Lecture 12
Generative Models: Autoencoders, GANs, Diffusion and Beyond (Part 1)

Olexandr Isayev

Department of Chemistry, CMU

olexandr@cmu.edu

- One-slide Class Project Update

- HW4

Homework #4: Graph based NNs

In this homework, you are required to implement a graph-based neural network machine learning model to predict the target property (Regression).

There are two datasets in the data folder: train.pt, test.pt. You will train a GNN on the training dataset, then predict on the test dataset.

This HW needs to be implemented with Pytorch Geometric (PyG): https://pytorch-geometric.readthedocs.io/en/latest/

Sign up link:
https://www.kaggle.com/t/8ad6bea774d5453fa4873d8a318124c5Links

Homework #4: Graph based NNs

You are allowed to build any type of CNN regression model using PyG (PyTorch Geometric)!

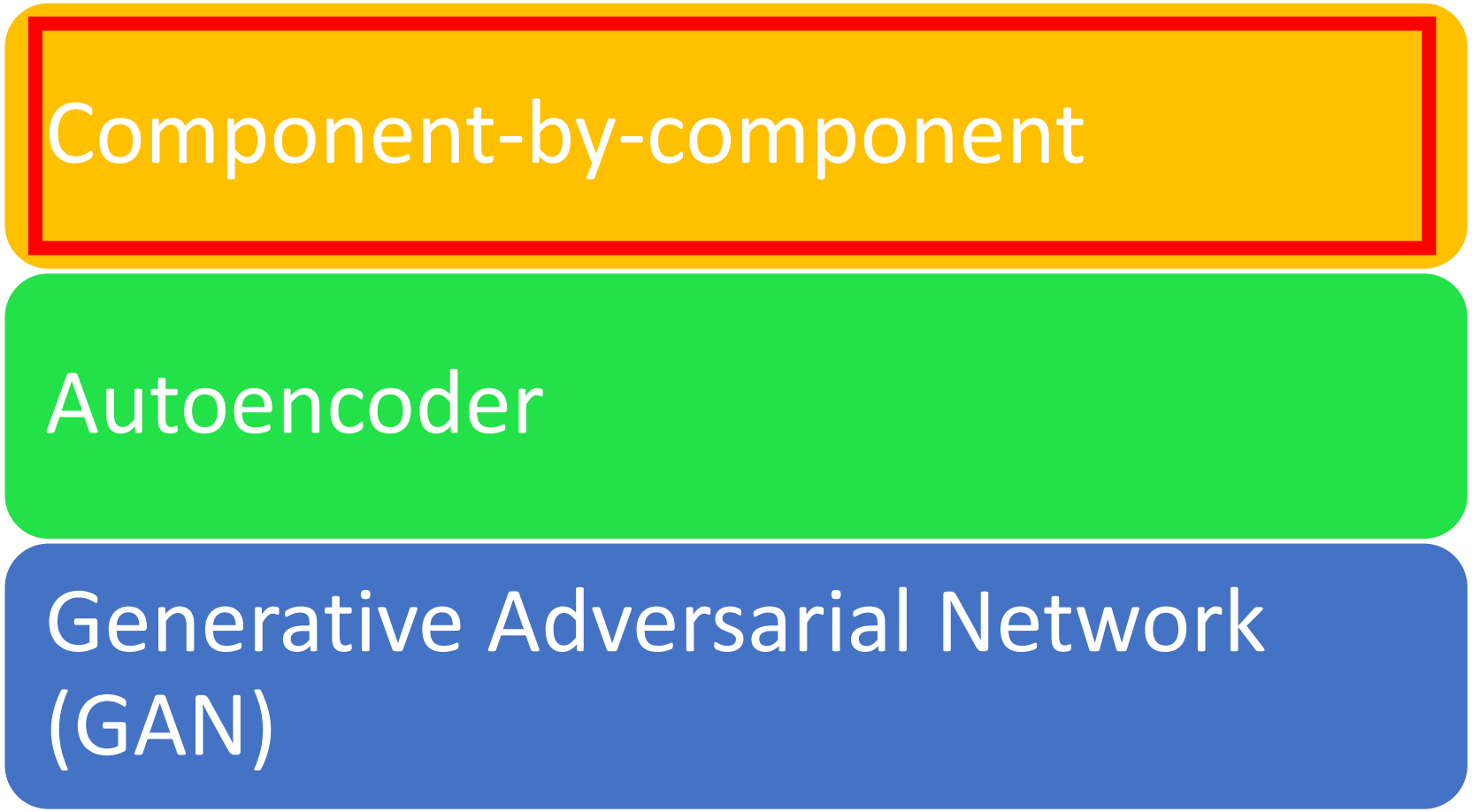
You are allowed to use any type of data processing (data augmentation).

Please explore training from scratch or finetuning/pertaining existing model.

Find the best performing model

Use your model to score test.pt

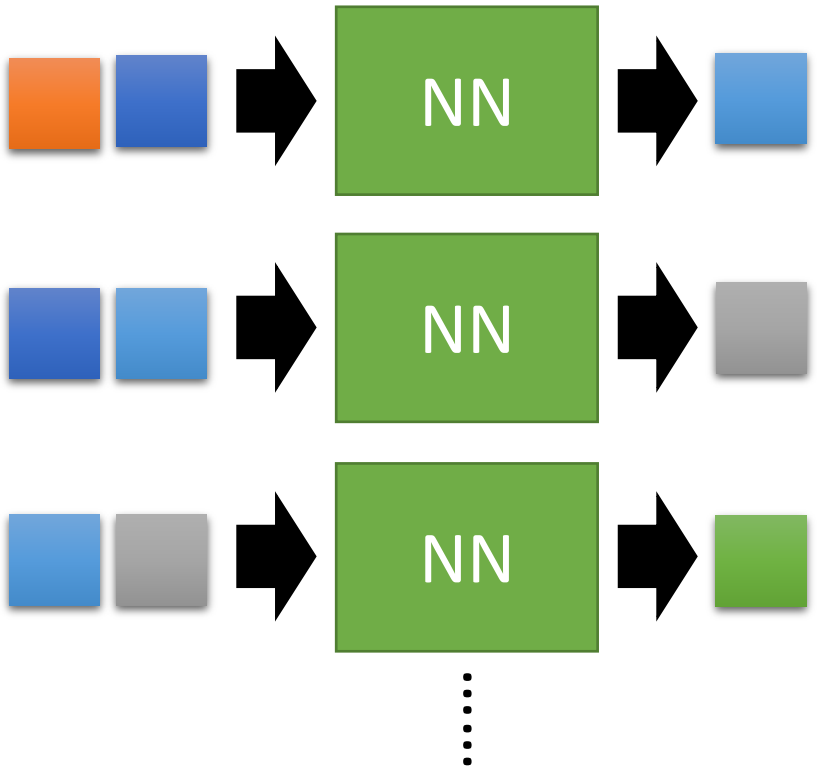
Generative Models



Component-by-component

- Image generation

E.g. 3 x 3 images

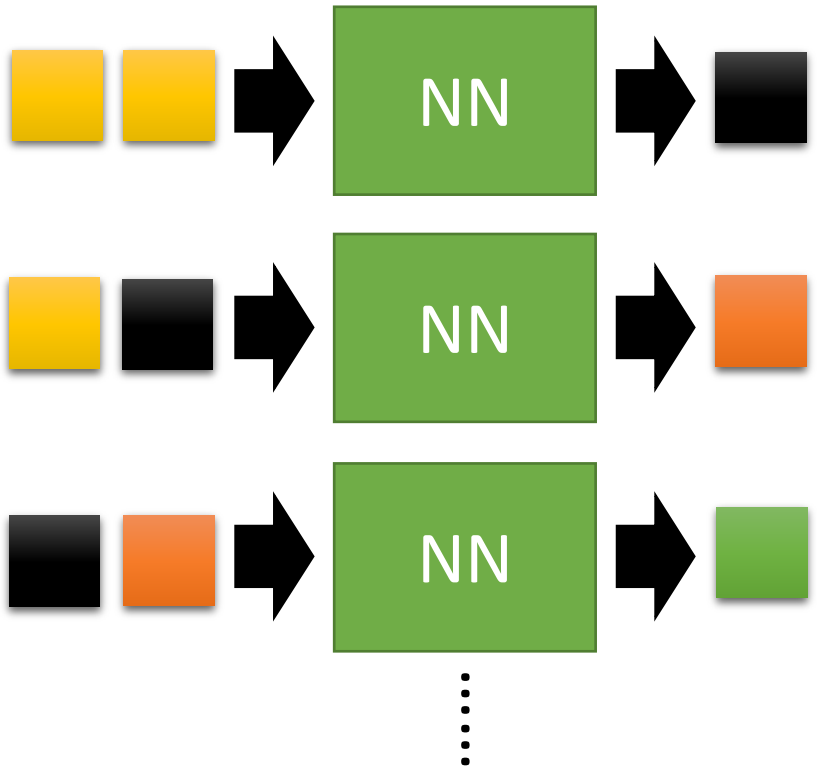


Can be trained just with a large collection of images without any annotation

Component-by-component

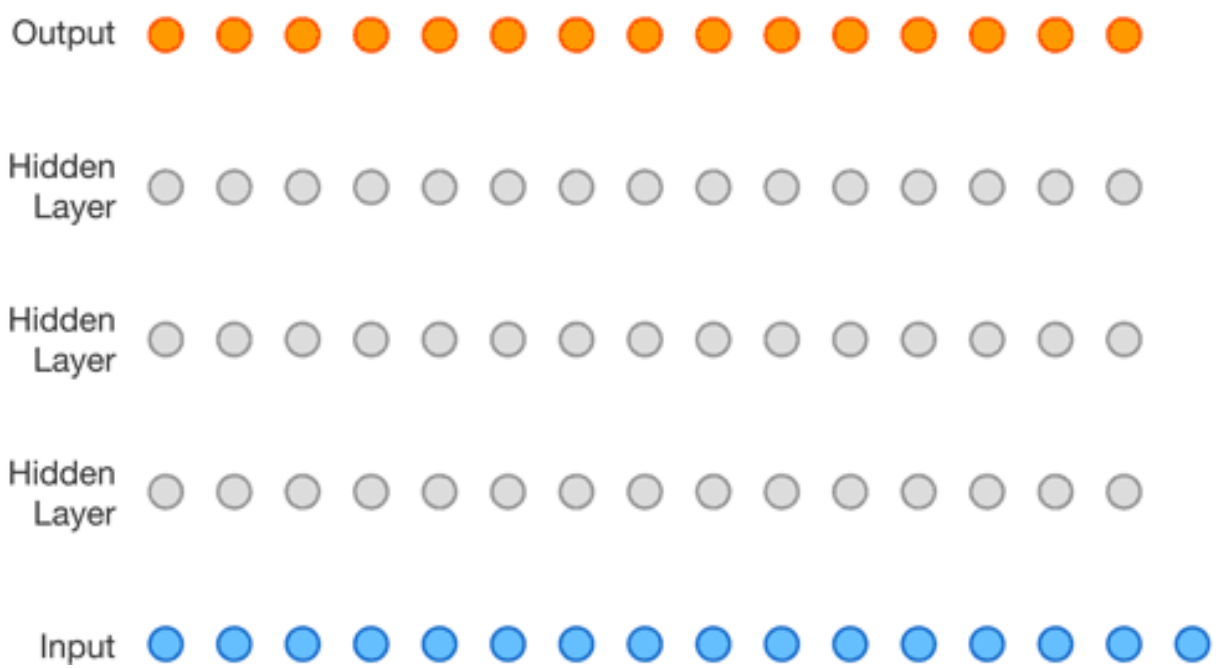
- Image generation

E.g. 3 x 3 images



Can be trained just with a large collection of images without any annotation

More than images ……



Audio: Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, WaveNet: A Generative Model for Raw Audio, arXiv preprint, 2016

Video: Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, Koray Kavukcuoglu, Video Pixel Networks , arXiv preprint, 2016

Autoregressive Models

An **autoregressive model** is when a value from a time series is regressed on previous values from that same time series. For example, y_t on y_{t-1}:

$$y_t=\beta_0+\beta_1 y_{t-1}+\epsilon_t.$$

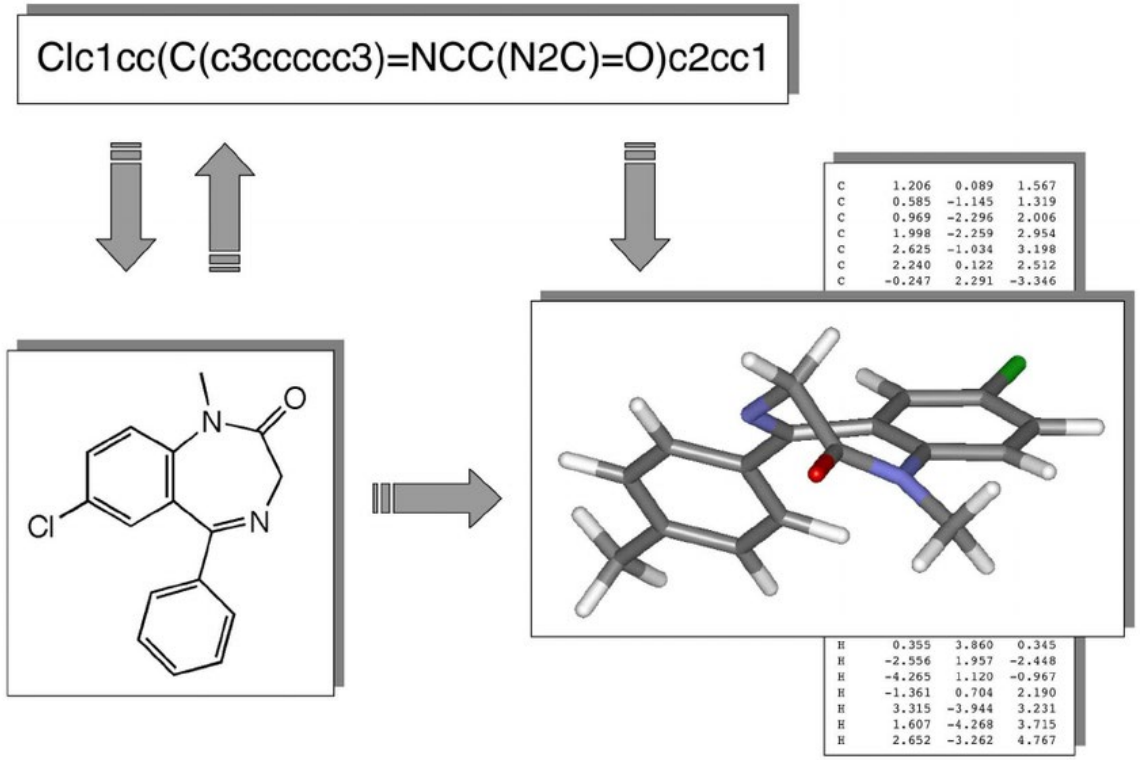
In this regression model, the response variable in the previous time period has become the predictor and the errors have our usual assumptions about errors in a simple linear regression model. The **order** of an autoregression is the number of immediately preceding values in the series that are used to predict the value at the present time. So, the preceding model is a first-order autoregression, written as AR(1).

If we want to predict y using measurements of the previous two observations (y_{t-1},y_{t-2}), then the autoregressive model for doing so would be:

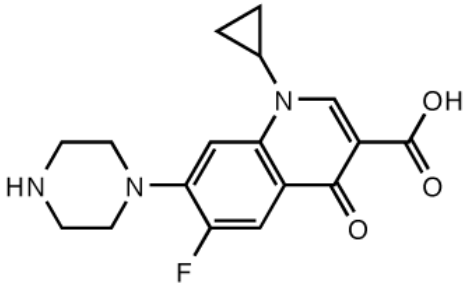
$$y_t=\beta_0+\beta_1 y_{t-1}+\beta_2 y_{t-2}+\epsilon_t.$$

This model is a second-order autoregression, written as AR(2), since the value at time t is predicted from the values at times t−1 and t−2.

