Image generation

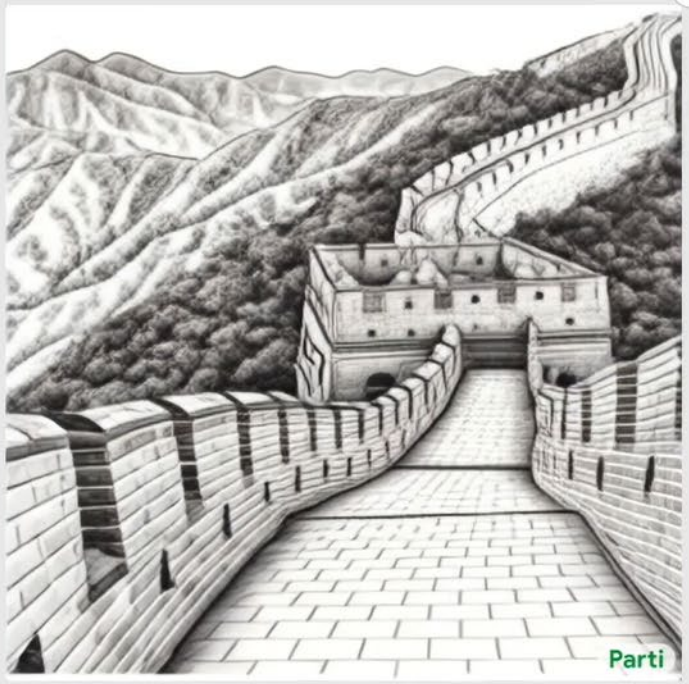
AI for Media, Art & Design, Spring 2024

Prof. Perttu Hämäläinen

Aalto University

Imagen

DALL-E 2



Parti

Text to Image

Steve Seitz

0:07 / 5:59

Text to Image in 5 minutes: Parti, Dall-E 2, Imagen

https://www.youtube.com/watch?v=GYyP7Ova8KA

Graphics in 5 Minutes
17,3 t. tilaajaa

Tilaa

363 Jaa Lataa Klippi Tallenna

Text to Image Part 2

Imagen

DALL-E 2

Steve Seitz

0:01 / 6:12

Text to Image: Part 2 -- how image diffusion works in 5 minutes

https://www.youtube.com/watch?v=lyodbLwb2lY

Graphics in 5 Minutes
17,3 t. tilaajaa

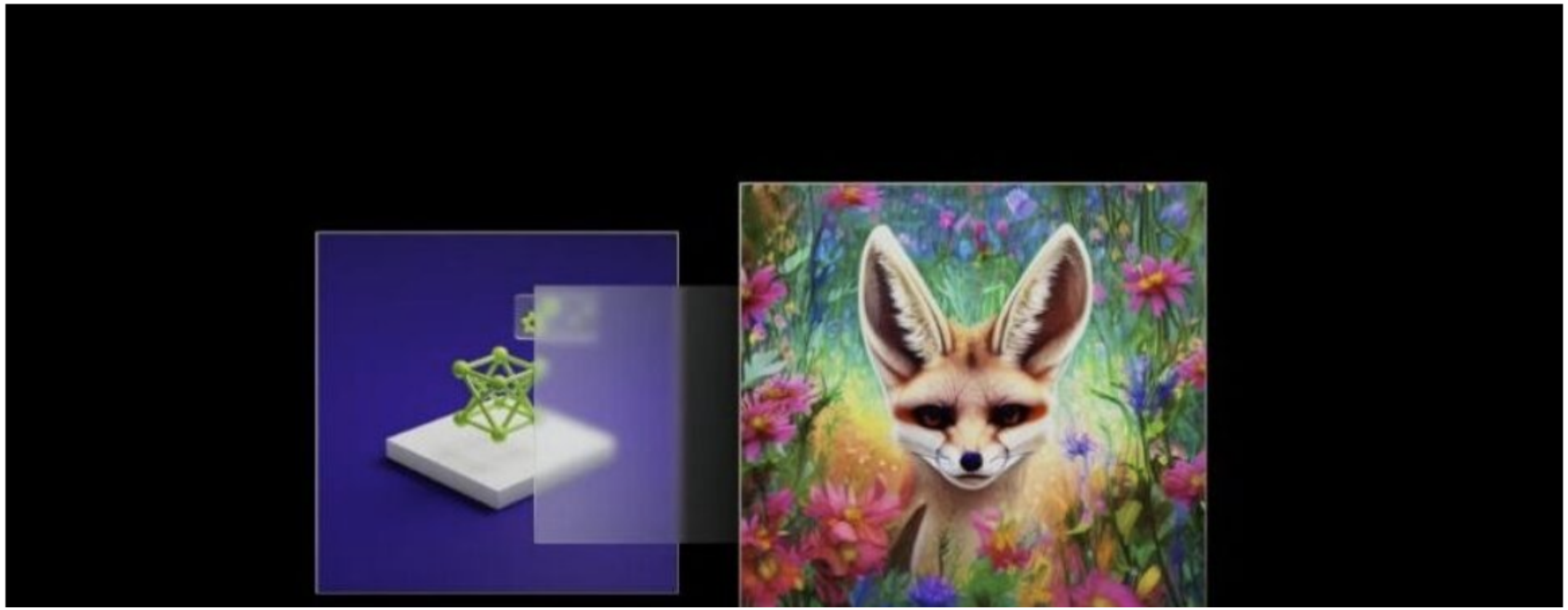
Tilaa

345 Jaa Lataa Klippi Tallenna

Generative AI Research Spotlight: Demystifying Diffusion-Based Models

Dec 14, 2023

By Miika Aittala



https://developer.nvidia.com/blog/generative-ai-research-spotlight-demystifying-diffusion-based-models/

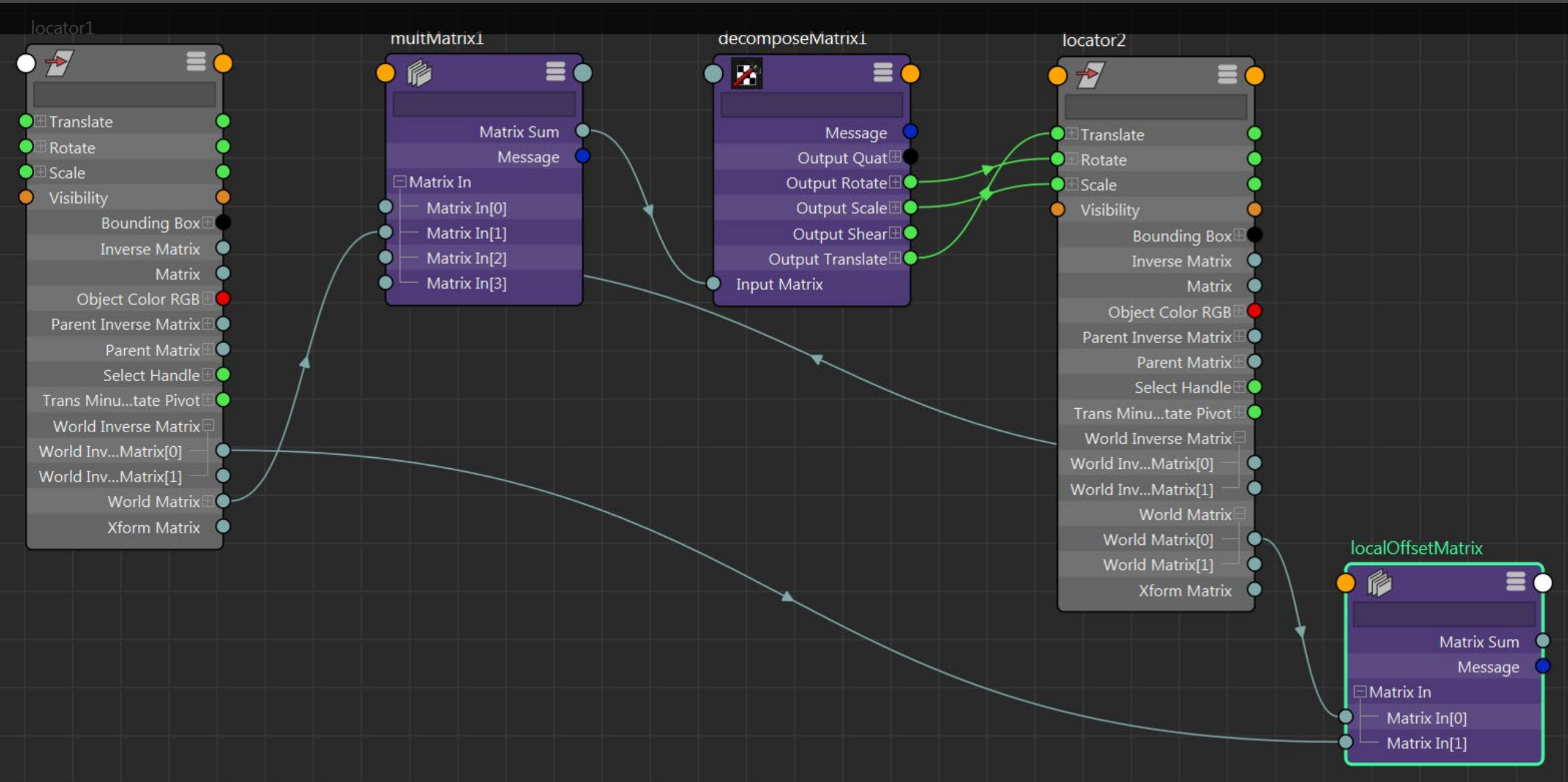
Contents

- **Preliminaries: compute graphs, artificial neurons, activation functions, loss functions, latent spaces**
- Understanding nonlinear activations
- Image generator architectures
- Quality metrics: FID, Precision & Recall

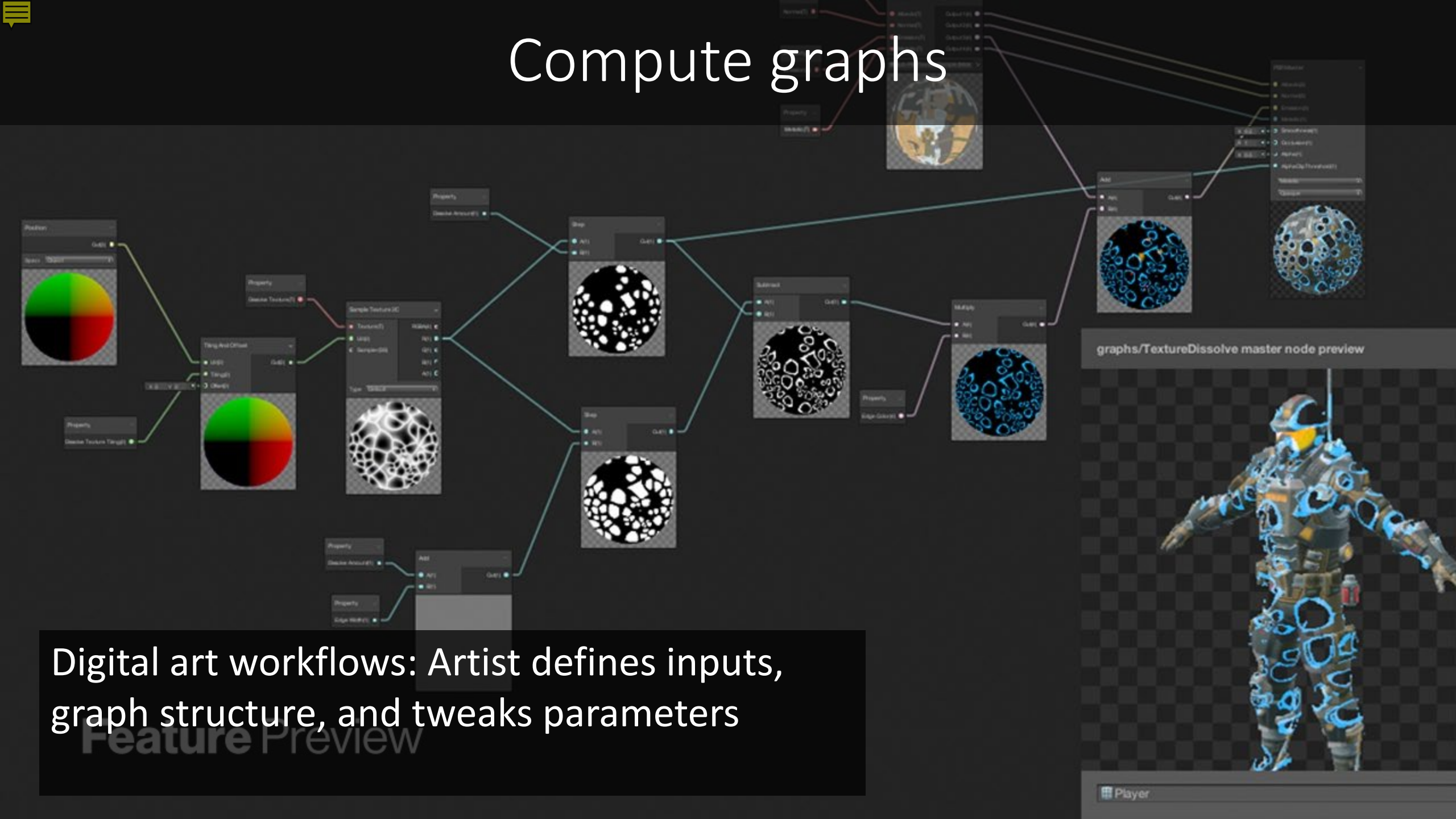
3Blue1Brown YouTube channel: Visual intuitions to math and neural networks

- 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 descend)
- 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

Compute graphs

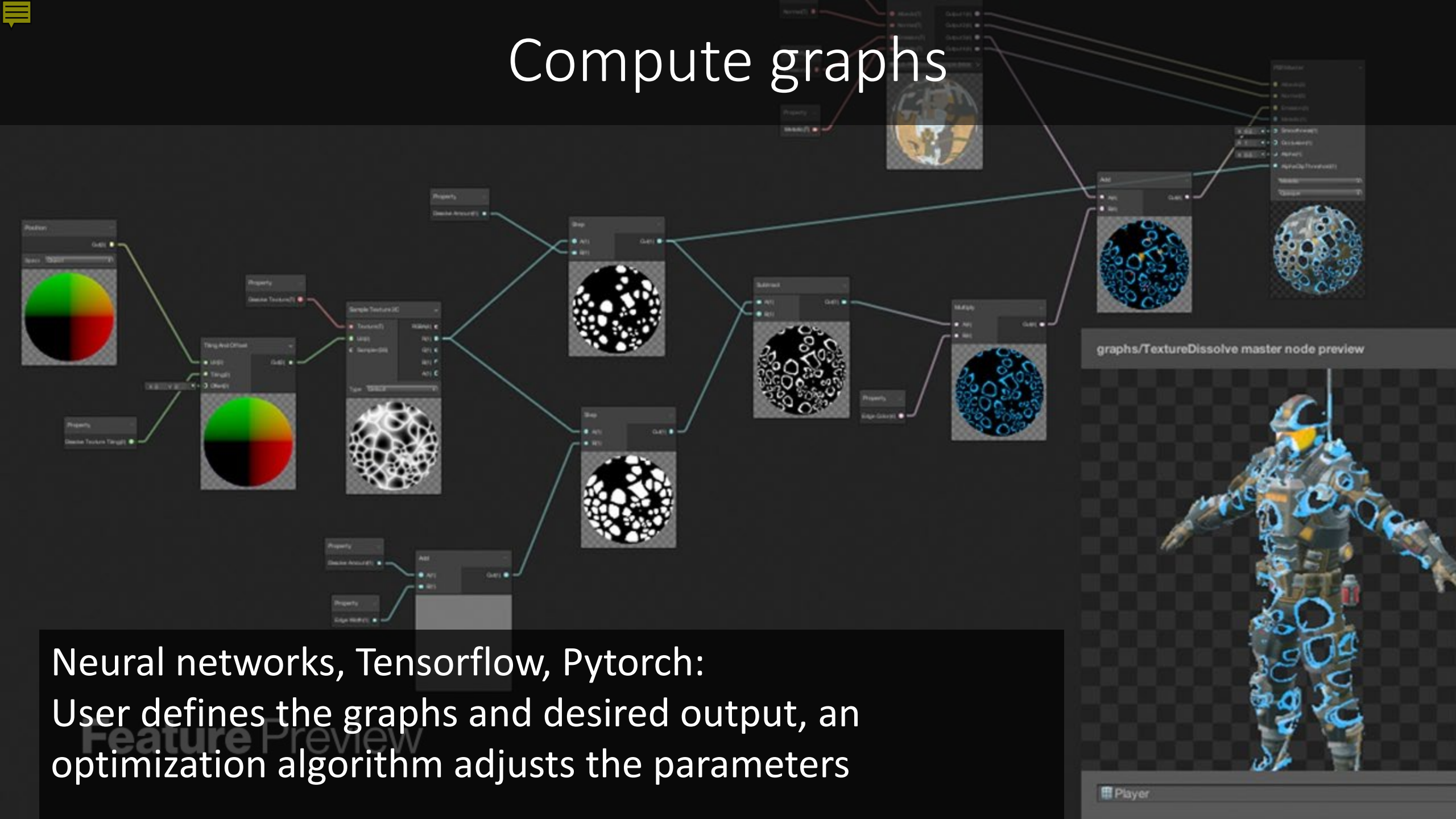


Compute graphs



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

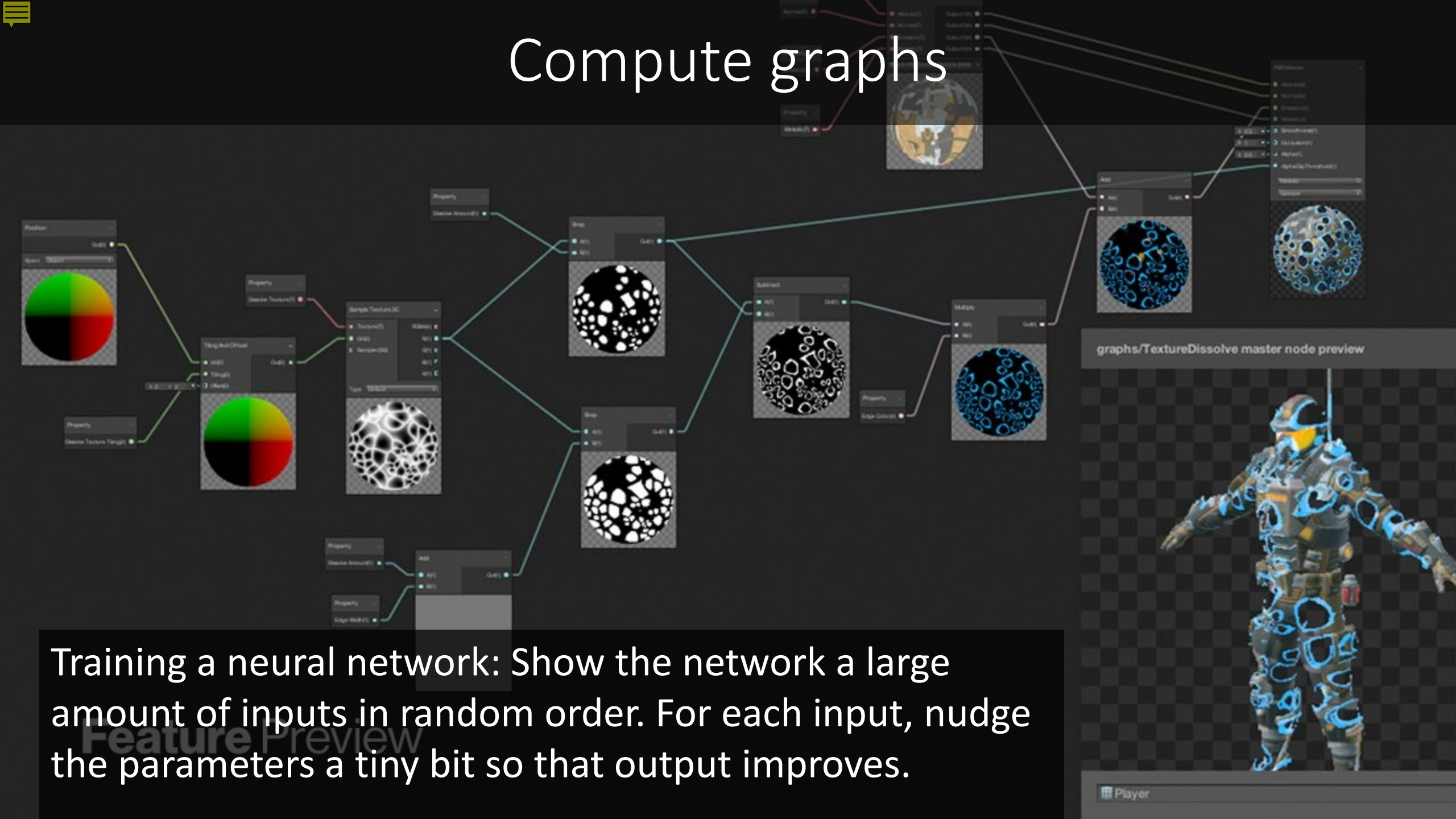
Compute graphs



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

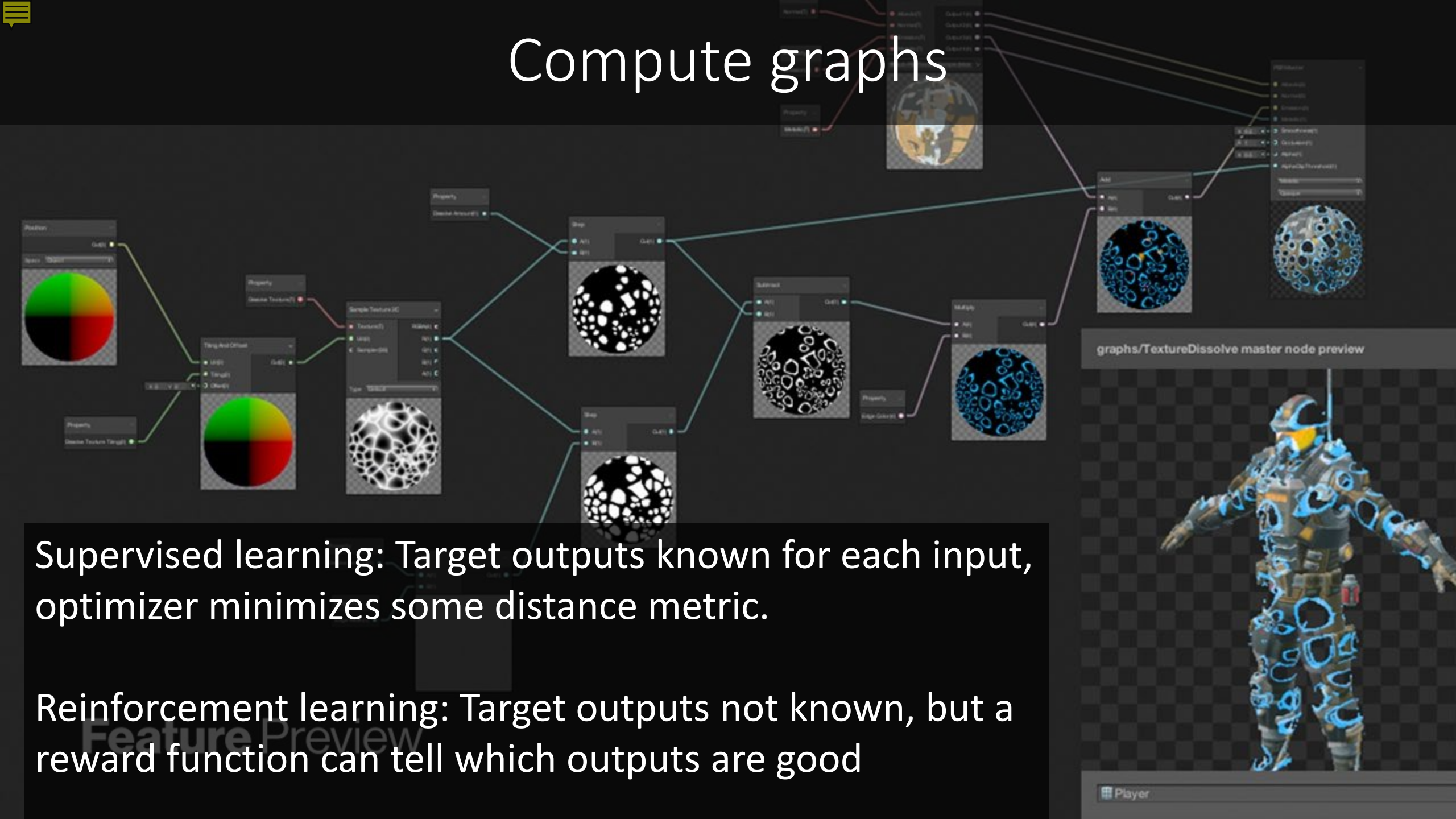
Feature Preview

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

Epoch
000,844

Learning rate: 0.003
Activation: ReLU
Regularization: None
Regularization rate: 0
Problem type: Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

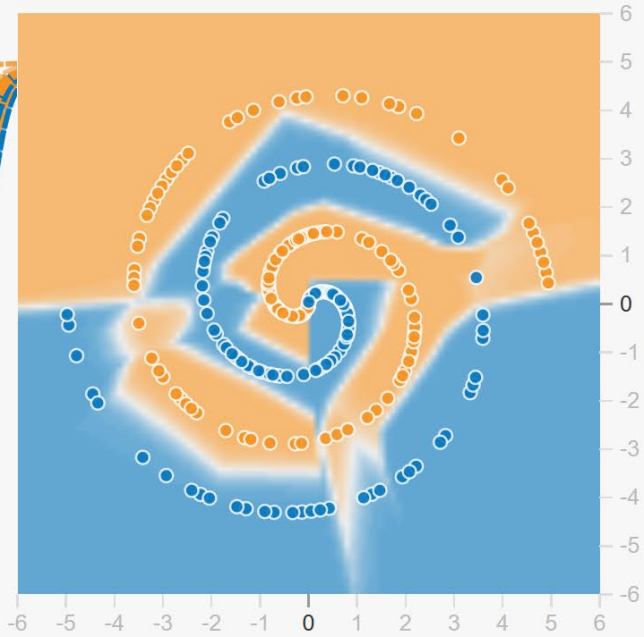
- X_1
- X_2
- X_1^2
- X_2^2
- X_1X_2
- $\sin(X_1)$
- $\sin(X_2)$

3 HIDDEN LAYERS

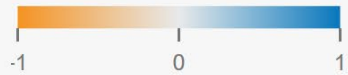
8 neurons | 8 neurons | 8 neurons

OUTPUT

Test loss 0.068
Training loss 0.010

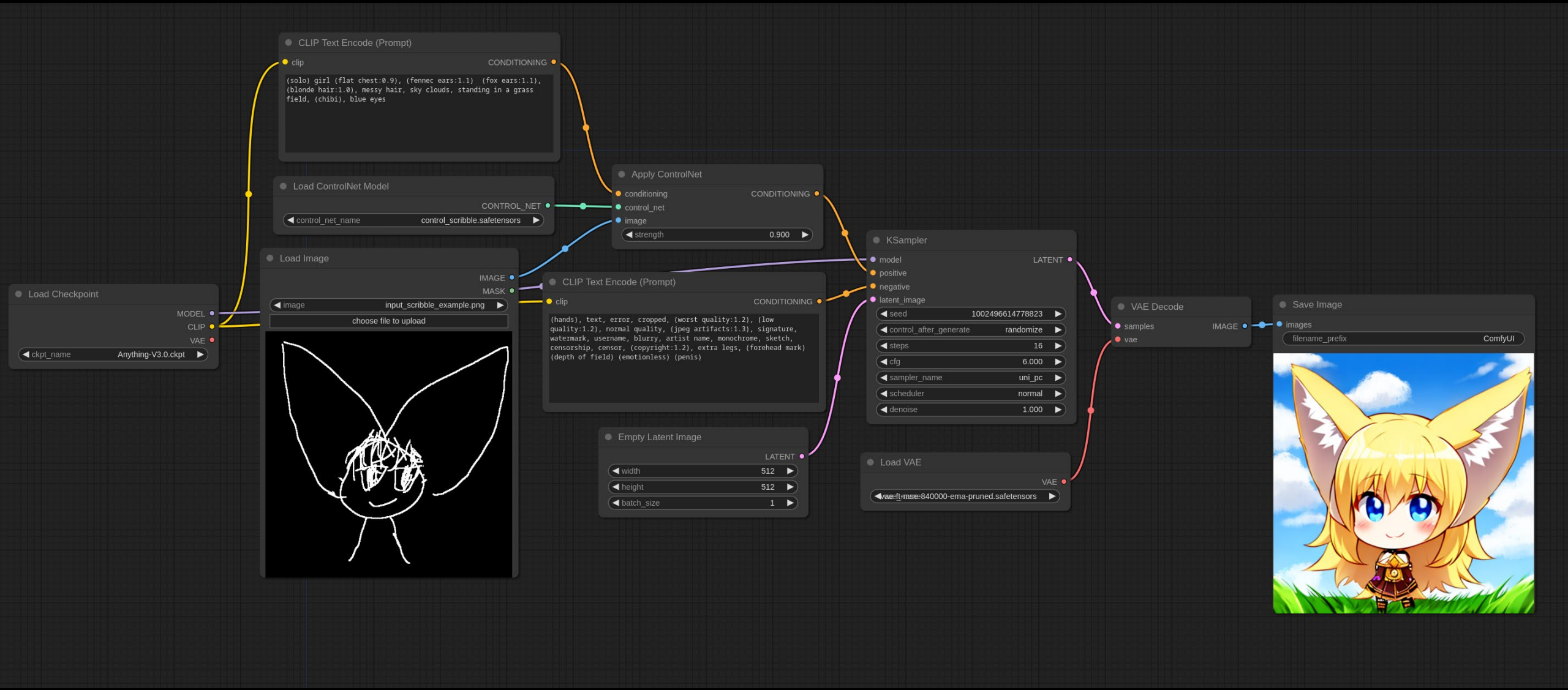


Colors shows data, neuron and weight values.



Show test data Discretize output

Compute graphs



ComfyUI: A popular interface for Stable diffusion

In neural networks, the data is arrays of numbers

- 1-dimensional arrays: Text (each number corresponding to a character), 1-channel audio
- 2-dimensional arrays: grayscale images, stereo audio
- 3-dimensional arrays: color images (array shape [width,height,channels]), multiple stereo audio files
- 4-dimensional arrays: multiple color images, [index,width,height,channels]

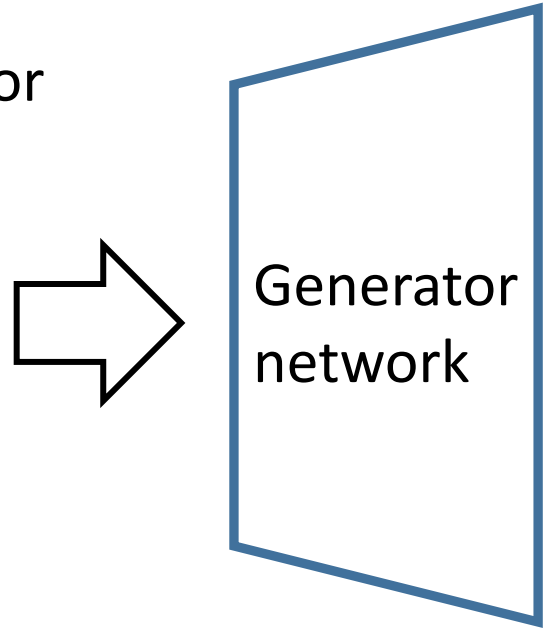
…

The compute graphs usually don’t care about the actual data types. When working with someone else’s code, the first thing is to usually check how data is formatted into arrays, and convert one’s own data accordingly, if needed.

Neural network blocks transform data from one representation to other.

Face descriptor vector
(e.g., a random one)

$[z_1, z_2, z_3 \ldots]$



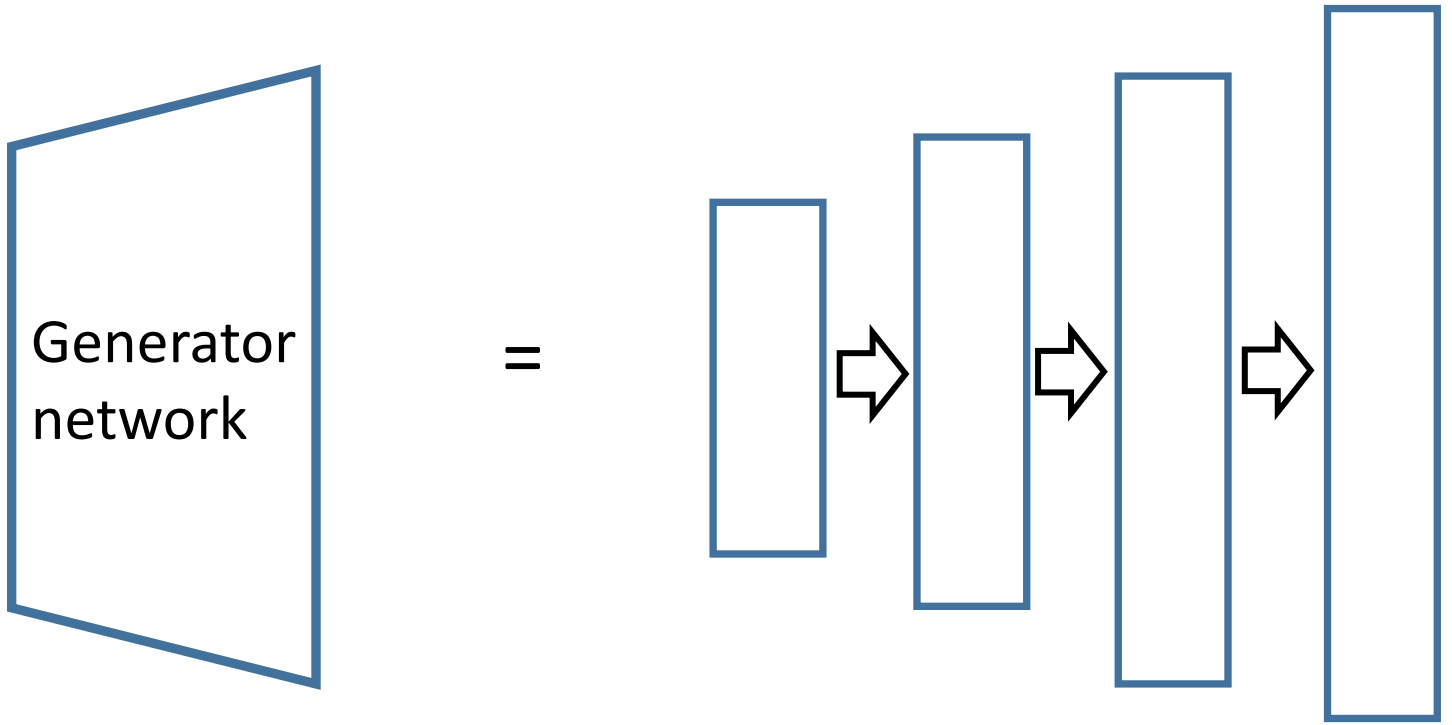
Output image: this is just an array of numbers
(pixel RGB values)

$[[r,g,b,\ldots]$
$[r,g,b,\ldots]$
$\vdots$
$\vdots$
$]$

=

Working & thinking with compute graphs

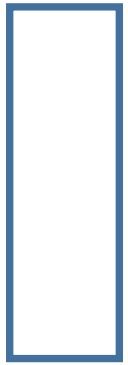
- Usually, a neural network can be broken down into layers
- What data moves between layers and how is defined by the network designer
- This is like an electronics device consisting of a few integrated circuits like a microcontroller and a Bluetooth chip.



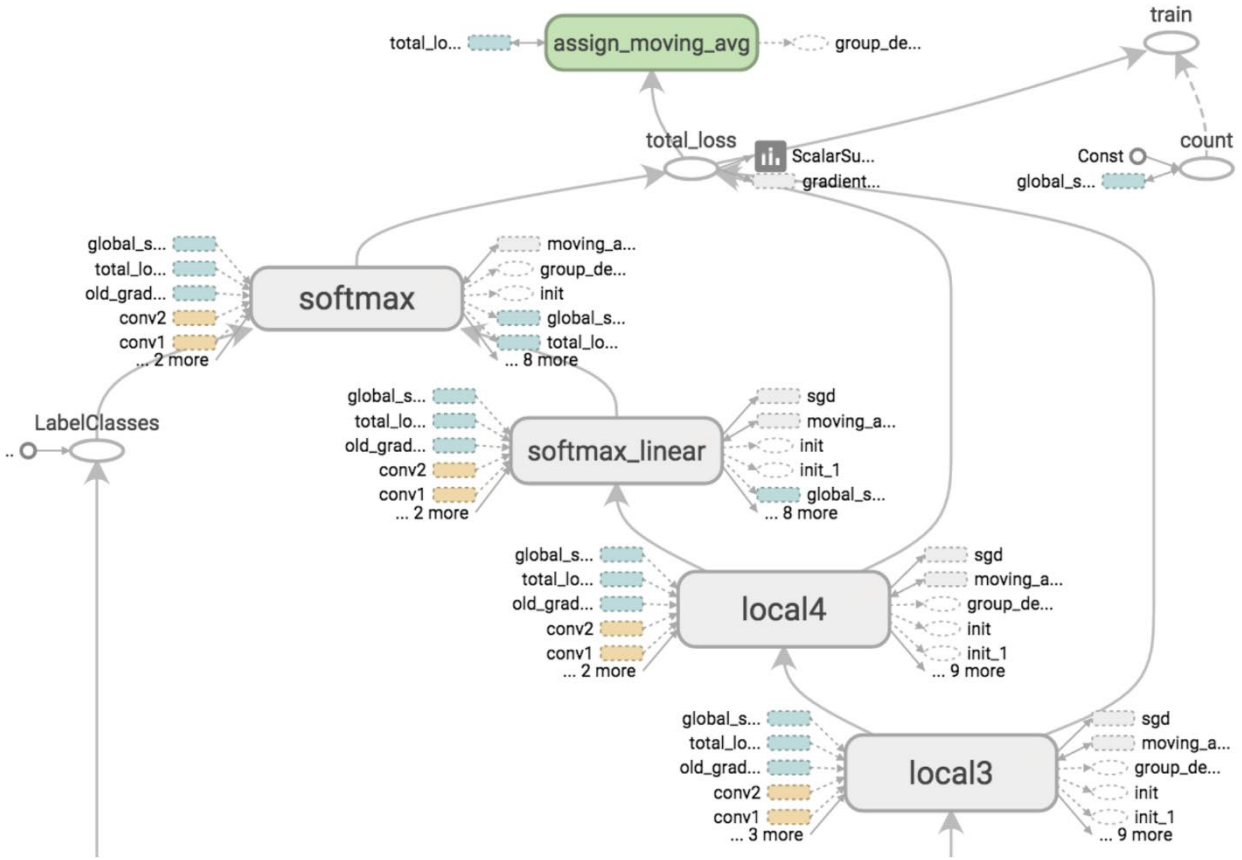
Working & thinking with compute graphs

- Layers can be further broken down into individual neurons and math operations
- This is like a CPU consisting of individual transistors

Network layer

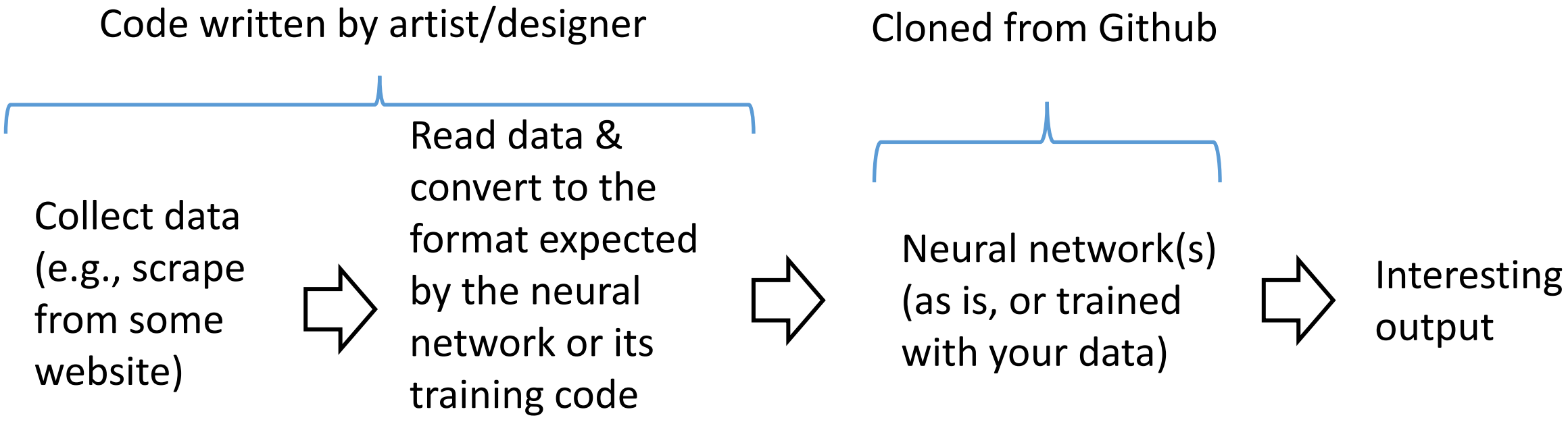


=



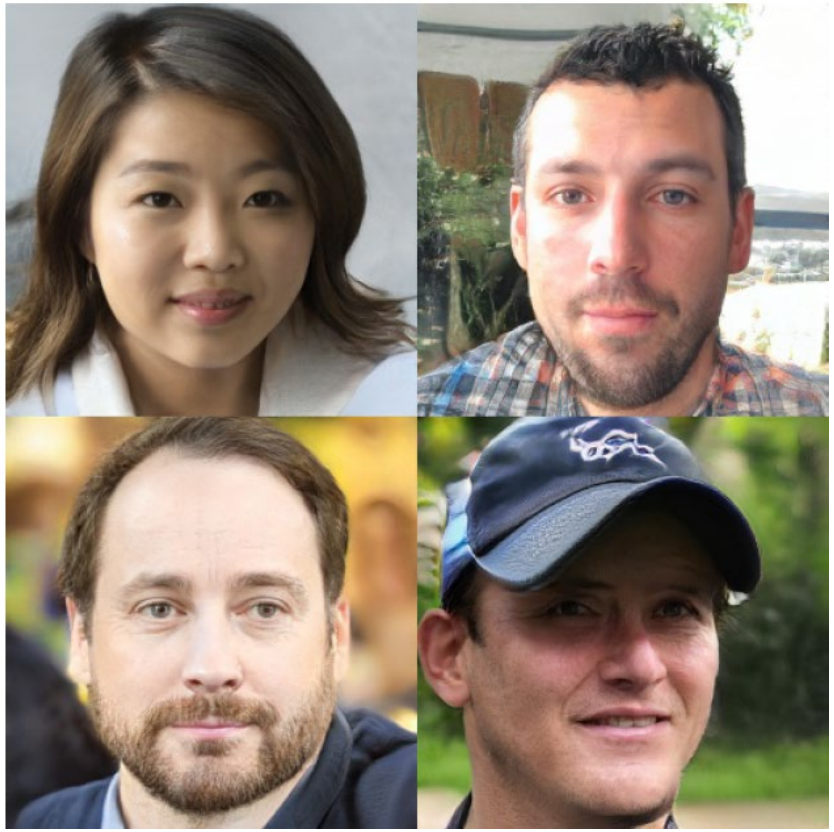
Working & thinking with compute graphs

- Usually, no need to work on the level of individual neurons, and only rarely on the level of layers
- Rather: Connecting readymade networks, like connecting an Arduino board to a movement tracking sensor
- Main questions: Which building blocks to use, where to get data, how to format it correctly?



