Convolutional Neural Networks

Computer Vision I

CSE 252A

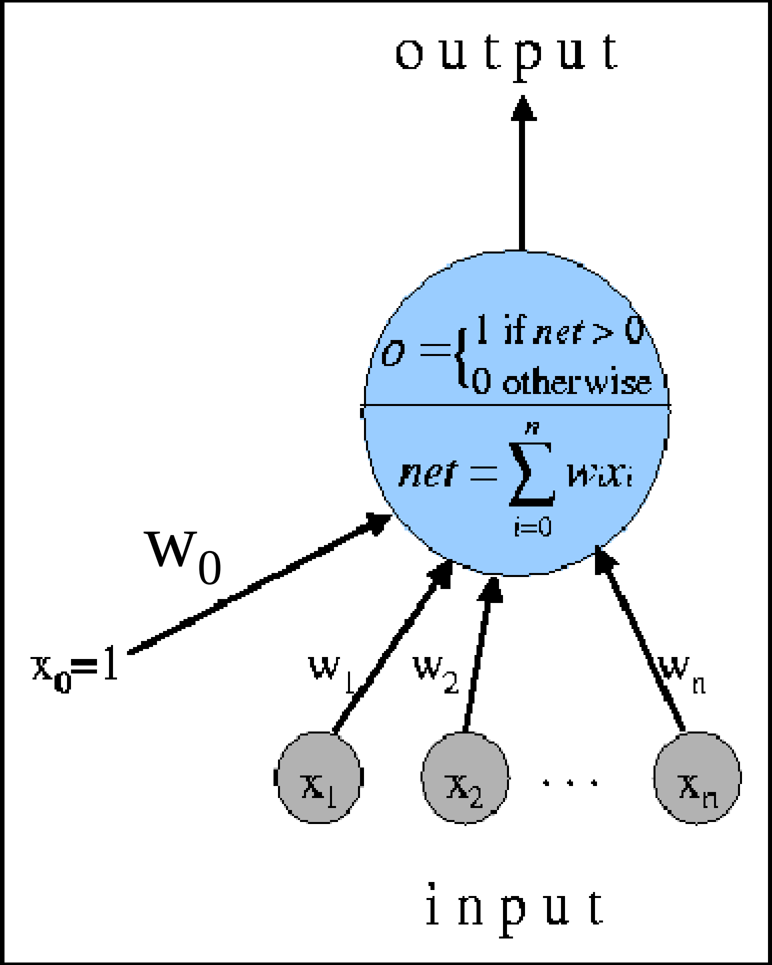
Lecture 16

Announcements

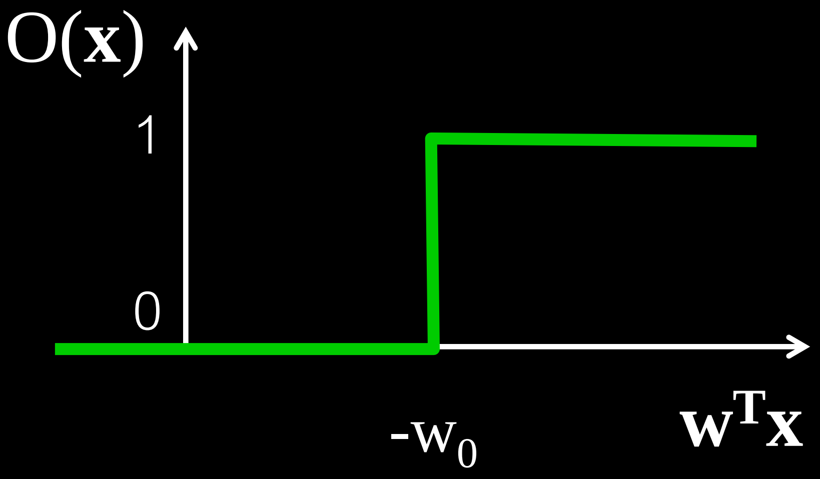
- Assignment 4 is due Dec 6, 11:59 PM

Neural Networks

Perceptron



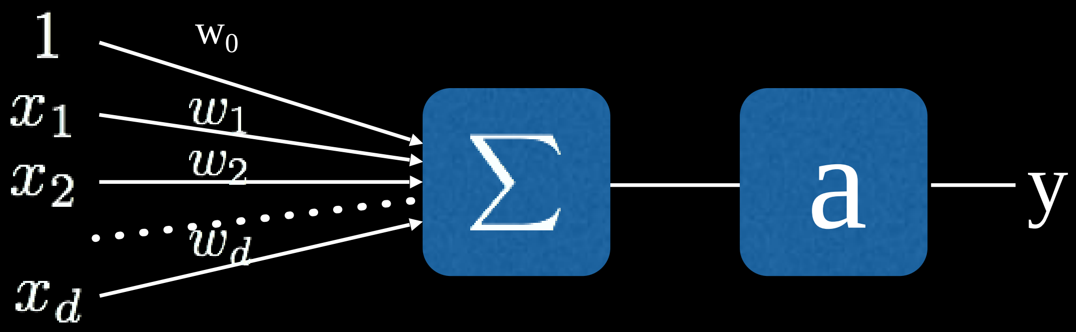
output
$o = \begin{cases} 1 \text{ if } net > 0 \\ 0 \text{ otherwise} \end{cases}$
$net = \sum_{i=0}^{n} w_i x_i$
w_0
$x_0=1$
w_1
w_2
w_n
x_1
x_2
...
x_n
input



Note: For $x=(x_1,\ldots, x_2)$, x_i can be binary or a real number

$$o(x_1,\ldots,x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + \cdots + w_nx_n > 0 \\ 0 & \text{otherwise.} \end{cases}$$

The nodes of multilayered network



$y(\mathbf{x}; \mathbf{w}) = a(\mathbf{w}^T\mathbf{x}+w_0)$

x: input vector

w: weights

w_0: bias term

a: activation function

$y(\mathbf{x}; \mathbf{w}) = a(\mathbf{w}^T\mathbf{x})$

x: input vector padded with 1

w: weights including bias

a: activation function

Feedforward Networks

- These networks are composed of functions represented as **“layers"**

$$y(\mathbf{x}) = a_3(a_2(a_1(\mathbf{x}; w_1); w_2); w_3)$$

with weights w_i associated with layer i and a_i is the activation function for layer i.

- y(***x***) can be a scalar or a vector function.

Universal Approximation Theorem

- **Universal Approximation Theorem**: A feedforward neural network with a linear output layer and one or more hidden layers with ReLU **[Leshno et al. ’93]**, or sigmoid or some other “squashing” activation function **[Hornik et al. ’89, Cybenko ’89]** can approximate any continuous function on a closed and bounded subset of $\mathbb{R}^n$ This holds for functions mapping finite dimensional discrete spaces as well.

- If we have enough hidden units we can approximate “any” function! … but we may not be able to train it.

Universal Approximation Theorem: Caveats

- So even though “any” function can be approximated with a network as described with single hidden layer, the network may fail to train, fail to generalize, or require so many hidden units as to be infeasible.
- This is both encouraging and discouraging!
- However, **[Montufar et al. 2014]** showed that **deeper networks are more efficient** in that a deep rectified net can represent functions that would require an exponential number of hidden units in a shallow one hidden layer network.
- Deep networks composed on many rectified hidden layers are good at approximating functions that can be composed from simpler functions. And lots of tasks such as image classification may fit nicely into this space.

Optimization for Deep Nets

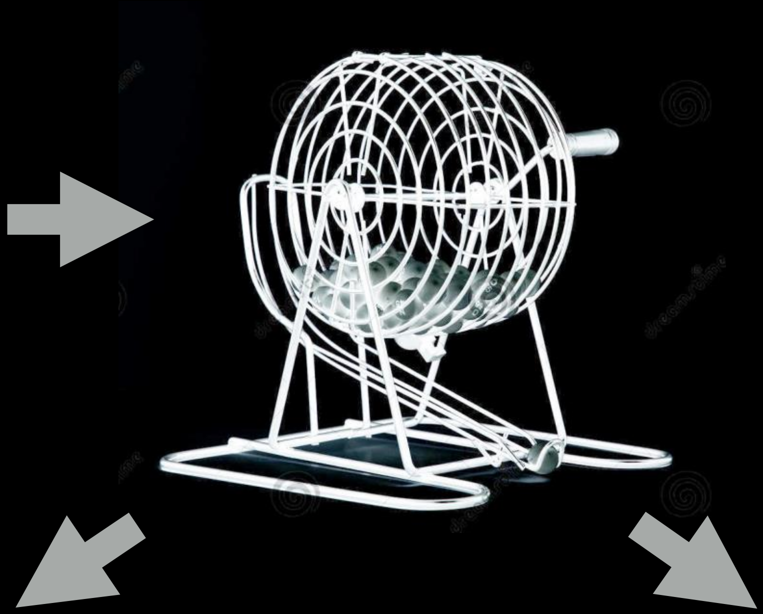
- Although there is a large literature on global optimization, gradient descent-based methods are used in practice.
- Our optimizations for deep learning are typically done in very high dimensional spaces, where the number of weights can run into the millions.
- And for these optimizations, when starting the training from scratch (i.e., some random initialization of the weights), we will need LOTS of labeled training data.

Back propagation

- Basically another name for gradient descent
- Because of nature of network $a_3(a_2(a_1(\mathbf{x};w_1);w_2);w_3)$, gradients with respect to w_i are determined by chain rule
- Can be thought of as “propagating” from loss function to input.
- Adaptive step size methods (e.g., ADAM).

Training and Validation Sets

Labeled Data



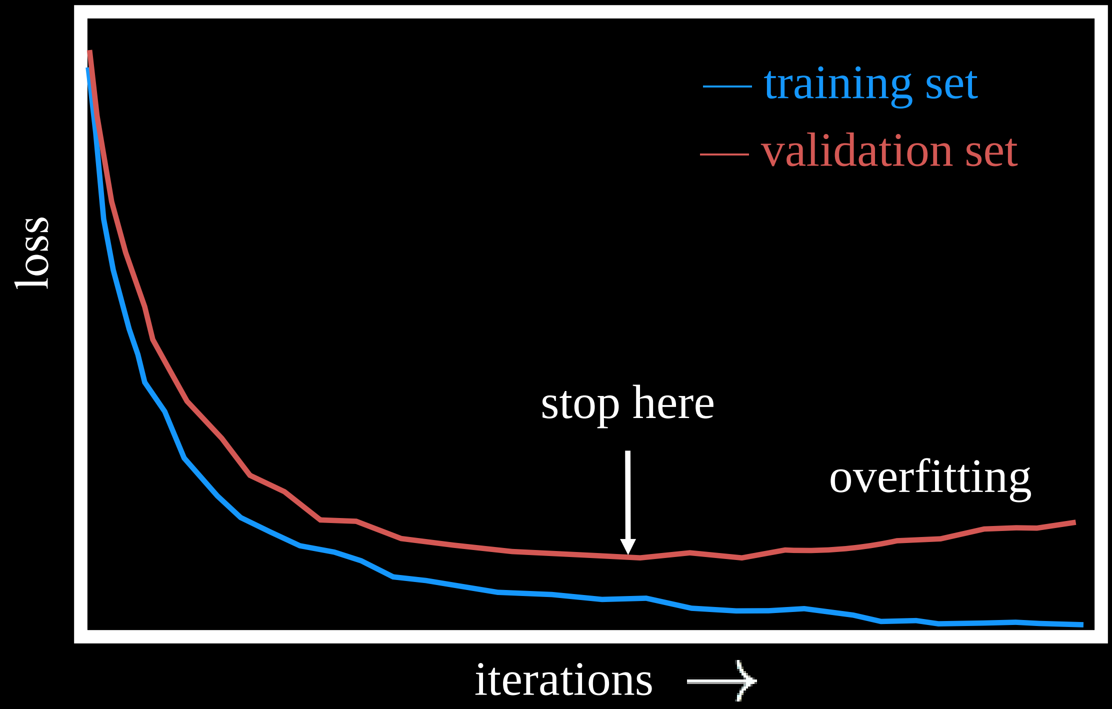
Training Data

Validation Data

NEVER TRAIN ON YOUR VALIDATION SET!

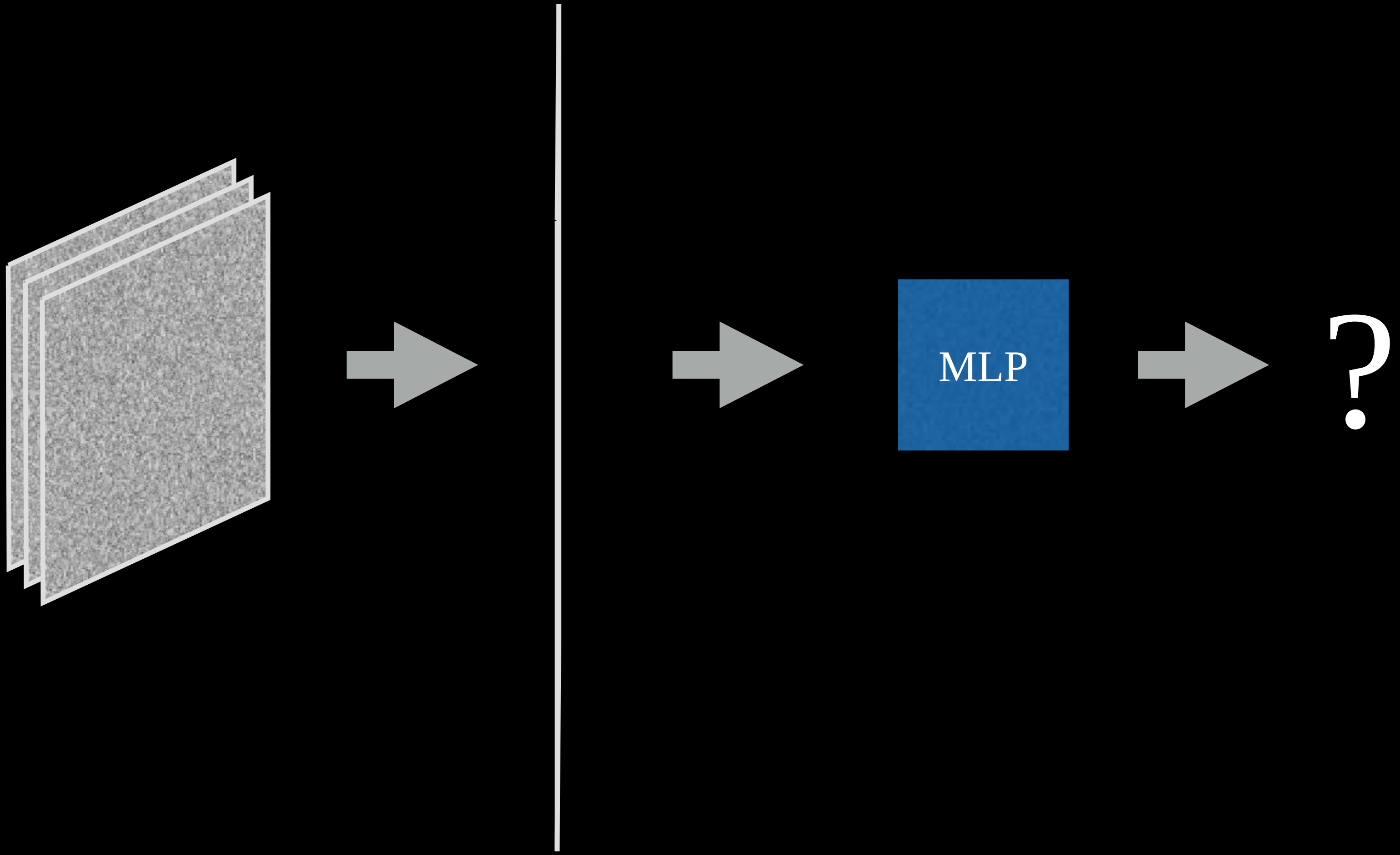
Training

- Regularization
- Early stopping

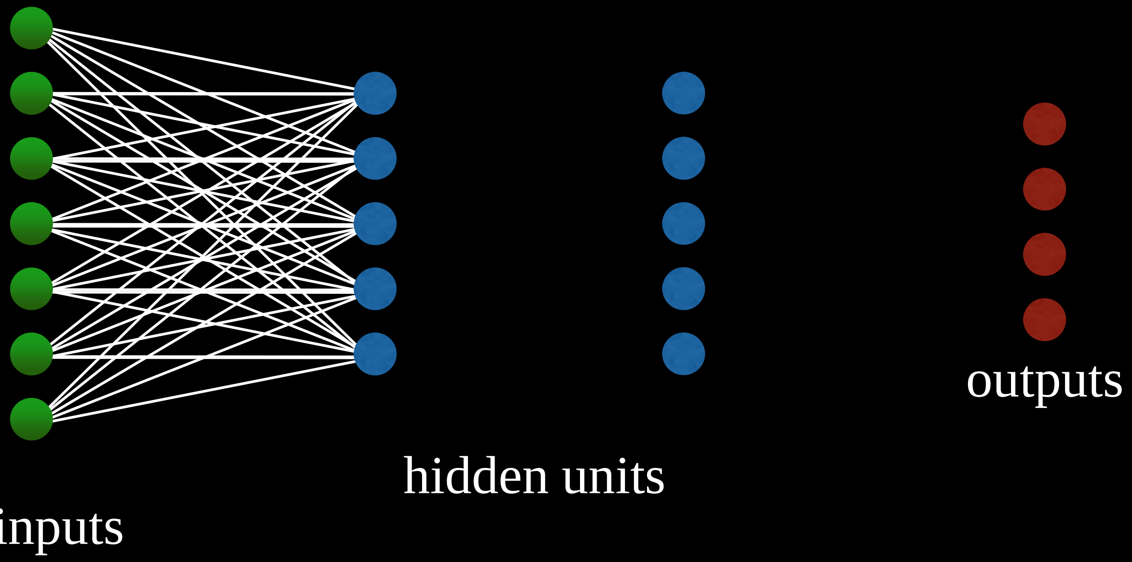


Finally, we get to images…

What if we just vectorized images and stuffed these into a MLP?

