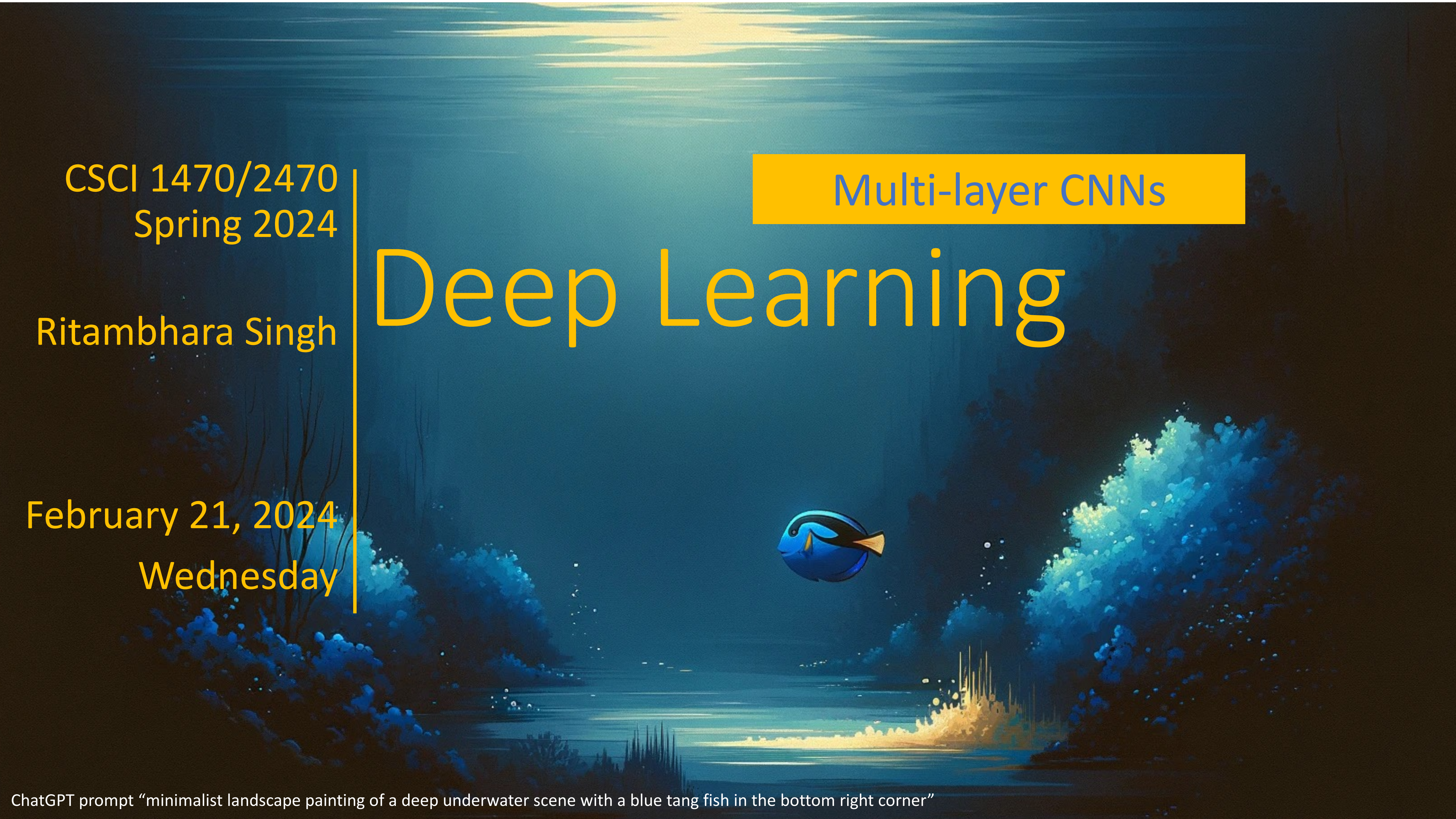
CSCI 1470/2470
Spring 2024

Ritambhara Singh

February 21, 2024
Wednesday

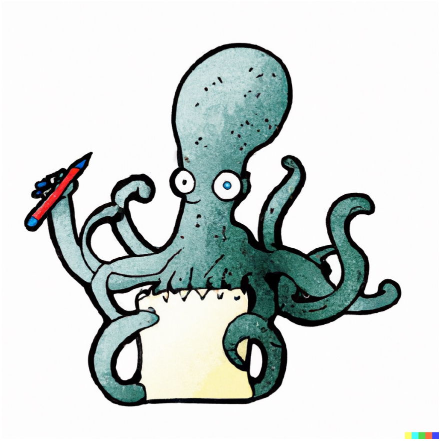
Multi-layer CNNs

Deep Learning



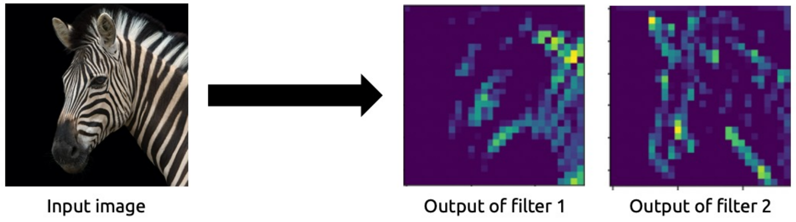
ChatGPT prompt “minimalist landscape painting of a deep underwater scene with a blue tang fish in the bottom right corner”

Recap



Convolution

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



Convolution in Tensorflow

- Tensorflow conv2d function
- Padding
- Application to MNIST/CIFAR

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

- input: Input Image (4-D Tensor)
- filter: Filter/Kernel (4-D Tensor)
- strides: Strides along each dimension
- padding: Type of Padding (String “Valid” or “Same”)

Today's goal – continue to learn about CNNs

(1) Convolutional Neural Network (CNN) architecture

(2) First successful CNN - AlexNet

Pooling and translational invariance

(3) Deeper CNNs!

Residual Blocks

Batch normalization

Bias Term in Convolution Layers

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

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

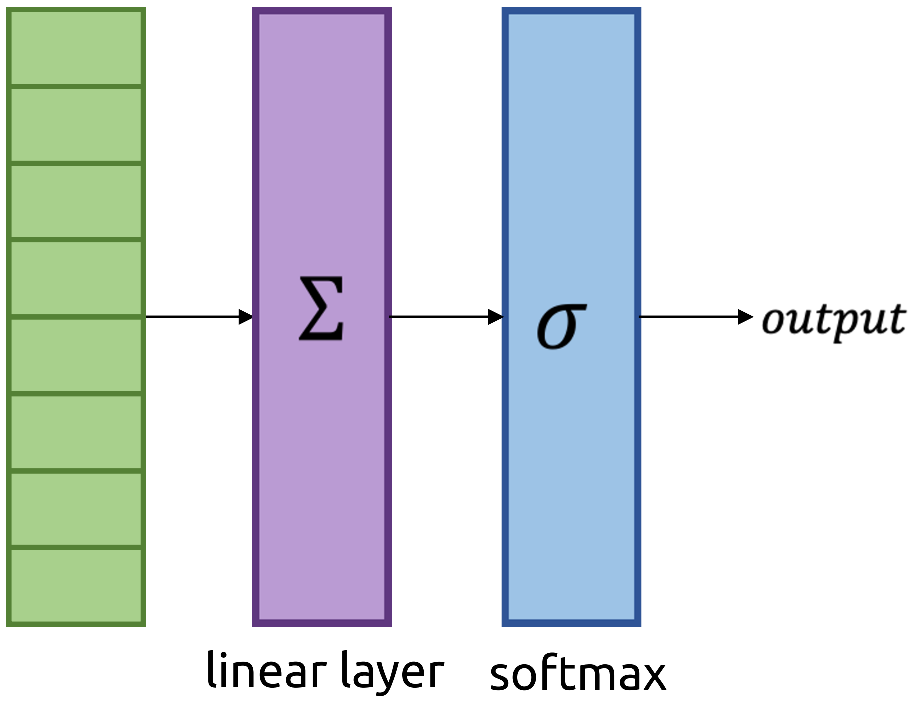
Filter Size

Strides along each dimension

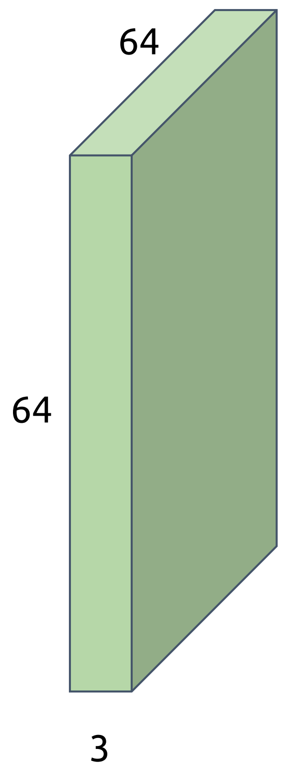
Type of Padding (VALID or SAME)

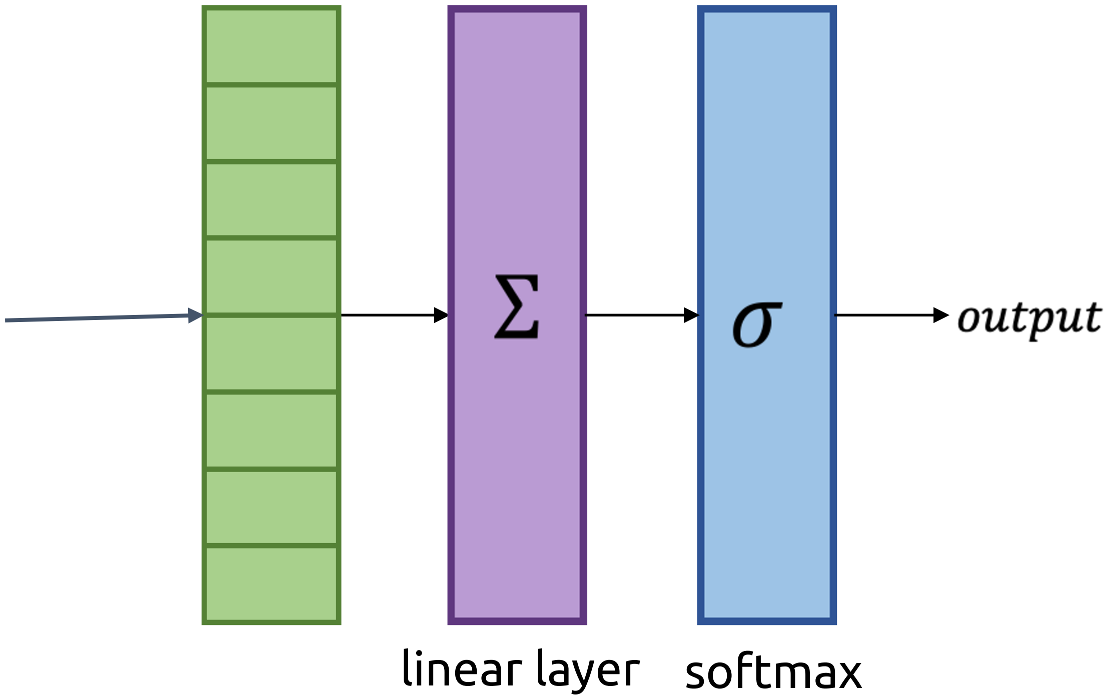
Full documentation here: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D

