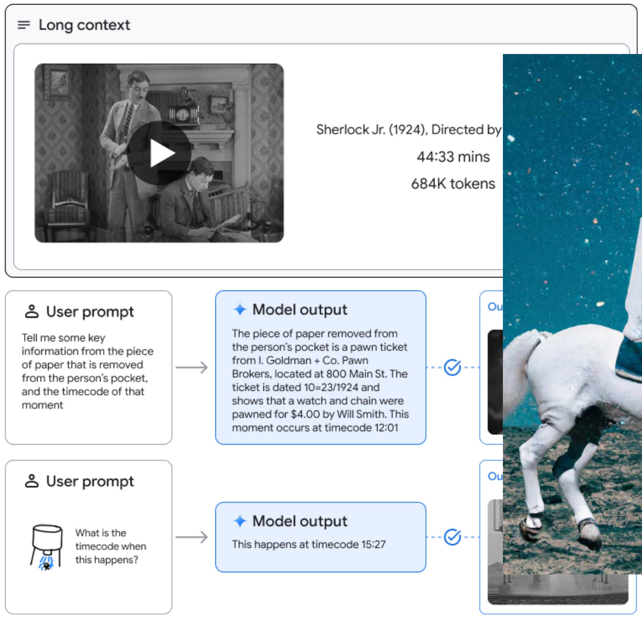
Generative machine learning models

Generative models

- Most of today's poster child applications of deep learning are generative models
  - LLMs: (Chat)-GPT, Gemini, Llama, Mistral, …
  - Image generation models: Stable diffusion, Dall-E, …
  - Video generation: Sora, …



Long context

Sherlock Jr. (1924), Directed by

44:33 mins

684K tokens

User prompt

Tell me some key information from the piece of paper that is removed from the person's pocket, and the timecode of that moment

Model output

The piece of paper removed from the person's pocket is a pawn ticket from I. Goldman + Co. Pawn Brokers, located at 800 Main St. The ticket is dated 10=23/1924 and shows that a watch and chain were pawned for $4.00 by Will Smith. This moment occurs at timecode 12:01

User prompt

What is the timecode when this happens?

Model output

This happens at timecode 15:27

https://storage.googleapis.com/deepmind-media/gemini/gemini_v1_5_report.pdf

https://openai.com/dall-e-2

https://openai.com/sora

Generative models

- Fundamental idea: learn probability distribution

$$p(x) \approx p_\theta(x)$$

- Distribution: natural language, natural images, atmospheric states, ..

Generative models

- Fundamental idea: learn probability distribution

$$p(x) \approx p_\theta(x)$$

- Distribution: natural language, natural images, atmospheric states, ..
- Neural network provides numerical model for distribution
  - Unparametric (non-Gaussian etc)
  - Network outputs (Monte Carlo-style) samples from the distribution since no explicit representation is possible
  - Learned by providing samples from distribution

Generative models

- Fundamental idea: learn probability distribution

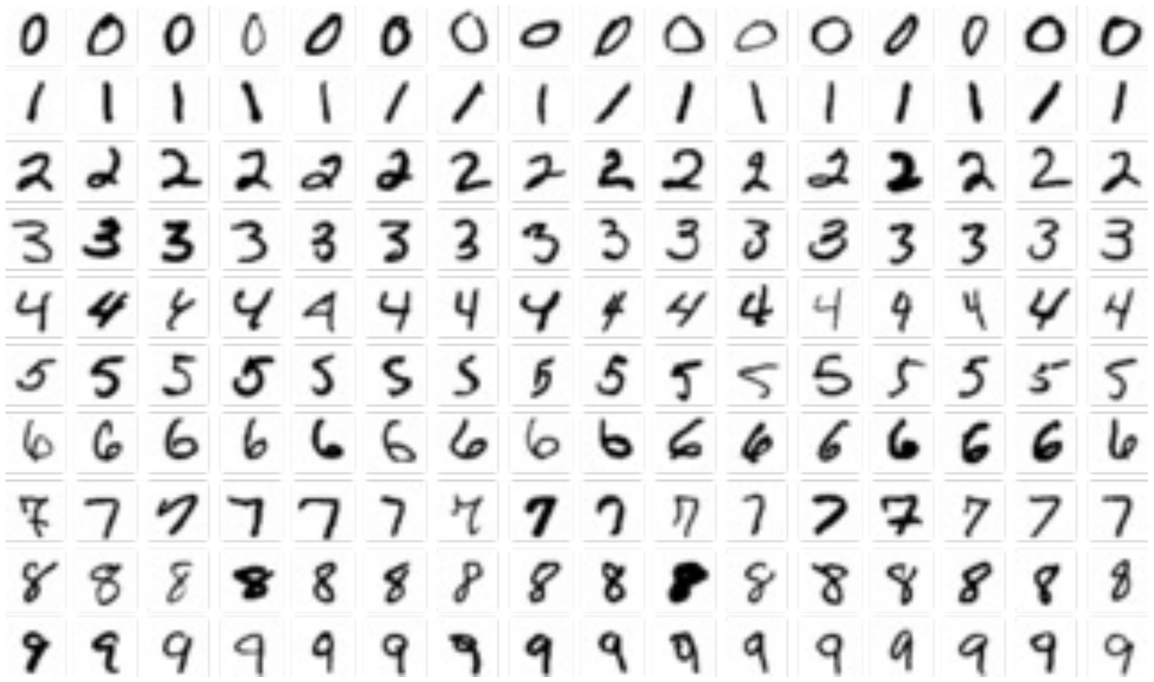
$$p(x) \approx p_\theta(x)$$

- Distribution: natural language, natural images, atmospheric states, ..
- Neural network provides numerical model for distribution
  - Unparametric (non-Gaussian etc)
  - Network outputs (Monte Carlo-style) samples from the distribution since no explicit representation is possible
  - Learned by providing samples from distribution
- Often we learn a joint or conditional probability distribution to have control over the output after learning

$$p(y, x) \approx p_\theta(y, x)$$

Simple Example: MNIST

- MNIST: digit recognition dataset (for automatic postal code classification)