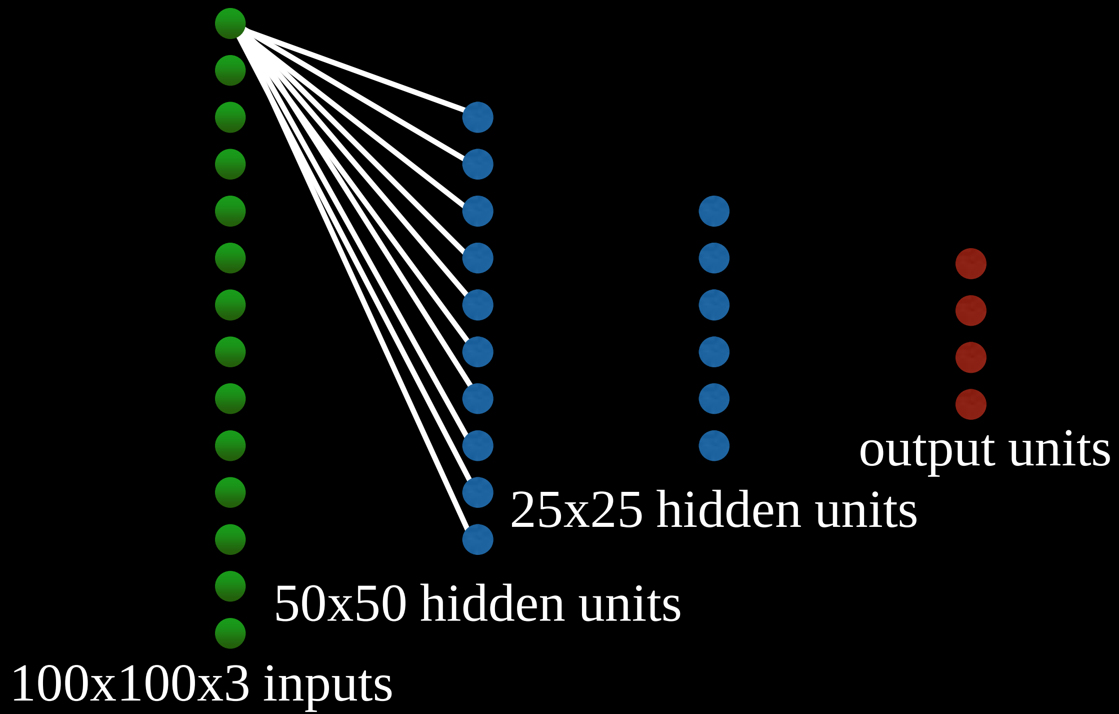


Fully Connected (FC) Layer



Every input unit is connected to every output unit.

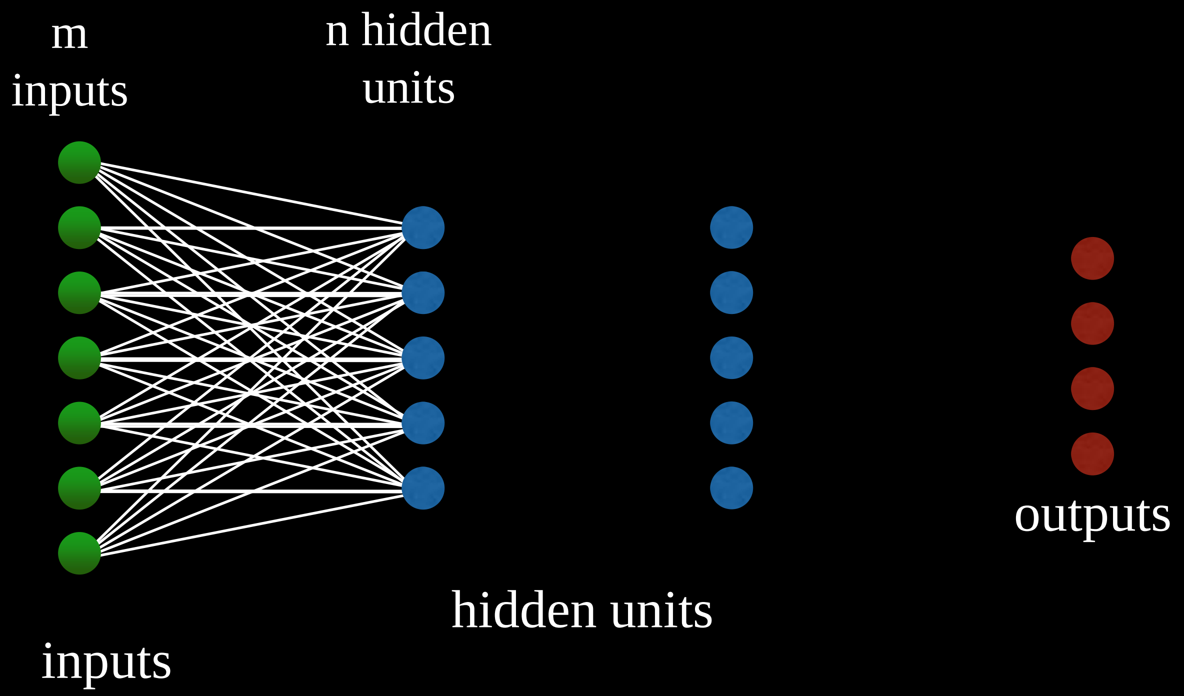
Too many weights and connections!



- This fully connected hidden layer might have 75 million weights!
- And this is just for a thumbnail image and a two layer net.

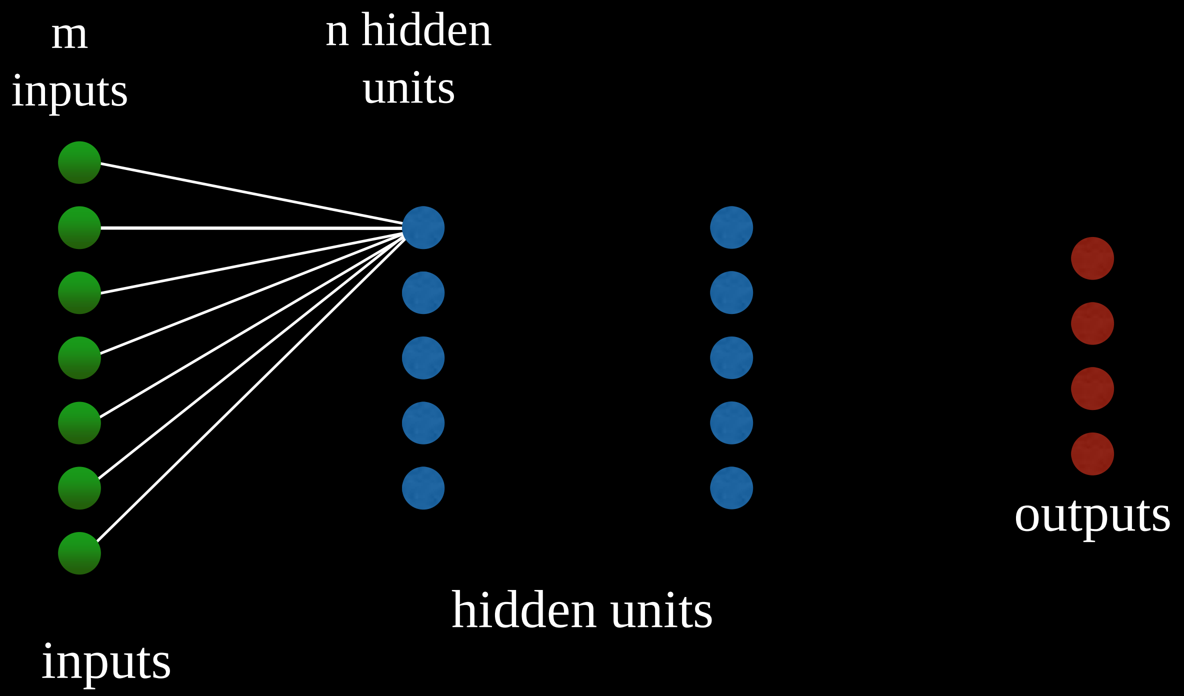
Convolutional Neural Networks

Fully Connected (FC) Layer



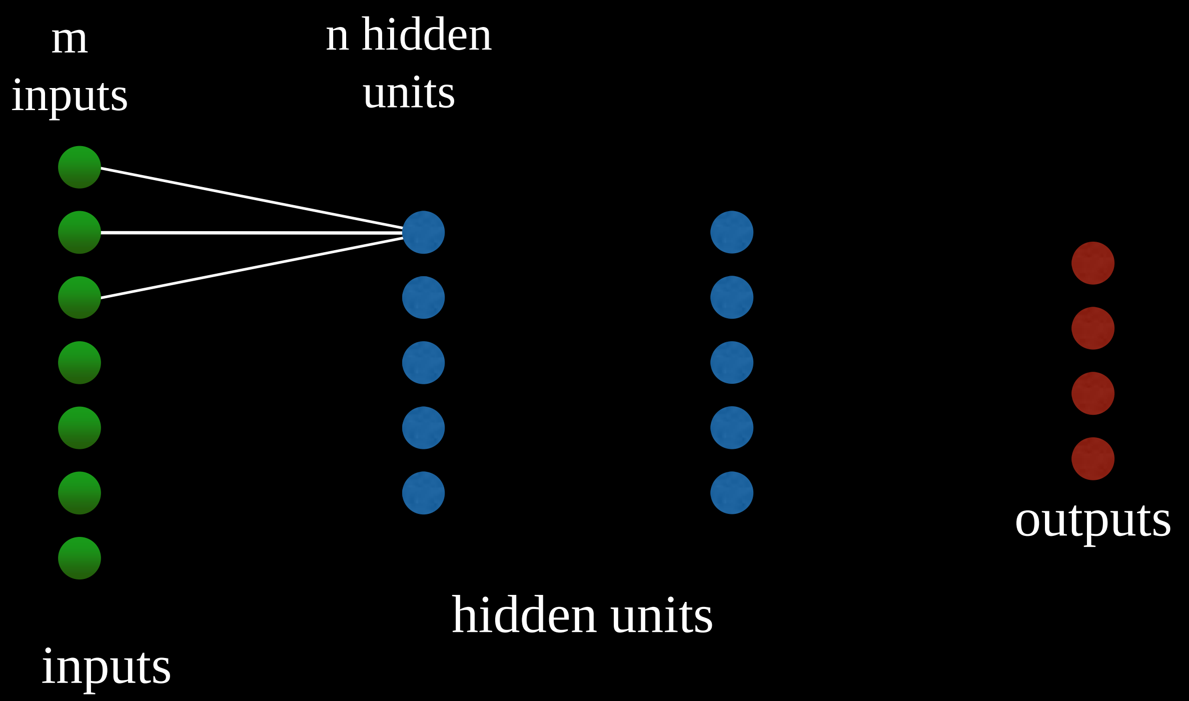
- Every input unit is connected to every output unit.
- m * n weights

Fully Connected (FC) Layer



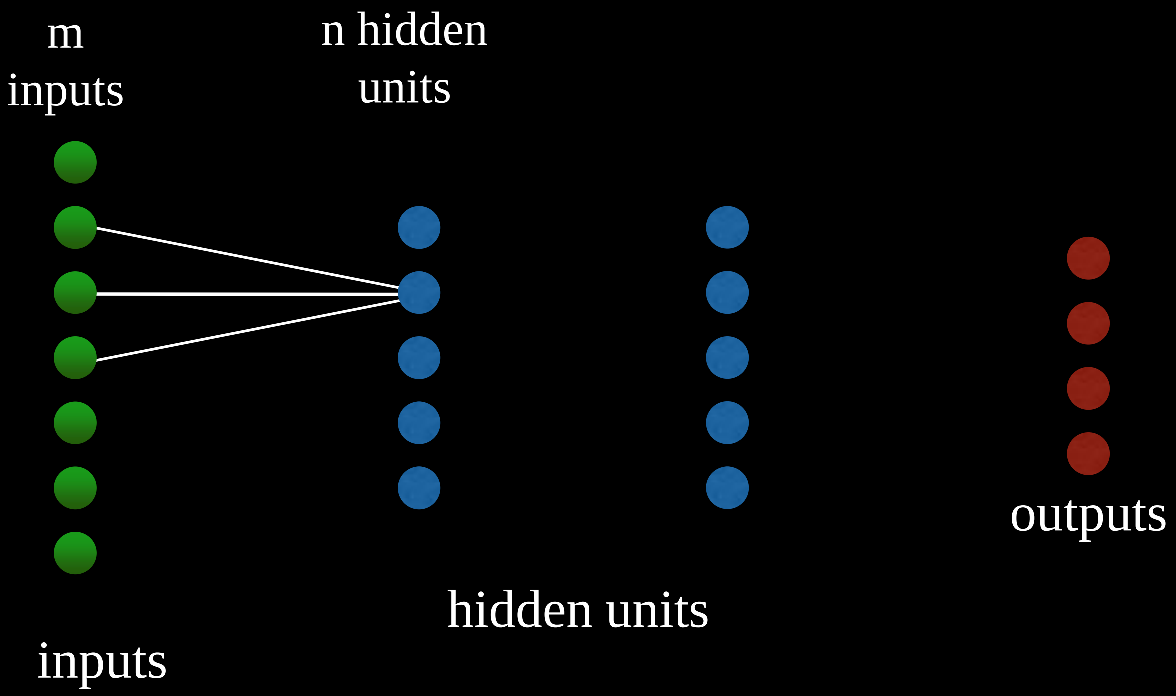
- Consider a hidden unit: it connects to all units from the previous layer
- m weights per hidden unit

Convolutional Layer: Local Connections



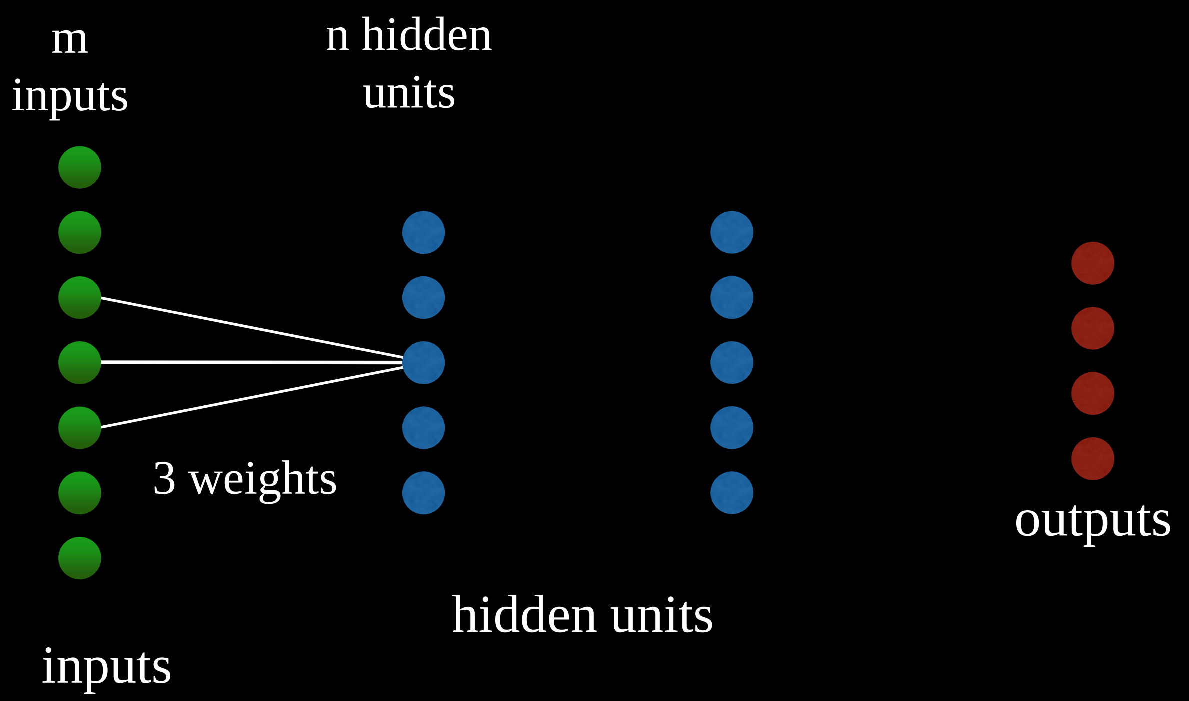
- The connections are spatially local and governed by the kernel size.

