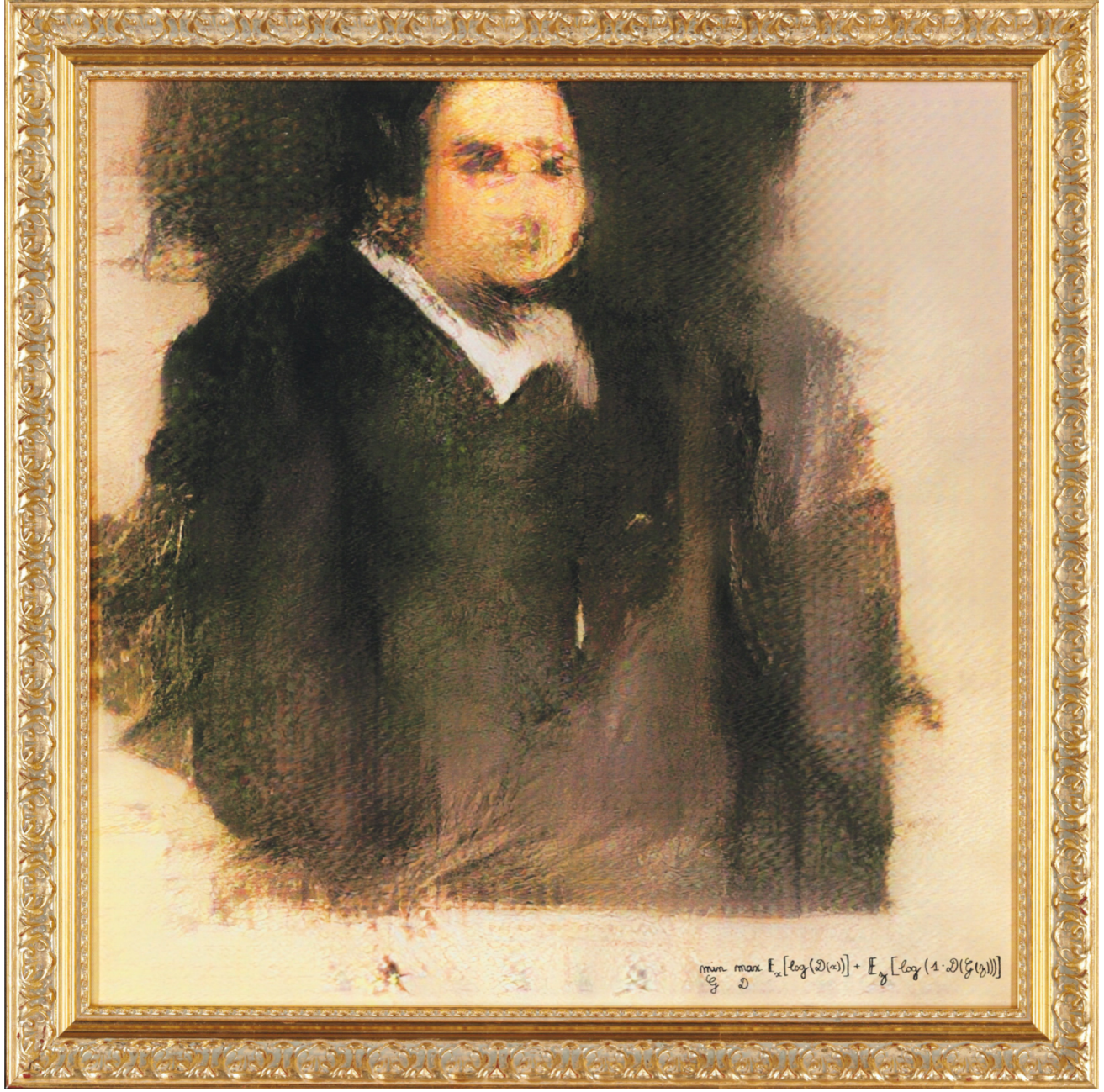
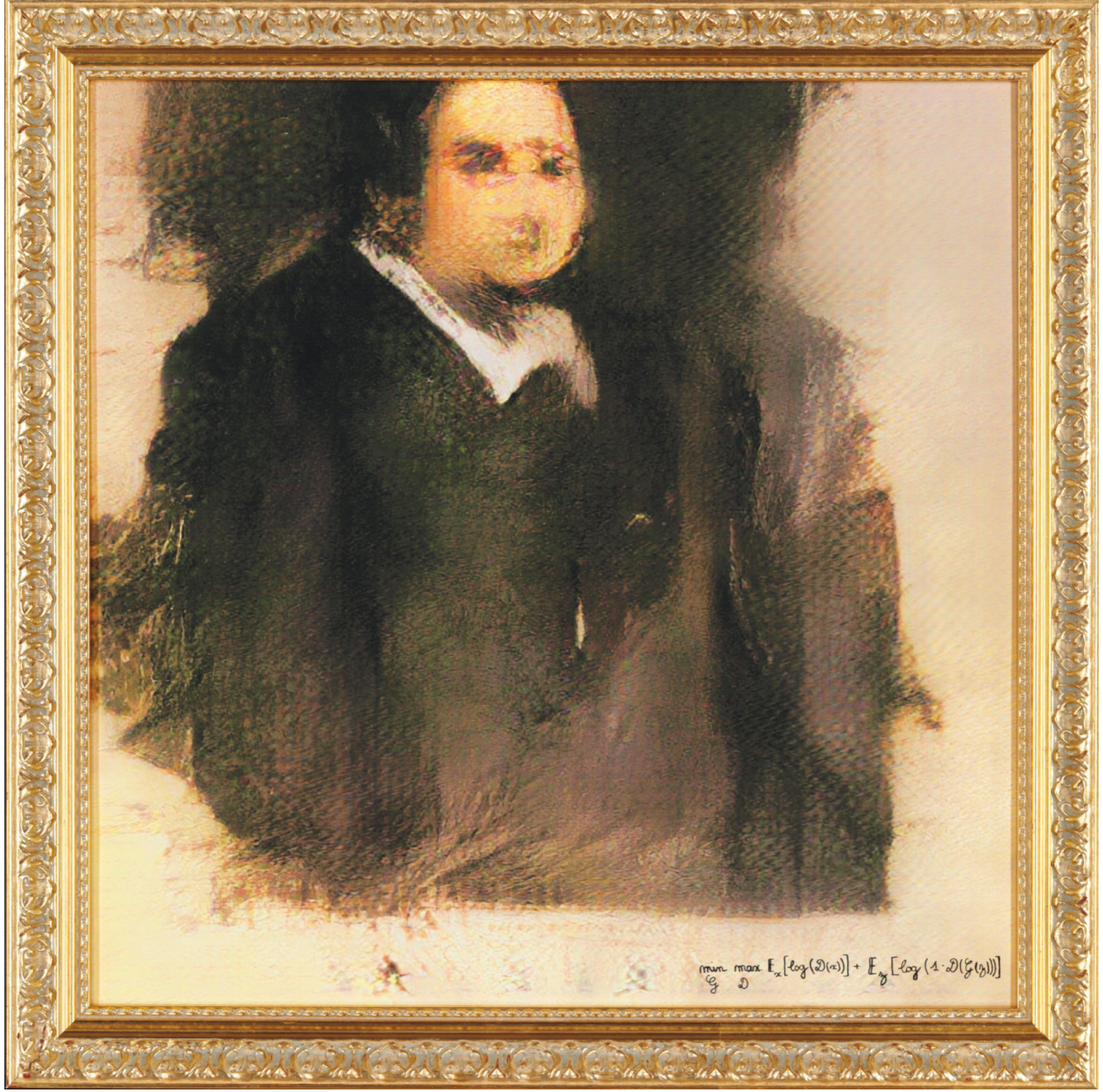
Generative Adversarial Networks for Image Data Augmentation

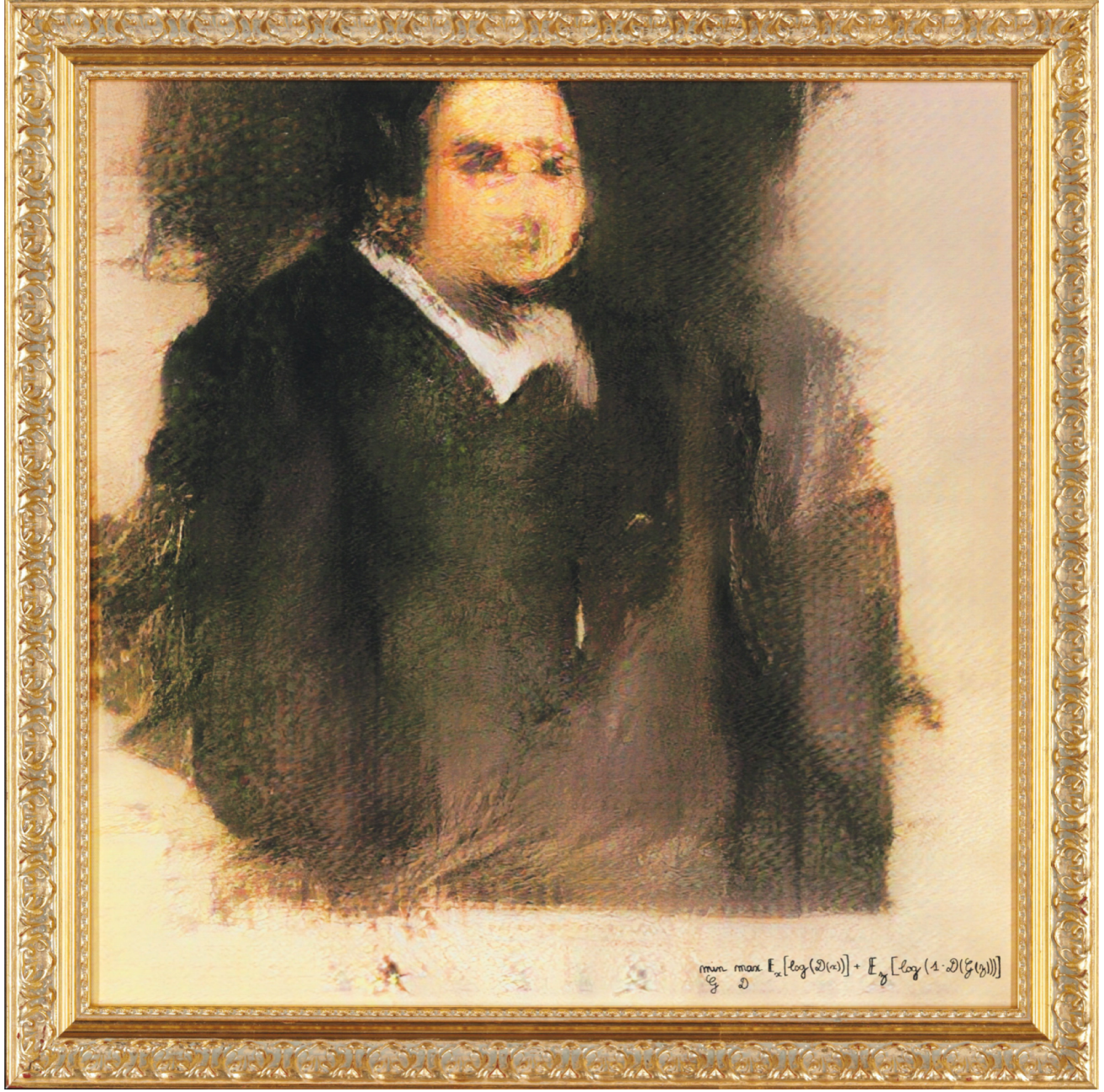
K. Horak, TU Wien, 2019



Portrait of Edmond Belamy. Sold for $432,500 on 25 October 2018 at Christie's in New York.



French portly judge, gilt frame, unfinished work (indistinct facial features, blank areas), composition is displaced to the north-west, style estimated between the 14th century to the 19th.



$432,500 – nearly 45 times its high estimate. What is so worthy on it?

Generative Adversarial Networks – Prerequisites

Linear algebra & statistics: vector/tensor calculus, eigenvalues, decomposition, PCA/ICA, joint and conditional probs.

Neural networks basics: neuron model, linear neuron, non-linear neuron (sigmoid, tanh, ReLU, Leaky ReLU), softmax output.

Feed-forward training: gradient descent, delta rule, learning rate, backpropagation alg., overfitting, regularization.

Convnet: convolutions, filters, feature (activation) maps, pooling, FC layer, well-known CNNs architectures (AlexNet, VGG).

