APS360 Fundamentals of AI

Lisa Zhang

Lecture 6; May 27, 2019

Agenda

Last class:

- Multi-class classification in PyTorch
- Defining a Neural Network in PyTorch
- Cross Entropy Loss for classification

Today:

- Discuss Labs 1, 2, 3
- Convolutional Neural Networks
- Debugging Neural Networks

How were last two classes?

How were last two classes?

Q: What is the difference between the validation set and the test set?

How were last two classes?

Q: What is the difference between the validation set and the test set?

Q: Should you choose hyperparameters based on validation accuracy or test accuracy?

How were last two classes?

Q: What is the difference between the validation set and the test set?

Q: Should you choose hyperparameters based on validation accuracy or test accuracy?

Q: What is the difference between the softmax and the signmoid activations?

How were last two classes?

Q: What is the difference between the validation set and the test set?

Q: Should you choose hyperparameters based on validation accuracy or test accuracy?

Q: What is the difference between the softmax and the signmoid activations?

Q: What do you think about the pacing of the course?

Labs

Lab 1

- Generally very well done
- Common issues:
  - Using builtins functions like `sum`, `input`, as variable names
  - Using `print` instead of `return`
  - Not vectorizing code when it is very easy to
  - The `<Tensor>.dtype` attribute: `float32` does not mean there are 32 floats
- Remark requests due by email by June 2nd
  - Your mark can go either up or down
  - Exception: counting error

Lab 2

I hope this lab was fun!

- Thank you for all the Piazza questions and answers!
- Huan (TA) is working on grading (partially complete)
- Common issue (so far):
  - Not re-randomize the weights when trying a new hyperparameter setting (i.e. reinitialize `large_net` or `small_net`)

Lab 3 Part A

- Jake (TA) is also currently working on grading, consolidating and anonymizing the data (partially complete)
- Data will be availabe in the next day or two
- Quality of the data might pose a challenge (background, photoshopping)

Lab 3 Part B

- You don't need the data to get starter
- You can finish Part 1 after this lecture
- You can also write code for Part 5 after this lecture

Lab 3 Part B

- You don't need the data to get starter
- You can finish Part 1 after this lecture
- You can also write code for Part 5 after this lecture

- Building a model from scratch is scary, but also really rewarding
- Give yourself a lot of time for training and debugging
- You can use any code from lecture or from the previous labs, but think about your architecture and choice of loss function!

Come to office hours if you need help!

Convolutional Neural Networks

Last class, we used fully-connected layers like this:

```
class MNISTClassifier(nn.Module):
    def __init__(self):
        super(MNISTClassifier, self).__init__()
        self.layer1 = nn.Linear(28 * 28, 50)
        self.layer2 = nn.Linear(50, 20)
        self.layer3 = nn.Linear(20, 10)
    def forward(self, img):
        flattened = img.view(-1, 28 * 28)
        activation1 = F.relu(self.layer1(flattened))
        activation2 = F.relu(self.layer2(activation1))
        output = self.layer3(activation2)
        return output
```

Last class, we used fully-connected layers like this:

```
class MNISTClassifier(nn.Module):
    def __init__(self):
        super(MNISTClassifier, self).__init__()
        self.layer1 = nn.Linear(28 * 28, 50)
        self.layer2 = nn.Linear(50, 20)
        self.layer3 = nn.Linear(20, 10)
    def forward(self, img):
        flattened = img.view(-1, 28 * 28)
        activation1 = F.relu(self.layer1(flattened))
        activation2 = F.relu(self.layer2(activation1))
        output = self.layer3(activation2)
        return output
```

Q: How many layers does this network have?

Last class, we used fully-connected layers like this:

```
class MNISTClassifier(nn.Module):
    def __init__(self):
        super(MNISTClassifier, self).__init__()
        self.layer1 = nn.Linear(28 * 28, 50)
        self.layer2 = nn.Linear(50, 20)
        self.layer3 = nn.Linear(20, 10)
    def forward(self, img):
        flattened = img.view(-1, 28 * 28)
        activation1 = F.relu(self.layer1(flattened))
        activation2 = F.relu(self.layer2(activation1))
        output = self.layer3(activation2)
        return output
```

Q: How many layers does this network have?

Q: How many weights are in the *first* layer of this network?

What if our network is bigger?

- Input image: 200×200 pixels
- First hidden layer: 500 units

Q: How many weights are there?

What if our network is bigger?

- Input image: 200×200 pixels
- First hidden layer: 500 units

Q: How many weights are there?

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

What if our network is bigger?

- Input image: 200×200 pixels
- First hidden layer: 500 units

Q: How many weights are there?

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

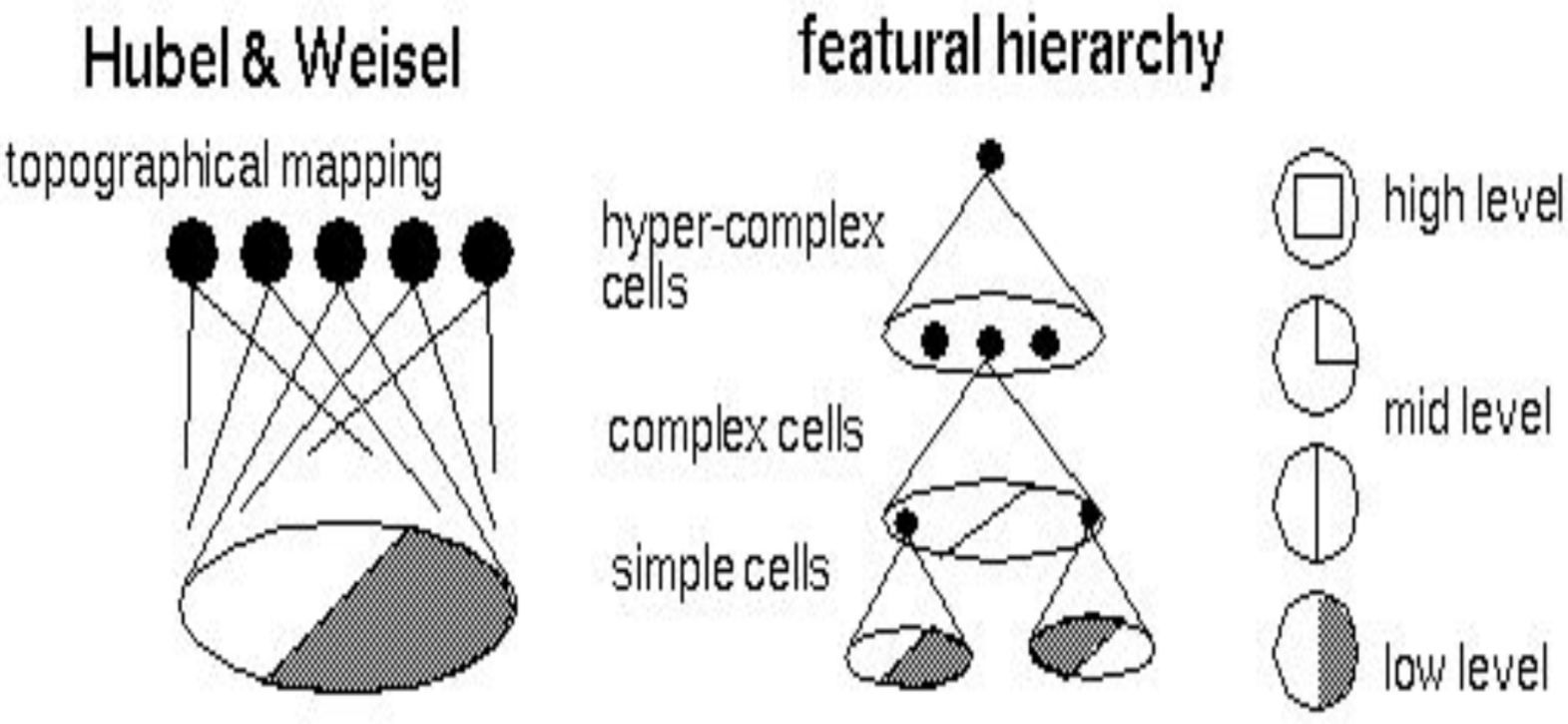
- computing predictions (forward pass) will take a long time
- a large number of weights requires a lot of training data to avoid overfitting
- small shift in image can result in large change in prediction
- does not make use of the geometry of the image

Convolutional Neural Network

Ideas:

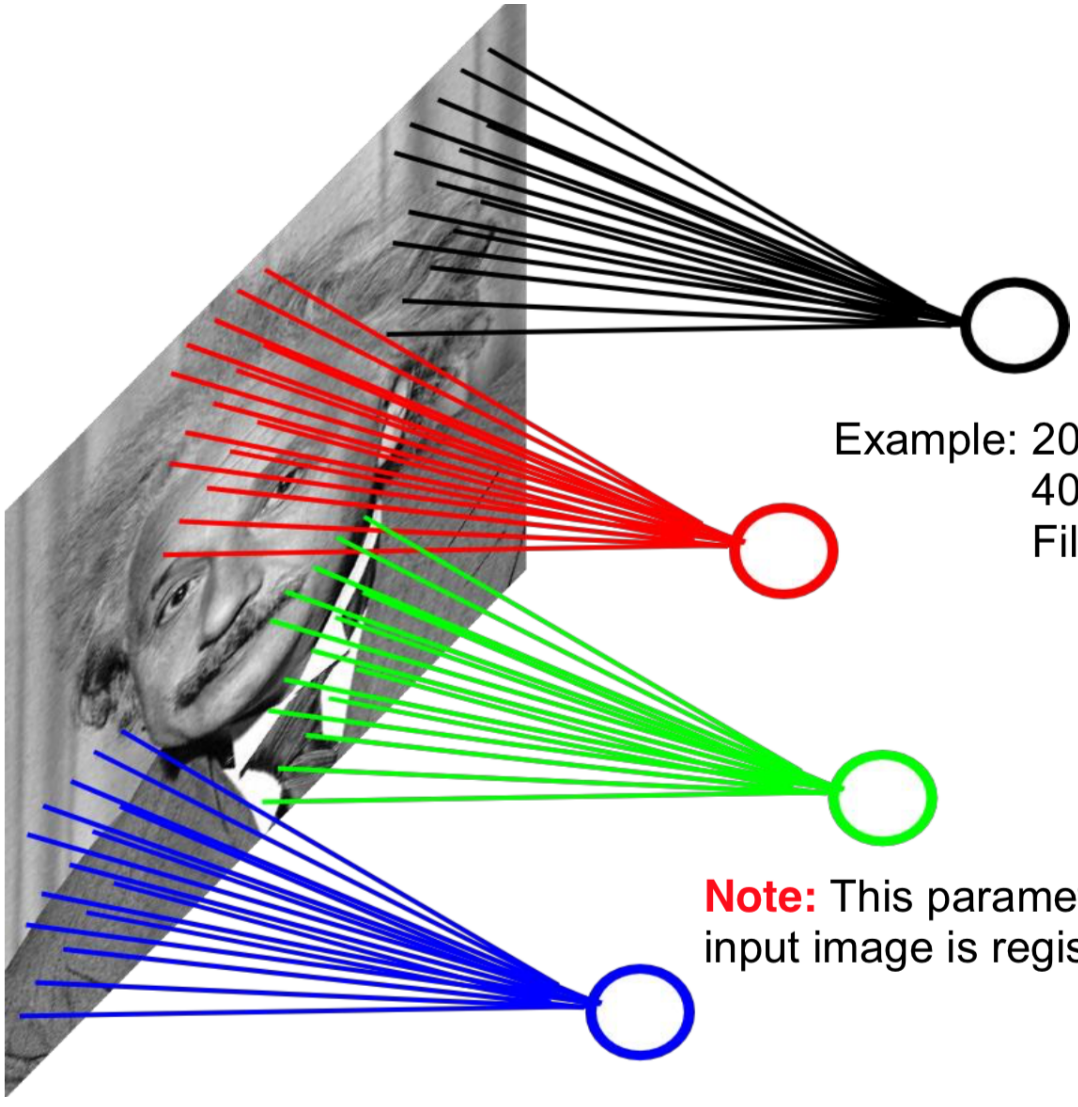
- Locally-connected layers: look for **local** features in small regions of the image
- Weight-sharing: detect the **same** local features across the entire image

Biological Influence



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

Locally Connected Layers

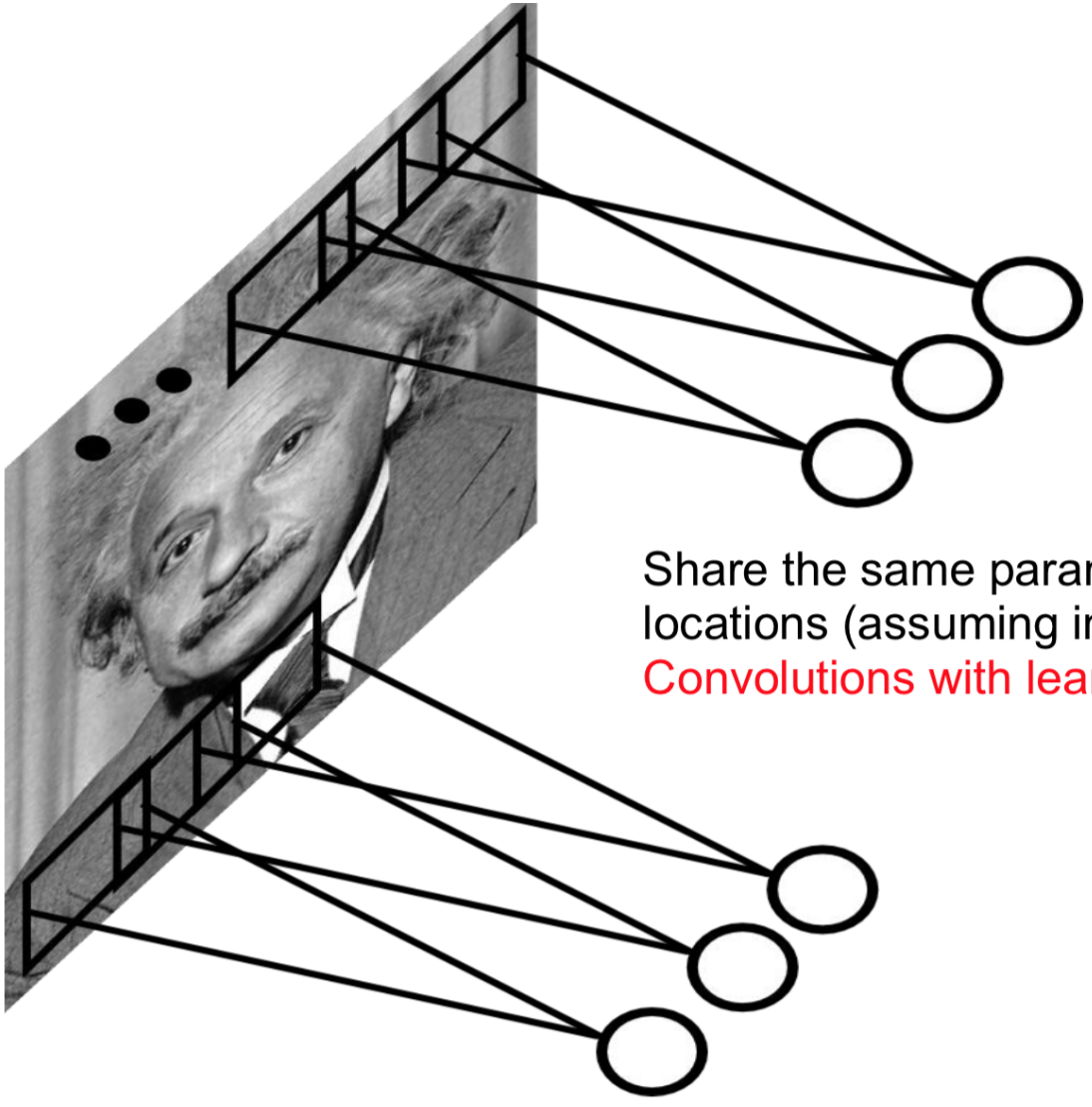


Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Ranzato

Weight Sharing



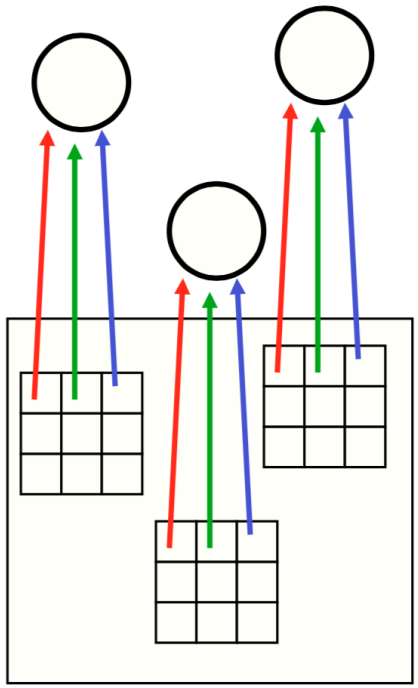
Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

Ranzato

Weight Sharing (continued)

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

The red connections all have the same weight.