Convolutional Layer: Local Connections



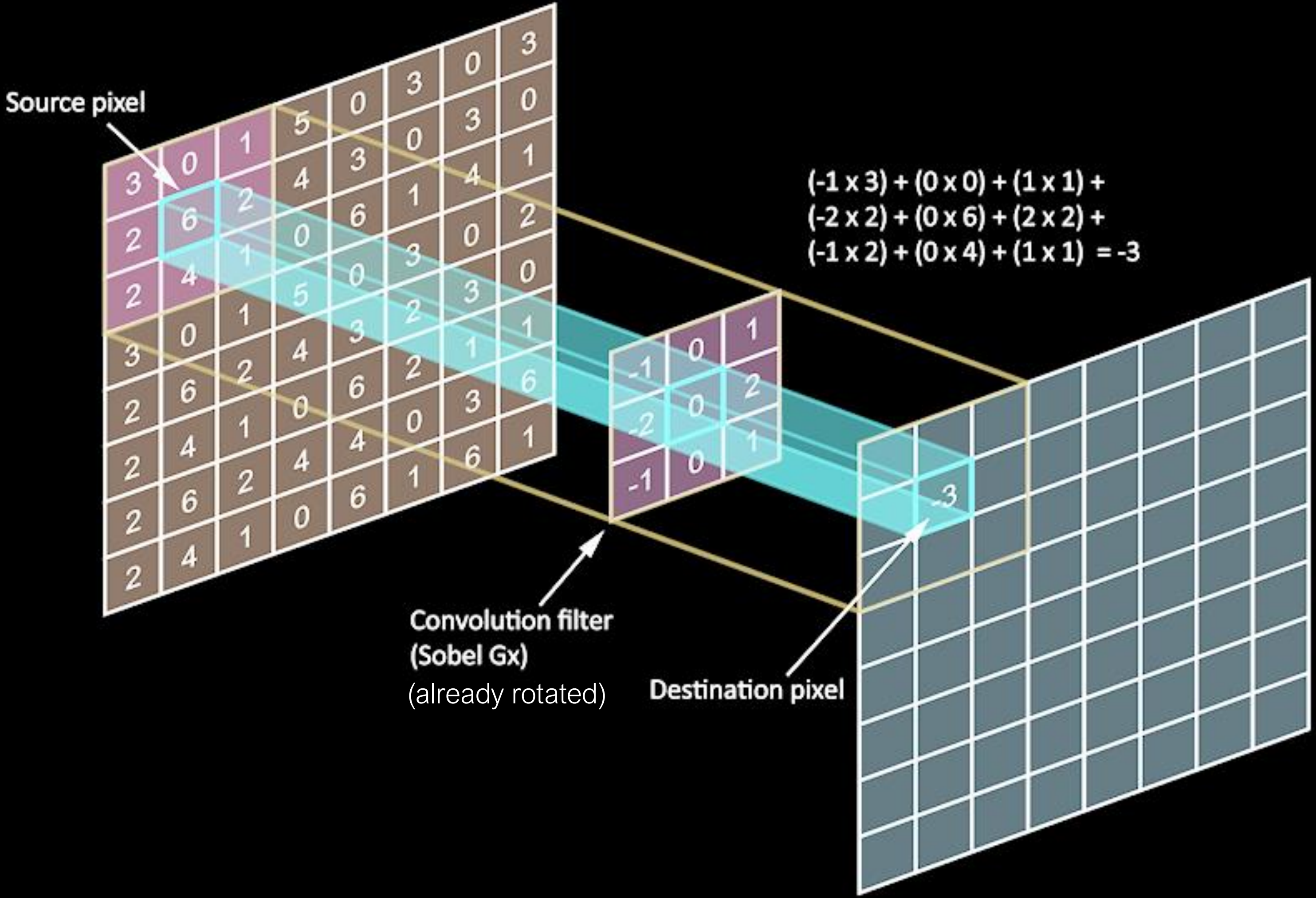
- The connections are spatially local and governed by the kernel size.

Convolutional Layer: Local Connections and Shared Weights



- The connections are spatially local and governed by the kernel size.
- The weights are shared. They are the same for each position.
- So, this is like a convolution kernel.

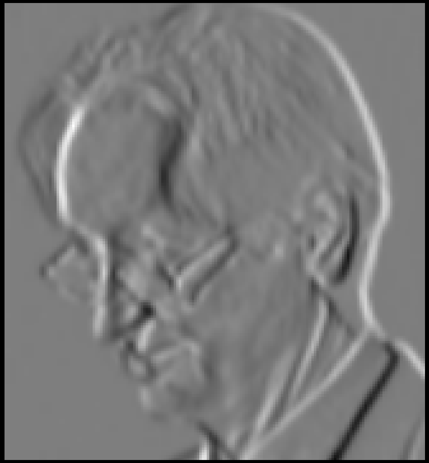
Convolution in 2D


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
(already rotated)
Destination pixel

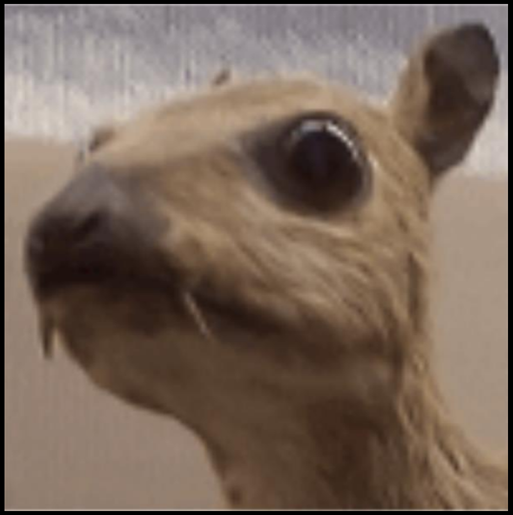
Convolution with 2D Kernel

$$* \quad \boxed{\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}} \quad =$$

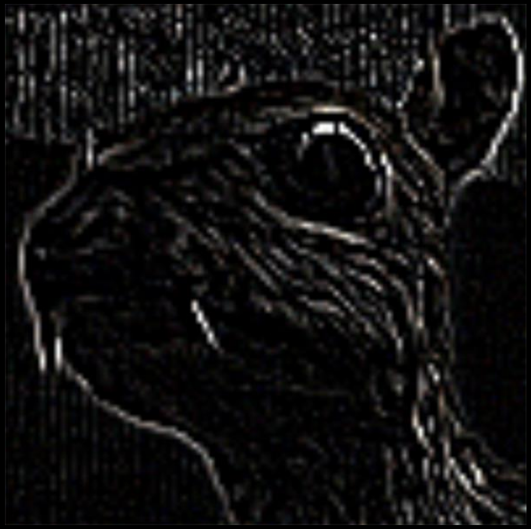
(already rotated)



Convolution with 2D Kernel

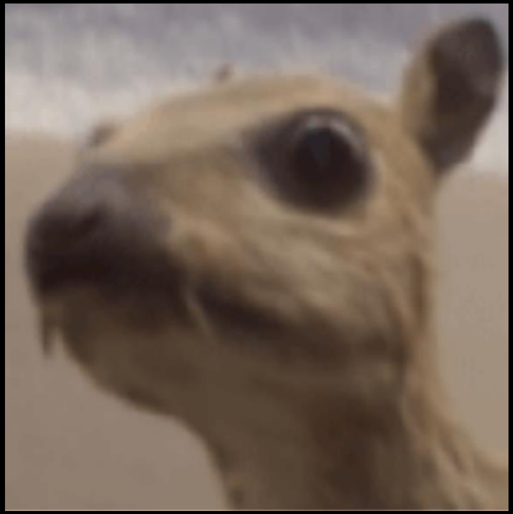


$$* \quad \boxed{\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}} \quad =$$

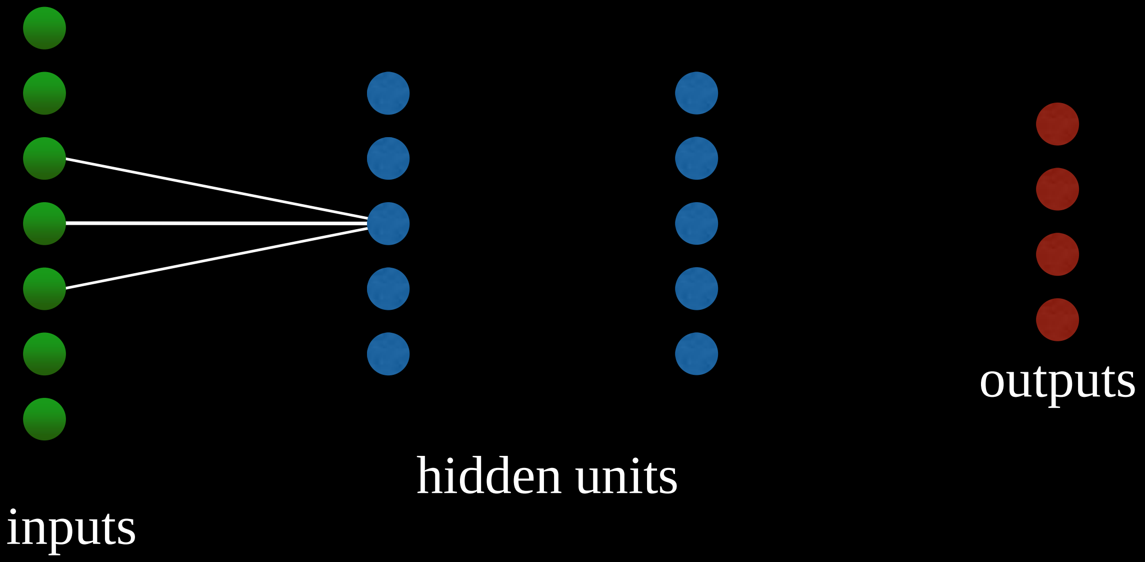


Convolution with 2D Kernel

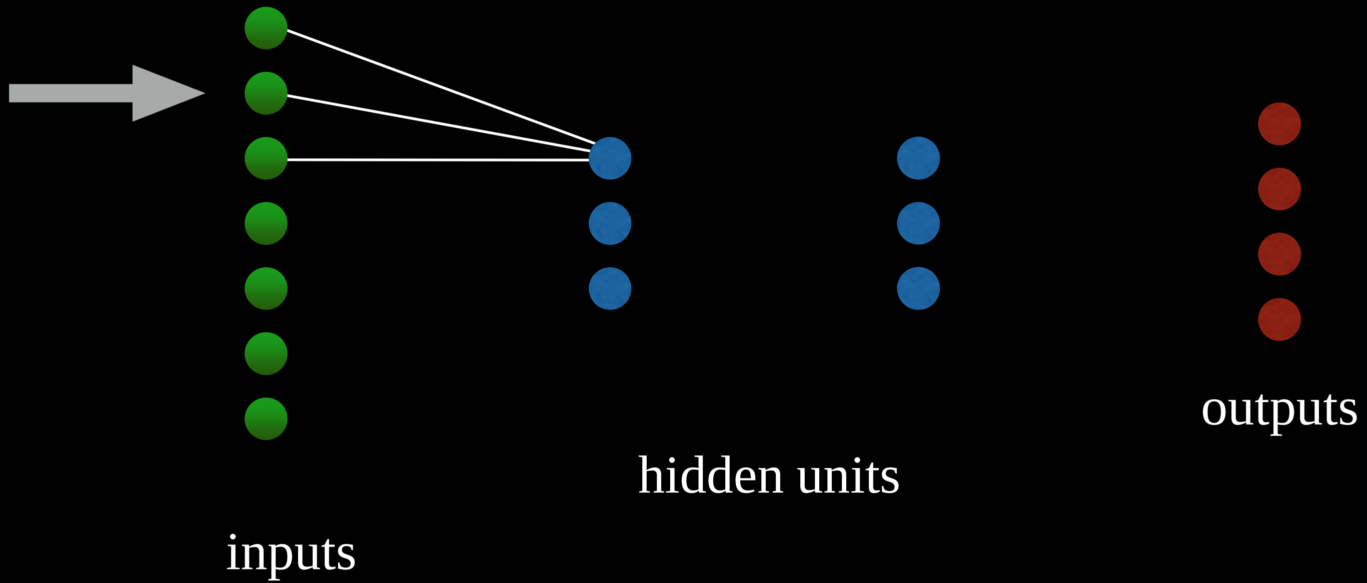
$$* \quad \boxed{\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}} \; / \, 16 \quad =$$



Convolutional Layer: Shared Weights

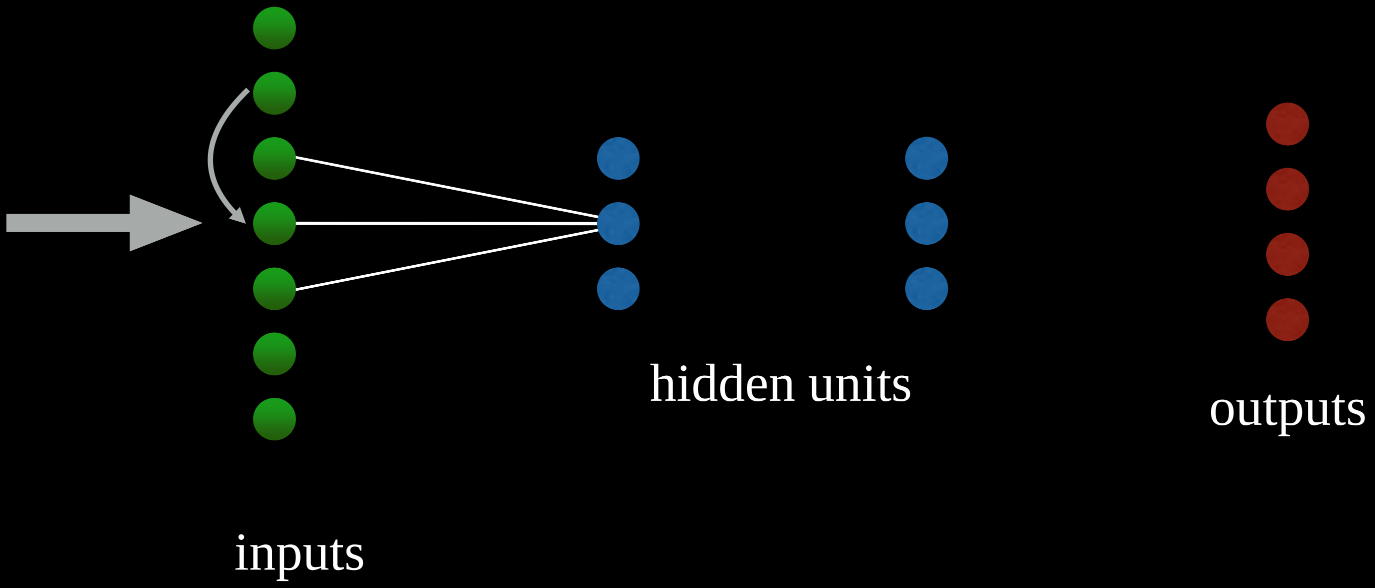


Convolutional Layer: Stride



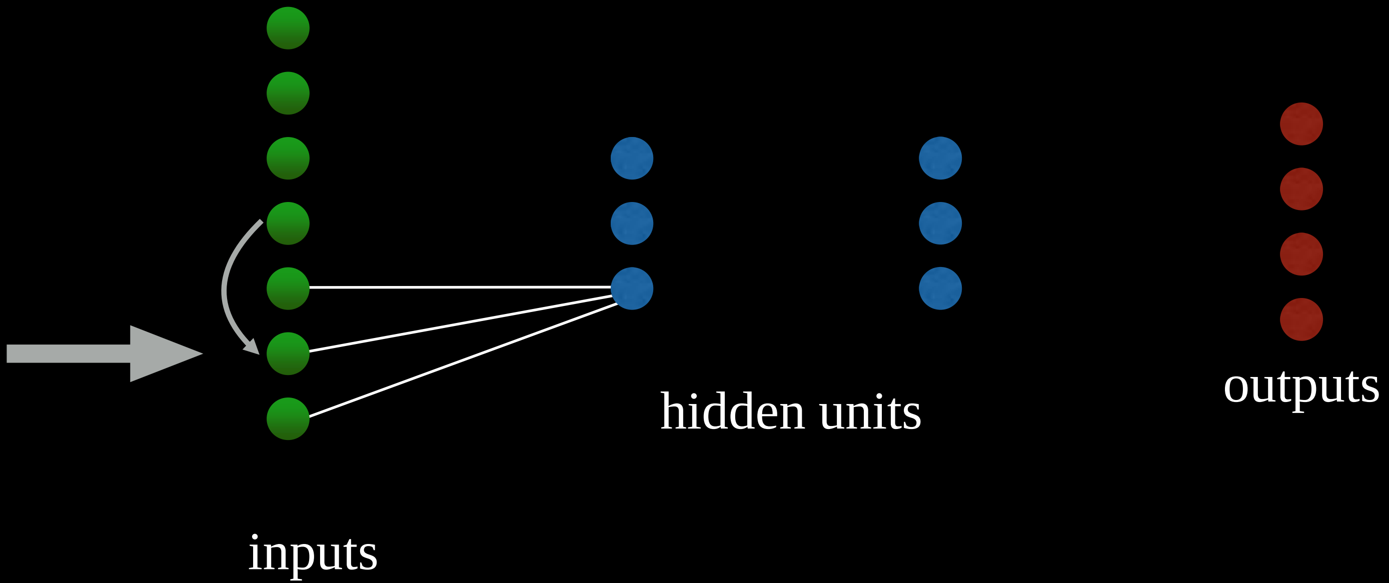
We can skip input pixels by choosing a *stride* > 1.

Convolutional Layer: Stride



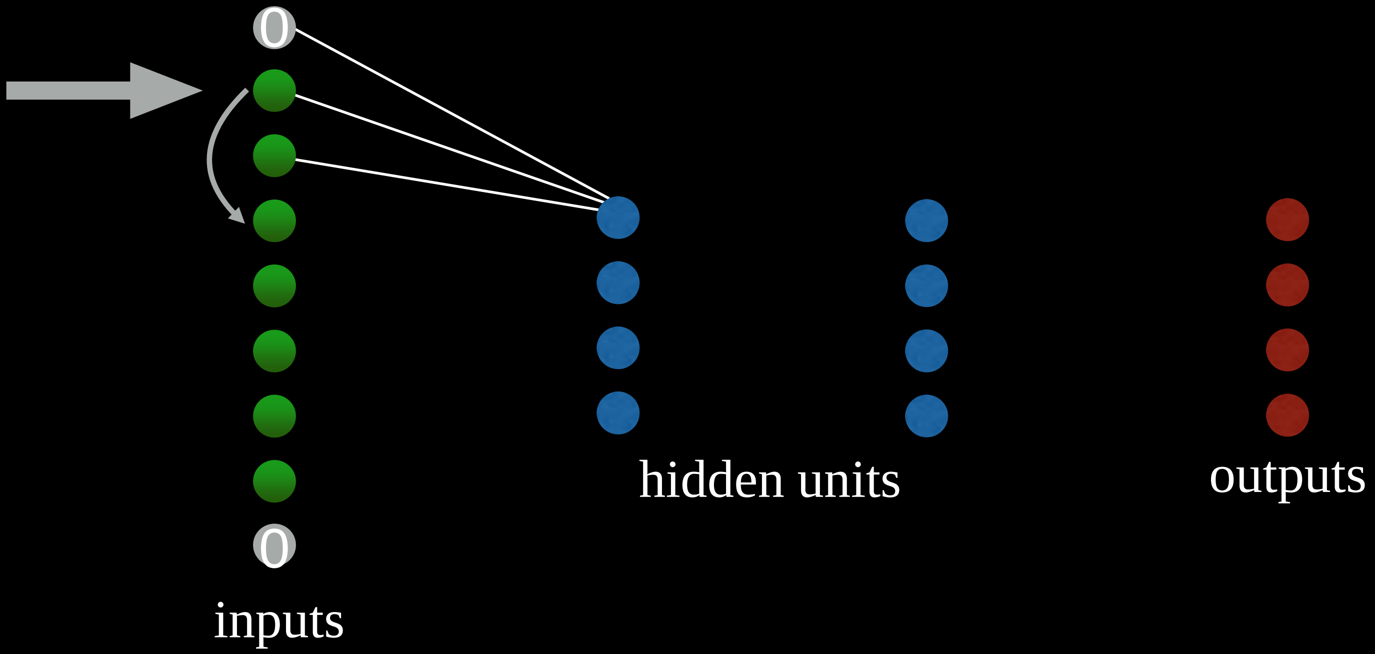
We can skip input pixels by choosing a *stride* > 1.

