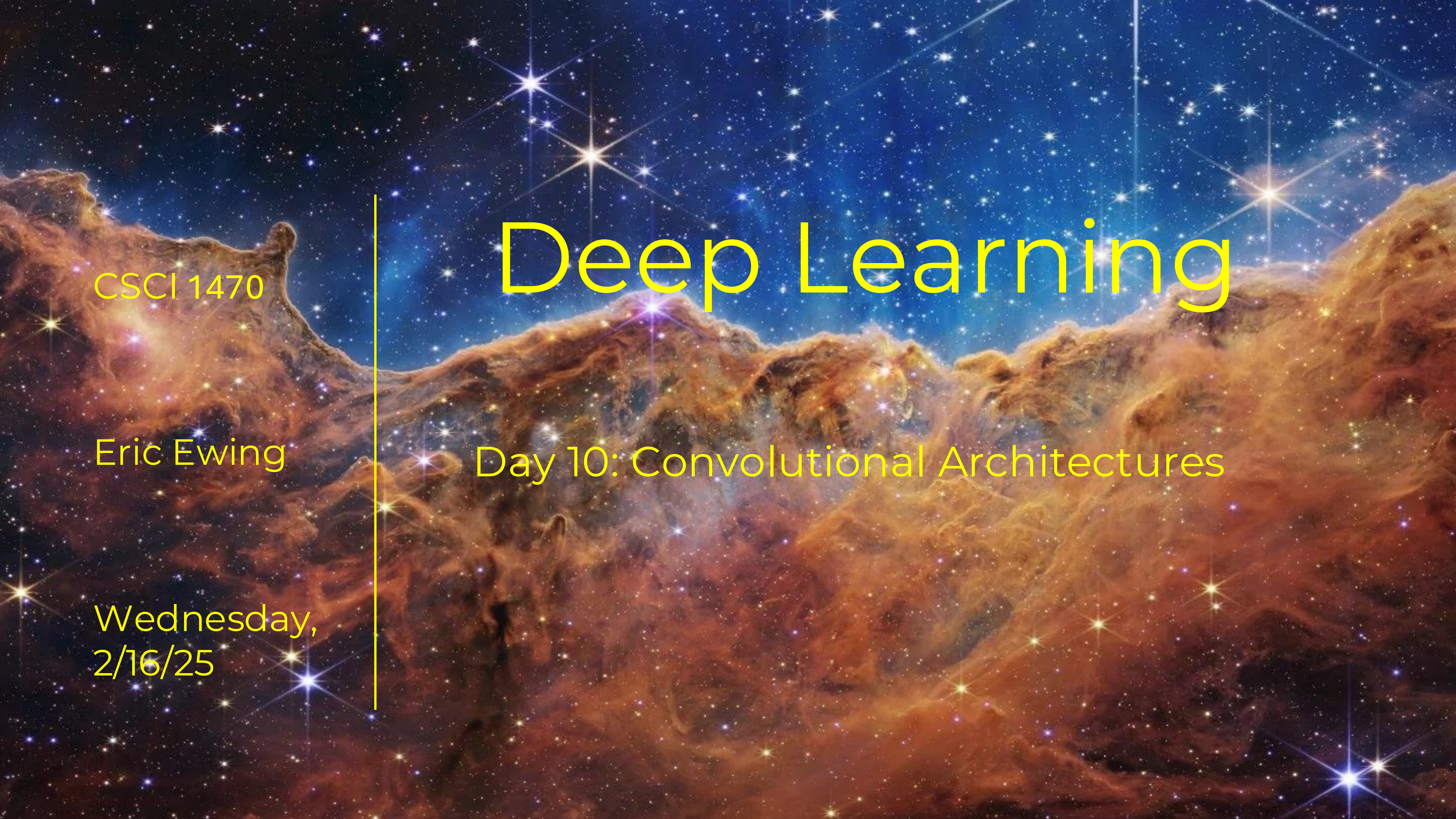


Deep Learning

CSCI 1470

Day 10: Convolutional Architectures

Eric Ewing

Wednesday, 2/16/25

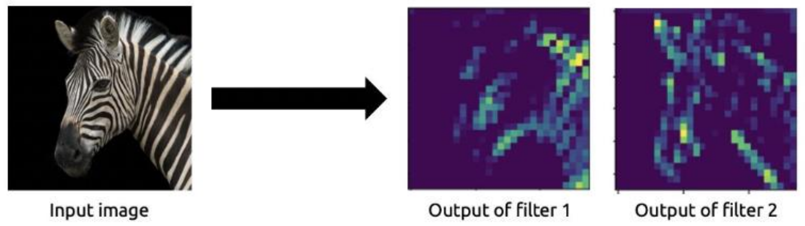
Logistics

- Beras Conceptual due 10pm tonight
- Two New Workshops
    - How to use GPUs/CUDA: How to accelerate code with GPUs? What GPU resources are available to you? How do you actually use those resources? All questions that may help you on your final projects.
    - Math of DL:

Recap

Convolution

- Filters/Kernels and Stride
- Learning filters
- CNNs are partially connected networks



Convolution in Tensorflow

- Tensorflow conv2d function

```
tf.nn.conv2d(input, filter, strides, padding)
```

Input Image (4-D Tensor)

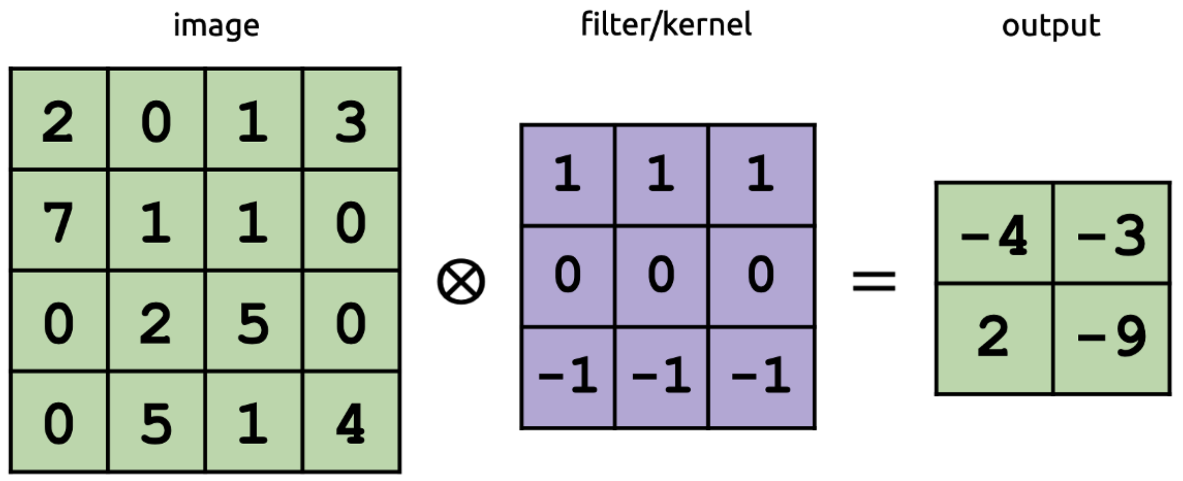
Filter/Kernel (4-D Tensor)

Strides along each dimension

Type of Padding (String "Valid" or "Same")

Convolutions in Tensorflow

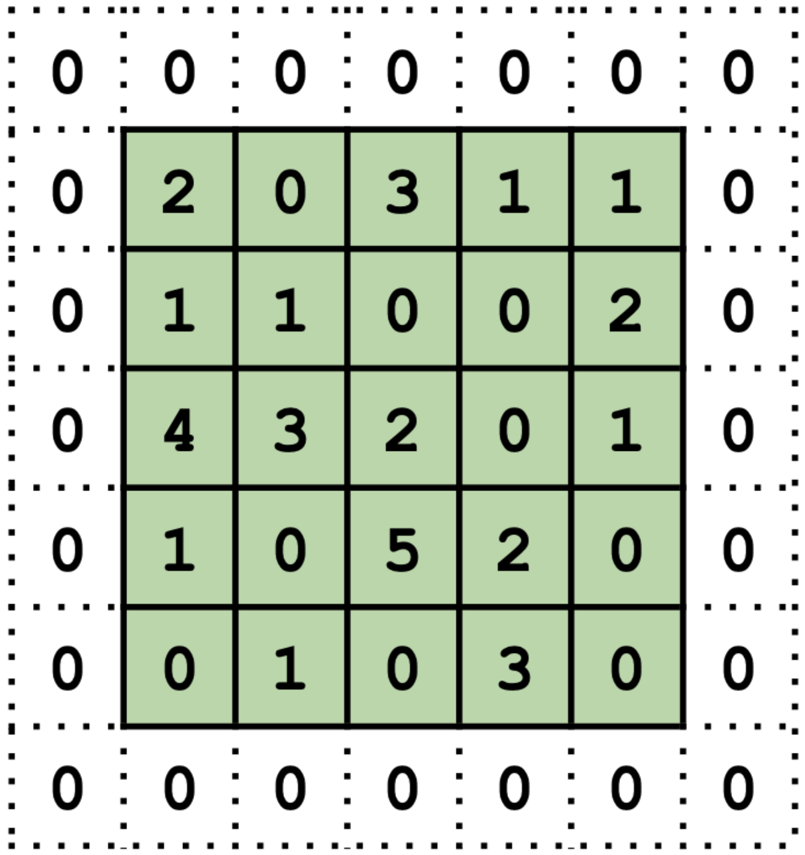
tf.nn.conv2d(input, filter, stride, padding)



Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d

What Values to Use For These Pixels?

Standard practice: fill with zeroes



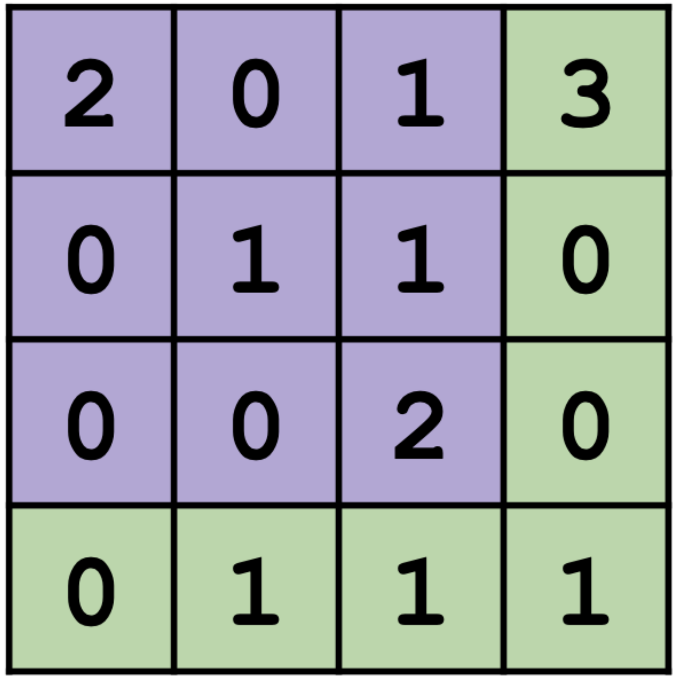
0	0	0	0	0	0	0
0	2	0	3	1	1	0
0	1	1	0	0	2	0
0	4	3	2	0	1	0
0	1	0	5	2	0	0
0	0	1	0	3	0	0
0	0	0	0	0	0	0

Padding Modes in Tensorflow

2 available options: ‘VALID’ and ‘SAME’:

Valid

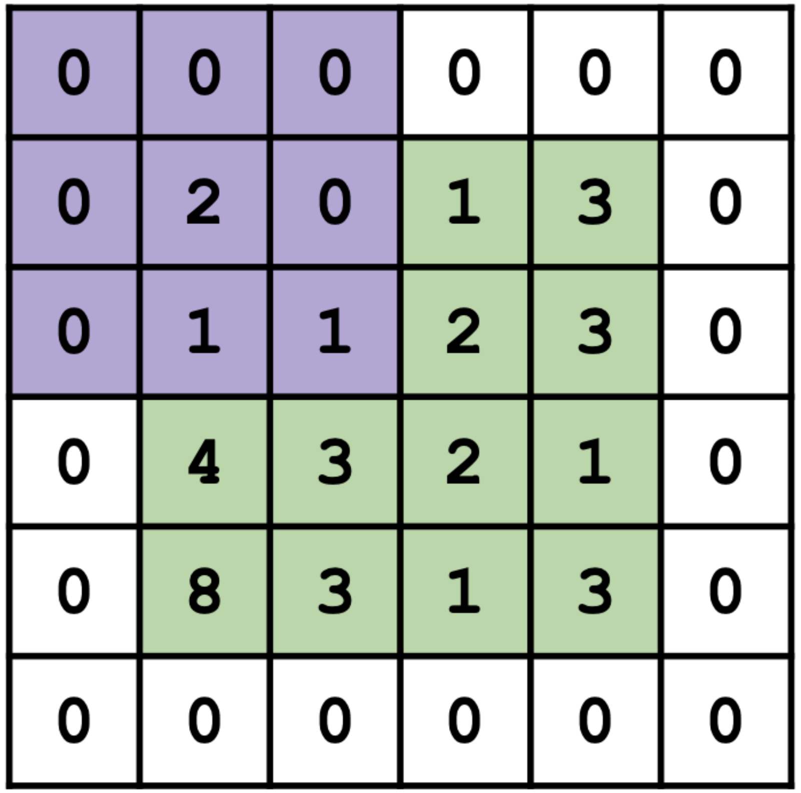
Filter only slides over “Valid” regions of the data



2	0	1	3
0	1	1	0
0	0	2	0
0	1	1	1

Same

Filter slides over the bounds of the data, ensuring output size is the “Same” as input size (when stride = 1)

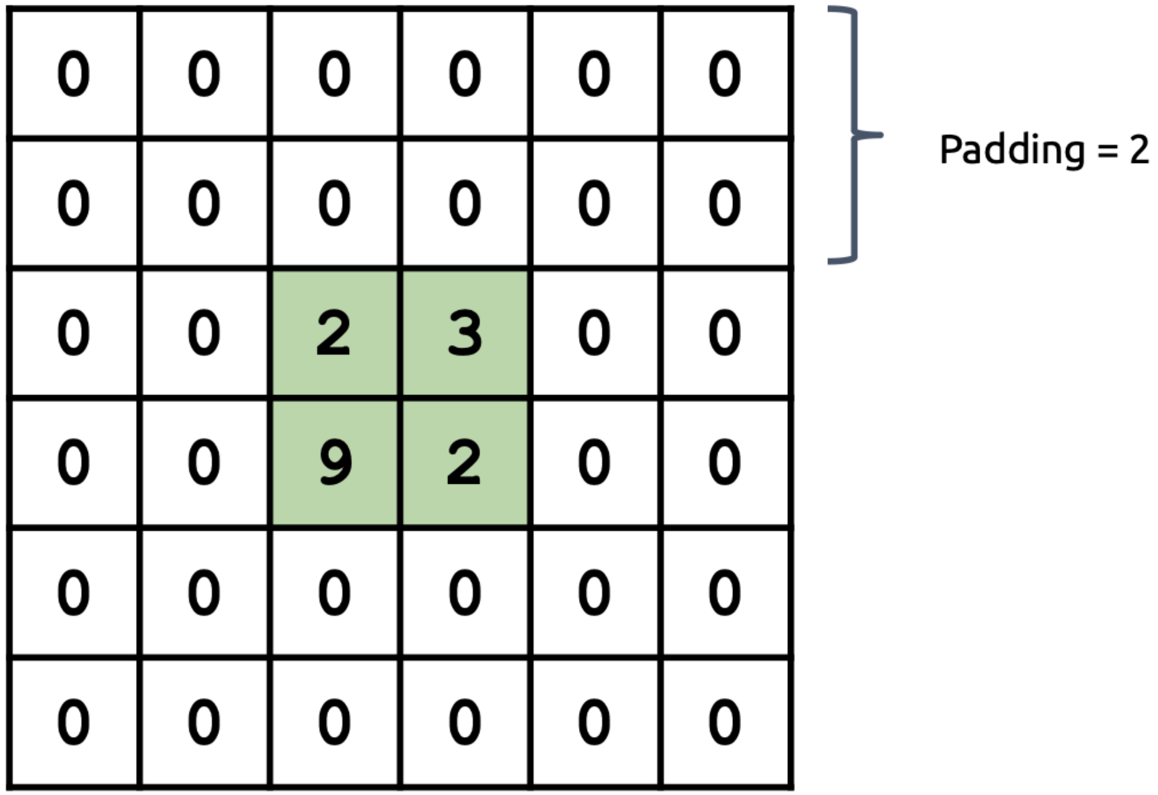


0	0	0	0	0	0
0	2	0	1	3	0
0	1	1	2	3	0
0	4	3	2	1	0
0	8	3	1	3	0
0	0	0	0	0	0

Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

- Number of filters, **N**
- The size of these filters, **F**
- The stride, **S**
- The amount of padding, **P**



0	0	0	0	0	0
0	0	0	0	0	0
0	0	2	3	0	0
0	0	9	2	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Output Size of a Convolution Layer

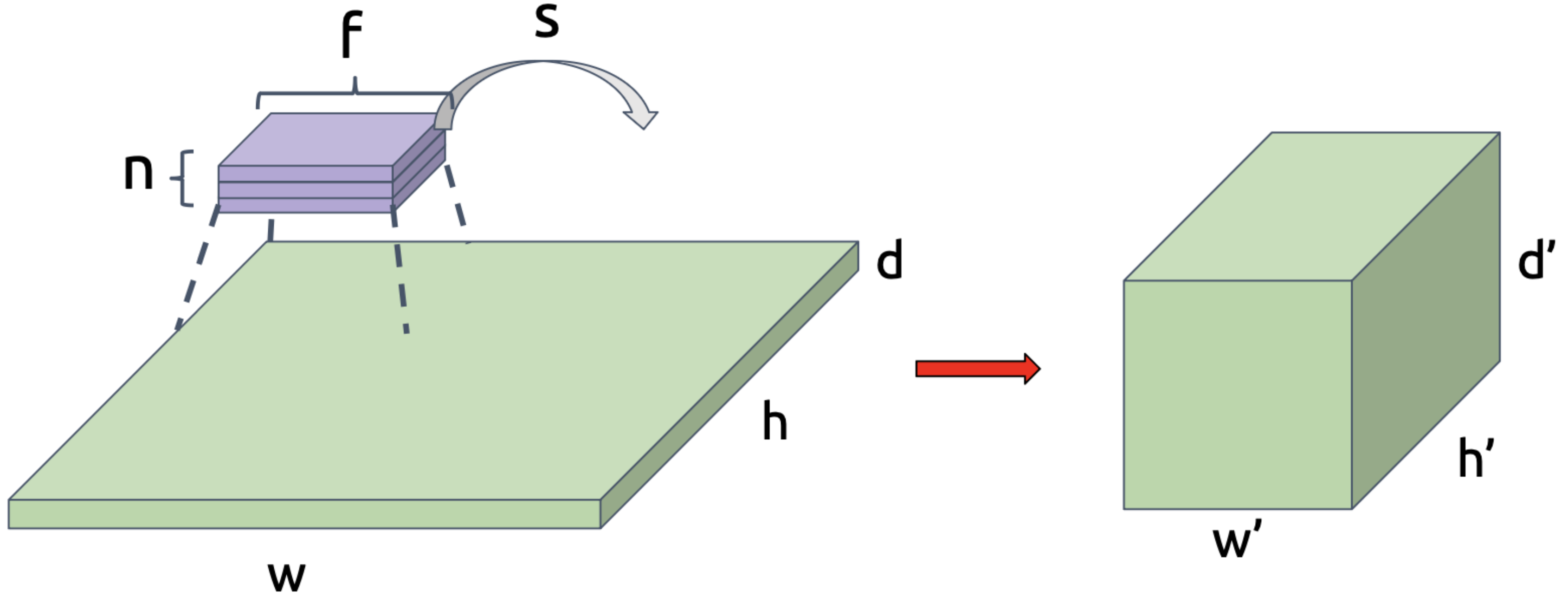
Suppose we know the number of filters, their size, the stride, and padding (**n,f,s,p**).

Then for a convolution layer with input dimension **w x h x d**, the output dimensions **w' x h' x d'** are:

$$w' = \frac{w - f + 2p}{s} + 1$$

$$h' = \frac{h - f + 2p}{s} + 1$$

$$d' = n$$



Output Size for “VALID” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

Let $w = 4$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 0

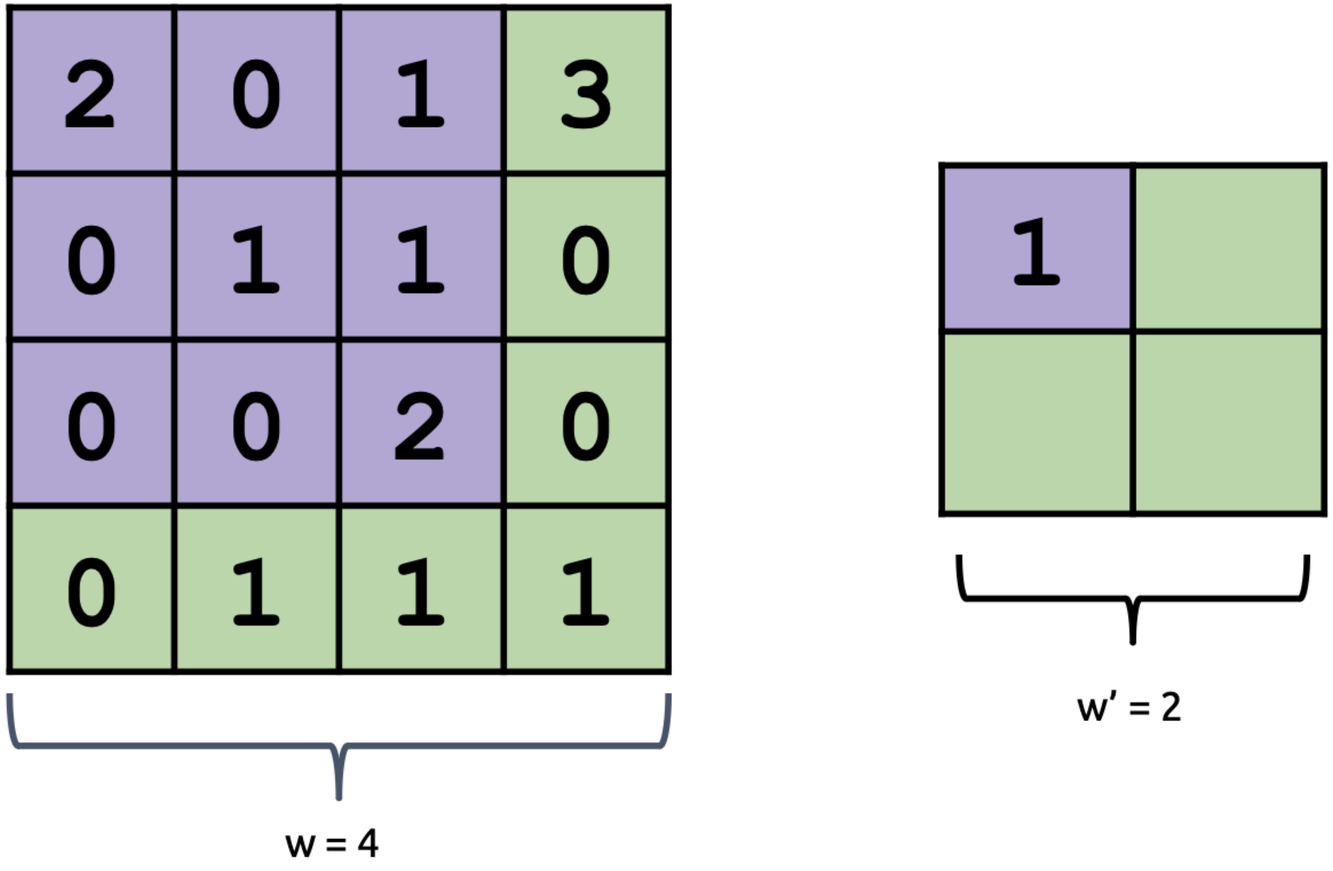
$$w' = \frac{4 - 3 + 2 \cdot 0}{1} + 1$$

$$= 1 + 1 = 2$$

Output Size for "VALID" Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

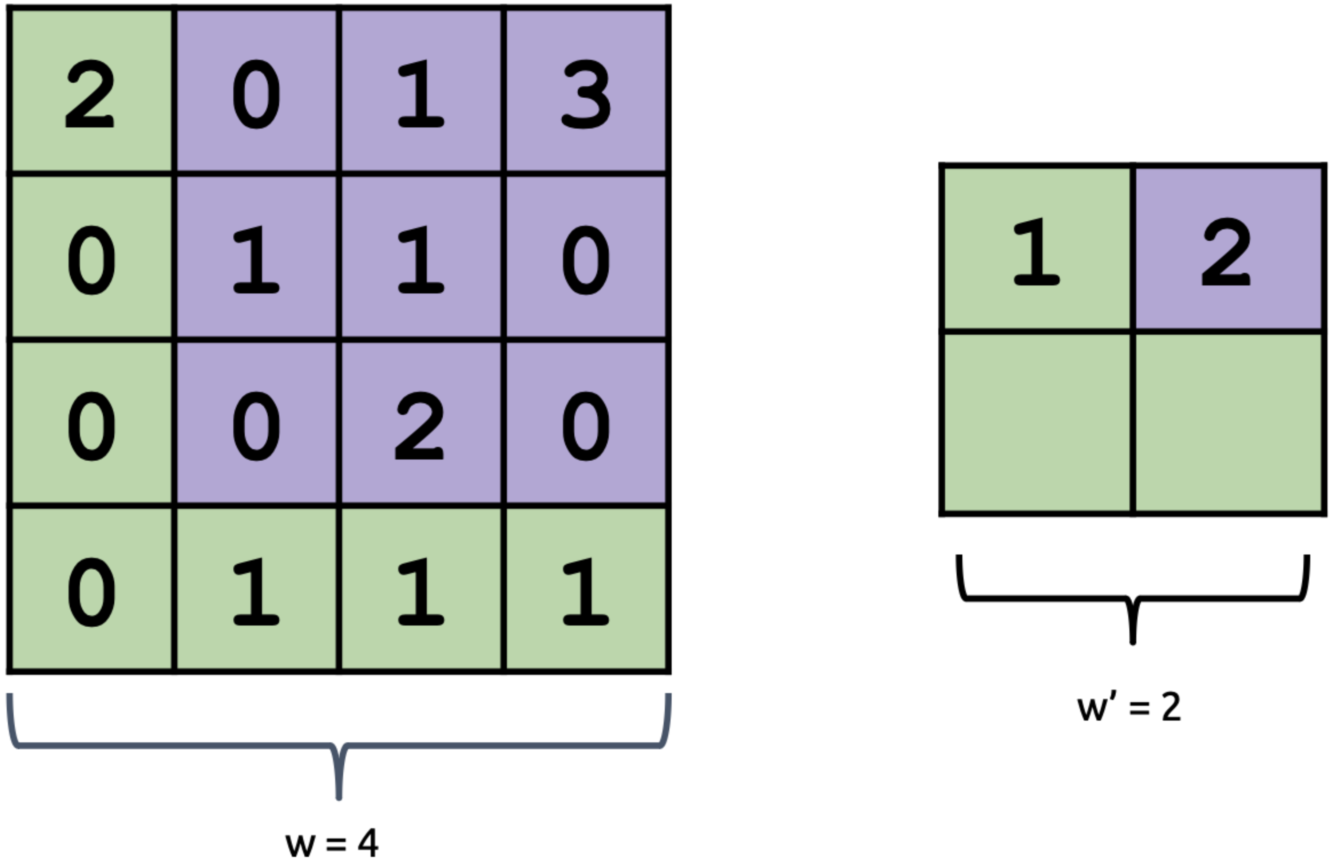
num filters n = 1
filter size f = 3
stride s = 1
padding p = 0