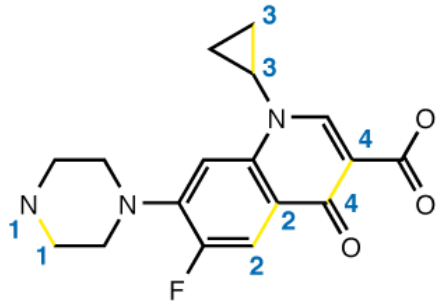
Language of SMILEs



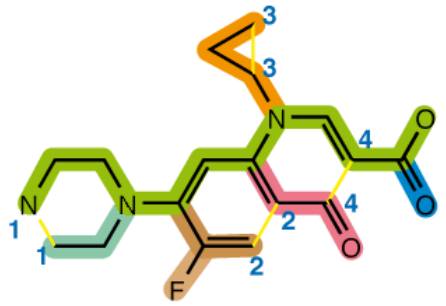
A



B

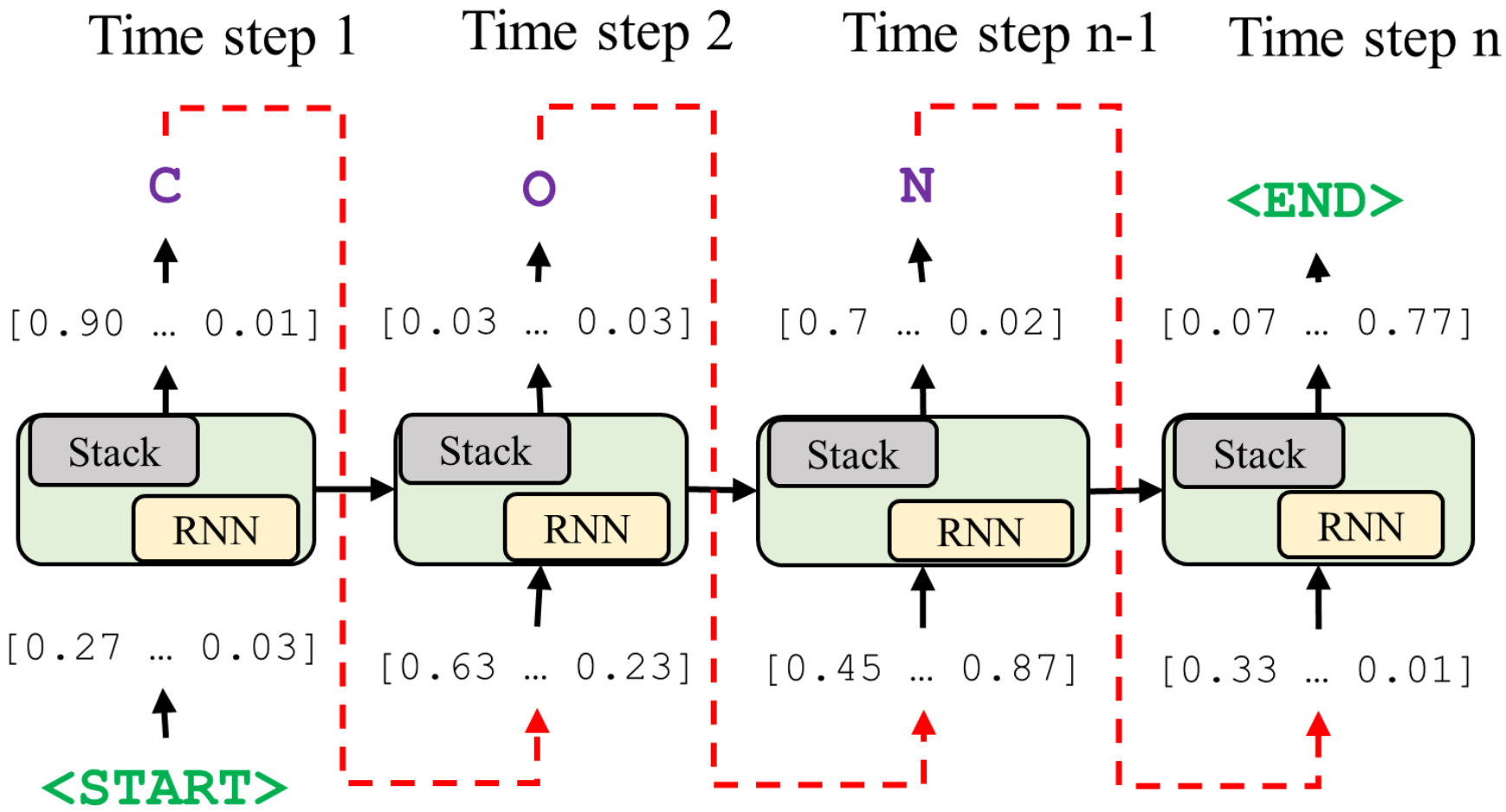


C



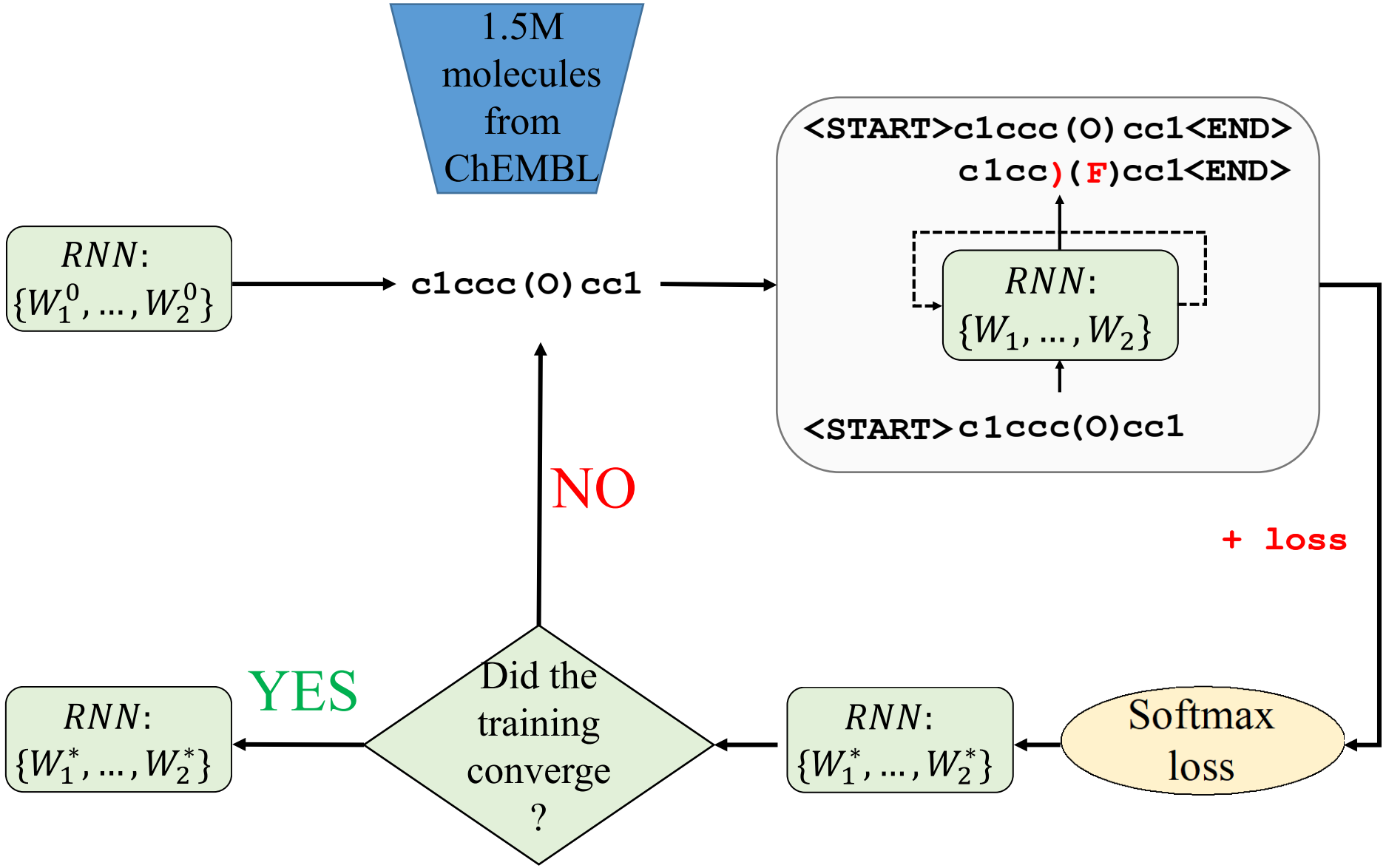
D

N1CCN(CC1)C(C(F)=C2)=CC(=C2C4=O)N(C3CC3)C=C4C(=O)O

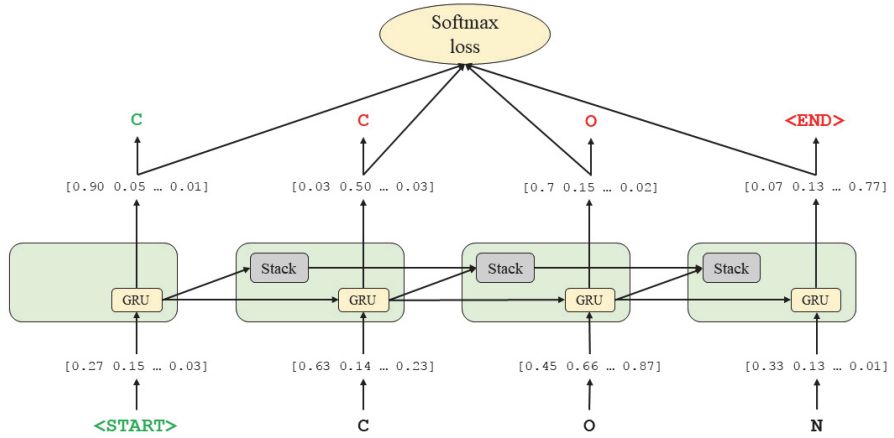


M. Popova & Co "Deep reinforcement learning for de-novo drug design." *Science Advances,* 2018, eaap7885 DOI: 10.1126/sciadv.aap7885

Generative model for molecules



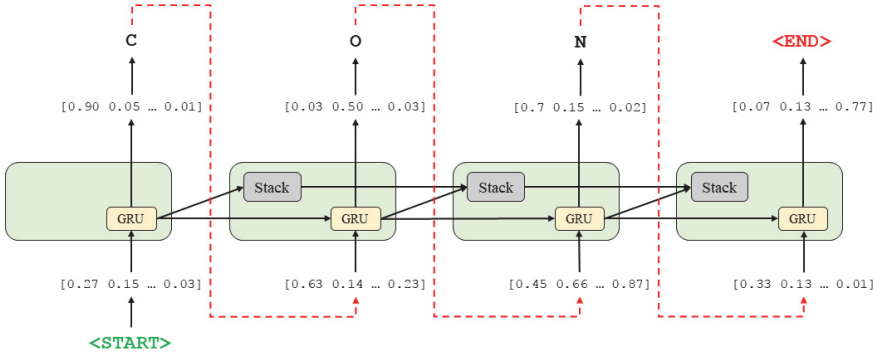
Stack-augmented GRU



A. Joulin & al. Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets (http://arxiv.org/abs/1503.01007)

A stack is a type of persistent memory, which can be only accessed through its topmost element. There are three operations supported by the stack: `POP` operation, which deletes an element from the top of the stack, `PUSH` operation, which puts a new element to the top of the stack; and `NO-OP` operation, which performs no action.

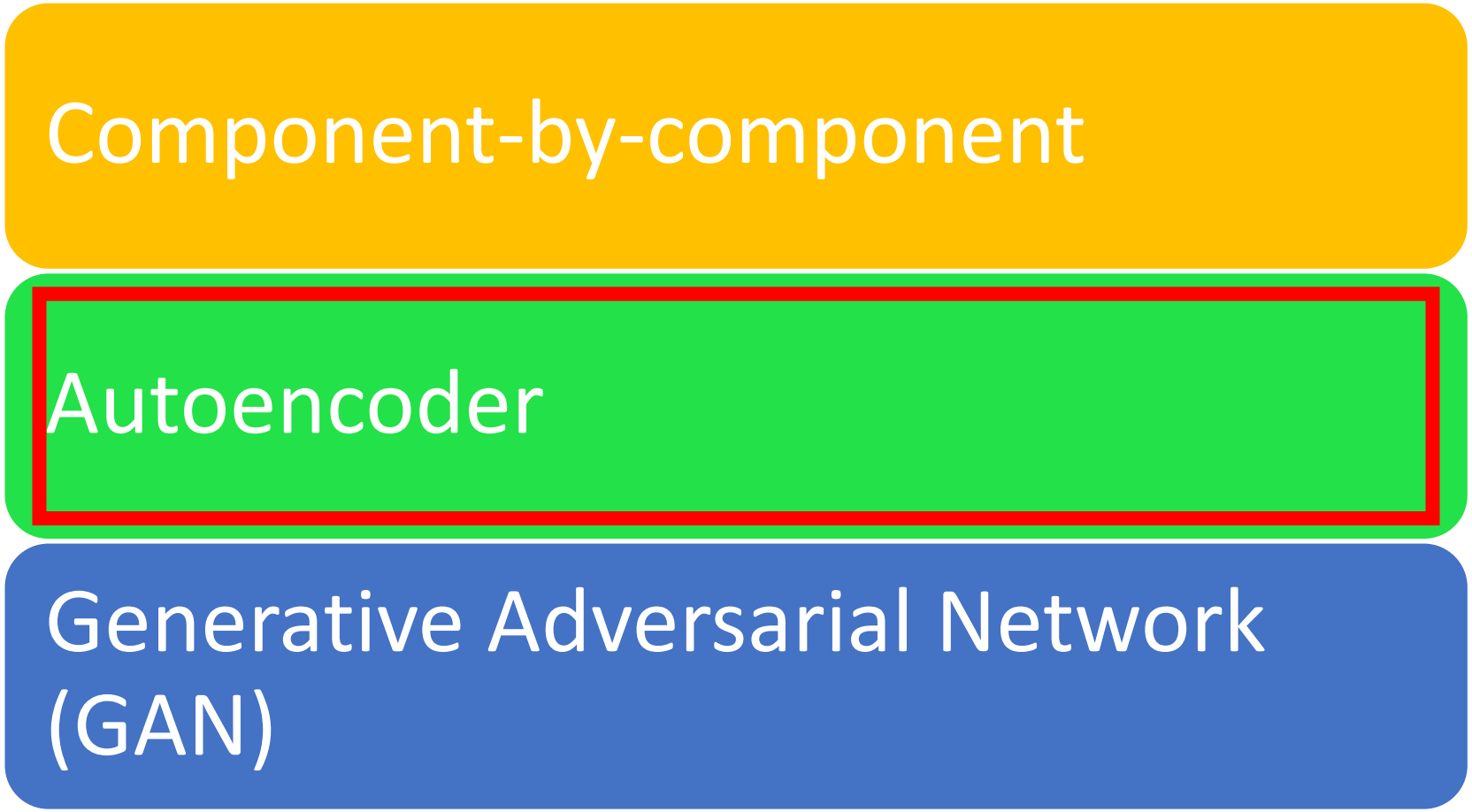
$$s_t[0] = a_t[PUSH]\sigma(Dh_t) + a_t[POP]s_{t-1}[1] + a_t[NO-OP]s_{t-1}[0]$$



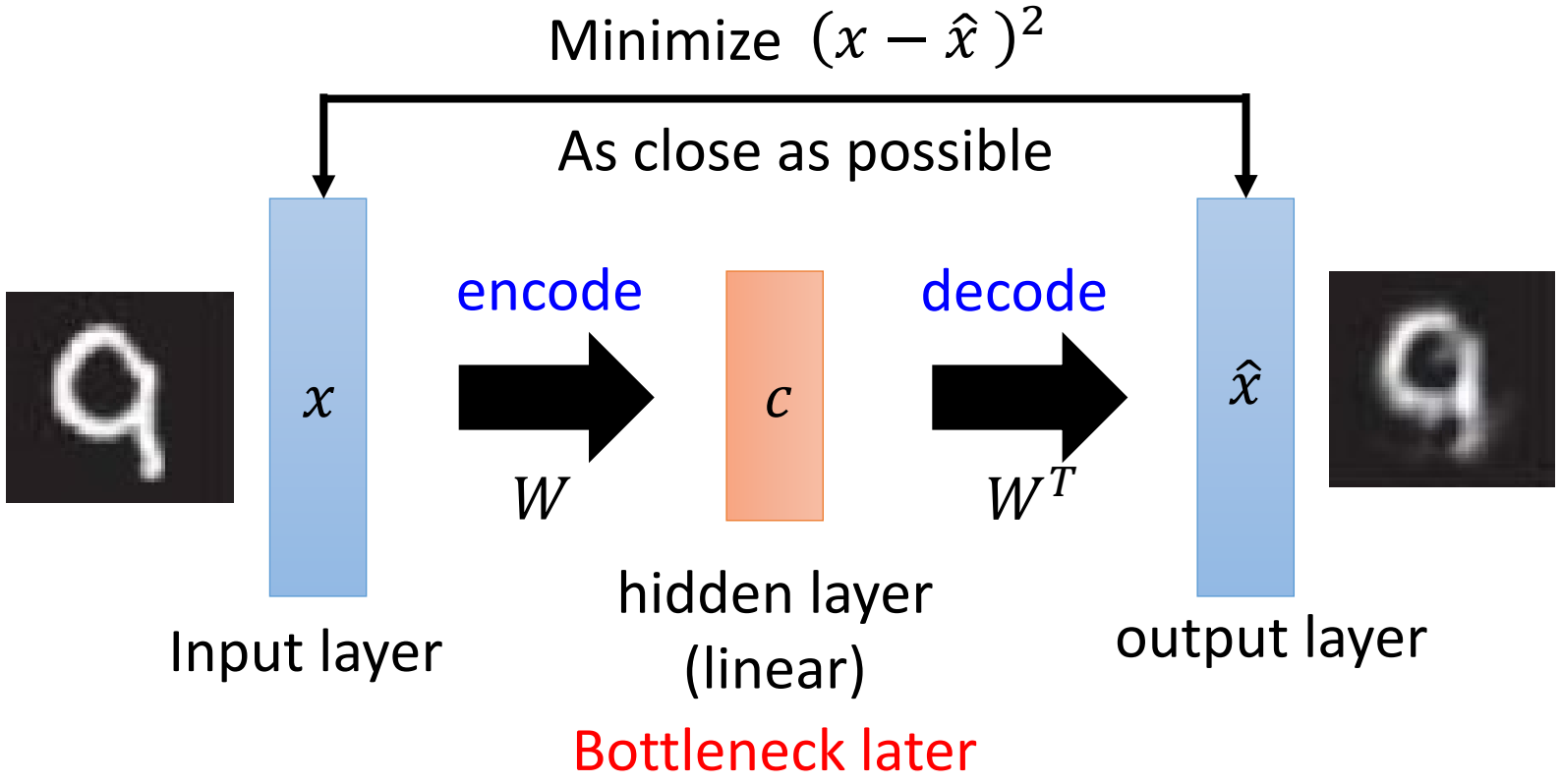
Similar to the Neural Turing Machines

M. Popova & Co "Deep reinforcement learning for de-novo drug design." *Science Advances,* 2018, eaap7885 DOI: 10.1126/sciadv.aap7885