Convolutional Layer: Stride



The output dim = (input dim - kernel size) / stride + 1.

Convolutional Layer: Padding + Stride



Output dimension = (input dim - kernel size + 2 * padding) / stride + 1

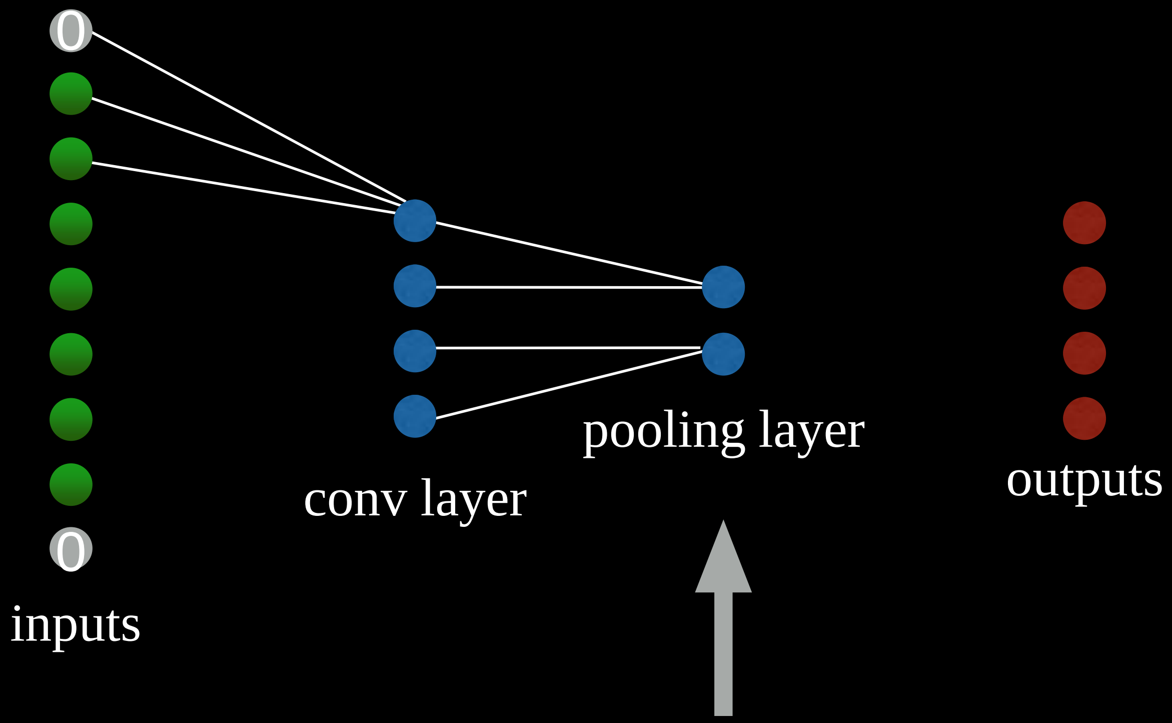
ReLU used with ConvNets

- Just like with our fully connected layers, for our convolutional layers we will follow the linear operation (convolution) with a non-linear squashing function.
- The function to use for now is ReLU.
- But we are not done…there’s one more thing!

Pooling

- We can spatially pool the output from the ReLU to reduce the size of subsequent layers in our network.
- This reduces both computation and the number of parameters that need to be fit and helps prevent overfitting.
- The pooling operation is often the **max** value in the region, but it could be **average**, or **median**, etc.
- The pooling has a stride associated with it that determines the downsampling of the input.

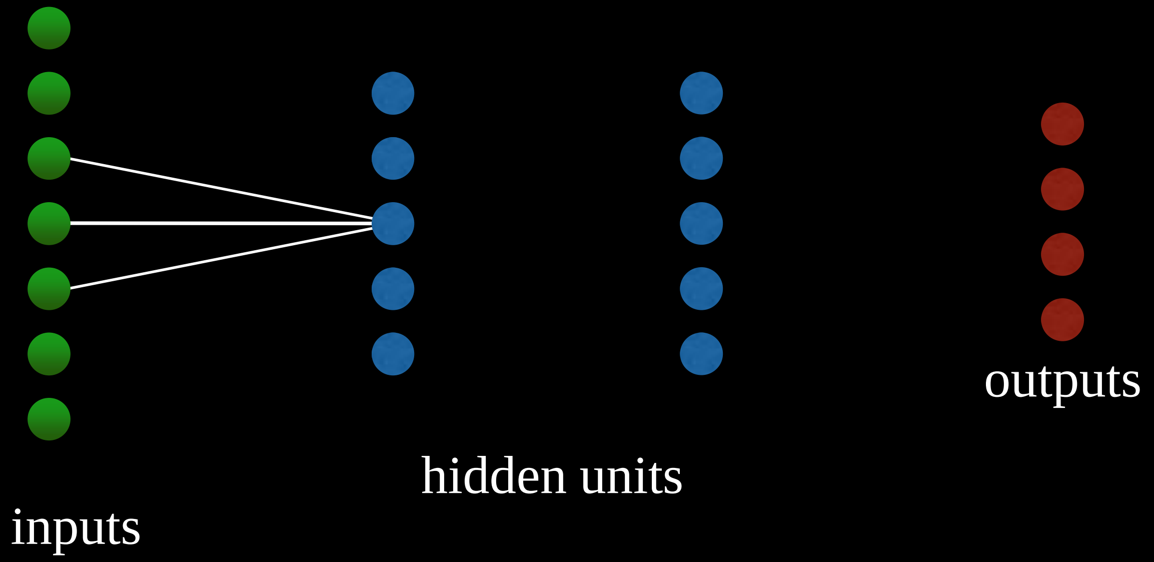
Pooling Layer



The pooling layer pools values in regions of the conv layer.

But wait, there’s more

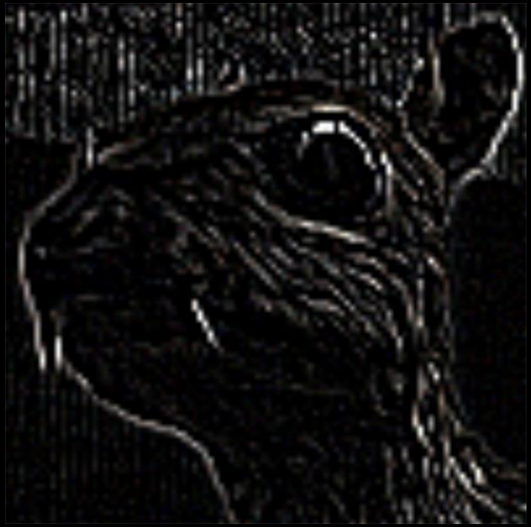
Convolutional Layer: Shared Weights



The weights for the kernel are shared. They are the same for each position.

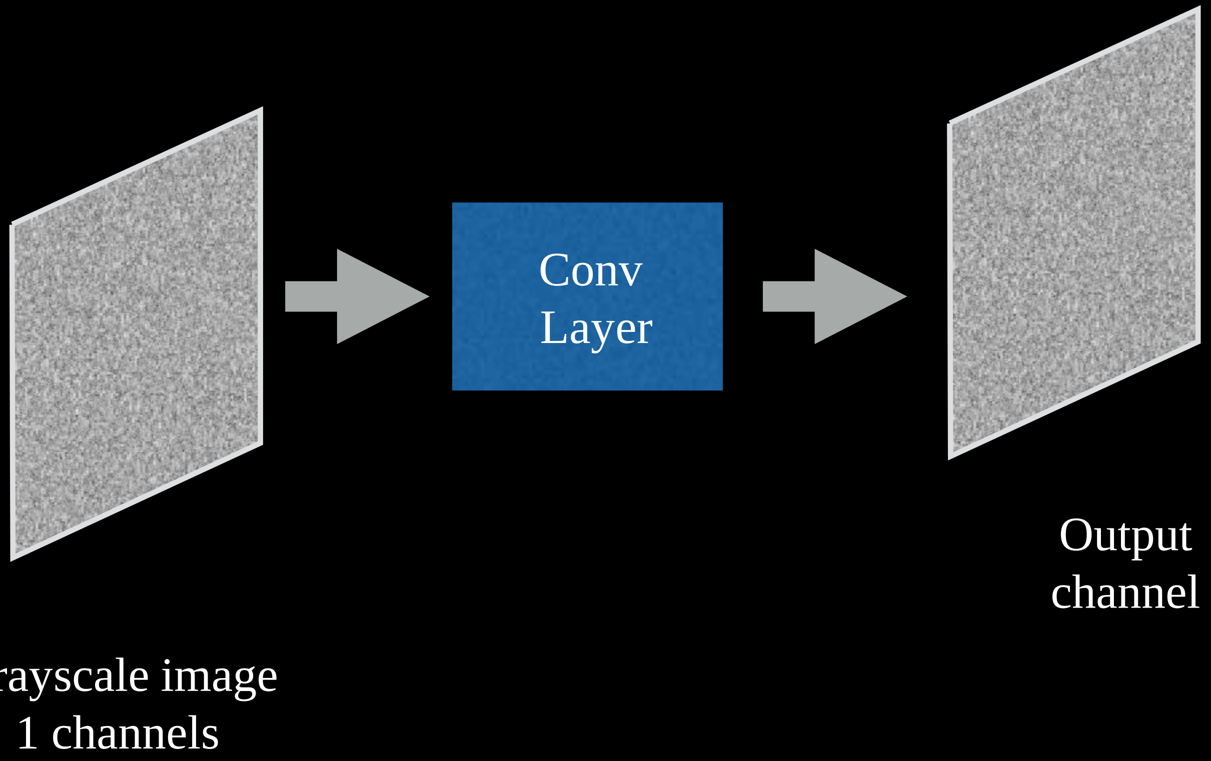
Kernel finds just one type of feature

$$* \begin{array}{|ccc|} \hline -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \\ \hline \end{array} =$$



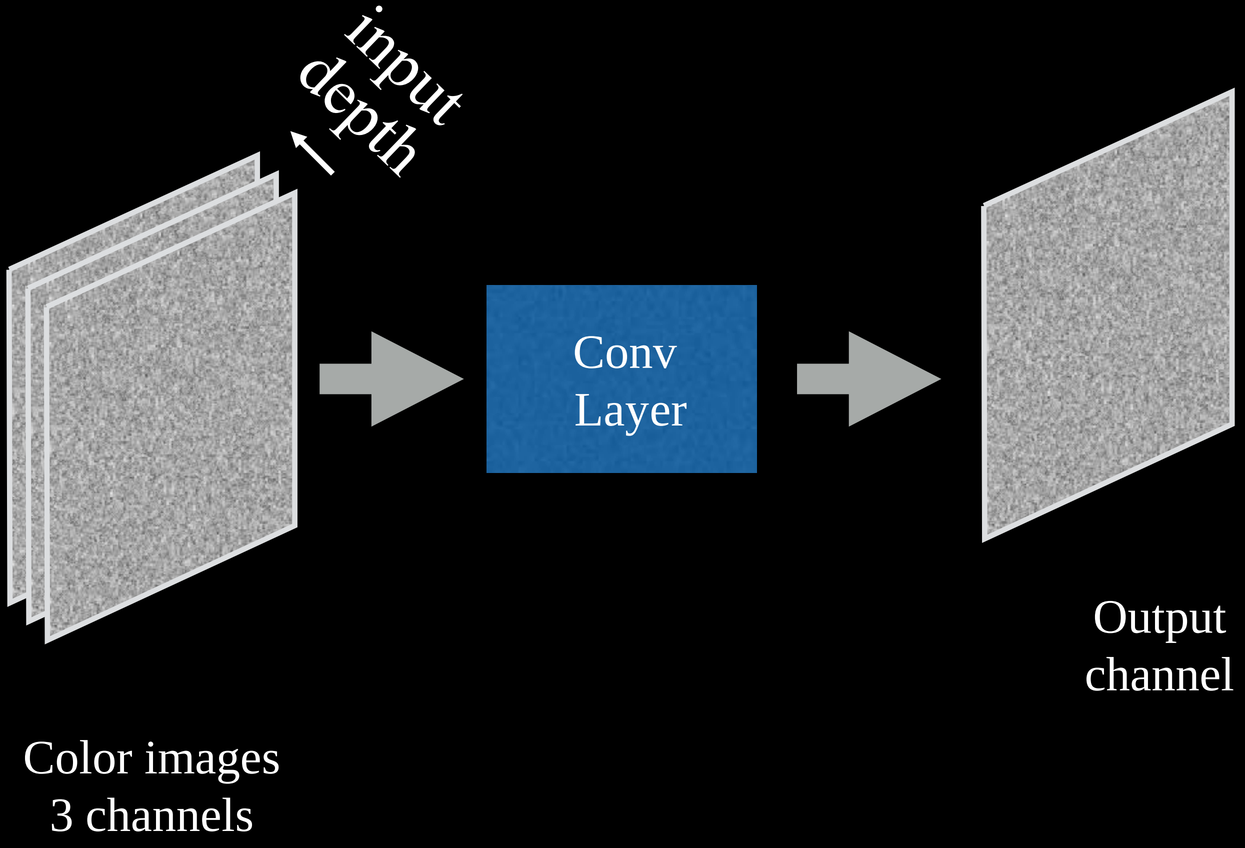
If a kernel shares weights, then it can only extract one type of feature

Single input channels



- Convolution kernel is 3-D: Goes across image dimensions & across channels
- Size = width x height x # input channels

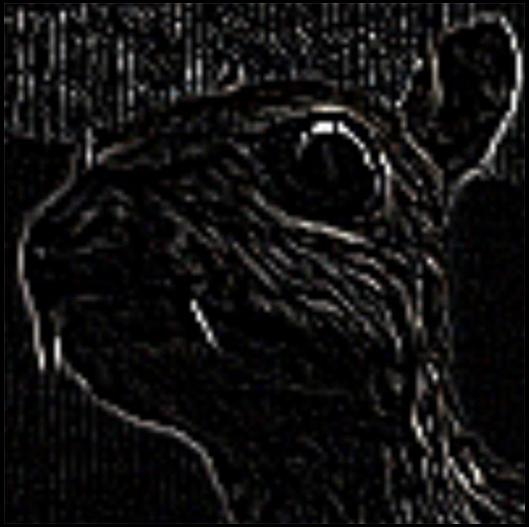
Multiple input channels



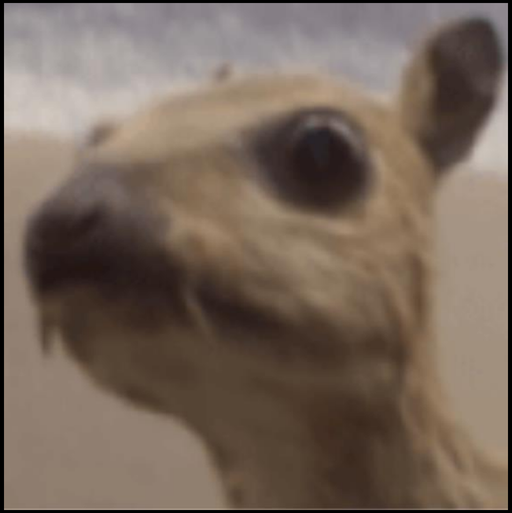
- Convolution kernel is 3-D: Goes across image dimensions & across channels
- Size = width x height x # input channels

Convolution with 2D Kernel

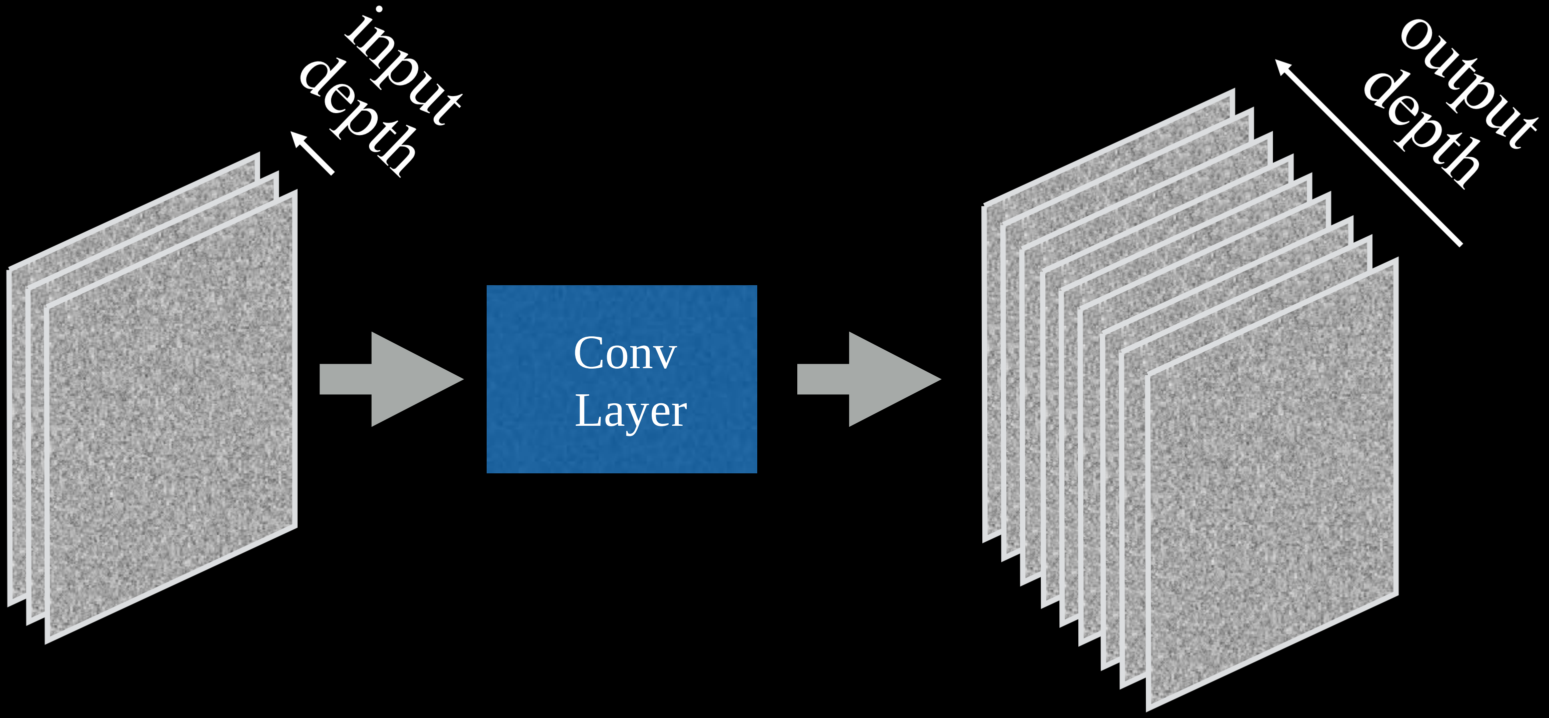
$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$



$$* \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} / 16 =$$



Many kernels yielding many features!