Generative Adversarial Networks – Roadmap

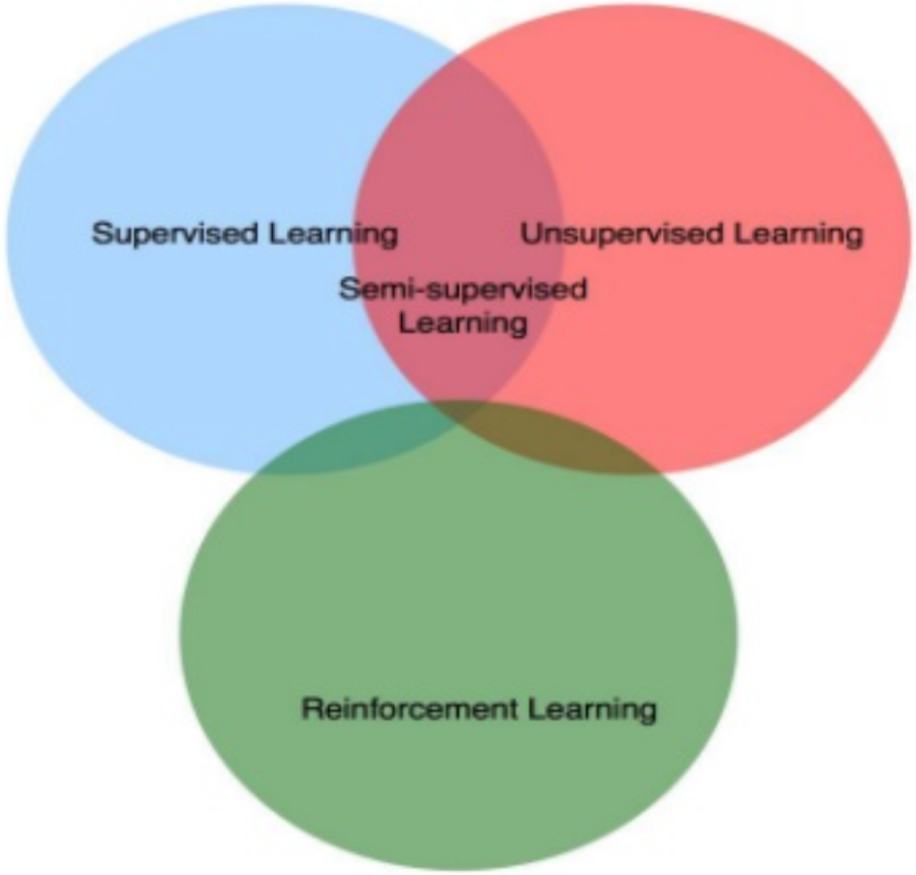
1. Machine Learning – Short Revision
2. Limited Dataset – Bootstrap & Cross-Validation
3. Generative Adversarial Networks (GANs) – General Framework
4. Deep Convolutional GANs – Architecture
5. DCGANs Examples
6. Who is Edmond Belamy?

Generative Adversarial Networks – Roadmap

1. Machine Learning – Short Revision
2. Limited Dataset – Bootstrap & Cross-Validation
3. Generative Adversarial Networks (GANs) – General Framework
4. Deep Convolutional GANs – Architecture
5. DCGANs Examples
6. Who is Edmond Belamy?

Machine Learning – Short Revision

Taxonomy of machine learning:



Machine Learning – Short Revision

Supervised learning in statistical classification:

- Find deterministic function ***f*** : $y = f(x)$, x : data, y : **label**

Supervised learning in statistical classification:

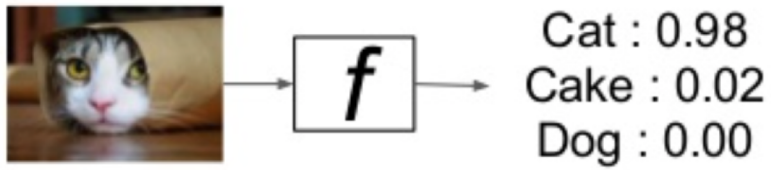
- Challenges
  - Image is **high dimensional** data
    - 224 x 224 x 3 = 150,528
  - Many **variations**
    - viewpoint, illumination, deformation, occlusion, background clutter, intraclass variation

Supervised learning in statistical classification:

- Solution
  - **Feature** Vector
    - 150,528 → 2,048
  - Synonyms
    - Latent Vector, Hidden Vector, Unobservable Vector, Feature, Representation

Supervised learning in statistical classification:

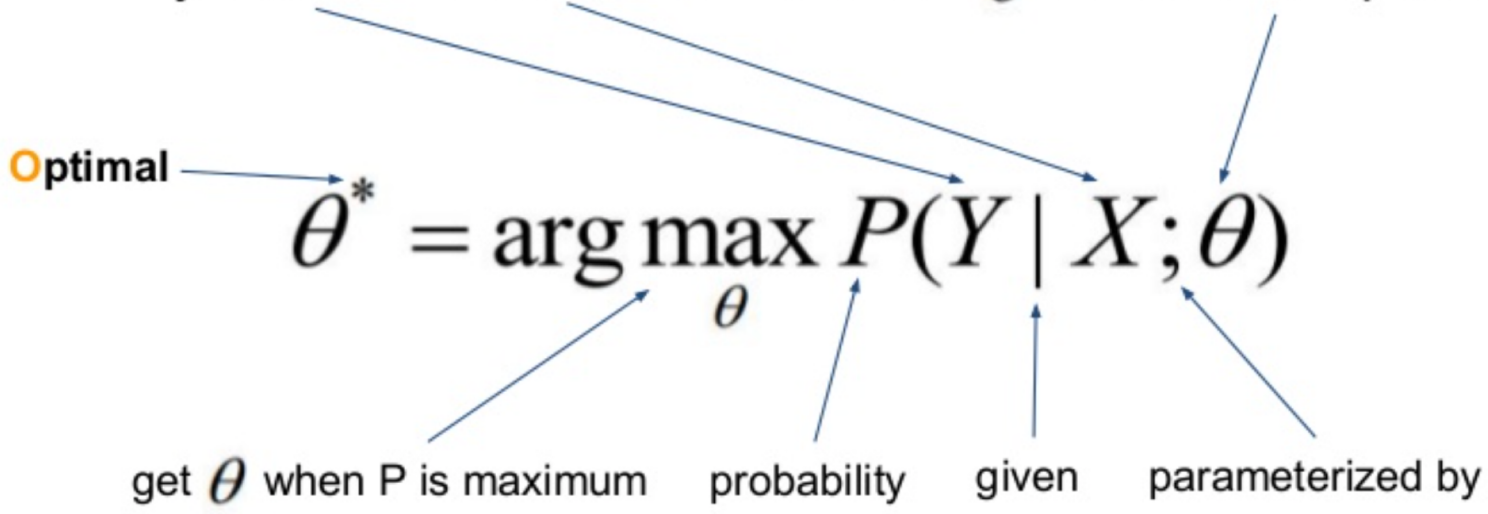
- More flexible solution
  - Get probability of the label for given data instead of label itself



Note: it is not about major class selection in the last layer!

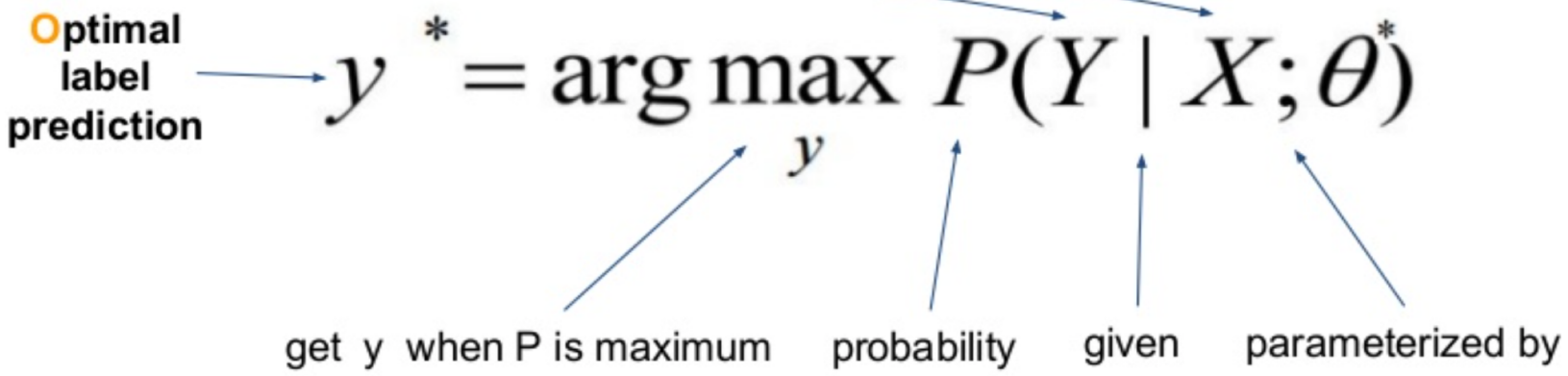
Supervised learning in statistical classification:

- Mathematical notation of **optimization**
  - y : label, x : data, z : latent, θ : learnable parameter



Supervised learning in statistical classification:

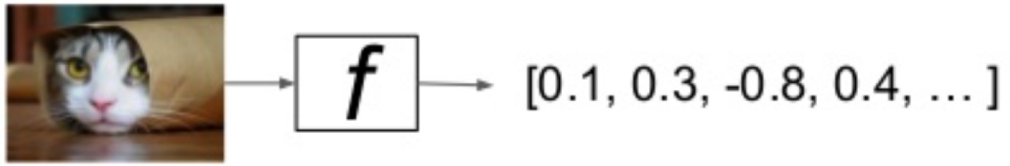
- Mathematical notation of **classifying** (greedy policy)
  - y : label, x : data, z : latent, θ^*: fixed optimal parameter



Unsupervised learning:

- latent structure analysis, i.e. to find hidden patterns

- Find deterministic function ***f*** : $z = f(x)$, x : data, z : **latent**



Machine Learning – Short Revision

Unsupervised learning:

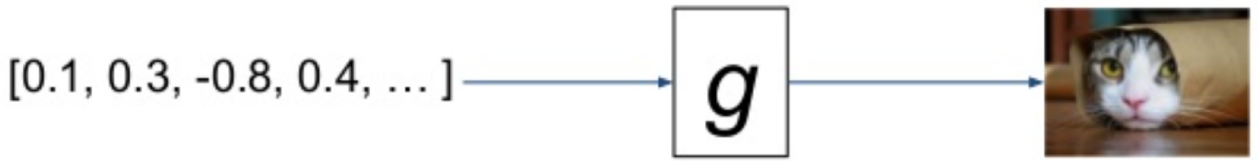
- latent structure analysis, i.e. to find hidden patterns

  - More challenging than supervised learning :
    - No label or curriculum → self learning

  - Some NN solutions :
    - Boltzmann machine
    - Auto-encoder or Variational Inference
    - Generative Adversarial Network

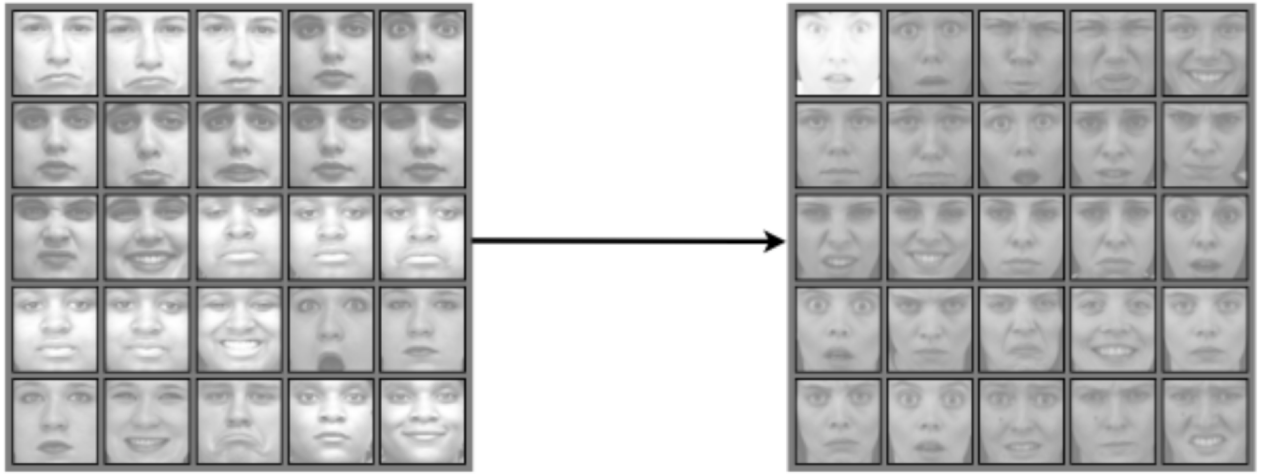
Generative model (joint distribution of observed and target variables):

- Find generation function ***g*** : $x = g(z)$, x : data, z : **latent**



Generative model (joint distribution of observed and target variables):

- Have training examples $x \sim p_{data}(x)$
- Want a model that can draw samples: $x \sim p_{model}(x)$
- Where $p_{data}(x) \approx p_{model}(x)$



$x \sim p_{data}(x)$ $x \sim p_{model}(x)$

Generative model (joint distribution of observed and target variables):

Unsupervised learning vs. Generative model

- ***z = f(x)*** *vs.* ***x = g(z)***
- ***P(z\|x)*** *vs.* ***P(x\|z)***
- ***Encoder*** *vs.* ***Decoder (Generator)***
  - P(x, z) needed. (cf : P(y|x) in supervised learning)
    - P(z|x) = P(x, z) / P(x) → P(x) is intractable (ELBO)
    - P(x|z) = P(x, z) / P(z) → P(z) is given. (prior)