Generative Models

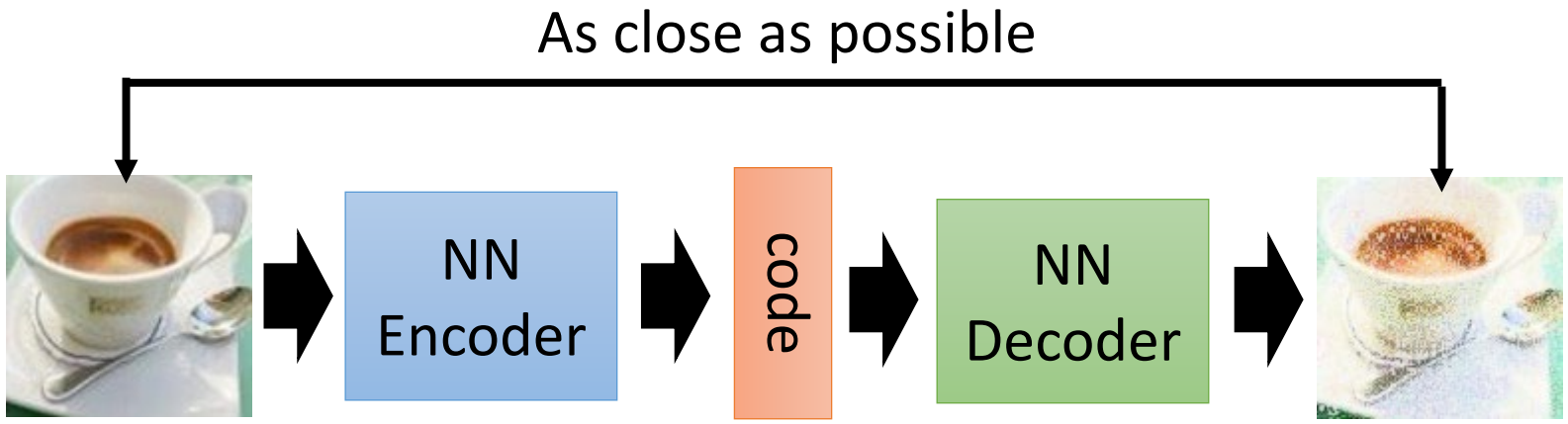


Recap: PCA

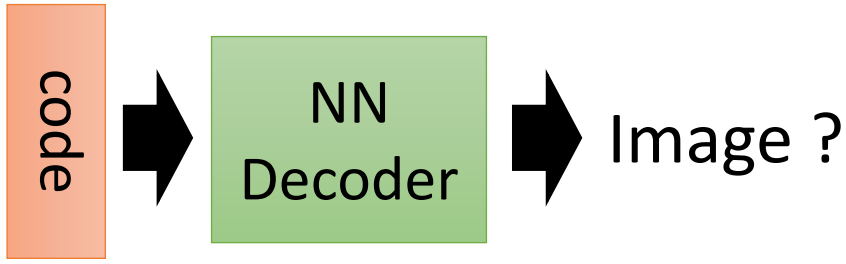


Output of the hidden layer is the code

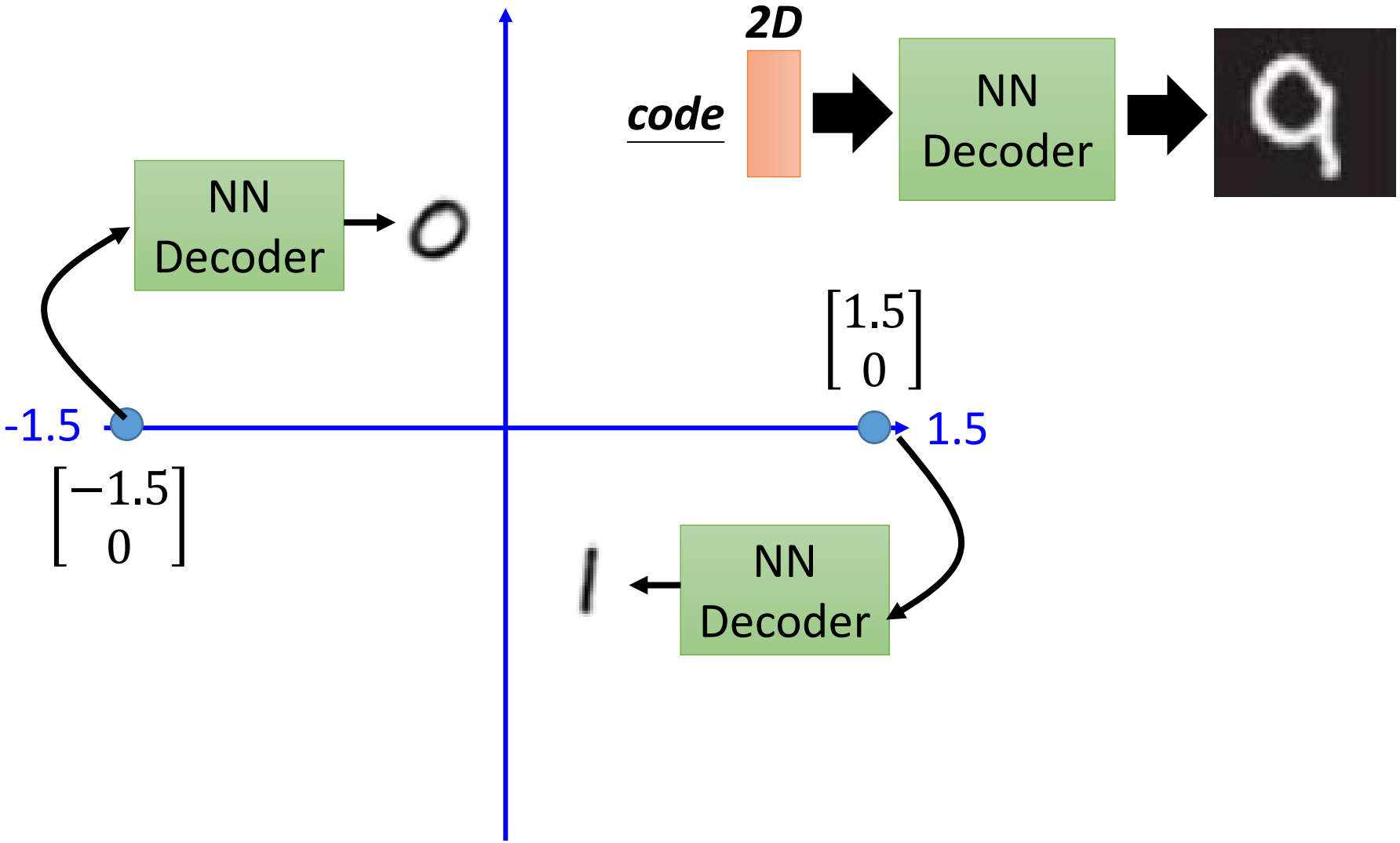
Auto-encoder



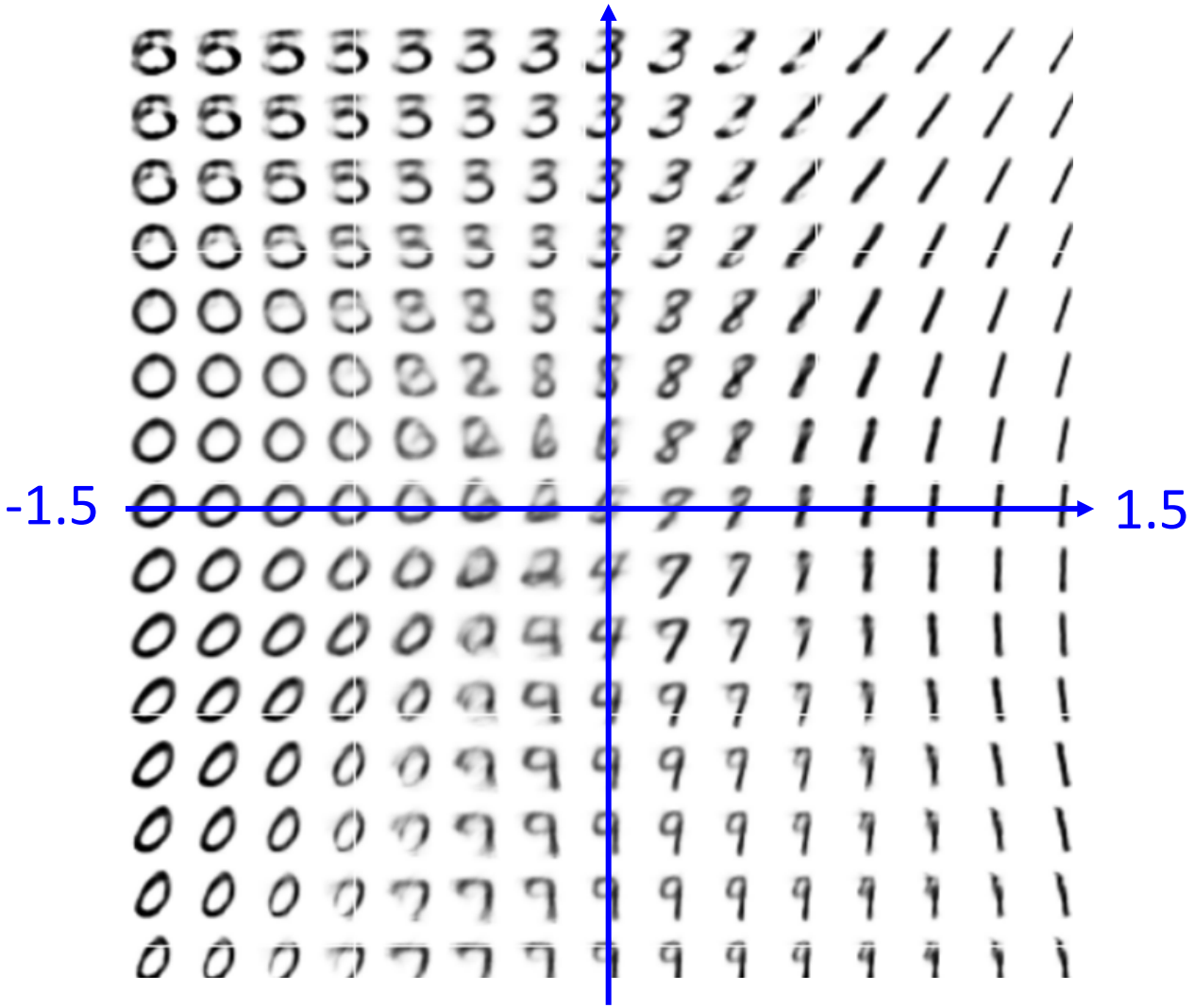
Randomly generate a vector as code



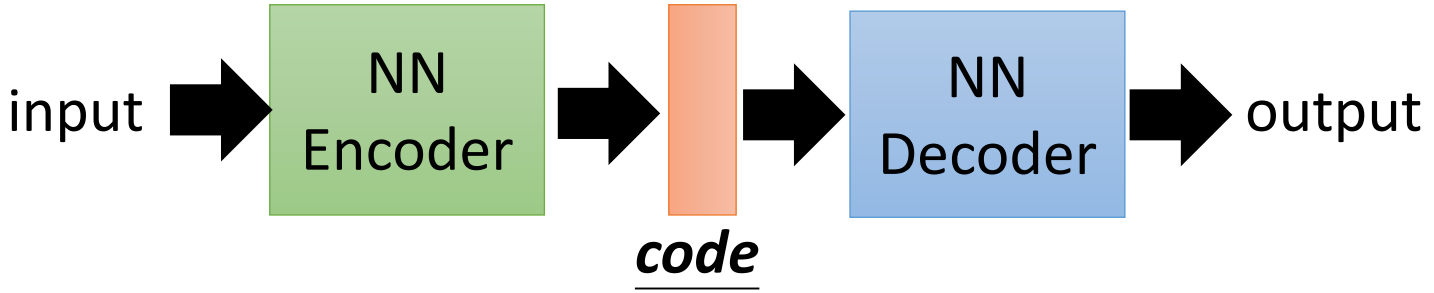
Auto-encoder



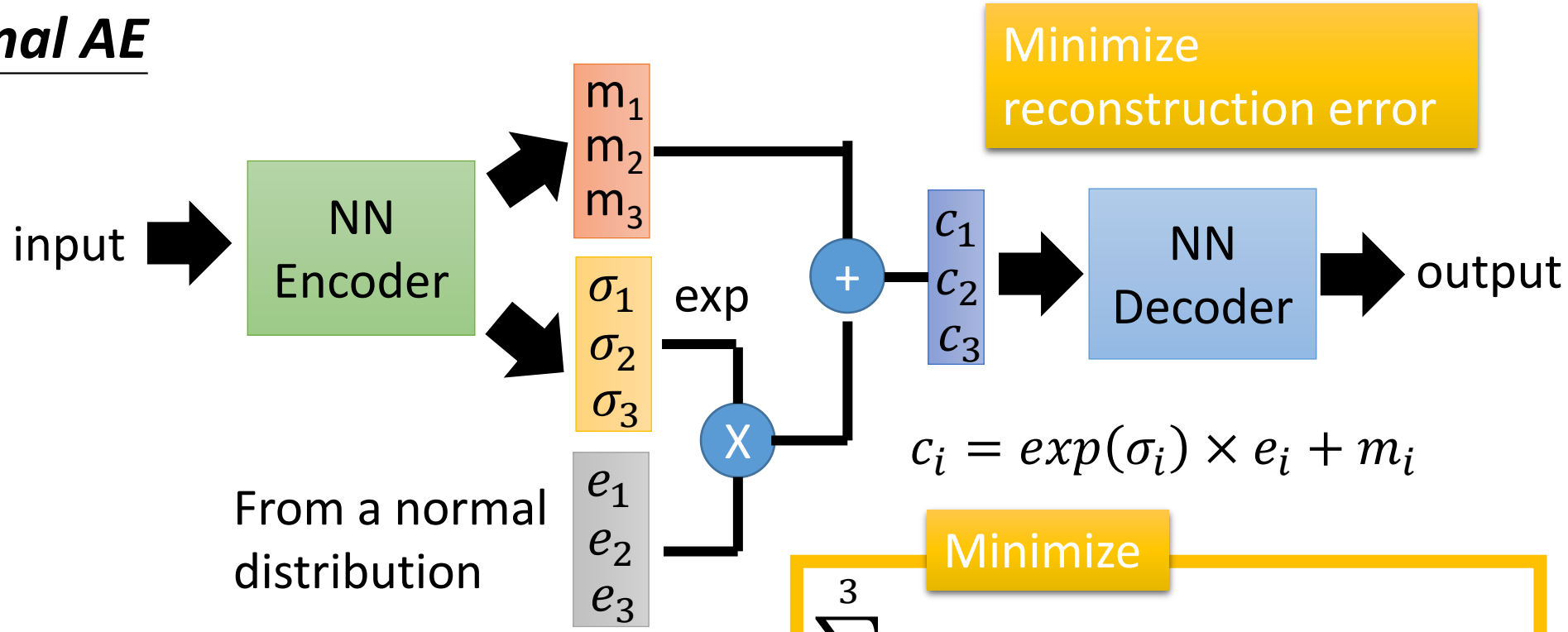
Auto-encoder



Auto-encoder

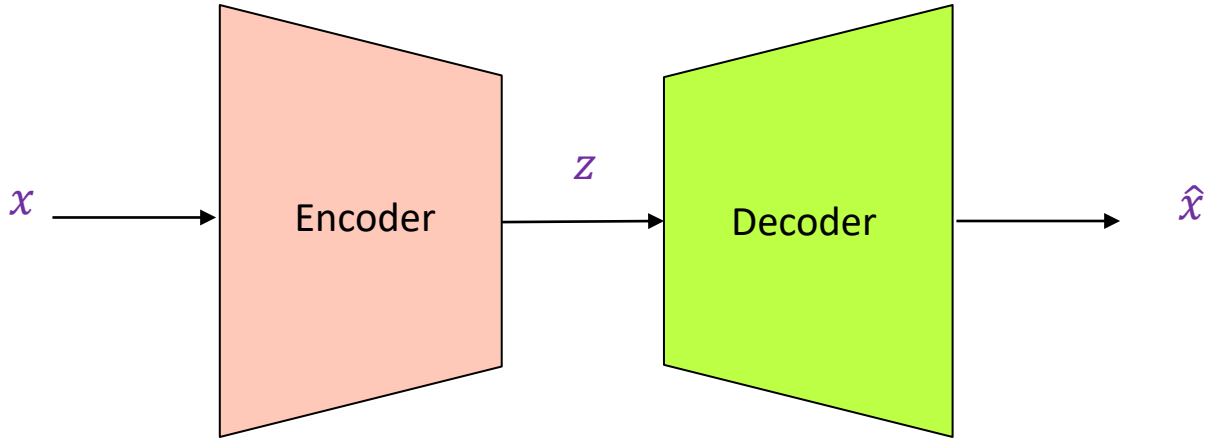


Variational AE



Variational autoencoders: Overview

- Probabilistic formulation based on *variational Bayes* framework
- At training time, jointly learn *encoder* and *decoder* by maximizing (a bound on) the data likelihood
- At test time, discard encoder and use decoder to sample from the learned distribution



D. Kingma and M. Welling, Auto-Encoding Variational Bayes, ICLR 2014

Variational autoencoders: Overview

- Probabilistic generative model of the data distribution:

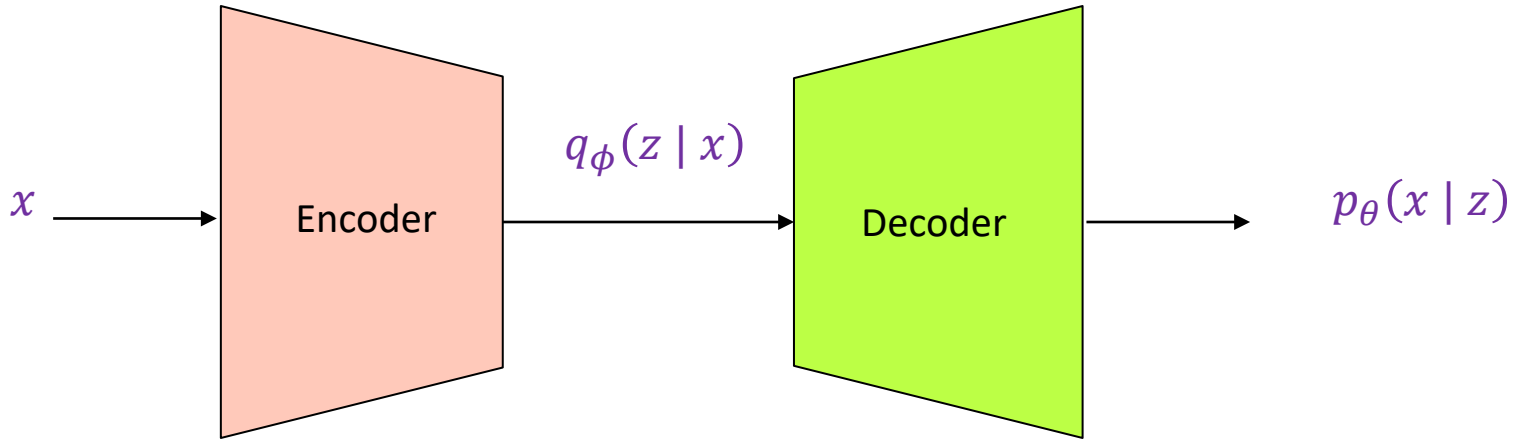
Sample z from *prior* $p(z)$ (usually Gaussian)

Sample x from *conditional* $p(x|z)$

Try to approximate the conditional with neural network

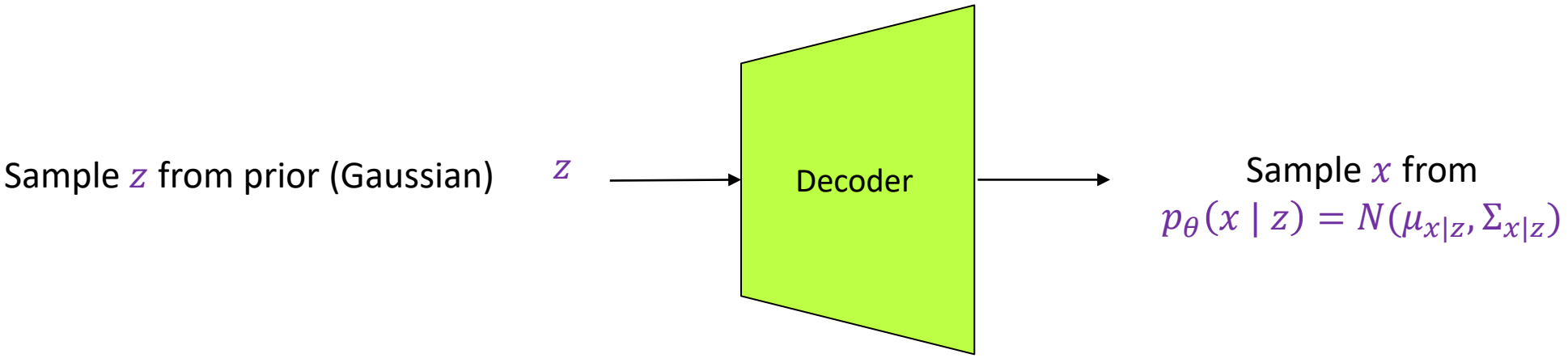
Variational autoencoders: Overview

- At training time, jointly learn *encoder* and *decoder*
- **Encoder:** given inputs x, output $q_\phi(z \mid x)$
  - Specifically, output mean and (diagonal) covariance, or $\mu_{z|x}$ and $\Sigma_{z|x}$, so that $q_\phi(z \mid x) = N(\mu_{z|x}, \Sigma_{z|x})$
- **Decoder:** given z, output $p_\theta(x \mid z)$
  - Specifically, output $\mu_{x|z}$ and $\Sigma_{x|z}$ so that $p_\theta(x \mid z) = N(\mu_{x|z}, \Sigma_{x|z})$
- **Training objective:** (a bound on) data likelihood under the model



Variational autoencoders: Overview

- At test time, discard encoder and use decoder to sample from $p_\theta(x \mid z) = N(\mu_{x|z}, \Sigma_{x|z})$



Variational autoencoders: Generating data

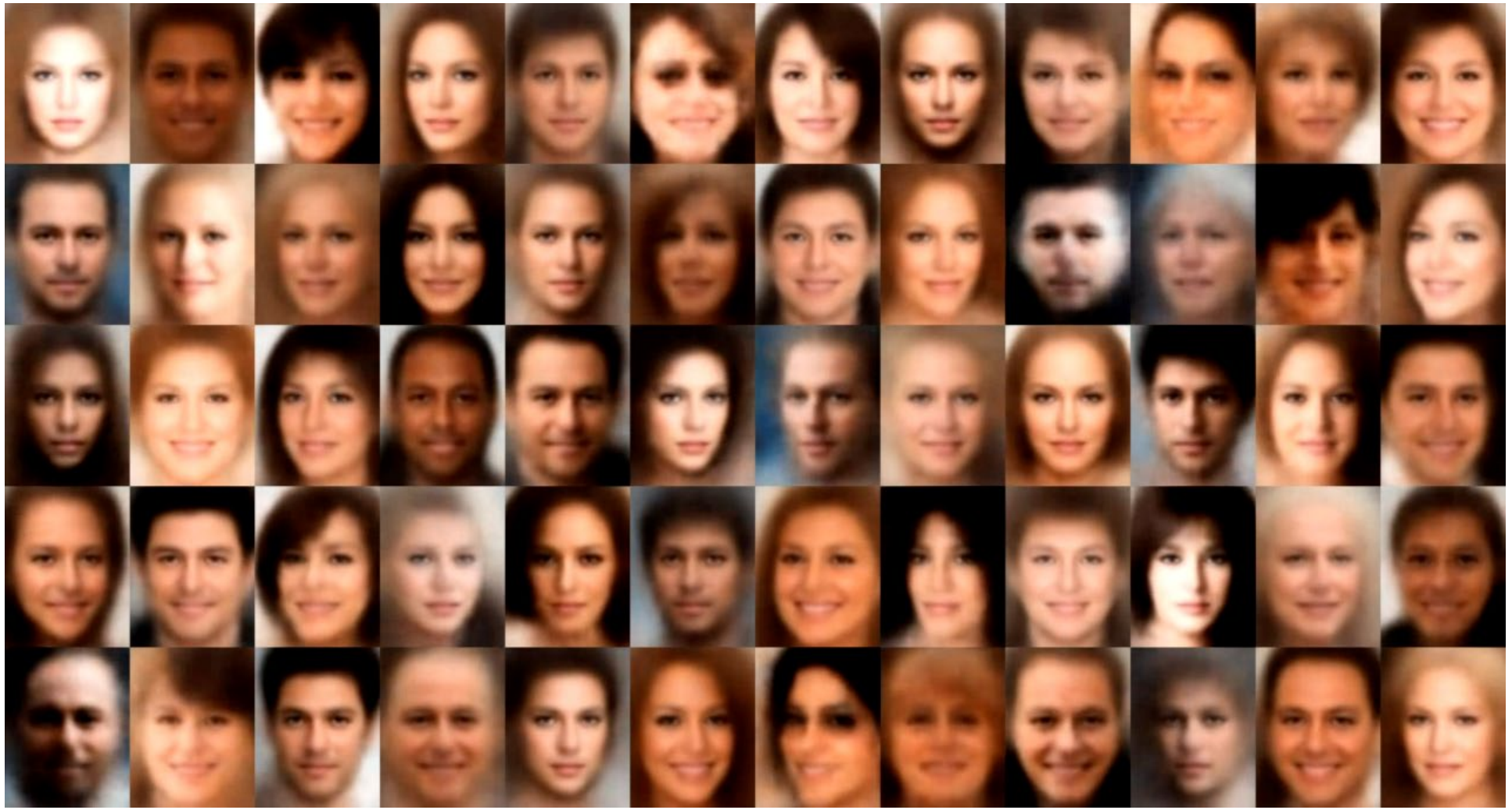


Image source

Basic VAE framework: Summary

Pros:

- Principled mathematical formalism for generative models
- Allows inference of code given image, can be useful for controlling the latent space

Cons:

- Samples blurrier and lower quality compared to GANs

Active areas of research:

- More powerful and flexible approximations for relevant probability distributions
- Combining VAEs and GANs
- Incorporating structure in latent variables, e.g., hierarchical or categorical distributions

