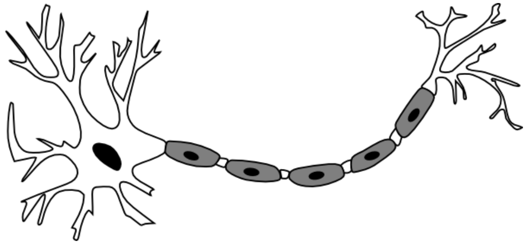
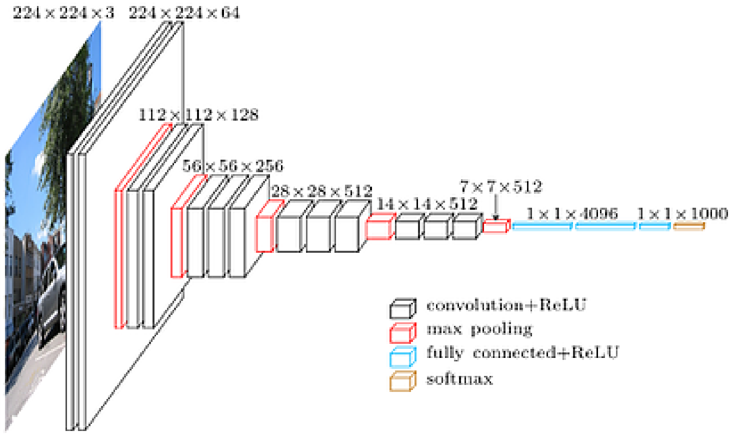
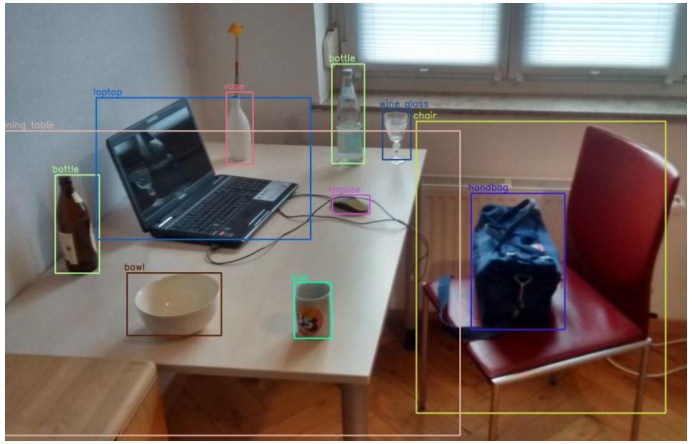
Neural Networks:
Tools and Principles Part 1

Intelligent Computational Media, Spring 2022

Nam Hee Gordon Kim



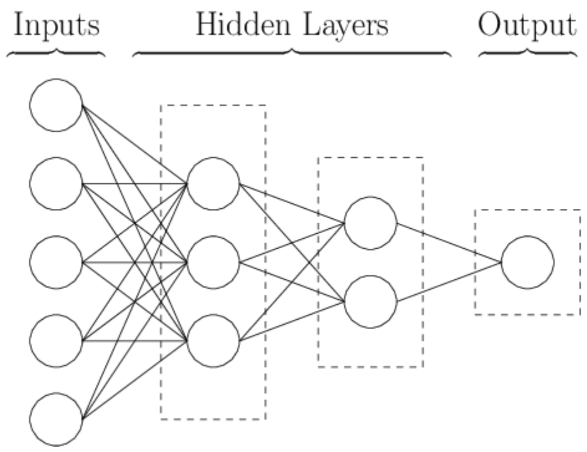




(c) Masks

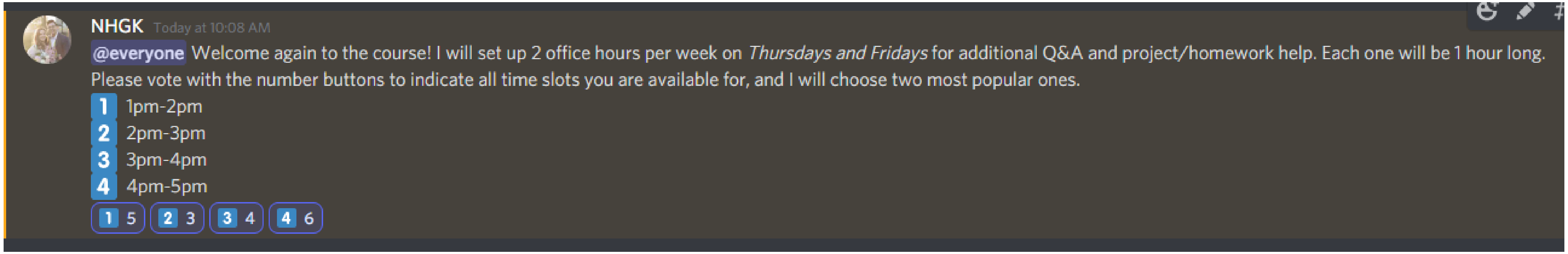
TL;DR

- What are artificial neural networks?
  - Versatile data-driven input/output mappings
  - Bread-and-butter of modern AI systems
  - Sequence of matrix multiplications and nonlinear activations
  - (Usually) deeper => *more expressive*



- What do we need for training a neural network?
  - Data (ideally: clean data)
  - Input/output representations
  - NN architecture, loss function
  - Optimization algorithm
  - Differentiable programming

Announcements



Vote for office hours (project + homework help)

Learning Goals

- Understand the relevance of neural networks (NNs) in ICM
- Get primitive understanding of inference and learning with NNs
- Glimpse at more advanced architectures
  - Convolutional neural networks
  - Autoencoders
  - Skip-connections and residual neural networks
- Understand fine tuning and transfer learning (warm-start)

Important Note on This Lecture

- Textbook: Perttu’s Old Slides (POS).
- This lecture will cover important first steps for project/assignments, and *very brief* overview of NN variants.
- It’s up for YOU to learn the materials in more depth. Ask questions, look into online course materials, watch YouTube videos, etc.

Bonus Slides

- We will cover basics, but I will leave “bonus topics” for you to study after introducing each concept.
- Bonus slides are marked with this background colour.
- POS covers overview of most of these topics.
- My UBC course also covers most of these topics:

(https://www.cs.ubc.ca/~nhgk/courses/cpsc340s21/)

Ian Goodfellow — "Deep Learning"

(Pause)

Why Do We Care?

Your Relationship to Neural Networks

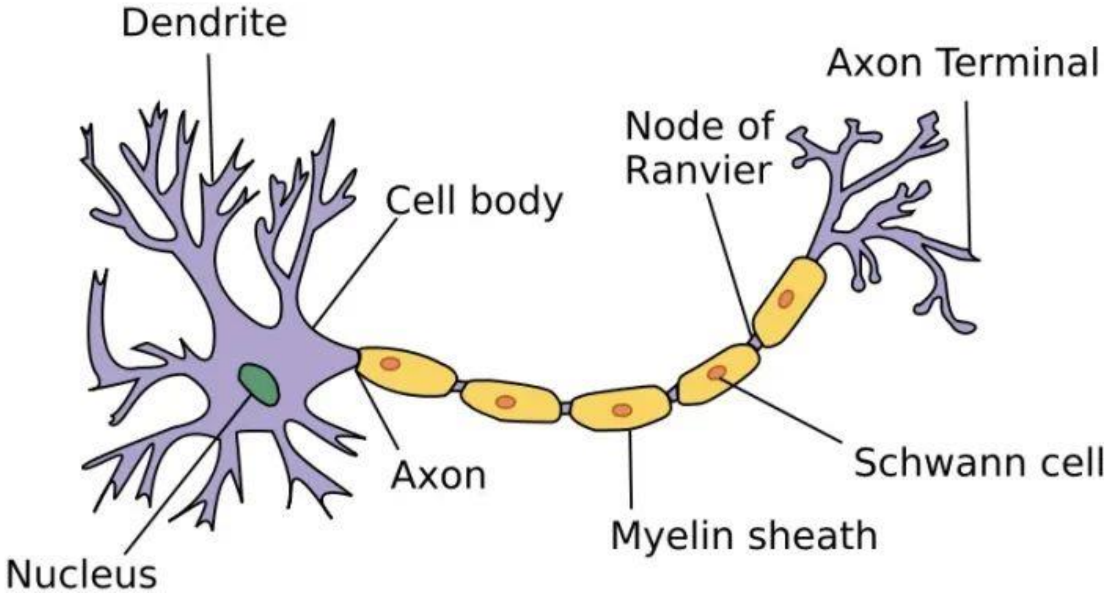
- NN is ___data-driven I/O mappings___ that is very versatile
  - e.g. image input to audio output
  - e.g. game state input to game control output
  - e.g. noise input to image output
  - ... and so many more!
- NNs can rapidly generate prototypes or starting points
  - e.g. use GPT-3 to generate a game synopsis
  - e.g. use GANs to generate texture assets
- NNs can solve complex problems by using data
  - e.g. controlling a physics-based character (robot)
- NNs can be imbued with imagination and enrich yours
  - e.g. poetry to describe sound of nature

What Are Neural Networks?

Nice Videos on Neural Networks (3Blue1Brown)

- https://www.youtube.com/watch?v=aircAruvnKk (what is a neural network)
- https://www.youtube.com/watch?v=IHZwWFHWa-w (how neural networks learn, i.e., gradient descent)
- https://www.youtube.com/watch?v=Ilg3gGewQ5U (what is backpropagation really doing)
- Essence of Linear Algebra (vectors, matrices, determinants etc., the branch of math at the heart of it all): https://www.youtube.com/watch?v=kjBOesZCoqc&list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab

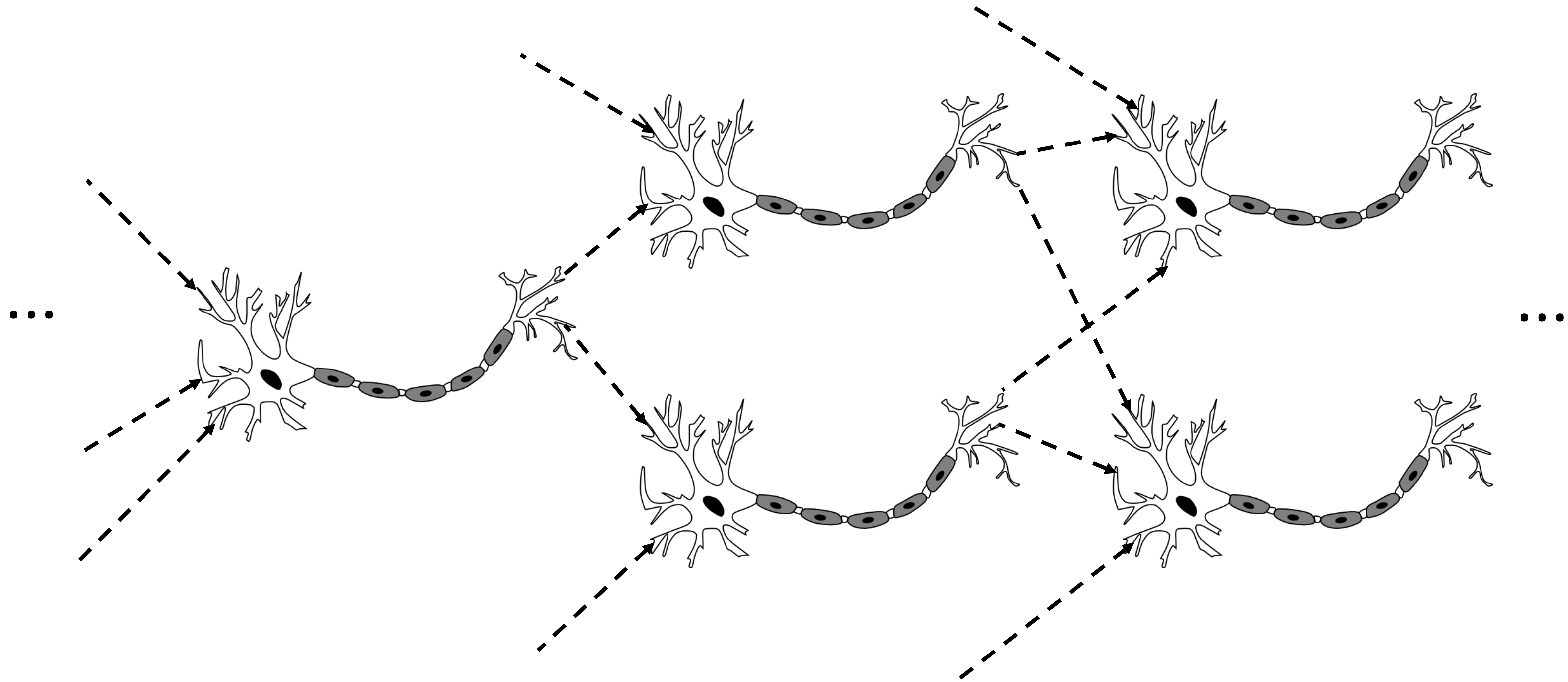
(Supposedly) Biological Motivation: Neuron



Gather information

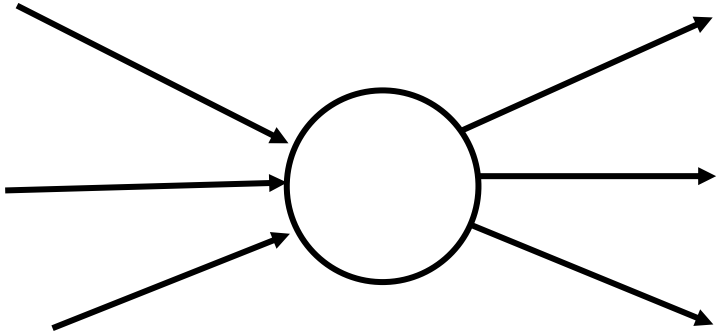
Propagate information

(Supposedly) Biological Motivation: Network

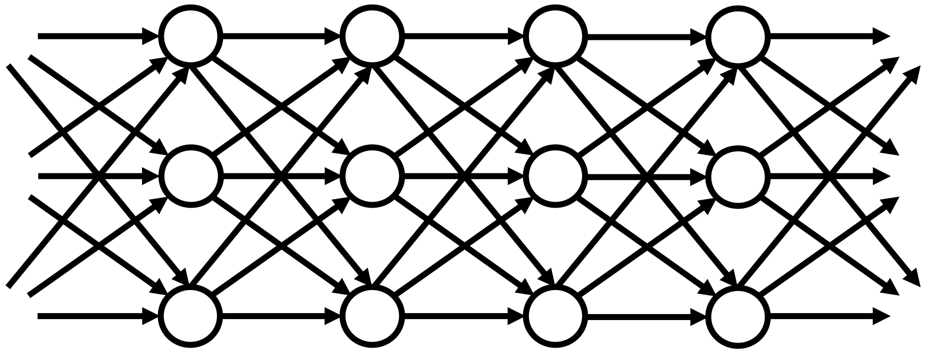


Artificial Neural Networks

Neuron

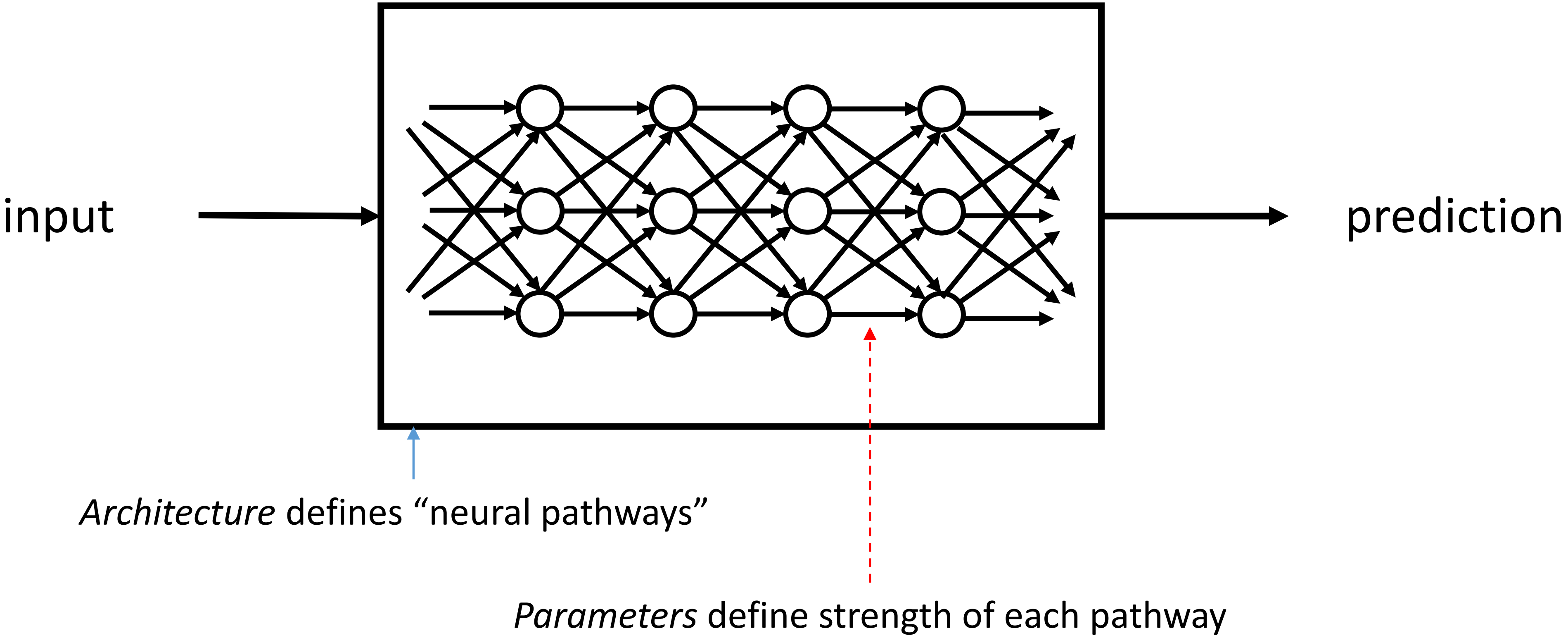


"Neuron"



