CSCI 1470

Eric Ewing

Friday, 2/21/25

Deep Learning

Day 13: Convolutions, Invariance, and Regularization

“Deep” Space News from NASA

NASA just changed the odds of asteroid YR4 hitting Earth in 2032 yet again

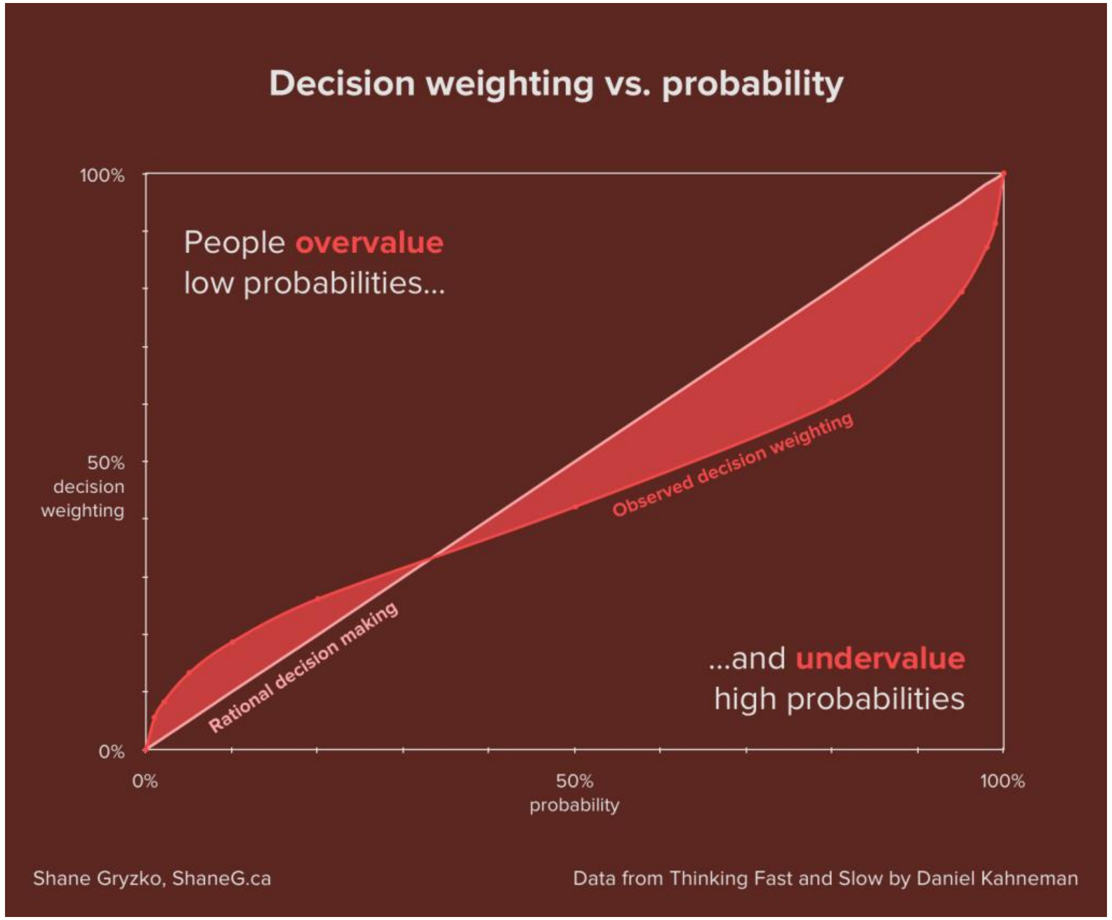
News By Patrick Pester published yesterday

NASA increased the chances of asteroid 2024 YR4 hitting Earth to 1 in 32, or 3.1%, on Tuesday, but they're now back down to 1 in 67, or 1.5%.

Humans and Probabilities

Humans tend to overvalue low probability events and undervalue high probability events

- Human decision making is often modeled as "boundedly rational"
    - We make "close" to the right decision
- One way of modelling this: Quantal Response
    - Humans make decisions with probabilities given by a softmax function



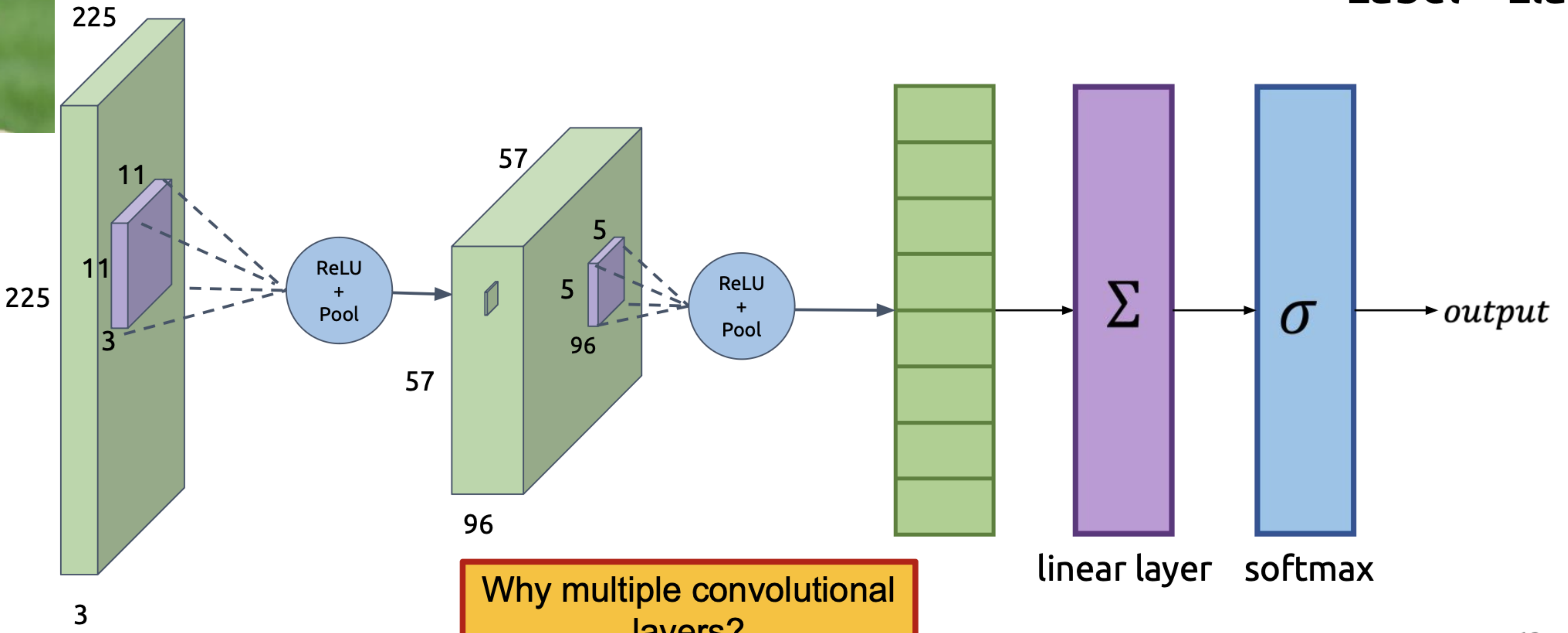
Today's Goals

(1) Finish talking about CNN architectures

(2) How can we train deeper Neural Networks?

CNN Architecture

Input



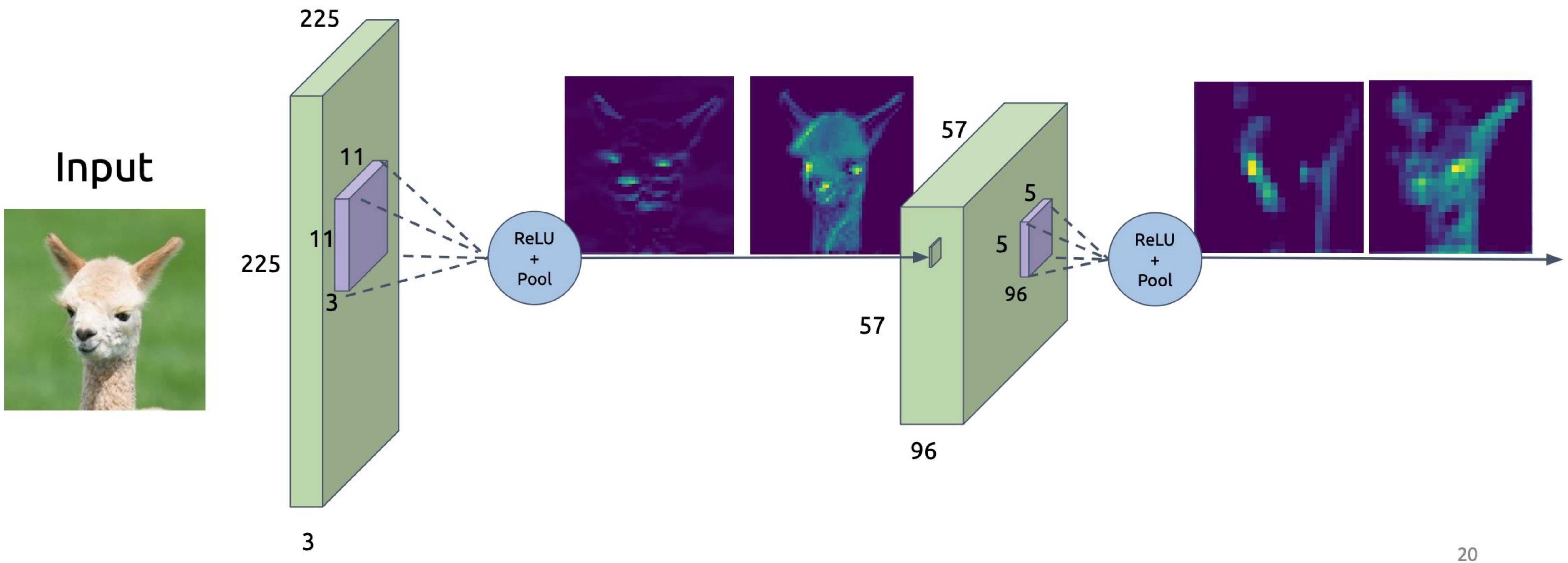
Label="Llama"

Why multiple convolutional layers?

Feature Extraction using multiple convolution layers

Hierarchy of features

Sequence of layers detect broader and broader features



Any questions?

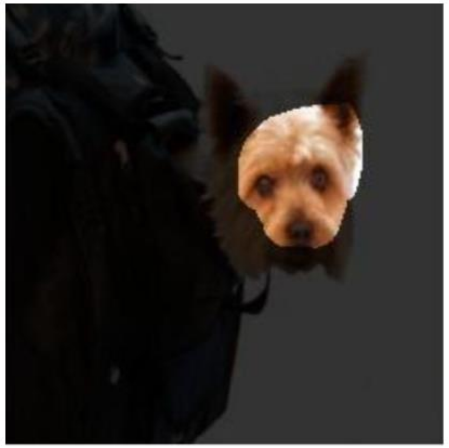
Example: Network Dissection

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

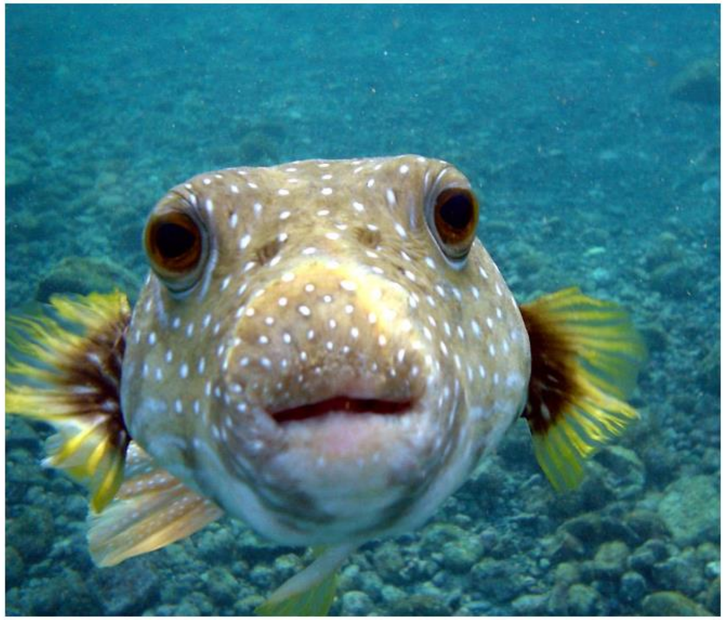
https://commons.wikimedia.org/wiki/File:Common_zebra_1.jpg

Predictions:

1. Carpet
2. 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

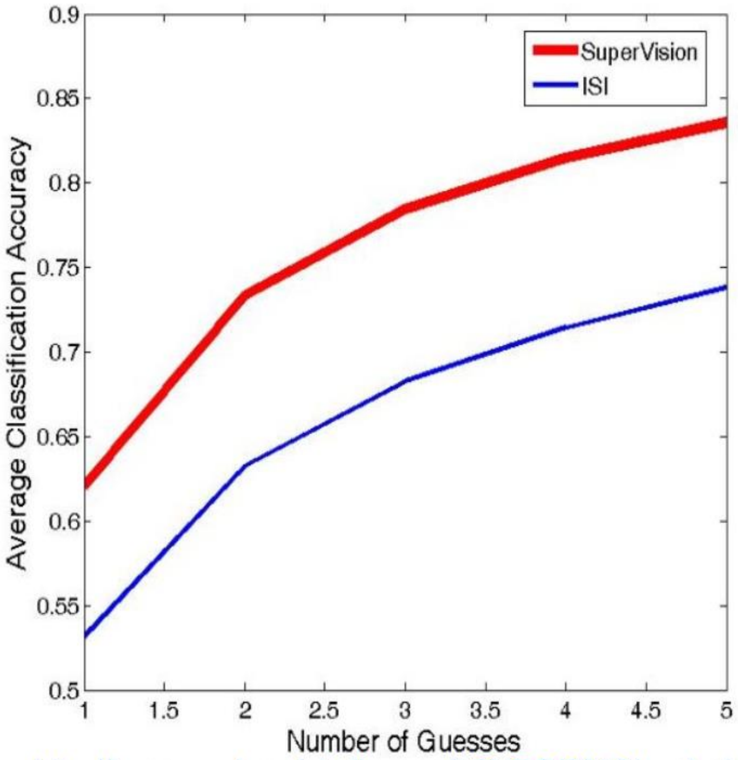
Predictions:

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

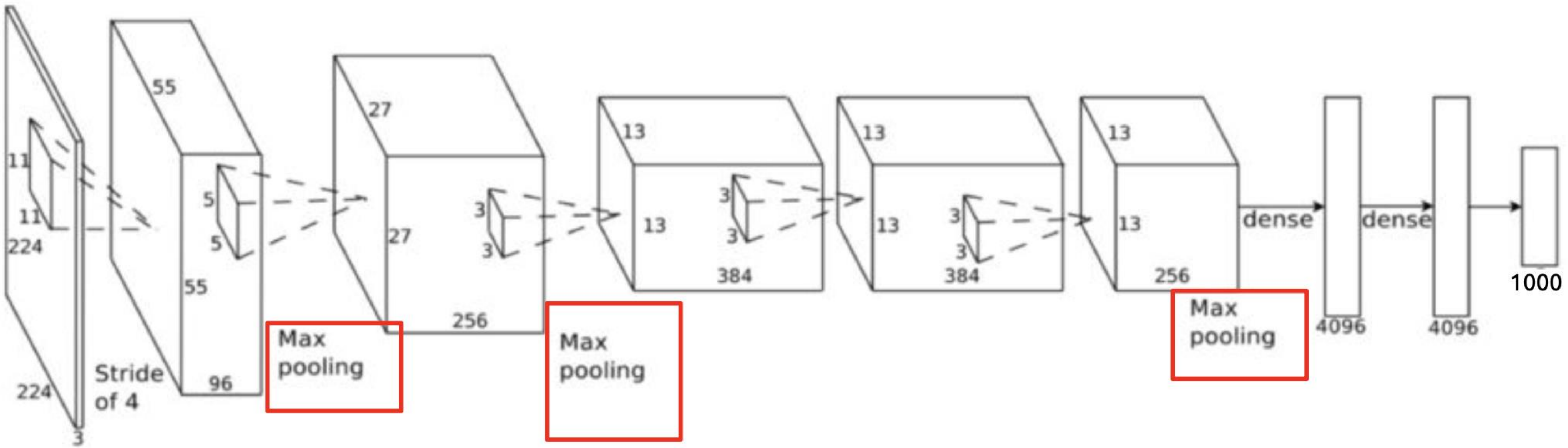


http://image-net.org/challenges/LSVRC/2012/analysis/

Note: SuperVision is the name of Alex's team

AlexNet

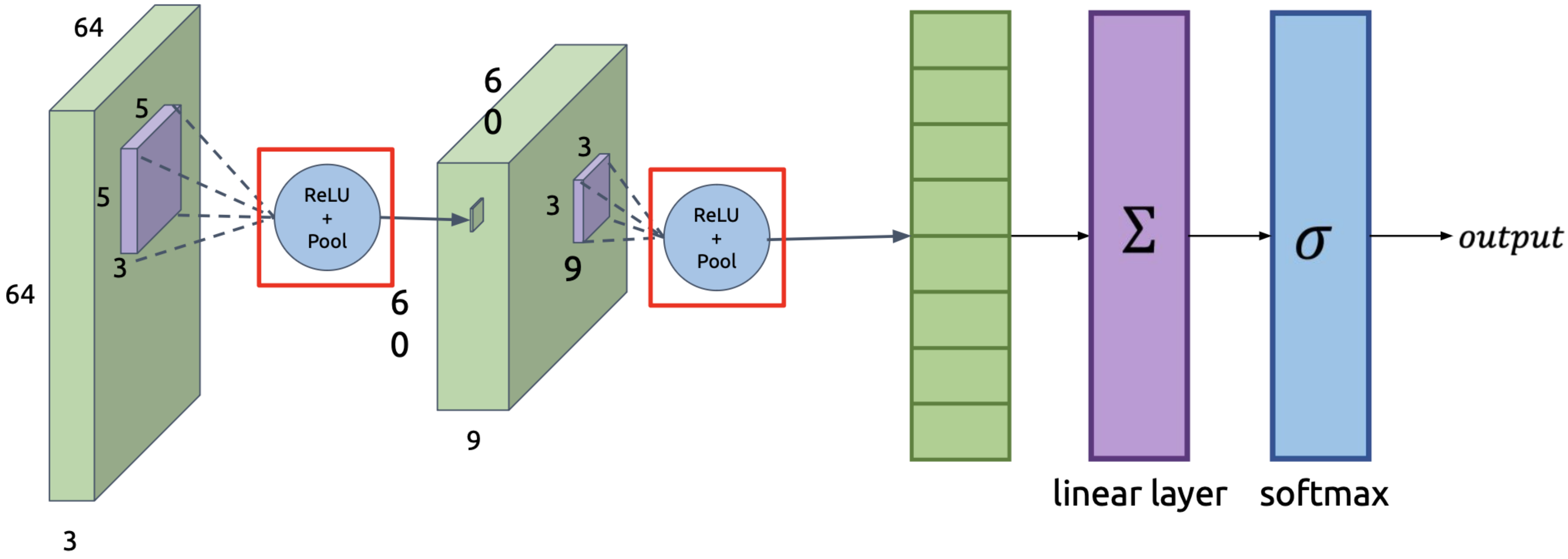
- 60 million parameters
- 5 Convolutional Layers
- 3 Fully Connected Layers



[Alex Krizhevsky et al. 2012]

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

Pooling

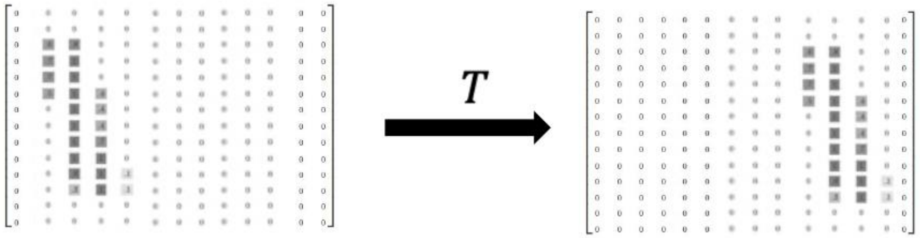


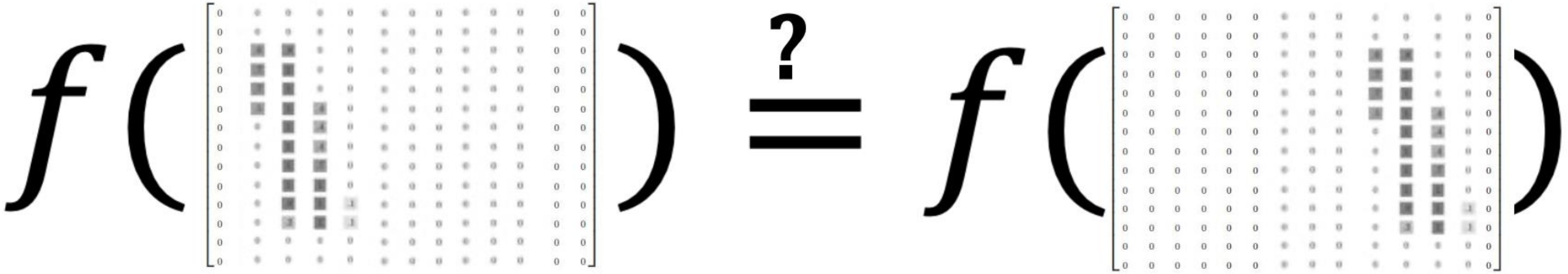
64
5
5
3
64
3
ReLU
+
Pool
6
0
6
0
3
3
9
9
ReLU
+
Pool
Σ
σ
output
linear layer
softmax

So…did we achieve our goal of translational invariance?

What was Translational Invariance again?

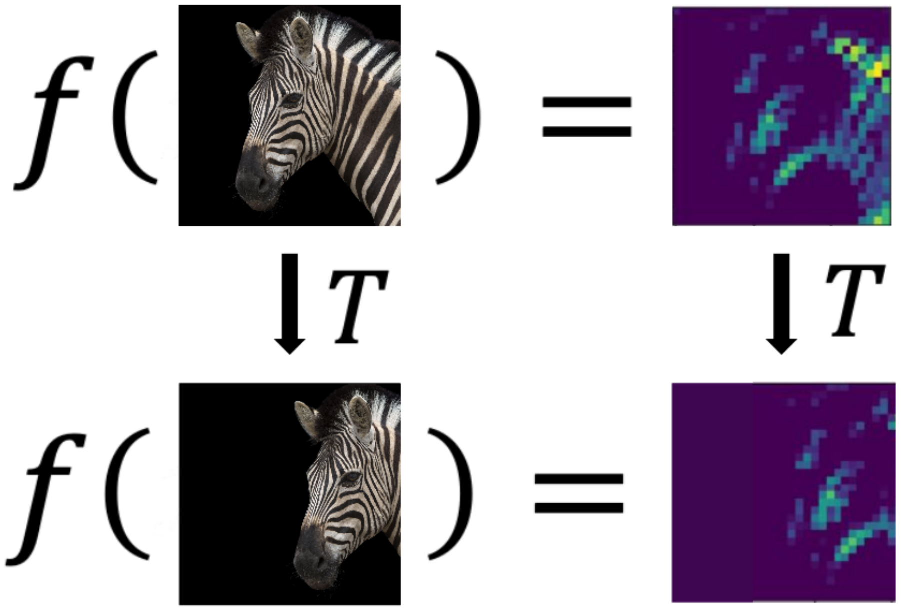
- To make a neural net f robust in this same way, it should ideally satisfy ***translational invariance***: $f\big(T(x)\big) = f(x)$, where
  - x is the input image
  - T is a translation (i.e. a horizonal and/or vertical shift)





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

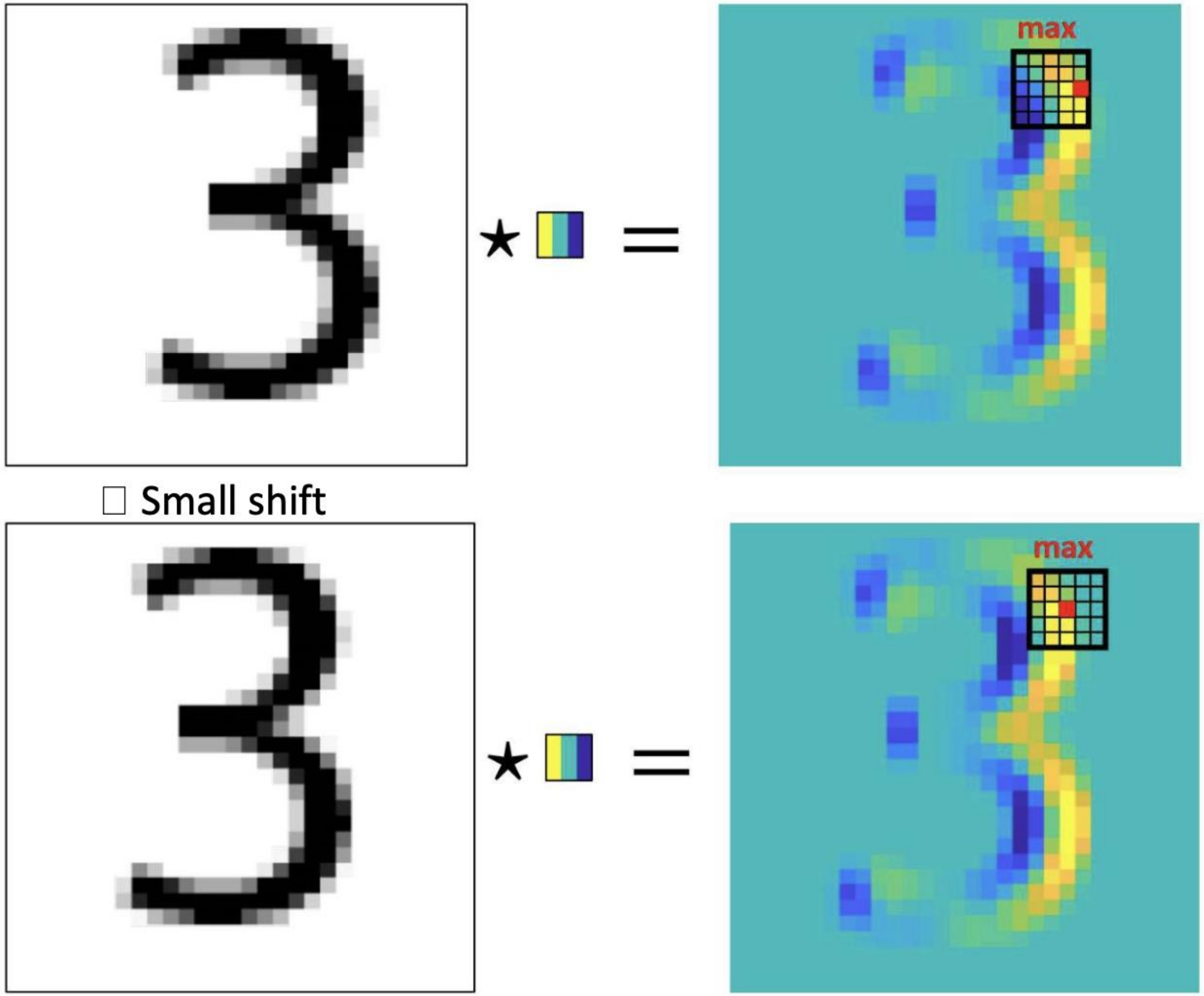
$$f\left(\begin{bmatrix} 6 & 3 \\ 4 & 1 \end{bmatrix}\right) = 6$$

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

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

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

So how does it all come together?