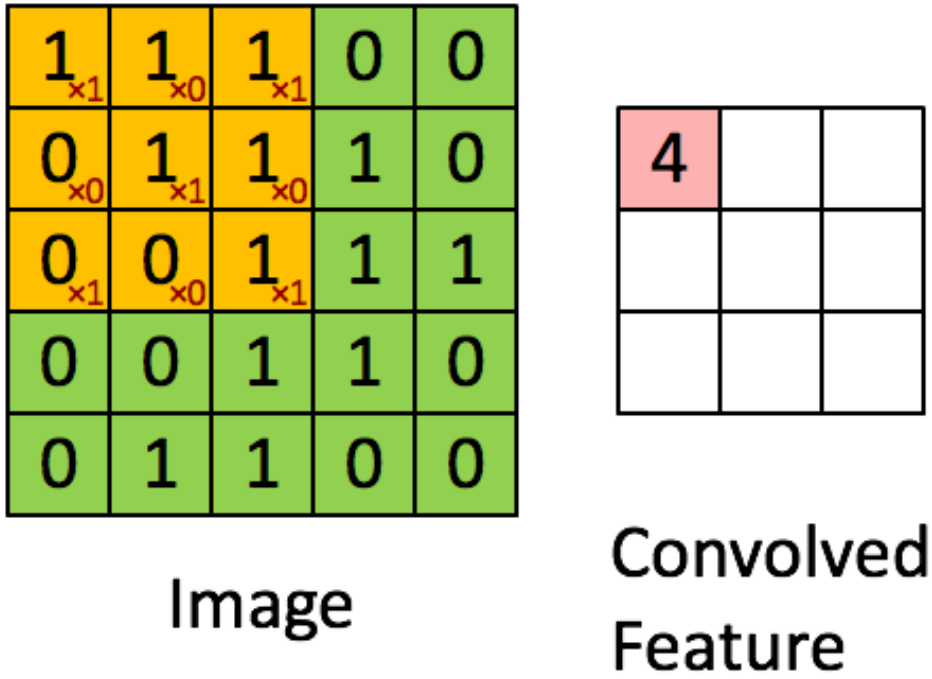


"Detecting" = the output (activation) is high if the feature is present

"Feature" = something in the image, like an edge, blob or shape

Forward Pass Example (Greyscale Image)

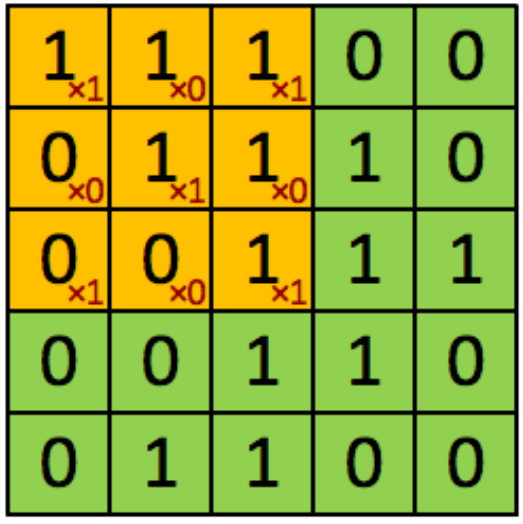
https://cdn-images-1.medium.com/max/1200/1*GcI7G-JLAQiEoCON

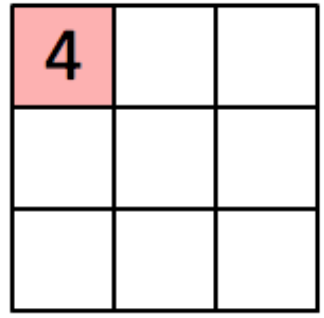


- The *kernel* or *filter* (red) contains the trainable weights. In this picture, the *kernel size* is 3×3
- The "*convolved features*" is another term for "convolution output"

Forward Pass Questions (Greyscale Image)

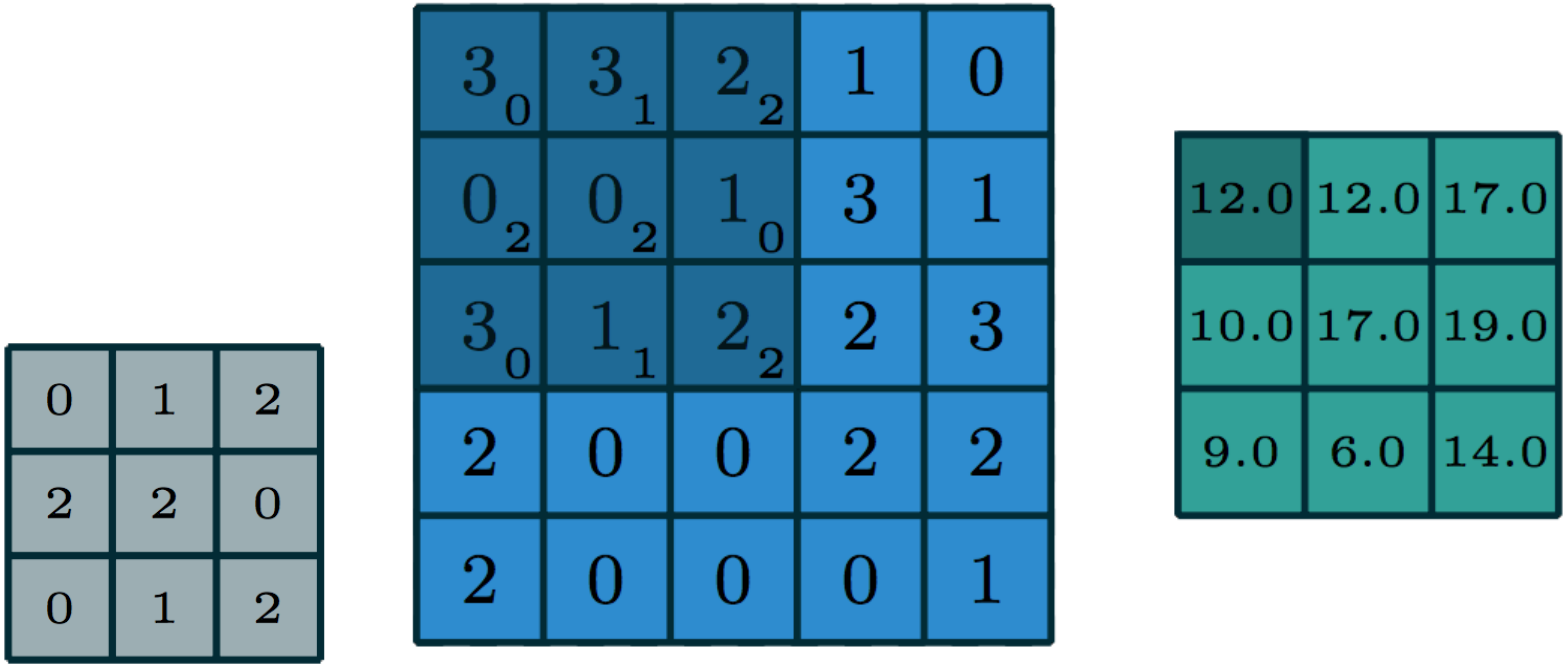
https://cdn-images-1.medium.com/max/1200/1*GcI7G-JLAQiEoCON





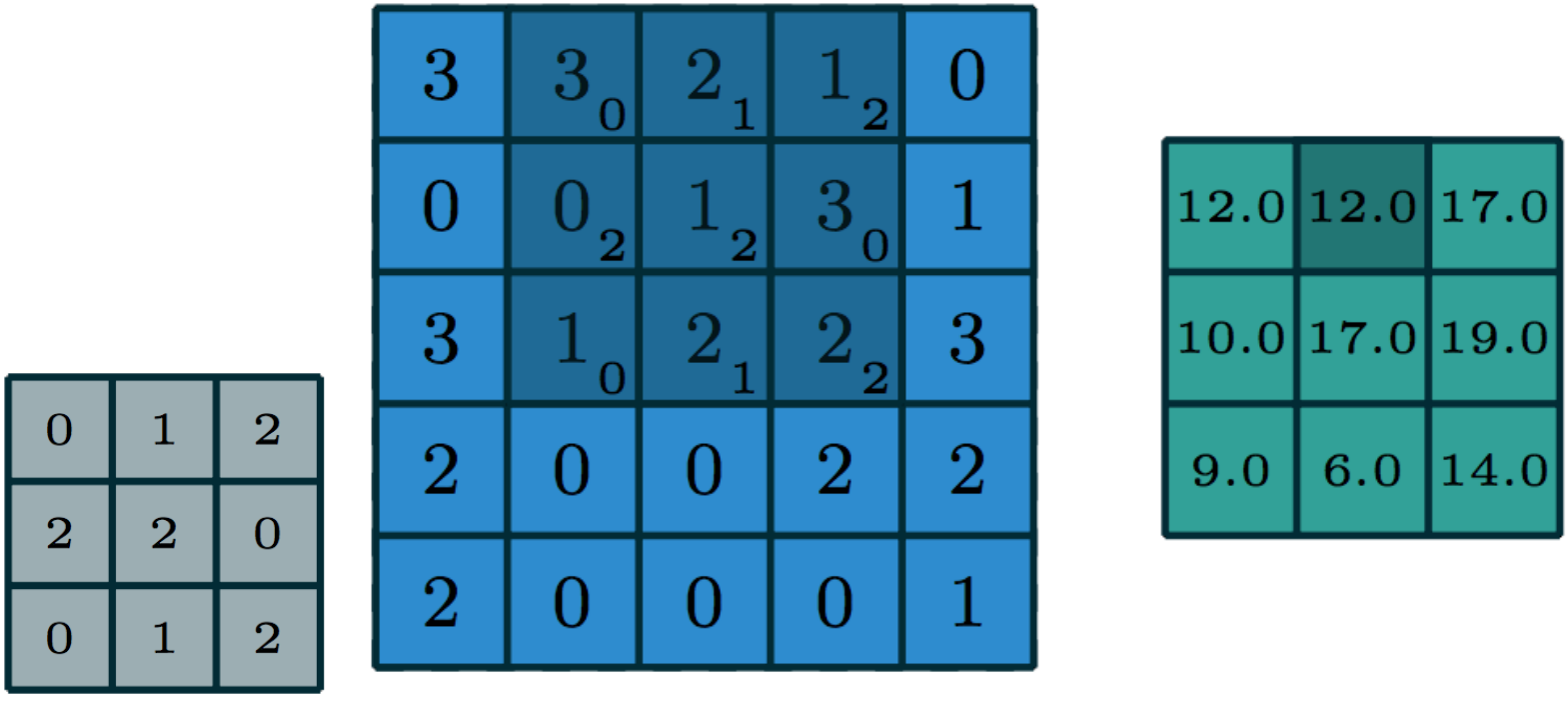
Q: What should the value of the next output (to the right of the 4) be?