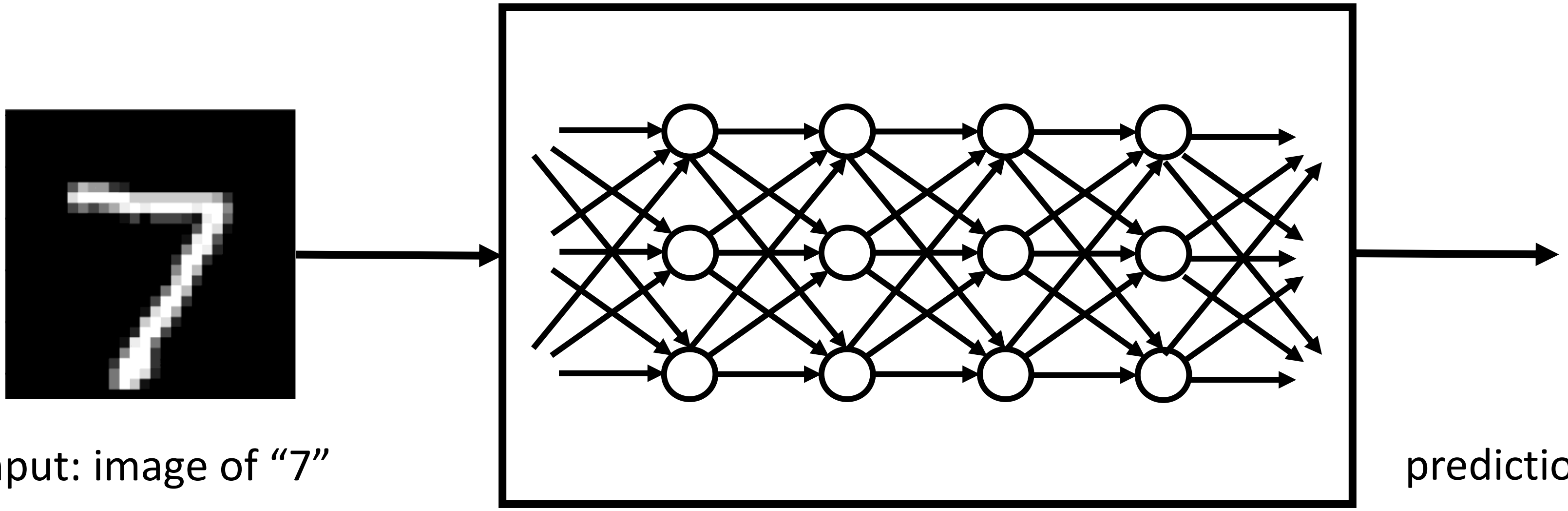
"Neural Network"

“Inference” with Neural Networks



Representations

Q: How are “input” and “output” represented?



input: image of “7”

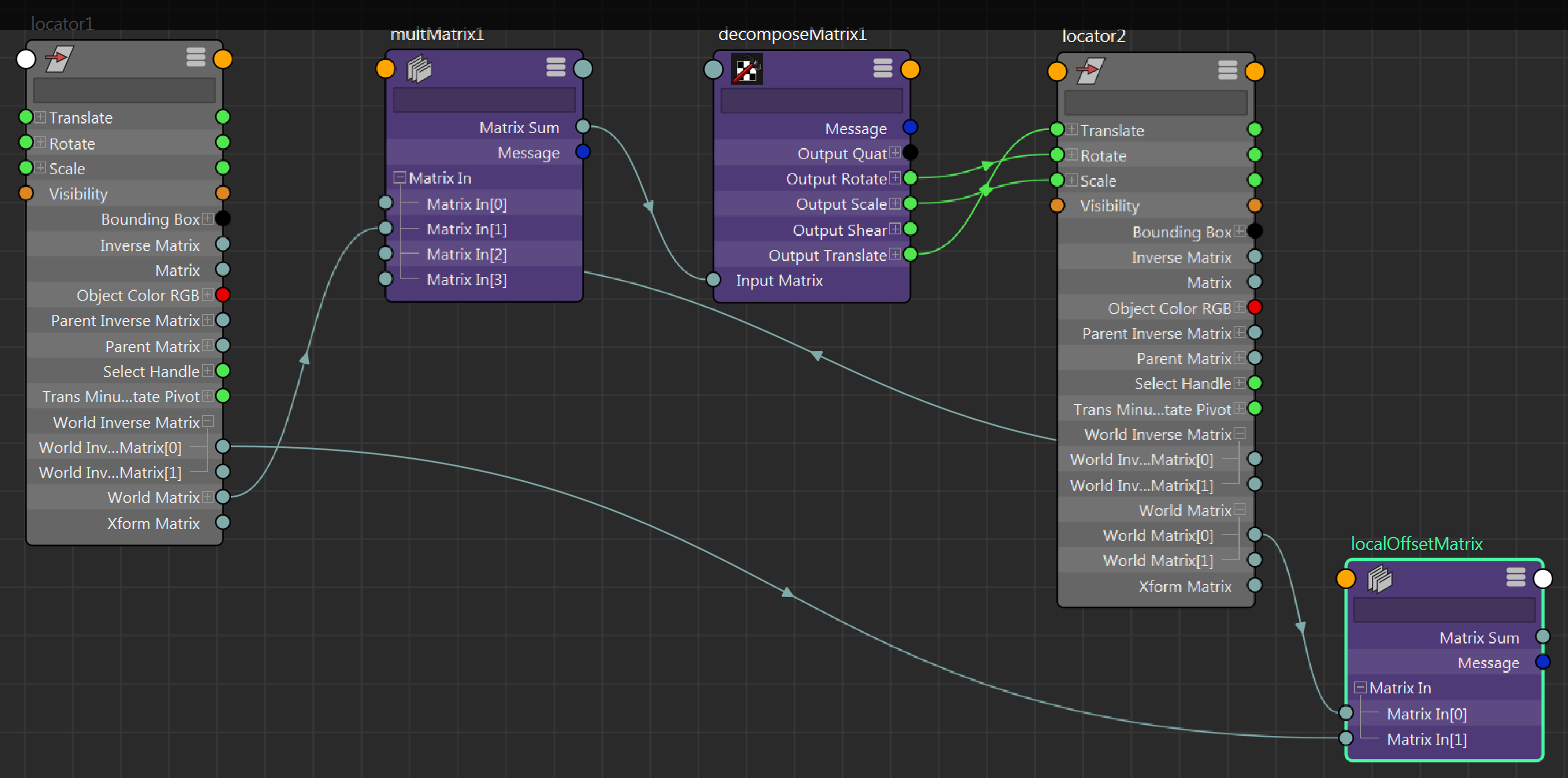
prediction: character “2”
(wrong)

Last Slide Before Demo 1

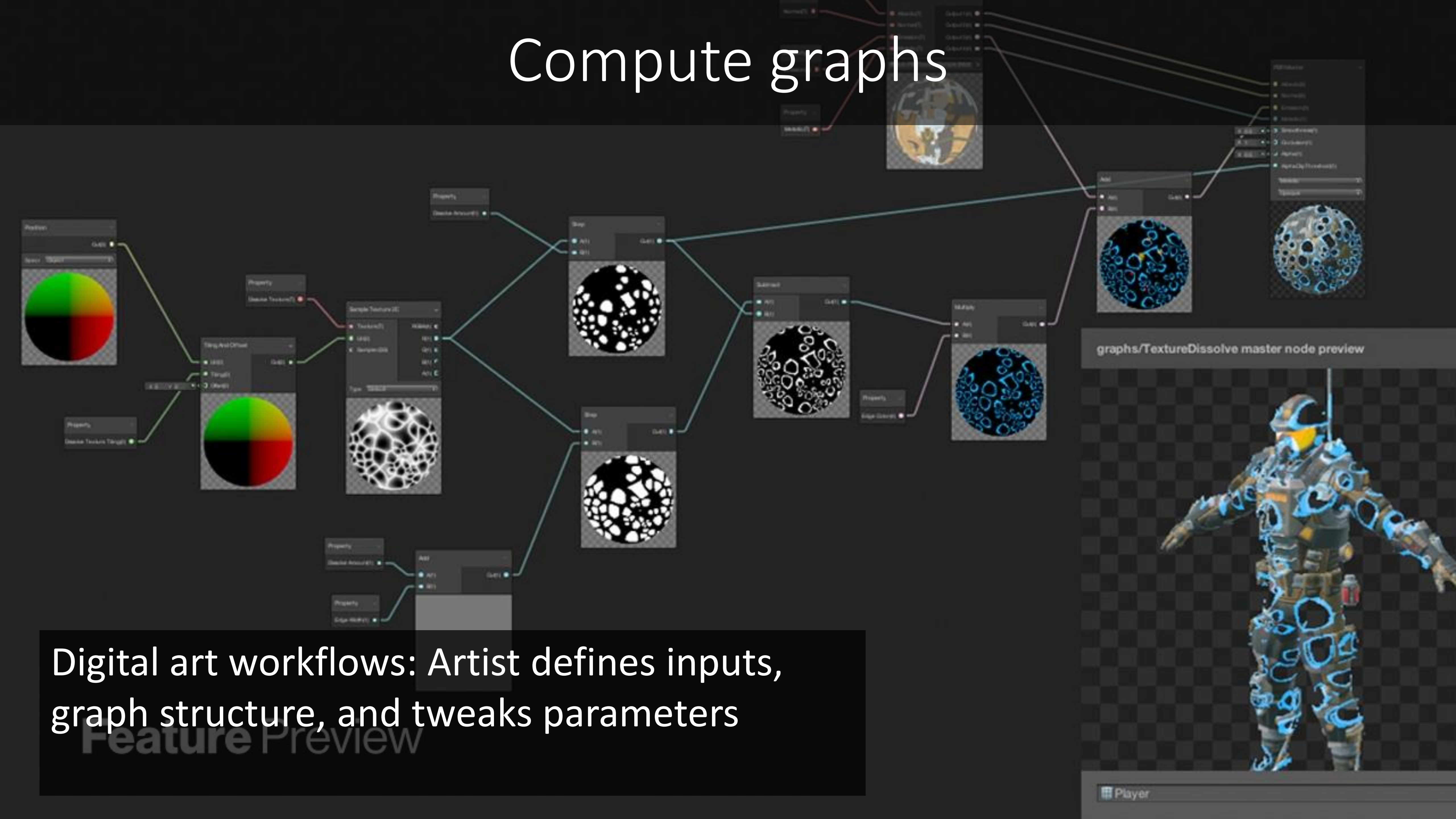
- Now we have all the ingredients for *inference!*
  - NN. (pre-trained)
  - Data
  - I/O representations.