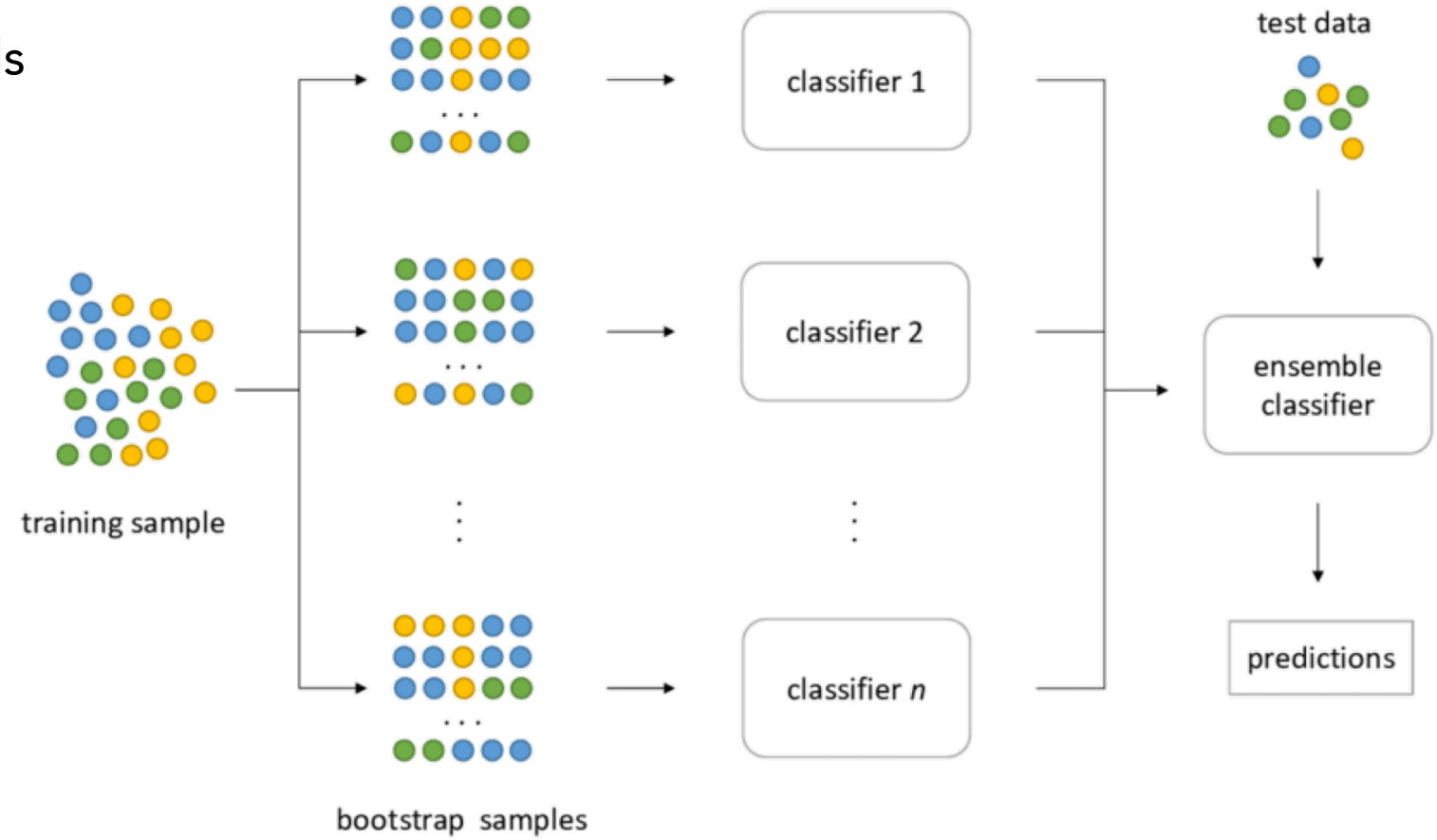
Generative Adversarial Networks – Roadmap

1. Machine Learning – Short Revision
2. Limited Dataset – Bootstrap & Cross-Validation
3. Generative Adversarial Networks (GANs) – General Framework
4. Deep Convolutional GANs – Architecture
5. DCGANs Examples
6. Who is Edmond Belamy?

Limited Dataset – Bootstrap

Bootstrap – if an input dataset of *N* samples is too small:

- Bootstrap generates *n* new training datasets B_i, so that $|B_i| = N, i \in \langle 1, n \rangle$
- Random combinations with repetition
- *n* recommended tens to thousands



Limited Dataset – Bootstrap

Bootstrap – if an input dataset of *N* samples is too small:

- Estimated error $\hat{E}rr_{boot}$ of a model is given by sum of loss functions related to the B_i

$$\hat{Err}_{boot} = \frac{1}{N}\frac{1}{|B|}\sum_{j=1}^{|B|}\sum_{i=1}^{N} LF\left(y_i, \tilde{f}^{B_j}(x_i)\right)$$

- where $\tilde{f}^{B_j}$ is the model adapted (trained on) to dataset B_j

Note: the so-called 0,632-Bootstrap is a Bootstrap version using unselected samples from *N* in a test set – estimated model error is then:

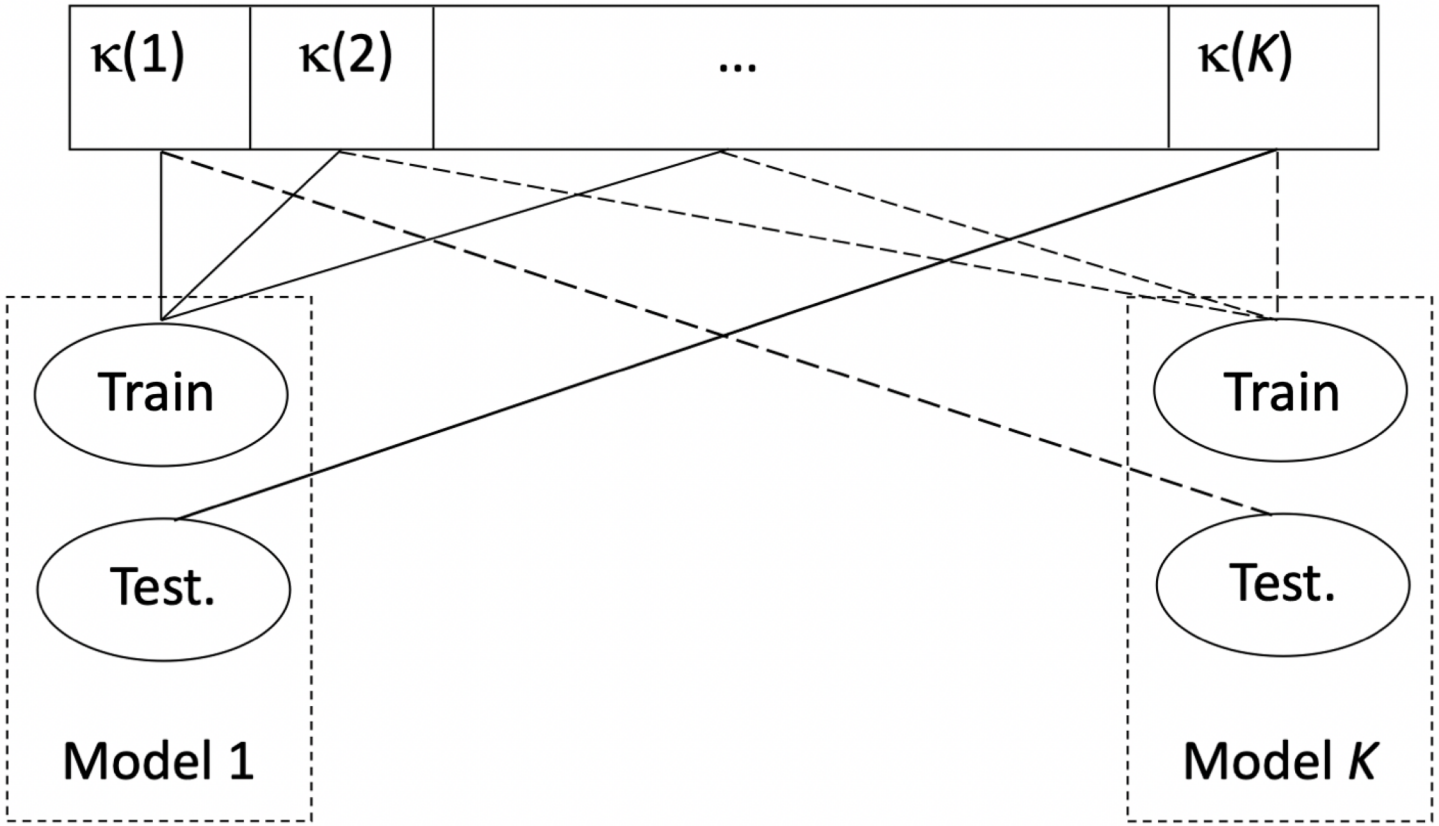
$$Err = 0{,}632 \,.\, Err_{Btest} + 0{,}368 \,.\, Err_{boot}$$

$$\hat{Err}_{Btest} = \frac{1}{|B|}\sum_{j=1}^{|B|}\frac{1}{|C_j|}\sum_{i=1}^{|C_j|} LF\left(y_i, \tilde{f}^{B_j}\left(x_{C_{ji}}\right)\right)$$

Limited Dataset – Cross-Validation

Cross-Validation – if an input dataset of *N* samples is small:

- Original dataset is divided into the *K* new disjunct subsets of the same volumes
- *K* models are trained separately ⇒ *K* errors $Err_{CV}{}^{i}, i \in \langle 1, K \rangle$ are computed



Limited Dataset – Cross-Validation

Cross-Validation – if an input dataset of *N* samples is small:

- Estimated overall error $\hat{E}rr_{CV}$ of the *K* models is given by:

$$Err_{CV} = \frac{1}{K}\sum_{i=1}^{K} Err\left(y_{\kappa(i)}, \tilde{f}^{-\kappa(i)}\left(x_{\kappa(i)}\right)\right)$$

- where $x_{\kappa(i)}$ and $y_{\kappa(i)}$ are input and output data of i^{th} subset $\kappa(i)$ respectively, and $\tilde{f}^{-\kappa(i)}$ represents model trained on a dataset not containing the subset $\kappa(i)$

- Typical design is so-called Tenfold Cross Validation, i.e. *K = 10*

- Leave-one-out: special case when *K=N* ⇒ each model is trained on a subset of *N-1* samples and tested on a remaining one

Limited Dataset – Bootstrap & Cross-Validation

To sum up:

- Bootstrap underestimates real model's error (re-substitution error)
- Cross-Validation overestimates real model's error
- Mainly, neither Bootstrap nor Cross-Validation create new content!

Generative Adversarial Networks – Today's Roadmap

1. Machine Learning – Short Revision
2. Limited Dataset – Bootstrap & Cross-Validation
3. Generative Adversarial Networks (GANs) – General Framework
4. Deep Convolutional GANs – Architecture
5. DCGANs Examples
6. Who is Edmond Belamy?

GANs – Mission

Can you guess what is common among all the faces in this image?

GANs – Mission

Can you guess what is common among all the faces in this image?

None of these people are real!

All these faces were generated by a computer vision technique called Generative Adversarial Networks (GANs).

GANs – Discriminative vs. Generative Approach

Discriminative approach:

- all well-known classifiers or estimators (regression) discriminates an input data to struct. output
- {OK, NOK} (dichotomy), or {class 0, class 1, class 2,…} (multiclass), or $v_{out} \in < 0; \infty)$ (regression)

Generative approach:

- has ability to create new content based on dataset distributions (joint probability distribution)
- GANs create such content through a sort of internal competition between two deep networks
- Deep Convolutional GAN learns to create image that resemble images in training dataset
- the first network is called generator, the other one discriminator – they play minimax game* against each other until the Nash equilibrium reached (or training collapsed)

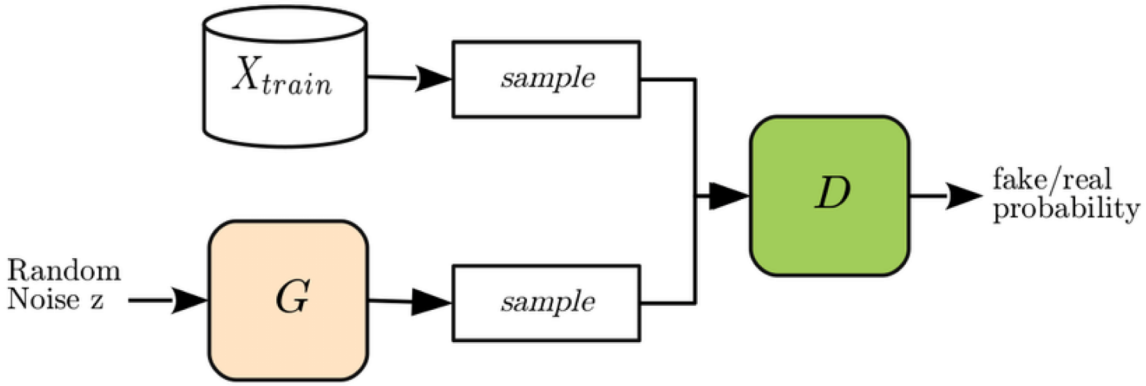
* decision rule strategy used in decision theory for minimizing the possible loss for a worst case (maximum loss) scenario

GANs – Framework

As outlined, GANs framework uses an adversarial training of two deep networks, each **trying** to defeat the other.

The **discriminator** is looking at real data from a training set and synthetic data from the generator. Its job is to classify each as incoming instance of data as either real or fake.

The **generator** attempts to fool the discriminator into thinking the data it is generating is real.



When both the generator and discriminator are configured correctly, they arrive at the Nash equilibrium where both are unable to find any advantage over the other.