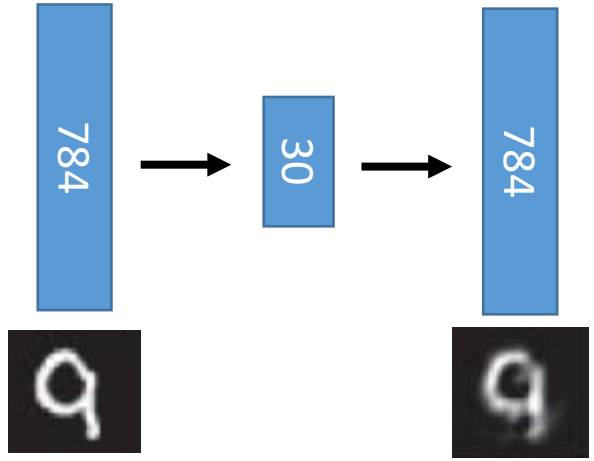
Adapted from J. Johnson

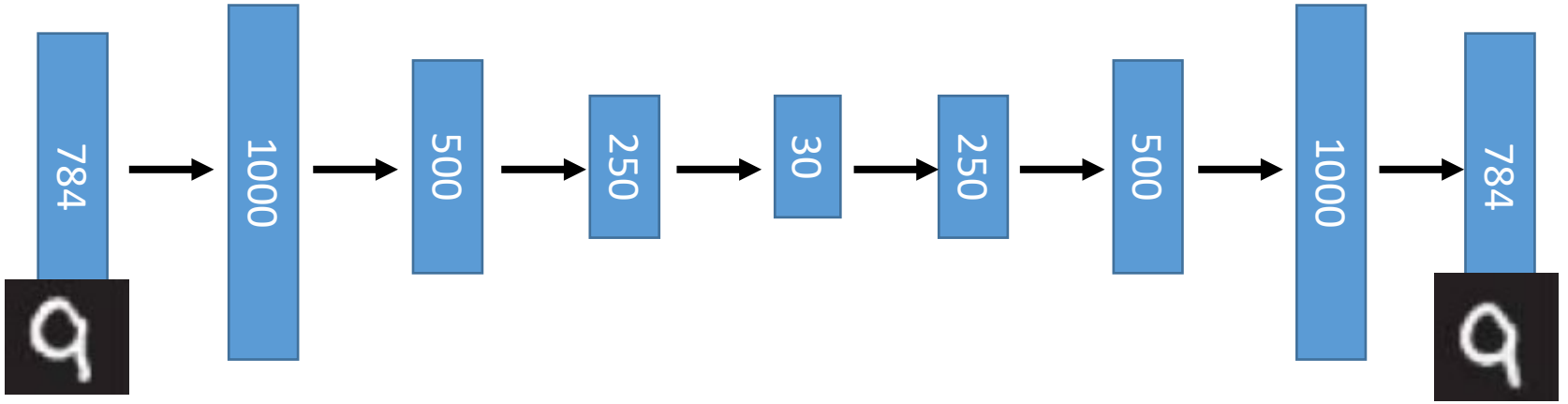
Deep Auto-encoder

Original Image

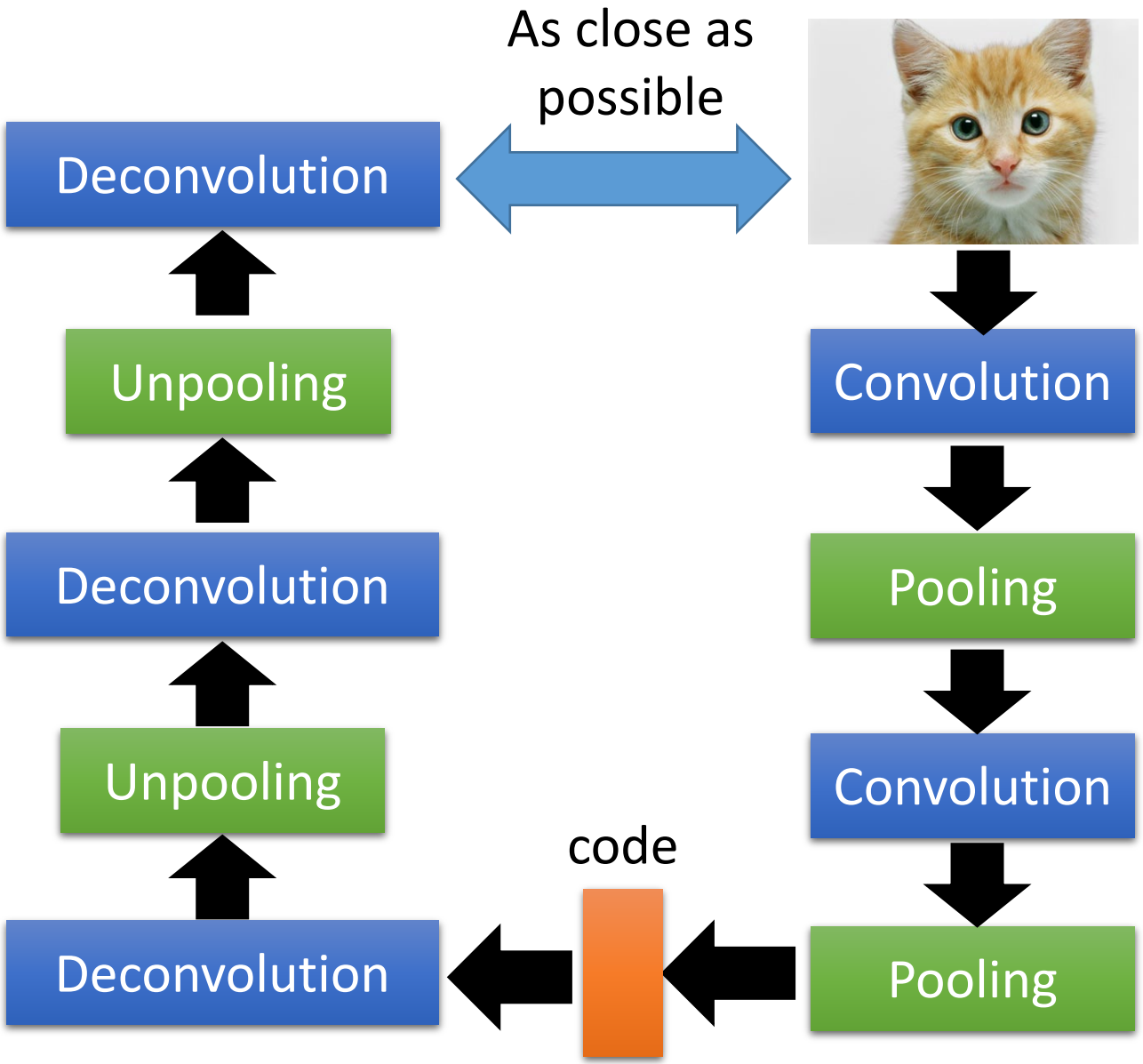
PCA

Deep Auto-encoder





Auto-encoder for CNN



Variational autoencoders: Generating data

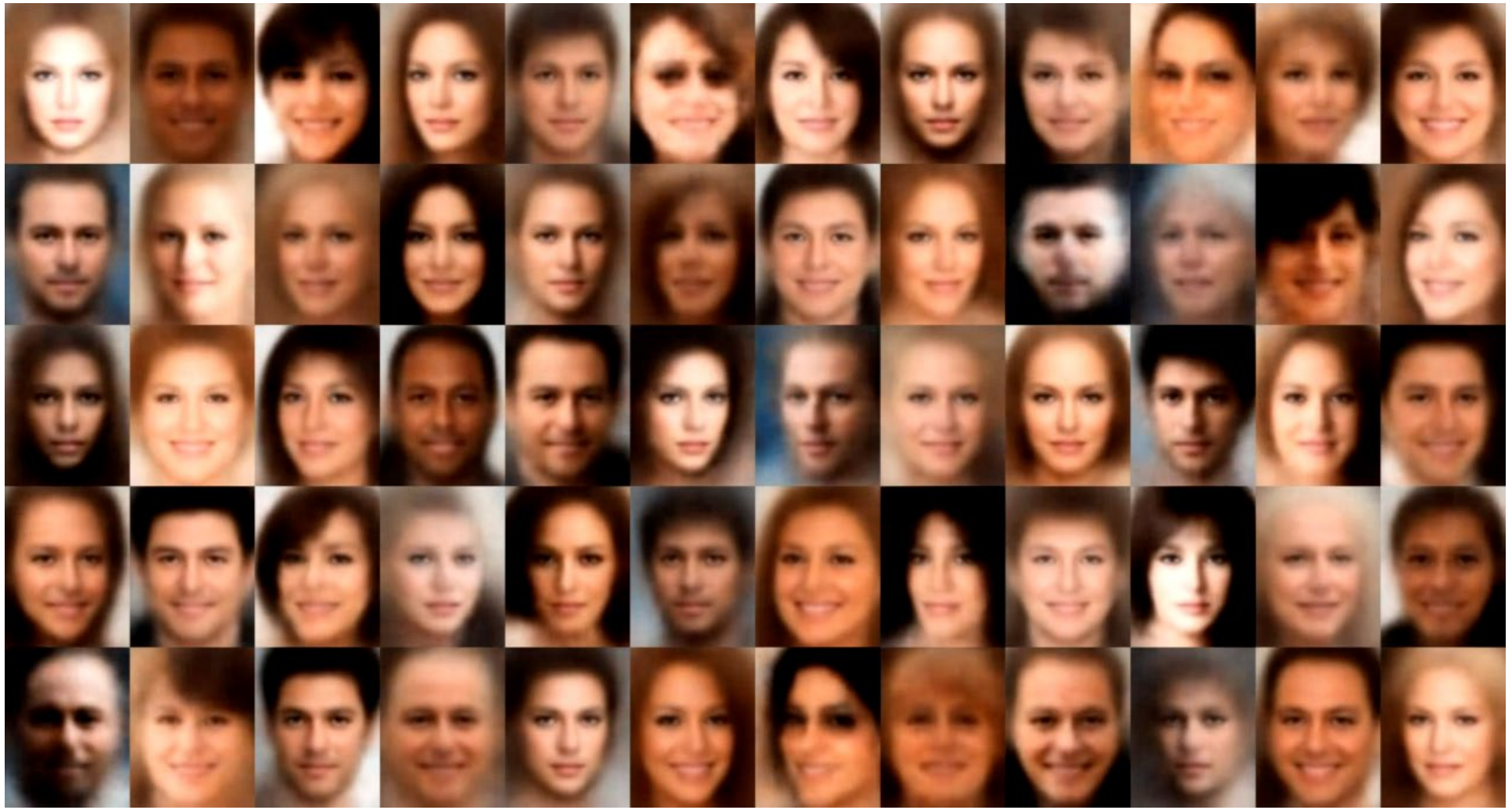


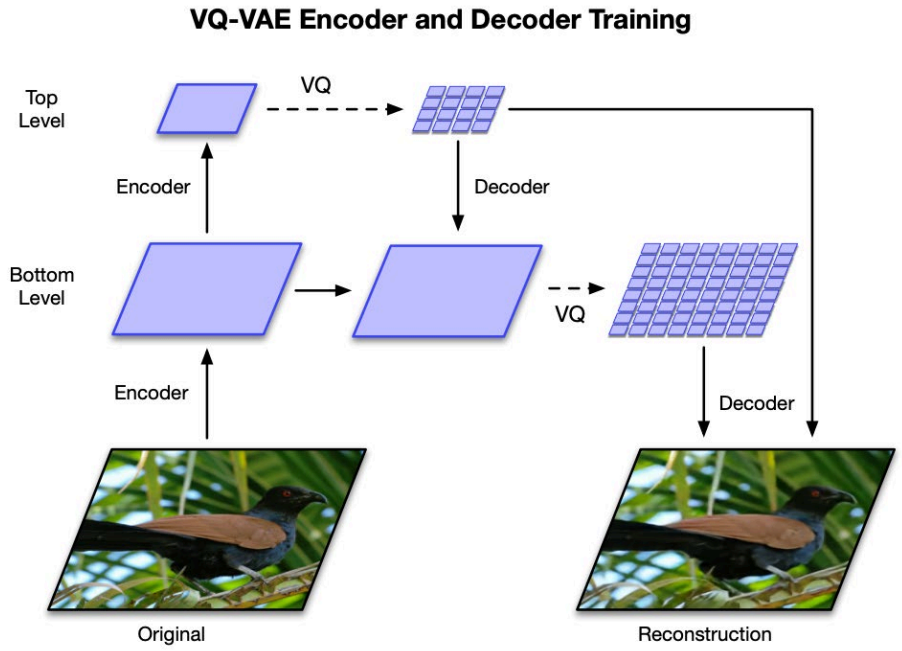
Image source

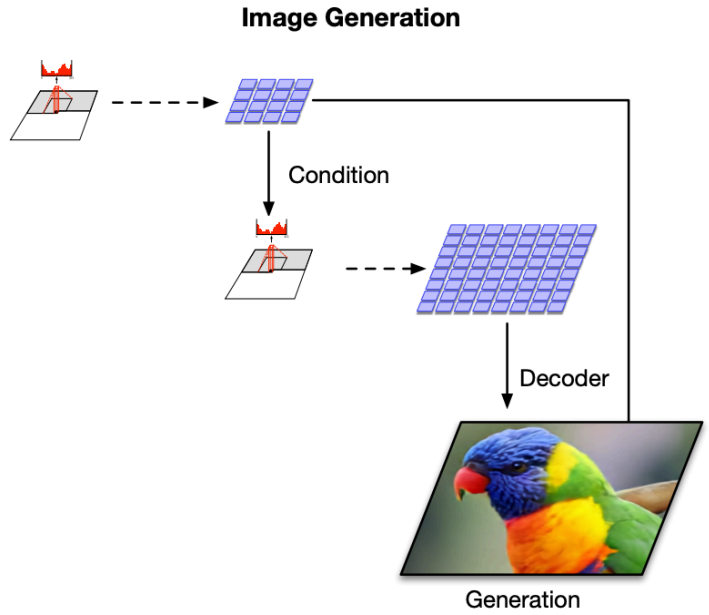
Generating better samples: VQ-VAE-2

Combining VAE and autoregressive models:

Train a VAE-like model to generate multiscale grids of latent codes

Use a multiscale autoregressive model (PixelCNN) to sample in latent code space





A. Razavi, A. van den Oord, O. Vinyals, Generating Diverse High-Fidelity Images with VQ-VAE-2, NeurIPS 2019

Generating better samples: VQ-VAE-2

256 x 256 class-conditional samples, trained on ImageNet:

Generating better samples: VQ-VAE-2

256 x 256 class-conditional samples, trained on ImageNet:

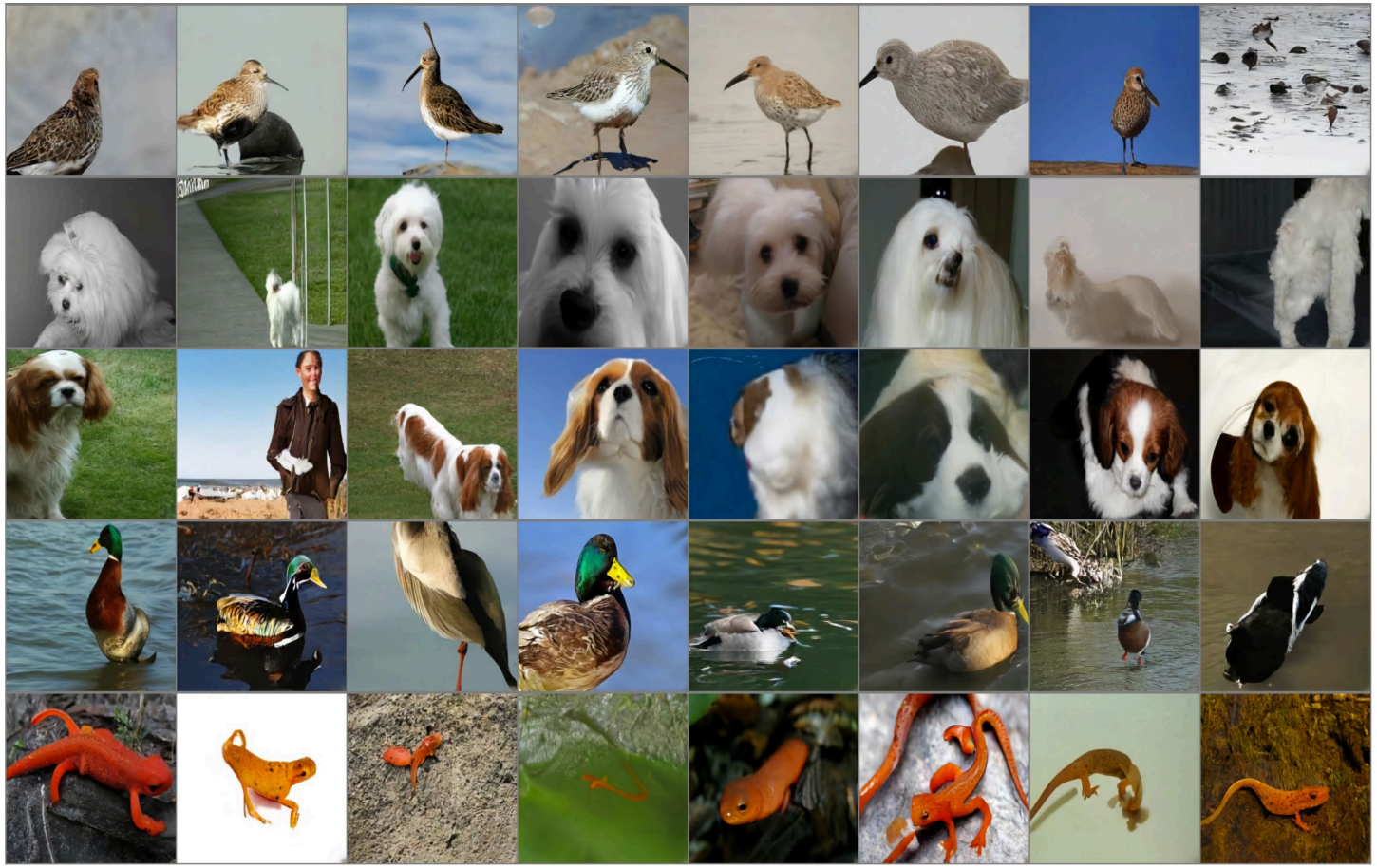
Redshank

Pekinese

Papillon

Drake

Spotted Salamander



Generating better samples: VQ-VAE-2

1024 x 1024 generated faces, trained on FFHQ:

Generating better samples: Hierarchical VAE

Figure 1: 256×256-pixel samples generated by NVAE, trained on CelebA HQ [28].

A. Vahdat, J. Kautz, NVAE: A Deep Hierarchical Variational Autoencoder, arXiv 2020

Combining VAEs and transformers: DALL-E

Train an encoder similar to VQ-VAE to compress images to 32x32 grids of discrete tokens (each assuming 8192 values)

Concatenate with text strings, learn a joint sequential transformer model that can be used to generate image based on text prompt

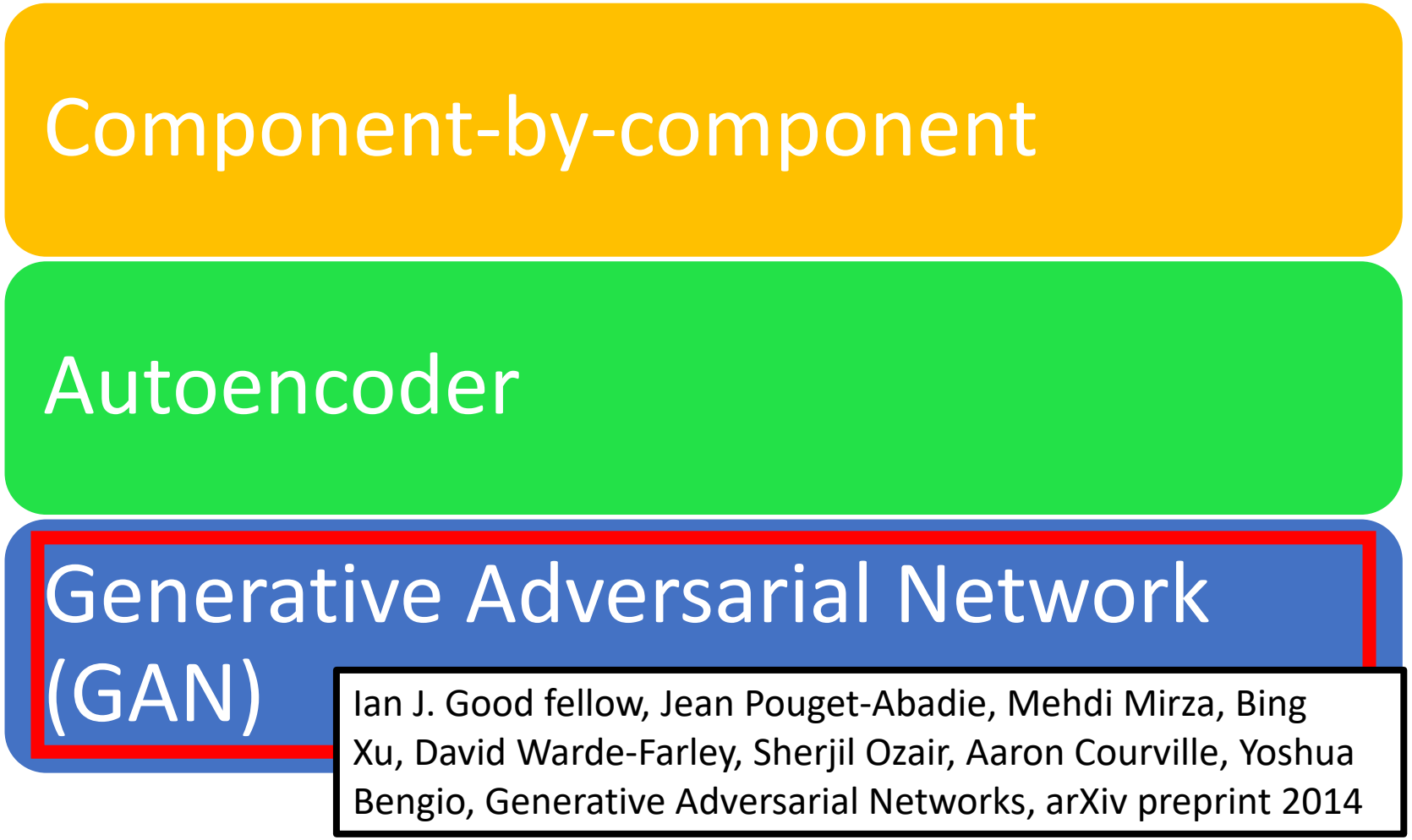
(a) a tapir made of accordion. a tapir with the texture of an accordion.

(b) an illustration of a baby hedgehog in a christmas sweater walking a dog

(c) a neon sign that reads "backprop". a neon sign that reads "backprop". backprop neon sign

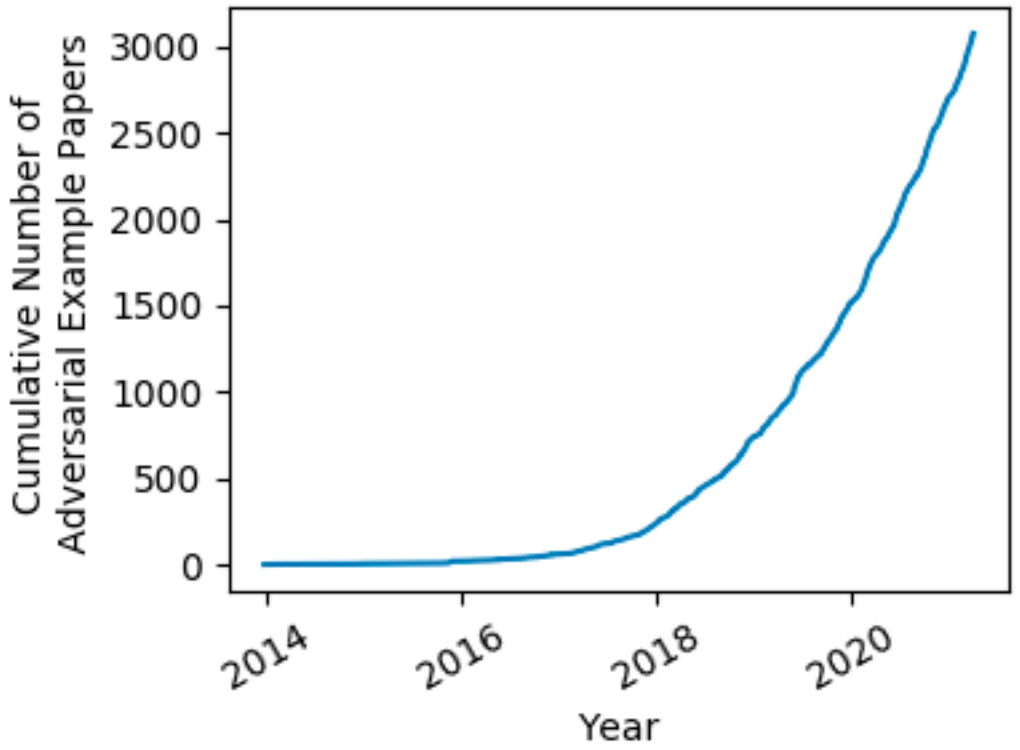
A. Ramesh et al., Zero-Shot Text-to-Image Generation, arXiv 2021
https://openai.com/blog/dall-e/

Generative Models



GANs

- https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html
- https://nicholas.carlini.com/writing/2018/adversarial-machine-learning-reading-list.html



Yann LeCun's comment

What are some recent and potentially upcoming breakthroughs in unsupervised learning?

Yann LeCun, Director of AI Research at Facebook and Professor at NYU
Written Jul 29 · Upvoted by Joaquin Quiñonero Candela, Director Applied Machine Learning at Facebook and Huang Xiao

Adversarial training is the coolest thing since sliced bread.

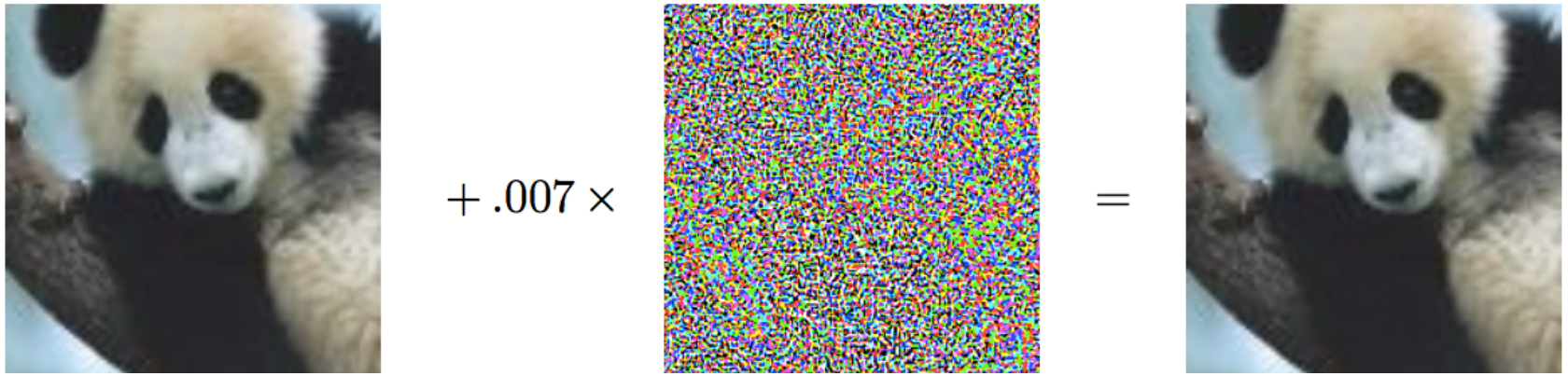
I've listed a bunch of relevant papers in a previous answer.