Working & thinking with compute graphs

- Case study: StyleGAN 2 “circuit bending”



NVIDIA’s pretrained StyleGAN2-ada neural network available through Github

The same network “broken” with unrelated images of rocks, trees, foliage

Working & thinking with compute graphs

These results:

- A pretrained (readymade) StyleGAN2-ada face generator from NVIDIA's Github
- Custom Python code: Scrape images of rocks, foliage etc. using Google Image search API. Package the images to a .zip file expected by StyleGAN (as explained on the Github page)
- Training the network just the right amount of time, using Python command line tools provided by NVIDIA

Working & thinking with compute graphs

Typical skills needed:

- Python network and file I/O
- Working with multidimensional number arrays using Python's Numpy package (basically all ML & AI builds on Numpy)
- Understanding what kinds of neural networks and processing blocks there are, what they can do, and what kind of data goes in and out

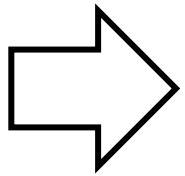
Latent spaces

Typically, generative models try to learn an *N*-dimensional ”latent space” where all positions (vectors of *N* numbers) in some region map to some meaningful image or sound, when passed through a generator or decoder network.

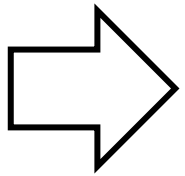
Output image: this is just an array of numbers (pixel RGB values)

Latent space vector

$\mathbf{z} = [z_1, z_2, z_3 \ldots]$



Generator network

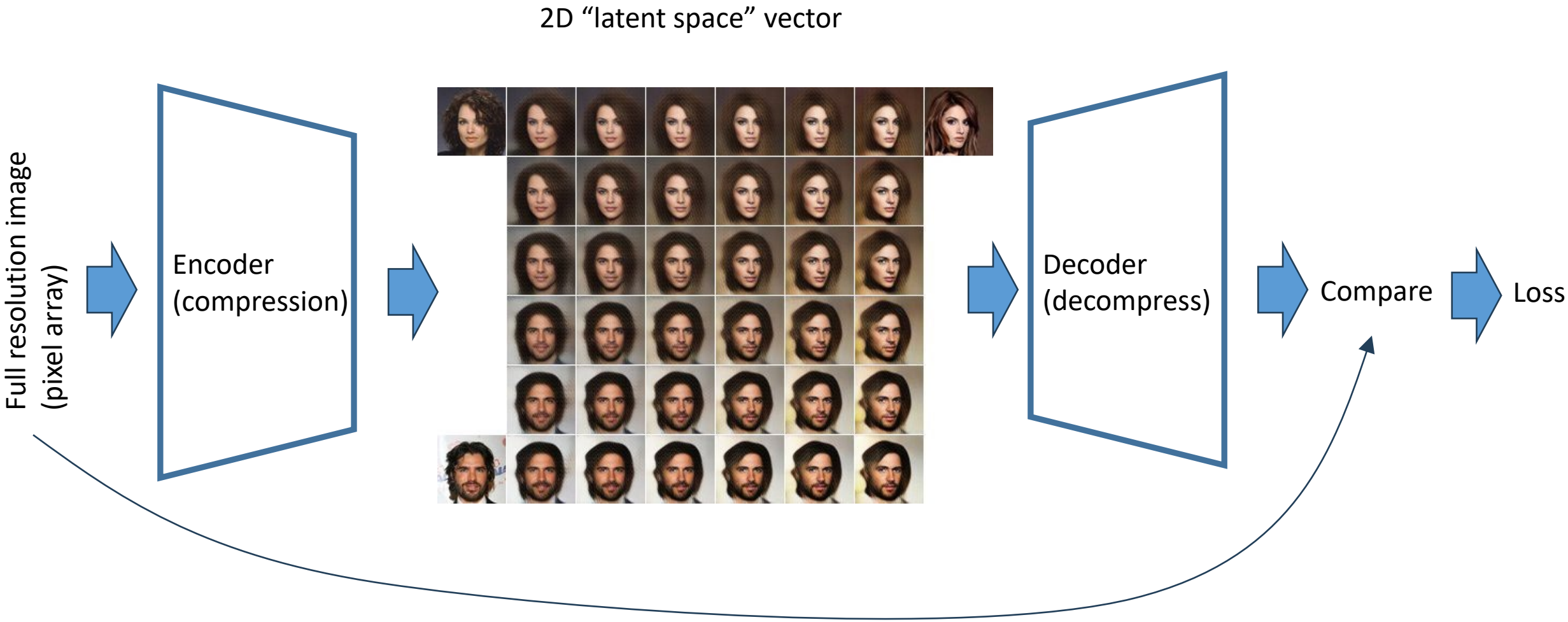


$$\begin{bmatrix} [r,g,b,\ldots] \\ [r,g,b,\ldots] \\ \vdots \\ \vdots \end{bmatrix} =$$

Latent spaces

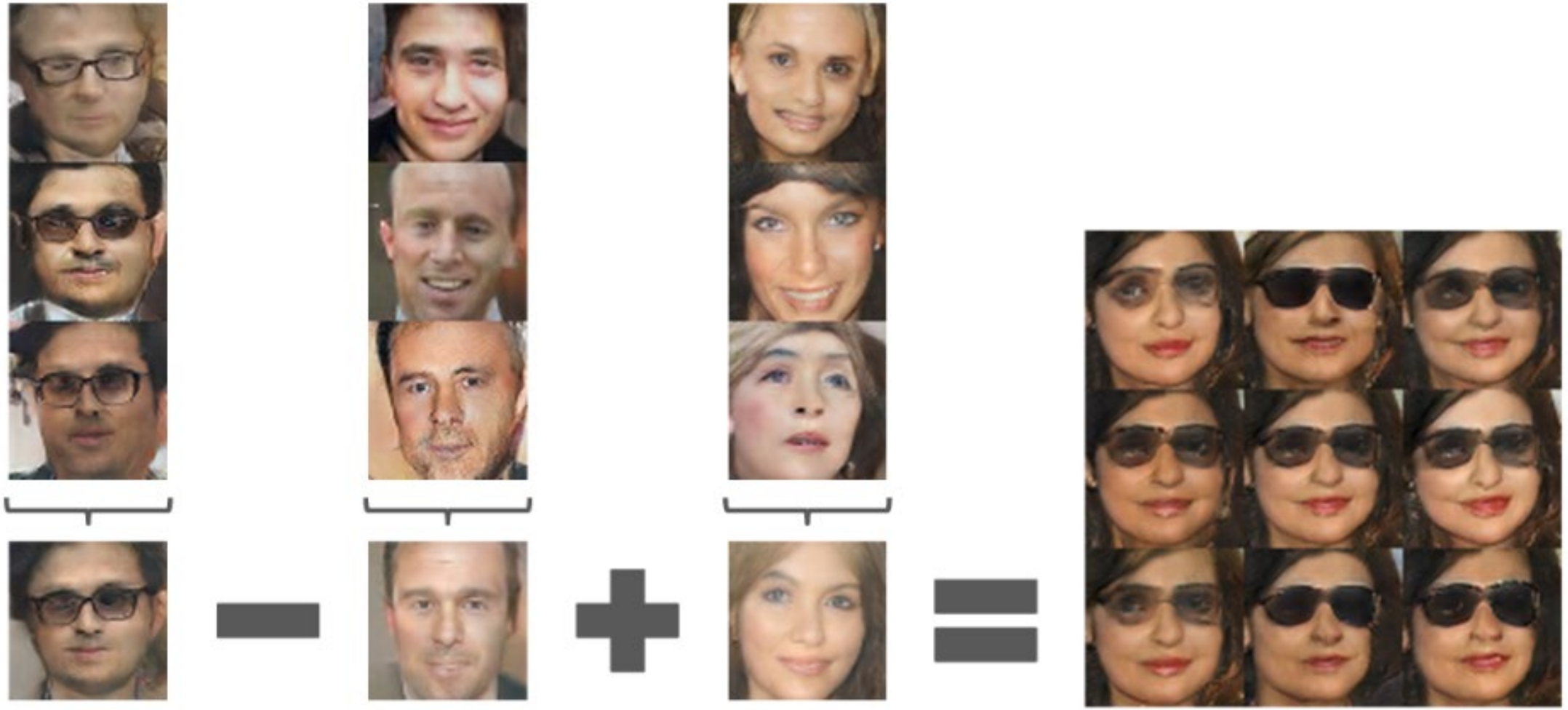
- Depending on context, may also be called ”*embedding space*”
- If the latent dimensionality is small, the encoder performs data compression.
- Efficient compression requires the network to learn to represent the underlying latent variables of a the ”generative process” the observed images originate from, e.g., gender, age, and orientation of faces.
- Latent space directions often have semantic meaning: Along some axis, male faces might be to the left, and female to the right

Training an autoencoder with a 2D ”bottleneck”



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

Latent space math

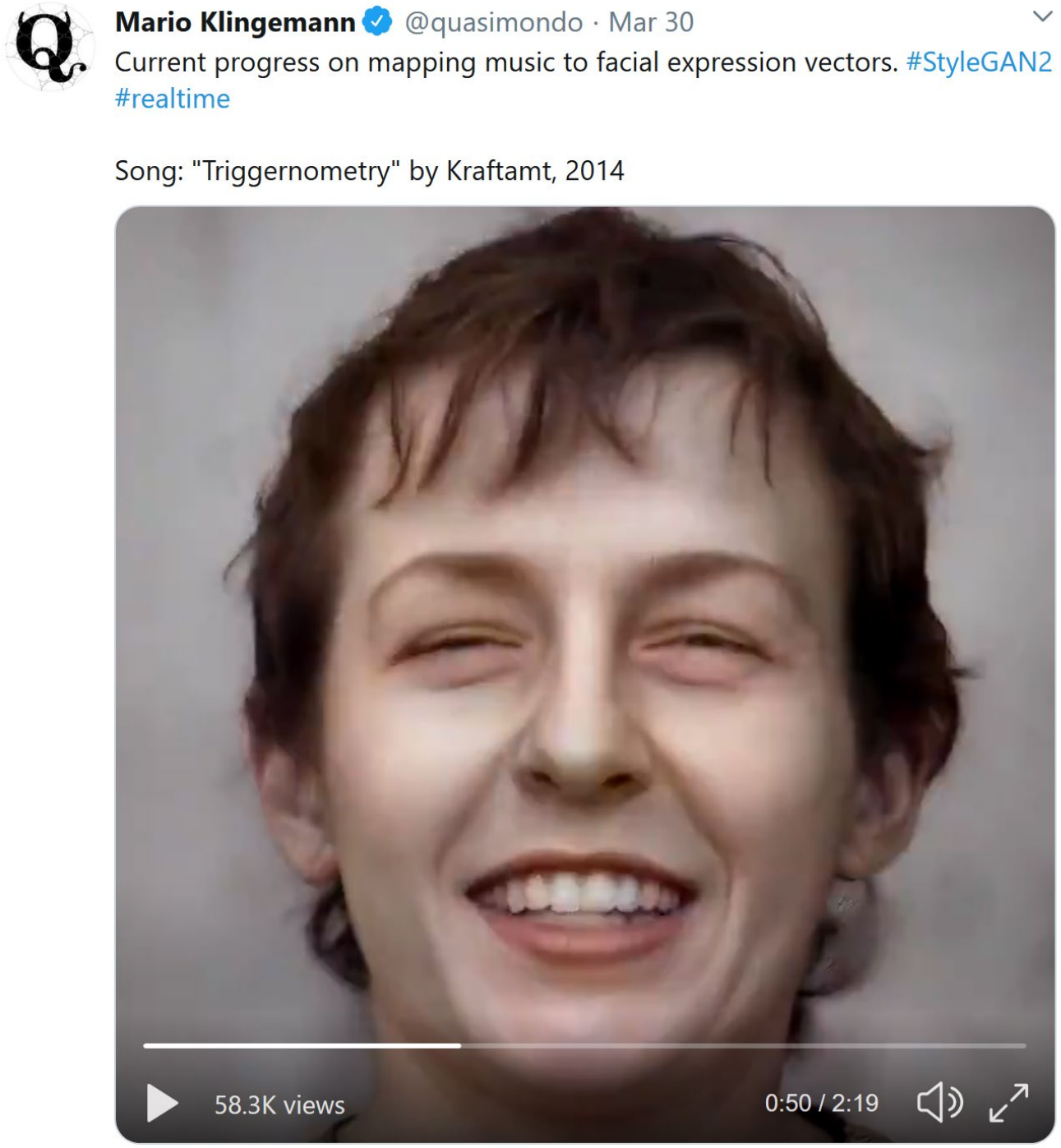


https://arxiv.org/pdf/1511.06434.pdf

Pixel-space math

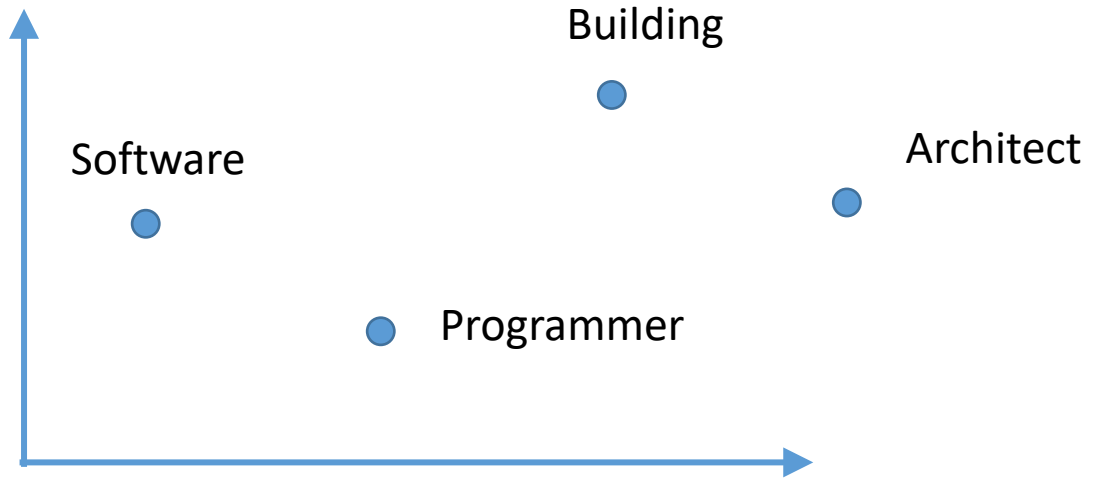
Music visualization using StyleGAN 2 latent space

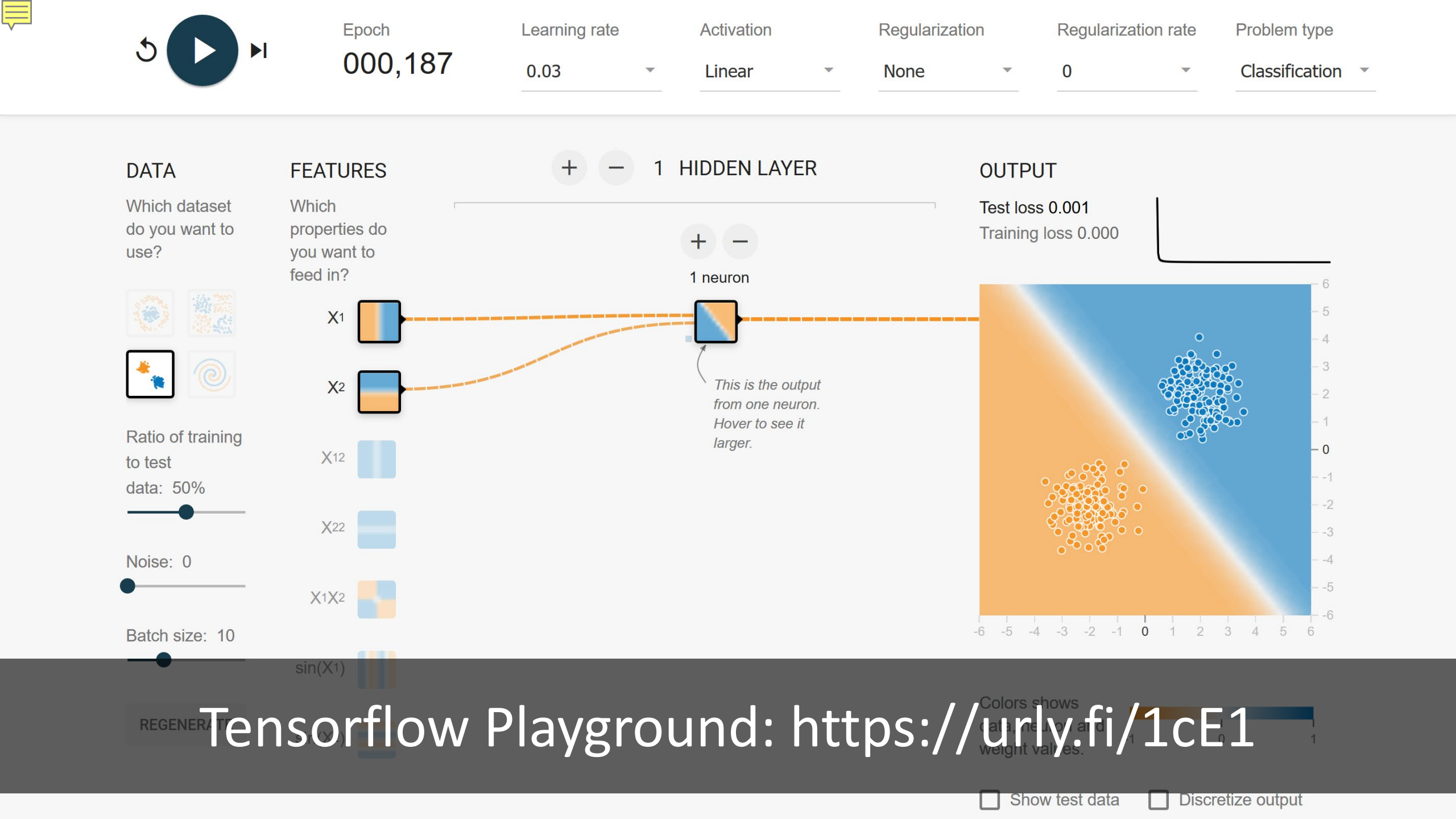
https://twitter.com/quasimondo/status/1244562140217905153?s=20

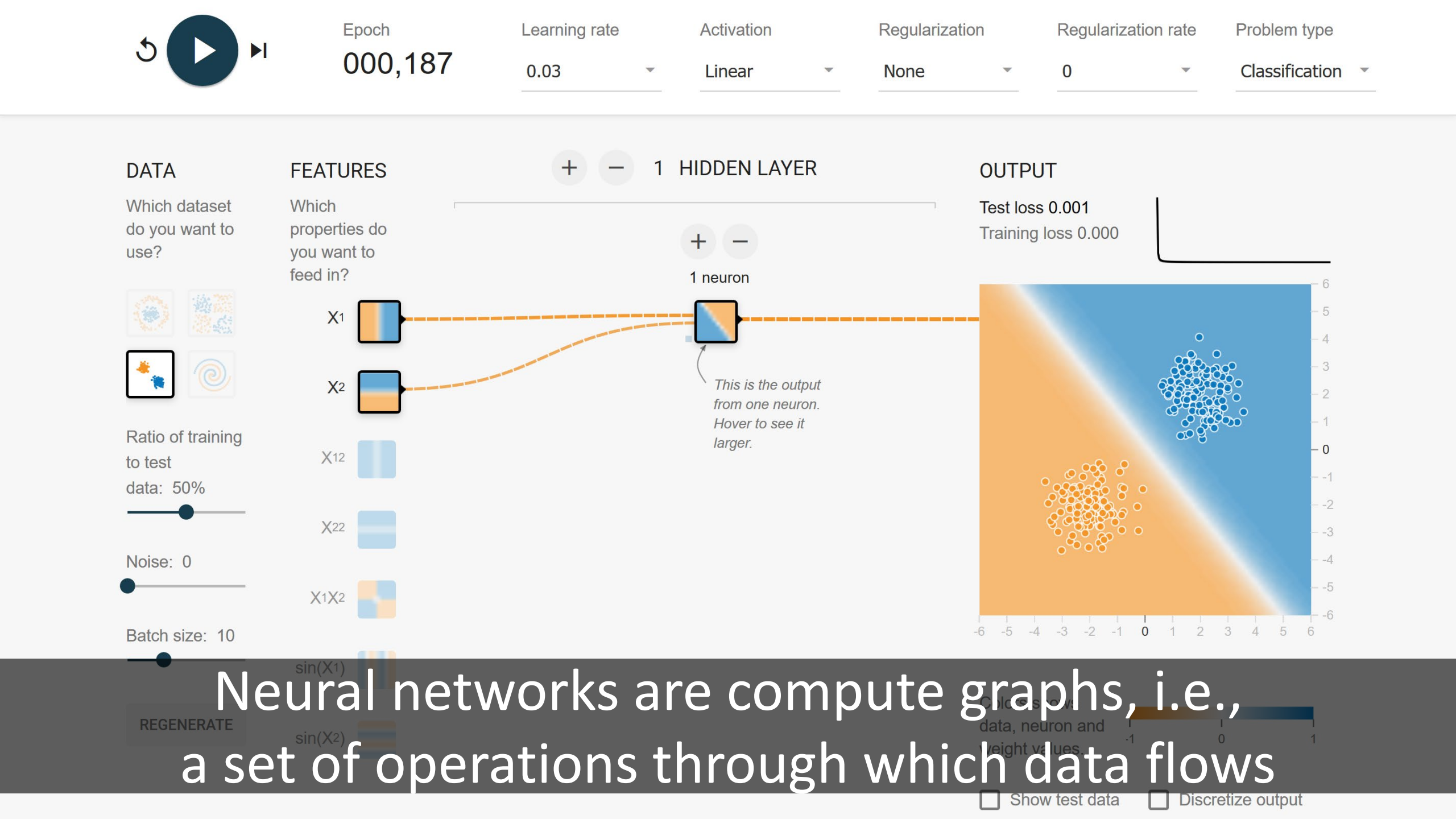


Same with words

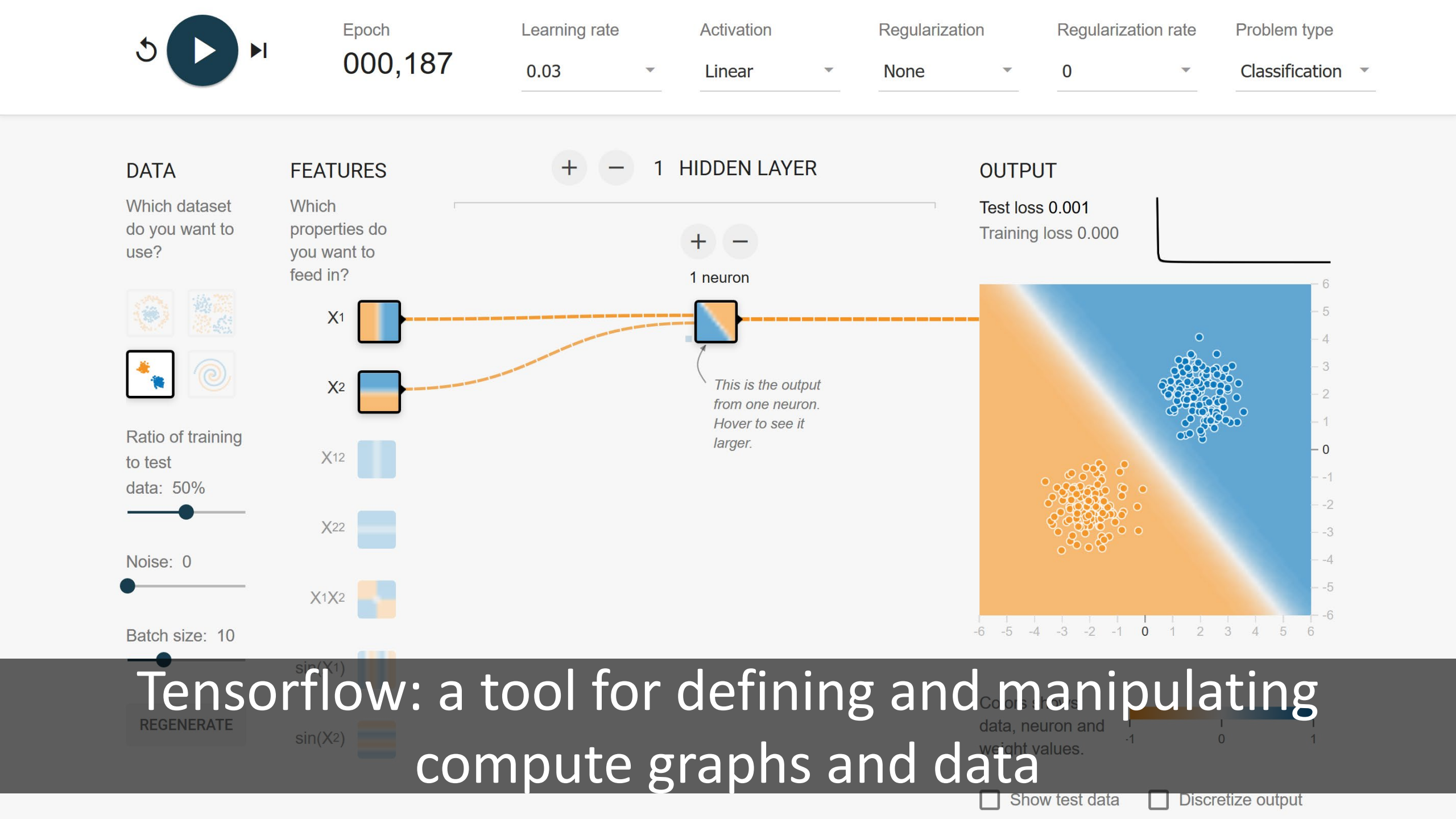
- King - Man + Woman = Queen
- Note that this is equivalent to King - Man = Queen - Woman
- Software - Building + Architect = Programmer
- Precomputed dictionaries of such embeddings are available, e.g., Fasttext (https://fasttext.cc/)
- OpenAI API and Sentence Transformers provide easy embedding of longer texts: https://platform.openai.com/docs/guides/embeddings, https://www.sbert.net/docs/hugging_face.html#using-hugging-face-models



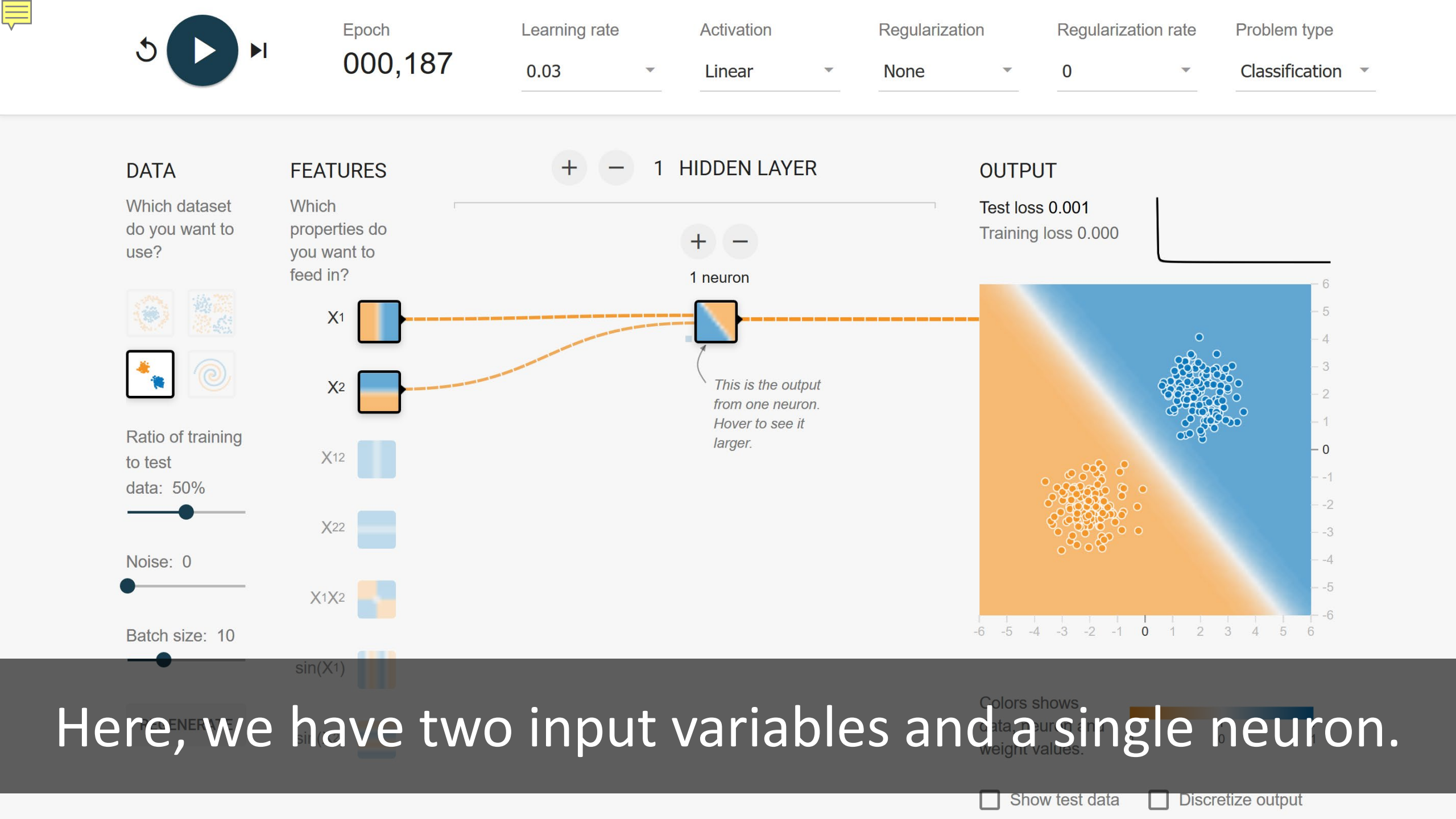

Epoch
000,187
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X12
X22
X1X2
sin(X1)
1 HIDDEN LAYER
1 neuron
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.001
Training loss 0.000
Colors shows data, neuron and weight values.
Show test data
Discretize output
Tensorflow Playground: https://urly.fi/1cE1

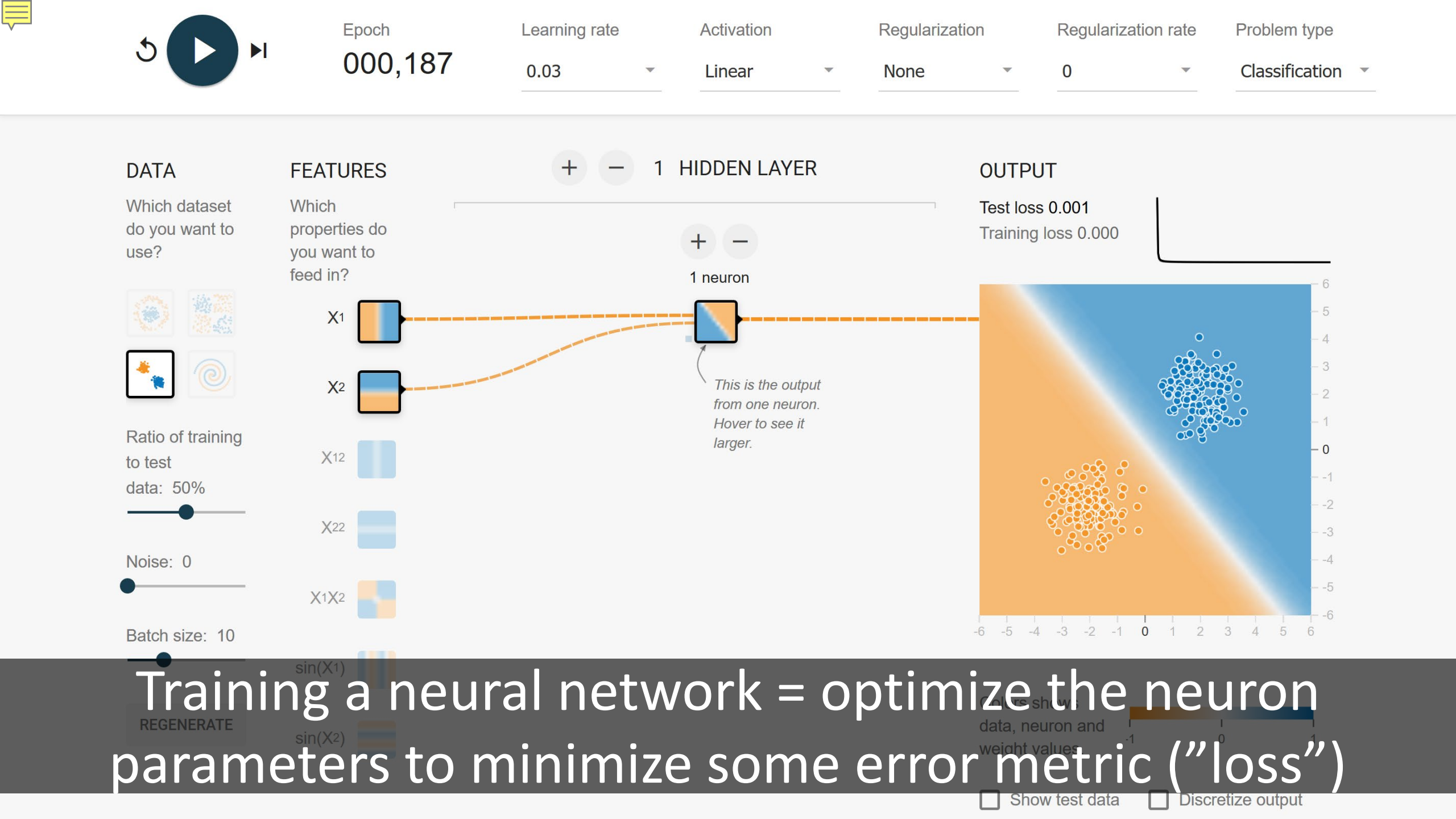


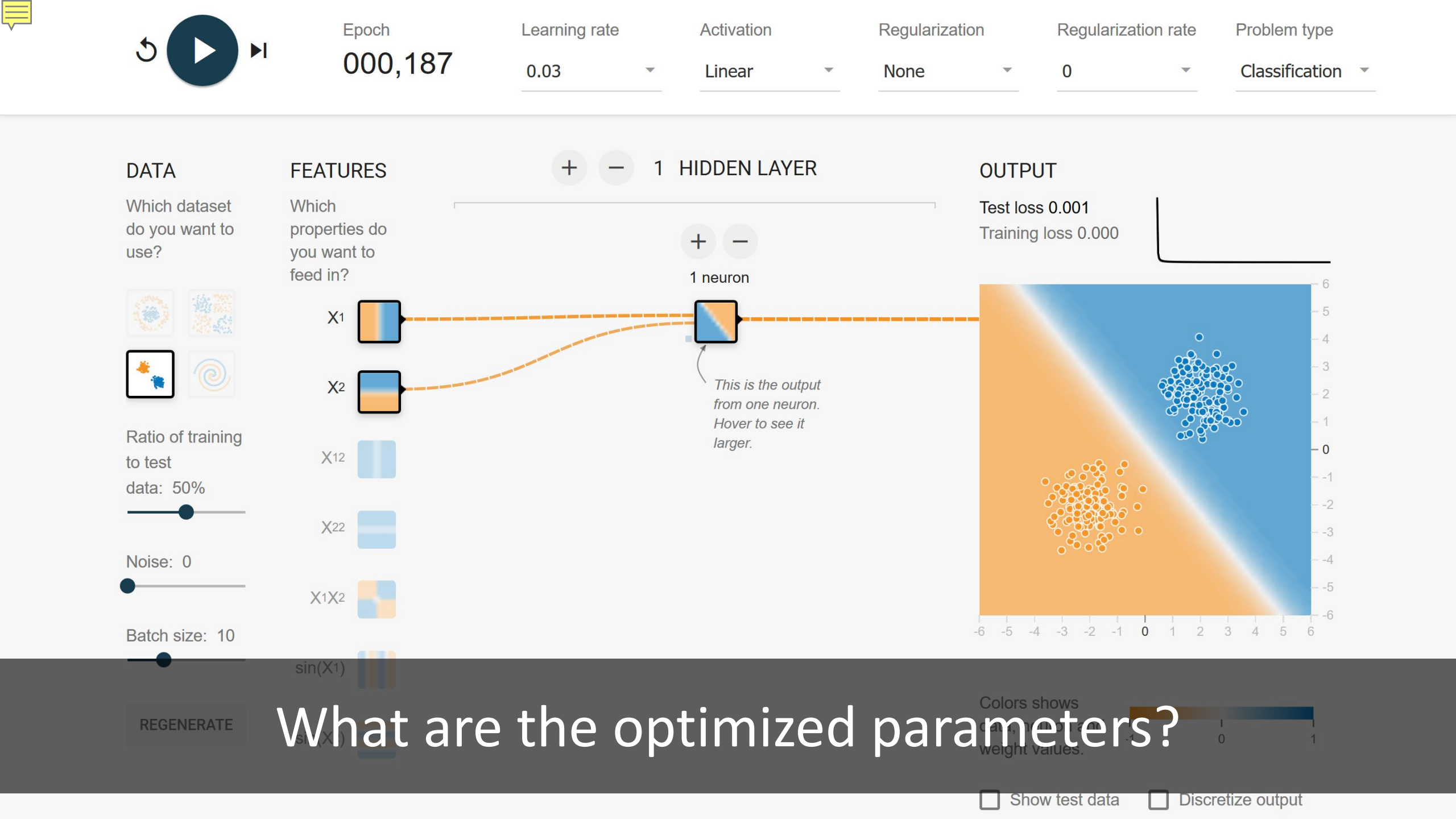
Neural networks are compute graphs, i.e.,
a set of operations through which data flows



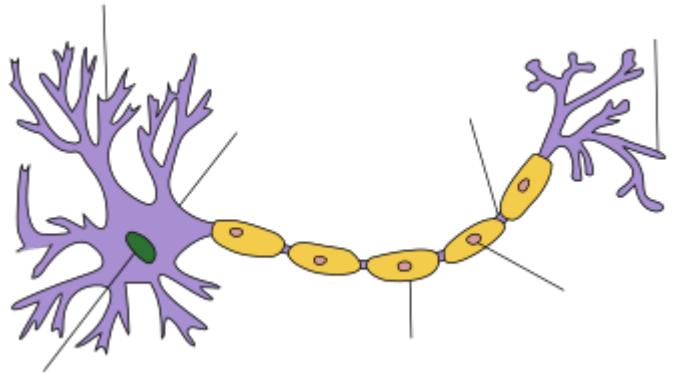
Tensorflow: a tool for defining and manipulating compute graphs and data

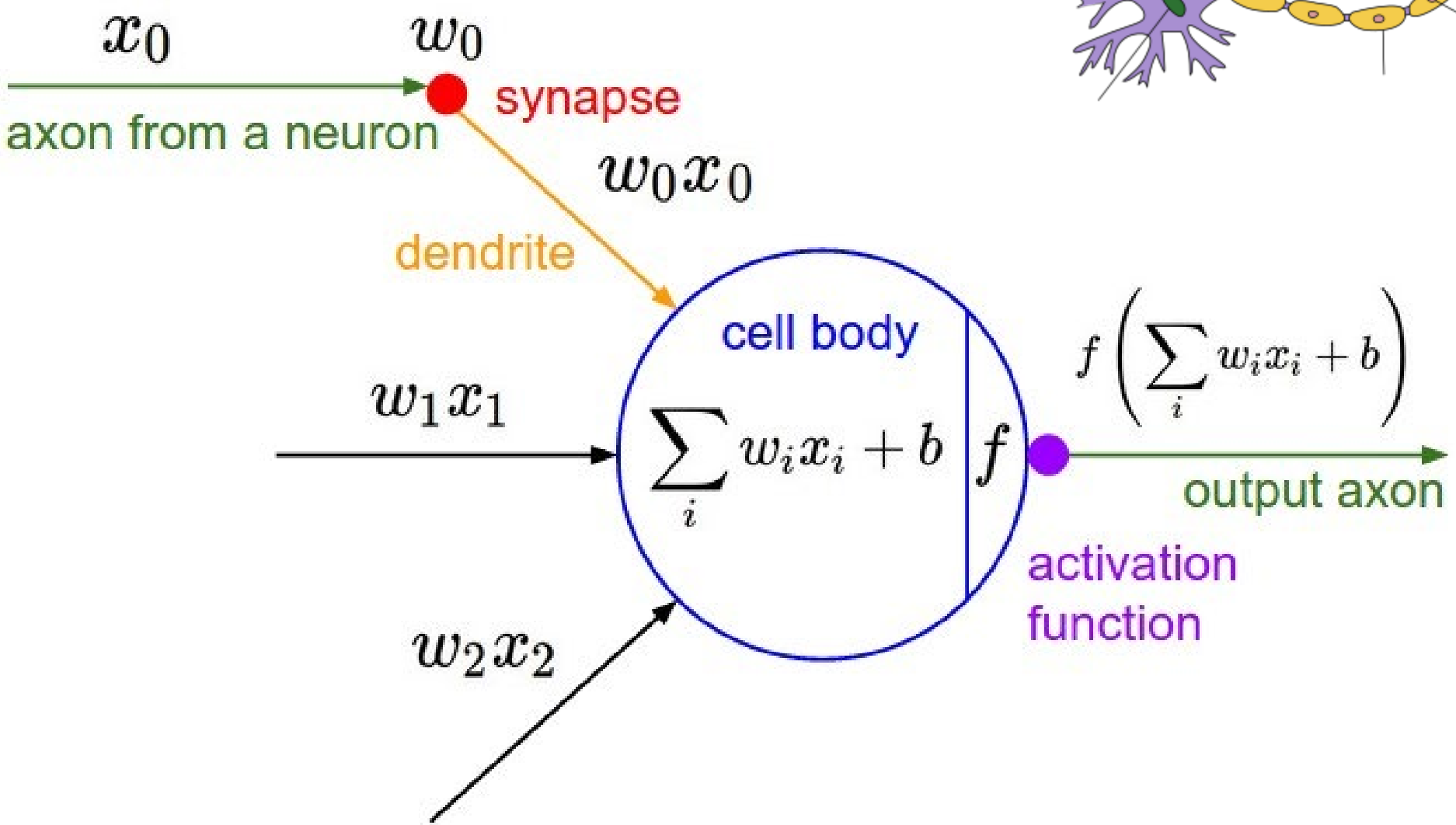

Epoch
000,187
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
FEATURES
Which properties do you want to feed in?
X1
X2
X12
X22
X1X2
sin(X1)
1 HIDDEN LAYER
1 neuron
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.001
Training loss 0.000
Colors shows data, neuron and weight values.
Show test data
Discretize output
Here, we have two input variables and a single neuron.