GANs – Nitti-gritty

Notation:

$p_{data}(x)$ = distribution of real data

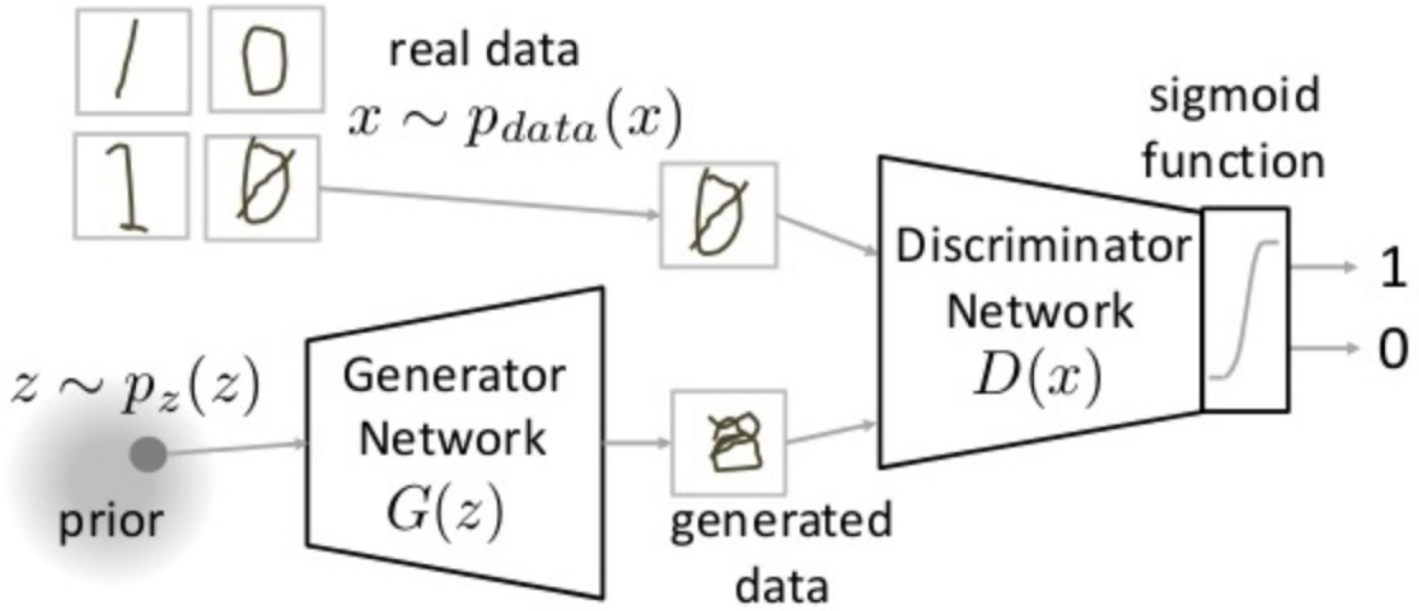
x = sample from distribution $p_{data}(x)$

$p_z(z)$ = distribution of generator

z = sample from distribution $p_z(z)$

$G(z,\theta_z)$ = Generator Network

$D(x,\theta_d)$ = Discriminator Network

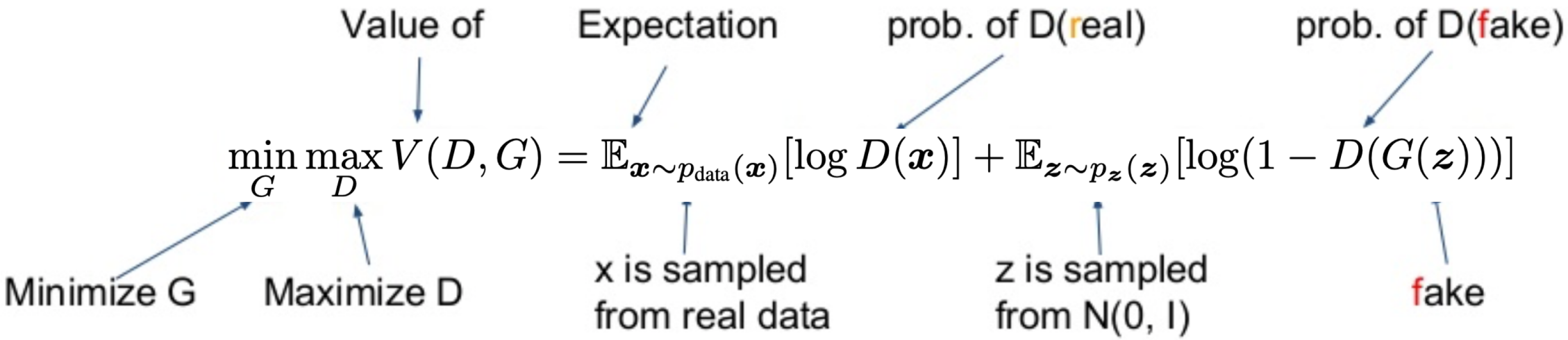


Competition between generator and discriminator is represented mathematically as:

$$\min_G \max_D V(D,G)$$

GANs – Nitti-gritty

Cost function *V(D,G)* is given by the cross entropy sums (see Kullback–Leibler divergence):



For better understanding:

$$V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

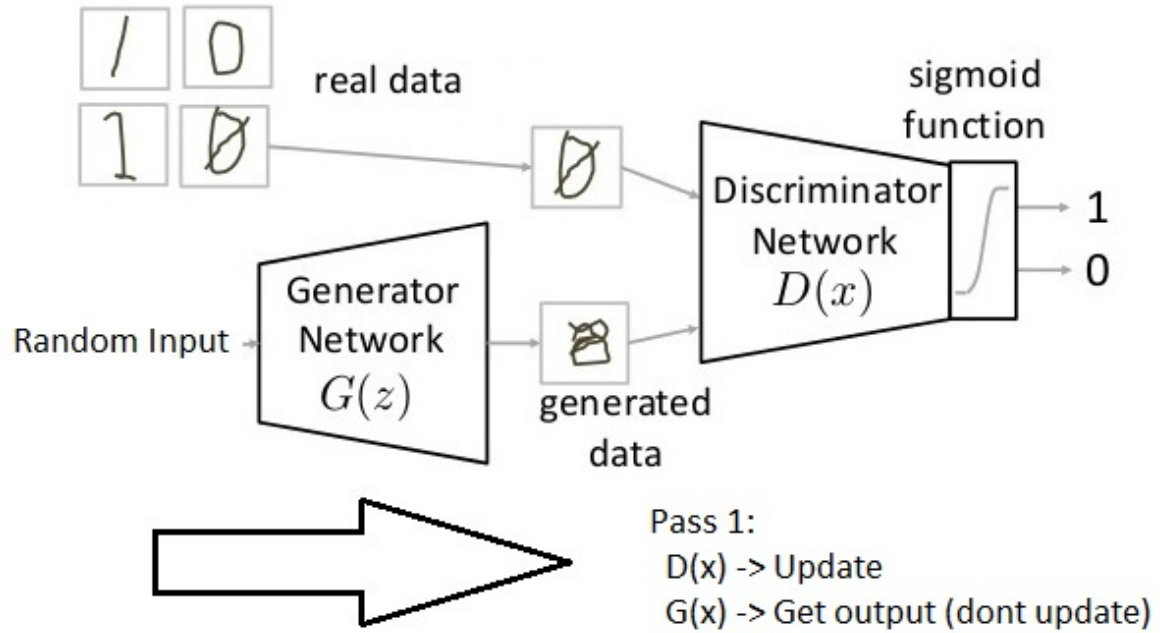
Entropy that the data *x* from real distribution $p_{data}(x)$ passes through the discriminator (best case scenario). "D" tries to maximize this to 1.

Entropy that the data *z* from random input $p_z(z)$ passes through „G", which then generates a fake sample subsequently passed through „D" to identify the fakeness (worst case scenario). „D" tries to maximize it to 0.

GANs – Training GAN

Training phase has two main subparts and they are done sequentially (by a minibatch):

Pass 1: Train discriminator and freeze generator (the G network does only forward pass and no backpropagation is applied).

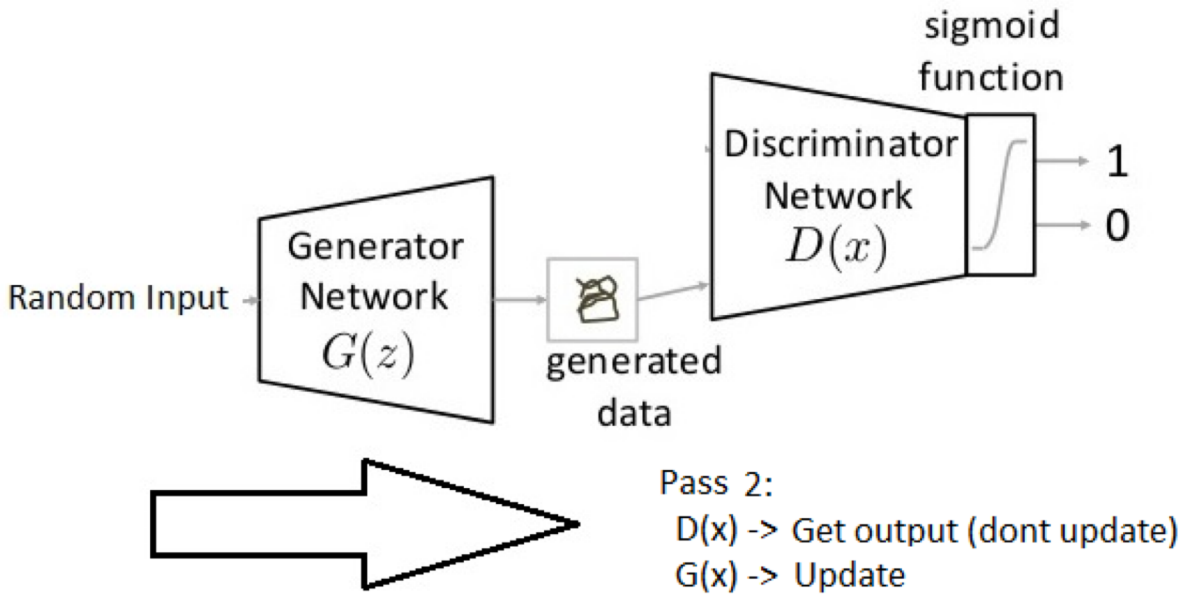


GANs – Training GAN

Training phase has two main subparts and they are done sequentially (by minibatch):

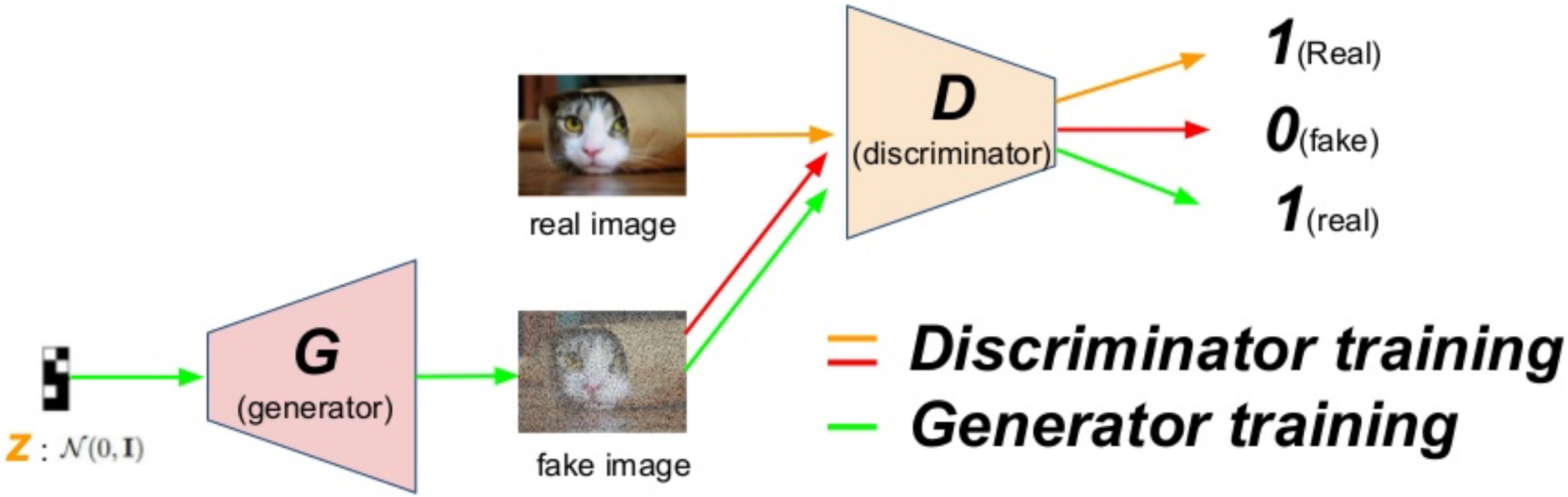
Pass 2: Train generator and freeze discriminator.

Note: in this step, the distribution $p_{data}(x)$ (coded into the $D(x)$ outputs in Pass 1) is back-propagated to the G(z) network, i.e. real data x knowledge is transferred into the G network.



GANs – Training GAN

Training phase has two main subparts and they are done sequentially – simply:



GANs – Training GAN

Mathematical notation – discriminator:

$$\max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Maximize prob. of D(real)

Minimize prob. of D(fake)

BCE(binary cross entropy) with label 1 for real, 0 for fake.

GANs – Training GAN

Mathematical notation – generator:

$$\min_G V(D,G) = \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \simeq \max_G \mathbb{E}_{z\sim p_z(z)}[\log(D(G(z)))]$$

Maximize prob. of D(fake)

BCE(binary cross entropy) with label 1 for fake.

