CSC321 Neural Networks and Machine Learning

Lecture 6

February 12, 2020

Agenda

- Projects 1 and 2
- Project 3 data collection
- Midterm
- Convolutional Neural Networks

Announcements

- Project 1 is released
  - If you received more than 4 points off due to your answers being cut off, please submit a remark request
  - Remarks are for possible TA mistakes
  - Feedback on common issues available on Quercus
- Project 2 is due tomorrow
  - FAQ on Piazza
  - Pouria has office hours later today

Project 3 Data Collection

For project 3, we'll be collecting our own data as a class.

Right Shoe	Left Shoe
	
	

If you submit your data by **Monday Feb 17th, 9pm** we will award you 3 extra grace tokens. (Why? So that we have more time to develop a fun assignment with this data set!)

Midterm

Midterm Logistics

The midterm will take place after reading week on **Feb 26th**

- 50 minutes midterm during the first half of the hour
  - 9:10am - 10:00am (LEC0101)
  - 11:10am - 12:00pm (LEC0101)
- We'll have a short lecture after
  - 10:20am-11am (LEC0101)
  - 12:20pm-1pm (LEC0102)

Midterm Coverage

- Weeks 1-6 materials, including convolutional neural networks forward pass
- Tutorials 2-6, Homeworks 1-3, Projects 1-2
- No aids will be permitted
- You won't need a calculator
- You may be asked to work with code, for example reason about issues with code that we give you
  - You don't need to memorize the numpy or PyTorch API
  - You do need to recognize (for example) a that trying to multiply a 3×4 matrix and a 5×2 matrix will fail

How to study?

- Previous midterms (available on the course website)
- Midterms from related courses (to see what multiple choice questions might look like)
- Review homework questions
- Review projects 1 and 2 (especially if you worked with a partner)

If you post study resources to share on Piazza (notes, solutions to posted problems, etc), we'll pin it.

Office Hours

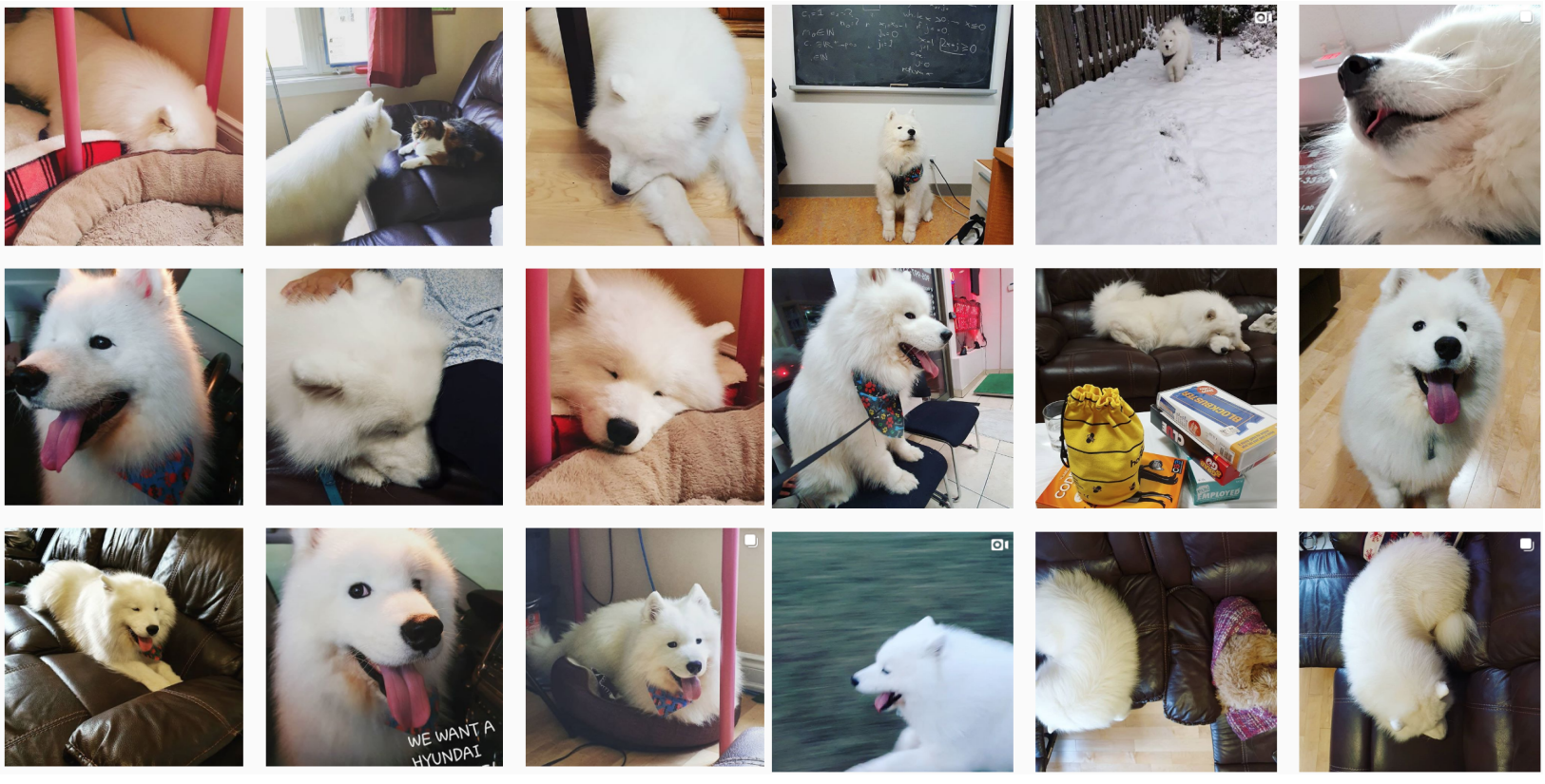
- No office hours during reading week
- Pouria will have an online office hour during reading week (to be announced)
- Lisa has office hours Monday Feb 24th
- Tuesday's tutorial time will be turned into office hours
- CANCELLED Pouria's office hours Wednesday Feb 26th

Questions?

Working with Images

Computer vision is hard

- Object change in pose, size, viewpoint, background
- Some objects are hidden behind others: *occlusion*

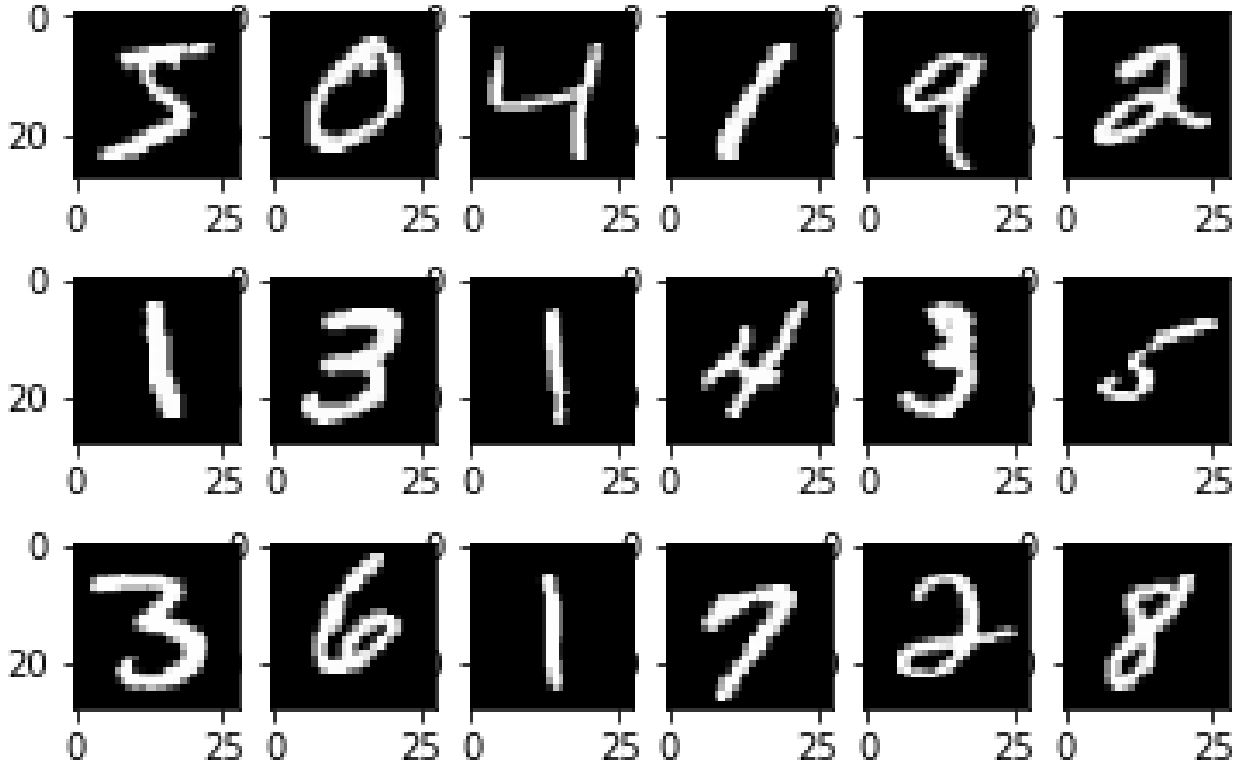


Computer vision is really hard

How can you “hard code” an algorithm that still recognizes that this is a cat?