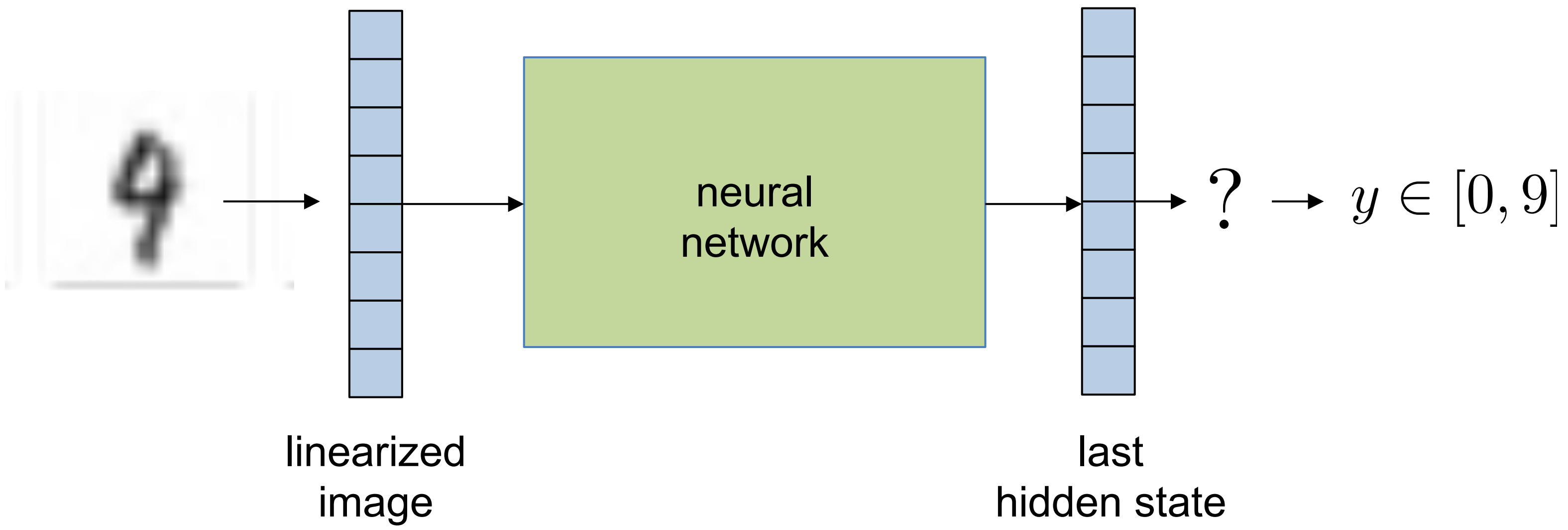
https://en.wikipedia.org/wiki/MNIST_database

Simple Example: MNIST

- MNIST: digit recognition dataset (for automatic postal code classification)
- What should our training objective be?
  - Natural choice: integer in [0, 9]
  - Network operates internally on hidden/latent states $h \in \mathbb{R}^n$

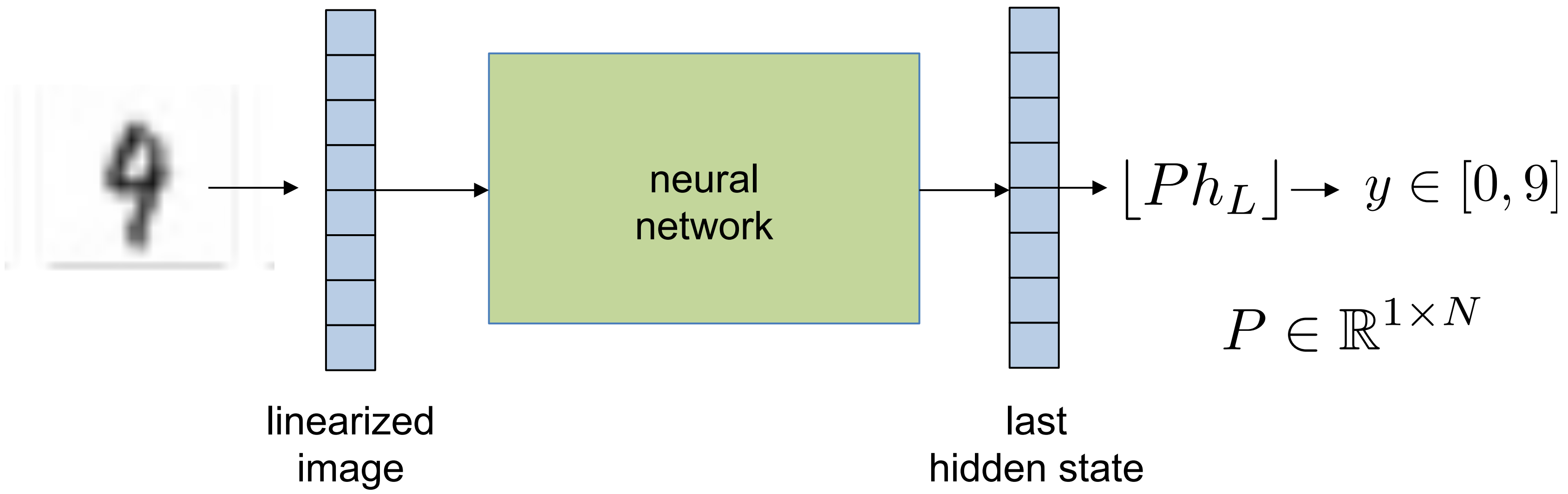
Simple Example: MNIST

- MNIST: digit recognition dataset (for automatic postal code classification)
- What should our training objective be?
  - Natural choice: integer in [0, 9]
  - Network operates internally on hidden/latent states $h \in \mathbb{R}^n$



