Does not introduce any trainable parameters

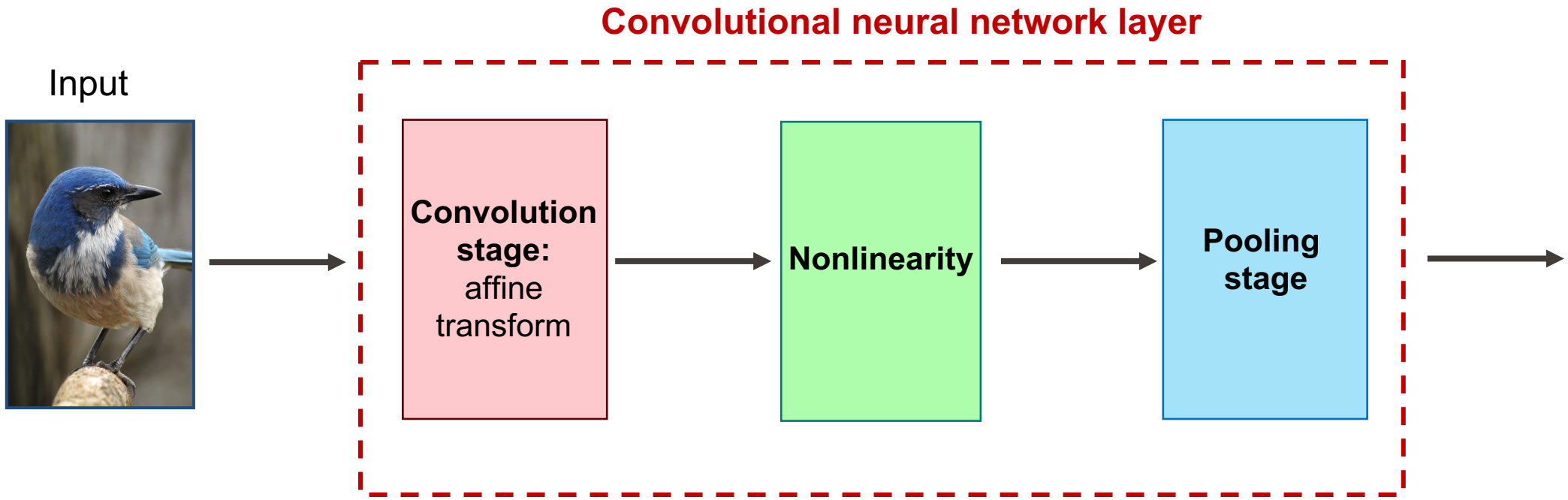


Max Pooling: Example

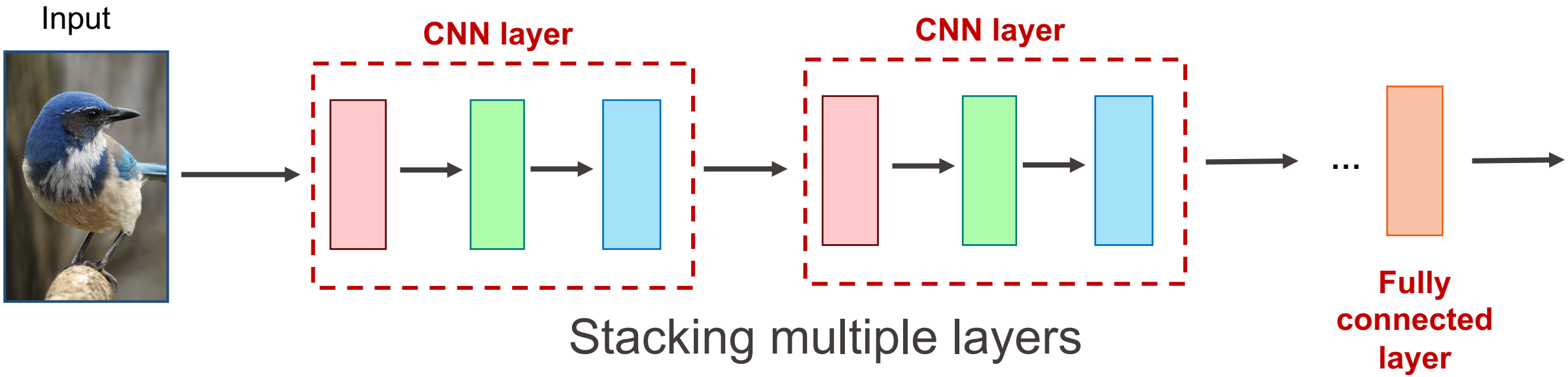
- Does not introduce any trainable parameters
- Applied to each channel separately



Convolutional Neural Network: Putting It All Together

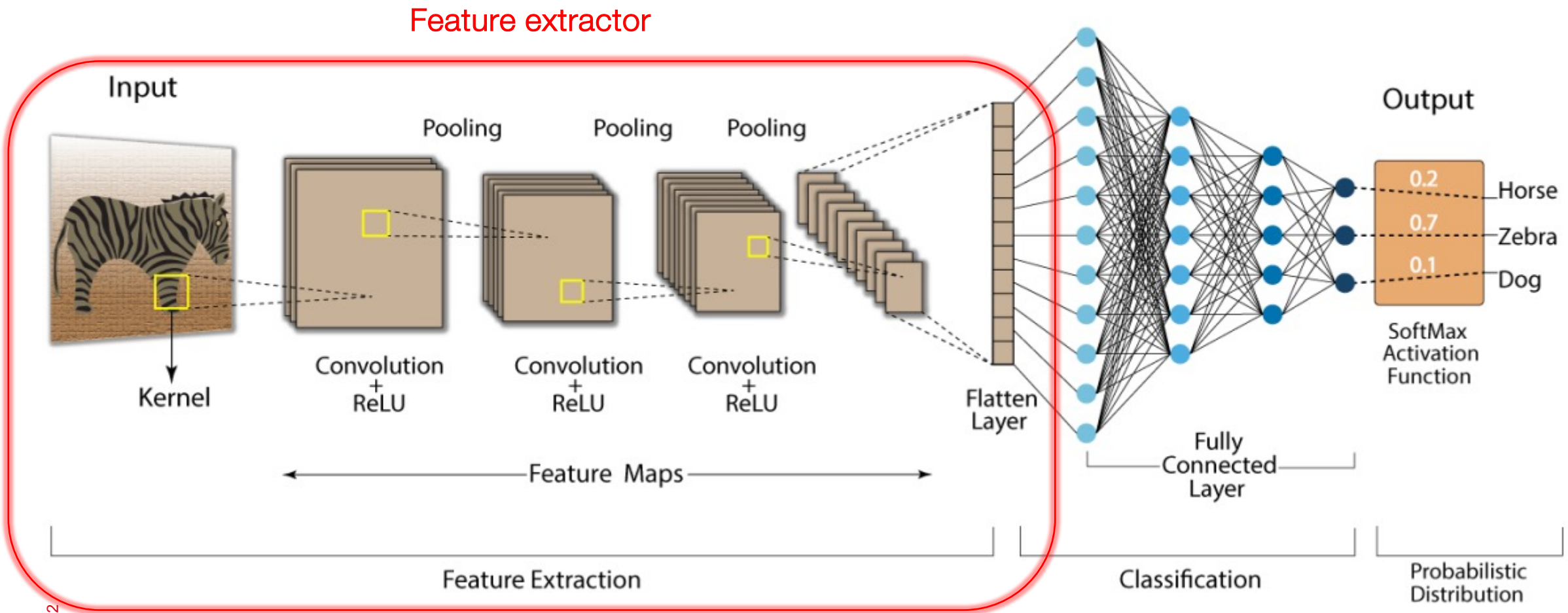


Convolutional Neural Network: Putting It All Together

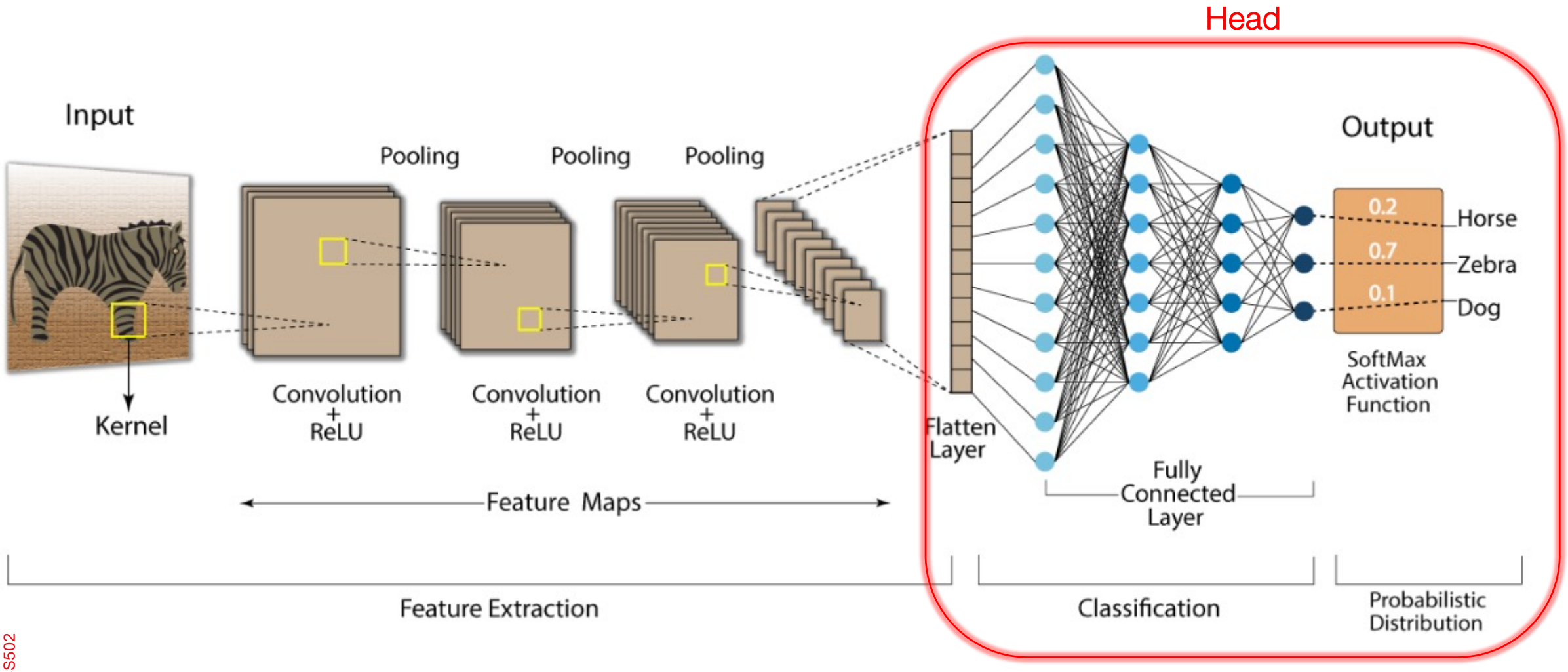


- Fully-connected layers are typically used in the last stages of the CNN to connect to the output layer and construct the desired number of outputs

CNN Architecture



CNN Architecture



Visualizing What CNNs Learn

- Visualization of CNN weights (filters)
  - These are usually most interpretable in the first convolutional layer
  - Well-trained networks usually display nice and smooth filters without any noisy patterns