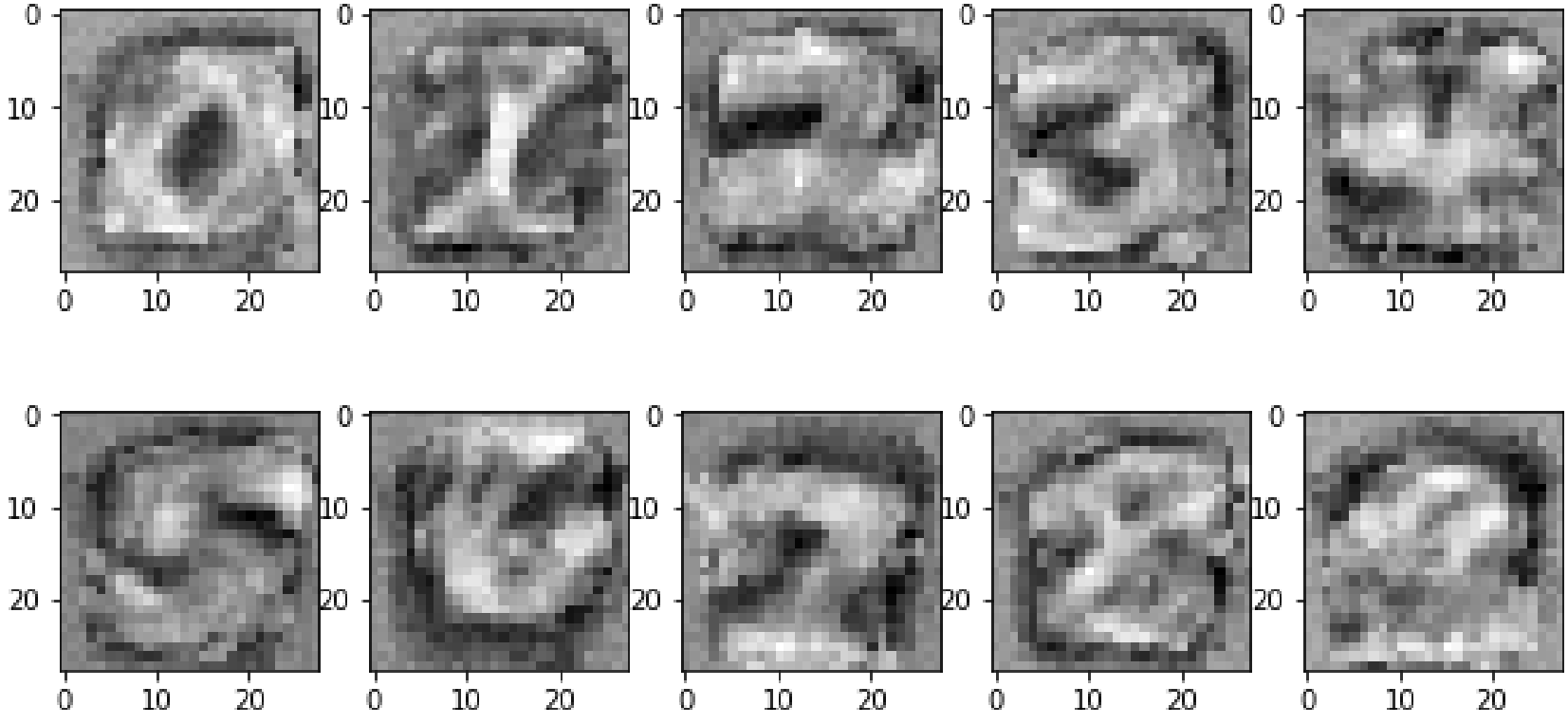
Working with Small Images

In tutorials 4 and 6, we worked with small images: 28×28 pixels, black and white.

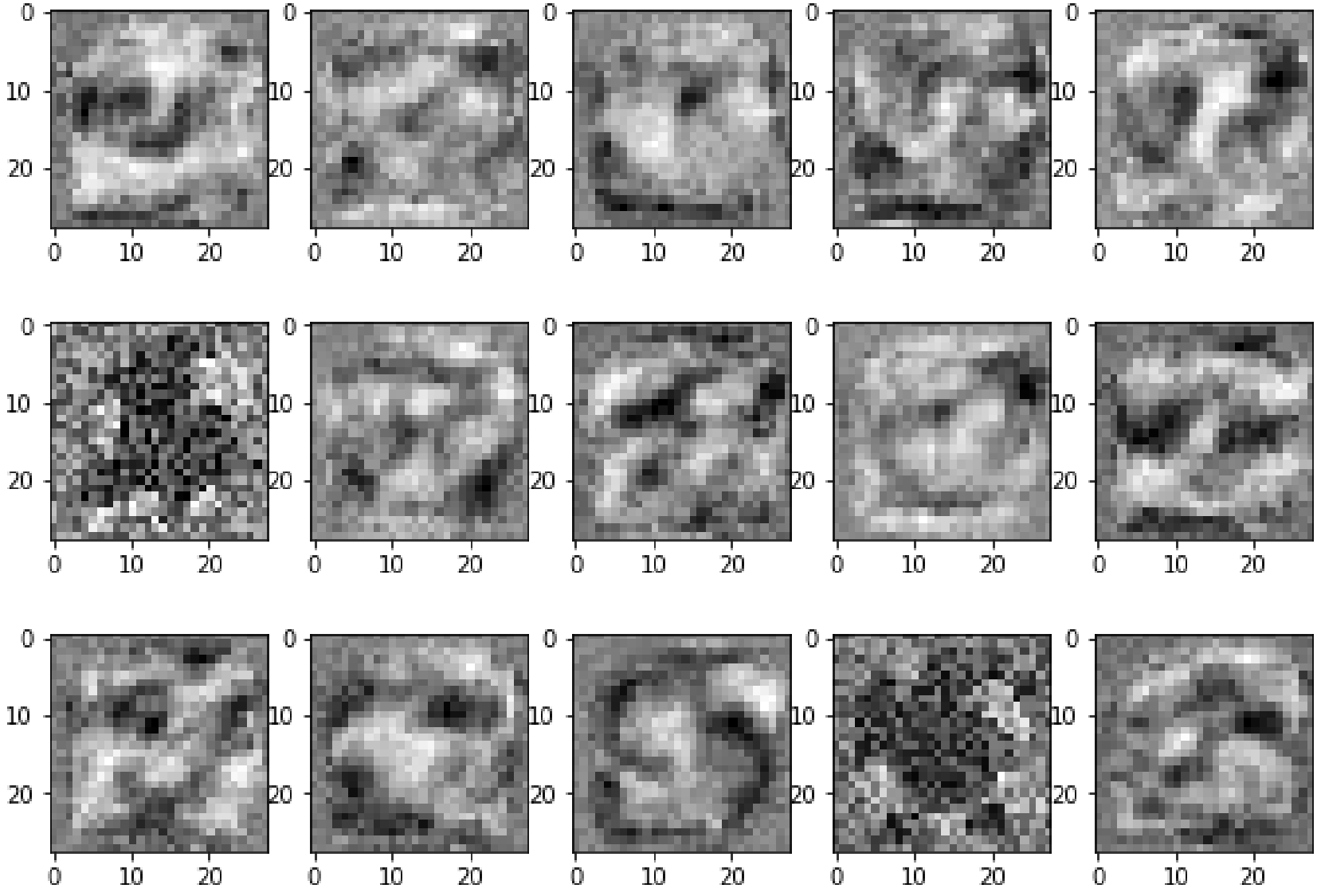


How do our models work?