Color images
3 channels

Conv layer features
9 channels

- Convolution kernel is 4-D: For each output channel, kernel goes across input dimensions and channels
- Size = width x height x # input channels x # output channels

An example deep convolution network

- Input: 28x28 grayscale image
- Output: 10 classes. One output per class.

A Convolutional Net

- Let’s assume we have 28x28 grayscale images as input to our conv net. So we will input 28x28x1 samples into the net.

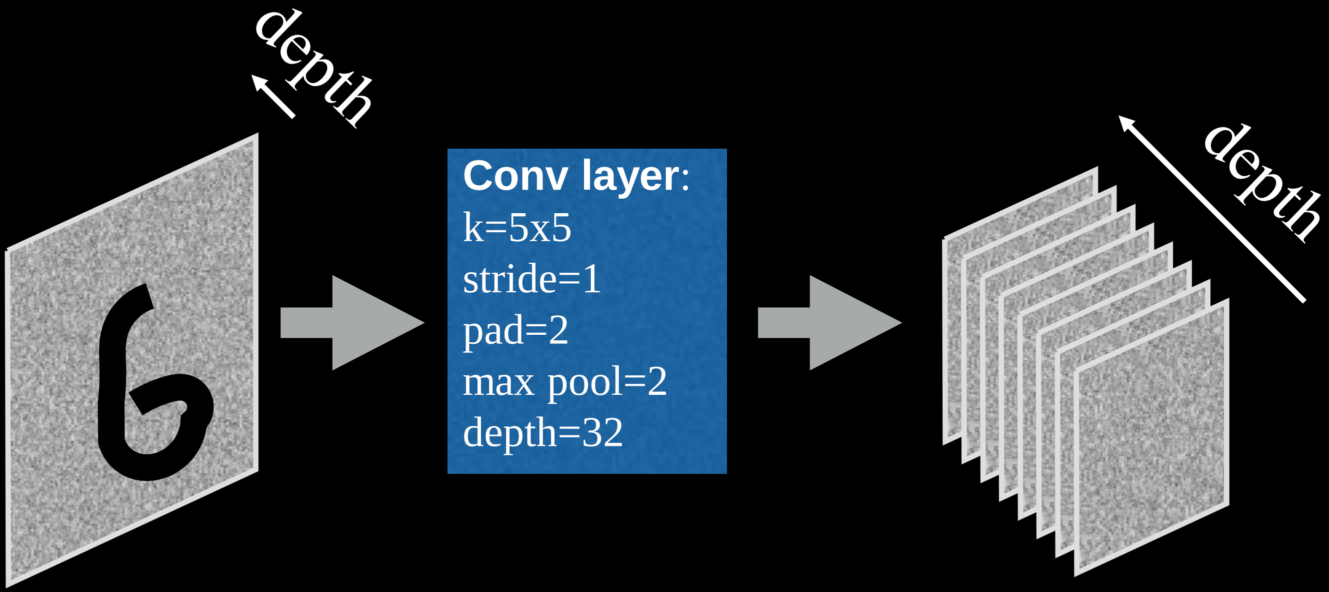
- Let’s fix our kernel size at 5x5 and, to make this simple, pad our images with zeros and use a stride = 1.

- Let’s use max pooling on the output, with a 2x2 pooling region and a stride of 2.

- Let’s extract 32 features after the first layer.

- So the output from this layer will be 14x14x32.

A Convolutional Net



Conv layer:
k=5x5
stride=1
pad=2
max pool=2
depth=32

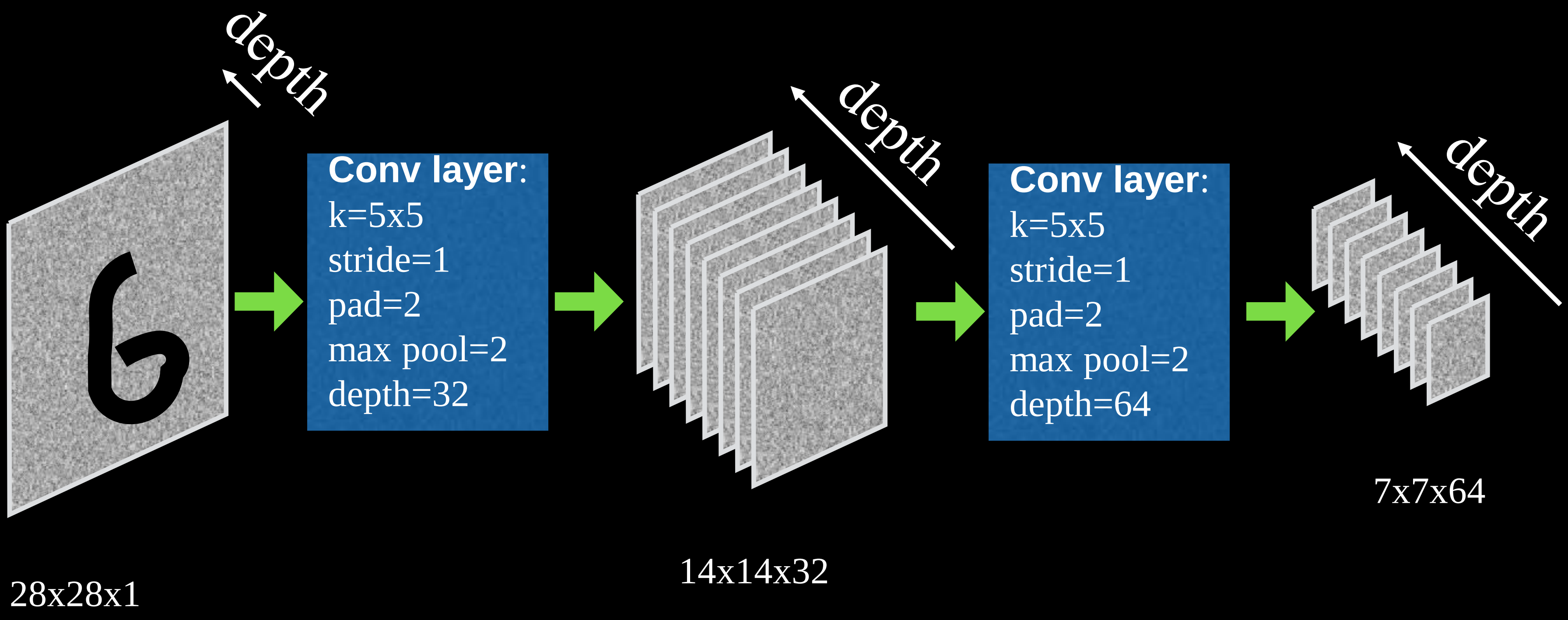
28x28x1

14x14x32

A Convolutional Net

- Now let’s make a second layer, also convolutional.
- Let’s fix our kernel size at 5x5, pad our images with zeros and use a stride = 1.
- Let’s use max pooling on the output again, with a 2x2 pooling region and a stride of 2.
- Let’s extract 64 features after the second layer.
- So the output from this layer will be 7x7x64.

A Convolutional Net



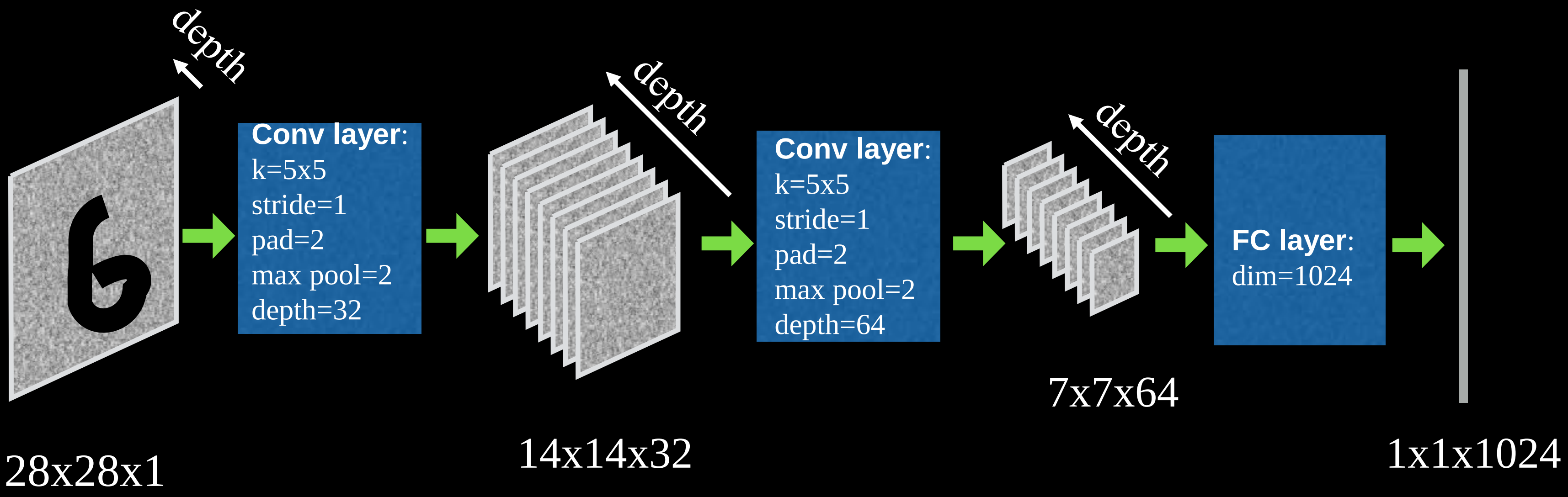
A Convolutional Net

- Our third layer will be a fully connected layer mapping our convolutional features to a 1024 dimensional feature space.

- This layer is just like any of the hidden layers you’ve seen before. It is a linear transformation followed by ReLU.

- So the output from this layer will be 1x1x1024.

A Convolutional Net

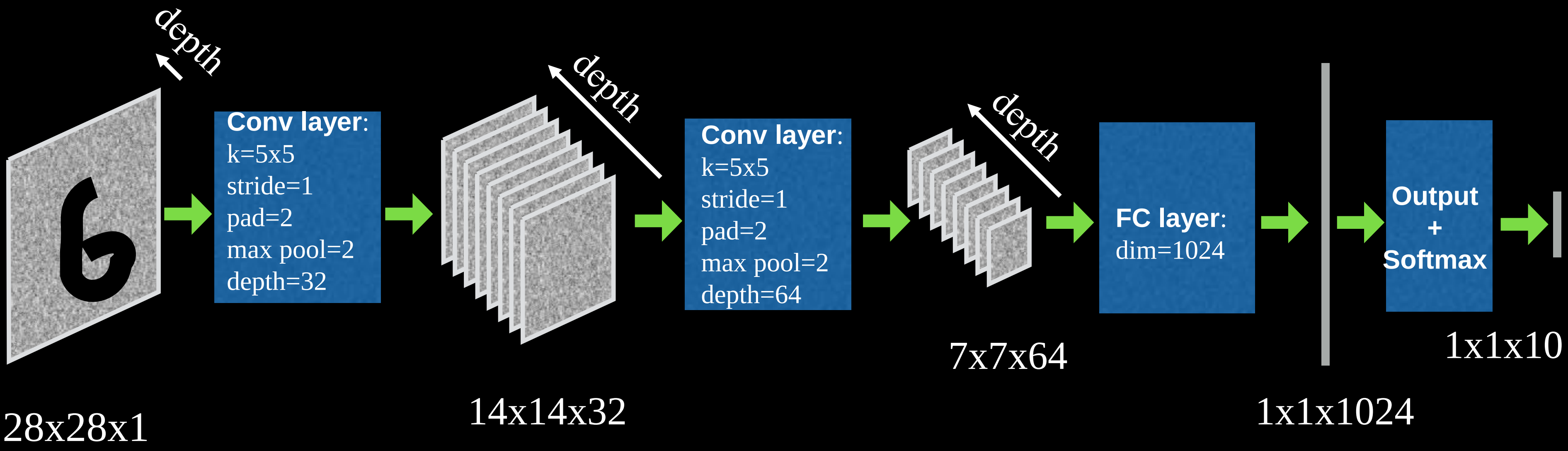


A Convolutional Net

- Finally, we'll map this feature space to a 10 class output space and use a softmax with a MLE/cross entropy loss function.

- And…we're done!

A Convolutional Net



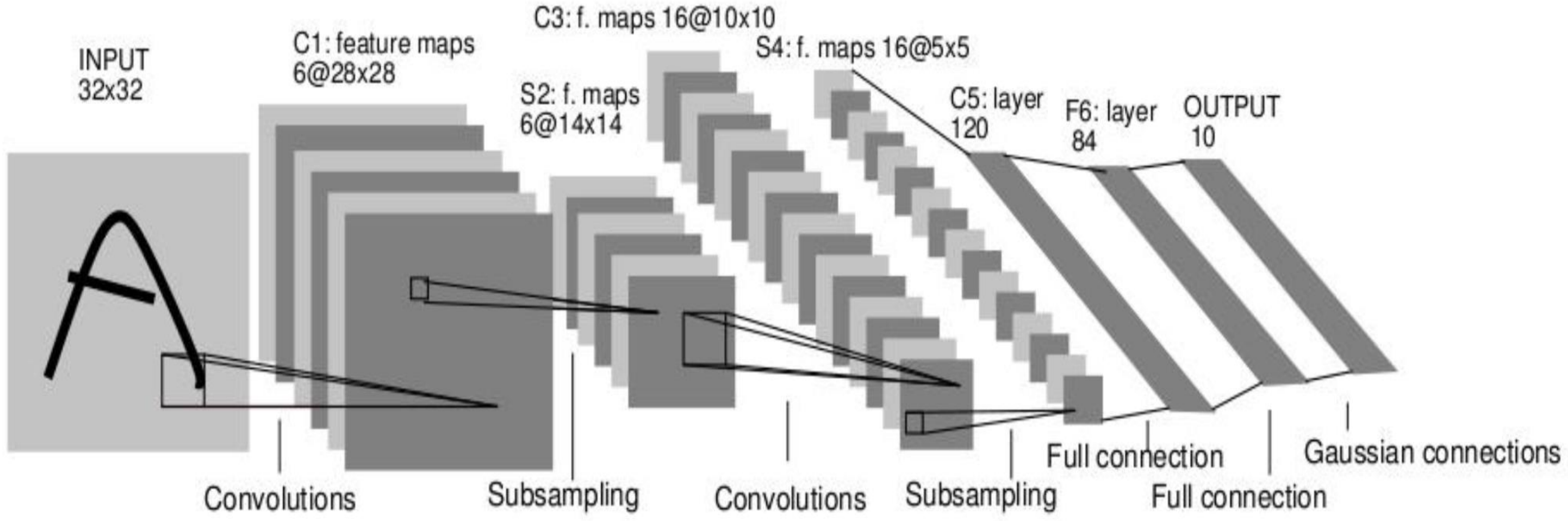
Parameters = (5x5x1x32+32) + (5x5x32x64+64) + (7x7x64x1024+1024) + (1024x10+10)