Output Size for “VALID” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

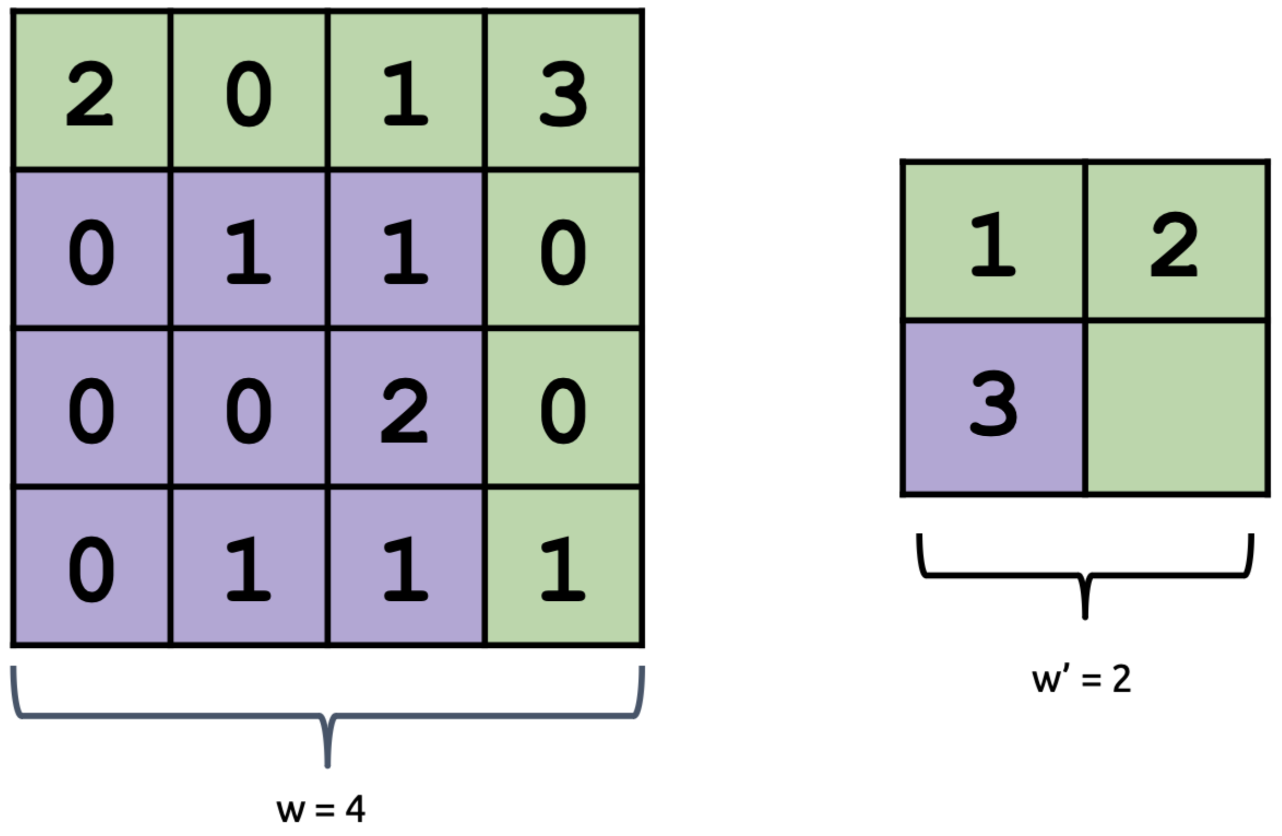
num filters n = 1
filter size f = 3
stride s = 1
padding p = 0



Output Size for “VALID” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

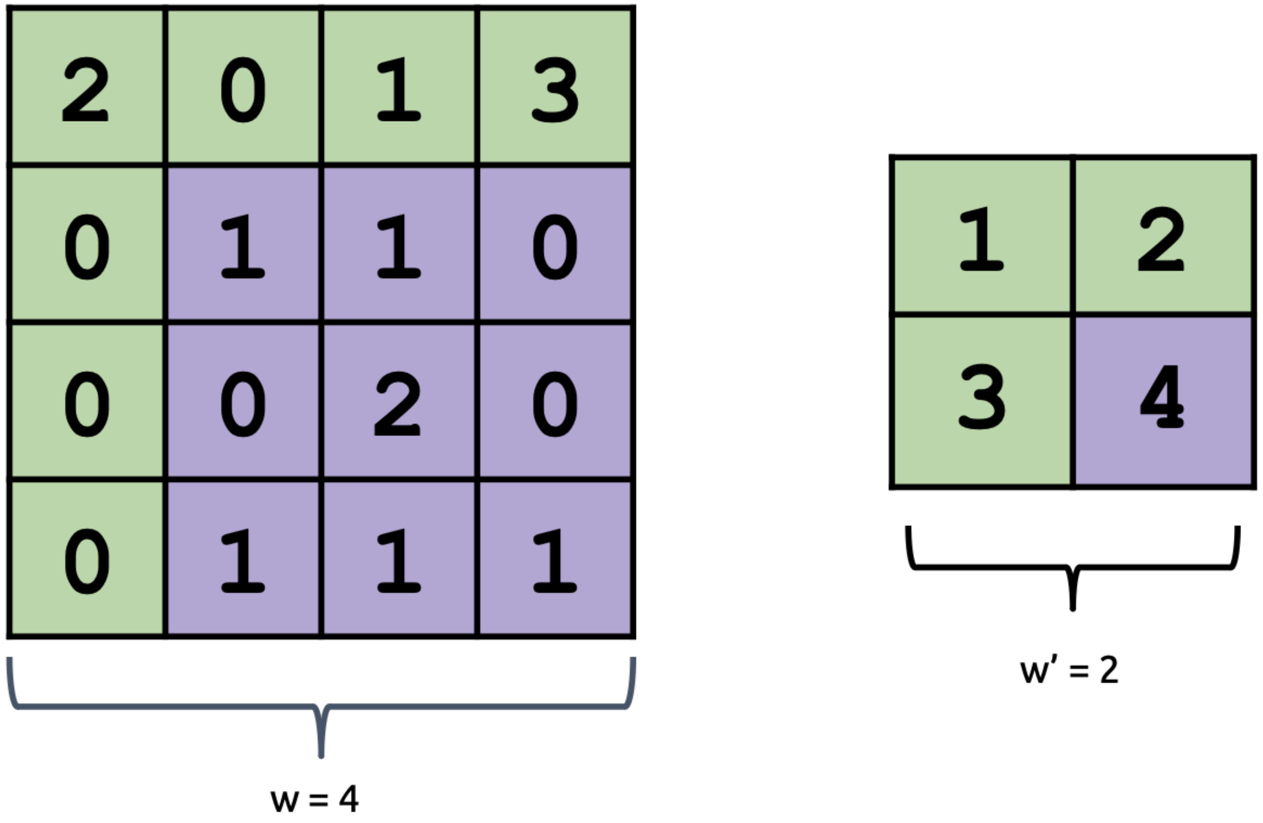
num filters n = 1
filter size f = 3
stride s = 1
padding p = 0



Output Size for “VALID” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 0



Output Size for “SAME” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

Let $w = 4$

num filters n = 1
filter size f = 3
stride s = 1
padding p = ??

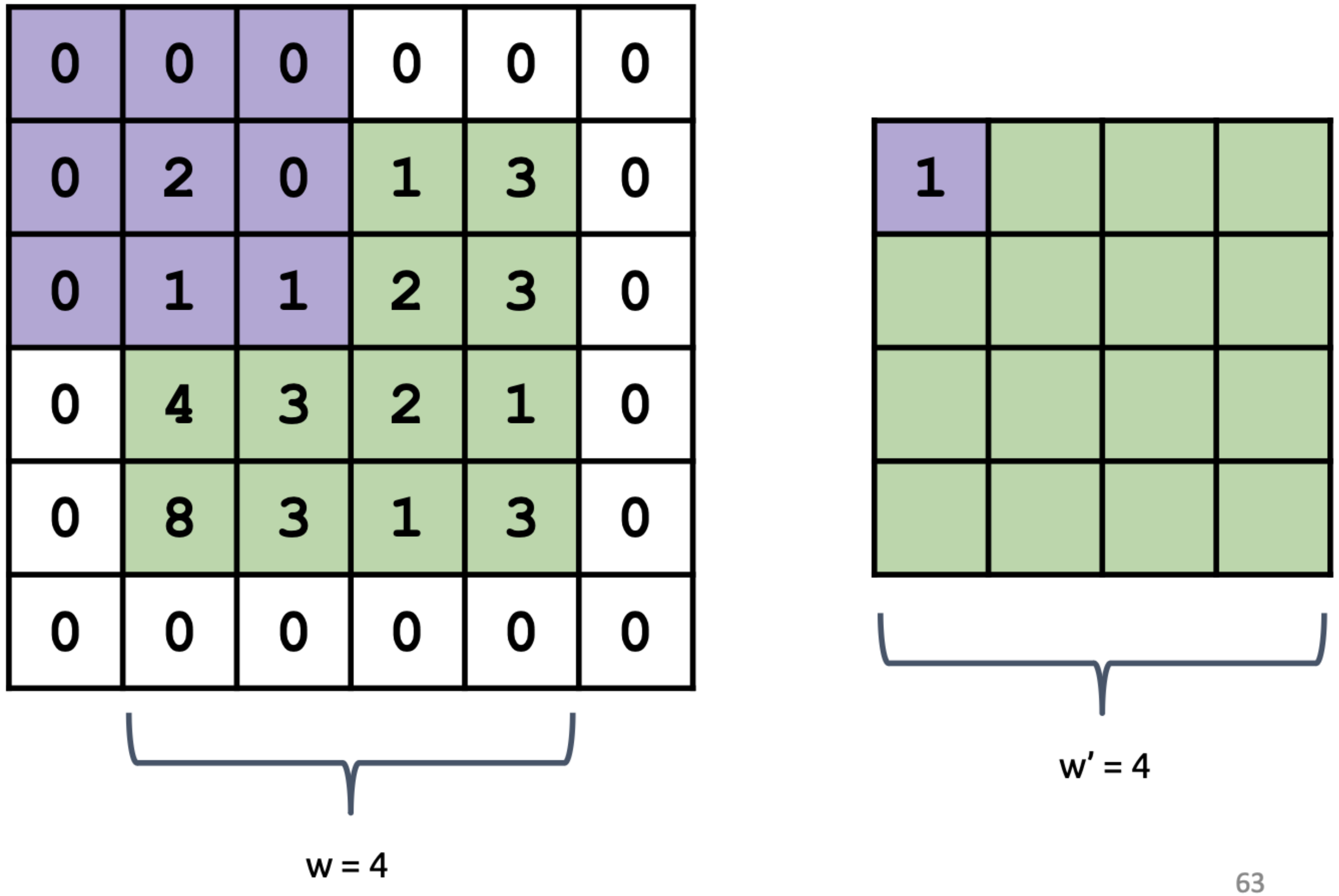
Padding size needs to be determined

Output Size for "SAME" Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined

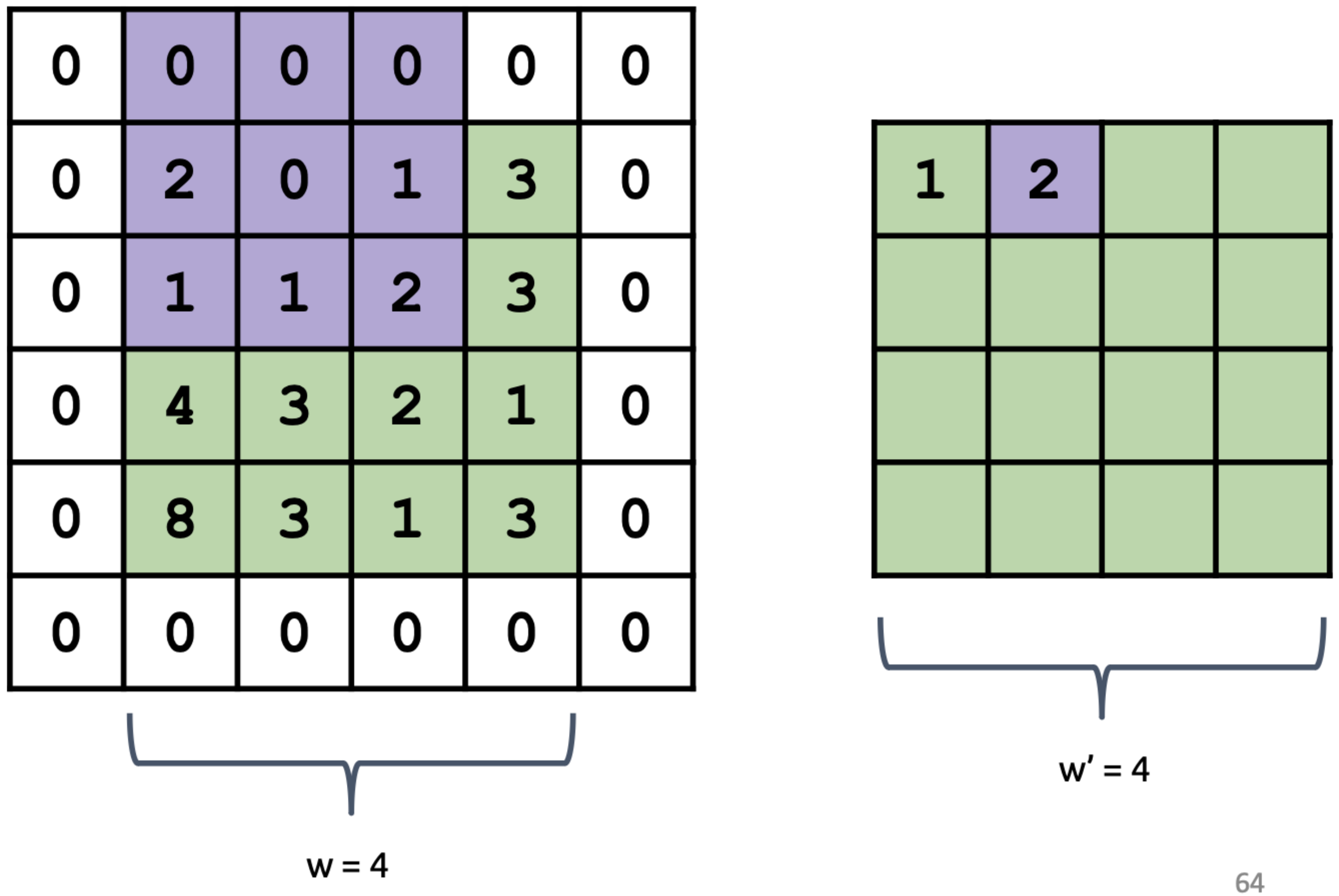


Output Size for "SAME" Padding

$$w' = \frac{w\ - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined

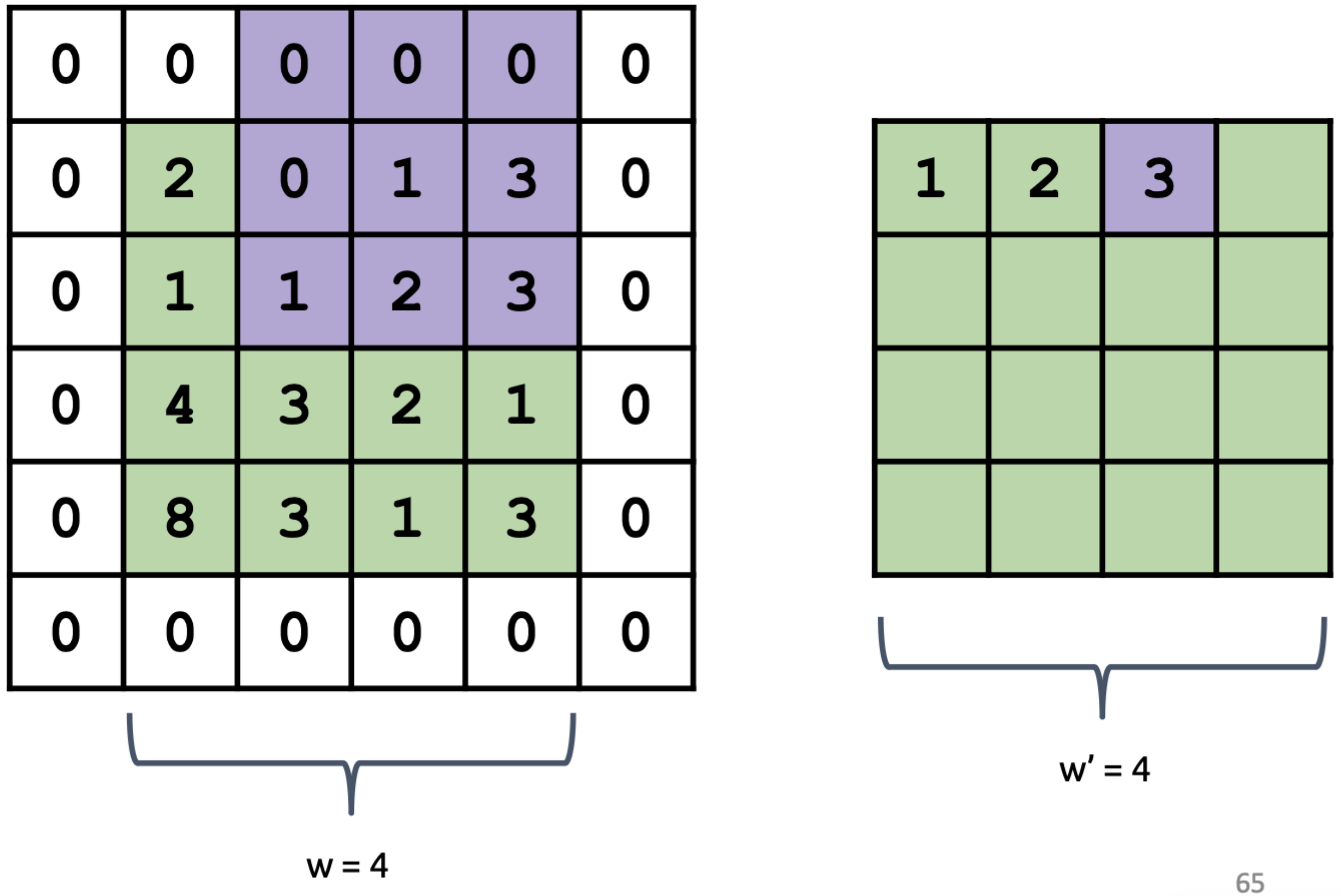


Output Size for "SAME" Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined



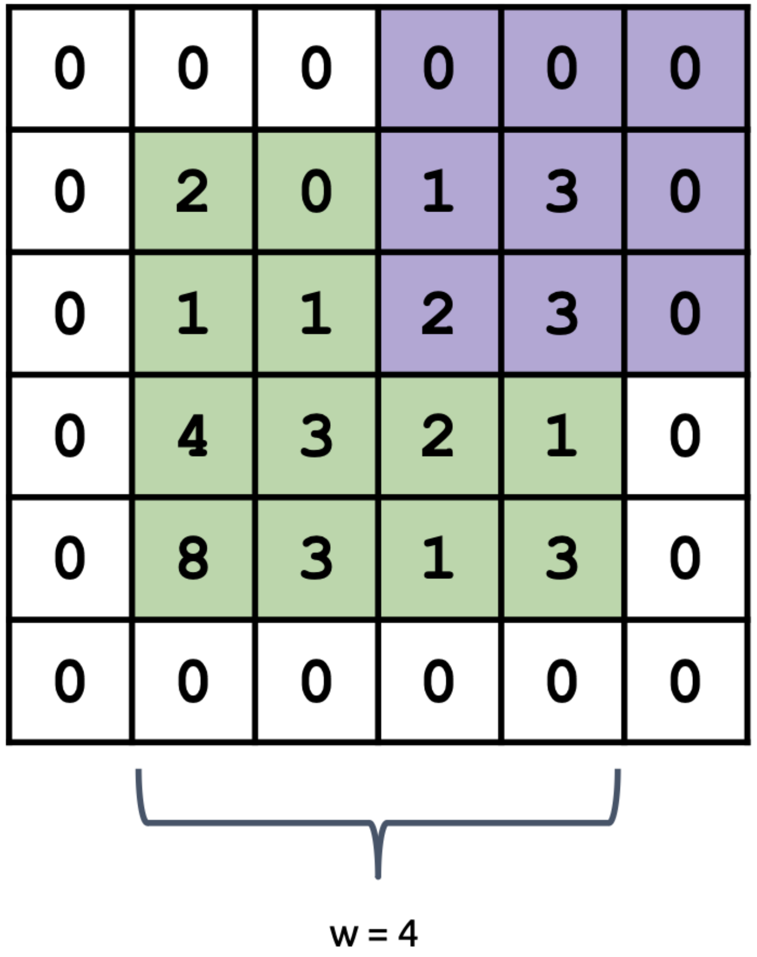
Any questions?