Live Demo 1:
Neural Networks in X lines of Python

Compute graphs

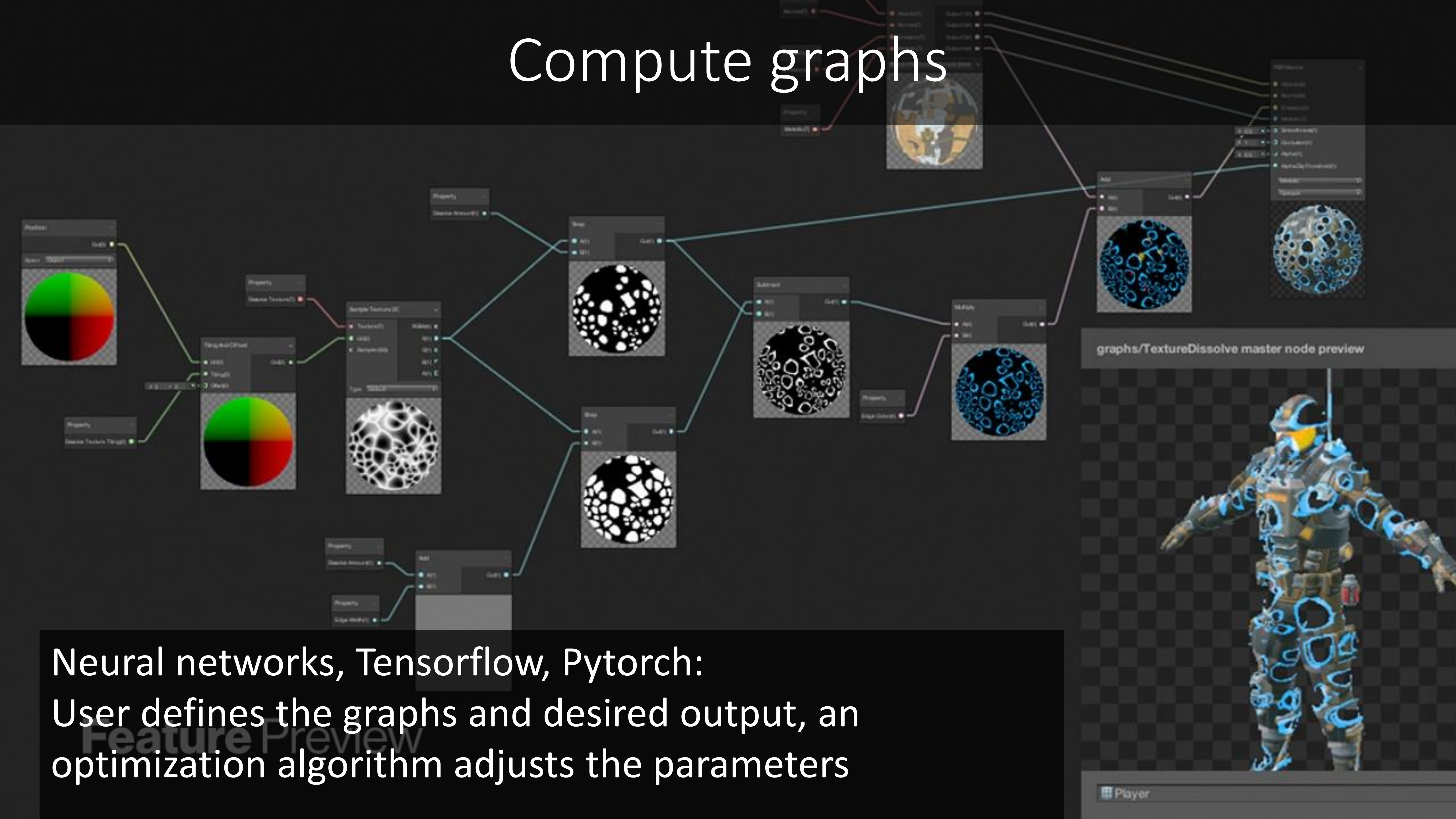


Compute graphs



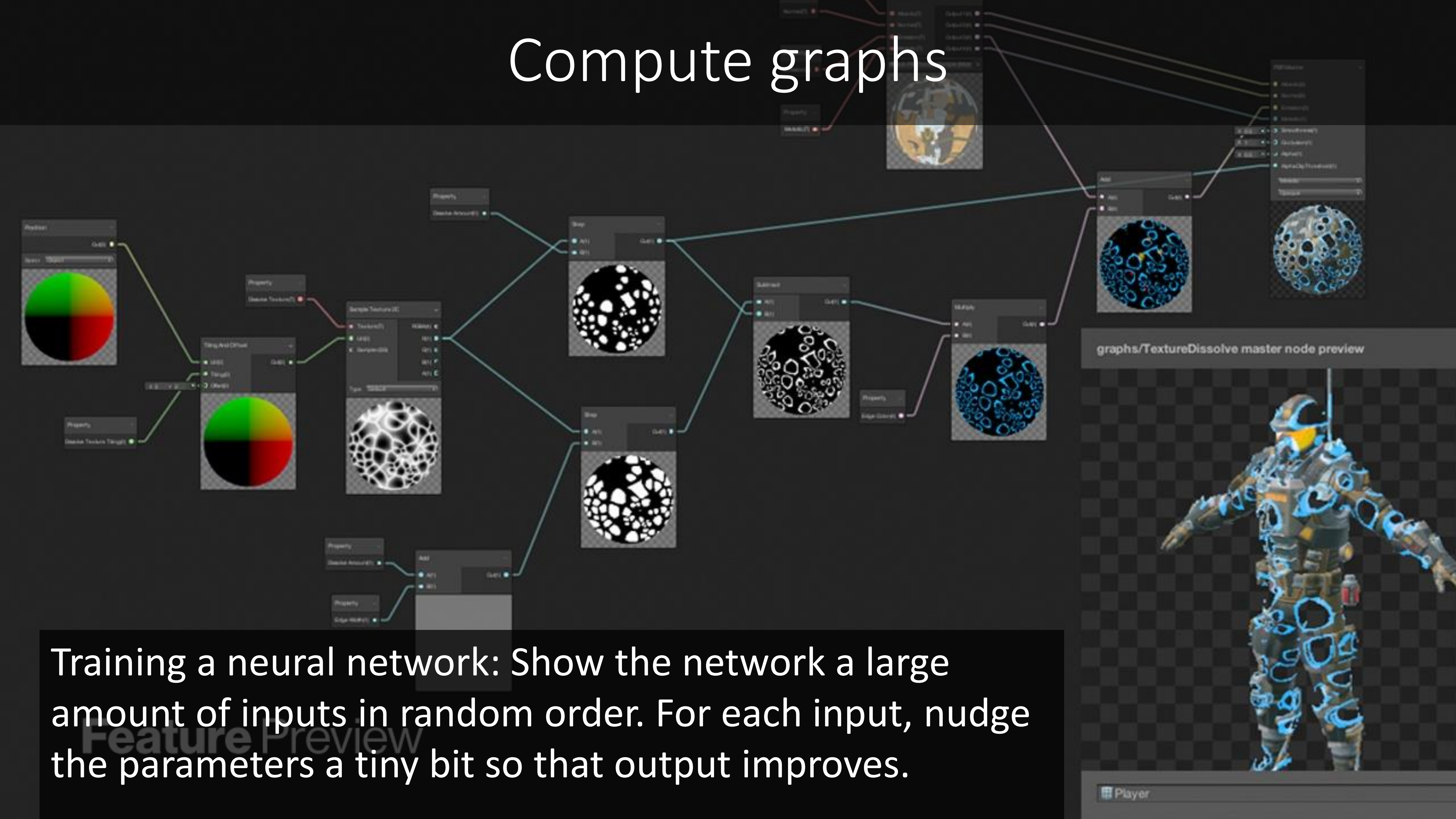
Digital art workflows: Artist defines inputs, graph structure, and tweaks parameters

Compute graphs



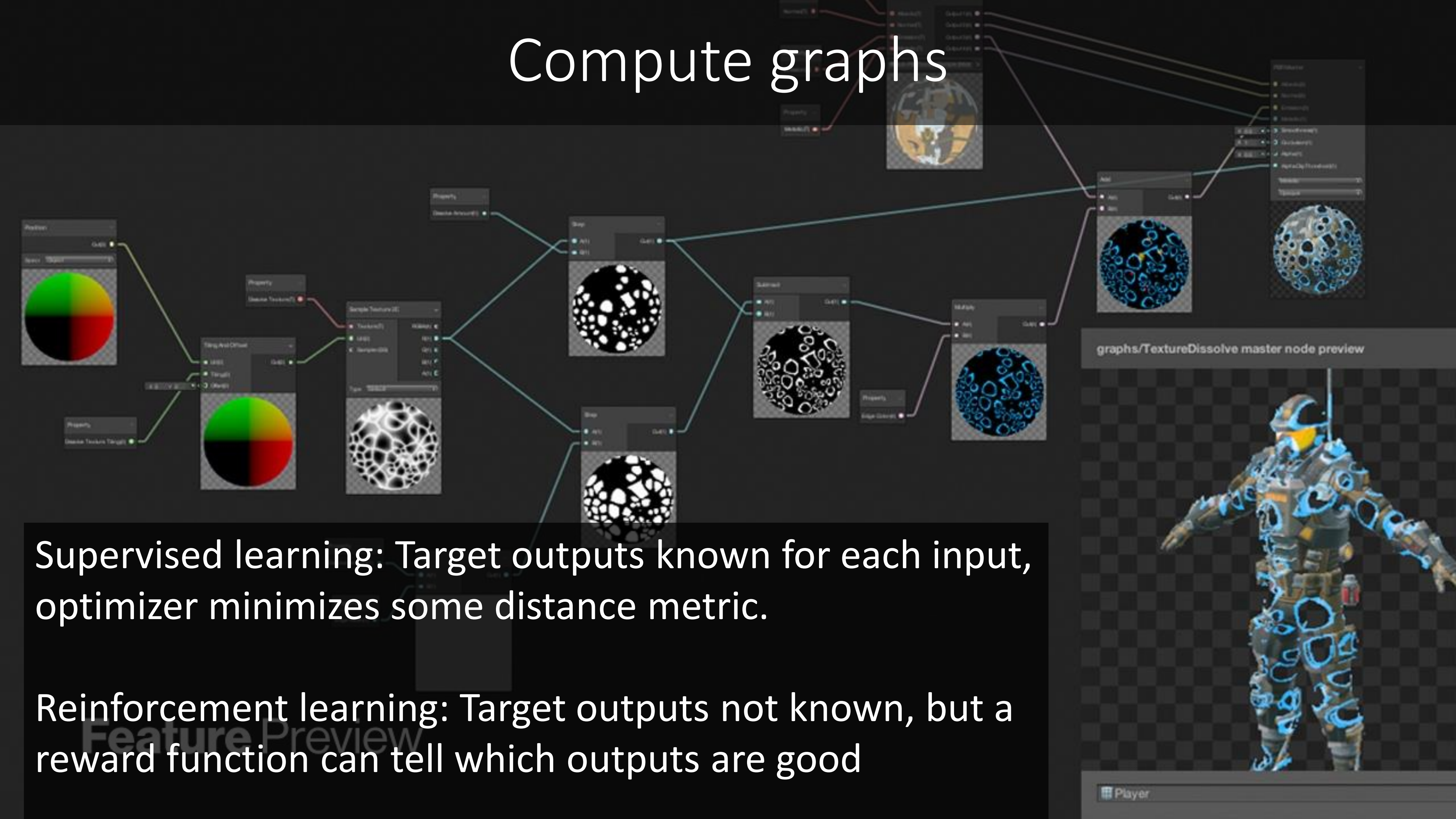
Neural networks, Tensorflow, Pytorch:
User defines the graphs and desired output, an optimization algorithm adjusts the parameters

Compute graphs



Training a neural network: Show the network a large amount of inputs in random order. For each input, nudge the parameters a tiny bit so that output improves.

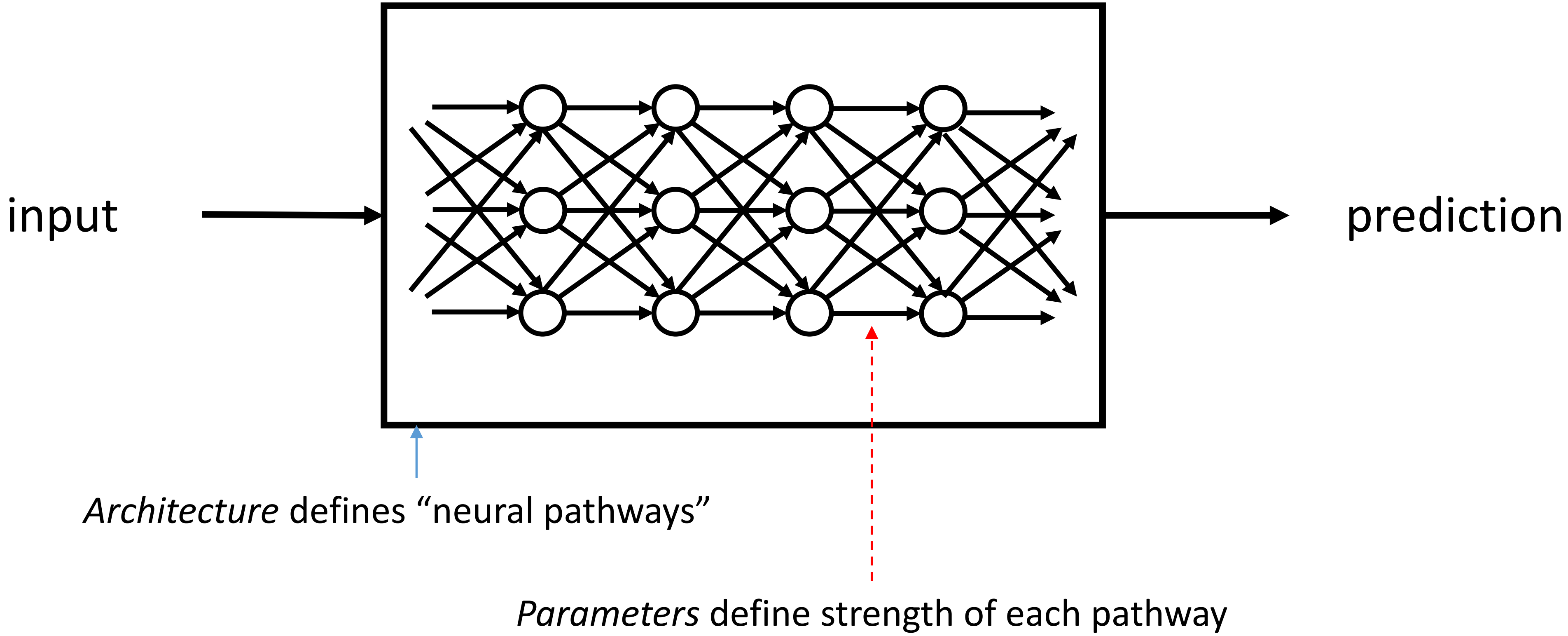
Compute graphs



Supervised learning: Target outputs known for each input, optimizer minimizes some distance metric.

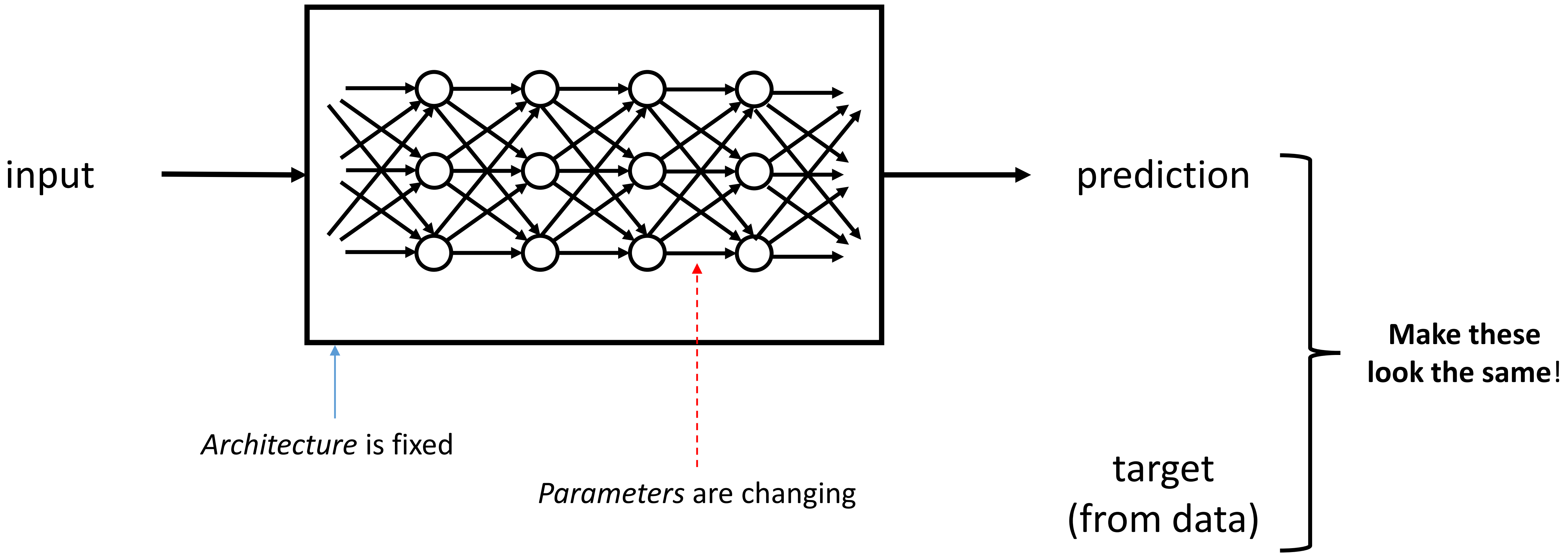
Reinforcement learning: Target outputs not known, but a reward function can tell which outputs are good

“Inference” with Neural Networks



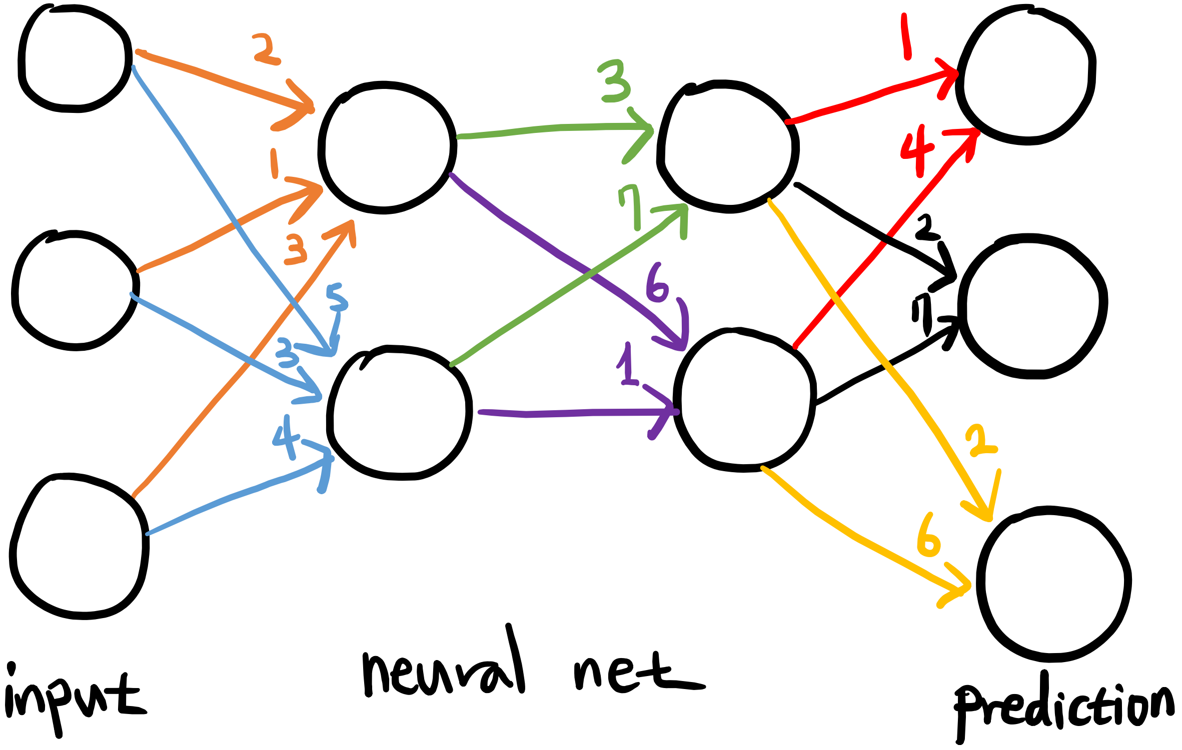
Q: What makes a prediction “good”?
Shouldn’t there be some kind of *reference*?