Major layers

- Convolutional layer
- Pooling layer
- Fully connected Layer
- Softmax

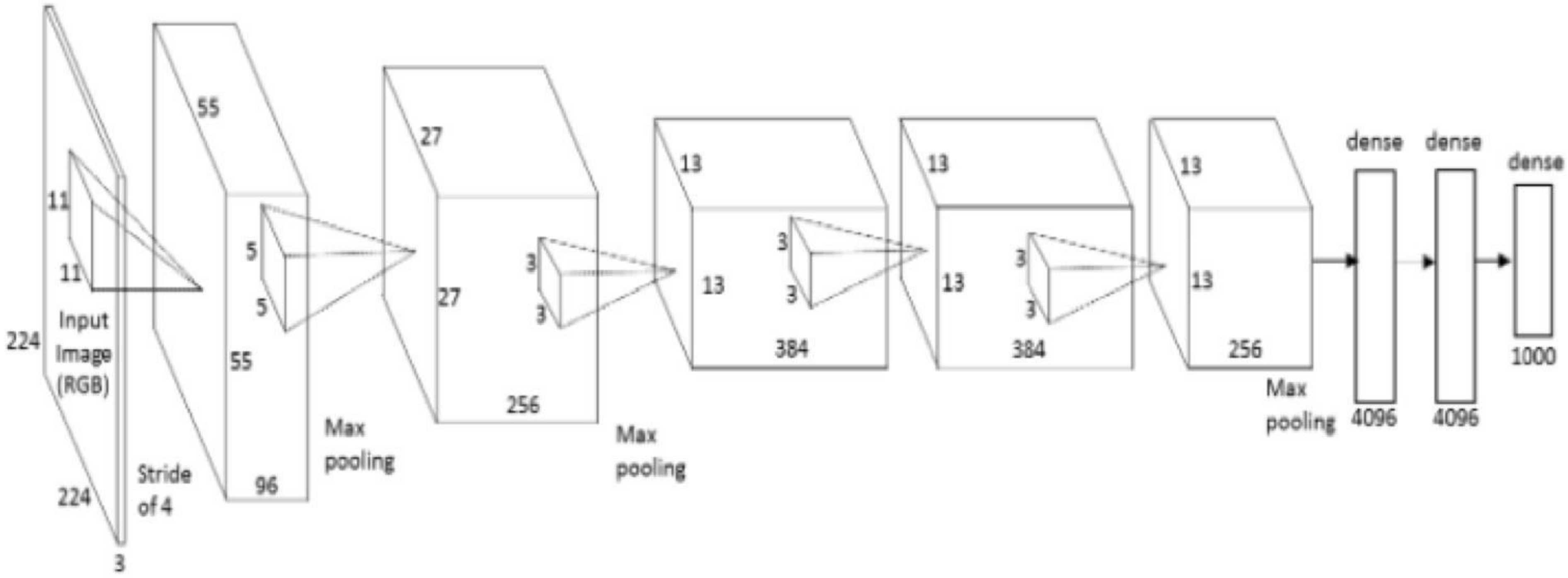
Some Famous Deep Nets and Data sets

LeNet 5



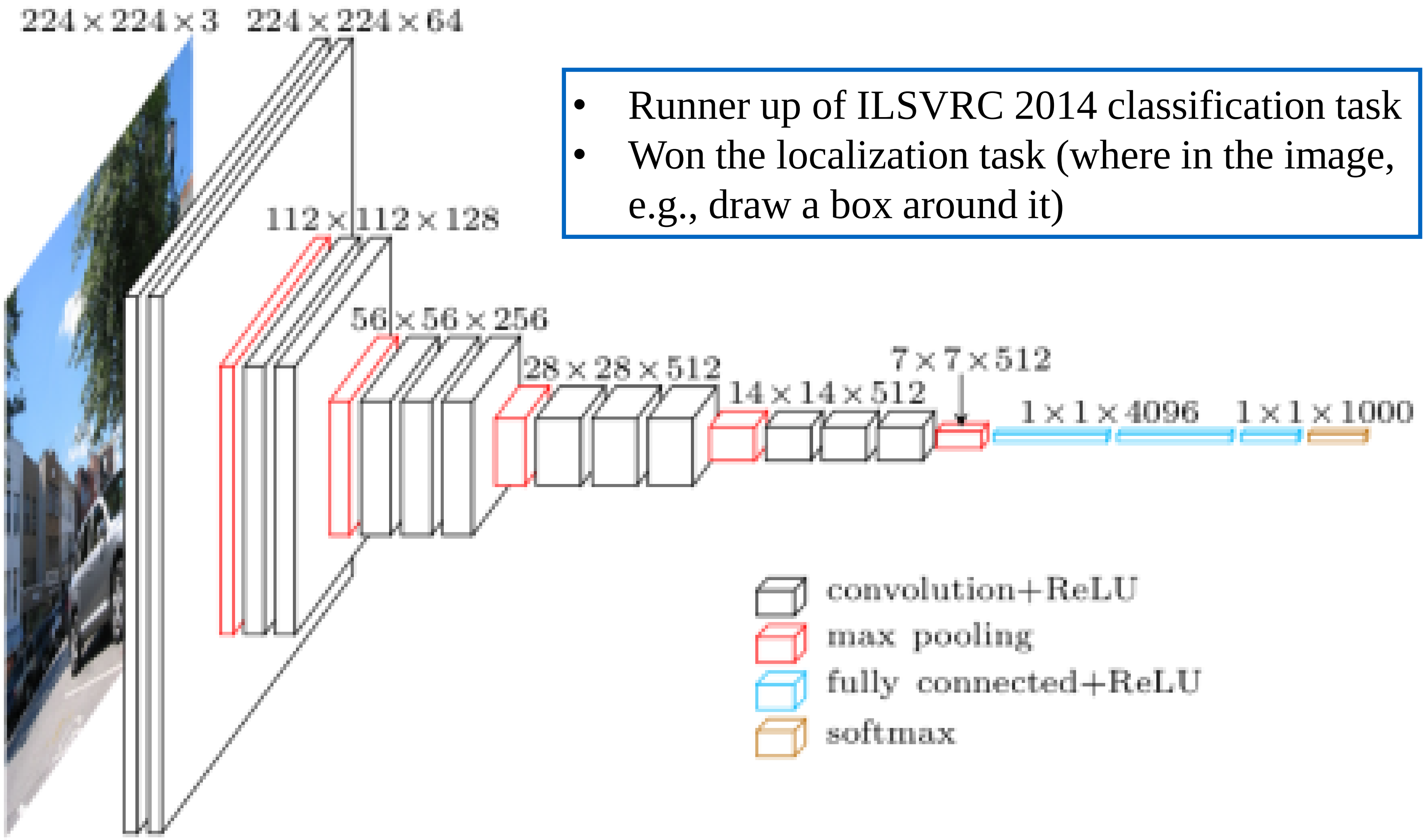
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner,
Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

AlexNet [Krizhevsky et al., 2012]



- Won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in September 2012
- The network achieved a top-5 error rate (rate of not finding the true label of a given image among its top 5 predictions) of 15.3%
- The next best result was 26.2%
- AlexNet used GPUs

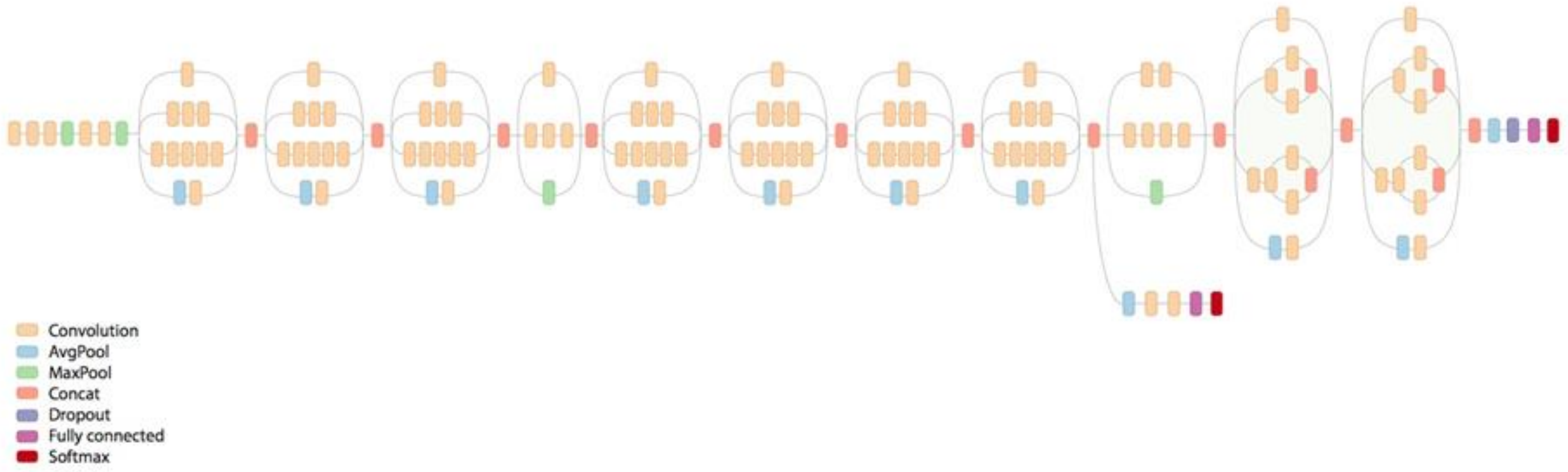
VGG16 [Simonyan and Zisserman, 2014]



- Runner up of ILSVRC 2014 classification task
- Won the localization task (where in the image, e.g., draw a box around it)

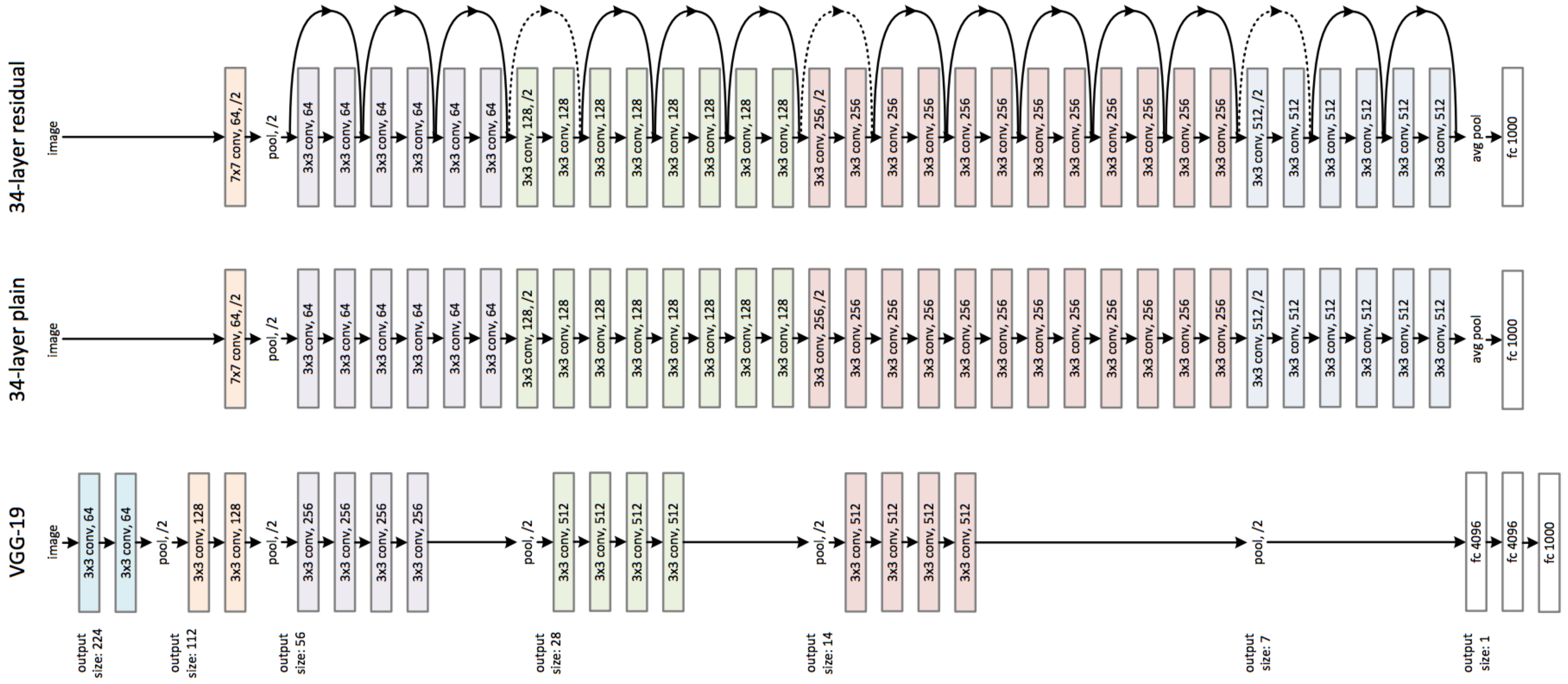
GoogLeNet [Szegedy et al., 2014]

Won the ILSVRC 2014 classification task



Another view of GoogLeNet's architecture.

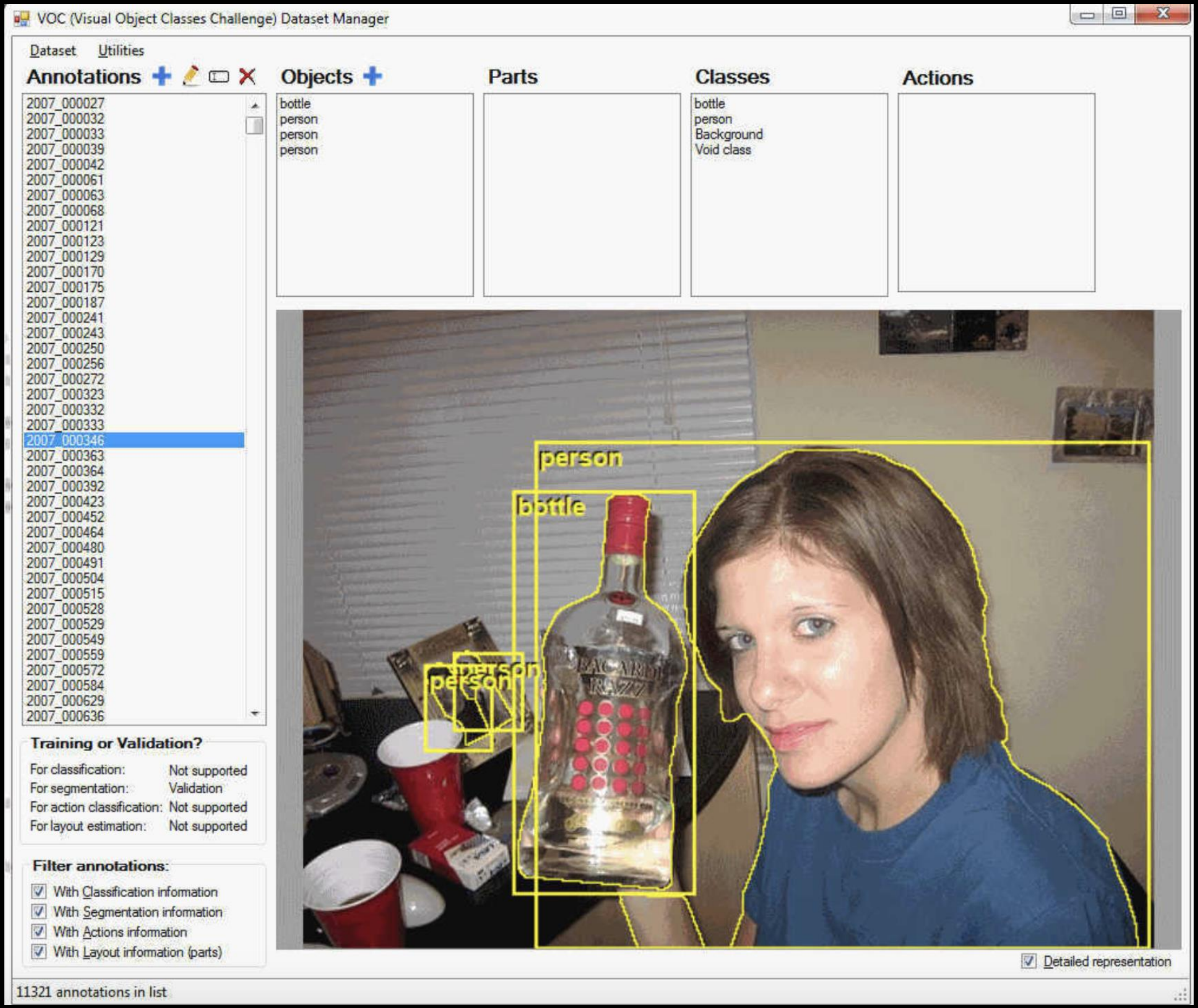
ResNet [He et al., 2016]



"Identity shortcut connection" skips one or more layers. A larger number of layers while mitigating the "vanishing gradient" problem (the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitively small).

PASCAL Visual Object Classes (VOC)

Provides standardized image data sets for object class recognition.



2007_000027
2007_000032
2007_000033
2007_000039
2007_000042
2007_000061
2007_000063
2007_000068
2007_000121
2007_000123
2007_000129
2007_000170
2007_000175
2007_000187
2007_000241
2007_000243
2007_000250
2007_000256
2007_000272
2007_000323
2007_000332
2007_000333
2007_000346
2007_000363
2007_000364
2007_000392
2007_000423
2007_000452
2007_000464
2007_000480
2007_000491
2007_000504
2007_000515
2007_000528
2007_000529
2007_000549
2007_000559
2007_000572
2007_000584
2007_000629
2007_000636

Objects: bottle, person, person, person

Classes: bottle, person, Background, Void class

Training or Validation?