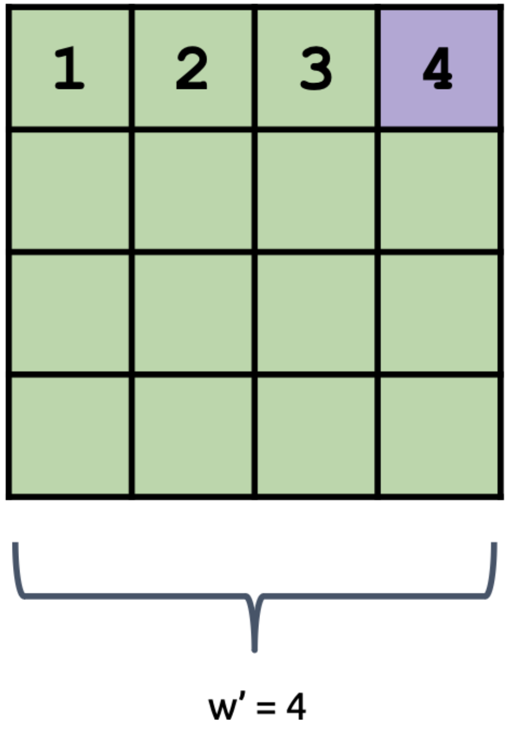
Output Size for “SAME” Padding

$$w' = \frac{w \; - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined

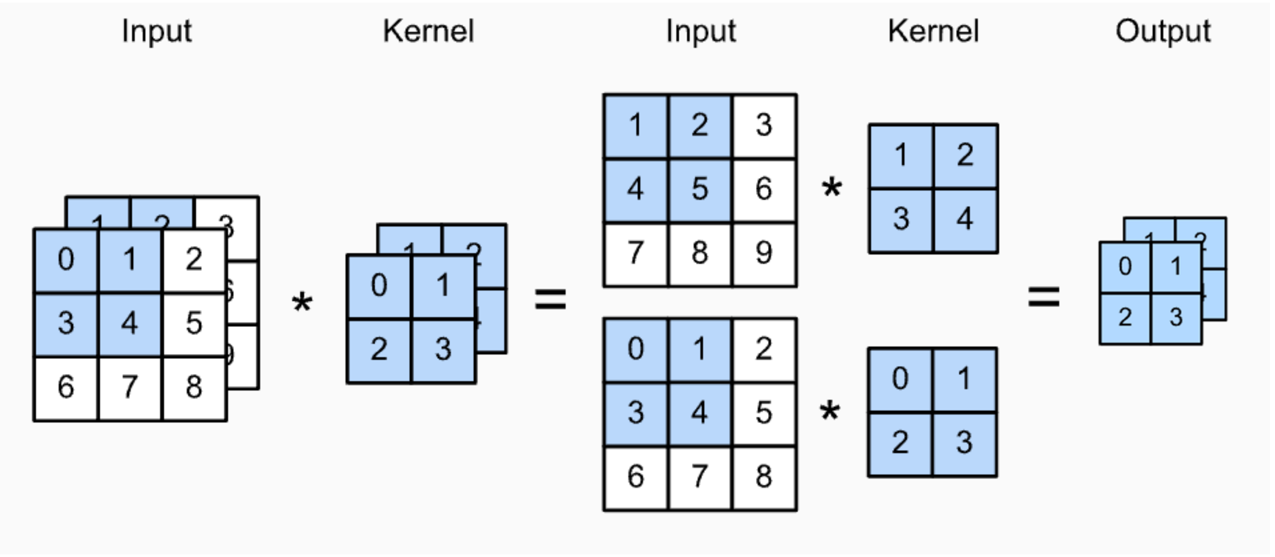




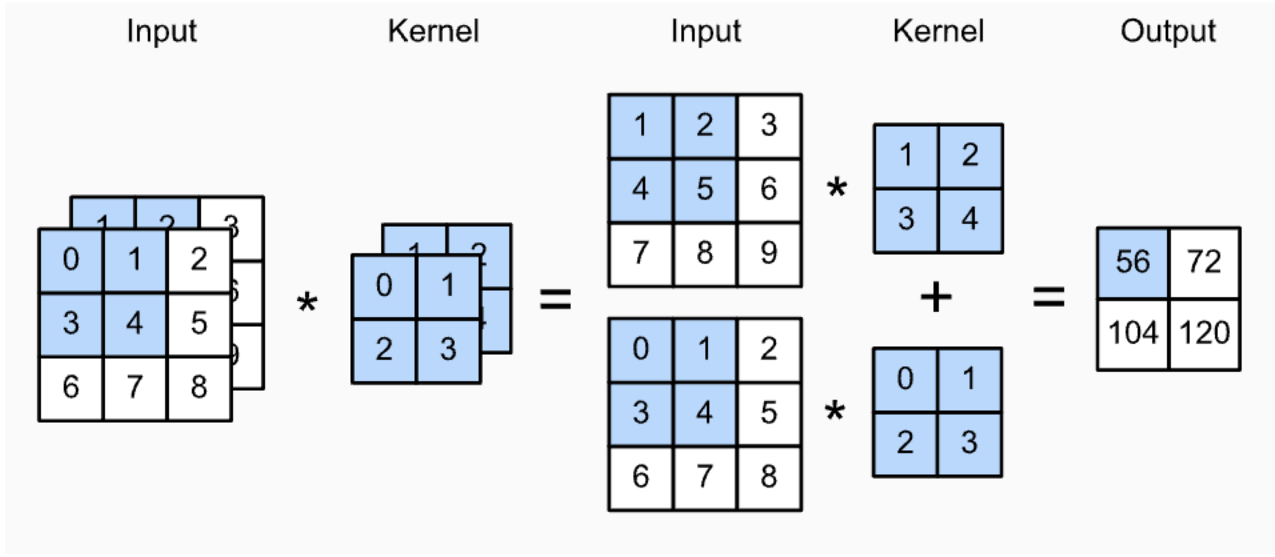
Multi-Channel Input

Which makes more sense?

Option #1
n channels to n outputs



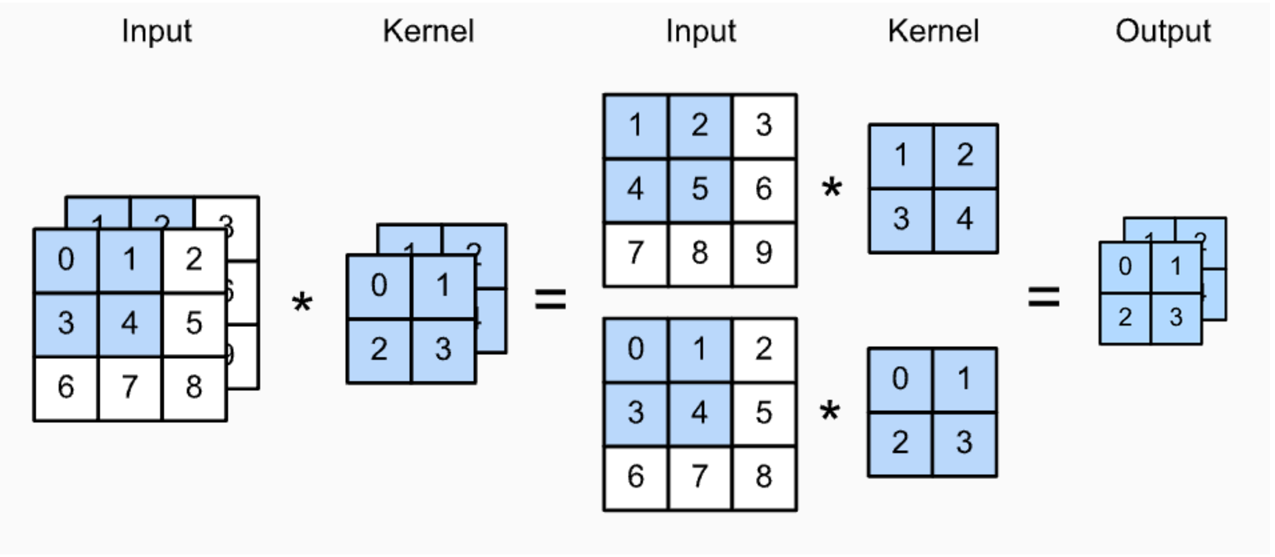
Option #2
N channels to 1 output



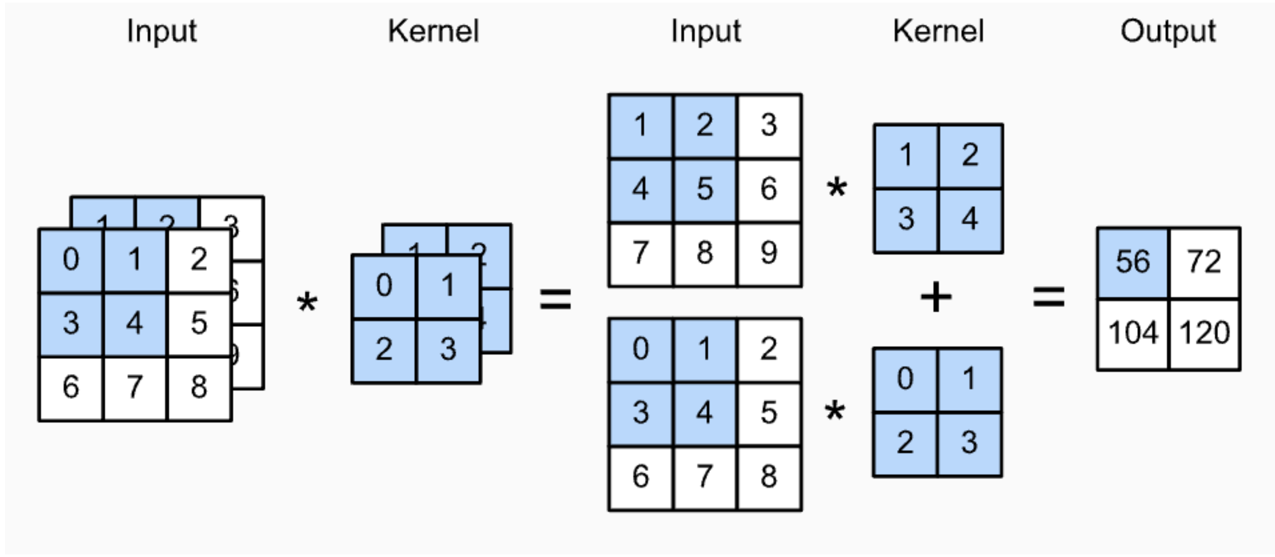
Multi-Channel Input

N-channels to 1 output allows information from separate channels to be used together

Option #1
n channels to n outputs



Option #2
N channels to 1 output



Today’s Goals

(1) What non-linear activation functions are available to us?

(2) Learn about Convolutional Architectures

    (1) Many more decisions to make about structure of network than MLPs

Bias Term in Convolution Layers

$$\left[\begin{array}{|c|c|c|c|}\hline 2 & 0 & 3 & 1 \\ \hline 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 2 & 0 \\ \hline 1 & 0 & 1 & 2 \\ \hline \end{array} \underset{\substack{\text{“VALID”} \\ \text{Stride = 1}}}{\otimes} \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \right] + \overset{\text{Bias}}{\begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array}}$$

Just like a fully connected layer, we can have a learnable additive bias for convolution.

Adding a Bias in Tensorflow

If you use tf.nn.conv2d, bias can be added with:

```
tf.nn.bias_add(value, bias)
```

Conv2D output

Bias variable to add
e.g.
`tf.Variable(tf.random.normal([16]))`
for a conv2d output with 16 channels

Full documentation here: https://www.tensorflow.org/api_docs/python/tf/nn/bias_add

Adding a Bias in Tensorflow

If you are using keras layers, bias is included by default:

```
tf.keras.layers.Conv2D(filters, kernel_sz, strides, padding, use_bias = True)
```

Number of filters

Filter Size

Strides along each dimension

Type of Padding (VALID or SAME)

Full documentation here: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D

Activation Functions

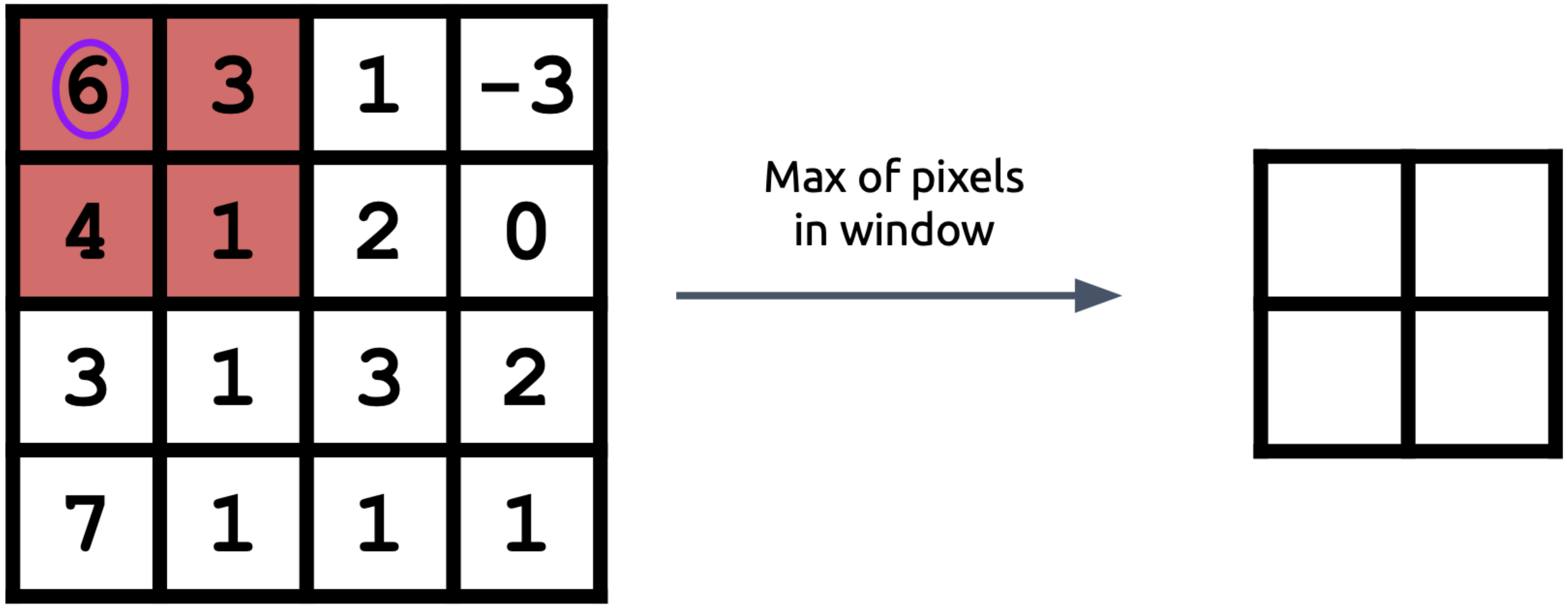
Remember, a linear combination of features, even if repeated many times, will always be linear.

Still need some type of non-linear activation (e.g., ReLUs)

We also have other convolution-specific activation functions called “pooling” operations