1st layer filters

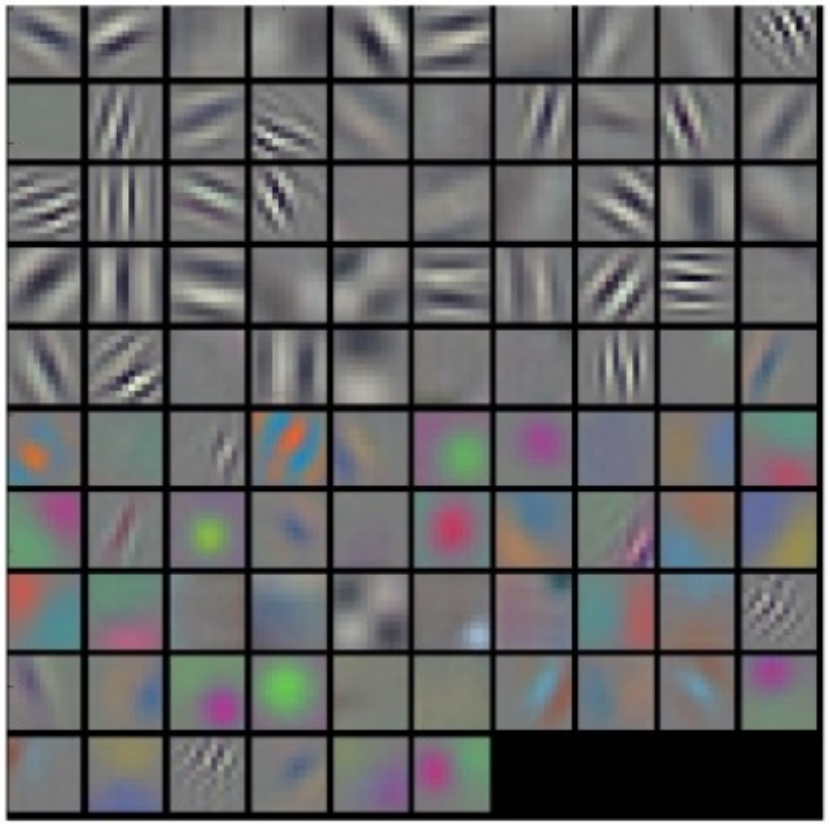


Figure from https://cs231n.github.io/understanding-cnn/

Visualizing What CNNs Learn

- Visualization of CNN weights (filters)
  - These are usually most interpretable in the first convolutional layer
  - Well-trained networks usually display nice and smooth filters without any noisy patterns
- Visualization of activations

1st layer activations for picture of a cat

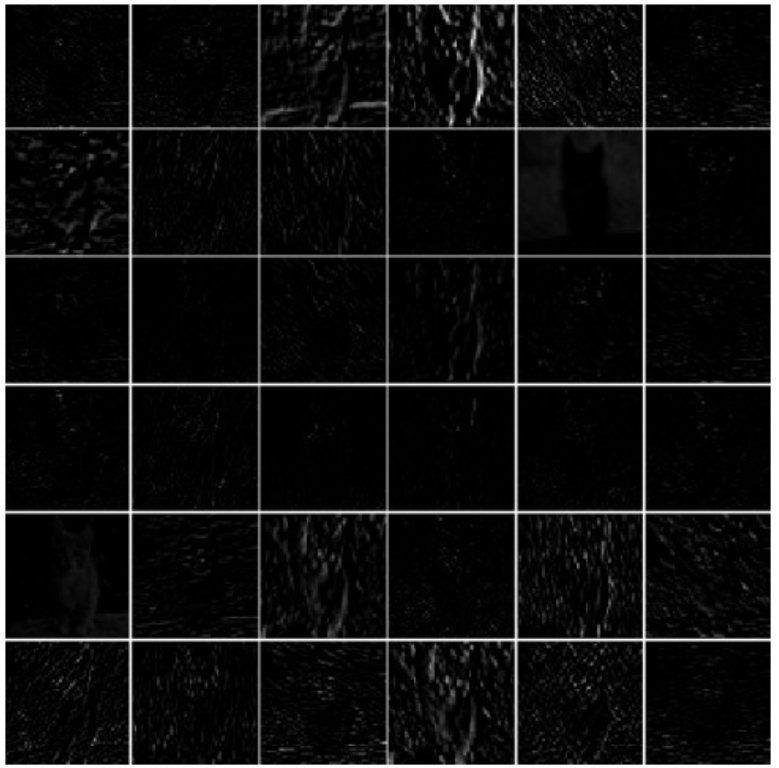


Figure from https://cs231n.github.io/understanding-cnn/

Visualizing What CNNs Learn

- Visualization of CNN weights (filters)
  - These are usually most interpretable in the first convolutional layer
  - Well-trained networks usually display nice and smooth filters without any noisy patterns
- Visualization of activations
- Also possible to retrieve images that maximally activate a neuron, occlude parts of the image etc
  - See https://cs231n.github.io/understanding-cnn/

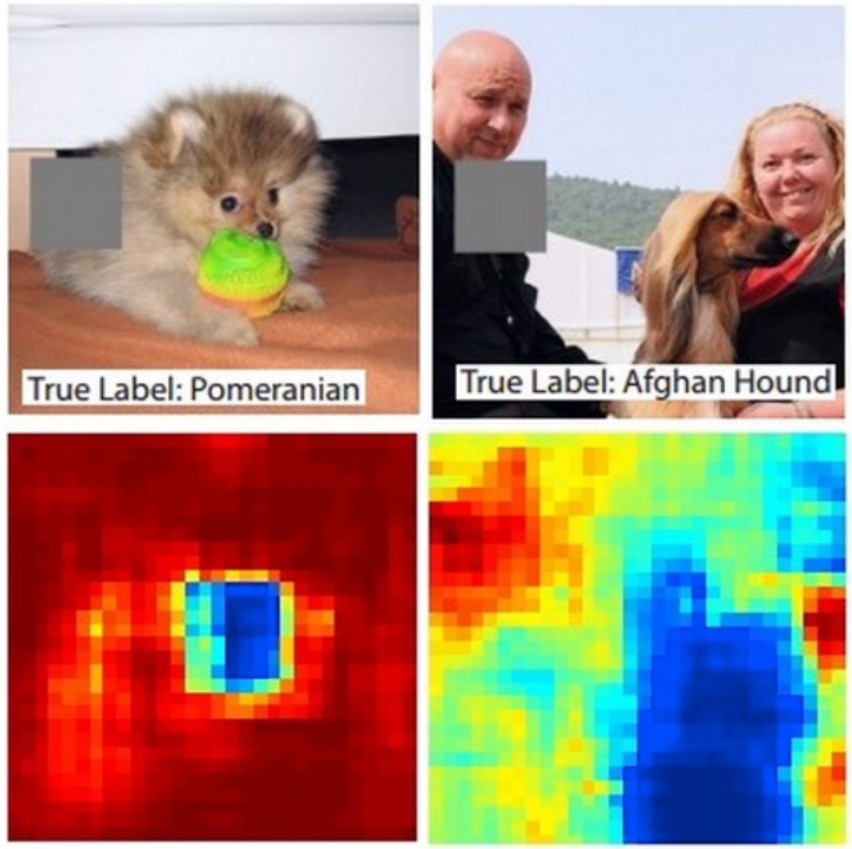
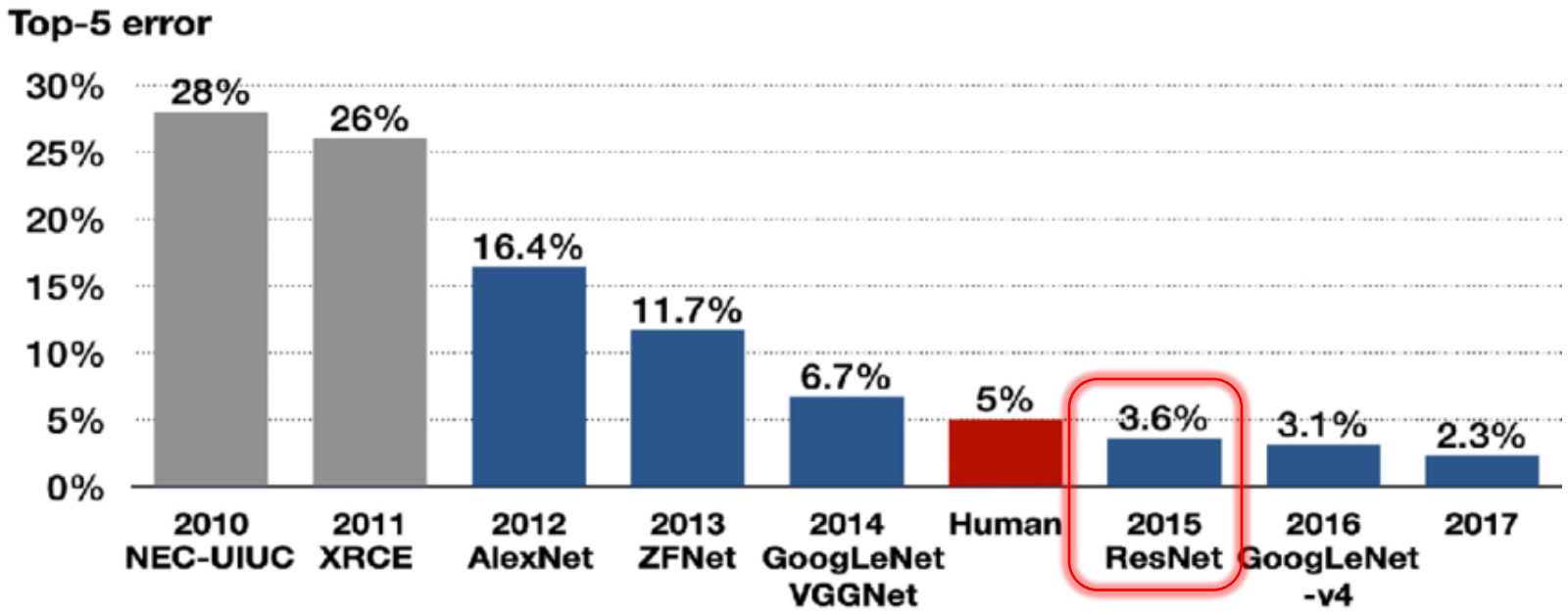


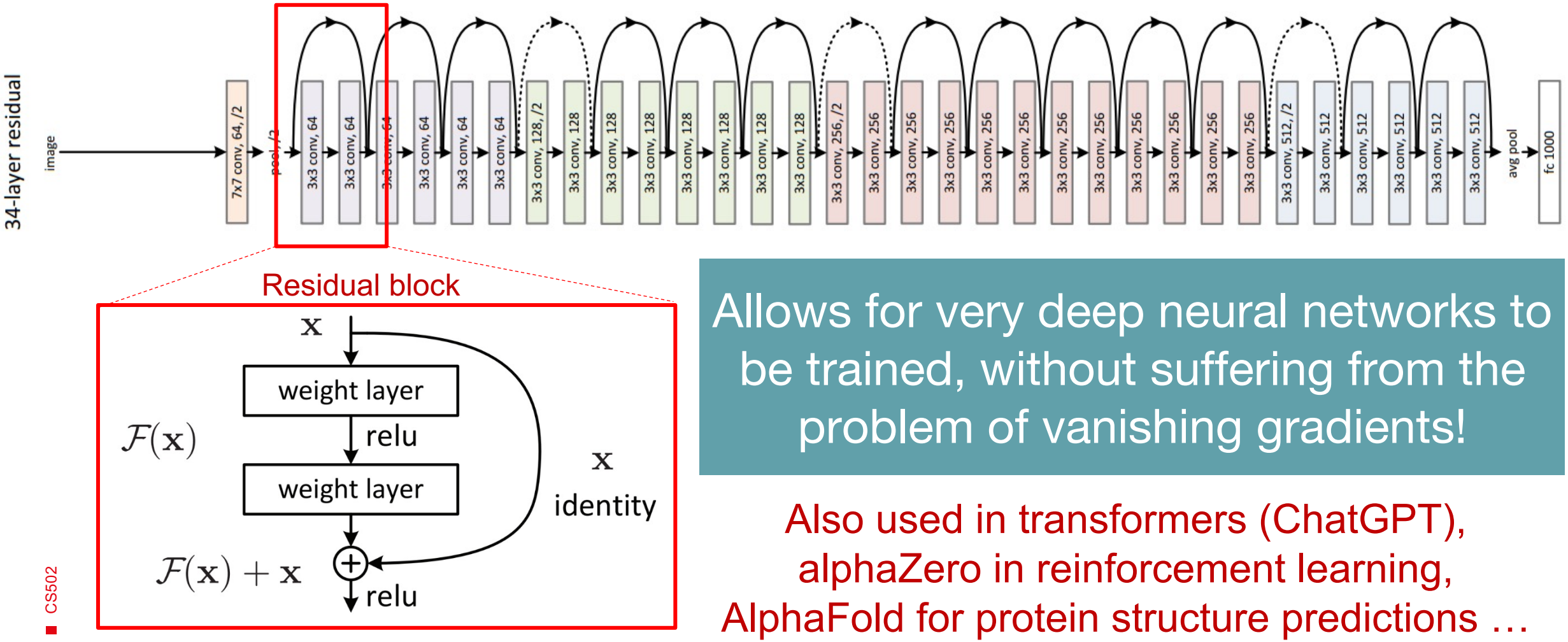
Figure from https://cs231n.github.io/understanding-cnn/

Famous Architectures



ResNet key idea: **skip connections**

ResNet Architecture



Allows for very deep neural networks to be trained, without suffering from the problem of vanishing gradients!