Expect more impressive results with this technique in the coming years.

What's missing at the moment is a good understanding of it so we can make it work reliably. It's very finicky. Sort of like ConvNet were in the 1990s, when I had the reputation of being the only person who could make them work (which wasn't true).

https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-unsupervised-learning

Fooling neural networks

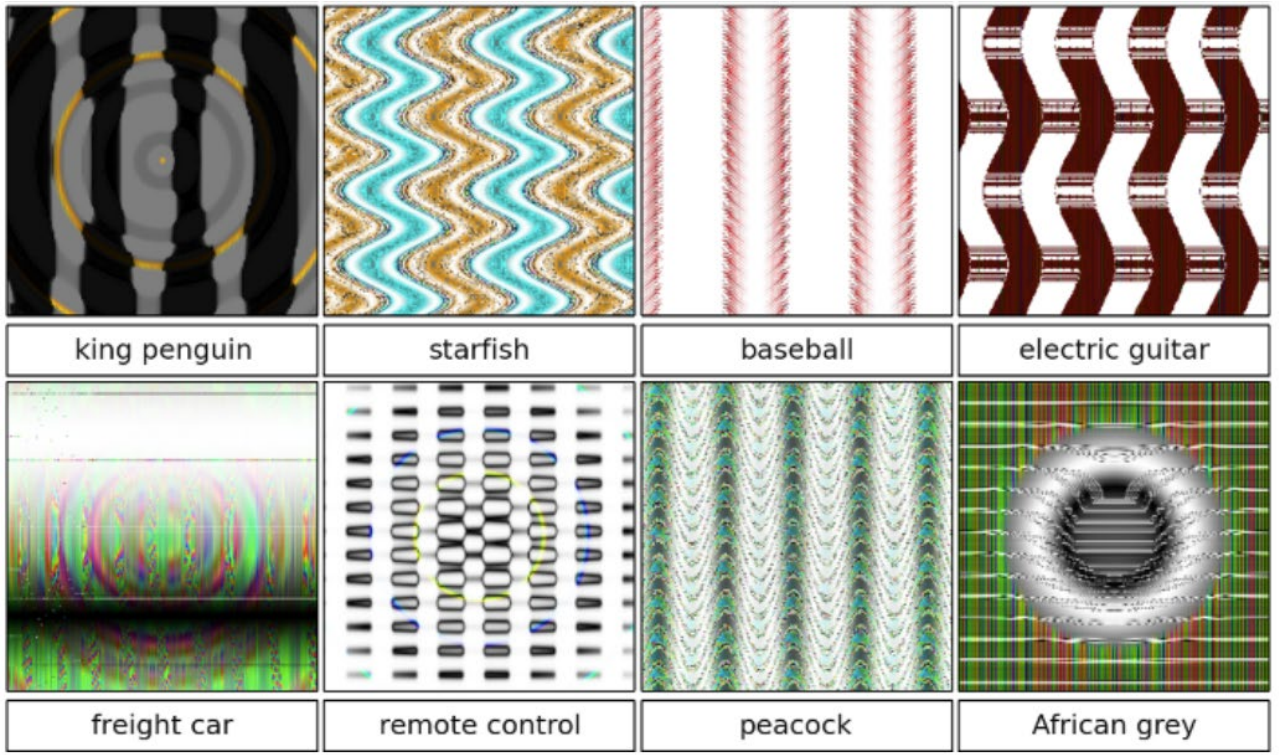


"panda"
57.7% confidence

"gibbon"
99.3 % confidence

Generating preferred inputs

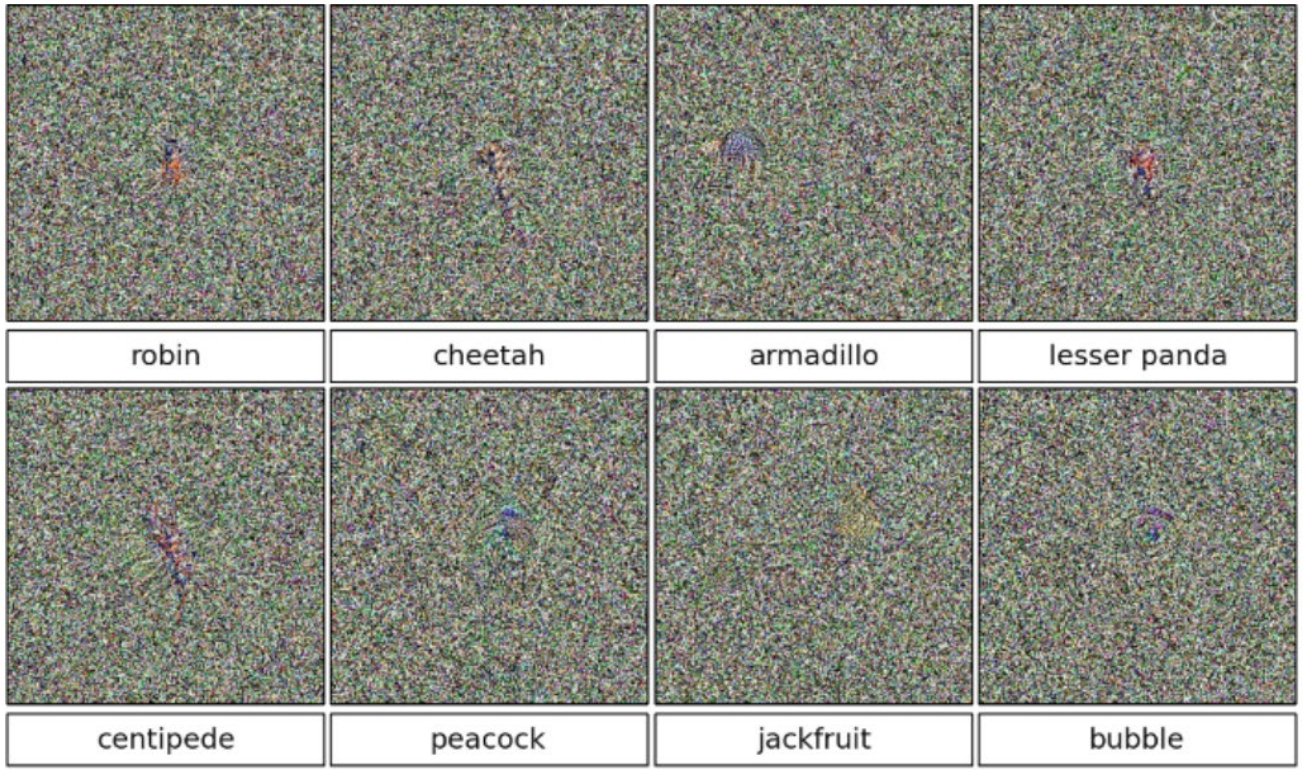
Related finding: it is easy to generate perceptually meaningless images that will be classified as any given class with high confidence



A. Nguyen, J. Yosinski, J. Clune, Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images, CVPR 2015

Generating preferred inputs

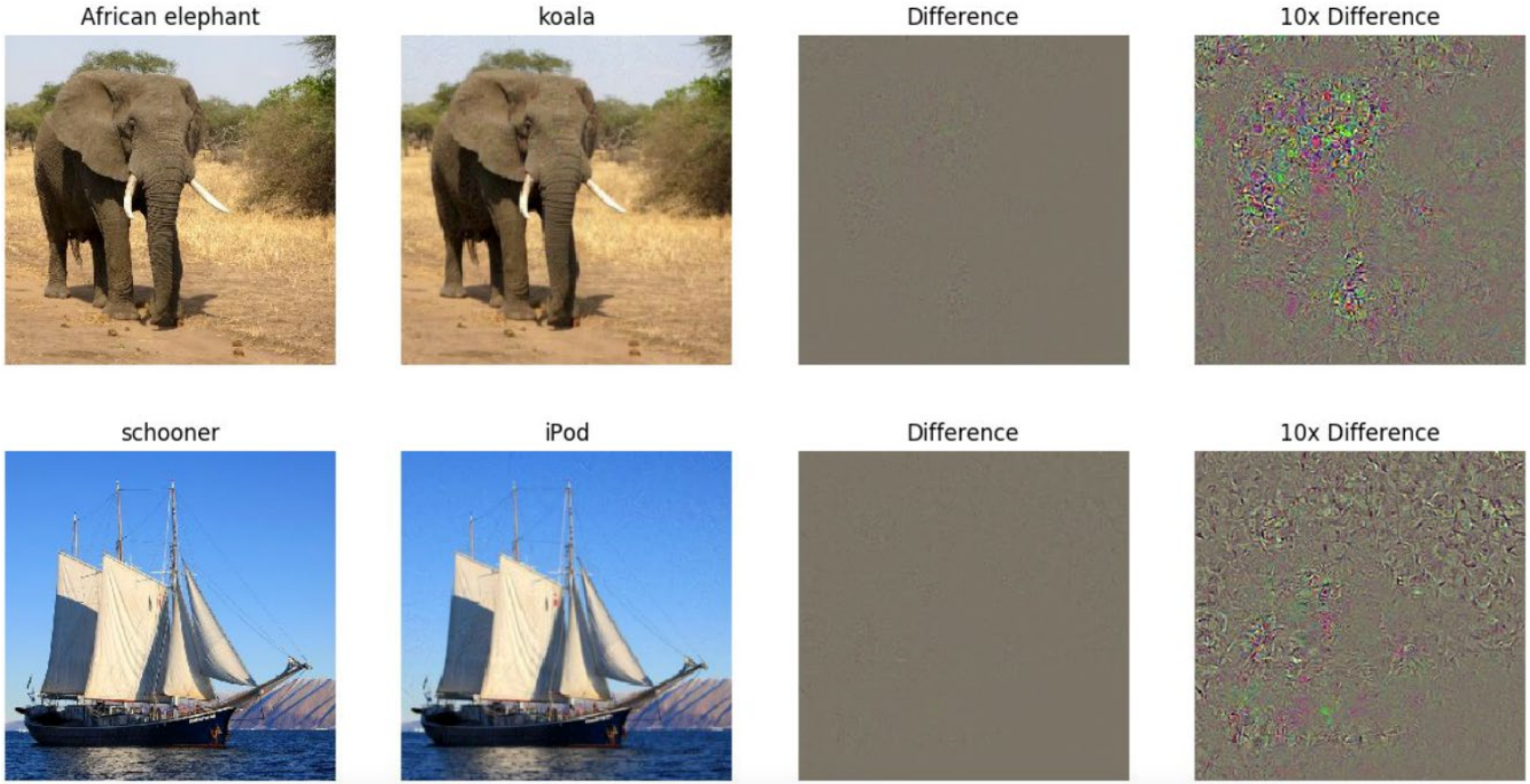
It is easy to generate perceptually meaningless images that will be classified as any given class with high confidence



A. Nguyen, J. Yosinski, J. Clune, Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images, CVPR 2015

Adversarial examples

We can “fool” a neural network by imperceptibly perturbing an input image so it is misclassified



Source: Stanford CS231n

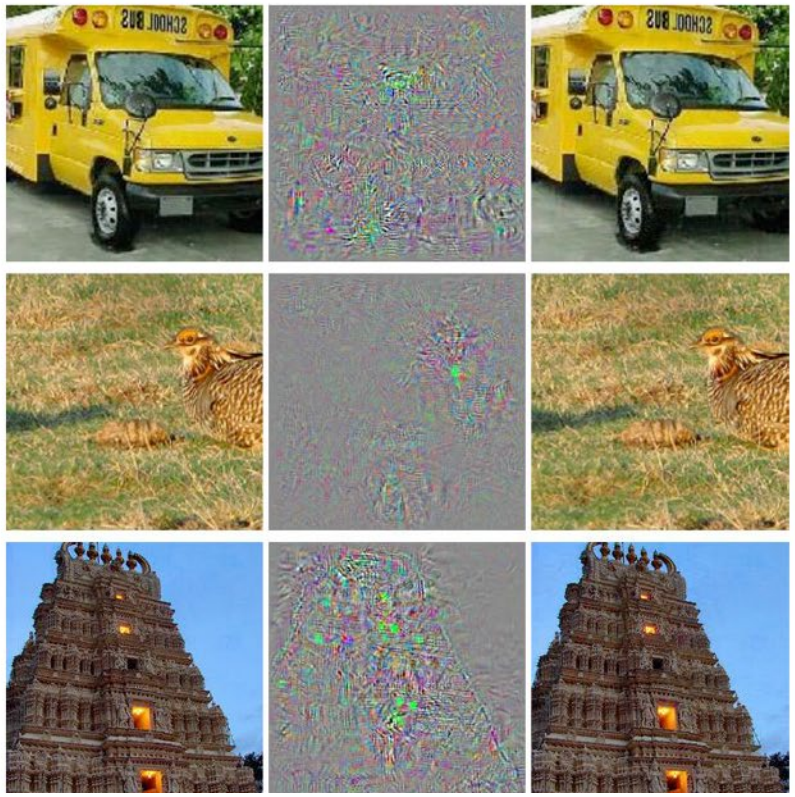
Finding the smallest adversarial perturbation

- Start with correctly classified image x
- Find perturbation r minimizing $\|r\|_2$ such that
  - $x + r$ is misclassified (or classified as specific target class)
  - All values of $x + r$ are in the valid range
- This is constrained non-convex optimization, which the authors solve with L-BFGS

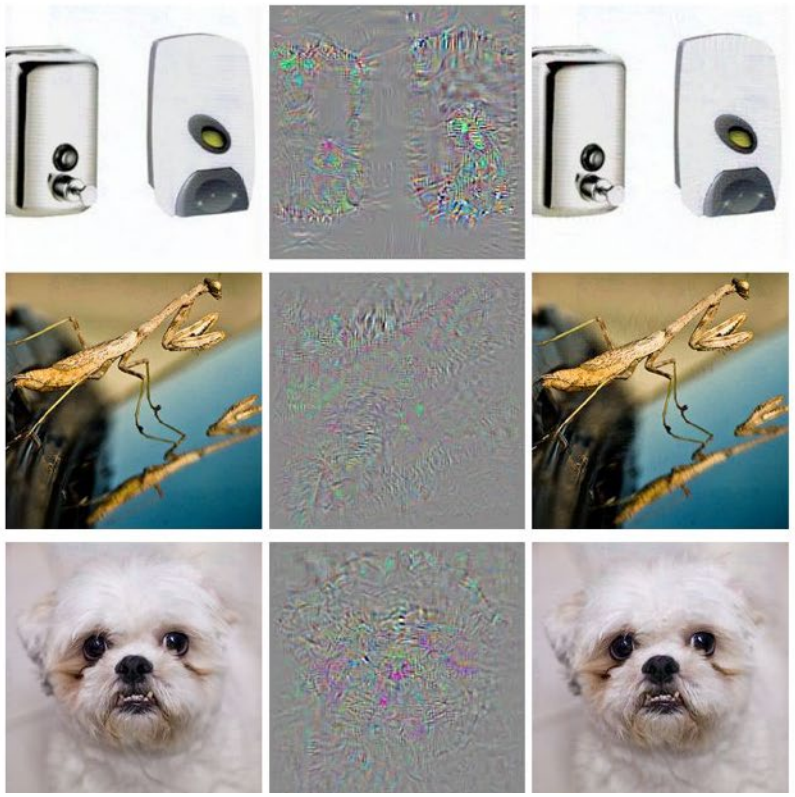
C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, ICLR 2014

Finding the smallest adversarial perturbation

Sample results:



Input | Perturbation x 10 | "Ostrich"



Input | Perturbation x 10 | "Ostrich"

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, ICLR 2014

Gradient ascent

- Rather than searching for the smallest possible perturbation, it is easier to take small gradient steps in desired direction
- Decrease score (increase loss) of *correct* class y^*:

$$x \leftarrow x - \eta \frac{\partial f(x, y^*)}{\partial x} \quad \text{or} \quad x \leftarrow x + \eta \frac{\partial L(x, y^*)}{\partial x}$$

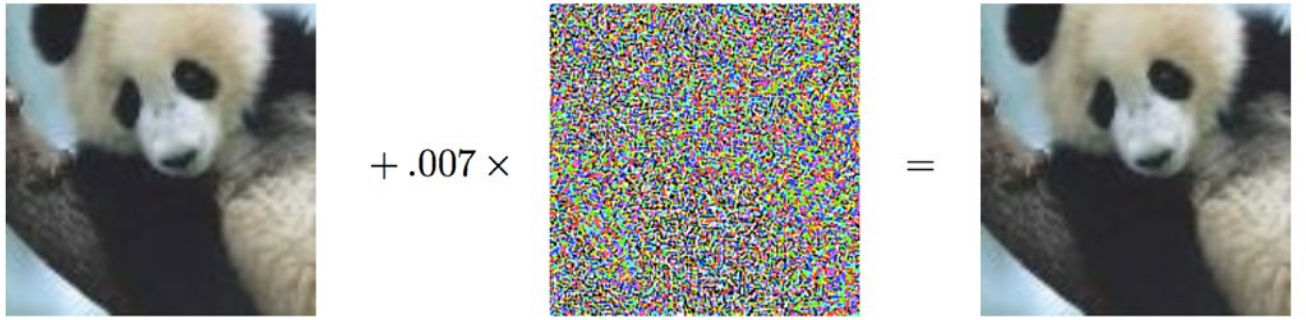
- Increase score (decrease loss) of *incorrect* target class $\hat{y}$:

$$x \leftarrow x + \eta \frac{\partial f(x, \hat{y})}{\partial x} \quad \text{or} \quad x \leftarrow x - \eta \frac{\partial L(x, \hat{y})}{\partial x}$$

Generating adversarial examples

- **Fast gradient sign method:** Find the gradient of the loss w.r.t. correct class y^*, take element-wise sign, update in resulting direction:

$$x \leftarrow x + \epsilon \, \mathrm{sgn}\left(\frac{\partial L(x, y^*)}{\partial x}\right)$$



I. Goodfellow, J. Schlens, C. Szegedy, Explaining and harnessing adversarial examples, ICLR 2015

Generating adversarial examples

- **Fast gradient sign method:**

$$x \leftarrow x + \epsilon \operatorname{sgn}\left(\frac{\partial L(x, y^*)}{\partial x}\right)$$

- **Iterative gradient sign method:** take multiple smaller steps until misclassified, each time clip result to be within ϵ-neighborhood of original image
- ***Least likely class* method:** try to misclassify image as class $\hat{y}$ with *smallest* initial score:

$$x \leftarrow x - \epsilon \operatorname{sgn}\left(\frac{\partial L(x, \hat{y})}{\partial x}\right)$$

A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the real world, ICLR 2017 workshop