Example from Notes (1)



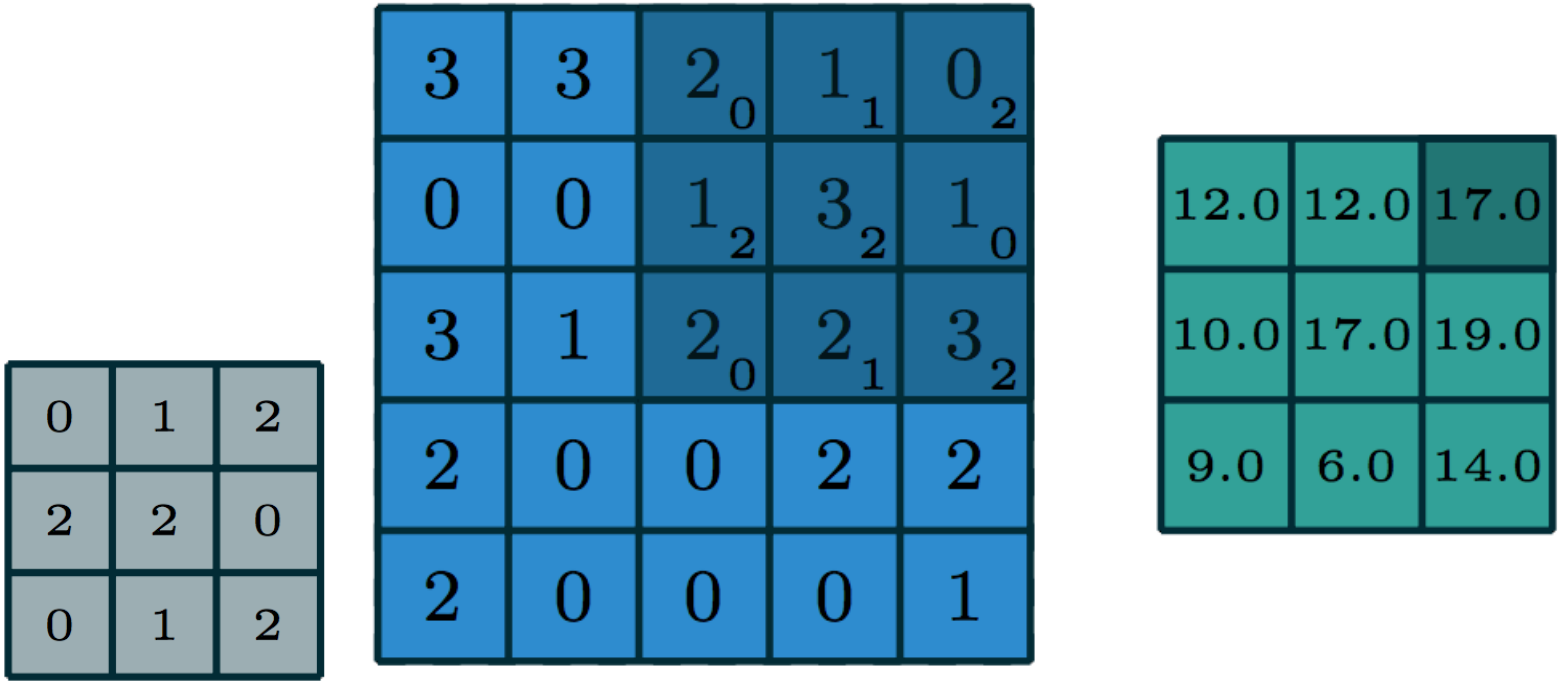
gray=kernel, blue=input, green=output

Example from Notes (2)



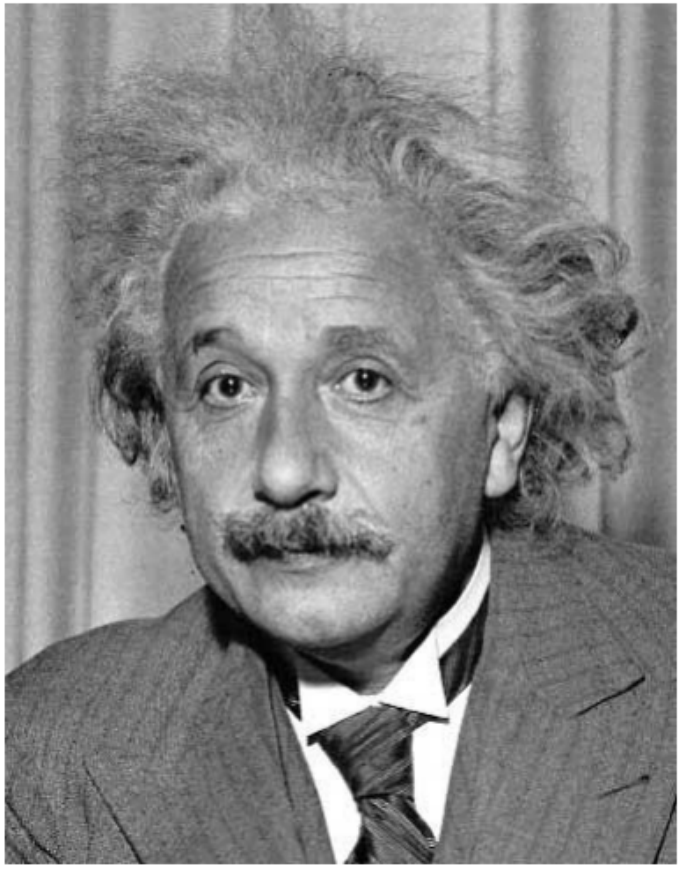
gray=kernel, blue=input, green=output

Example from Notes (3)