Our neural network so far

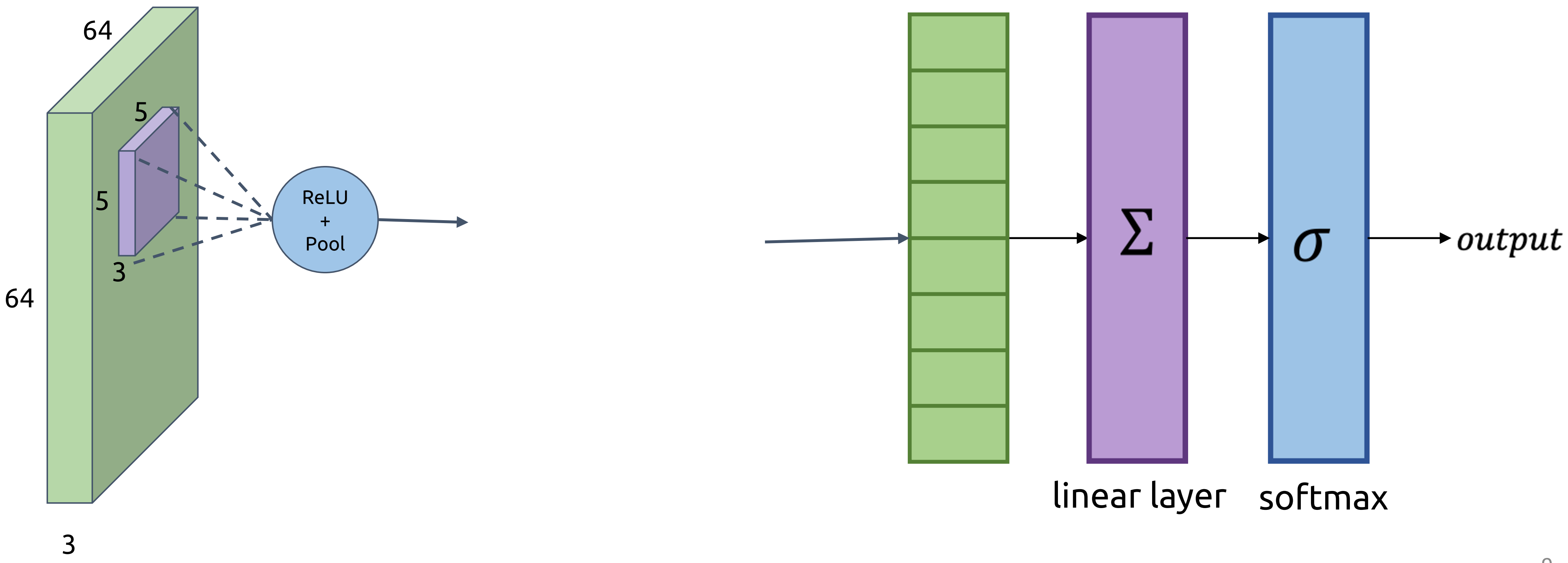


Convolutional Neural Network Architecture

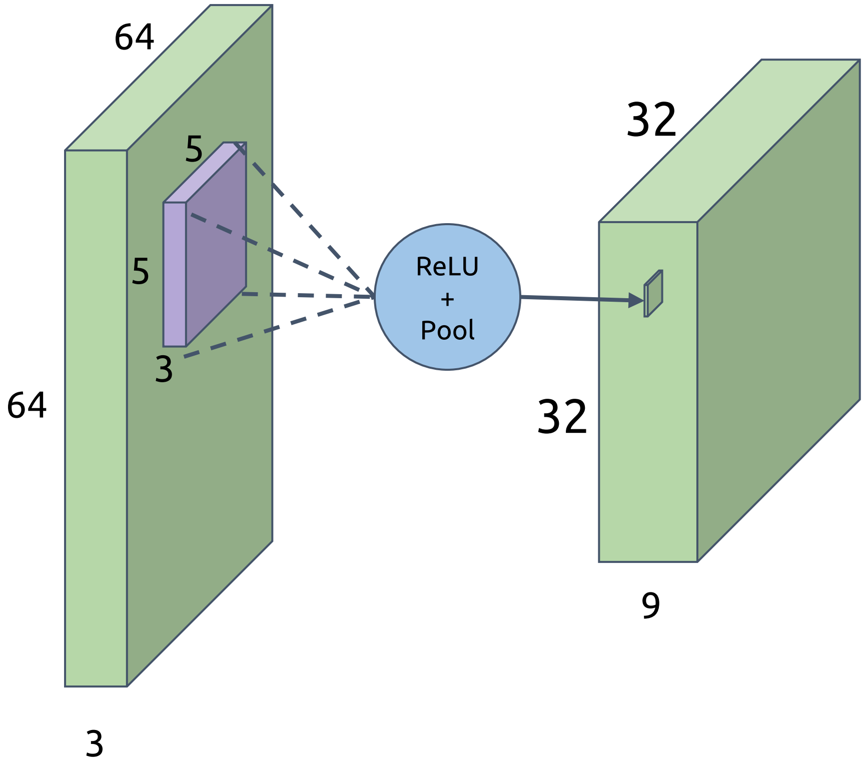


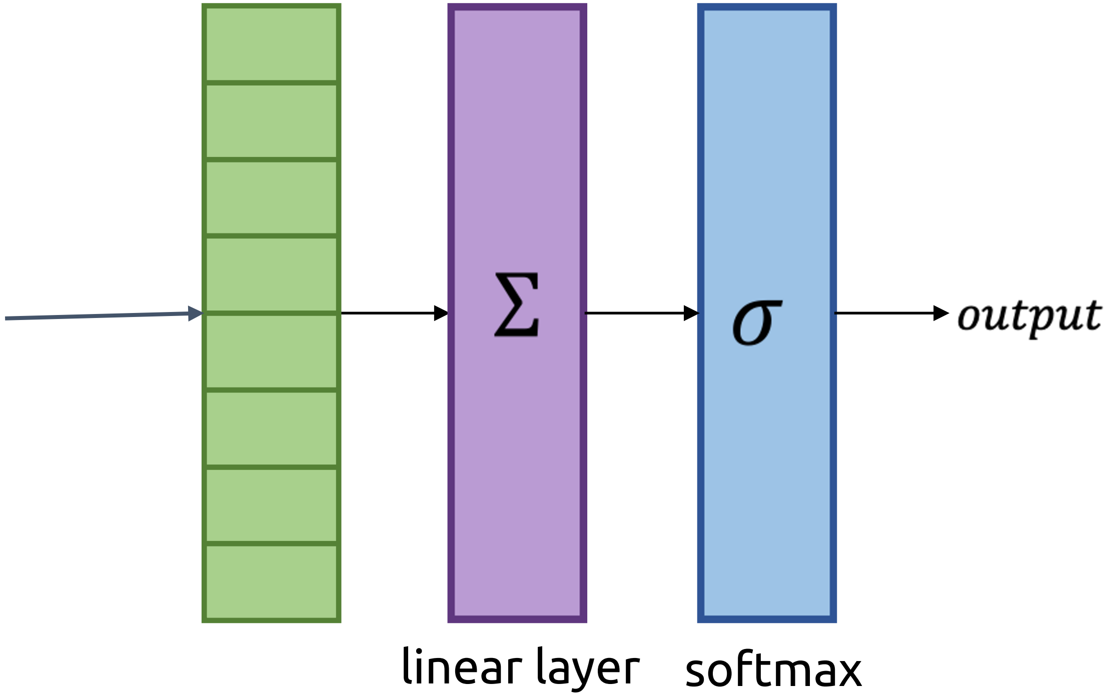


CNN Architecture

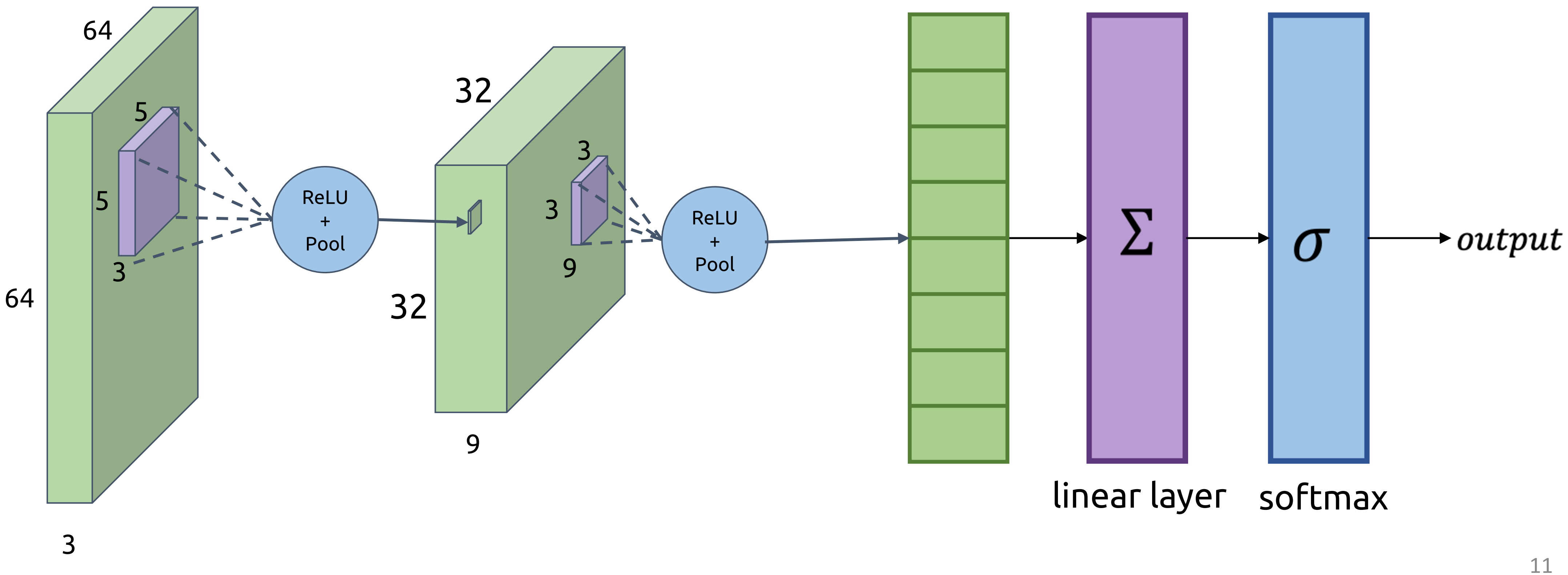


CNN Architecture



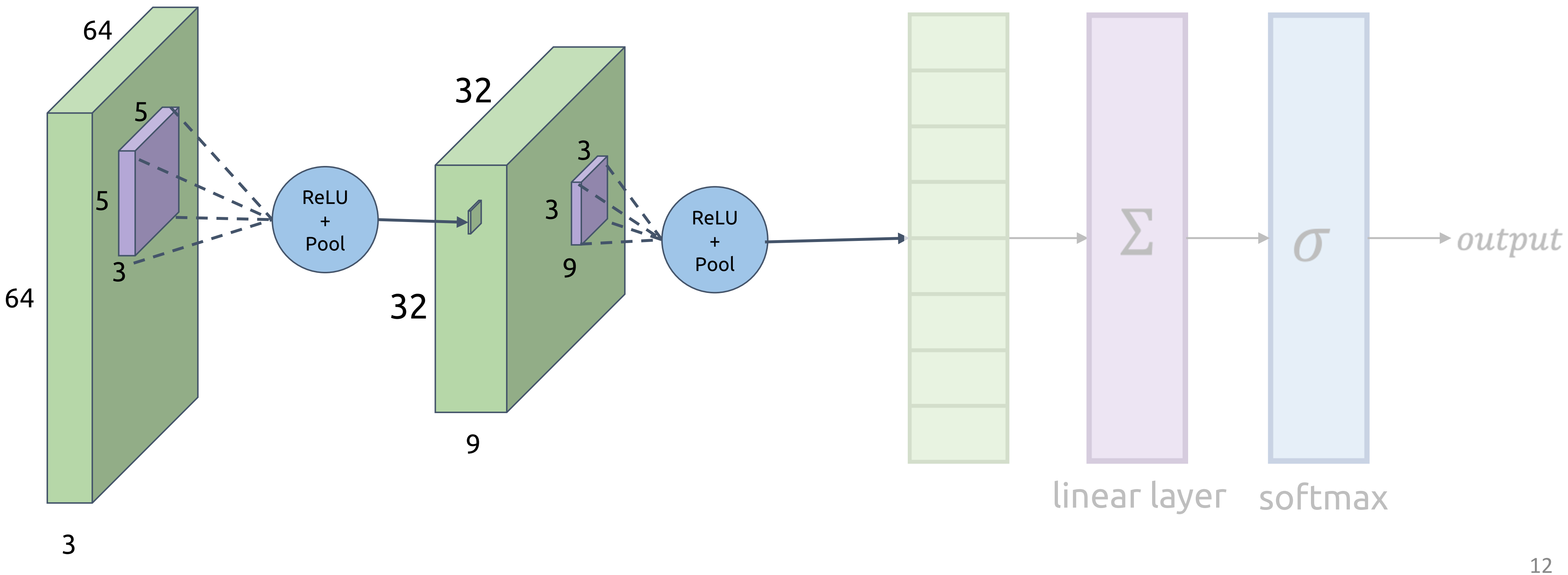


CNN Architecture

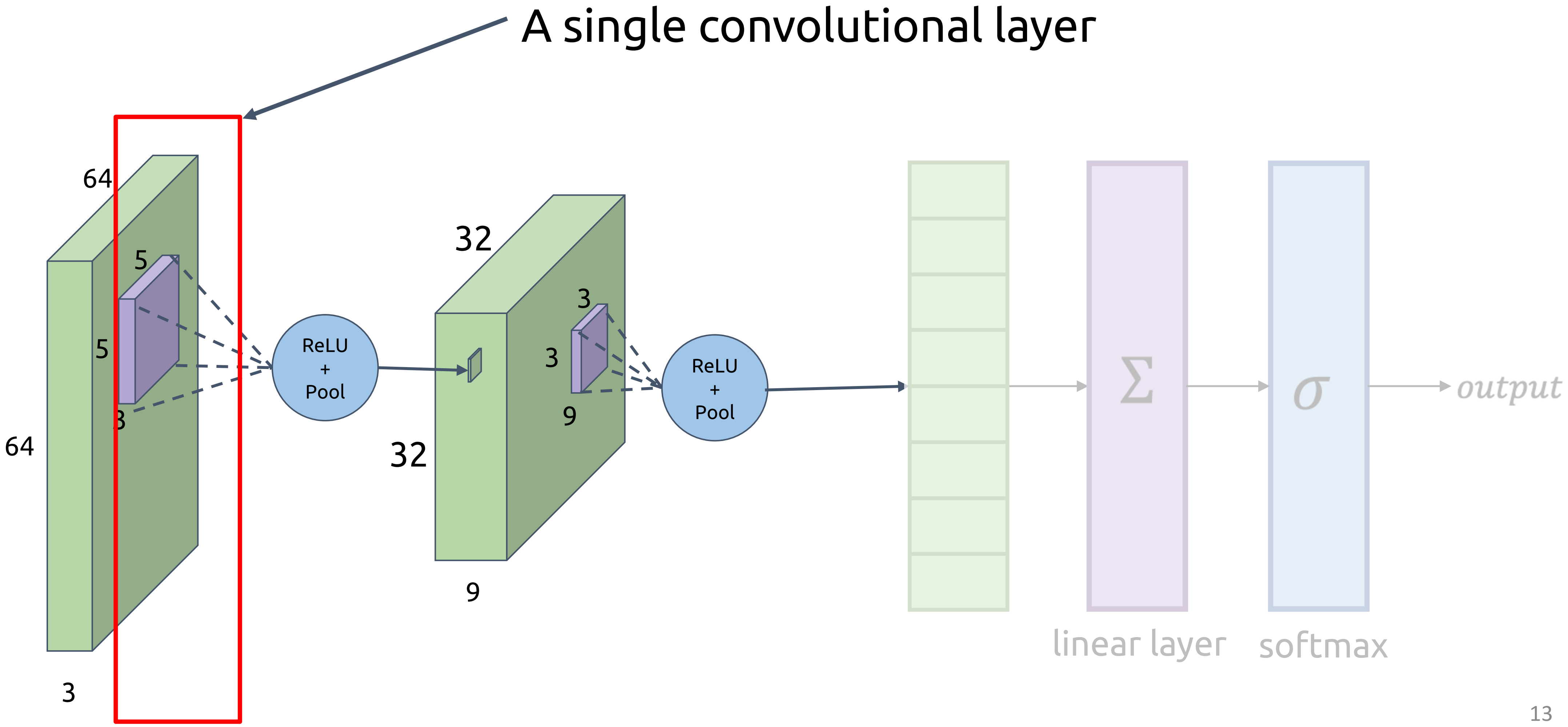


CNN Architecture

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

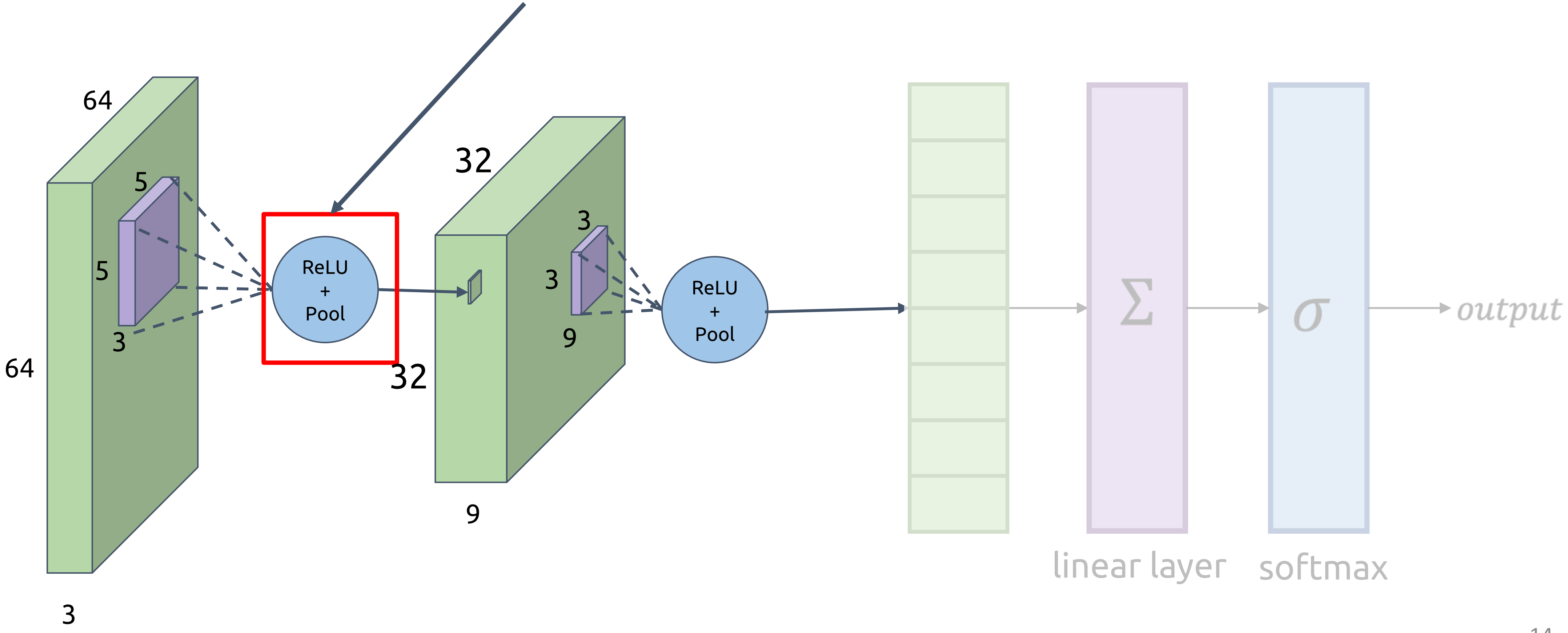