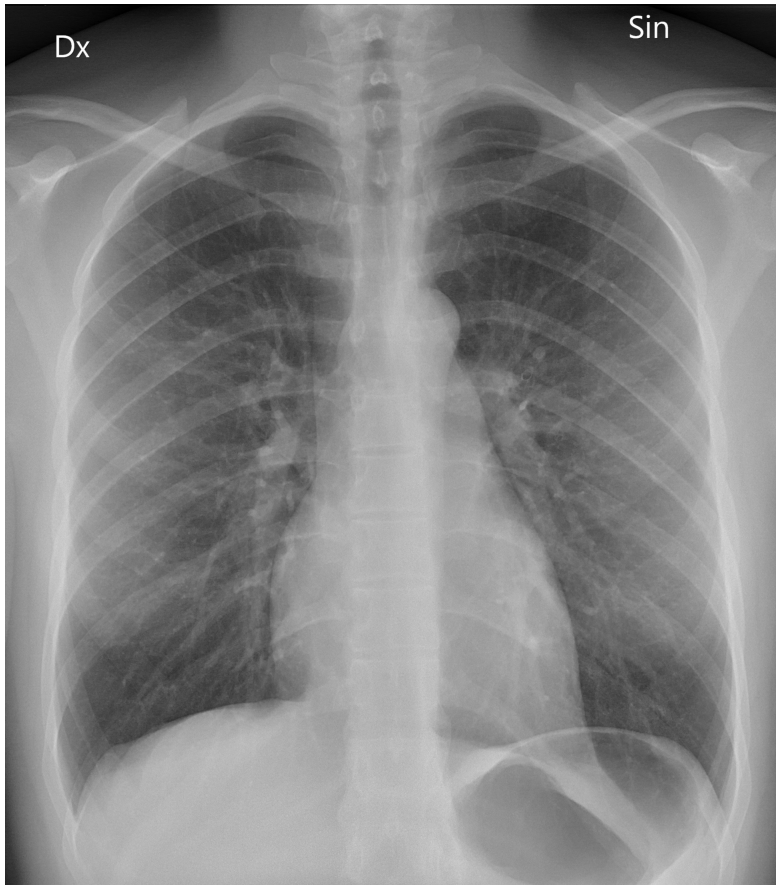
Also used in transformers (ChatGPT), alphaZero in reinforcement learning, AlphaFold for protein structure predictions …

Biomedical Applications: Medical Imaging

Image Classification

- Deep neural networks are really good in classifiying images!

Can We Use DNNs for Medical Imaging?



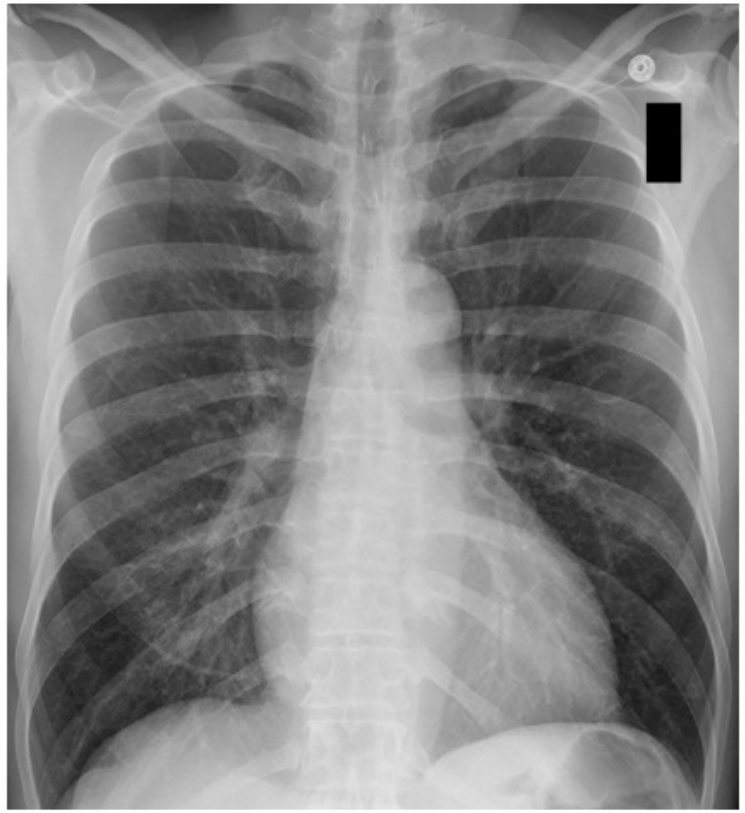
Why We Need Computational Methods for Medical Images?

- Diagnostic errors are frequent!
  - Estimation that more than 250,000 patients per year die in the US because of medical error[1]
  - Most frequent type of errors: radiologist does not “see” the abnormality on the imaging exam
  - Others: the radiologist sees an abnormality but fails to render a correct diagnoses
- Many people in less developed countries lack access to radiology specialists

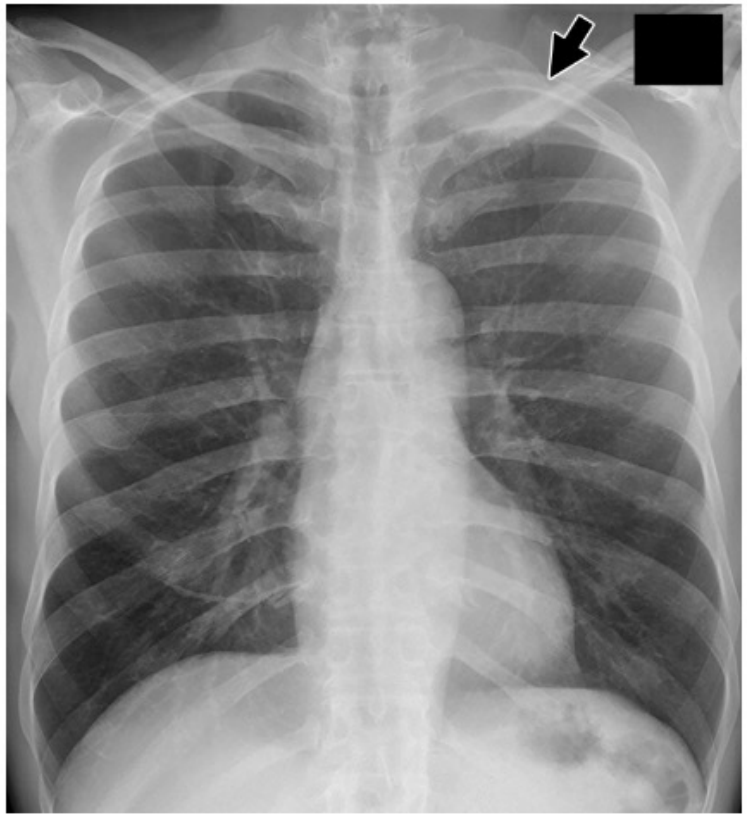
[1] Waite et al. Interpretive Error in Radiology

Diagnostic Errors: Examples

Correctly interpreted as normal



Wrongly interpreted as normal



Search-related omission error. Patient presented 8 months later with complaints of left arm pain. Chest radiograph was interpreted as normal; however, left apical mass (*arrow*) was missed. Apices are common blind spot in interpretation, and radiologist reported not looking in this region.

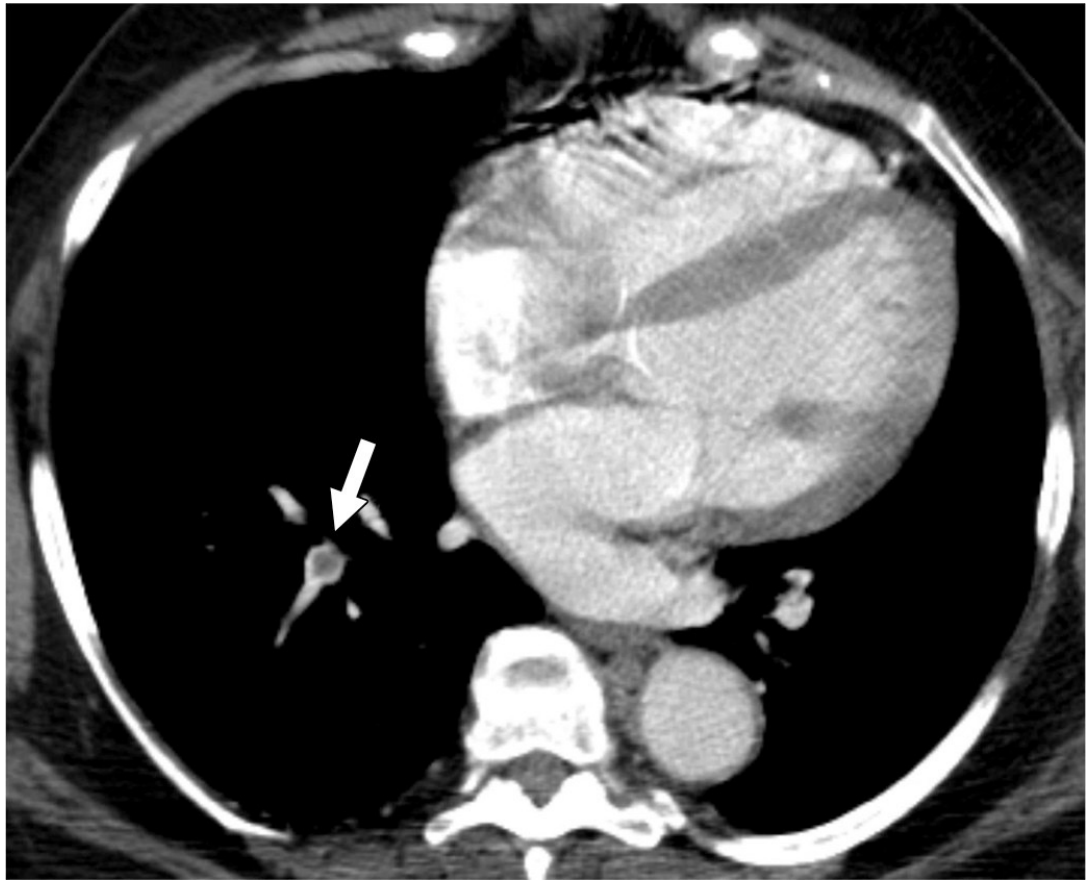
Waite et al. Interpretive Error in Radiology

Diagnostic Errors: Examples

Distraction error. Radiograph of wrist and hand of 32-year-old man show subluxation of fifth carpometacarpal articulation (oval) missed on initial interpretation. Radiologist recalled being on telephone with clinician while interpreting this study.

Waite et al. Interpretive Error in Radiology

Diagnostic Errors: Examples



Search error. Abdominal CT scan of 53-year-old man who presented with abdominal pain shows pulmonary embolus (*arrow*) in right lower lobe overlooked on interpretation. Abnormalities of lung bases are commonly missed on abdominal CT because they are not primary target of examination.

