Deutsches Primatenzentrum
Leibniz-Institut für Primatenforschung

Neuromatch Academy: Deep Learning Executive Summary

James Goodman
NBL
05.10.2021

Mitglied der Leibniz-Gemeinschaft

Quick aside

- I put way too much stuff in these slides
- I wanted to these slides to not just be a presentation aid, but also a reference material
- I also didn't totally understand 100% of the course
- All this means we'll be flying through some slides with minimal explanation

Mitglied der Leibniz-Gemeinschaft

DPZ
Deutsches Primatenzentrum
Leibniz-Institut für Primatenforschung

Background: What was it?

Mitglied der
Leibniz-Gemeinschaft

What was the Neuromatch Academy?

- Two summer schools offered under the Neuromatch banner
  - Computational Neuroscience
  - **Deep Learning**

- Each a 3-week long intensive course
  - Participants separated into "pods" (8+ people per pod)
  - Led through a series of Colab / Jupyter notebooks by a TA
  - Notebooks included a mix of
    - coding exercises
    - video lectures
  - Split into further groups of 3+ for independent projects

What was the Neuromatch Academy?

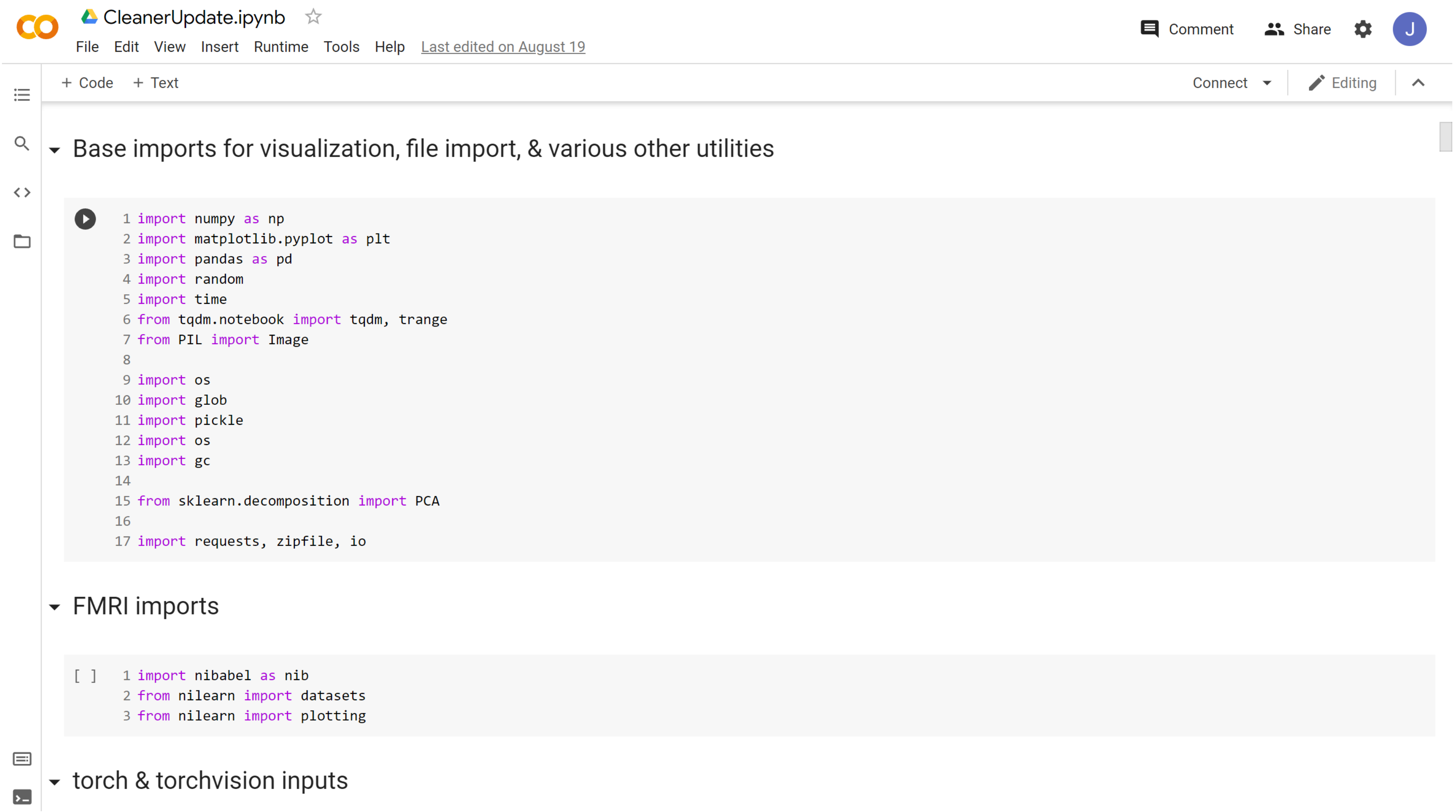
- Deep learning notable lecturers
  - Konrad Kording
  - Alexander Ecker
  - Surya Ganguli
  - Tim Lillicrap

- Participants were mostly
  - students just out of undergrad
  - in the first few years of their Ph.D.s
  - in the first few years of their professional careers
  - Swathi and I were not the only postdocs, though!

Map of concepts covered by the course

Mitglied der
Leibniz-Gemeinschaft

Google Colab: the environment we worked in



Base imports for visualization, file import, & various other utilities

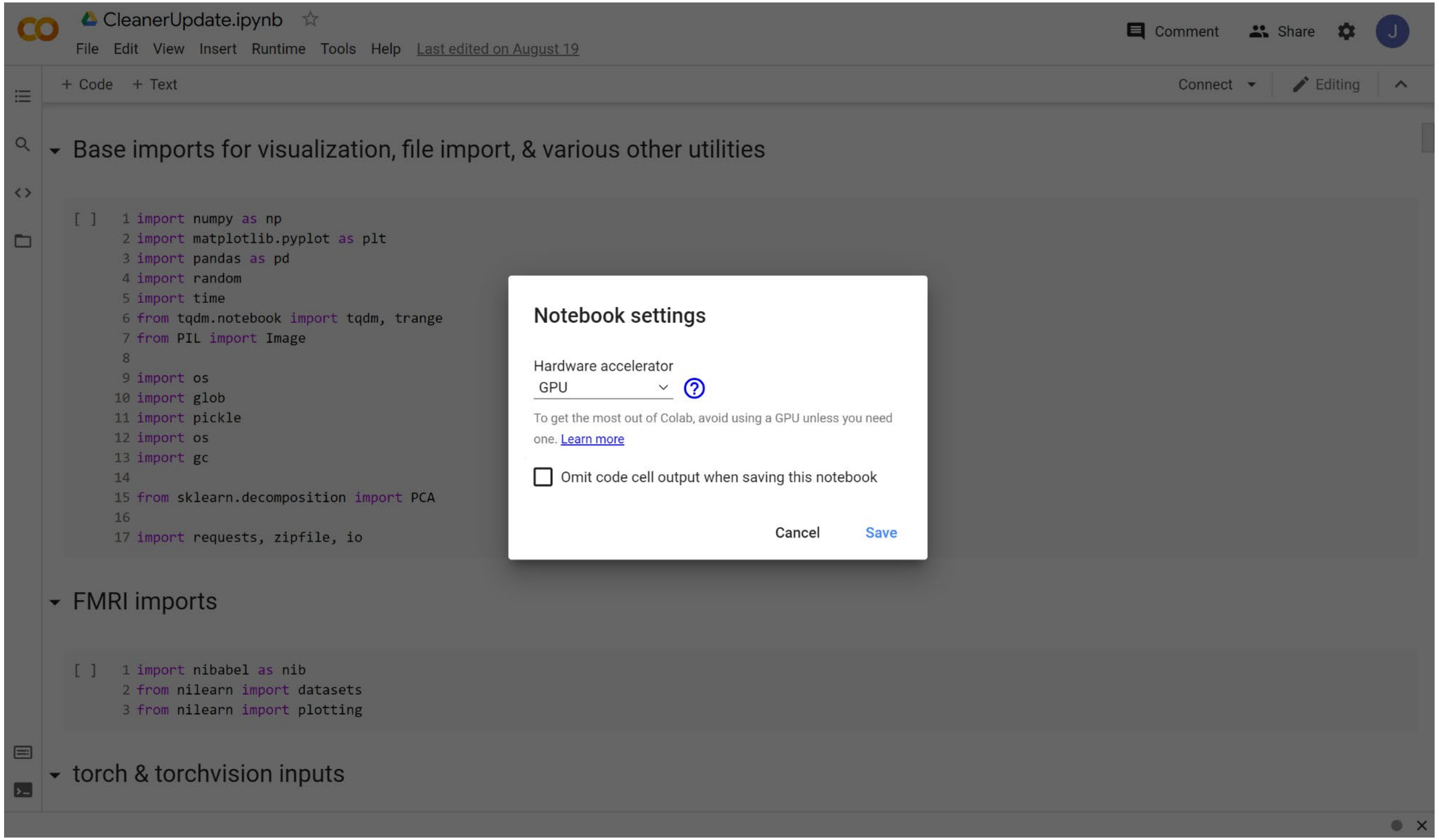
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random
import time
from tqdm.notebook import tqdm, trange
from PIL import Image

import os
import glob
import pickle
import os
import gc

from sklearn.decomposition import PCA

import requests, zipfile, io
```

FMRI imports

```
[ ]
import nibabel as nib
from nilearn import datasets
from nilearn import plotting
```

torch & torchvision inputs

Google Colab: the environment we worked in



Base imports for visualization, file import, & various other utilities

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
     import random
     import time
     from tqdm.notebook import tqdm, trange
     from PIL import Image

     import os
     import glob
     import pickle
     import os
     import gc

     from sklearn.decomposition import PCA

     import requests, zipfile, io
```

Notebook settings

Hardware accelerator: GPU

To get the most out of Colab, avoid using a GPU unless you need one. Learn more

☐ Omit code cell output when saving this notebook

FMRI imports

```
[ ]  import nibabel as nib
     from nilearn import datasets
     from nilearn import plotting
```

torch & torchvision inputs

Alternatives

- Kaggle
- Deepnote (especially collaboration-focused)
- Amazon Web Services / Google Cloud (for bigger jobs)
- Institutional compute resources (e.g., GWDG)
- Jupyter Notebook + Custom machine

- Only Colab offered free GPU access (albeit with tight restrictions)

PyTorch: a framework for deep learning in Python

PyTorch

PyTorch: a framework for deep learning in Python

PyTorch

"Old" standard:

K + TensorFlow

PyTorch Features

- Importable standard models (both pretrained & randomly initialized) (e.g., Alexnet, Resnet)
- Importable standard datasets (e.g., MNIST, ImageNet)
- Community-vetted classes for standard network layers
- Community-vetted classes for standard optimizers
- Community-vetted classes and methods for data loading & minibatching
- Autograd & GPU support
- Documentation!
  - https://pytorch.org/docs/stable/index.html
  - Doesn't quite compare to MATLAB's, but very good given how fast this field is moving

DPZ
Deutsches Primatenzentrum
Leibniz-Institut für Primatenforschung

Week 1: "The Basics"

Mitglied der Leibniz-Gemeinschaft

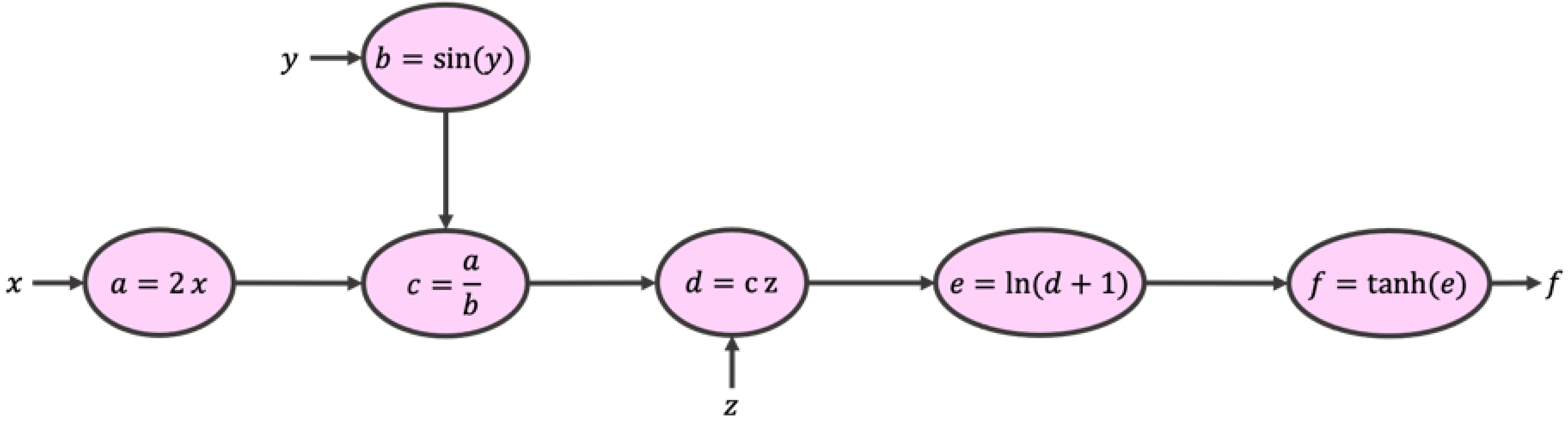
Computational graphs, gradient descent, and backpropagation

$$f(x, y, z) = \tanh\left(\ln\left[1 + z\frac{2x}{\sin(y)}\right]\right)$$

Mitglied der Leibniz-Gemeinschaft

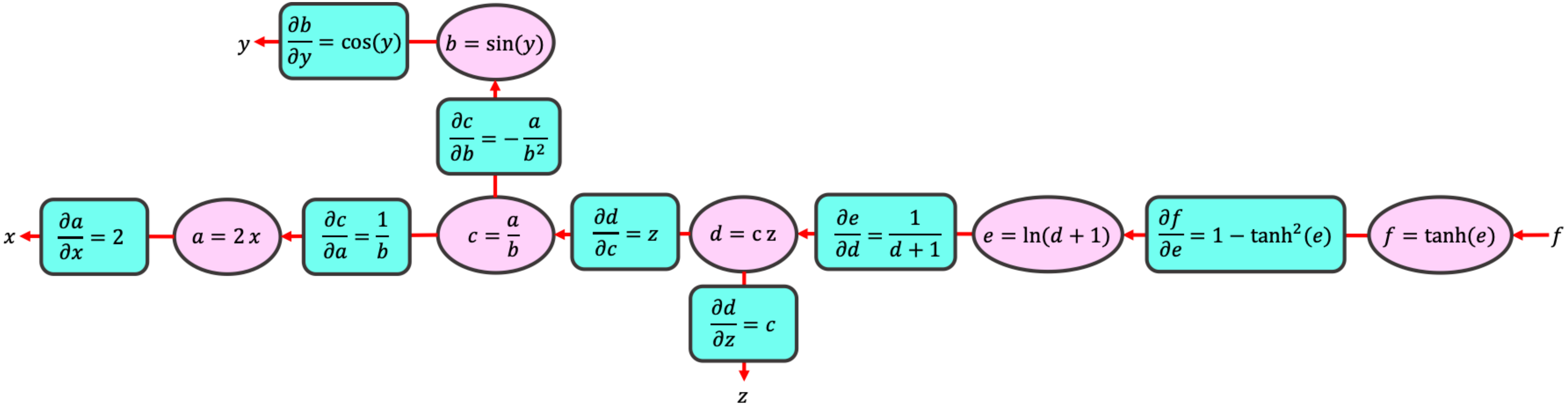
Gradient descent, computational graphs, and backpropagation

$$f(x, y, z) = \tanh\left(\ln\left[1 + z\frac{2x}{\sin(y)}\right]\right)$$



Computational graphs, gradient descent, and backpropagation

$$f(x,y,z) = \tanh\left(\ln\left[1 + z\frac{2x}{\sin(y)}\right]\right)$$



Note on Gradient Descent Optimizers

- Generally can't load entire dataset into memory
- Losses and gradients are therefore usually estimated from **minibatches**
- Optimizers come in two general flavors:
  - Stochastic Gradient Descent (SGD)
    - "Stochastic" because minibatch gradients are noisy estimates of your "true" gradient
    - One hyperparameter: learning rate
  - SGD with bells and whistles
    - Momentum: average over minibatches to get a better gradient estimate (simply called: Momentum)
    - Adaptive learning rate (e.g. RMSprop)
    - These are not mutually exclusive! (e.g. Adam)
    - These all add hyperparameters!

All sorts of knobs to tweak

- Hyperparameter: a value or model decision determined by the researcher or engineer which affects learning, but is not subject to learning

- Typical hyperparameters (which can interact!)
  - Choice of loss function (e.g., MSE, Cross-Entropy)
  - Choice of loss regularization terms and coefficients
  - Choice of model architecture
  - Choice of activation function(s) (e.g., ReLU, tanh, linear)
  - Choice of learning rate
  - Choice of momentum coefficient
  - Choice of (mini)batch size
  - ...and many more!

- Hyperparameter tuning is a bigger part of the process than one might hope...