Epoch
000,187
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
1 HIDDEN LAYER
1 neuron
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.001
Training loss 0.000
Colors shows data, neuron and weight values.
Show test data
Discretize output
Training a neural network = optimize the neuron parameters to minimize some error metric (”loss”)

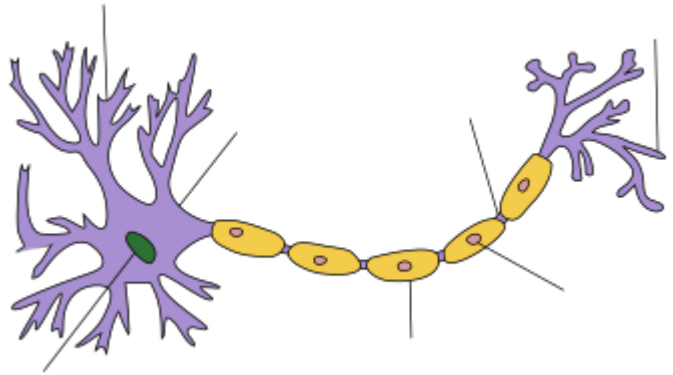

Epoch
000,187
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
X1
X2
X12
X22
X1X2
sin(X1)
1 HIDDEN LAYER
1 neuron
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.001
Training loss 0.000
Colors shows data, neuron and weight values.
Show test data
Discretize output
What are the optimized parameters?

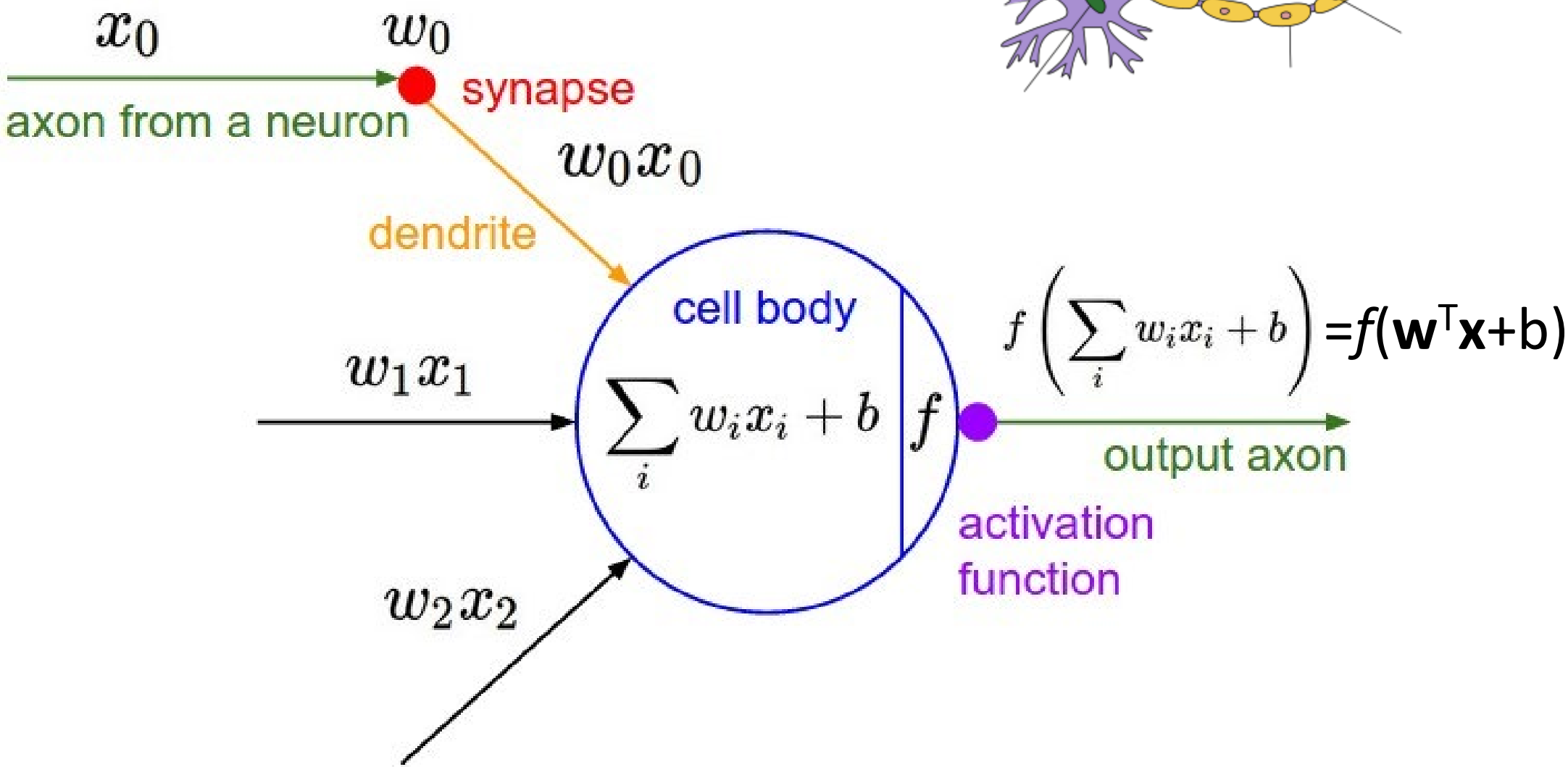
An artificial neuron



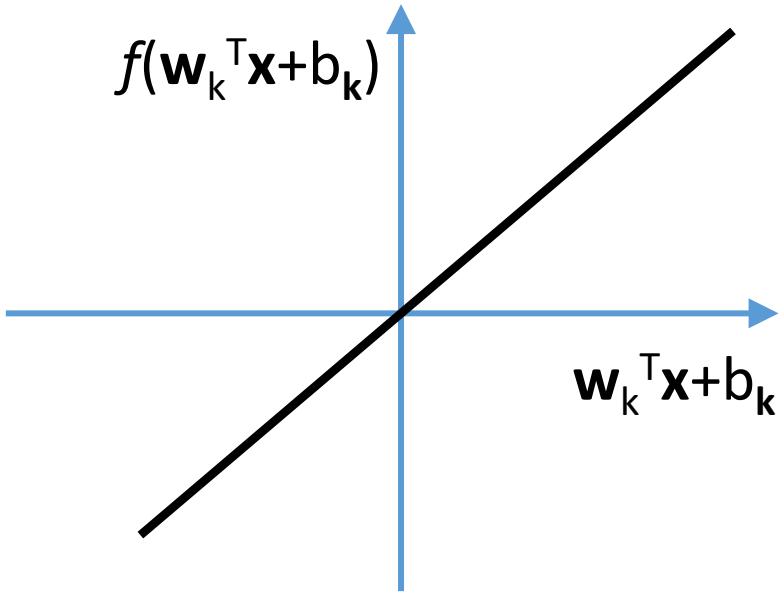


An artificial neuron

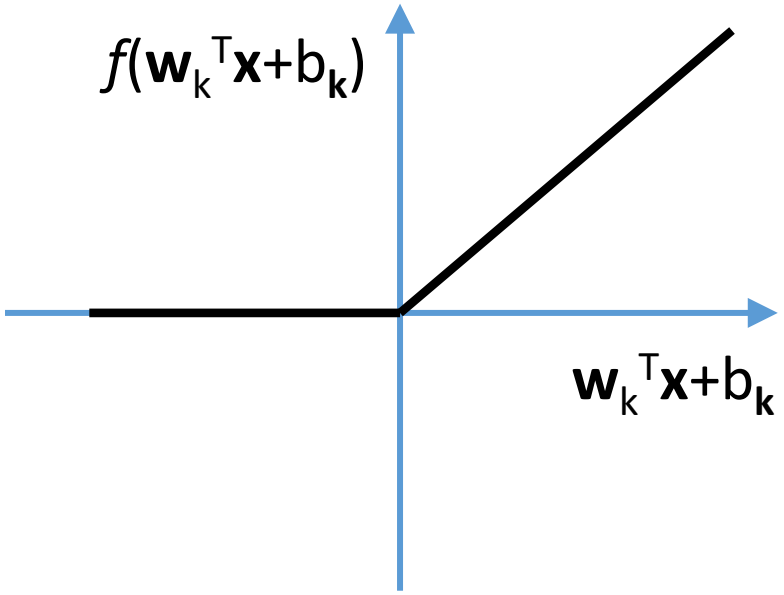




Linear activation



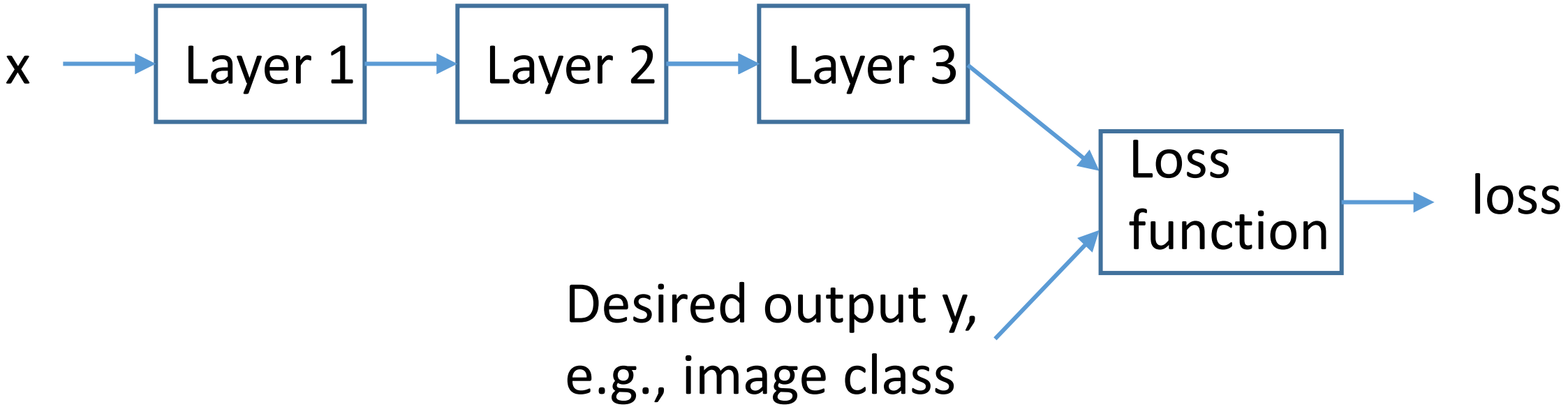
ReLU activation



ReLU: The neuron only "fires" when $\mathbf{w}_k^T\mathbf{x}+b_k$ is positive. Otherwise, the output is zero.

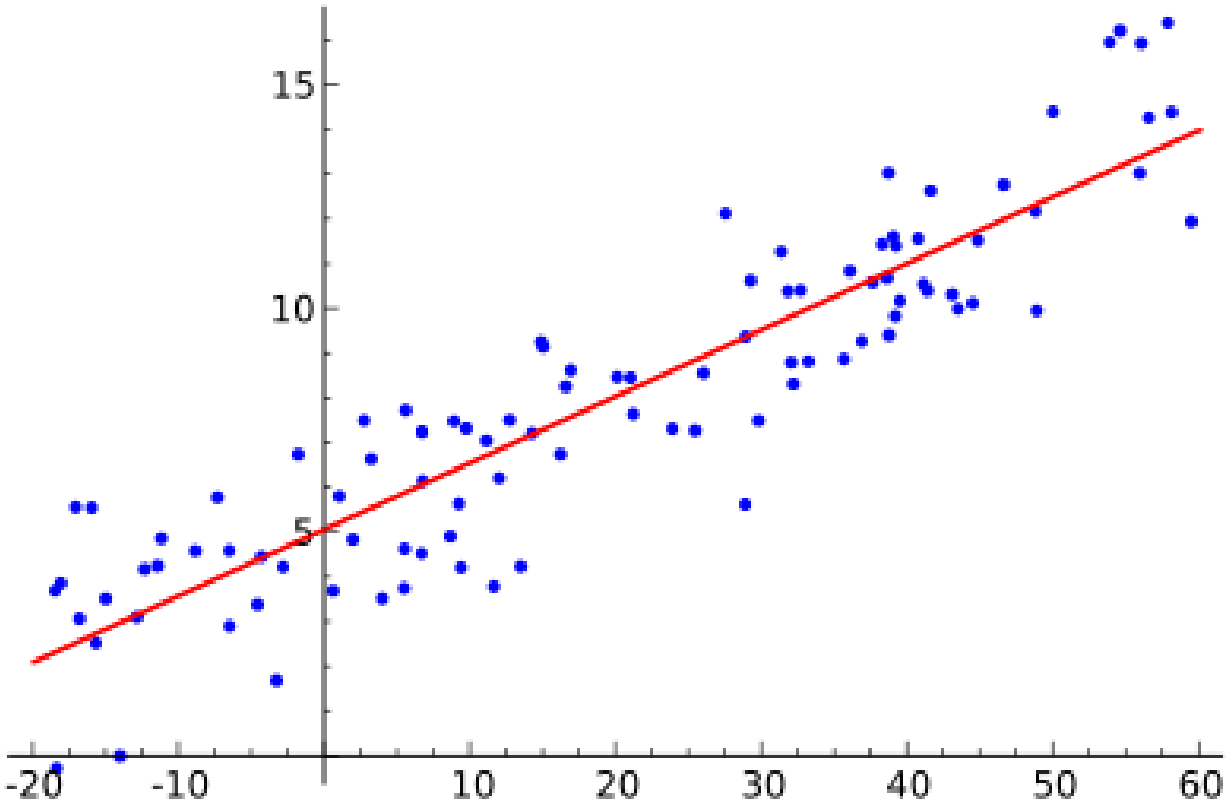
Training = optimization, minimizing loss

1. Feed a random *minibatch* of input data x through the network
2. Compute the loss function for each input
3. Change the weights such that loss decreases, on average
4. Rinse & repeat!



Common loss functions

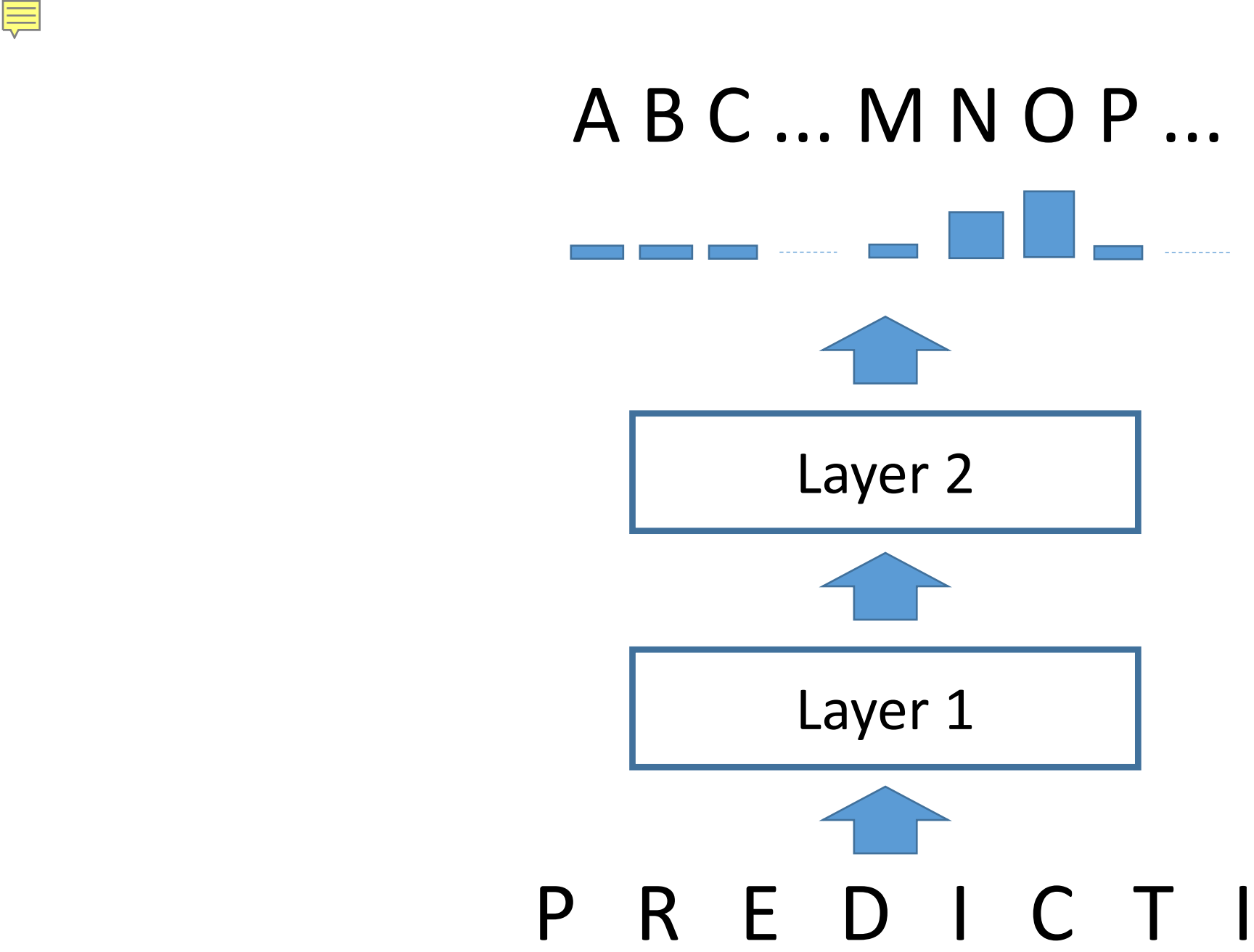
- Sum of squared errors, i.e., differences between network output variables and desired output variables



Common loss functions

- Softmax cross-entropy, in classification where one output neuron for each class

```
def cross_entropy(X,y):
    """
    X is the output from fully connected layer (num_examples x num_classes)
    y is labels (num_examples x 1)
    """
    m = y.shape[0]
    p = softmax(X)
    log_likelihood = -np.log(p[range(m),y])
    loss = np.sum(log_likelihood) / m
    return loss
```


A B C … M N O P …
Layer 2
Layer 1
P R E D I C T I

Contents

- Preliminaries: compute graphs, artificial neurons, activation functions, loss functions, latent spaces
- **Understanding nonlinear activations**
- Image generator architectures
- Quality metrics: FID, Precision & Recall

Epoch
000,071

Learning rate: 0.03

Activation: Linear

Regularization: None

Regularization rate: 0

Problem type: Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

- X_1
- X_2
- X_1^2
- X_2^2
- X_1X_2
- $\sin(X_1)$
- $\sin(X_2)$

1 HIDDEN LAYER

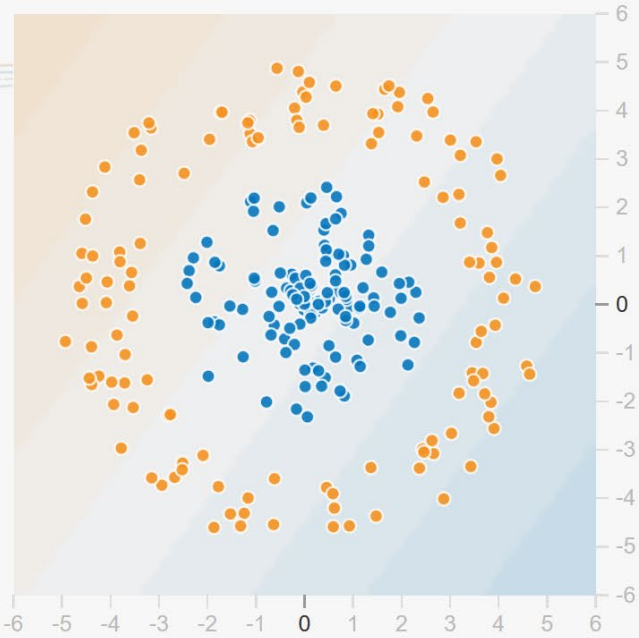
4 neurons

This is the output from one neuron. Hover to see it larger.

OUTPUT

Test loss 0.513

Training loss 0.498



Colors shows data, neuron and weight values.

-1 0 1

Show test data

Discretize output

Epoch
000,071

Learning rate
0.03

Activation
Linear

Regularization
None

Regularization rate
0

Problem type
Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

- X_1
- X_2
- X_1^2
- X_2^2
- X_1X_2
- $\sin(X_1)$
- $\sin(X_2)$

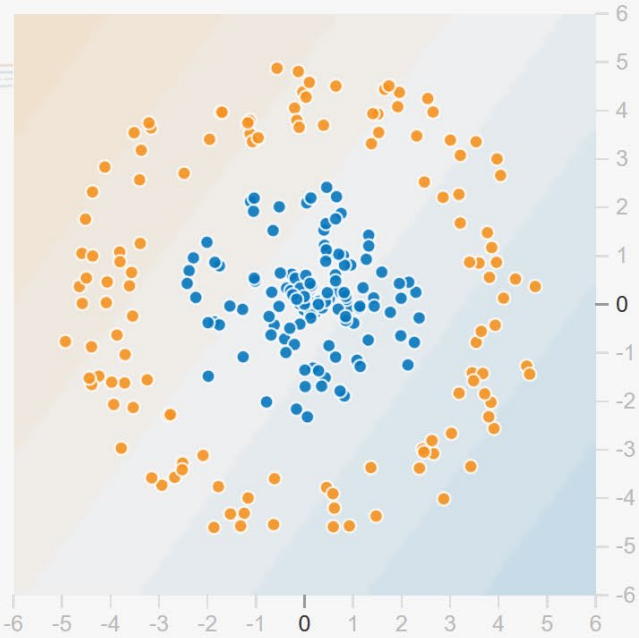
1 HIDDEN LAYER

4 neurons

This is the output from one neuron. Hover to see it larger.

OUTPUT

Test loss 0.513
Training loss 0.498



Colors shows data, neuron and weight values.

-1 0 1

Show test data Discretize output

Epoch
000,124

Learning rate
0.03

Activation
ReLU

Regularization
None

Regularization rate
0

Problem type
Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 21

REGENERATE

FEATURES

Which properties do you want to feed in?

X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)

1 HIDDEN LAYER

4 neurons

This is the output from one neuron. Hover to see it larger.

OUTPUT

Test loss 0.025
Training loss 0.013

Colors shows data, neuron and weight values.

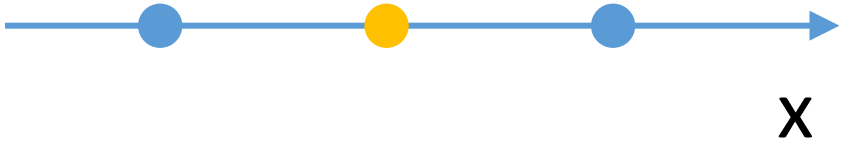
-1 0 1

Show test data

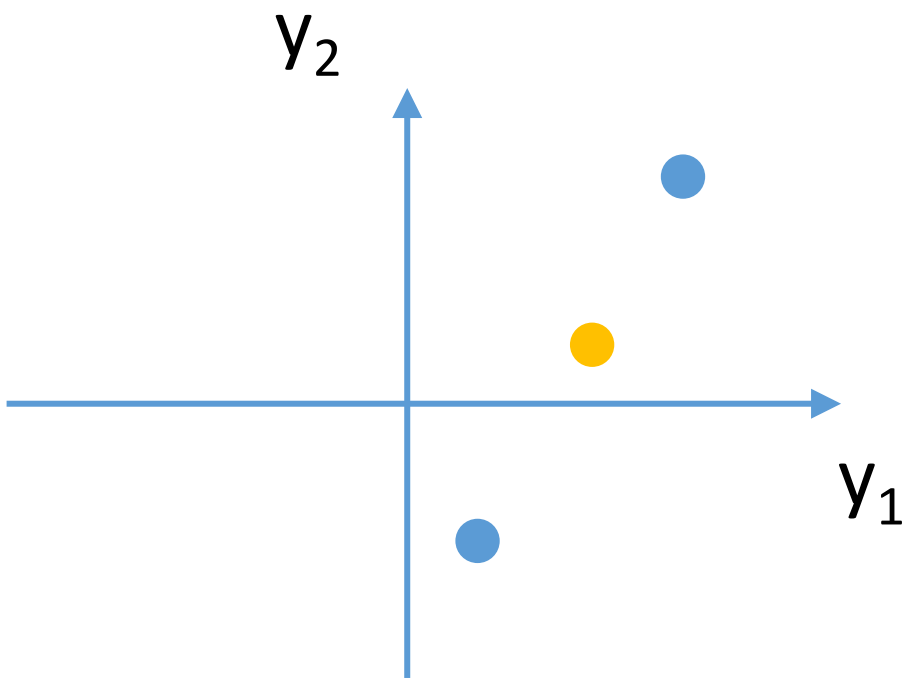
Discretize output

The effect of nonlinearity

1D input

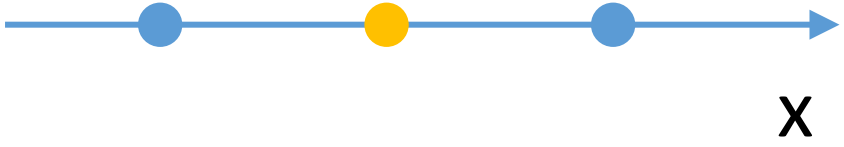


2D output of 2 linear neurons



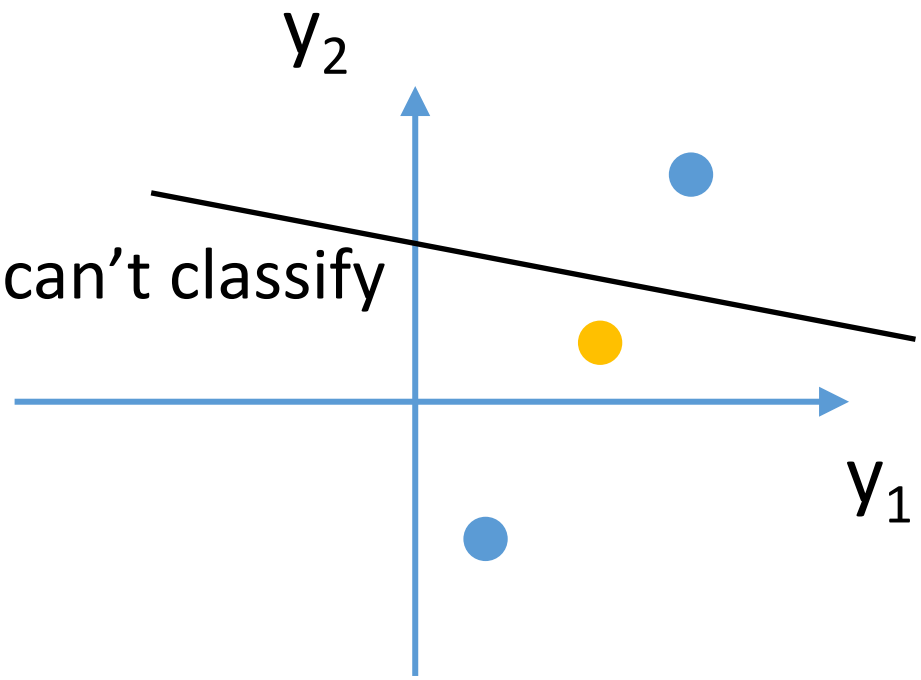
The effect of nonlinearity

1D input



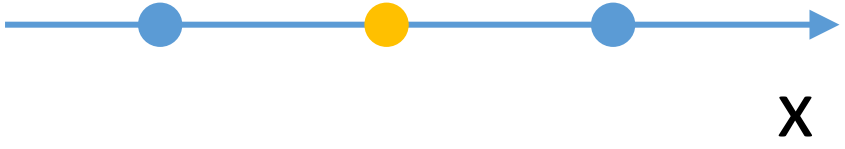
Output neuron can’t classify

2D output of 2 linear neurons

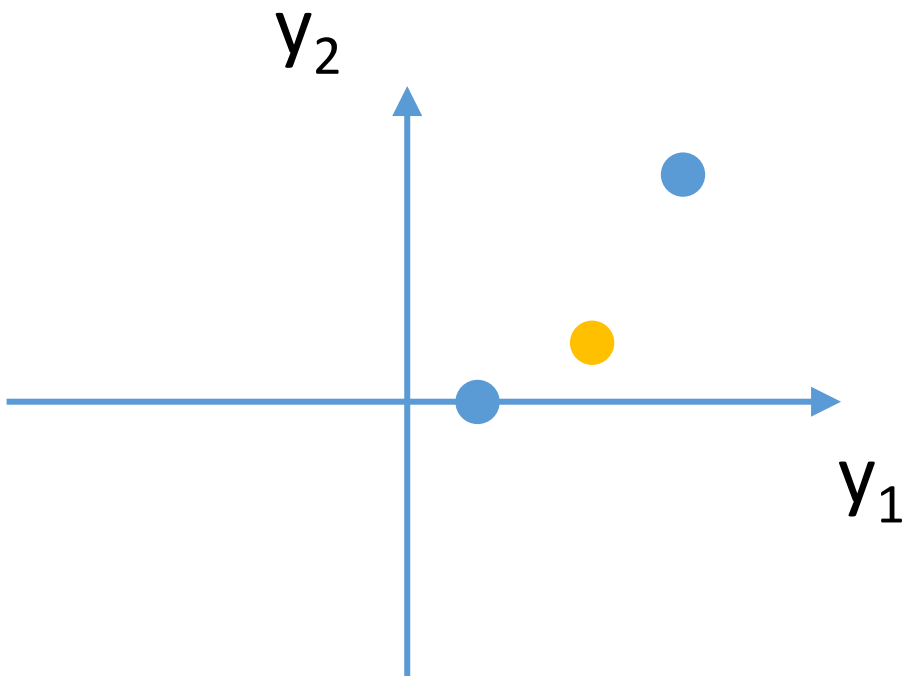


The effect of nonlinearity

1D input

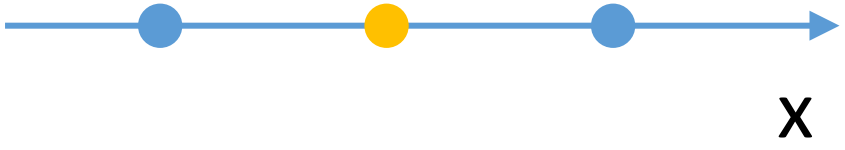


2D output of 2 RELU neurons

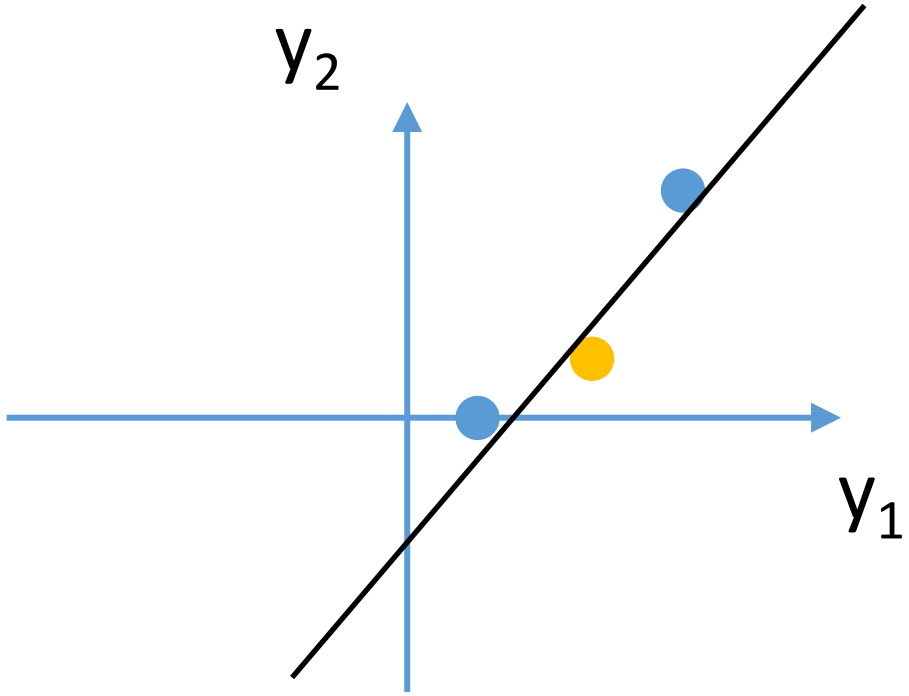


The effect of nonlinearity

1D input



2D output of 2 RELU neurons



Linear split works now!

Epoch
000,844

Learning rate: 0.003
Activation: ReLU
Regularization: None
Regularization rate: 0
Problem type: Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

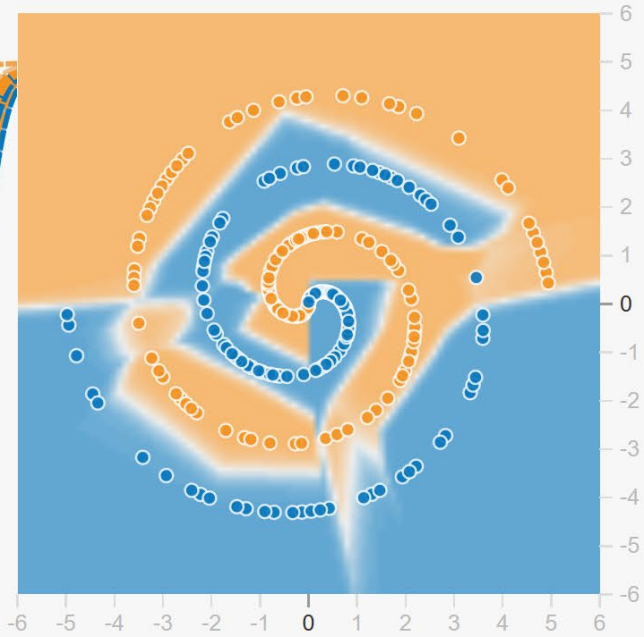
- X_1
- X_2
- X_1^2
- X_2^2
- X_1X_2
- $\sin(X_1)$
- $\sin(X_2)$

3 HIDDEN LAYERS

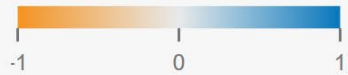
8 neurons | 8 neurons | 8 neurons

OUTPUT

Test loss 0.068
Training loss 0.010



Colors shows data, neuron and weight values.