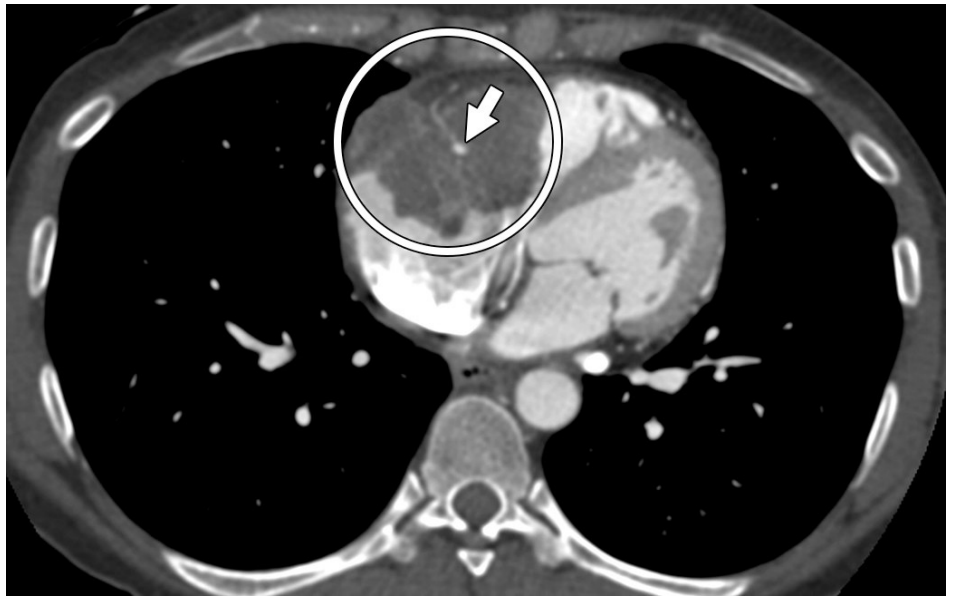
Waite et al. Interpretive Error in Radiology

Why is medical image classification hard?

Medical Images: Why Is It Hard?

- Look from our own perspective:
  - How hard for us is to classify radiologist images vs general images?

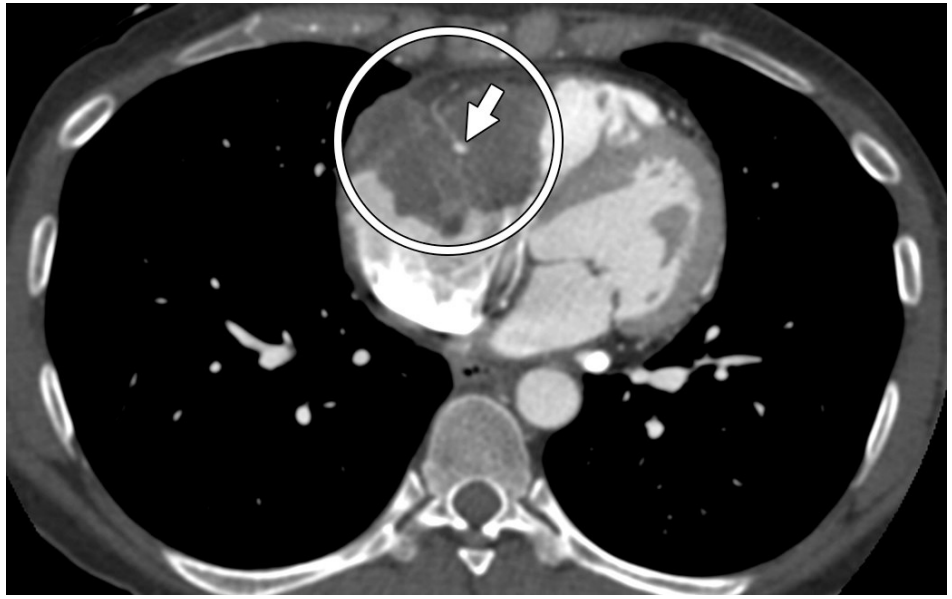
We need specialists to do this



Children can do it

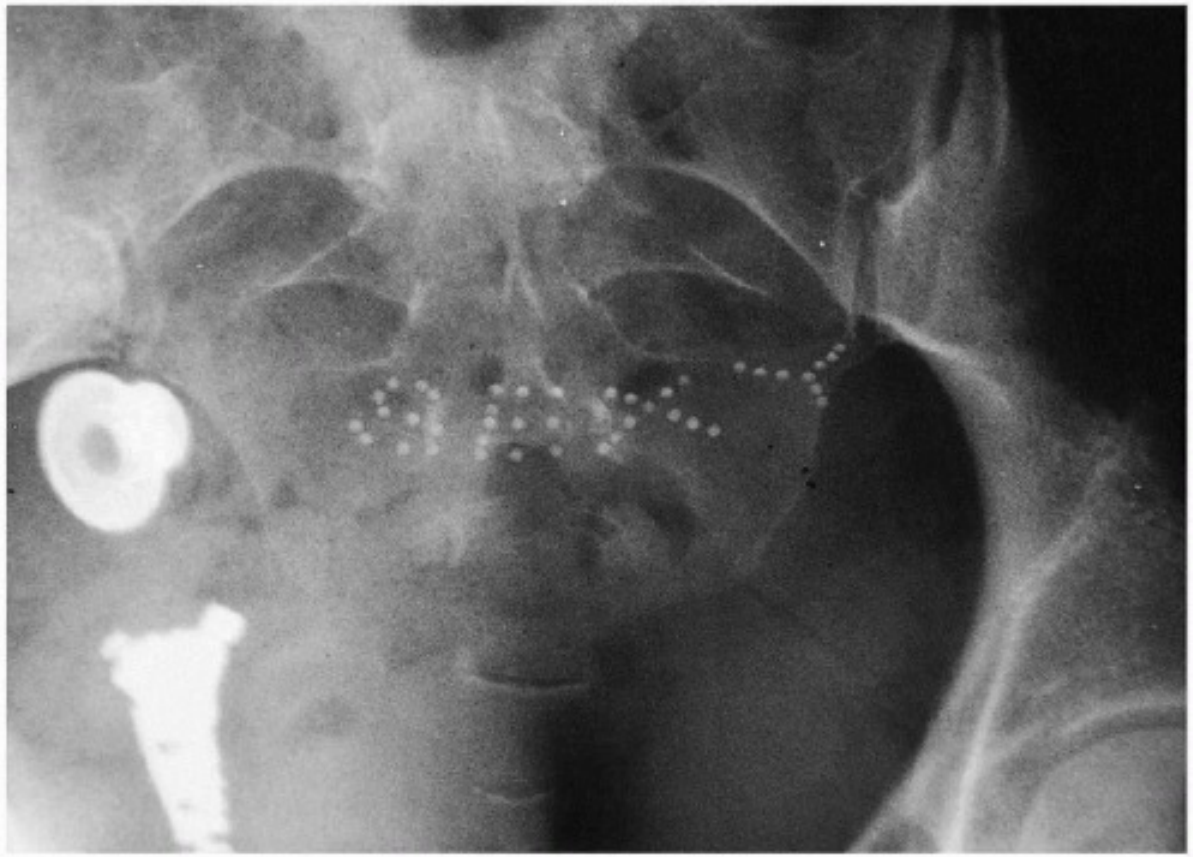
Medical Images: Why Is It Hard?

- Medical images are large and complex!
  - Only a small part of the image is relevant for classification
  - Standard computer vision datasets: classes with high variation



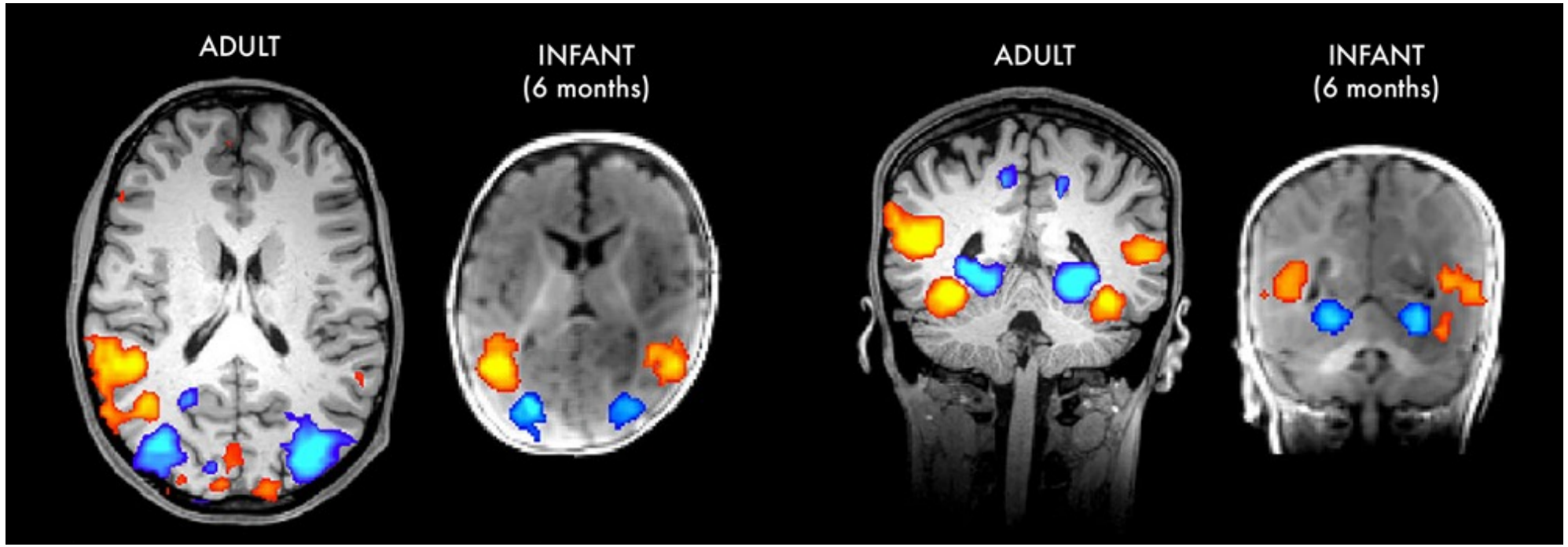
Medical Images: Why Is It Hard?

- Image artifacts can cause wrong classification



Medical Images: Why Is It Hard?

- Context is important
  - E.g., age or gender of a patient



Medical Images: Why Is It Hard?

- Available data for training is small!
  - Hard to label at scale
  - Requires experts and time effort
- Dataset bias
  - A cohort may not appropriately represent patients and symptoms
  - Measurement bias
  - Labeling errors
- Limited benefits of transfer learning from non-image datasets
  - The first training task is typically not medically relevant

Different Techniques

- Knee images generated with different techniques

X-ray

Magnetic resonance imaging (MRI)

CT scan

Ultrasound

Biomedical Applications: Pneumonia Detection

Rajpurkar et al. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. *arXiv* 2017

Radiology: Pneumonia Detection

- More than 1 million adults are hospitalized with pneumonia every year in the US
  - Chest X-rays are typical method for pneumonia diagnosis
- Chest X-Ray14 dataset[1,2]
  - > 100,000 frontal images from >30,800 patients released as public domain
  - Images are classified to 14 different thoracic diseases, including pneumonia
- CheX-Net[2]
  - Convolutional neural network for pneumonia detection

14 thoracic diseases

Atelectasis
Cardiomegaly
Effusion
Infiltration
Mass
Nodule
Pneumonia
Pneumothorax
Consolidation
Edema
Emphysema
Fibrosis
Pleural Thickening
Hernia

[1] Wang et al. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks
[2] Rajpurkar et al. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning

CheX-Net Approach

- Pretrain the network on large-scale dataset of general images
  - ImageNet: 14,197,122 images annotated to 1000 classes