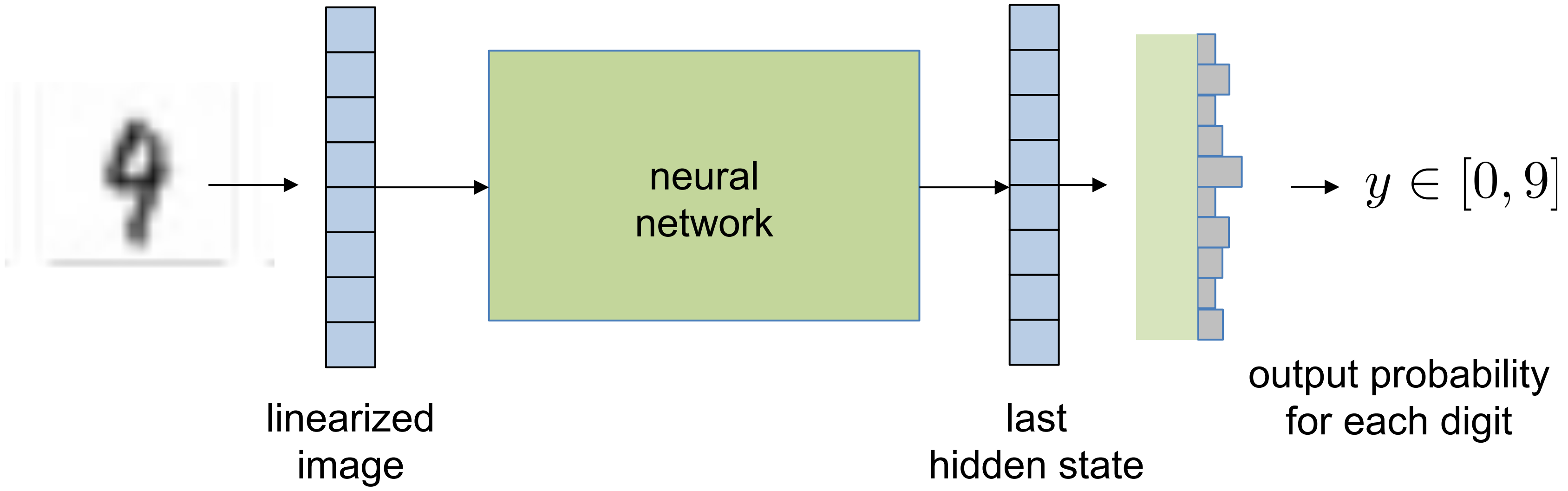
Simple Example: MNIST

- MNIST: digit recognition dataset (for automatic postal code classification)
- What should our training objective be?
  - Natural choice: integer in [0, 9]
  - Network operates internally on hidden/latent states $h \in \mathbb{R}^n$


neural
network
linearized
image
last
hidden state
$\lfloor P h_L \rfloor \rightarrow y \in [0, 9]$
$P \in \mathbb{R}^{1 \times N}$

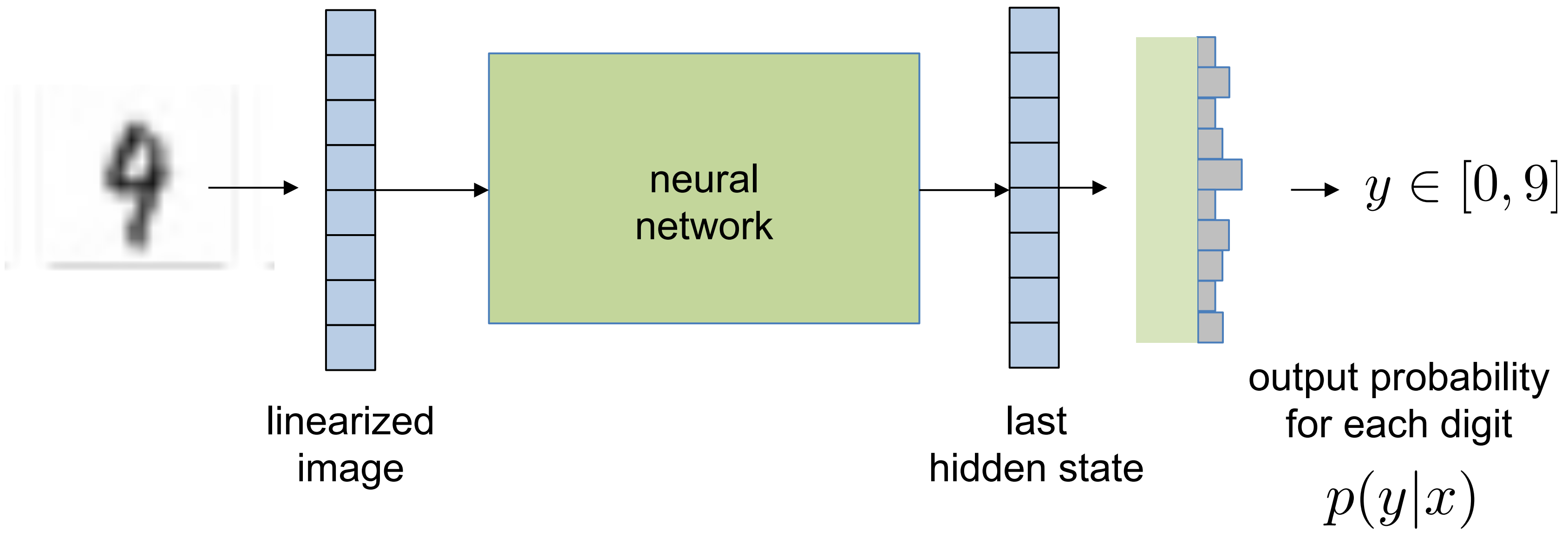
Simple Example: MNIST

- MNIST: digit recognition dataset (for automatic postal code classification)
- What should our training objective be?
  - Natural choice: integer in [0, 9]
  - Network operates internally on hidden/latent states $h \in \mathbb{R}^n$