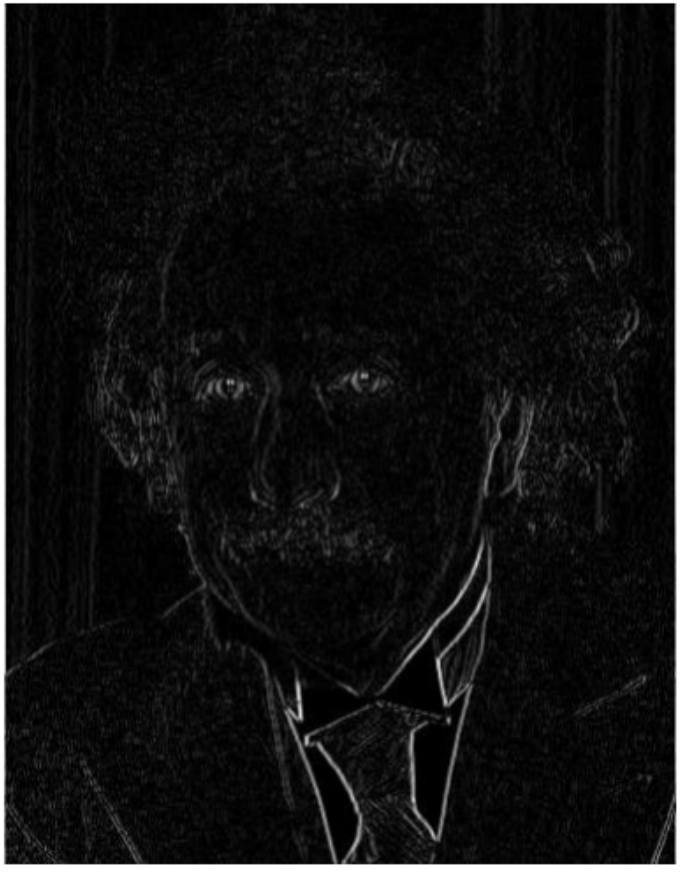
gray=kernel, blue=input, green=output

Sobel Filter - Weights to Detect Vertical Edges



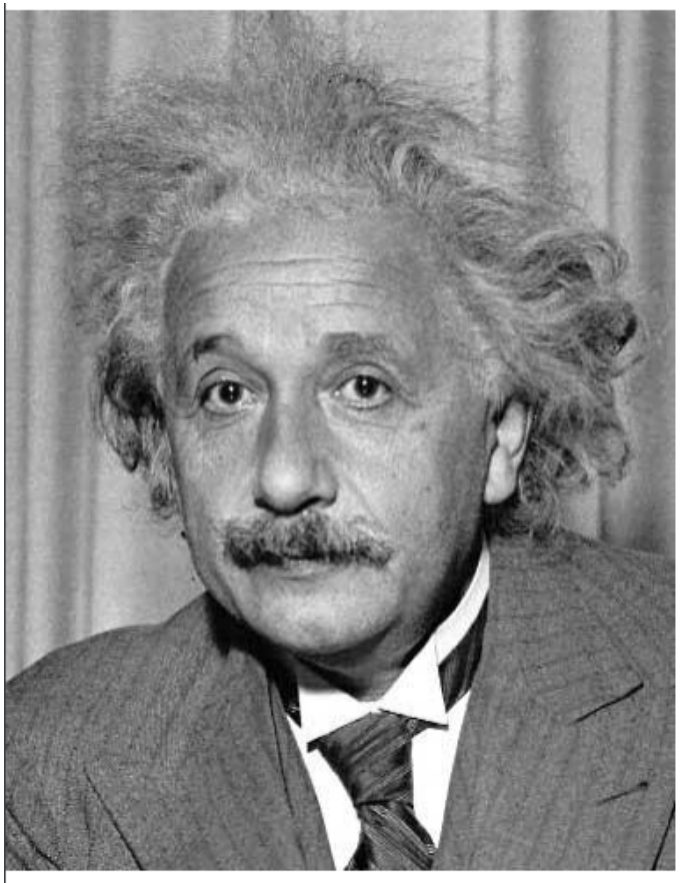
*

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



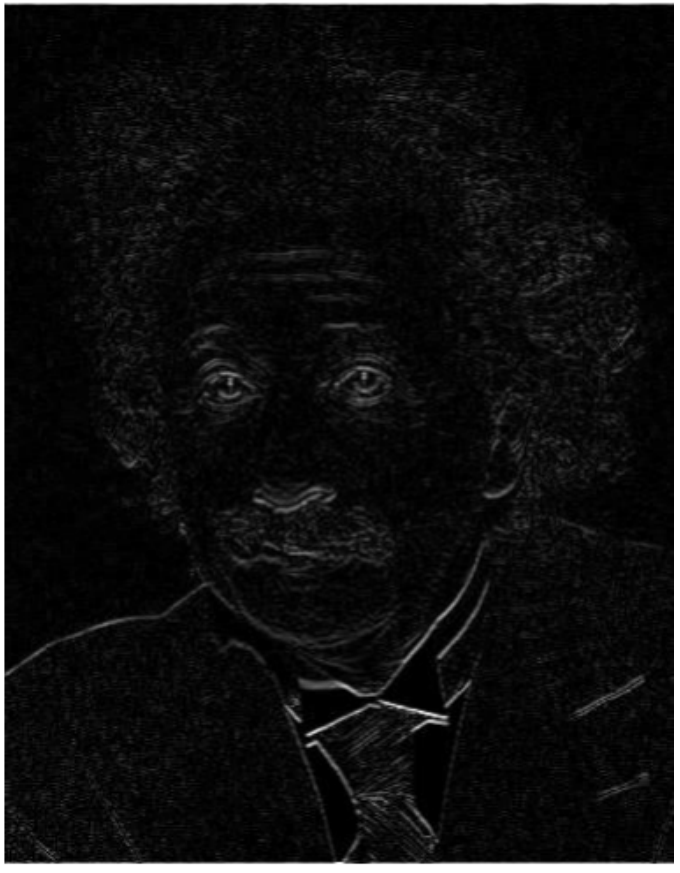
Vertical Edge
(absolute value)

Sobel Filter - Weights to Detect Horizontal Edges



*

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



Horizontal 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: 7×7

Convolution **kernel**: 3×3

Questions:

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

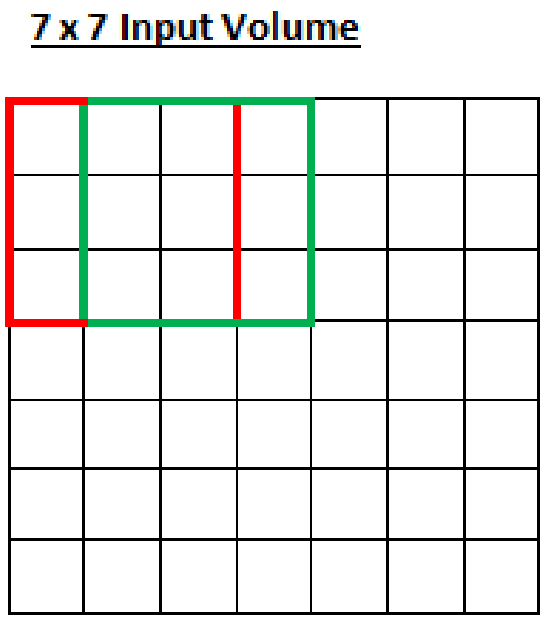
Example:

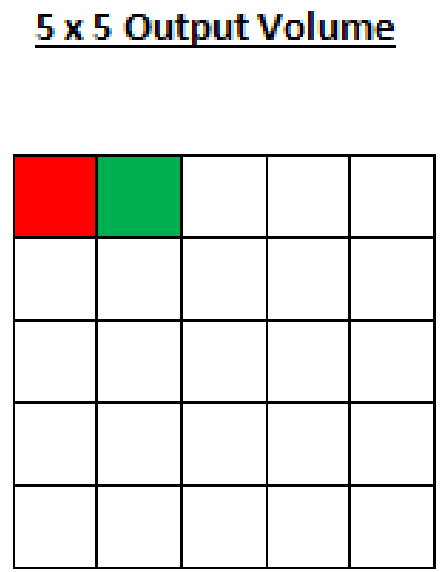
Greyscale input image: 7×7

Convolution **kernel**: 3×3

Questions:

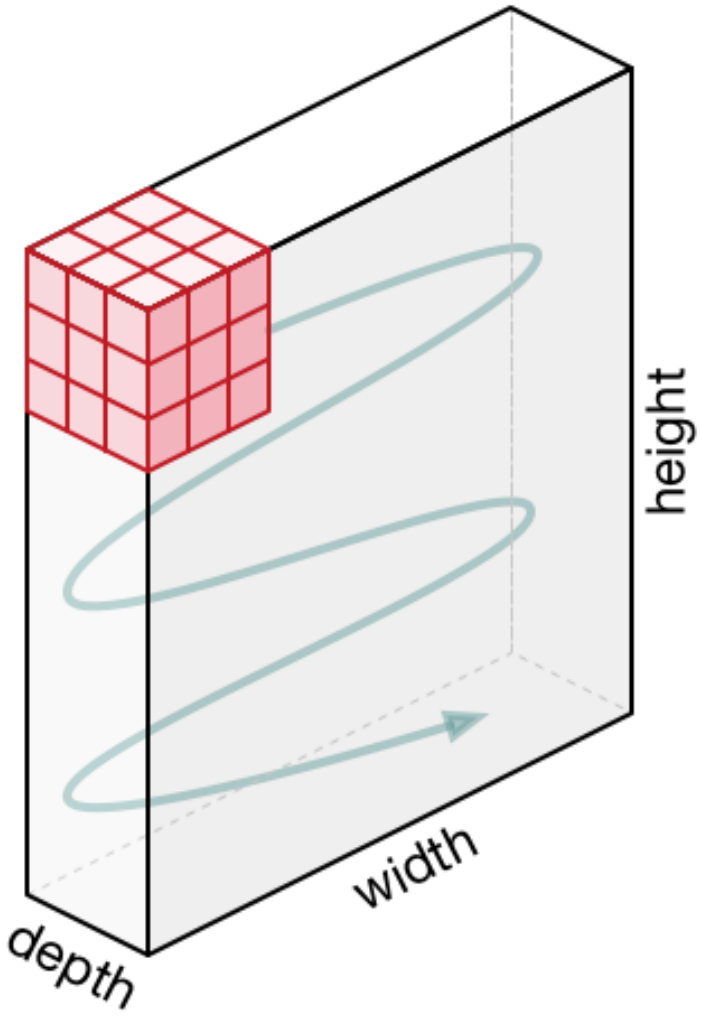
- How many units are in the output?
- How many trainable weights are there?