Generating adversarial examples

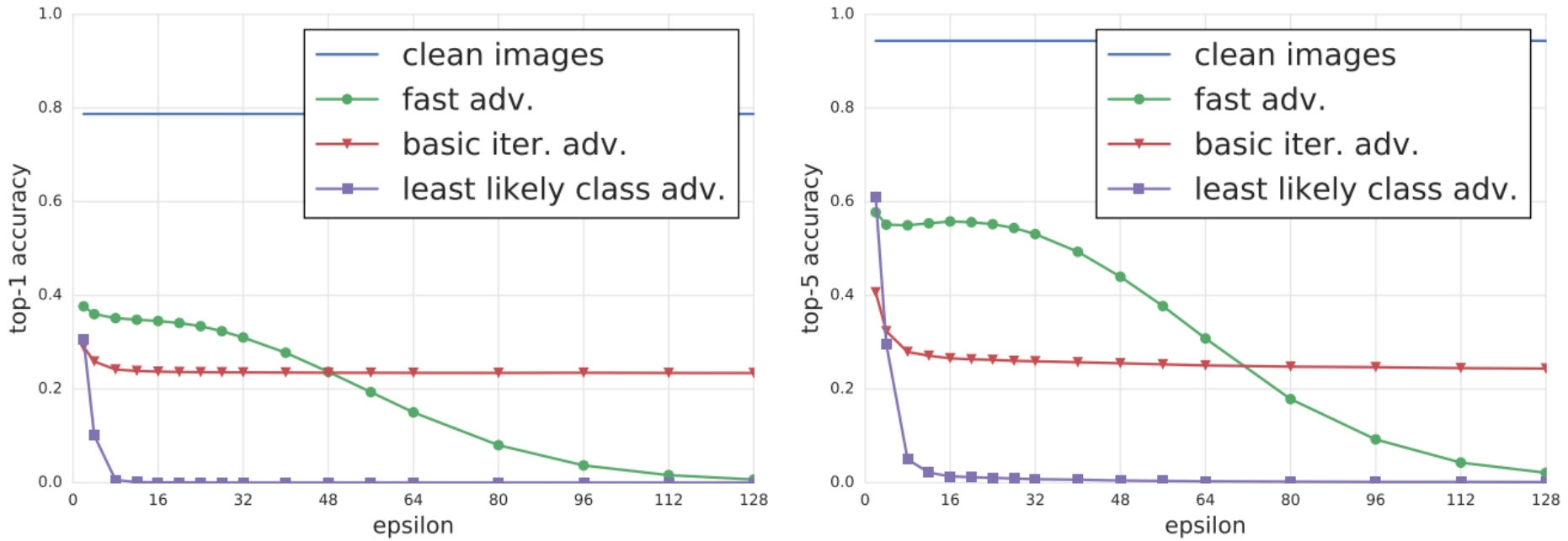
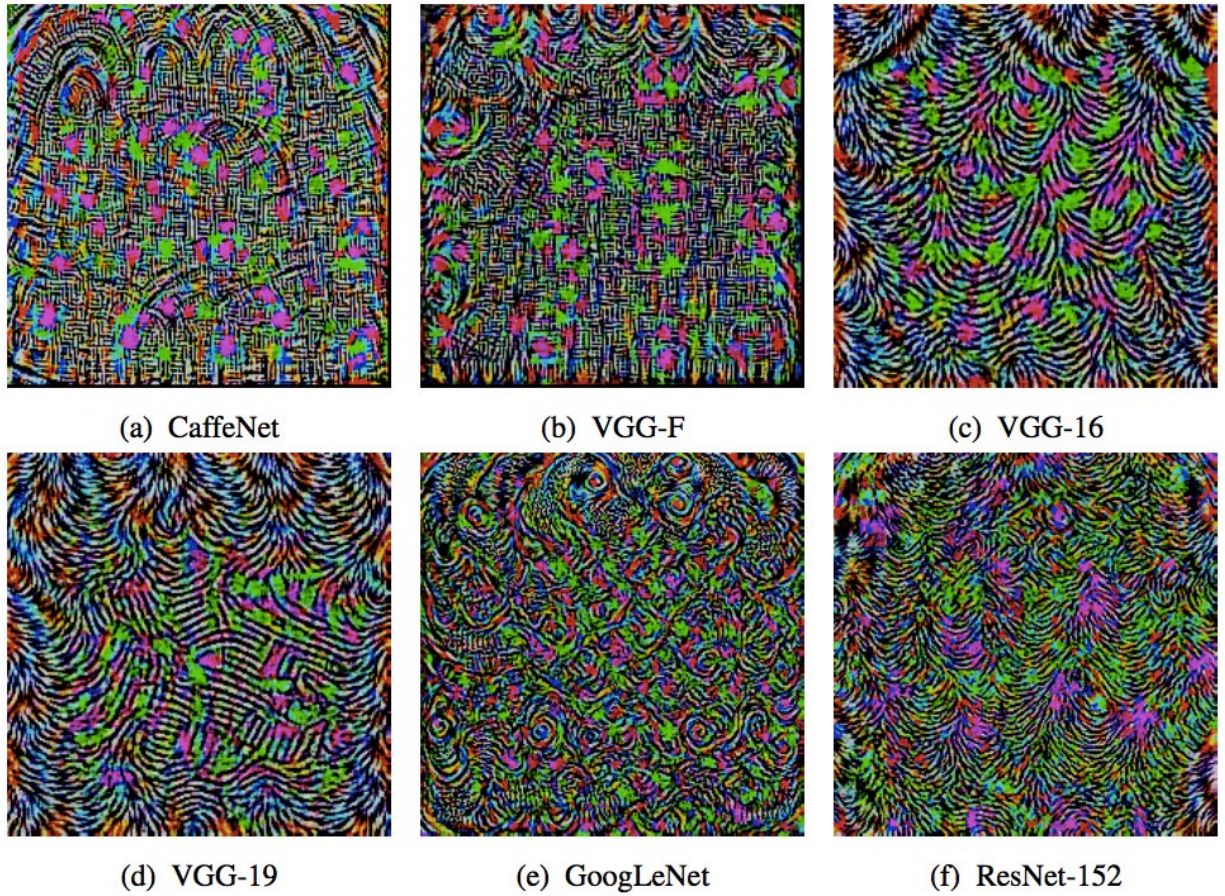


Figure 2: Top-1 and top-5 accuracy of Inception v3 under attack by different adversarial methods and different ϵ compared to "clean images" — unmodified images from the dataset. The accuracy was computed on all $50,000$ validation images from the ImageNet dataset. In these experiments ϵ varies from 2 to 128.

A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the real world, ICLR 2017 workshop

Universal adversarial perturbations

- Perturbation vectors computed from different architectures:



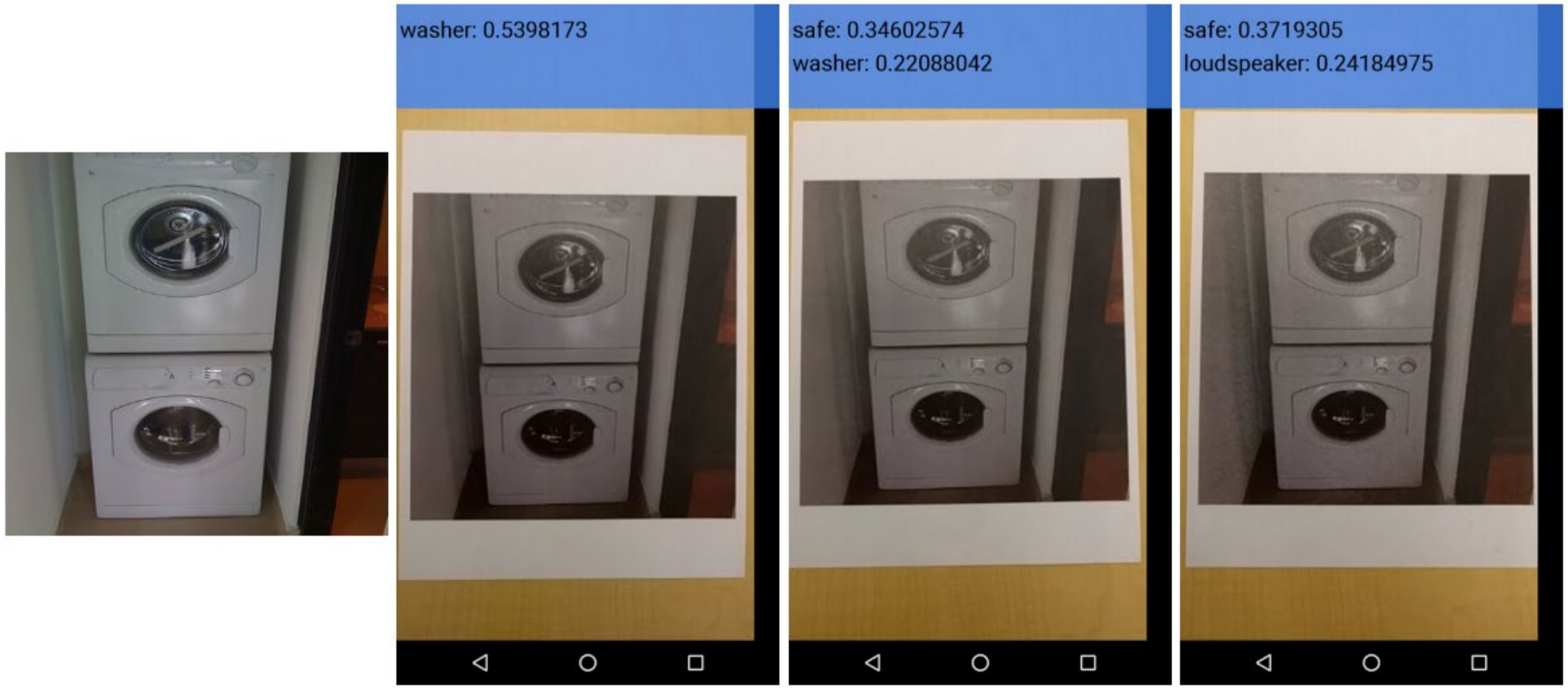
(a) CaffeNet (b) VGG-F (c) VGG-16

(d) VGG-19 (e) GoogLeNet (f) ResNet-152

S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, Universal adversarial perturbations, CVPR 2017

Printed adversarial examples

"Black box" attack on a cell phone app: take a clean image, add perturbation, print out, classify with TensorFlow Camera Demo app



(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the real world, ICLR 2017 workshop

Defending against adversarial examples

- Adversarial training: networks can be made somewhat resistant by augmenting or regularizing training with adversarial examples (Goodfellow et al. 2015, Tramer et al. 2018)

Defending against adversarial examples

- Adversarial training: networks can be made somewhat resistant by augmenting or regularizing training with adversarial examples (Goodfellow et al. 2015, Tramer et al. 2018)
- Adversarial objectives can also be formulated (Madry et al., 2018)

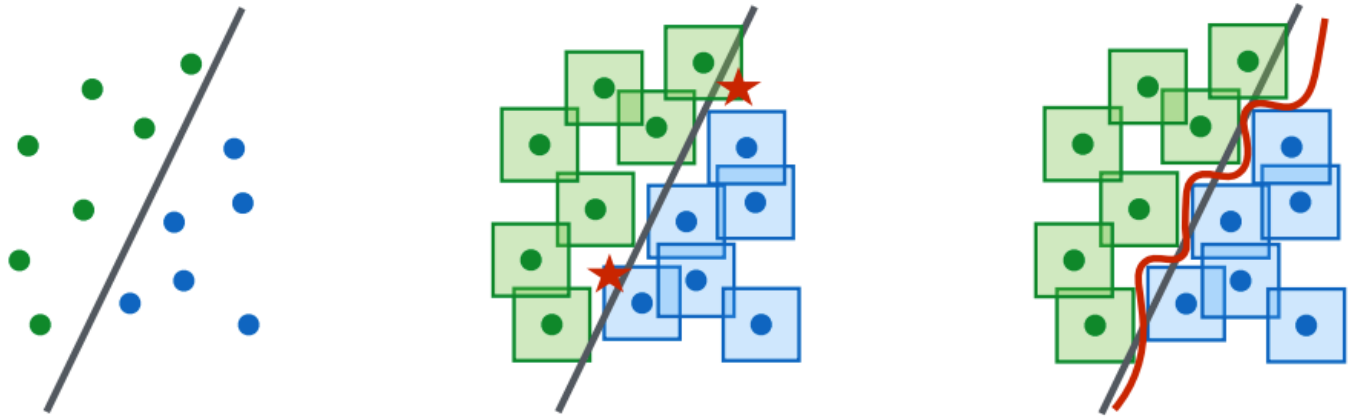


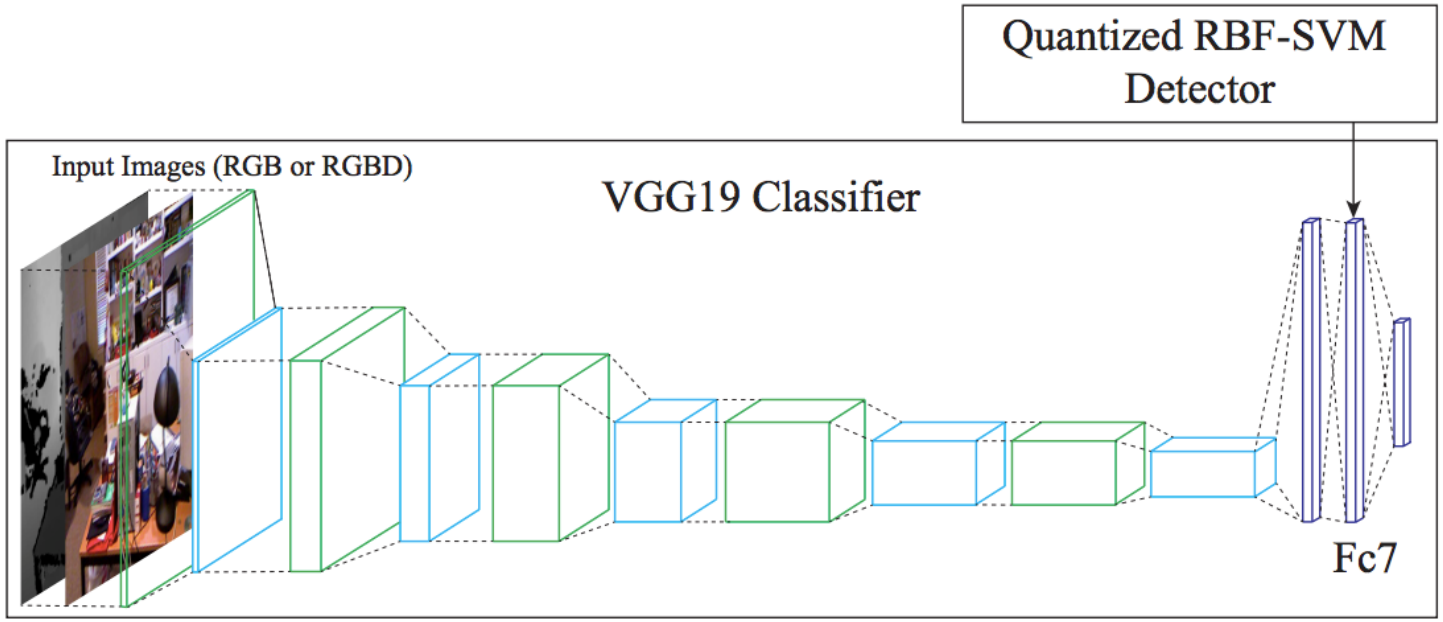
Figure 3: A conceptual illustration of standard vs. adversarial decision boundaries. Left: A set of points that can be easily separated with a simple (in this case, linear) decision boundary. Middle: The simple decision boundary does not separate the ℓ_∞-balls (here, squares) around the data points. Hence there are adversarial examples (the red stars) that will be misclassified. Right: Separating the ℓ_∞-balls requires a significantly more complicated decision boundary. The resulting classifier is robust to adversarial examples with bounded ℓ_∞-norm perturbations.

Defending against adversarial examples

- Adversarial training: networks can be made somewhat resistant by augmenting or regularizing training with adversarial examples (Goodfellow et al. 2015, Tramer et al. 2018)
- Adversarial objectives can also be formulated (Madry et al., 2018)
  - There is even evidence that such models may give better performance when transferred to different tasks (Salman et al., 2020)

Defending against adversarial examples

- Train a separate model to reject adversarial examples: SafetyNet



J. Lu, T. Issaranon, D. Forsyth, SafetyNet: Detecting and Rejecting Adversarial Examples Robustly, CVPR 2017

Defending against adversarial examples

- Robust architectures

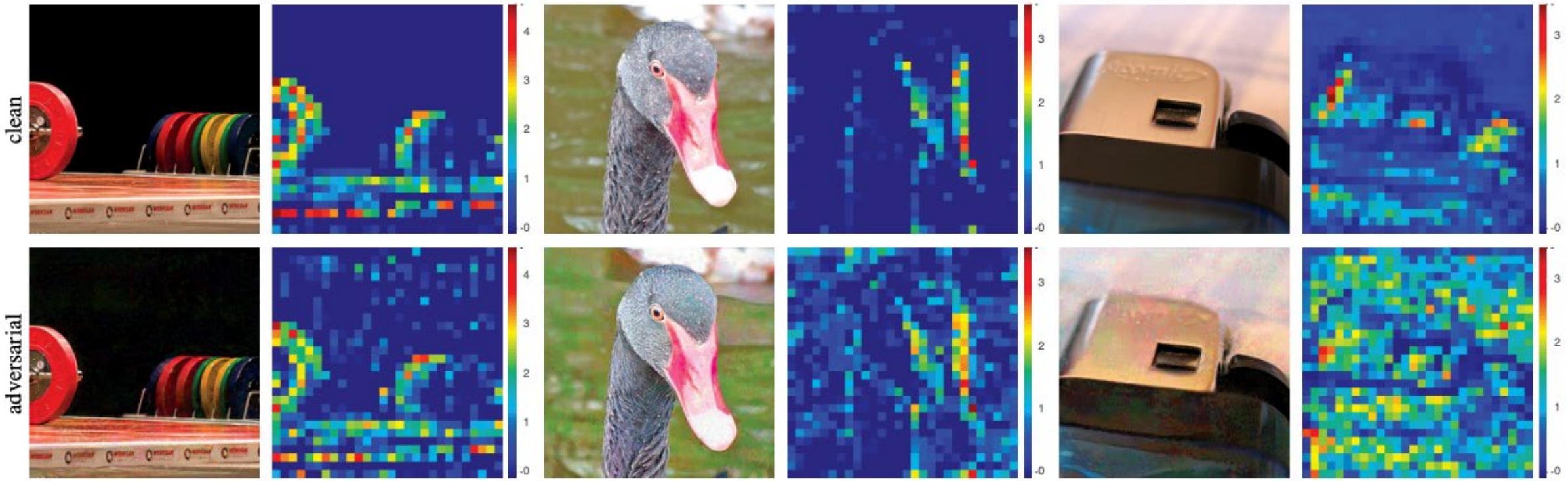
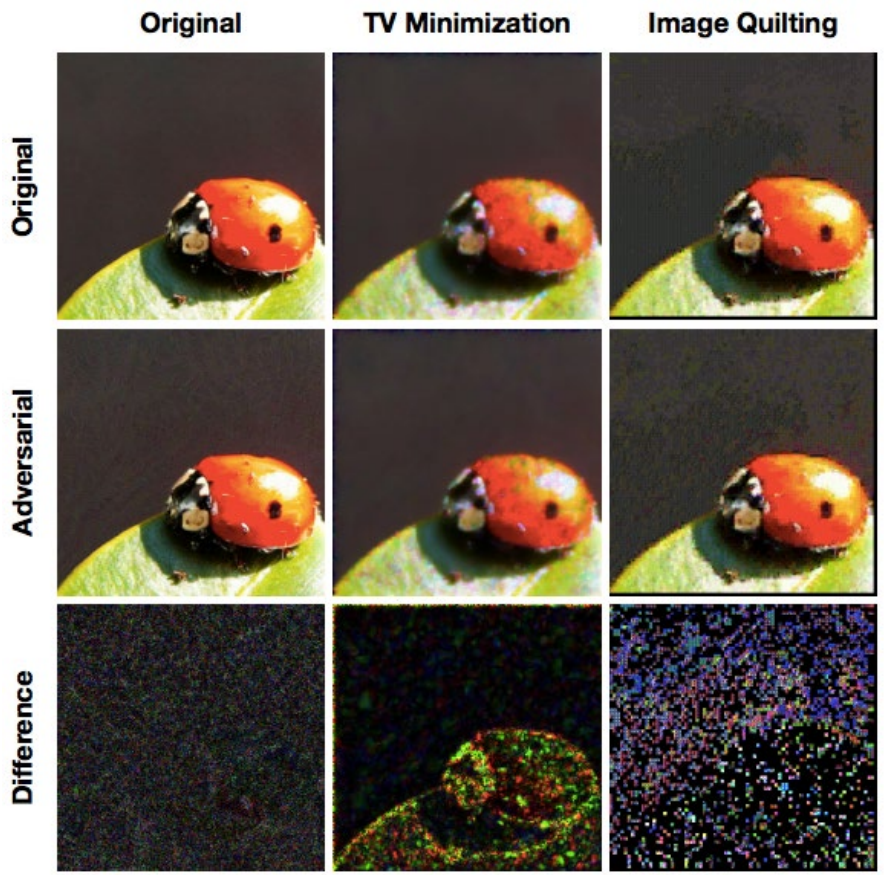


Figure 2. More examples similar to Figure 1. We show feature maps corresponding to clean images (top) and to their adversarial perturbed versions (bottom). The feature maps for each pair of examples are from the same channel of a res_3 block in the same ResNet-50 trained on clean images. The attacker has a maximum perturbation $\epsilon = 16$ in the pixel domain.

C. Xie et al., Feature Denoising for Improving Adversarial Robustness, CVPR 2018

Defending against adversarial examples

- Pre-process input images to disrupt adversarial perturbations



C. Guo, M. Rana, M. Cisse, L. van der Maaten, Countering Adversarial Images Using Input Transformations, ICLR 2018

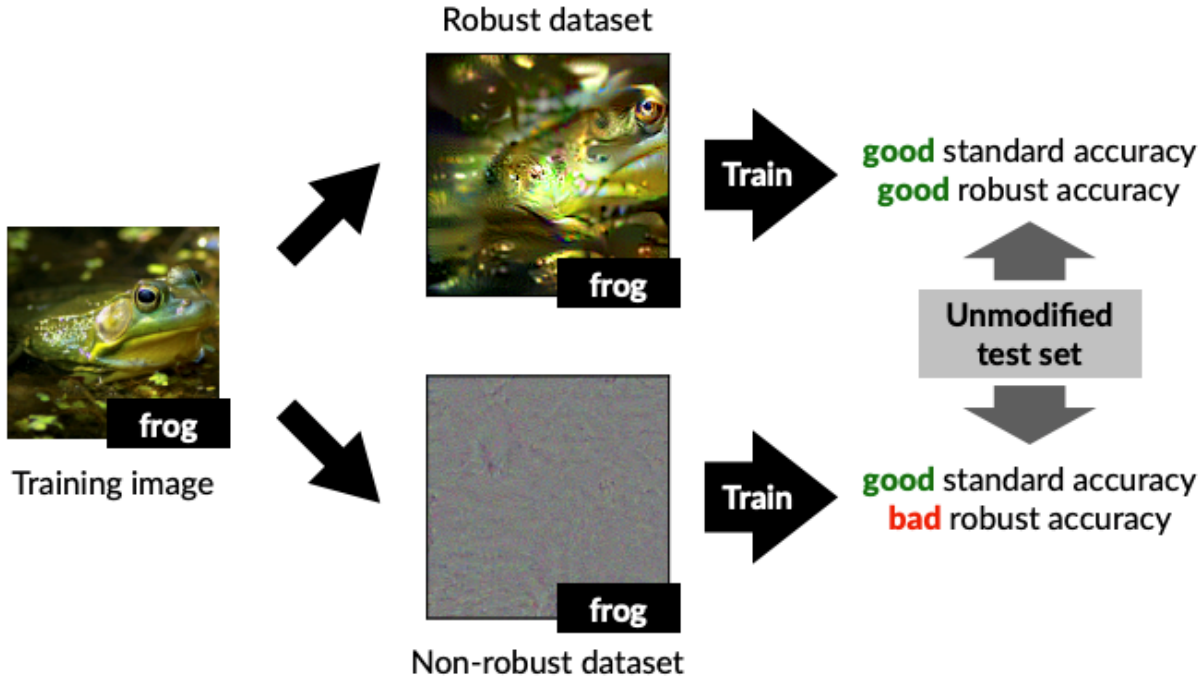
Properties of adversarial examples: Summary

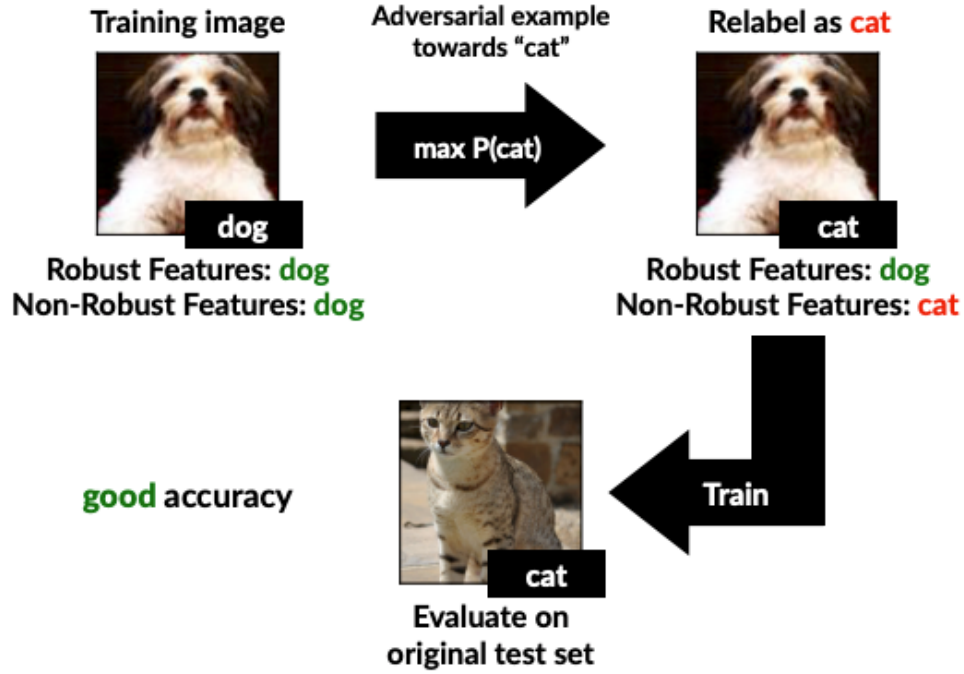
- For any input image, it is usually easy to generate a very similar image that gets misclassified by the same network
- To obtain an adversarial example, one does not need to do precise gradient ascent
- Adversarial images can (somewhat) survive transformations like being printed and photographed
- It is possible to attack many images with the same perturbation
- Adversarial examples that can fool one network have a high chance of fooling a network with different parameters and even architecture

Why are deep networks easy to fool?