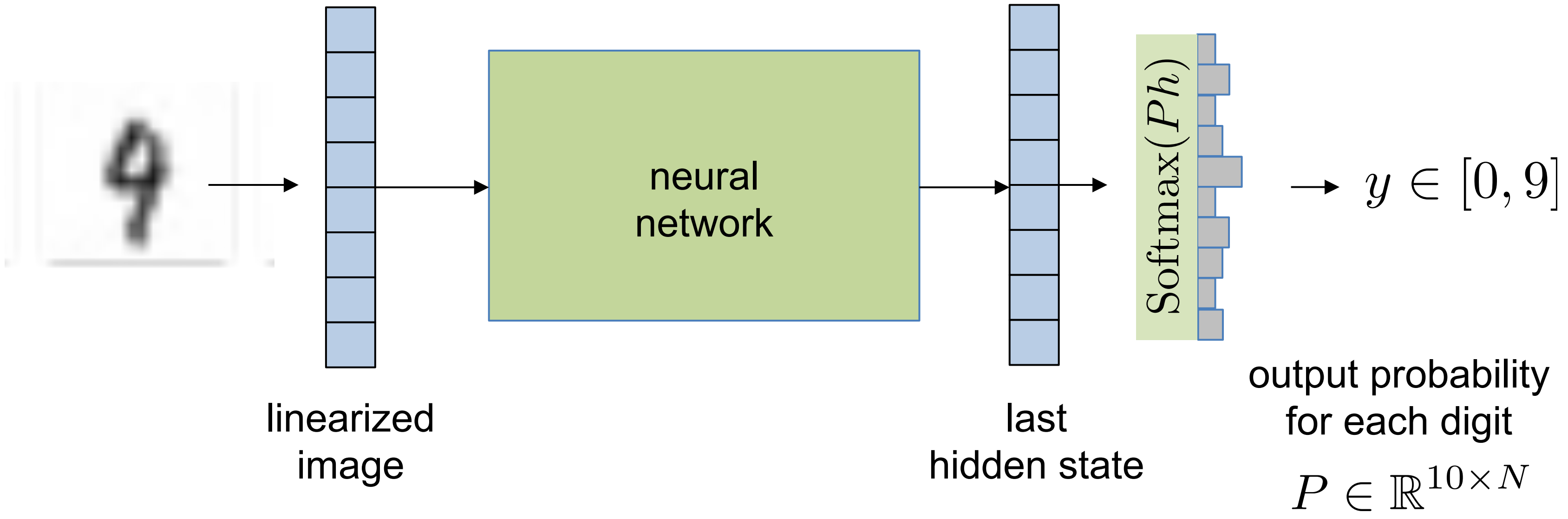
Simple Example: MNIST

- MNIST: digit recognition dataset (for automatic postal code classification)
- What should our training objective be?
  - Natural choice: integer in [0, 9]
  - Network operates internally on hidden/latent states $h \in \mathbb{R}^n$



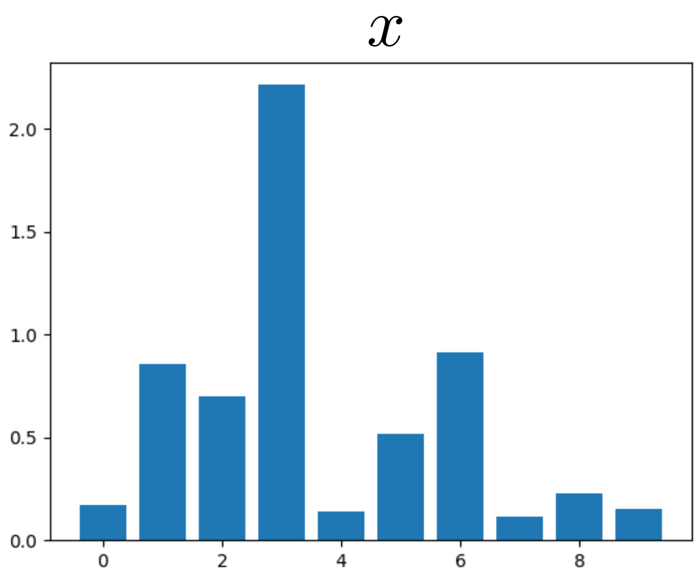
Simple Example: MNIST

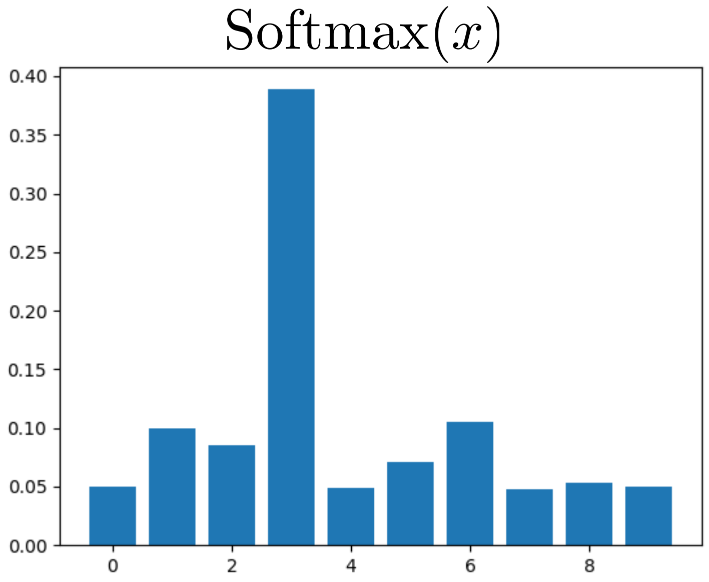
- MNIST: digit recognition dataset (for automatic postal code classification)
- What should our training objective be?
  - Natural choice: integer in [0, 9]
  - Network operates internally on hidden/latent states $h \in \mathbb{R}^n$