GANs – Training GAN

Mathematical notation – equilibrium:

$$P_{data}(x) = P_{gen}(x) \ \forall x$$
$$\mathrm{D}(x) = \frac{1}{2} \ \forall x$$
→ Nash equilibrium

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \longrightarrow 0.5$$

$$\begin{aligned} C(G) &= \max_D V(G, D) \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{data}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] \end{aligned}$$

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right)$$

$$2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

Jansen-Shannon divergence

GANs – Implementation

Minibatch SGD alg.:

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

Discriminator updates

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

Generator updates

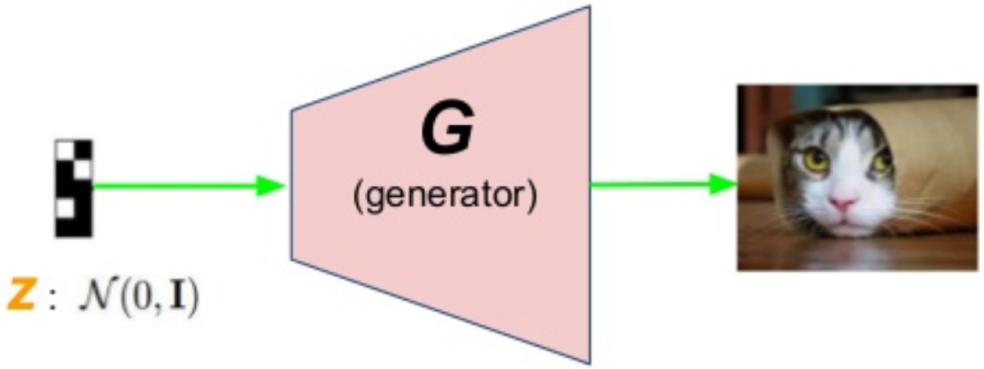
- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

GANs – Generating Data

When the Nash equilibrium between the discriminator and generator reached and required precision* of the generator achieved, the generator can create new data itself based on noise input and learnt distributions:



* Precision achievement is conditioned (not sufficient condition) by satisfactory capacity of *G* and *D* networks.

GANs – Architectures

GAN is a framework – architectures are its implementations:

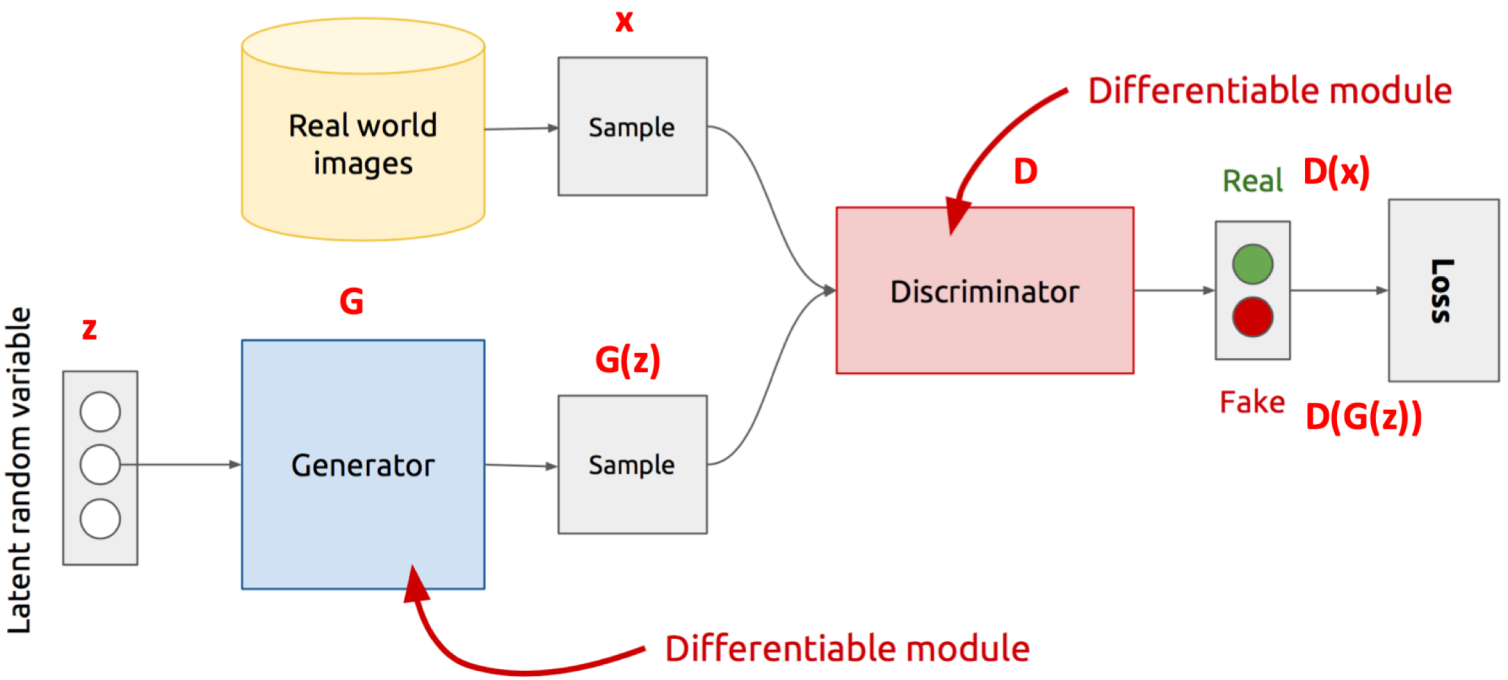
- Deep Convolutional GAN (DCGAN)
- Conditional GAN (cGAN)
- Stack GAN
- Info GAN
- Wasserstein GAN, …

Generative Adversarial Networks – Roadmap

1. Machine Learning – Short Revision
2. Limited Dataset – Bootstrap & Cross-Validation
3. Generative Adversarial Networks (GANs) – General Framework
4. Deep Convolutional GANs – Architecture
5. DCGANs Examples
6. Who is Edmond Belamy?

DCGAN – Adversarial Training Architecture

DCGAN – architecture of the GAN where both the discriminator and generator are represented by convolutional network (note these networks are not the same as in the case of CNNs!).

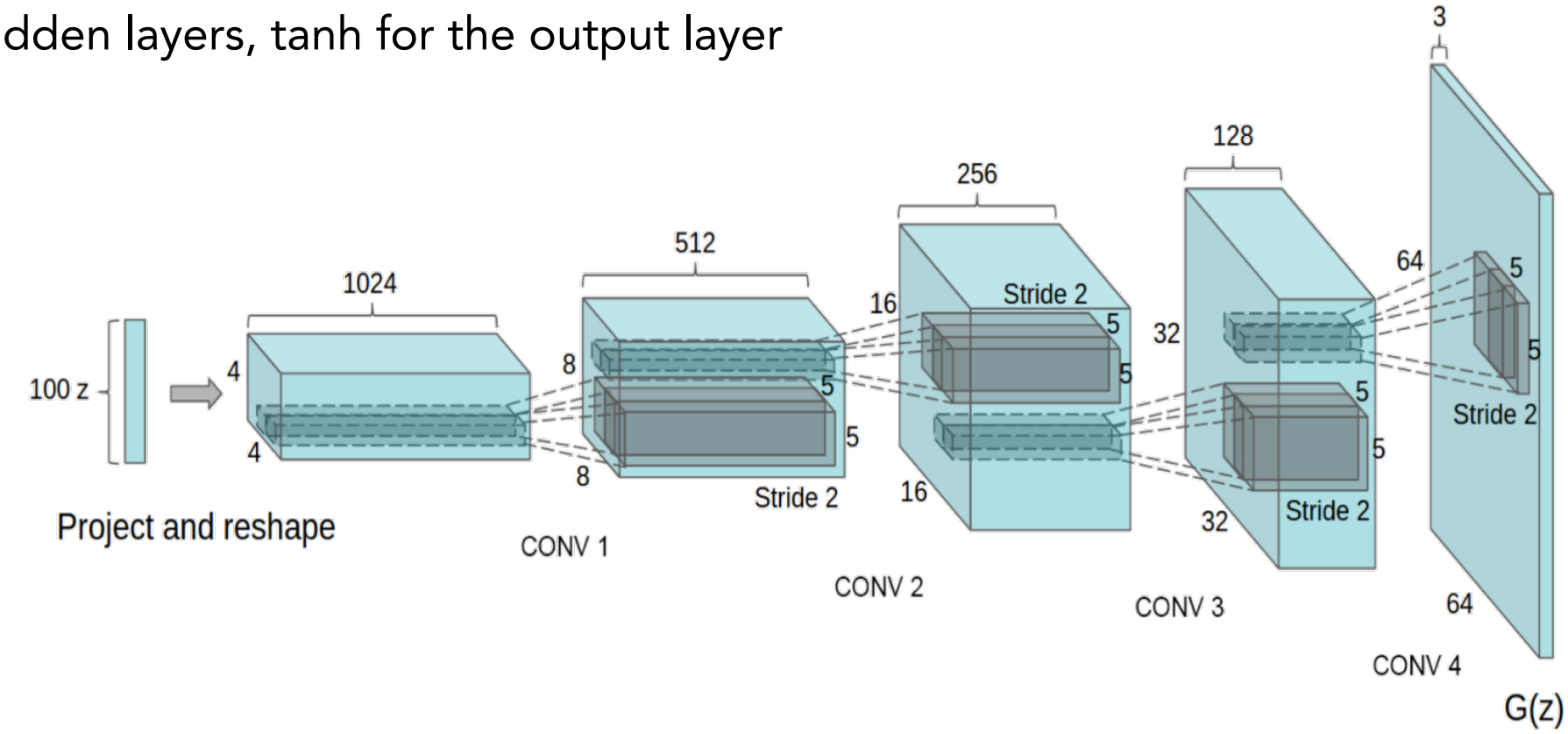


- The discriminator and generator are both trained together, in a stack. As one improves, the other also improves until hopefully the generator produces such good output that the discriminator is no longer able to identify the difference between *G(z)* and the training data *x*.

DCGAN – Generator Architecture

Generator architecture ideas:

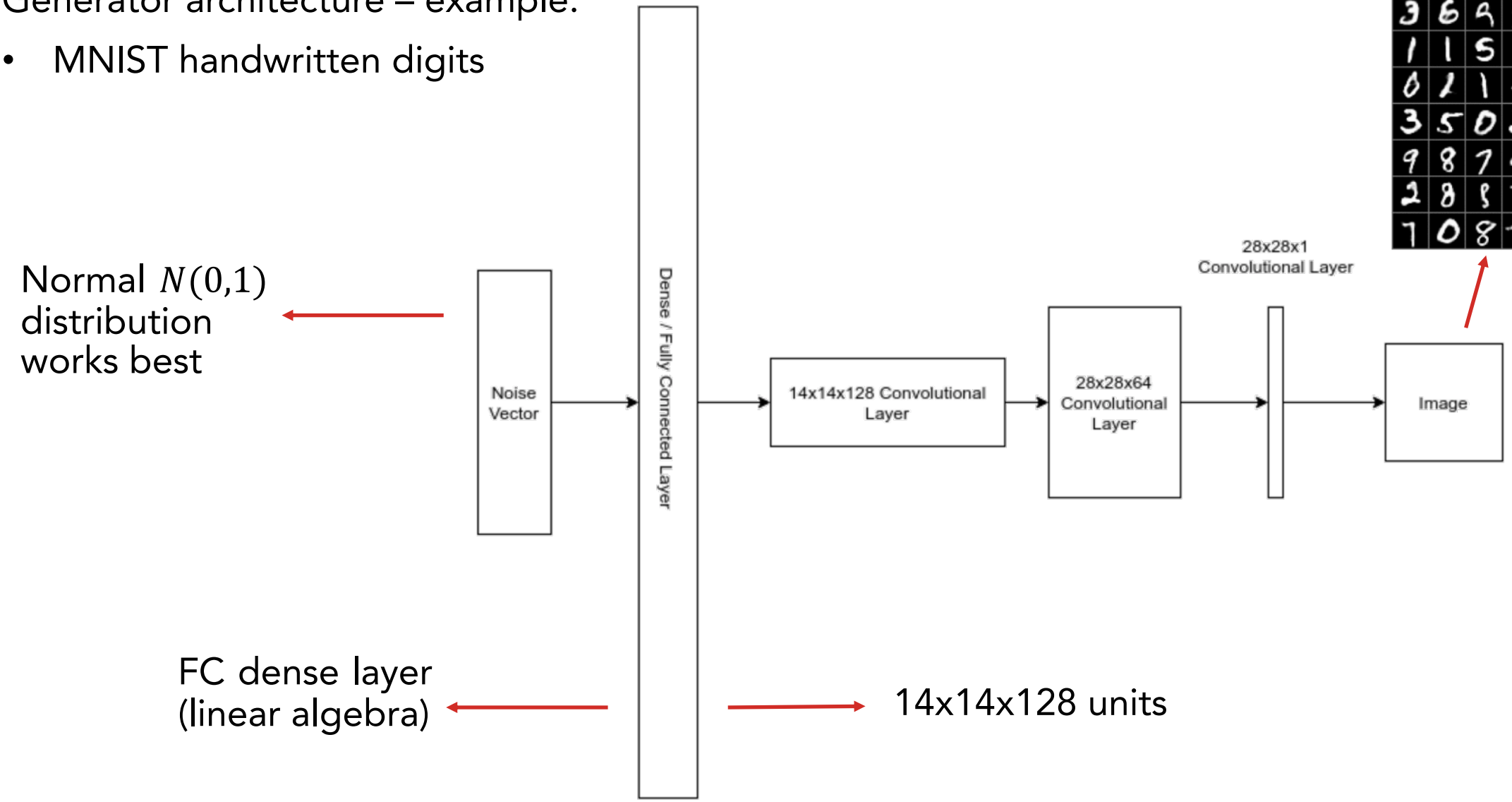
- FC hidden layers of a NN are replaced with convolutions (fractional-strided to upscale)
- Batch normalization after each layer
- ReLU for hidden layers, tanh for the output layer