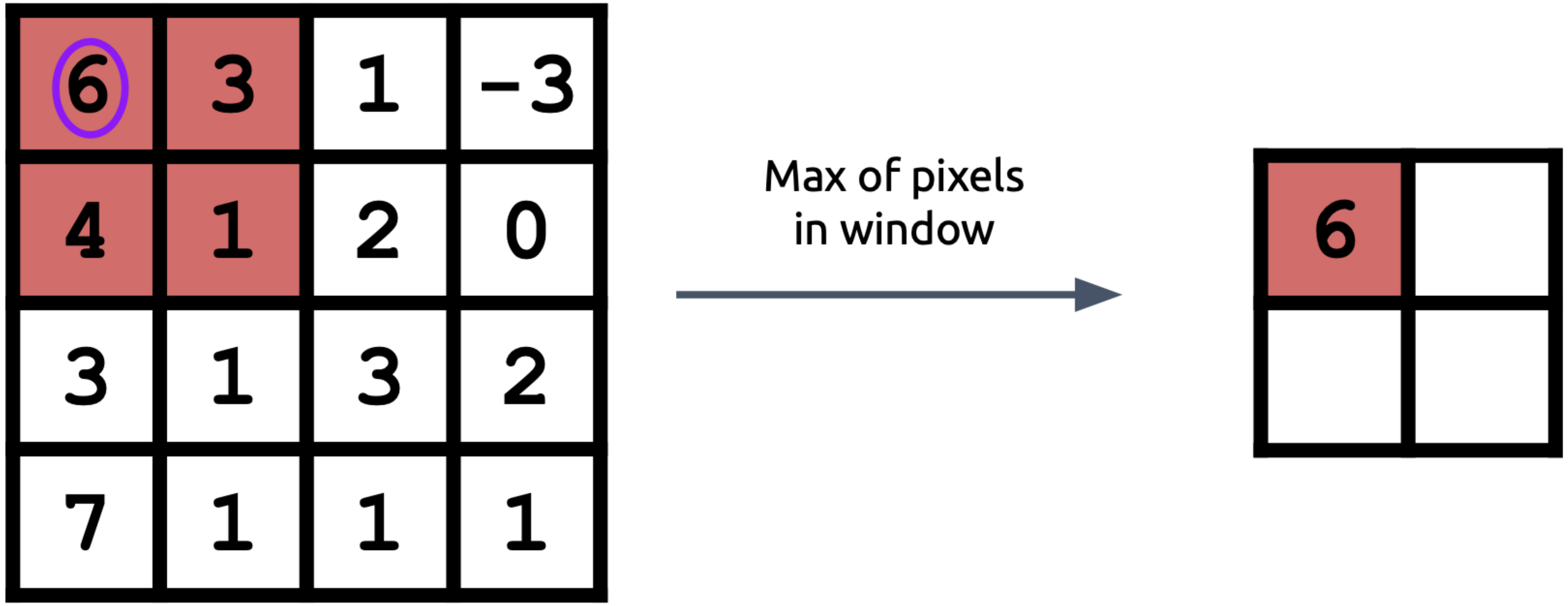
Max Pooling

Max pooling with stride 2 and 2x2 filters



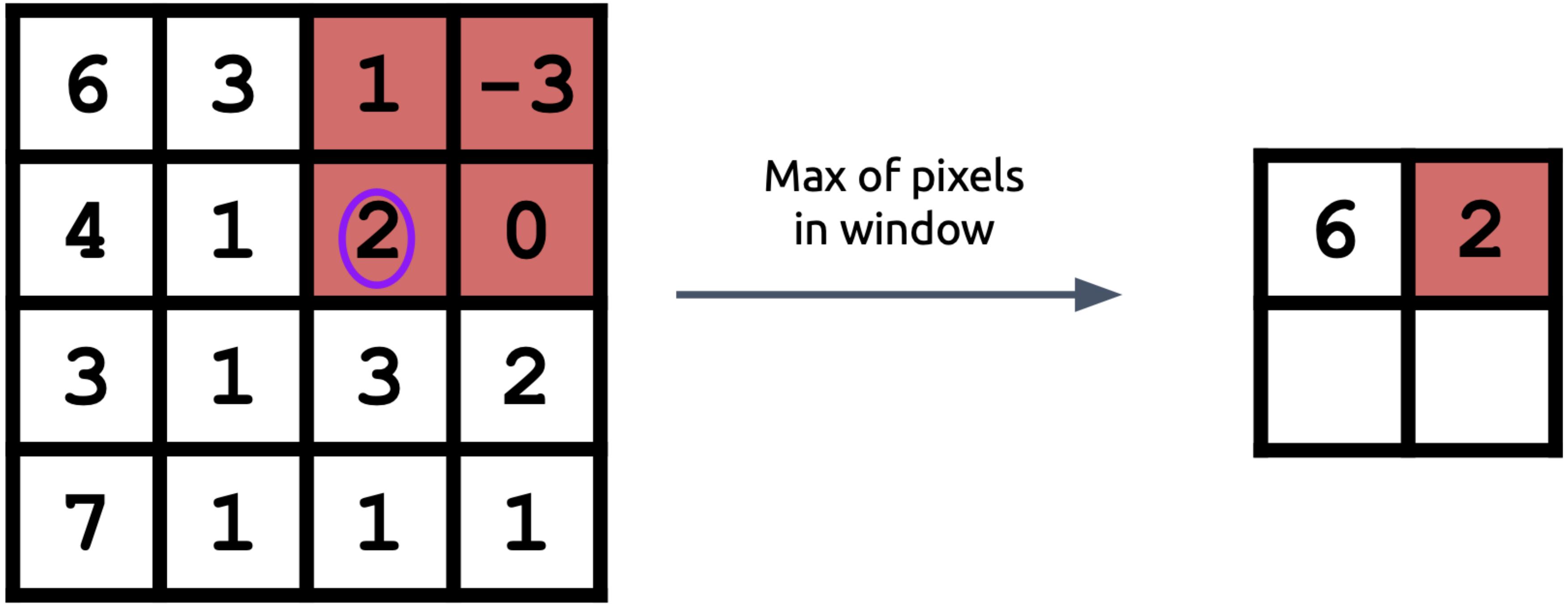
Max Pooling

Max pooling with stride 2 and 2x2 filters



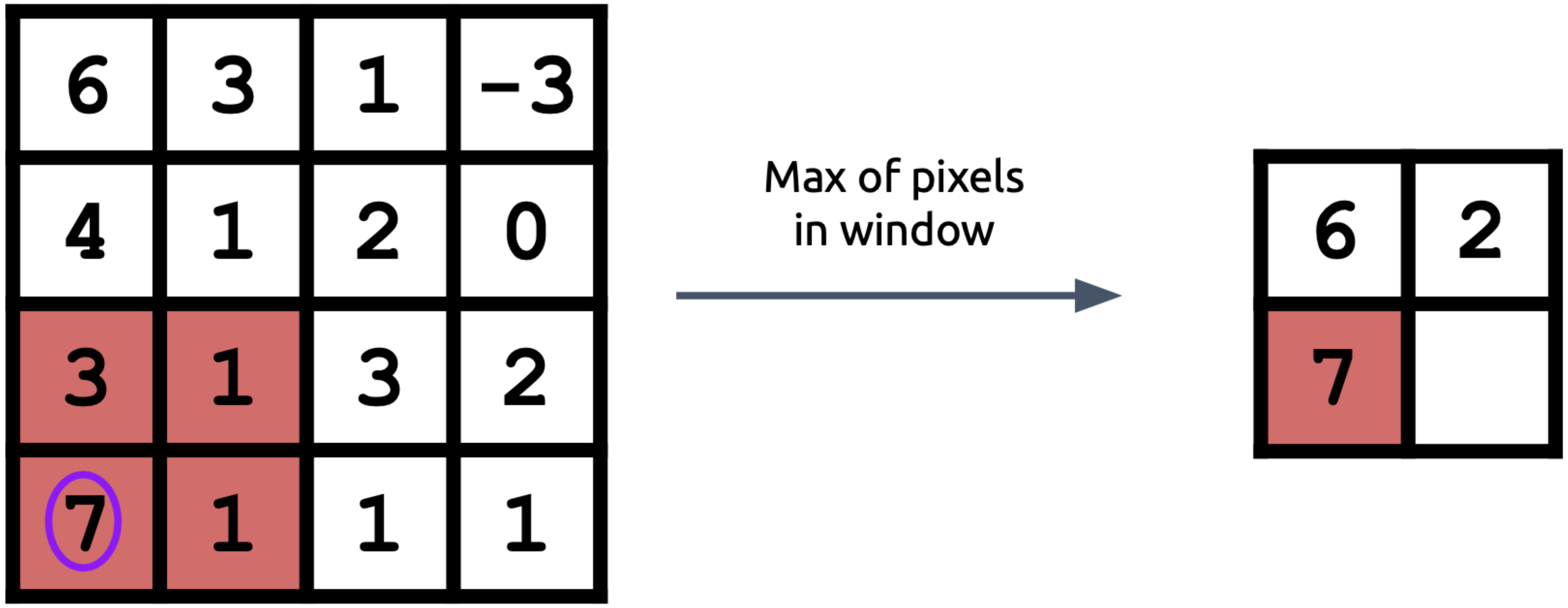
Max Pooling

Max pooling with stride 2 and 2x2 filters



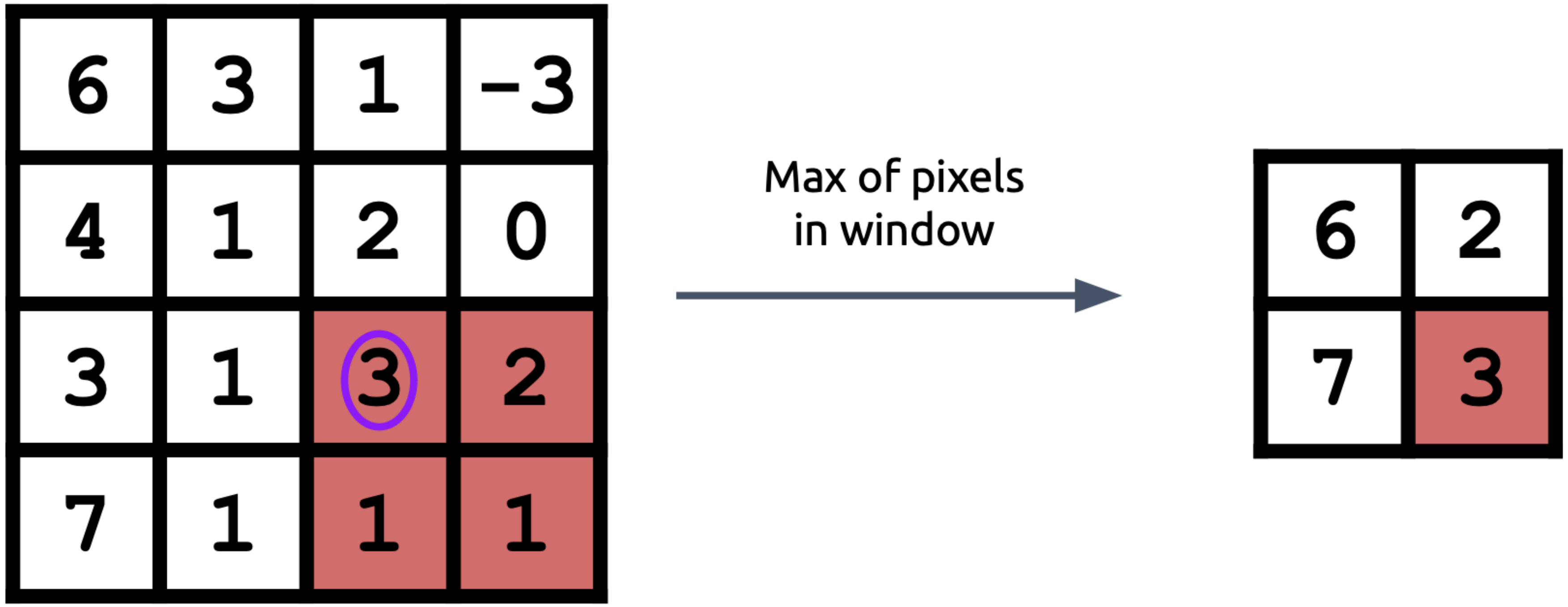
Max Pooling

Max pooling with stride 2 and 2x2 filters



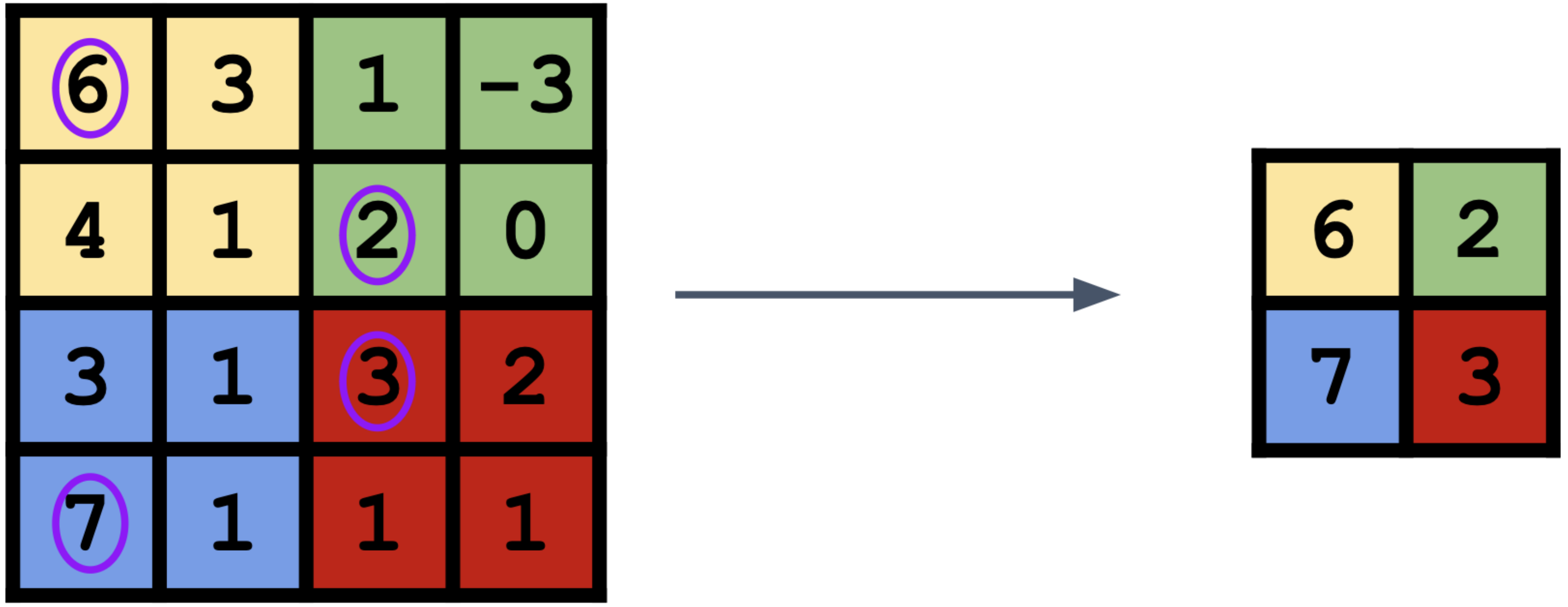
Max Pooling

Max pooling with stride 2 and 2x2 filters



Max Pooling

Max pooling with stride 2 and 2x2 filters



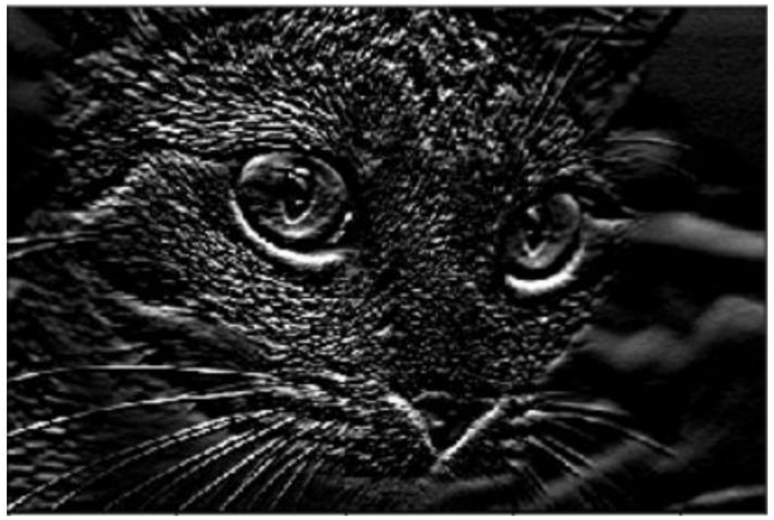
Why use Max Pooling?

Pooling: Motivation

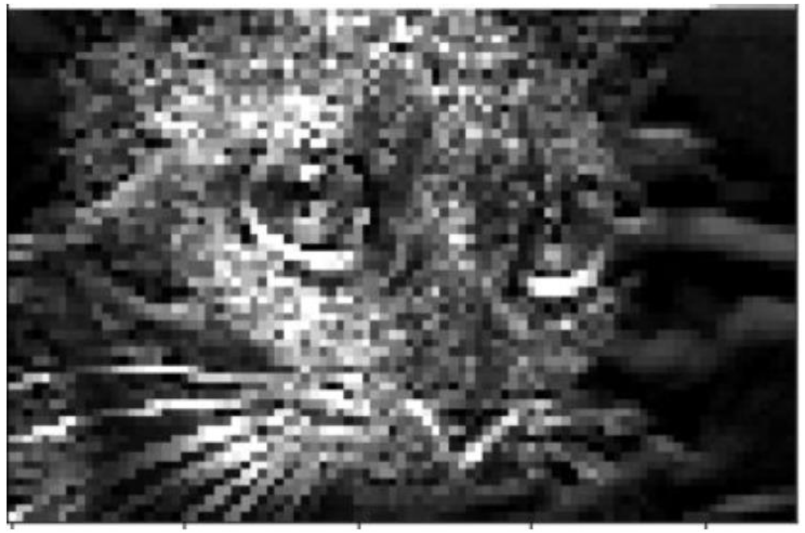
Max Pooling

- Keeps track of regions with highest activations, indicating object presence
- Controllable way to lower (coarser) resolution (down sample the convolution output)

Original Image



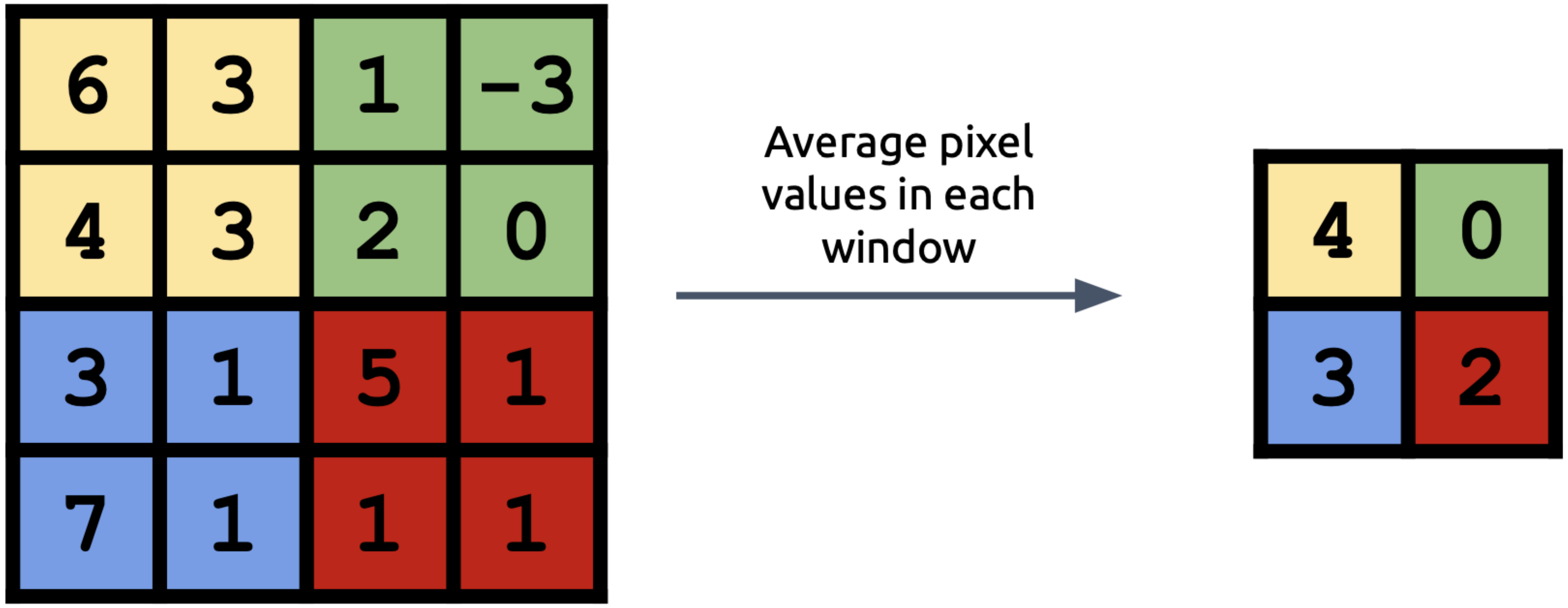
Convolution Output



After Pooling

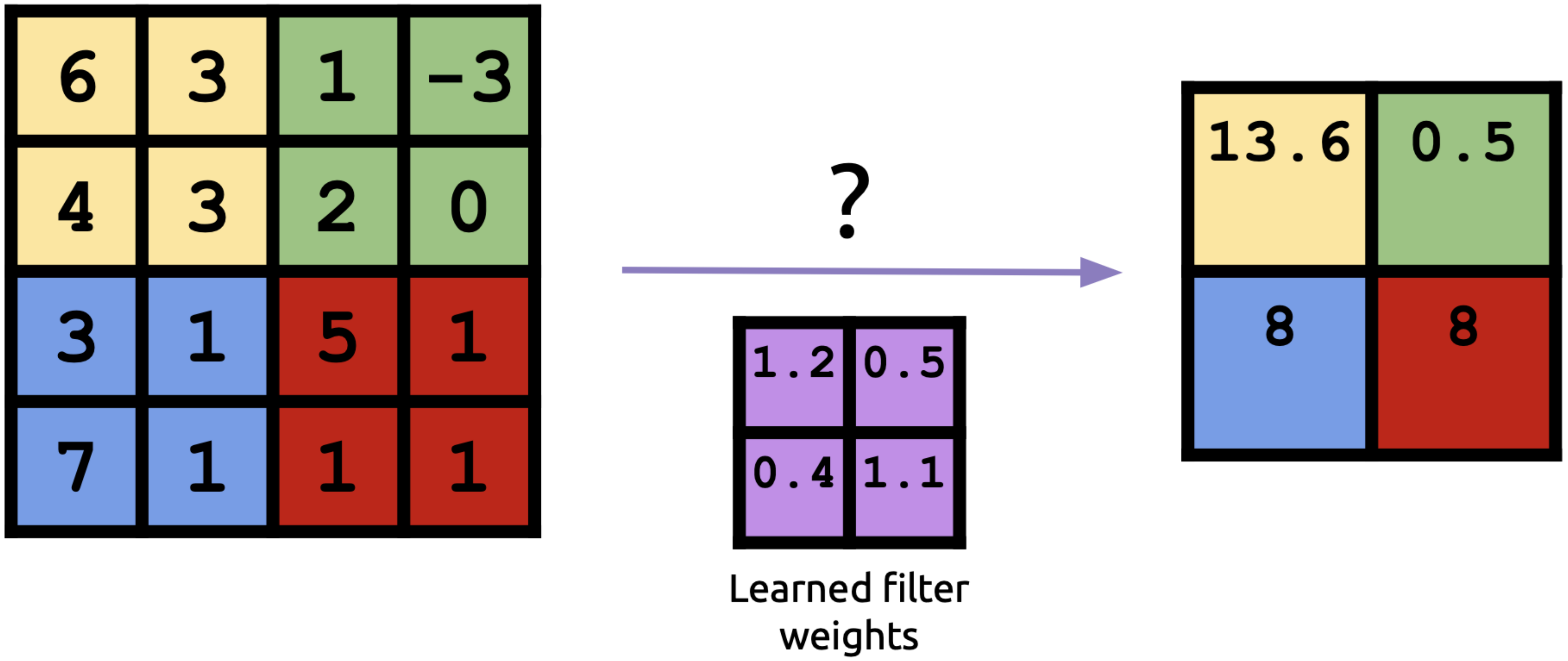
Other Pooling Techniques

Average pooling with stride 2 and 2x2 filters

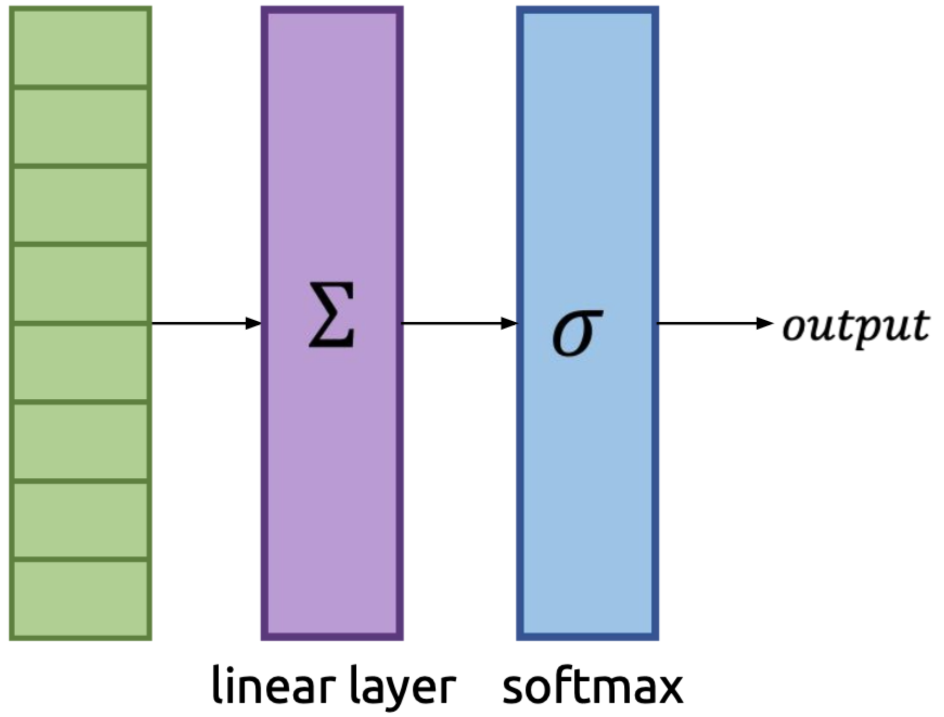


Learning a Pooling Function

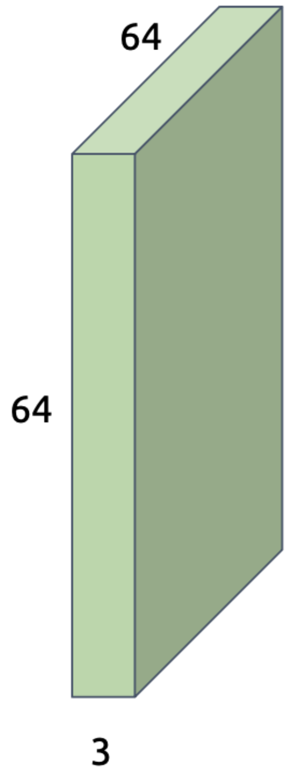
- The network can learn its own pooling function
- Implement via a strided convolution layer

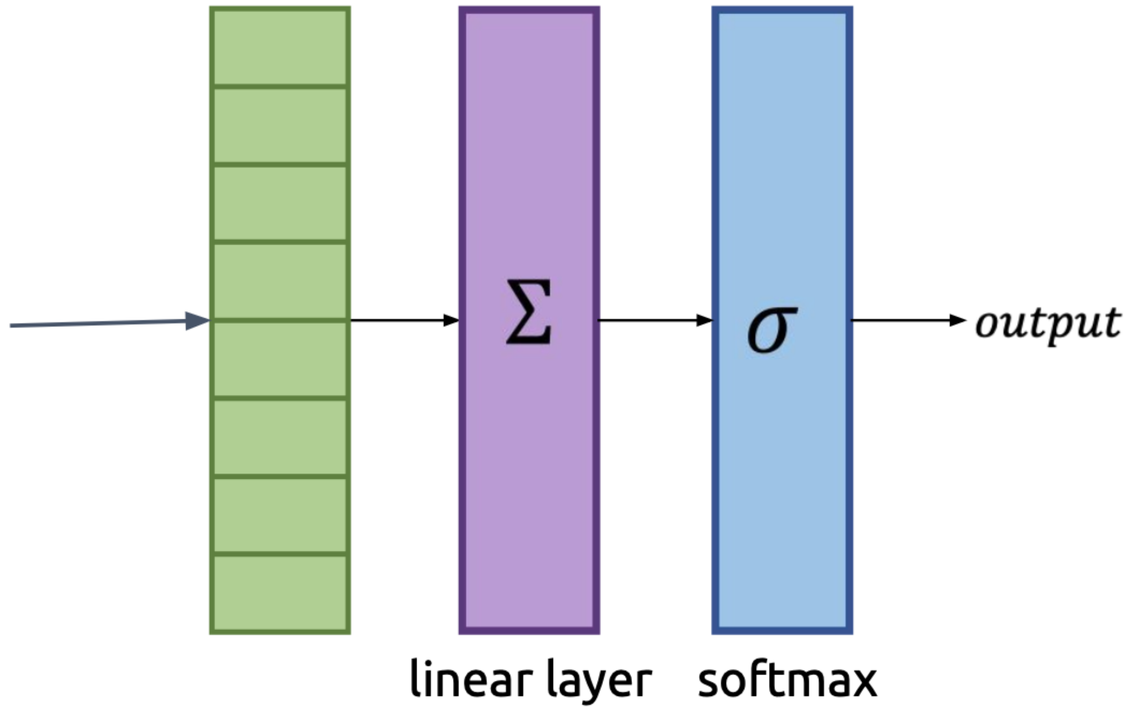


Our neural network so far

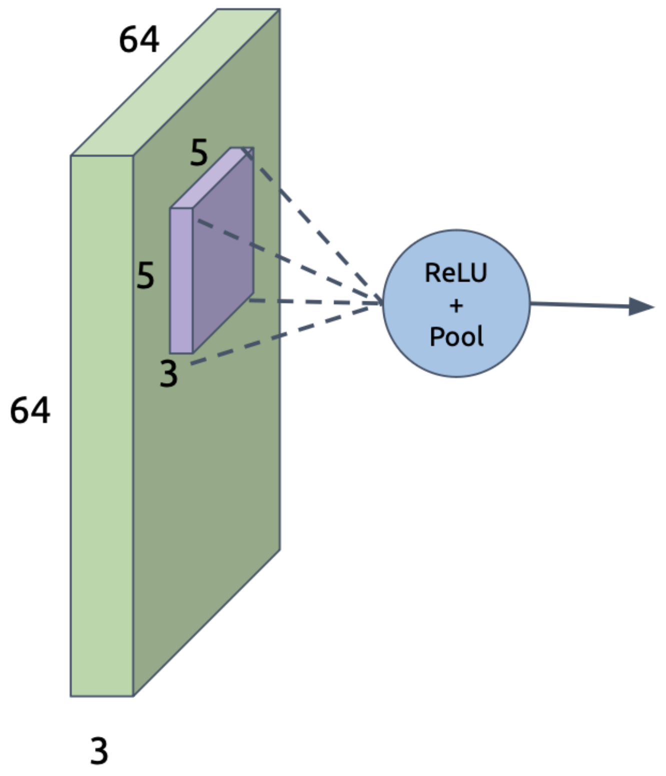


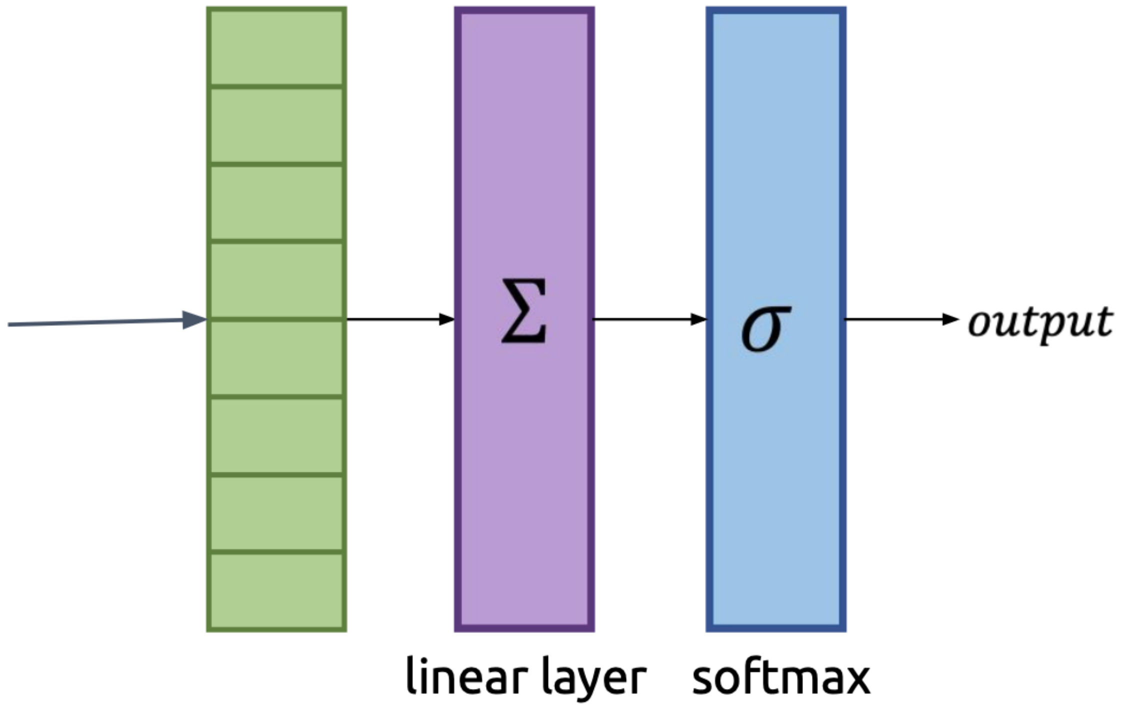
Convolutional Neural Network Architecture