Recap: softmax

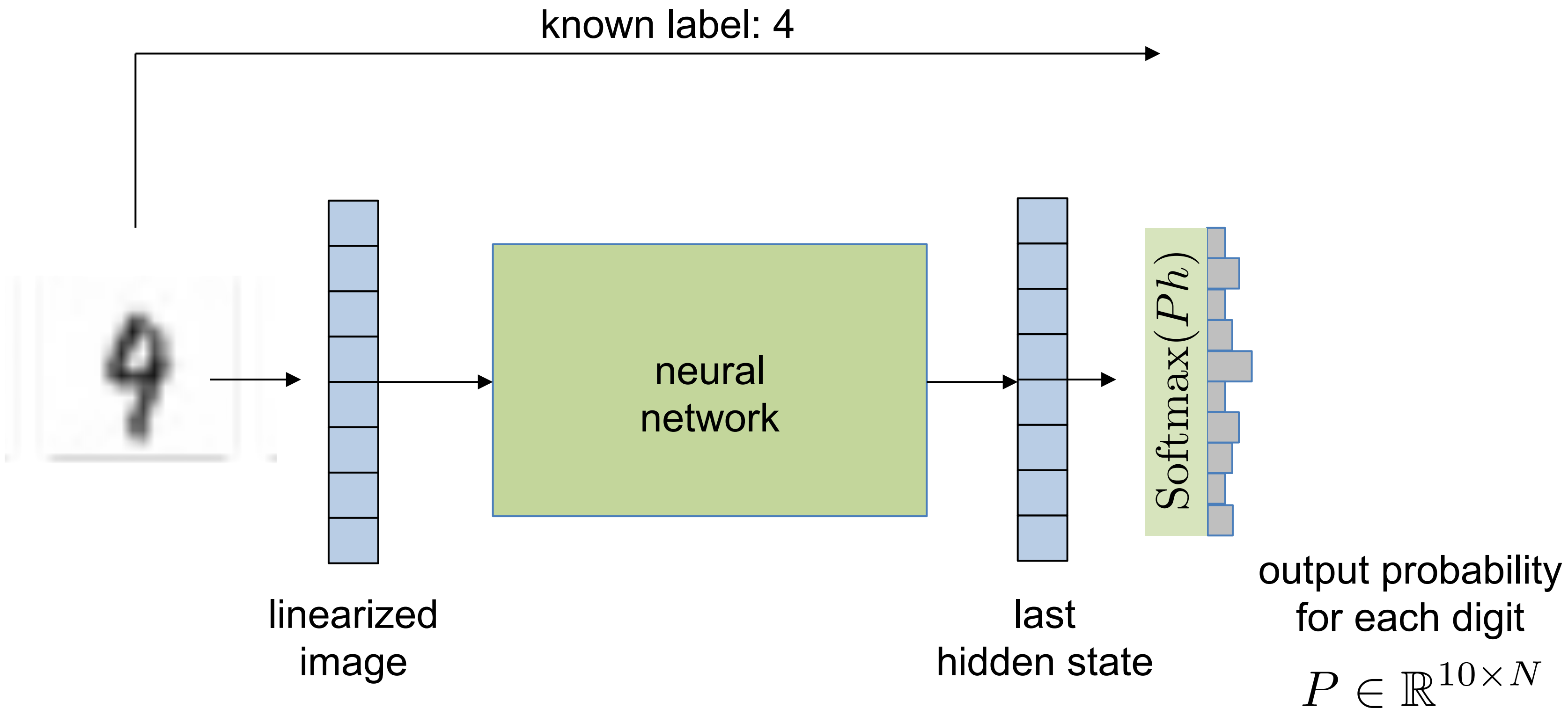
$$\mathrm{Softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad , \; x \in \mathbb{R}^n$$