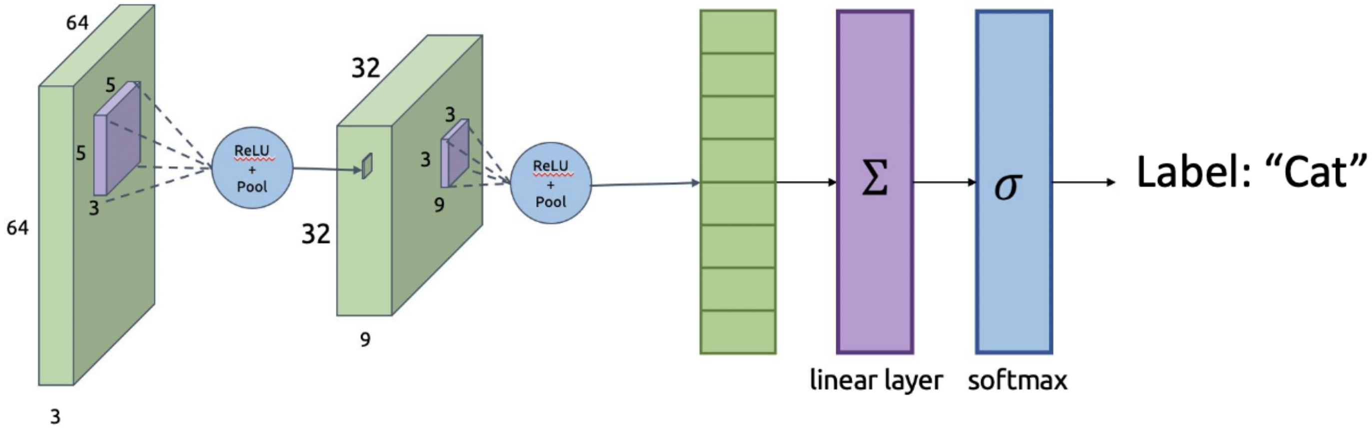
Convolution is ***translation equivariant***

Max pooling gives invariance to small translations

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

Are CNNs Translation Invariant?

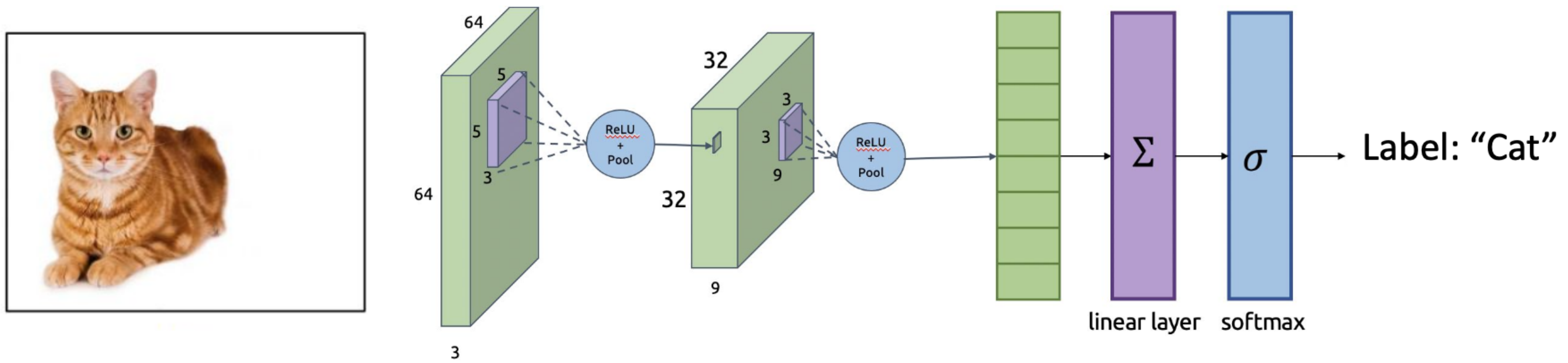
- Answer: CNNs are **“sort of” translation invariant**
  - Shifting the content of the image around tends not to drastically effect the output classification probabilities...



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?

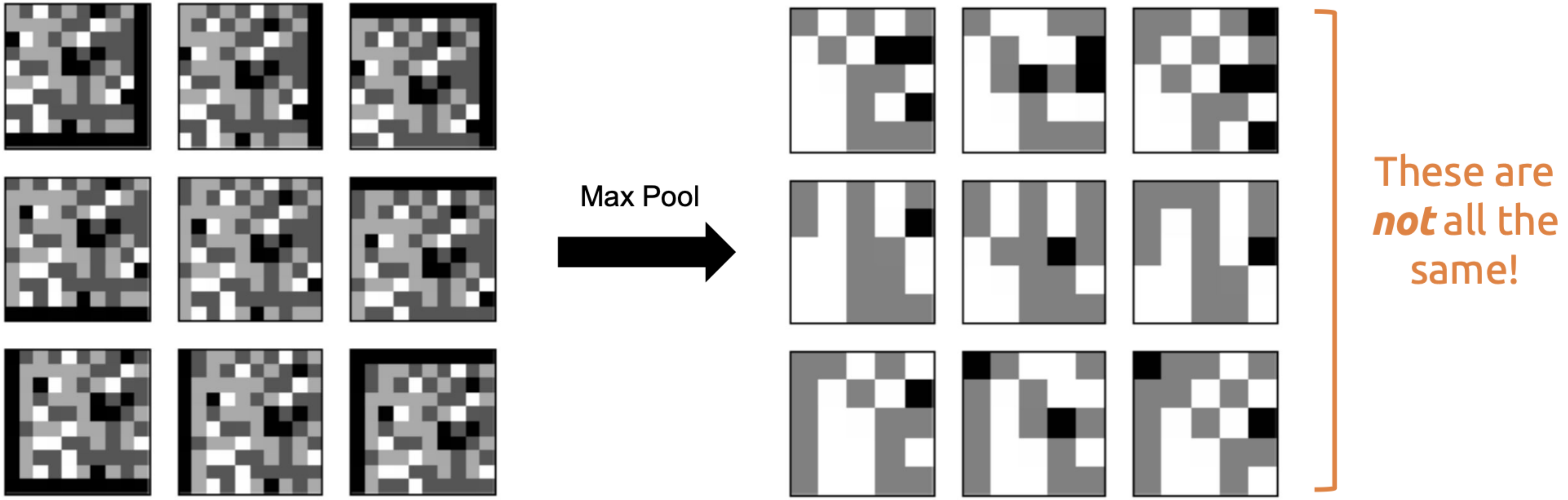
- Answer: CNNs are **"sort of" translation invariant**
  - Shifting the content of the image around tends not to drastically effect the output classification probabilities...



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

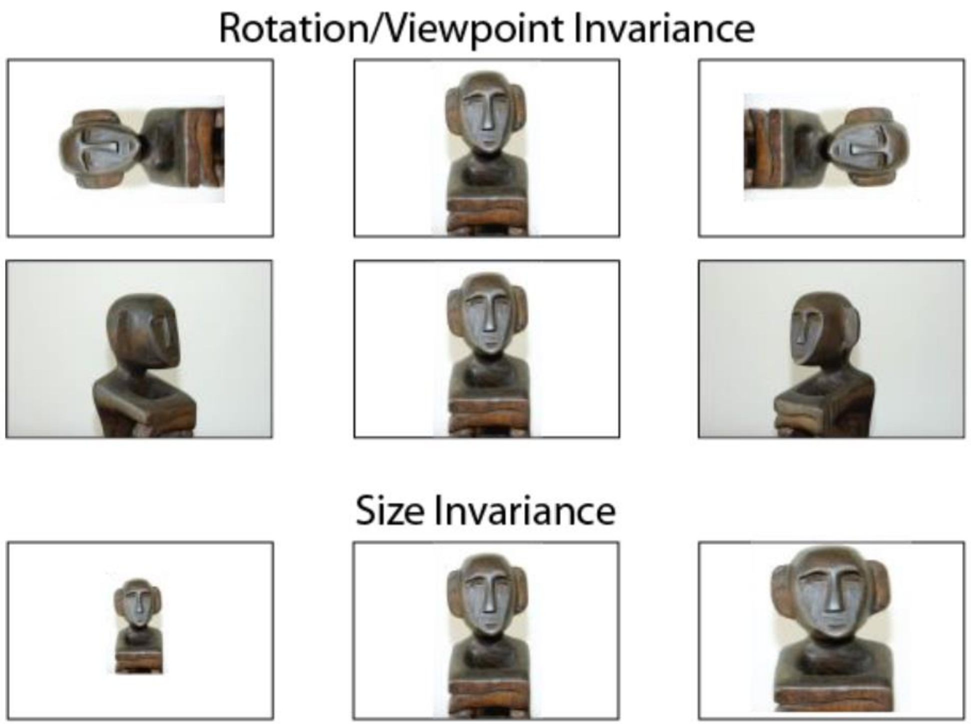


https://dsp.stackexchange.com/questions/24900/translation-invariance-in-max-pooling-and-cascading-with-convolutional-layer