Cross-validation

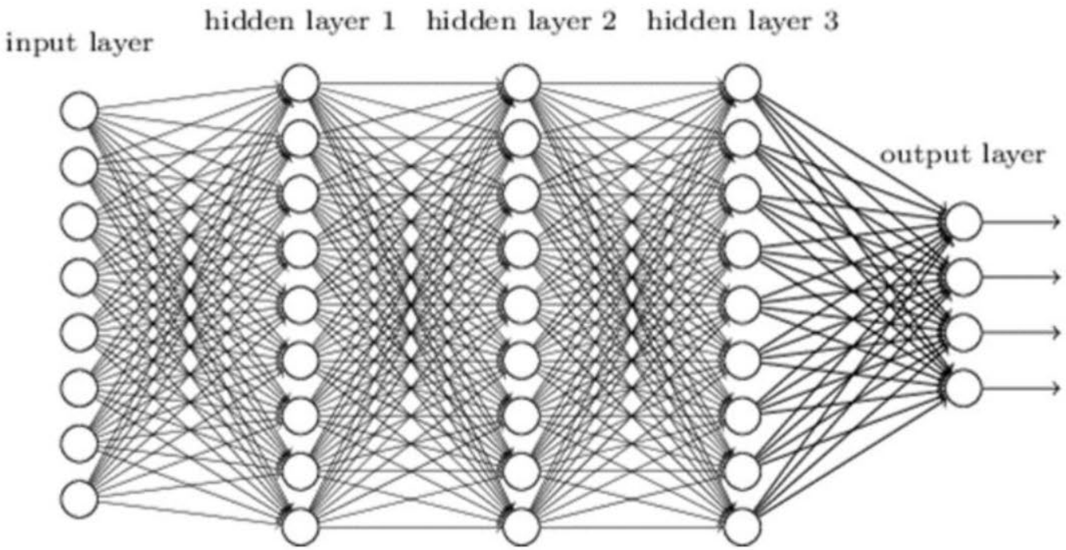
- Typical form of cross-validation: holdout
- Split data into three separate sets
    - Training: defines your parameter gradients
    - Validation: tweak hyperparameters until this looks good
    - Test: stop tweaking, just evaluate performance
- Pytorch offers built-in methods for doing this bookkeeping

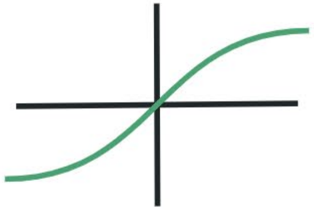
Typical methods for regularization and combating vanishing gradients

- Regularization: constrains models to help them generalize
  - Early stopping (i.e., when validation loss stagnates, halt training)
  - Dropout layers (during each training epoch, randomly fix X% of units to 0 activation)
  - Explicit regularization terms in a loss function (e.g., L2 penalty in ridge regression)

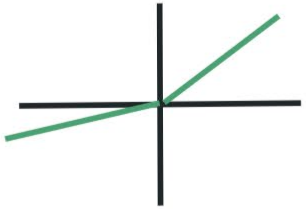
- Combating vanishing gradients: a notorious problem of machine learning
  - Residual blocks (see more in section on ConvNets)
  - Normalization (see more in section on ConvNets)
    - Batch normalization
    - Layer normalization

Our first network: the multilayer perceptron!



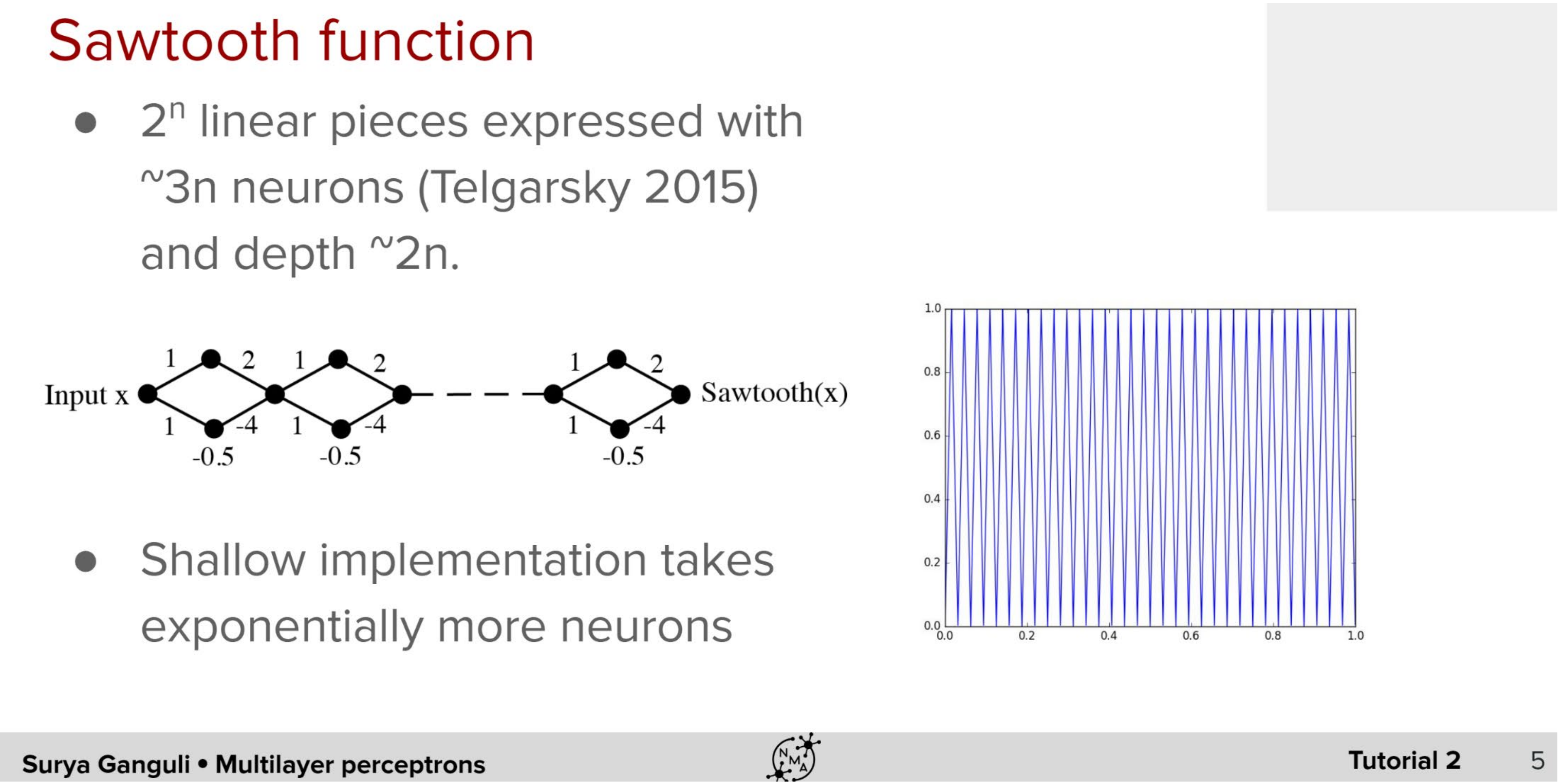


Sigmoid



Leaky ReLU

All nonlinear neural nets are universal approximators, but deeper nets have higher "expressivity"

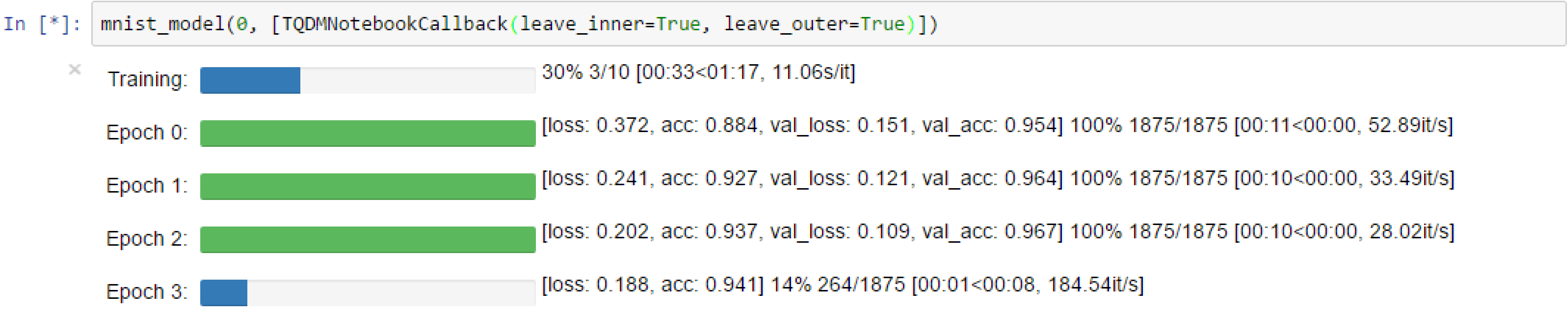


Sawtooth function

- 2^n linear pieces expressed with ~3n neurons (Telgarsky 2015) and depth ~2n.
- Shallow implementation takes exponentially more neurons

Surya Ganguli • Multilayer perceptrons **Tutorial 2** 5

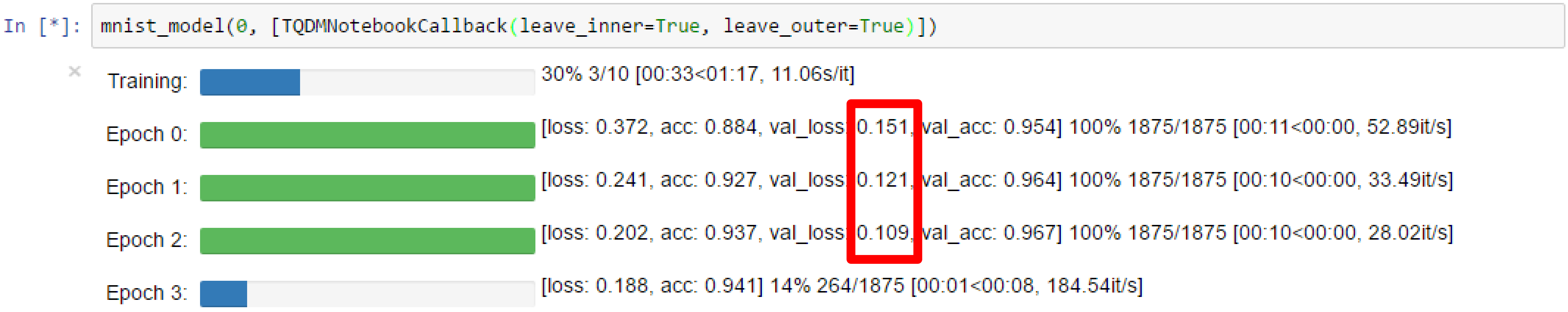
POV: you finally got a deep learning model to run



```
In [*]: mnist_model(0, [TQDMNotebookCallback(leave_inner=True, leave_outer=True)])
```

```
Training: 30% 3/10 [00:33<01:17, 11.06s/it]
Epoch 0: [loss: 0.372, acc: 0.884, val_loss: 0.151, val_acc: 0.954] 100% 1875/1875 [00:11<00:00, 52.89it/s]
Epoch 1: [loss: 0.241, acc: 0.927, val_loss: 0.121, val_acc: 0.964] 100% 1875/1875 [00:10<00:00, 33.49it/s]
Epoch 2: [loss: 0.202, acc: 0.937, val_loss: 0.109, val_acc: 0.967] 100% 1875/1875 [00:10<00:00, 28.02it/s]
Epoch 3: [loss: 0.188, acc: 0.941] 14% 264/1875 [00:01<00:08, 184.54it/s]
```

POV: you finally got a deep learning model to run



```
In [*]: mnist_model(0, [TQDMNotebookCallback(leave_inner=True, leave_outer=True)])

Training: 30% 3/10 [00:33<01:17, 11.06s/it]
Epoch 0:  [loss: 0.372, acc: 0.884, val_loss: 0.151, val_acc: 0.954] 100% 1875/1875 [00:11<00:00, 52.89it/s]
Epoch 1:  [loss: 0.241, acc: 0.927, val_loss: 0.121, val_acc: 0.964] 100% 1875/1875 [00:10<00:00, 33.49it/s]
Epoch 2:  [loss: 0.202, acc: 0.937, val_loss: 0.109, val_acc: 0.967] 100% 1875/1875 [00:10<00:00, 28.02it/s]
Epoch 3:  [loss: 0.188, acc: 0.941] 14% 264/1875 [00:01<00:08, 184.54it/s]
```

Validation loss:

- Not NaN
- Decreasing across epochs
- This is the dream

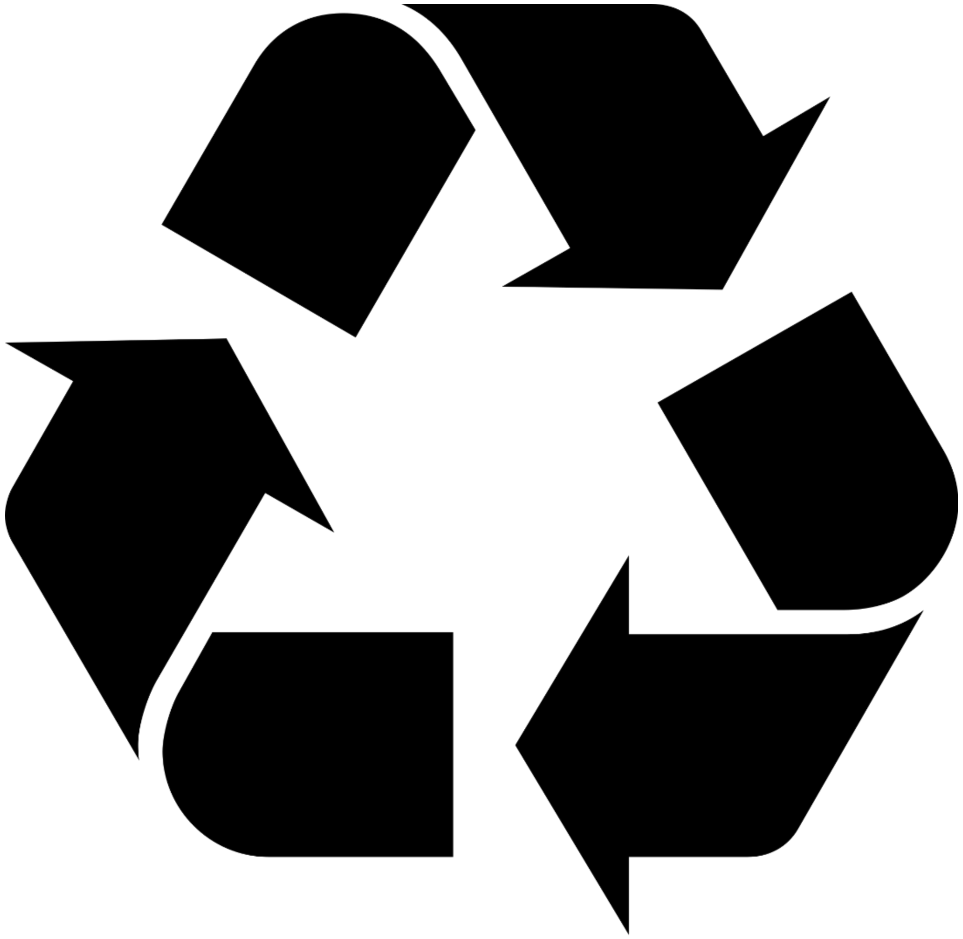
Note: A training "epoch" is a set of minibatches which uses each sample of the training set once

DPZ
Deutsches Primatenzentrum
Leibniz-Institut für Primatenforschung

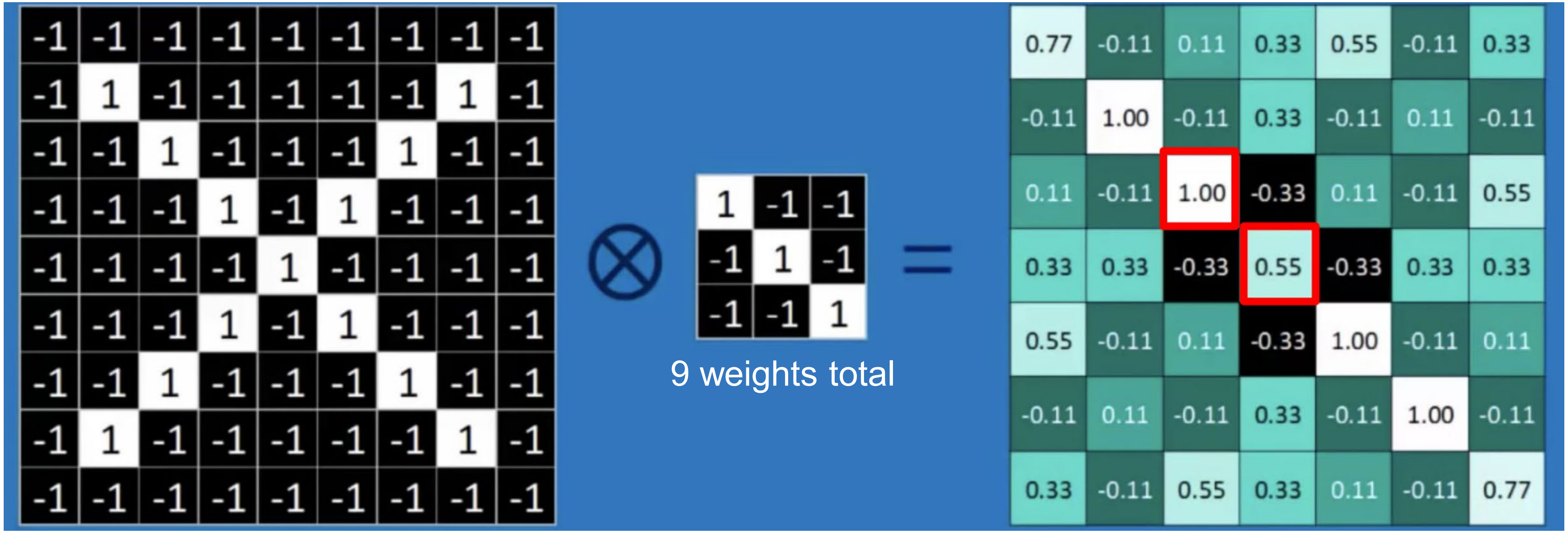
Week 2: Doing more with fewer parameters

Mitglied der
Leibniz-Gemeinschaft

MLPs are too dense, let's recycle some parameters



The convolutional network (ConvNet) saves parameters by recycling them across space