Notebook Demo - Logistic Regression Weights



Notebook Demo - MLP Weights (first layer)



Working with Large Images

- Suppose you have an image that is 200 pixels x 200 pixels
- There are 500 units in the first hidden layer

Q: How many **parameters** will there be in the first layer?

Working with Large Images

- Suppose you have an image that is 200 pixels x 200 pixels
- There are 500 units in the first hidden layer

Q: How many **parameters** will there be in the first layer?

A: $200 \times 200 \times 500 + 500 =$ over 20 million!

Working with Large Images

- Suppose you have an image that is 200 pixels x 200 pixels
- There are 500 units in the first hidden layer

Q: How many **parameters** will there be in the first layer?

A: $200 \times 200 \times 500 + 500 =$ over 20 million!

Q: Why might using a fully connected layer be problematic?

Working with Large Images

- Suppose you have an image that is 200 pixels x 200 pixels
- There are 500 units in the first hidden layer

Q: How many **parameters** will there be in the first layer?

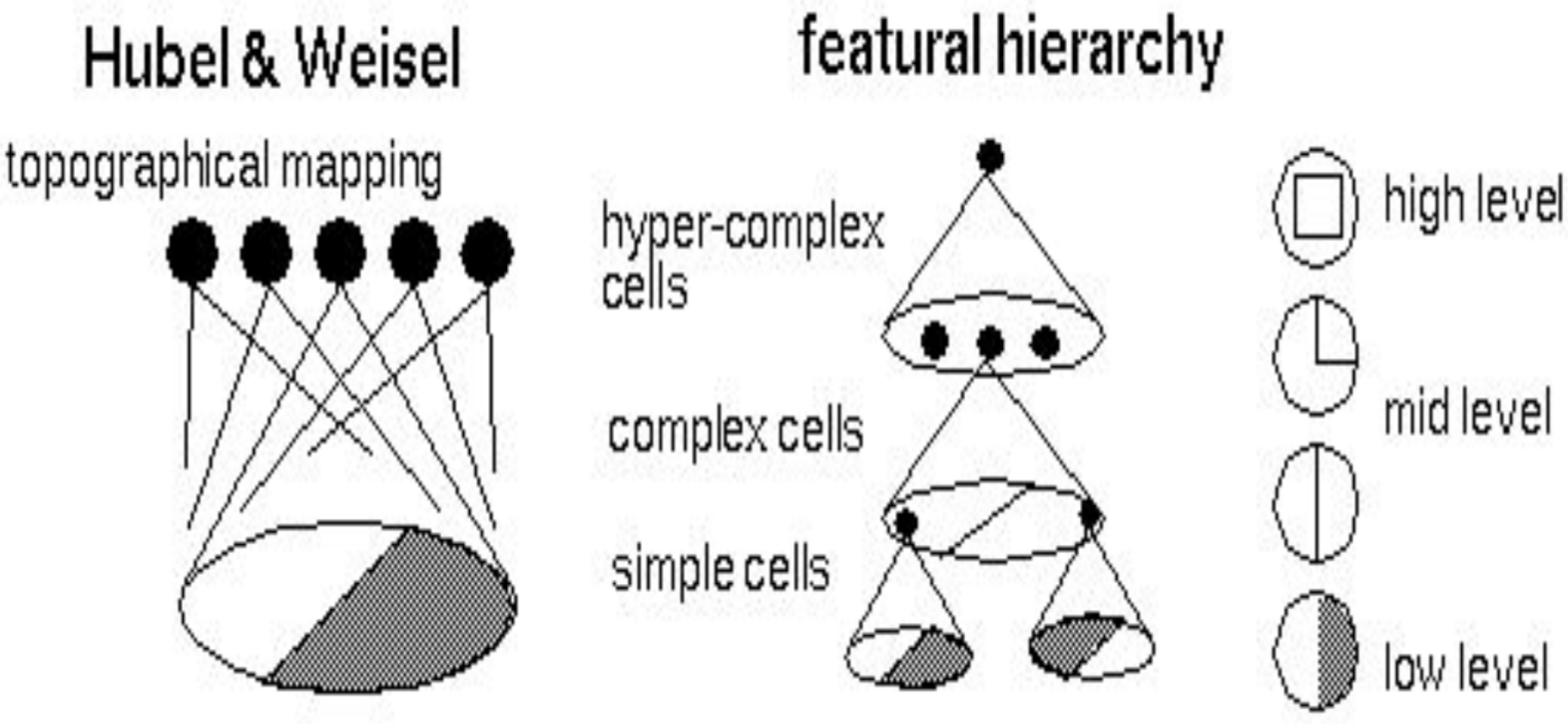
A: $200 \times 200 \times 500 + 500 =$ over 20 million!

Q: Why might using a fully connected layer be problematic?

- computing predictions (forward pass) will take a long time
- large number of weights requires a lot of training data to avoid overfitting
- small shift in image can result in large change in prediction

Convolutions

Biological Influence



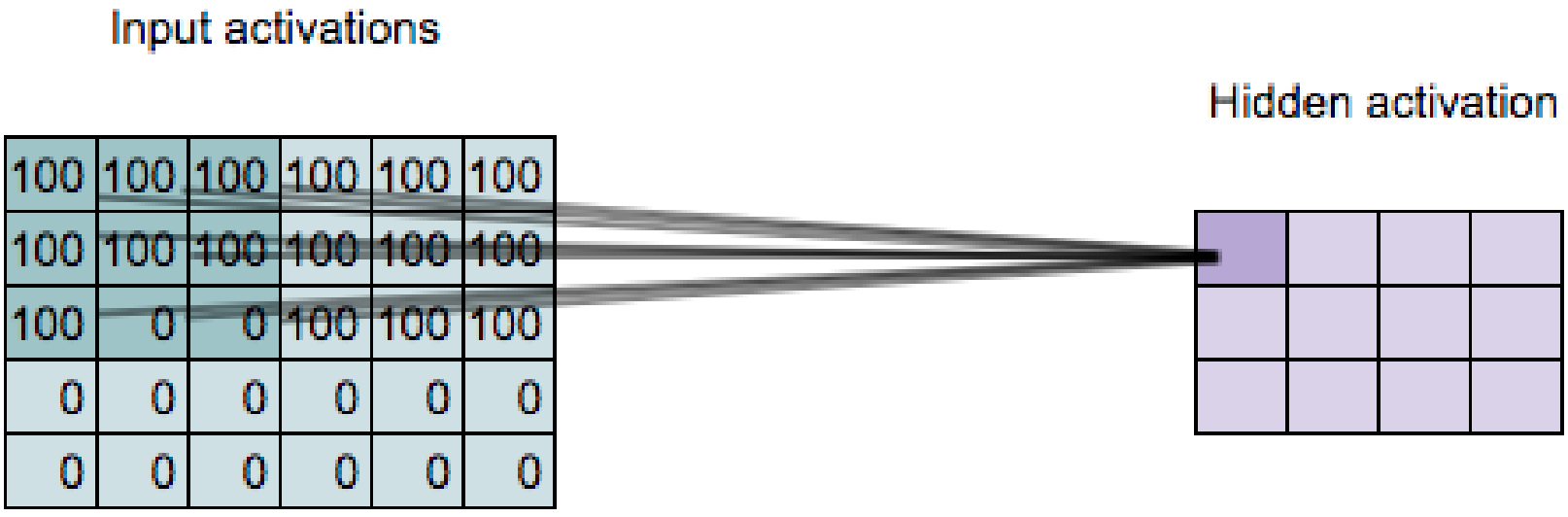
There is evidence that biological neurons in the visual cortex have *locally-connected* connections

Convolutional Neural Network

Ideas:

- **Locally-connected layers**: compute *local* features based on small regions of the image
  - Examples of *features*:
    - a horizontal edge in an area
    - a vertical edge in an area
    - a blob (no edges) in the area
    - a circular shape in the area
- **Weight-sharing**: detect the *same* local features across the entire image

Locally Connected Layers



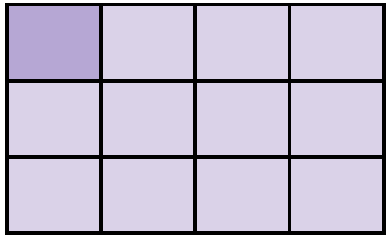
Each hidden unit connects to a small region of the input (in this case a 3×3 region)