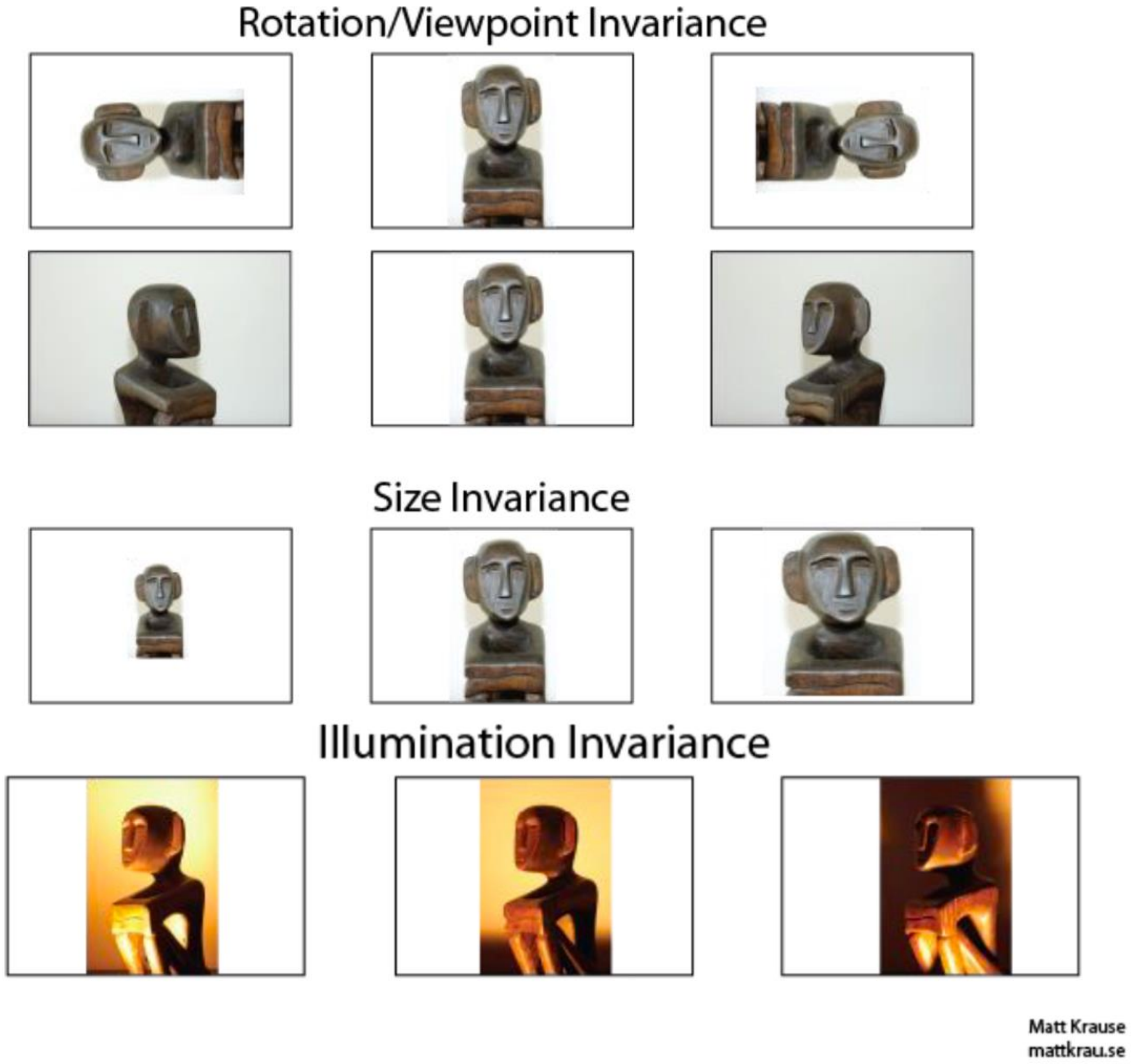
Other Invariances

Rotation/Viewpoint Invariance

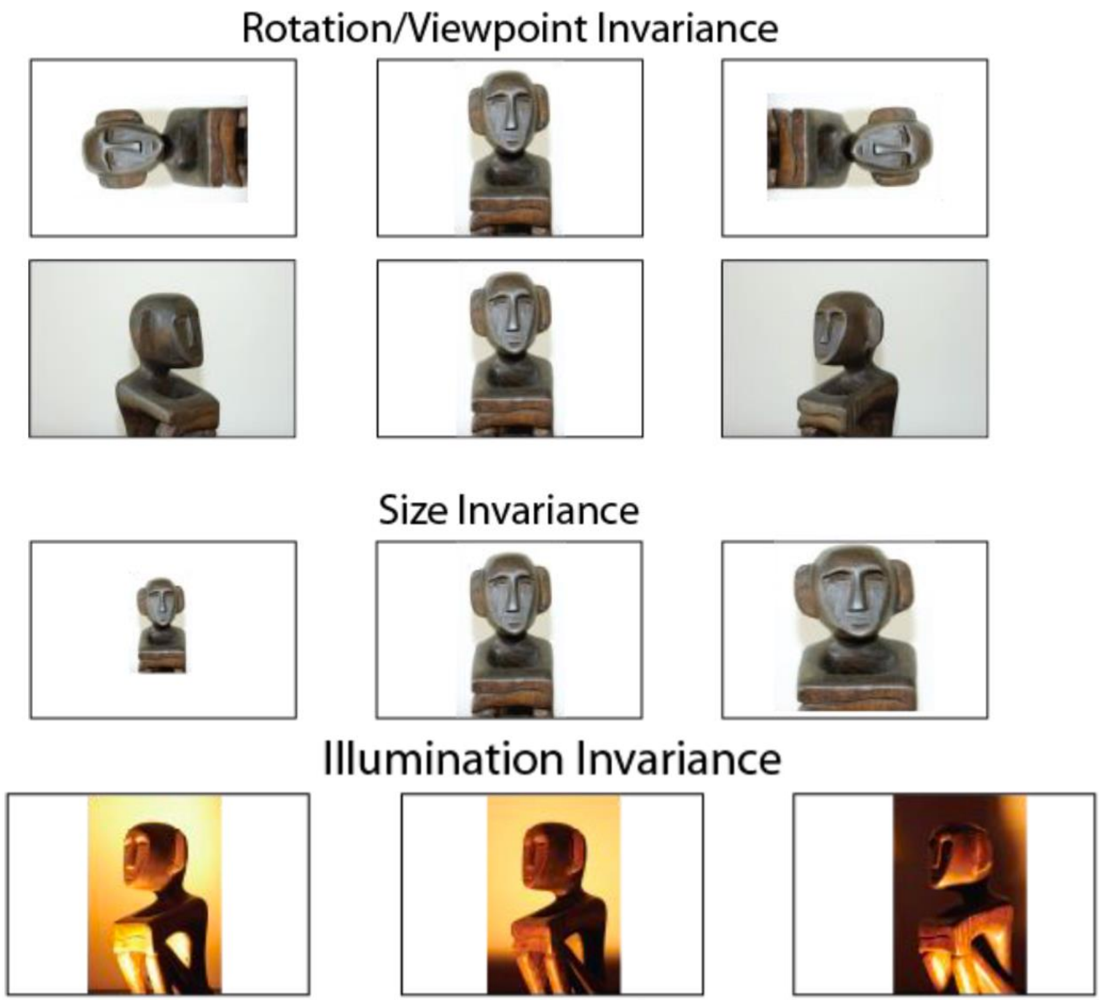
Other Invariances



Other Invariances



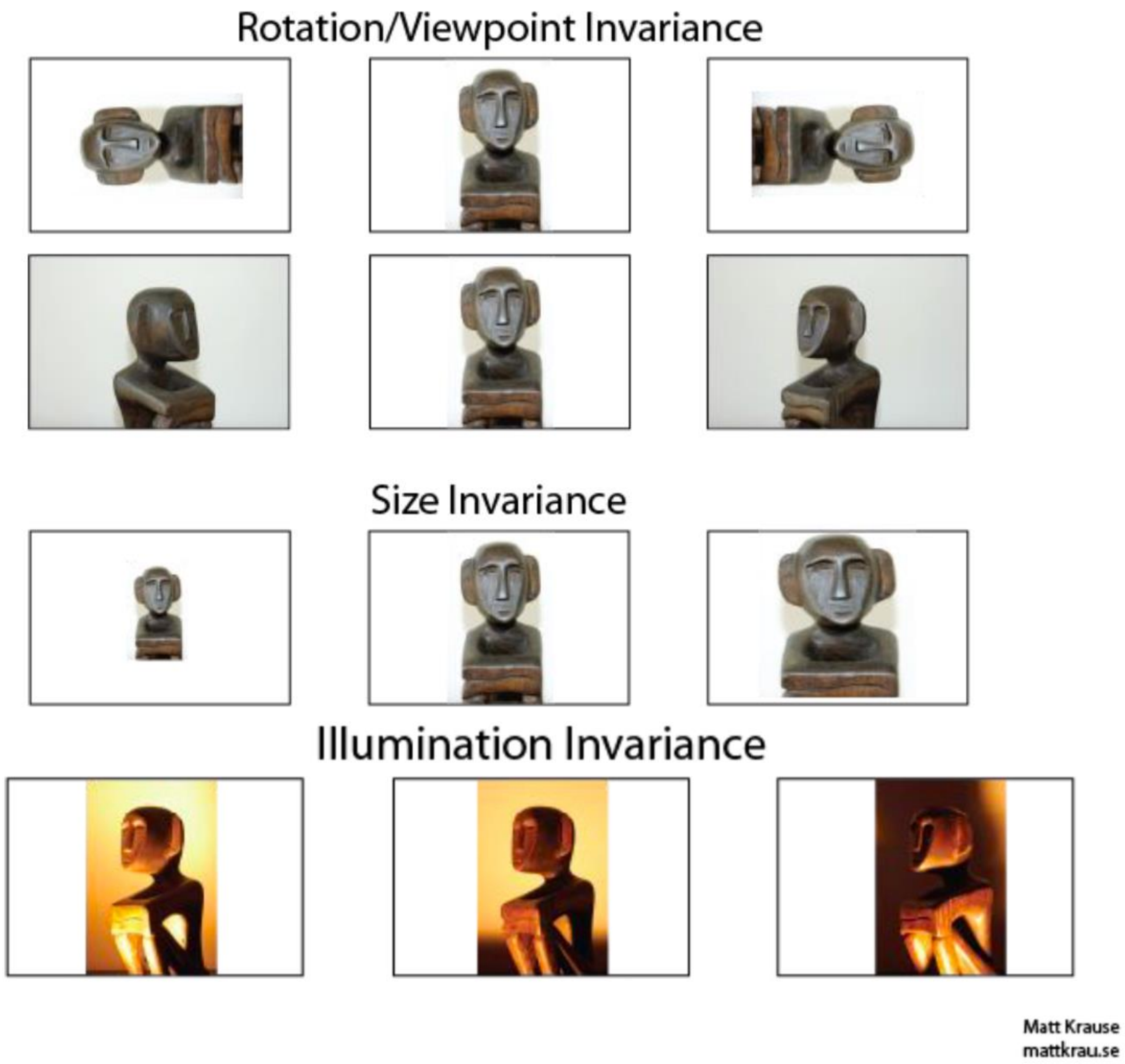
Other Invariances



Matt Krause
mattkrau.se

- 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.

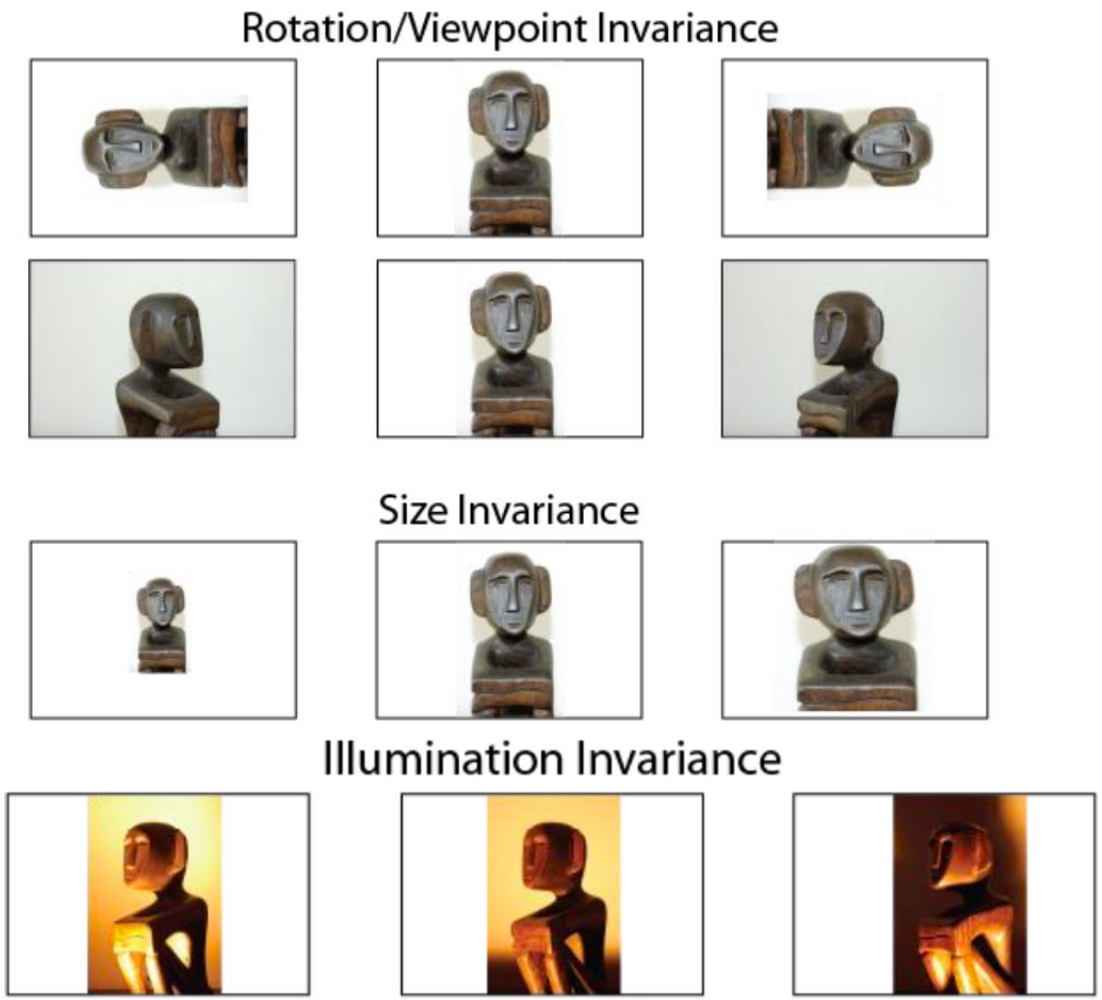
Other Invariances



- 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?

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?

Data Augmentation

If this is a cat in our dataset, it is an image with a label (cat)

Data Augmentation

If this is a cat in our dataset, it is an image with a label (cat)

This is also a cat

Data Augmentation

If this is a cat in our dataset, it is an image with a label (cat)

This is also a cat

This is also a cat

Data Augmentation

This is also a cat

If this is a cat in our dataset, it is an image with a label (cat)

This is also a cat

This is also a cat

Data Augmentation

This is also a cat

If this is a cat in our dataset, it is an image with a label (cat)

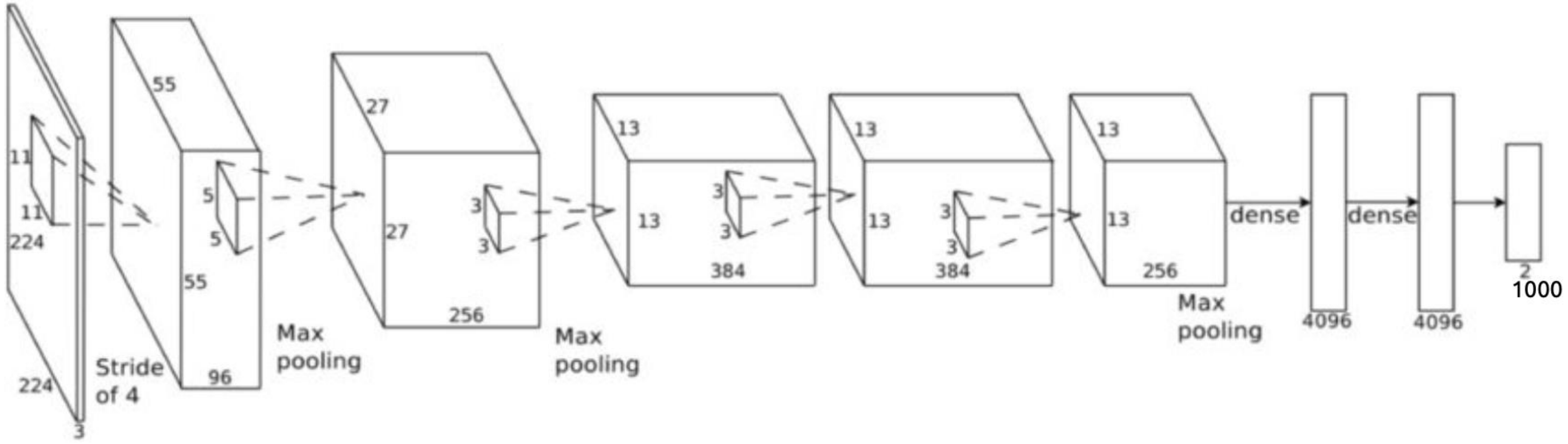
This is also a cat

This is also a cat

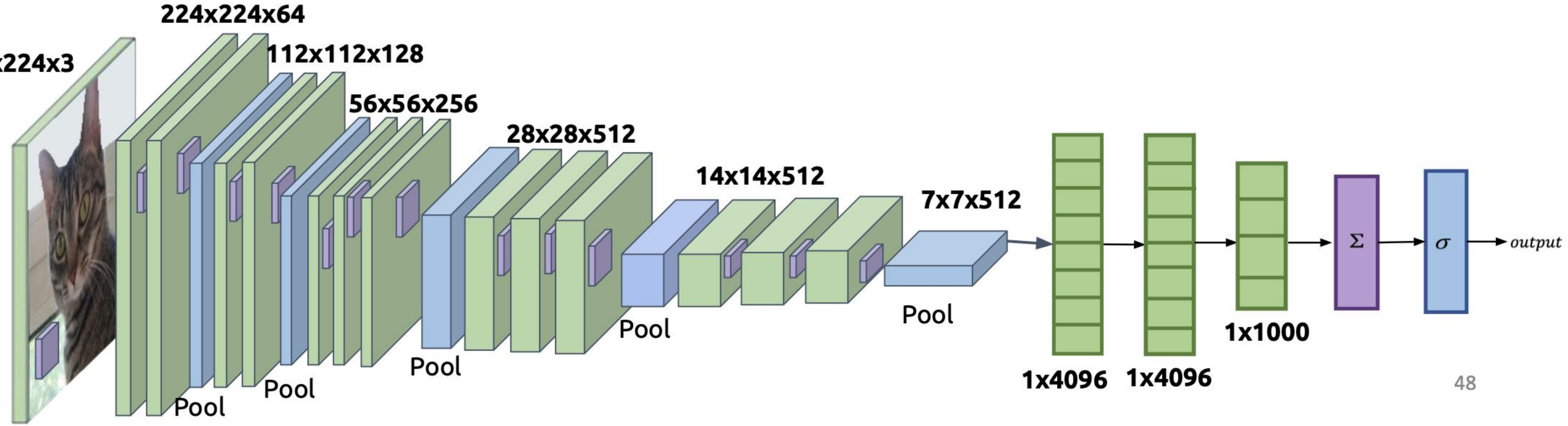
This is also a cat

More Complicated Networks

AlexNet:

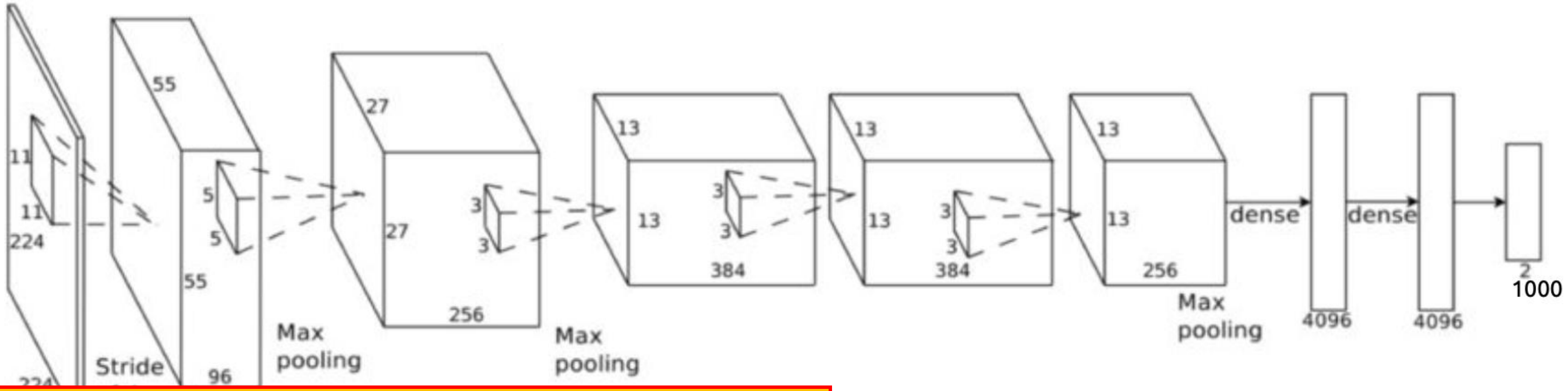


VGG:



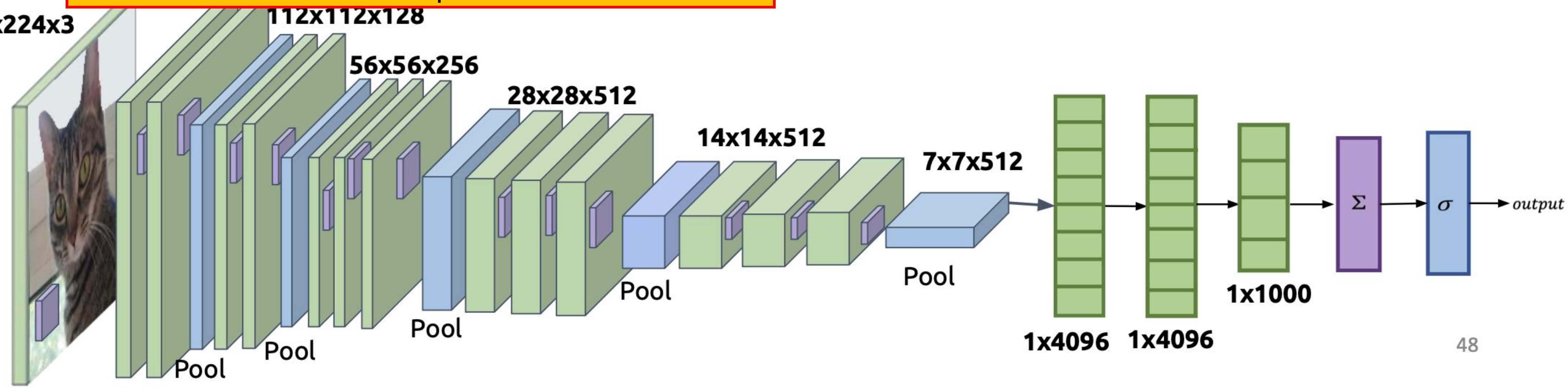
More Complicated Networks

AlexNet:



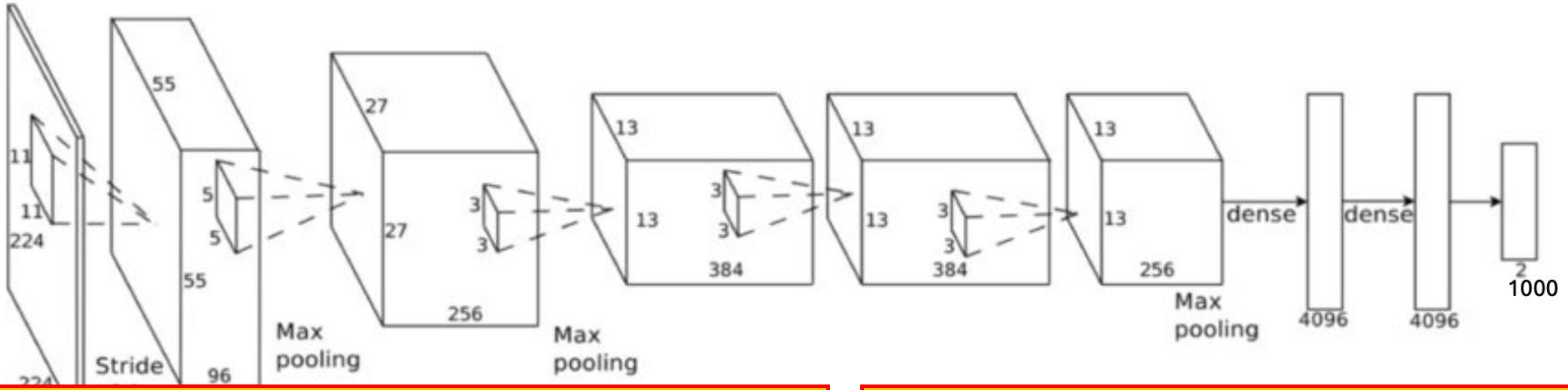
VGG uses 3x3 filters for everything, AlexNet uses 11, 5, 3, 3. Number of channels typically increases as depth increases

VGG:



More Complicated Networks

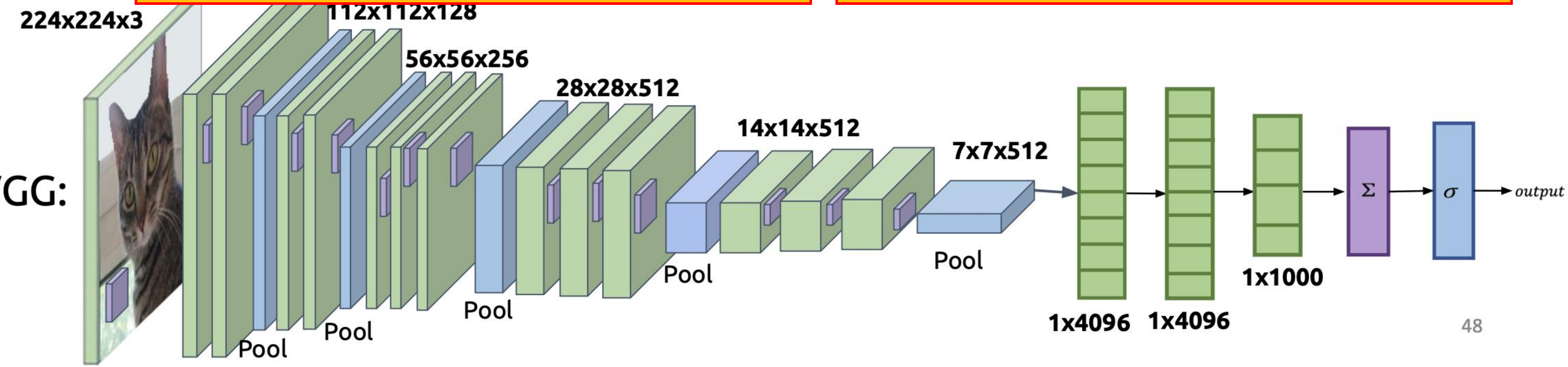
AlexNet:



VGG uses 3x3 filters for everything, AlexNet uses 11, 5, 3, 3. Number of channels typically increases as depth increases

Filters tend to get smaller as depth increases, number of output channels (num filters) increases (or stay the same)

VGG:



What if we didn't use a convolution?

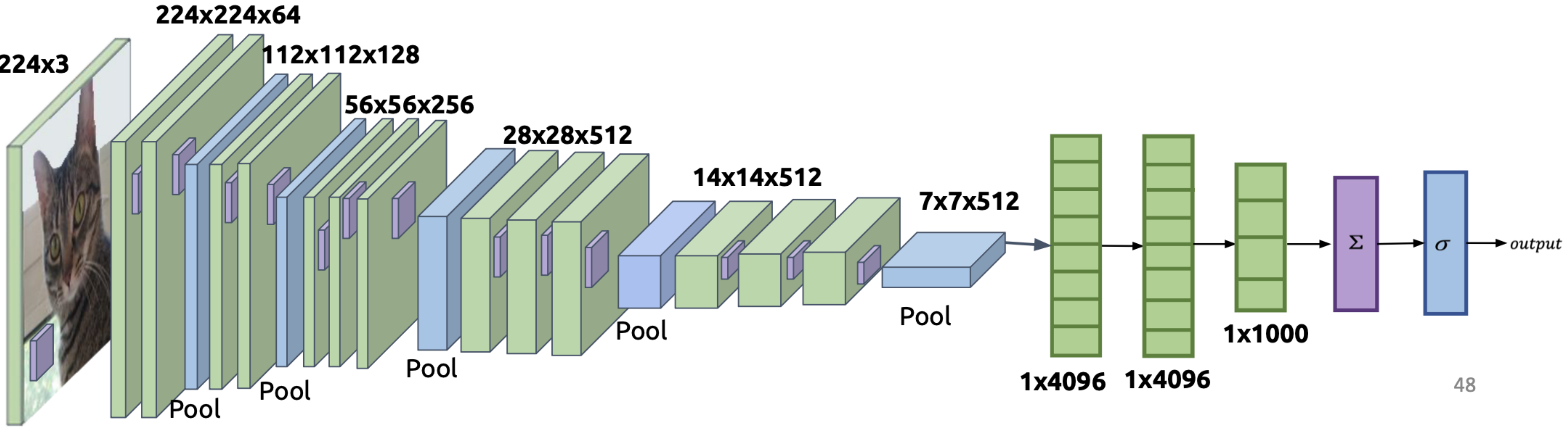
How many weights would there be if we have an input image of 224x224x3 and want to go to a hidden layer size of 4096?

What is the size of the Jacobian $\frac{\partial z}{\partial W}$?

With Convolutions

VGG uses 3x3 convolutions, how many weights are in the first filter bank to go from 224x224x3 to 224x224x64?

VGG:



Convolutions and Depth

Convolutions are much faster to run than a linear layer on the same size input

We can add more layers to CNNs than MLPs with the same inference time

Theory: Having more layers gives better performance with the same number of total weights (with lots of caveats)

Convolutions and Depth

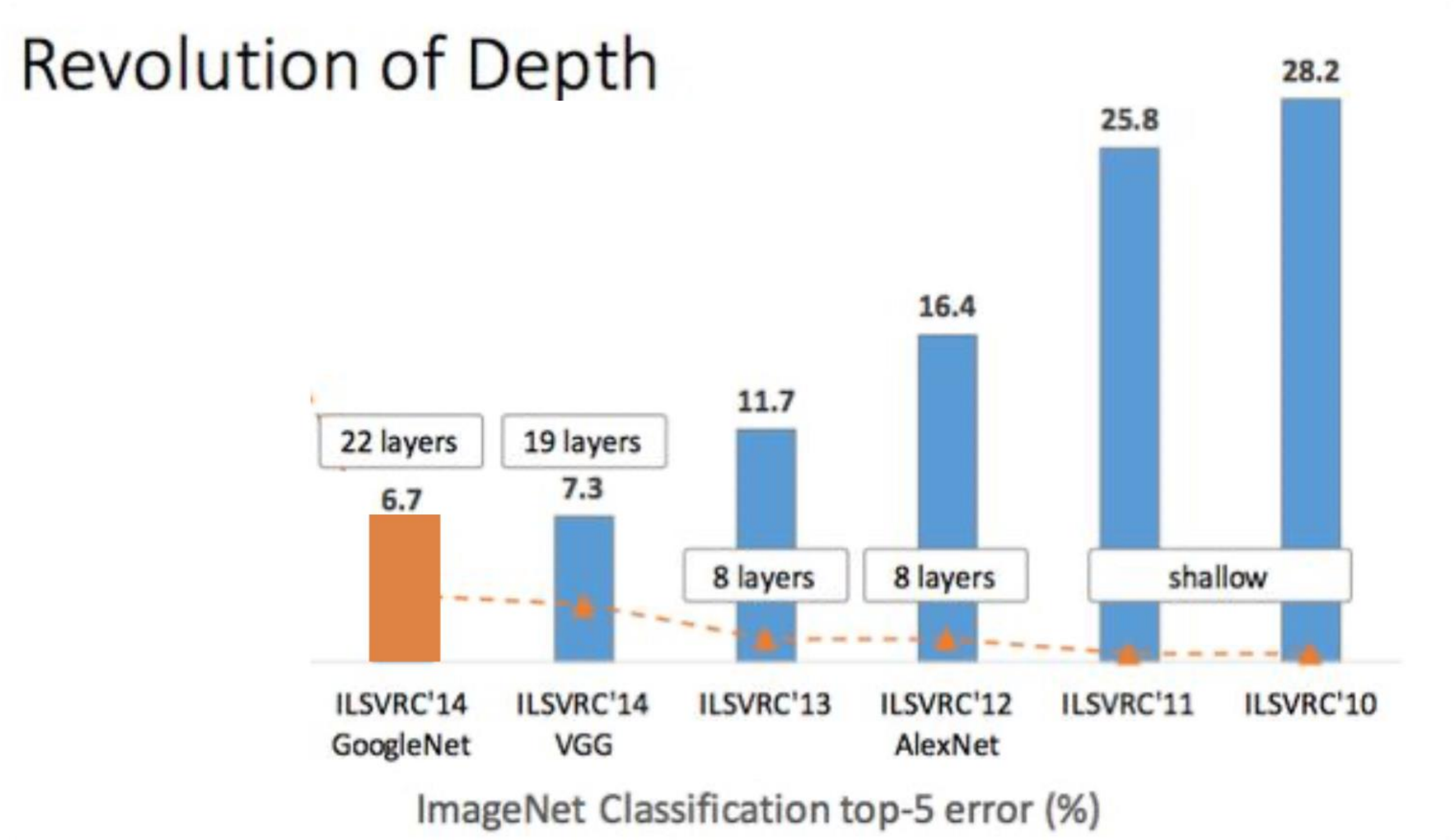
Convolutions are much faster to run than a linear layer on the same size input

We can add more layers to CNNs than MLPs with the same inference time

Theory: Having more layers gives better performance with the same number of total weights (with lots of caveats)

But we start to run into other issues as the depth of our neural networks increase…

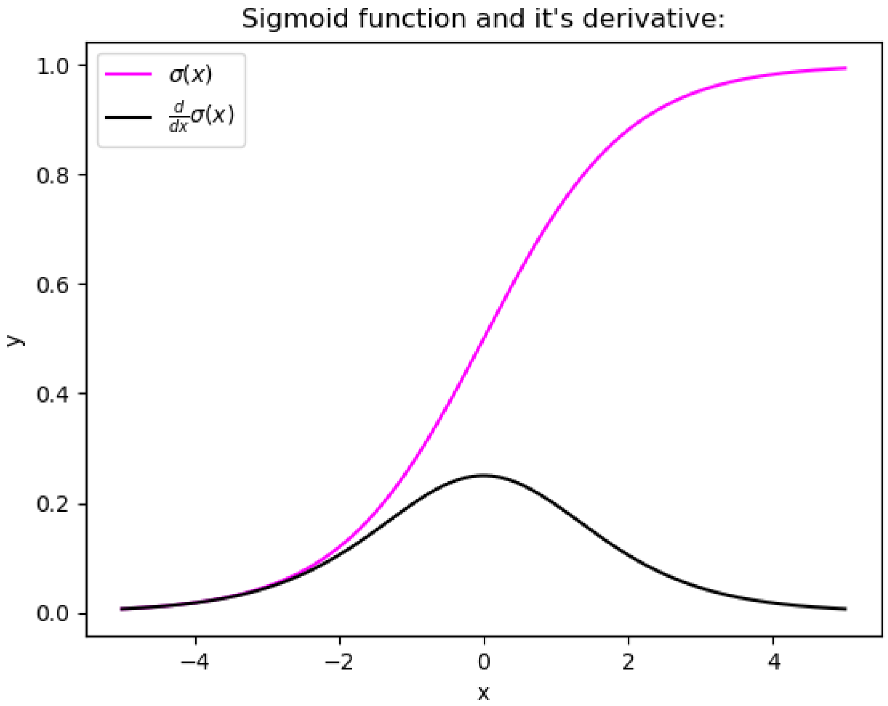
What's the biggest limitation in increasing depth?