Show test data Discretize output

Rules of thumb

- Various network blocks transform the input to a new representation (latent space)
- Transforming nonlinearly to a sufficiently high-dimensional space allows linear regression or classification of practically any data
  - A single high-dimensional linear split can carve out complex nonlinear regions of the input space
  - Generalizing to new data points may be poor => regularization techniques needed
- Transforming to a lower-dimensional space = compression, forces learning the most important underlying structure of the data

Contents

- Preliminaries: compute graphs, artificial neurons, activation functions, loss functions, latent spaces
- Understanding nonlinear activations
- **Image generator architectures**
- Quality metrics: FID, Precision & Recall

Generative models

- Learn a probability distribution $p(\mathbf{x})$ – e.g., facial images – and a way to draw samples from it
- More common in practice: learn a conditional distribution $p(\mathbf{y} \mid \mathbf{x})$
- Same as approximating some function $\mathbf{y}=f(\mathbf{x})$, but with multiple possible **y** for each **x**
- Example: image colorization. Many possible interpretations.

input

U-Net
(discriminative)

GAN, WAE...
(generative)

Ground truth

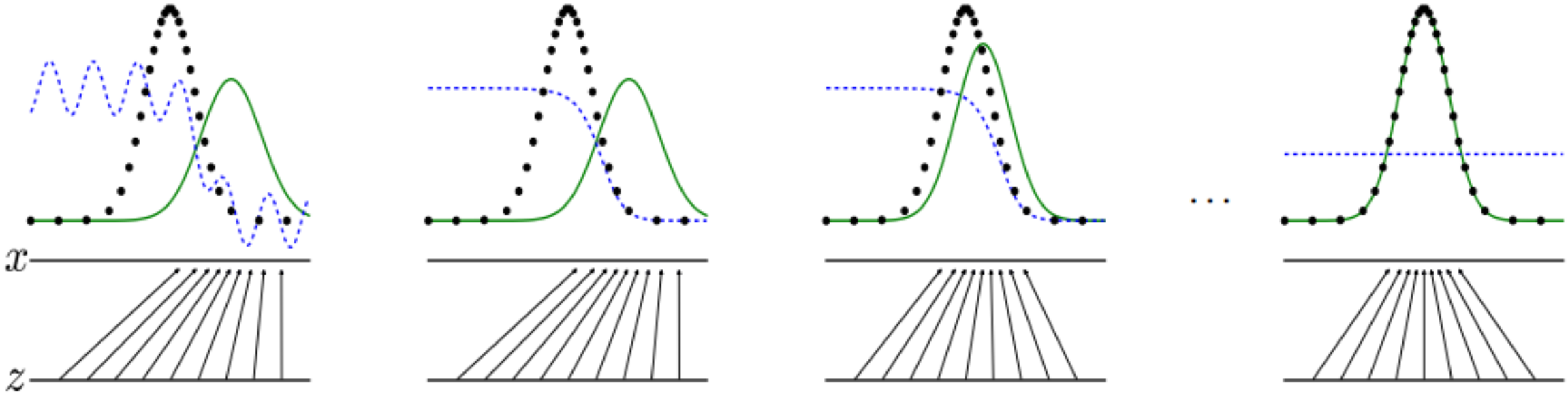
Types of image generators

Model type	Generation speed	Image quality	Image diversity	Interpolable latent space	Examples
VAE (Variational Autoencoder)	fast	Low	high	Yes	https://arxiv.org/abs/1312.6114
GAN (Generative Adversarial Networks)	fast	high	low	Yes	StyleGAN 1-3, BigGAN, https://github.com/NVlabs/stylegan3, https://www.tensorflow.org/hub/tutorials/biggan_generation_with_tf_hub
Flow-based	fast	medium	high	Yes	https://openai.com/research/glow
Transformer (autoregressive generation patch-by-patch)	slow	high	high	No	https://github.com/google-research/parti
Diffusion	slow	very high	high	Somewhat (non-smooth)	DALL-E, Midjourney, Stable Diffusion

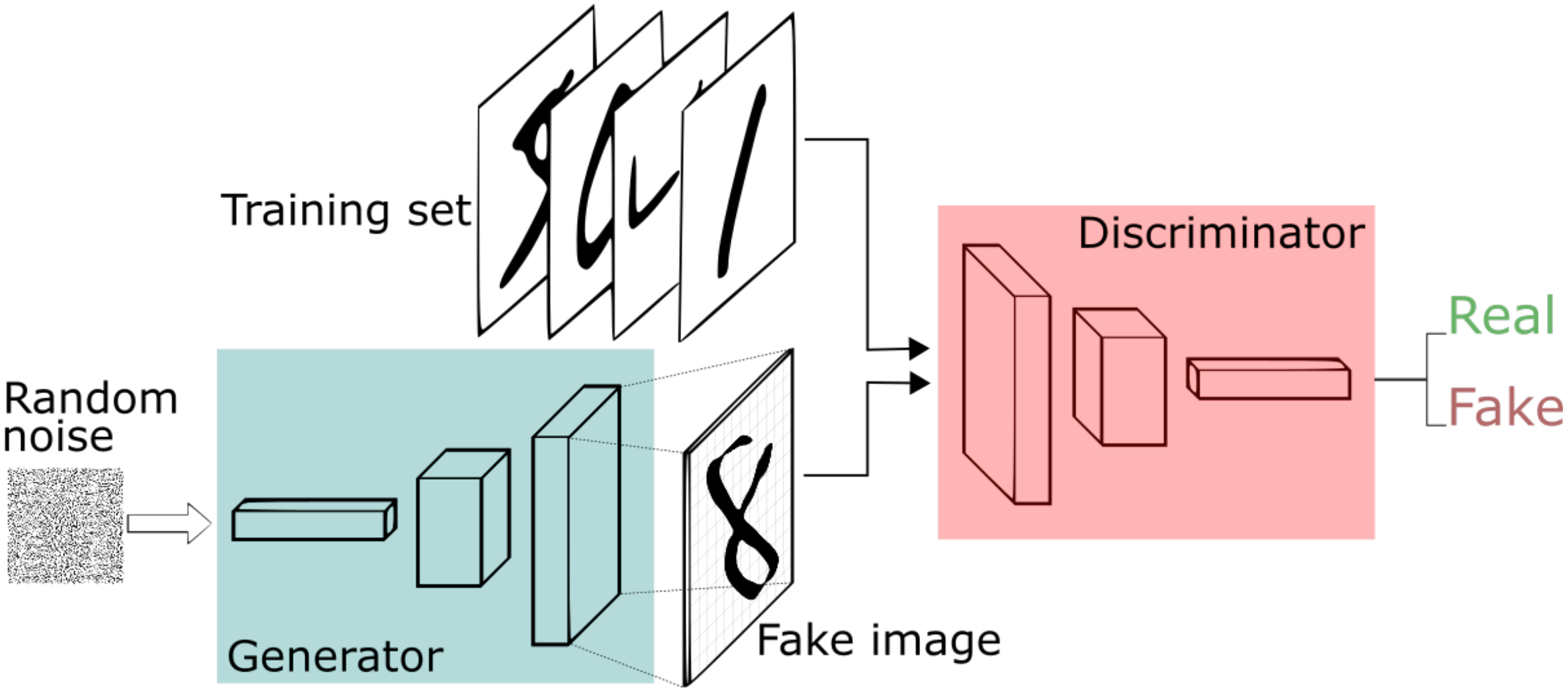
Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs)

- A *generator* network maps random vectors **z** to data vector **x**
- A *discriminator* network: 2-class classifier, trained to distinguish actual data from **x**

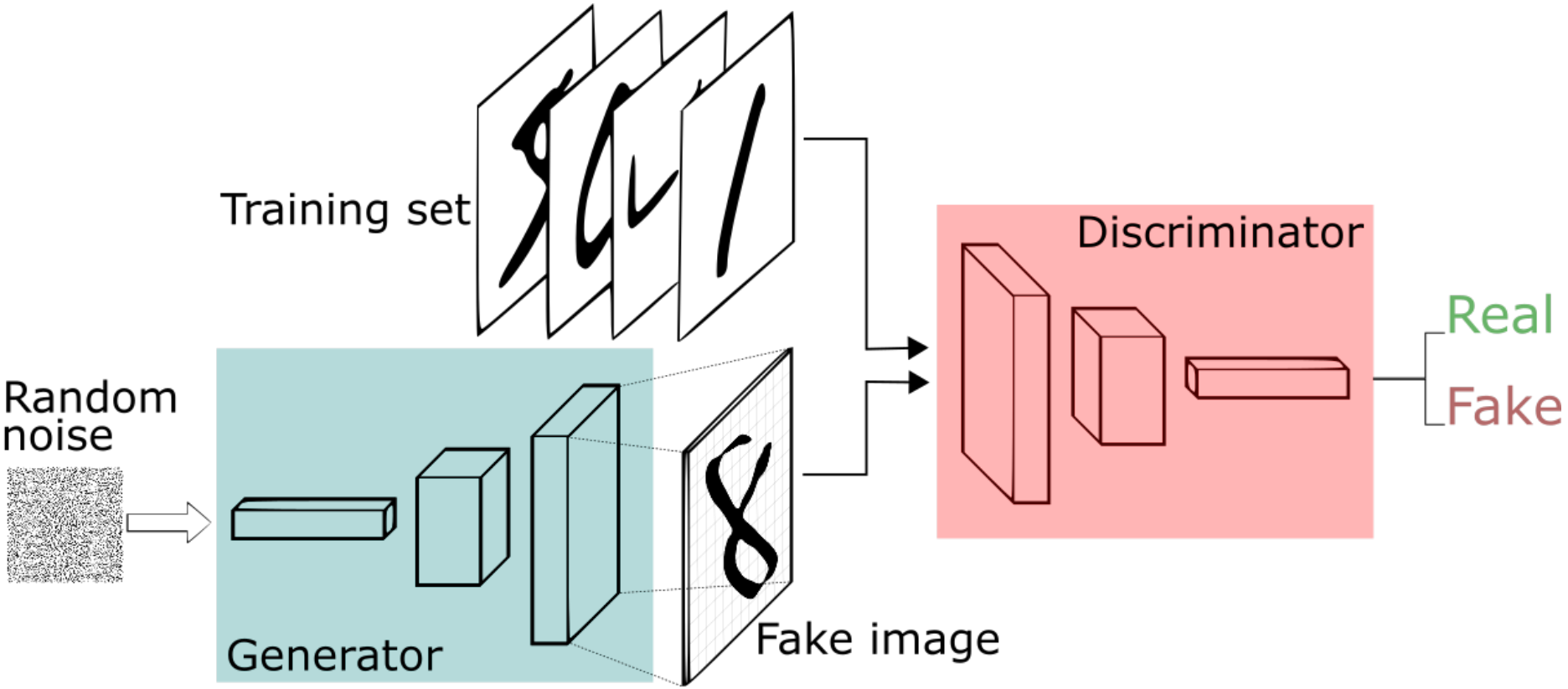


GAN: two networks in a single compute graph



GANs are trained in interleaved manner

1. Keep generator fixed, optimize discriminator to maximize discriminator performance with both generated and real data
2. Keep discriminator fixed, optimize generator to minimize discriminator performance with generated data



Demos and source code

- Browser-based interactive visualization: https://cs.stanford.edu/people/karpathy/gan/
- Browser-based image-to-image (pix2pix) translation: https://affinelayer.com/pixsrv/
- An accessible tutorial: https://deeplearning4j.org/generative-adversarial-network
- BigGAN art creation tool: https://www.artbreeder.com/
- State of the art code from NVIDIA:
- https://github.com/NVlabs/stylegan3
- https://github.com/NVlabs/stylegan2-ada-pytorch
- https://github.com/NVIDIA/pix2pixHD

The original GAN paper (Goodfellow et al. 2014)

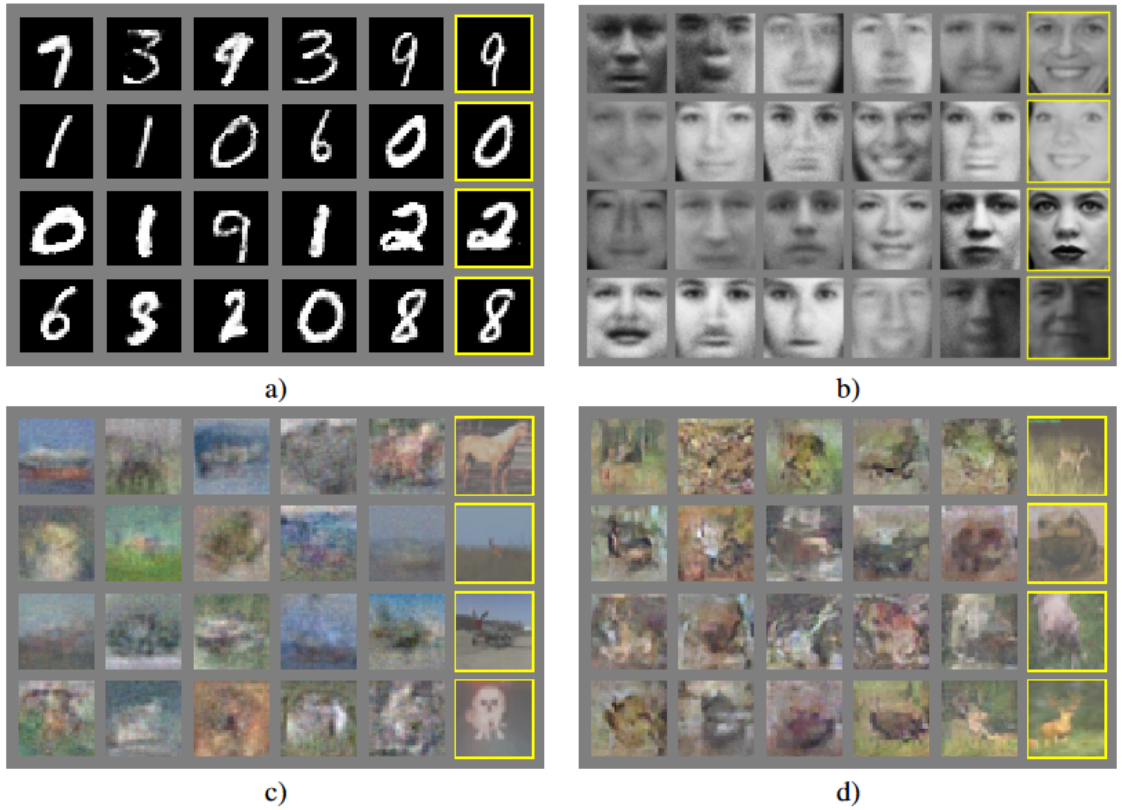


Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator)

https://proceedings.neurips.cc/paper_files/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

Generative Adversarial Nets

Ian J. Goodfellow[*], **Jean Pouget-Abadie**[†], **Mehdi Mirza**, **Bing Xu**, **David Warde-Farley**, **Sherjil Ozair**[‡], **Aaron Courville**, **Yoshua Bengio**[§]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

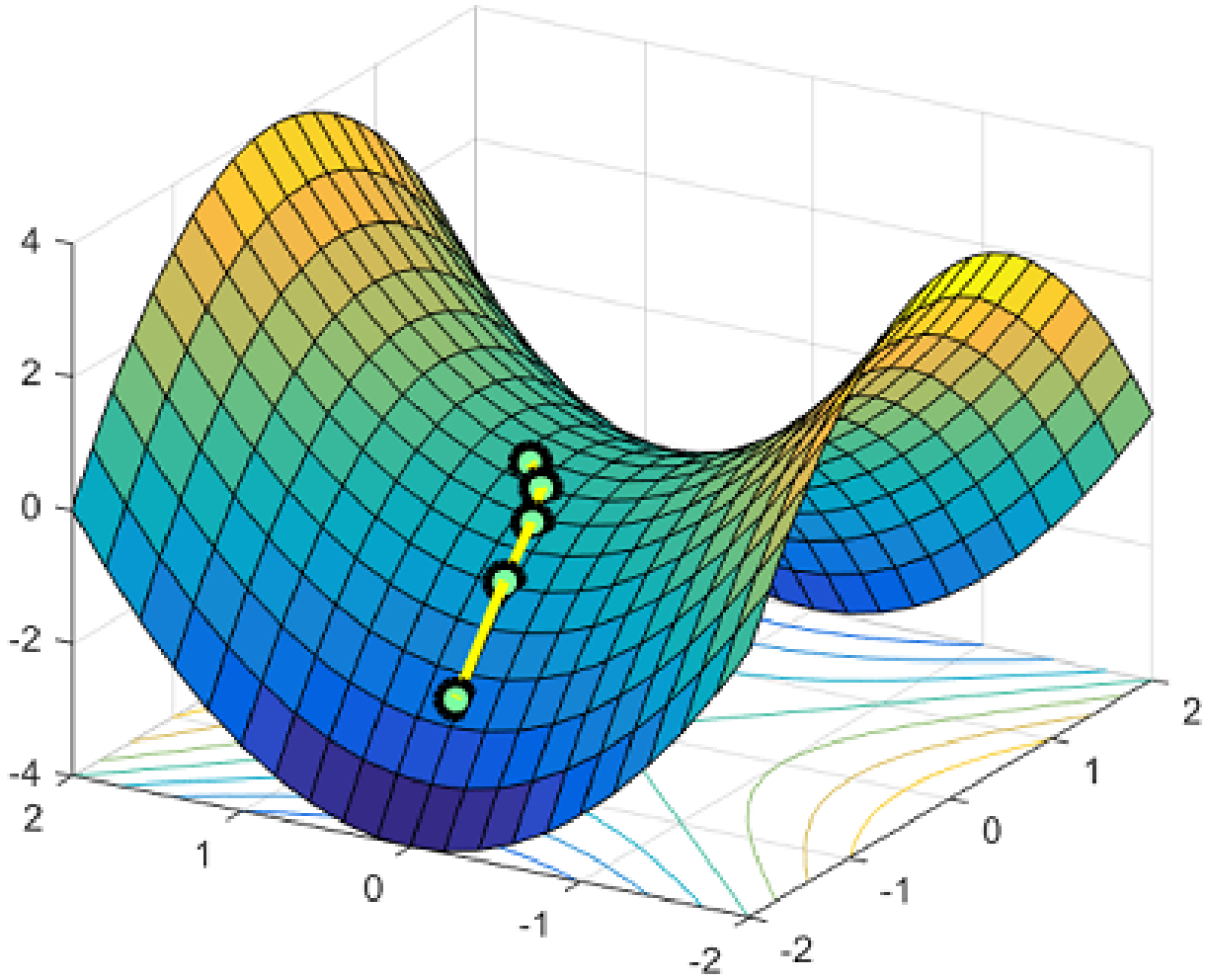
1 Introduction

The promise of deep learning is to discover rich, hierarchical models [2] that represent probability distributions over the kinds of data encountered in artificial intelligence applications, such as natural images, audio waveforms containing speech, and symbols in natural language corpora. So far, the most striking successes in deep learning have involved discriminative models, usually those that map a high-dimensional, rich sensory input to a class label [14, 20]. These striking successes have primarily been based on the backpropagation and dropout algorithms, using piecewise linear units [17, 8, 9] which have a particularly well-behaved gradient . Deep *generative* models have had less of an impact, due to the difficulty of approximating many intractable probabilistic computations that arise in maximum likelihood estimation and related strategies, and due to difficulty of leveraging the benefits of piecewise linear units in the generative context. We propose a new generative model estimation procedure that sidesteps these difficulties. [1]

In the proposed *adversarial nets* framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles.

[*] Ian Goodfellow is now a research scientist at Google, but did this work earlier as a UdeM student
[†] Jean Pouget-Abadie did this work while visiting Université de Montréal from Ecole Polytechnique.
[‡] Sherjil Ozair is visiting Université de Montréal from Indian Institute of Technology Delhi
[§] Yoshua Bengio is a CIFAR Senior Fellow.
[1] All code and hyperparameters available at `http://www.github.com/goodfeli/adversarial`

Problem: the generator-discrimination competition is unstable



Improving GAN training stability

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans. *Advances in neural information processing systems*, *29*.

Yadav, A., Shah, S., Xu, Z., Jacobs, D., & Goldstein, T. (2017). Stabilizing Adversarial Nets With Prediction Methods. *arXiv preprint arXiv:1705.07364*. (Using saddle-point optimization theory)

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of Wasserstein gans. In *Advances in Neural Information Processing Systems* (pp. 5769-5779).

Improved GAN training (Salimans et al. 2016)

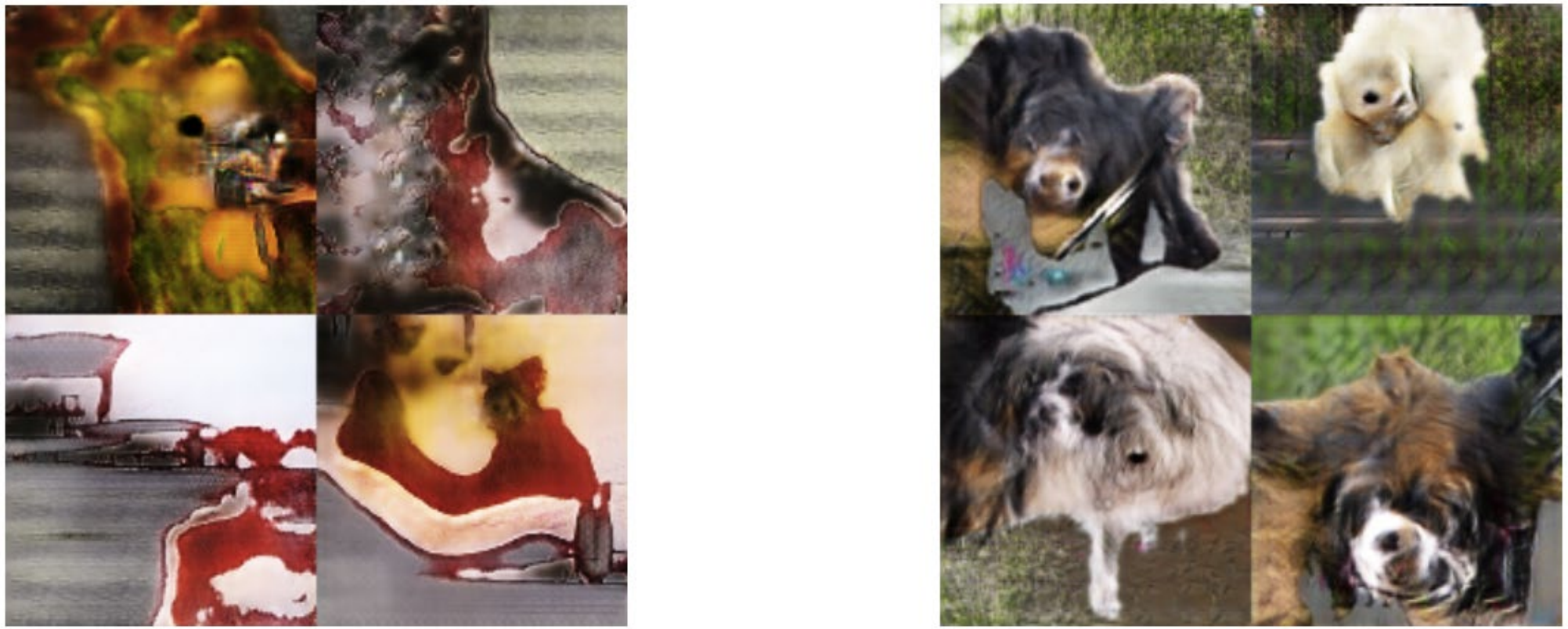


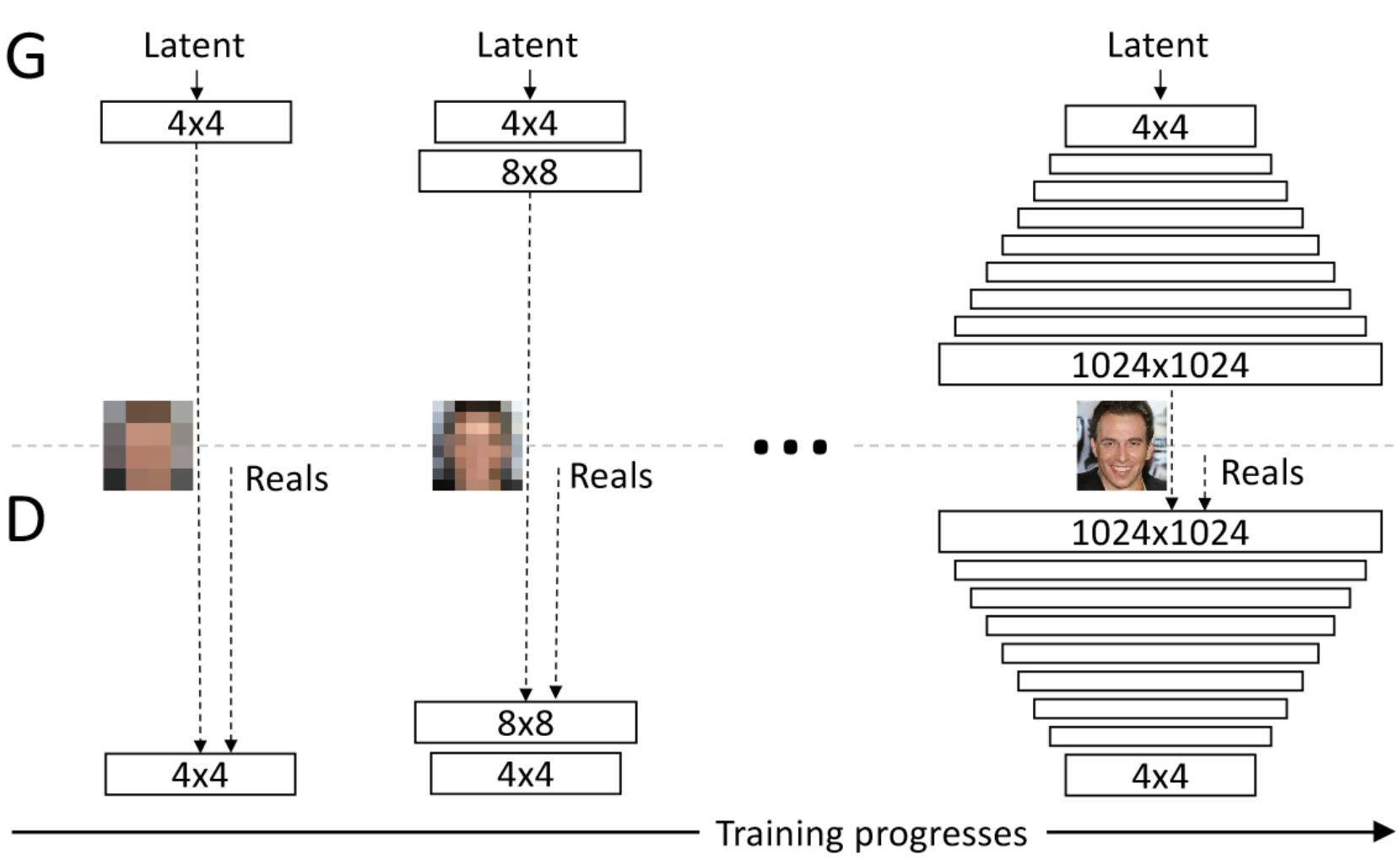
Figure 6: Samples generated from the ImageNet dataset. *(Left)* Samples generated by a DCGAN. *(Right)* Samples generated using the techniques proposed in this work. The new techniques enable GANs to learn recognizable features of animals, such as fur, eyes, and noses, but these features are not correctly combined to form an animal with realistic anatomical structure.

https://proceedings.neurips.cc/paper_files/paper/2016/hash/8a3363abe792db2d8761d6403605aeb7-Abstract.html

Progressive GANs (Karras et al. 2017)

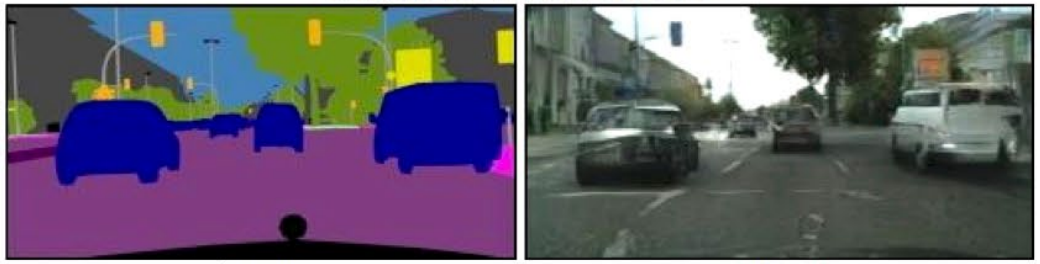
https://arxiv.org/pdf/1710.10196.pdf

Progressive GANs (Karras et al. 2017)



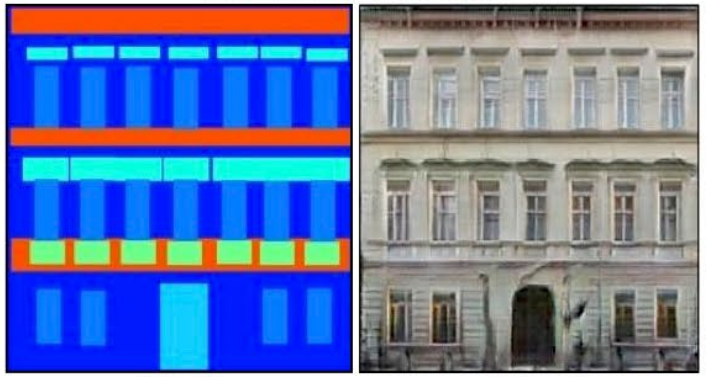
pix2pix (Isola et al. 2017)

Labels to Street Scene



input output

Labels to Facade



input output

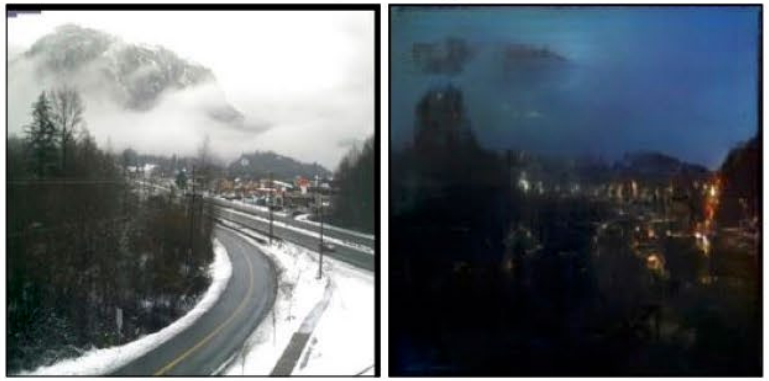
BW to Color

input output

Aerial to Map

input output

Day to Night



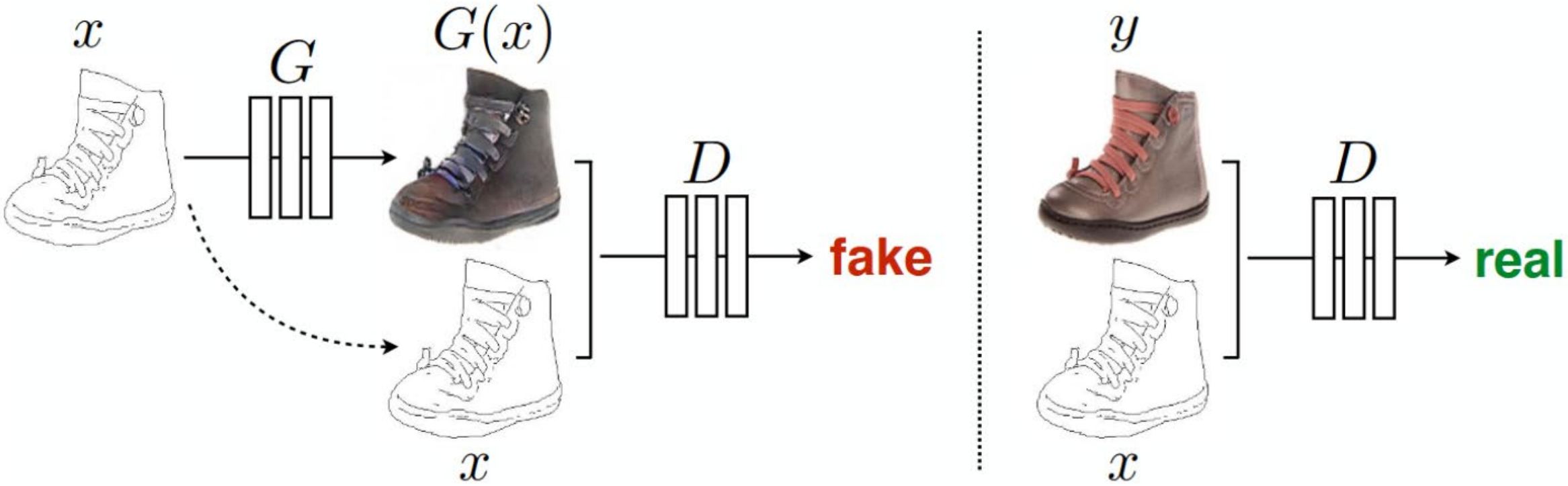
input output

Edges to Photo

input output

pix2pix (Isola et al. 2017)

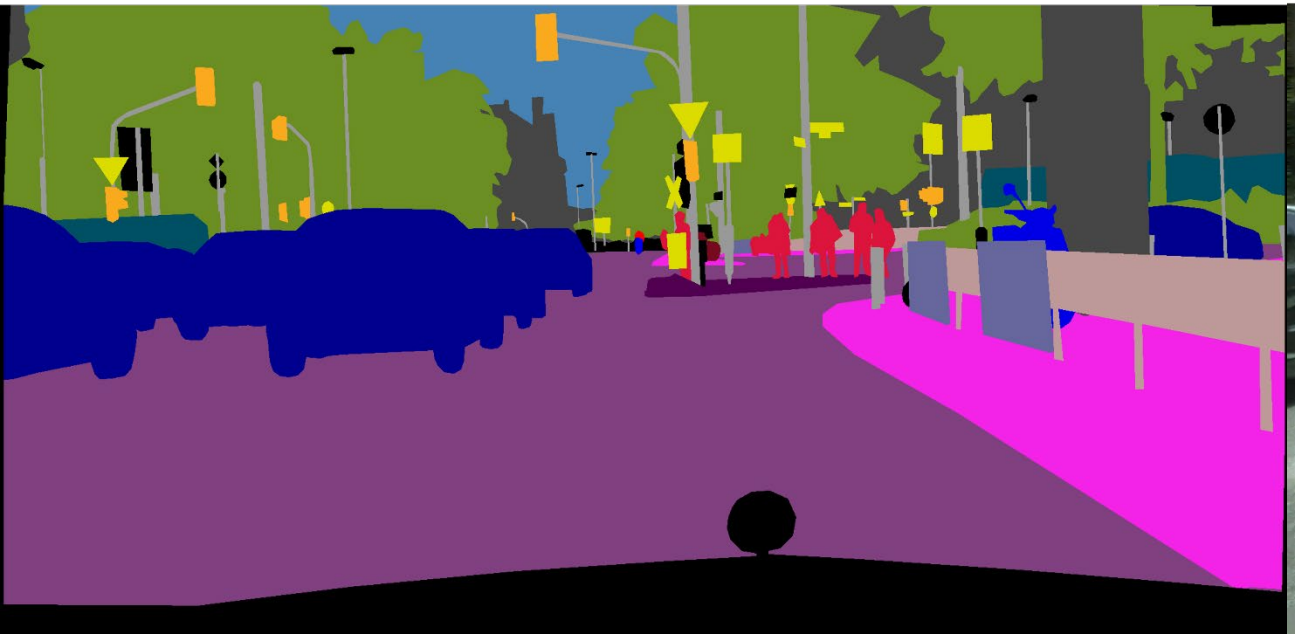
- pix2pix is an example of conditional GAN
- Generator input: edge map, noise
- Discriminator input: edge map, real or generated images



pix2pixHD

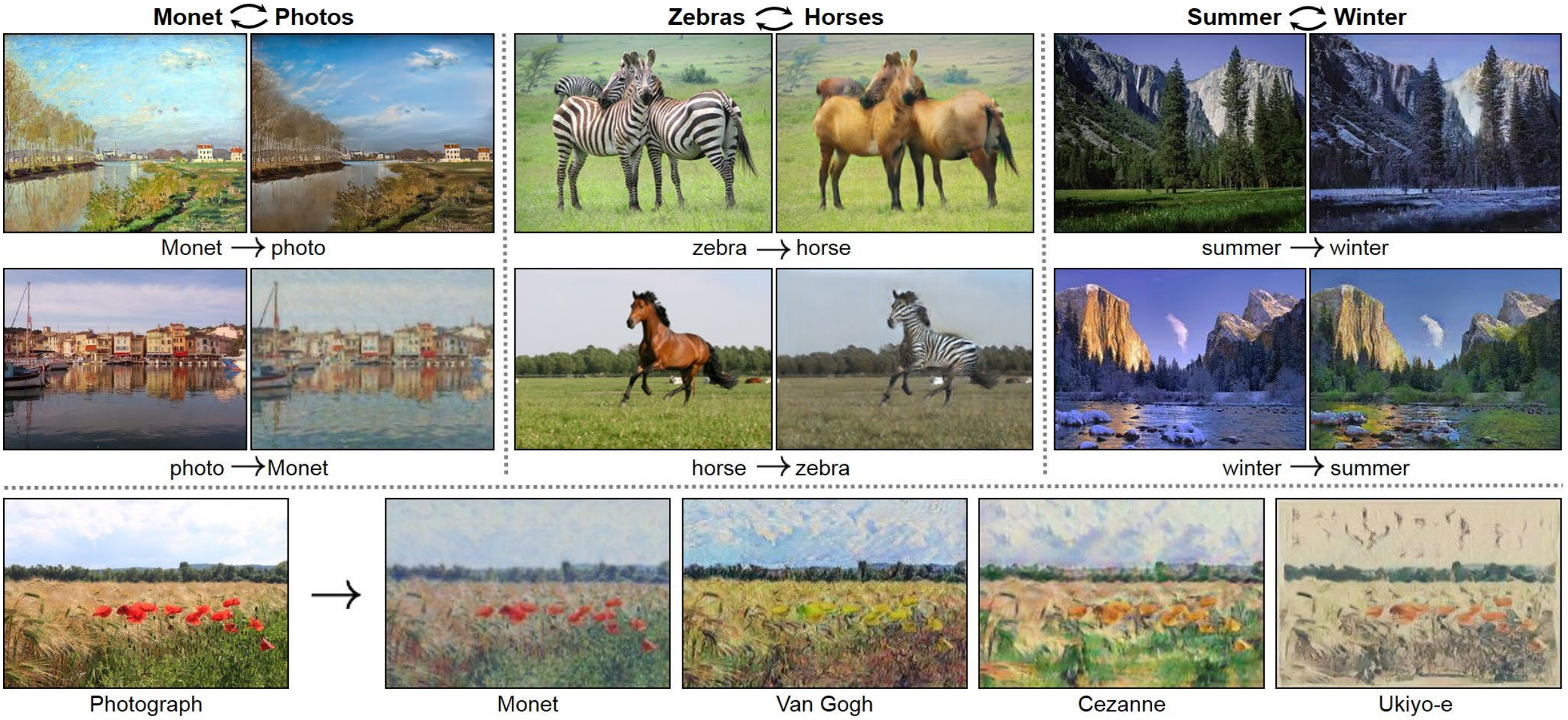
https://github.com/NVIDIA/pix2pixHD

Input labels

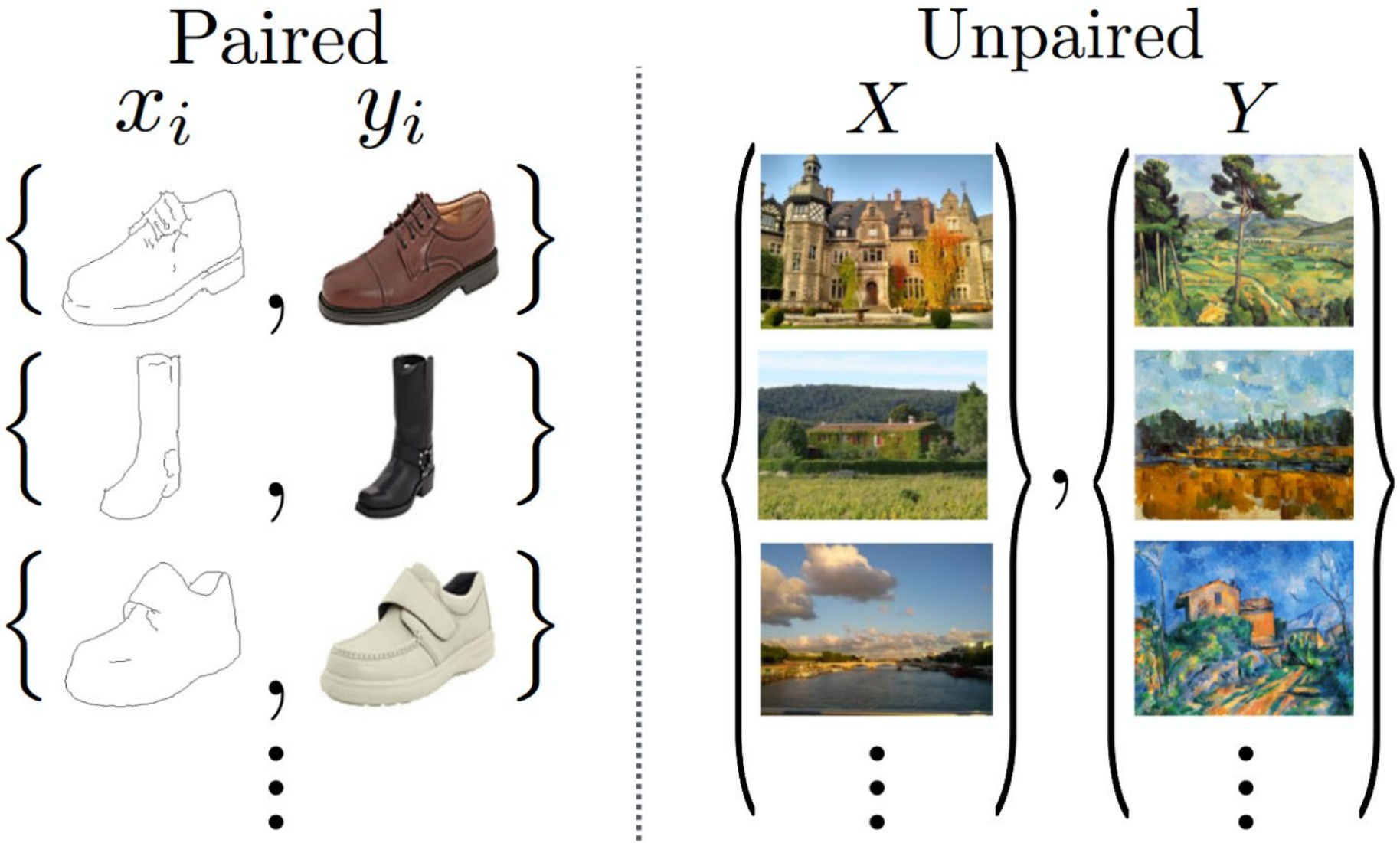


Synthesized image

CycleGAN (Jun-Yan Zhu et al. 2017)



CycleGAN: no training pairs needed!



CycleGAN: no training pairs needed!

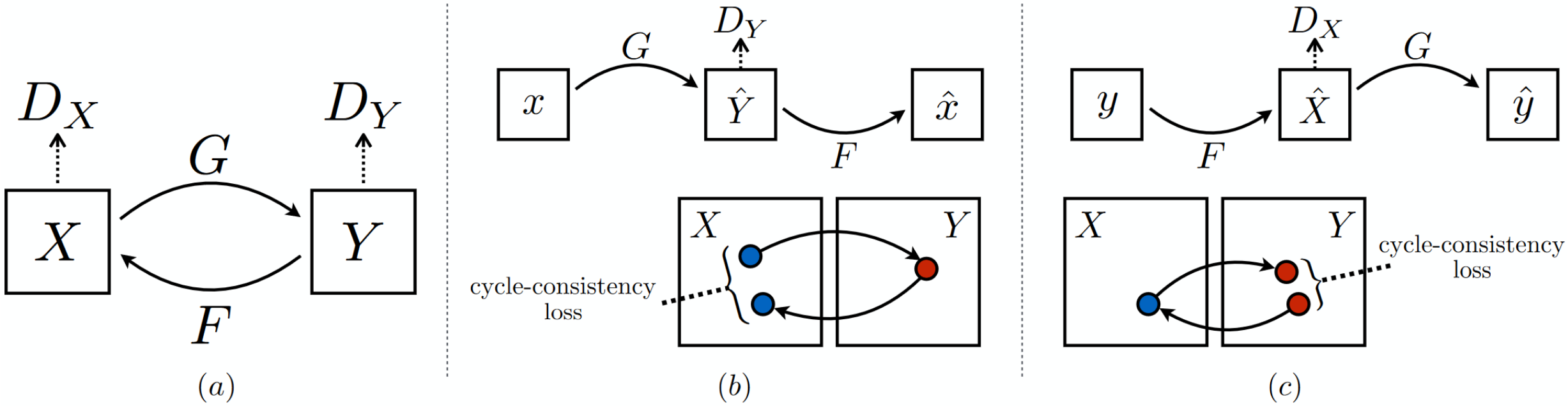
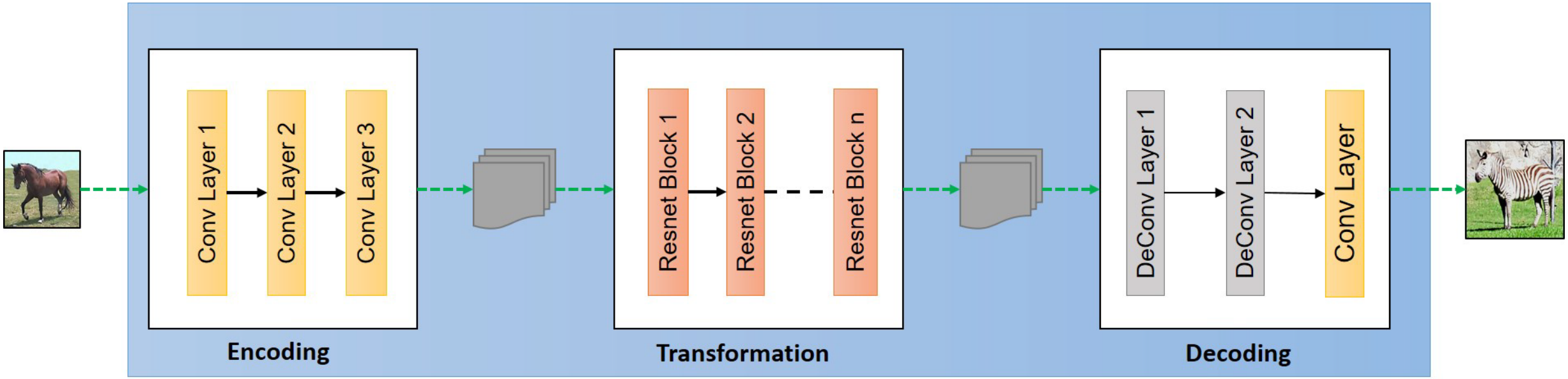
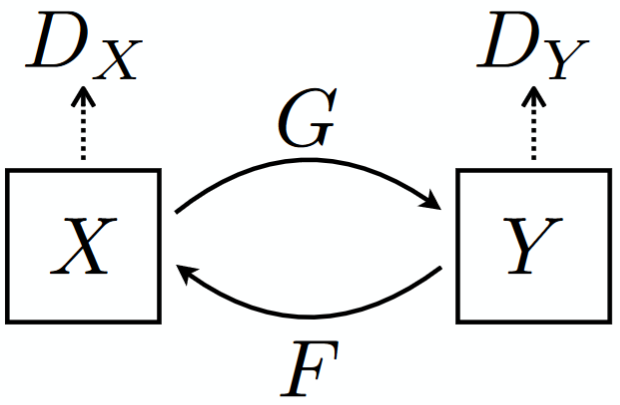


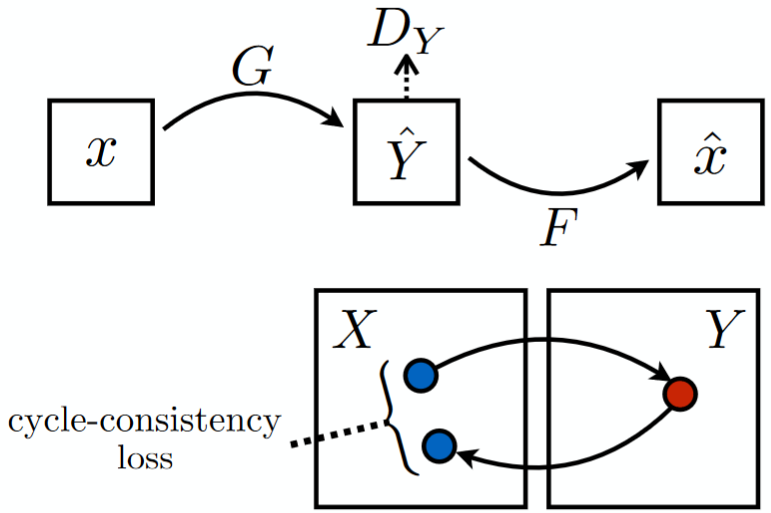
Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

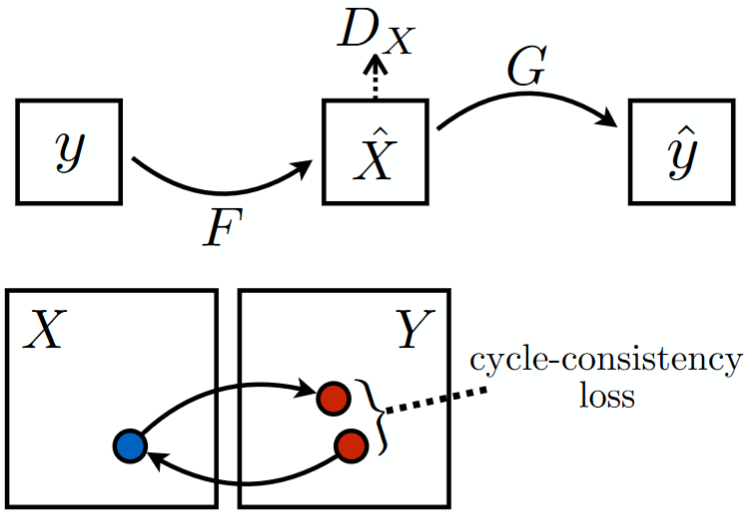
CycleGAN: no training pairs needed!