What if we have 3 colours?

Convolution in RGB



The kernel becomes a 3-dimensional tensor!

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

Convolutions: RGB Input

Colour input image: $\mathbf{3} \times 7 \times 7$

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

Questions:

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

Terminology

Input image: $3 \times 32 \times 32$

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

- 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

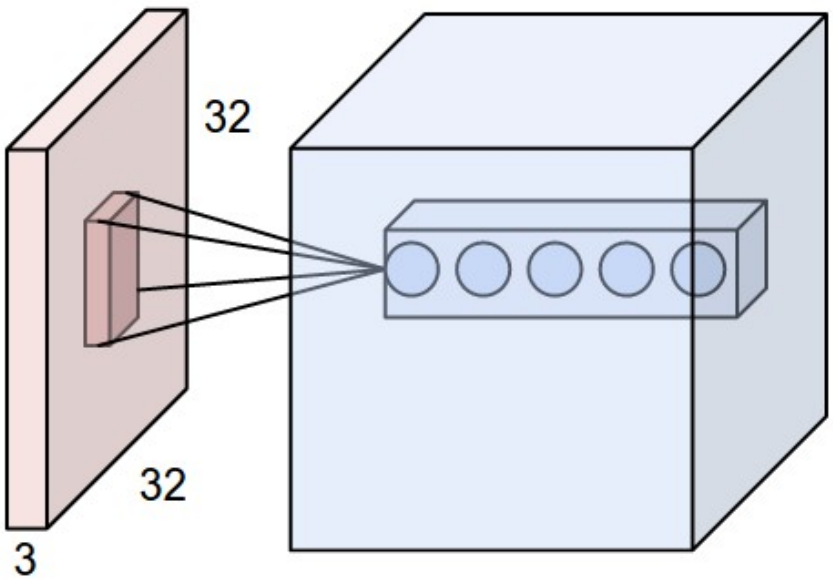
Q: What if we want to detect many features of the input? (i.e. **both** horizontal edges and vertical edges, and maybe even other features?)

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: $3 \times 7 \times 7$

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

Questions:

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

More Terminology

Input image of size $3 \times 32 \times 32$

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

- The number 3 is the number of **input channels** or **input feature maps**
- The number 5 is the number of **output channels** or **output feature maps**

Example

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

Convolution kernel: $5 \times 3 \times 3 \times 10$

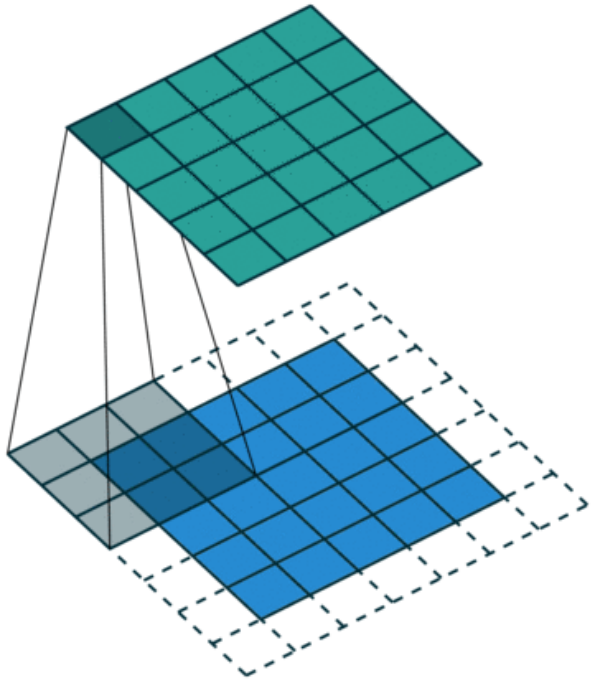
Questions:

- 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?

Convolutional Layers in PyTorch

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

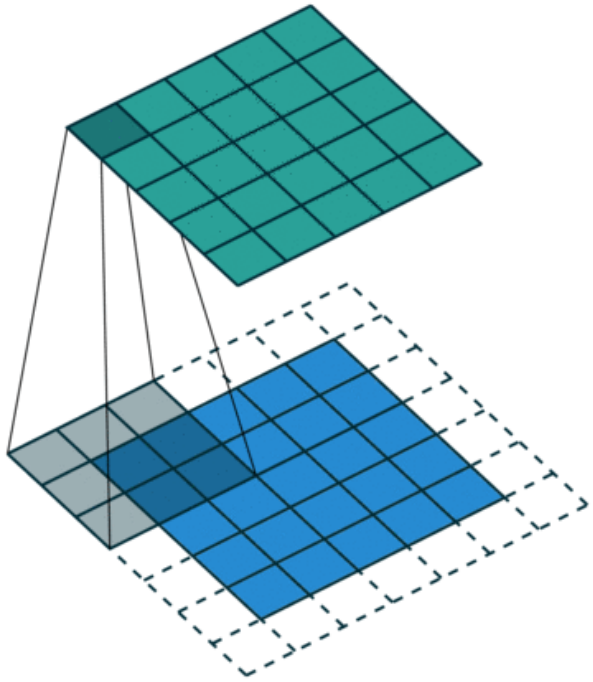
Zero Padding



- 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

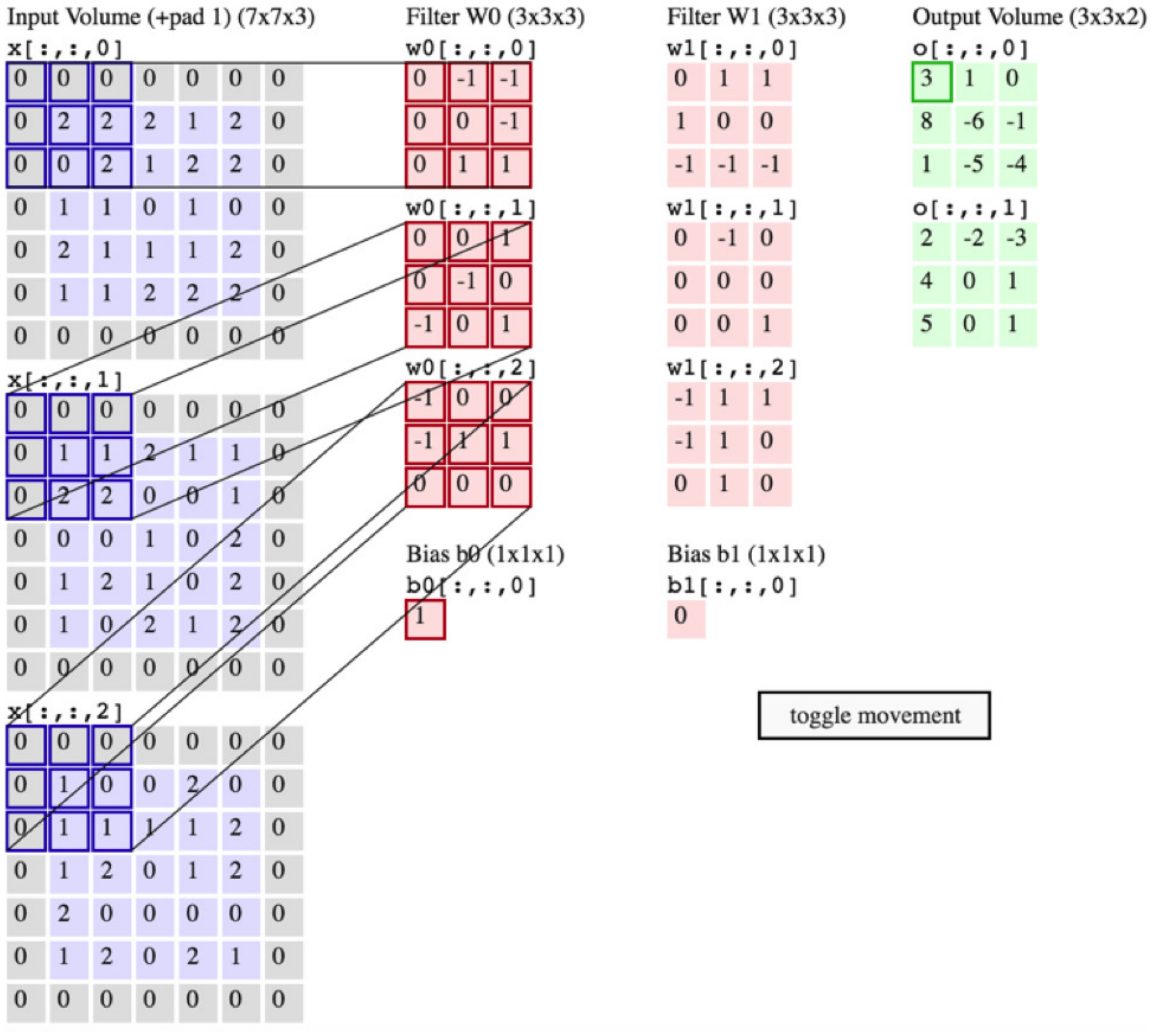


- Add zeros around the border of the image
- (Can add more than one pixel of zeros)

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

Zero Padding: Multiple Channels



Zero Padding in PyTorch

PyTorch's `nn.Conv2D` has a `padding` parameter (default 0, you are not expected to know).