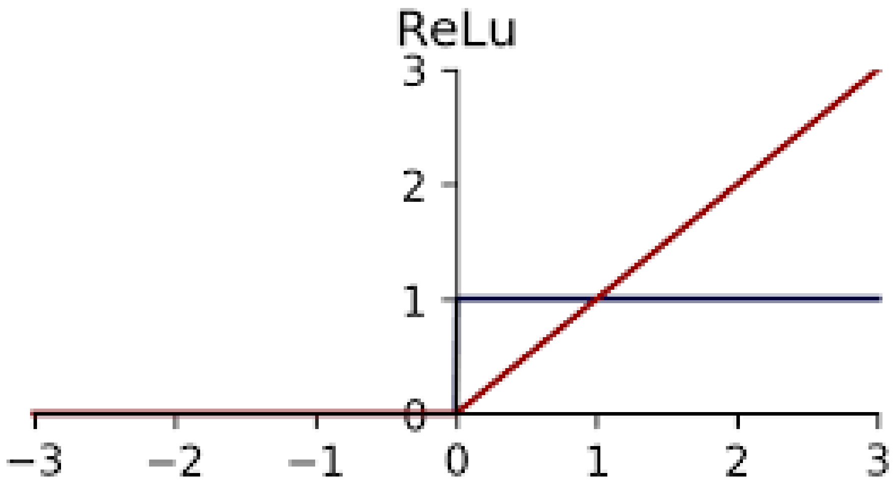


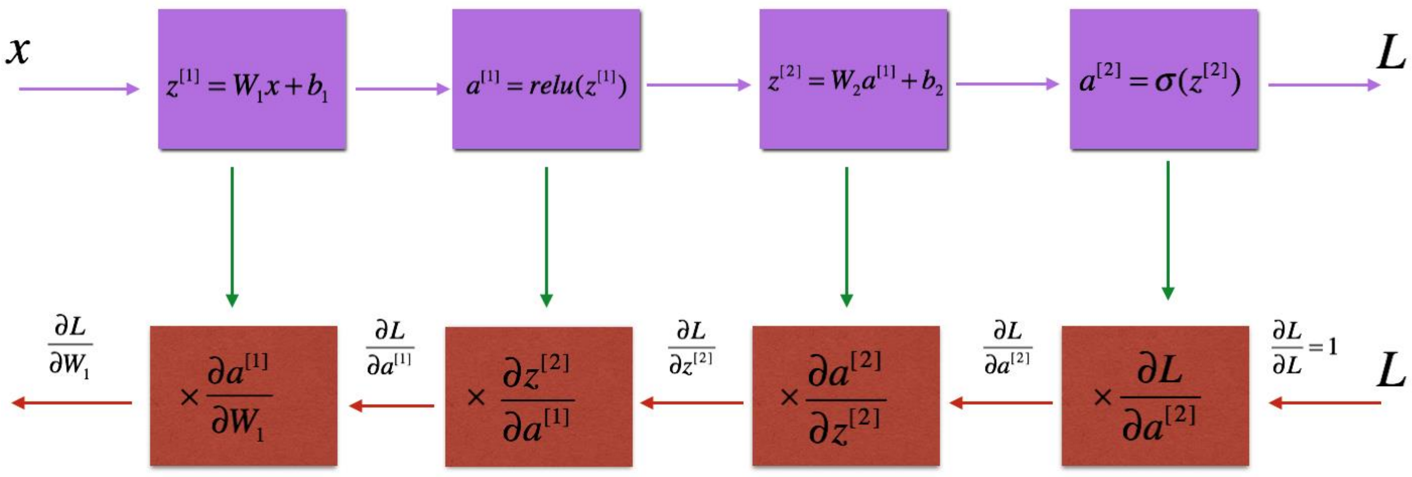
https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

The Return of Gradients

Common activation functions typically have a derivative smaller than 1 (or at least not more than 1)

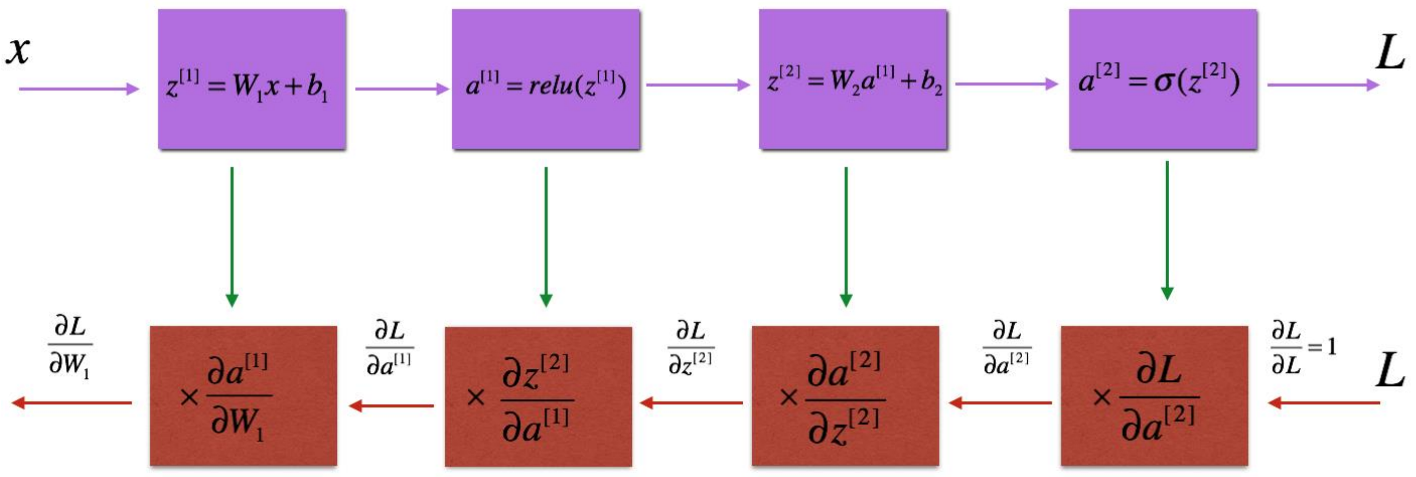






$$\boxed{\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} \frac{\partial a^{[1]}}{\partial W_1}}$$

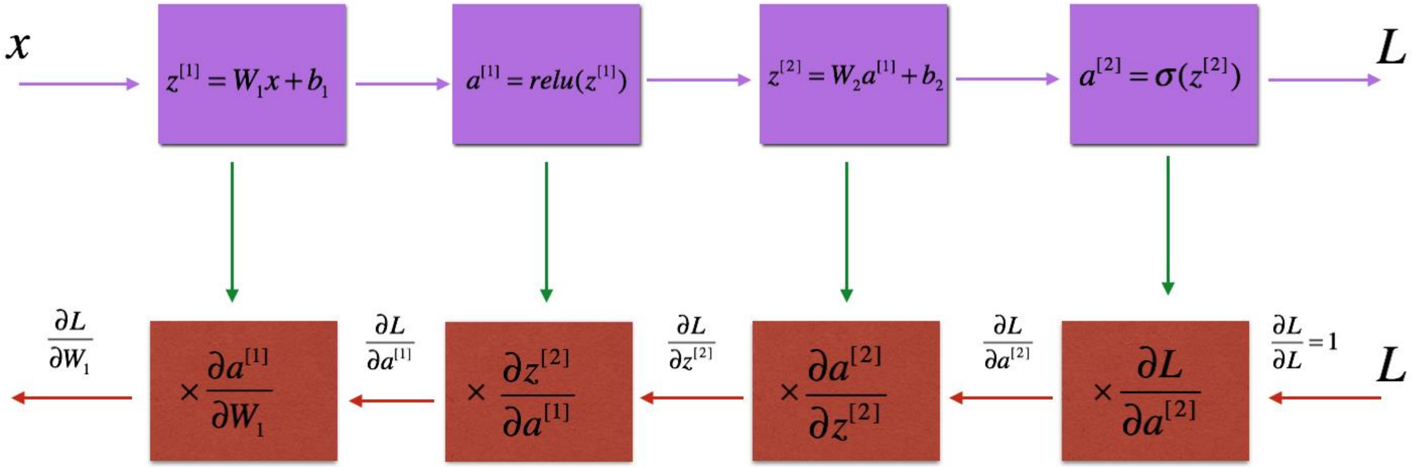
≤ 1



$$\boxed{\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} \frac{\partial a^{[1]}}{\partial W_1}}$$

≤ 1

Adding more layers adds more terms with gradient ≤ 1



$$\boxed{\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} \frac{\partial a^{[1]}}{\partial W_1}}$$

≤1

Multiplying by terms ≤1 makes things smaller...
Gradients earlier in the network tend to “Vanish”

Adding more layers adds more terms with gradient ≤1

Could we fix it by making everything "steeper"

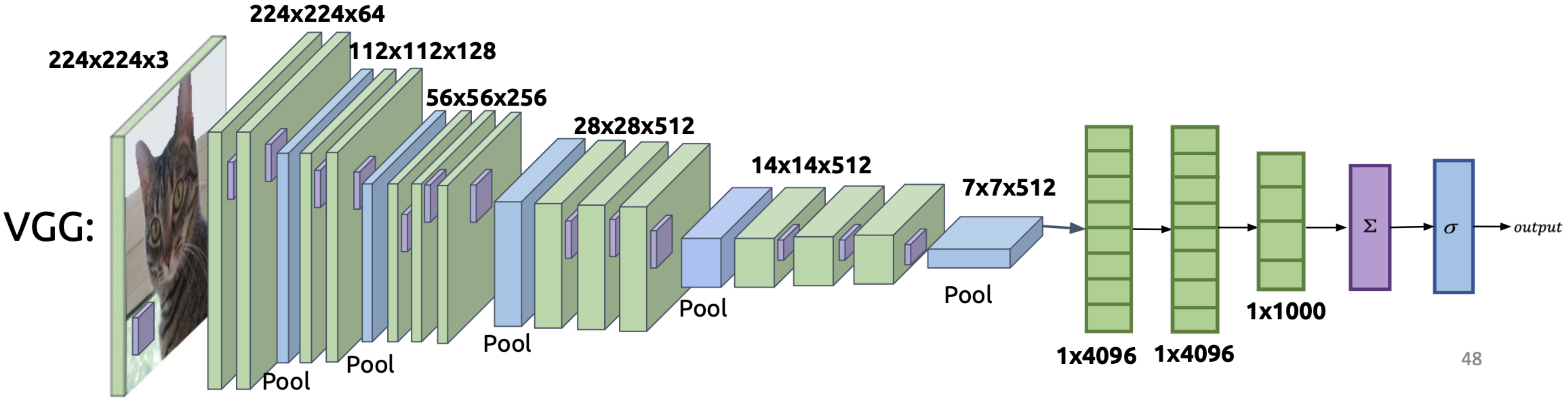
- Vanishing gradients are caused by the repeated multiplication of numbers smaller than 1
- If we make those numbers larger than 1, we have a separate problem…

Could we fix it by making everything "steeper"

- Vanishing gradients are caused by the repeated multiplication of numbers smaller than 1
- If we make those numbers larger than 1, we have a separate problem…