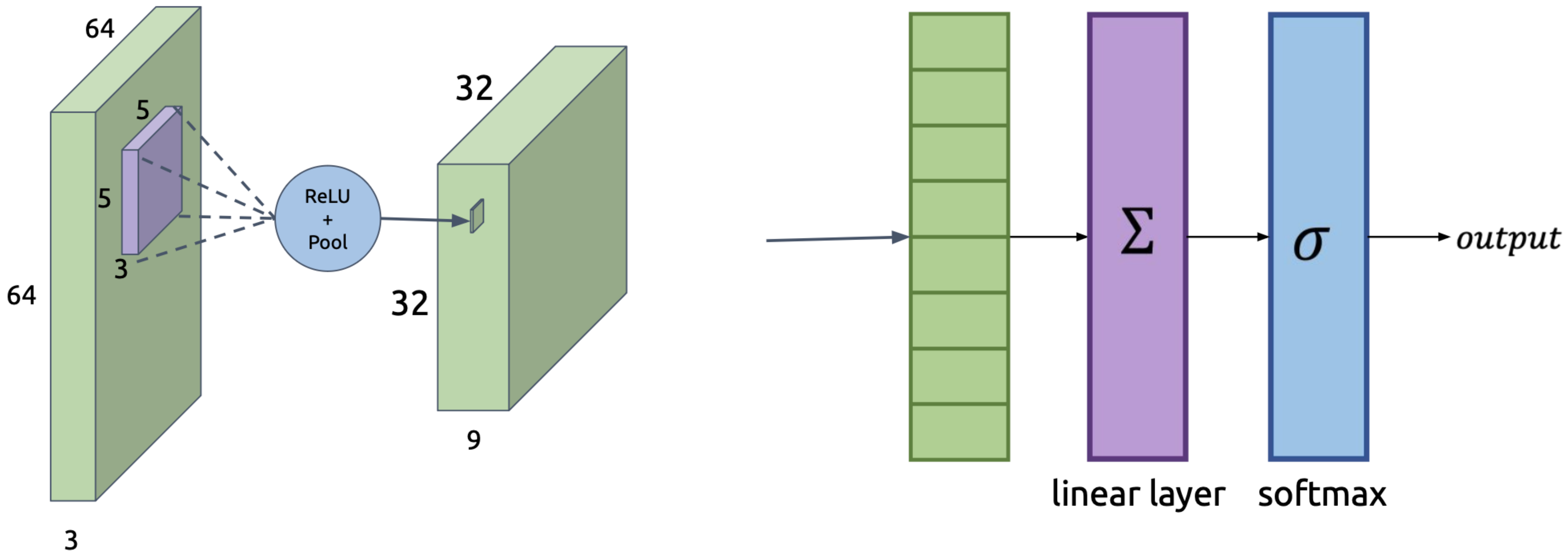


CNN Architecture

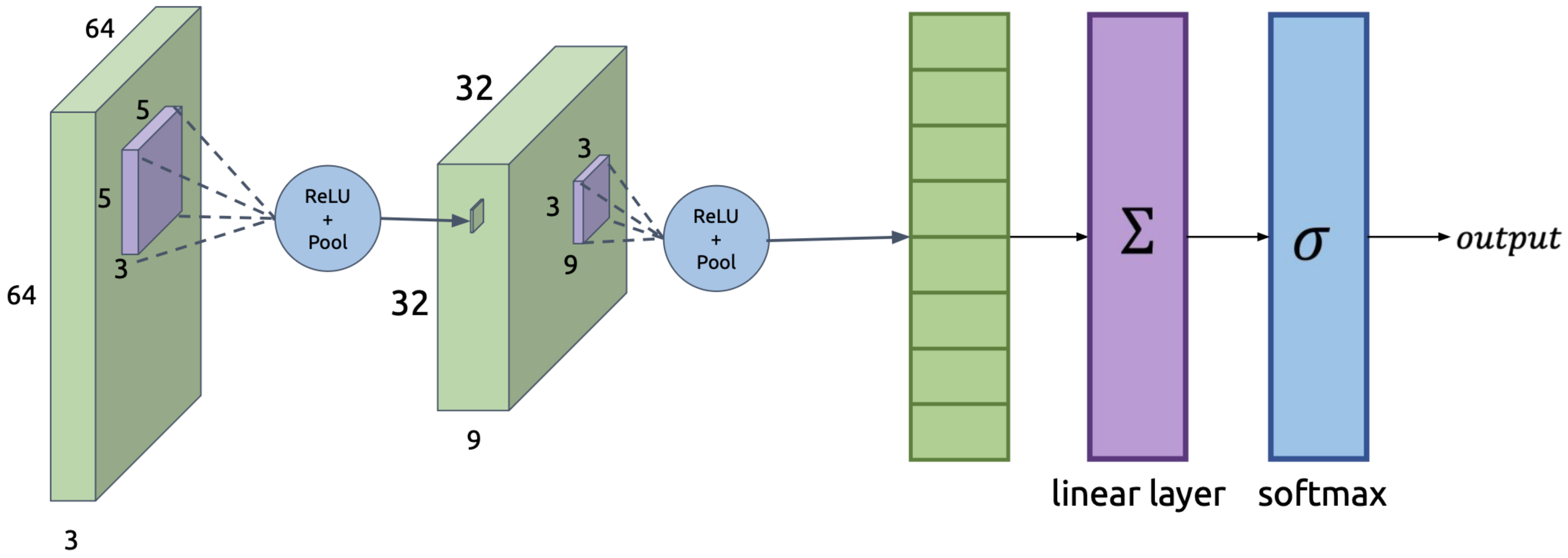




CNN Architecture

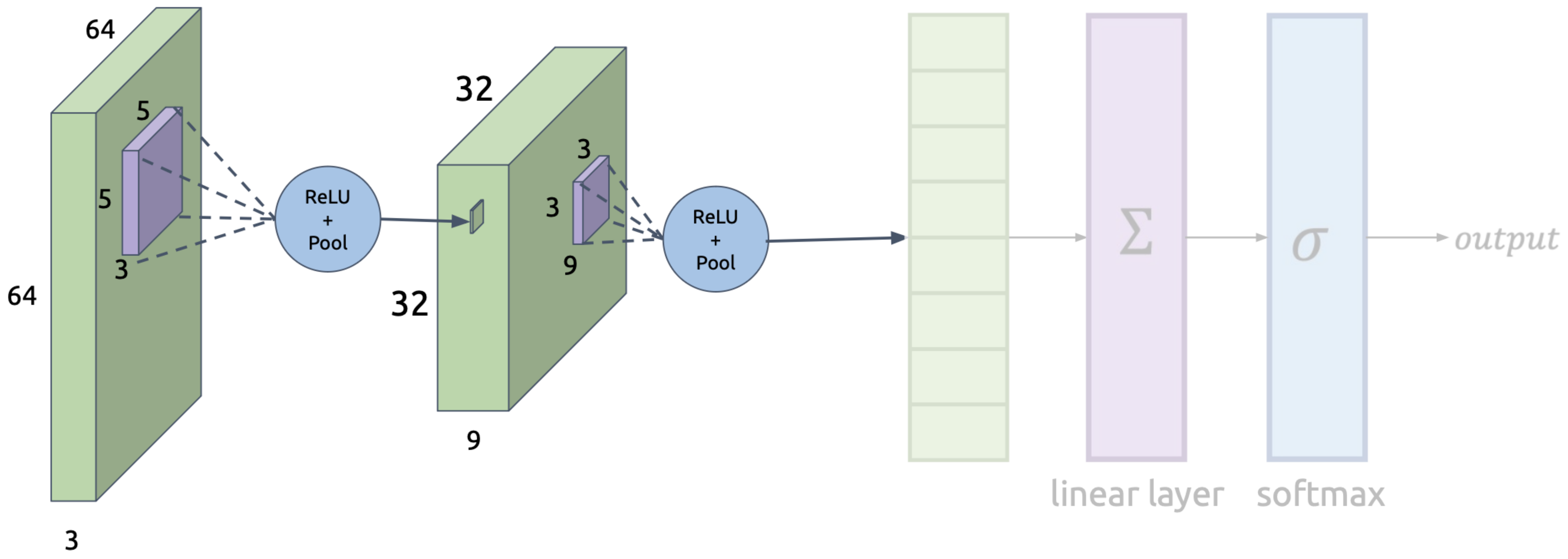


CNN Architecture



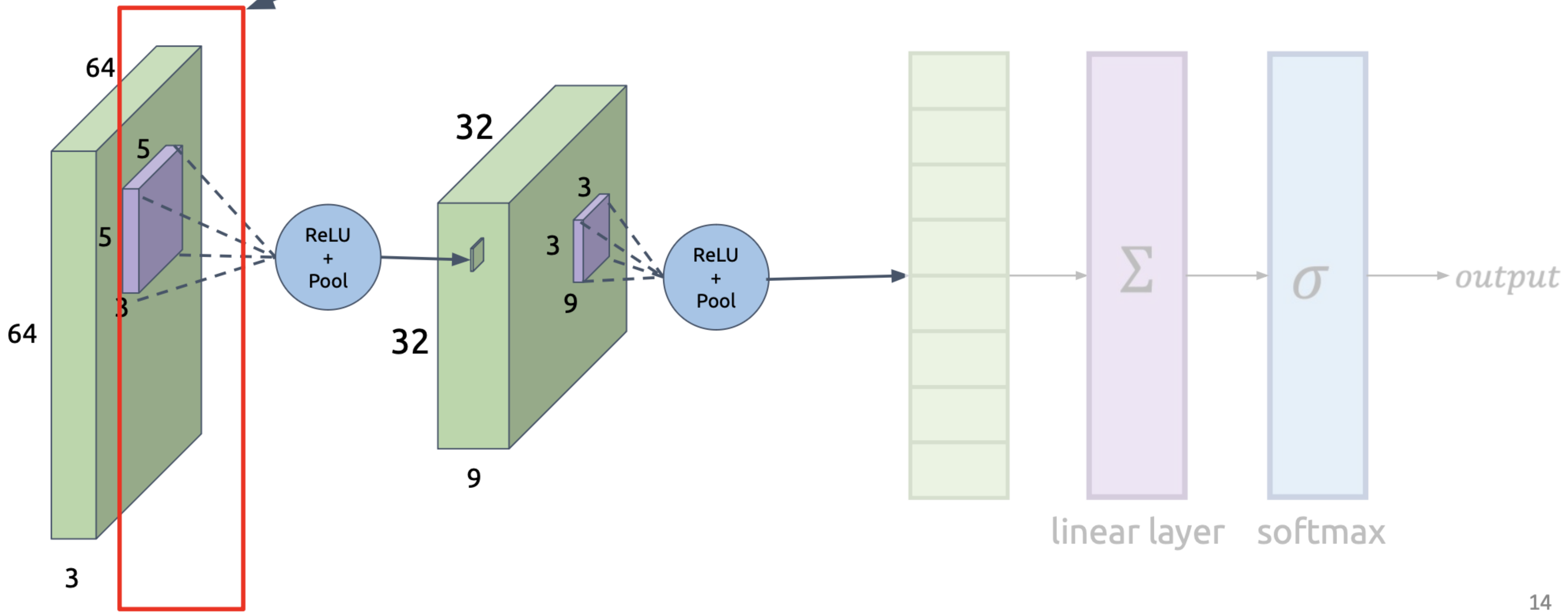
CNN Architecture

This part learns to extract ***features*** from the image



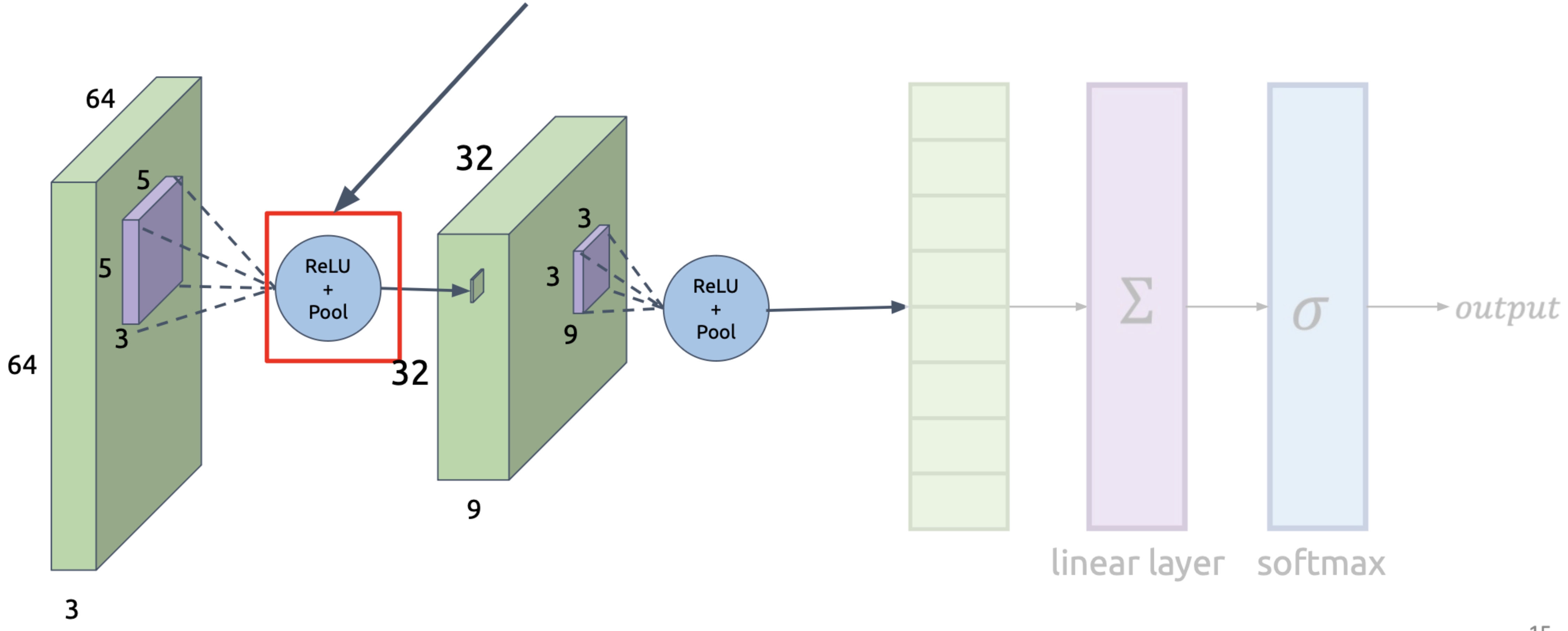
CNN Architecture

A single convolutional layer



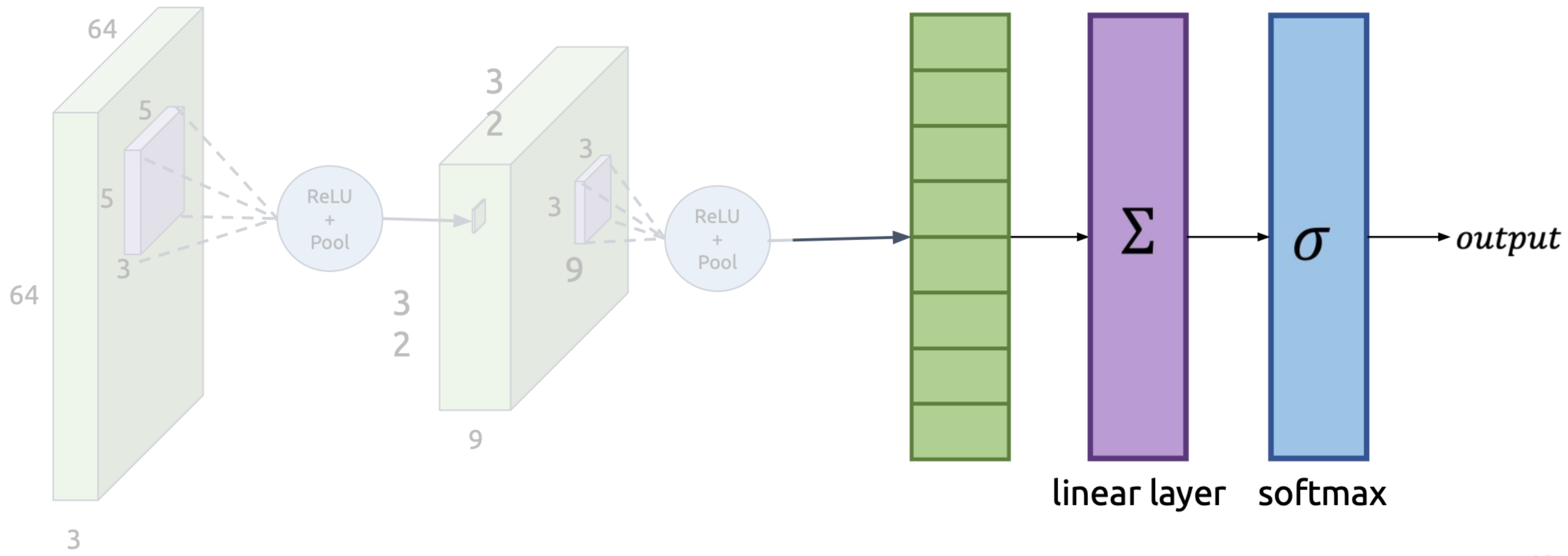
CNN Architecture

Activation after filter passes over image

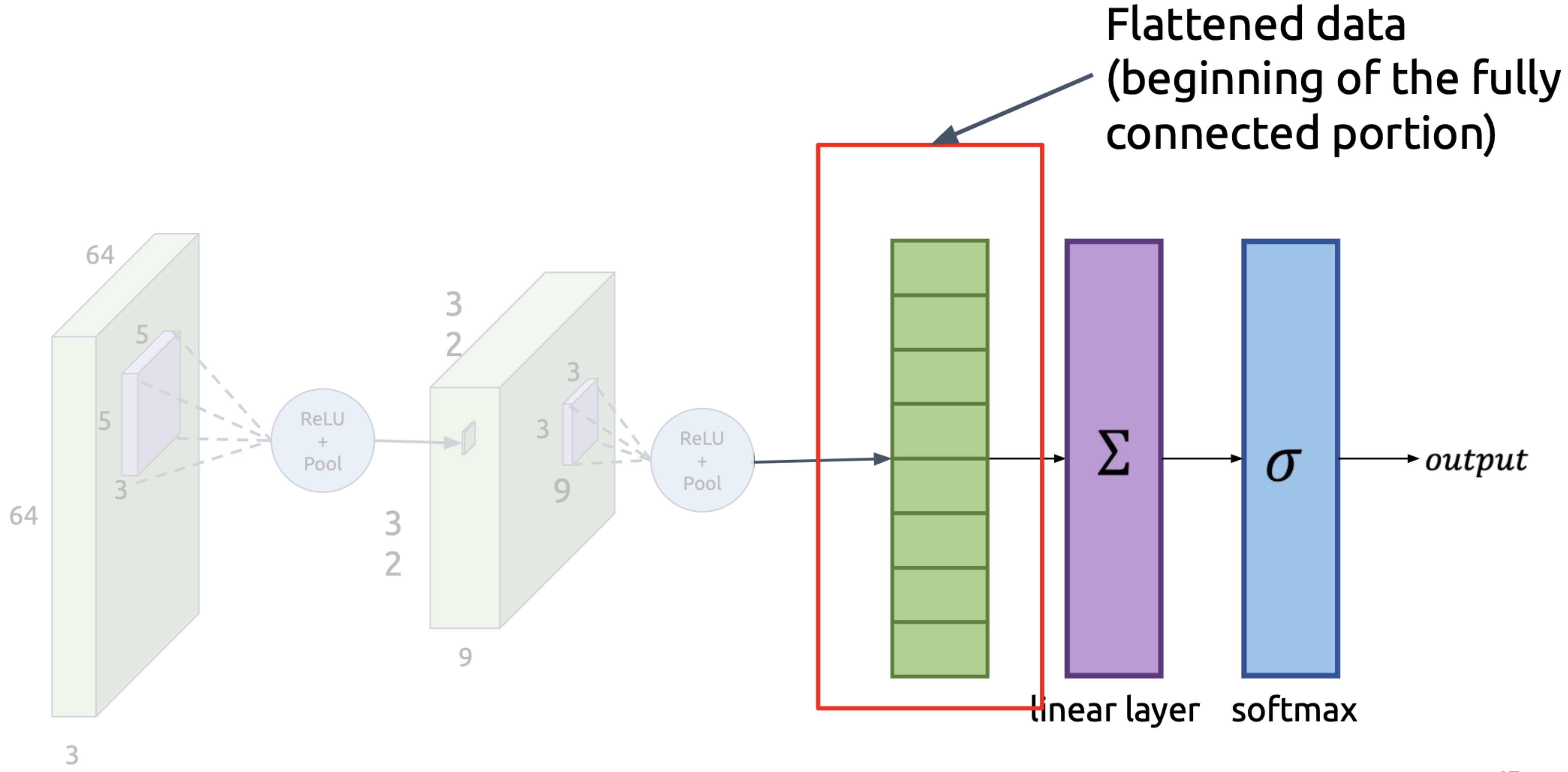


CNN Architecture

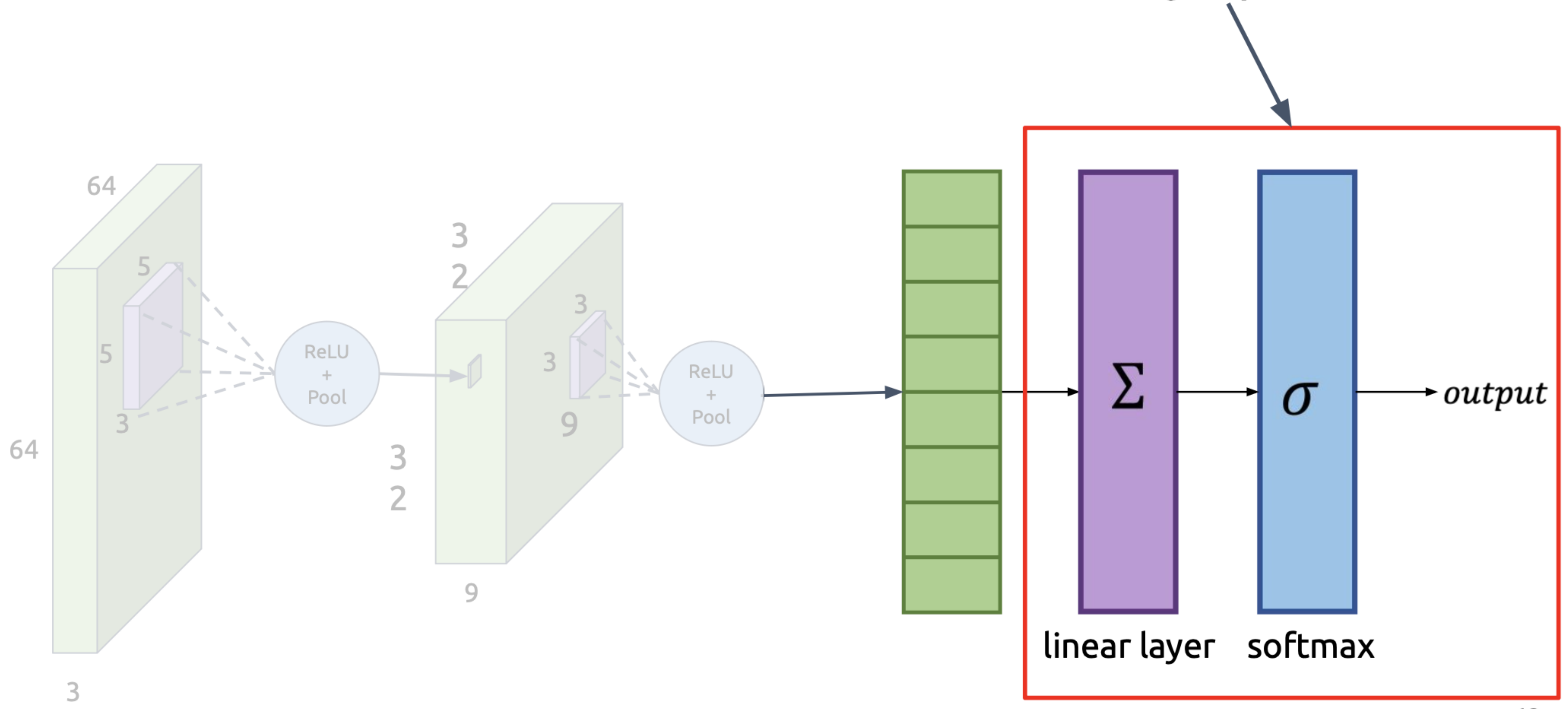
This part learns to perform a specific task (e.g. classification) using those features