- Networks are “too linear”: it is easy to manipulate output in a predictable way given the input
- The input dimensionality is high, so one can get a large change in the output by changing individual inputs by small amounts
- Neural networks can fit anything, but nothing prevents them from behaving erratically between training samples
  - Counter-intuitively, a network can both generalize well on natural images and be susceptible to adversarial examples
- Adversarial examples generalize well because different models learn similar functions when trained to perform the same task (or because adversarial examples are a function of the data rather than of the network)?

Adversarial examples are not bugs, but features?





Disentangle features into robust and non-robust

Construct a dataset which appears mislabeled to human (via adversarial examples) but results in good accuracy on the original test set

A. Ilyas et al. Adversarial Examples are not Bugs, they are Features. NeurIPS 2019

Adversarial examples and humans

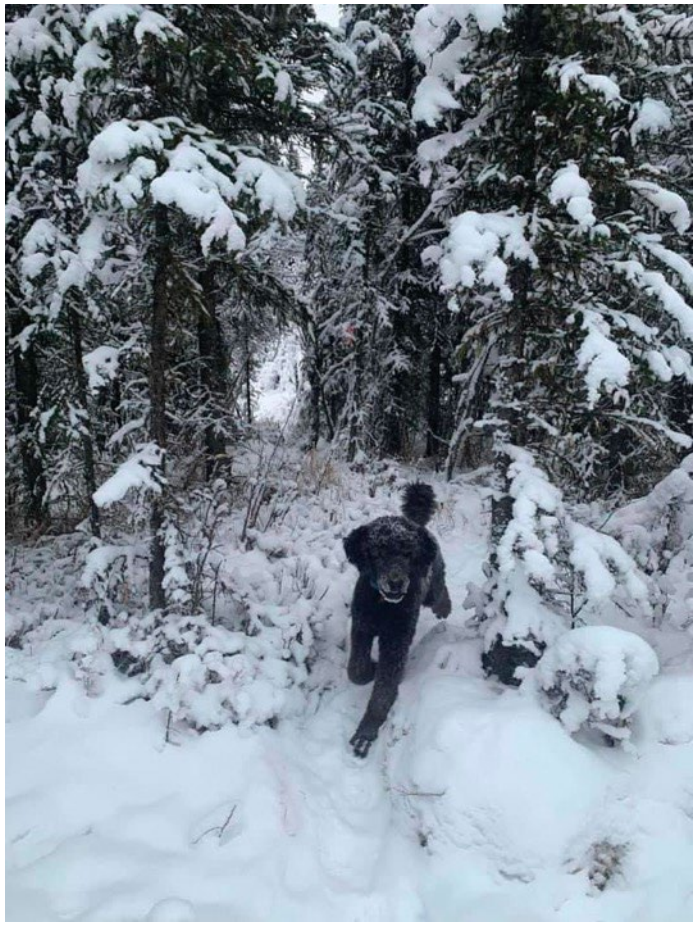
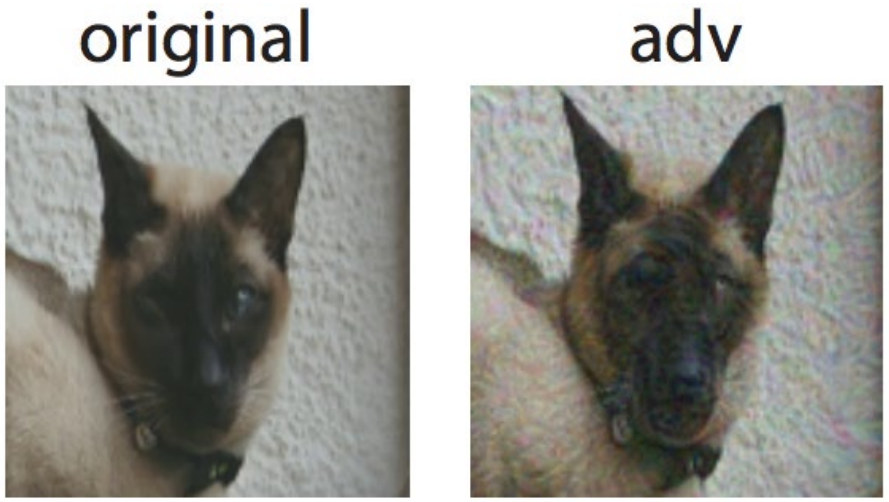


Image source

Adversarial examples and humans

Adversarial examples and humans

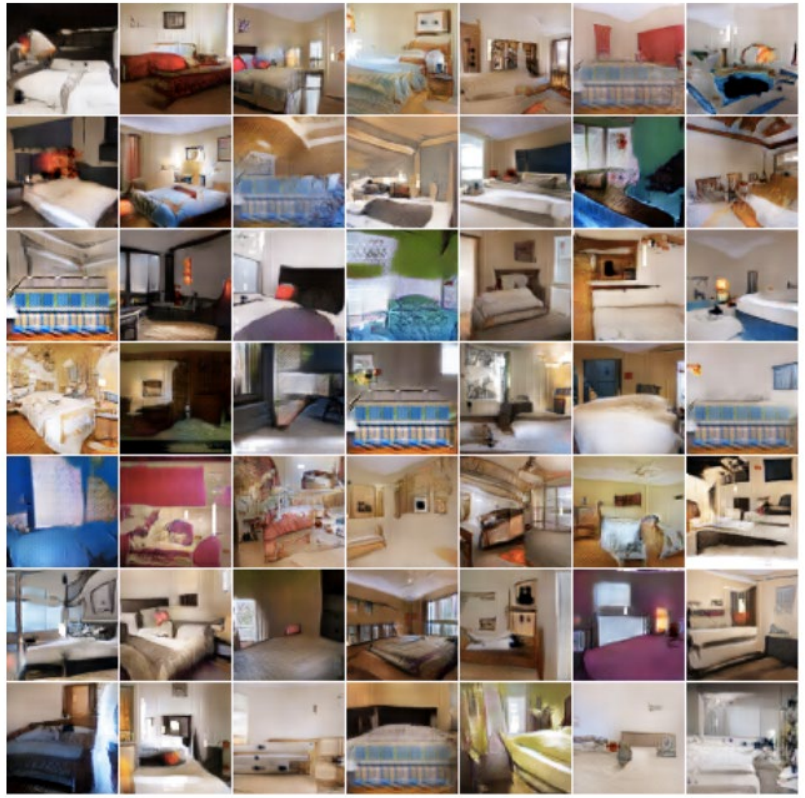
- Adversarial examples that are designed to transfer across multiple architectures can also be shown to confuse the human visual system in rapid presentation settings



G. Elsayed et al., Adversarial Examples that Fool both Computer Vision and Time-Limited Humans, NeurIPS 2018

Generative tasks

- Generation: learn to sample from the distribution represented by the training set
  - *Unsupervised learning* task



Generative tasks

- Generation conditioned on class label

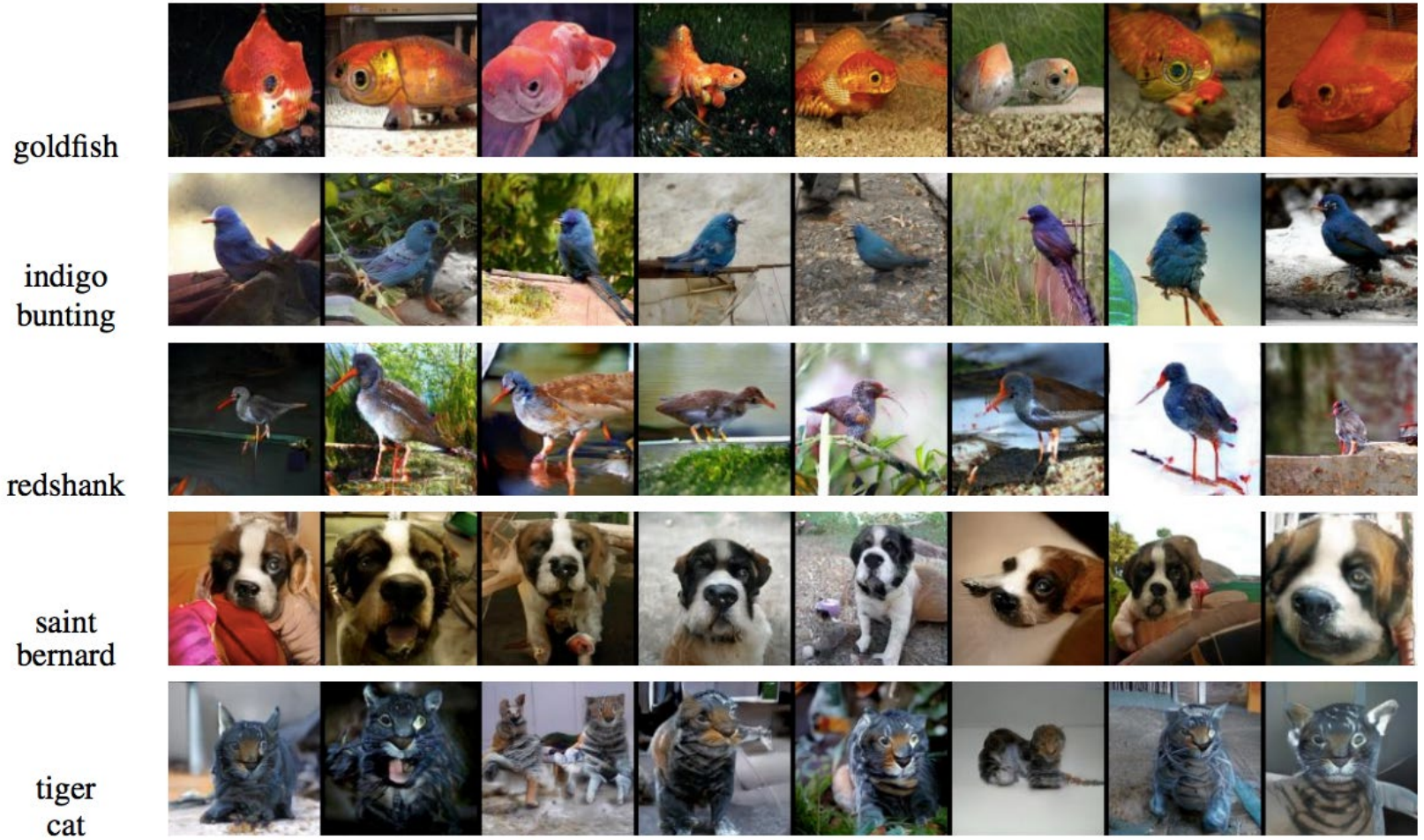
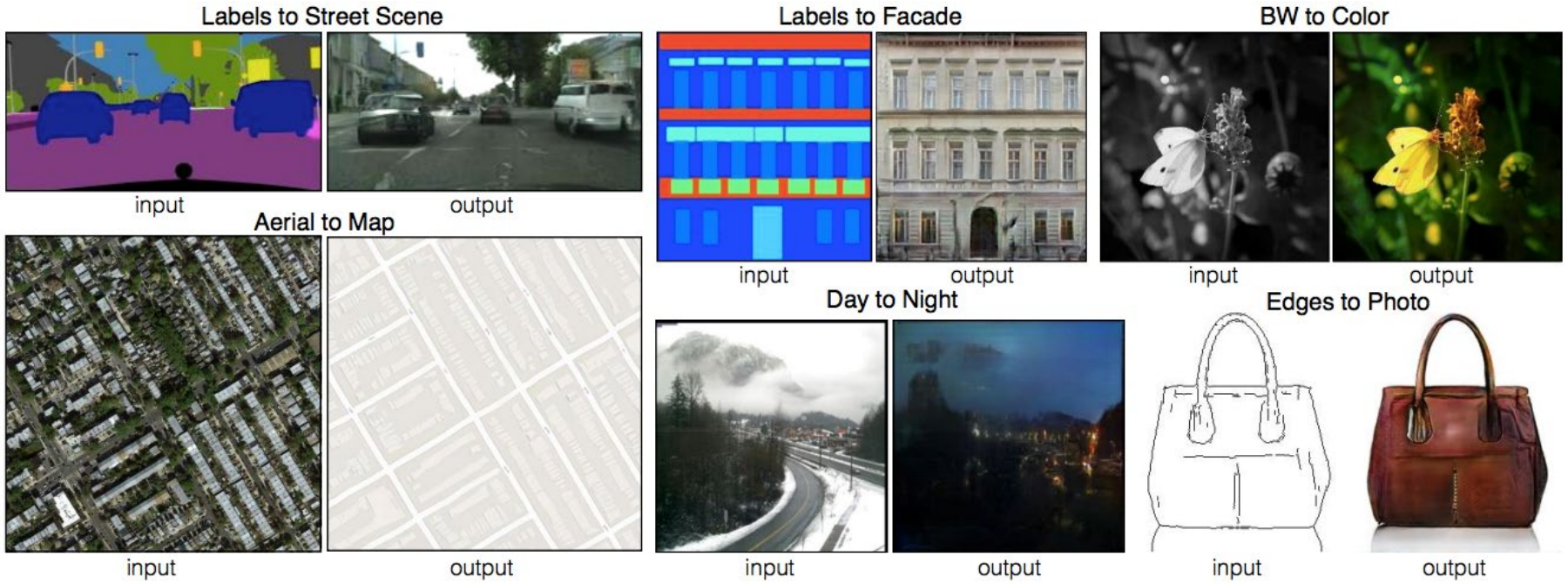


Figure source

Generative tasks

- Generation conditioned on image (image-to-image translation)



P. Isola, J.-Y. Zhu, T. Zhou, A. Efros, Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017

Generative adversarial networks (GANs)

Train two networks with opposing objectives:

- **Generator:** learns to generate samples
- **Discriminator:** learns to distinguish between generated and real samples

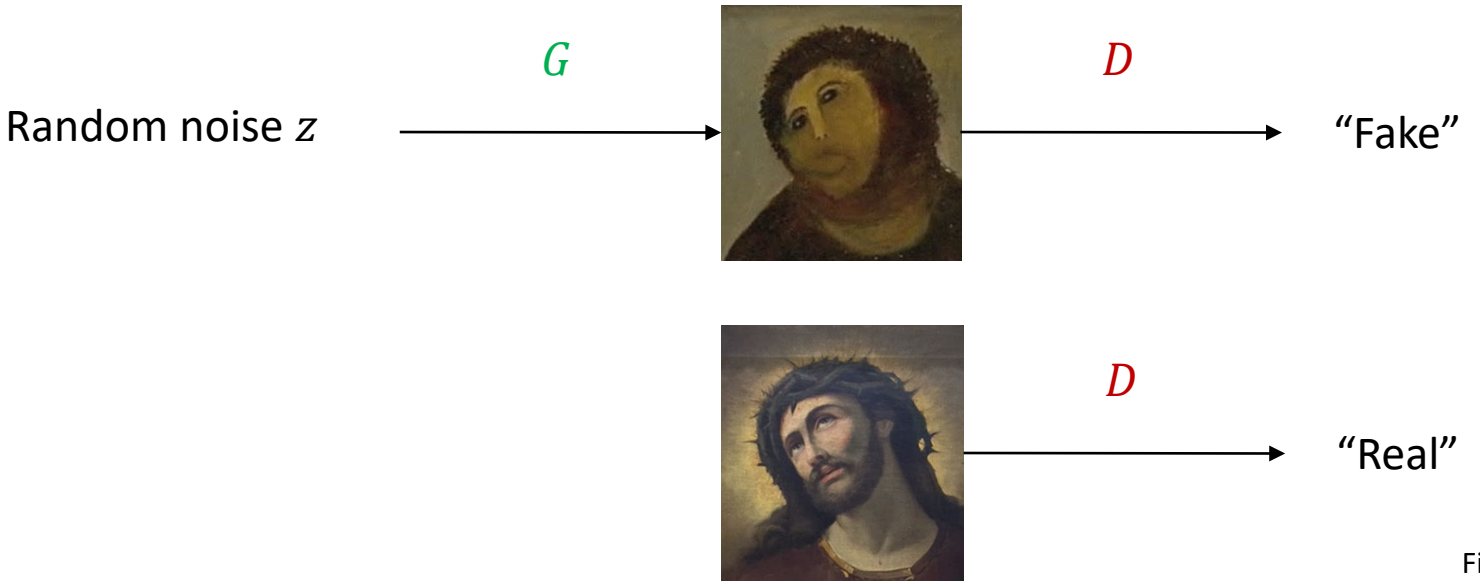
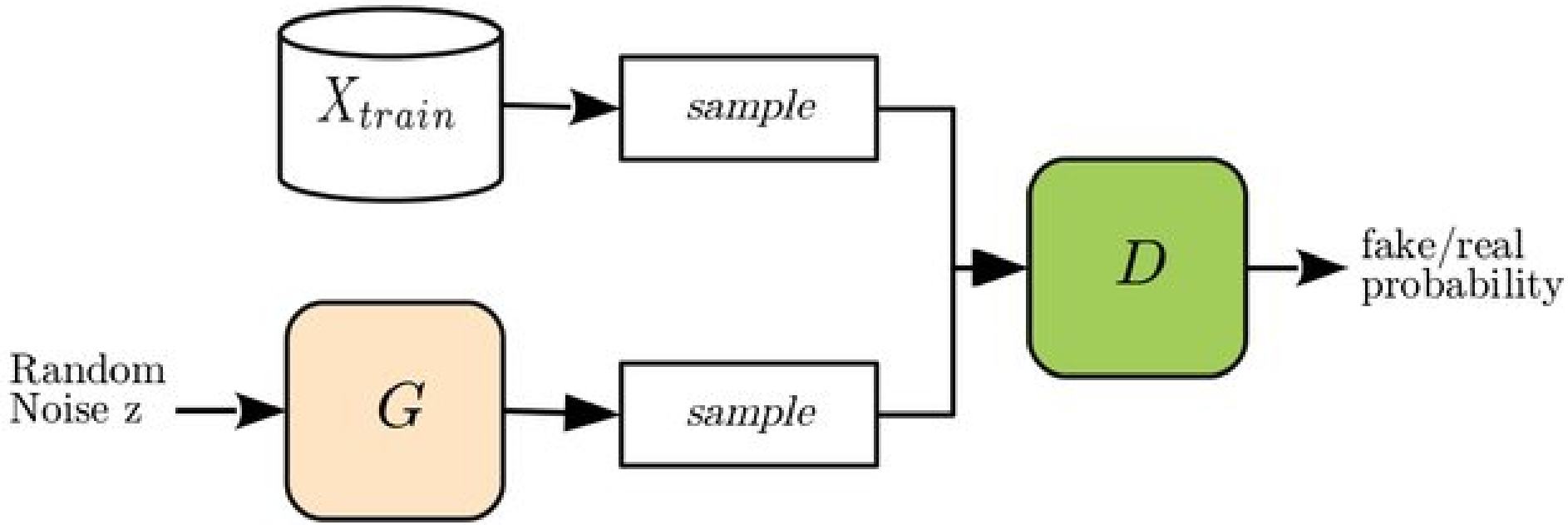