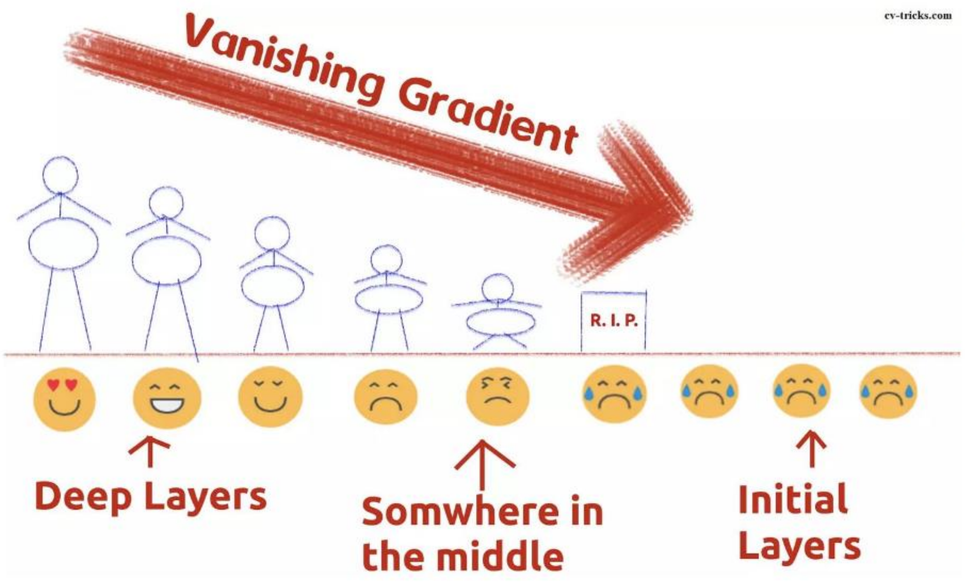
Exploding Gradients

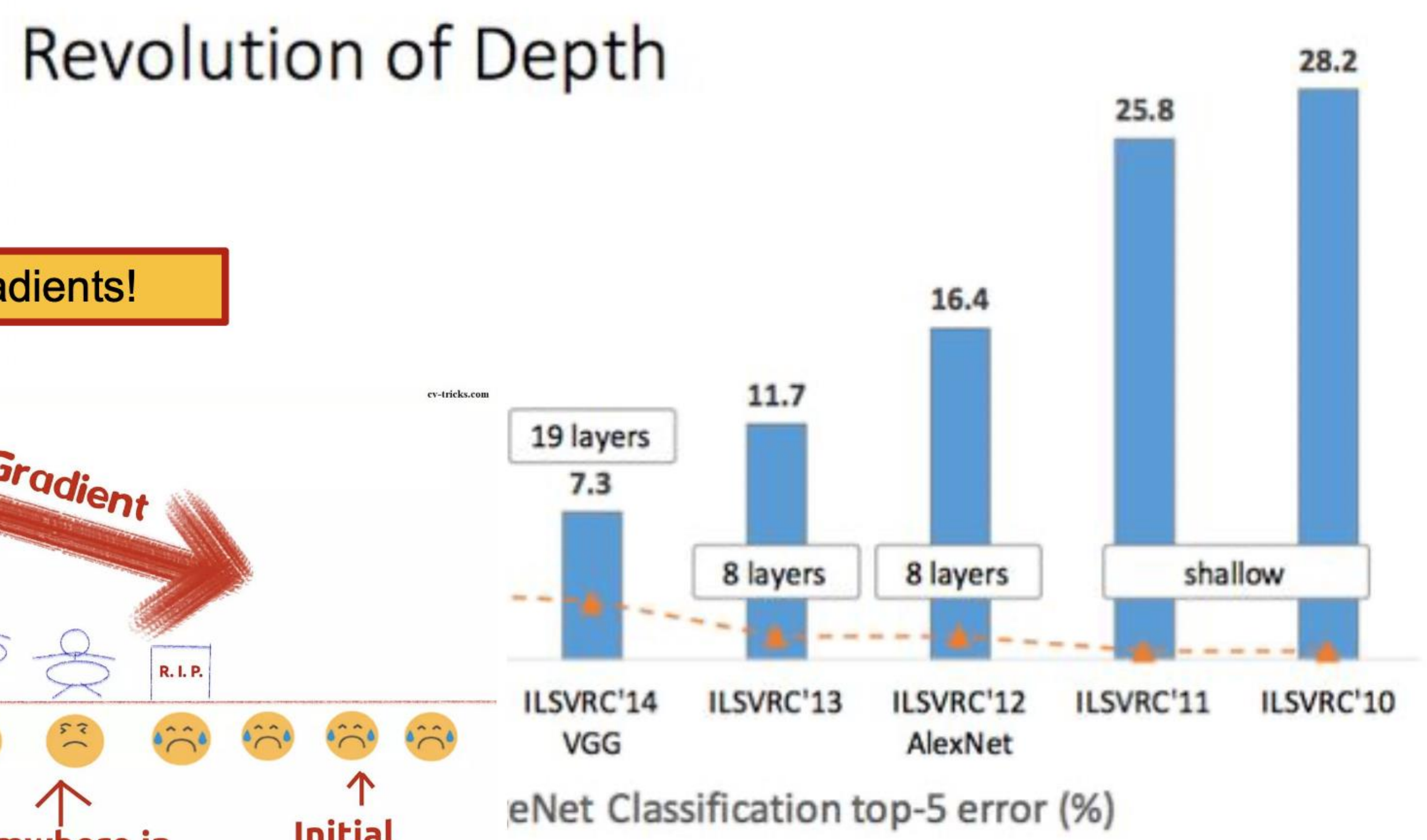
If you could make one change to a weight to have the biggest change on output, which weight would you pick?



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

Vanishing gradients!



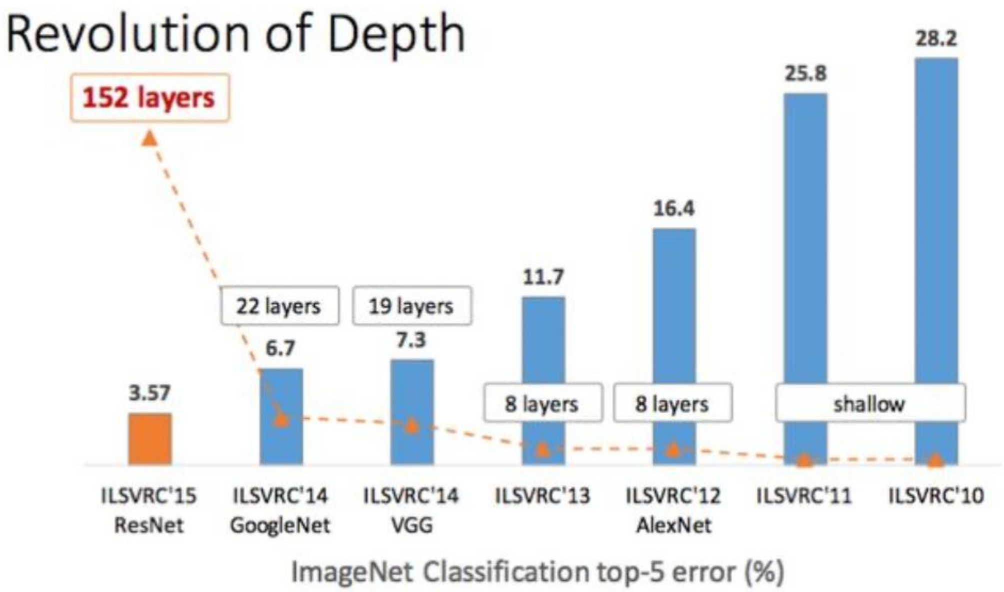


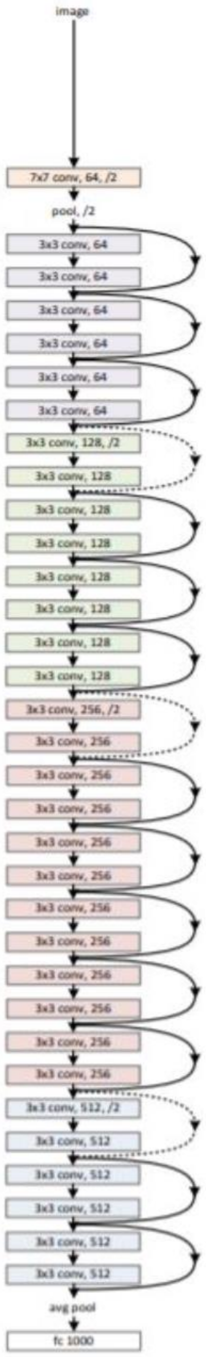
More Complicated Networks

ResNet:

Lots of layers, tons of learnable parameters

Avoids Vanishing Gradient problem
but how?



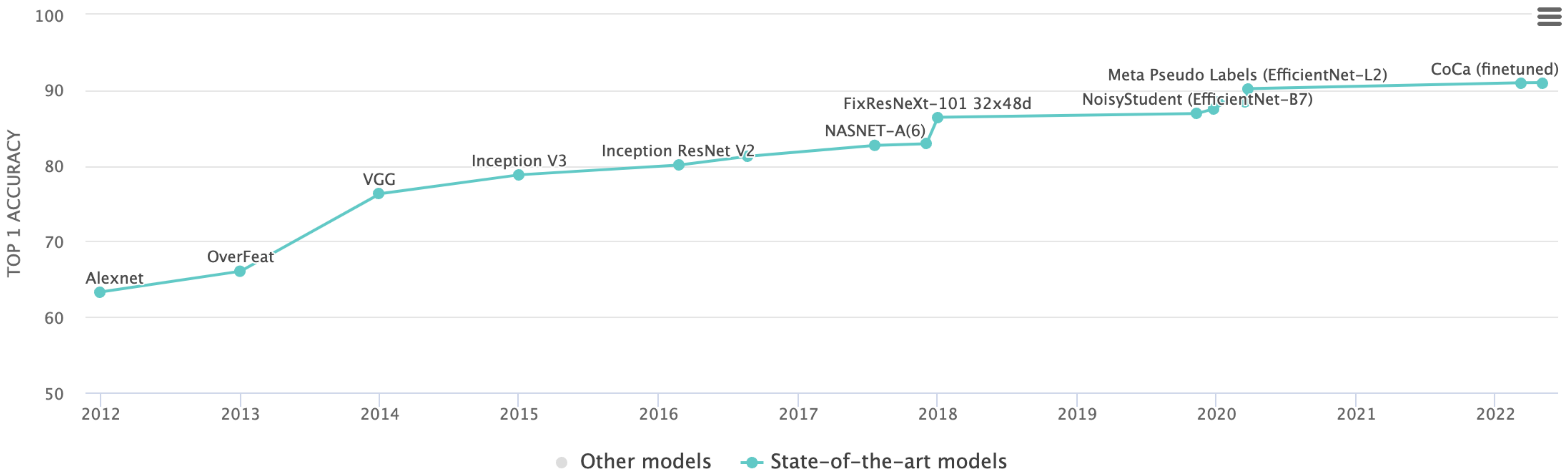


K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

Image Classification on ImageNet

Leaderboard Community Models Dataset

View Top 1 Accuracy by Date for All models



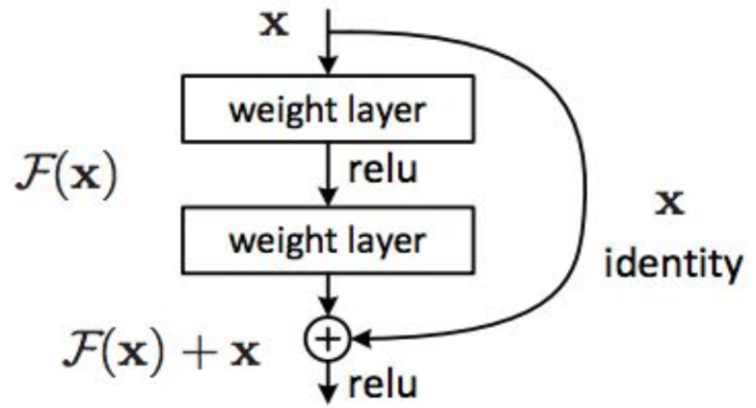
More Complicated Networks

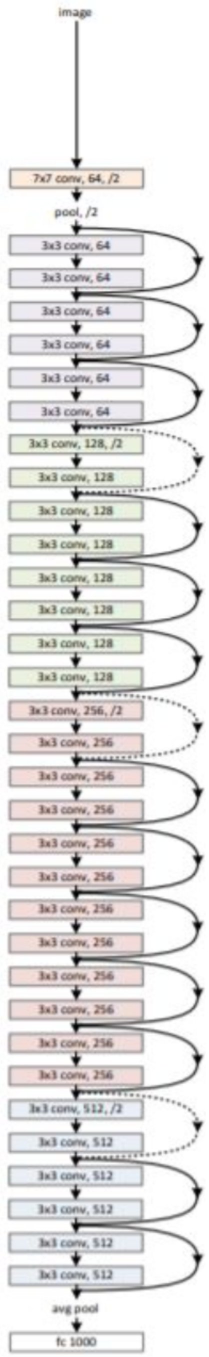
ResNet:

Lots of layers, tons of learnable parameters

Avoids Vanishing Gradient problem

Residual Block

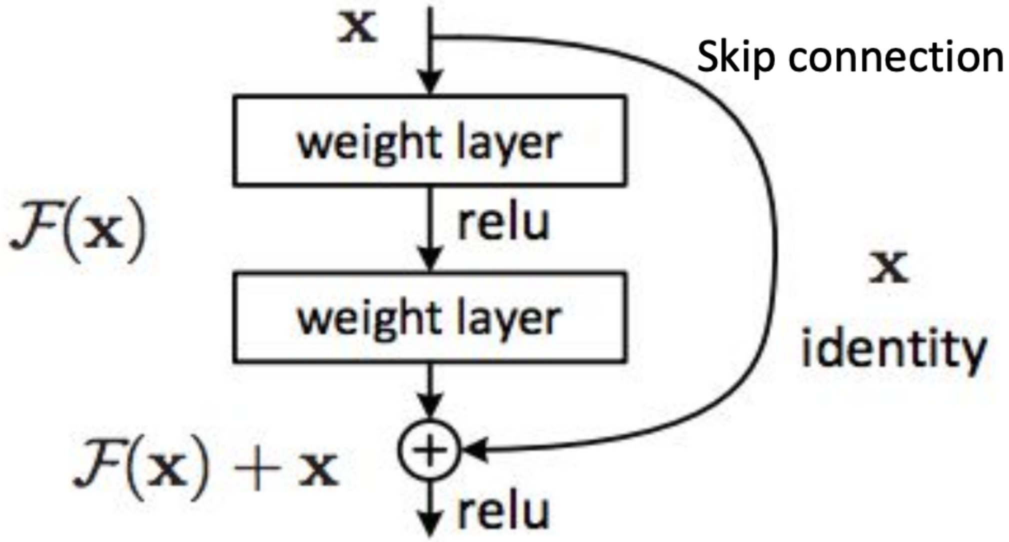




K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

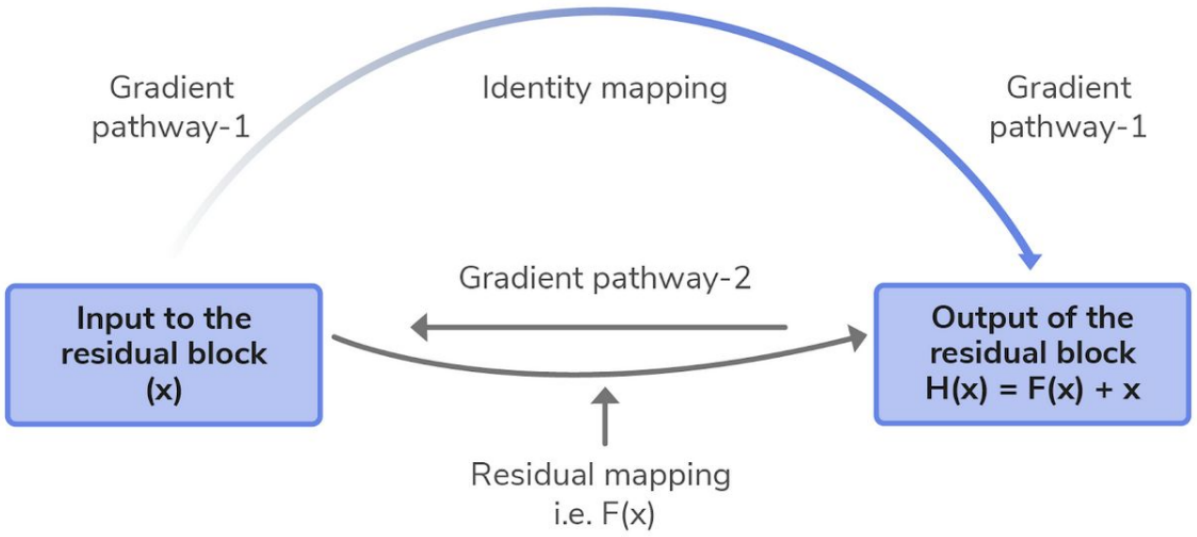
Residual Blocks

- 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

- 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 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**