Locally Connected Layers

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Hidden activation



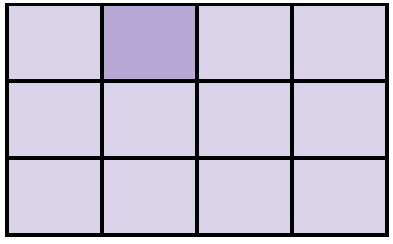
(Remove lines for readability)

Locally Connected Layers

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Hidden activation



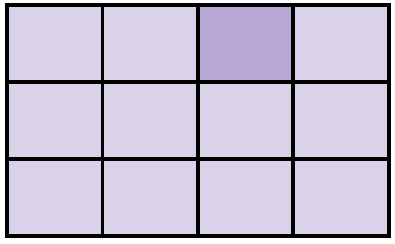
Hidden unit geometry has a 2D geometry consistent with the input

Locally Connected Layers

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Hidden activation

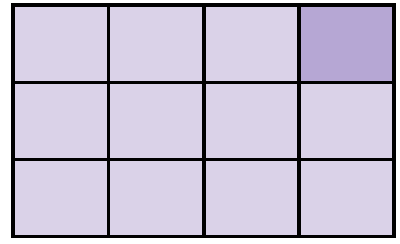


Locally Connected Layers

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Hidden activation

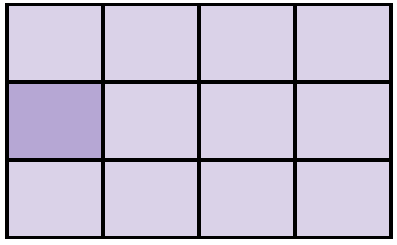


Locally Connected Layers

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Hidden activation

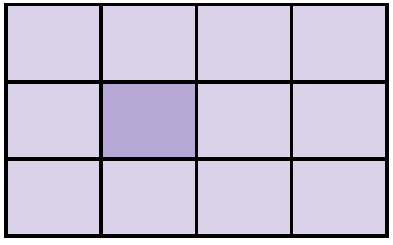


Locally Connected Layers

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Hidden activation

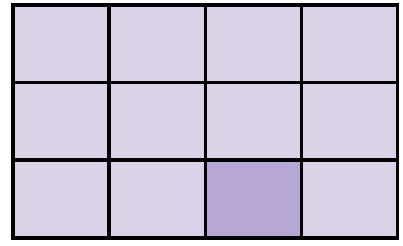


Locally Connected Layers

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Hidden activation



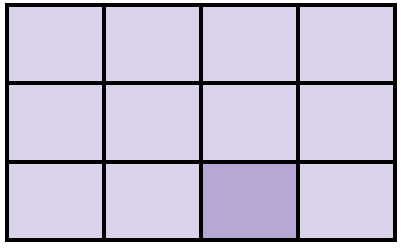
Q: Which region of the input is this hidden unit connected to?

Locally Connected Layers

Input activations

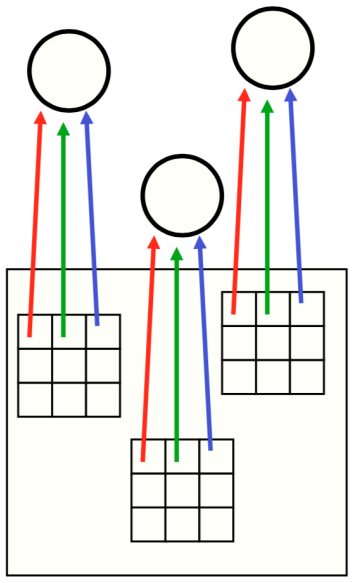
100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Hidden activation



Weight Sharing

Use the *same weights* across each region (each colour represents the same weight)



Each neuron on the higher layer is detecting the same feature, but in different locations on the lower layer

"Detecting" = output (activation) is high if feature is present
"Feature" = something in a part of the image, like an edge or shape

Convolution Computation

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Kernel

1	2	1
0	0	0
-1	-2	-1

Hidden activation

300			

$$\begin{aligned} 300 = & 100 \times 1 + 100 \times 2 + 100 \times 1 + \\ & 100 \times 0 + 100 \times 0 + 100 \times 0 + \\ & 100 \times (-1) + 0 \times (-2) + 0 \times (-1) \end{aligned}$$

- The **kernel** or **filter** (middle) contains the trainable weights
- In our example, the **kernel size** is 3×3
- The "**convolved features**" or "**feature map**" are other terms for the output hidden activation

Convolution Computation

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Kernel

1	2	1
0	0	0
-1	-2	-1

Hidden activation

300	300		

$$\begin{aligned} 300 = & 100 \times 1 + 100 \times 2 + 100 \times 1 + \\ & 100 \times 0 + 100 \times 0 + 100 \times 0 + \\ & 0 \times (-1) + 0 \times (-2) + 100 \times (-1) \end{aligned}$$

Convolution Computation (Your Turn!)

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Weights

1	2	1
0	0	0
-1	-2	-1

Hidden activation

300	300	?	

Q: What is the value of the highlighted hidden activation?

Convolution Computation

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

Kernel

1	2	1
0	0	0
-1	-2	-1

Hidden activation

300	300	100	

$$
\begin{aligned}
100 = & 100 \times 1 + 100 \times 2 + 100 \times 1 + \\
& 100 \times 0 + 100 \times 0 + 100 \times 0 + \\
& 0 \times (-1) + 100 \times (-2) + 100 \times (-1)
\end{aligned}
$$

Convolution Computation

Input activations

100	100	100	100	100	100
100	100	100	100	100	100
100	0	0	100	100	100
0	0	0	0	0	0
0	0	0	0	0	0

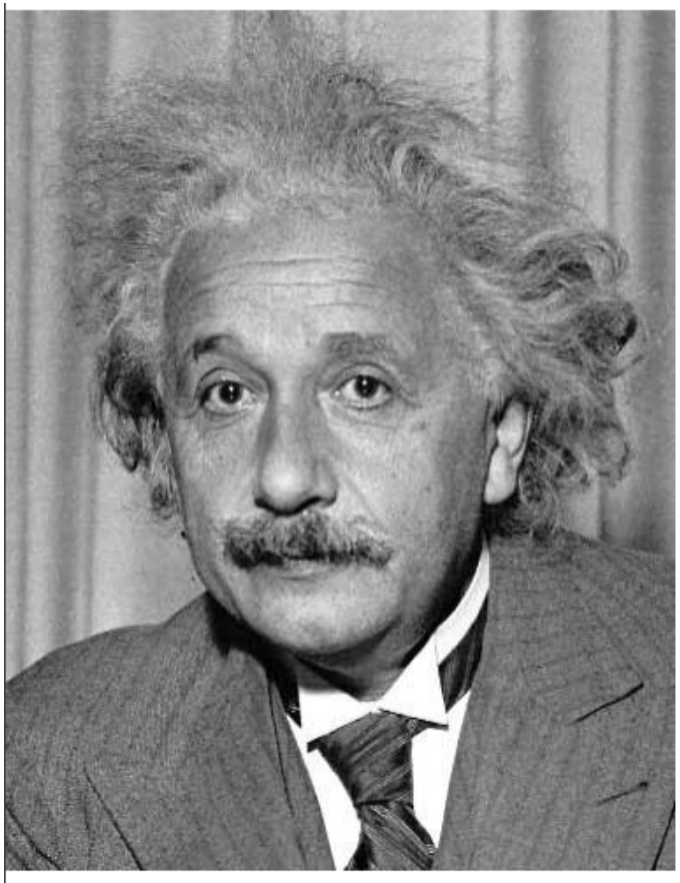
Kernel

1	2	1
0	0	0
-1	-2	-1

Hidden activation

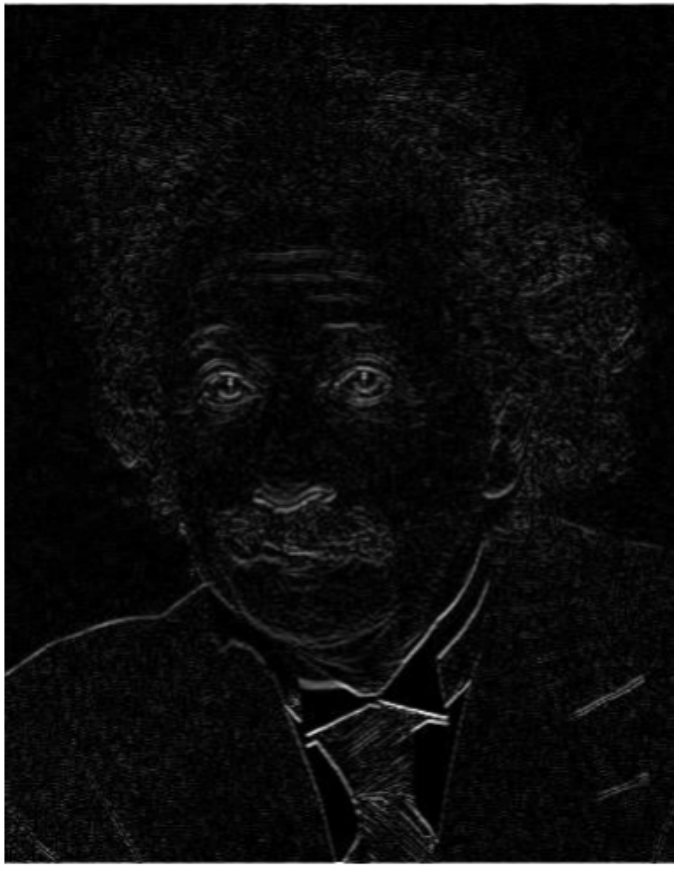
300	300	100	0
400	400	400	400
100	100	300	300