CNN Architecture



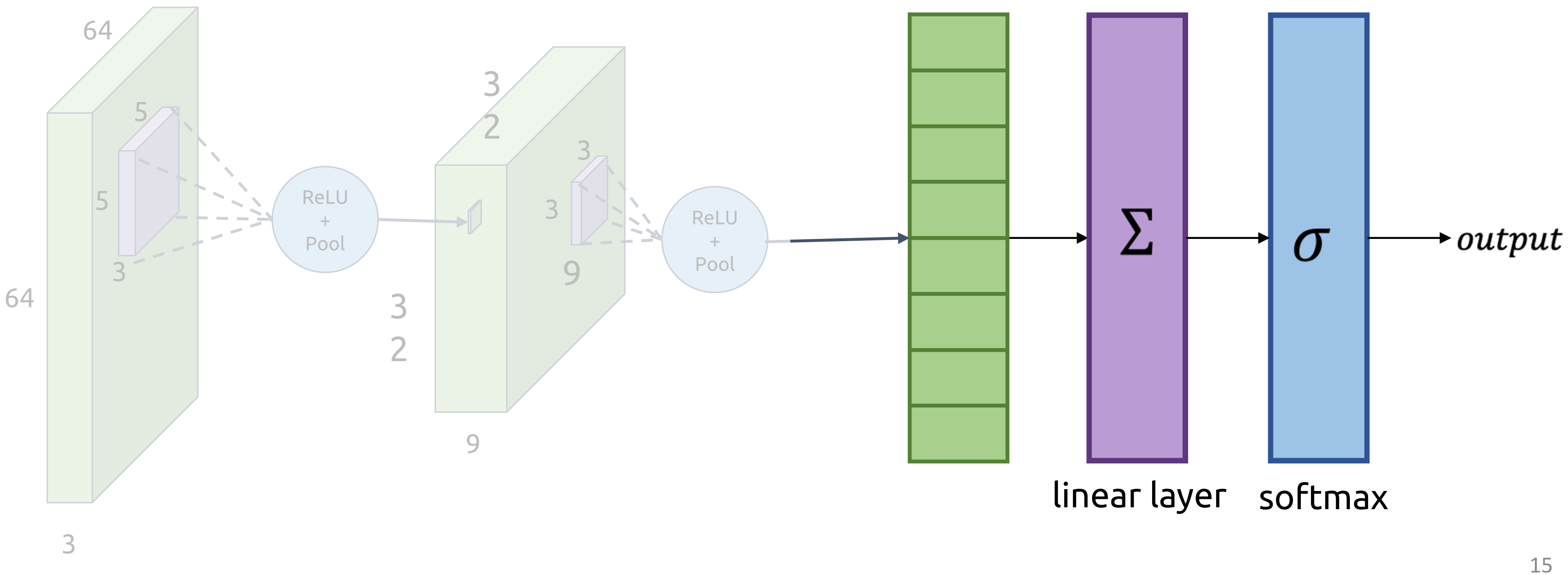
CNN Architecture

Activation after filter passes over image

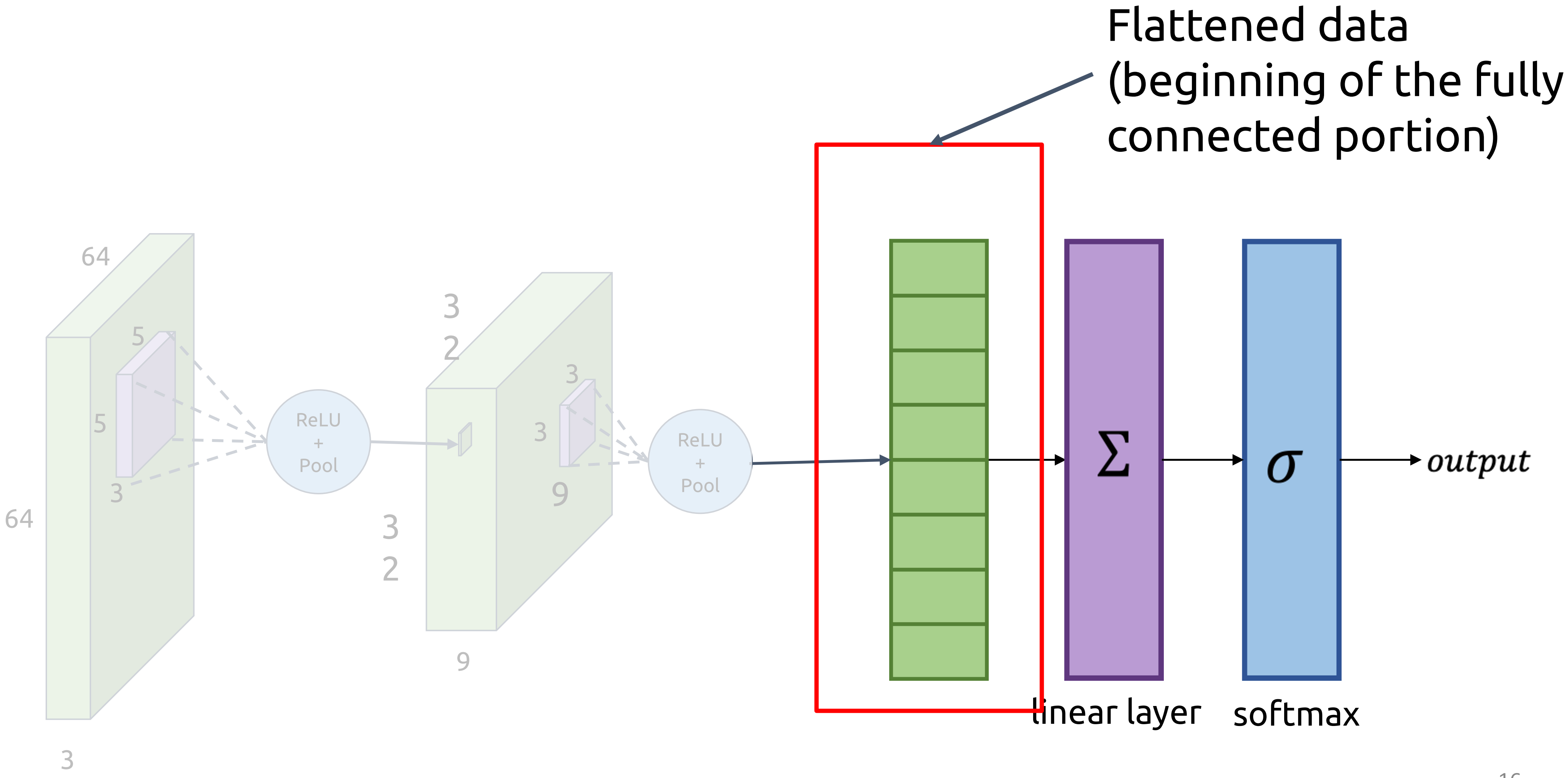


CNN Architecture

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

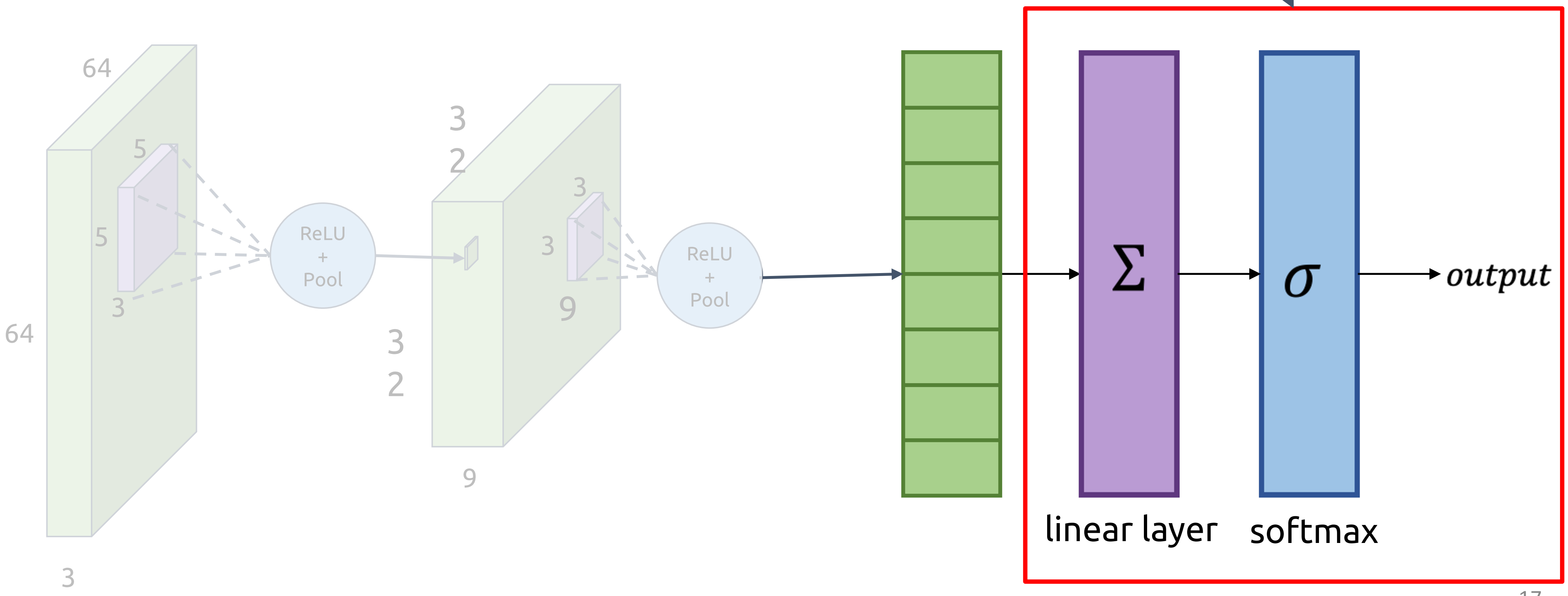


CNN Architecture



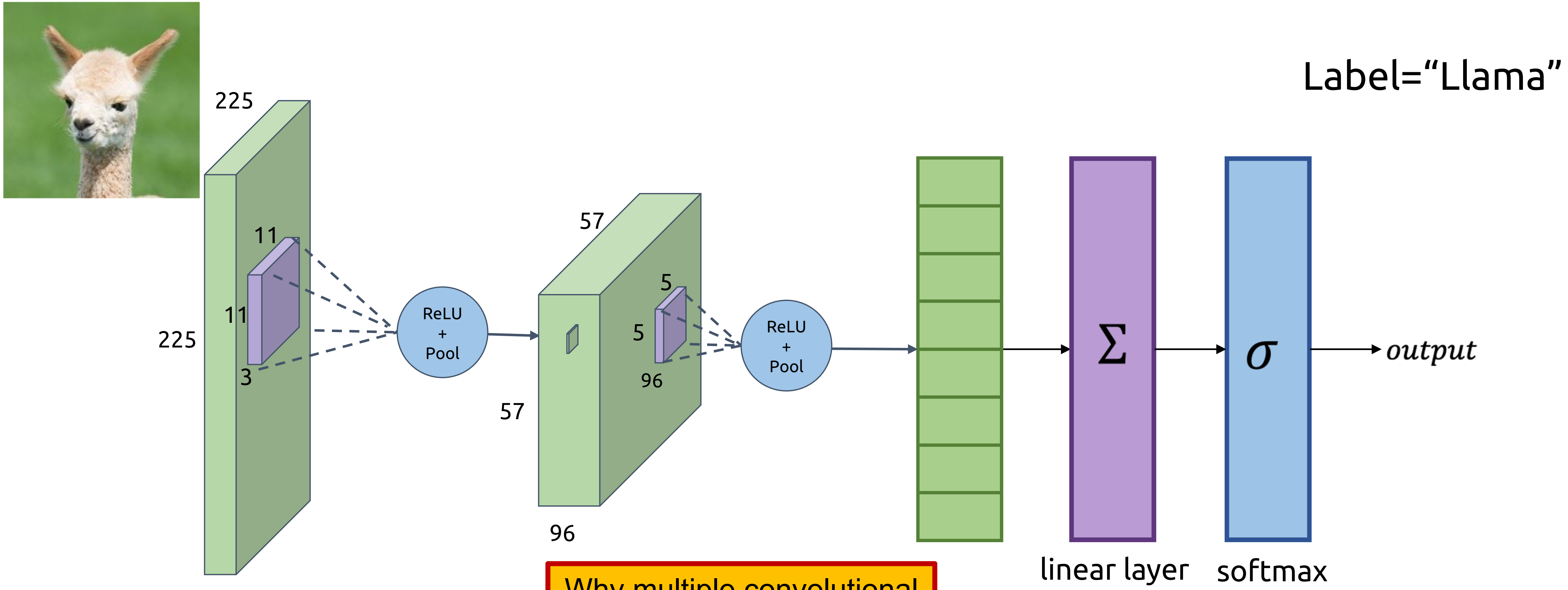
CNN Architecture

Fully connected layers
to classify input



CNN Architecture

Input

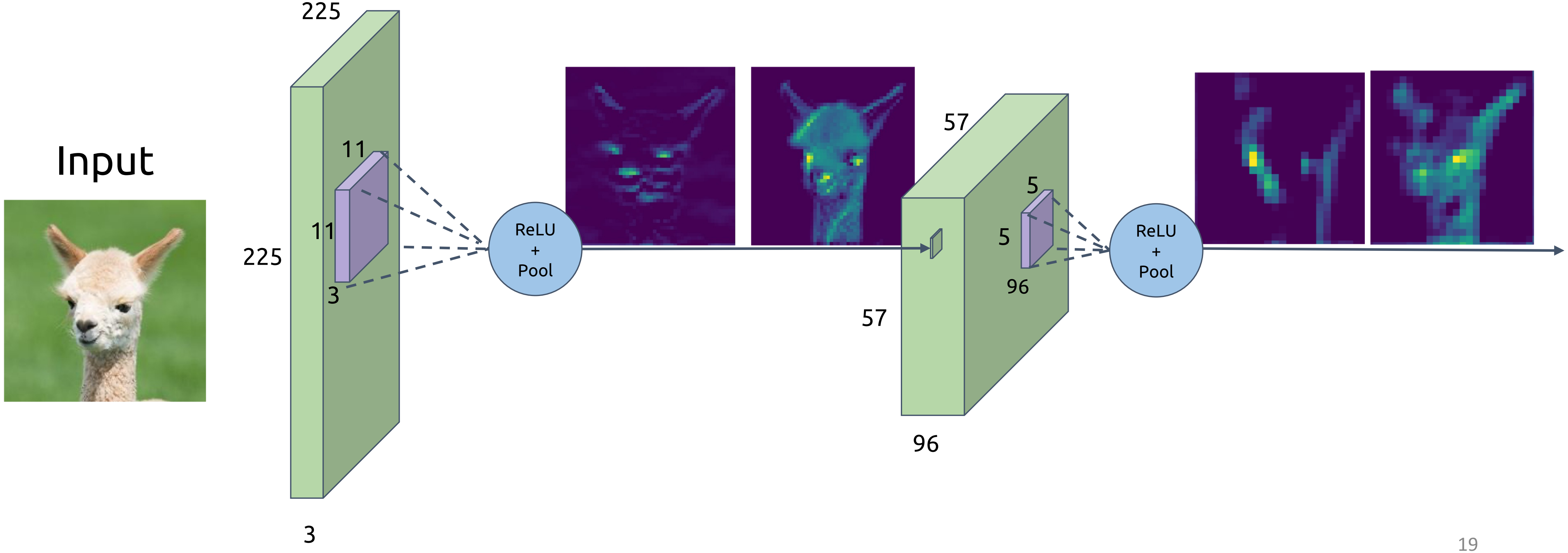


Why multiple convolutional layers?