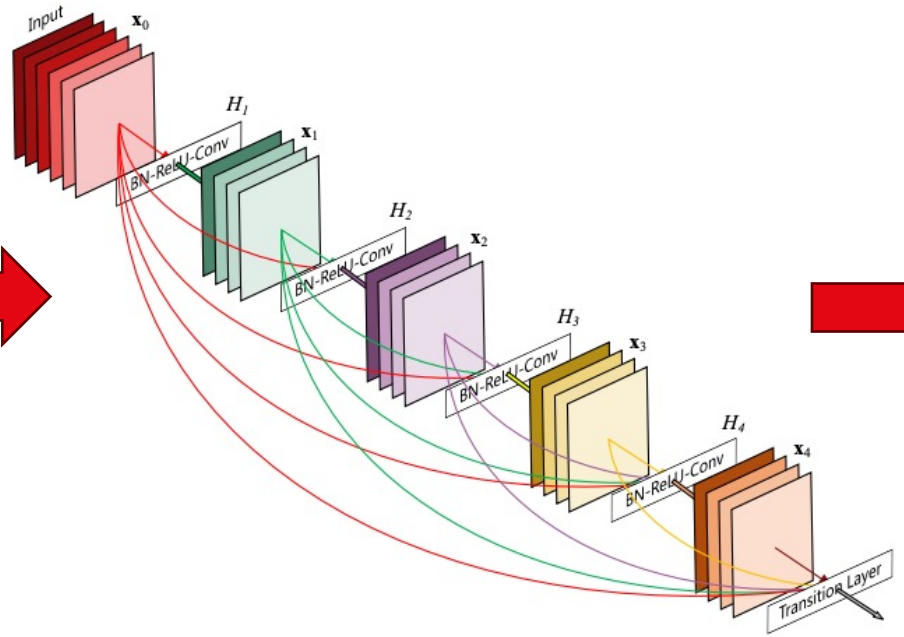
- Fine-tune on Chest X-Ray14 dataset

CheX-Net: Pneumonia Detection

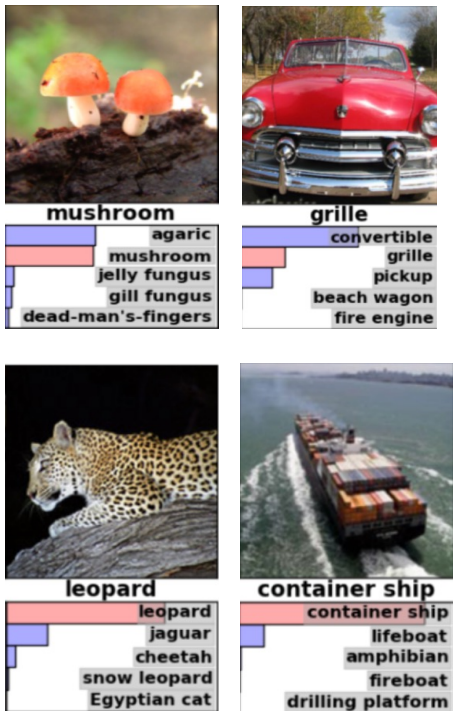
1) Pretraining

ImageNet

DenseNet
121 convolutional layers



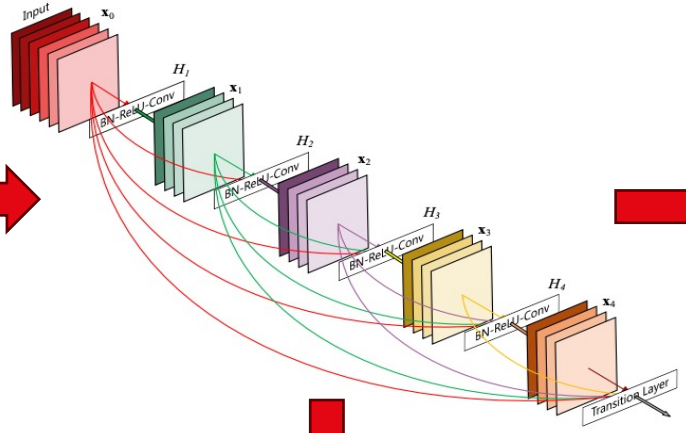
1000 classes

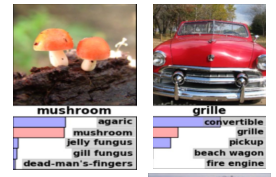


In reality: Take the ImageNet pretrained model!

CheX-Net: Pneumonia Detection

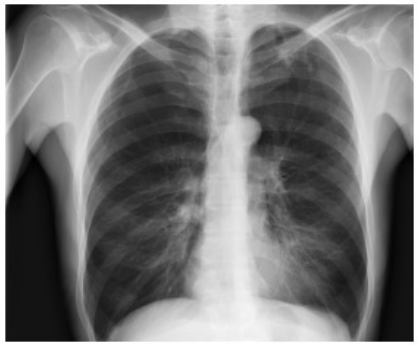
2) Fine-tuning

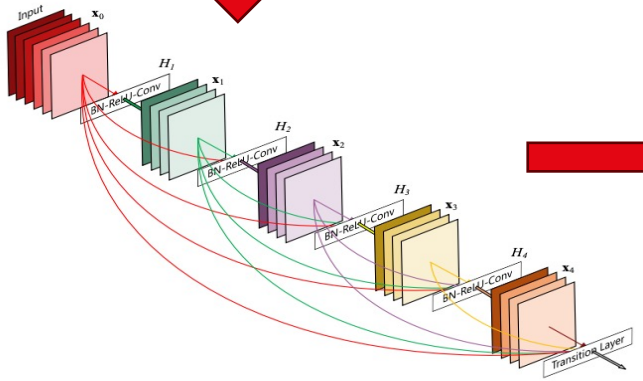




Transfer learning!

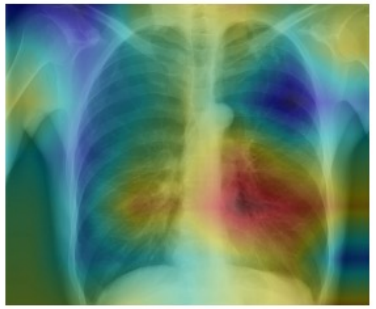
Transfer weights





Classification layer

Pneumonia Positive (85%)

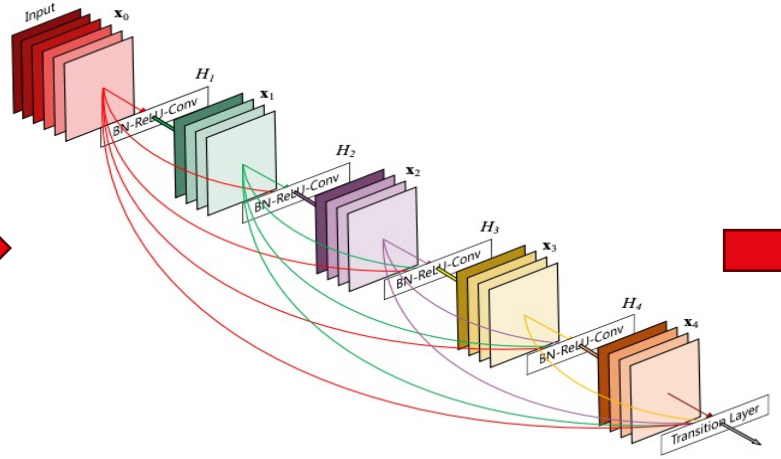


CheX-Net: Pneumonia Detection

2) Fine-tuning

Chest X-ray image

DenseNet
121 convolutional layers



Tasks

Pneumonia detection

Pneumonia positive
Pneumonia negative

- Weighted binary cross entropy loss:

$$\mathcal{L}(X, y) = -w_+ \, y \log p(Y = 1|X) - w_-(1 - y) \log p(Y = 0|X)$$

positive class weight

predicted probability that the image is pneumonia positive

negative class weight

predicted probability that the image is pneumonia negative

$$w_+ = \frac{|N|}{|P| + |N|}$$

Proportion of pneumonia negative cases

$$w_- = \frac{|P|}{|P| + |N|}$$

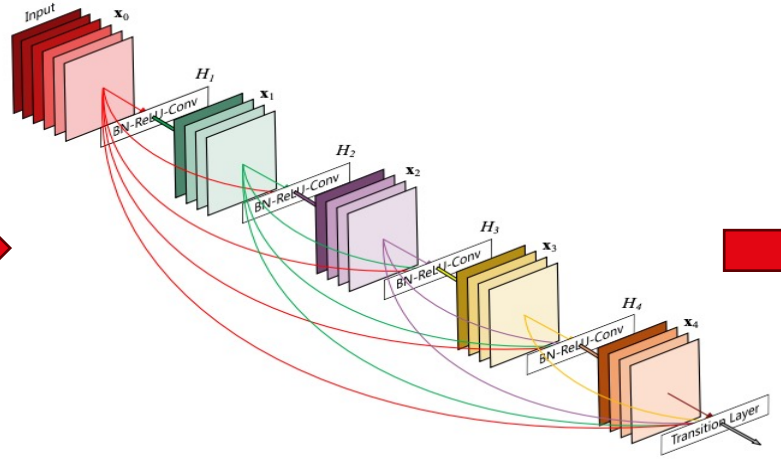
Proportion of pneumonia positive cases

CheX-Net: Pneumonia Detection

2) Fine-tuning

Chest X-ray image

DenseNet
121 convolutional layers



Tasks

Pneumonia detection

Pneumonia positive
Pneumonia negative

Pathology classification

Atelectasis
Cardiomegaly
Effusion
Infiltration
…

Replace the last layer with a layer with 14-dimensional output

- Sum of binary cross entropy losses:

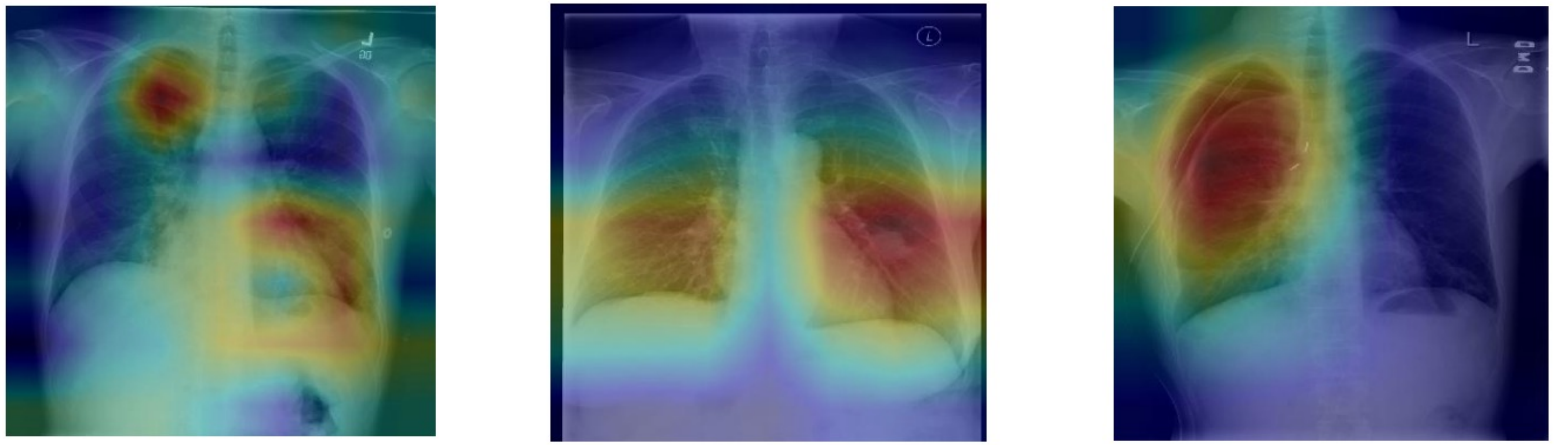
$$\mathcal{L}(X, y) = \sum_{c=1}^{14} [-y_c \log p(Y_c = 1|X) - (1 - y_c) \log p(Y_c = 0|X)]$$

predicted probability that the image does contain pathology c

predicted probability that the image does not contain pathology c

Model Interpretation

- Produce heatmaps to visualize the areas of the image most indicative of the disease using class activation mappings[3](CAM)
- To generate the CAMs, feed an image into the fully trained network and extract the feature maps that are output by the final convolutional layer
  - Obtain a map of the most salient features used in classifying the image as having pathology c



[3] Zhou et al. Learning deep features for discriminative localization. *CVPR* 2015