DCGAN – Generator Architecture

Generator architecture – example:

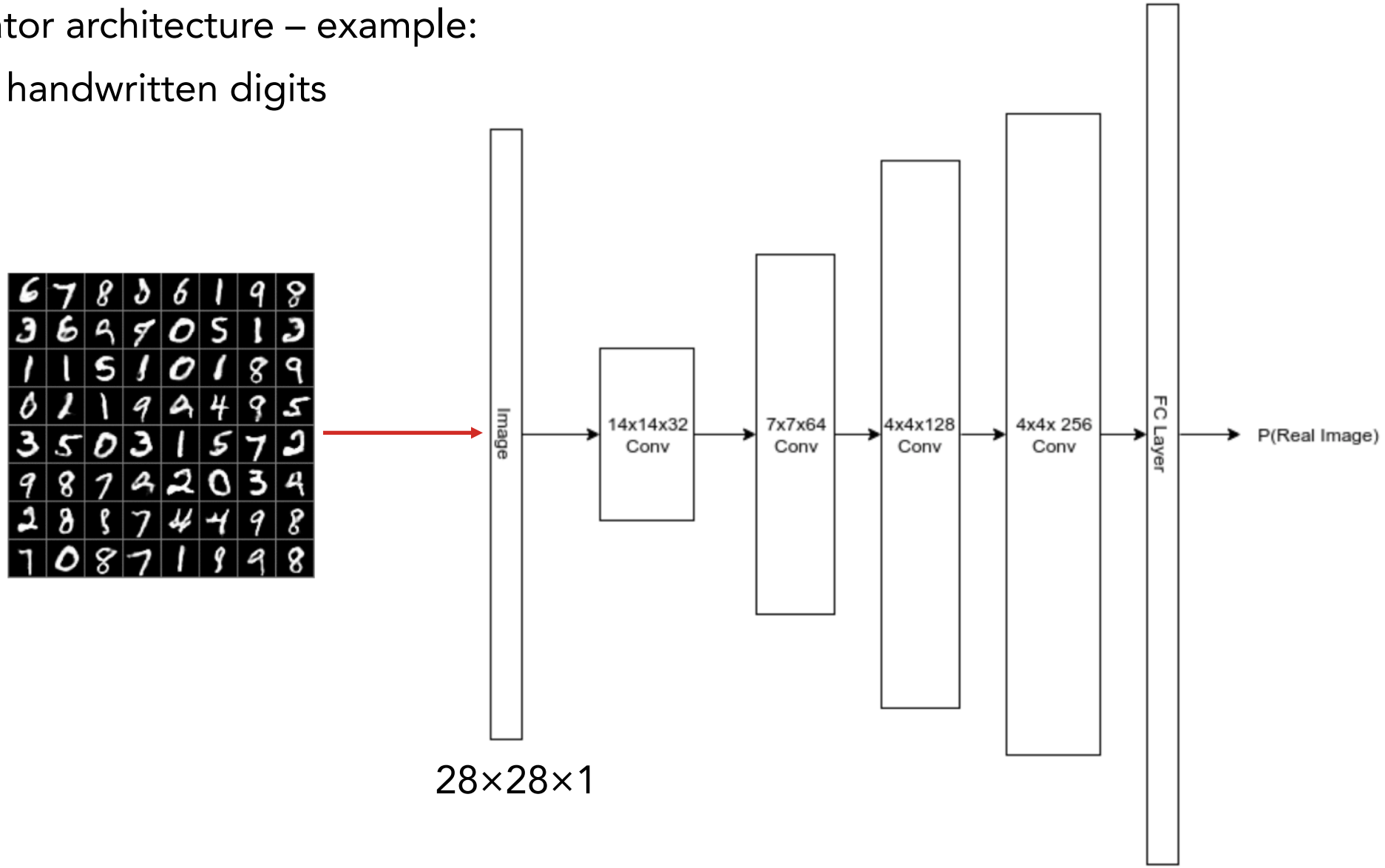
- MNIST handwritten digits



DCGAN – Discriminator Architecture

Discriminator architecture – example:

- MNIST handwritten digits



DCGAN – Stacked Training

The DCGAN framework is trained using so-called minibatches (GAN training requires several models to update their weights over the same batch, so it's slightly more complicated than a single parameter update as we know from the NN theory).

Training a DCGAN happens in two steps, for each batch:

Step 1 – train the discriminator

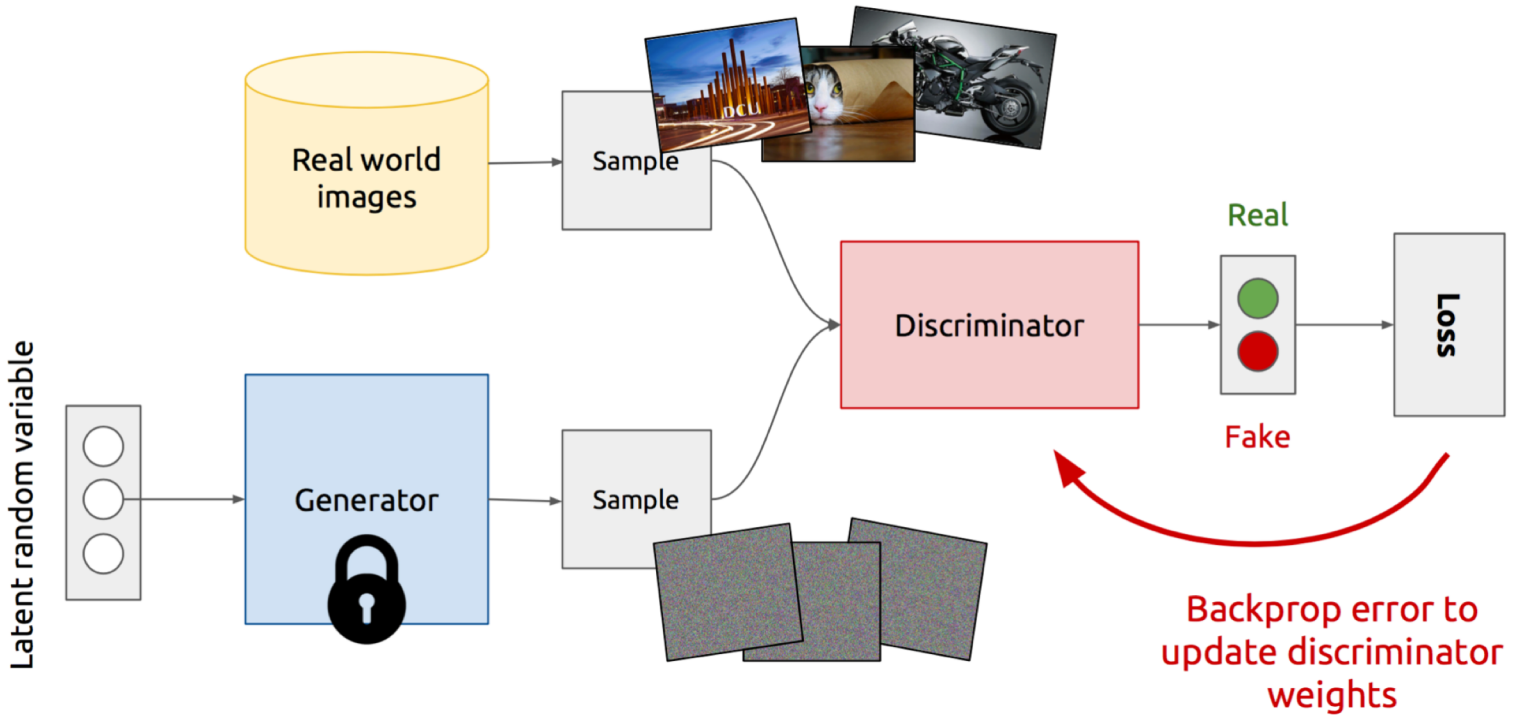
Train a DCGAN discriminator on both real data and generated data in supervised learning way. The label given to real data will obviously be 1 and the label for synthetic data is 0.

Step 2 – train the stack (generator using output of discriminator)

After the discriminator has updated it's weights, both the discriminator and generator together as a single model will be trained. Discriminator's weights are non-trainable – frozen in place but still allowing the discriminator to reverse propagate a gradient to the generator so that the generator can update its weights.

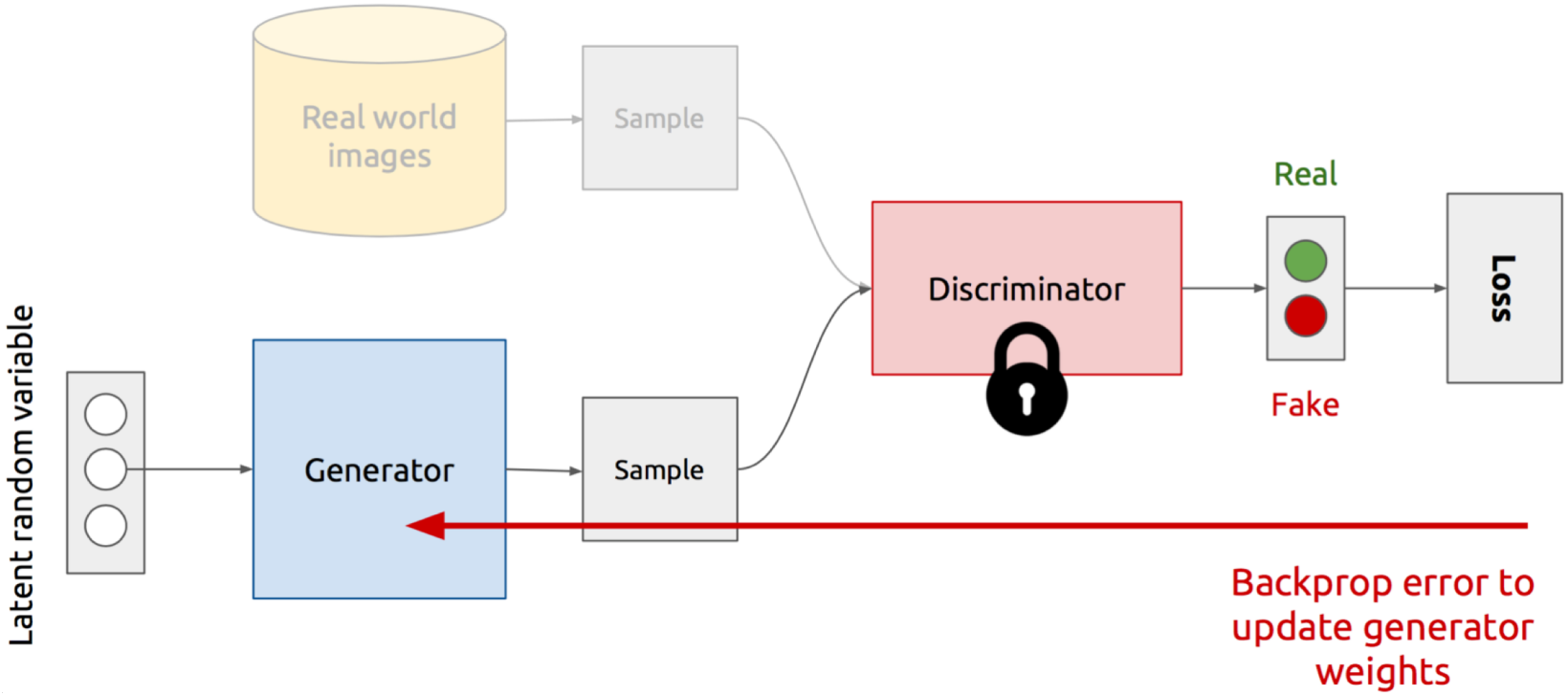
DCGAN – Training Discriminator

Step 1 – Training Discriminator:



DCGAN – Training Generator

Step 2 – Training Generator:



Generative Adversarial Networks – Roadmap

1. Machine Learning – Short Revision
2. Limited Dataset – Bootstrap & Cross-Validation
3. Generative Adversarial Networks (GANs) – General Framework
4. Deep Convolutional GANs – Architecture
5. DCGANs Examples
6. Who is Edmond Belamy?