Feature Extraction using multiple convolution layers

Hierarchy of features

Sequence of layers detect broader and broader features



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

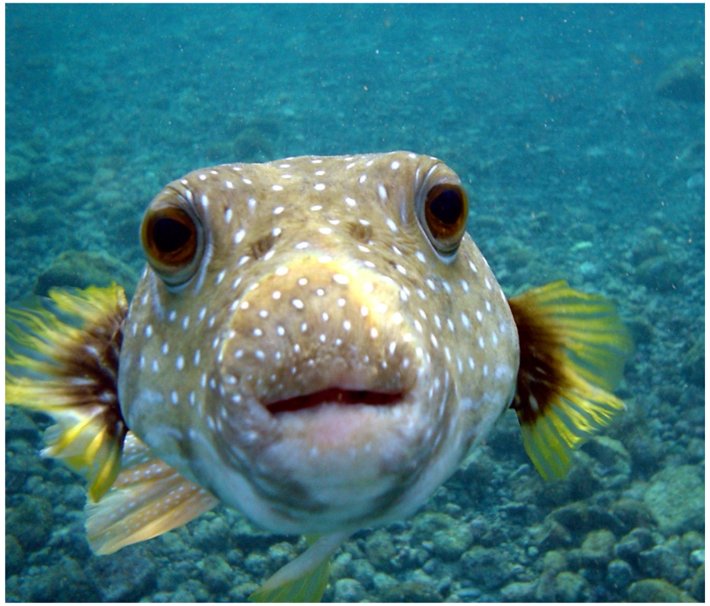
https://commons.wikimedia.org/wiki/File:Common_zebra_1.jpg

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

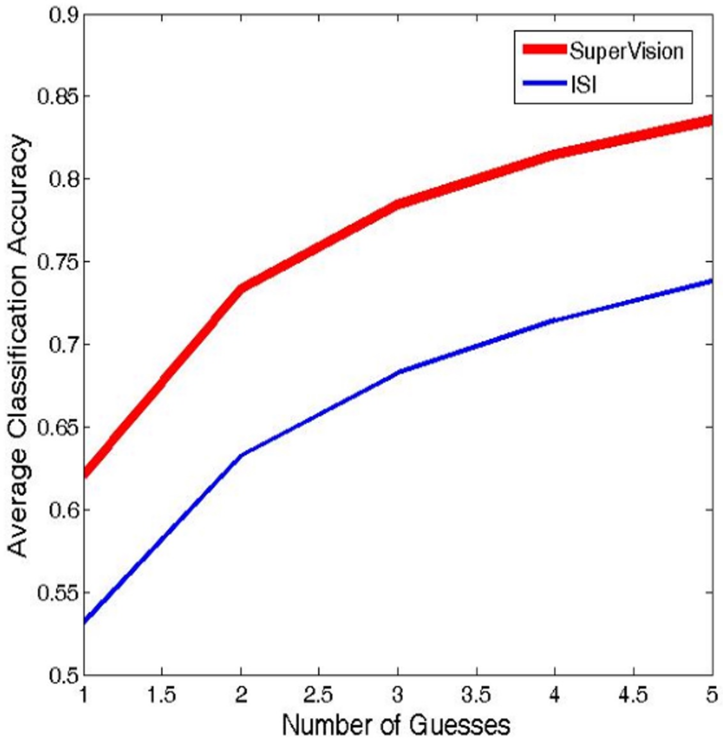
Predictions:

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

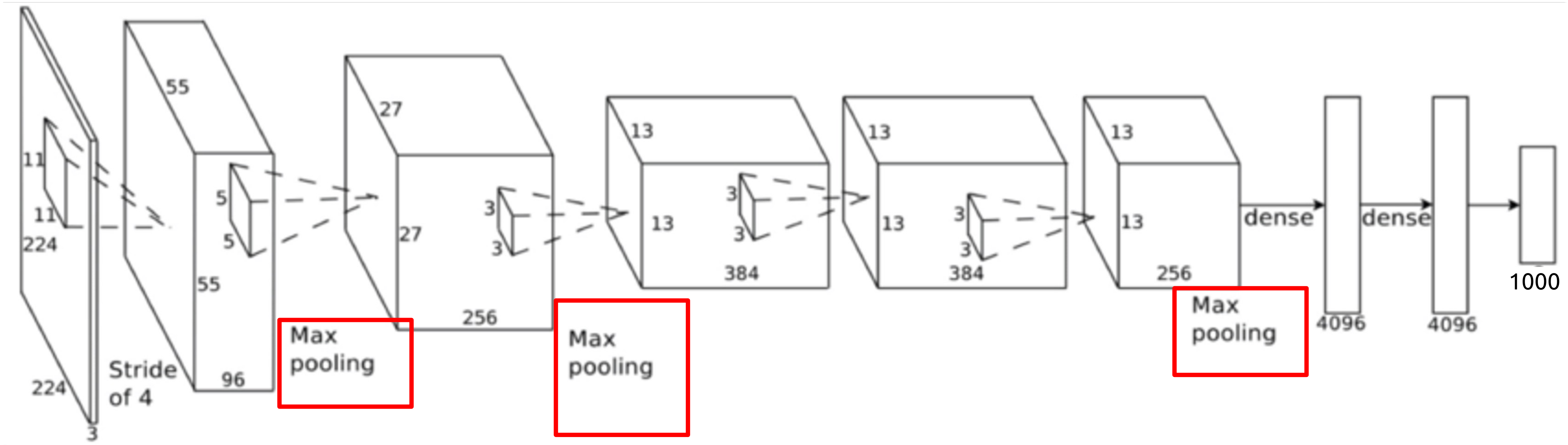


http://image-net.org/challenges/LSVRC/2012/analysis/

Note: SuperVision is the name of Alex's team

AlexNet

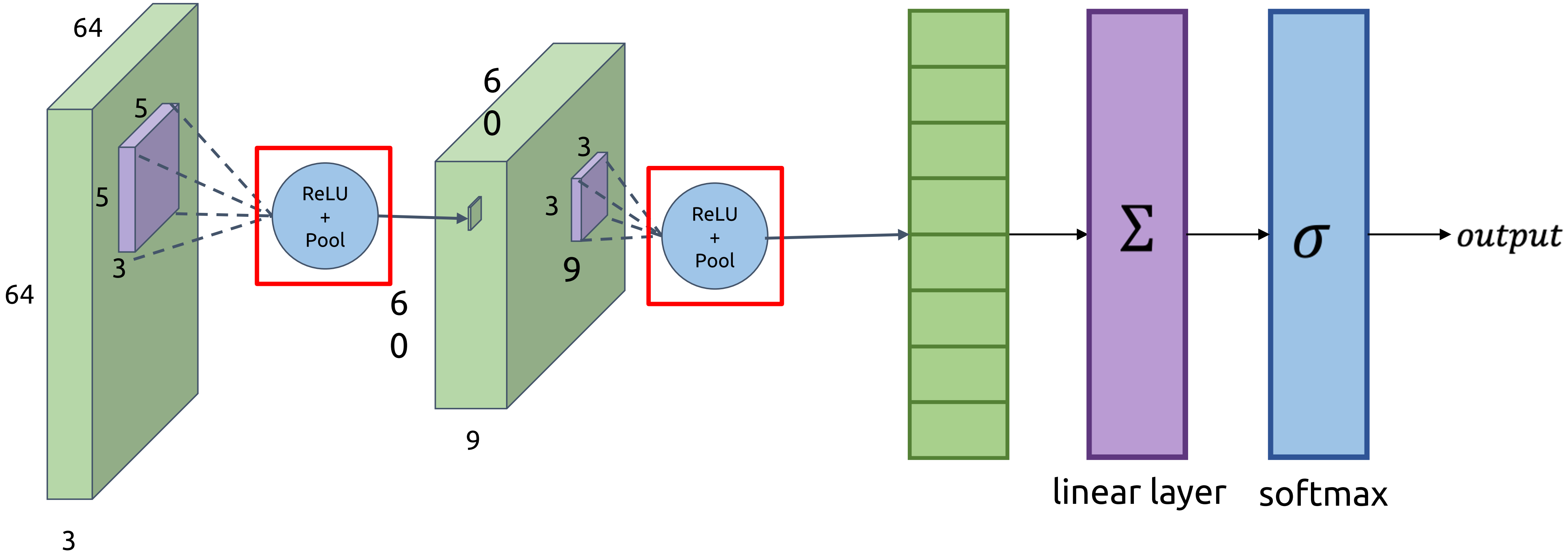
- 60 million parameters
- 5 Convolutional Layers
- 3 Fully Connected Layers



[Alex Krizhevsky et al. 2012]

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

Pooling

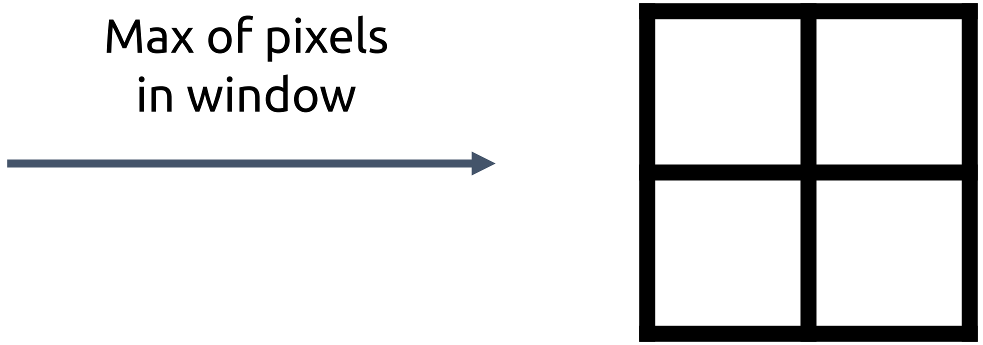


64
64
3
5
5
3
ReLU
+
Pool
6
0
6
0
9
3
3
9
ReLU
+
Pool
Σ
σ
output
linear layer
softmax

Max Pooling

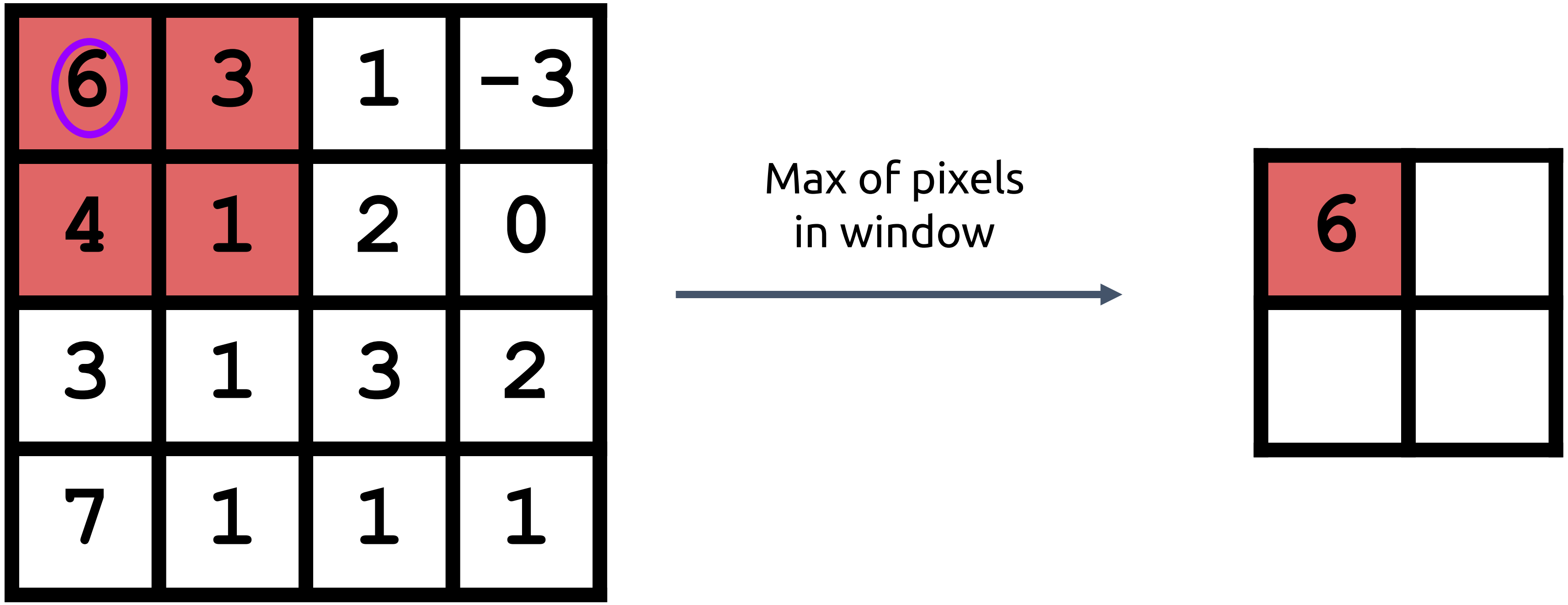
Max pooling with stride 2 and 2x2 filters