“Learning” with Neural Networks



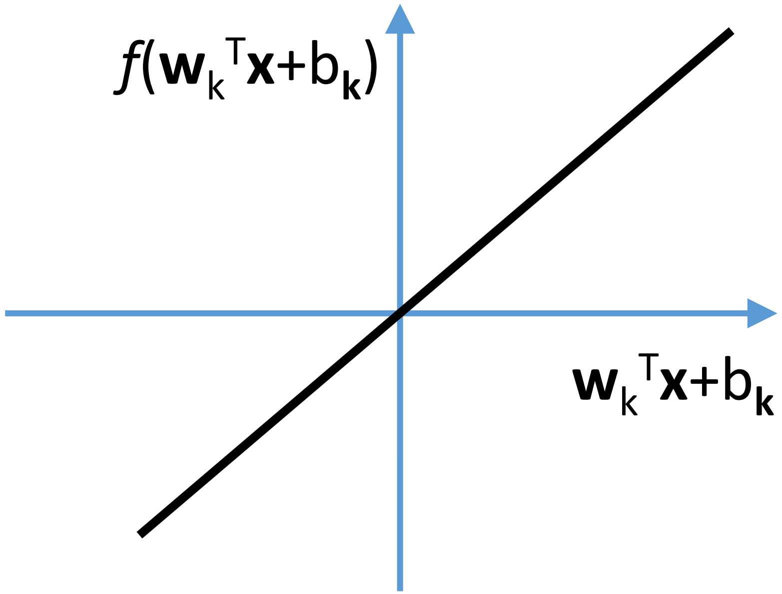
Neural Network is
Matrix Multiplication + Nonlinear Activation

- “Matrix multiplication”: lets you express networks that does weighted sum
- “Nonlinear activation”: lets you compound matrix multiplications

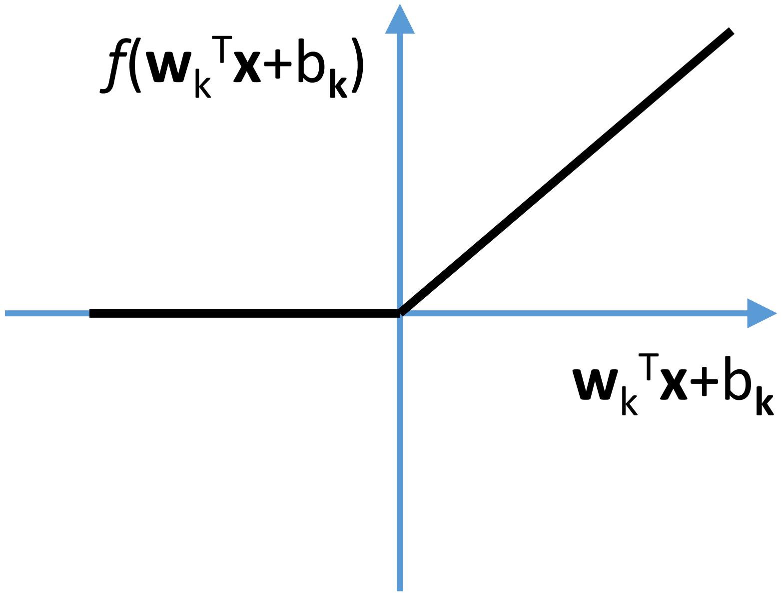


$$\begin{bmatrix} 2 & 1 & 3 \\ 5 & 3 & 4 \end{bmatrix}, \begin{bmatrix} 3 & 7 \\ 6 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 4 \\ 2 & 7 \\ 2 & 6 \end{bmatrix}$$

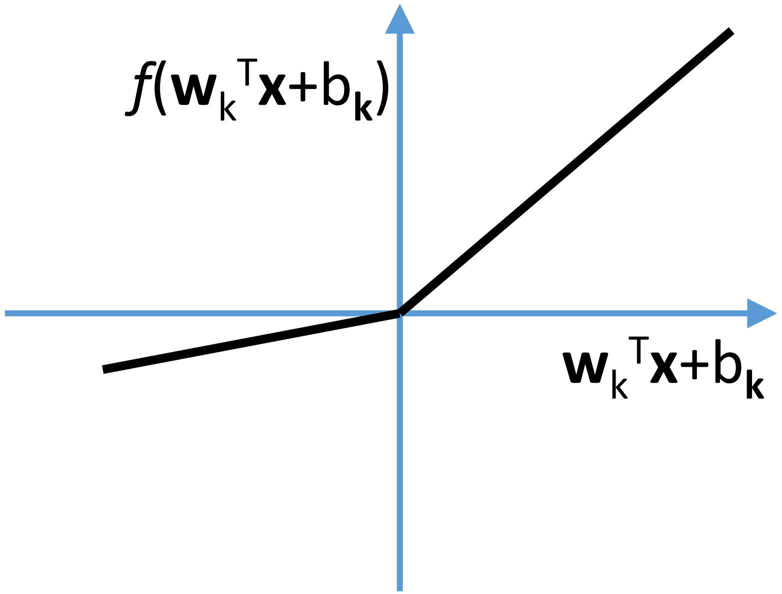
Linear activation



ReLU activation

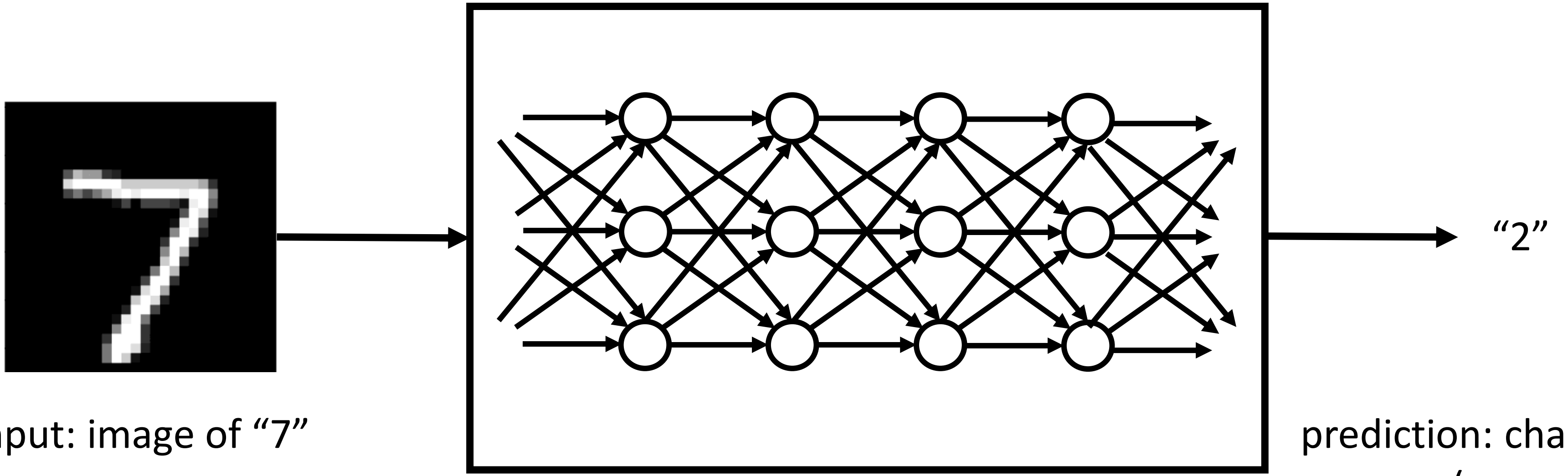


Leaky ReLU



Neural Networks is *Differentiable Operation*

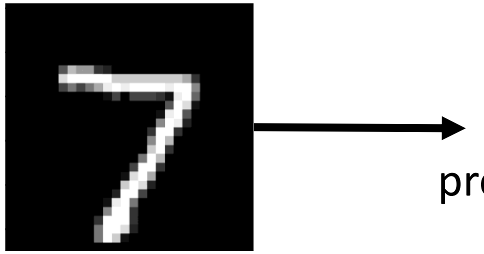
- "Differentiable": property that lets you change parameters to optimize a loss function with *gradient descent*
- "Differentiable programming": math/programming framework that lets you compose differentiable operations (like neural networks)



input: image of "7"

prediction: character "2"
(wrong)

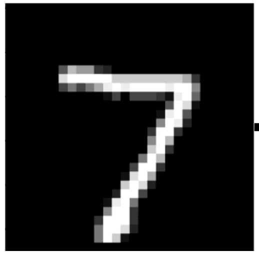
more concisely...



"2"
prediction

input

Loss Function



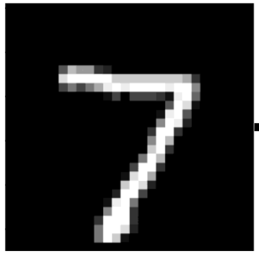
input

“2”
prediction

“7”
target

Make these look the same!
(by changing neural network’s parameters)

Loss Function



input

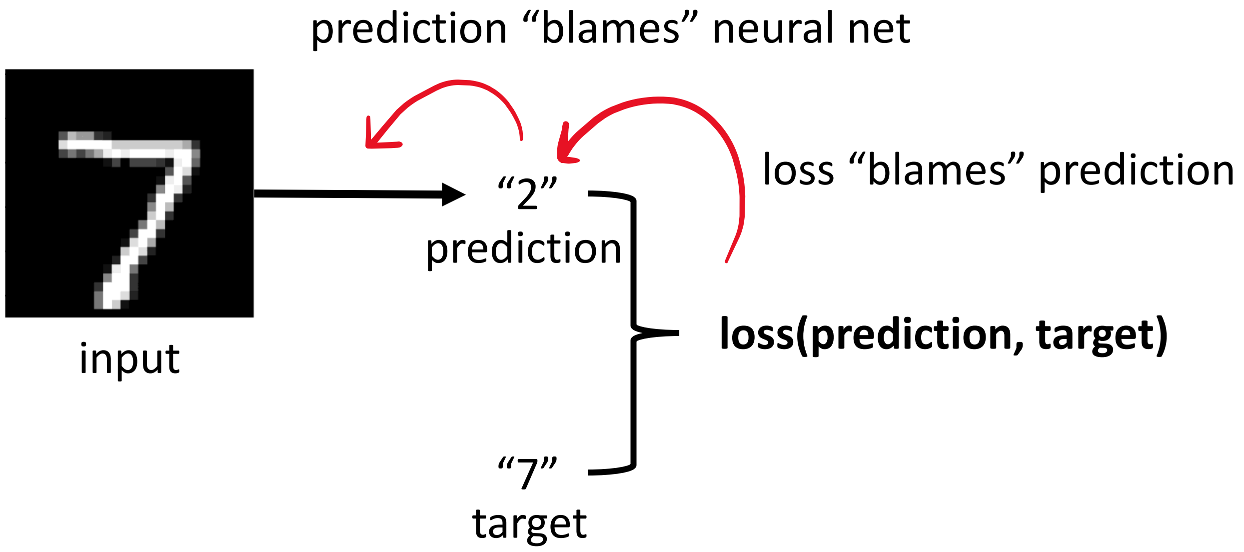
“2”
prediction

“7”
target

loss(prediction, target) ← *Minimize* this using *gradient descent*

Backpropagation and Optimization Algorithm

- "Backpropagation": computes gradients for setting the NN parameters. (very fast with differentiable programming)
- "Optimization algorithm": e.g. Adam, SGD, uses gradients to change neural network's parameters

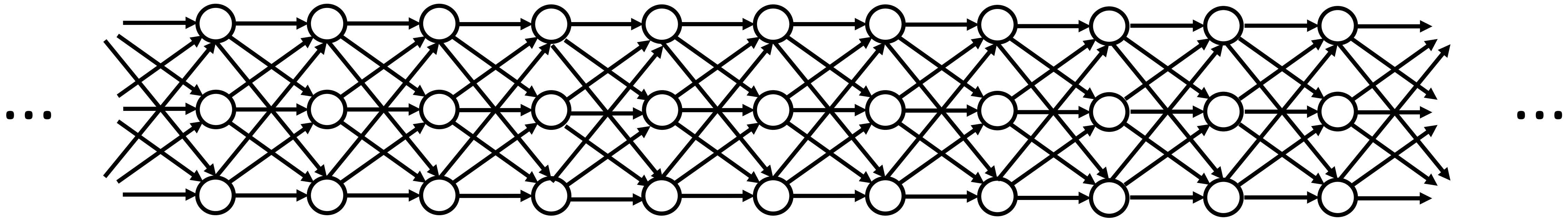


Last Slide Before Demo 2

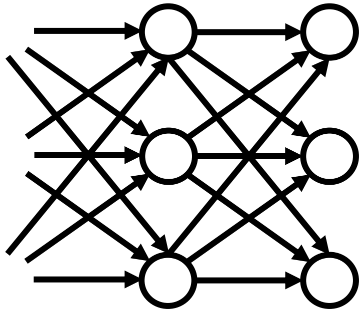
- Now we have all ingredients for *training!*
  - Neural network
  - Input/output representation
  - Data (ideally *clean*)
  - Differentiable programming
  - Loss function
  - Optimization algorithm