See more at:
https://pytorch.org/docs/stable/nn.html#torch.nn.Conv2d

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?

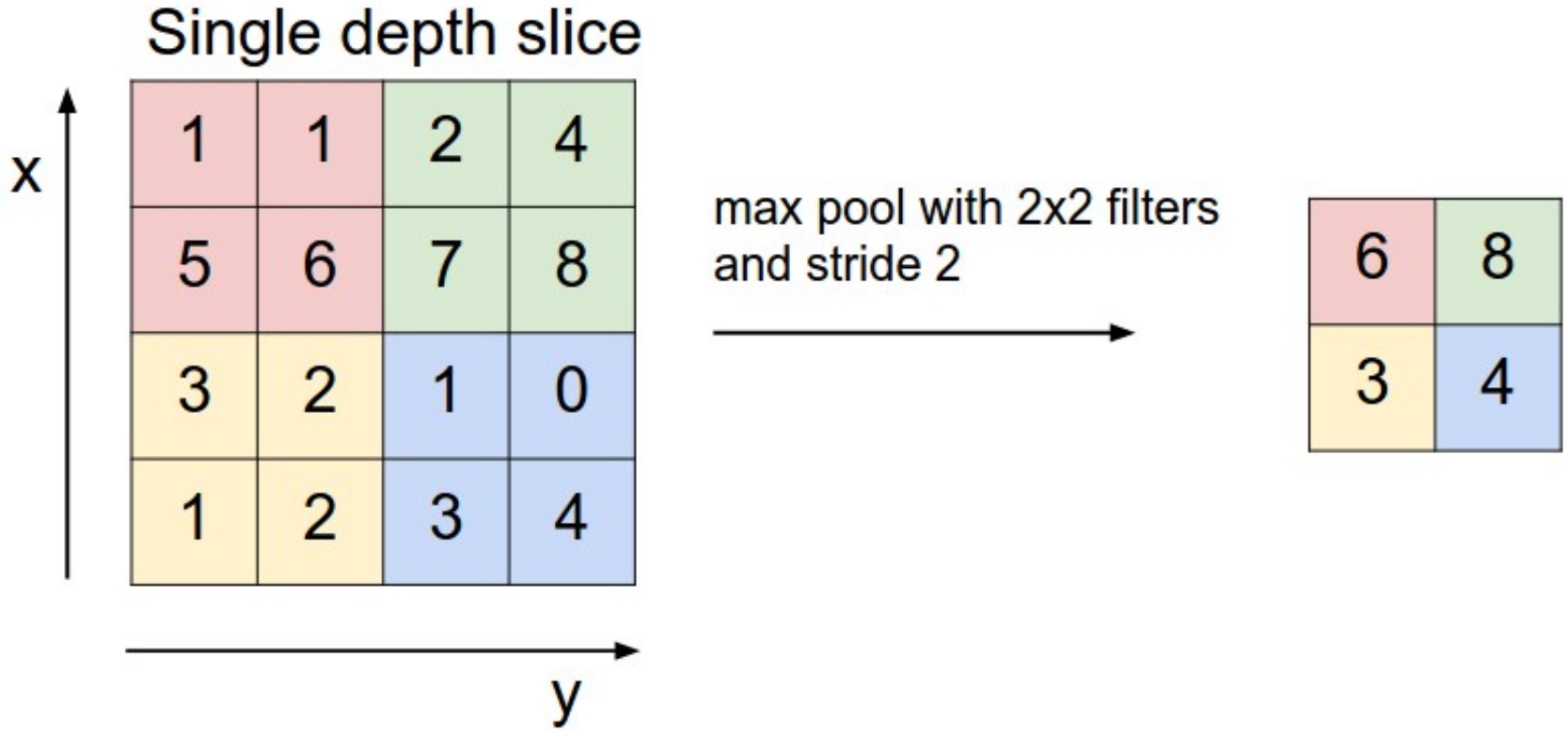
- To be able to consolidate information, and remove out information not useful for the current task