CNN Architecture

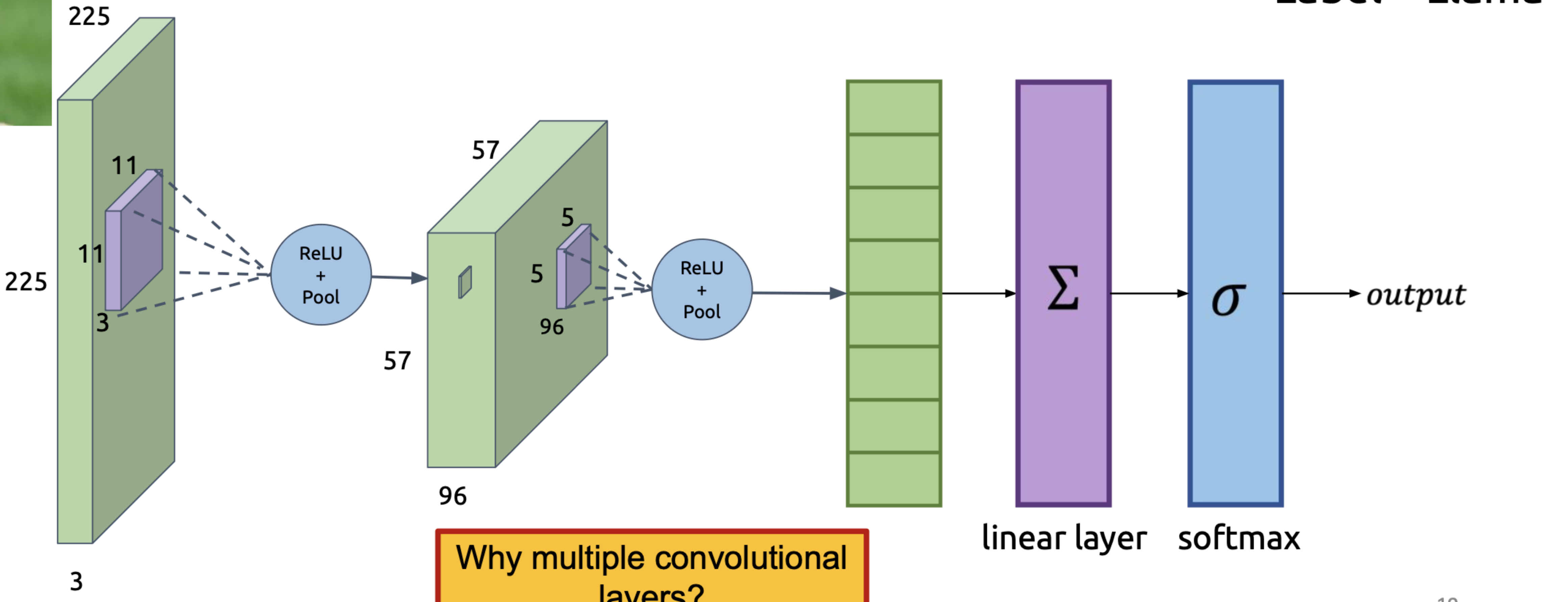


CNN Architecture



CNN Architecture

Input

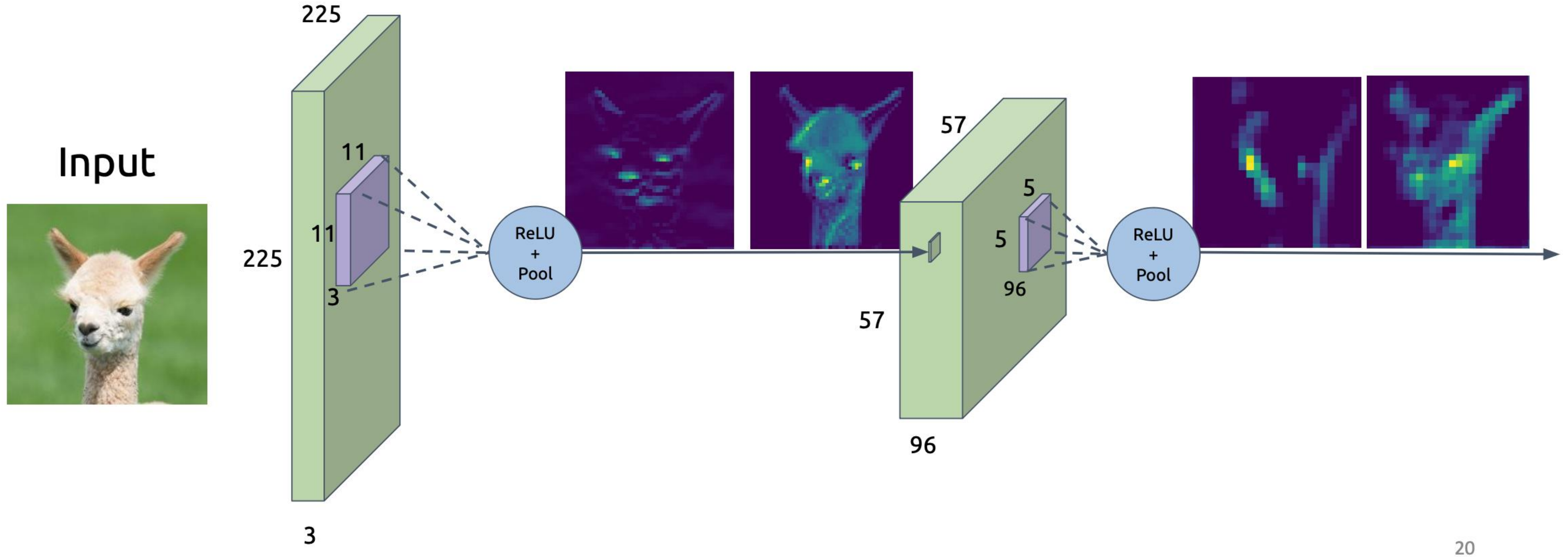


Why multiple convolutional layers?

Feature Extraction using multiple convolution layers

Hierarchy of features

Sequence of layers detect broader and broader features

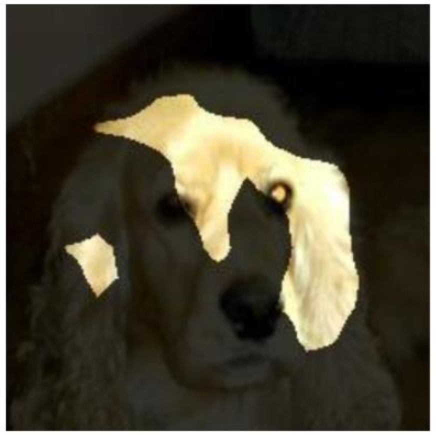


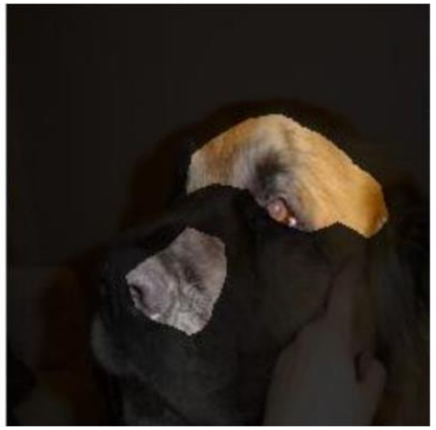
Any questions?

Example: Network Dissection

http://netdissect.csail.mit.edu/

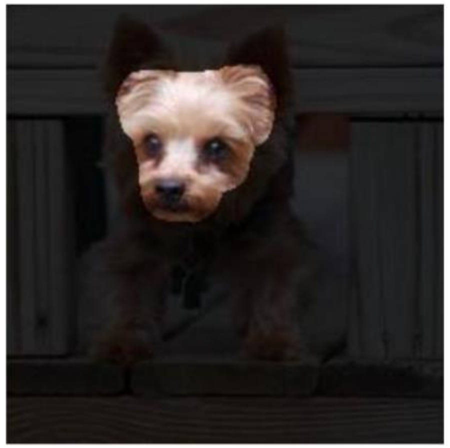
Layer 3 active regions

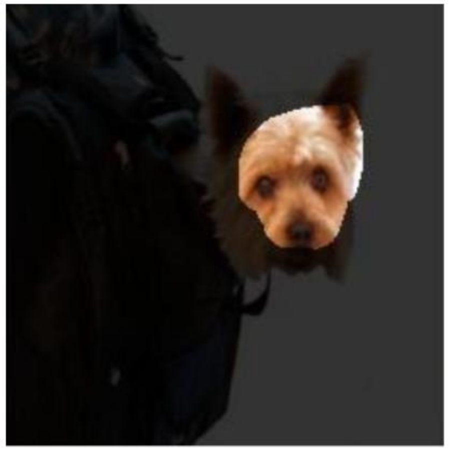




“Eye Detector”

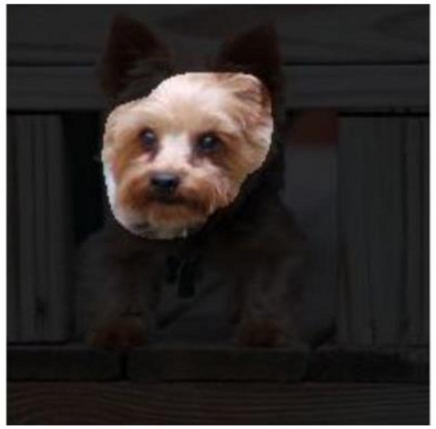
Layer 4 active regions

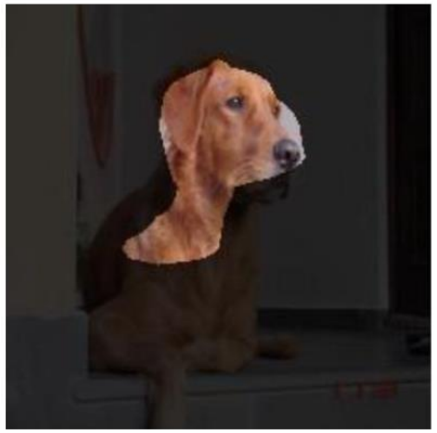




“Eyes and Nose Detector”

Layer 5 active regions





“Dog Face Detector”

ILSVRC 2012

(ImageNet Large Scale Visual Recognition Challenge)

The classification task on ImageNet:

For each image, assign 5 labels in order of decreasing confidence. Success if one of these labels matches the ground truth

https://commons.wikimedia.org/wiki/File:Common_zebra_1.jpg

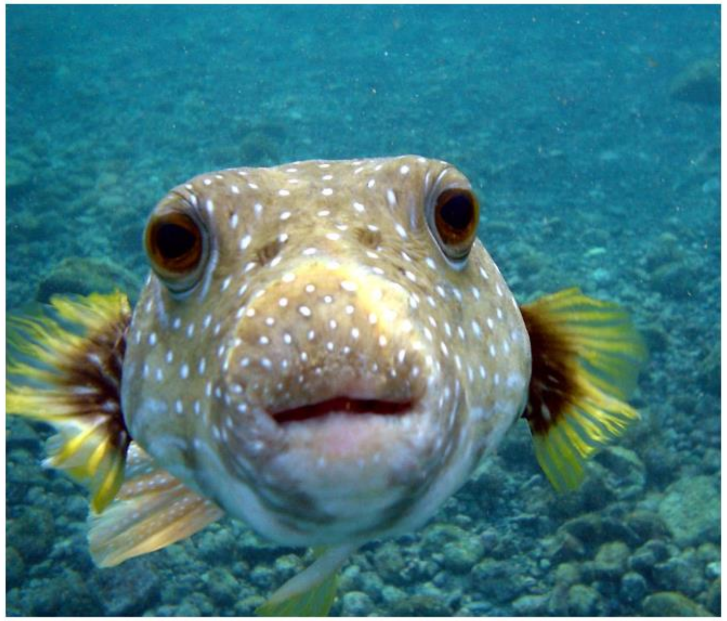
Predictions:

1. Carpet
2. $\boxed{\text{Zebra}}$
3. Llama
4. Flower
5. Horse

✓

ILSVRC 2012

Percentage that model fails to classify is known as ***Top 5 Error Rate***



https://commons.wikimedia.org/wiki/File:Puffer_Fish_DSC01257.JPG

Predictions:

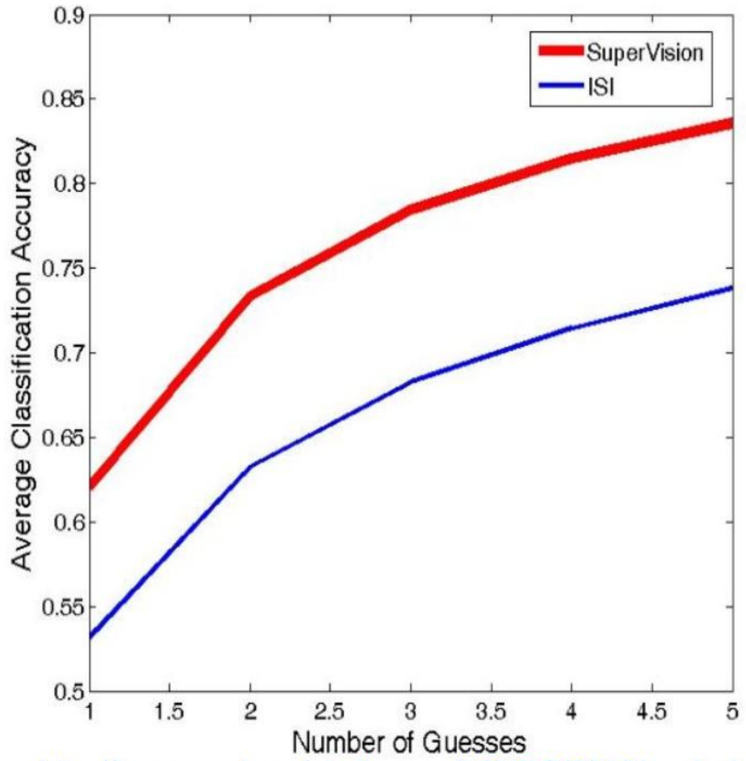
1. Sponge
2. Person
3. Llama
4. Flower
5. Boat

✗

AlexNet: Why CNNs Are a Big Deal

Major performance boost on ImageNet at ILSCRV 2012

Top 5 error rate of 15.3% compared to 26.2% achieved by 2nd place

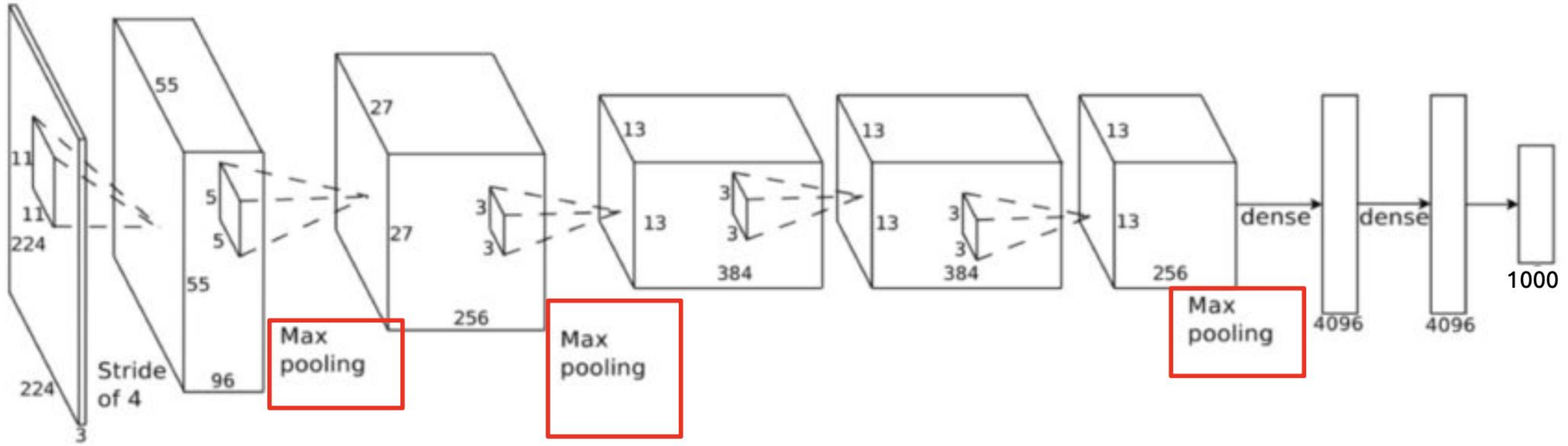


http://image-net.org/challenges/LSVRC/2012/analysis/

Note: SuperVision is the name of Alex's team

AlexNet

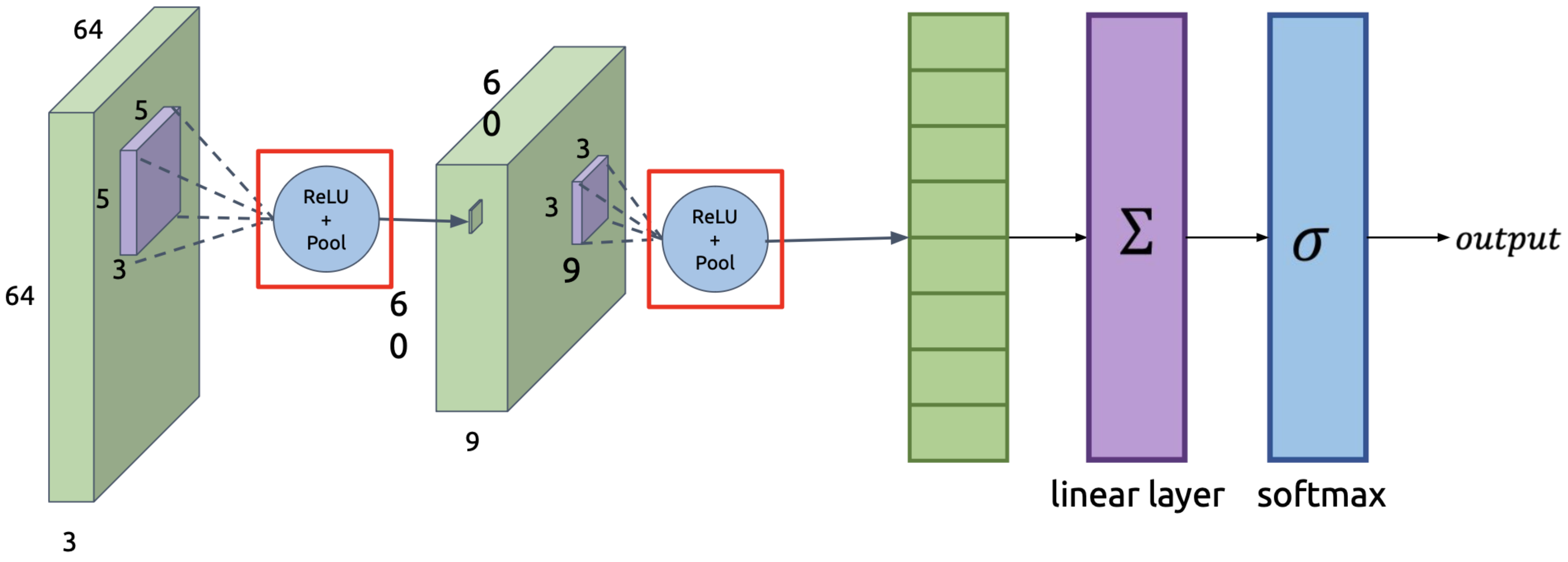
- 60 million parameters
- 5 Convolutional Layers
- 3 Fully Connected Layers



[Alex Krizhevsky et al. 2012]

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

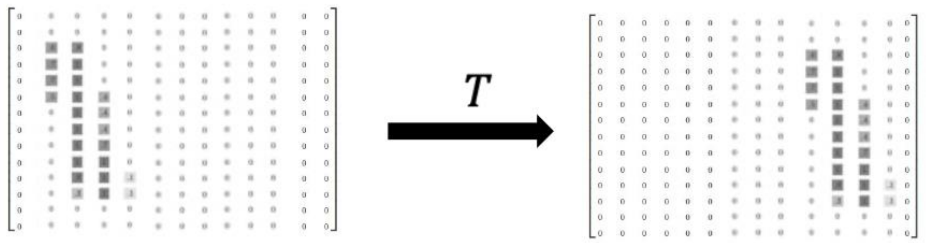
Pooling

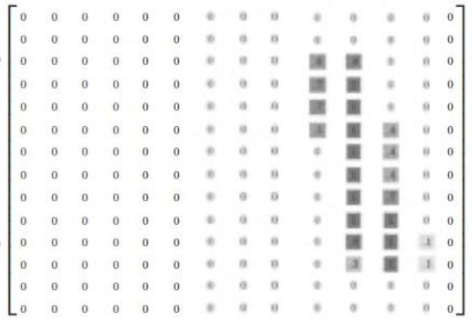

64
5
5
3
64
3
ReLU + Pool
6
0
6
0
3
3
9
9
ReLU + Pool
Σ
σ
output
linear layer
softmax

So…did we achieve our goal of translational invariance?

What was Translational Invariance again?

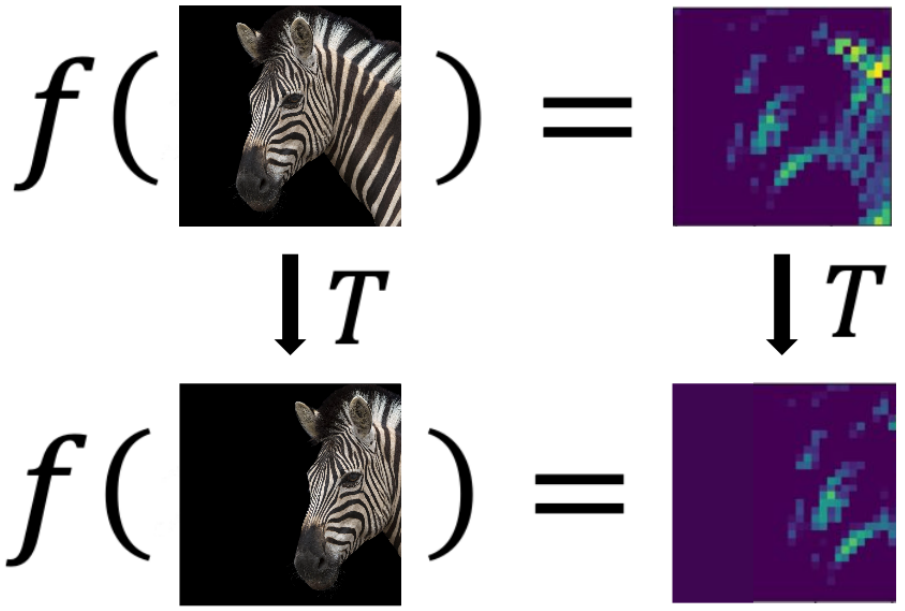
- To make a neural net f robust in this same way, it should ideally satisfy ***translational invariance***: $f\big(T(x)\big) = f(x)$, where
  - x is the input image
  - T is a translation (i.e. a horizonal and/or vertical shift)



$f($ $) \overset{?}{=} f($  $)$

Are CNNs Translation Invariant?

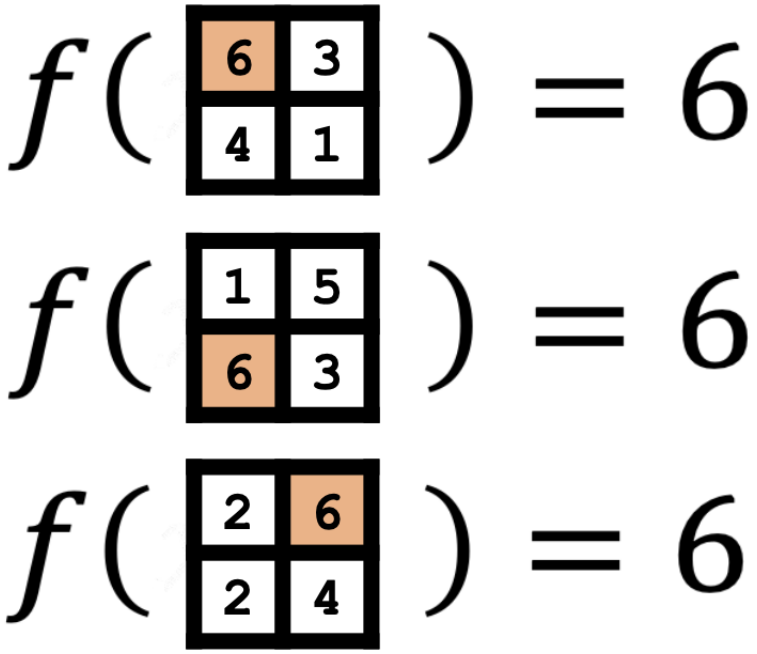
- Convolution is ***translation equivariant***
  - A translated input results in an output translated by the same amount
  - $f\big(T(I)\big) = T\big(f(I)\big)$
  - $(T(I) \otimes K)(x,y) = T(I \otimes K)(x,y)$



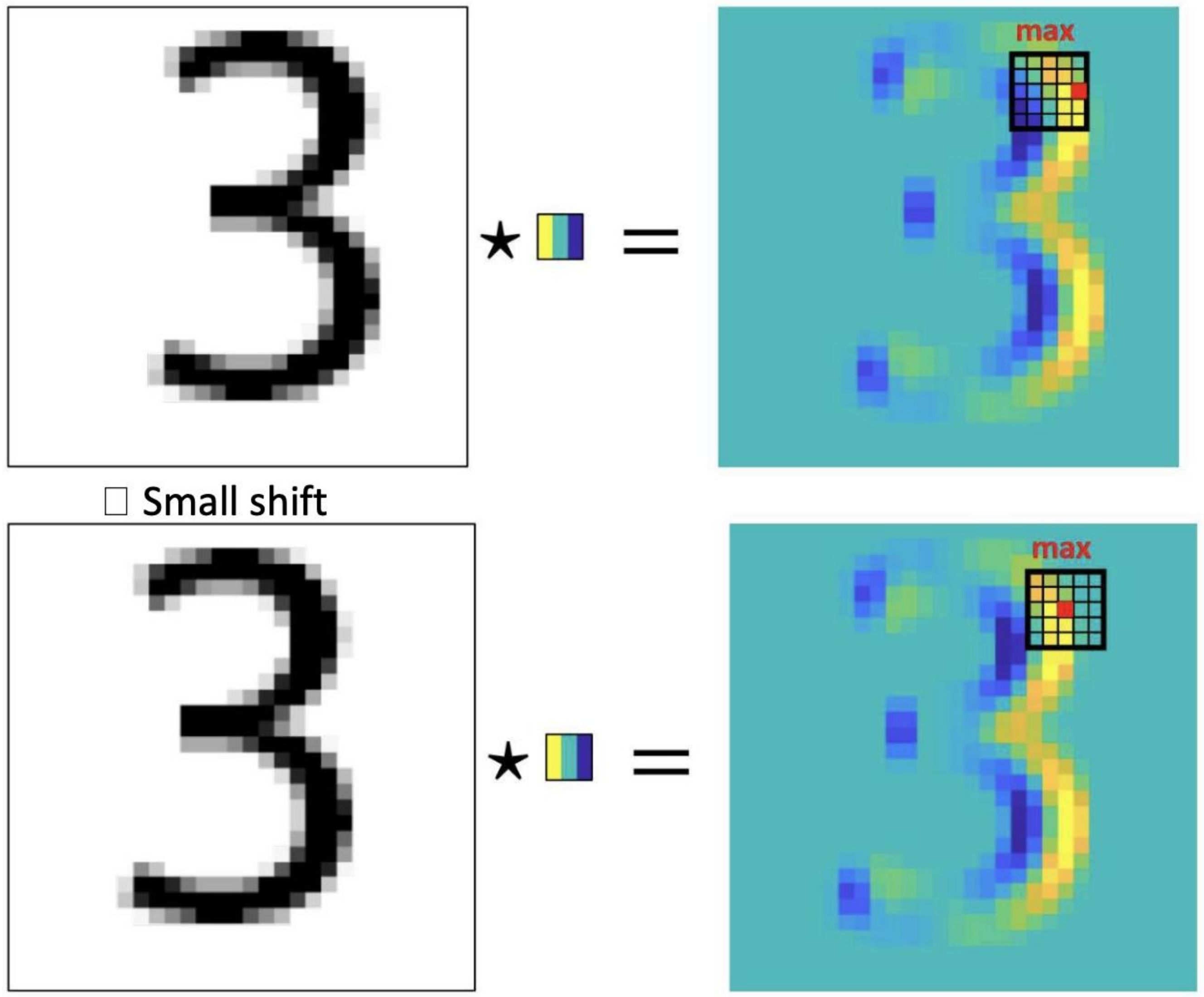
* $Here, (I \otimes K)(x,y) = \sum_{m} \sum_{n} I(x+m, y+n)K(m,n)$

Are CNNs Translation Invariant?