Sobel Filter - Weights to Detect Horizontal Edges



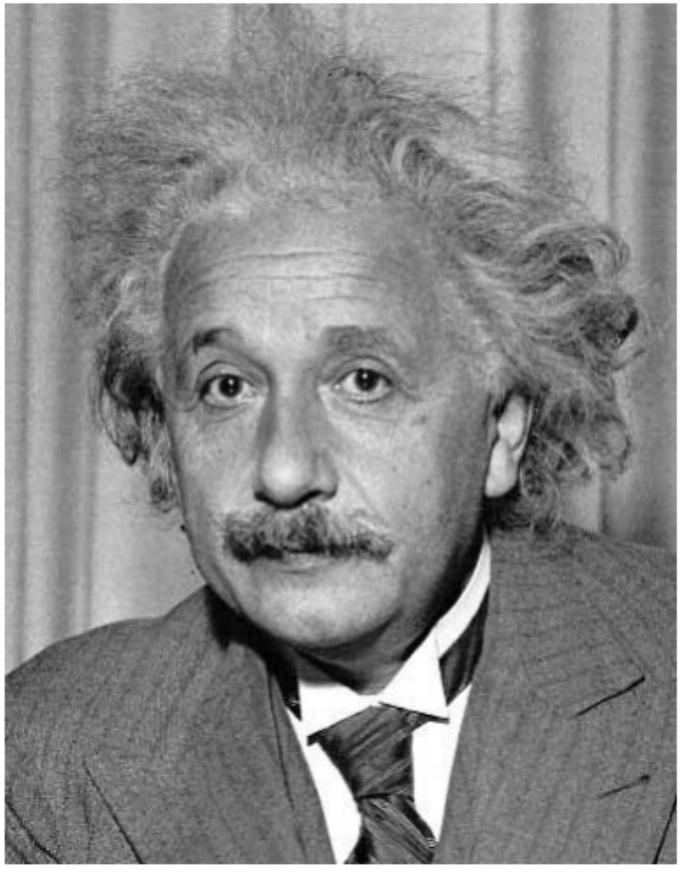
*

1	2	1
0	0	0
-1	-2	-1



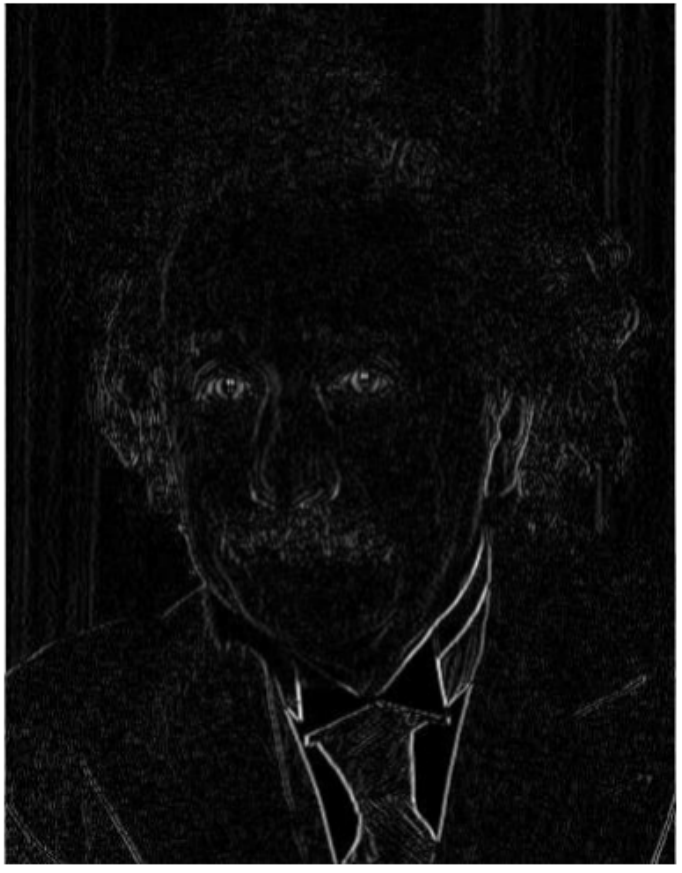
Horizontal Edge
(absolute value)

Sobel Filter - Weights to Detect Vertical Edges



*

1	0	-1
2	0	-2
1	0	-1



Vertical Edge
(absolute value)

Weights to Detect Blobs

$*$

$$\begin{pmatrix} 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \\ 0 & 2 & 3 & 5 & 5 & 5 & 3 & 2 & 0 \\ 3 & 3 & 5 & 3 & 0 & 3 & 5 & 3 & 3 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \\ 2 & 5 & 0 & -23 & -40 & -23 & 0 & 5 & 2 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \\ 3 & 3 & 5 & 3 & 0 & 3 & 5 & 3 & 3 \\ 0 & 2 & 3 & 5 & 5 & 5 & 3 & 2 & 0 \\ 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \end{pmatrix}$$

Q: What is the *kernel size* of this convolution?

Example:

Greyscale input image: 8×8

Convolution **kernel**: 5×5

Q: How many hidden units are in the output of this convolution?

Q: How many trainable parameters are there?

Example:

Greyscale input image: 8×8

Convolution **kernel**: 5×5

Q: How many hidden units are in the output of this convolution?

Q: How many trainable parameters are there?

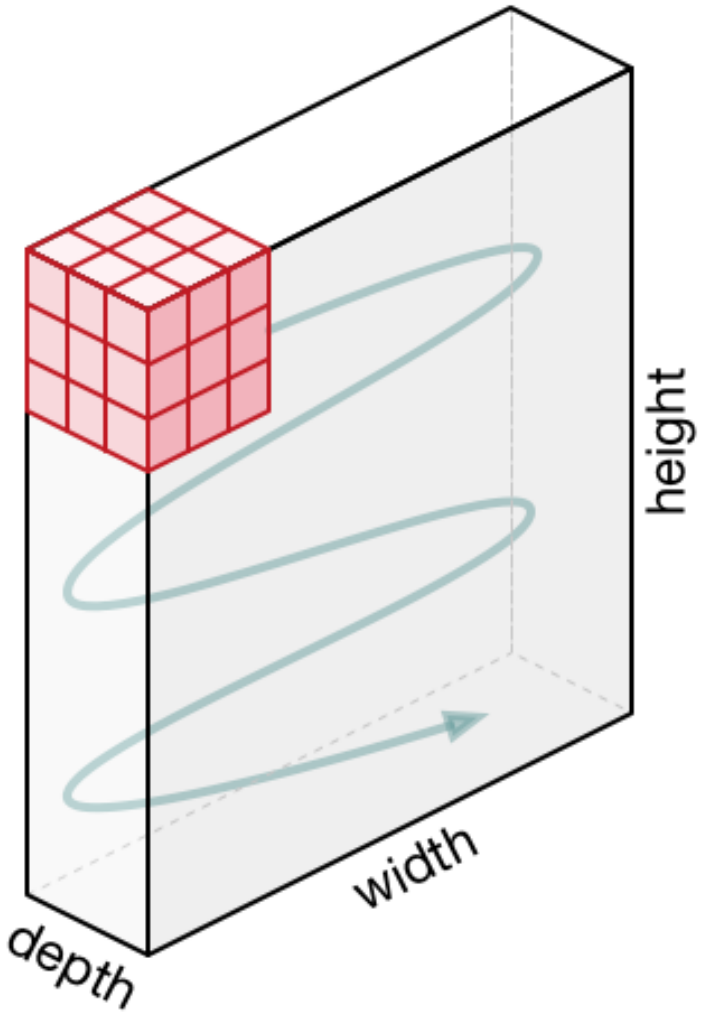
There are $5 \times 5 + 1$ trainable parameters. We have been ignoring the bias, but there is a bias applied.