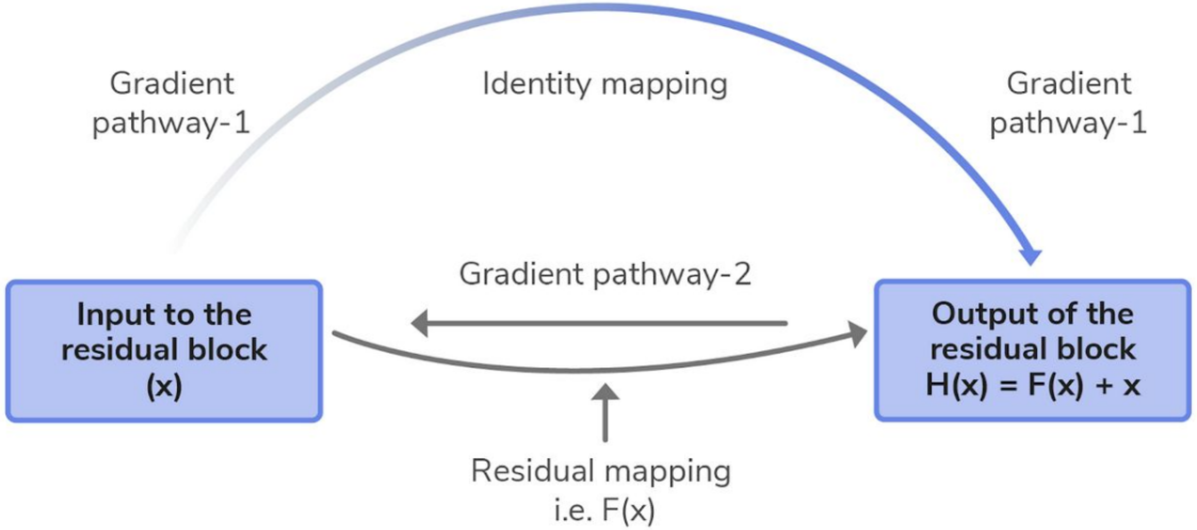


https://blog.perceptilabs.com/using-resnets-to-detect-anomalies-in-industrial-iot-textile-production/

Any questions?

Residual Blocks

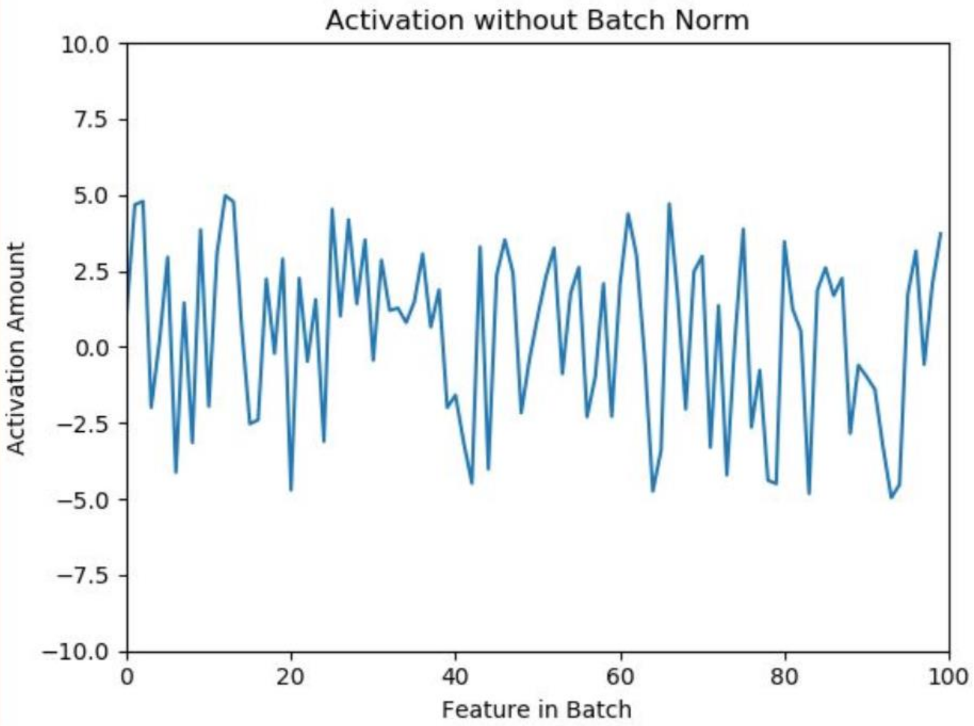
- In very deep nets, each layer often needs to learn just a small transformation of th preceding layer (identity + change)
- Idea: explicitly design the network such that the output of each layer is the identi + some deviation from it
  - Deviation is known as a residual
- Allows gradient to flow through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

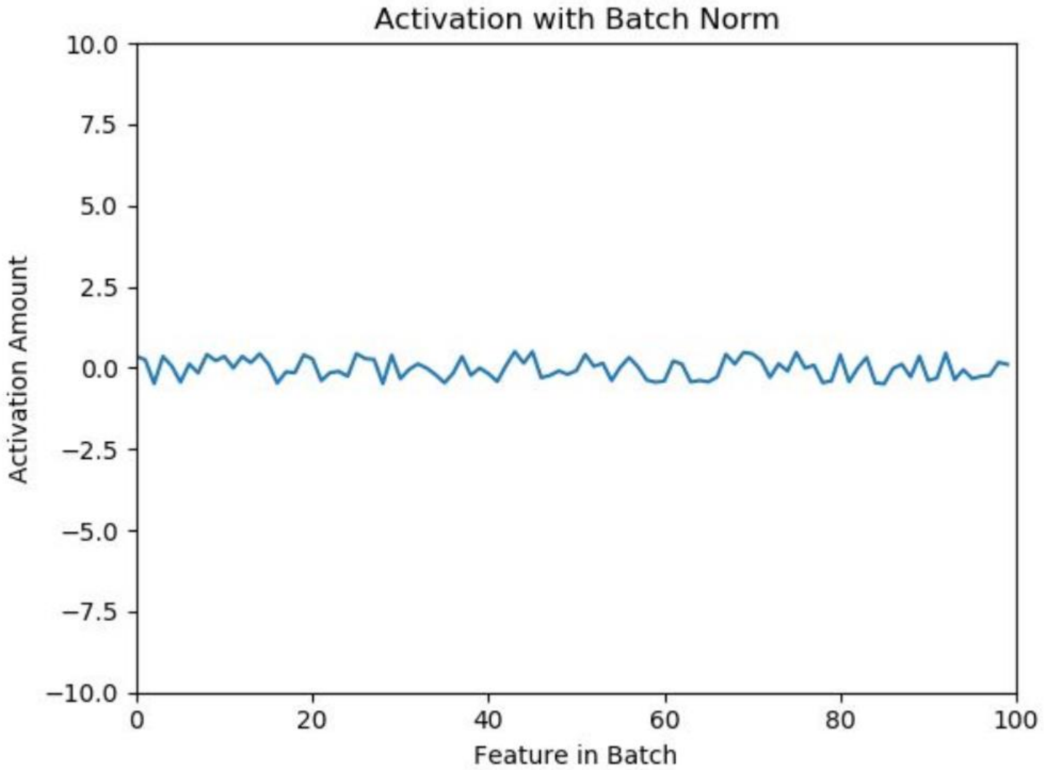


https://blog.perceptilabs.com/using-resnets-to-detect-anomalies-in-industrial-iot-textile-production/

Batch Normalization (stabilizing training)

Idea: normalize the activations for each feature at each layer



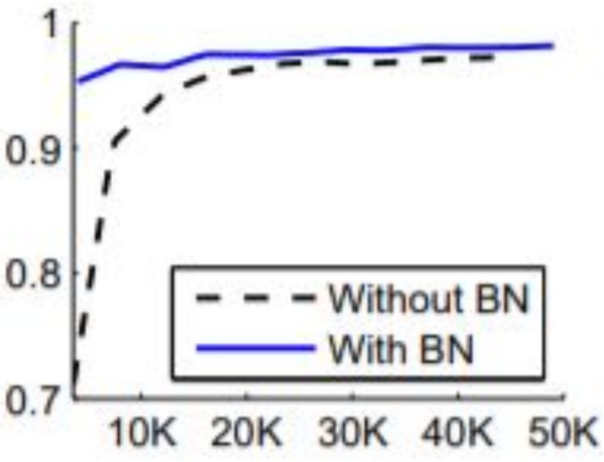


Why might we want to do this?

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

Motivation of BatchNorm

- Reduce “internal co-variate shift”
- Neural networks are trained on a certain distribution of data and are expected to be tested on the same distribution
- If we were to scale the colors of an image significantly at test time, we wouldn’t expect a neural network to do well
- The same can be said for our intermediate layers
  - They expect a certain distribution of inputs, if that changes significantly from example to example, it will be hard to learn
- (Most commonly cited reason for using BatchNorm)

The only issue is that controlling internal covariate shift does not matter that much…

How Does Batch Normalization Help Optimization?

Shibani Santurkar* MIT shibani@mit.edu

Dimitris Tsipras* MIT tsipras@mit.edu

Andrew Ilyas* MIT ailyas@mit.edu

Aleksander Mądry MIT madry@mit.edu

Abstract

Batch Normalization (BatchNorm) is a widely adopted technique that enables faster and more stable training of deep neural networks (DNNs). Despite its pervasiveness, the exact reasons for BatchNorm's effectiveness are still poorly understood. The popular belief is that this effectiveness stems from controlling the change of the layers' input distributions during training to reduce the so-called "internal covariate shift". In this work, we demonstrate that such distributional stability of layer inputs has little to do with the success of BatchNorm. Instead, we uncover a more fundamental impact of BatchNorm on the training process: it makes the optimization landscape significantly smoother. This smoothness induces a more predictive and stable behavior of the gradients, allowing for faster training.

BatchNorm makes the loss landscape smoother with fewer local minima, saddle points, and other problematic areas for gradient descent

How Does Batch Normalization Help Optimization?

Shibani Santurkar* MIT shibani@mit.edu

Dimitris Tsipras* MIT tsipras@mit.edu

Andrew Ilyas* MIT ailyas@mit.edu

Aleksander Mądry MIT madry@mit.edu

Abstract

Batch Normalization (BatchNorm) is a widely adopted technique that enables faster and more stable training of deep neural networks (DNNs). Despite its pervasiveness, the exact reasons for BatchNorm's effectiveness are still poorly understood. The popular belief is that this effectiveness stems from controlling the change of the layers' input distributions during training to reduce the so-called "internal covariate shift". In this work, we demonstrate that such distributional stability of layer inputs has little to do with the success of BatchNorm. Instead, we uncover a more fundamental impact of BatchNorm on the training process: it makes the optimization landscape significantly smoother. This smoothness induces a more predictive and stable behavior of the gradients, allowing for faster training.

Theory, intuition, and experimental results can all tell you different things

Why does BatchNorm work so well?
Intuition: If normalizing input data works so well for training, why not normalize input features to intermediate layers?

Theory/experiments: Makes gradients of loss function “better”

Why do CNNs work so well?
Intuition: Looking for a way to get “spatial reasoning” or translational invariance

Theory/experiments: Maybe it’s just that using fewer weights lets us go deeper and deep networks learn better (and also they have spatial reasoning)

Recap

Translations

Convolutions and Pooling give us translationally equivariant layers in our network

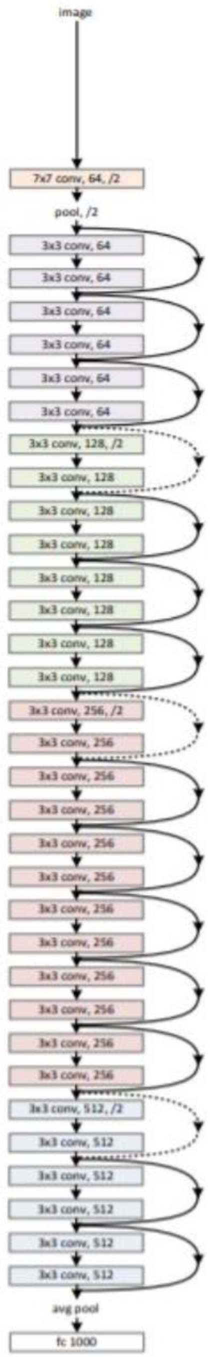
Small translations in input cause translations in output

CNN Architectures

Convolutions let us train train deeper networks than MLPs

Adding significantly more depth presents new challenges (vanishing/exploding gradients)

Residual layers and batch norm can help reduce those effects



Looking Ahead

- Can we extend convolutions to other types of structured data (e.g. graphs)?
- Is there any way to get around the fact that CNNs must take a constant input size?
  - Graph Convolutional NNs
- Do CNNs learn features that humans use for image identification, or are they doing something else entirely?
  - Interpretation and Adversarial Learning