Live Demo 2:
Training a Neural Network

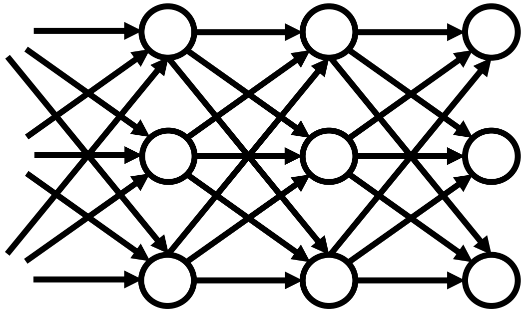
Neural Networks Can Be *Deep*



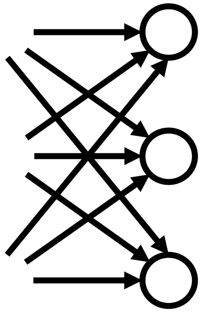
and *Modular*



Neural net 1

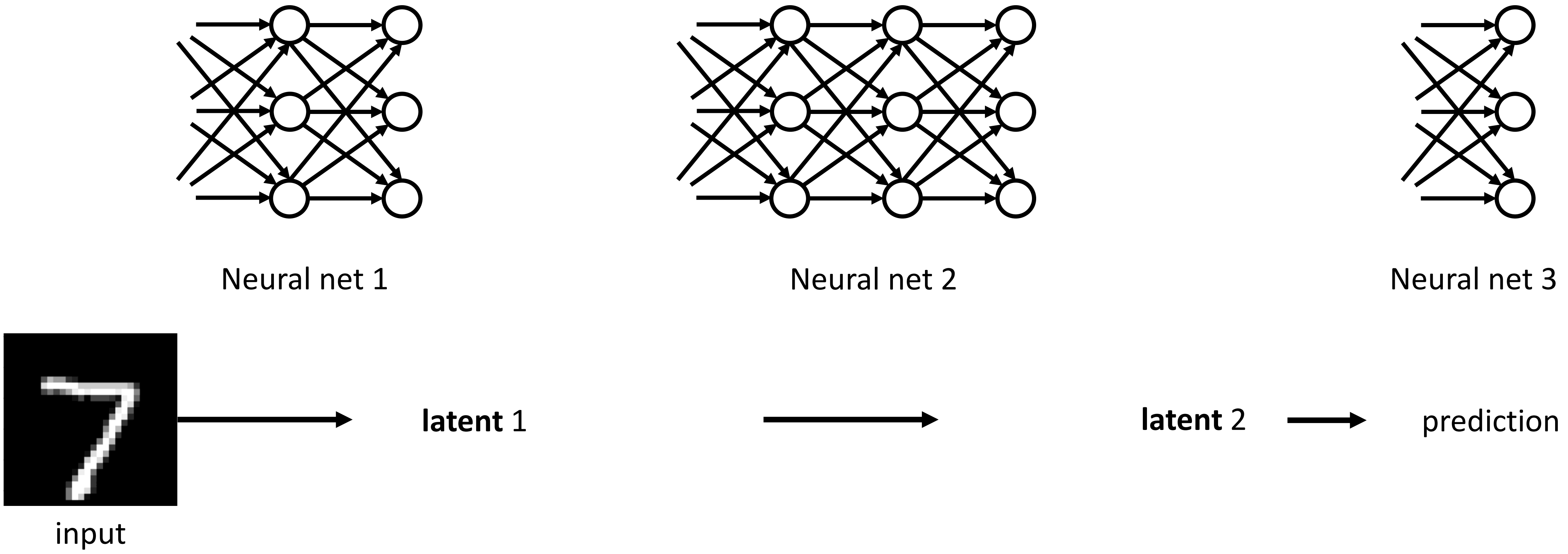


Neural net 2



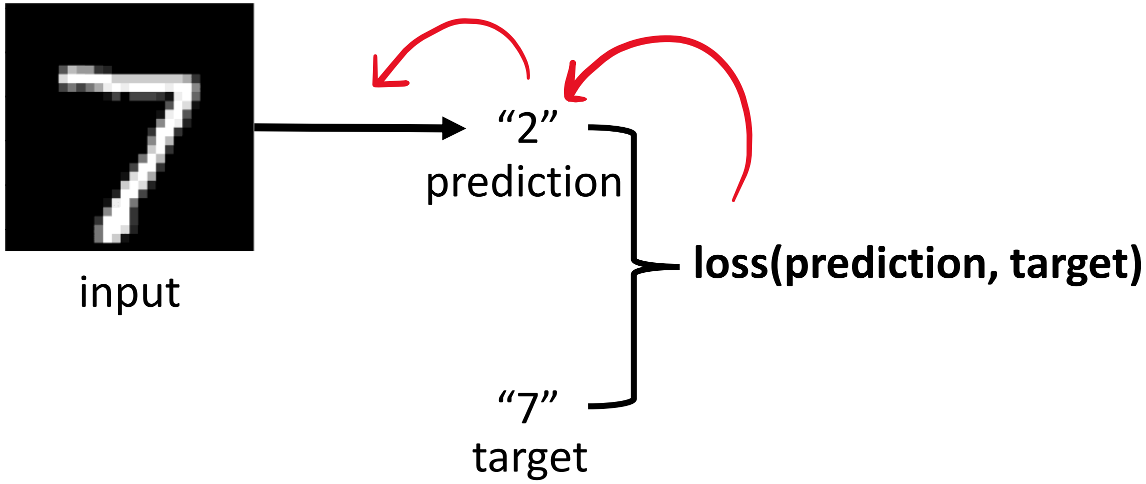
Neural net 3

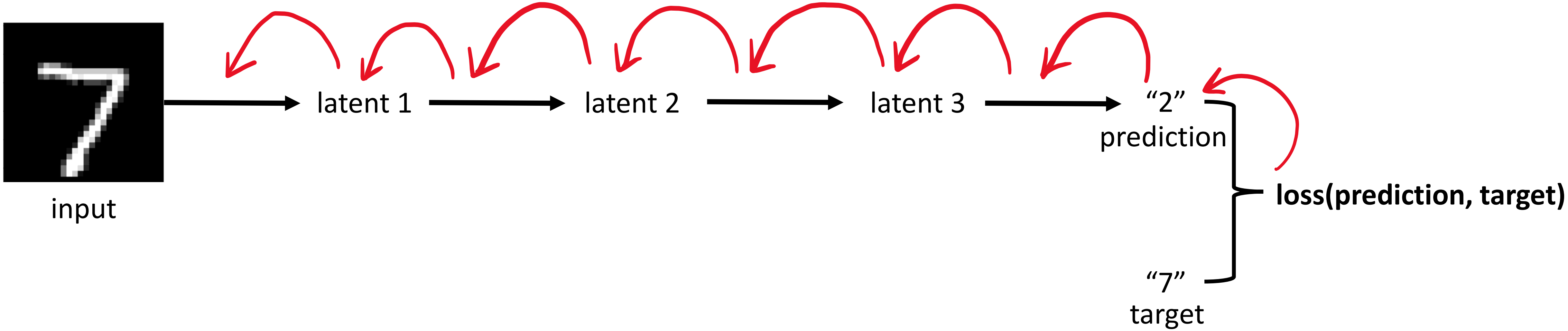
Latents



- “Latents”: intermediate representations resulting from neural net modules (aka “embeddings” or “hidden features”)

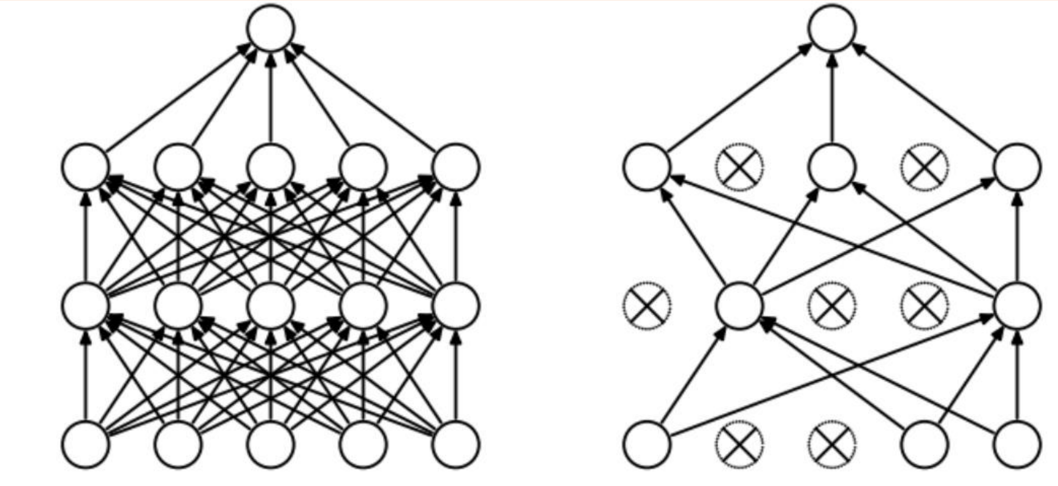
Showing Backpropagation Again





Bonus Topics for Neural Network Basics

- What are weights and biases? (https://deepai.org/machine-learning-glossary-and-terms/weight-artificial-neural-network)
- Choice of nonlinear activations (https://towardsdatascience.com/visualizing-the-non-linearity-of-neural-networks-c55b2a14ad7a)
- Supervised learning, unsupervised learning, reinforcement learning
- Fundamental tradeoff of machine learning
  - Overfitting
- Regularization
- Mini-batching


(a) Standard Neural Net (b) After applying dropout.

(Pause)

Overview of Notable
Neural Networks in the Wild

Notation

- We'll show both original diagrams and my own notation for clarity
- Here's my notation:

$\longrightarrow$	$\hat{y}$	y	X	z
neural network module/layer	prediction	target	input	latent

$$X \rightarrow \left[z^{(1)} \rightarrow z^{(2)} \right] \rightarrow \hat{y}$$

$$X \longrightarrow \hat{y}$$

Image Classifier

Classic: MNIST digit classification

Input: digit image
Prediction: digit character

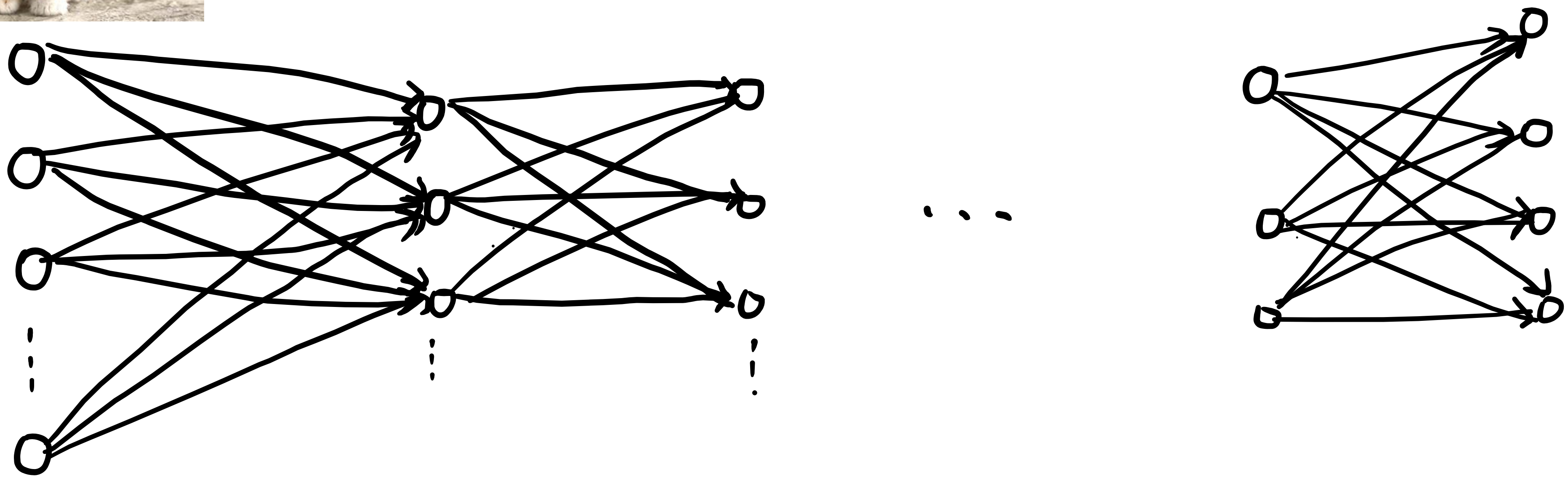
ImageNet image classification

Input: RGB image
Prediction: one of 1000 categories

Architecture Matters

$$\to z^{(1)} \to z^{(2)} \to \cdots \to z^{(h)} \to \hat{y}$$

these "layers" are fully-connected, i.e.



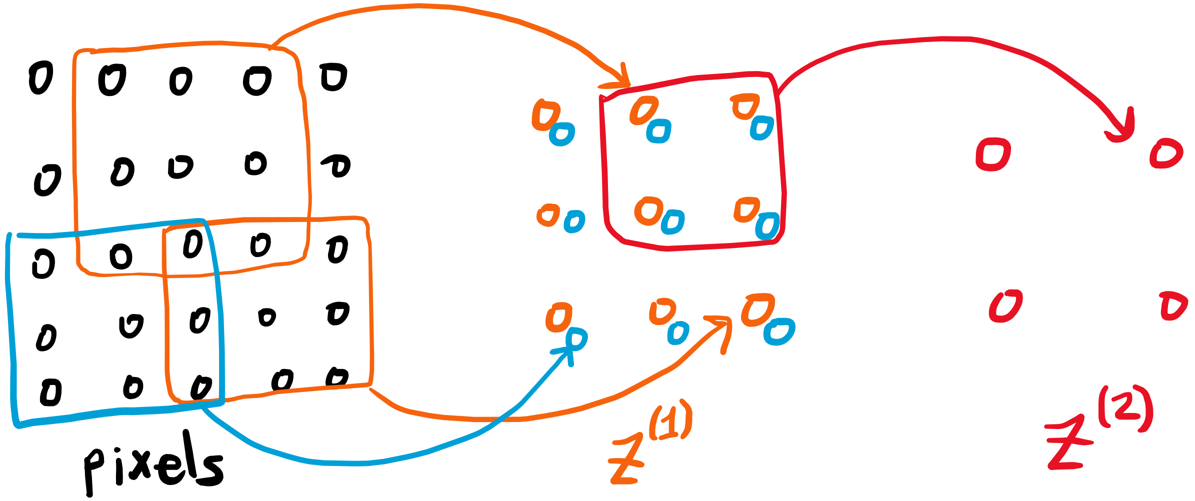
Q: Why would fully-connected layers be a bad idea?

Fully-Connected Networks (FCNs) are often inoptimal for images/sequences

- Lots of parameters, slow to optimize
- In real data, relations are often local, e.g., nearby pixels have stronger relationships. FCNs do not leverage this.
- Convolutional networks to the rescue! (fewer parameters, leverage locality)

“ear” pixels appear in the same region

Convolutional Neural Networks (CNNs)

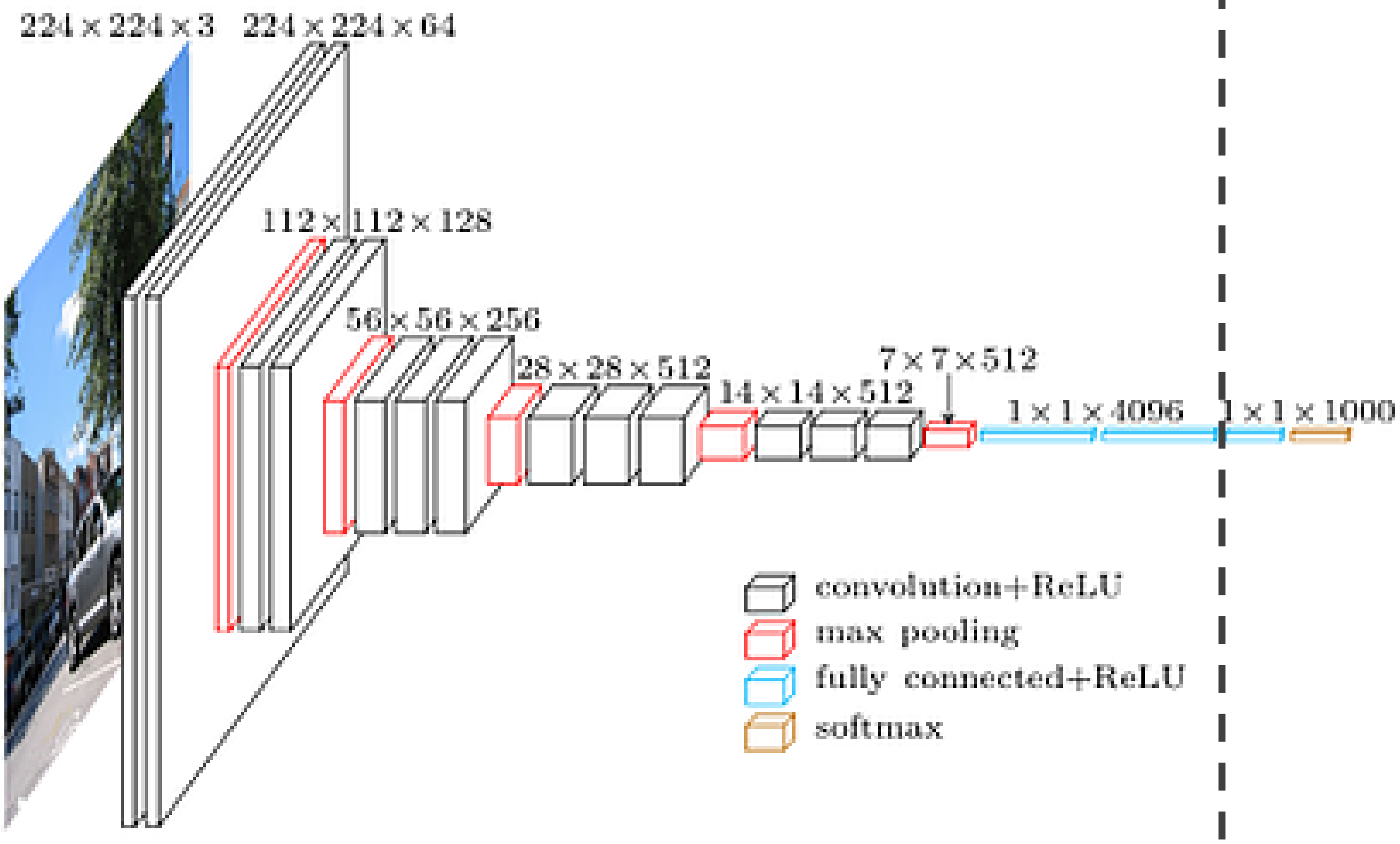


- An alternative architecture with far fewer connections
- A neuron accepts connections from *neighbouring* pixels
- Each kernel (blue, orange) has same weight for every neighbourhood
- Can be stacked: neighbourhoods of neurons, and neighbourhoods of neighbourhoods, etc.
- VERY popular for audiovisual input! (or any “sequential” input)