CheX-Net: Results

Pneumonia detection

	F1 Score (95% CI)
Radiologist 1	0.383 (0.309, 0.453)
Radiologist 2	0.356 (0.282, 0.428)
Radiologist 3	0.365 (0.291, 0.435)
Radiologist 4	0.442 (0.390, 0.492)
Radiologist Avg.	0.387 (0.330, 0.442)
CheXNet	0.435 (0.387, 0.481)

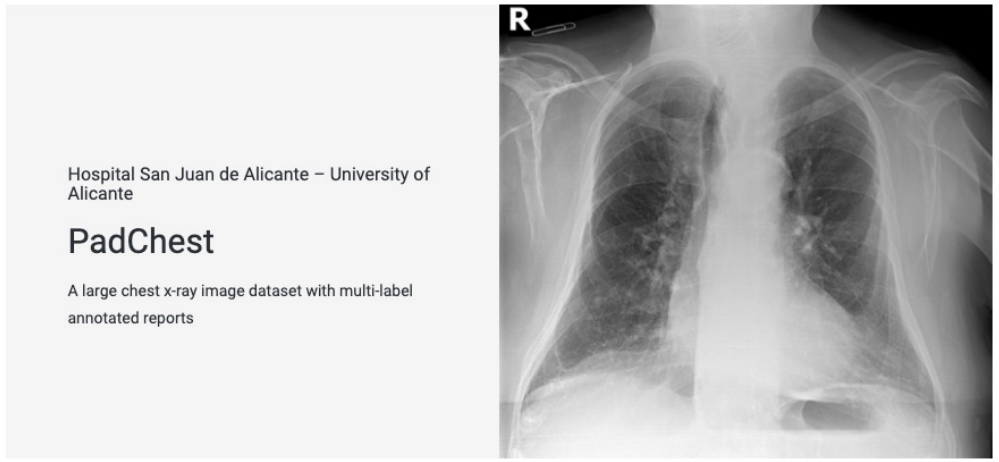
Pathology classification (AUROC)

Pathology	Wang et al. (2017)	Yao et al. (2017)	CheXNet (ours)
Atelectasis	0.716	0.772	**0.8094**
Cardiomegaly	0.807	0.904	**0.9248**
Effusion	0.784	0.859	**0.8638**
Infiltration	0.609	0.695	**0.7345**
Mass	0.706	0.792	**0.8676**
Nodule	0.671	0.717	**0.7802**
Pneumonia	0.633	0.713	**0.7680**
Pneumothorax	0.806	0.841	**0.8887**
Consolidation	0.708	0.788	**0.7901**
Edema	0.835	0.882	**0.8878**
Emphysema	0.815	0.829	**0.9371**
Fibrosis	0.769	0.767	**0.8047**
Pleural Thickening	0.708	0.765	**0.8062**
Hernia	0.767	0.914	**0.9164**

- **Limitations:**
  - Only frontal radiographs were presented to the radiologists and model
  - Neither the model nor the radiologists were not permitted to use patient history

Other Large-Scale X-Ray Image Datasets

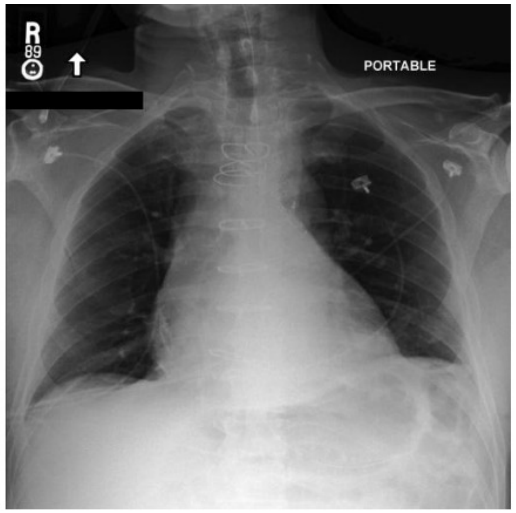
- Most of them use automatic labeler to assign labels to each image given a text-based radiology report, while others are manually labeled by hand
- All published in 2019



PadChest: >160,000 images from 67,000 patients

CheXpert: >224,000 images from 65,240 patients from Stanford

MIMIC-CXR



MIMIC-CXR: >371,000 images from 65,000 patients from Beth Israel Deaconess Medical Center in Boston (MIT)

Data loaders for all of these datasets available at torchxrayvision library!

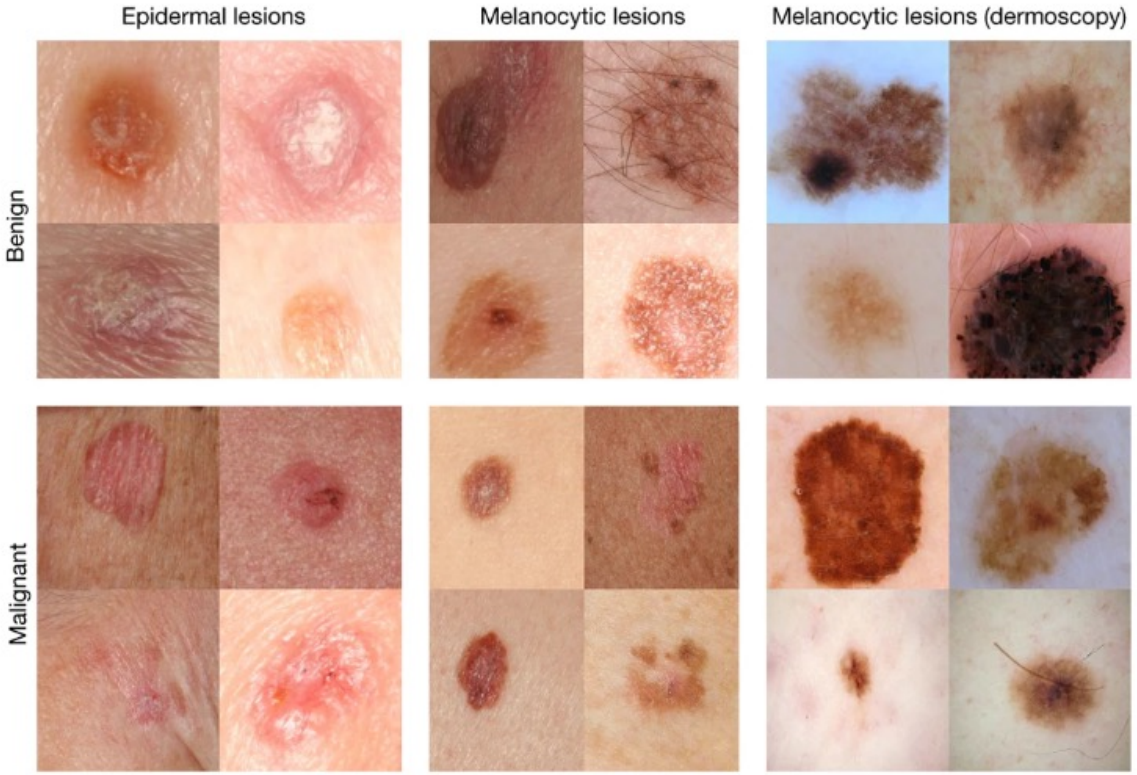
Biomedical Applications: Skin lesions classification

Esteva et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 2017


nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
Melanoma
LESIONS LEARNT
Artificial intelligence powers detection of skin cancer from images PAGES 36 & 115
NATURE.COM/NATURE
2 February 2017 £10
Vol. 542, No. 7639

Skin Lesion Detection

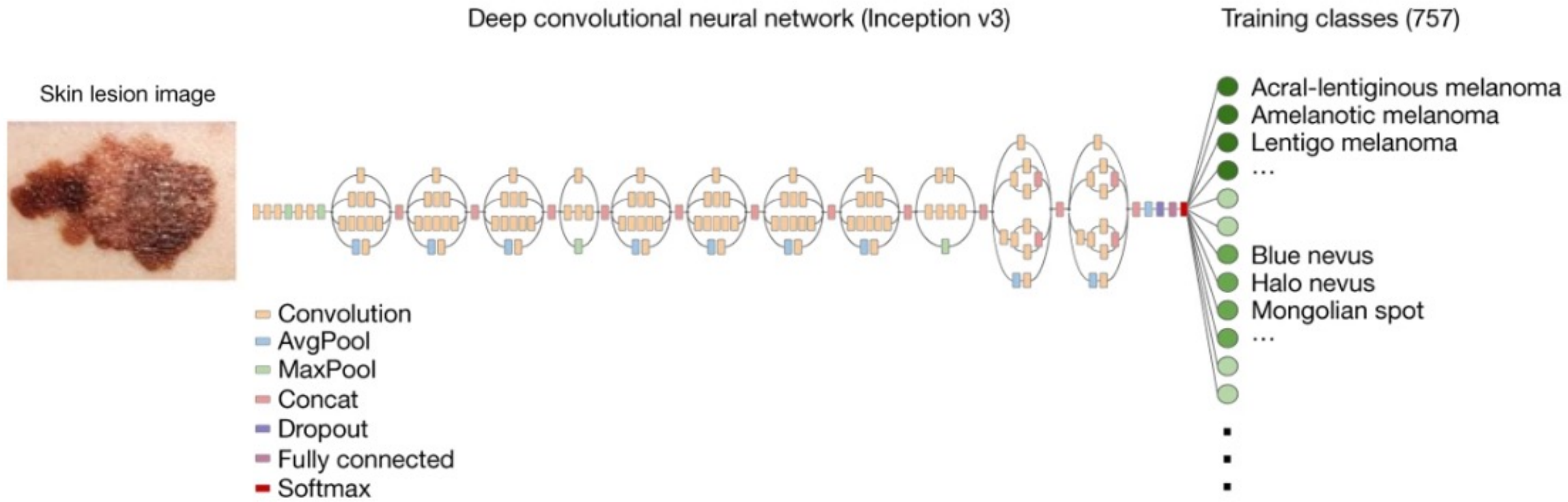
- 5.4 million new cases of skin cancer in the United States every year
- Early detection is critical!
  - 5-year survival rate for melanoma drops from over 99% if detected in its earliest stages to about 14% if detected in its latest stages
- Dataset:
  - 129,450 clinical images (not radiology dataset but standard images!)