Convolutions in Practice

What if we have a coloured image?

What if we want to compute *multiple* features?

Convolution in RGB



The kernel becomes a 3-dimensional tensor!

In this example, the kernel has size $\mathbf{3} \times 5 \times 5$

Convolutions: RGB Input

Colour input image: $\mathbf{3} \times 8 \times 8$

Convolution kernel: $\mathbf{3} \times 5 \times 5$

Questions:

- How many units are in the **output** of this convolution?
- How many trainable parameters are there?

Convolutions: RGB Input

Colour input image: $\mathbf{3} \times 8 \times 8$

Convolution kernel: $\mathbf{3} \times 5 \times 5$

Questions:

- How many units are in the **output** of this convolution?
- How many trainable parameters are there?

There are $3 \times 5 \times 5 + 1$ trainable parameters.

Terminology

Input image: $3 \times 8 \times 8$

Convolution kernel: $\mathbf{3} \times 5 \times 5$

- The number 3 is the number of **input channels** or **input feature maps**

Detecting Multiple Features

Q: What if we want to detect many features of the input? (i.e. **both** horizontal edges and vertical edges, and maybe even other features?)

Detecting Multiple Features

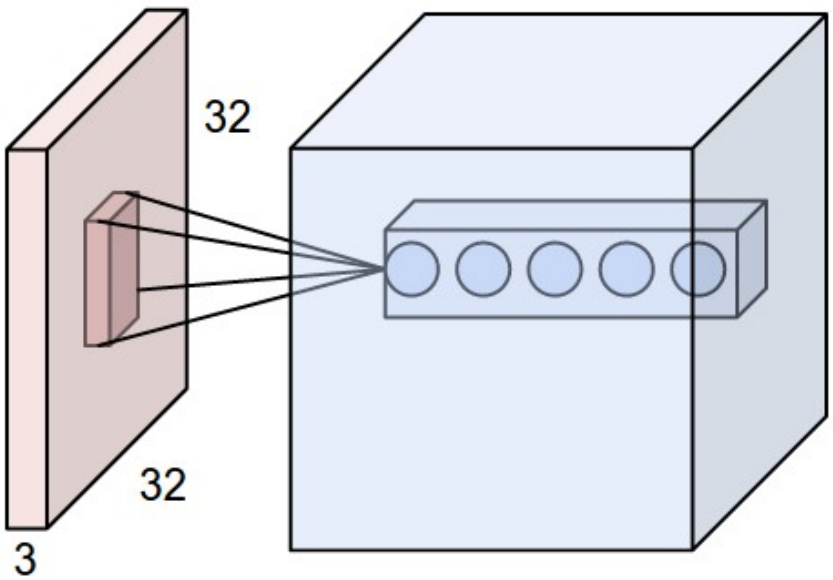
Q: What if we want to detect many features of the input? (i.e. **both** horizontal edges and vertical edges, and maybe even other features?)

A: Have many convolutional filters!

Detecting Multiple Features

Q: What if we want to detect many features of the input? (i.e. **both** horizontal edges and vertical edges, and maybe even other features?)

A: Have many convolutional filters!



Many Convolutional Filters

Input image: $3 \times 8 \times 8$

Convolution kernel: $\mathbf{10} \times 3 \times 5 \times 5$

Q:

- How many units are in the output of this convolution?
- How many trainable weights are there?

Many Convolutional Filters

Input image: $3 \times 8 \times 8$

Convolution kernel: $\mathbf{10} \times 3 \times 5 \times 5$

Q:

- How many units are in the output of this convolution?
- How many trainable weights are there?

There are $10 \times (3 \times 5 \times 5 + 1)$ trainable parameters.

More Terminology

Input image of size $3 \times 8 \times 8$

Convolution kernel of $\mathbf{10} \times \mathbf{3} \times 5 \times 5$

- The number 10 is the number of **output channels** or **output feature maps**
- The number 3 is the number of **input channels** or **input feature maps**

Example

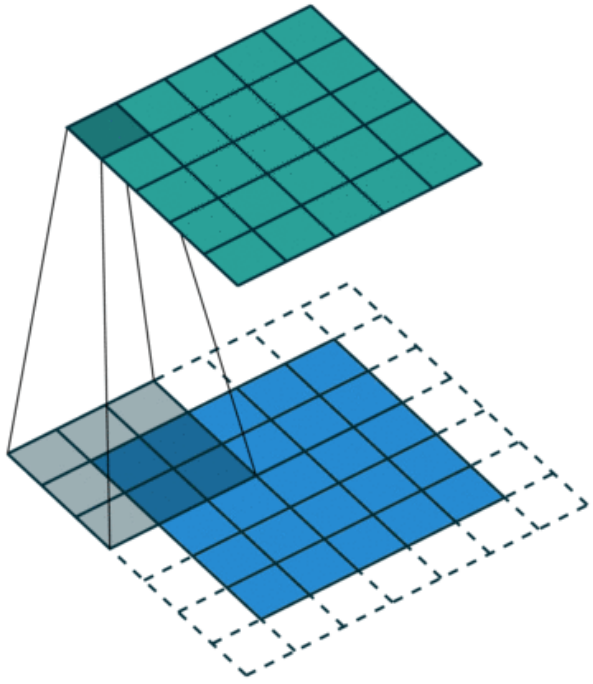
Input features: $5 \times 32 \times 32$

Convolution kernel: $8 \times 5 \times 3 \times 3$

Questions:

- How many input channels are there?
- How many output channels are there?
- How many units are in the higher layer?
- How many trainable weights are there?

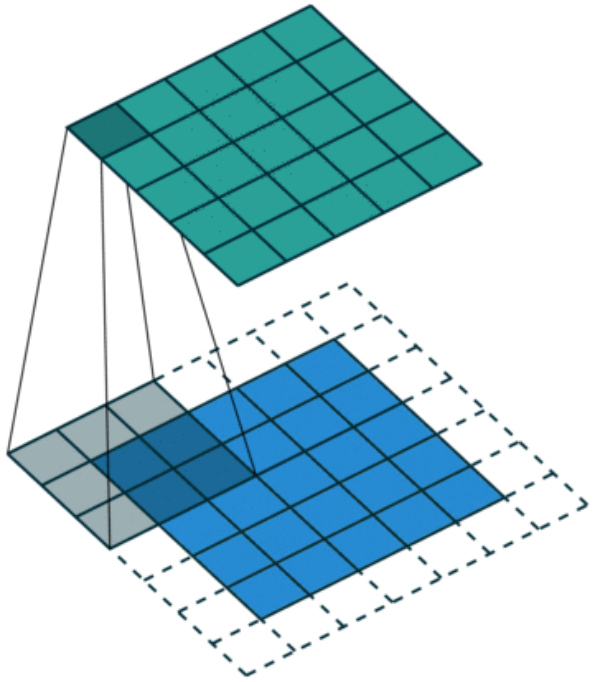
Zero Padding



- Add zeros around the border of the image
- (Can add more than one pixel of zeros)

Q: Why might we want to add zero padding?

Zero Padding



- Add zeros around the border of the image
- (Can add more than one pixel of zeros)

Q: Why might we want to add zero padding?

- Keep the next layer's width and height consistent with the previous
- Keep the information around the border of the image

Convolutional Layers in PyTorch

Let's take a look at convolutional layers in PyTorch!

Consolidating Information

In a neural network with fully-connected layers, we reduced the number of units in each hidden layer

Q: Why?

Consolidating Information

In a neural network with fully-connected layers, we reduced the number of units in each hidden layer

Q: Why?

- To be able to consolidate information, and remove out information not useful for the current task

Q: How can we consolidate information in a neural network with convolutional layers?

Consolidating Information

In a neural network with fully-connected layers, we reduced the number of units in each hidden layer

Q: Why?