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

- max pooling, average pooling, strided convolutions

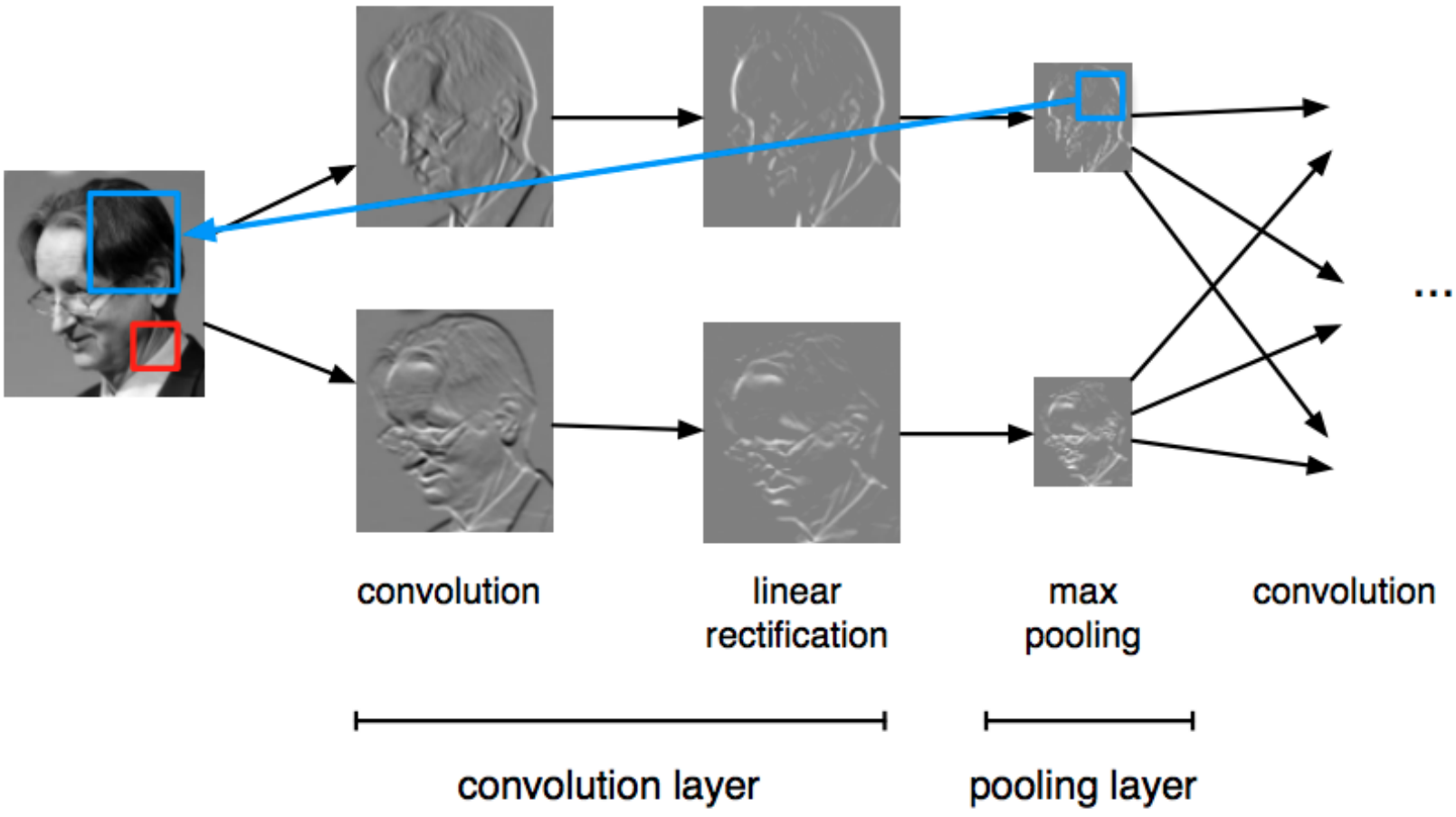
Max-Pooling

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



Max-Pooling Example

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



Max-Pooling in PyTorch

PyTorch has a `nn.MaxPool2d` layer

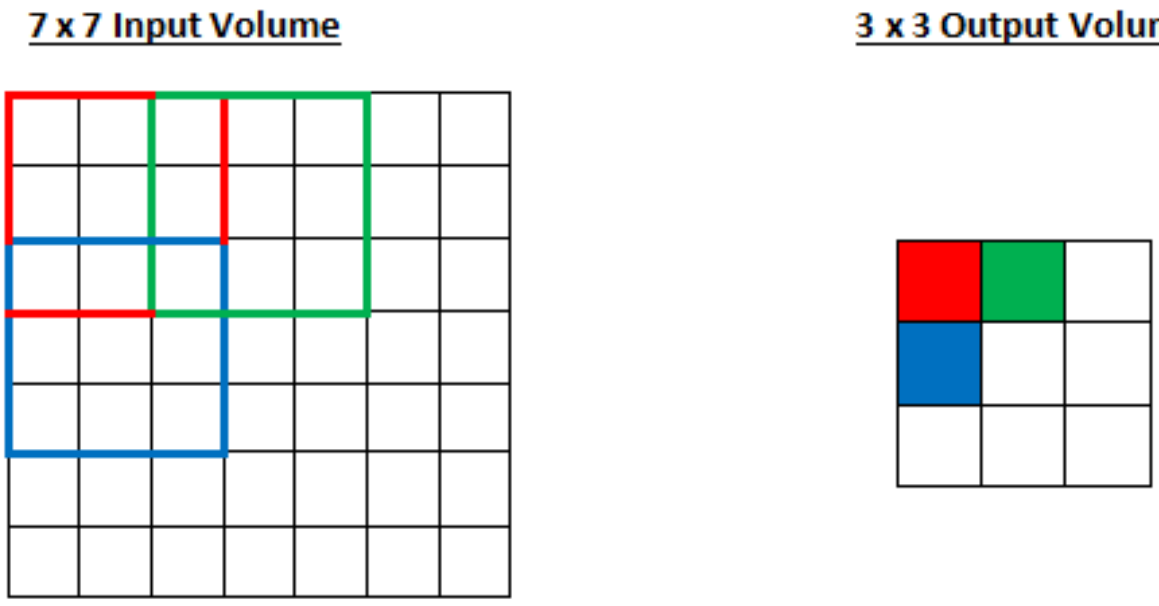
See more at: https://pytorch.org/docs/stable/nn.html#maxpool2d

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

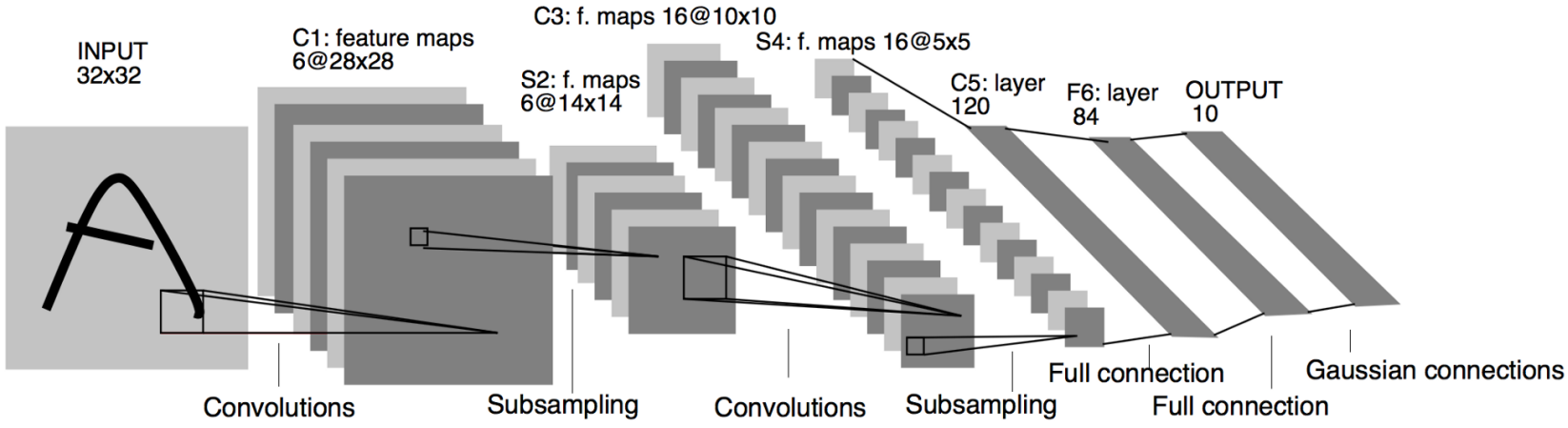
https://arxiv.org/pdf/1603.07285.pdf
https://github.com/vdumoulin/conv_arithmetic

Strided Convolution in PyTorch

PyTorch's `nn.Conv2D` has a `stride` parameter (default 1, you *are* expected to know).

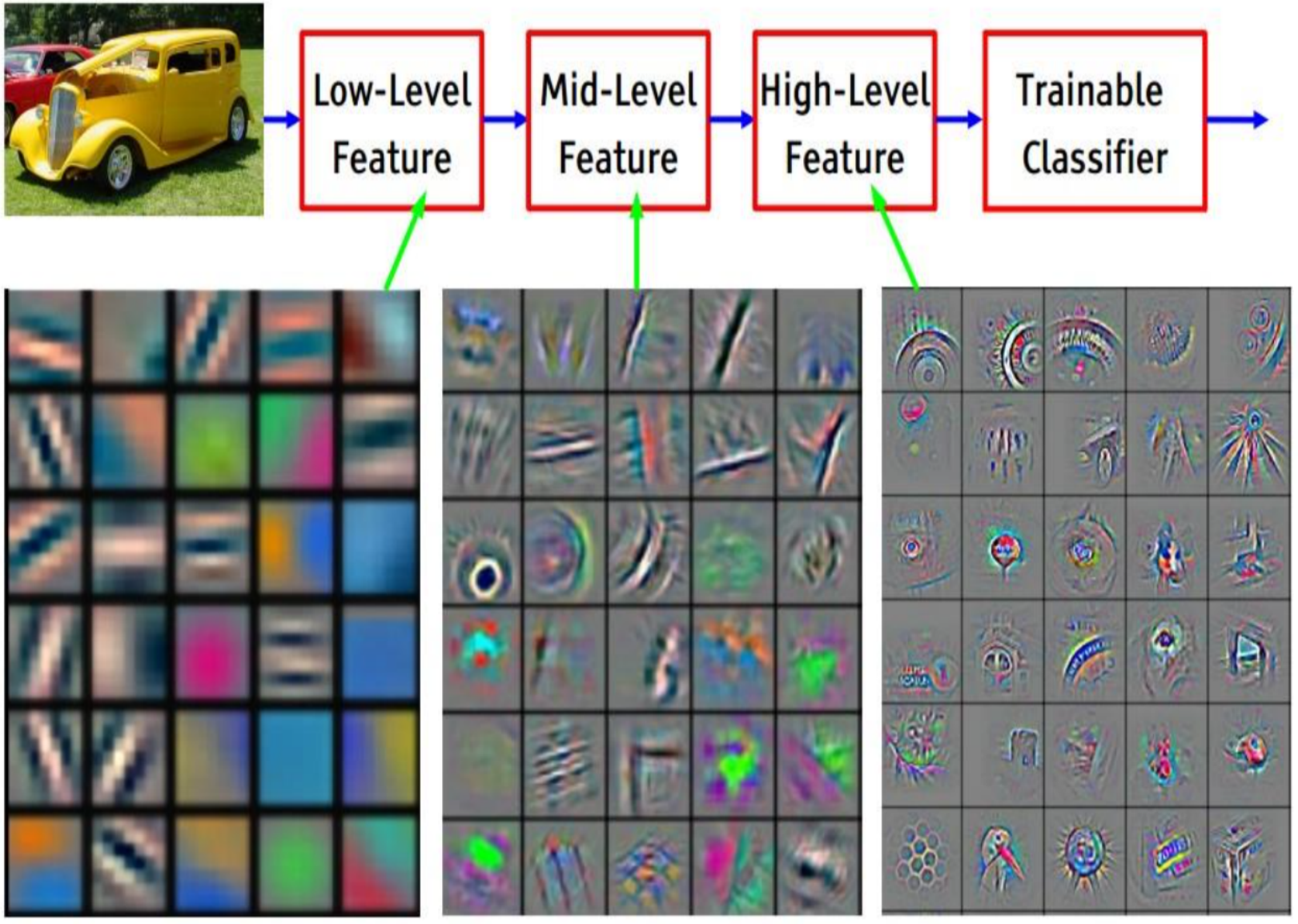
See more at:
https://pytorch.org/docs/stable/nn.html#torch.nn.Conv2d

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]

More Convolutional Networks in PyTorch

Let's look at some more code!

- LargeNet
- AlexNet (Transfer Learning)