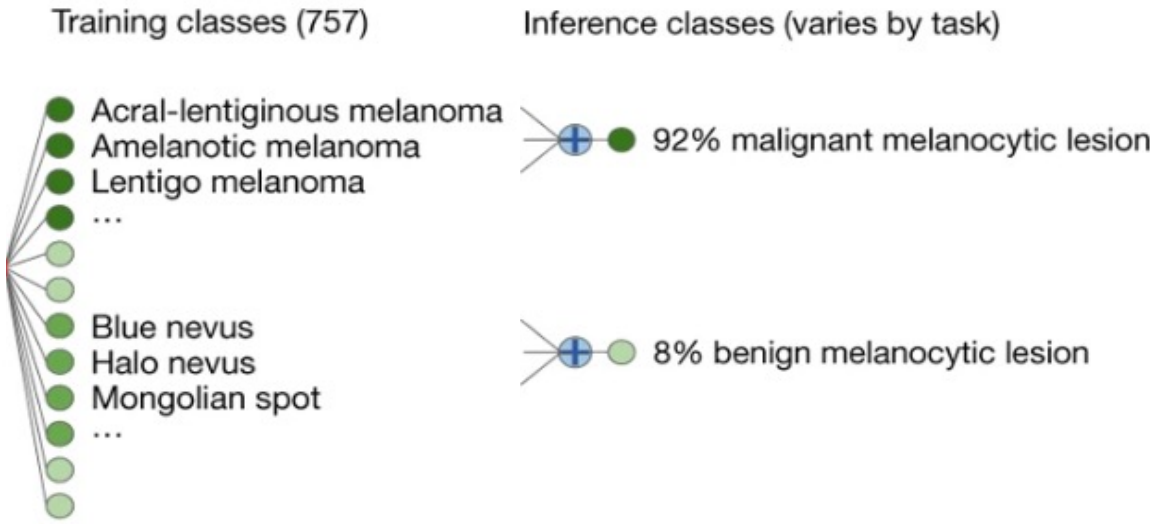
CNNs for Skin Lesion Classification

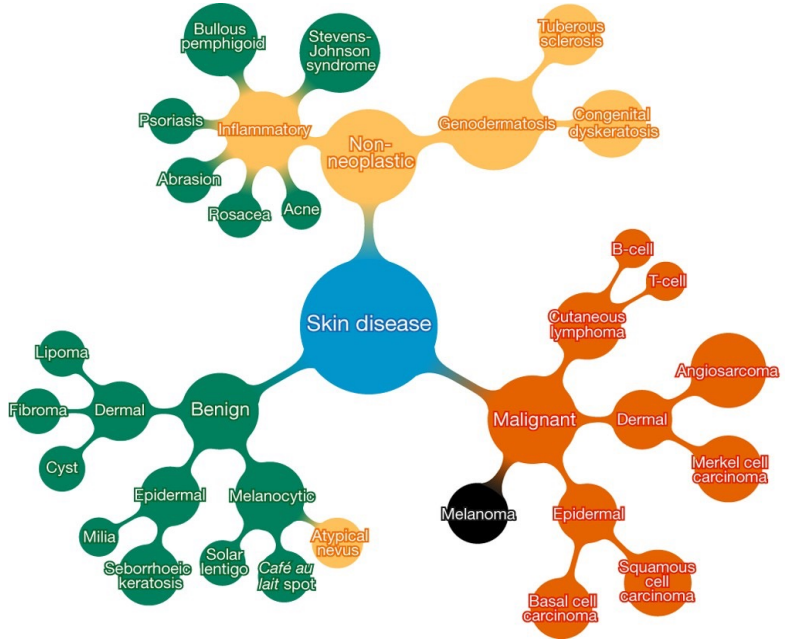
- **Methodology:**
  - **Pretraining:** Pretrain the backbone on the ImageNet dataset
  - **Fine-tuning:** Remove and retrain the final classification layer and fine-tune (all layers) on the skin lesion dataset
- **Backbone:**
  - Google's Inception v3 CNN
  - 48 layers deep
- **Optimization:**
  - Standard cross-entropy loss over 757 training classes

CNNs for Skin Lesion Classification: Model



CNNs for Skin Lesion Classification: Classes





- Training: 757 classes
- Inference:
  - 3 classes: benign, non-neoplastic, malignant
  - 9 classes: second-level nodes

$$P(u) = \sum_{v \in C(u)} P(v)$$

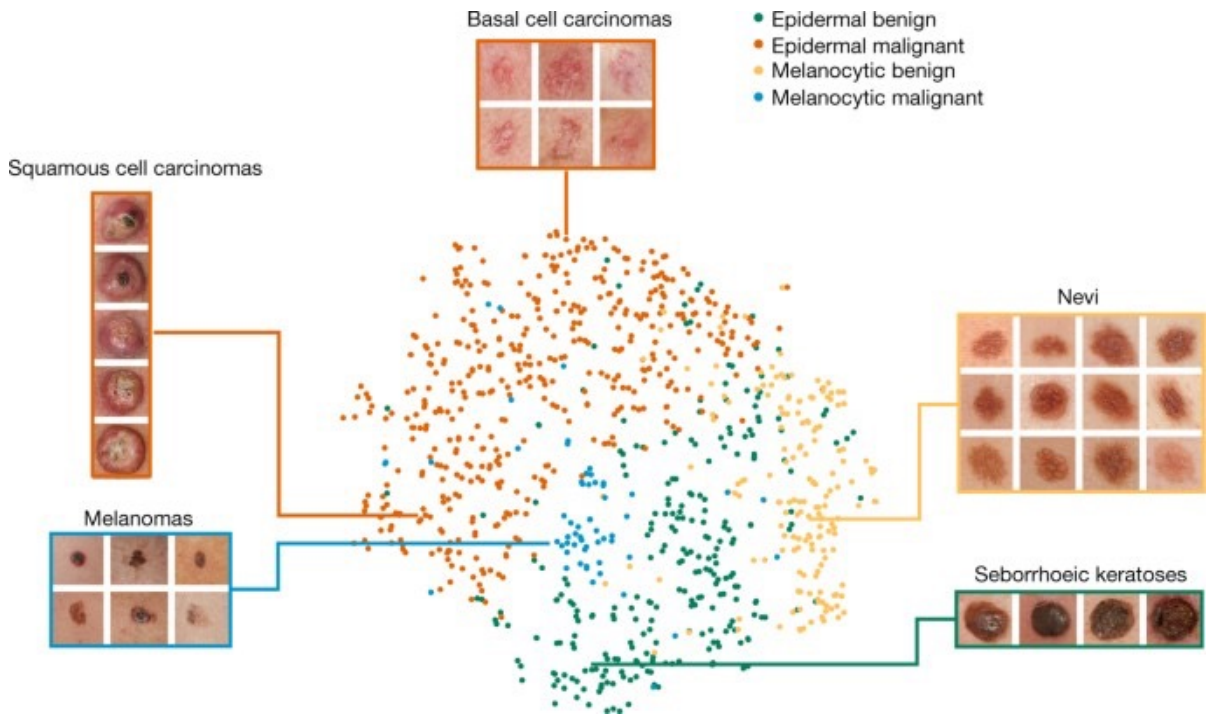
child nodes of u according to taxonomy

CNNs for Skin Lesion Classification: Results

Classifier	Three-way accuracy
Dermatologist 1	65.6%
Dermatologist 2	66.0%
CNN	69.4 ± 0.8%
CNN - PA	**72.1 ± 0.9%**

Classifier	Nine-way accuracy
Dermatologist 1	53.3%
Dermatologist 2	55.0%
CNN	48.9 ± 1.9%
CNN - PA	**55.4 ± 1.7%**

How do CNN internal representations look like?



Discussion on ImageNet Pretraining

Are features from ImageNet well transferable to medical images?

Can we do better?

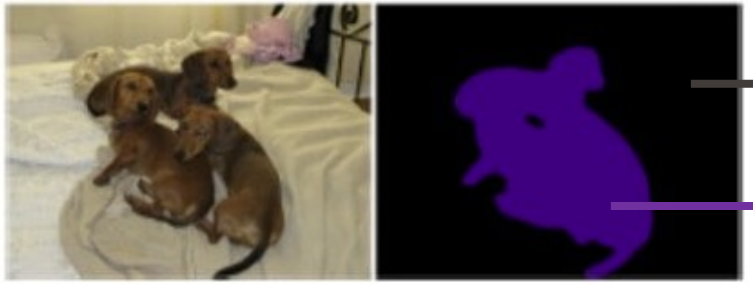
Self-supervised learning (Lecture 10)

Biomedical Applications: Medical Image Segmentation

- Reza Azad et al. Medical Image Segmentation Review: The Success of U-Net. *aRxiv* 2022
- Mohammad Havaei et al. Brain tumor segmentation with Deep Neural Networks. *Medical Image Analysis* 2017

Image Segmentation

Image segmentation is a **pixel-level** classification.

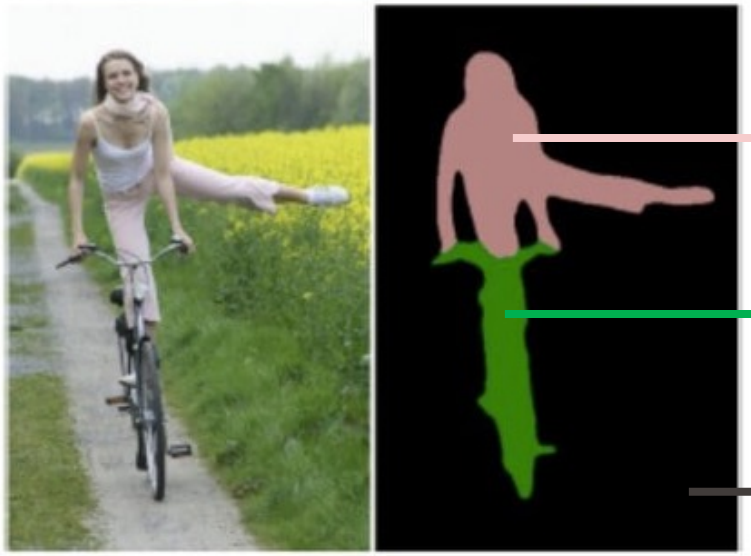


Background

Foreground

We can segment the images into foreground and background (pixel-level **binary classification**).

OR



Human

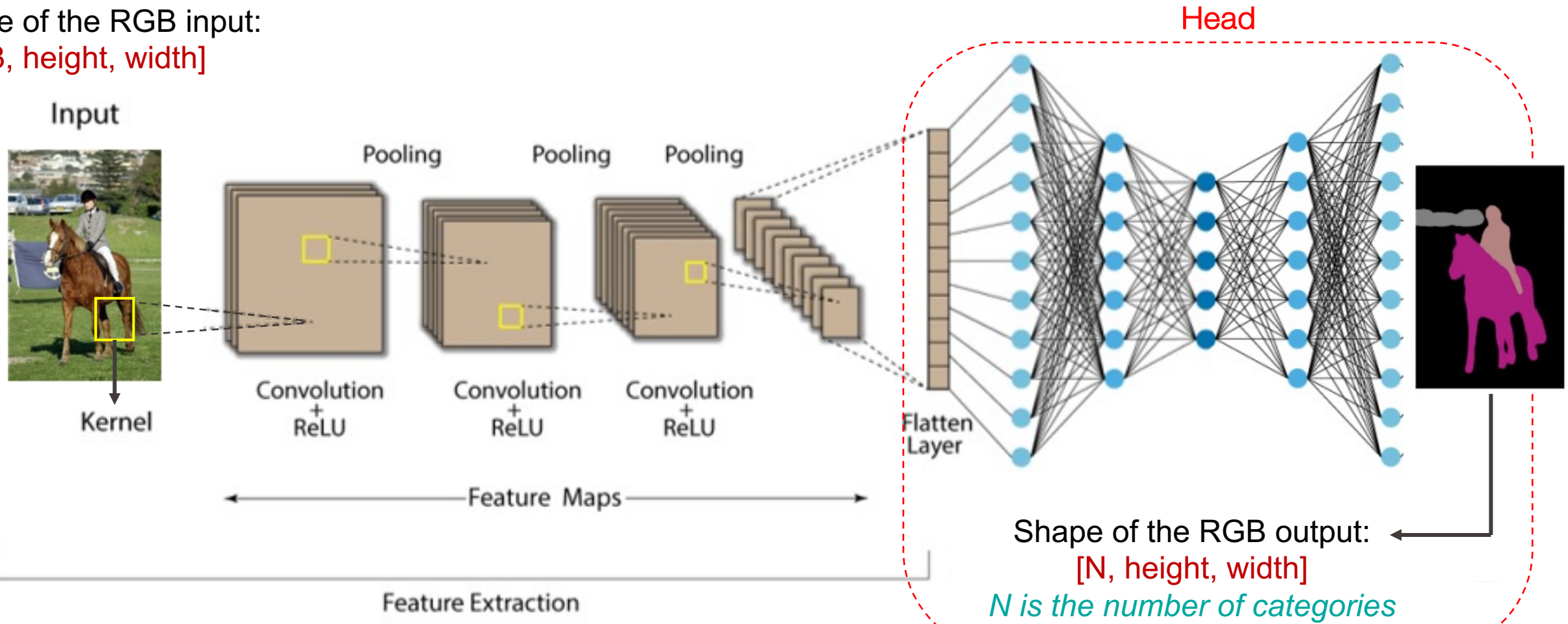
Bicycle

Background

We can segment the images based on different categories. (pixel-level **multiclass classification**)