6	3	1	-3
4	1	2	0
3	1	3	2
7	1	1	1



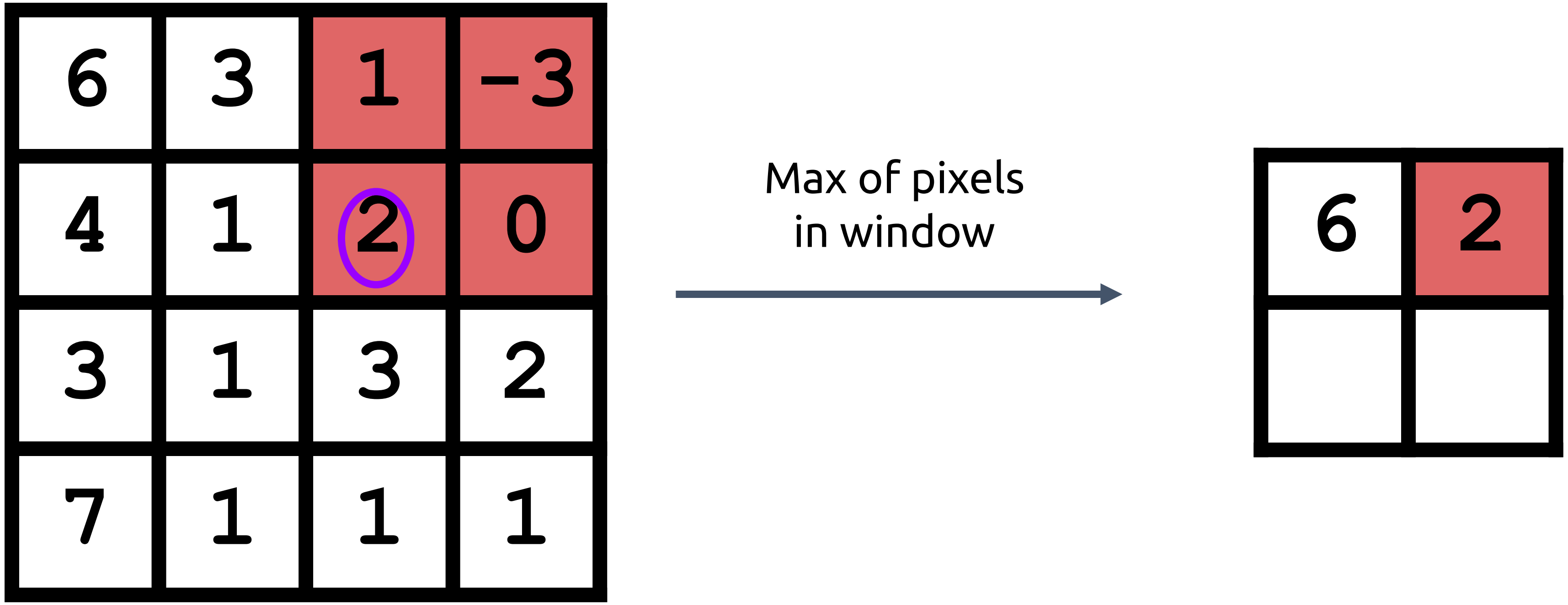
Max Pooling

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

6	3	1	-3
4	1	2	0
3	1	3	2
7	1	1	1

Max of pixels
in window

6	2
7	

Max Pooling

Max pooling with stride 2 and 2x2 filters

6	3	1	-3
4	1	2	0
3	1	3	2
7	1	1	1

Max of pixels
in window

$\longrightarrow$

6	2
7	3

Max Pooling

Max pooling with stride 2 and 2x2 filters

6	3	1	-3
4	1	2	0
3	1	3	2
7	1	1	1

6	2
7	3

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)

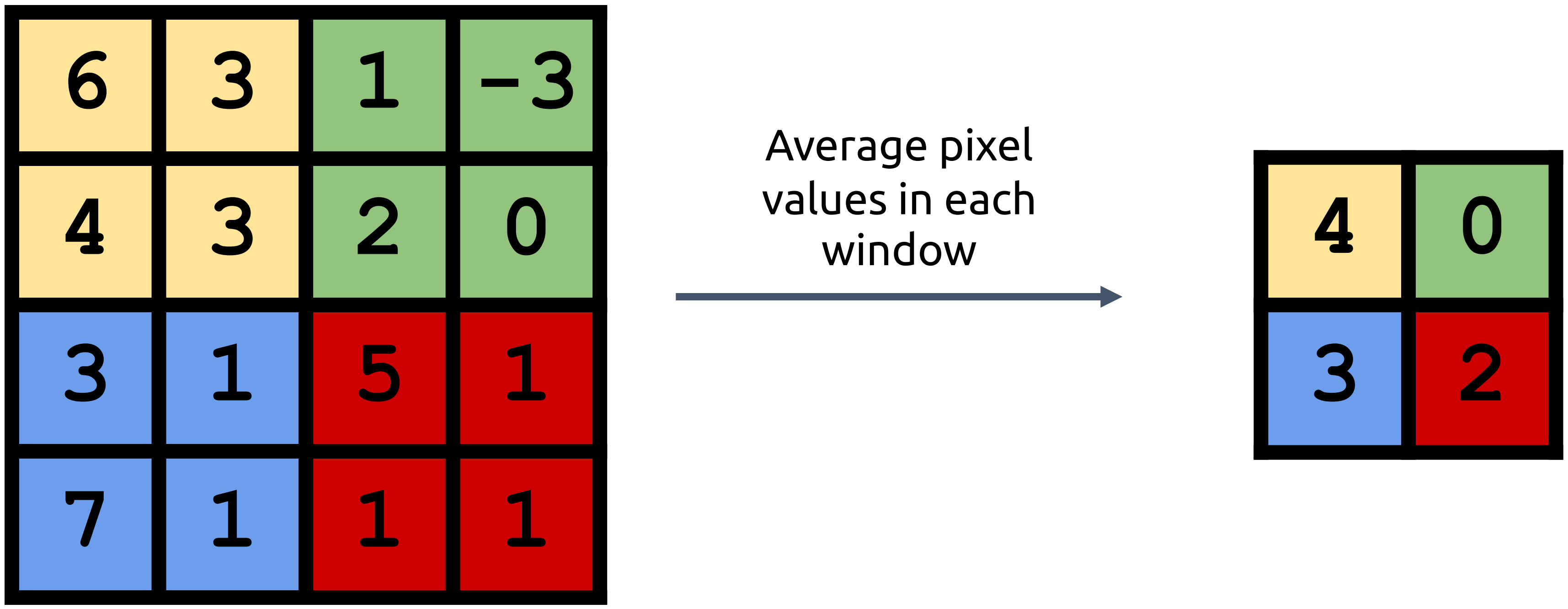
Original Image

Convolution Output

After Pooling

Other Pooling Techniques

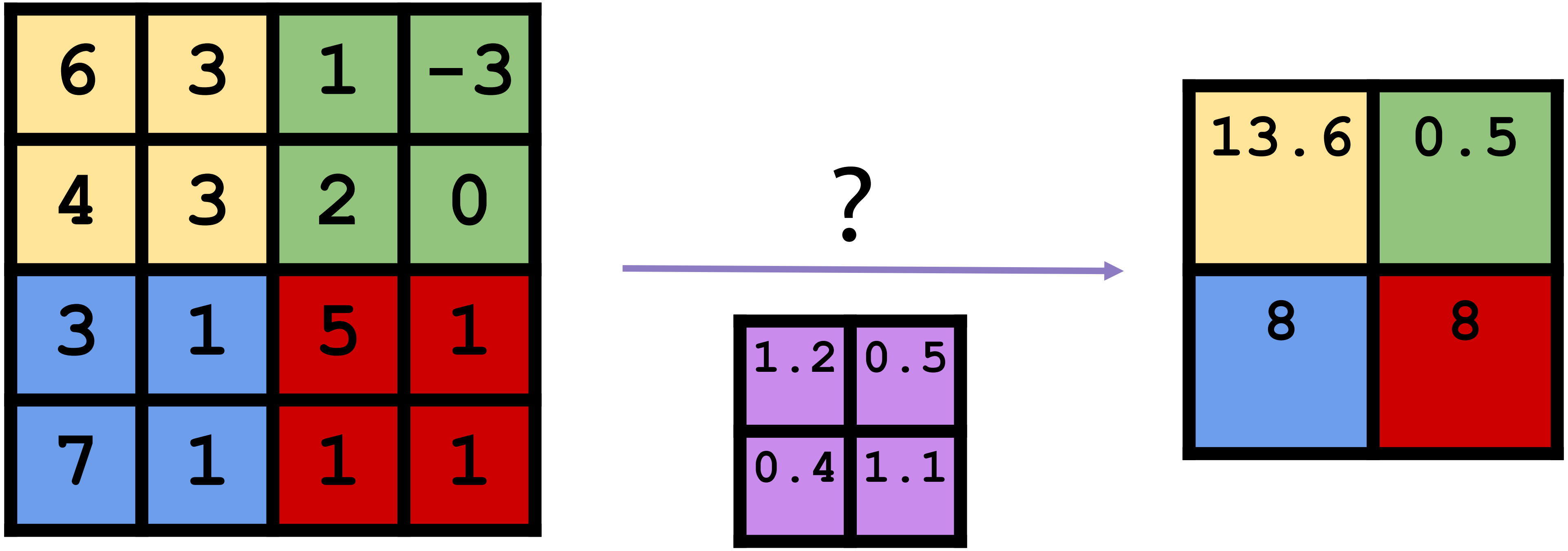
Average pooling with stride 2 and 2x2 filters



Any questions?

Learning a Pooling Function

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

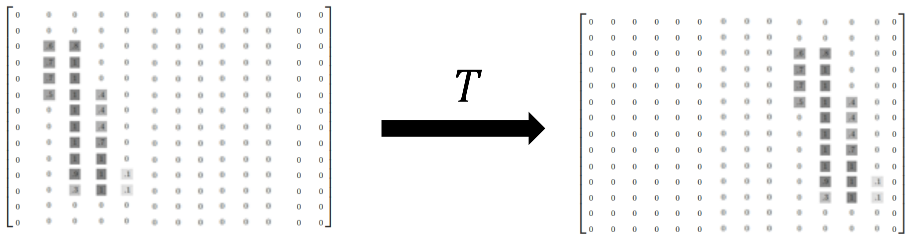


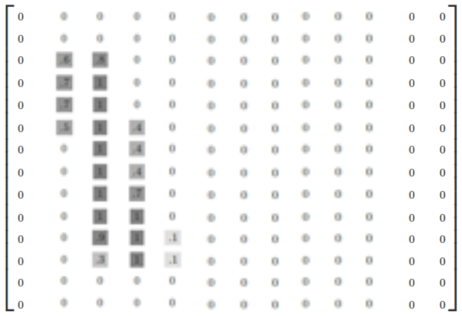
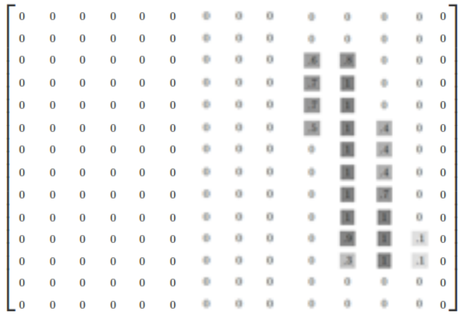
Learned filter weights

So…did we achieve our goal of translational invariance?

What was Translational Invariance again?

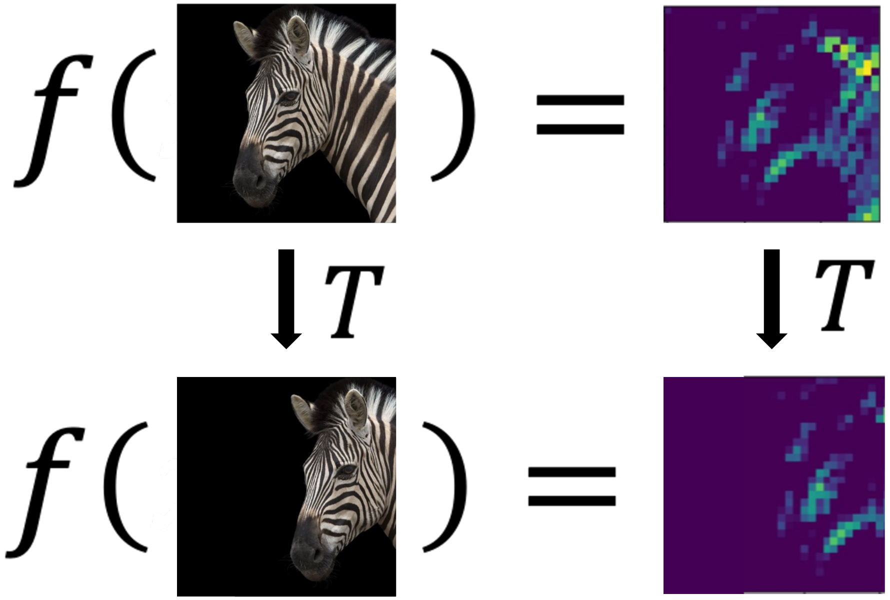
- To make a neural net f robust in this same way, it should ideally satisfy ***translational invariance***: $f\big(T(x)\big) = f(x)$, where
  - x is the input image
  - T is a translation (i.e. a horizonal and/or vertical shift)