-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

⊗

1	-1	-1
-1	1	-1
-1	-1	1

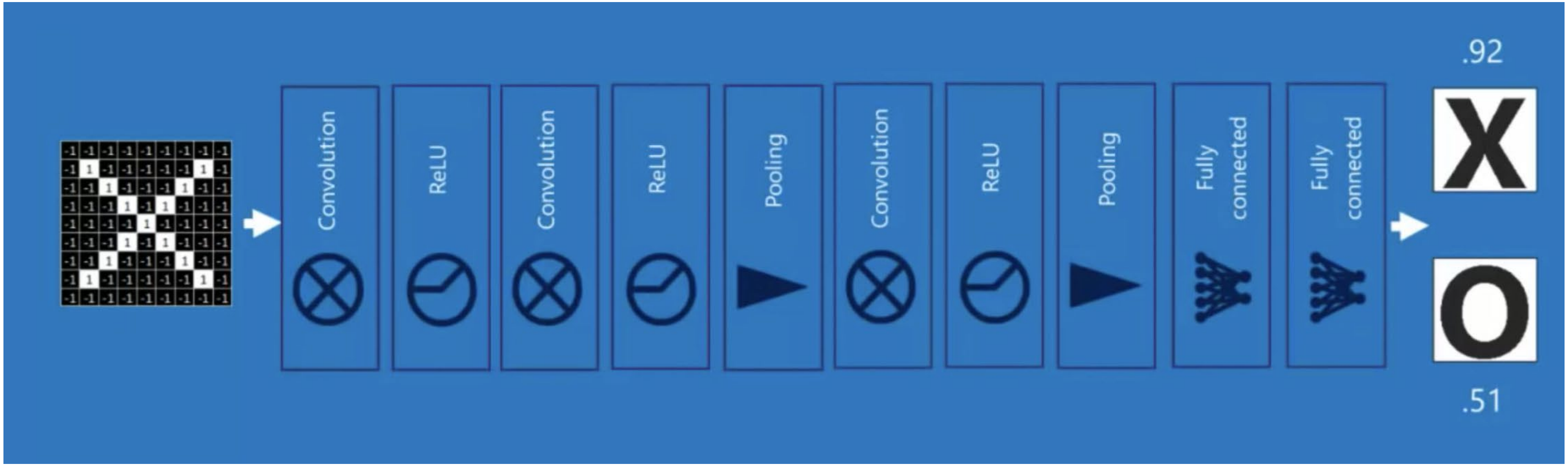
9 weights total

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

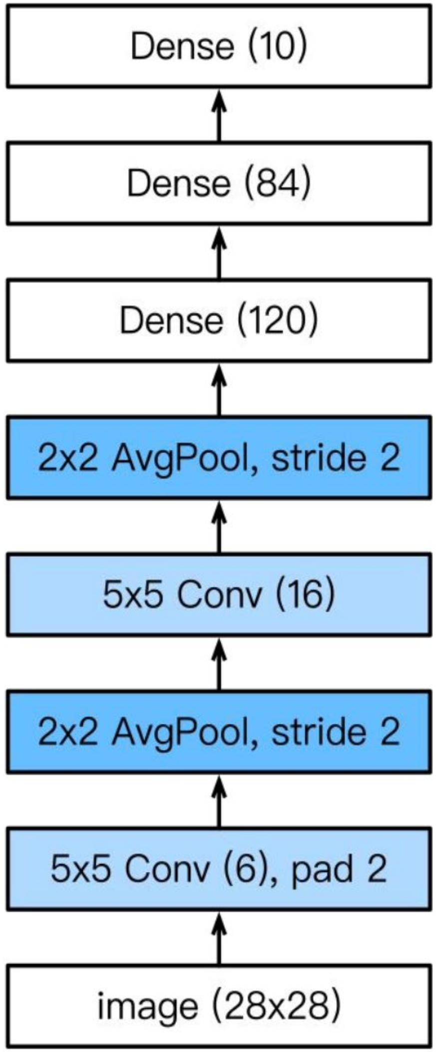
MLP would require 3969 weights

Typical ConvNet pipeline



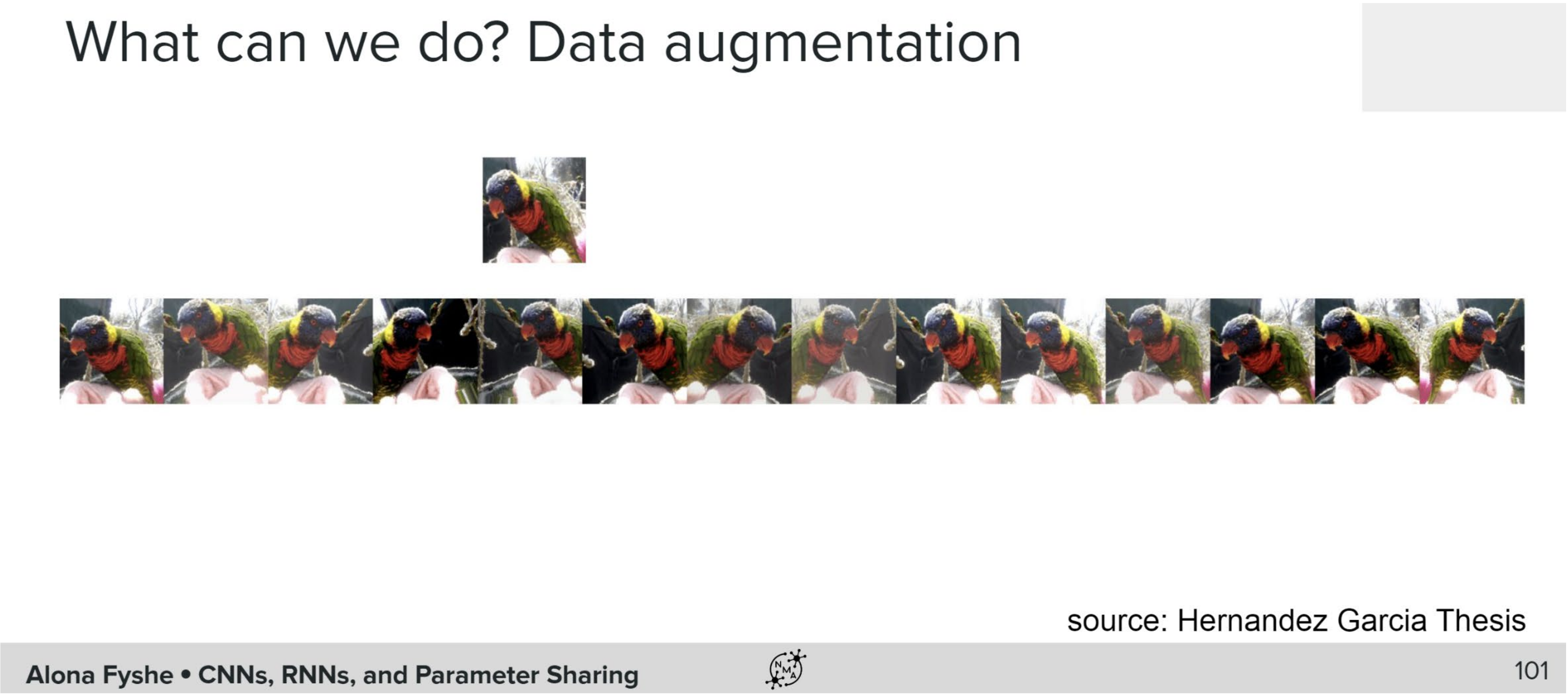
Mitglied der Leibniz-Gemeinschaft

A more typical visualization of the typical ConvNet pipeline



(slides from: Alona Fyshe)

Training ConvNets for computer vision requires a "data augmentation" step



What can we do? Data augmentation

source: Hernandez Garcia Thesis

Alona Fyshe • CNNs, RNNs, and Parameter Sharing

101

Adding normalization (batch or otherwise) combats the problem of too-small or too-large gradients

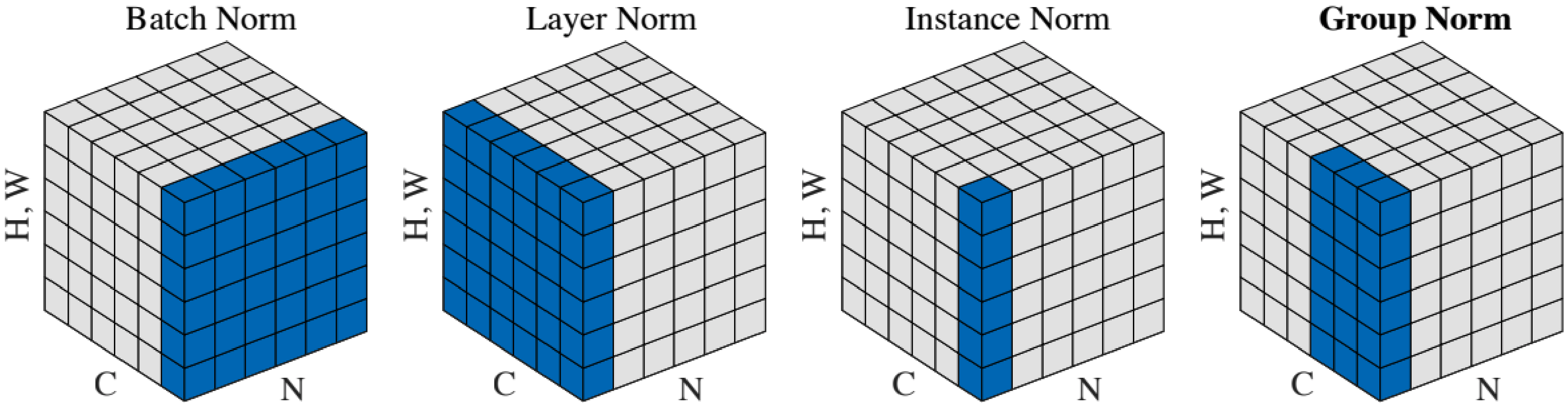
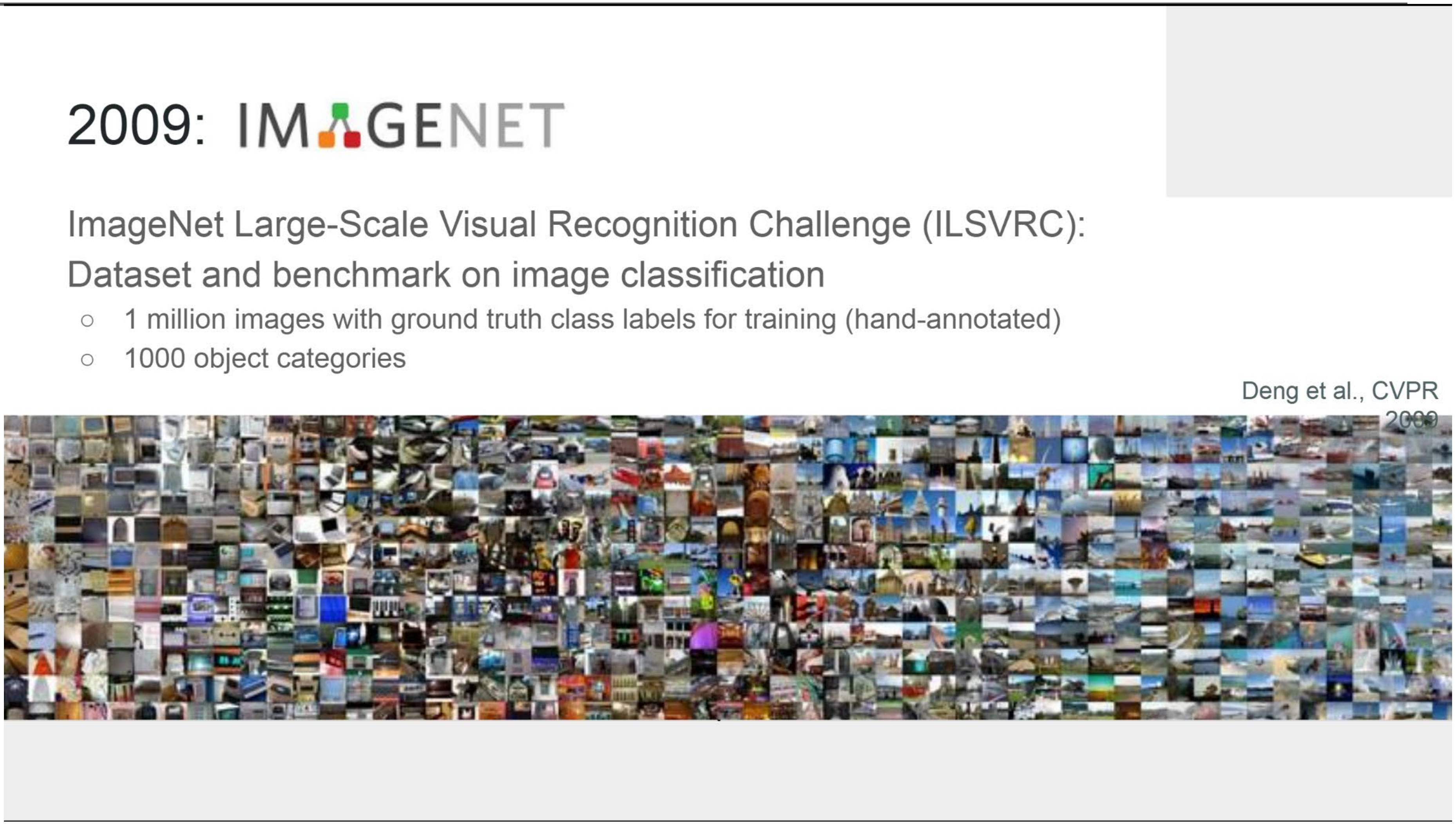


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

Wu, Y. and He, K., 2018. Group normalization. arXiv preprint arXiv: 1803.08494.

A brief history of ImageNet and the Convnets that solved it



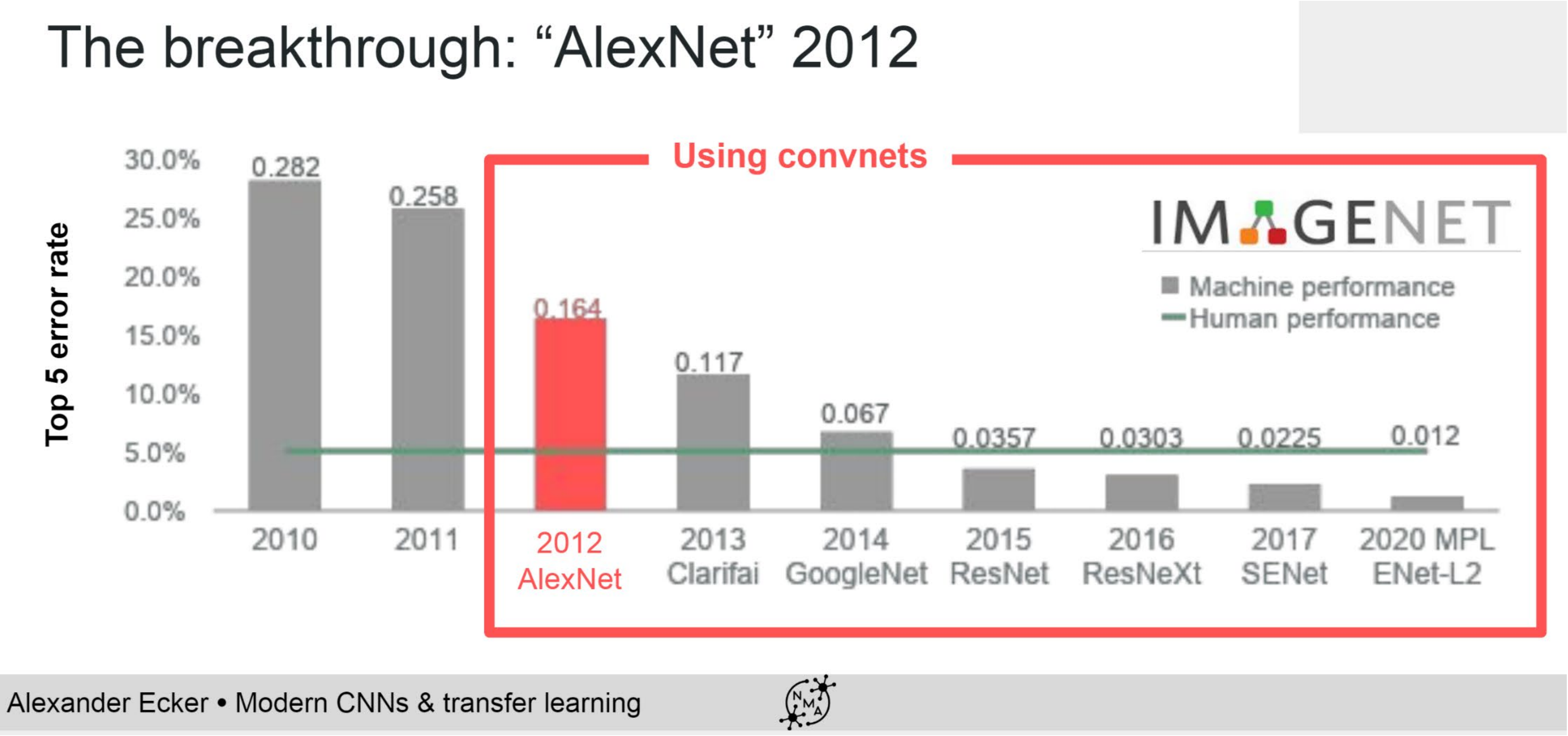
2009: IMAGENET

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC):
Dataset and benchmark on image classification

- 1 million images with ground truth class labels for training (hand-annotated)
- 1000 object categories