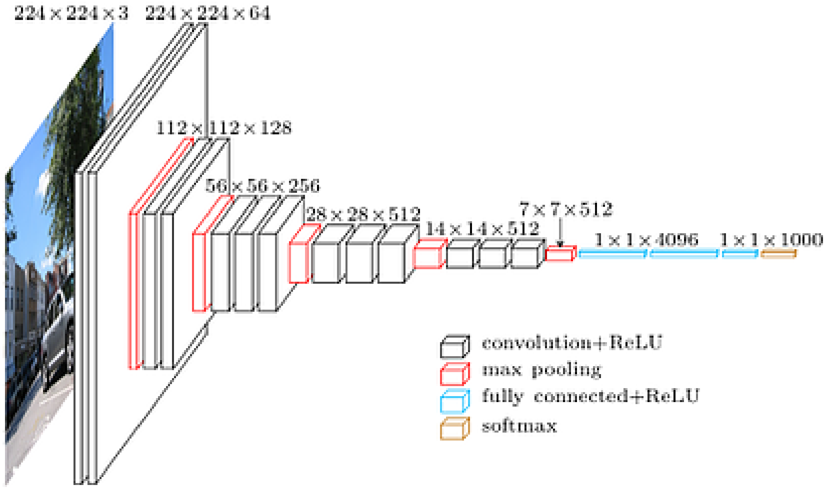
Simply put: CNNs make latents from images

Encoder, a compact representation of image contents

Simple classifier:
One neuron per
Image class



Simply put: CNNs make latents from images



$$\to z^{(1)} \to z^{(2)} \to \cdots \to z^{(h)} \to \hat{y}$$

these "layers" are **CNN**

prediction is fully-connected

$\underbrace{\hspace{6em}}$

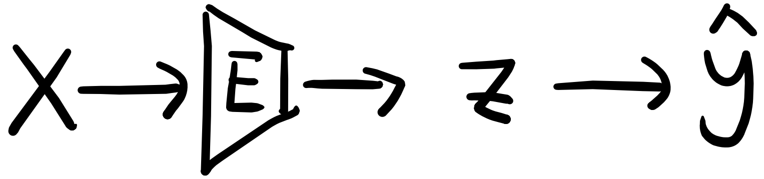
$\to \boxed{E} \to$ "encoder"

Bonus Topics for CNNs

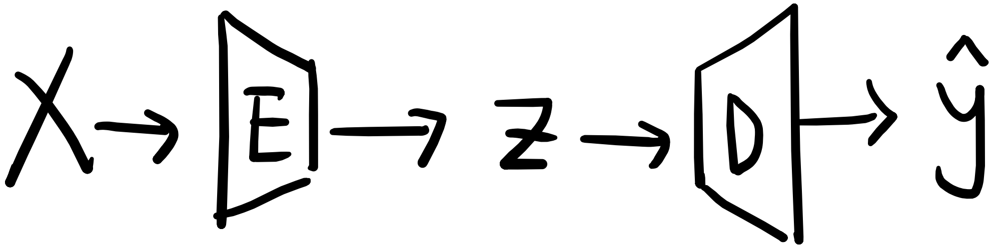
- Great resource: https://towardsdatascience.com/visualizing-the-fundamentals-of-convolutional-neural-networks-6021e5b07f69
- Pooling layers
- Invariance and data augmentation (https://www.cs.ubc.ca/~nhgk/courses/cpsc340s21/slides/L7_updated.pdf, slide 22 onwards)
- Image filters
- Edge detectors

Encoder-Decoder View of Neural Networks

Instead of:



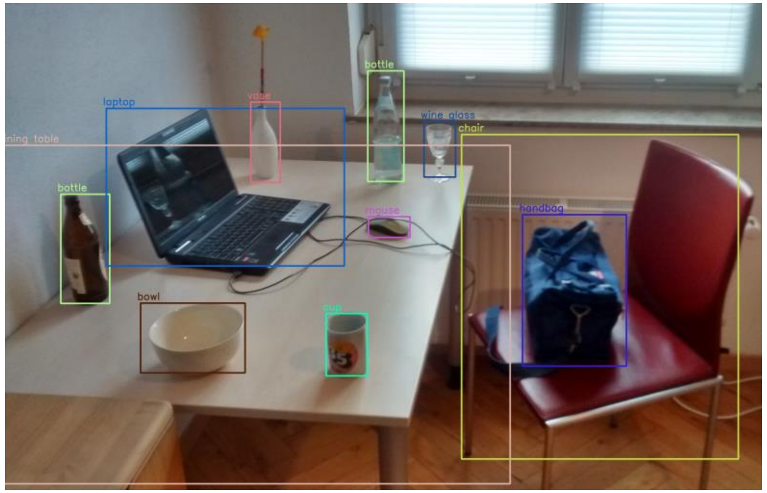
Let’s write:



encoder

decoder

Object Detector



Input: image
Prediction: *bounding boxes*

Input: image
Prediction: *image segments*

Input: image
Prediction: *human pixels*

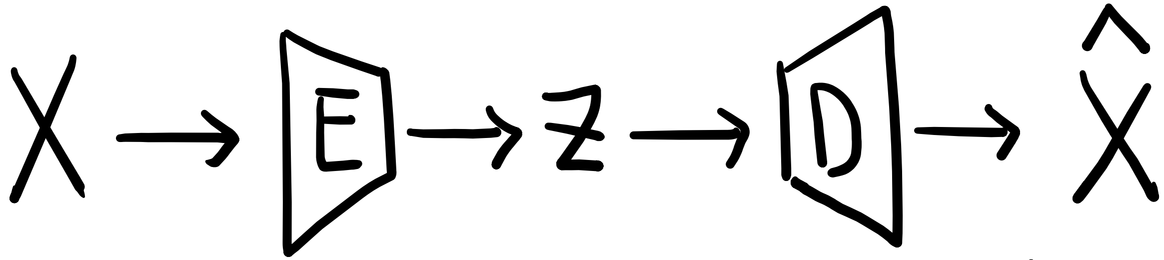
$$X \rightarrow E \rightarrow z \rightarrow D \rightarrow \hat{y}$$

RGB image

image, but labels
instead of RGB

Encoder-Decoder CNN is the underlying architecture of many object detectors

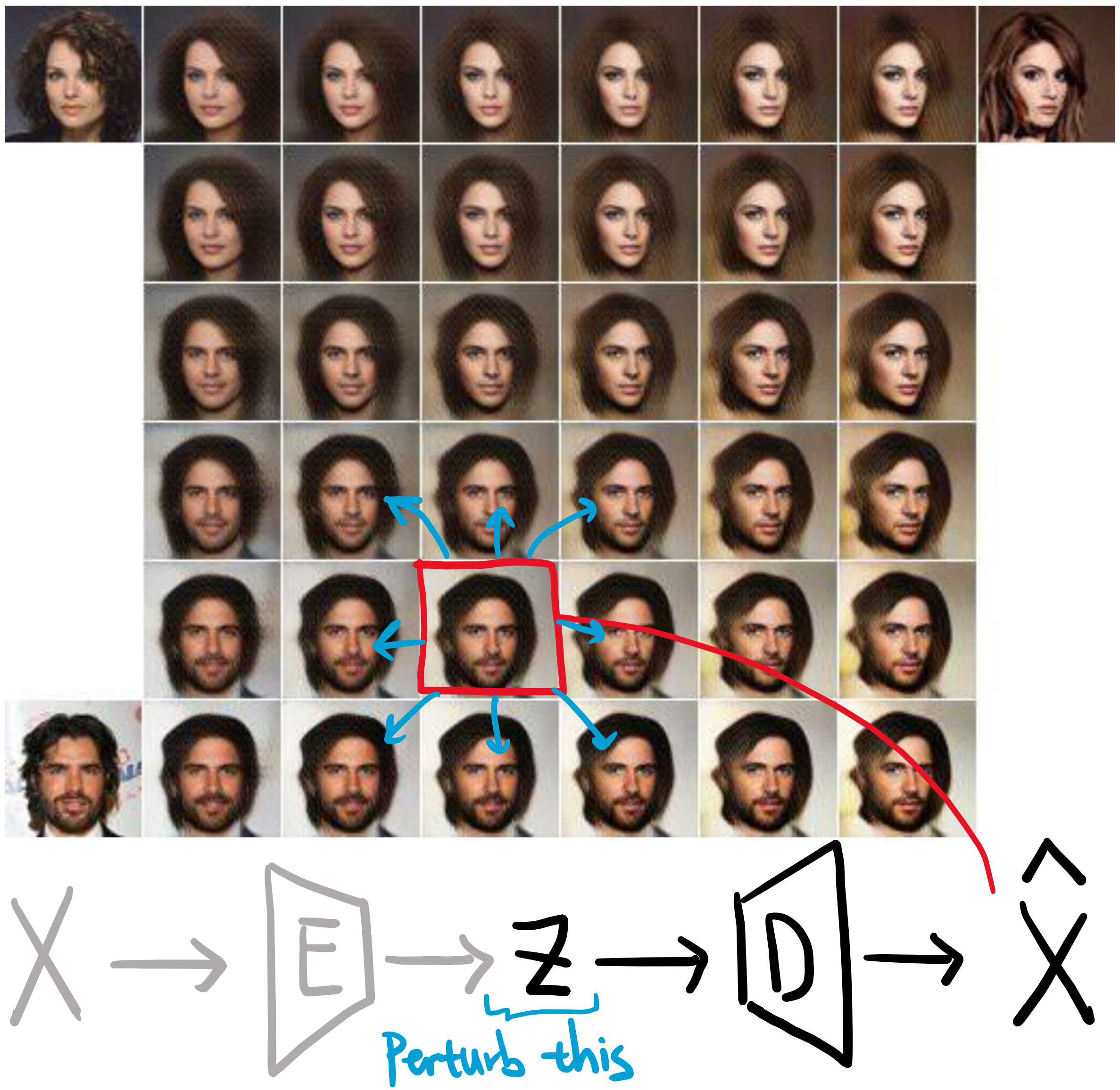
Autoencoders



prediction is input?!

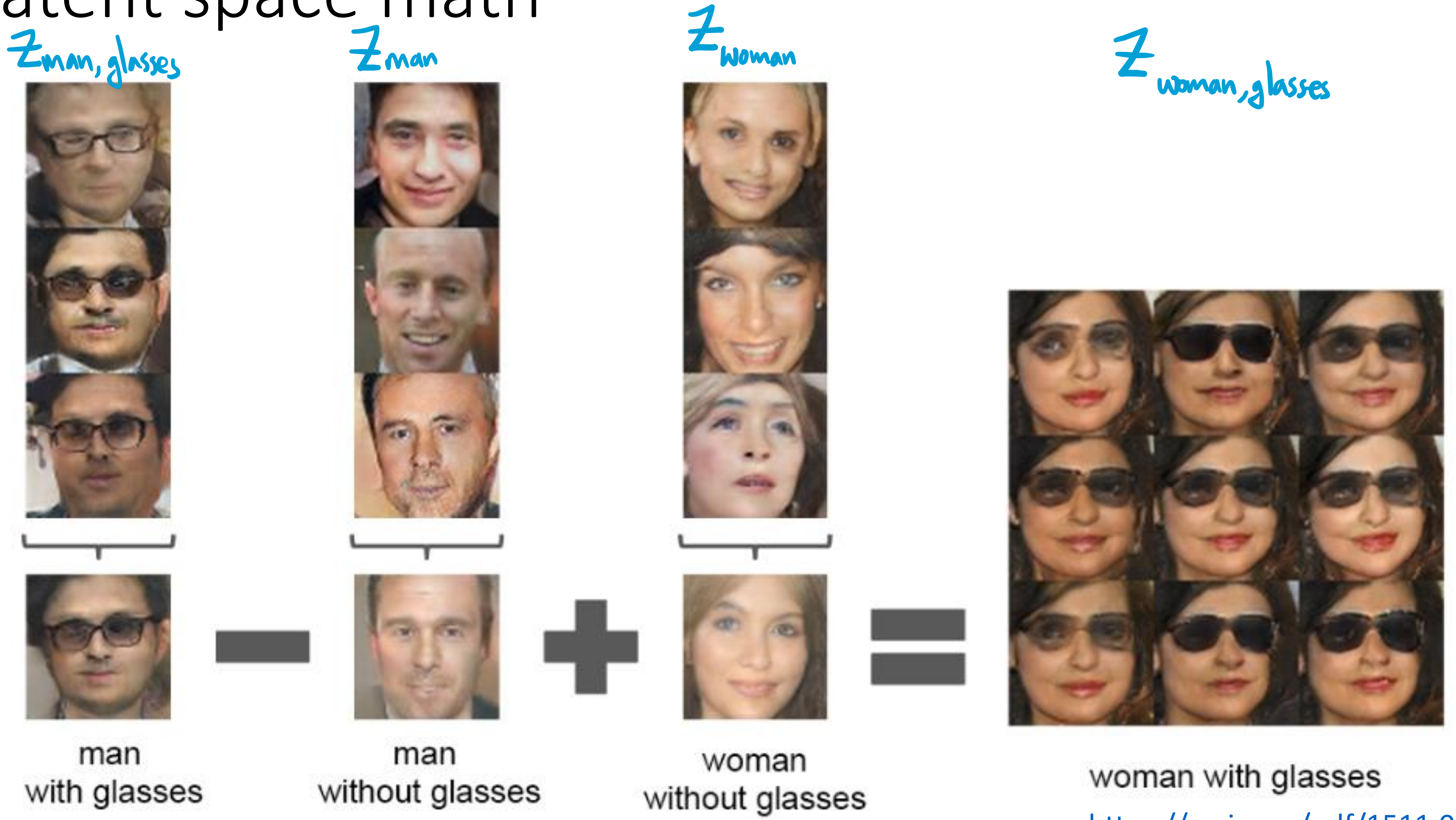
Q: Why would this be useful?