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

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

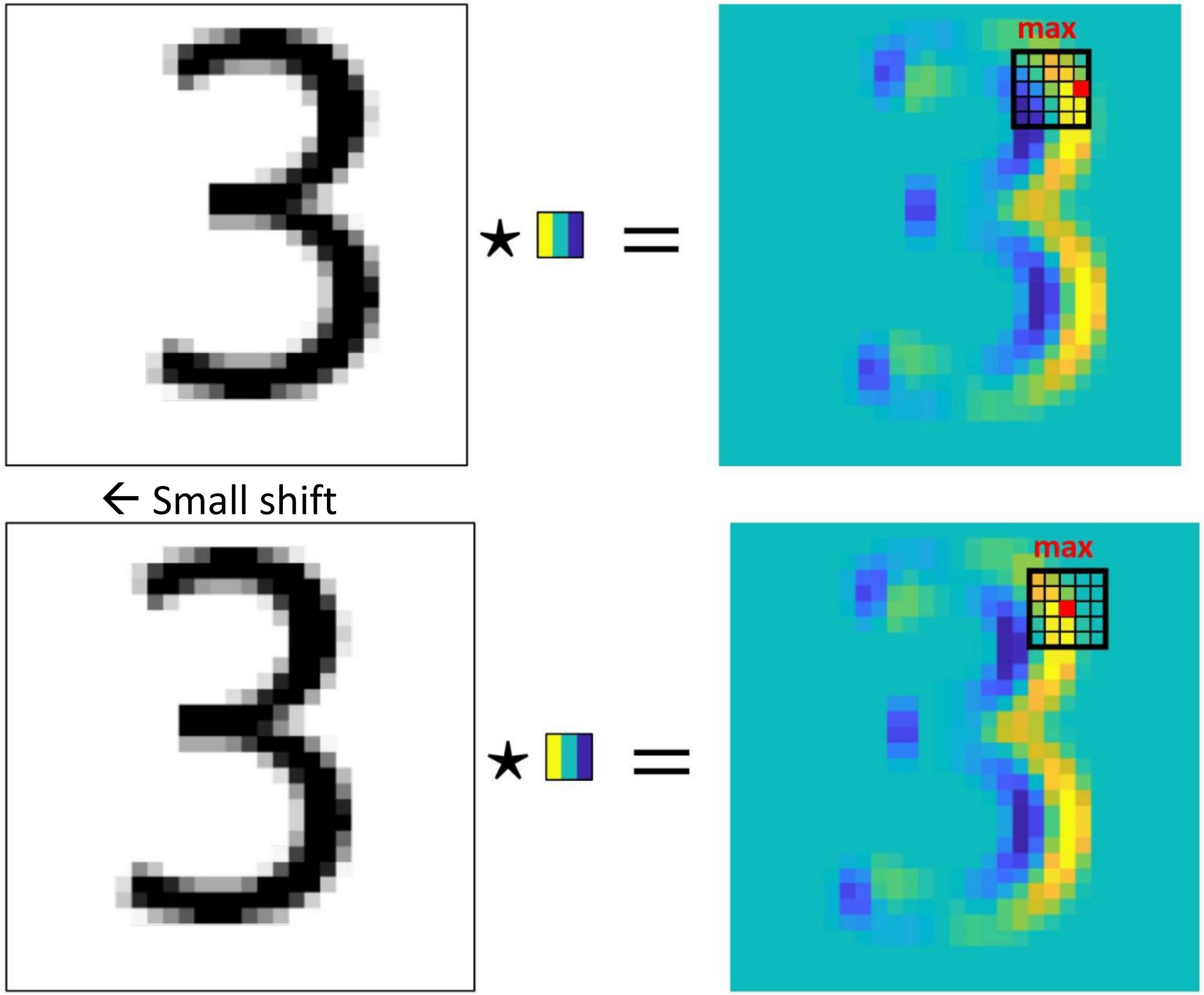
$$f\left(\begin{bmatrix}6 & 3\\4 & 1\end{bmatrix}\right) = 6$$

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

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

https://divsoni2012.medium.com/translation-invariance-in-convolutional-neural-networks-61d9b6fa03df

So how does it all come together?



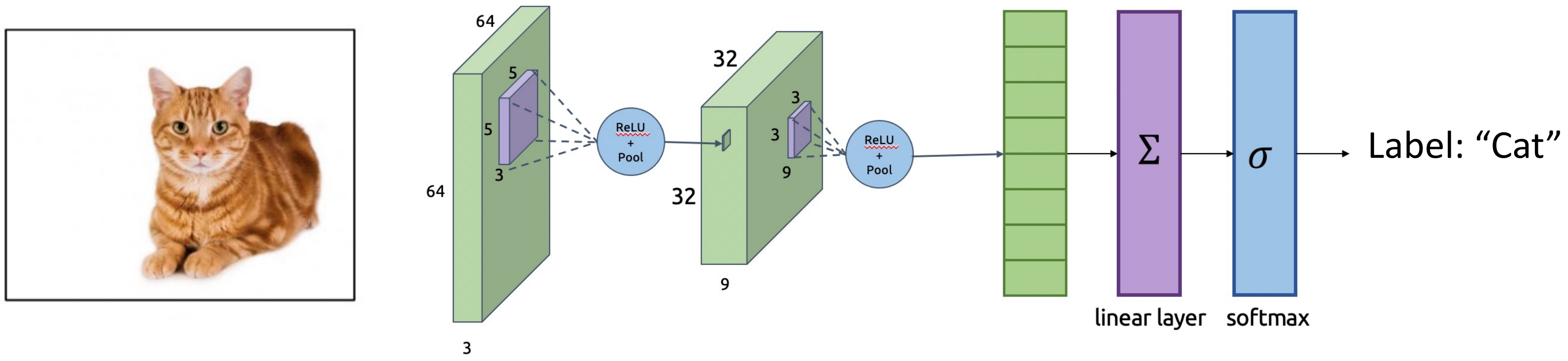
Convolution is ***translation equivariant***

Max pooling gives invariance to small translations

https://www.doc.ic.ac.uk/~bkainz/teaching/DL/notes/equivariance.pdf

Are CNNs Translation Invariant?

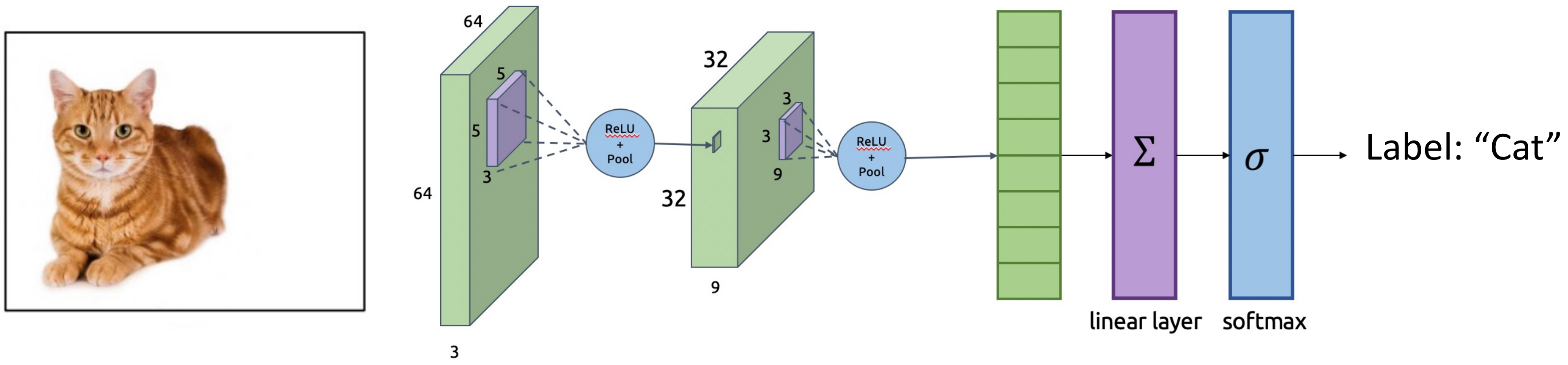
- Answer: CNNs are **“sort of” translation invariant**
  - Shifting the content of the image around tends not to drastically effect the output classification probabilities...



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?

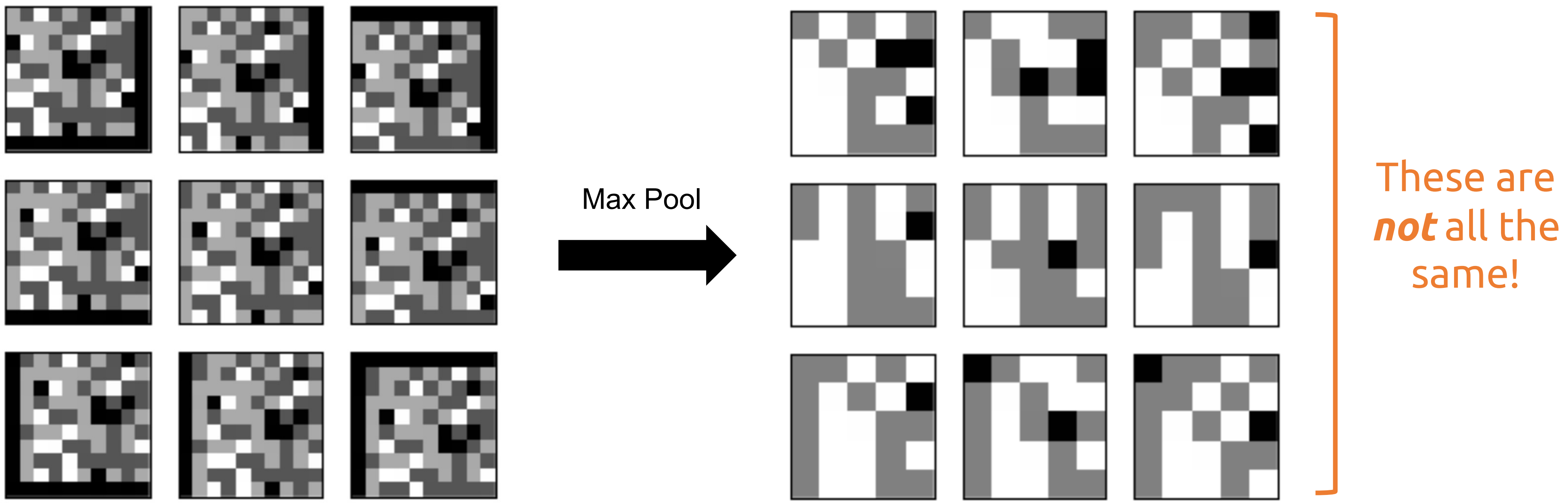
- Answer: CNNs are **“sort of” translation invariant**
    - Shifting the content of the image around tends not to drastically effect the output classification probabilities...



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...
  - ...but they are ***not***, strictly speaking, translation invariant



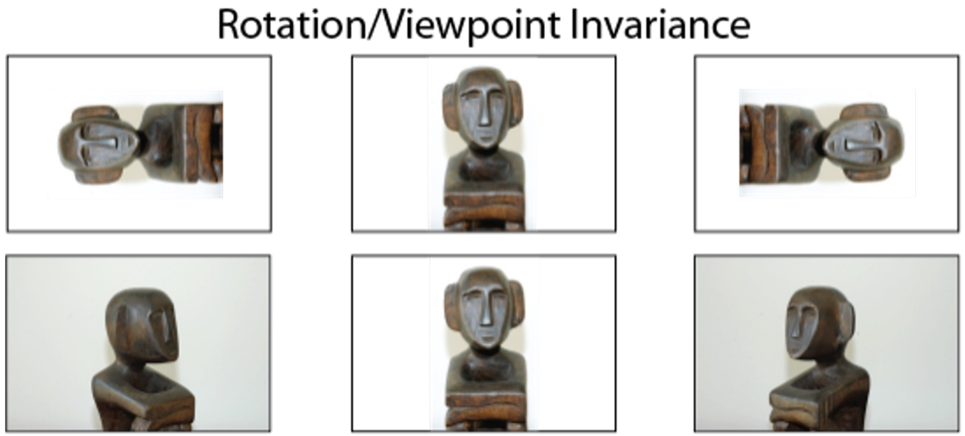
https://dsp.stackexchange.com/questions/24900/translation-invariance-in-max-pooling-and-cascading-with-convolutional-layer

Are CNNs Translation Invariant?

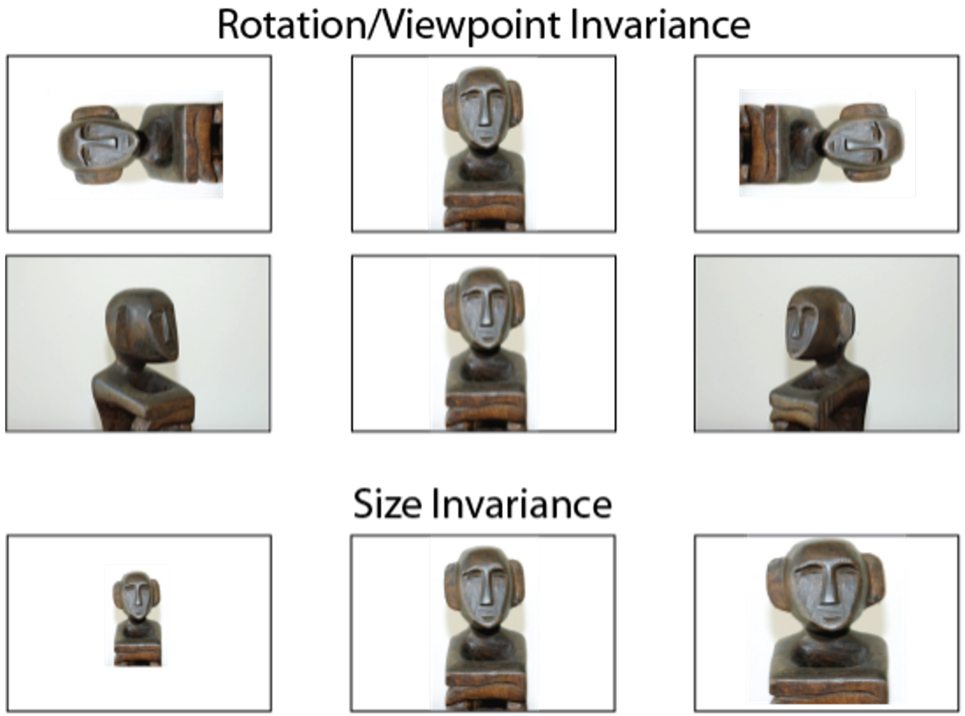
- Is it possible to build a truly translation invariant CNN?
  - Yes!
  - Have to properly “pre-filter” images before pooling them
  - Comes from signal processing theory (The Sampling Theorem)
  - Take CS 1230 (Computer Graphics) if you want to learn about this!

- One effort to make a translation-invariant CNN: https://arxiv.org/pdf/1904.11486.pdf

Other Invariances



Other Invariances



Other Invariances

Matt Krause
mattkrau.se

Any questions?

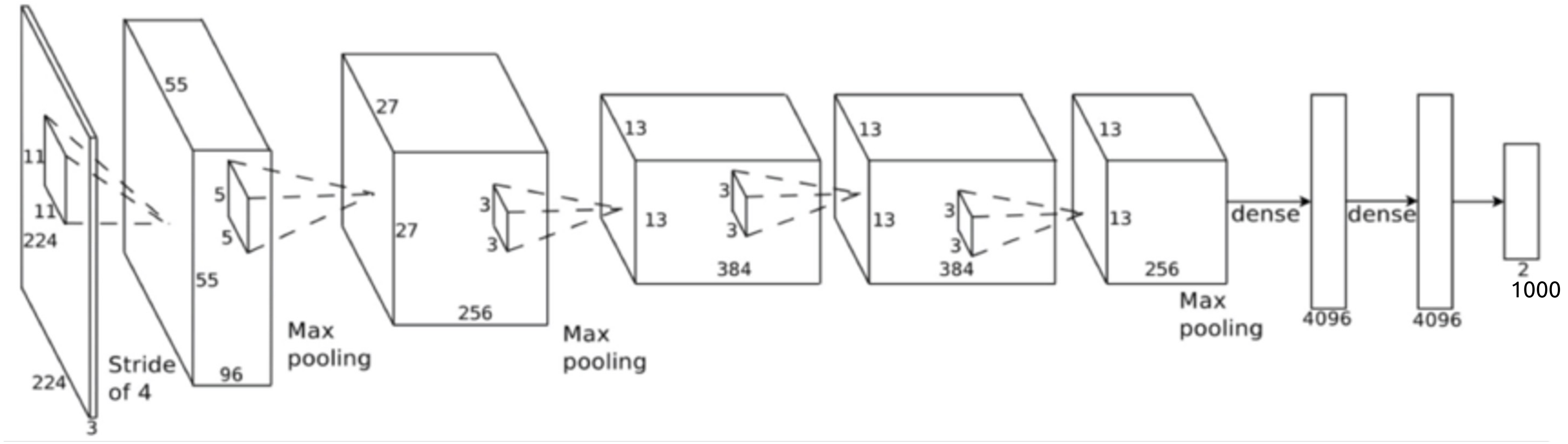
- All of these are desirable
- How do CNNs fare?
  - Max pooling gives some small amount of size invariance...
  - ...but in general, CNNs don't do well with big changes in size, pose, or lighting

What should we we do?