Deng et al., CVPR 2009

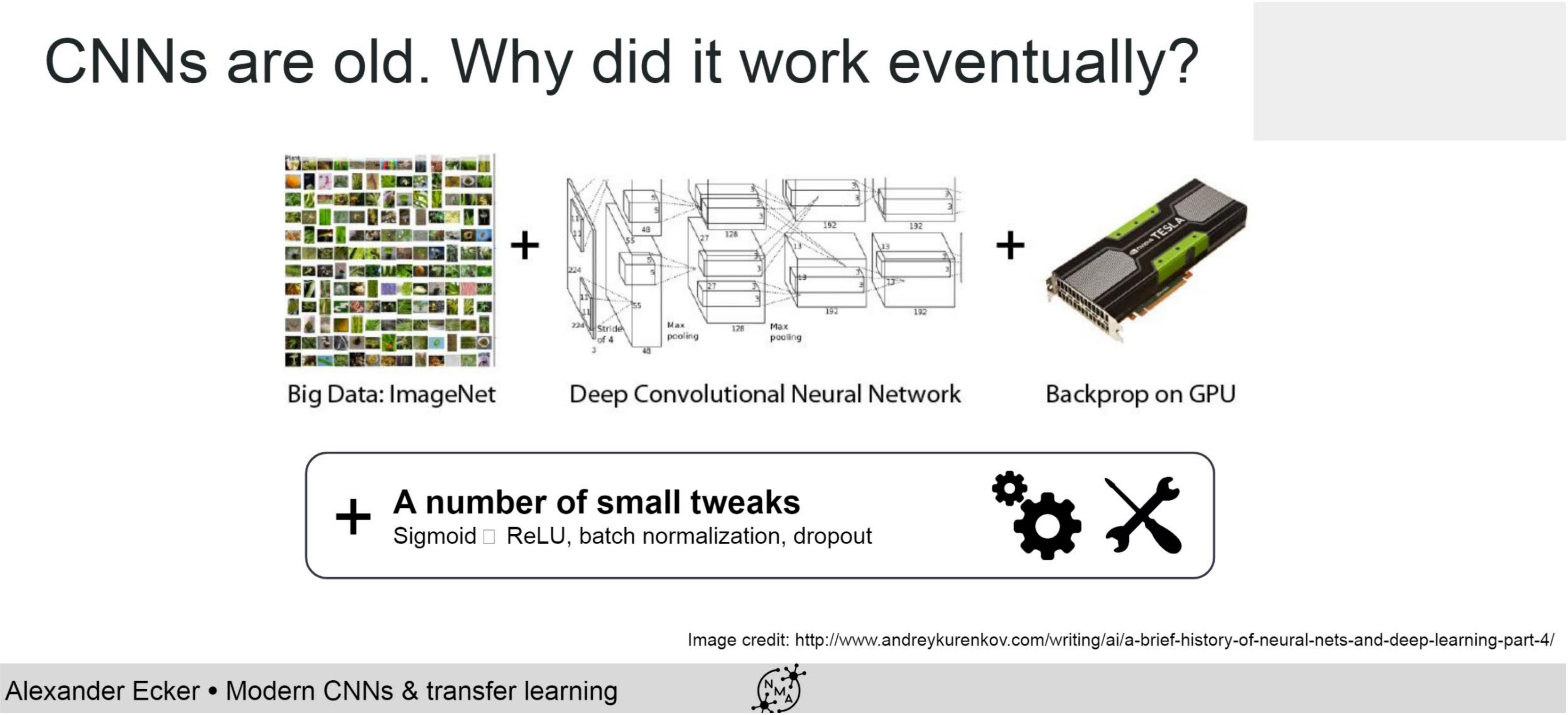
CNNs take over from 2012



The breakthrough: “AlexNet” 2012

Alexander Ecker • Modern CNNs & transfer learning

Old ideas meet new technology (and clever "hacks" to enable training on 2 GPUs at once)



CNNs are old. Why did it work eventually?

Big Data: ImageNet + Deep Convolutional Neural Network + Backprop on GPU

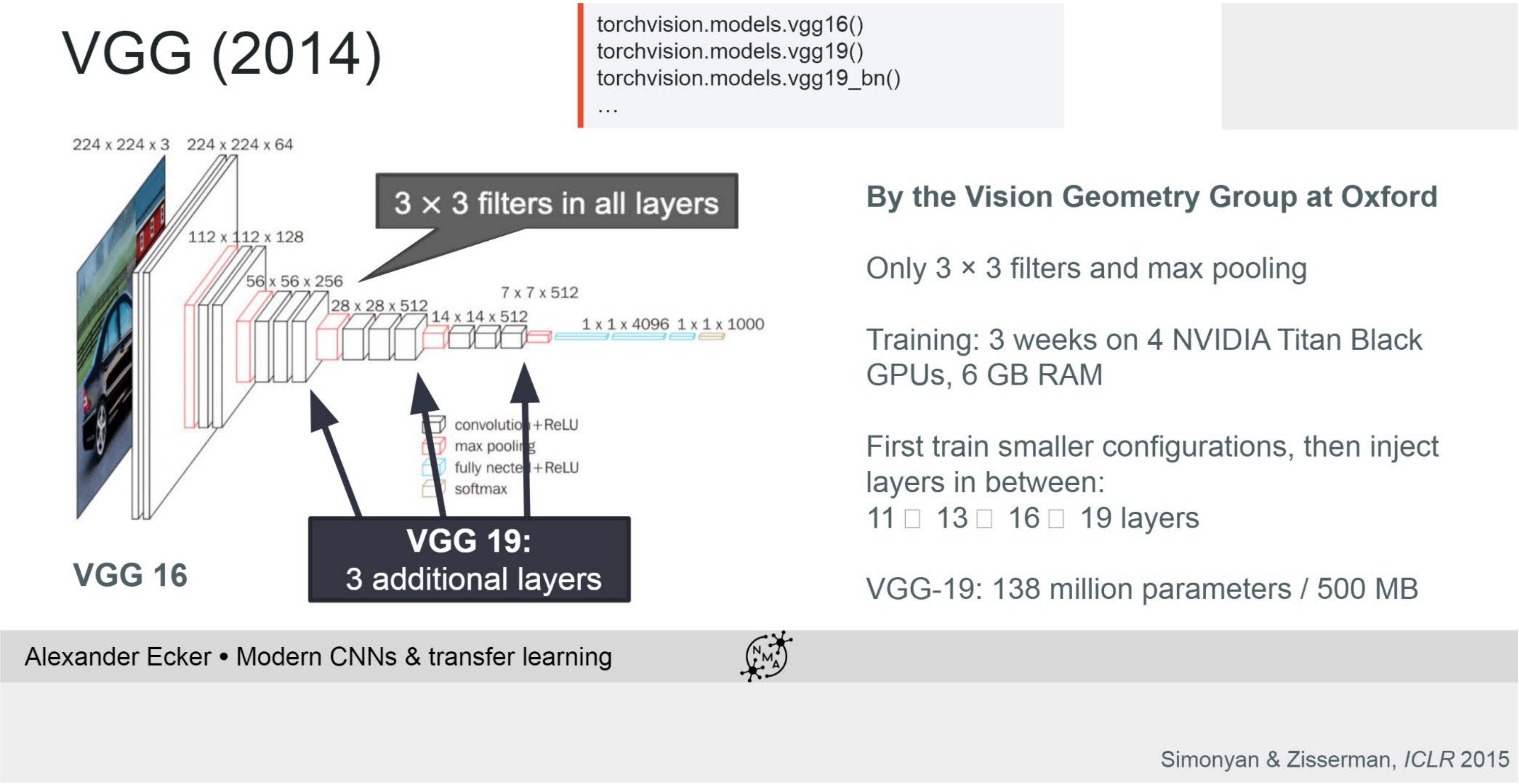
+ A number of small tweaks

Sigmoid □ ReLU, batch normalization, dropout

Image credit: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning-part-4/

Alexander Ecker • Modern CNNs & transfer learning

VGG: inefficient, but popular and a triumph of model sharing



VGG (2014)

```
torchvision.models.vgg16()
torchvision.models.vgg19()
torchvision.models.vgg19_bn()
…
```

3 × 3 filters in all layers

VGG 19:
3 additional layers

VGG 16

By the Vision Geometry Group at Oxford

Only 3 × 3 filters and max pooling

Training: 3 weeks on 4 NVIDIA Titan Black GPUs, 6 GB RAM

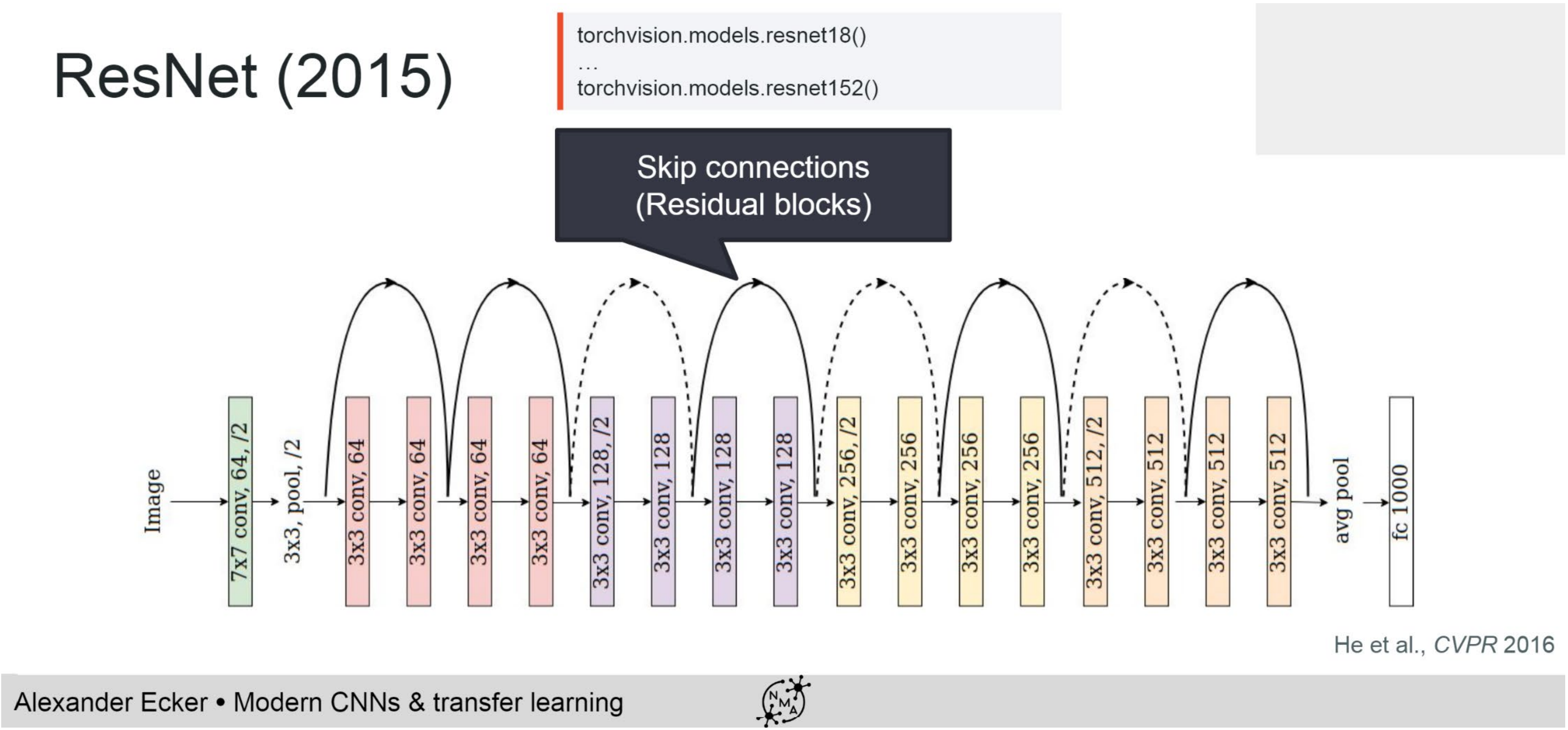
First train smaller configurations, then inject layers in between:
11 □ 13 □ 16 □ 19 layers

VGG-19: 138 million parameters / 500 MB

Alexander Ecker • Modern CNNs & transfer learning

Simonyan & Zisserman, *ICLR* 2015

Resnet: residual blocks (skip connections) enable a gradient superhighway, combating the vanishing gradient problem



ResNet (2015)

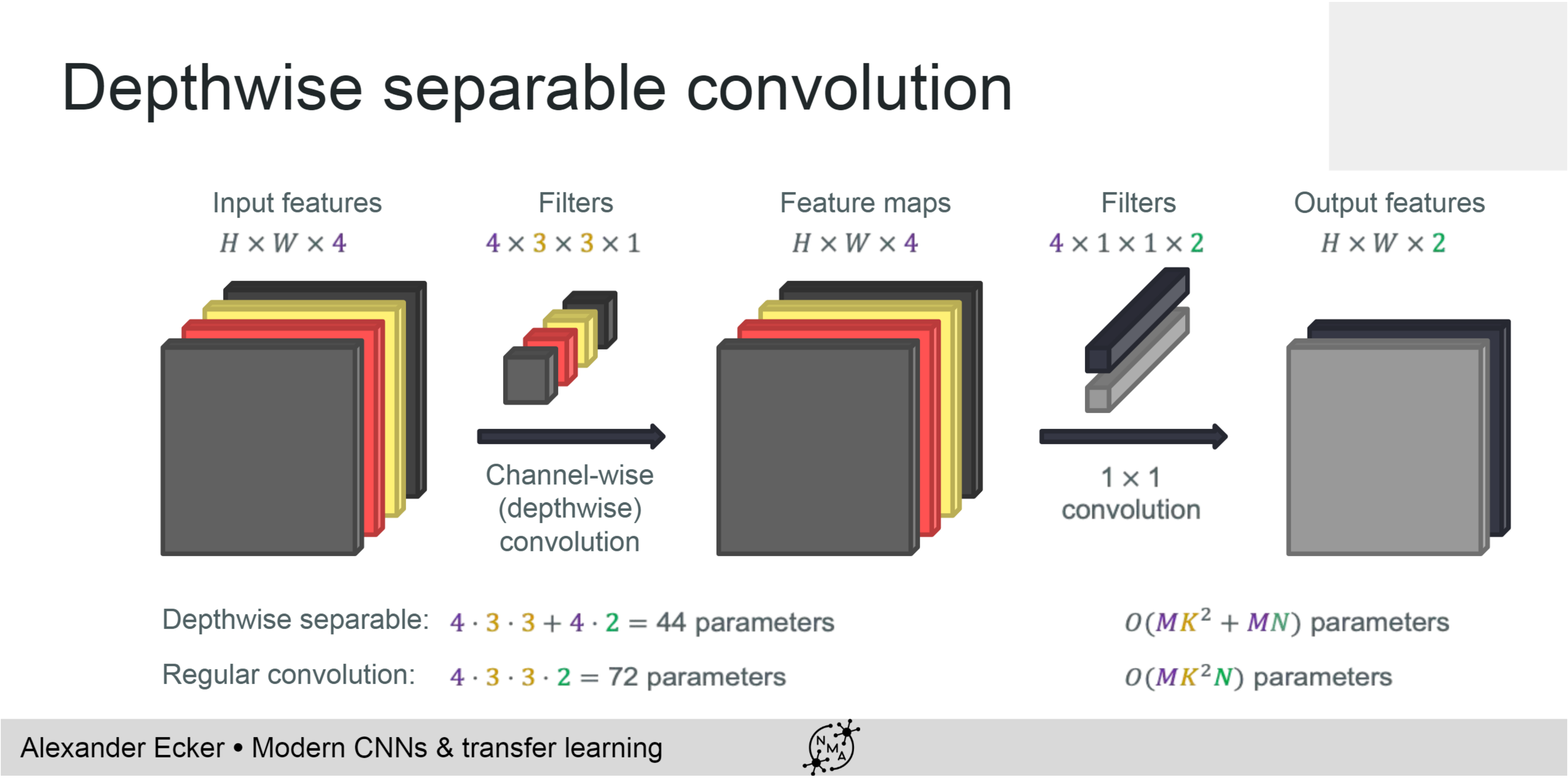
```
torchvision.models.resnet18()
…
torchvision.models.resnet152()
```

Skip connections
(Residual blocks)

He et al., *CVPR* 2016

Alexander Ecker • Modern CNNs & transfer learning

Depthwise separable convolution: compose filters as outer products to save more parameters (GoogLeNet and ResNeXt)



Depthwise separable convolution

Input features
$H \times W \times 4$

Filters
$4 \times 3 \times 3 \times 1$

Channel-wise (depthwise) convolution

Feature maps
$H \times W \times 4$

Filters
$4 \times 1 \times 1 \times 2$

1×1 convolution

Output features
$H \times W \times 2$

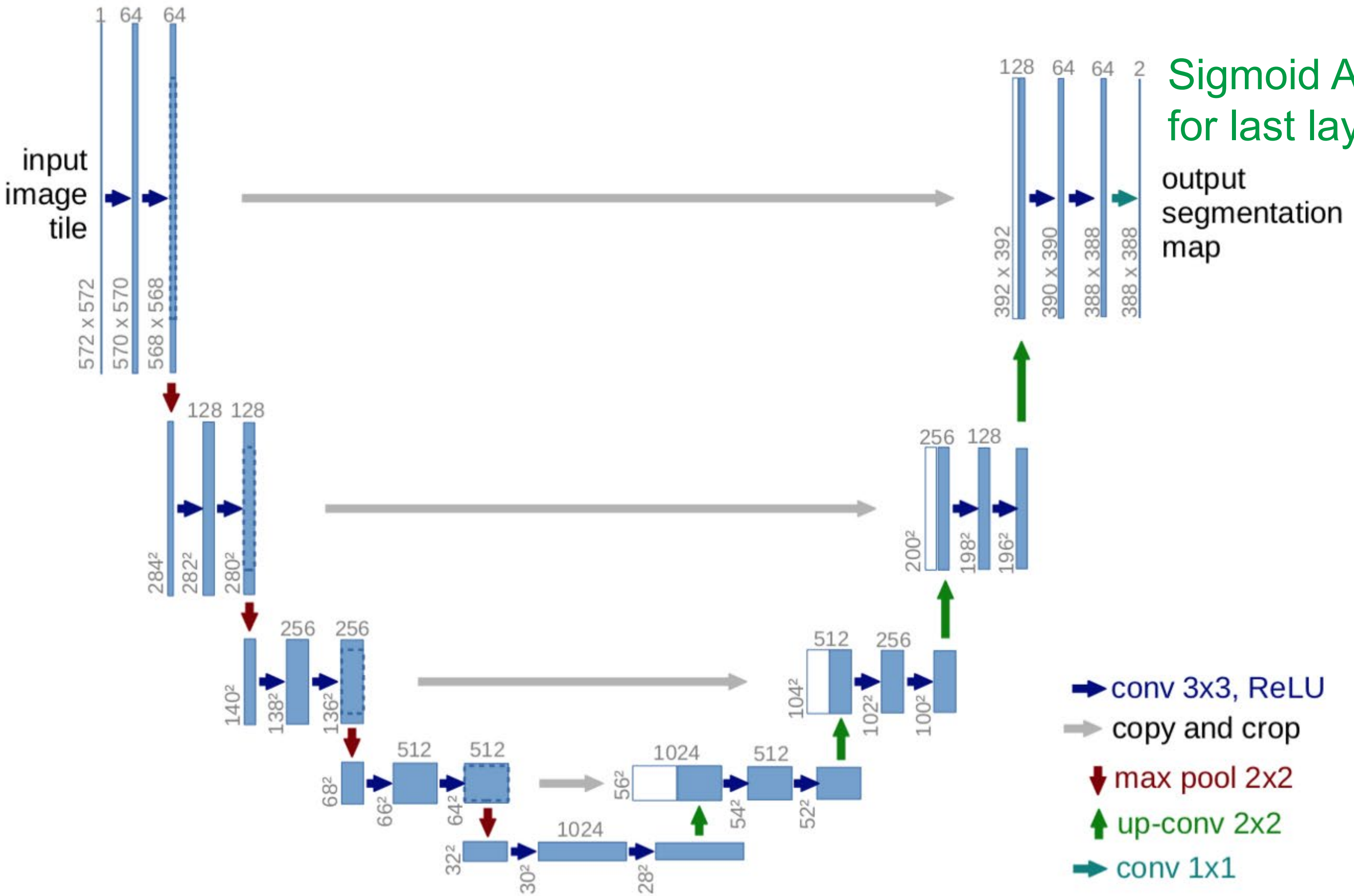
Depthwise separable: $4 \cdot 3 \cdot 3 + 4 \cdot 2 = 44$ parameters $O(MK^2 + MN)$ parameters