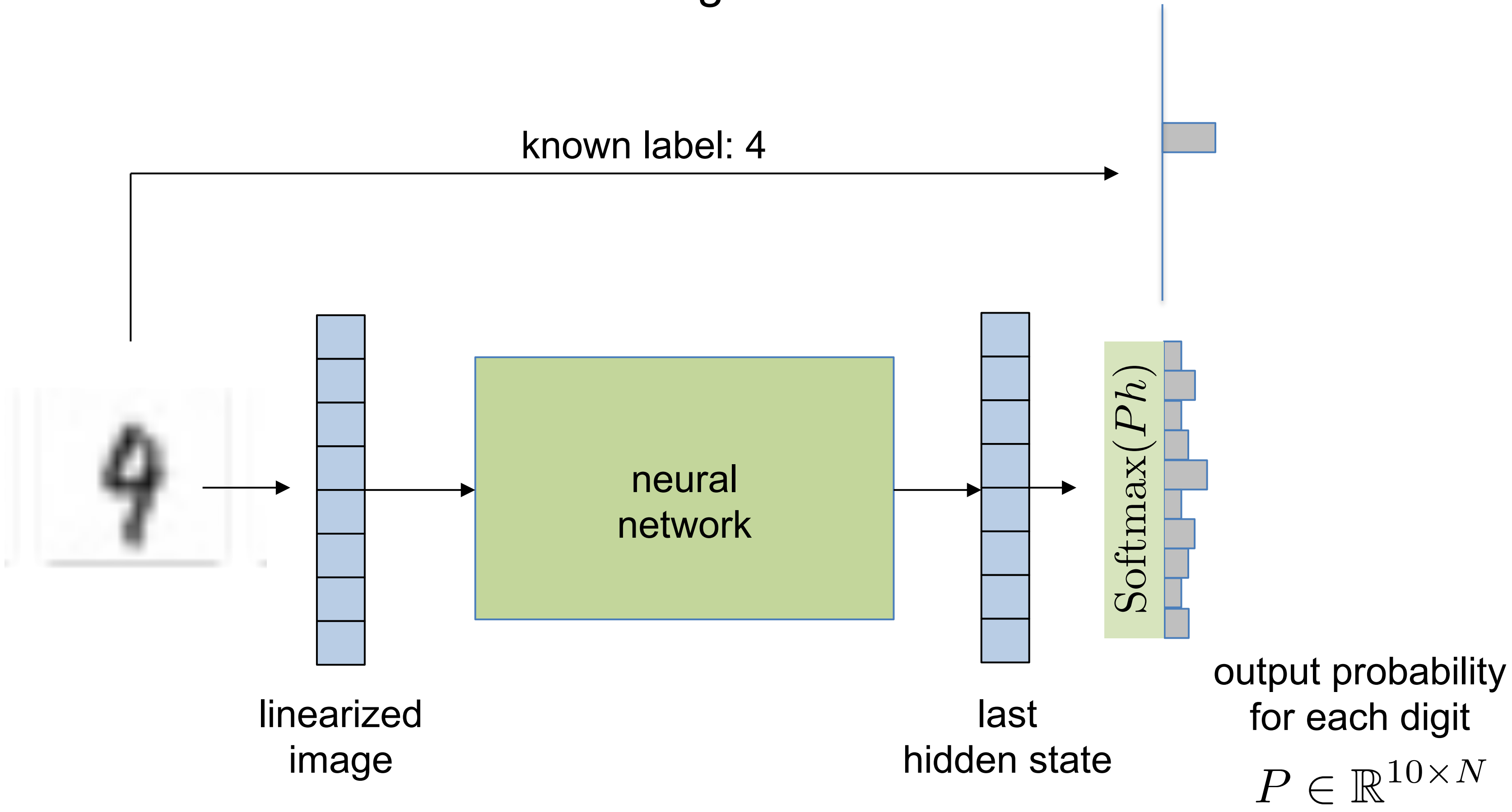
Simple Example: MNIST

- Training: how can we train the probability distribution—we only have one label for each image?



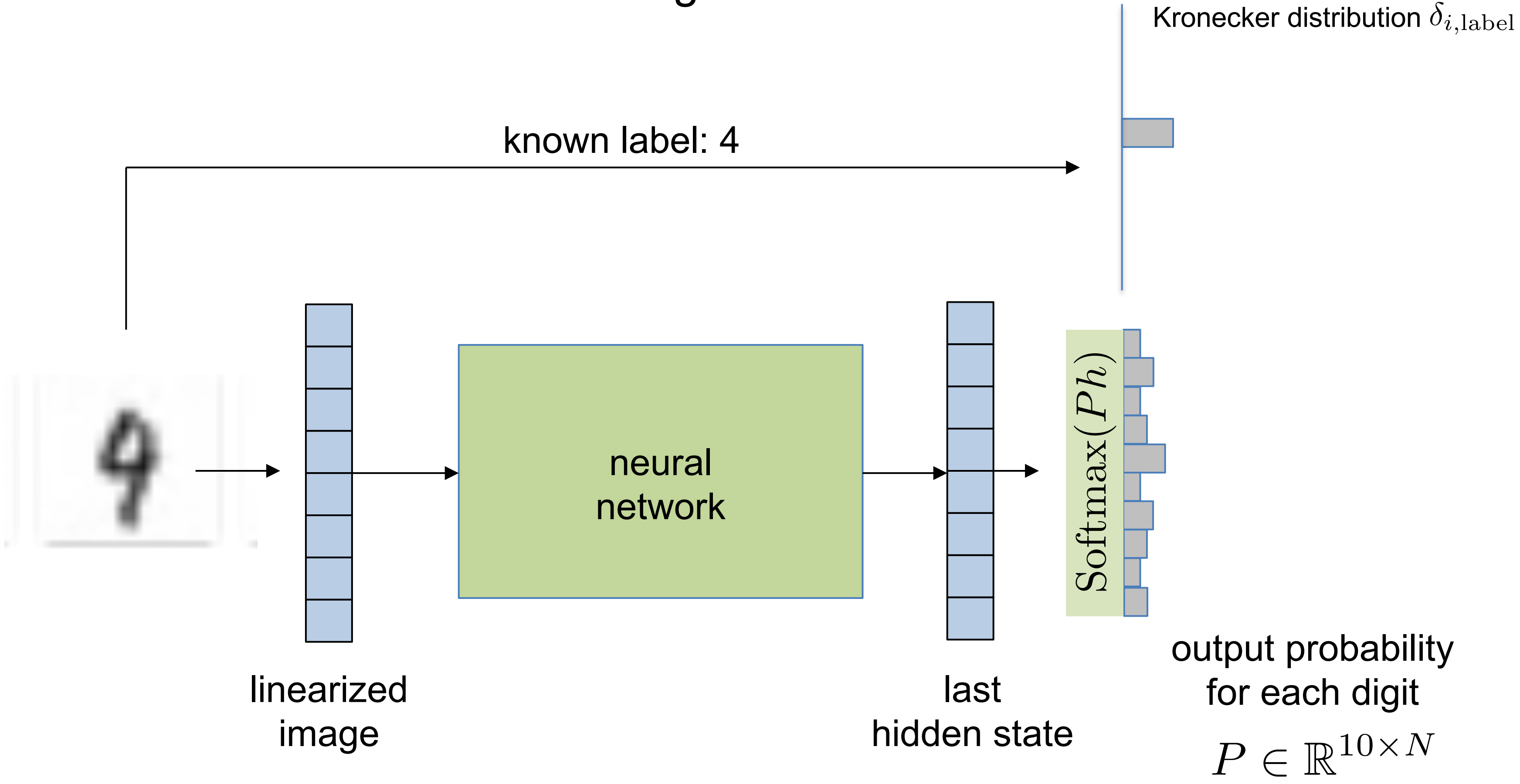
Simple Example: MNIST

- Training: how can we train the probability distribution—we only have one label for each image?