DCGAN – Example: Generating MNIST Images

Example 1: MNIST digits generated conditioned on their class label:

- Modified National Institute of Standards and Technology dataset
- 70 000 small square 28×28 pixel grayscale images of handwritten single digits

$[1,0,0,0,0,0,0,0,0,0] \rightarrow$

$[0,1,0,0,0,0,0,0,0,0] \rightarrow$

$[0,0,1,0,0,0,0,0,0,0] \rightarrow$

$[0,0,0,1,0,0,0,0,0,0] \rightarrow$

$[0,0,0,0,1,0,0,0,0,0] \rightarrow$

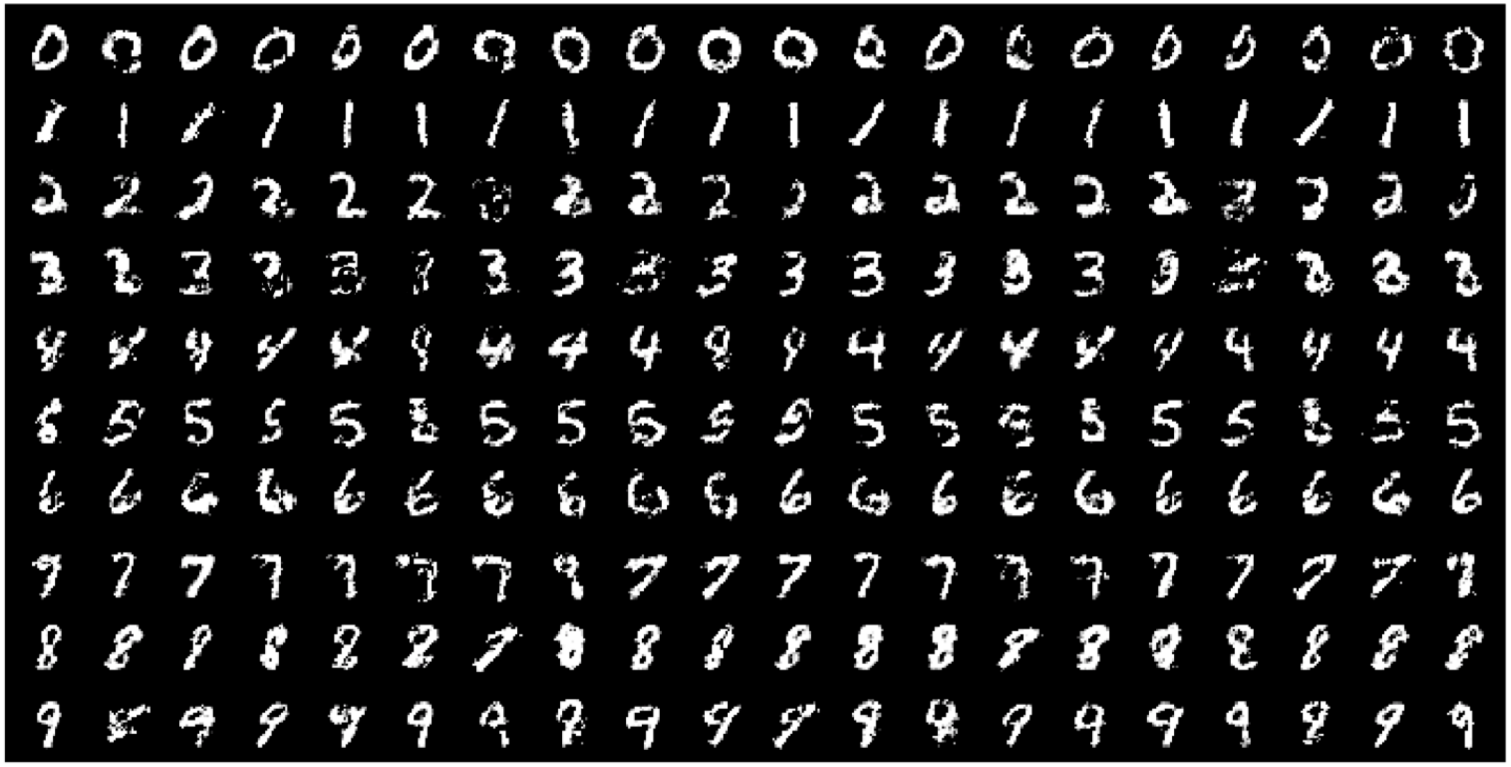
$[0,0,0,0,0,1,0,0,0,0] \rightarrow$

$[0,0,0,0,0,0,1,0,0,0] \rightarrow$

$[0,0,0,0,0,0,0,1,0,0] \rightarrow$

$[0,0,0,0,0,0,0,0,1,0] \rightarrow$

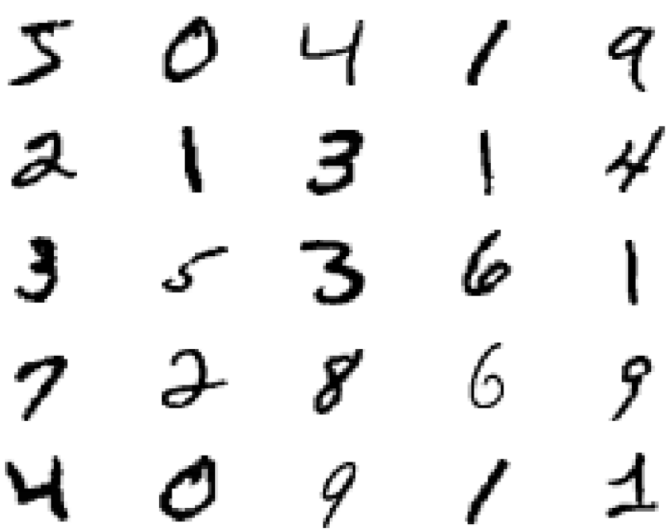
$[0,0,0,0,0,0,0,0,0,1] \rightarrow$



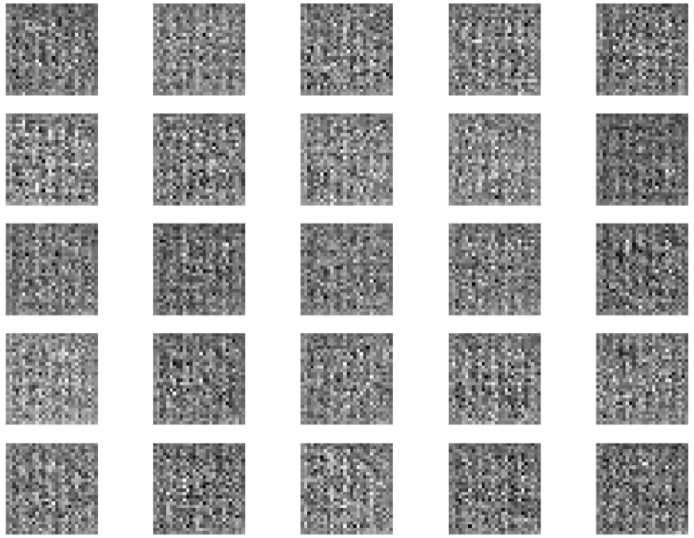
DCGAN – Example: Generating MNIST Images

Example 1: MNIST digits generated conditioned on their class label:

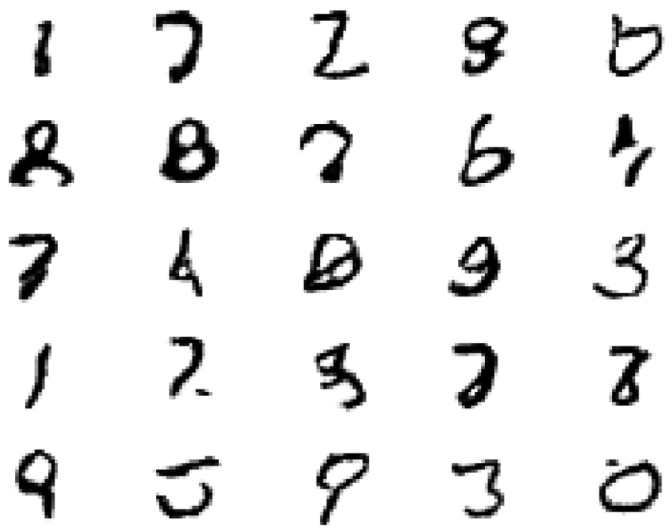
- Modified National Institute of Standards and Technology dataset
- 70,000 small square 28×28 pixel grayscale images of handwritten single digits



First 25 handwritten digits from the MNIST dataset



Output of the untrained generator model – $G^0(z)$

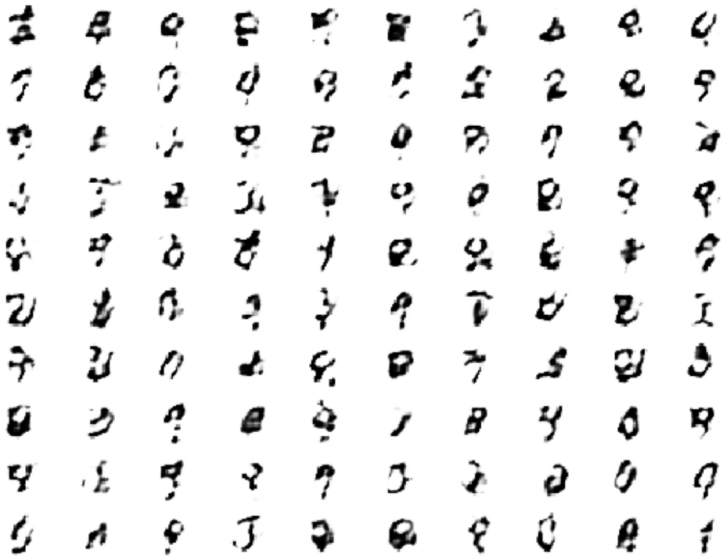


GAN generated MNIST handwritten images

DCGAN – Example: Generating MNIST Images

Example 1: MNIST digits generated conditioned on their class label:

- Inside training – after 40 epochs the model begins to generate plausible MNIST figures
- GAN generated MNIST figures conditioned by a number of epochs accomplished:



after 10 epochs

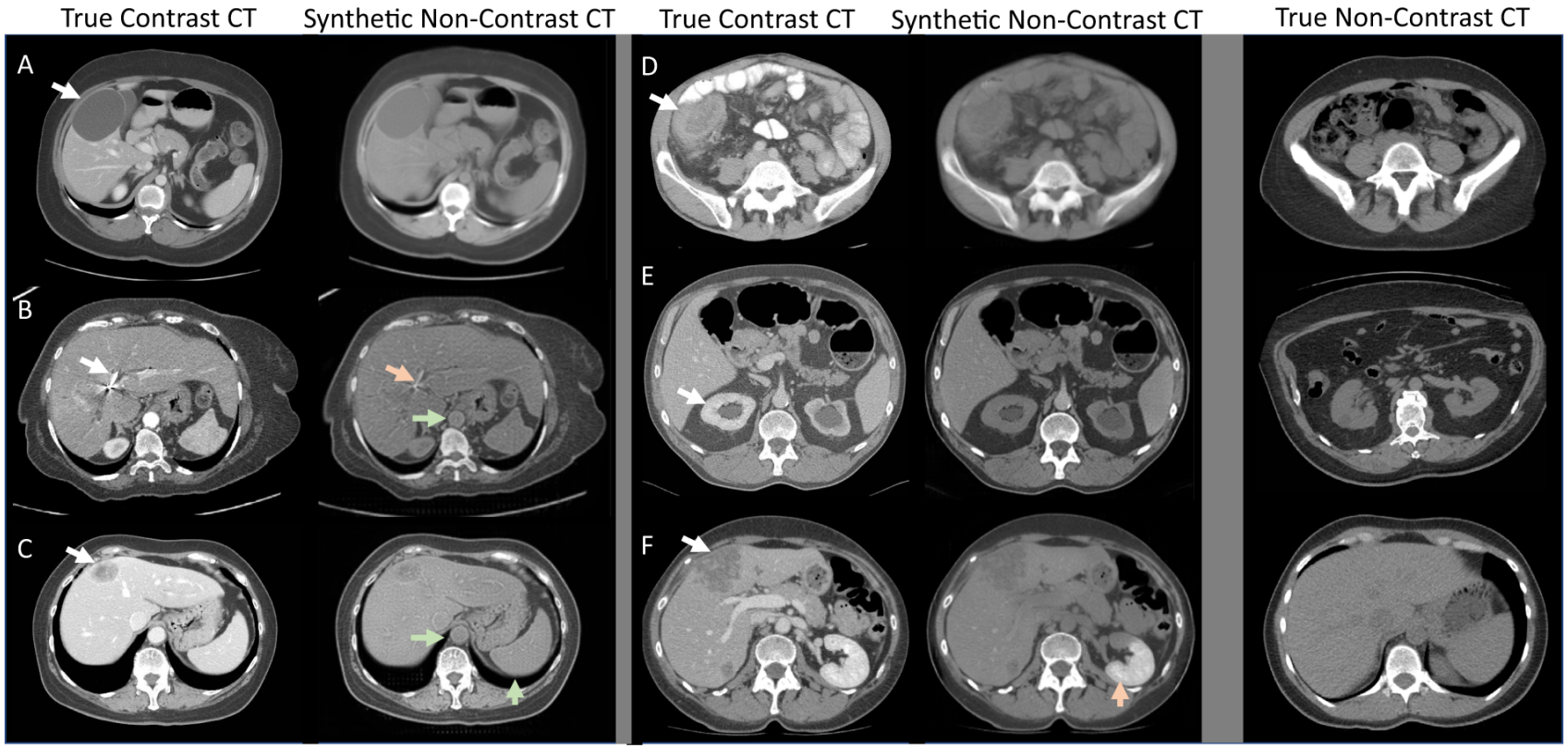
after 40 epochs

after 100 epochs

DCGAN – Example: CT Images Segmentation

Example 2: Data augmentation using Generative Adversarial Networks (CycleGAN) to improve generalizability in CT segmentation tasks (11-2019, Nature).