Regular convolution: $4 \cdot 3 \cdot 3 \cdot 2 = 72$ parameters $O(MK^2N)$ parameters

Alexander Ecker • Modern CNNs & transfer learning

Related concepts: "Inception", 1x1 convolution

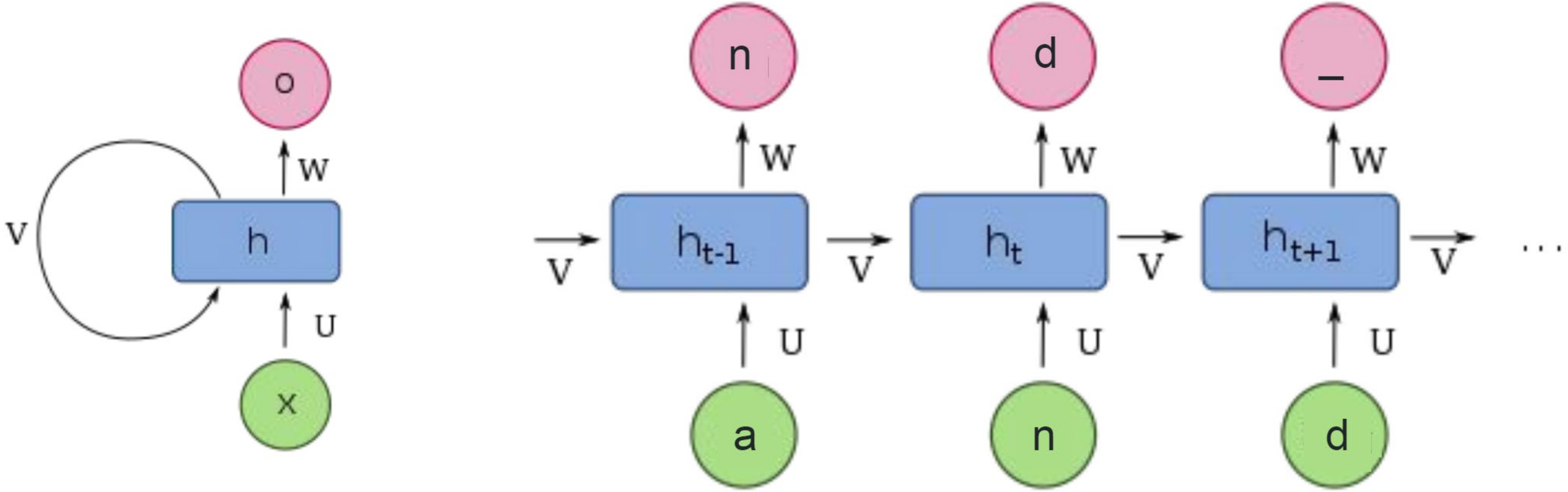
Bonus material: U-nets for image segmentation, look a lot like autoencoders (or rather, encoder-decoder chains)



Sigmoid Activation for last layer

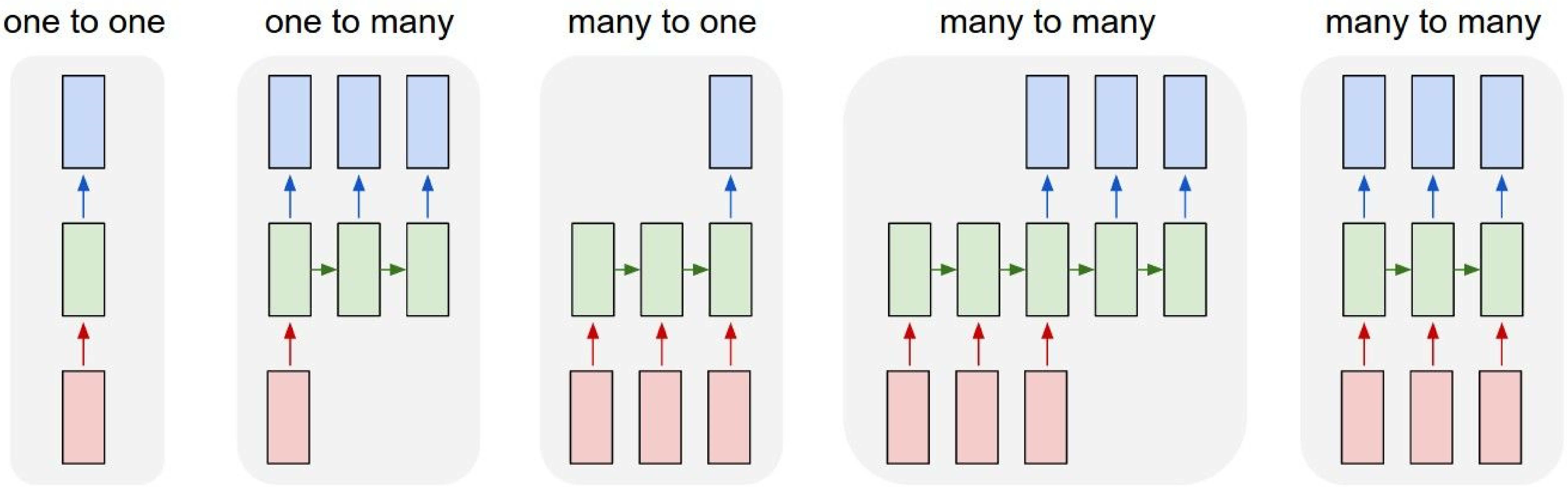
Robbegerger et al. 2015 *arXiv*

Recurrent neural networks (RNNs) save parameters by sharing them across time



Mitglied der Leibniz-Gemeinschaft

Various sequence learning frameworks ideal for RNN application



blog.floydhub.com

Mitglied der Leibniz-Gemeinschaft

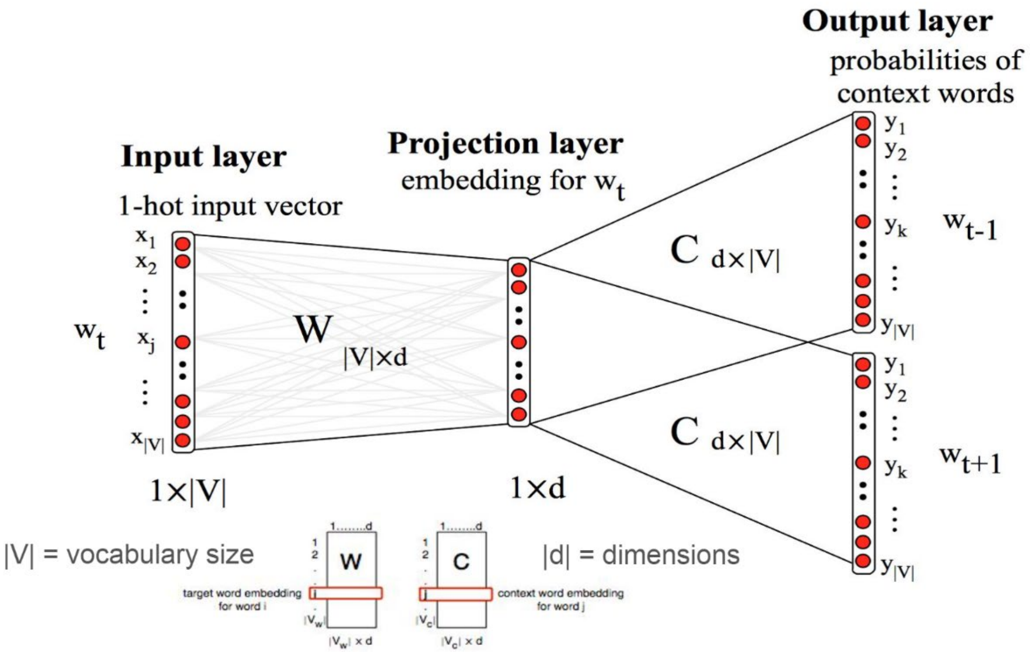
Why not just do convolution? (ARMA theory)

- Convolution = moving average filter
  - recent information only
  - limited memory unless we use many parameters

- RNN = autoregressive filter
  - includes a memory even of sequence elements far in the past
  - arbitrarily long-lasting memory (in principle) using very few parameters

In many sequence learning applications ("language models") one must first learn an "embedding"

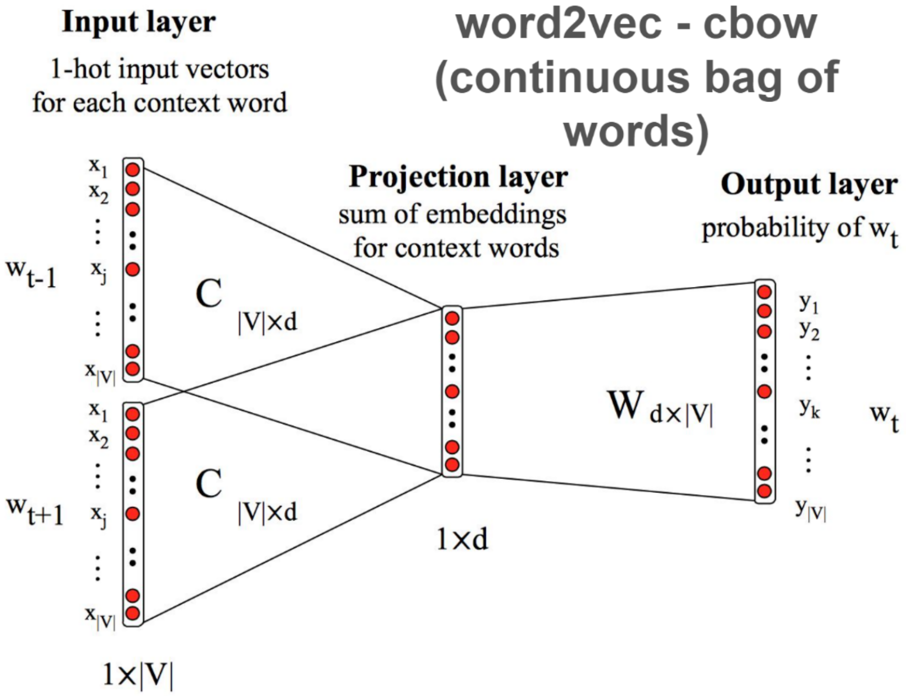
word2vec - skipgram



|V| = vocabulary size

|d| = dimensions

word2vec - cbow (continuous bag of words)

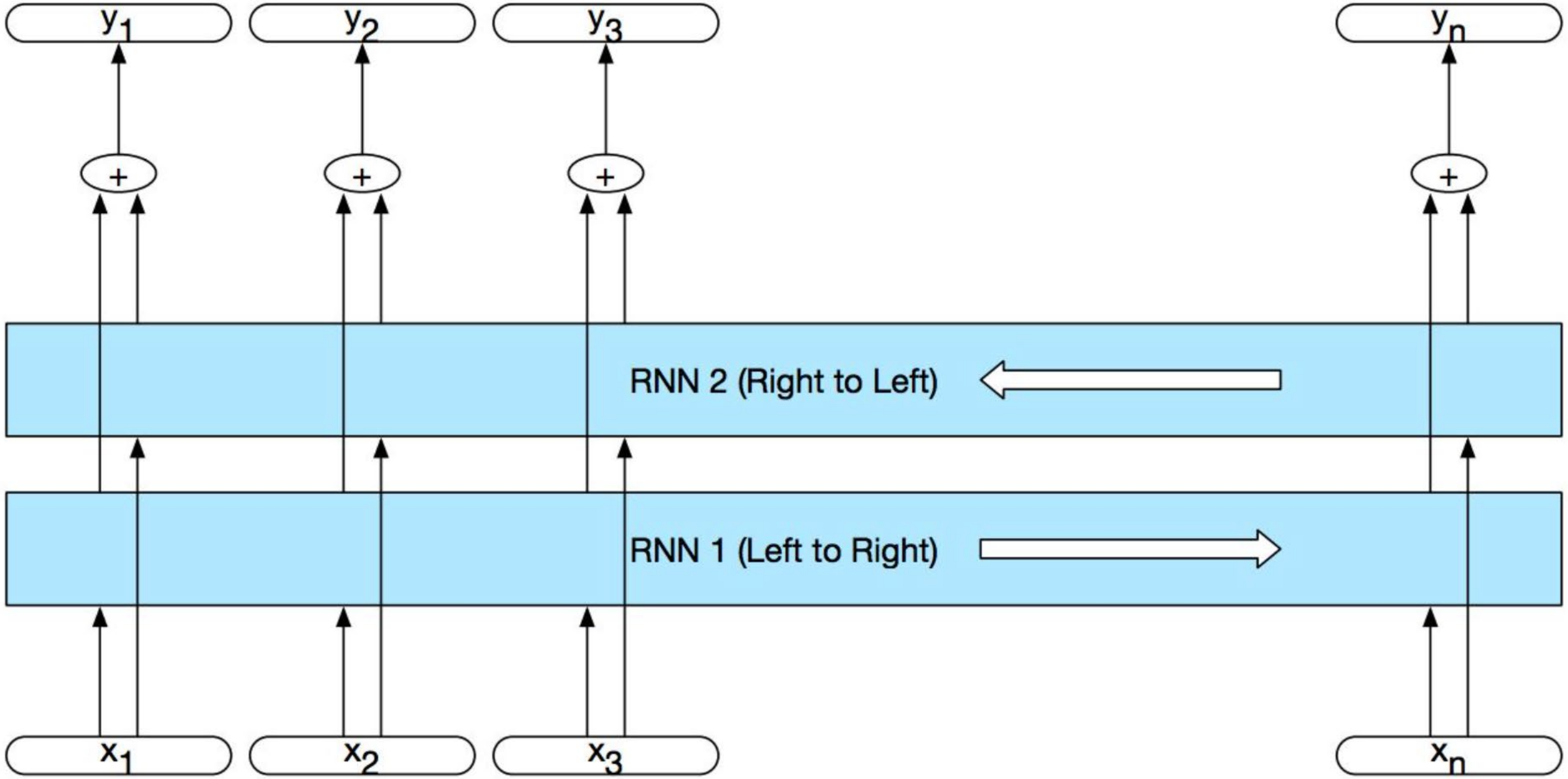


Mitglied der Leibniz-Gemeinschaft

Problems with RNNs

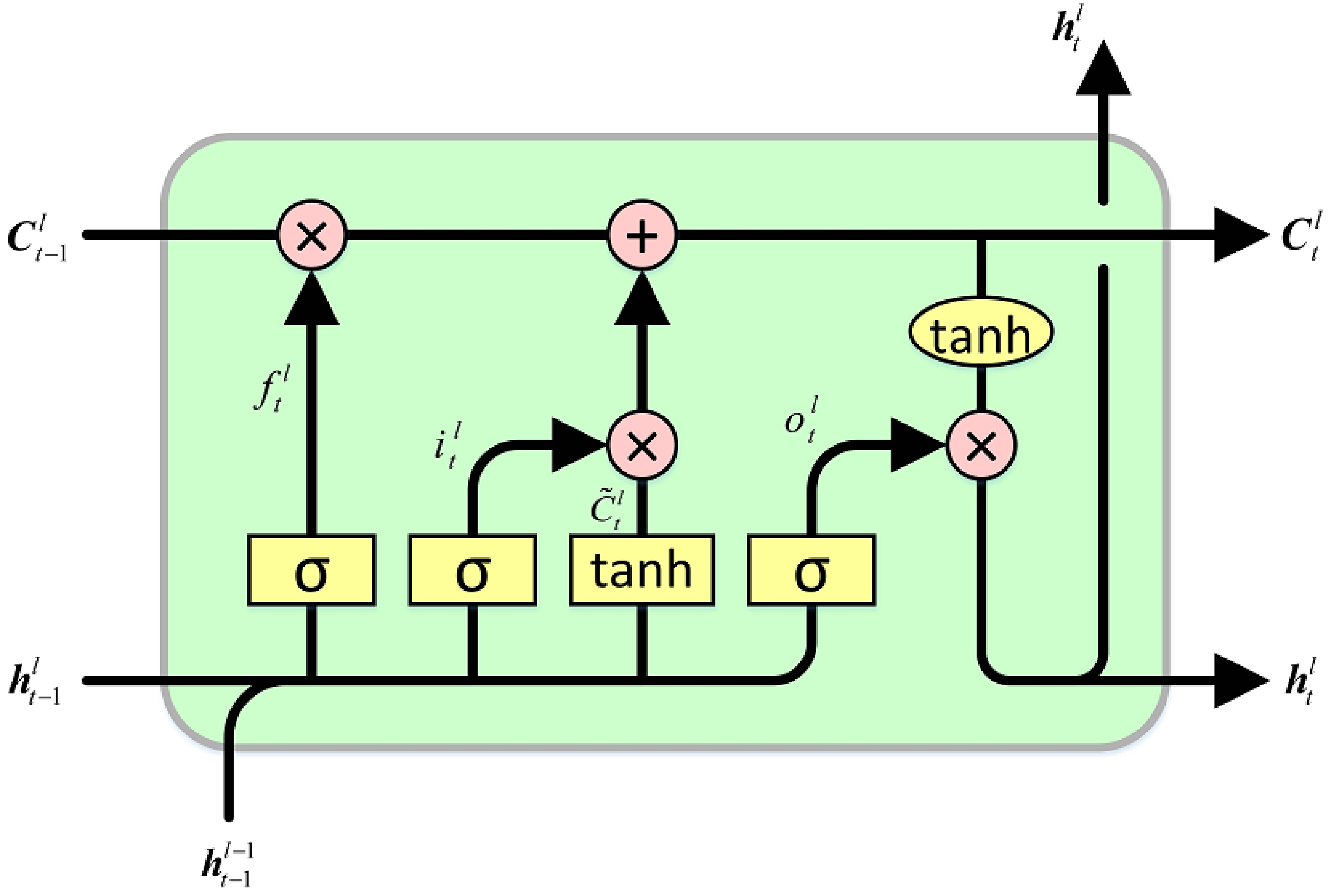
- In practice, memories that are stored over long time periods influence the state only very weakly ("forgetful")
- Also, gradients very easily explode or vanish to zero since one must "unfold" RNNs ("Backpropagation through time") to perform backpropagation
- This "unfolding" also means that gradient descent requires a lot of operations to compute