Image Segmentation

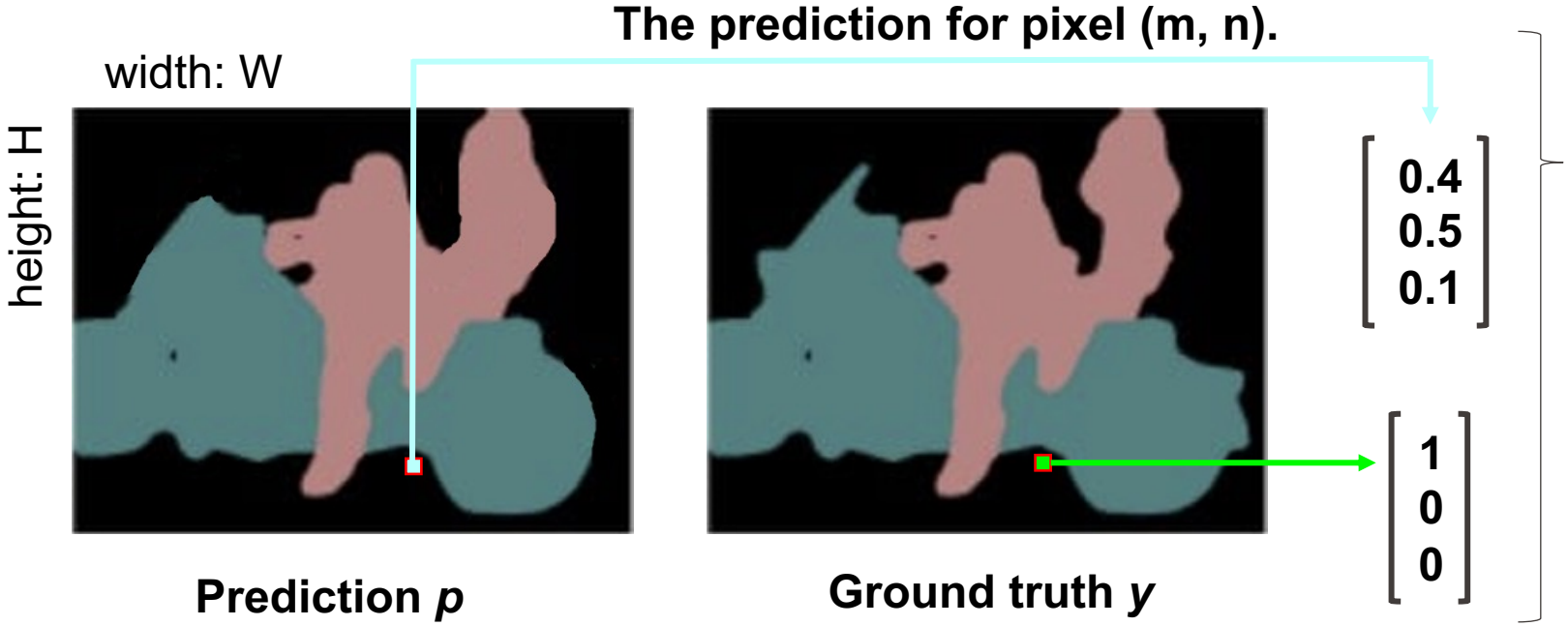
Shape of the RGB input:
[3, height, width]



For each pixel, the model output a one-hot prediction of N categories.

Loss Function for Image Segmentation

Since image segmentation is equal to pixel-level classification, it is natural to use **pixel-level cross-entropy loss.**



We apply cross-entropy to each pixel-wise prediction :

$$\mathcal{L}_{CE}(\mathrm{m},\mathrm{n}) = -\sum_i y_i^{mn} \log(p_i^{mn})$$

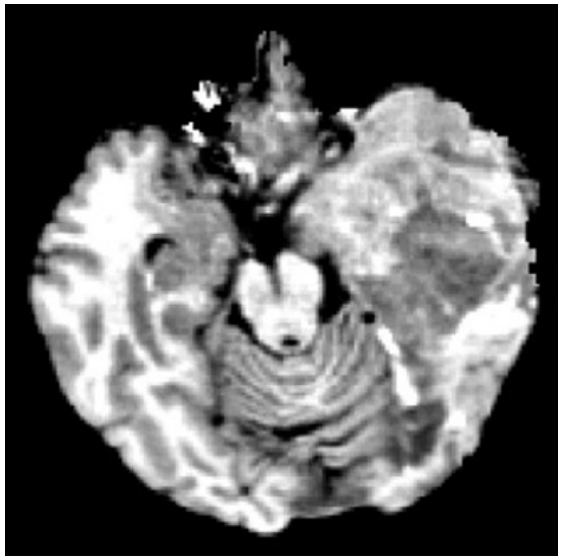
Then we add all the pixel-wise cross-entropy loss :

$$\mathcal{L}_{total} = \frac{1}{HW}\sum_{m,n} \mathcal{L}_{CE}(\mathrm{m},\mathrm{n})$$

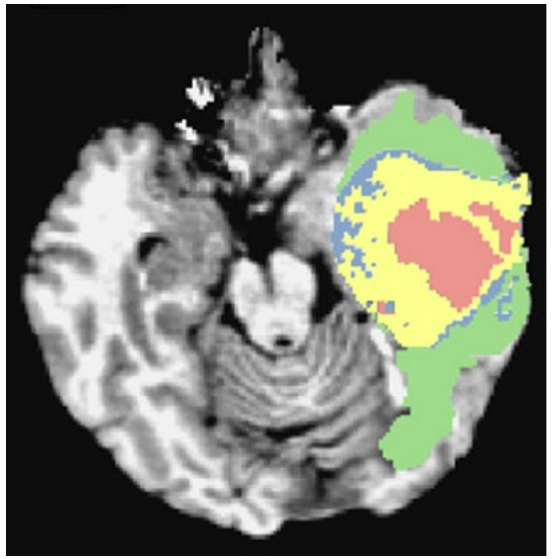
Click this link to see more loss functions for image segmentation.
Dice Loss will be covered in tomorrow's exercise session!

Brain Tumor Segmentation

- Brain tumor segmentation from MRI images can have a great impact on improved diagnostics, growth rate prediction, and treatment planning
- Dataset: 6,000 2D images



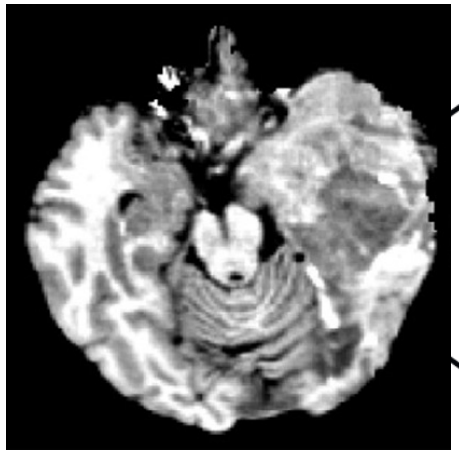
MR image



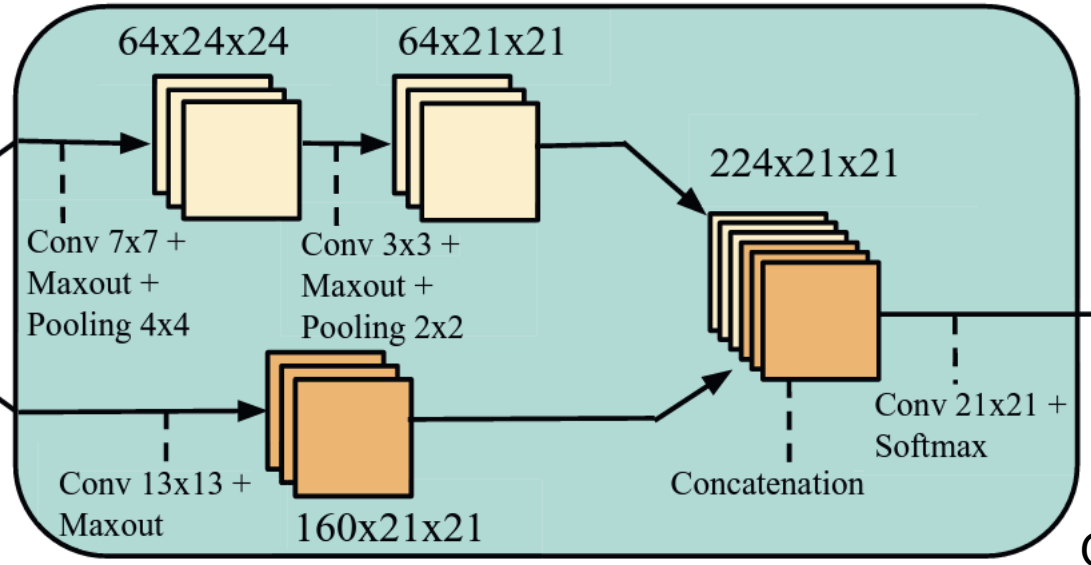
MR image + masks

Mohammad Havaei et al. Brain tumor segmentation with Deep Neural Networks. *Medical Image Analysis* 2017

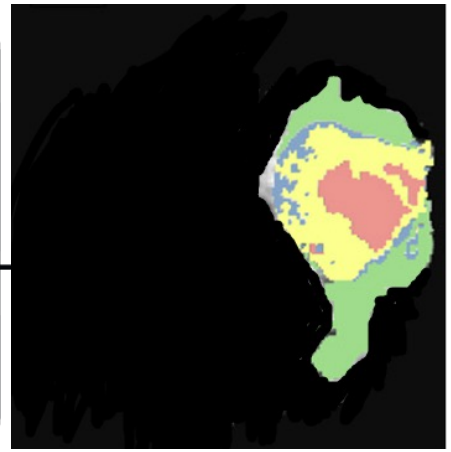
CNN for Brain Tumor Segmentation



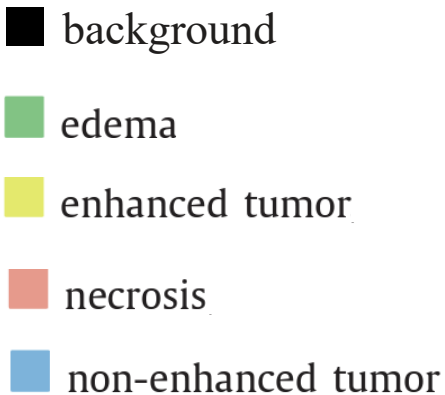
Input: MR image



CNN



Output: segmentation



U-Net is typically used in recent medical image segmentation, and it will be covered in tomorrow's exercise session!

Brain Tumor Segmentation: Results

- **Evaluation metric – Dice coefficient:**

$$Dice = \frac{2|G \cap P|}{|G| + |P|}$$ (G: ground truth P: prediction)

Since G and P are in the same size, Dice coefficient represent the overlapping rate.

- **Result:**

Rank	Method	Dice
4	TwoPathCNN*	0.85
9	LocalPathCNN*	0.85
10	AverageCNN*	0.84
14	GlobalPathCNN*	0.82
14	TwoPathCNN	0.78
15	LocalPathCNN	0.77

The best method achieves 0.85 of overlapping rate!

Recap

- CNNs main building blocks:
  - Convolutional layer: sparse connectivity, parameter sharing, equivariant representations
  - Pooling layer: no trainable parameters
- CNNs learn hierarchical representations across layers
- We can visualize what CNNs learn
- ResNet architecture: skip connections
- Medical images challenges:
  - Large and complex images, training data is small, fine-grained differences and only small part of image is relevant for classification, context is important
- Typical strategy:
  - Pretrain on other datasets (e.g., ImageNet)
  - Fine-tune on medical images

Additional Readings

- Stanford CS231N: https://cs231n.github.io/understanding-cnn
- Waite et al. Interpretive Error in Radiology. *American Journal of Roentgenology 2016*
- Rajpurkar et al. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. *arXiv* 2017
- Esteva et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 2017
- Reza Azad et al. Medical Image Segmentation Review: The Success of U-Net. aRxiv 2022
- Mohammad Havaei et al. Brain tumor segmentation with Deep Neural Networks. *Medical Image Analysis* 2017
- Zhou et al. Learning deep features for discriminative localization. *CVPR* 2015

Any Feedback?

Give us feedback on the lecture:

- https://go.epfl.ch/cs502-lecture-2-feedback