- Max pooling is intended to give invariance to small translations
  - The highest activation pixel can shift around within the pooling window, and the output does not change


$f\left(\begin{bmatrix}6&3\\4&1\end{bmatrix}\right)=6$
$f\left(\begin{bmatrix}1&5\\6&3\end{bmatrix}\right)=6$
$f\left(\begin{bmatrix}2&6\\2&4\end{bmatrix}\right)=6$

So how does it all come together?

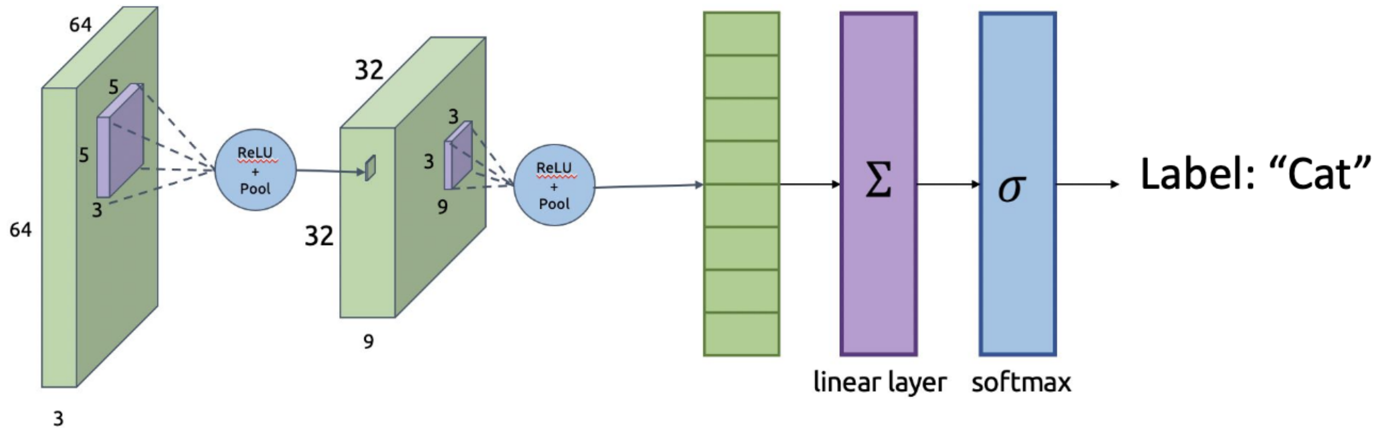


Convolution is ***translation equivariant***

Max pooling gives invariance to small translations

Are CNNs Translation Invariant?

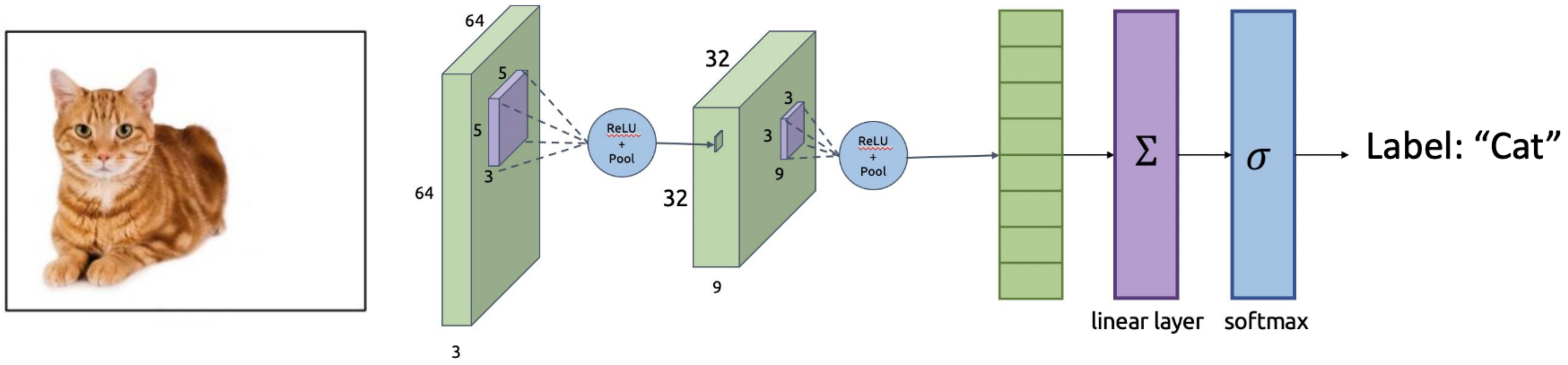
- Answer: CNNs are **"sort of" translation invariant**
  - Shifting the content of the image around tends not to drastically effect the output classification probabilities...



https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.quora.com%2FWhy-and-how-are-convolutional-neural-networks-translation-invariant&psig=AOvVaw3CGbr5n49raEoDHt7opgHk&ust=1645798061134000&source=images&cd=vfe&ved=0CAwQjhxqFwoTCPDfkdvBmPYCFQAAAAAdAAAAABAc

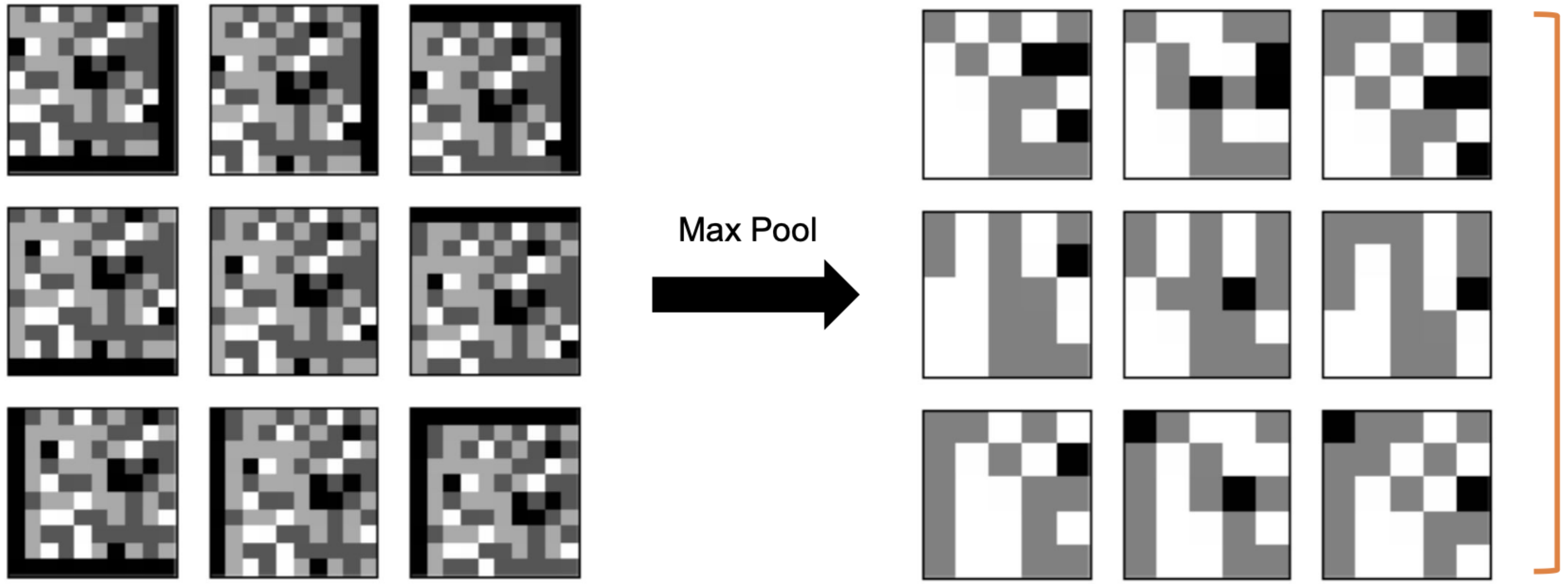
Are CNNs Translation Invariant?

- Answer: CNNs are **“sort of” translation invariant**
  - Shifting the content of the image around tends not to drastically effect the output classification probabilities...



Are CNNs Translation Invariant?

- Answer: CNNs are “sort of” translation invariant
  - Shifting the content of the image around tends not to drastically effect the output classification probabilities…
  - …but they are ***not***, strictly speaking, translation invariant

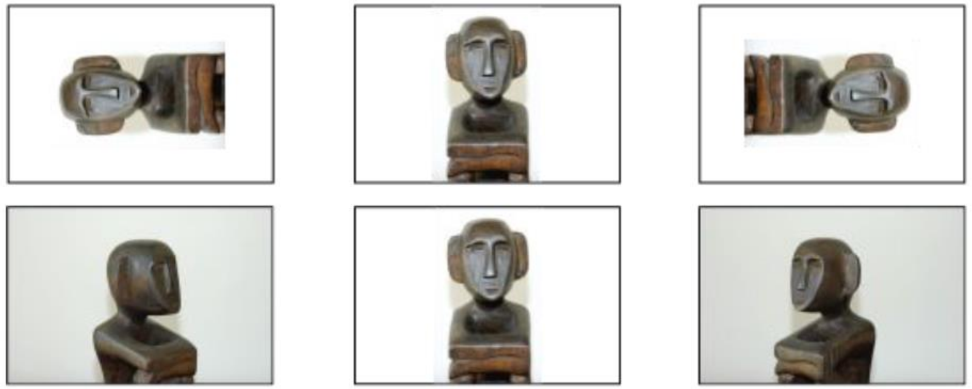


These are ***not*** all the same!

https://dsp.stackexchange.com/questions/24900/translation-invariance-in-max-pooling-and-cascading-with-convolutional-layer

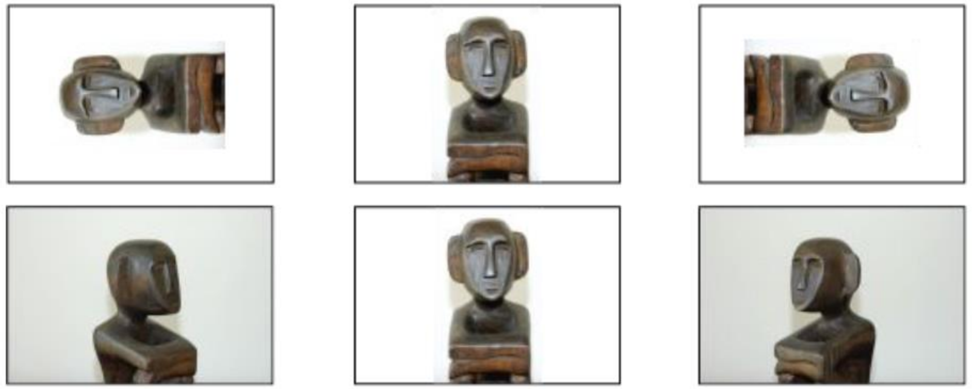
Other Invariances

Rotation/Viewpoint Invariance



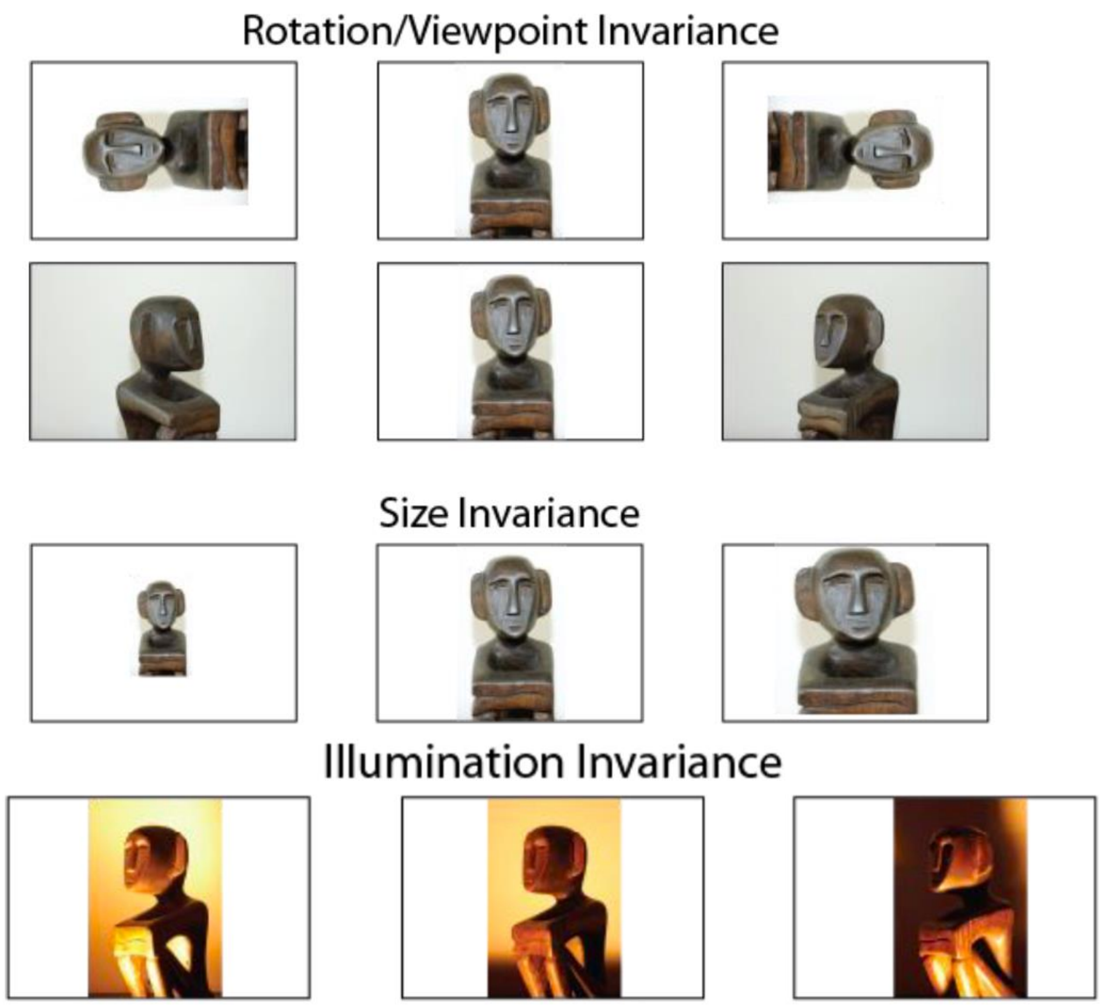
Other Invariances

Rotation/Viewpoint Invariance



Size Invariance

Other Invariances



Matt Krause
mattkrau.se

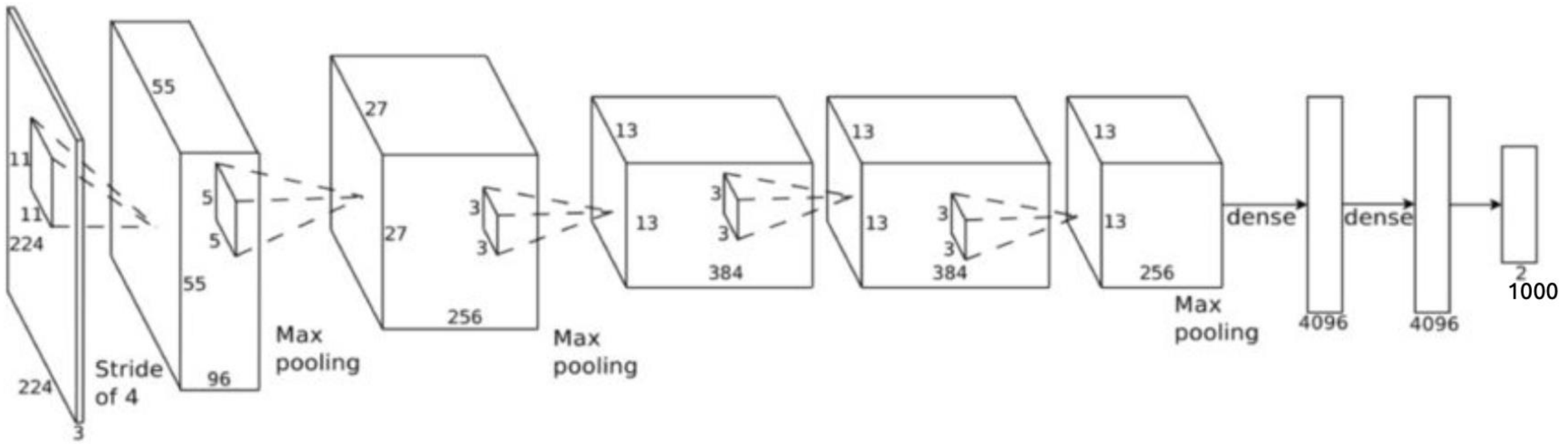
Data Augmentation! Use rotated/scaled/shifted images from your dataset to train

- All are desirable properties!
- How do CNNs fare?
  - Max pooling gives some amount of size and translational invariance
  - But in general, CNNs do not fare well with large changes in lighting or scale.
- Consequences of not having these invariances?
  - Require *lots* of training data
  - Have to show network many examples of lighting changes, scale changes, etc.

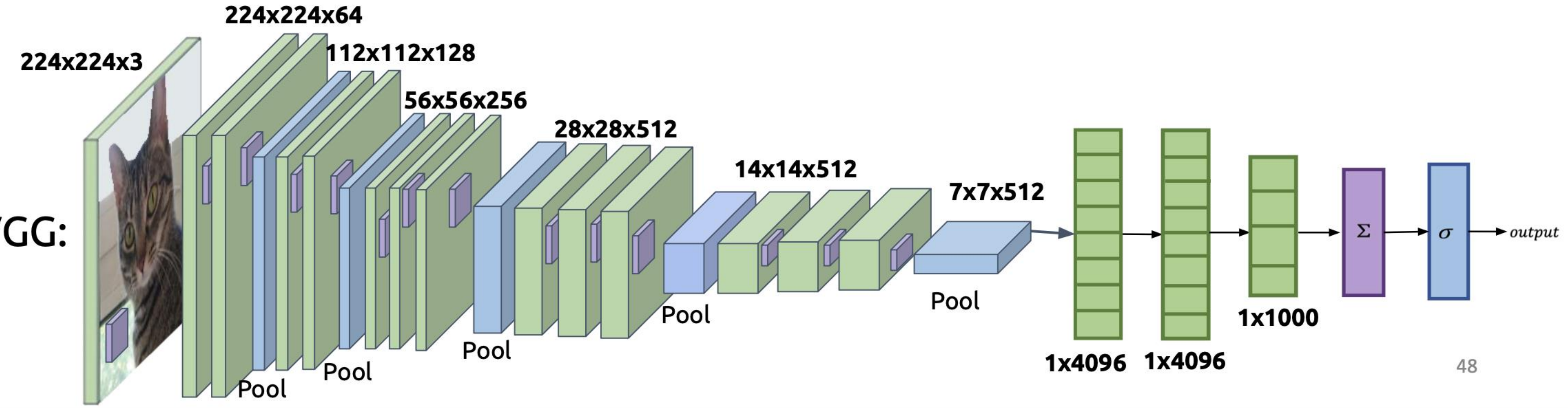
Can we address these concerns without collecting additional data?