A trick to deal with forgetfulness: bidirectional RNNs



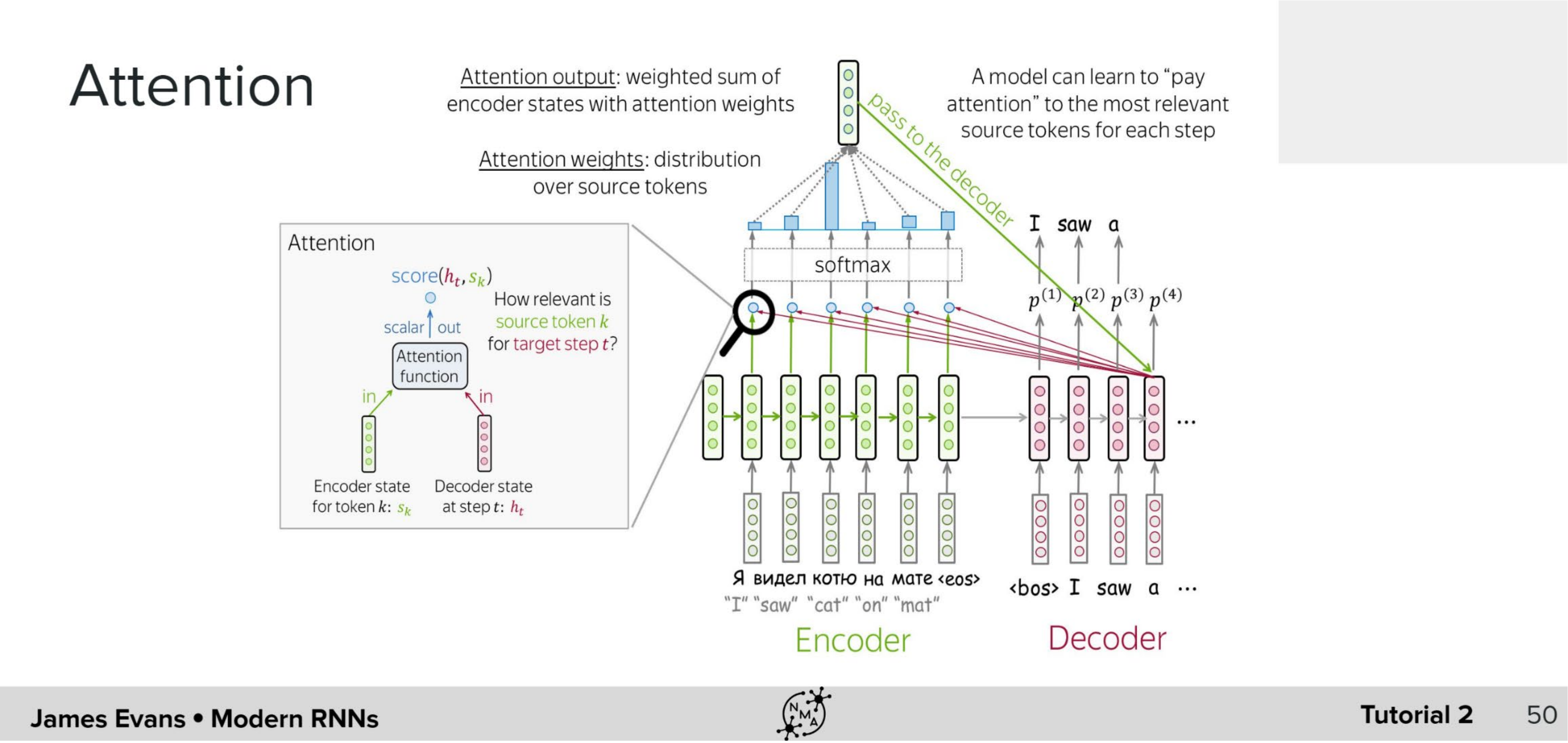
Mitglied der Leibniz-Gemeinschaft

A trick to deal with forgetfulness: LSTMs and gating



related, simpler network: GRU

Attention: weighting states by contextual similarity (relevance)



Attention

Attention output: weighted sum of encoder states with attention weights

Attention weights: distribution over source tokens

A model can learn to “pay attention” to the most relevant source tokens for each step

James Evans • Modern RNNs

Tutorial 2 50

Transformers: Ditch the RNN, "attention is all you need"!

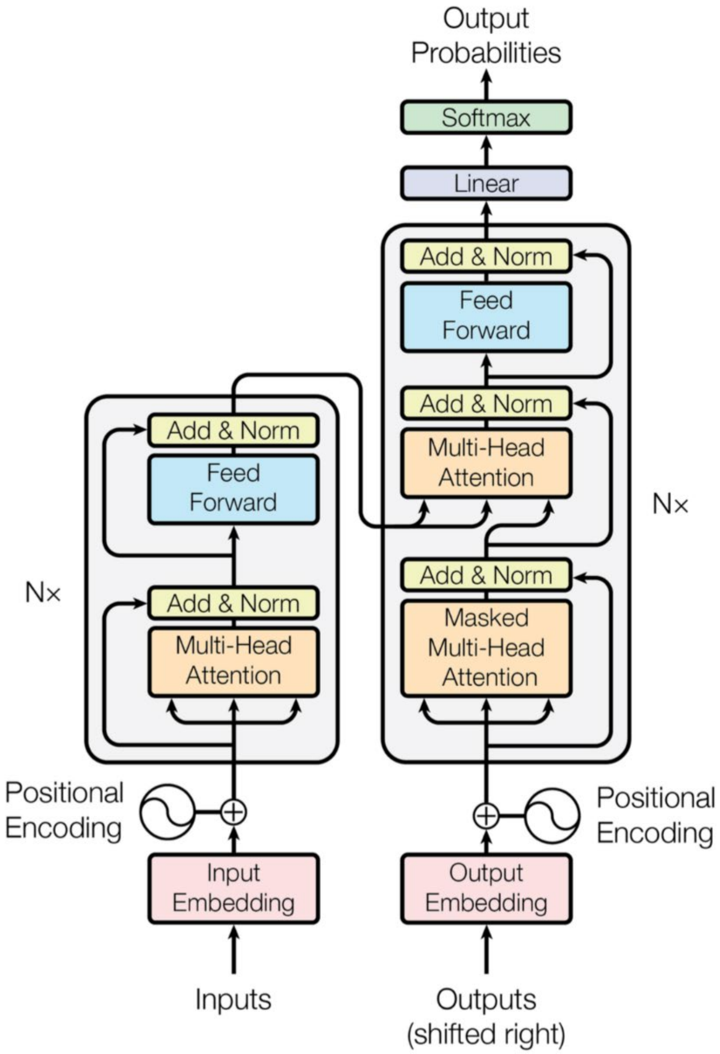


Figure 1: The Transformer - model architecture.

Vaswani et al. 2017 *arXiv*

Transformers are quite difficult to explain

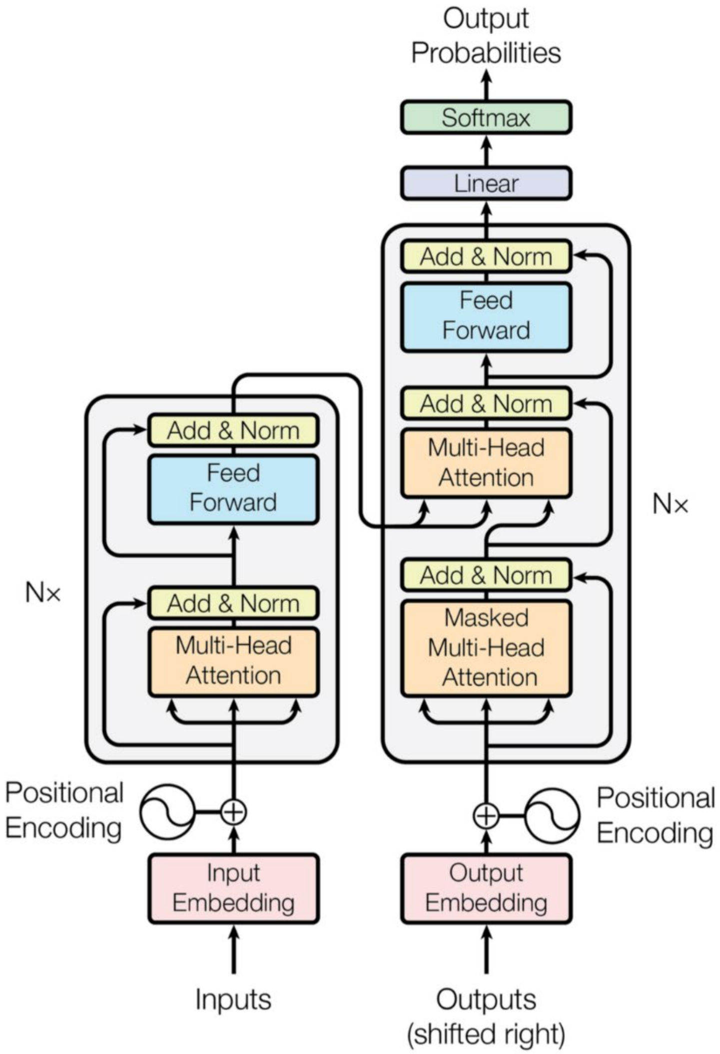


Figure 1: The Transformer - model architecture.

do the terms

"keys"

"queries"

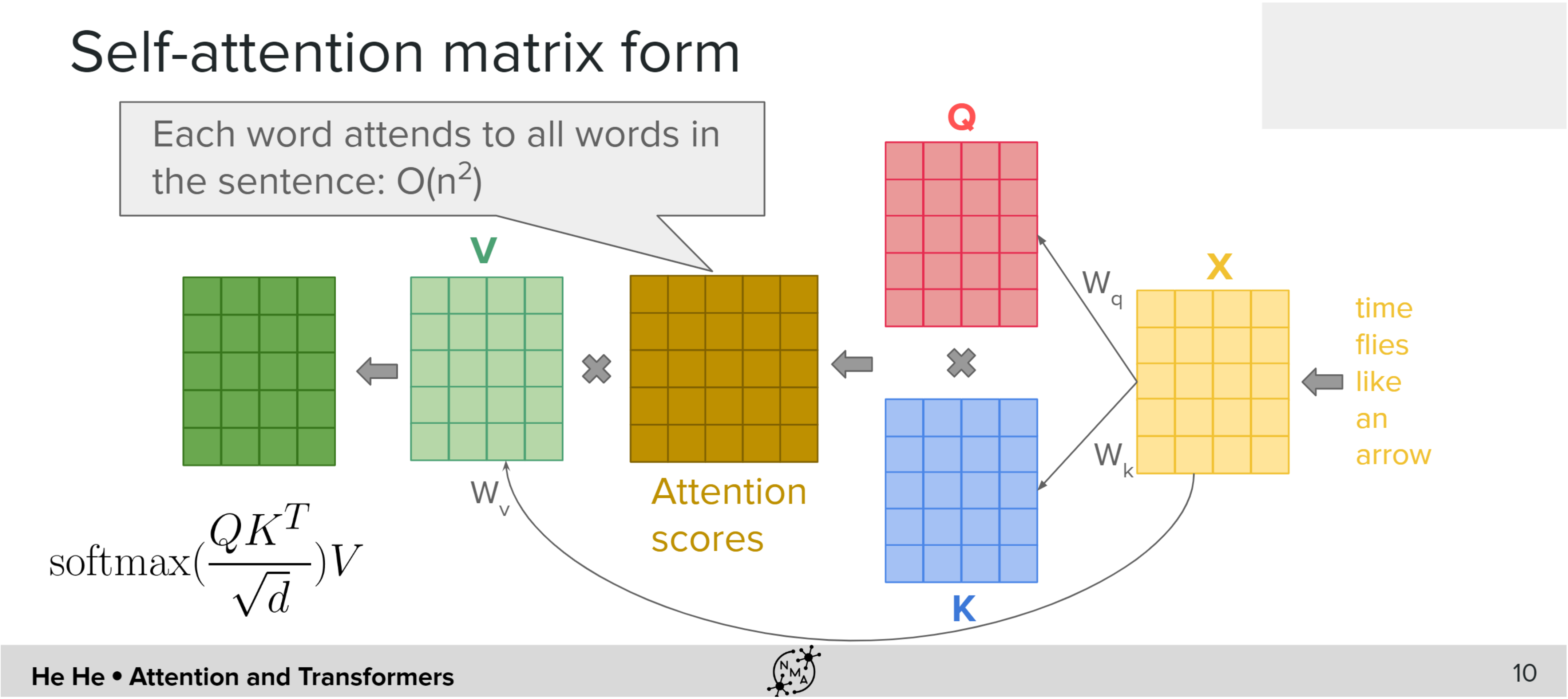
"values"

mean anything to you?

If so, you might *get* it!