Face Autoencoder's Latent Space



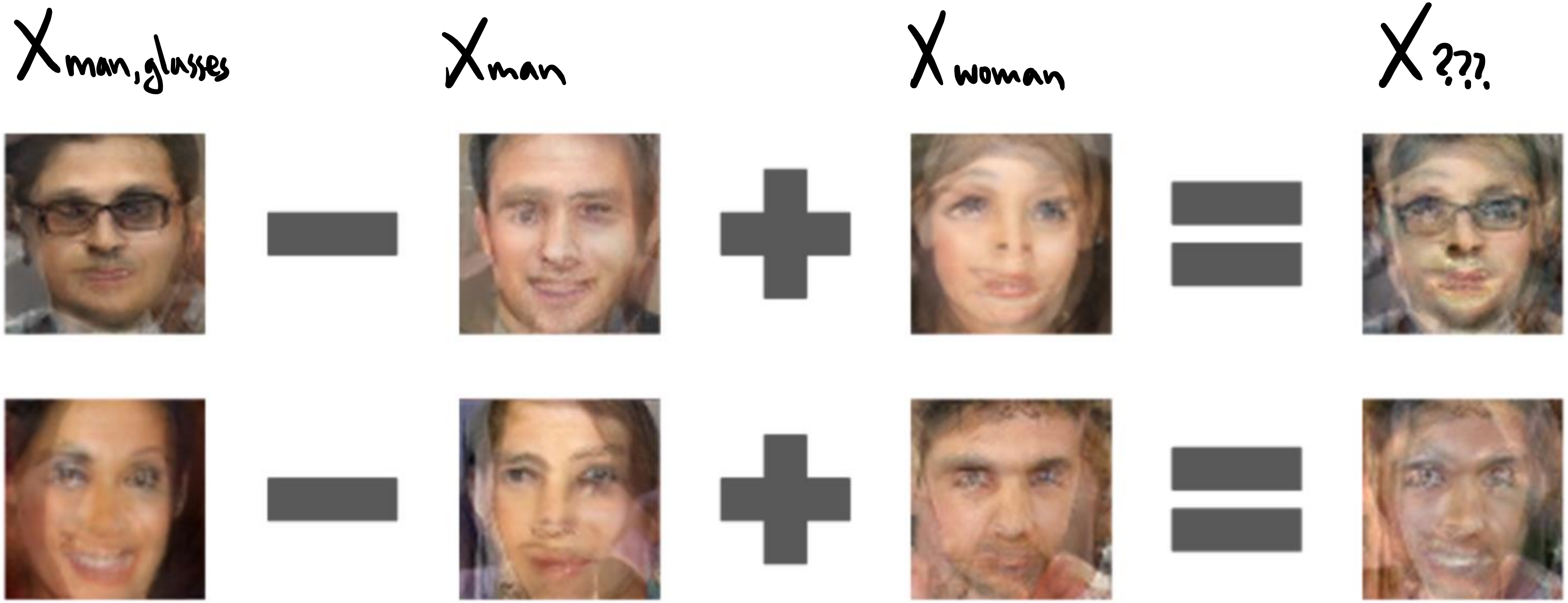
https://medium.com/@juliendespois/latent-space-visualization-deep-learning-bits-2-bd09a46920df

Latent space math



https://arxiv.org/pdf/1511.06434.pdf

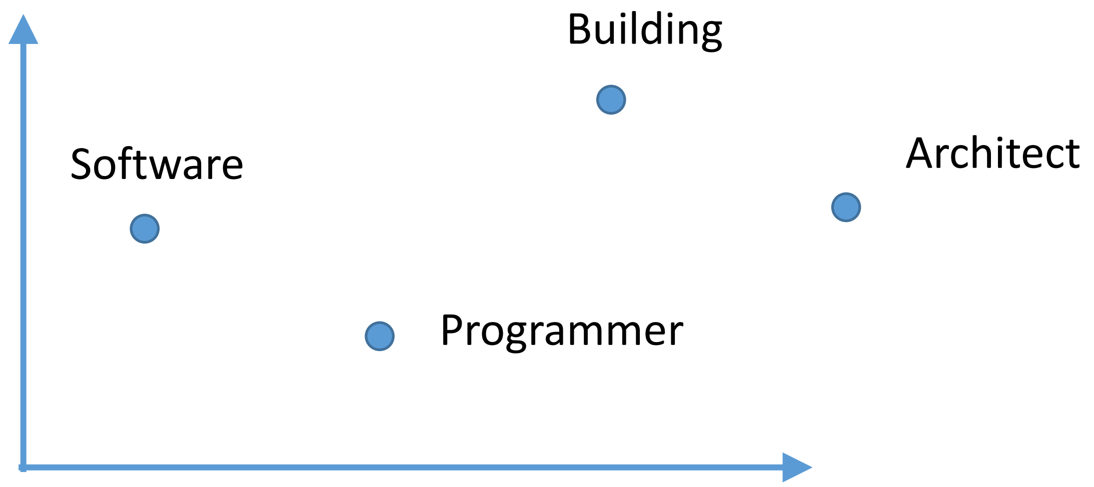
Pixel-space math



Same with words

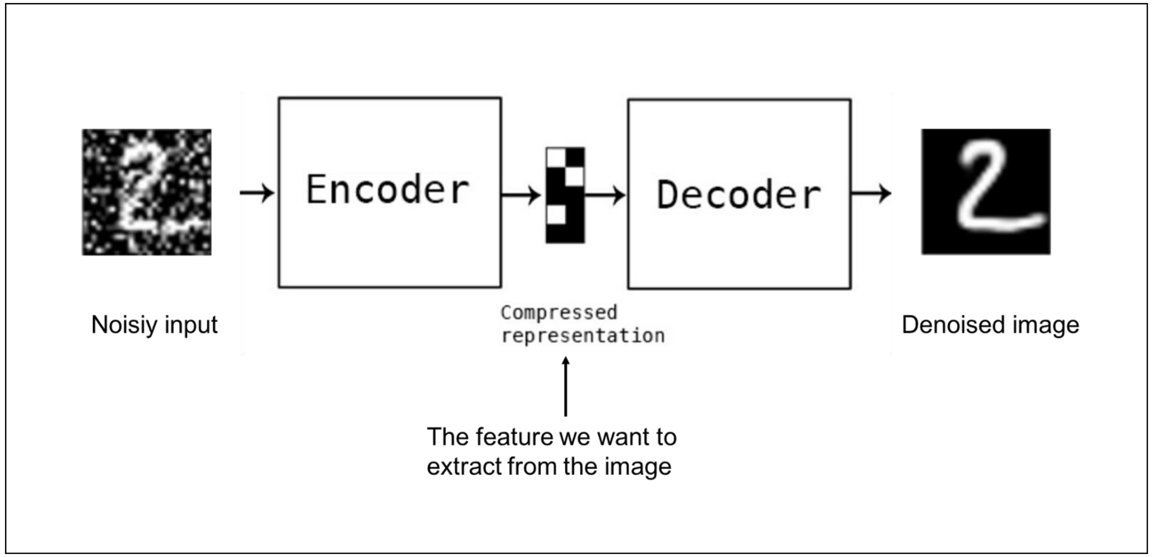
"is to"   "is to"

- King - Man + Woman = Queen
- Note that this is equivalent to King - Man = Queen - Woman
- Software - Building + Architect = Programmer
- Precomputed dictionaries of such encodings are available, e.g., Fasttext (https://fasttext.cc/)



Denoising Autoencoders

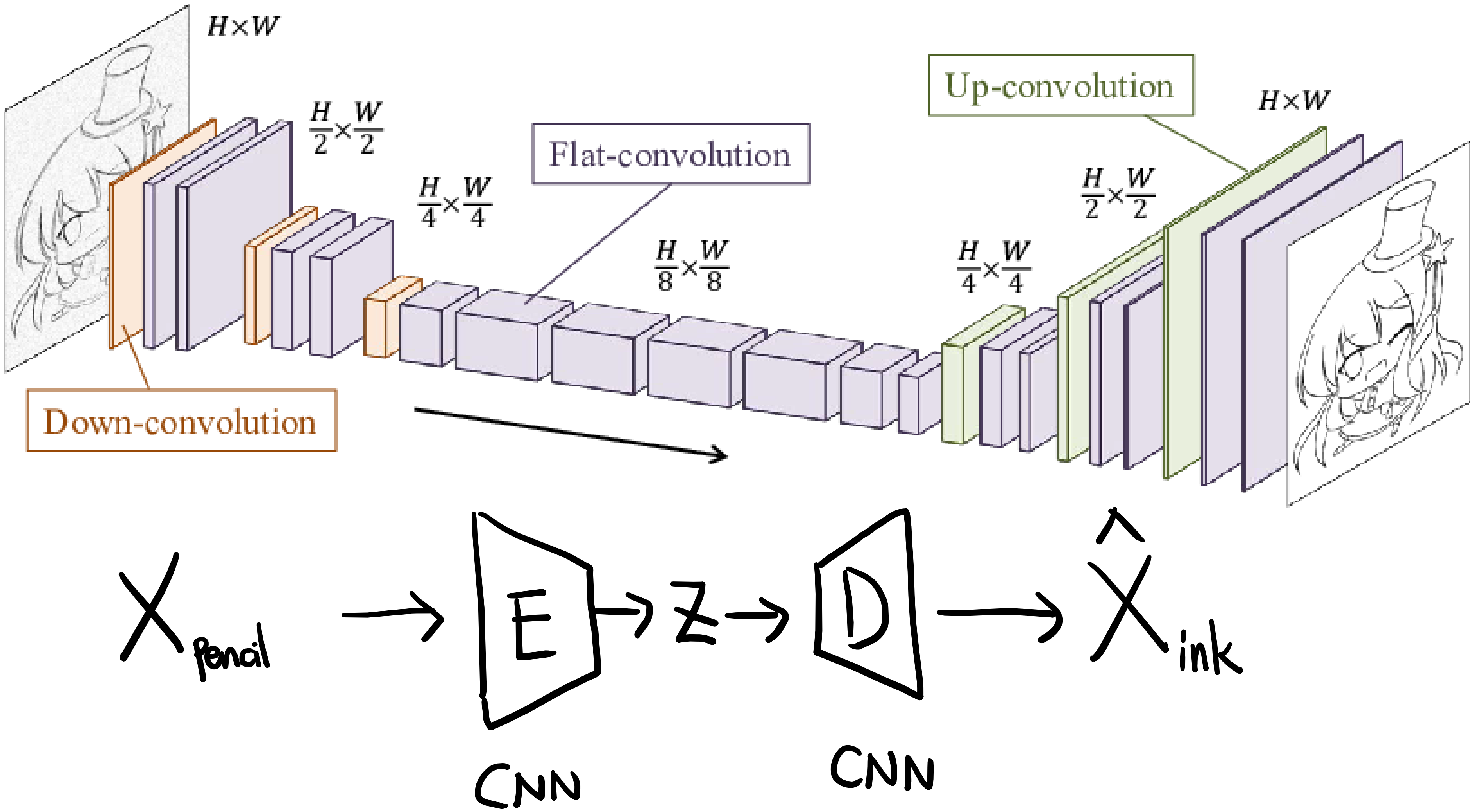
- Neural networks can learn to remove noise from input



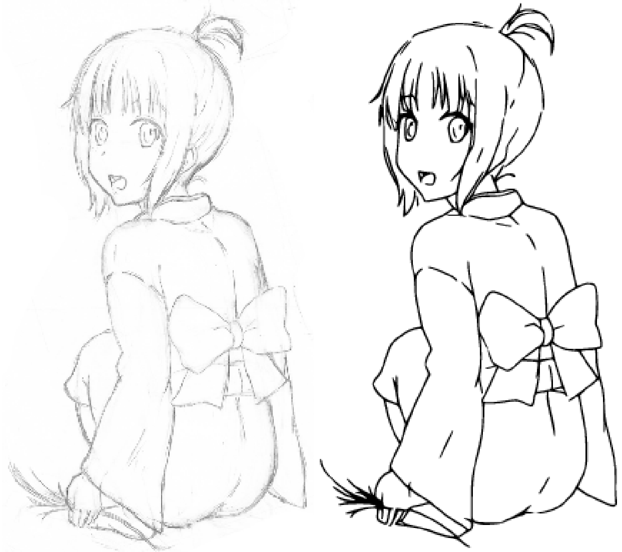
$$X_{clean} \xrightarrow{\text{add noise}} X_{noised} \rightarrow \boxed{E} \rightarrow Z \rightarrow \boxed{D} \rightarrow \hat{X}_{clean}$$

For learning, we need **clean** images, but... what about for inference?

Convolutional Autoencoder



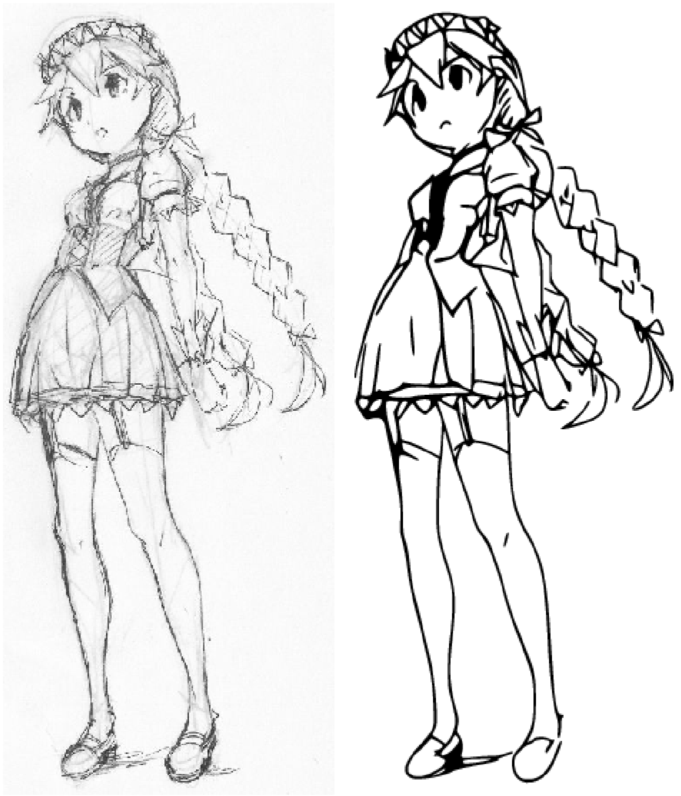
(a) Animals



(b) Kimono

(c) Masks

(d) Book



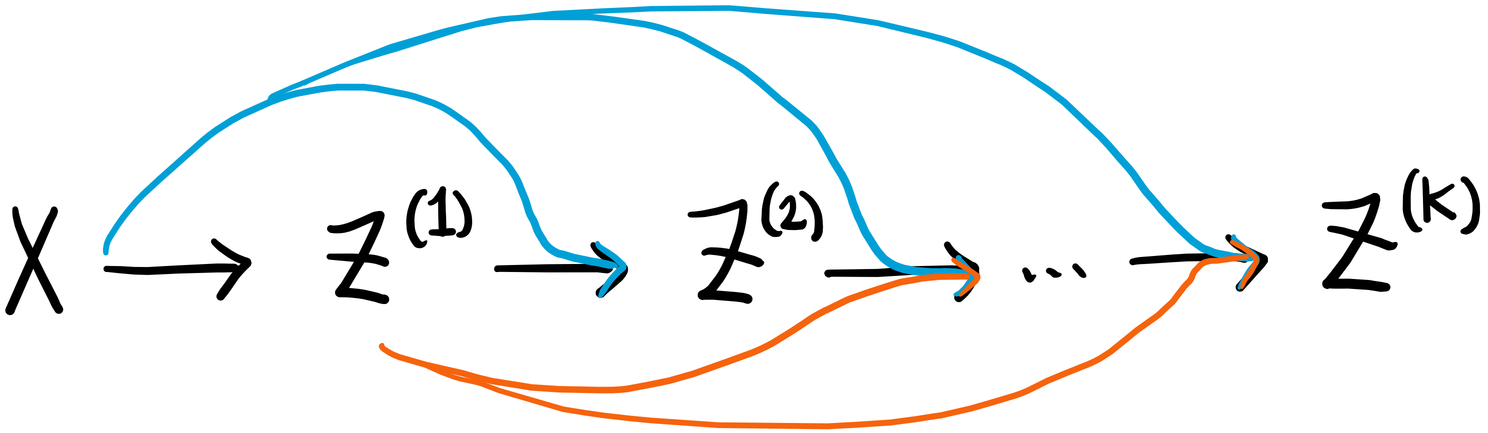
(e) Standing girl

(Pause)