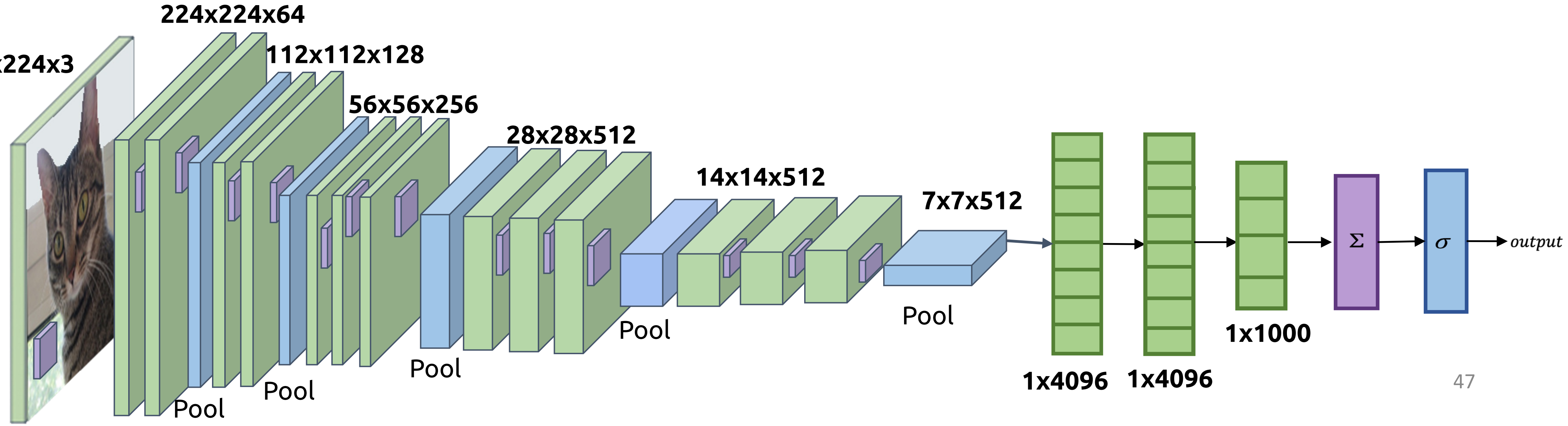
- Consequence of not having these invariances?
  - Need ***lots*** of training data
  - Have to show the network examples of everything under different poses, lighting, etc.
  - Data Augmentation

More Complicated Networks

AlexNet:

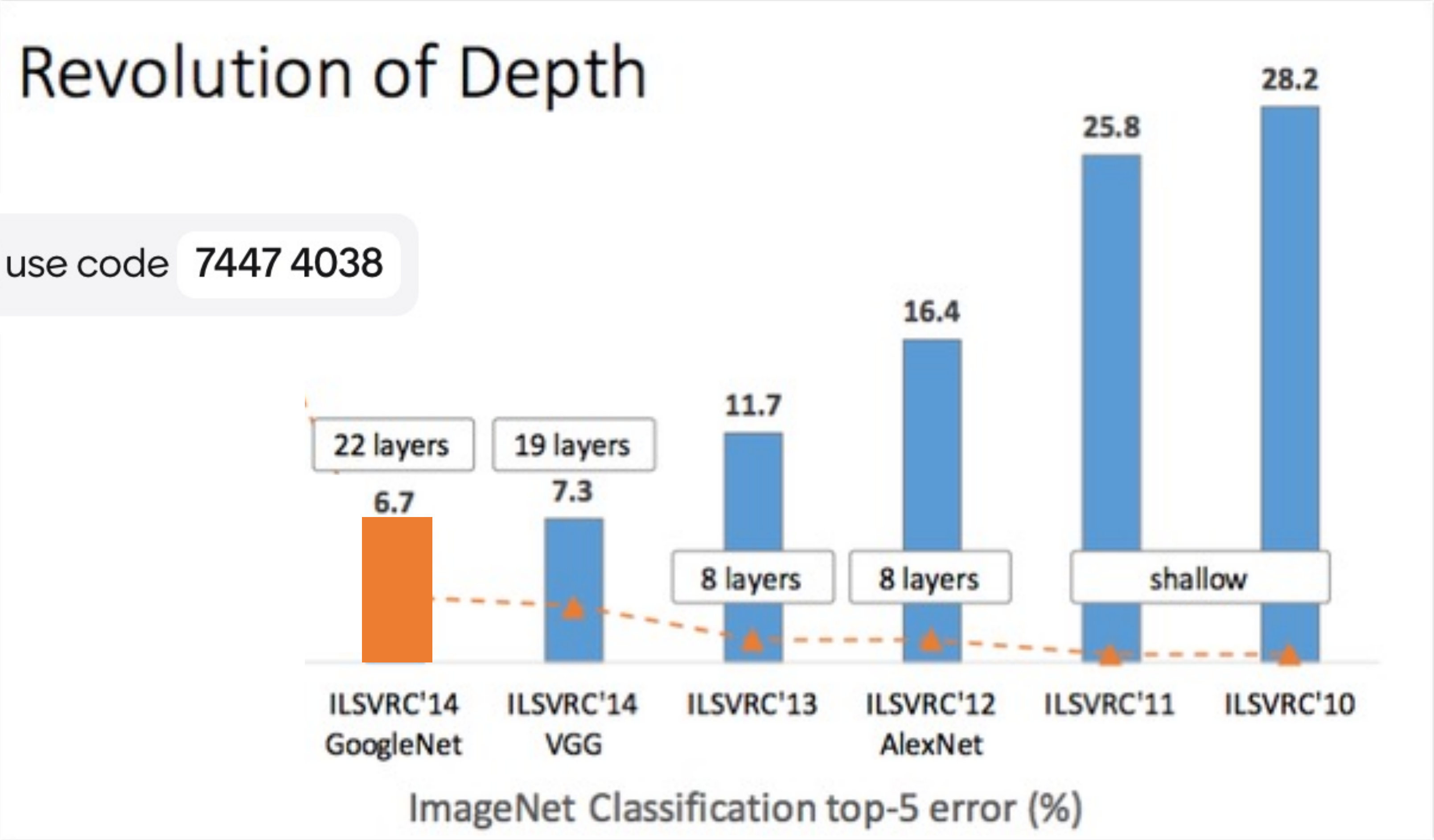


VGG:



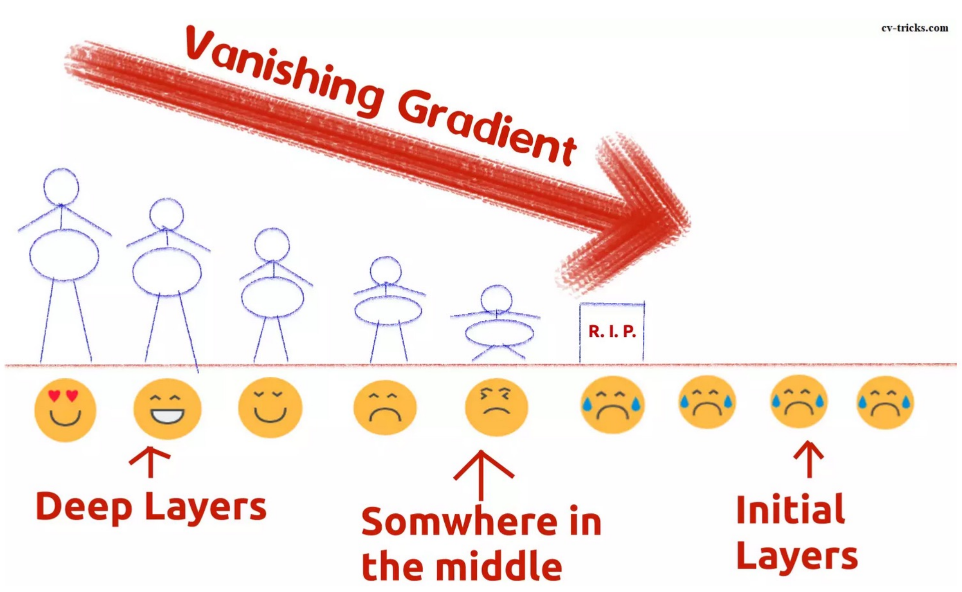
Can you guess what was the biggest bottleneck to adding more layers?

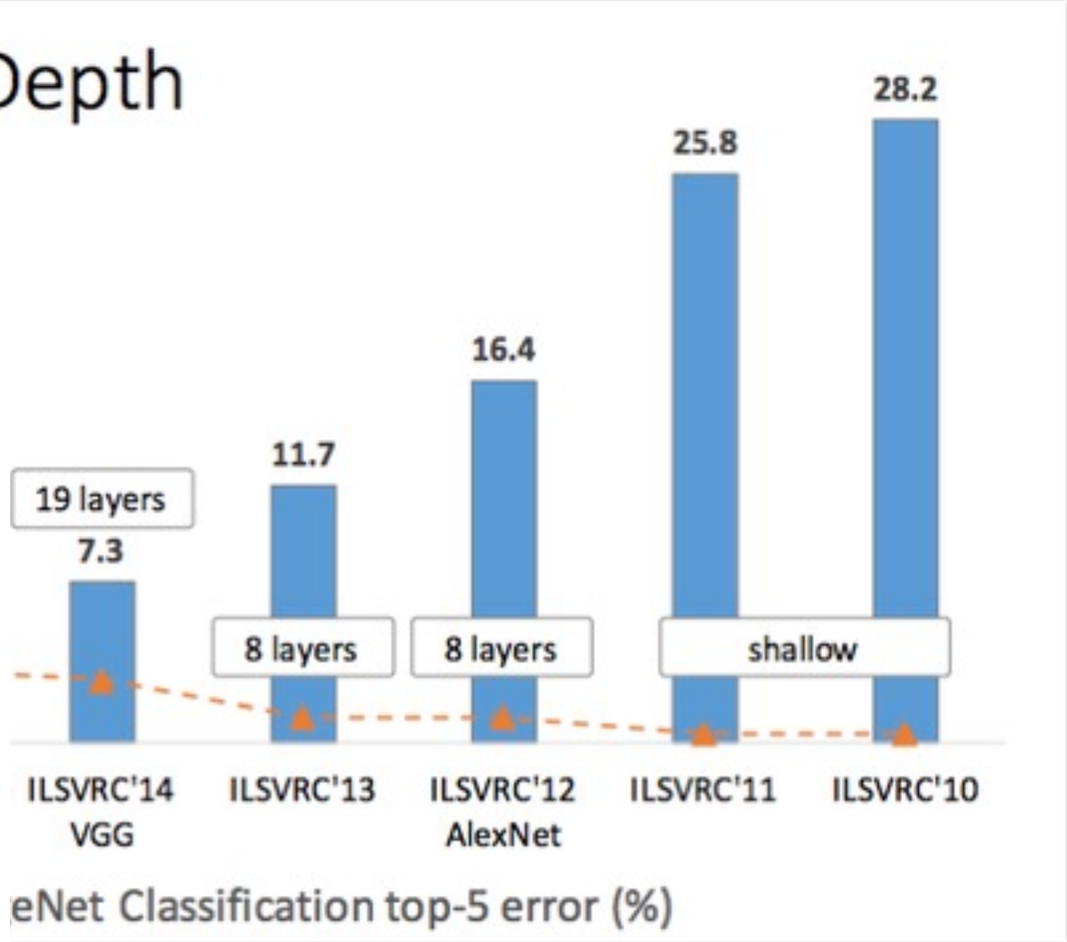
Join at menti.com | use code 7447 4038



https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

Vanishing gradients!



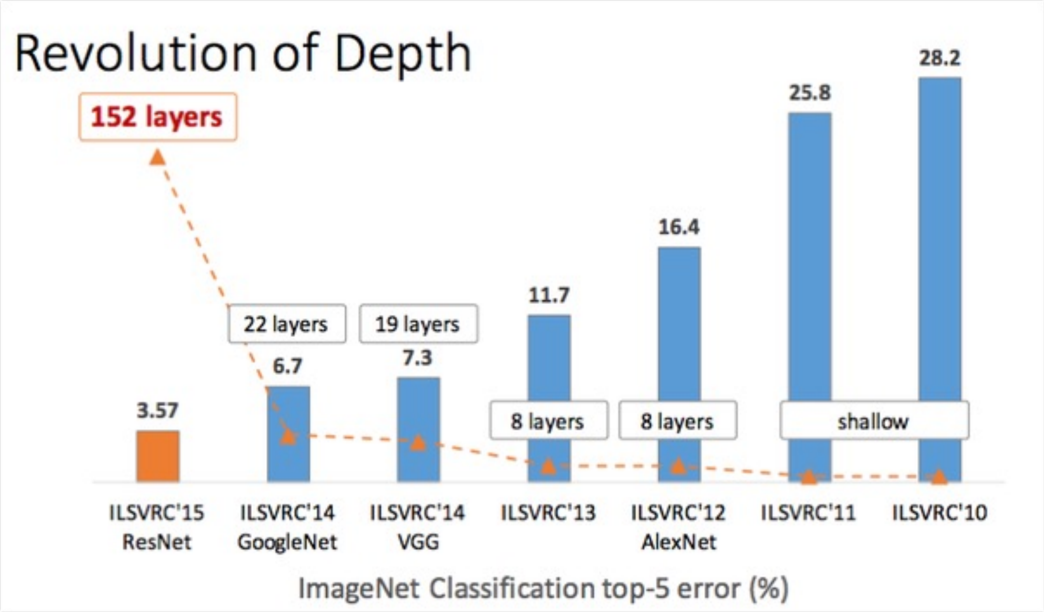


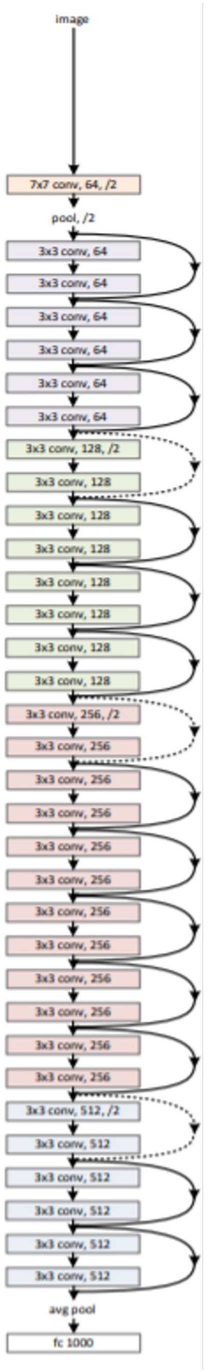
https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33, https://cv-tricks.com/keras/understand-implement-resnets/

More Complicated Networks

ResNet:

Lots of layers, tons of learnable parameters
Avoids Vanishing Gradient problem
but how?