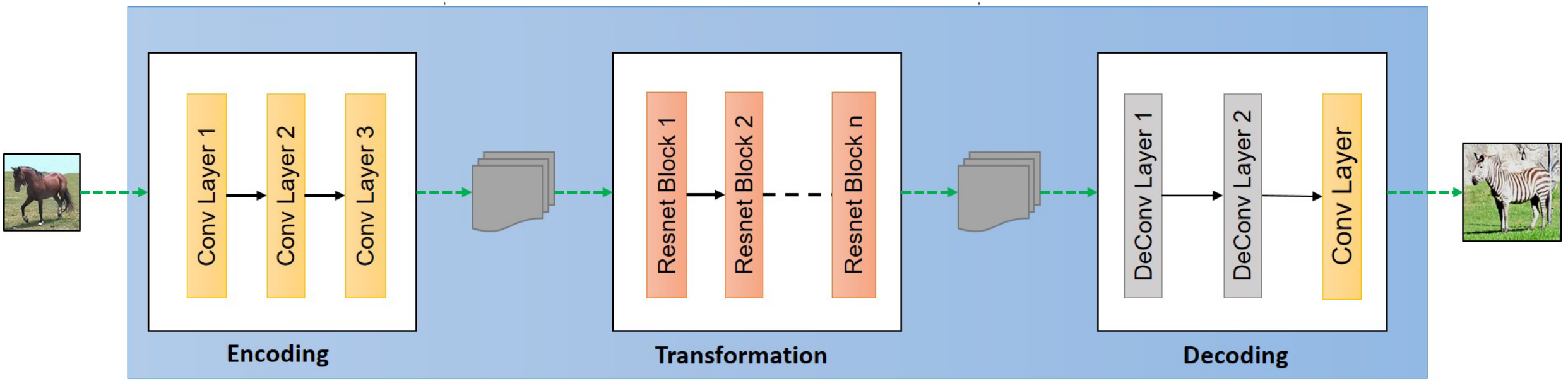


A key takeaway: infinite ways to combine basic compute graph blocks









BigGAN: Generating all 1000 ImageNet image classes

[Submitted on 28 Sep 2018 (v1), last revised 25 Feb 2019 (this version, v2)]

Large Scale GAN Training for High Fidelity Natural Image Synthesis

Andrew Brock, Jeff Donahue, Karen Simonyan

Despite recent progress in generative image modeling, successfully generating high-resolution, diverse samples from complex datasets such as ImageNet remains an elusive goal. To this end, we train Generative Adversarial Networks at the largest scale yet attempted, and study the instabilities specific to such scale. We find that applying orthogonal regularization to the generator renders it amenable to a simple "truncation trick," allowing fine control over the trade-off between sample fidelity and variety by reducing the variance of the Generator's input. Our modifications lead to models which set the new state of the art in class-conditional image synthesis. When trained on ImageNet at 128x128 resolution, our models (BigGANs) achieve an Inception Score (IS) of 166.5 and Frechet Inception Distance (FID) of 7.4, improving over the previous best IS of 52.52 and FID of 18.6.

ARTBREEDER

Extend your imagination

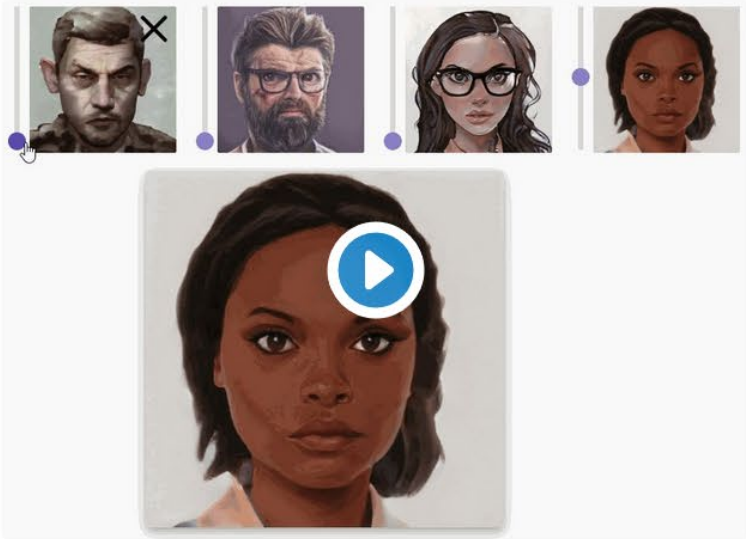
Thousands of users have collectively made 59946402 images

View Gallery | Start | Watch Intro

Bay Raitt
@bayraitt

Artbreeder is a nuclear powered pencil.

have a spin remixing some of mine here:artbreeder.com/bayraitt/starr…#artbreeder #ai #ganbreeder #conceptart #comics



♡ 8,991 2:29 AM - Sep 18, 2019

2,942 people are talking about this

Henry Lynch
@HenryLynch_Art

'New World Officer'

Character created with AI image breeding and paint-over. Fast and interesting. It is the future. @Artbreeder 👏#ConceptArt #CharacterDesign #SciFi #AI #ConceptArtist

♡ 58 5:18 AM - Oct 9, 2019

See Henry Lynch's other Tweets

TELTHONA
@telthona

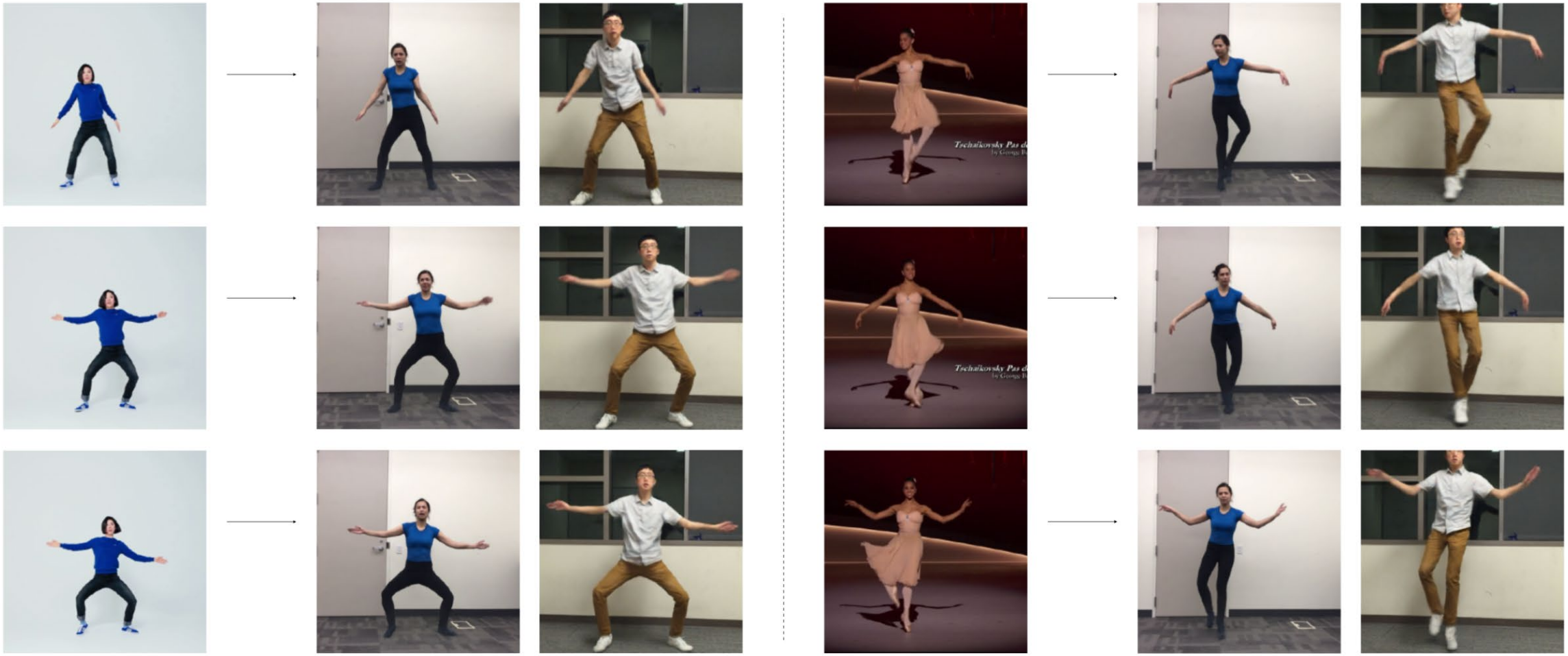
I generated bunch of concepts with new AI powered website - it's the best AI app that i tried so far! It's amaizng for creature exploration, mood thumbnailing and more❤️ finally AI that i can truly use in the creative process! ganbreeder.app #Aimakesart #ganbreeder

♡ 69 8:41 PM - Jun 29, 2019

See TELTHONA's other Tweets

Everybody Dance Now (Chan et al. 2018)

- Conditioning GAN image synthesis with neural network motion tracking



StyleGAN (Karras et al. 2019)

Facebook Removes Accounts With AI-Generated Profile Photos

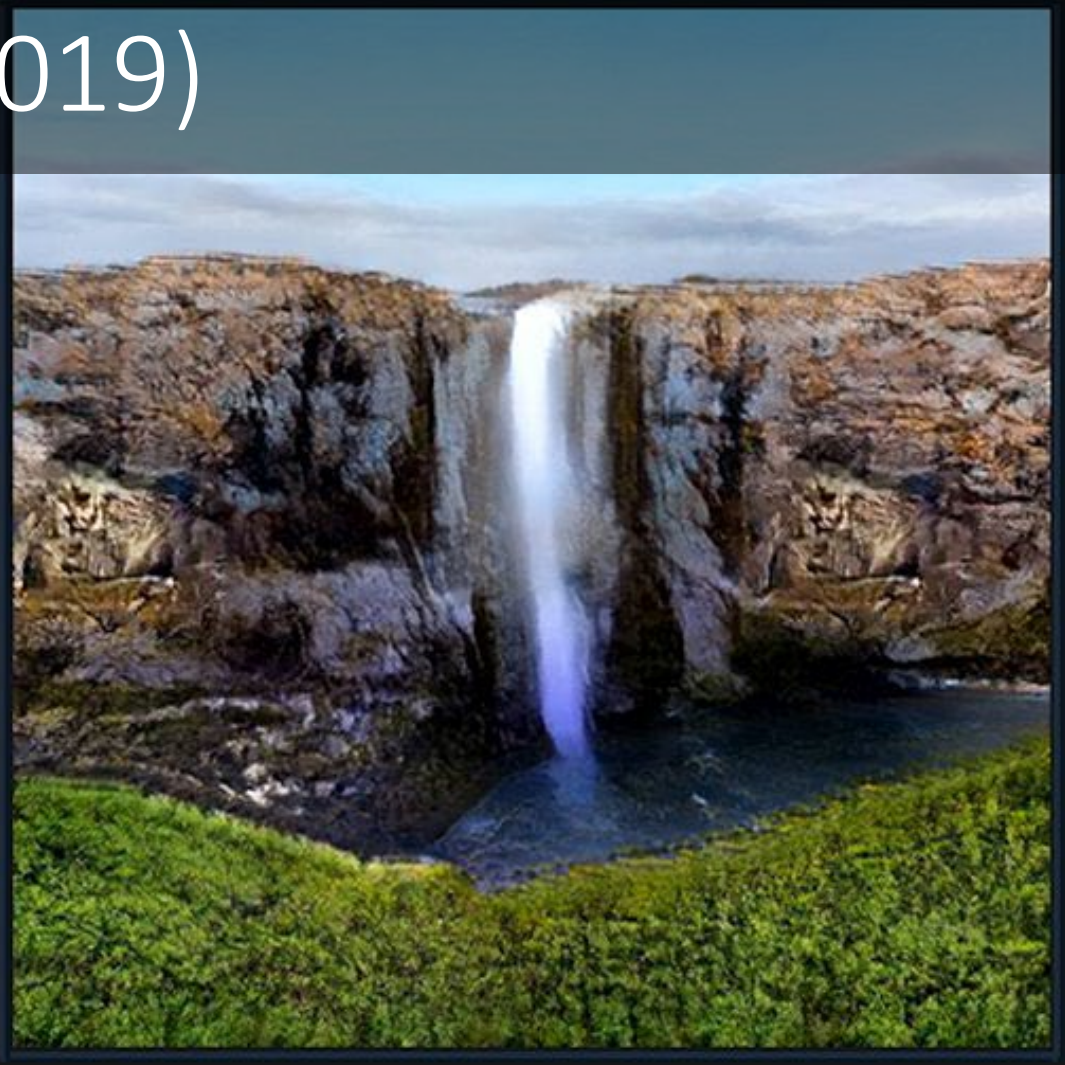
Researchers said it appears to be the first use of artificial intelligence to support an inauthentic social media campaign.

ILLUSTRATION: WIRED STAFF; DIMITRI OTIS/GETTY IMAGES

2019

https://www.wired.com/story/facebook-removes-accounts-ai-generated-photos/

Painting with GANs (2019)



sky | tree | cloud | mountain

snow | water | hill | dirt

GauGAN: https://github.com/NVlabs/SPADE

GAN Dissection: https://gandissect.csail.mit.edu

StyleGAN 2 (2020)

StyleGAN can provide similar results if one is lucky, but StyleGAN 2 produces high quality much more consistently and predictably

StyleGAN 2 ”circuit bending”

Finetuning a pretrained Nvidia StyleGAN 2 network with only 250 images (google image search with ”dragons”)

https://twitter.com/Norod78/status/1218282356391530496?s=20

StyleGAN font generation

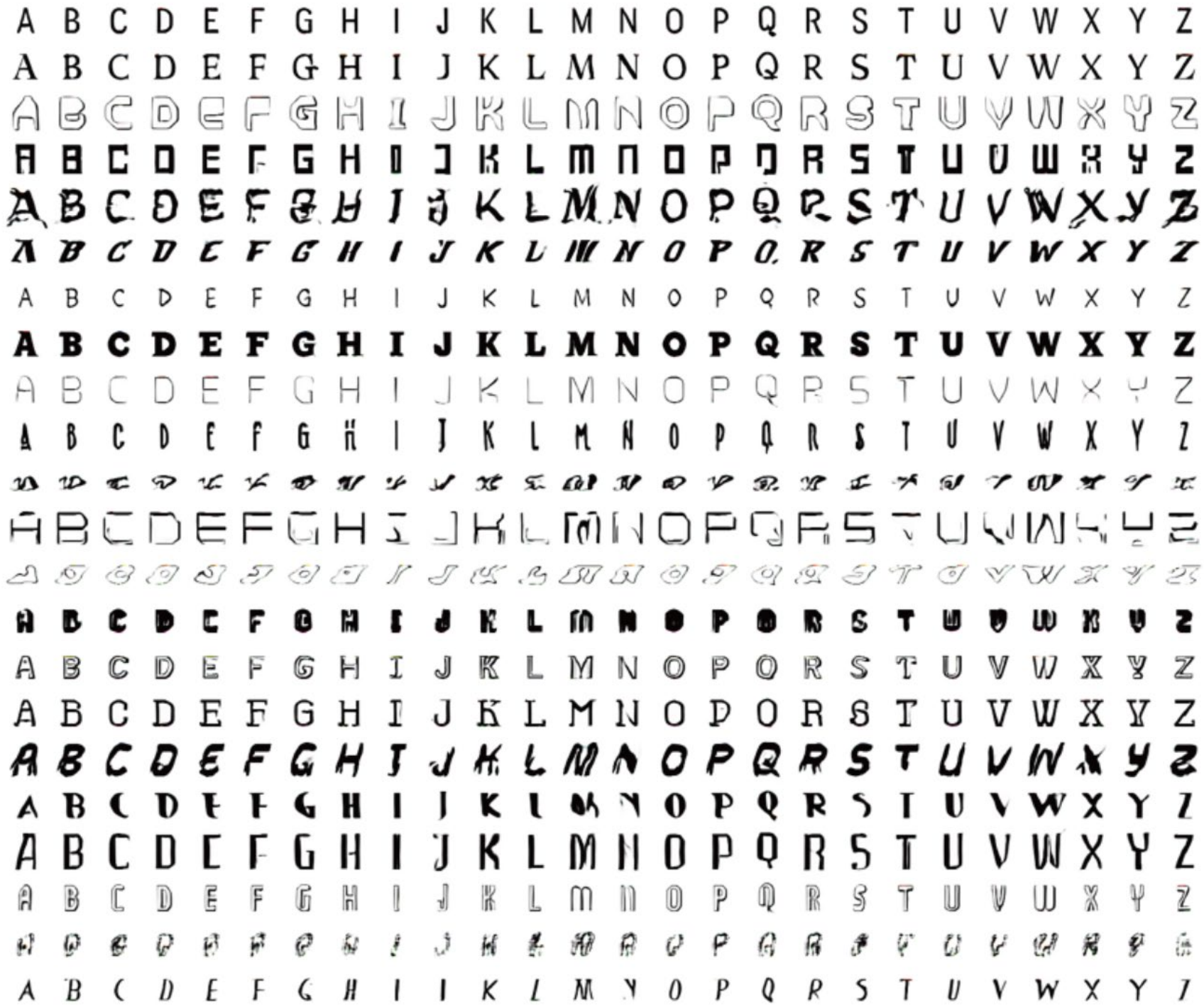
http://www.machinelearningfont.com/#mlfabout

[Submitted on 29 May 2019 (v1), last revised 30 May 2019 (this version, v2)]

GlyphGAN: Style-Consistent Font Generation Based on Generative Adversarial Networks

Hideaki Hayashi, Kohtaro Abe, Seiichi Uchida

In this paper, we propose GlyphGAN: style-consistent font generation based on generative adversarial networks (GANs). GANs are a framework for learning a generative model using a system of two neural networks competing with each other. One network generates synthetic images from random input vectors, and the other discriminates between synthetic and real images. The motivation of this study is to create new fonts using the GAN framework while maintaining style consistency over all characters. In GlyphGAN, the input vector for the generator network consists of two vectors: character class vector and style vector. The former is a one-hot vector and is associated with the character class of each sample image during training. The latter is a uniform random vector without supervised information. In this way, GlyphGAN can generate an infinite variety of fonts with the character and style independently controlled. Experimental results showed that fonts generated by GlyphGAN have style consistency and diversity different from the training images without losing their legibility.

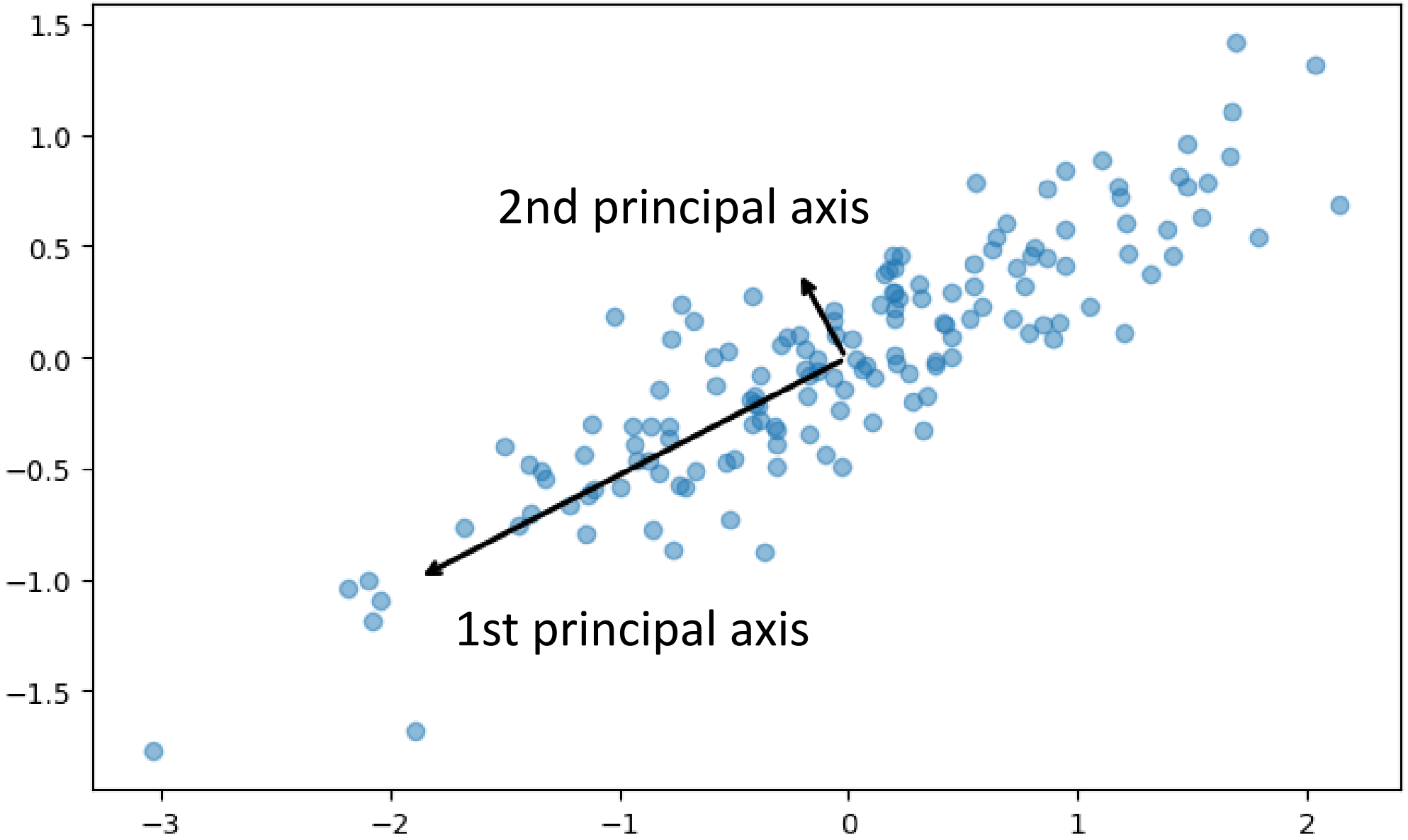


Other font generation projects

- https://github.com/erikbern/deep-fonts
- https://github.com/uchidalab/fontdesign_gan
- https://github.com/patrickgadd/feel-the-kern (with kerning)

PCA-based GAN control (Härkönen et al. 2020)

Principal Component Analysis (PCA)



StyleGAN 3(2021)

Much better interpolation

Key: Corrects the aliasing artefacts caused by the upsampling stages of the decoder

Video: https://youtu.be/0zaGYLPj4Kk?si=ni0fHn7V1Bcut7GO&t=87

Alias-Free Generative Adversarial Networks

Tero Karras
NVIDIA
tkarras@nvidia.com

Miika Aittala
NVIDIA
maittala@nvidia.com

Samuli Laine
NVIDIA
slaine@nvidia.com

Erik Härkönen*
Aalto University and NVIDIA
erik.harkonen@aalto.fi

Janne Hellsten
NVIDIA
jhellsten@nvidia.com

Jaakko Lehtinen
NVIDIA and Aalto University
jlehtinen@nvidia.com

Timo Aila
NVIDIA
taila@nvidia.com

Abstract

We observe that despite their hierarchical convolutional nature, the synthesis process of typical generative adversarial networks depends on absolute pixel coordinates in an unhealthy manner. This manifests itself as, e.g., detail appearing to be glued to image coordinates instead of the surfaces of depicted objects. We trace the root cause to careless signal processing that causes aliasing in the generator network. Interpreting all signals in the network as continuous, we derive generally applicable, small architectural changes that guarantee that unwanted information cannot leak into the hierarchical synthesis process. The resulting networks match the FID of StyleGAN2 but differ dramatically in their internal representations, and they are fully equivariant to translation and rotation even at subpixel scales. Our results pave the way for generative models better suited for video and animation.

1 Introduction

The resolution and quality of images produced by generative adversarial networks (GAN) [19] have seen rapid improvement recently [27, 11, 29, 30]. They have been used for a variety of applications, including image editing [42, 47, 37, 20, 34, 3], domain translation [62, 32, 53, 36], and video generation [49, 15, 21]. While several ways of controlling the generative process have been found [8, 26, 10, 36, 22, 2, 7, 41, 6], the foundations of the synthesis process remain only partially understood.

In the real world, details of different scale tend to transform hierarchically. For instance, moving a head causes the nose to move, which in turn moves the skin pores on it. The structure of a typical GAN generator is analogous: coarse, low-resolution features are hierarchically refined by upsampling layers, locally mixed by convolutions, and new detail is introduced through nonlinearities. We observe that despite this superficial similarity, current GAN architectures do not synthesize images in a natural hierarchical manner: the coarse features mainly control the *presence* of finer features, but not their precise positions. Instead, much of the fine detail appears to be fixed in pixel coordinates. This disturbing "texture sticking" is clearly visible in latent interpolations (see Figure 1 and our accompanying videos on the project page https://nvlabs.github.io/stylegan3), breaking the illusion of a solid and coherent object moving in space. Our goal is an architecture that

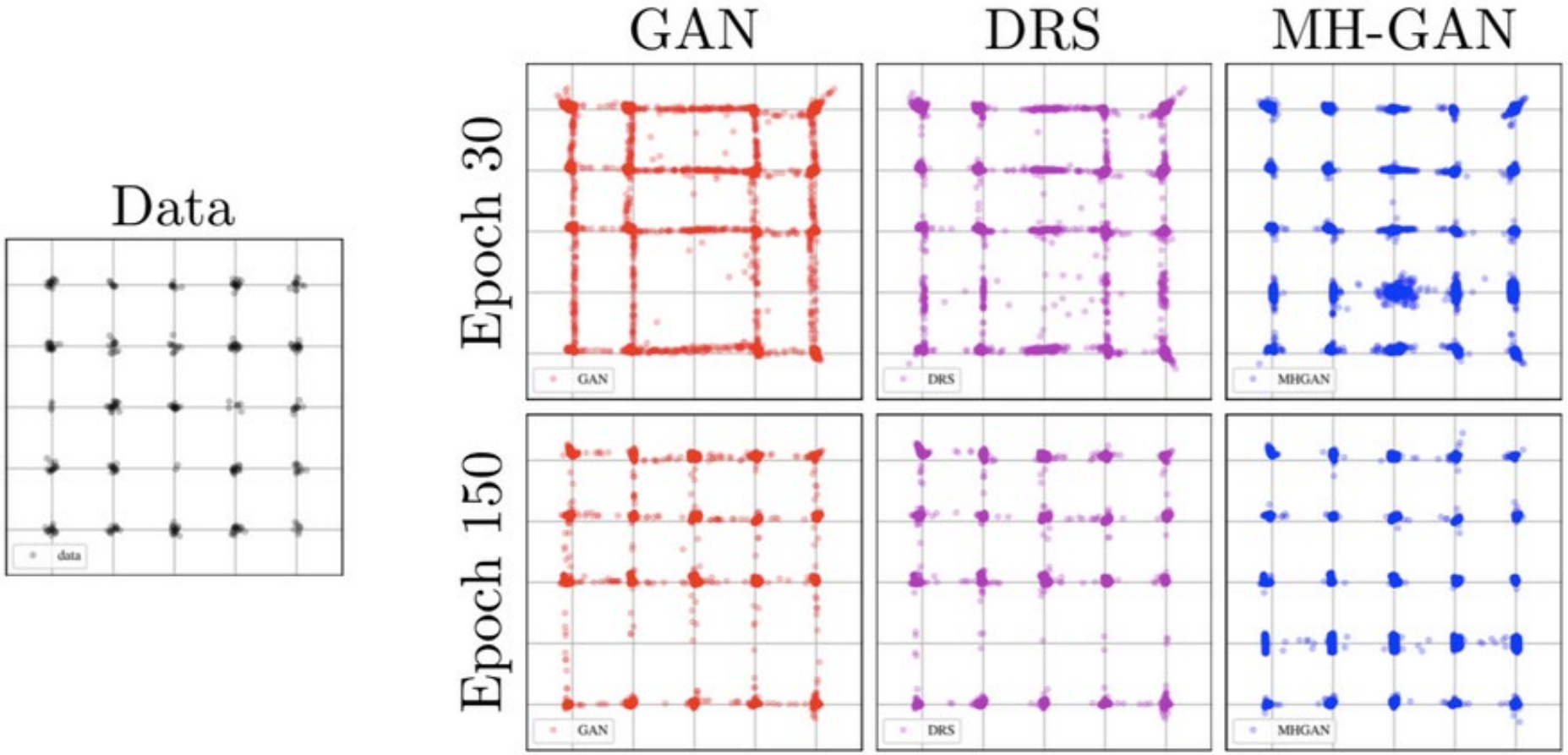
*This work was done during an internship at NVIDIA.

- Applications using GANs
  - Font generation
  - Anime character generation
  - Interactive Image generation
  - Text2Image (text to image)
  - 3D Object generation
  - Image Editing
  - Face Aging
  - Human Pose Estimation
  - Domain-transfer (e.g. style-transfer, pix2pix, sketch2image)
  - Image Inpainting (hole filling)

https://github.com/nashory/gans-awesome-applications

Diffusion models

The latent-to-image mapping in GANs is hard



https://www.uber.com/en-FI/blog/mh-gan/

Diffusion model motivation: It can be easier to train a denoiser than a generator

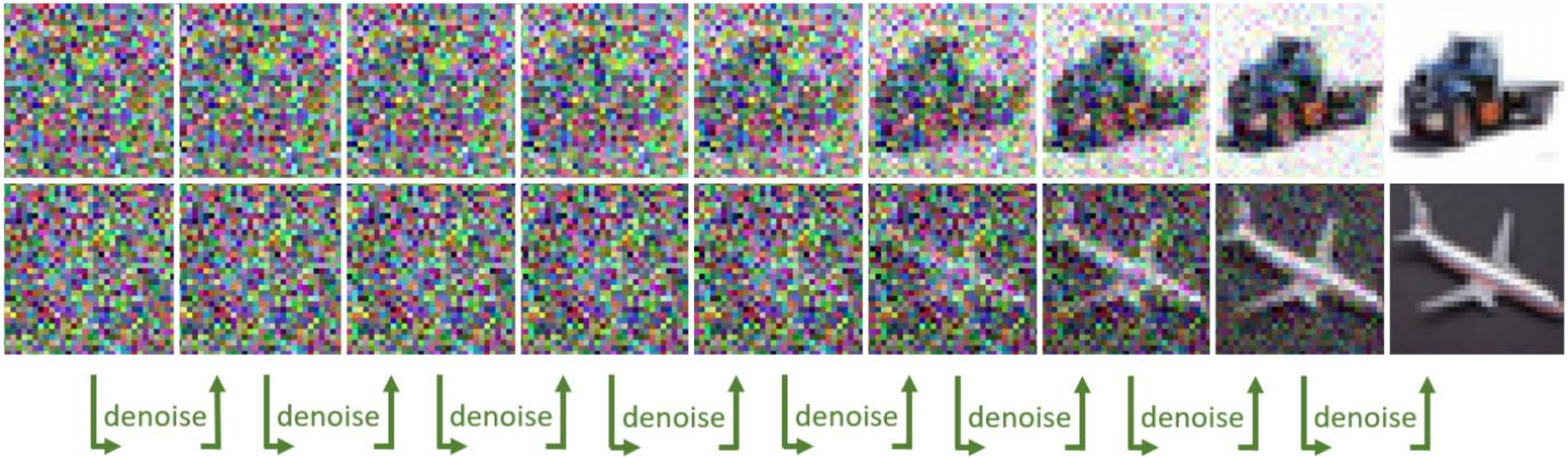


Figure 1. Denoising diffusion reveals novel images from pure noise through repeated denoising

https://developer.nvidia.com/blog/generative-ai-research-spotlight-demystifying-diffusion-based-models/

The code below is a first guess of how to implement this idea, assuming a neural network function `denoise` is available.

```
# start with an image of pure large-magnitude noise
sigma = 80      # initial noise level
x = sigma * torch.randn(img_shape)

for step in range(256):
    # keep 98% of current noisy image, and mix in 2% of denoising
    x = 0.98 * x + 0.02 * denoise(x, sigma)

    # keep track of current noise level
    sigma *= 0.98
```

If you've ever looked at code bases or scientific papers in the field, filled with pages of equations, you might be surprised to learn that this near-trivial piece of code is actually a theoretically valid implementation of something called a *probability flow ordinary differential equation solver*. While this snippet is hardly optimal, it embodies surprisingly many of the key good practices explained in the paper. The team's top-of-the-line final sampler is essentially just a few more lines.

https://developer.nvidia.com/blog/generative-ai-research-spotlight-demystifying-diffusion-based-models/

What about that function `denoise`? At its core, it's surprisingly straightforward as well: the denoiser must output the blurry average of all possible clean images that could have been hiding under the noise. The desired output at various noise levels might look like the examples in Figure 2.

Figure 2. Examples of ideal denoiser outputs at different noise levels. At high noise levels, the image details are uncertain, and remain blurry in the output

https://developer.nvidia.com/blog/generative-ai-research-spotlight-demystifying-diffusion-based-models/

2015: The original paper

Was slept on when GANs dominated

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

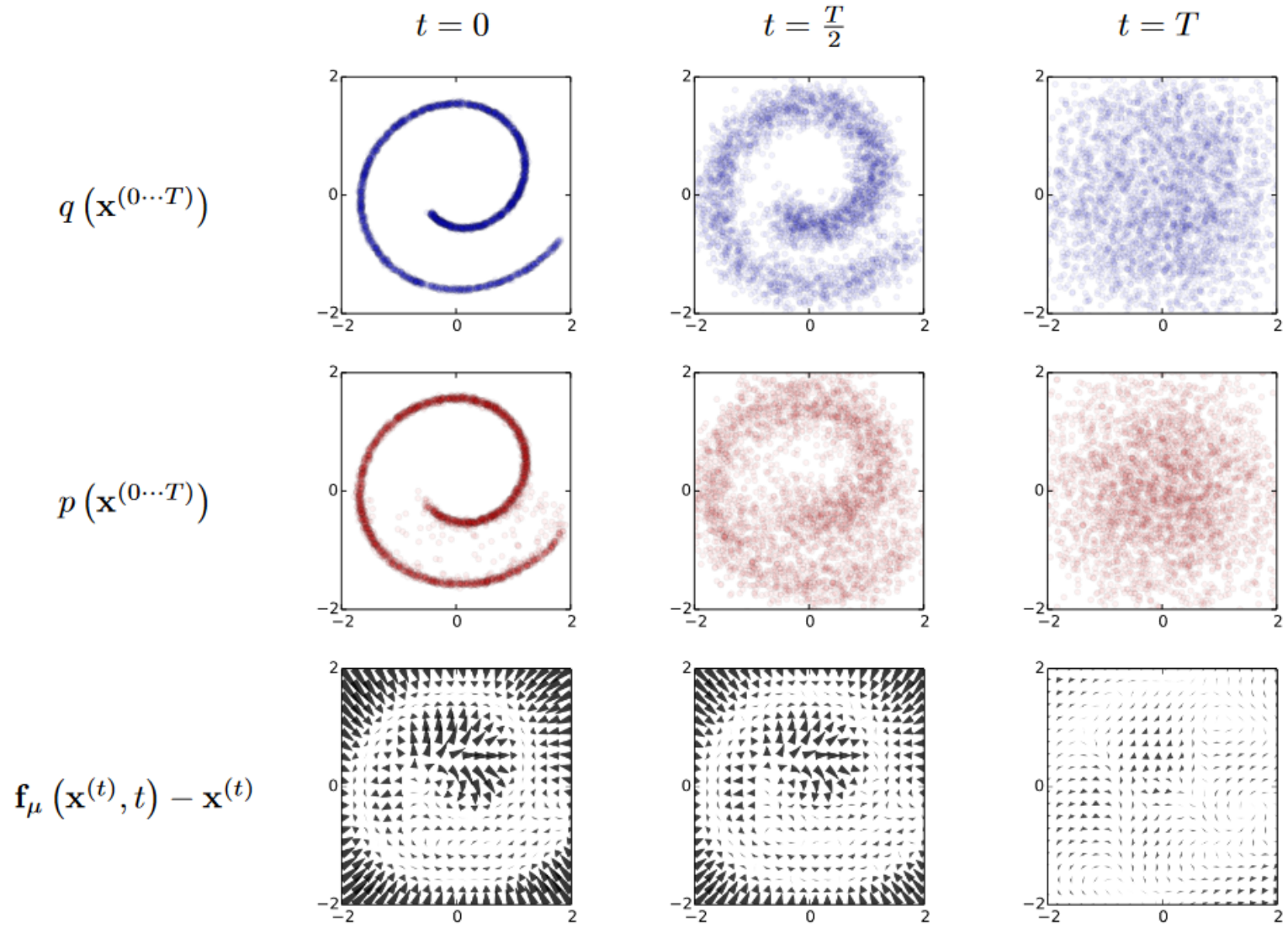


Figure 1. The proposed modeling framework trained on 2-d swiss roll data. The top row shows time slices from the forward trajectory $q\left(\mathbf{x}^{(0\cdots T)}\right)$. The data distribution (left) undergoes Gaussian diffusion, which gradually transforms it into an identity-covariance Gaussian (right). The middle row shows the corresponding time slices from the trained reverse trajectory $p\left(\mathbf{x}^{(0\cdots T)}\right)$. An identity-covariance Gaussian (right) undergoes a Gaussian diffusion process with learned mean and covariance functions, and is gradually transformed back into the data distribution (left). The bottom row shows the drift term, $\mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right) - \mathbf{x}^{(t)}$, for the same reverse diffusion process.

Figure 3. The proposed framework trained on the CIFAR-10 (Krizhevsky & Hinton, 2009) dataset. *(a)* Example training data. *(b)* Random samples generated by the diffusion model.

Understanding Diffusion Models: A Unified Perspective

Calvin Luo
Google Research, Brain Team
calvinluo@google.com

August 26, 2022

Contents

Introduction: Generative Models

Given observed samples $\boldsymbol{x}$ from a distribution of interest, the goal of a **generative model** is to learn to *model* its true data distribution $p(\boldsymbol{x})$. Once learned, we can *generate* new samples from our approximate model at will. Furthermore, under some formulations, we are able to use the learned model to evaluate the likelihood of observed or sampled data as well.

There are several well-known directions in current literature, that we will only introduce briefly at a high level. Generative Adversarial Networks (GANs) model the sampling procedure of a complex distribution, which is learned in an adversarial manner. Another class of generative models, termed "likelihood-based", seeks to learn a model that assigns a high likelihood to the observed data samples. This includes autoregressive models, normalizing flows, and Variational Autoencoders (VAEs). Another similar approach is energy-based modeling, in which a distribution is learned as an arbitrarily flexible energy function that is then normalized.