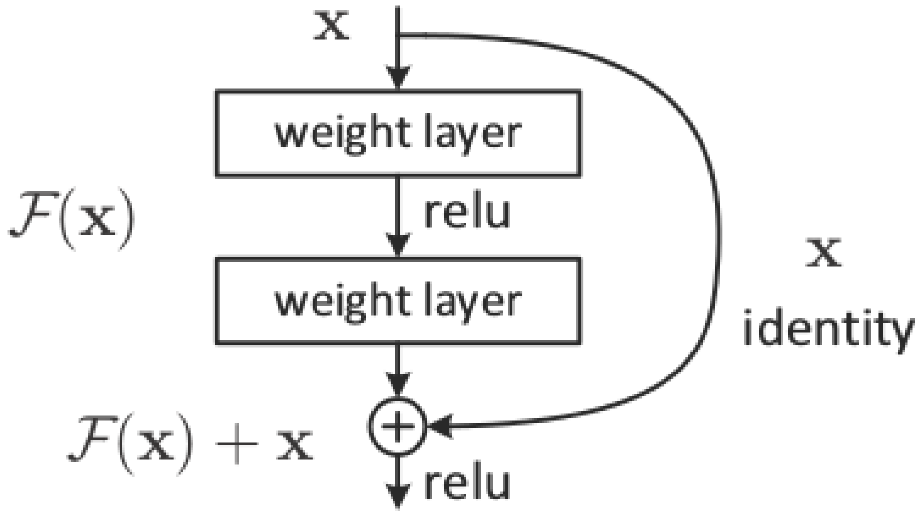
Skip-Connections

$$X \to Z^{(1)} \to Z^{(2)} \to \cdots \to Z^{(k)}$$

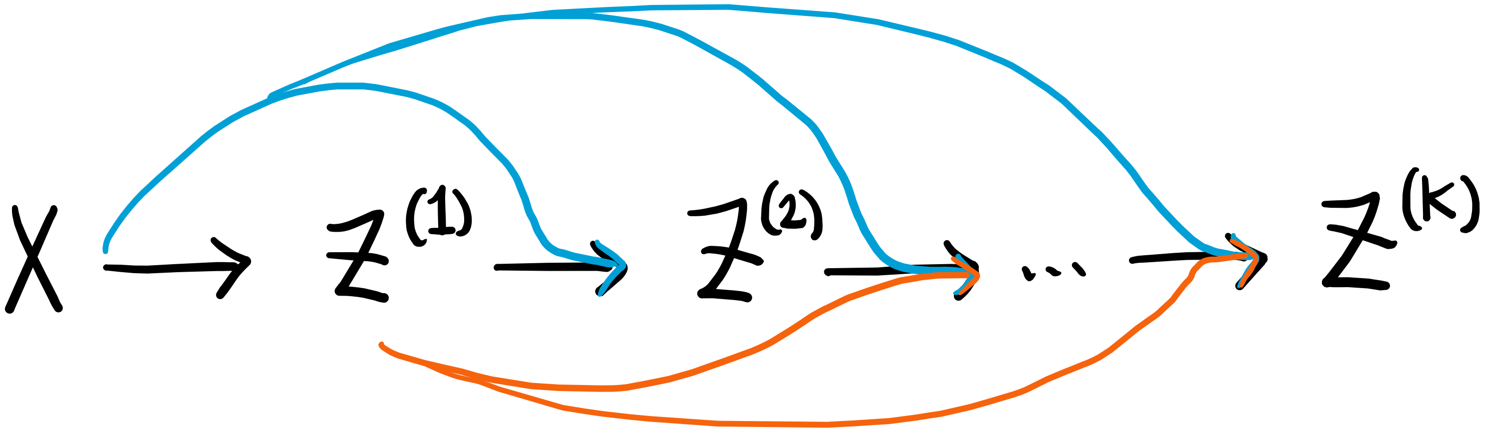
FCN

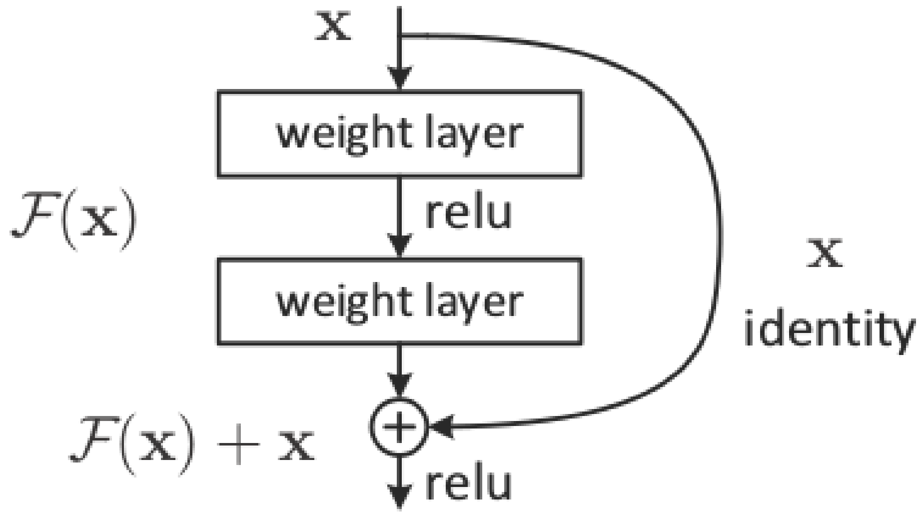


Skip-connections



Skip-Connections





- Black + blue + orange make a latent
- Black arrows are output “residues” after blue and orange
- Main benefit: makes very-deep neural networks (100+ layers) easier to train

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [40] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test *set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.*

The depth of representations is of central importance

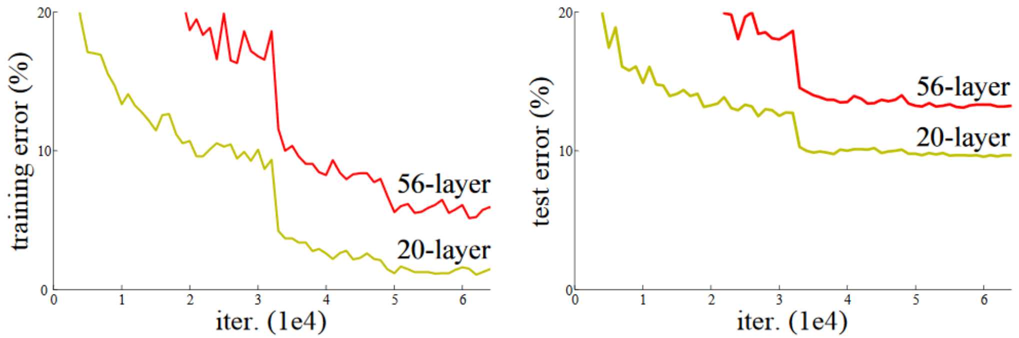
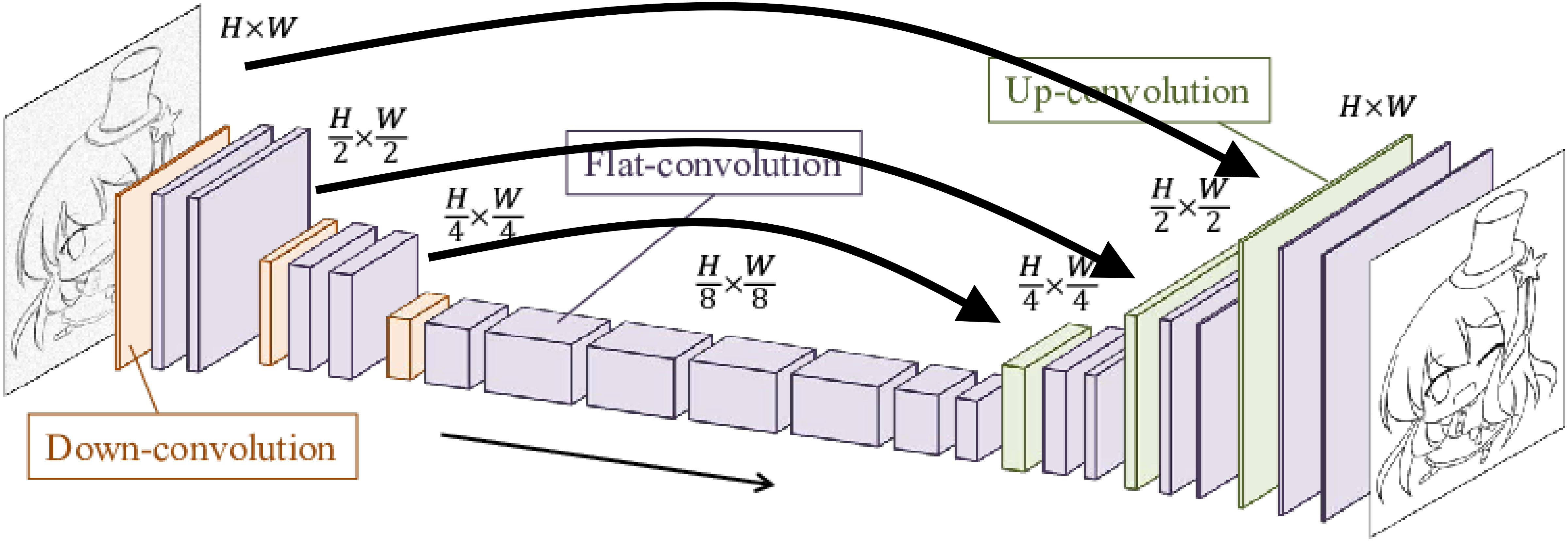


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

greatly benefited from very deep models.

Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious problem of vanishing/exploding gradients [14, 1, 8], which hamper convergence from the beginning. This problem,

CNN + Skip-Connections



Skip-connections and convolutions: U-Net (2015)

U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies,
University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

Abstract. There is large consent that successful training of deep networks requires many thousand annotated training samples. In this paper, we present a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. We show that such a network can be trained end-to-end from very few images and outperforms the prior best method (a sliding-window convolutional network) on the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks. Using the same network trained on transmitted light microscopy images (phase contrast and DIC) we won the ISBI cell tracking challenge 2015 in these categories by a large margin. Moreover, the network is fast. Segmentation of a 512x512 image takes less than a second on a recent GPU. The full implementation (based on Caffe) and the trained networks are available at http://lmb.informatik.uni-freiburg.de/people/ronneber/u-net.

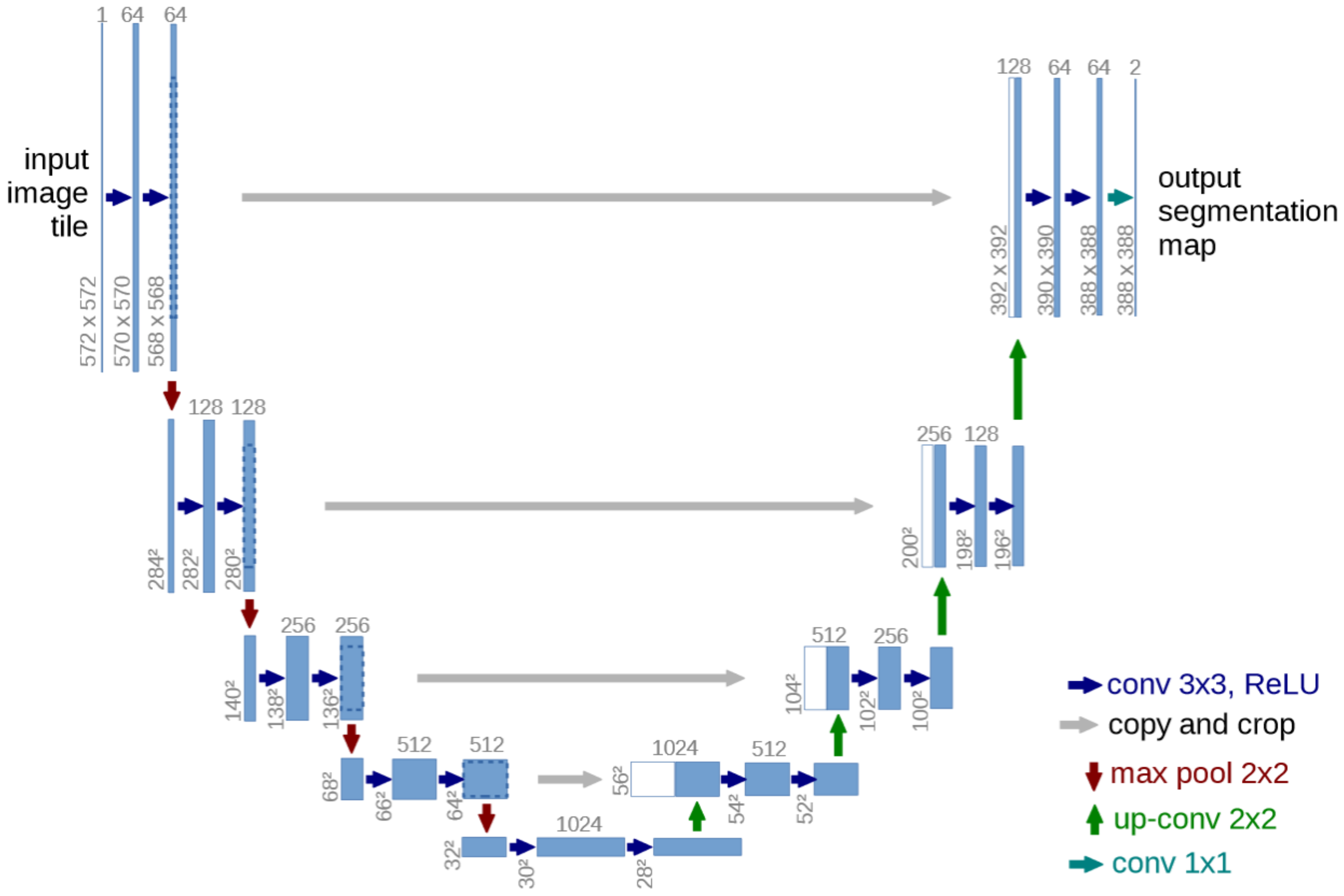


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Last Slide Before Demo 3

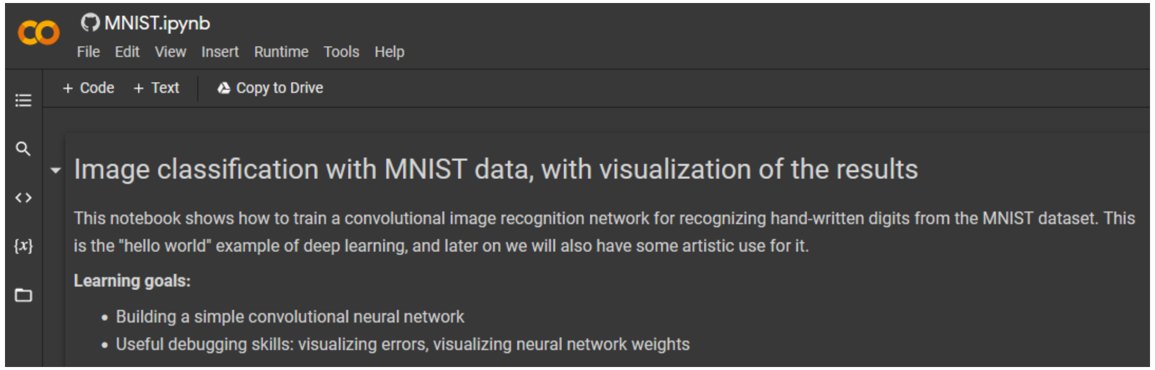
- “ResNet”: family of *very-deep* neural nets with skip-connections. Works best out-of-the-box for many classification tasks.
- “U-Net”: CNN + Skip-Connections. Popular for autoencoders and generative tasks (next lecture).

Live Demo 3:
Denoising CNN Autoencoder

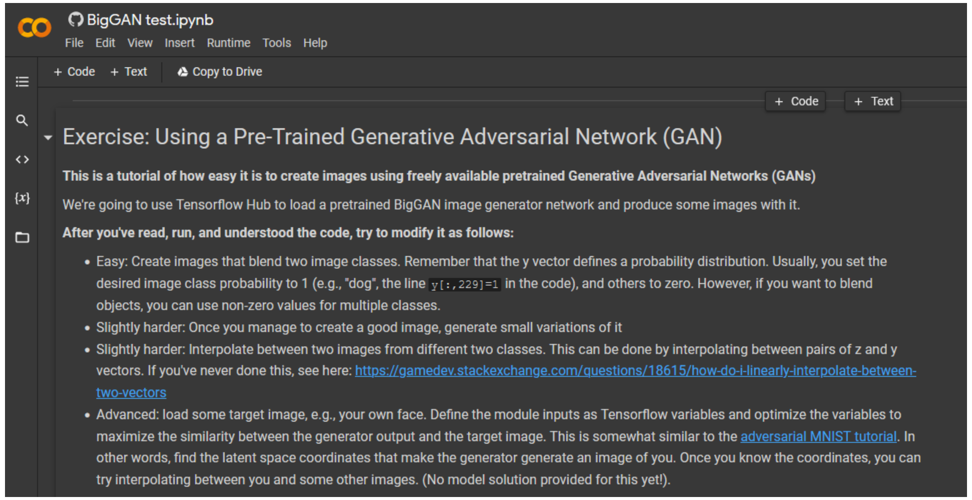
More Bonus Topics

- Normalizations in NN: https://towardsdatascience.com/what-is-group-normalization-45fe27307be7
- Batch/layer/group normalization
- Padding, stride, dilation

Today’s Exercises



Hands-on NN training experience



Inference with pre-trained GAN

Questions?