- CT scans are usually performed with intravenous contrast agent (iodide) to detect lesions
- These scans are not convenient for automatic data augmentation algs. (segmentation) → synthetic non-contrast image was generated by the trained CycleGAN



Note: in radiology, besides brain analysis, CT is used often for abdominal organs

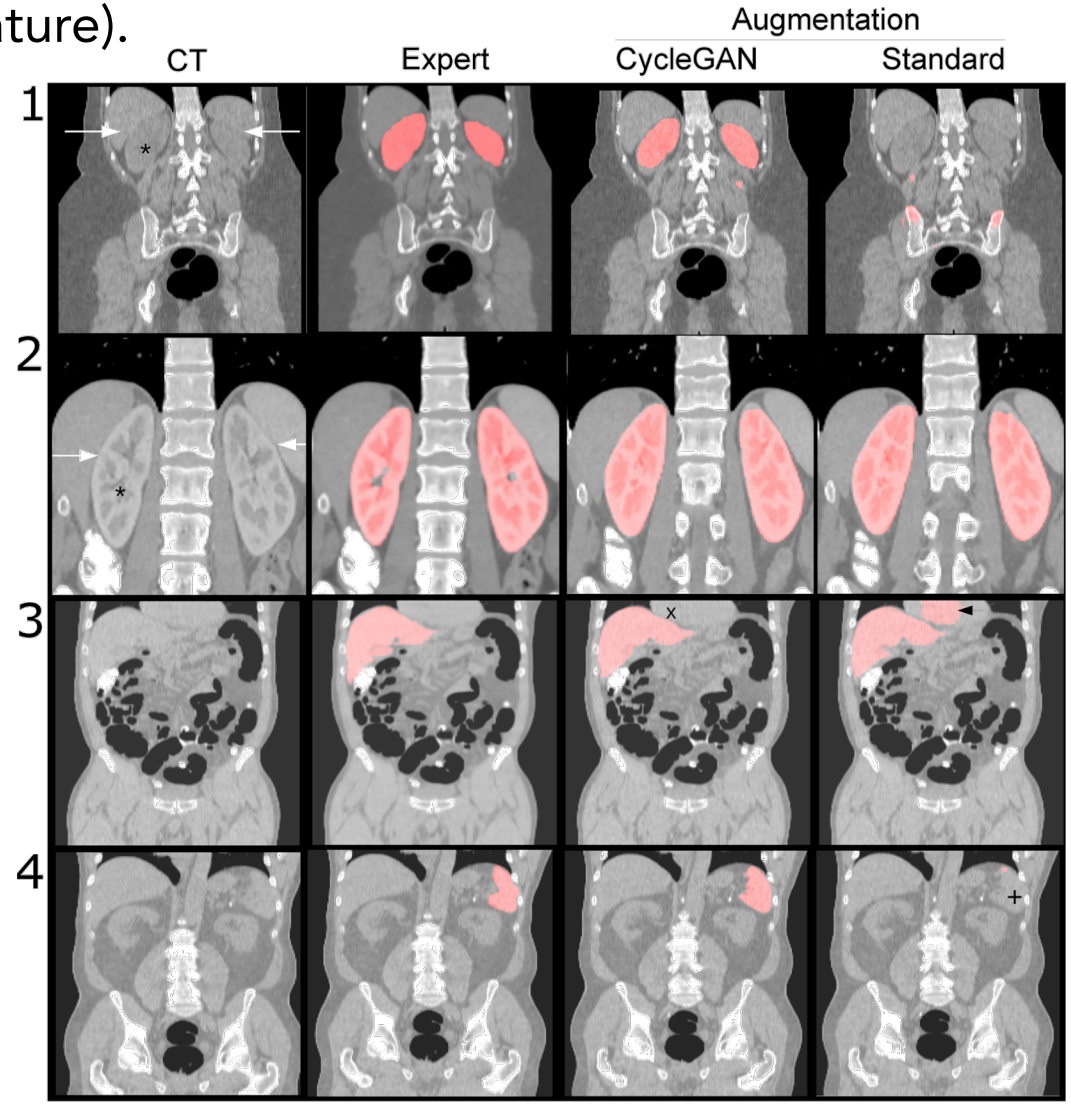
DCGAN – Example: CT Images Segmentation

Example 2: Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks (11-2019, Nature).

Examples of kidney, liver and spleen segs. ⟶

- first column – original CT images
- second column – expert segmentation
- third column – CycleGAN images segmentation
- fourth column – standard augmented training

See closely "*x*" in the third row – liver/heart boundary:



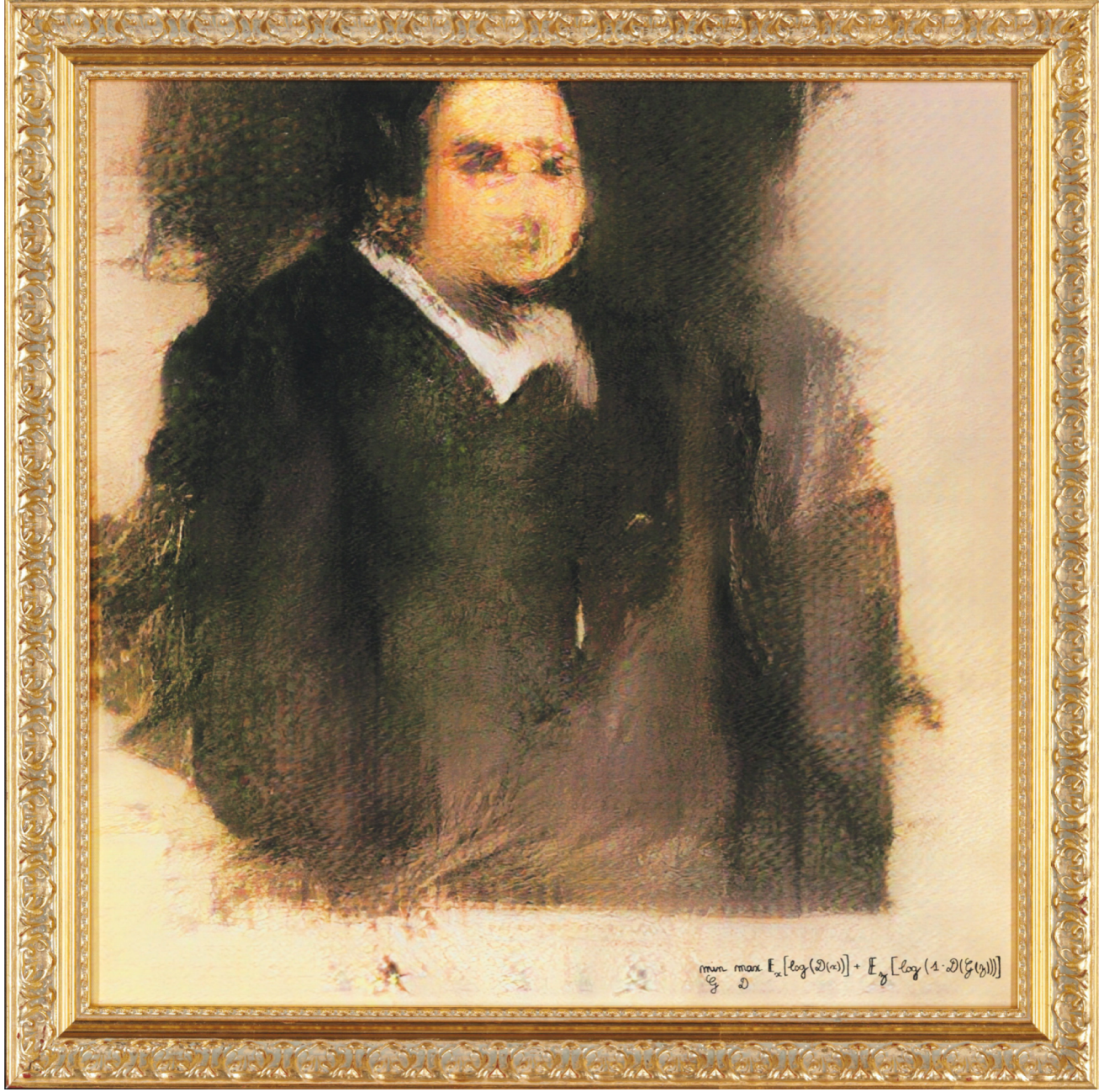
References

Courtesy of:

- Goodfellow, J.I., et al.: Generative Adversarial Networks. Advances in Neural Information Processing Systems. Curran Associates Inc., 2014, pp. 2672—2680.
- Bernico, M.: Deep learning quick reference. Packt Publishing, Birmingham, 2018, 261 p. ISBN 9781788838917.
- Faizan Shaikh: Introductory guide to Generative Adversarial Networks (GANs) and their promise! https://www.analyticsvidhya.com/blog/2017/06/introductory-generative-adversarial-networks-gans/?utm_source=blog&utm_medium=top-5-GANs-applications.
- Namju Kim: Generative adversarial networks. https://www.slideshare.net/ssuser77ee21/generative-adversarial-networks-70896091.
- Sandfort, V., Yan, K., Pickhardt, P.J. et al. Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks. Sci Rep 9, 2019, Nature, 16884.

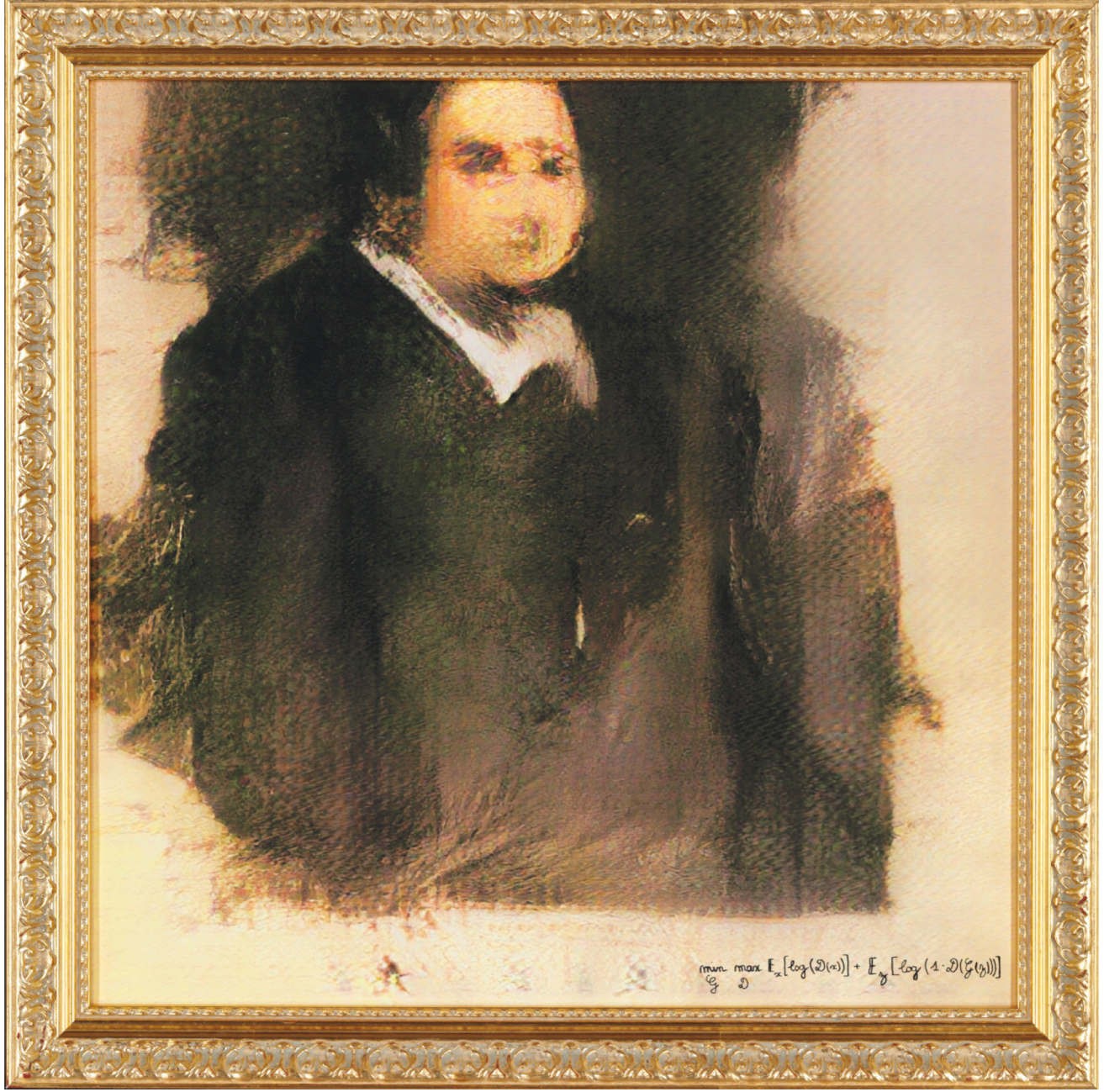
Generative Adversarial Networks – Roadmap

1. Machine Learning – Short Revision
2. Limited Dataset – Bootstrap & Cross-Validation
3. Generative Adversarial Networks (GANs) – General Framework
4. Deep Convolutional GANs – Architecture
5. DCGANs Examples
6. Who is Edmond Belamy?



Let’s get back to Edmond… $432,500 – nearly 45 times its high estimate. What is so worthy on it?

Author, apparently. In cursive Gallic script you can read this algebraic formula at the bottom right.



The portrait was created by the artificial intelligence algorithm, Generative Adversarial Network, learnt on a data set of 15000 portraits painted between the 14th century to the 20th by human painters. The Belamy family is fictional.