Vaswani et al. 2017 *arXiv*

"Self-attention" in more detail

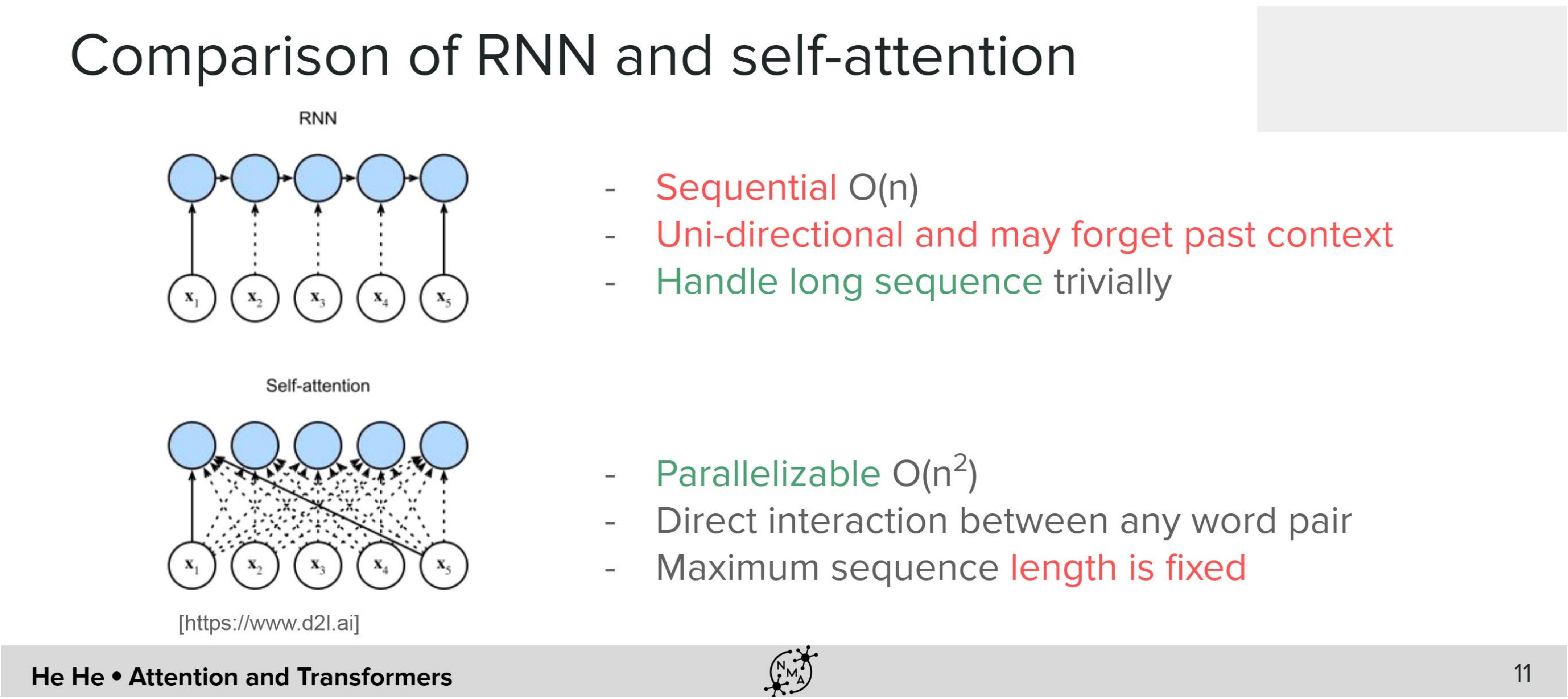


Self-attention matrix form

Each word attends to all words in the sentence: $O(n^2)$

$$\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

Pros and cons: RNNs vs. Pure Attention



Comparison of RNN and self-attention

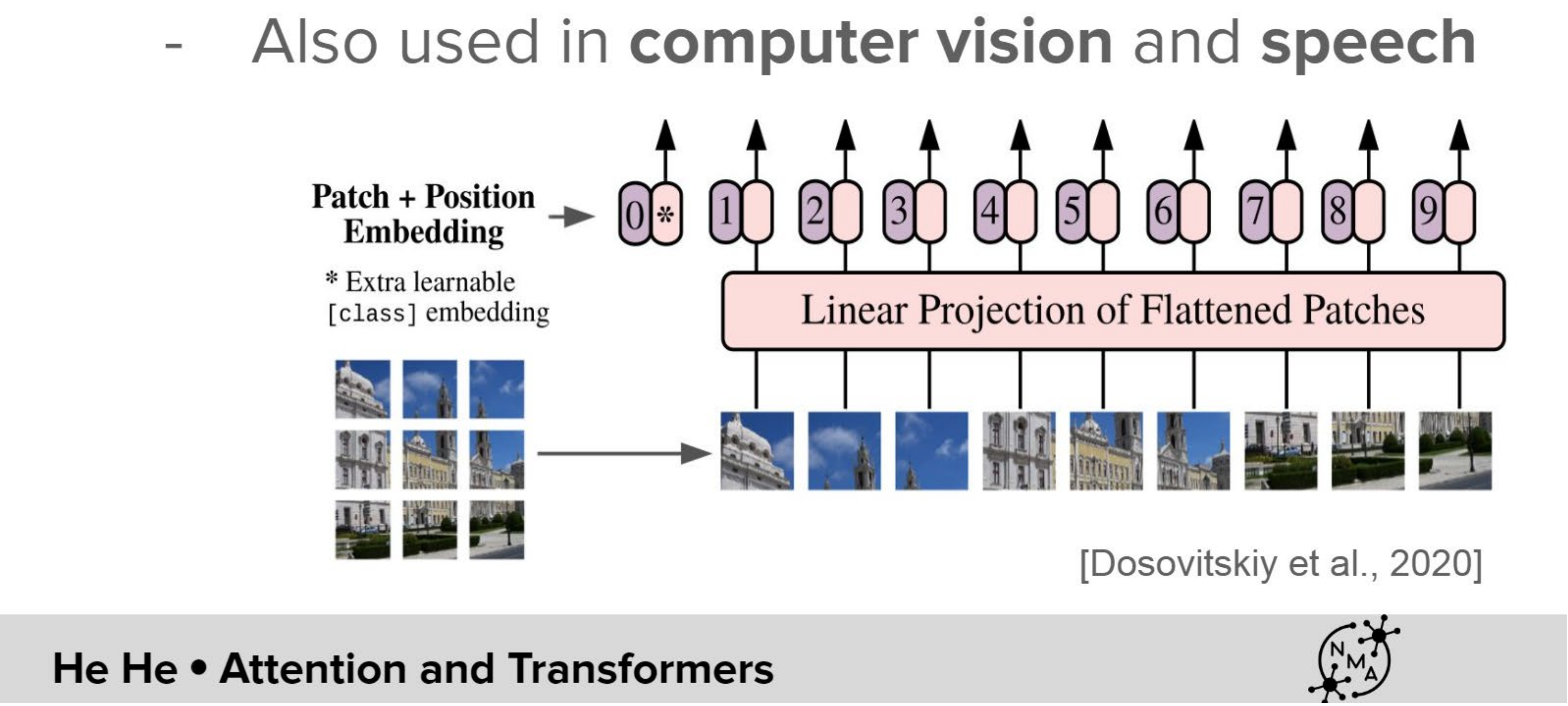
- Sequential O(n)
- Uni-directional and may forget past context
- Handle long sequence trivially

- Parallelizable $O(n^2)$
- Direct interaction between any word pair
- Maximum sequence length is fixed

[https://www.d2l.ai]

He He • Attention and Transformers 11

Transformers: more than just "language models"



- Also used in **computer vision** and **speech**

[Dosovitskiy et al., 2020]

He He • Attention and Transformers

Autoencoders: the classic generative framework

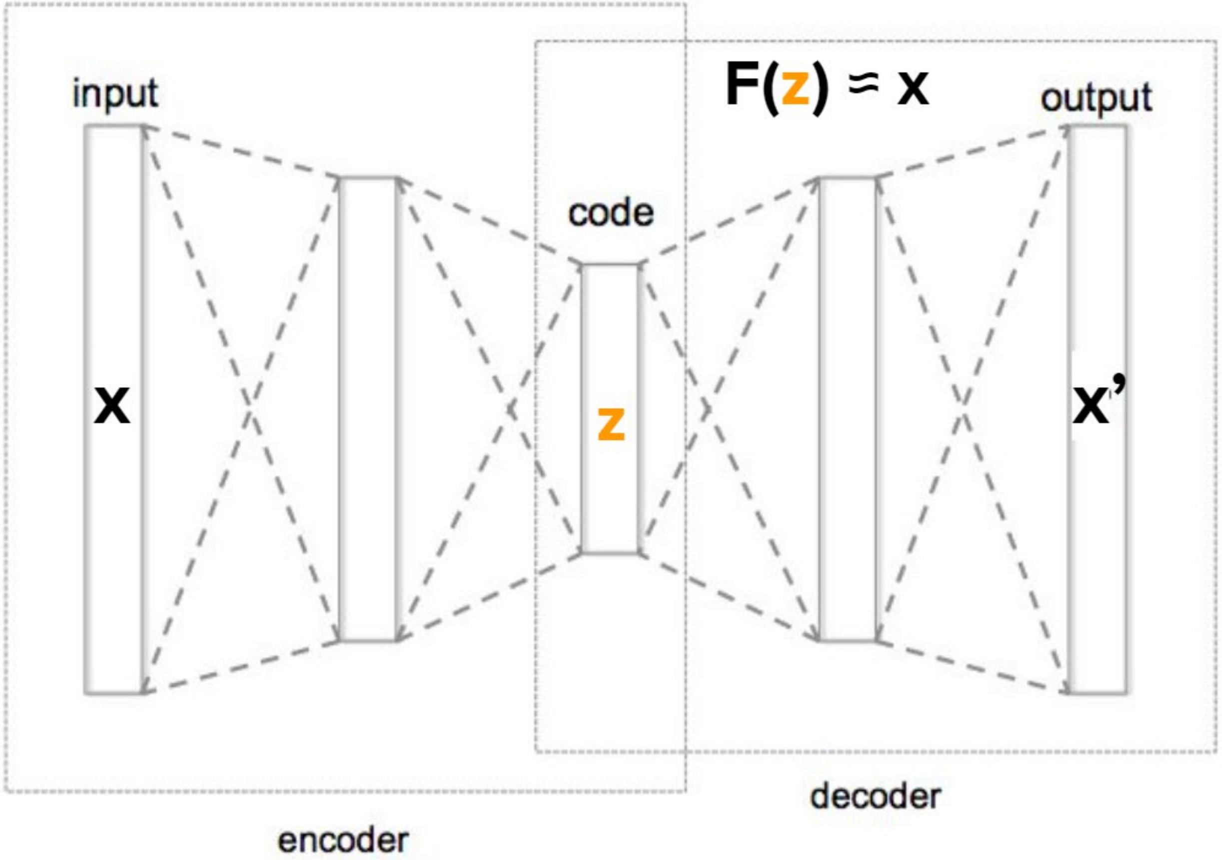
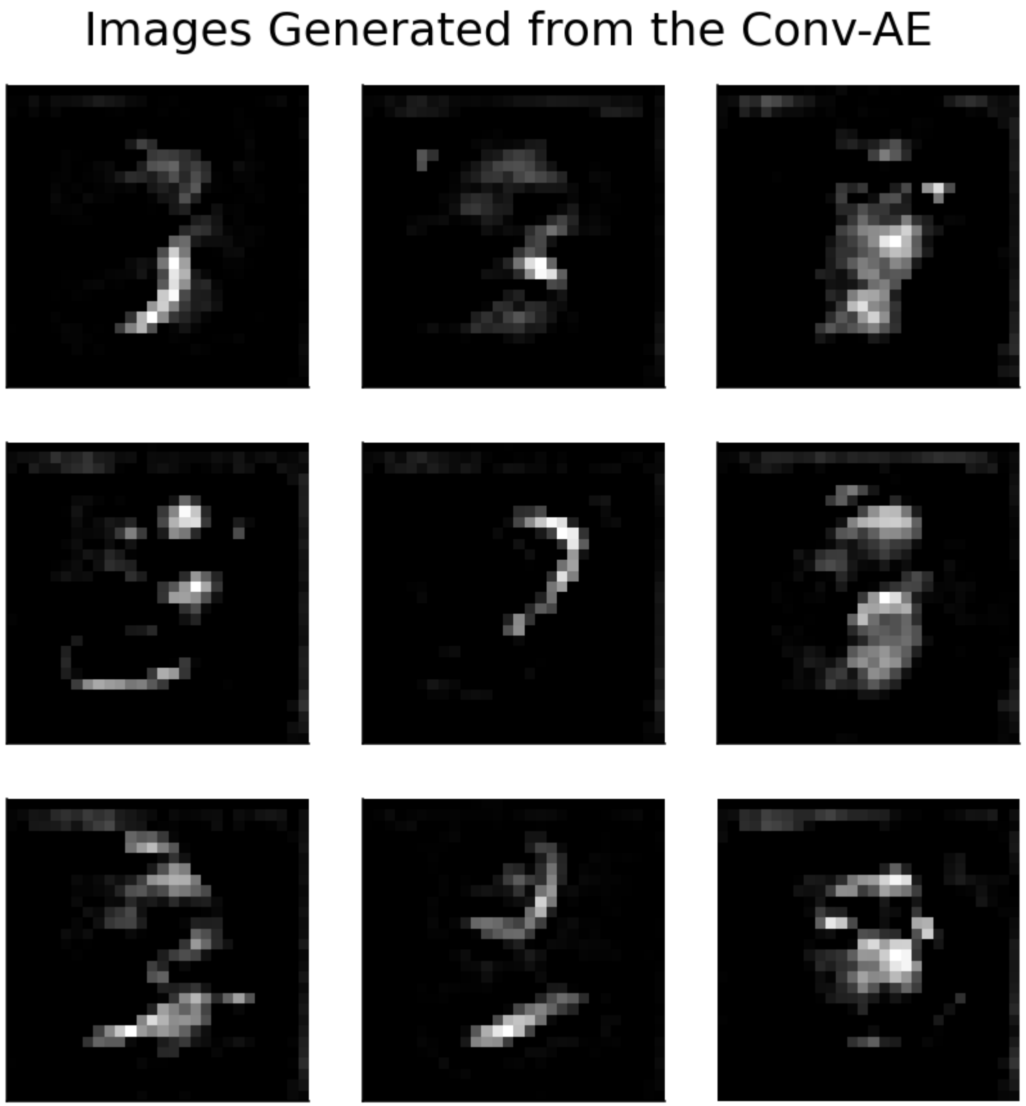
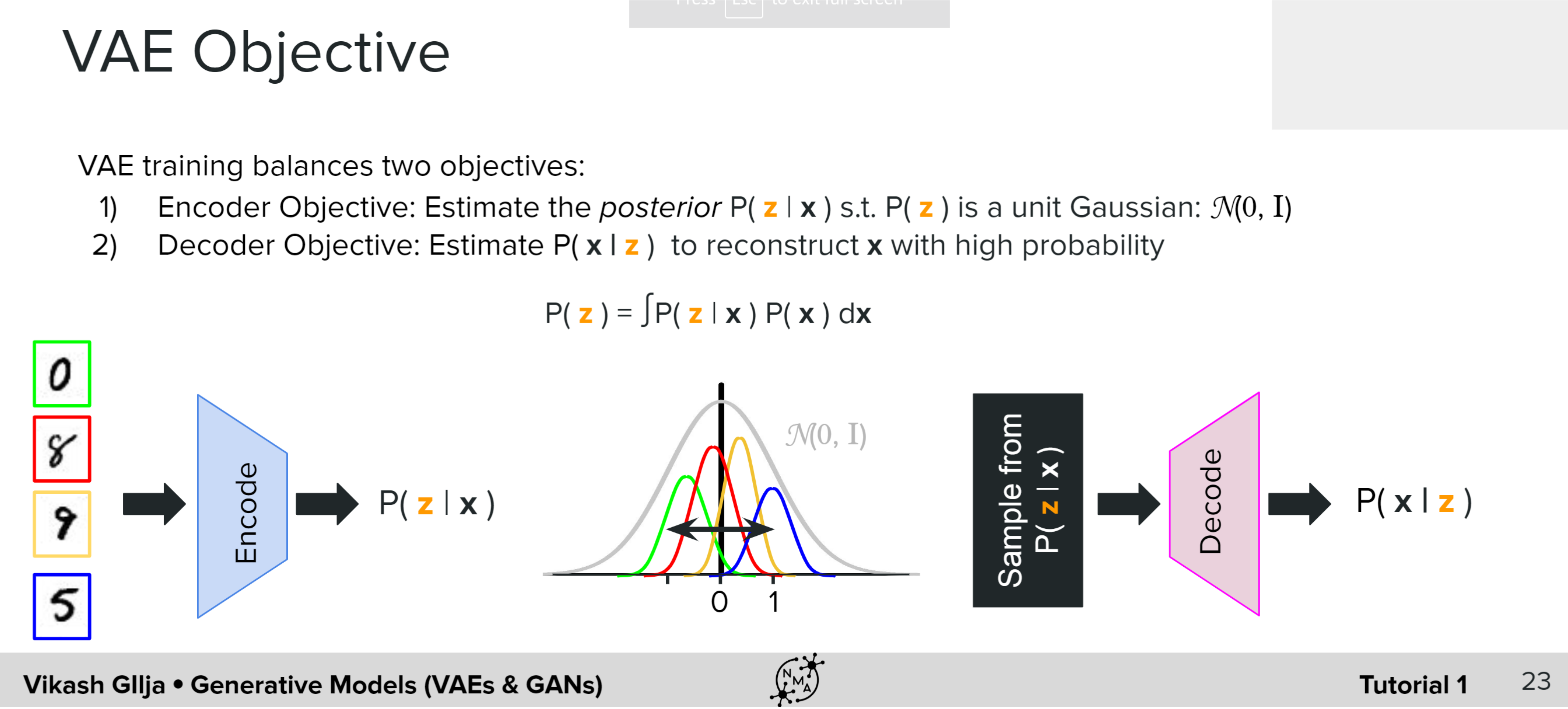


Image Credit: Chervinskii, CC BY-SA 4.0, via Wikimedia Commons

Classical autoencoders don't tile latent state space evenly


Images Generated from the Conv-AE

Variational autoencoders smooth the underlying state space



VAE Objective

VAE training balances two objectives:

1) Encoder Objective: Estimate the *posterior* $P(\mathbf{z} \mid \mathbf{x})$ s.t. $P(\mathbf{z})$ is a unit Gaussian: $\mathcal{N}(0, I)$
2) Decoder Objective: Estimate $P(\mathbf{x} \mid \mathbf{z})$ to reconstruct **x** with high probability

$$P(\mathbf{z}) = \int P(\mathbf{z} \mid \mathbf{x})\, P(\mathbf{x})\, d\mathbf{x}$$

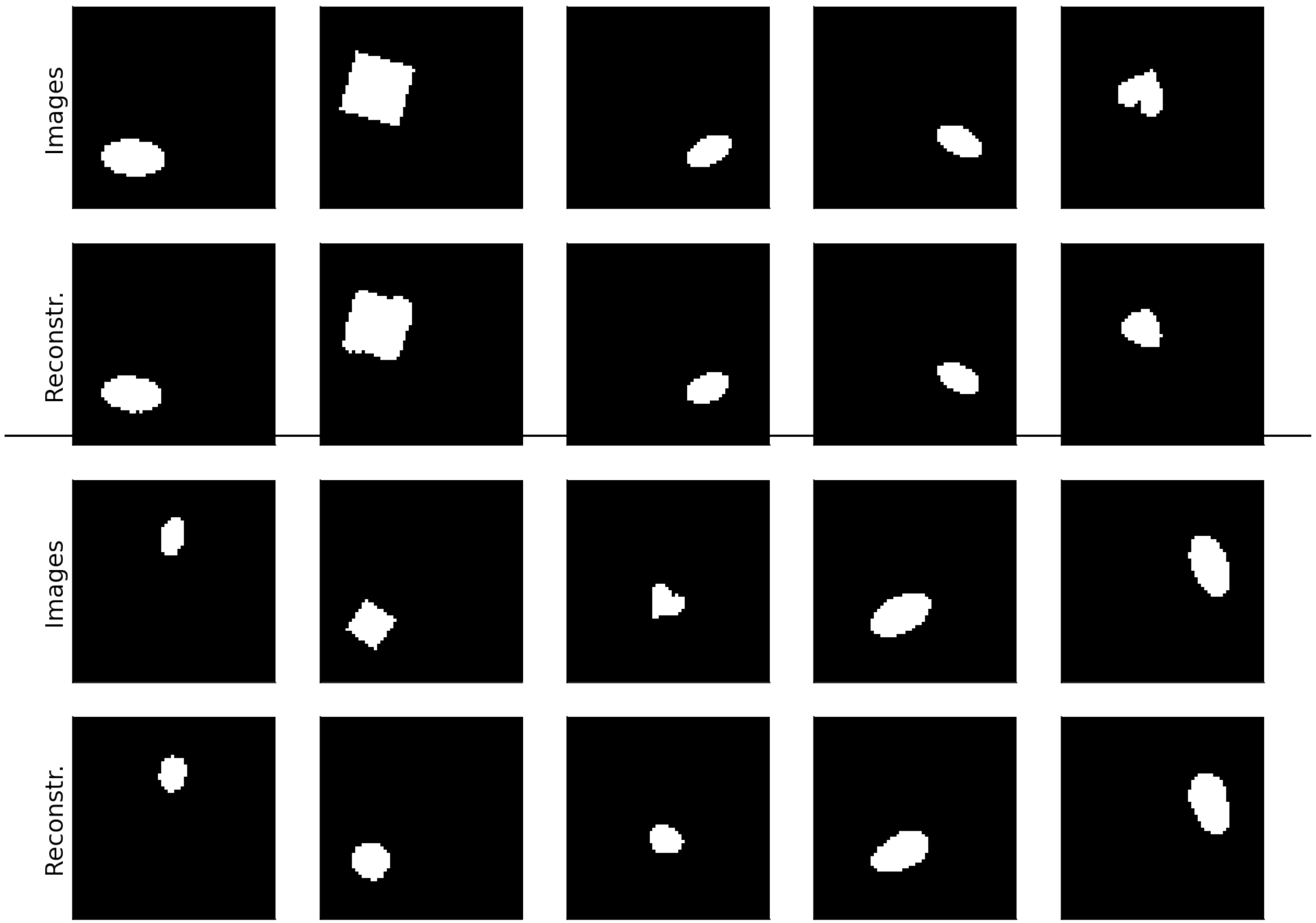
Vikash GIlja • Generative Models (VAEs & GANs) **Tutorial 1** 23