Figure adapted from F. Fleuret

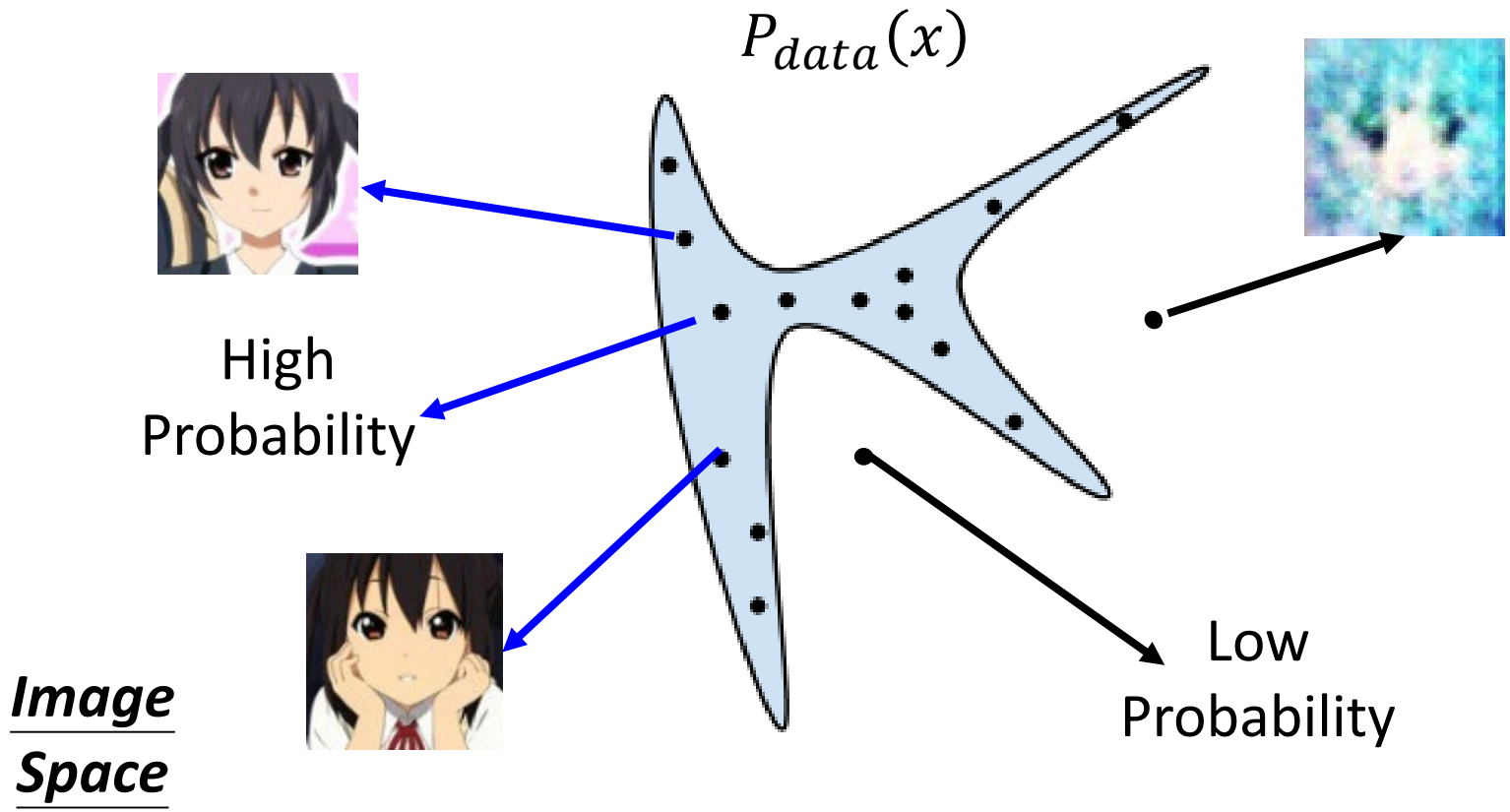
I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, NIPS 2014

Generative adversarial networks (GANs)



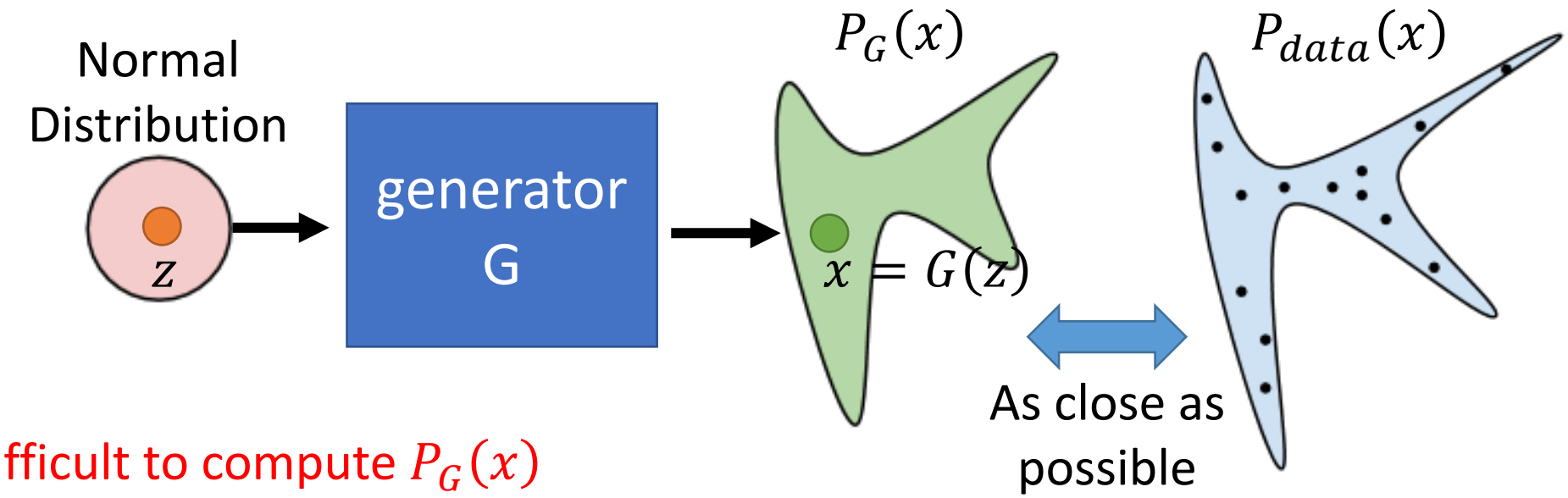
Basic Idea of GAN

- The data we want to generate has a distribution $P_{data}(x)$



Basic Idea of GAN

- A generator G is a network. The network defines a probability distribution.



It is difficult to compute $P_G(x)$

We do not know what the distribution looks like.

https://blog.openai.com/generative-models/

GAN objective

The discriminator $D(x)$ should output the probability that the sample x is real

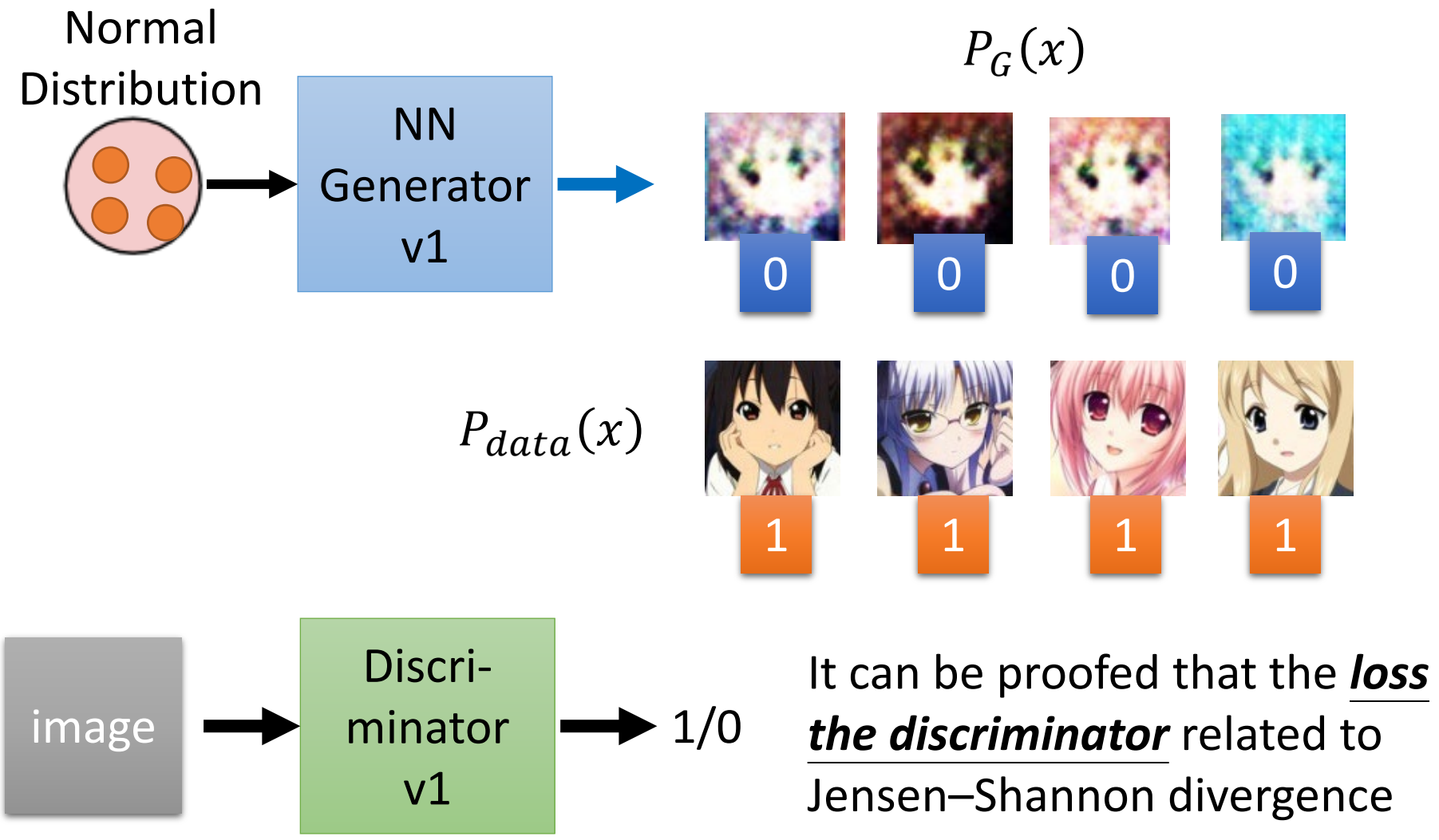
- That is, we want $D(x)$ to be close to 1 for real data and close to 0 for fake

Expected conditional log likelihood for real and generated data:

- $\mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{x \sim p_{\text{gen}}} \log\big(1 - D(x)\big)$
- $= \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z)))$

We seed the generator with noise z drawn from a simple distribution p (Gaussian or uniform)

Basic Idea of GAN



It can be proofed that the ***loss the discriminator*** related to Jensen–Shannon divergence

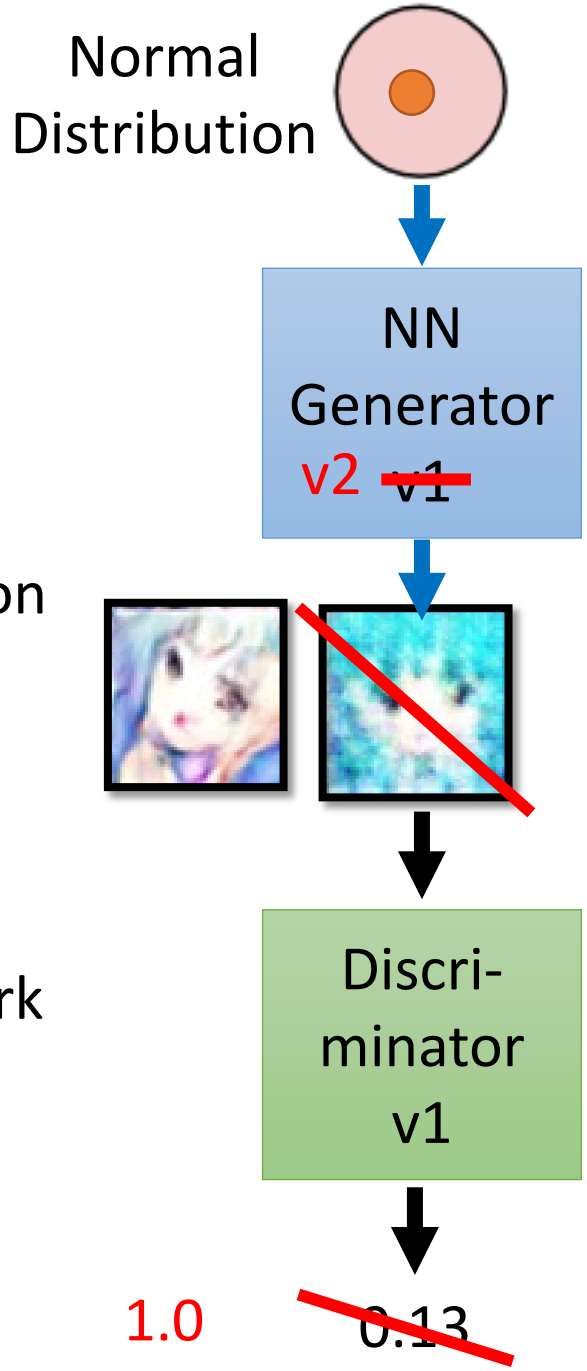
Basic Idea of GAN

- ***Next step***:
  - Updating the parameters of generator
  - To minimize the Jensen–Shannon divergence

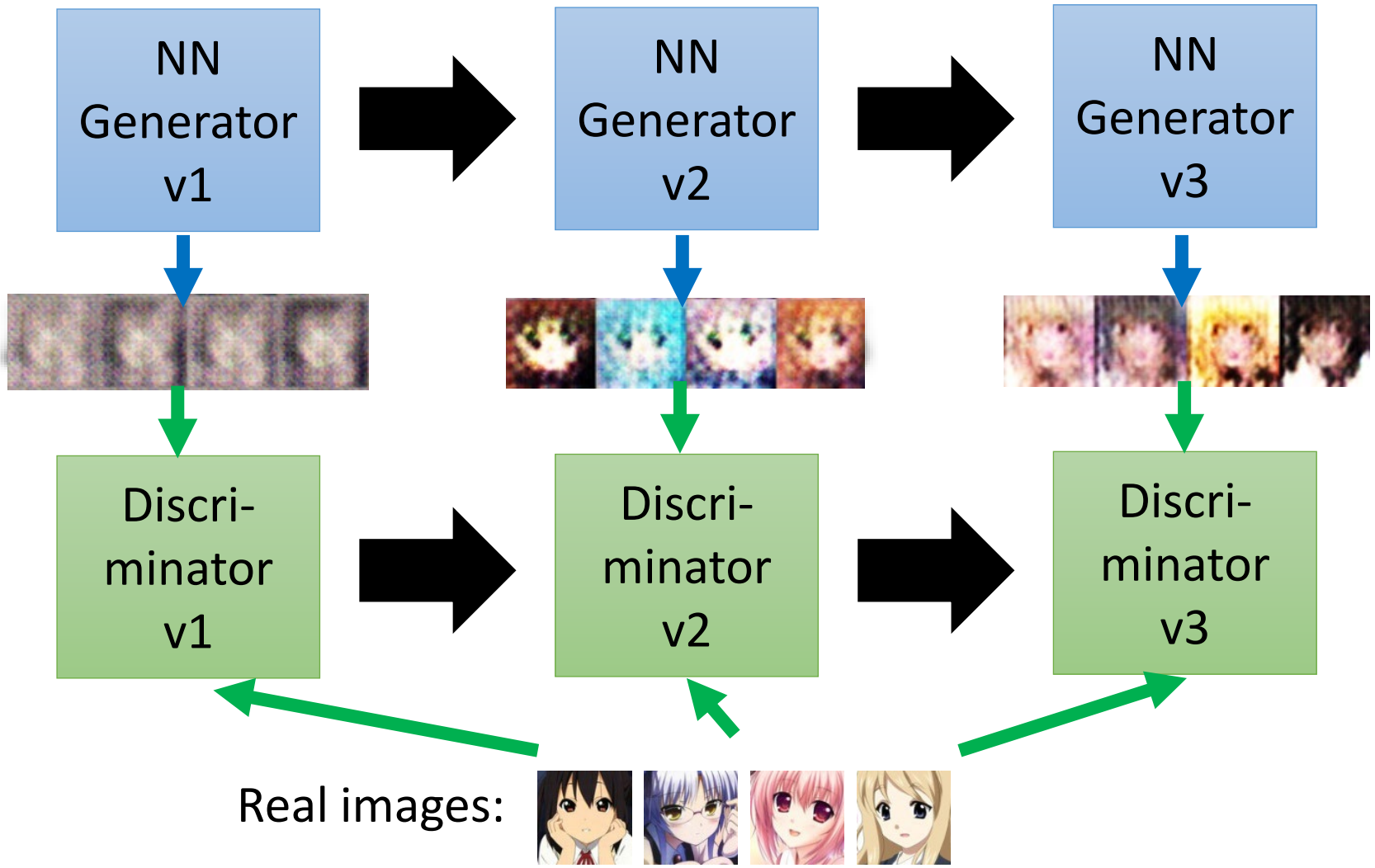
➡ The output be classified as "real" (as close to 1 as possible)

Generator + Discriminator = a network

Using gradient descent to update the parameters in the generator, but fix the discriminator

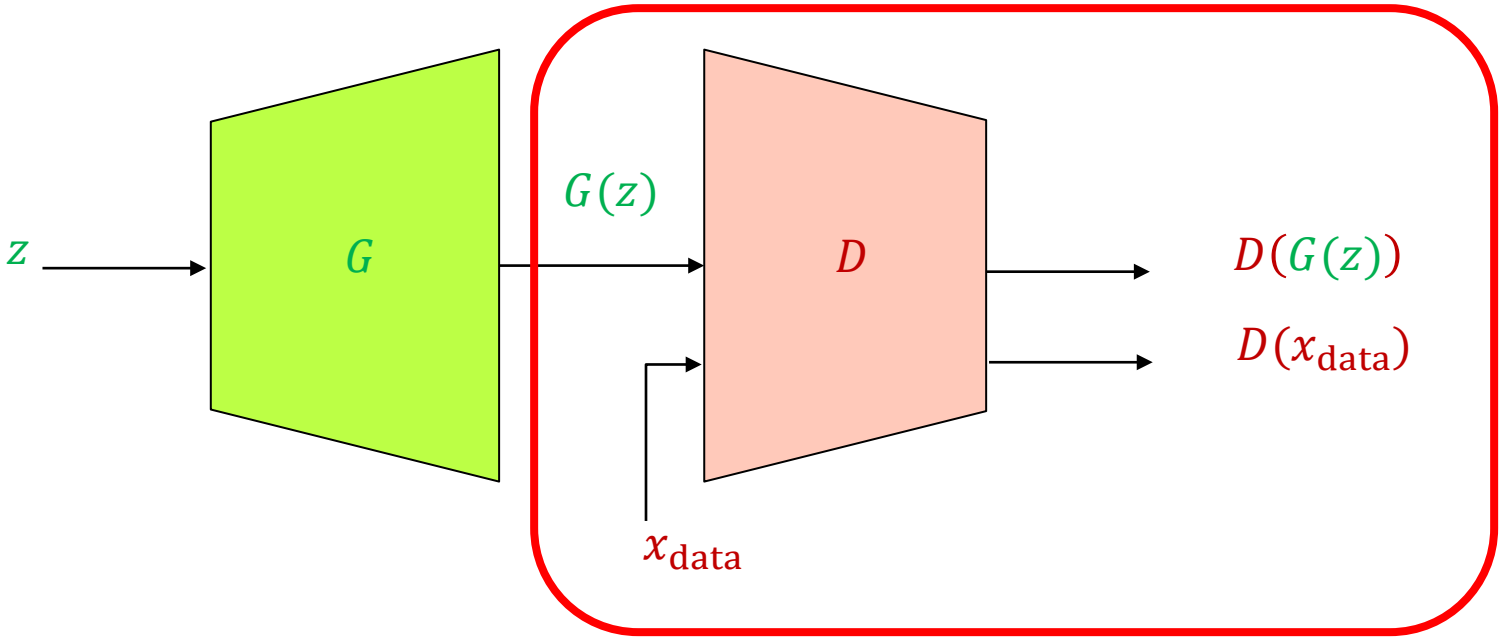


The evolution of generation



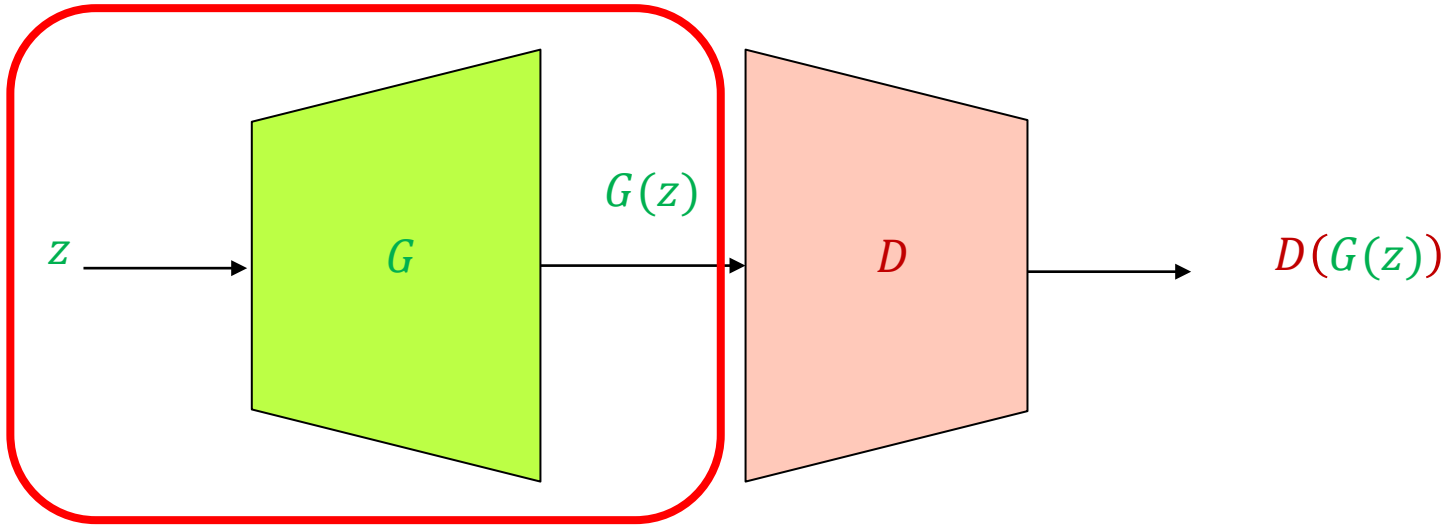
GAN: Conceptual picture

- Update discriminator: push $D(x_{\text{data}})$ close to 1 and $D(G(z))$ close to 0
  - The generator is a “black box” to the discriminator



GAN: Conceptual picture

- Update generator: increase $D(G(z))$
  - Requires back-propagating through the composed generator-discriminator network (i.e., the discriminator cannot be a black box)
  - The generator is exposed to real data only via the output of the discriminator (and its gradients)



GAN: Conceptual picture

- Test time – the discriminator is discarded