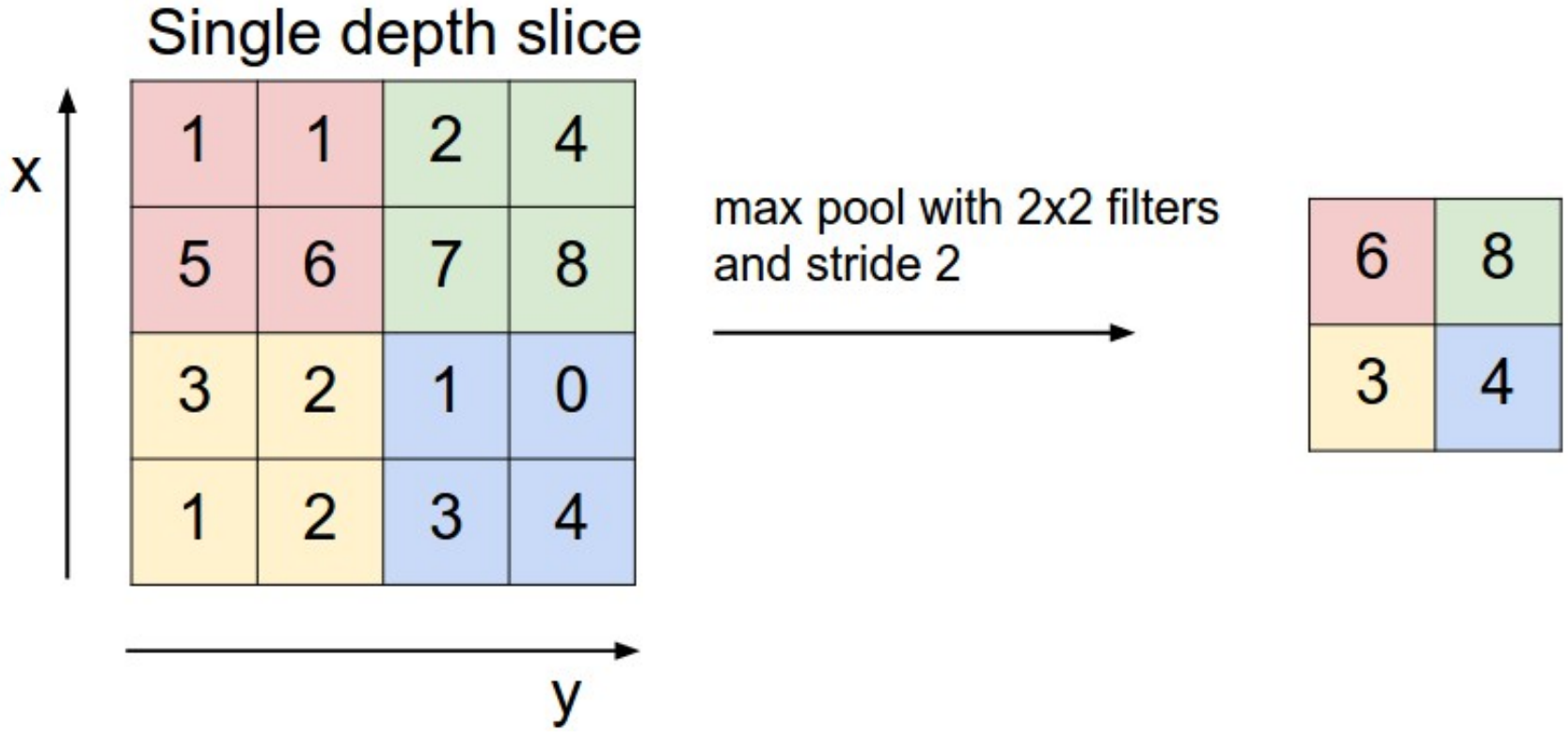
- To be able to consolidate information, and remove out information not useful for the current task

Q: How can we consolidate information in a neural network with convolutional layers?

- max pooling, average pooling, strided convolutions

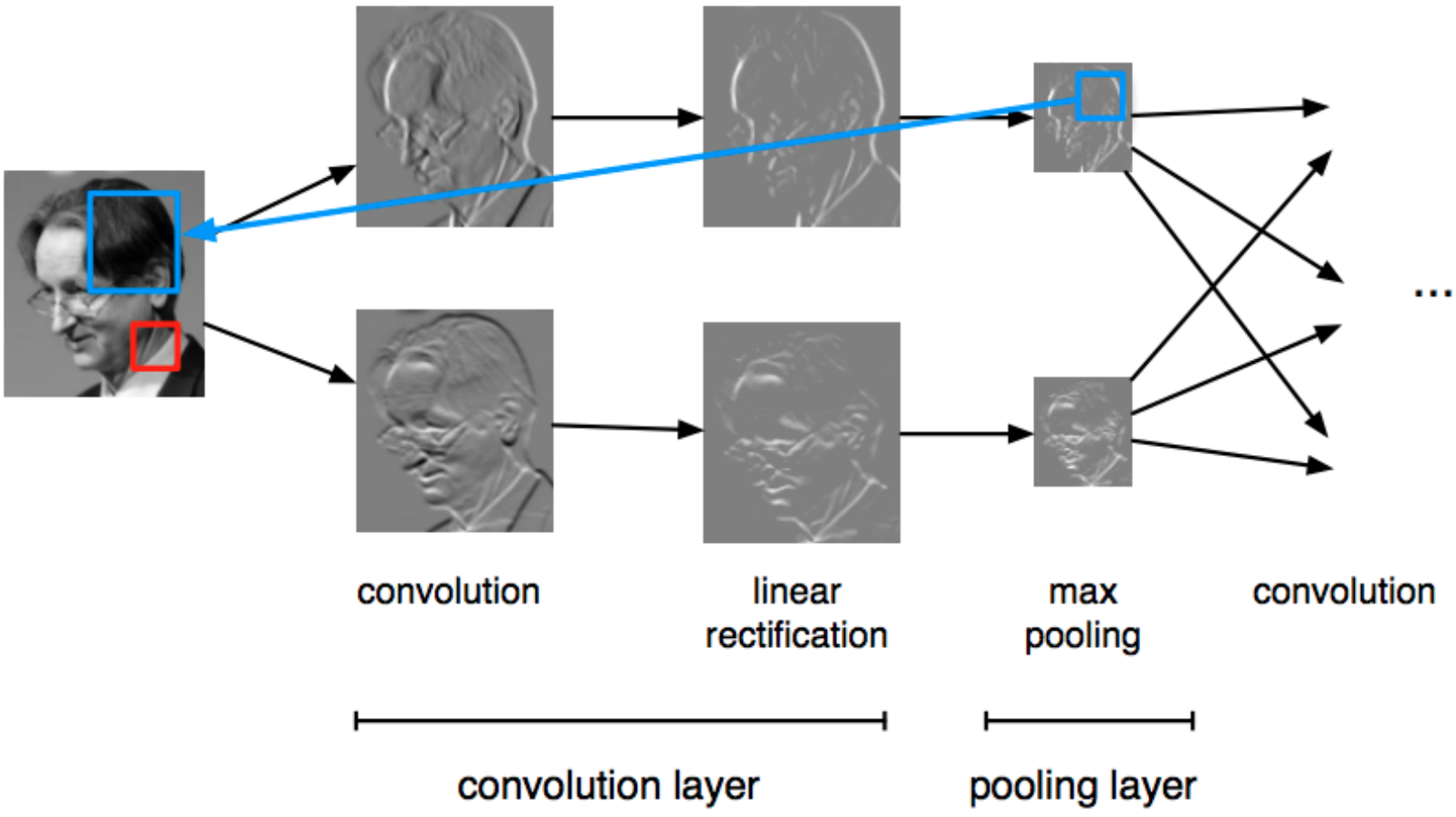
Max-Pooling

Idea: take the **maximum value** in each 2×2 grid.