For classification:	Not supported
For segmentation:	Validation
For action classification:	Not supported
For layout estimation:	Not supported

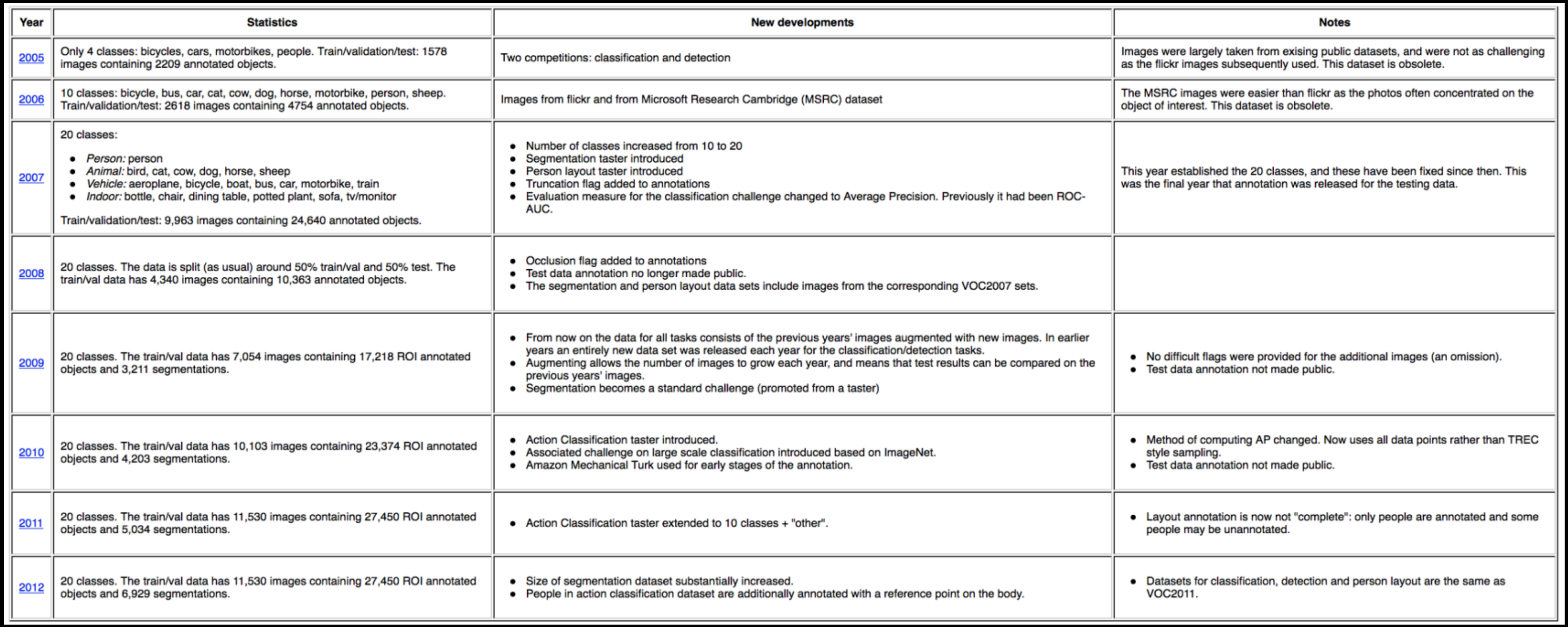
Filter annotations:

- [✓] With Classification information
- [✓] With Segmentation information
- [✓] With Actions information
- [✓] With Layout information (parts)

[✓] Detailed representation

11321 annotations in list

PASCAL VOC



Year	Statistics	New developments	Notes
2005	Only 4 classes: bicycles, cars, motorbikes, people. Train/validation/test: 1578 images containing 2209 annotated objects.	Two competitions: classification and detection	Images were largely taken from exising public datasets, and were not as challenging as the flickr images subsequently used. This dataset is obsolete.
2006	10 classes: bicycle, bus, car, cat, cow, dog, horse, motorbike, person, sheep. Train/validation/test: 2618 images containing 4754 annotated objects.	Images from flickr and from Microsoft Research Cambridge (MSRC) dataset	The MSRC images were easier than flickr as the photos often concentrated on the object of interest. This dataset is obsolete.
2007	20 classes: • *Person:* person • *Animal:* bird, cat, cow, dog, horse, sheep • *Vehicle:* aeroplane, bicycle, boat, bus, car, motorbike, train • *Indoor:* bottle, chair, dining table, potted plant, sofa, tv/monitor Train/validation/test: 9,963 images containing 24,640 annotated objects.	• Number of classes increased from 10 to 20 • Segmentation taster introduced • Person layout taster introduced • Truncation flag added to annotations • Evaluation measure for the classification challenge changed to Average Precision. Previously it had been ROC-AUC.	This year established the 20 classes, and these have been fixed since then. This was the final year that annotation was released for the testing data.
2008	20 classes. The data is split (as usual) around 50% train/val and 50% test. The train/val data has 4,340 images containing 10,363 annotated objects.	• Occlusion flag added to annotations • Test data annotation no longer made public. • The segmentation and person layout data sets include images from the corresponding VOC2007 sets.	
2009	20 classes. The train/val data has 7,054 images containing 17,218 ROI annotated objects and 3,211 segmentations.	• From now on the data for all tasks consists of the previous years' images augmented with new images. In earlier years an entirely new data set was released each year for the classification/detection tasks. • Augmenting allows the number of images to grow each year, and means that test results can be compared on the previous years' images. • Segmentation becomes a standard challenge (promoted from a taster)	• No difficult flags were provided for the additional images (an omission). • Test data annotation not made public.
2010	20 classes. The train/val data has 10,103 images containing 23,374 ROI annotated objects and 4,203 segmentations.	• Action Classification taster introduced. • Associated challenge on large scale classification introduced based on ImageNet. • Amazon Mechanical Turk used for early stages of the annotation.	• Method of computing AP changed. Now uses all data points rather than TREC style sampling. • Test data annotation not made public.
2011	20 classes. The train/val data has 11,530 images containing 27,450 ROI annotated objects and 5,034 segmentations.	• Action Classification taster extended to 10 classes + "other".	• Layout annotation is now not "complete": only people are annotated and some people may be unannotated.
2012	20 classes. The train/val data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations.	• Size of segmentation dataset substantially increased. • People in action classification dataset are additionally annotated with a reference point on the body.	• Datasets for classification, detection and person layout are the same as VOC2011.

20 categories in 11,530 images with 27,450 ROIs and 6,929 segmentations

[Everingham et al. 2005—2012]

Object Detection: PASCAL VOC mean Average Precision (mAP)

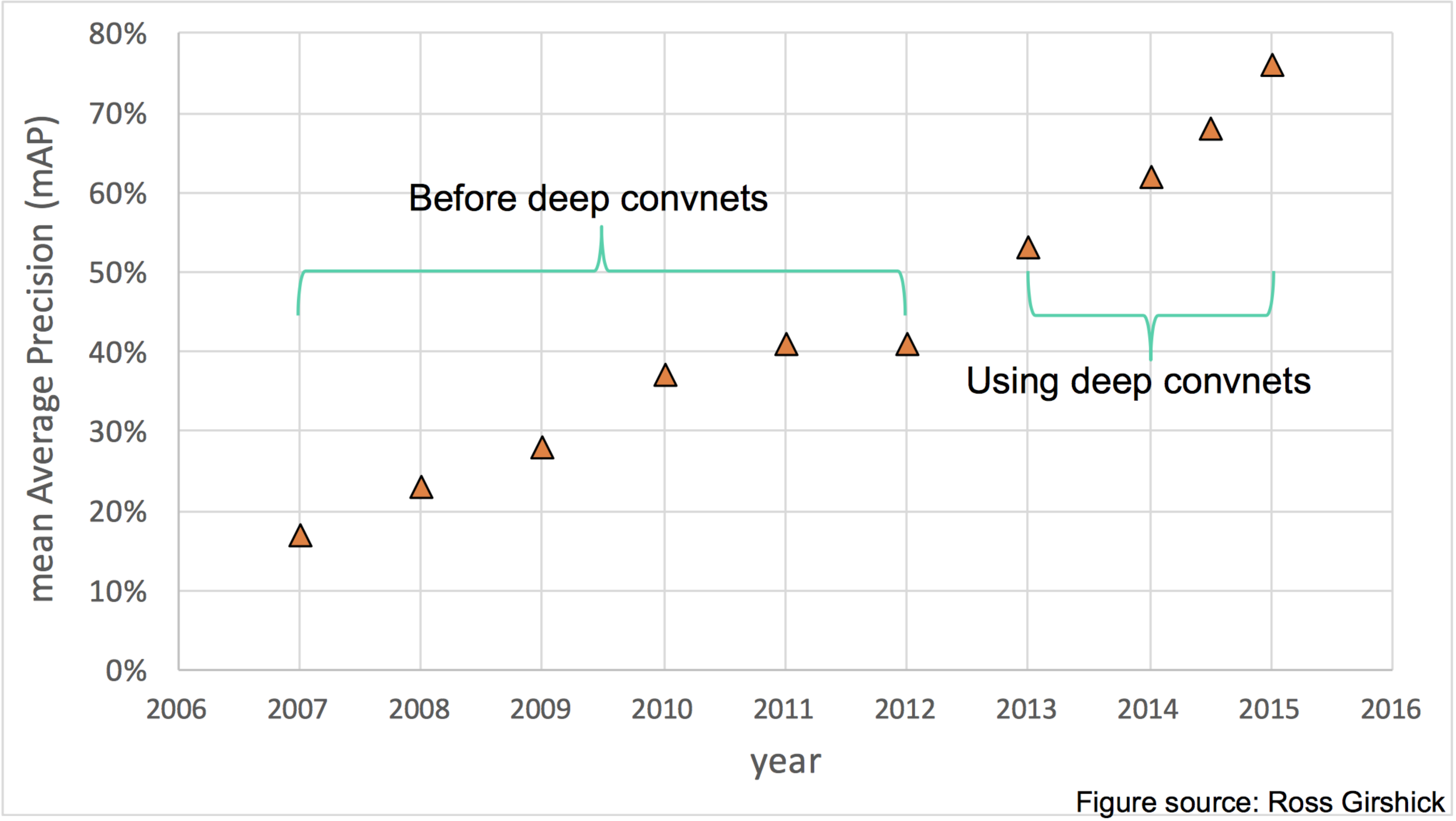


Figure source: Ross Girshick

ImageNet

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
  - Classification task.
  - Localization

[Deng et al. 2009]

20,000+ categories x ~1000 instances = 14,000,000+ images

Classification: ImageNet Challenge top-5 error

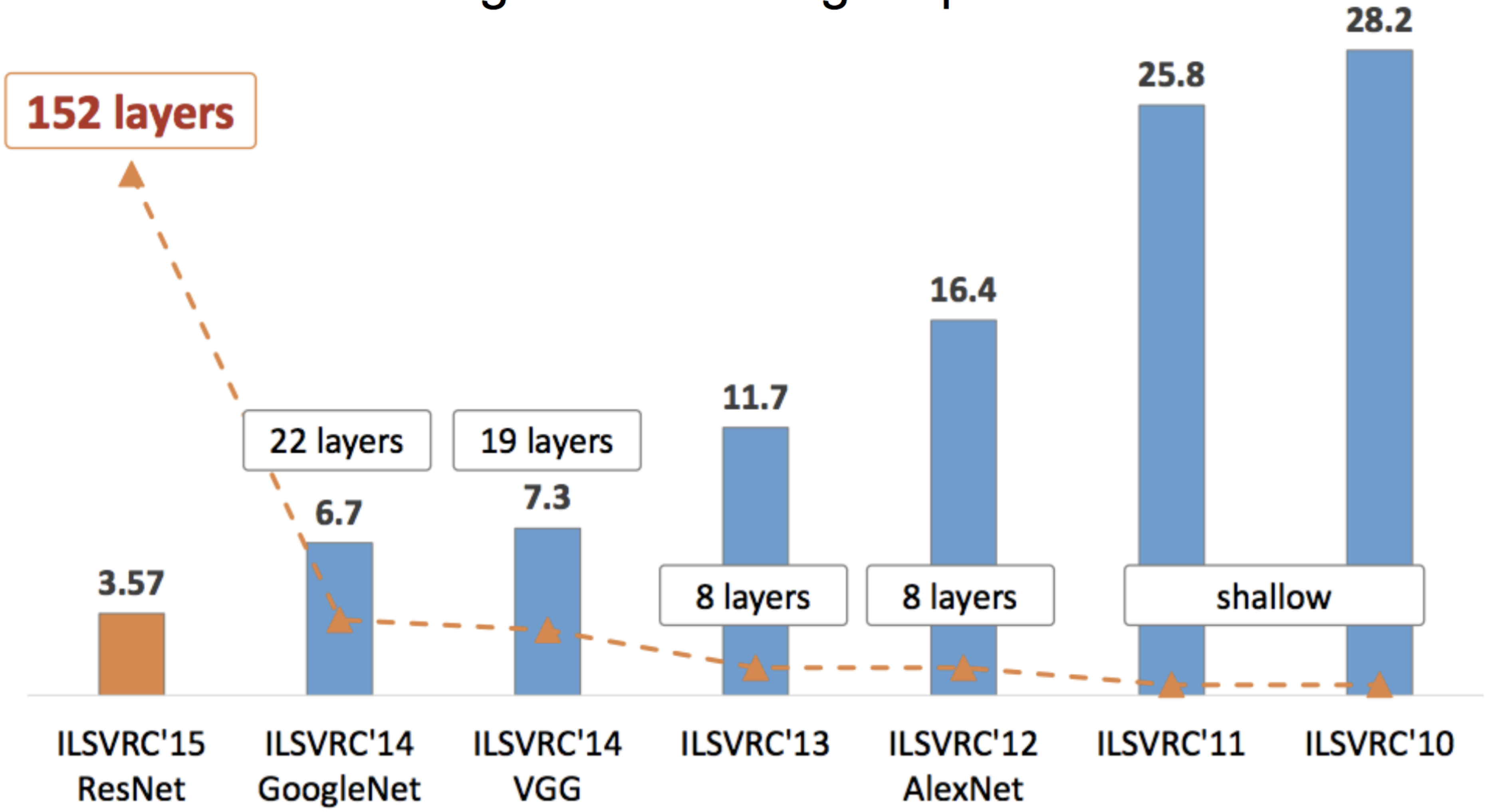


Figure source: [Kaiming He]

Summary

- Artificial neural networks have been around for over 60 years
- Universal Approximation Theorem
  - If we have enough hidden units, then we can approximate any function
  - Deeper networks (more hidden layers) are more efficient than a single hidden layer
- Convolutional neural networks share weights
  - Significantly less weights than a fully connected network
- Training a neural network is a nonlinear optimization problem
  - Never train on your validation set!

Different Classes
Network Surgery
+
Fine Tuning

Cats vs. Dogs

Data from [Kaggle.com]

vs.

2 categories x 11,000 instances = 22,000 images

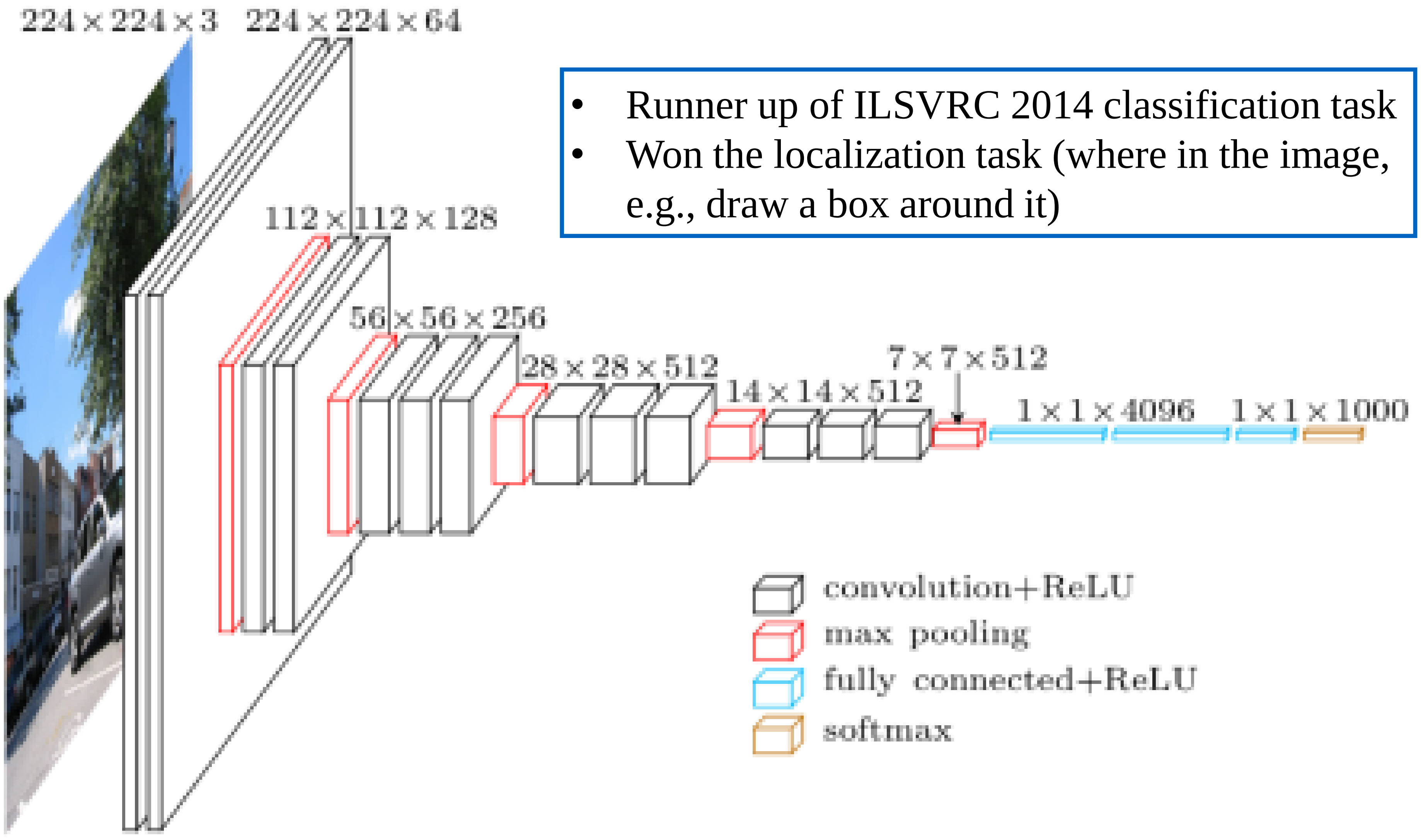
We usually don’t have enough data!

…but here is a FANTASTIC trick!

Domain Transfer

- Train a **deep** ConvNet using lots of data on a large image classification problem like ImageNet.
- Save the weights.
- Chop off the output layer (or final layers) of this ConvNet.
- Replace the output layer (or final layers) with one that fits your problem.
- Freeze the weights in the old layers and train on your data, allowing the weights in the new layer to settle.
- Unfreeze the whole network and train.