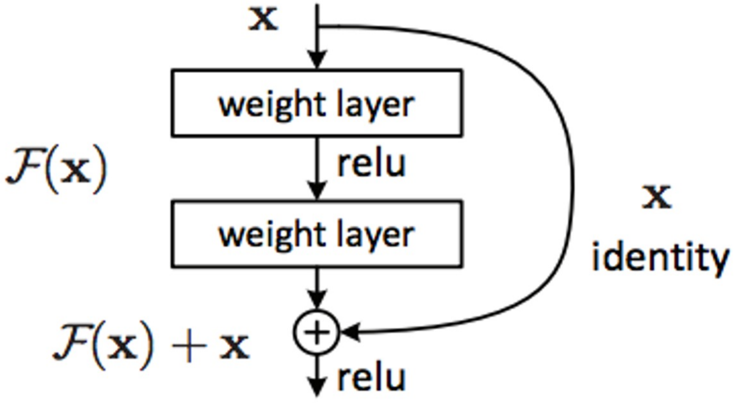
K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

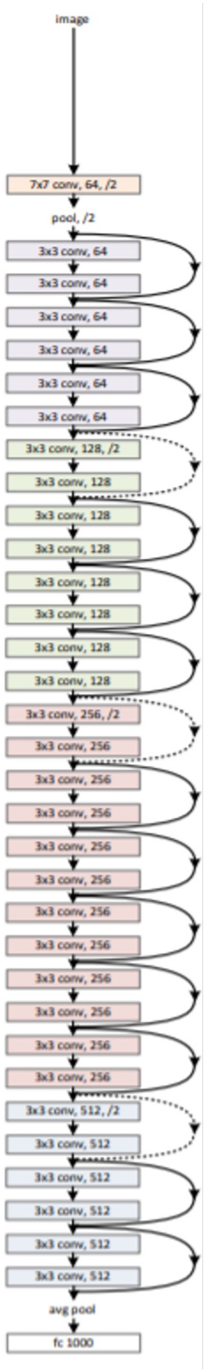
More Complicated Networks

ResNet:

Lots of layers, tons of learnable parameters
Avoids Vanishing Gradient problem

Residual Block

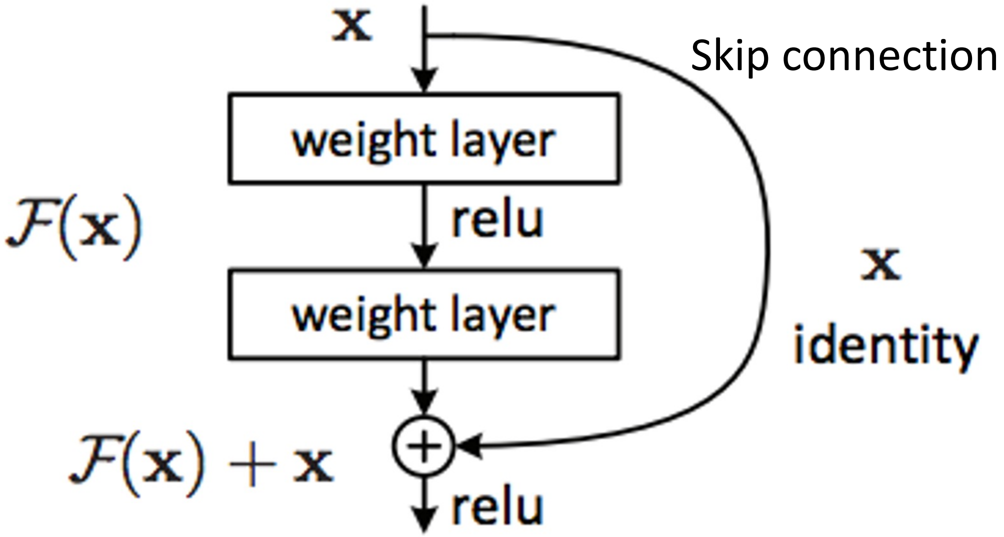




K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

Residual Blocks

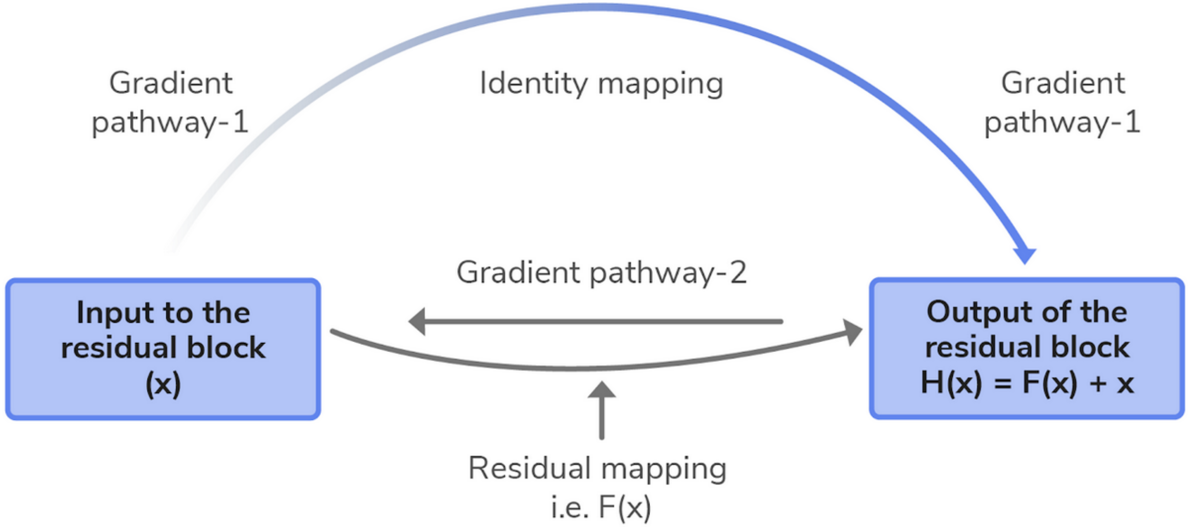
- In very deep nets, each layer often needs to learn just a small transformation of the preceding layer (identity + change)
- Idea: explicitly design the network such that the output of each layer is the identity + some deviation from it
  - Deviation is known as a residual



Any questions?

Residual Blocks

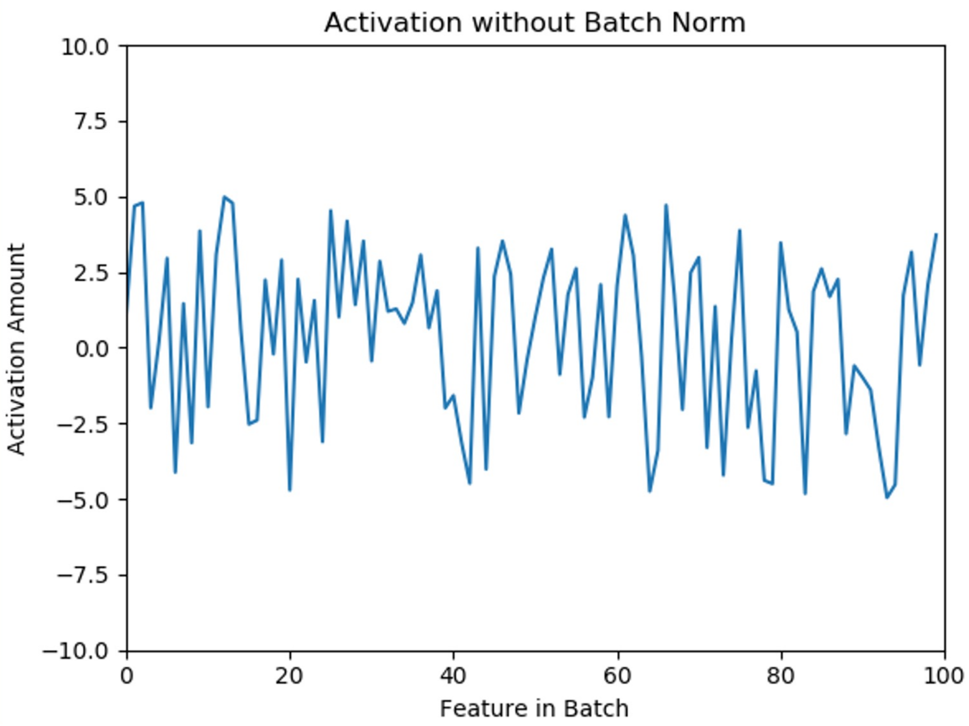
- In very deep nets, each layer often needs to learn just a small transformation of th preceding layer (identity + change)
- Idea: explicitly design the network such that the output of each layer is the ident + some deviation from it
  - Deviation is known as a residual
- Allows gradient to flow through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

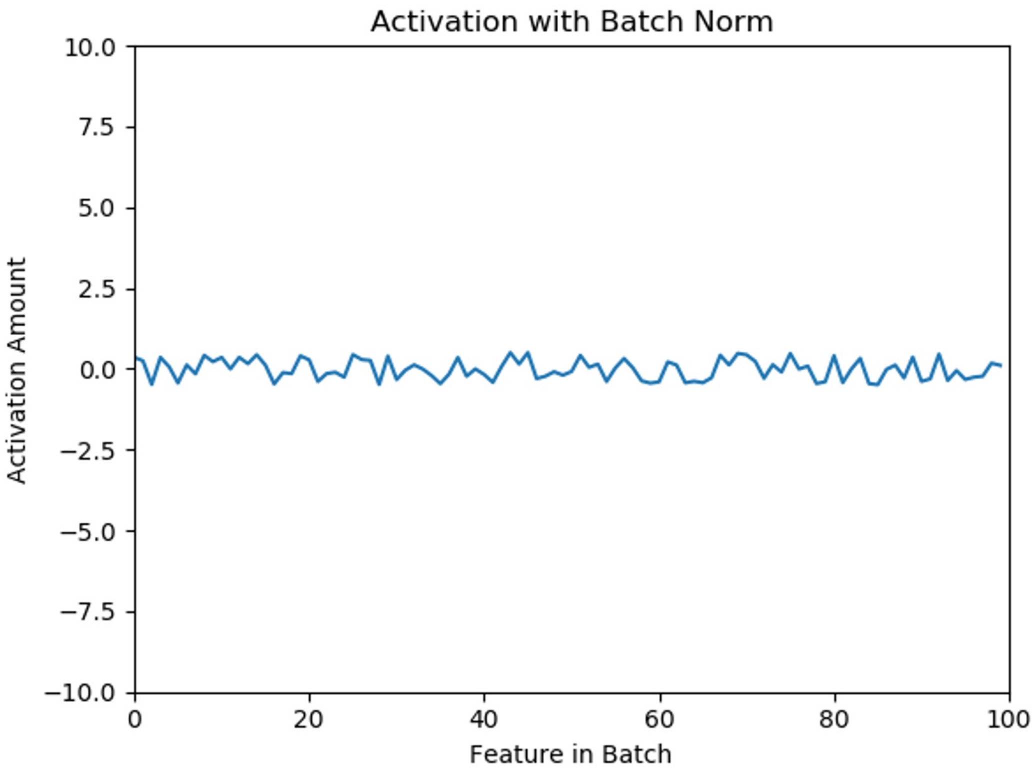


https://blog.perceptilabs.com/using-resnets-to-detect-anomalies-in-industrial-iot-textile-production/

Batch Normalization (stabilizing training)

Idea: normalize the activations for each feature at each layer



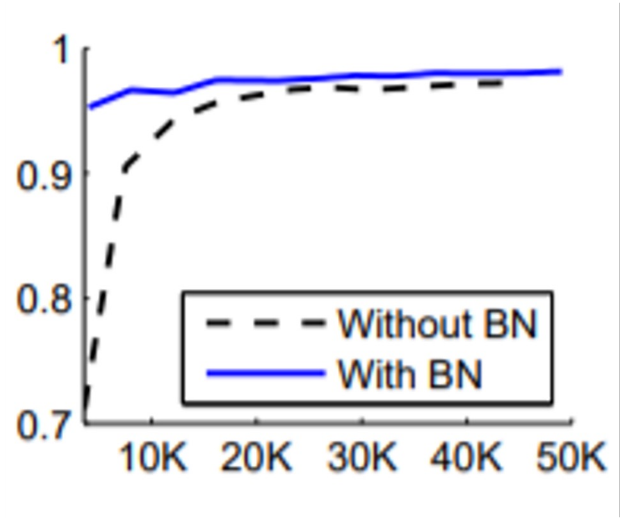


Why might we want to do this?

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:

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

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

Batch Normalization: Implementation

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

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

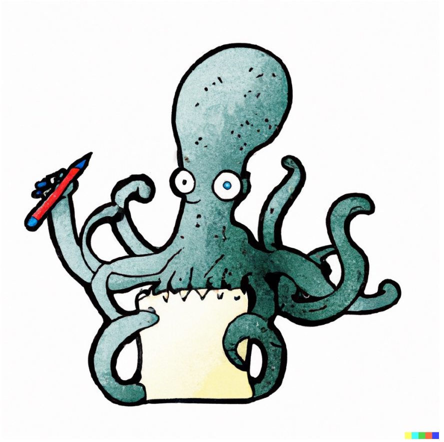
Feature x now has mean 0 and variance 1 along the batch

Batch Normalization in Tensorflow

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

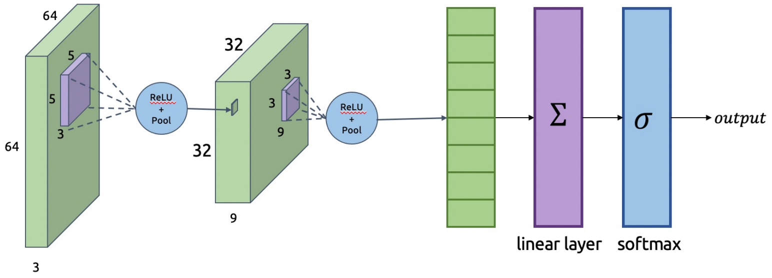
Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/BatchNormalization

Recap



CNNs

- Architecture
- AlexNet + Pooling
- CNNs are “sort of” translationally invariant



Deeper CNNs

- Many layers = vanishing gradient
- ResNet + Residual blocks
- Batch normalization