Simple Example: MNIST

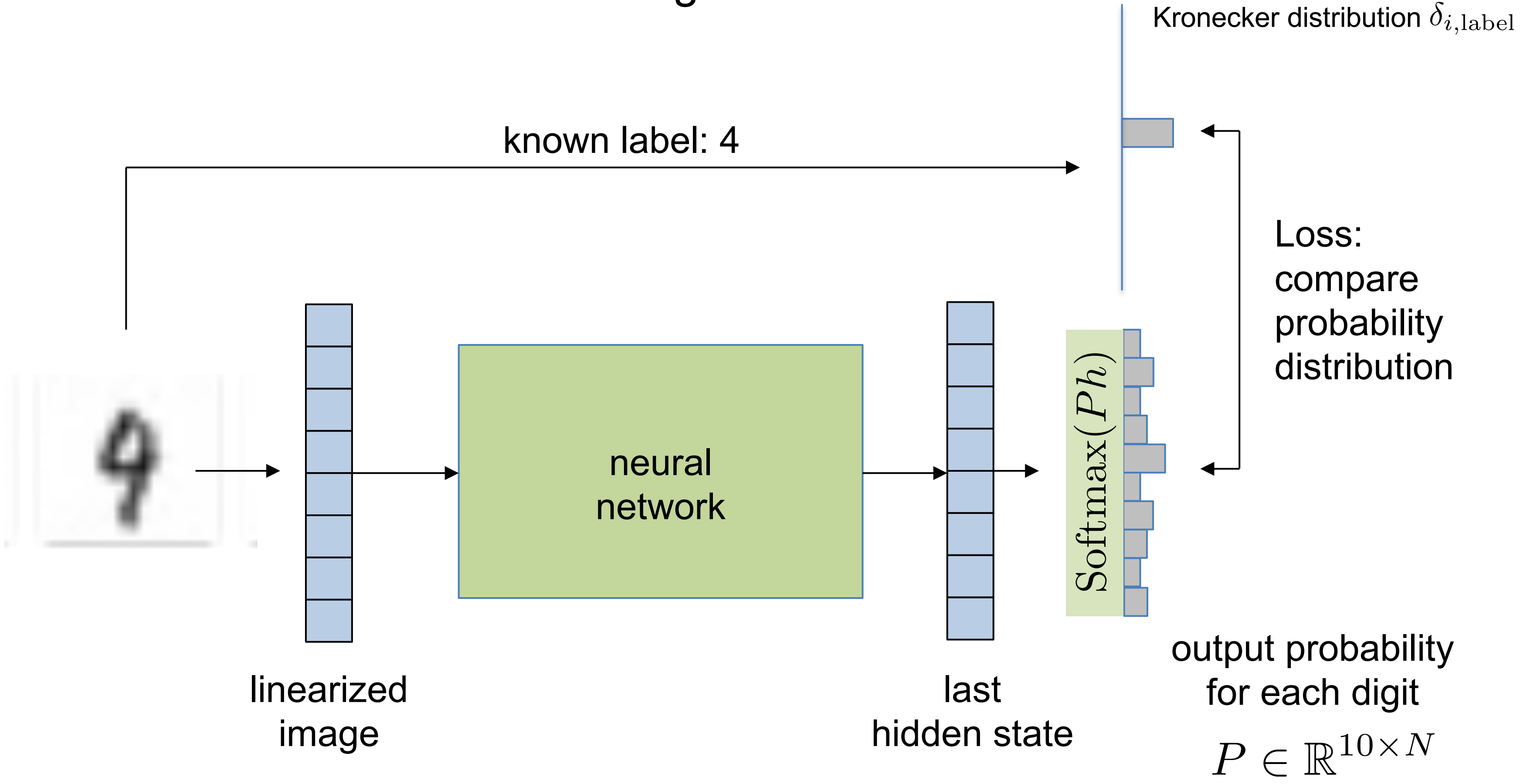
- Training: how can we train the probability distribution—we only have one label for each image?



Kronecker distribution $\delta_{i,\mathrm{label}}$
known label: 4
neural network
$\mathrm{Softmax}(Ph)$
linearized image
last hidden state
output probability for each digit
$P \in \mathbb{R}^{10\times N}$

Simple Example: MNIST

- Training: how can we train the probability distribution—we only have one label for each image?


Kronecker distribution $\delta_{i,\text{label}}$
known label: 4
Loss: compare probability distribution
neural network
$\text{Softmax}(Ph)$
linearized image
last hidden state
output probability for each digit
$P \in \mathbb{R}^{10\times N}$

Simple Example: MNIST

- Cross-entropy loss: the workhorse for discrete generative models

$$\mathrm{Loss}(\mathrm{label}, x) = \log\left(\frac{e^{x_{\mathrm{label}}}}{\sum_i e^{x_i}}\right)$$

Simple Example: MNIST

- Cross-entropy between discrete prob. distributions p, q

$$H(p,q) = -\mathrm{E}_p\big[-\log q\big]$$

Simple Example: MNIST

- Cross-entropy between discrete prob. distributions p, q

$$H(p,q) = -\mathrm{E}_p\big[-\log q\big]$$

From information theory: roughly, discrepancy between q (observed) and p (true) in bits

(can also be written in terms of Kullback-Leibler divergence)

Simple Example: MNIST

- Cross-entropy between discrete prob. distributions p, q

$$H(p,q) = -\mathrm{E}_p\big[-\log q\big]$$

$$= \sum_x p(x)\log\big(q(x)\big) \quad \text{(def of expectation)}$$

Simple Example: MNIST

- Cross-entropy between discrete prob. distributions p, q

$$
\begin{aligned}
H(p,q) &= -\mathrm{E}_p\big[-\log q\big] \\
&= \sum_x p(x)\log\big(q(x)\big) \quad \text{(def of expectation)} \\
&= \sum_x \delta_{x,\text{label}}\log\big(q(x)\big) \quad \text{(Kronecker dist for label)}
\end{aligned}
$$

Simple Example: MNIST

- Cross-entropy between discrete prob. distributions p, q

$$
\begin{aligned}
H(p,q) &= -\mathrm{E}_p\big[-\log q\big] \\
&= \sum_x p(x)\log\big(q(x)\big) \quad \text{(def of expectation)} \\
&= \sum_x \delta_{x,\text{label}}\log\big(q(x)\big) \quad \text{(Kronecker dist for label)} \\
&= \log\big(q(x_{\text{label}})\big) \quad \text{(Properties of Kronecker dist)}
\end{aligned}
$$