VAEs yield better random samples

Images Generated from a Conv-Variational-AE

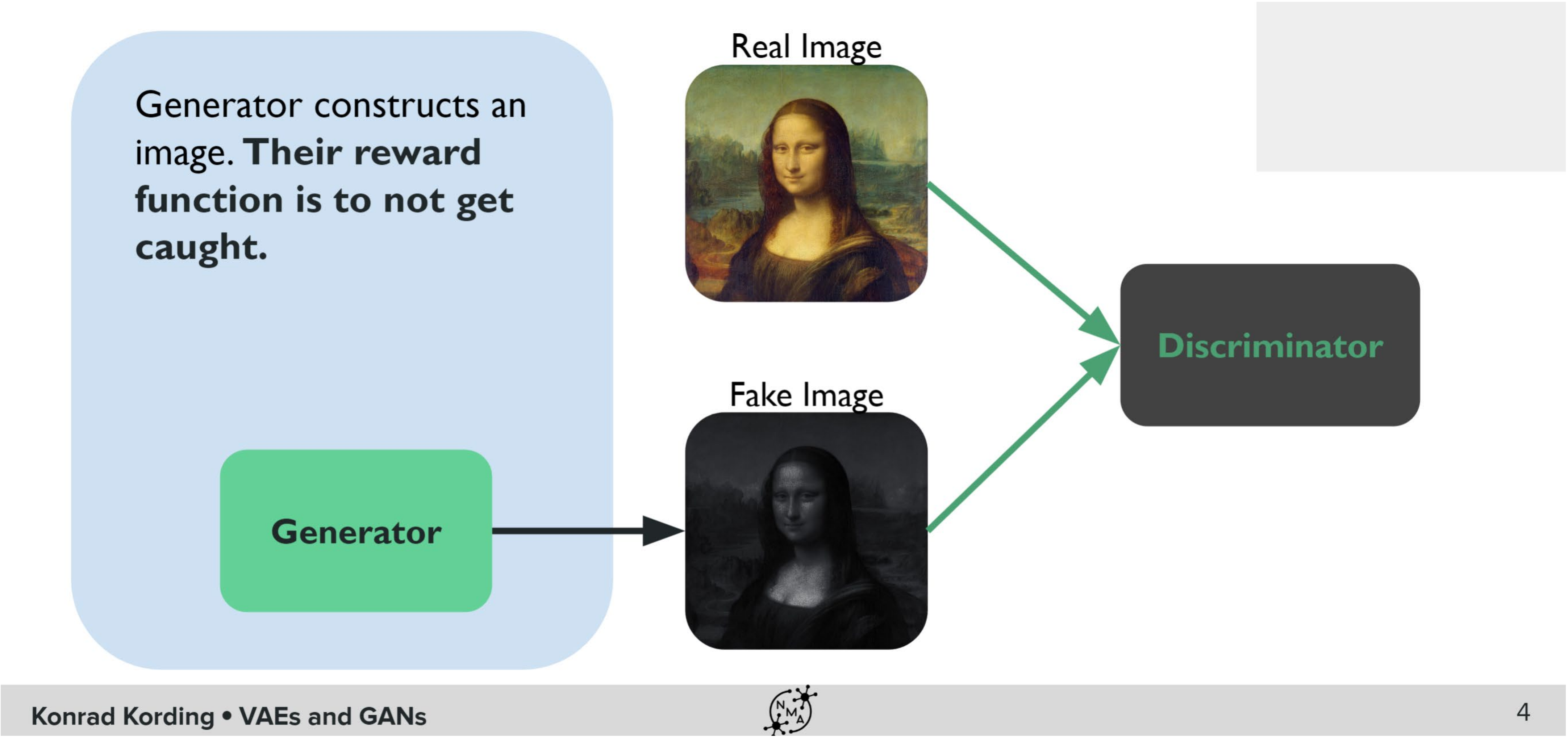
Note: Convolutional VAEs are generally bad at invariant representations, unlike regular CNNs for image classification

VAE test set image reconstructions



Mitglied der Leibniz-Gemeinschaft

Generative Adversarial Networks (GANs): a forger-critic model of learning

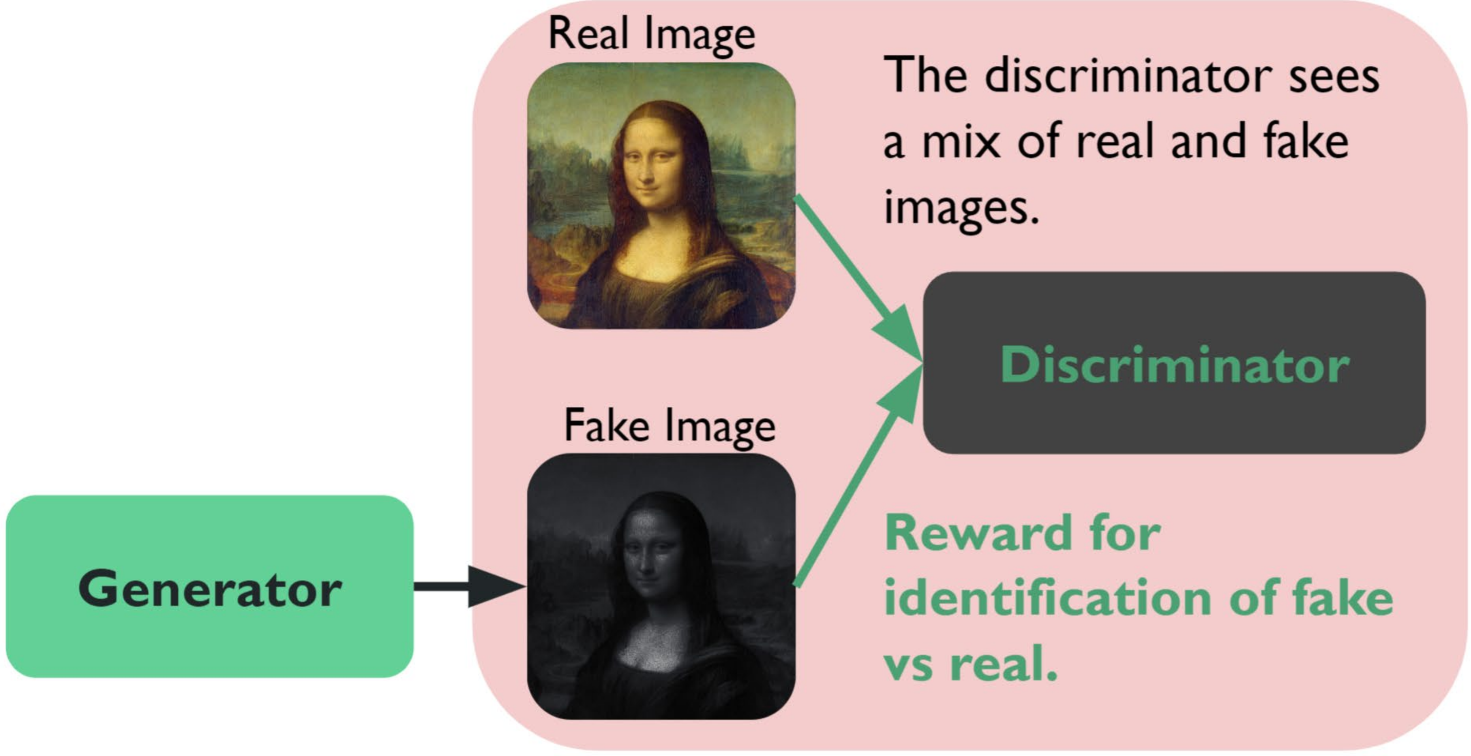


Generator constructs an image. **Their reward function is to not get caught.**

Konrad Kording • VAEs and GANs

4

Generative Adversarial Networks (GANs): a forger-critic model of learning



Real Image

The discriminator sees a mix of real and fake images.

Discriminator

Fake Image

Generator

Reward for identification of fake vs real.

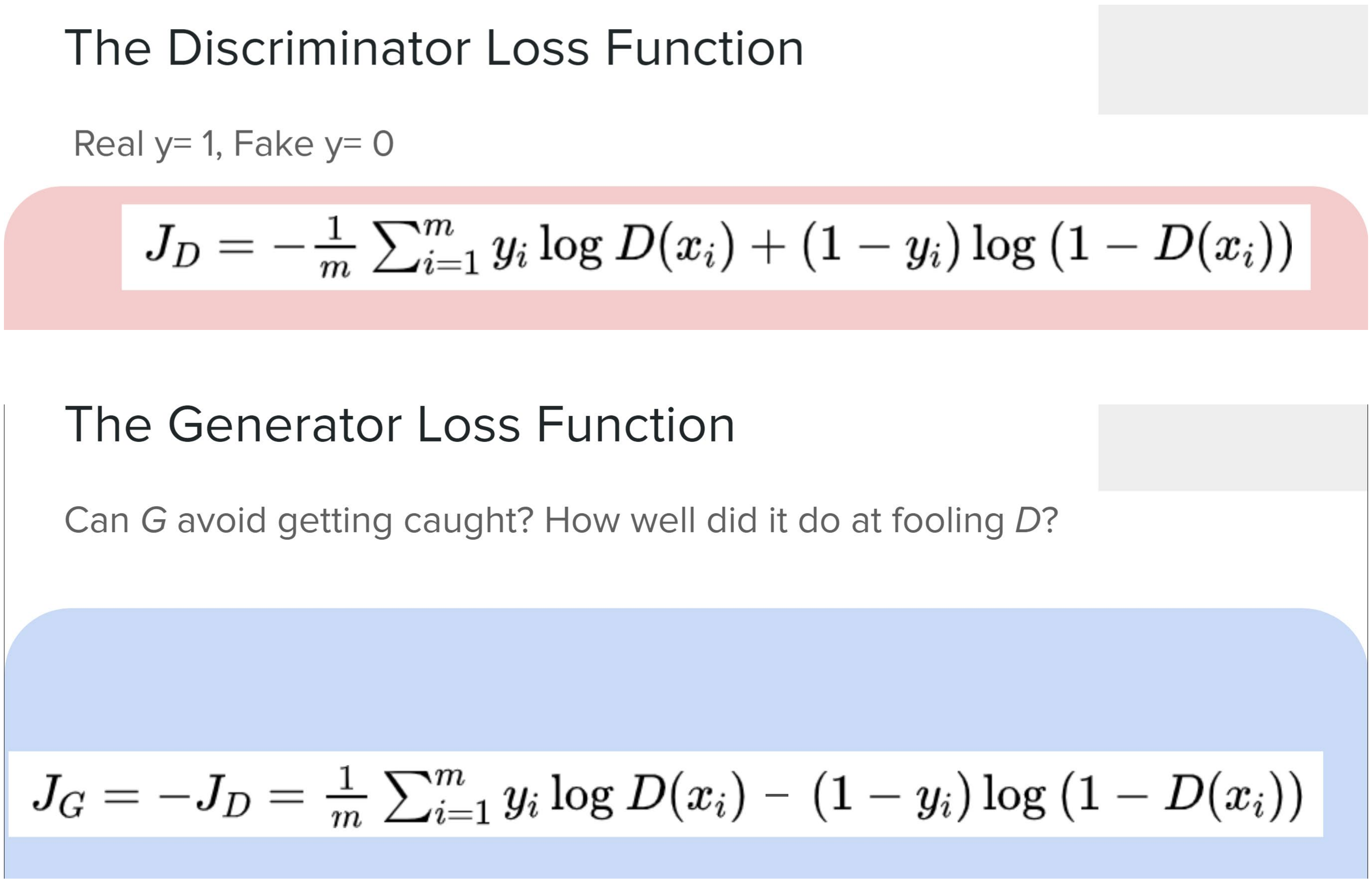
P(Real)=.9

Konrad Kording • VAEs and GANs

5

Mitglied der Leibniz-Gemeinschaft

Loss functions for those who care (cross-entropy)



The Discriminator Loss Function

Real y= 1, Fake y= 0

$$J_D = -\frac{1}{m}\sum_{i=1}^{m} y_i \log D(x_i) + (1-y_i)\log\left(1-D(x_i)\right)$$

The Generator Loss Function

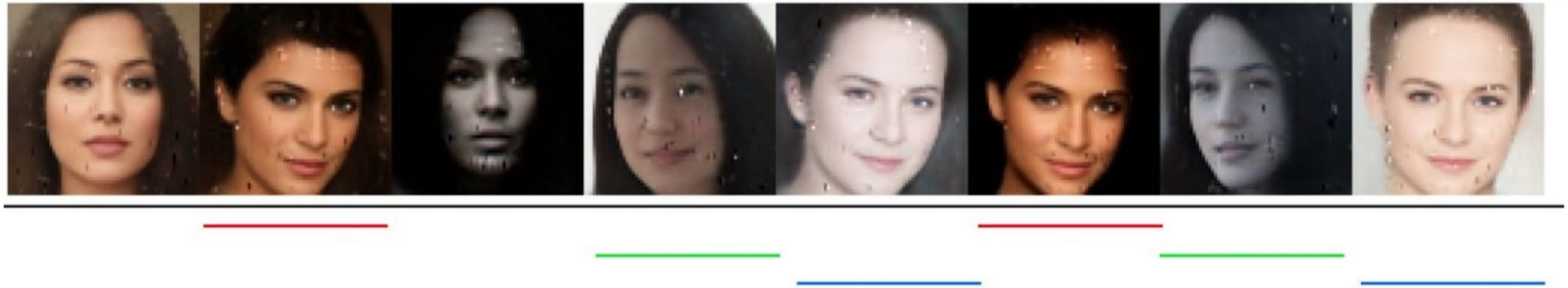
Can *G* avoid getting caught? How well did it do at fooling *D*?

$$J_G = -J_D = \frac{1}{m}\sum_{i=1}^{m} y_i \log D(x_i) - (1-y_i)\log\left(1-D(x_i)\right)$$

Cat-and-mouse

Finicky to train!

GANs generate crisp output, but suffer from mode collapse



Neuromatch was a very anti-GAN summer school…

Deutsches Primatenzentrum
Leibniz-Institut für Primatenforschung

Week 3: Advanced Topics

Mitglied der Leibniz-Gemeinschaft

Unsupervised and self-supervised learning

- Unsupervised
  - "Deep belief networks"
  - Proper unsupervised methods aren't there yet

- "Self-supervised"
  - Take an image dataset
  - Data augmentation to hell and back
  - Perform image classification, assign augmented images to label of source image
  - Basically, a way to try to learn invariant representations without human labels

Reinforcement learning

- A fairly complicated topic, because it's not *just* machine learning

- In a nutshell
  - "state", "action", "reward", "policy" – important vocabulary for any RL model
  - If you are familiar with Q-learning and Dynamic Programming, then you have the framework you need to understand this
  - If you have this background: **Deep Q learning** uses an ANN to map states to Q (prospective reward) values instead of trying to populate a full Q table through exploration alone
  - Optimal policy can then be inferred as the one that maximizes Q from a given state
  - **Policy gradient** methods learn the policy directly, and thus may be more intuitive for non-CS folks (and seem to do better, too)

- Probably of interest to learn
  - Not data limited!
  - Training agents to move a body effectively in an environment: that's RL!

Continual learning

- Catastrophic forgetting
  - Train on one task
  - Then train on another
  - Uh oh! The network forgot how to do the first task

- Strategies to counteract:
  - Rehearsal / Replay, e.g., Gradient episodic memory (GEM)
  - Regularization, e.g., Elastic weight consolidation (EWC)
  - Supermasks in superposition

- CORe50 dataset to stress-test continual learning paradigms

"Out-of-distribution" learning

- Transfer learning
  - Train a net on one task/dataset (e.g., Imagenet)
  - Then use this to initialize your net for a new task/dataset
  - Normally, random initialization, e.g. Xavier initialization, is used

- Meta-learning
  - "Learning to learn"
  - One goal: given only 5-10 exemplars, learn to identify a new image class
  - Not much covered in the way of methods...

- Continual learning fits under this umbrella, too

Ethics topics covered along the way

- Don't buy into "hype" (or "anti-hype" for that matter)

- Being aware of biases (gender, race, age, ...)
  - And how DL can reinforce harmful biases

- Deepfakes

- Environmental impacts

- RL for, say, self-driving cars
  - Trolley problem?
  - https://www.moralmachine.net/

Mitglied der Leibniz-Gemeinschaft

Projects: loads of resources/datasets provided

- https://deeplearning.neuromatch.io/projects/docs/datasets_and_models.html
- https://deeplearning.neuromatch.io/projects/Neuroscience/ideas_and_datasets.html
- https://deeplearning.neuromatch.io/projects/Neuroscience/algonauts_videos.html
- In many cases, "domain adaptation" of an existing model was often the most effective solution...

Final remarks

- A lot of material, could not retain it all
- Net valuable experience
- Decent starting point for developing a DL aspect of a project