More Complicated Networks

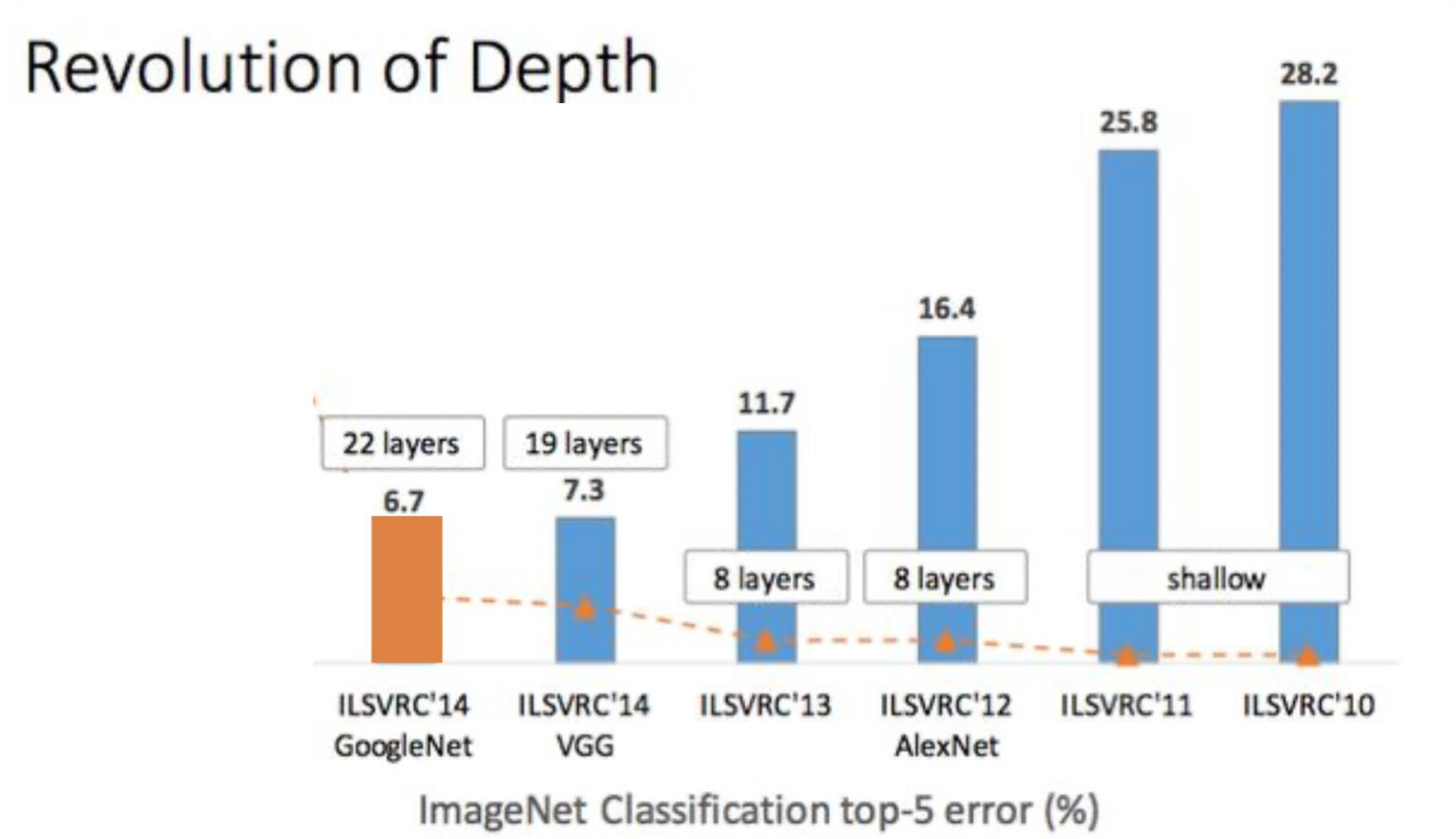
AlexNet:



VGG:

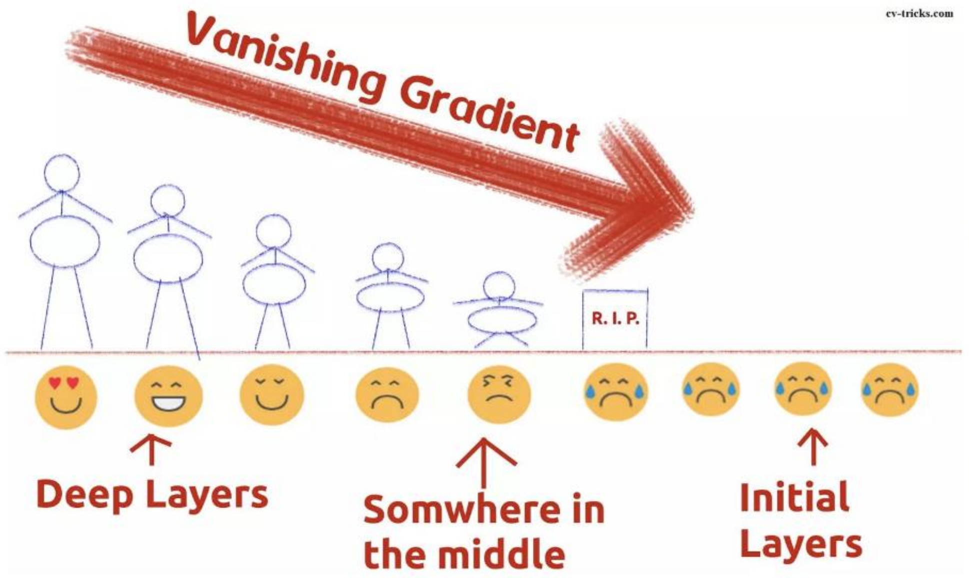


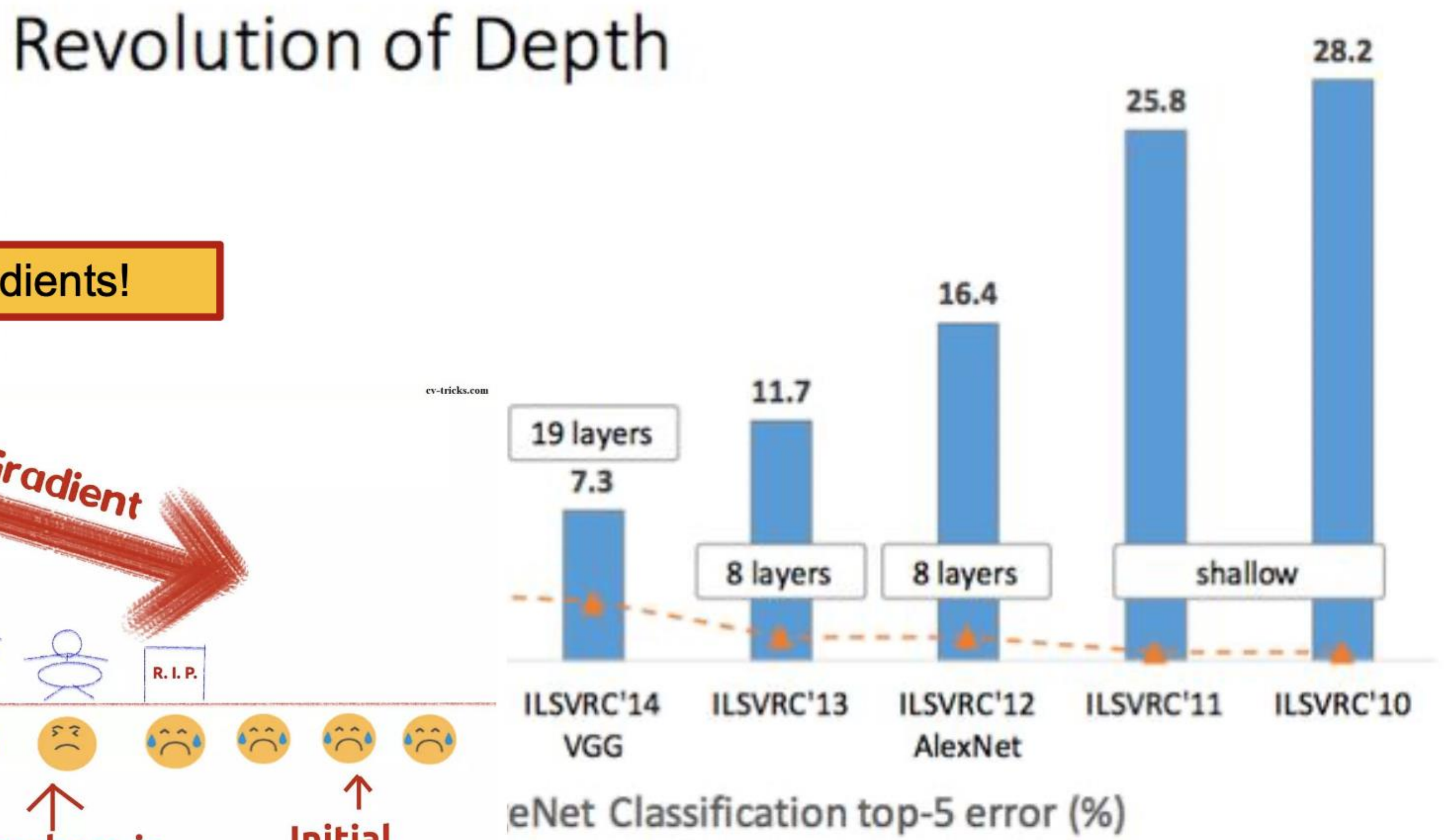
Can you guess what was the biggest bottleneck to adding more layers?



https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33 https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33, https://cv-tricks.com/keras/understand-implement-resnets/

Vanishing gradients!



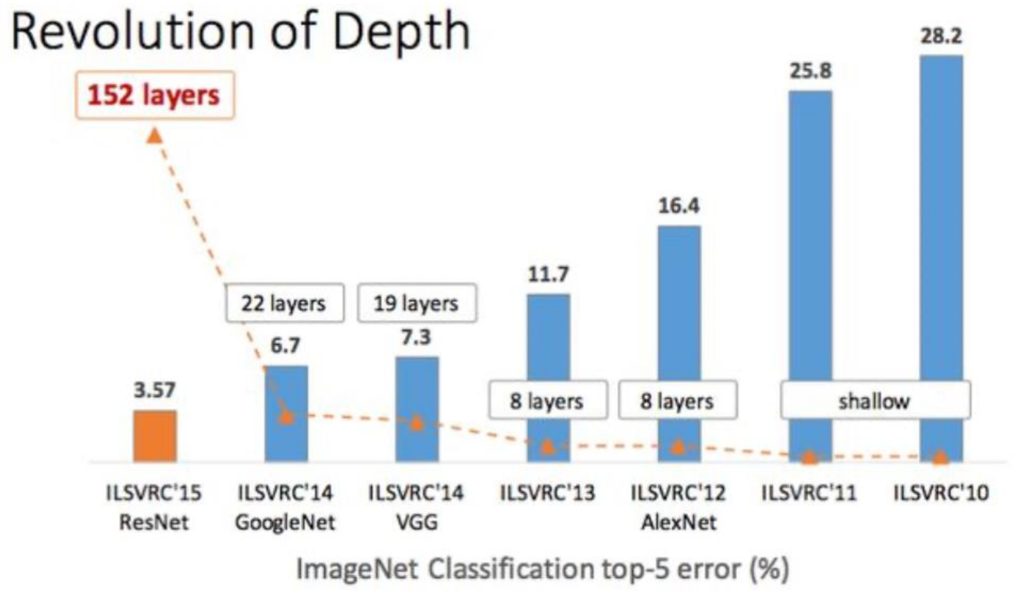


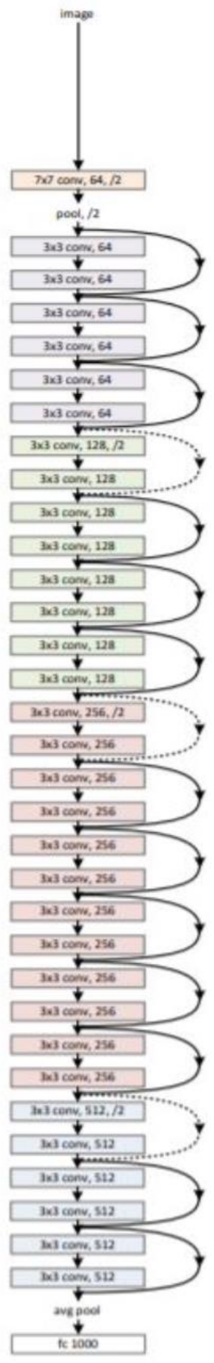
More Complicated Networks

ResNet:

Lots of layers, tons of learnable parameters

Avoids Vanishing Gradient problem
but how?





K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

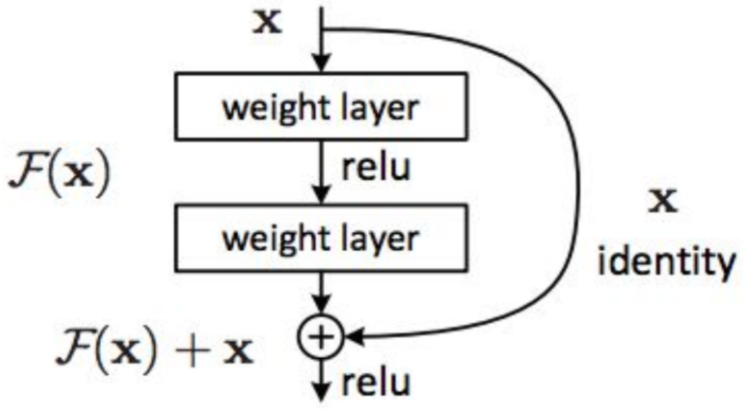
More Complicated Networks

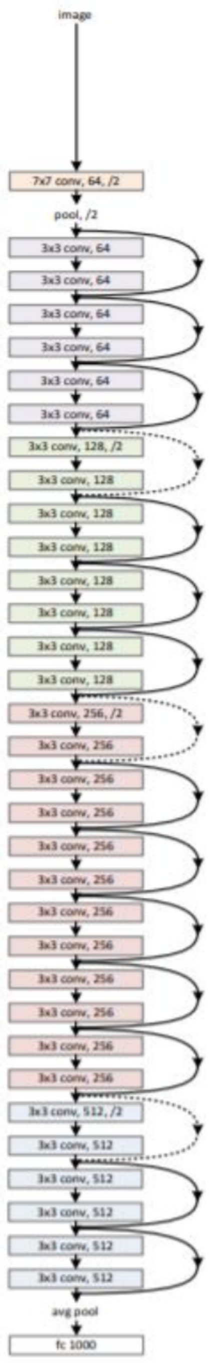
ResNet:

Lots of layers, tons of learnable parameters

Avoids Vanishing Gradient problem

Residual Block

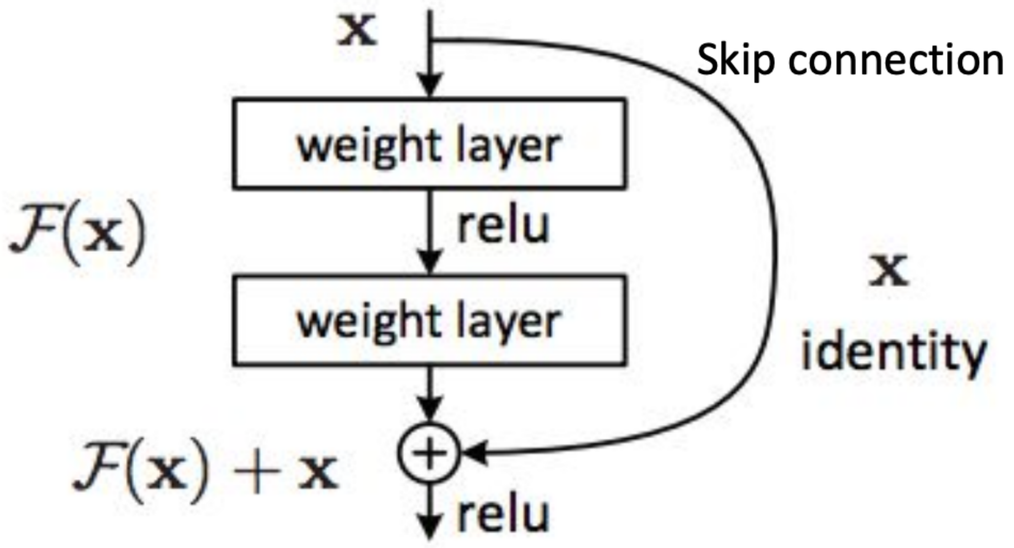




K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

Residual Blocks

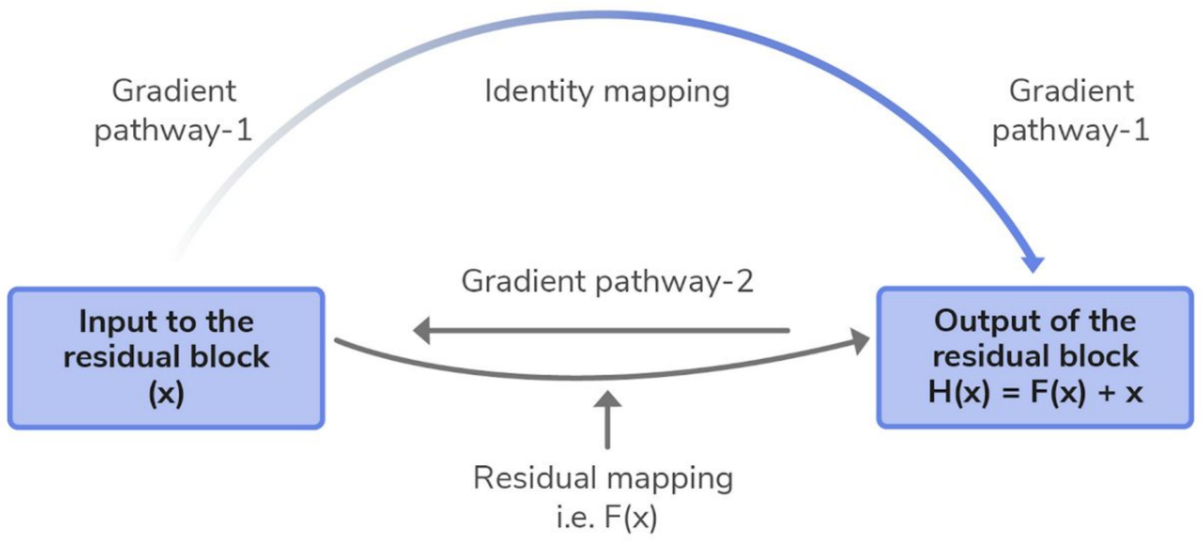
- In very deep nets, each layer often needs to learn just a small transformation of the preceding layer (identity + change)
- Idea: explicitly design the network such that the output of each layer is the identity + some deviation from it
  - Deviation is known as a residual



Residual Blocks

Any questions?

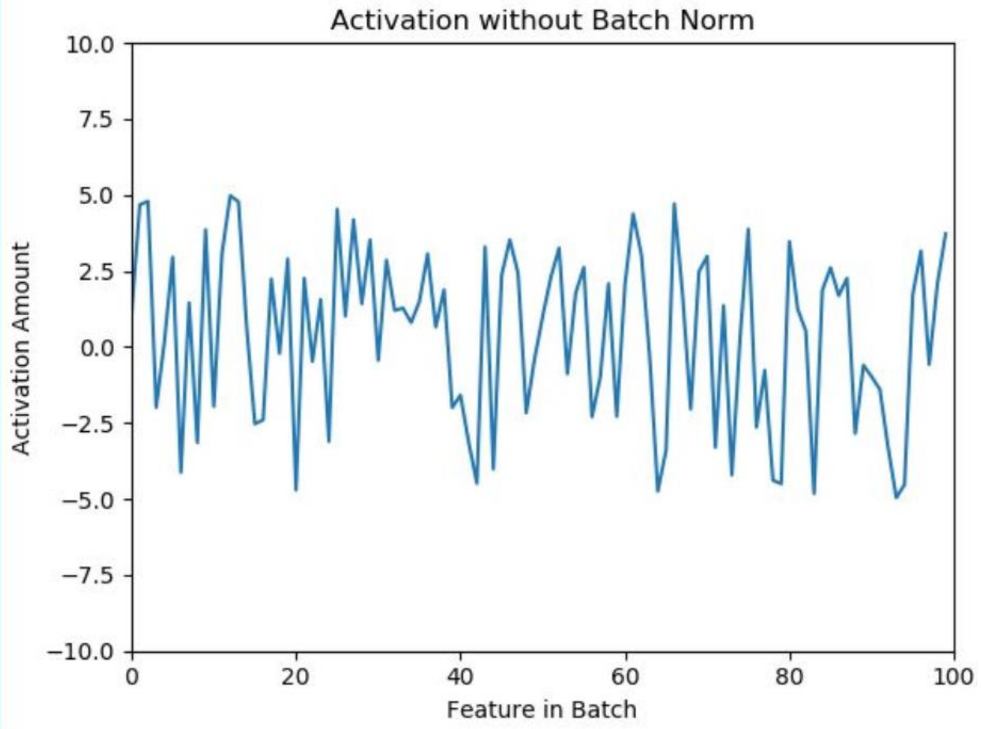
- In very deep nets, each layer often needs to learn just a small transformation of th preceding layer (identity + change)
- Idea: explicitly design the network such that the output of each layer is the identi + some deviation from it
  - Deviation is known as a residual
- Allows gradient to flow through two pathways
- **Significantly stabilizes training of very deep networks**

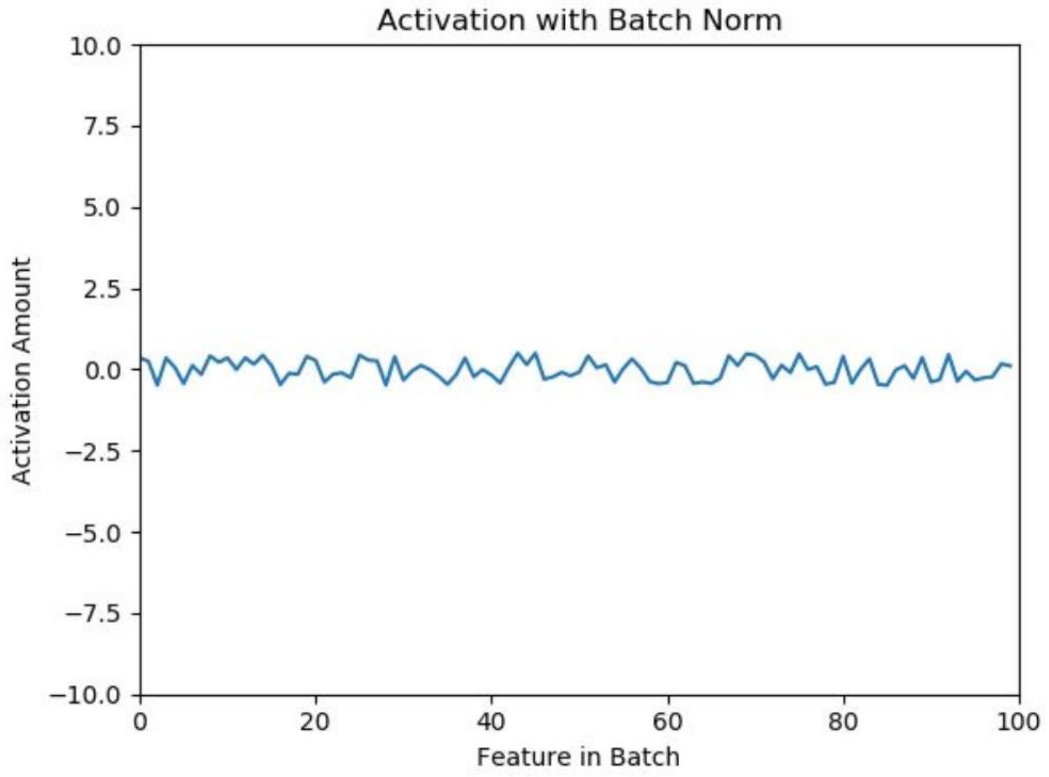

Gradient pathway-1
Identity mapping
Gradient pathway-1
Gradient pathway-2
Input to the residual block (x)
Output of the residual block H(x) = F(x) + x
Residual mapping i.e. F(x)

https://blog.perceptilabs.com/using-resnets-to-detect-anomalies-in-industrial-iot-textile-production/

Batch Normalization (stabilizing training)

Idea: normalize the activations for each feature at each layer



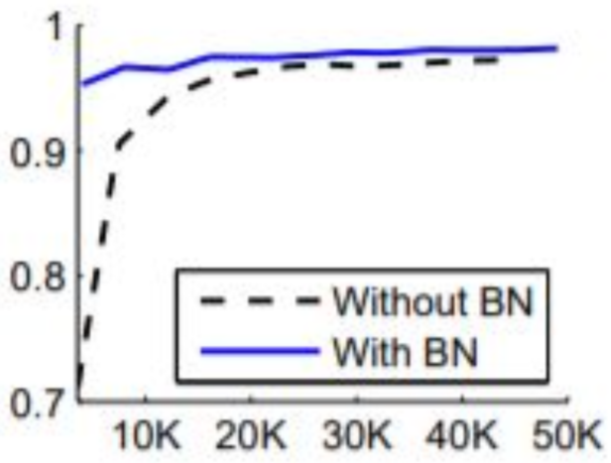


Why might we want to do this?

Batch Normalization: Motivation

More stable inputs = faster training

MNIST test accuracy vs number of training steps



https://arxiv.org/pdf/1502.03167.pdf

Batch Normalization: Implementation

For each feature x, Start by calculating the batch mean and standard deviation for each feature:

$$\mu_{batch} = \frac{\sum_{i=0}^{batch\_size} x_i}{batch\_size}$$

$$\sigma_{batch} = \sqrt{\frac{\sum_{i=0}^{batch\_size} \left(x_i - \mu_{batch}\right)^2}{batch_{size}}}$$

Batch Normalization: Implementation

Normalize by subtracting feature x’s batch mean, then divide by batch standard deviation.

$$x' = \frac{x\ - \mu_{batch}}{\sigma_{batch}}$$

Feature x now has mean 0 and variance 1 along the batch

Batch Normalization in Tensorflow

```
tf.keras.layers.BatchNormalization(input)
```

Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/BatchNormalization

Recap

CNNs

Architecture

AlexNet + Pooling

CNNs are “sort of” translationally invariant

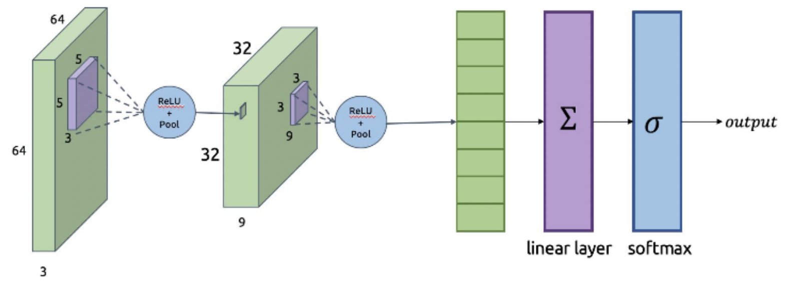
Weekly quiz #4 out now!

Deeper CNNs

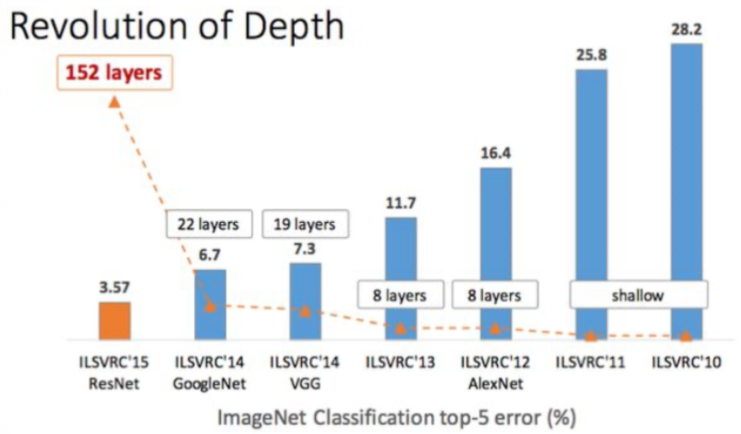
Many layers = vanishing gradient

ResNet + Residual blocks

Batch normalization



64
64
3
5
5
3
ReLU + Pool
32
32
9
3
3
9
ReLU + Pool
Σ
σ
output
linear layer
softmax


Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)