Max-Pooling Example

We can add a max-pooling layer *after* each convolutional layer

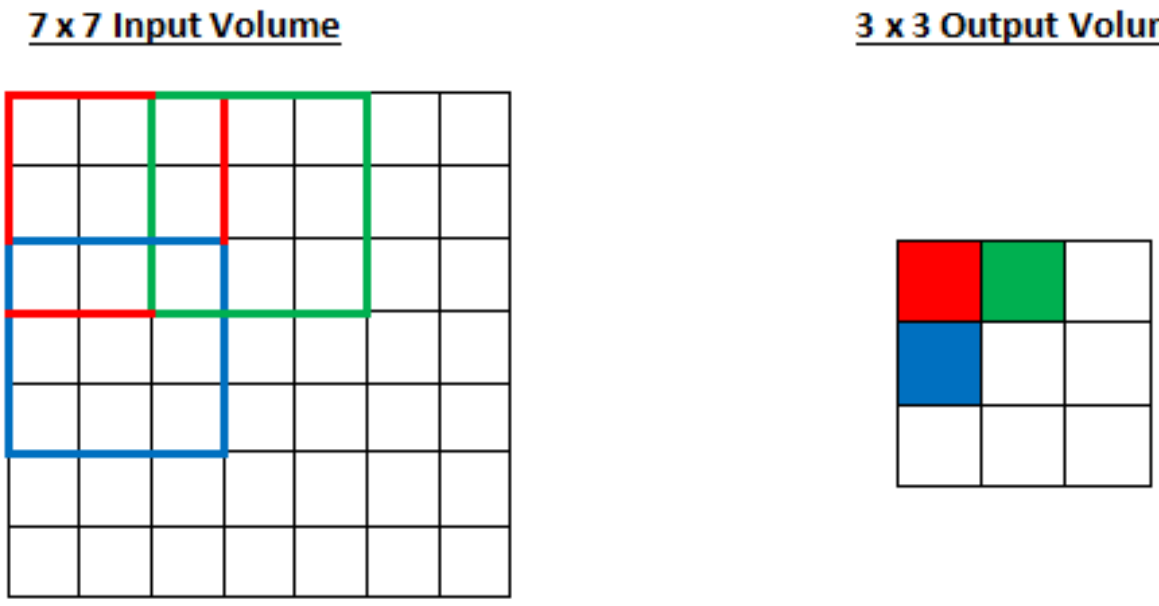


Average Pooling

- Average pooling (compute the average activation of a region)
- Max pooling generally works better

Strided Convolution

More recently people are doing away with pooling operations, using **strided** convolutions instead:

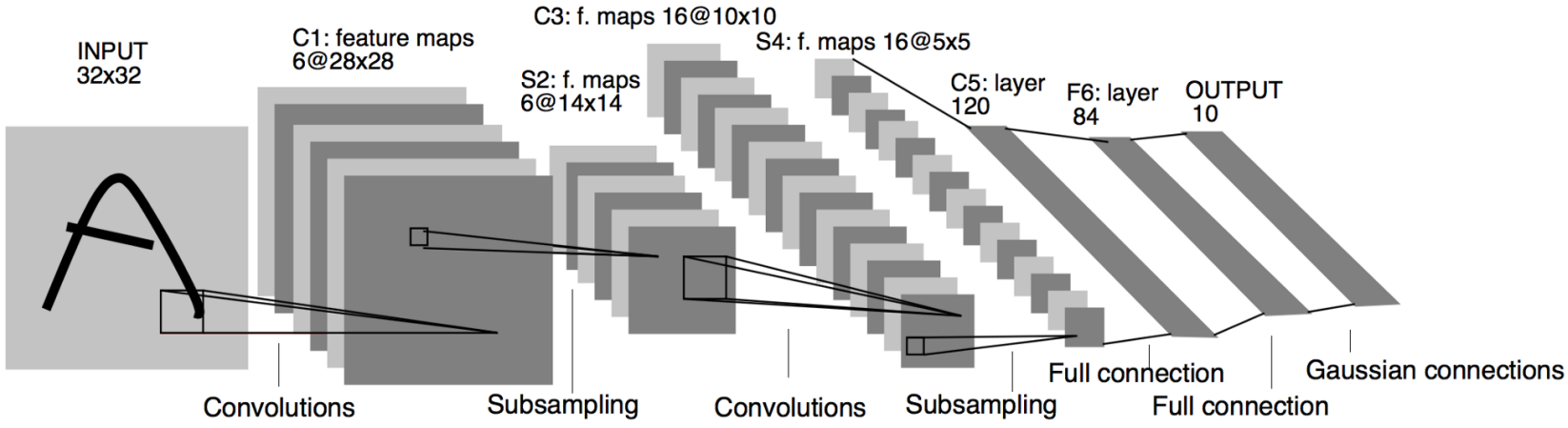


Shift the kernel by **2** (stride=2) when computing the next output feature.

Visuals

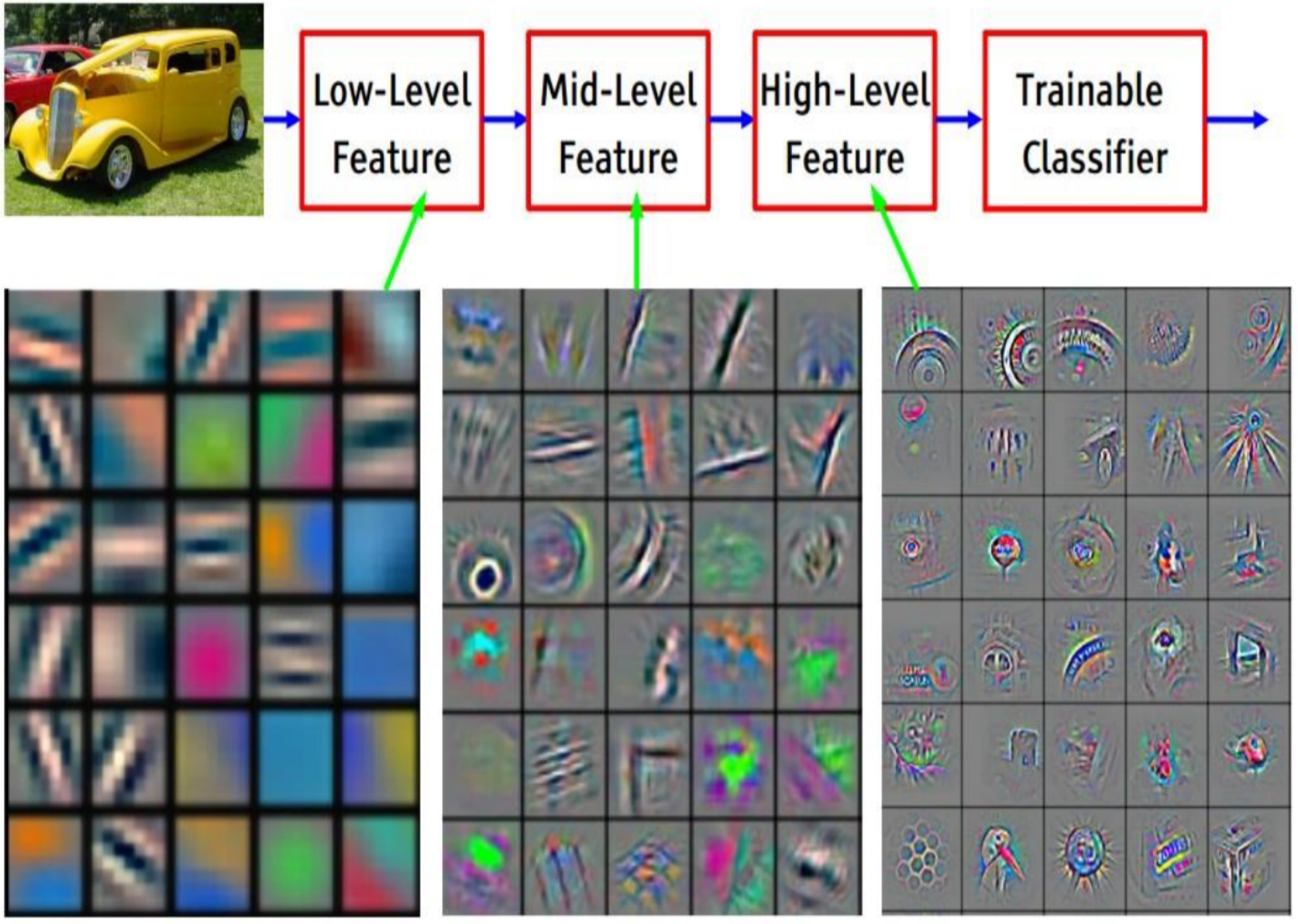
https://arxiv.org/pdf/1603.07285.pdf
https://github.com/vdumoulin/conv_arithmetic

Early Convolutional Architecture: LeNet Architecture



- Input: 32x32 pixel, greyscale image
- First convolution has 6 output features (5x5 convolution?)
- First subsampling is probably a max-pooling operation
- Second convolution has 16 output features (5x5 convolution?)
- ...
- Some number of fully-connected layers at the end

What features do CNN's detect?



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Training a CNN

- We train using backpropagation!
- We already know how to work with shared weights (project 2), and computing CNN weight updates works the same way