Armed with this new equation, we can retry the derivation resuming from the ELBO in Equation 37:

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\right] \tag{47}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \tag{48}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)\prod_{t=2}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \tag{49}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)\prod_{t=2}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)}\right] \tag{50}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)}\right] \tag{51}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}}\right] \tag{52}$$

https://arxiv.org/pdf/2208.11970.pdf

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)\cancel{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}}{\cancel{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}}}\right] \tag{53}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{\cancel{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}} + \log \frac{\cancel{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)} + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}\right] \tag{54}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)} + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}\right] \tag{55}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\right] + \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}\right] \tag{56}$$

$$= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] + \mathbb{E}_{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_T)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}\right] + \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}\right] \tag{57}$$

$$= \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T))}_{\text{prior matching term}} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}} \tag{58}$$

2020

Introduced U-Net as the denoiser architecture

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

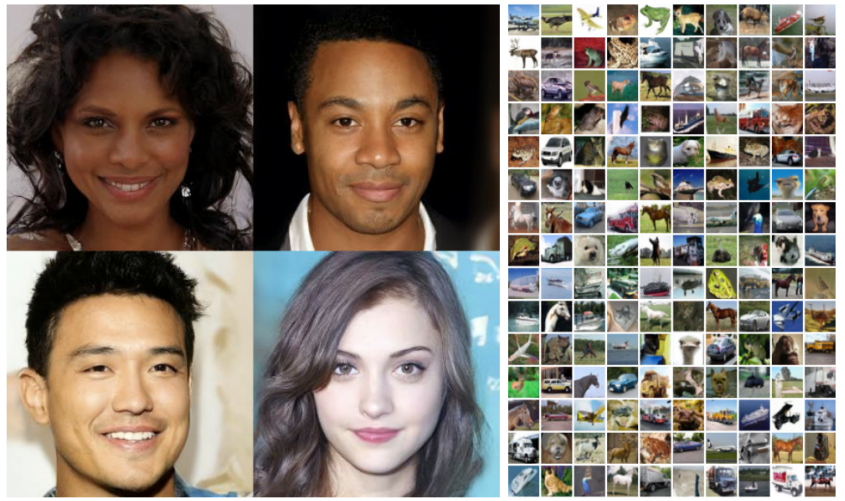
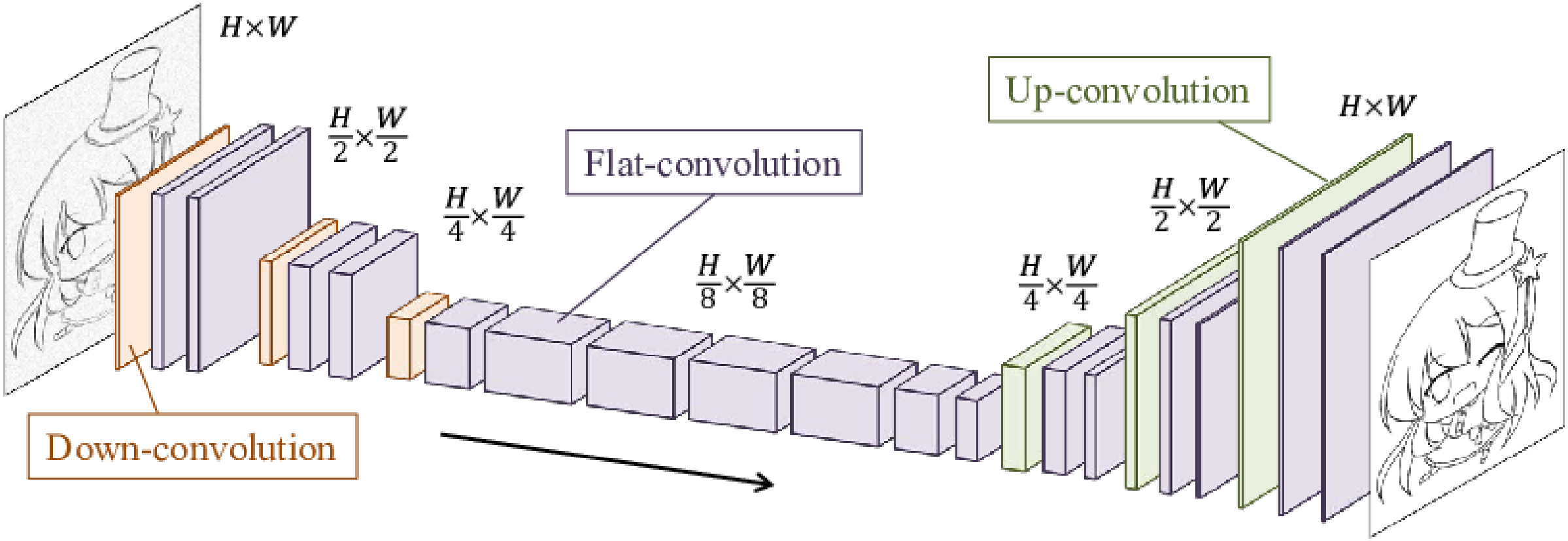


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

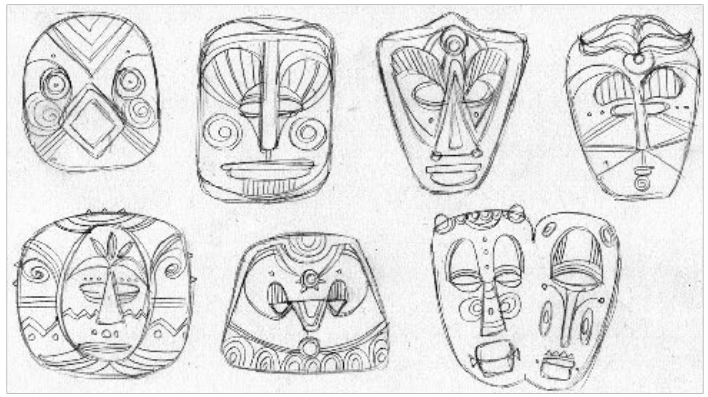
Encoder-decoder image networks: Output is the same as input, but denoised, inked, segmented…



http://hi.cs.waseda.ac.jp/%7Eesimo/en/research/sketch/

(a) Animals

(b) Kimono



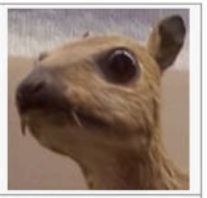
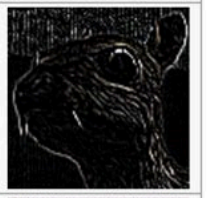
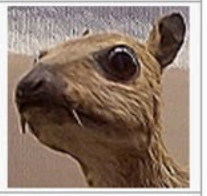
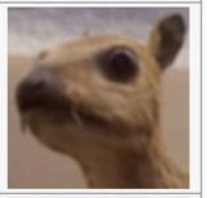
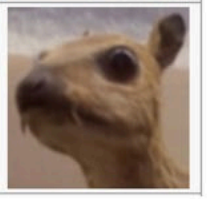
(c) Masks

(d) Book

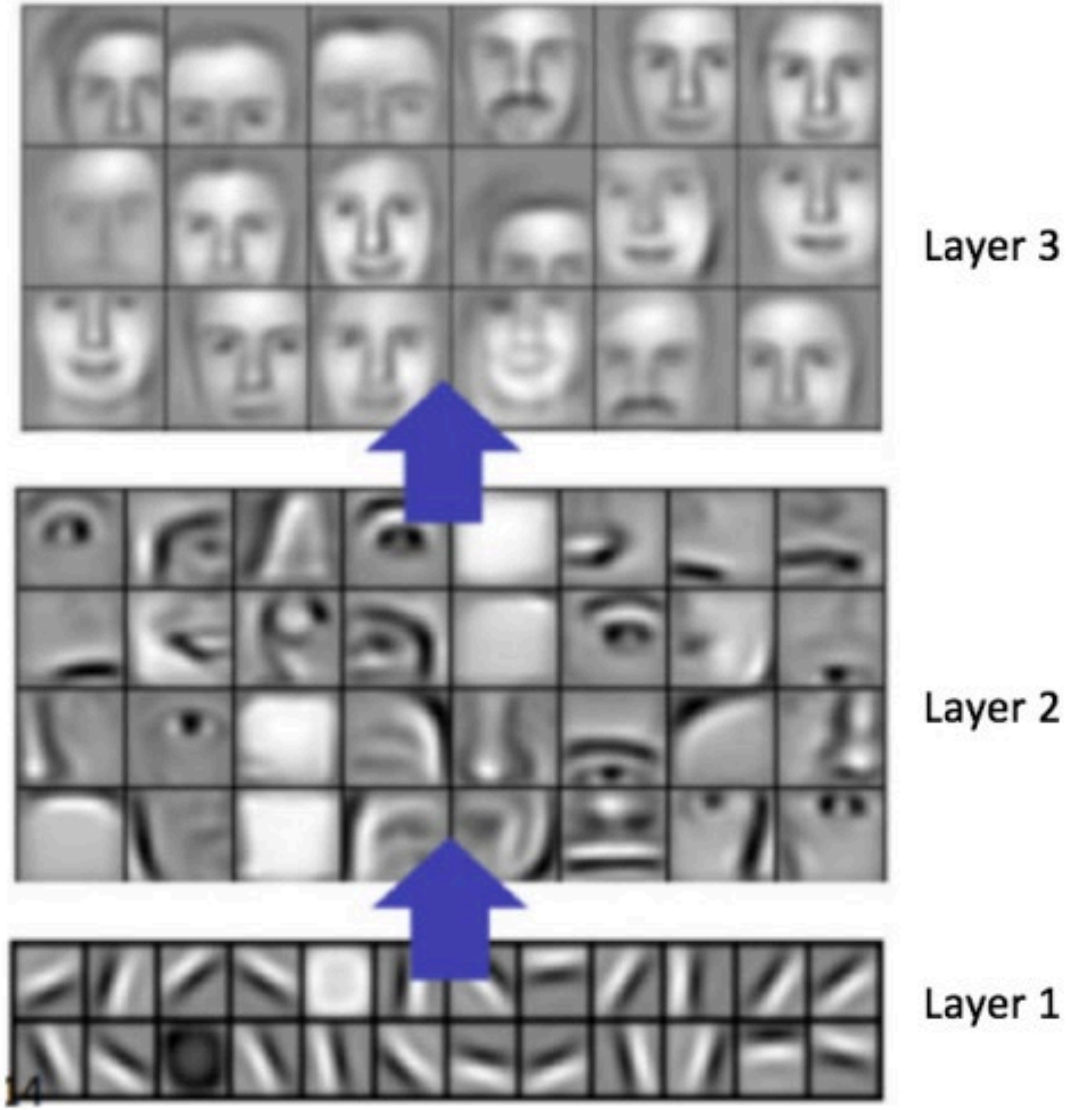
(e) Standing girl

Convolutions

- Each neuron in a neural network can be considered to measure the similarity between the neuron's weights **w** and the input data vector **x** using the dot product $\mathbf{w}^T\mathbf{x}$
- Convolutional networks: Each neuron only sees a small patch of the image and $\mathbf{w}^T\mathbf{x}$ measures similarity with the patch
- The output of neuron responses is also an image. The weights **w** can be considered an image *filter*

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Examples of filters learned in image classification



https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

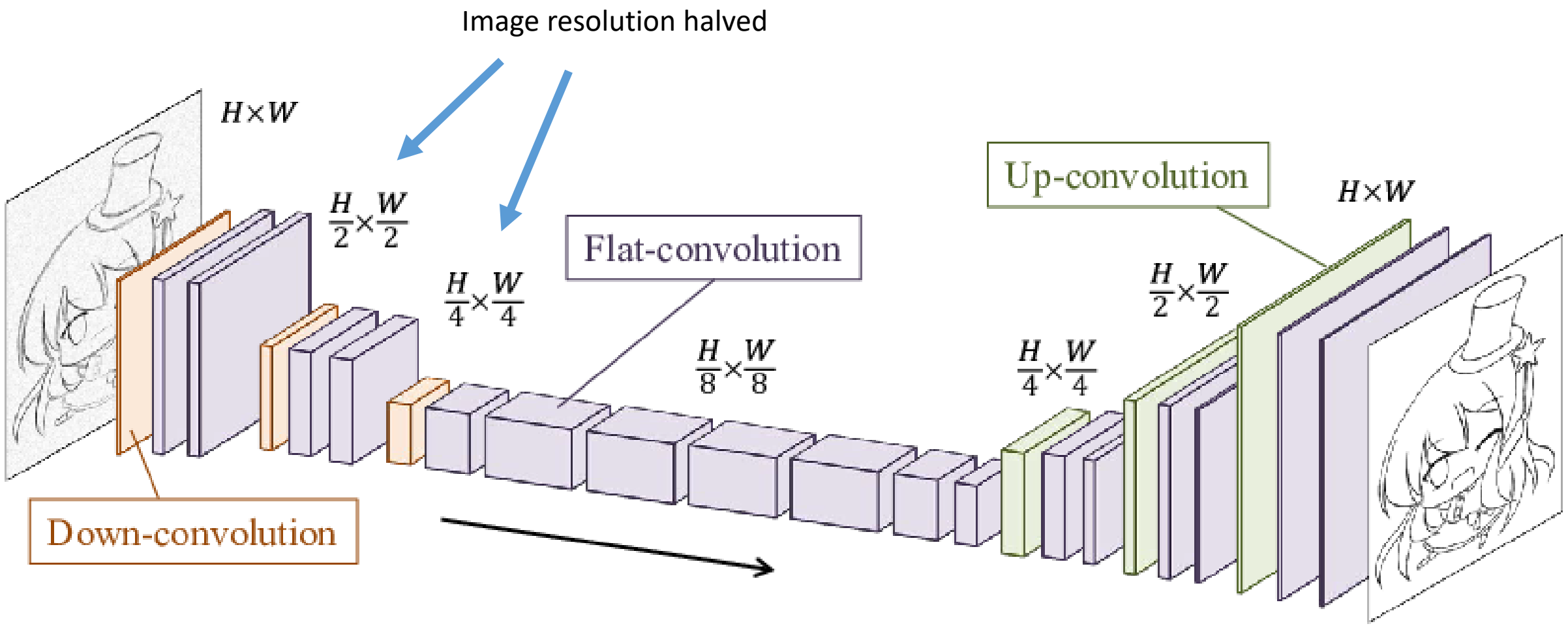
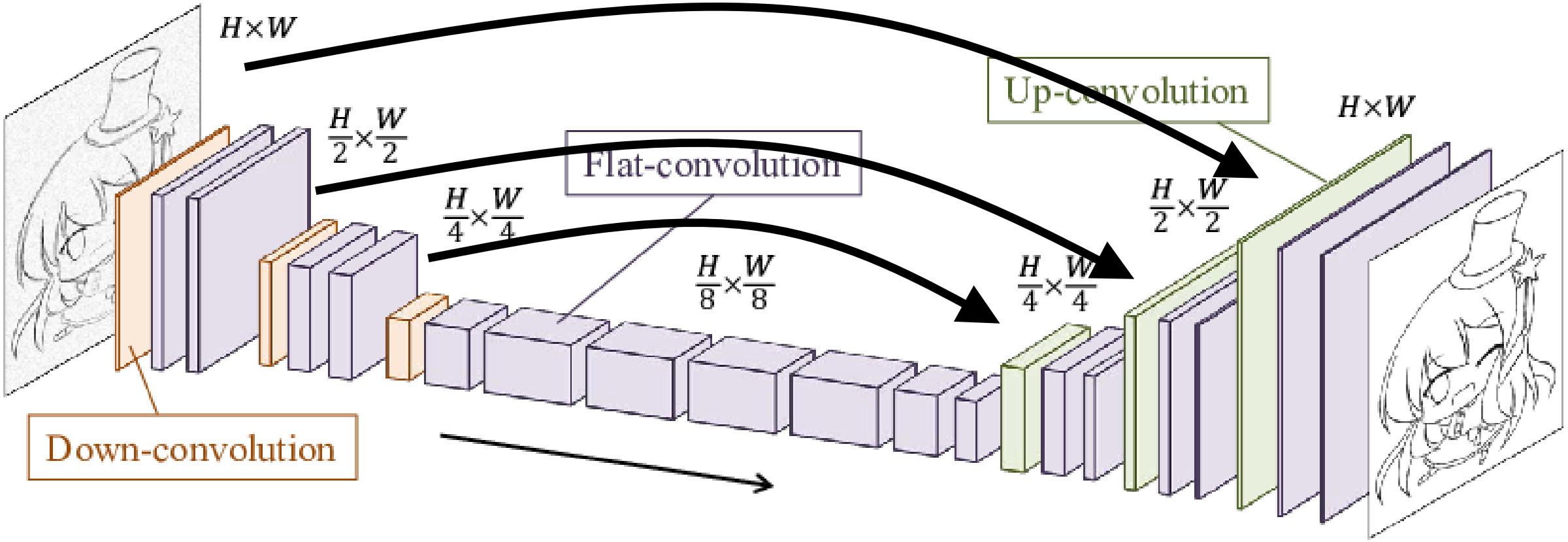


Image resolution halved
$H \times W$
$\frac{H}{2} \times \frac{W}{2}$
$\frac{H}{4} \times \frac{W}{4}$
$\frac{H}{8} \times \frac{W}{8}$
$\frac{H}{4} \times \frac{W}{4}$
$\frac{H}{2} \times \frac{W}{2}$
$H \times W$
Down-convolution
Flat-convolution
Up-convolution

U-Net adds skip-connections: high-resolution local decisions utilize local input info



The U-Net paper (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.

2021

- State-of-the-art image quality in small images
- No text prompting yet, but could be conditioned on a class number
- Not yet scaled to large images

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song*
Stanford University
yangsong@cs.stanford.edu

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Diederik P. Kingma
Google Brain
durk@google.com

Abhishek Kumar
Google Brain
abhishk@google.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Ben Poole
Google Brain
pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024×1024 images for the first time from a score-based generative model.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017; Du & Mordatch, 2019), have proven effective at generation of images (Song & Ermon, 2019; 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai

*Work partially done during an internship at Google Brain.

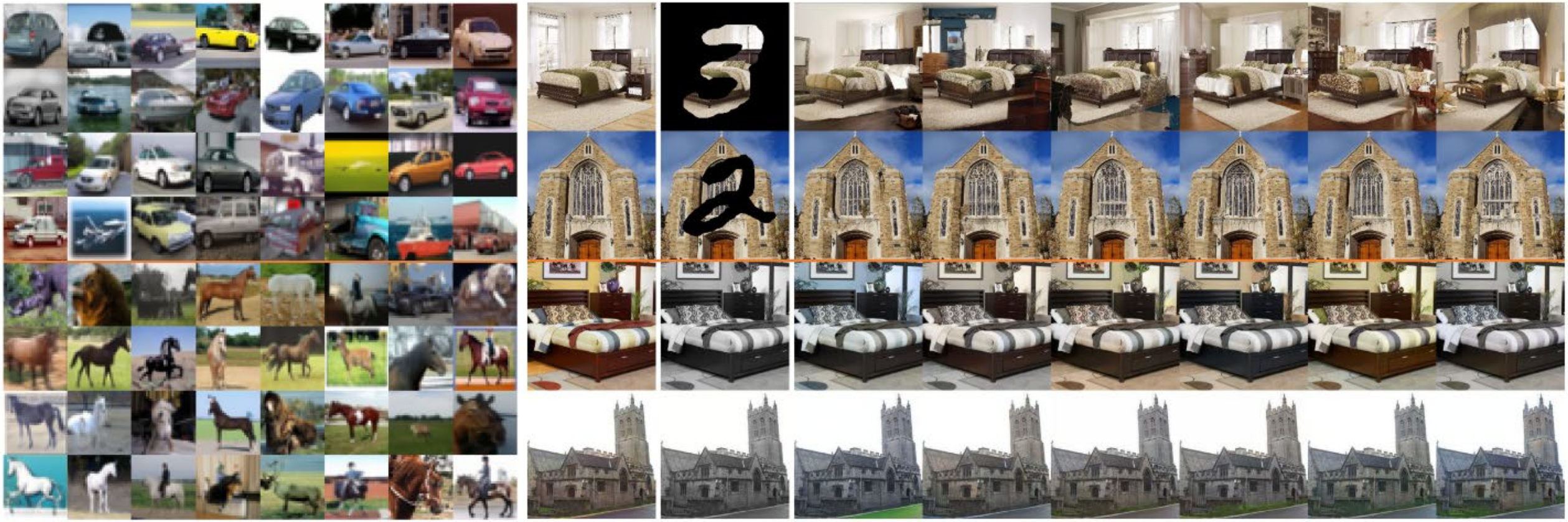


Figure 4: *Left*: Class-conditional samples on 32 × 32 CIFAR-10. Top four rows are automobiles and bottom four rows are horses. *Right:* Inpainting (top two rows) and colorization (bottom two rows) results on 256 × 256 LSUN. First column is the original image, second column is the masked/gray-scale image, remaining columns are sampled image completions or colorizations.

Beating GANs (2021)

- Scaling up the image synthesis and improving quality
- Still no text prompting

Diffusion Models Beat GANs on Image Synthesis

Prafulla Dhariwal*
OpenAI
prafulla@openai.com

Alex Nichol*
OpenAI
alex@openai.com

Abstract

We show that diffusion models can achieve image sample quality superior to the current state-of-the-art generative models. We achieve this on unconditional image synthesis by finding a better architecture through a series of ablations. For conditional image synthesis, we further improve sample quality with classifier guidance: a simple, compute-efficient method for trading off diversity for fidelity using gradients from a classifier. We achieve an FID of 2.97 on ImageNet 128×128, 4.59 on ImageNet 256×256, and 7.72 on ImageNet 512×512, and we match BigGAN-deep even with as few as 25 forward passes per sample, all while maintaining better coverage of the distribution. Finally, we find that classifier guidance combines well with upsampling diffusion models, further improving FID to 3.94 on ImageNet 256×256 and 3.85 on ImageNet 512×512. We release our code at https://github.com/openai/guided-diffusion.

1 Introduction

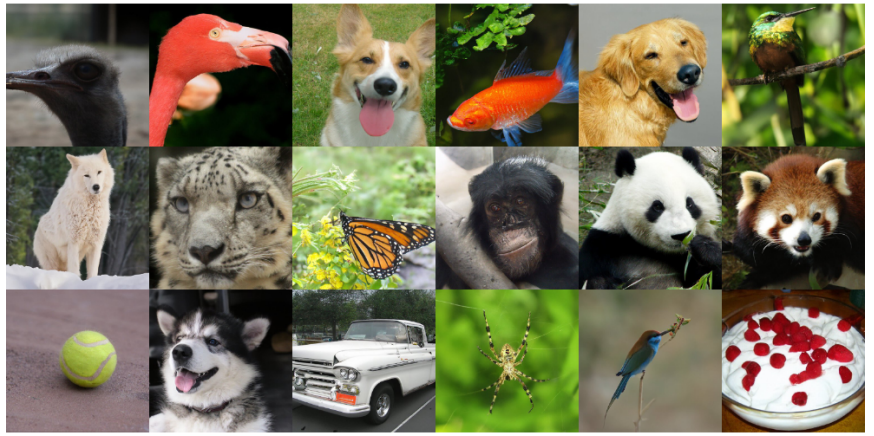


Figure 1: Selected samples from our best ImageNet 512×512 model (FID 3.85)

Over the past few years, generative models have gained the ability to generate human-like natural language [6], infinite high-quality synthetic images [5, 28, 51] and highly diverse human speech and music [64, 13]. These models can be used in a variety of ways, such as generating images from text prompts [72, 50] or learning useful feature representations [14, 7]. While these models are already

*Equal contribution

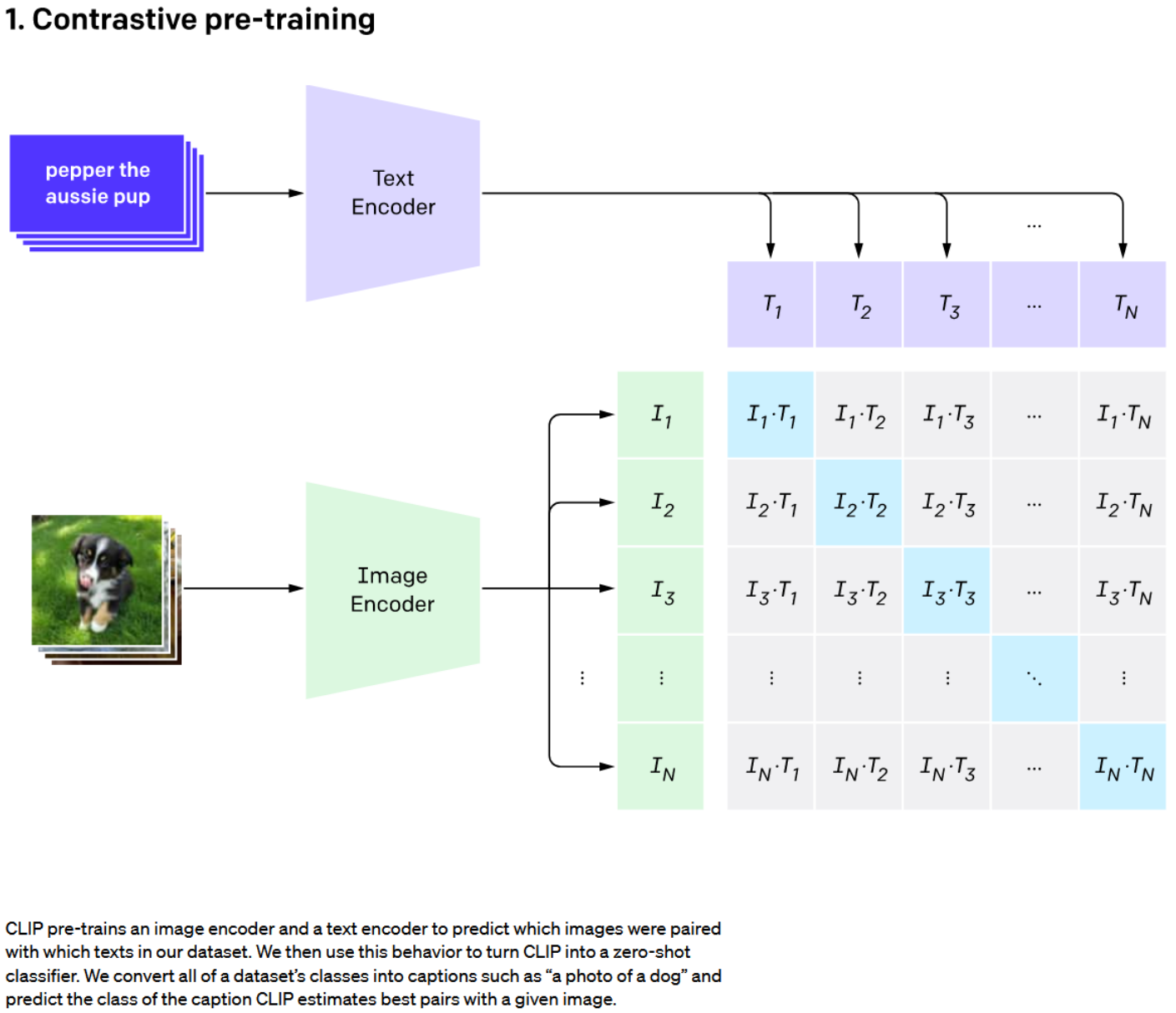
CLIP-guided diffusion (2021)

- Text prompting added by artist and coder Catherine Crowson
- No paper, only Colab that evolved into Disco Diffusion
- Introduced CLIP (next slide) as the guiding mechanism for the denoising

https://arxiv.org/abs/2112.10741

CLIP

- Contrastive Language Image Pretraining
- Perhaps OpenAI's highest-impact open model
- Embeds both text and images so that the embeddings of an image and its caption are approximately the same
- Diffusion: CLIP can nudge the denoising so that image's CLIP embedding gets closer to the prompt's embedding



CLIP pre-trains an image encoder and a text encoder to predict which images were paired with which texts in our dataset. We then use this behavior to turn CLIP into a zero-shot classifier. We convert all of a dataset's classes into captions such as "a photo of a dog" and predict the class of the caption CLIP estimates best pairs with a given image.

https://openai.com/research/clip

CLIP building blocks

- The text encoder is a Transformer LLM
- The image encoder is a Vision Transformer (ViT) LLM, where the tokens are image patches (4x4 tokens per image)

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy*,†, Lucas Beyer*, Alexander Kolesnikov*, Dirk Weissenborn*, Xiaohua Zhai*, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby*,†
*equal technical contribution, †equal advising
Google Research, Brain Team
{adosovitskiy, neilhoulsby}@google.com

ABSTRACT

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.[1]

https://arxiv.org/abs/2010.11929

1 INTRODUCTION

Self-attention-based architectures, in particular Transformers (Vaswani et al., 2017), have become the model of choice in natural language processing (NLP). The dominant approach is to pre-train on a large text corpus and then fine-tune on a smaller task-specific dataset (Devlin et al., 2019). Thanks to Transformers' computational efficiency and scalability, it has become possible to train models of unprecedented size, with over 100B parameters (Brown et al., 2020; Lepikhin et al., 2020). With the models and datasets growing, there is still no sign of saturating performance.

In computer vision, however, convolutional architectures remain dominant (LeCun et al., 1989; Krizhevsky et al., 2012; He et al., 2016). Inspired by NLP successes, multiple works try combining CNN-like architectures with self-attention (Wang et al., 2018; Carion et al., 2020), some replacing the convolutions entirely (Ramachandran et al., 2019; Wang et al., 2020a). The latter models, while theoretically efficient, have not yet been scaled effectively on modern hardware accelerators due to the use of specialized attention patterns. Therefore, in large-scale image recognition, classic ResNet-like architectures are still state of the art (Mahajan et al., 2018; Xie et al., 2020; Kolesnikov et al., 2020).

Inspired by the Transformer scaling successes in NLP, we experiment with applying a standard Transformer directly to images, with the fewest possible modifications. To do so, we split an image into patches and provide the sequence of linear embeddings of these patches as an input to a Transformer. Image patches are treated the same way as tokens (words) in an NLP application. We train the model on image classification in supervised fashion.

When trained on mid-sized datasets such as ImageNet without strong regularization, these models yield modest accuracies of a few percentage points below ResNets of comparable size. This seemingly discouraging outcome may be expected: Transformers lack some of the inductive biases

[1] Fine-tuning code and pre-trained models are available at https://github.com/google-research/vision_transformer

CLIP-guided diffusion (2021)

- Further development of the CLIP-guided approach with OpenAI's resources
- Found that *classifier-free guidance* using CLIP is better than vanilla CLIP guidance

https://arxiv.org/abs/2112.10741

GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models

Alex Nichol * Prafulla Dhariwal * Aditya Ramesh * Pranav Shyam Pamela Mishkin Bob McGrew
Ilya Sutskever Mark Chen

Abstract

Diffusion models have recently been shown to generate high-quality synthetic images, especially when paired with a guidance technique to trade off diversity for fidelity. We explore diffusion models for the problem of text-conditional image synthesis and compare two different guidance strategies: CLIP guidance and classifier-free guidance. We find that the latter is preferred by human evaluators for both photorealism and caption similarity, and often produces photorealistic samples. Samples from a 3.5 billion parameter text-conditional diffusion model using classifier-free guidance are favored by human evaluators to those from DALL-E, even when the latter uses expensive CLIP reranking. Additionally, we find that our models can be fine-tuned to perform image inpainting, enabling powerful text-driven image editing. We train a smaller model on a filtered dataset and release the code and weights at https://github.com/openai/glide-text2im.

1. Introduction

Images, such as illustrations, paintings, and photographs, can often be easily described using text, but can require specialized skills and hours of labor to create. Therefore, a tool capable of generating realistic images from natural language can empower humans to create rich and diverse visual content with unprecedented ease. The ability to edit images using natural language further allows for iterative refinement and fine-grained control, both of which are critical for real world applications.

Recent text-conditional image models are capable of synthesizing images from free-form text prompts, and can compose unrelated objects in semantically plausible ways (Xu et al., 2017; Zhu et al., 2019; Tao et al., 2020; Ramesh et al., 2021; Zhang et al., 2021). However, they are not yet able to generate photorealistic images that capture all aspects of their corresponding text prompts.

On the other hand, unconditional image models can synthesize photorealistic images (Brock et al., 2018; Karras et al., 2019a;b; Razavi et al., 2019), sometimes with enough fidelity that humans can't distinguish them from real images (Zhou et al., 2019). Within this line of research, diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2020b) have emerged as a promising family of generative models, achieving state-of-the-art sample quality on a number of image generation benchmarks (Ho et al., 2020; Dhariwal & Nichol, 2021; Ho et al., 2021).

To achieve photorealism in the class-conditional setting, Dhariwal & Nichol (2021) augmented diffusion models with *classifier guidance*, a technique which allows diffusion models to condition on a classifier's labels. The classifier is first trained on noised images, and during the diffusion sampling process, gradients from the classifier are used to guide the sample towards the label. Ho & Salimans (2021) achieved similar results without a separately trained classifier through the use of *classifier-free guidance*, a form of guidance that interpolates between predictions from a diffusion model with and without labels.

Motivated by the ability of guided diffusion models to generate photorealistic samples and the ability of text-to-image models to handle free-form prompts, we apply guided diffusion to the problem of text-conditional image synthesis. First, we train a 3.5 billion parameter diffusion model that uses a text encoder to condition on natural language descriptions. Next, we compare two techniques for guiding diffusion models towards text prompts: CLIP guidance and classifier-free guidance. Using human and automated evaluations, we find that classifier-free guidance yields higher-quality images.

We find that samples from our model generated with classifier-free guidance are both photorealistic and reflect a wide breadth of world knowledge. When evaluated by human judges, our samples are preferred to those from DALL-E (Ramesh et al., 2021) 87% of the time when evaluated for photorealism, and 69% of the time when evaluated for caption similarity.

*Equal contribution. Correspondence to alex@openai.com, prafulla@openai.com, aramesh@openai.com

“a hedgehog using a calculator”

“a corgi wearing a red bowtie and a purple party hat”

“robots meditating in a vipassana retreat”

“a fall landscape with a small cottage next to a lake”

“a surrealist dream-like oil painting by salvador dalí of a cat playing checkers”

“a professional photo of a sunset behind the grand canyon”

“a high-quality oil painting of a psychedelic hamster dragon”

“an illustration of albert einstein wearing a superhero costume”

Classifier-free guidance

Use CLIP in a way that allows higher quality at the cost of reduced diversity

https://arxiv.org/abs/2207.12598

CLASSIFIER-FREE DIFFUSION GUIDANCE

Jonathan Ho & Tim Salimans
Google Research, Brain team
{jonathanho,salimans}@google.com

ABSTRACT

Classifier guidance is a recently introduced method to trade off mode coverage and sample fidelity in conditional diffusion models post training, in the same spirit as low temperature sampling or truncation in other types of generative models. Classifier guidance combines the score estimate of a diffusion model with the gradient of an image classifier and thereby requires training an image classifier separate from the diffusion model. It also raises the question of whether guidance can be performed without a classifier. We show that guidance can be indeed performed by a pure generative model without such a classifier: in what we call classifier-free guidance, we jointly train a conditional and an unconditional diffusion model, and we combine the resulting conditional and unconditional score estimates to attain a trade-off between sample quality and diversity similar to that obtained using classifier guidance.

1 INTRODUCTION

Diffusion models have recently emerged as an expressive and flexible family of generative models, delivering competitive sample quality and likelihood scores on image and audio synthesis tasks (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020; Song et al., 2021b; Kingma et al., 2021; Song et al., 2021a). These models have delivered audio synthesis performance rivaling the quality of autoregressive models with substantially fewer inference steps (Chen et al., 2021; Kong et al., 2021), and they have delivered ImageNet generation results outperforming BigGAN-deep (Brock et al., 2019) and VQ-VAE-2 (Razavi et al., 2019) in terms of FID score and classification accuracy score (Ho et al., 2021; Dhariwal & Nichol, 2021).

Dhariwal & Nichol (2021) proposed *classifier guidance*, a technique to boost the sample quality of a diffusion model using an extra trained classifier. Prior to classifier guidance, it was not known how to generate "low temperature" samples from a diffusion model similar to those produced by truncated BigGAN (Brock et al., 2019) or low temperature Glow (Kingma & Dhariwal, 2018): naive attempts, such as scaling the model score vectors or decreasing the amount of Gaussian noise added during diffusion sampling, are ineffective (Dhariwal & Nichol, 2021). Classifier guidance instead mixes a diffusion model's score estimate with the input gradient of the log probability of a

Figure 1: Classifier-free guidance on the malamute class for a 64x64 ImageNet diffusion model. Left to right: increasing amounts of classifier-free guidance, starting from non-guided samples on the left.

A short version of this paper appeared in the NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications: https://openreview.net/pdf?id=qw8AKxfYbI

DALL-E 2 (2022)

- Uses classifier-free guidance with CLIP
- Learns a correction from CLIP text embedding to the corresponding image embedding
- Adds an upscaling stage

https://cdn.openai.com/papers/dall-e-2.pdf

Hierarchical Text-Conditional Image Generation with CLIP Latents

Aditya Ramesh* OpenAI aramesh@openai.com

Prafulla Dhariwal* OpenAI prafulla@openai.com

Alex Nichol* OpenAI alex@openai.com

Casey Chu* OpenAI casey@openai.com

Mark Chen OpenAI mark@openai.com

Abstract

Contrastive models like CLIP have been shown to learn robust representations of images that capture both semantics and style. To leverage these representations for image generation, we propose a two-stage model: a prior that generates a CLIP image embedding given a text caption, and a decoder that generates an image conditioned on the image embedding. We show that explicitly generating image representations improves image diversity with minimal loss in photorealism and caption similarity. Our decoders conditioned on image representations can also produce variations of an image that preserve both its semantics and style, while varying the non-essential details absent from the image representation. Moreover, the joint embedding space of CLIP enables language-guided image manipulations in a zero-shot fashion. We use diffusion models for the decoder and experiment with both autoregressive and diffusion models for the prior, finding that the latter are computationally more efficient and produce higher-quality samples.

1 Introduction

Recent progress in computer vision has been driven by scaling models on large datasets of captioned images collected from the internet [10, 44, 60, 39, 31, 16]. Within this framework, CLIP [39] has emerged as a successful representation learner for images. CLIP embeddings have a number of desirable properties: they are robust to image distribution shift, have impressive zero-shot capabilities, and have been fine-tuned to achieve state-of-the-art results on a wide variety of vision and language tasks [45]. Concurrently, diffusion models [46, 48, 25] have emerged as a promising generative modeling framework, pushing the state-of-the-art on image and video generation tasks [11, 26, 24]. To achieve best results, diffusion models leverage a guidance technique [11, 24] which improves sample fidelity (for images, photorealism) at the cost of sample diversity.

In this work, we combine these two approaches for the problem of text-conditional image generation. We first train a diffusion *decoder* to invert the CLIP image *encoder*. Our inverter is non-deterministic, and can produce multiple images corresponding to a given image embedding. The presence of an encoder and its approximate inverse (the decoder) allows for capabilities beyond text-to-image translation. As in GAN inversion [62, 55], encoding and decoding an input image produces semantically similar output images (Figure 3). We can also interpolate between input images by inverting interpolations of their image embeddings (Figure 4). However, one notable advantage of using the CLIP latent space is the ability to semantically modify images by moving in the direction of any encoded text vector (Figure 5), whereas discovering these directions in GAN latent space involves

*Equal contribution

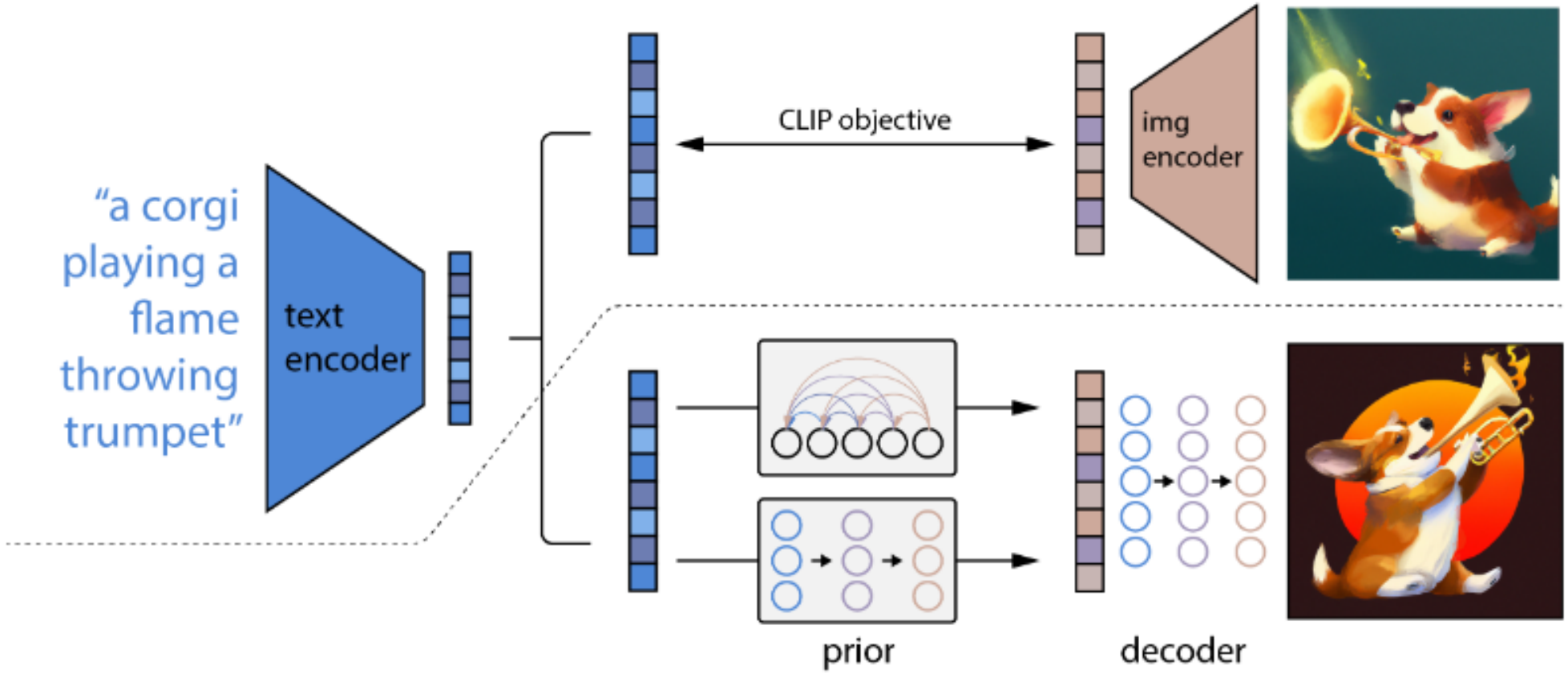


Figure 2: A high-level overview of unCLIP. Above the dotted line, we depict the CLIP training process, through which we learn a joint representation space for text and images. Below the dotted line, we depict our text-to-image generation process: a CLIP text embedding is first fed to an autoregressive or diffusion prior to produce an image embedding, and then this embedding is used to condition a diffusion decoder which produces a final image. Note that the CLIP model is frozen during training of the prior and decoder.

https://cdn.openai.com/papers/dall-e-2.pdf

DALL-E 3 (2023)

- Problem: Most of the training image captions are bad quality (E.g., alt-text of scraped images may be wrong or only describe a part of the image)
- Solution:
  - Create synthetic very elaborate and verbose captions using a captioning model
  - Instruct GPT-4 to add detail to the user's prompt if needed
- Chat-based interface: Bing chat, MS Copilot app, GPT-4

https://cdn.openai.com/papers/dall-e-3.pdf

Improving Image Generation with Better Captions

James Betker*† jbetker@openai.com
Gabriel Goh*† ggoh@openai.com
Li Jing*† lijing@openai.com
Tim Brooks†
Jianfeng Wang‡ Linjie Li‡ Long Ouyang† Juntang Zhuang† Joyce Lee† Yufei Guo†
Wesam Manassra† Prafulla Dhariwal† Casey Chu† Yunxin Jiao†
Aditya Ramesh*† aramesh@openai.com

Abstract

We show that prompt following abilities of text-to-image models can be substantially improved by training on highly descriptive generated image captions. Existing text-to-image models struggle to follow detailed image descriptions and often ignore words or confuse the meaning of prompts. We hypothesize that this issue stems from noisy and inaccurate image captions in the training dataset. We address this by training a bespoke image captioner and use it to recaption the training dataset. We then train several text-to-image models and find that training on these synthetic captions reliably improves prompt following ability. Finally, we use these findings to build DALL-E 3: a new text-to-image generation system, and benchmark its performance on an evaluation designed to measure prompt following, coherence, and aesthetics, finding that it compares favorably to competitors. We publish samples and code for these evaluations so that future research can continue optimizing this important aspect of text-to-image systems.