VGG16 [Simonyan and Zisserman, 2014]


$224 \times 224 \times 3$
$224 \times 224 \times 64$
$112 \times 112 \times 128$
$56 \times 56 \times 256$
$28 \times 28 \times 512$
$14 \times 14 \times 512$
$7 \times 7 \times 512$
$1 \times 1 \times 4096$
$1 \times 1 \times 1000$
convolution+ReLU
max pooling
fully connected+ReLU
softmax

- Runner up of ILSVRC 2014 classification task
- Won the localization task (where in the image, e.g., draw a box around it)

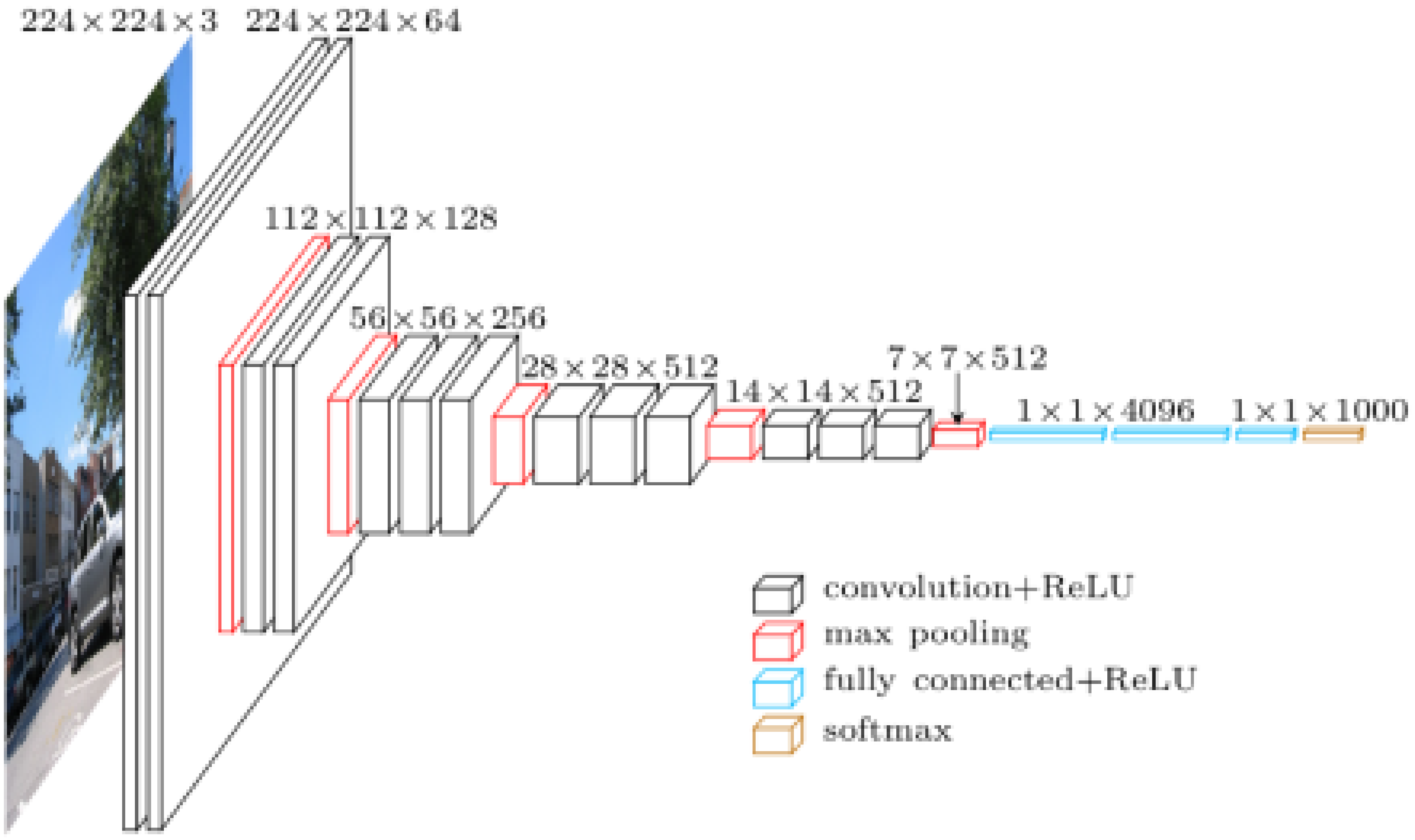
ImageNet

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
    - Classification task.
    - Localization

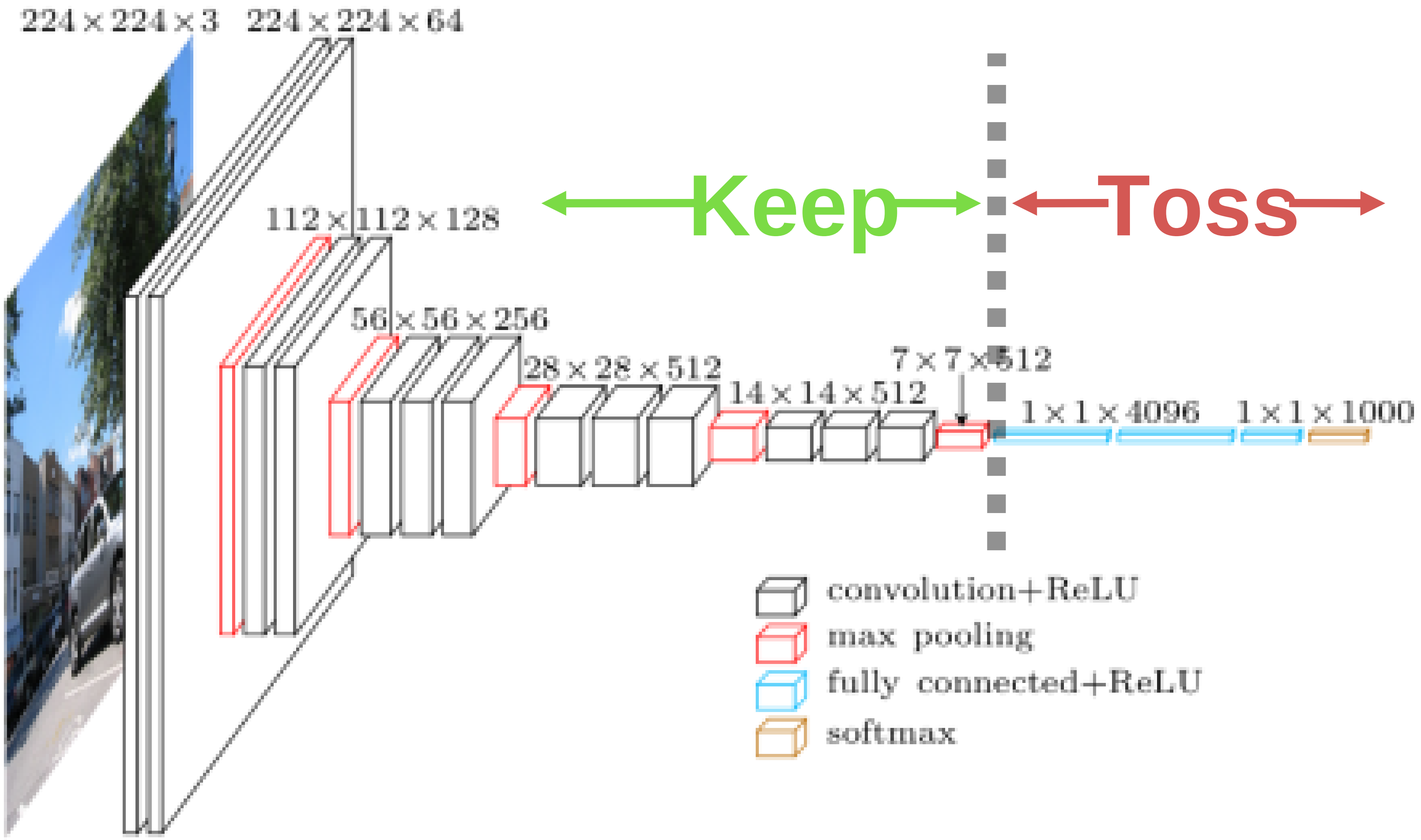
[Deng et al. 2009]

20,000+ categories x ~1000 instances = 14,000,000+ images

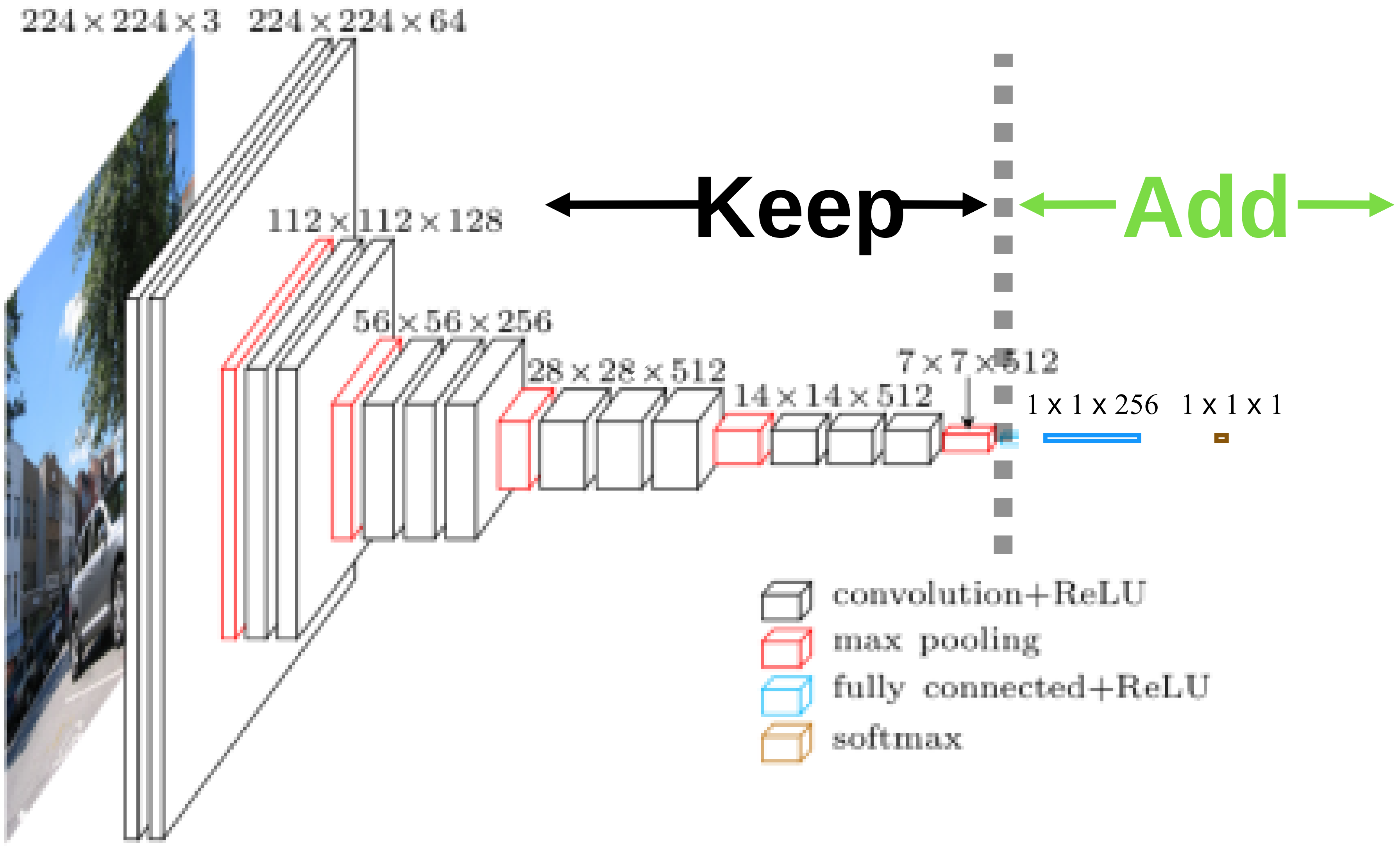
Domain Transfer + Fine-Tuning



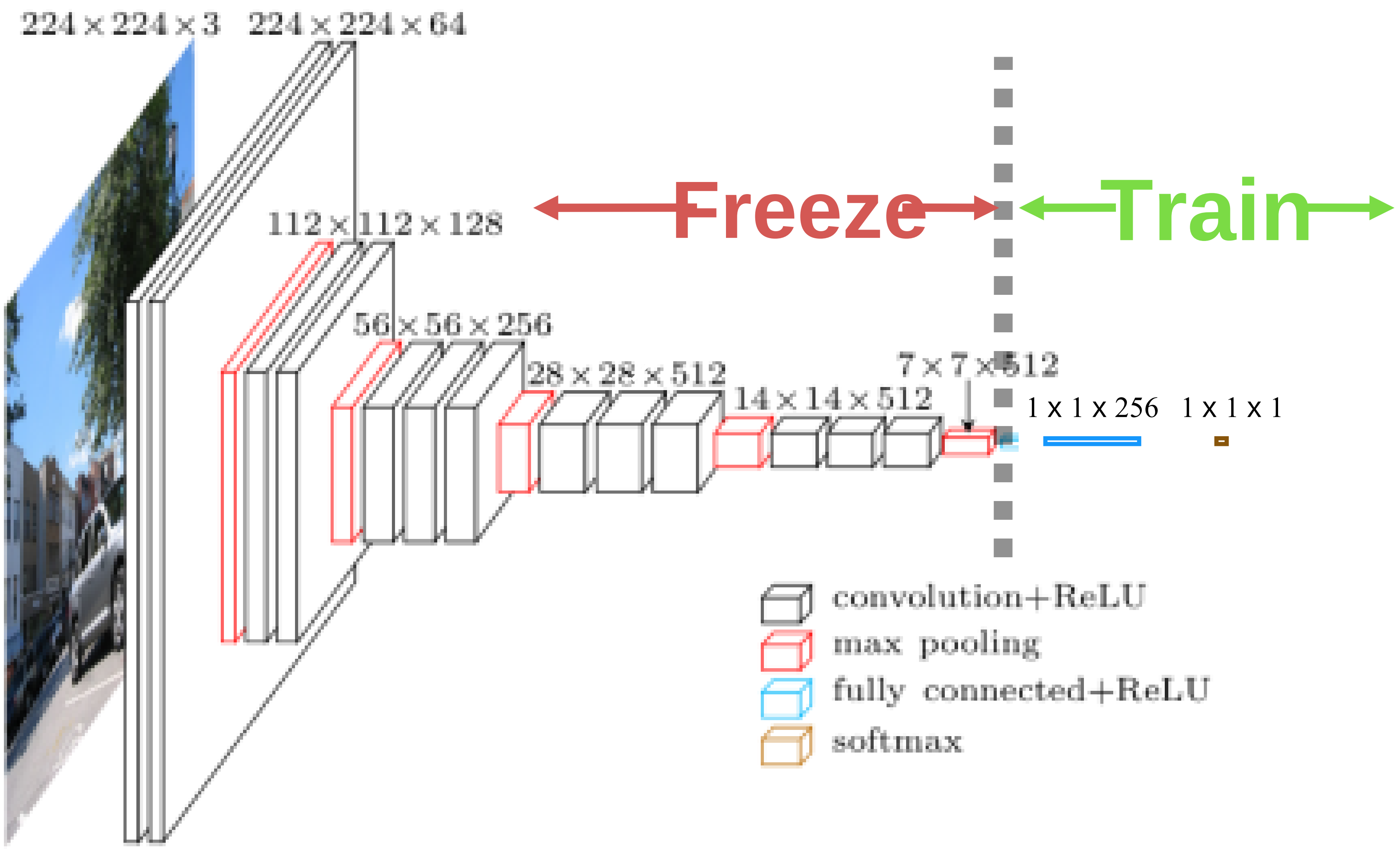
Network Surgery



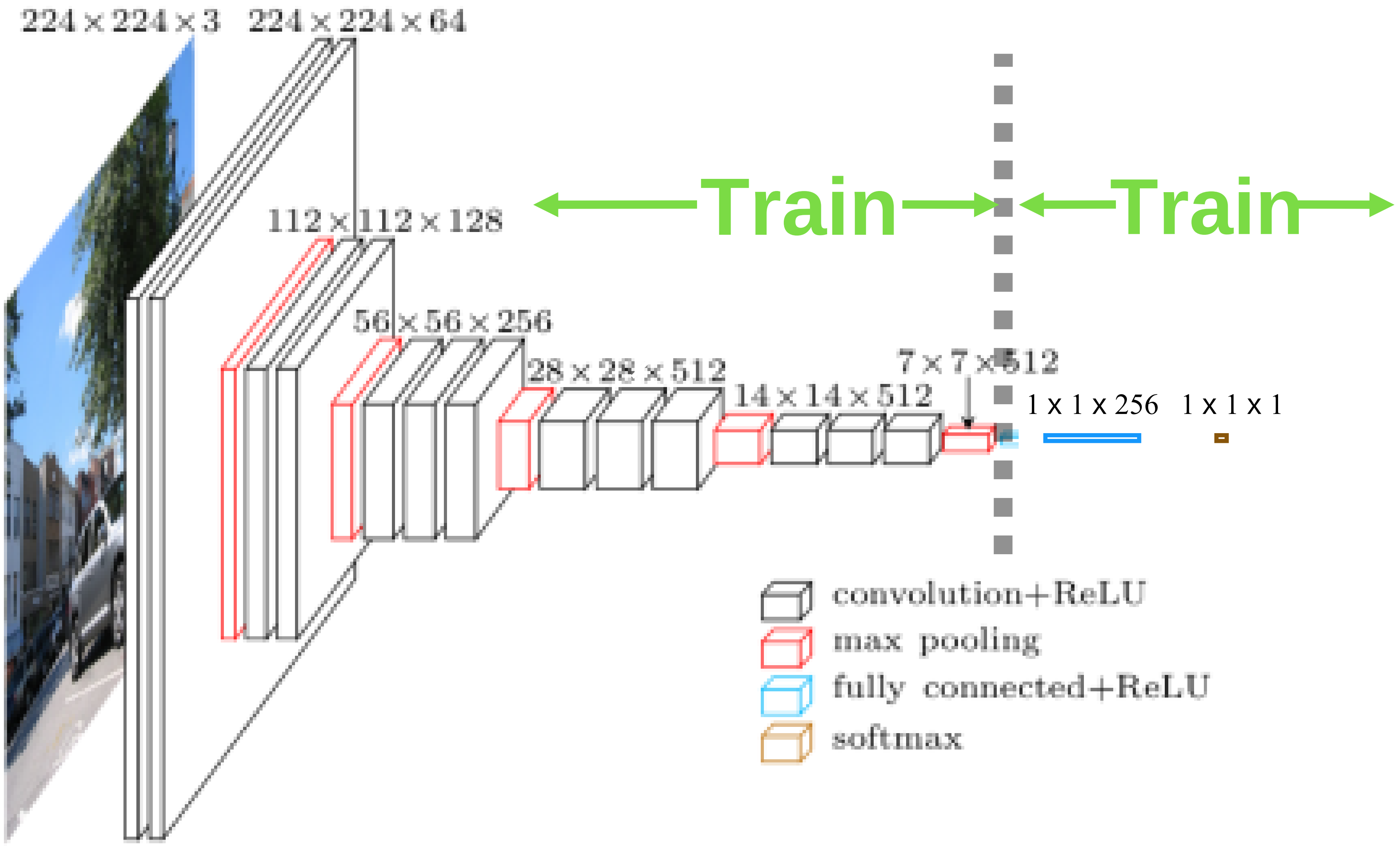
Network Surgery



First, train the top layers



Then, train the whole network



Next Lecture

- Color