1 Introduction

Recent advances in generative modeling have allowed text-to-image generative models to achieve drastic performance improvements. In particular, tackling the problem with sampling-based approaches such as autoregressive generative modeling[27, 2, 1, 20, 30] or using diffusion processes[25, 6, 11, 12, 19, 22] have allowed us to decompose the problem of image generation into small, discrete steps which are more tractable for neural networks to learn.

In parallel, researchers have found ways to build image generators out of stacks of self-attention layers[15, 3, 4]. Decoupling image generation from the implicit spatial biases of convolutions has allowed text-to-image models to reliably improve via the well-studied scaling properties of transformers.

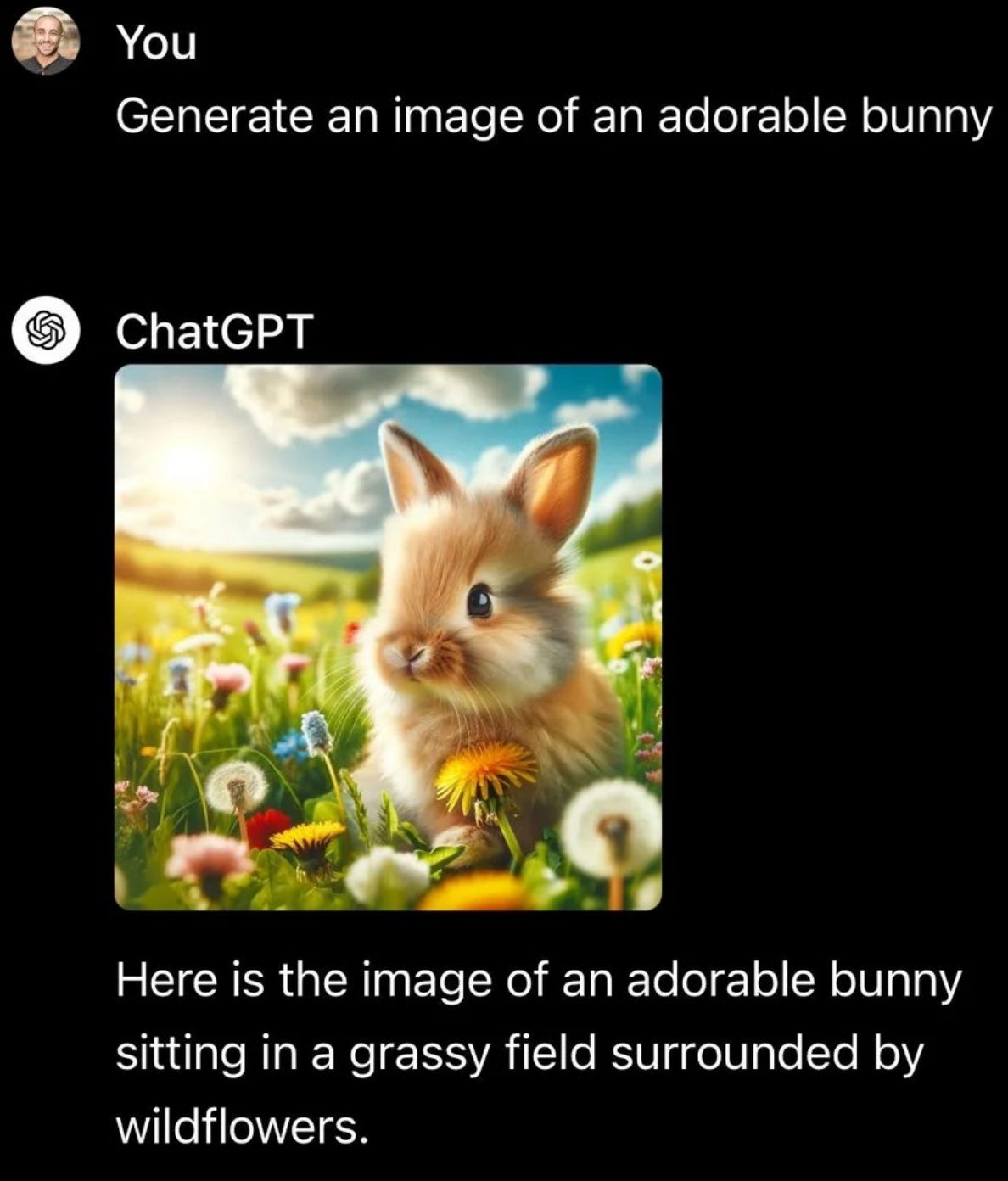
Combined with a sufficiently large dataset, these approaches have enabled the training of large text-to-image models which are capable of generating imagery which is rapidly approaching the quality of photographs and artwork that humans can produce.

*Equal contribution
†OpenAI
‡Microsoft

Chat integration

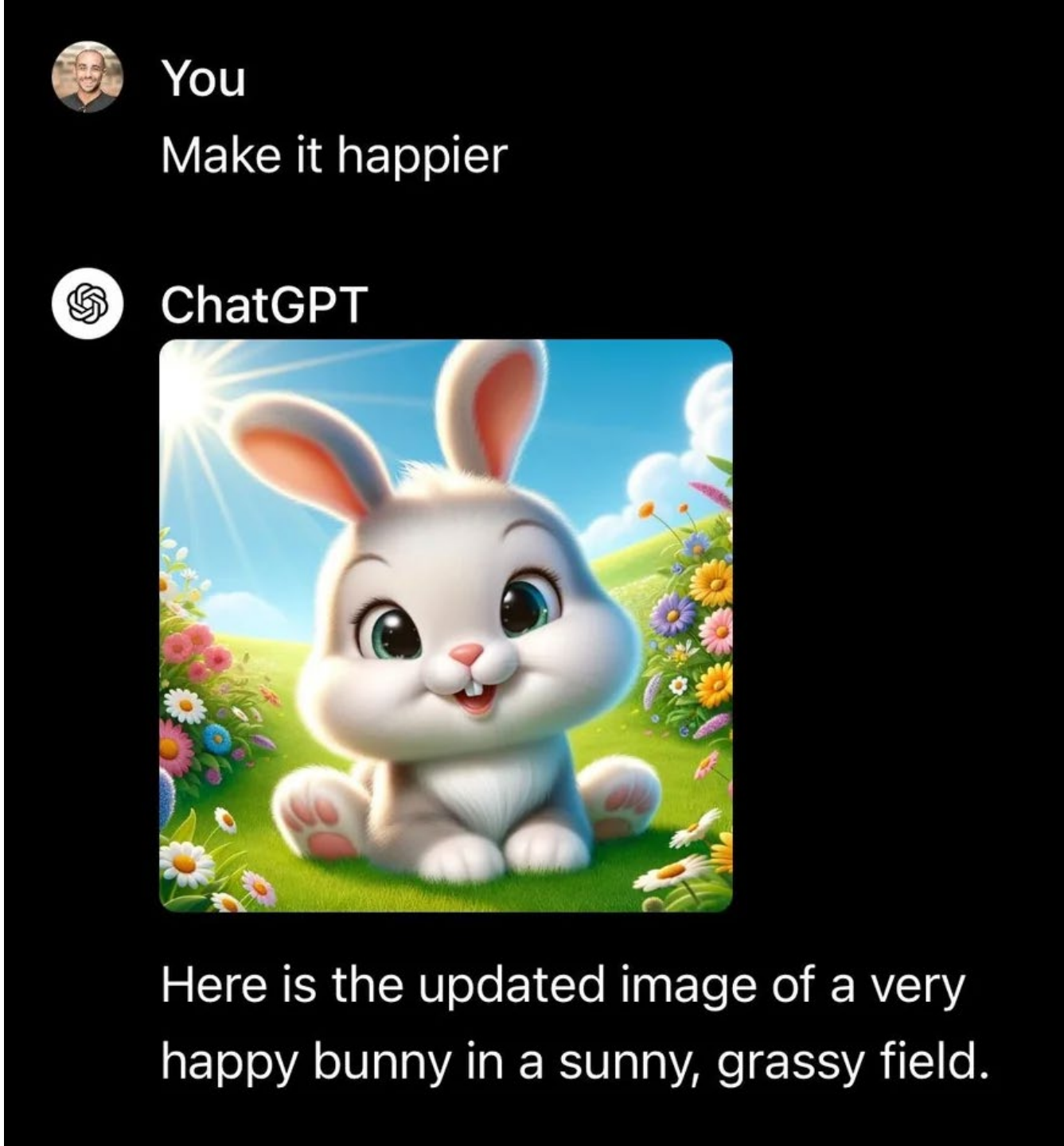
- Allows one to chat with the “AI artist”, suggesting changes

https://www.reddit.com/r/ChatGPT/comments/1839fgo/asking_gpt_to_make_a_bunny_happier/



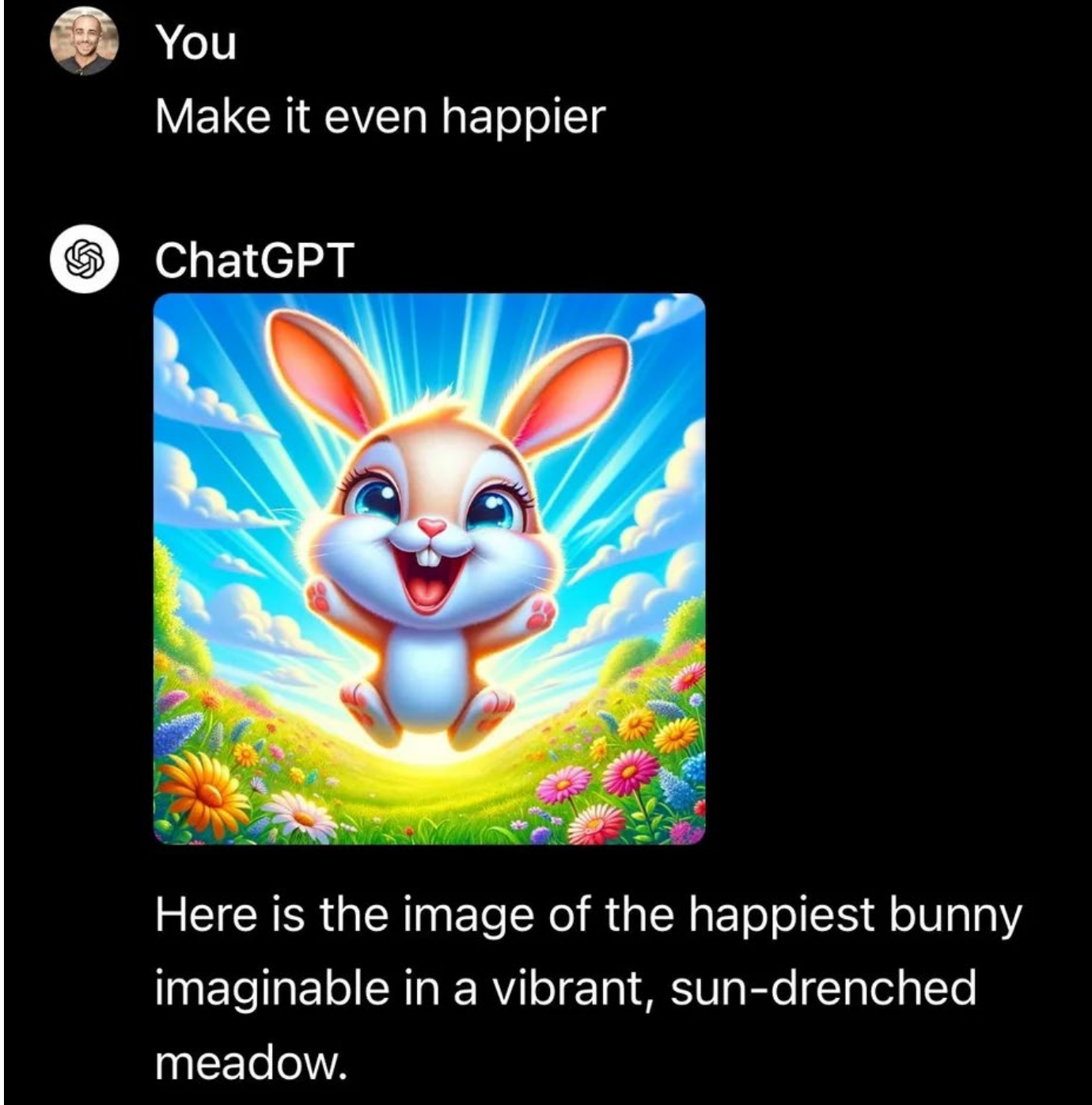
Chat integration

- Allows one to chat with the “AI artist”, suggesting changes



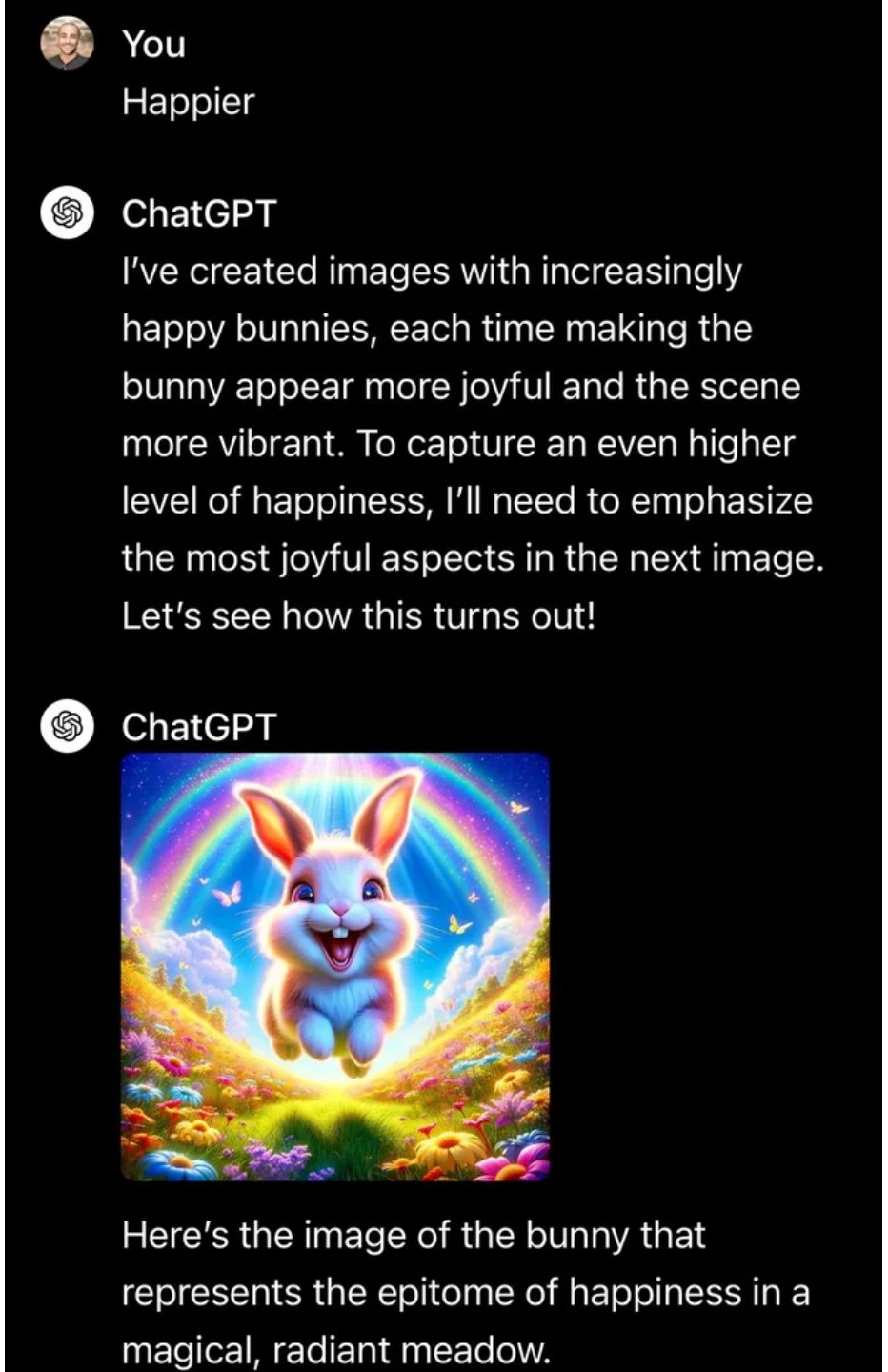
Chat integration

- Allows one to chat with the “AI artist”, suggesting changes



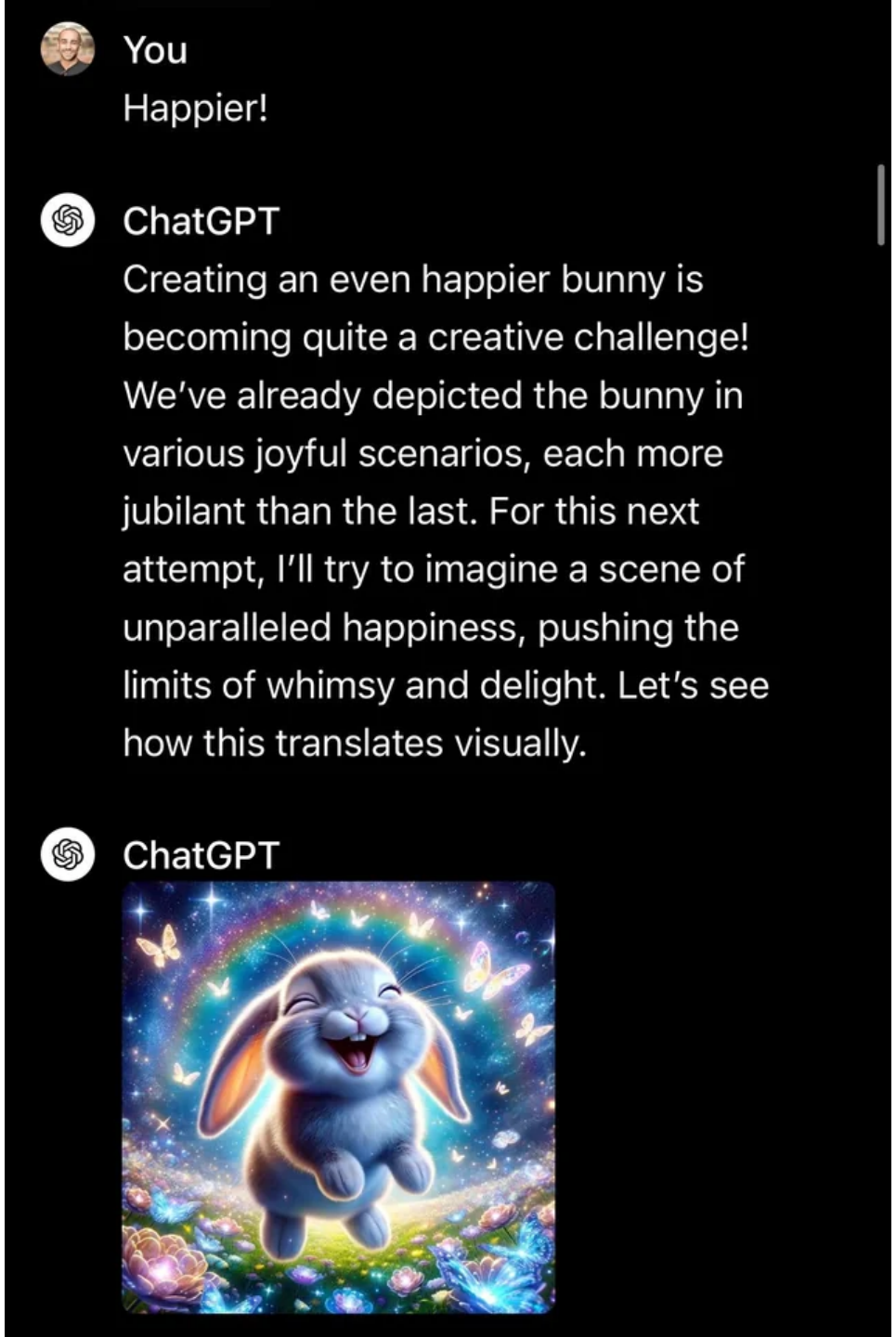
Chat integration

- Allows one to chat with the “AI artist”, suggesting changes



Chat integration

- Allows one to chat with the “AI artist”, suggesting changes



AaltoMediaAI @aaltomediaai · Nov 9, 2023

A fascinating Reddit thread of DALL•E images based on emoji prompts
reddit.com/r/dalle2/s/QwP...

https://www.reddit.com/r/dalle2/comments/17qqhpr/when_you_just_put_emojis_in_the_prompt/

Stable Diffusion (2022)

- Key feature: Instead of pixels, the denoising is done in a latent space of image patch vectors
- Allows conditioning with text prompt and/or image
- Also adds Transformer-like attention to the denoiser U-net
- Fully open source, including trained models.

High-Resolution Image Synthesis with Latent Diffusion Models

Robin Rombach[1] * Andreas Blattmann[1] * Dominik Lorenz[1] Patrick Esser[R] Björn Ommer[1]
[1]Ludwig Maximilian University of Munich & IWR, Heidelberg University, Germany [R]Runway ML

https://github.com/CompVis/latent-diffusion

https://arxiv.org/abs/2112.10752

Abstract

By decomposing the image formation process into a sequential application of denoising autoencoders, diffusion models (DMs) achieve state-of-the-art synthesis results on image data and beyond. Additionally, their formulation allows for a guiding mechanism to control the image generation process without retraining. However, since these models typically operate directly in pixel space, optimization of powerful DMs often consumes hundreds of GPU days and inference is expensive due to sequential evaluations. To enable DM training on limited computational resources while retaining their quality and flexibility, we apply them in the latent space of powerful pretrained autoencoders. In contrast to previous work, training diffusion models on such a representation allows for the first time to reach a near-optimal point between complexity reduction and detail preservation, greatly boosting visual fidelity. By introducing cross-attention layers into the model architecture, we turn diffusion models into powerful and flexible generators for general conditioning inputs such as text or bounding boxes and high-resolution synthesis becomes possible in a convolutional manner. Our latent diffusion models (LDMs) achieve new state-of-the-art scores for image inpainting and class-conditional image synthesis and highly competitive performance on various tasks, including text-to-image synthesis, unconditional image generation and super-resolution, while significantly reducing computational requirements compared to pixel-based DMs.

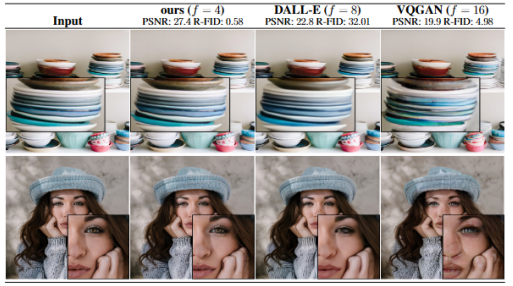


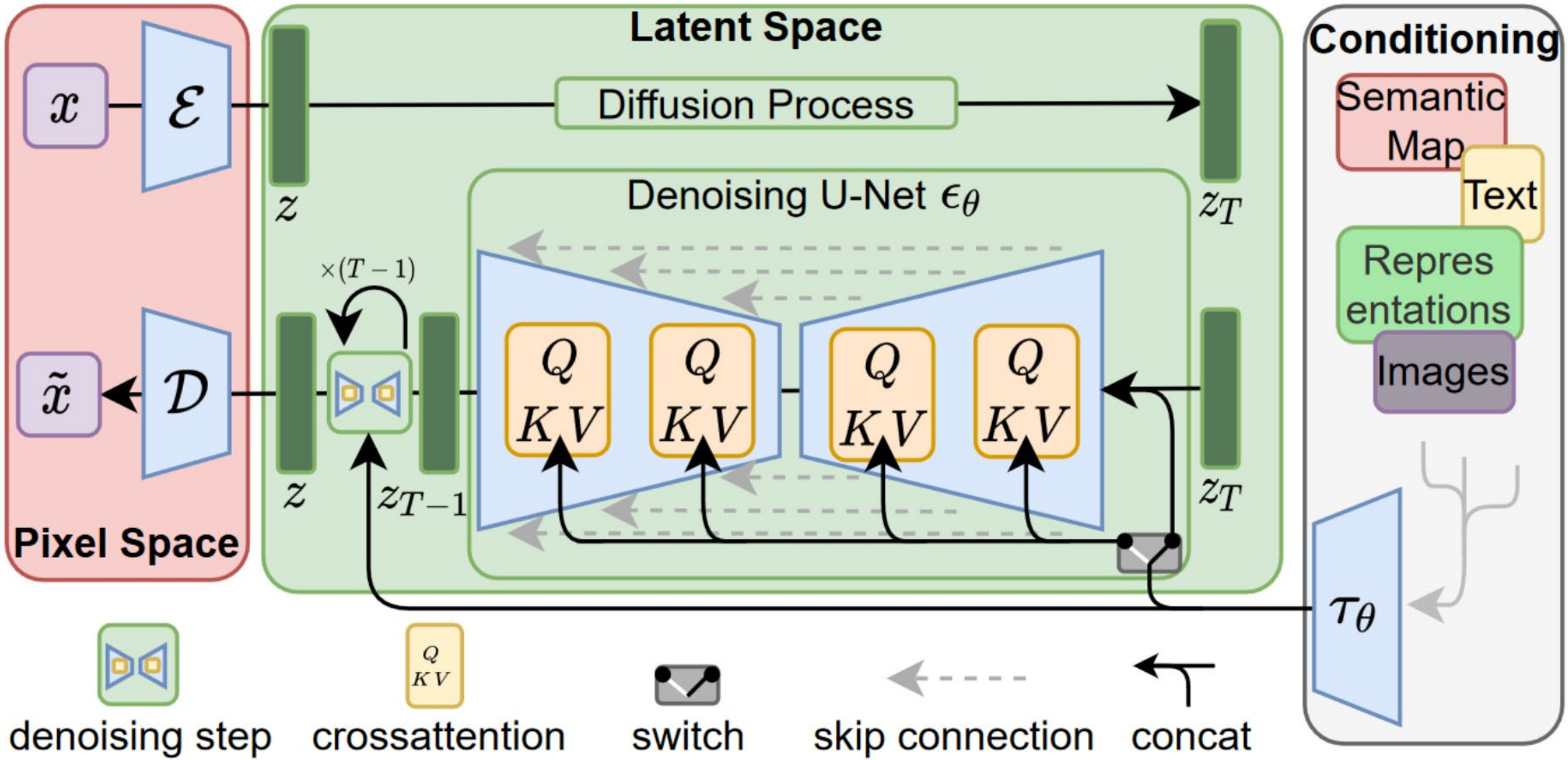
Figure 1. Boosting the upper bound on achievable quality with less agressive downsampling. Since diffusion models offer excellent inductive biases for spatial data, we do not need the heavy spatial downsampling of related generative models in latent space, but can still greatly reduce the dimensionality of the data via suitable autoencoding models, see Sec. 3. Images are from the DIV2K [1] validation set, evaluated at 512^2 px. We denote the spatial downsampling factor by f. Reconstruction FIDs [29] and PSNR are calculated on ImageNet-val. [12]; see also Tab. 8.

1. Introduction

Image synthesis is one of the computer vision fields with the most spectacular recent development, but also among those with the greatest computational demands. Especially high-resolution synthesis of complex, natural scenes is presently dominated by scaling up likelihood-based models, potentially containing billions of parameters in autoregressive (AR) transformers [66,67]. In contrast, the promising results of GANs [3, 27, 40] have been revealed to be mostly confined to data with comparably limited variability as their adversarial learning procedure does not easily scale to modeling complex, multi-modal distributions. Recently, diffusion models [82], which are built from a hierarchy of denoising autoencoders, have shown to achieve impressive results in image synthesis [30,85] and beyond [7,45,48,57], and define the state-of-the-art in class-conditional image synthesis [15,31] and super-resolution [72]. Moreover, even unconditional DMs can readily be applied to tasks such as inpainting and colorization [85] or stroke-based synthesis [53], in contrast to other types of generative models [19,46,69]. Being likelihood-based models, they do not exhibit mode-collapse and training instabilities as GANs and, by heavily exploiting parameter sharing, they can model highly complex distributions of natural images without involving billions of parameters as in AR models [67].

Democratizing High-Resolution Image Synthesis DMs belong to the class of likelihood-based models, whose mode-covering behavior makes them prone to spend excessive amounts of capacity (and thus compute resources) on modeling imperceptible details of the data [16, 73]. Although the reweighted variational objective [30] aims to address this by undersampling the initial denoising steps, DMs are still computationally demanding, since training and evaluating such a model requires repeated function evaluations (and gradient computations) in the high-dimensional space of RGB images. As an example, training the most powerful DMs often takes hundreds of GPU days (*e.g.* 150 - 1000 V100 days in [15]) and repeated evaluations on a noisy version of the input space render also inference expensive,

*The first two authors contributed equally to this work.


Latent Space
Diffusion Process
Denoising U-Net ϵ_θ
Conditioning
Semantic Map
Text
Repres entations
Images
x
$\mathcal{E}$
z
z_T
$\times(T-1)$
$\tilde{x}$
$\mathcal{D}$
z
z_{T-1}
z_T
Q
K V
Pixel Space
τ_θ
denoising step
crossattention
switch
skip connection
concat

2023

High-resolution update of Stable Diffusion

SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis

Dustin Podell **Zion English** **Kyle Lacey** **Andreas Blattmann** **Tim Dockhorn**

Jonas Müller **Joe Penna** **Robin Rombach**

Stability AI, Applied Research

Code: https://github.com/Stability-AI/generative-models *Model weights*: https://huggingface.co/stabilitya

ControlNet: versatile image-based conditioning

An example of the innovations and rapid developments empowered by Stable Diffusion's open source release

Adding Conditional Control to Text-to-Image Diffusion Models

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala
Stanford University
{lvmin, anyirao, maneesh}@cs.stanford.edu

Figure 1: Controlling Stable Diffusion with learned conditions. ControlNet allows users to add conditions like Canny edges (top), human pose (bottom), *etc.*, to control the image generation of large pretrained diffusion models. The default results use the prompt "a high-quality, detailed, and professional image". Users can optionally give prompts like the "chef in kitchen".

Abstract

We present ControlNet, a neural network architecture to add spatial conditioning controls to large, pretrained text-to-image diffusion models. ControlNet locks the production-ready large diffusion models, and reuses their deep and robust encoding layers pretrained with billions of images as a strong backbone to learn a diverse set of conditional controls. The neural architecture is connected with "zero convolutions" (zero-initialized convolution layers) that progressively grow the parameters from zero and ensure that no harmful noise could affect the finetuning. We test various conditioning controls, e.g., edges, depth, segmentation, human pose, etc., with Stable Diffusion, using single or multiple conditions, with or without prompts. We show that the training of ControlNets is robust with small (<50k) and large (>1m) datasets. Extensive results show that ControlNet may facilitate wider applications to control image diffusion models.

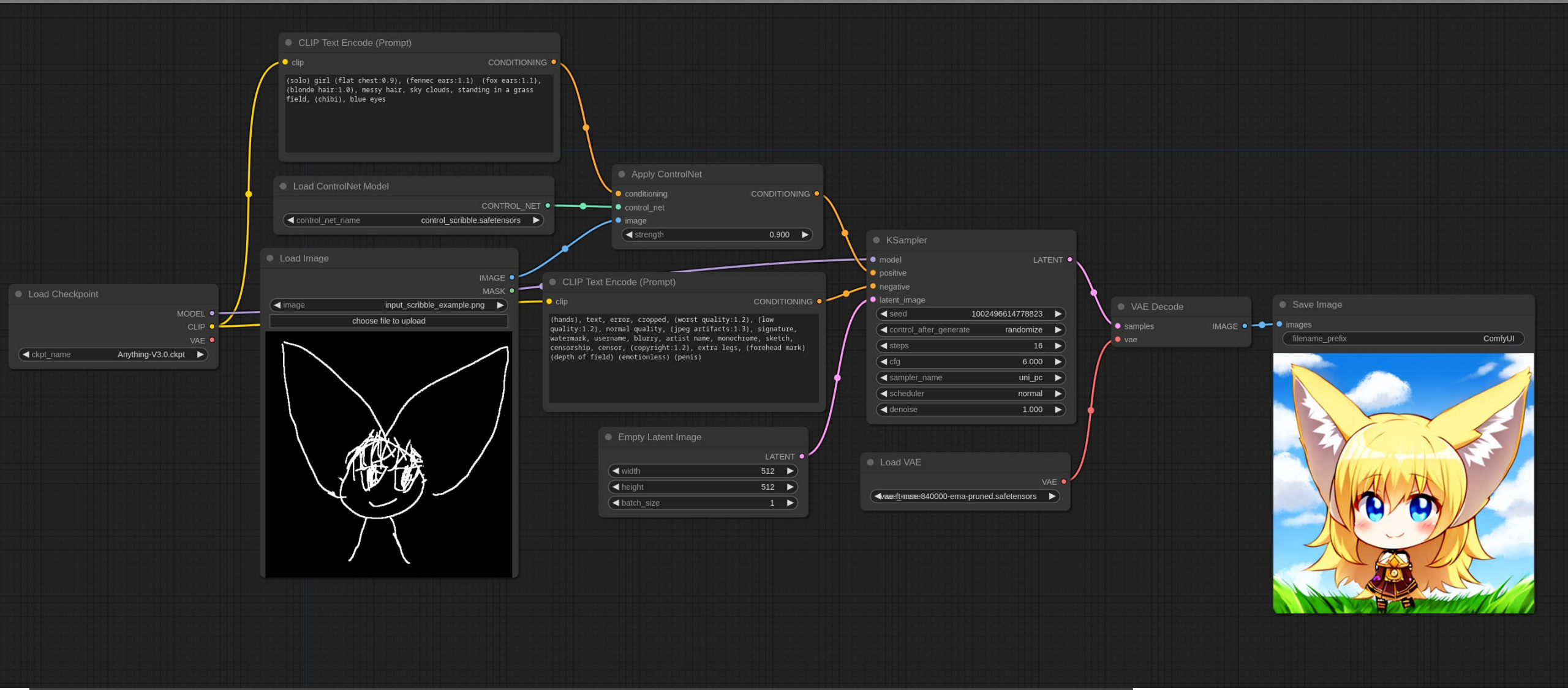
1. Introduction

Many of us have experienced flashes of visual inspiration that we wish to capture in a unique image. With the advent of text-to-image diffusion models [54, 62, 72], we can now create visually stunning images by typing in a text prompt. Yet, text-to-image models are limited in the control they provide over the spatial composition of the image; precisely expressing complex layouts, poses, shapes and forms can be difficult via text prompts alone. Generating an image that accurately matches our mental imagery often requires numerous trial-and-error cycles of editing a prompt, inspecting the resulting images and then re-editing the prompt.

Can we enable finer grained spatial control by letting users provide additional images that directly specify their desired image composition? In computer vision and machine learning, these additional images (*e.g.*, edge maps, human pose skeletons, segmentation maps, depth, normals, *etc.*) are often treated as conditioning on the image generation process. Image-to-image translation models [34, 98] learn

https://openaccess.thecvf.com/content/ICCV2023/html/Zhang_Adding_Conditional_Control_to_Text-to-Image_Diffusion_Models_ICCV_2023_paper.html

ControlNet in ComfyUI



ComfyUI: A popular interface for Stable diffusion

Finetuning StableDiffusion

- You can finetune StableDiffusion with your own images
- E.g., https://huggingface.co/hakurei/waifu-diffusion
- Requires a lot of memory, e.g., 40GB A100 GPU
- Free Colab GPU (16GB) works with a quantized model and LoRA (Low-Rank Adaptation):

LoRA

- Allows finetuning LLMs and StableDiffusion with limited memory
- Additive low-rank linear transforms (“adapters”) modify the outputs of various network blocks
- The original network is kept fixed and gradients only need to be computed and stored for the adapters
- QLoRA: The original network is quantized to further save memory

LoRA: Low-Rank Adaptation of Large Language Models

Edward Hu* **Yelong Shen*** **Phillip Wallis** **Zeyuan Allen-Zhu**
Yuanzhi Li **Shean Wang** **Lu Wang** **Weizhu Chen**
Microsoft Corporation
{edwardhu, yeshe, phwallis, zeyuana,
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu
(Version 2)

Abstract

An important paradigm of natural language processing consists of large-scale pre-training on general domain data and adaptation to particular tasks or domains. As we pre-train larger models, full fine-tuning, which retrains all model parameters, becomes less feasible. Using GPT-3 175B as an example – deploying independent instances of fine-tuned models, each with 175B parameters, is prohibitively expensive. We propose **Lo**w-**R**ank **A**daptation, or LoRA, which freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture, greatly reducing the number of trainable parameters for downstream tasks. Compared to GPT-3 175B fine-tuned with Adam, LoRA can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times. LoRA performs on-par or better than fine-tuning in model quality on RoBERTa, DeBERTa, GPT-2, and GPT-3, despite having fewer trainable parameters, a higher training throughput, and, unlike adapters, *no additional inference latency*. We also provide an empirical investigation into rank-deficiency in language model adaptation, which sheds light on the efficacy of LoRA. We release a package that facilitates the integration of LoRA with PyTorch models and provide our implementations and model checkpoints for RoBERTa, DeBERTa, and GPT-2 at https://github.com/microsoft/LoRA.

1 Introduction

Many applications in natural language processing rely on adapting *one* large-scale, pre-trained language model to *multiple* downstream applications. Such adaptation is usually done via *fine-tuning*, which updates all the parameters of the pre-trained model. The major downside of fine-tuning is that the new model contains as many parameters as in the original model. As larger models are trained every few months, this changes from a mere “inconvenience” for GPT-2 (Radford et al., b) or RoBERTa large (Liu et al., 2019) to a critical deployment challenge for GPT-3 (Brown et al., 2020) with 175 billion trainable parameters.[1]

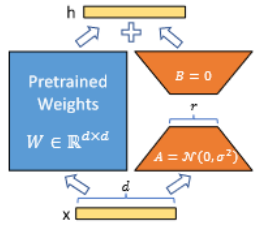


Figure 1: Our reparametrization. We only train A and B.

Many sought to mitigate this by adapting only some parameters or learning external modules for new tasks. This way, we only need to store and load a small number of task-specific parameters in addition to the pre-trained model for each task, greatly boosting the operational efficiency when deployed. However, existing techniques

*Equal contribution.
[0]Compared to V1, this draft includes better baselines, experiments on GLUE, and more on adapter latency.
[1]While GPT-3 175B achieves non-trivial performance with few-shot learning, fine-tuning boosts its performance significantly as shown in Appendix A.

DreamBooth

- Allows finetuning so that it only affects the look of a particular character or object
- E.g., teach StableDiffusion to generate images of your own pet, based on a handful of images

DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation

Nataniel Ruiz*,[1,2] Yuanzhen Li[1] Varun Jampani[1]
Yael Pritch[1] Michael Rubinstein[1] Kfir Aberman[1]
[1] Google Research [2] Boston University

Figure 1. With just a few images (typically 3-5) of a subject (left), *DreamBooth*—our AI-powered photo booth—can generate a myriad of images of the subject in different contexts (right), using the guidance of a text prompt. The results exhibit natural interactions with the environment, as well as novel articulations and variation in lighting conditions, all while maintaining high fidelity to the key visual features of the subject.

Abstract

Large text-to-image models achieved a remarkable leap in the evolution of AI, enabling high-quality and diverse synthesis of images from a given text prompt. However, these models lack the ability to mimic the appearance of subjects in a given reference set and synthesize novel renditions of them in different contexts. In this work, we present a new approach for "personalization" of text-to-image diffusion models. Given as input just a few images of a subject, we fine-tune a pretrained text-to-image model such that it learns to bind a unique identifier with that specific subject. Once the subject is embedded in the output domain of the model, the unique identifier can be used to synthesize novel photorealistic images of the subject contextualized in different scenes. By leveraging the semantic prior embedded in the model with a new autogenous class-specific prior preservation loss, our technique enables synthesizing the subject in diverse scenes, poses, views and lighting conditions that do not appear in the reference images. We apply our technique to several previously-unassailable tasks, including subject recontextualization, text-guided view synthesis, and artistic rendering, all while preserving the subject's key features. We also provide a new dataset and evaluation protocol for this new task of subject-driven generation. Project page: https://dreambooth.github.io/

*This research was performed while Nataniel Ruiz was at Google.

1. Introduction

Can you imagine your own dog traveling around the world, or your favorite bag displayed in the most exclusive showroom in Paris? What about your parrot being the main character of an illustrated storybook? Rendering such imaginary scenes is a challenging task that requires synthesizing instances of specific subjects (e.g., objects, animals) in new contexts such that they naturally and seamlessly blend into the scene.

Recently developed large text-to-image models have shown unprecedented capabilities, by enabling high-quality and diverse synthesis of images based on a text prompt written in natural language [54, 61]. One of the main advantages of such models is the strong semantic prior learned from a large collection of image-caption pairs. Such a prior learns, for instance, to bind the word "dog" with various instances of dogs that can appear in different poses and contexts in an image. While the synthesis capabilities of these models are unprecedented, they lack the ability to mimic the appearance of subjects in a given reference set, and synthesize novel renditions of the *same subjects* in different contexts. The main reason is that the expressiveness of their output domain is limited; even the most detailed textual description of an object may yield instances with different appearances.

Colab notebooks (try at your own risk)

- Colab runs on Google's Linux virtual machines => can't mess up your own computer
- However, allowing a Colab notebook connect to your Google drive can pose some risk. If a notebook asks for that, Google if others have used the notebook with good results. Also, do not input any passwords to a notebook you don't trust.
- LoRA finetuning of Stable Diffusion: https://colab.research.google.com/github/Linaqruf/kohya-trainer/blob/main/kohya-LoRA-finetuner.ipynb
- LoRA Dreambooth for Stable Diffusion: https://colab.research.google.com/github/Linaqruf/kohya-trainer/blob/main/kohya-LoRA-dreambooth.ipynb

Recent progress

Understanding of diffusion model best practices is gradually maturing.

Analyzing and Improving the Training Dynamics of Diffusion Models

Tero Karras, NVIDIA
Miika Aittala, NVIDIA
Jaakko Lehtinen, NVIDIA, Aalto University
Janne Hellsten, NVIDIA
Timo Aila, NVIDIA
Samuli Laine, NVIDIA

Abstract

Diffusion models currently dominate the field of data-driven image synthesis with their unparalleled scaling to large datasets. In this paper, we identify and rectify several causes for uneven and ineffective training in the popular ADM diffusion model architecture, without altering its high-level structure. Observing uncontrolled magnitude changes and imbalances in both the network activations and weights over the course of training, we redesign the network layers to preserve activation, weight, and update magnitudes on expectation. We find that systematic application of this philosophy eliminates the observed drifts and imbalances, resulting in considerably better networks at equal computational complexity. Our modifications improve the previous record FID of 2.41 in ImageNet-512 synthesis to 1.81, achieved using fast deterministic sampling.

As an independent contribution, we present a method for setting the exponential moving average (EMA) parameters post-hoc, i.e., after completing the training run. This allows precise tuning of EMA length without the cost of performing several training runs, and reveals its surprising interactions with network architecture, training time, and guidance.

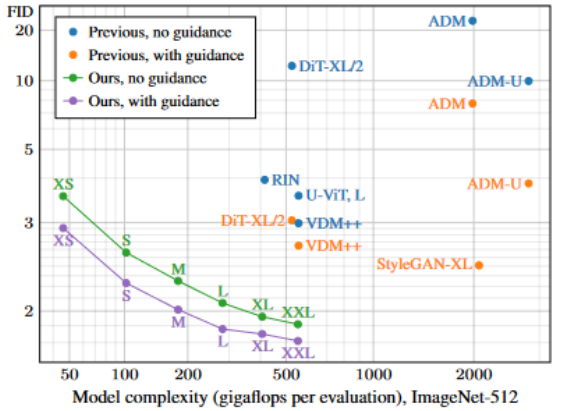


Figure 1. Our contributions significantly improve the quality of results w.r.t. model complexity, surpassing the previous state-of-the-art with a 5× smaller model. In this plot, we use gigaflops per single model evaluation as a measure of a model's intrinsic computational complexity; a similar advantage holds in terms of parameter count, as well as training and sampling cost (see Appendix A).

1. Introduction

High-quality image synthesis based on text prompts, example images, or other forms of input has become widely popular thanks to advances in denoising diffusion models [22, 52, 71–74, 81]. Diffusion-based approaches produce high-quality images while offering versatile controls [9, 18, 21, 50, 88] and convenient ways to introduce novel subjects [13, 65], and they also extend to other modalities such as audio [41, 58], video [6, 23, 25], and 3D shapes [46, 57, 60, 70]. A recent survey of methods and applications is given by Yang et al. [83].

On a high level, diffusion models convert an image of pure noise to a novel generated image through repeated application of image denoising. Mathematically, each denoising step can be understood through the lens of score matching [28], and it is typically implemented using a U-Net [22, 64] equipped with self-attention [80] layers. Since we do not contribute to the theory behind diffusion models, we refer the interested reader to the seminal works of Sohl-Dickstein et al. [71], Song and Ermon [73], and Ho et al. [22], as well as to Karras et al. [36], who frame various mathematical frameworks in a common context.

Despite the seemingly frictionless scaling to very large datasets and models, the training dynamics of diffusion models remain challenging due to the highly stochastic loss function. The final image quality is dictated by faint image details predicted throughout the sampling chain, and small mistakes at intermediate steps can have snowball effects in subsequent iterations. The network must accurately estimate the average clean image across a vast range of noise levels, Gaussian noise realizations, and conditioning inputs. Learn-

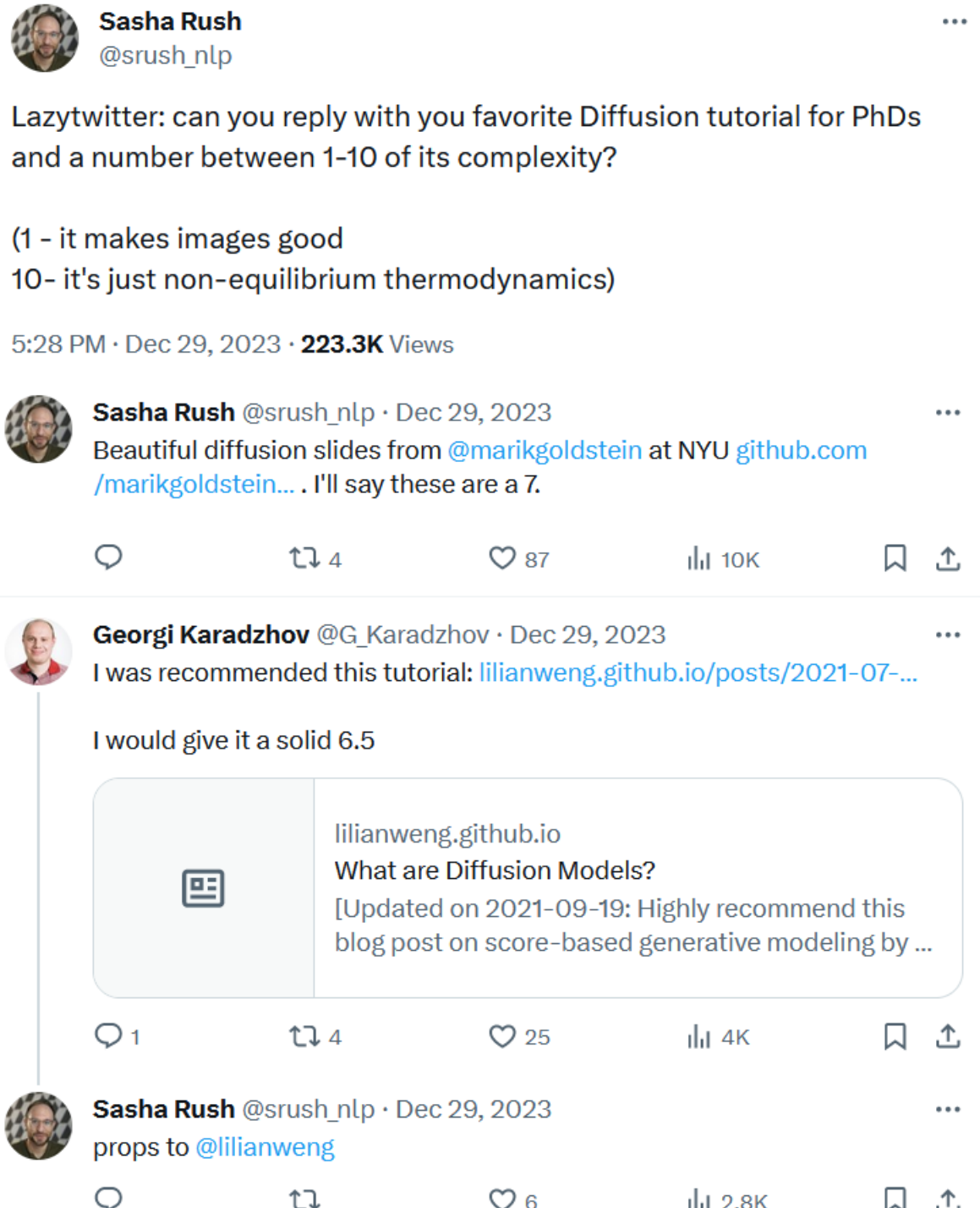
Want to learn more?

https://developer.nvidia.com/blog/generative-ai-research-spotlight-demystifying-diffusion-based-models/

http://yang-song.net/blog/2021/score/

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

https://twitter.com/srush_nlp/status/1740756730307629530



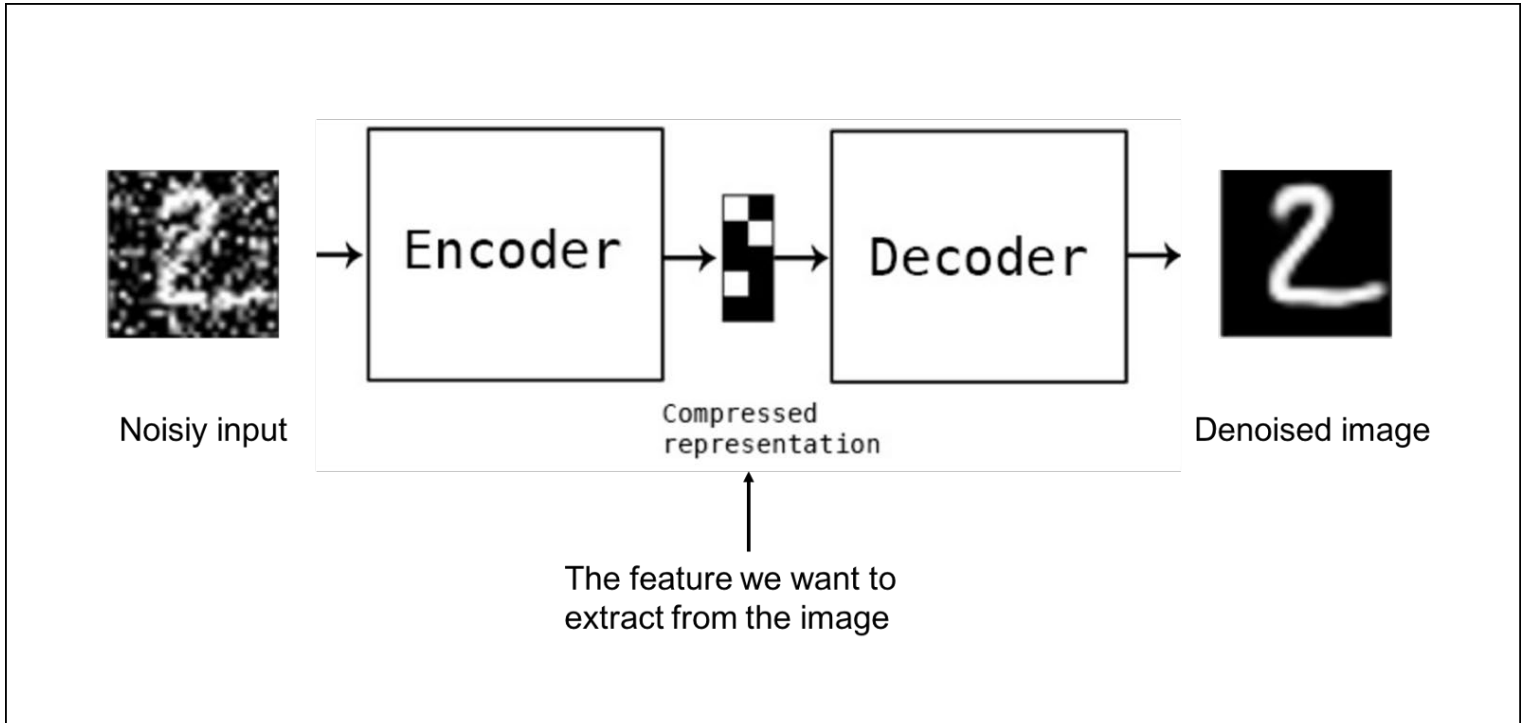
Midjourney

- Commercial diffusion-based image generator
- Interface: Discord chat
- Internals not published
- Rapid evolution of model versions—if v3 or v4 was not good for you, check the latest v5 and v6 models

Other image generators

Variational autoencoder (VAE, 2014)

- Random samples in the latent space (the encoder outputs) of an autoencoder sometimes generate valid decoded output, sometimes not
- A basic autoencoder does not guarantee:
  - An easy-to-sample latent distribution
  - Interpolation: What happens when a randomly sampled latent vector lies between training examples?
- VAE is an autoencoder where the compressed representation is forced to be normally distributed => easy to sample



Wasserstein Autoencoder (WAE, ICLR 2018)

- VAE is old and usually doesn't give great results (details beyond the scope of this lecture)
- WAE is the modern version, but a bit more costly to train

Wasserstein Auto-Encoders

Ilya Tolstikhin[1], Olivier Bousquet[2], Sylvain Gelly[2], and Bernhard Schölkopf[1]

[1]Max Planck Institute for Intelligent Systems
[2]Google Brain

Abstract

We propose the Wasserstein Auto-Encoder (WAE)—a new algorithm for building a generative model of the data distribution. WAE minimizes a penalized form of the Wasserstein distance between the model distribution and the target distribution, which leads to a different regularizer than the one used by the Variational Auto-Encoder (VAE) [1]. This regularizer encourages the encoded training distribution to match the prior. We compare our algorithm with several other techniques and show that it is a generalization of adversarial auto-encoders (AAE) [2]. Our experiments show that WAE shares many of the properties of VAEs (stable training, encoder-decoder architecture, nice latent manifold structure) while generating samples of better quality, as measured by the FID score.

Glow: Generative Flow with Invertible 1×1 Convolutions

Diederik P. Kingma[*†], **Prafulla Dhariwal**[*]
[*]OpenAI
[†]Google AI

Abstract

Flow-based generative models (Dinh et al., 2014) are conceptually attractive due to tractability of the exact log-likelihood, tractability of exact latent-variable inference, and parallelizability of both training and synthesis. In this paper we propose *Glow*, a simple type of generative flow using an invertible 1×1 convolution. Using our method we demonstrate a significant improvement in log-likelihood on standard benchmarks. Perhaps most strikingly, we demonstrate that a flow-based generative model optimized towards the plain log-likelihood objective is capable of efficient realistic-looking synthesis and manipulation of large images. The code for our model is available at `https://github.com/openai/glow`.

1 Introduction

Two major unsolved problems in the field of machine learning are (1) data-efficiency: the ability to learn from few datapoints, like humans; and (2) generalization: robustness to changes of the task or its context. AI systems, for example, often do not work at all when given inputs that are different

[*]Equal contribution.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

Figure 1: Synthetic celebrities sampled from our model; see Section 3 for architecture and method, and Section 5 for more results.

DALL-E v1

- Generating images patch-by-patch like text, using a GPT-like transformer
- The training sequences start with text tokens (the prompt) and continue with image patch tokens

Zero-Shot Text-to-Image Generation

Aditya Ramesh [1] Mikhail Pavlov [1] Gabriel Goh [1] Scott Gray [1]
Chelsea Voss [1] Alec Radford [1] Mark Chen [1] Ilya Sutskever [1]

Abstract

Text-to-image generation has traditionally focused on finding better modeling assumptions for training on a fixed dataset. These assumptions might involve complex architectures, auxiliary losses, or side information such as object part labels or segmentation masks supplied during training. We describe a simple approach for this task based on a transformer that autoregressively models the text and image tokens as a single stream of data. With sufficient data and scale, our approach is competitive with previous domain-specific models when evaluated in a zero-shot fashion.

Figure 1. Comparison of original images (top) and reconstructions from the discrete VAE (bottom). The encoder downsamples the spatial resolution by a factor of 8. While details (e.g., the texture of the cat's fur, the writing on the storefront, and the thin lines in the illustration) are sometimes lost or distorted, the main features of the image are still typically recognizable. We use a large vocabulary size of 8192 to mitigate the loss of information.

1. Introduction

Modern machine learning approaches to text to image synthesis started with the work of Mansimov et al. (2015), who showed that the DRAW Gregor et al. (2015) generative model, when extended to condition on image captions, could also generate novel visual scenes. Reed et al. (2016b) later demonstrated that using a generative adversarial network (Goodfellow et al., 2014), rather than a recurrent variational auto-encoder, improved image fidelity. Reed et al. (2016b) showed that this system could not only generate objects with recognizable properties, but also could *zero-shot* generalize to held-out categories.

Over the next few years, progress continued using a combination of methods. These include improving the generative model architecture with modifications like multi-scale generators (Zhang et al., 2017; 2018), integrating attention and auxiliary losses (Xu et al., 2018), and leveraging additional sources of conditioning information beyond just text (Reed et al., 2016a; Li et al., 2019; Koh et al., 2021).

Separately, Nguyen et al. (2017) propose an energy-based framework for conditional image generation that obtained a large improvement in sample quality relative to contemporary methods. Their approach can incorporate pretrained discriminative models, and they show that it is capable of performing text-to-image generation when applied to a captioning model pretrained on MS-COCO. More recently, Cho et al. (2020) also propose a method that involves optimizing the input to a pretrained cross-modal masked language model. While significant increases in visual fidelity have occurred as a result of the work since Mansimov et al. (2015), samples can still suffer from severe artifacts such as object distortion, illogical object placement, or unnatural blending of foreground and background elements.

Recent advances fueled by large-scale generative models suggest a possible route for further improvements. Specifically, when compute, model size, and data are scaled carefully, autoregressive transformers (Vaswani et al., 2017) have achieved impressive results in several domains such as text (Radford et al., 2019), images (Chen et al., 2020), and audio (Dhariwal et al., 2020).

By comparison, text-to-image generation has typically been evaluated on relatively small datasets such as MS-COCO and CUB-200 (Welinder et al., 2010). Could dataset size and

[1]OpenAI, San Francisco, California, United States. Correspondence to: Aditya Ramesh <_@adityaramesh.com>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 139, 2020.

Parti

- Like DALL-E v1, but with a separate upsampler and encoder-decoder transformer

Scaling Autoregressive Models for Content-Rich Text-to-Image Generation

Jiahui Yu* Yuanzhong Xu† Jing Yu Koh† Thang Luong† Gunjan Baid†
Zirui Wang† Vijay Vasudevan† Alexander Ku†
Yinfei Yang Burcu Karagol Ayan Ben Hutchinson
Wei Han Zarana Parekh Xin Li Han Zhang
Jason Baldridge† Yonghui Wu*
{jiahuiyu, yuanzx, jykoh, thangluong, gunjanbaid, ziruiw, vrv, alexku,
jasonbaldridge, yonghui}@google.com§

* Equal contribution. † Core contribution.

Google Research

https://3dvar.com/Yu2022Scaling.pdf

Figure 1: Example images generated by Parti. **Top row**: "*Oil-on-canvas painting of a blue night sky with roiling energy. A fuzzy and bright yellow crescent moon shining at the top. Below the exploding yellow stars and radiating swirls of blue, a distant village sits quietly on the right. Connecting earth and sky is a flame-like cypress tree with curling and swaying branches on the left. A church spire rises as a beacon over rolling blue hills.*" (a 67-word description of the Starry Night by Vincent van Gogh). **Middle row**: "*A close-up high-contrast photo of Sydney Opera House sitting next to Eiffel tower, under a blue night sky of roiling energy, exploding yellow stars, and radiating swirls of blue*". **Last row**: Similar to the middle row, but with "*anime illustration*" and different landmarks (*the Great Pyramid and the Parthenon*).

§Correspondence to {jiahuiyu, jasonbaldridge, yonghui}@google.com.

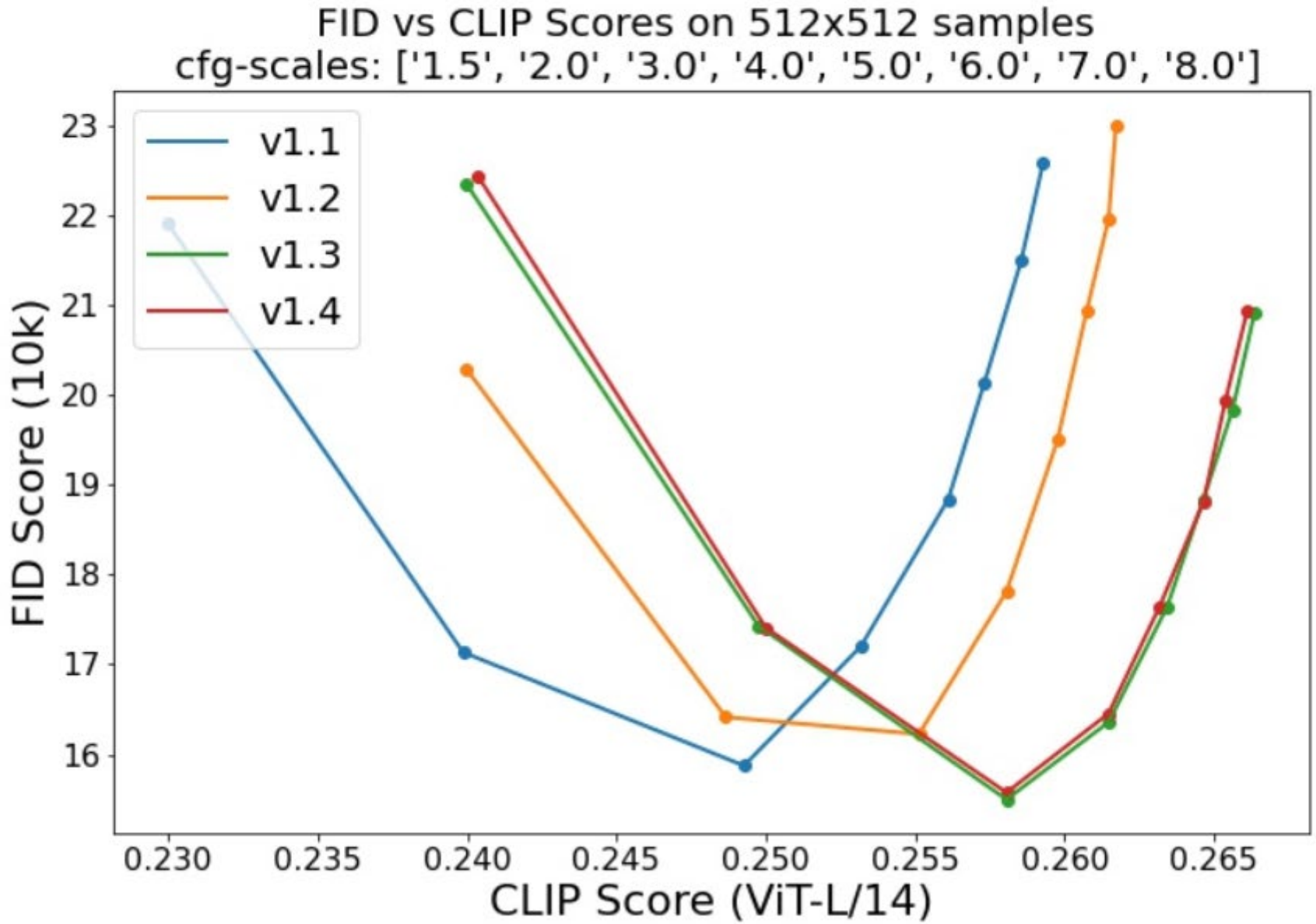
Contents

- Preliminaries: compute graphs, artificial neurons, activation functions, loss functions, latent spaces
- Understanding nonlinear activations
- Image generator architectures
- **Quality metrics: FID, Precision & Recall**

FID: A common measure of image generation quality

- Many valid outputs for the same prompt—can’t simply compare to a single reference image
- Subjective evaluation by humans is costly (many raters and images needed for reliable result)
- Frechet Inception Distance (FID): Frechet distance between Normal distribution approximations of generated and real images in the embedding space of the Inception image classifier network. Lower FID is better

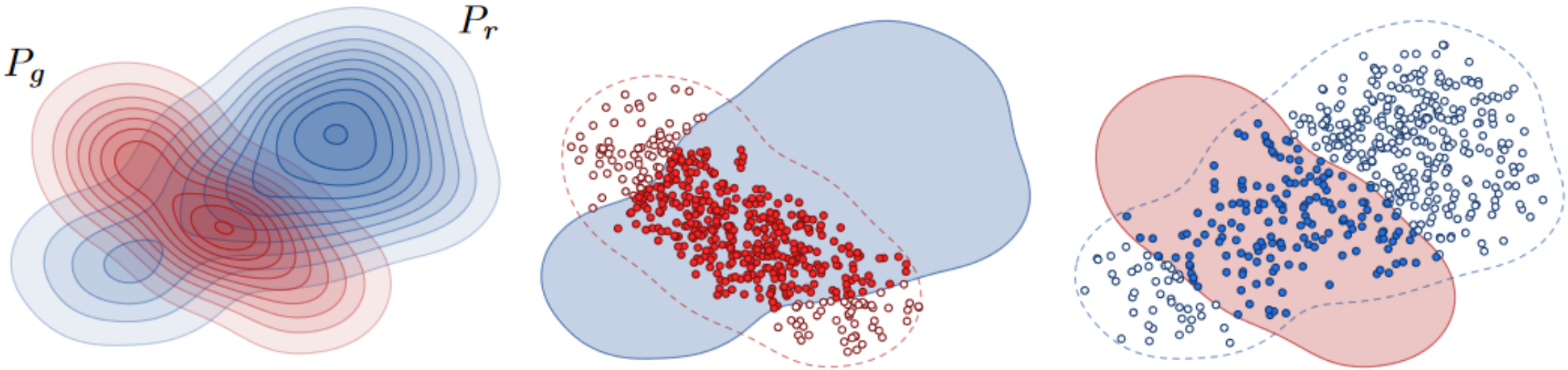
Evaluations with different classifier-free guidance scales (1.5, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0) and 50 PLMS sampling steps show the relative improvements of the checkpoints:



https://github.com/CompVis/stable-diffusion

Precision and Recall

- Precision: how well the samples stay within the training data
- Recall: how well the samples cover all training data
- Range 0…1, higher is better.

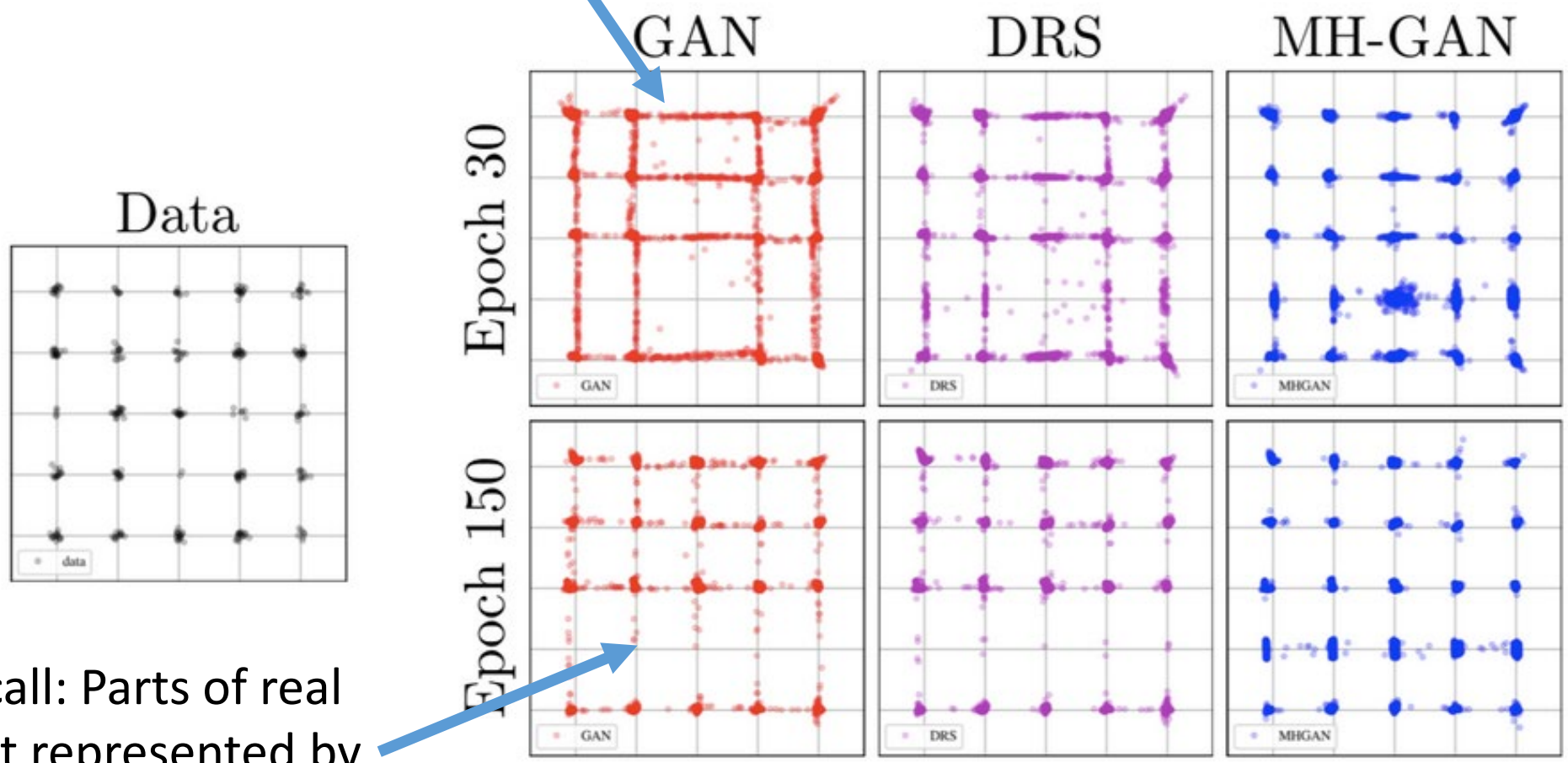


(a) Example distributions (b) Precision (c) Recall

Figure 1: Definition of precision and recall for distributions [24]. (a) Denote the distribution of real images with P_r (blue) and the distribution of generated images with P_g (red). (b) Precision is the probability that a random image from P_g falls within the support of P_r. (c) Recall is the probability that a random image from P_r falls within the support of P_g.

https://proceedings.neurips.cc/paper/2019/hash/0234c510bc6d908b28c70ff313743079-Abstract.html

Low precision: Generated samples outside real data



Low recall: Parts of real data not represented by samples

https://www.uber.com/en-FI/blog/mh-gan/

How to compute precision and recall for images?

- K-nearest neighbors –based approach
- Neighbors searched for in an embedding space where similarity of vectors corresponds to perceptual image similarity
- Recall: For each real image, are there similar generated images?
- Precision: For each generated image, are there similar real images?

https://proceedings.neurips.cc/paper/2019/hash/0234c510bc6d908b28c70ff313743079-Abstract.html

Improved Precision and Recall Metric for Assessing Generative Models

Tuomas Kynkäänniemi*
Aalto University
NVIDIA
tuomas.kynkaanniemi@aalto.fi

Tero Karras
NVIDIA
tkarras@nvidia.com

Samuli Laine
NVIDIA
slaine@nvidia.com

Jaakko Lehtinen
Aalto University
NVIDIA
jlehtinen@nvidia.com

Timo Aila
NVIDIA
taila@nvidia.com

Abstract

The ability to automatically estimate the quality and coverage of the samples produced by a generative model is a vital requirement for driving algorithm research. We present an evaluation metric that can separately and reliably measure both of these aspects in image generation tasks by forming explicit, non-parametric representations of the manifolds of real and generated data. We demonstrate the effectiveness of our metric in StyleGAN and BigGAN by providing several illustrative examples where existing metrics yield uninformative or contradictory results. Furthermore, we analyze multiple design variants of StyleGAN to better understand the relationships between the model architecture, training methods, and the properties of the resulting sample distribution. In the process, we identify new variants that improve the state-of-the-art. We also perform the first principled analysis of truncation methods and identify an improved method. Finally, we extend our metric to estimate the perceptual quality of individual samples, and use this to study latent space interpolations.

1 Introduction

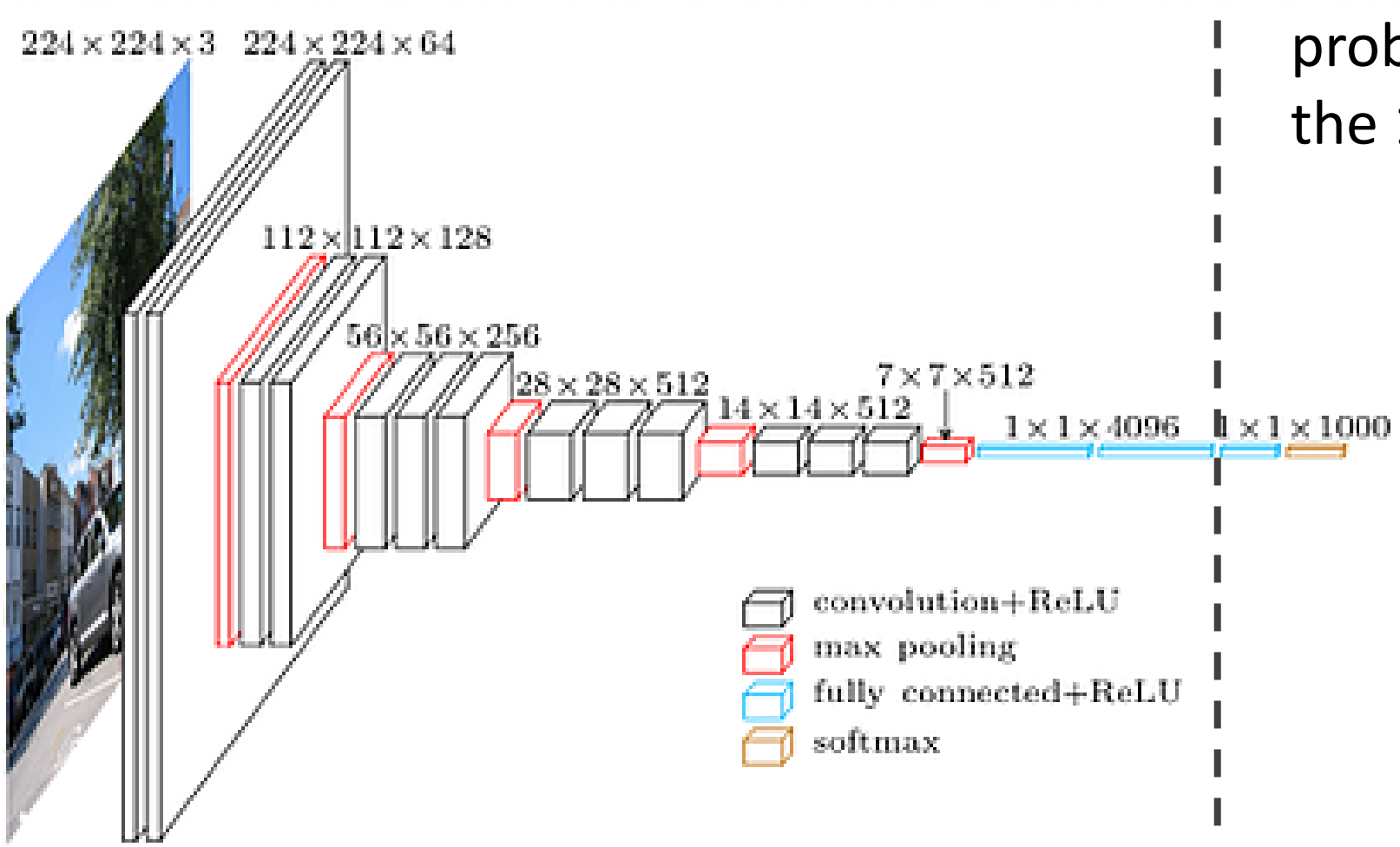
The goal of generative methods is to learn the manifold of the training data so that we can subsequently generate novel samples that are indistinguishable from the training set. While the quality of results from generative adversarial networks (GAN) [7], variational autoencoders (VAE) [14], autoregressive models [29, 30], and likelihood-based models [6, 13] have seen rapid improvement recently [11, 8, 28, 20, 4, 12], the automatic evaluation of these results continues to be challenging.

When modeling a complex manifold for sampling purposes, two separate goals emerge: individual samples drawn from the model should be faithful to the examples (they should be of "high quality"), and their variation should match that observed in the training set. The most widely used metrics, such as Fréchet Inception Distance (FID) [9], Inception Score (IS) [25], and Kernel Inception Distance (KID) [2], group these two aspects to a single value without a clear tradeoff. We illustrate by examples that this makes diagnosis of model performance difficult. For instance, it is interesting that while recent state-of-the-art generative methods [4, 13, 12] claim to optimize FID, in the end the (uncurated) results are almost always produced using another model that explicitly sacrifices variation, and often FID, in favor of higher quality samples from a truncated subset of the domain [17].

*This work was done during an internship at NVIDIA.

VGG-16 onvolutional encoder, produces a 4096-dimensional embedding vector

Fully connected layer and softmax:
Maps embedding to probabilities of each of the 1000 image classes



Summary

- A *generative model* is needed when there are multiple possible outputs for a single input, e.g., can’t simply map a text prompt to a single correct output image.
- Precision: how well the samples stay within the training data
- Recall: how well the samples cover all training data
- GANs used to be the dominant image generators, now superseded by diffusion models
- GANs have high precision but no guarantees of high recall.
- Diffusion models have both high precision and recall, but require multiple denoising passes through the network => typically slower

3 main types of generation

- AE, VAE, WAE, GAN, Flow-based: Sample a latent vector (e.g., multivariate Gaussian) and feed it through a network that converts them into images, sound, text...
- Parti, DALL-E v1: Sample an image autoregressively patch-by-patch, just like text generators sample text tokens. A patch = an image word.
- Diffusion (DALL-E v2 & v3, Midjourney, StableDiffusion): Sample a noise image, then gradually denoise it guided by a reference text and/or an image.

Training data

Artist finds private medical record photos in popular AI training data set

LAION scraped medical photos for AI research use. Who's responsible for taking them down?

BENJ EDWARDS - 9/21/2022, 6:43 PM

Enlarge / Censored medical images found in the LAION-5B data set used to train AI. The black bars and distortion have been added.

https://arstechnica.com/information-technology/2022/09/artist-finds-private-medical-record-photos-in-popular-ai-training-data-set/

LAION-5B: An open large-scale dataset for training next generation image-text models

Christoph Schuhmann[1] §§°° **Romain Beaumont**[1] §§°° **Richard Vencu**[1,3,8] §§°°
Cade Gordon[2] §§°° **Ross Wightman**[1] §§ **Mehdi Cherti** [1,10] §§
Theo Coombes[1] **Aarush Katta**[1] **Clayton Mullis**[1] **Mitchell Wortsman**[6]
Patrick Schramowski[1,4,5] **Srivatsa Kundurthy**[1] **Katherine Crowson**[1,8,9]
Ludwig Schmidt[6] °° **Robert Kaczmarczyk**[1,7] °° **Jenia Jitsev**[1,10] °°
LAION[1] UC Berkeley[2] Gentec Data[3] TU Darmstadt[4] Hessian.AI[5]
University of Washington, Seattle[6] Technical University of Munich[7] Stability AI[8]
EleutherAI[9] Juelich Supercomputing Center (JSC), Research Center Juelich (FZJ)[10]
contact@laion.ai
§§ Equal first contributions, °° Equal senior contributions

Abstract

Groundbreaking language-vision architectures like CLIP and DALL-E proved the utility of training on large amounts of noisy image-text data, without relying on expensive accurate labels used in standard vision unimodal supervised learning. The resulting models showed capabilities of strong text-guided image generation and transfer to downstream tasks, while performing remarkably at zero-shot classification with noteworthy out-of-distribution robustness. Since then, large-scale language-vision models like ALIGN, BASIC, GLIDE, Flamingo and Imagen made further improvements. Studying the training and capabilities of such models requires datasets containing billions of image-text pairs. Until now, no datasets of this size have been made openly available for the broader research community. To address this problem and democratize research on large-scale multi-modal models, we present LAION-5B - a dataset consisting of 5.85 billion CLIP-filtered image-text pairs, of which 2.32B contain English language. We show successful replication and fine-tuning of foundational models like CLIP, GLIDE and Stable Diffusion using the dataset, and discuss further experiments enabled with an openly available dataset of this scale. Additionally we provide several nearest neighbor indices, an improved web-interface for dataset exploration and subset generation, and detection scores for watermark, NSFW, and toxic content detection. [1]

1 Introduction

Learning from multimodal data such as text, images, and audio is a longstanding research challenge in machine learning [31, 51, 56, 83, 86]. Recently, contrastive loss functions combined with large neural networks have led to breakthroughs in the generalization capabilities of vision and language models [58, 59, 66]. For instance, OpenAI's CLIP models [58] achieved large gains in zero-shot classification on ImageNet [65], improving from the prior top-1 accuracy of 11.5% [41] to 76.2%. In addition, CLIP achieved unprecedented performance gains on multiple challenging distribution shifts [3, 23, 61, 70, 78, 82]. Inspired by CLIP's performance, numerous groups have further improved image-text models by increasing the amount of computation and the training set size [28, 54, 89, 94]. Another recent success of multimodal learning is in image generation, where DALL-E [59] and later

[1]Project page: https://laion.ai/laion-5b-a-new-era-of-open-large-scale-multi-modal-datasets/

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.

Types of image generators

Model type	Generation speed	Image quality	Image diversity	Interpolable latent space	Examples
VAE (Variational Autoencoder)	fast	Low	high	Yes	https://arxiv.org/abs/1312.6114
GAN (Generative Adversarial Networks)	fast	high	low	Yes	StyleGAN 1-3, BigGAN, https://github.com/NVlabs/stylegan3, https://www.tensorflow.org/hub/tutorials/biggan_generation_with_tf_hub
Flow-based	fast	medium	high	Yes	https://openai.com/research/glow
Transformer (autoregressive generation patch-by-patch)	slow	high	high	No	https://github.com/google-research/parti
Diffusion	slow	very high	high	Somewhat (non-smooth)	DALL-E, Midjourney, Stable Diffusion