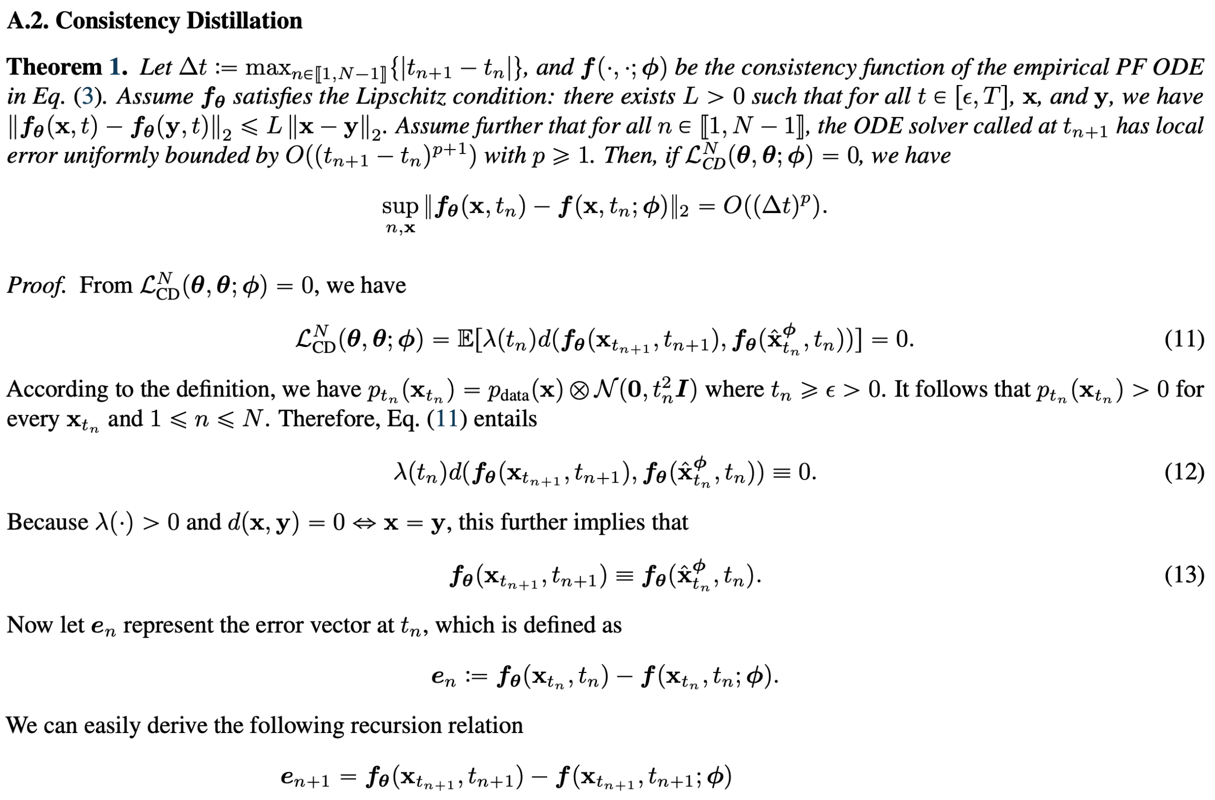
Simple Example: MNIST

- Cross-entropy between discrete prob. distributions p, q

$$
\begin{aligned}
H(p,q) &= -\mathrm{E}_p\big[-\log q\big] \\
&= \sum_x p(x)\log\big(q(x)\big) \quad \text{(def of expectation)} \\
&= \sum_x \delta_{x,\mathrm{label}}\log\big(q(x)\big) \quad \text{(Kronecker dist for label)} \\
&= \log\big(q(x_{\mathrm{label}})\big) \quad \text{(Properties of Kronecker dist)} \\
&= \log\left(\frac{e^{x_{\mathrm{label}}}}{\sum_i e^{x_i}}\right) \quad \text{(q is given by softmax)}
\end{aligned}
$$

Simple Example: MNIST

- Mathematically deep subject:



A.2. Consistency Distillation

Theorem 1. *Let* $\Delta t := \max_{n \in [\![1, N-1]\!]}\{|t_{n+1} - t_n|\}$, *and* $\boldsymbol{f}(\cdot, \cdot; \boldsymbol{\phi})$ *be the consistency function of the empirical PF ODE in Eq. (3). Assume* $\boldsymbol{f}_{\boldsymbol{\theta}}$ *satisfies the Lipschitz condition: there exists* $L > 0$ *such that for all* $t \in [\epsilon, T]$, $\mathbf{x}$, *and* $\mathbf{y}$, *we have* $\|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) - \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{y}, t)\|_2 \leqslant L\|\mathbf{x} - \mathbf{y}\|_2$. *Assume further that for all* $n \in [\![1, N-1]\!]$, *the ODE solver called at* t_{n+1} *has local error uniformly bounded by* $O((t_{n+1} - t_n)^{p+1})$ *with* $p \geqslant 1$. *Then, if* $\mathcal{L}_{\text{CD}}^N(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, *we have*

$$\sup_{n, \mathbf{x}} \|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t_n) - \boldsymbol{f}(\mathbf{x}, t_n; \boldsymbol{\phi})\|_2 = O((\Delta t)^p).$$

Proof. From $\mathcal{L}_{\text{CD}}^N(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, we have

$$\mathcal{L}_{\text{CD}}^N(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = \mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n))] = 0. \tag{11}$$

According to the definition, we have $p_{t_n}(\mathbf{x}_{t_n}) = p_{\text{data}}(\mathbf{x}) \otimes \mathcal{N}(\mathbf{0}, t_n^2 \boldsymbol{I})$ where $t_n \geqslant \epsilon > 0$. It follows that $p_{t_n}(\mathbf{x}_{t_n}) > 0$ for every $\mathbf{x}_{t_n}$ and $1 \leqslant n \leqslant N$. Therefore, Eq. (11) entails

$$\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n)) \equiv 0. \tag{12}$$

Because $\lambda(\cdot) > 0$ and $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$, this further implies that

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) \equiv \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n). \tag{13}$$

Now let $\boldsymbol{e}_n$ represent the error vector at t_n, which is defined as

$$\boldsymbol{e}_n := \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n) - \boldsymbol{f}(\mathbf{x}_{t_n}, t_n; \boldsymbol{\phi}).$$

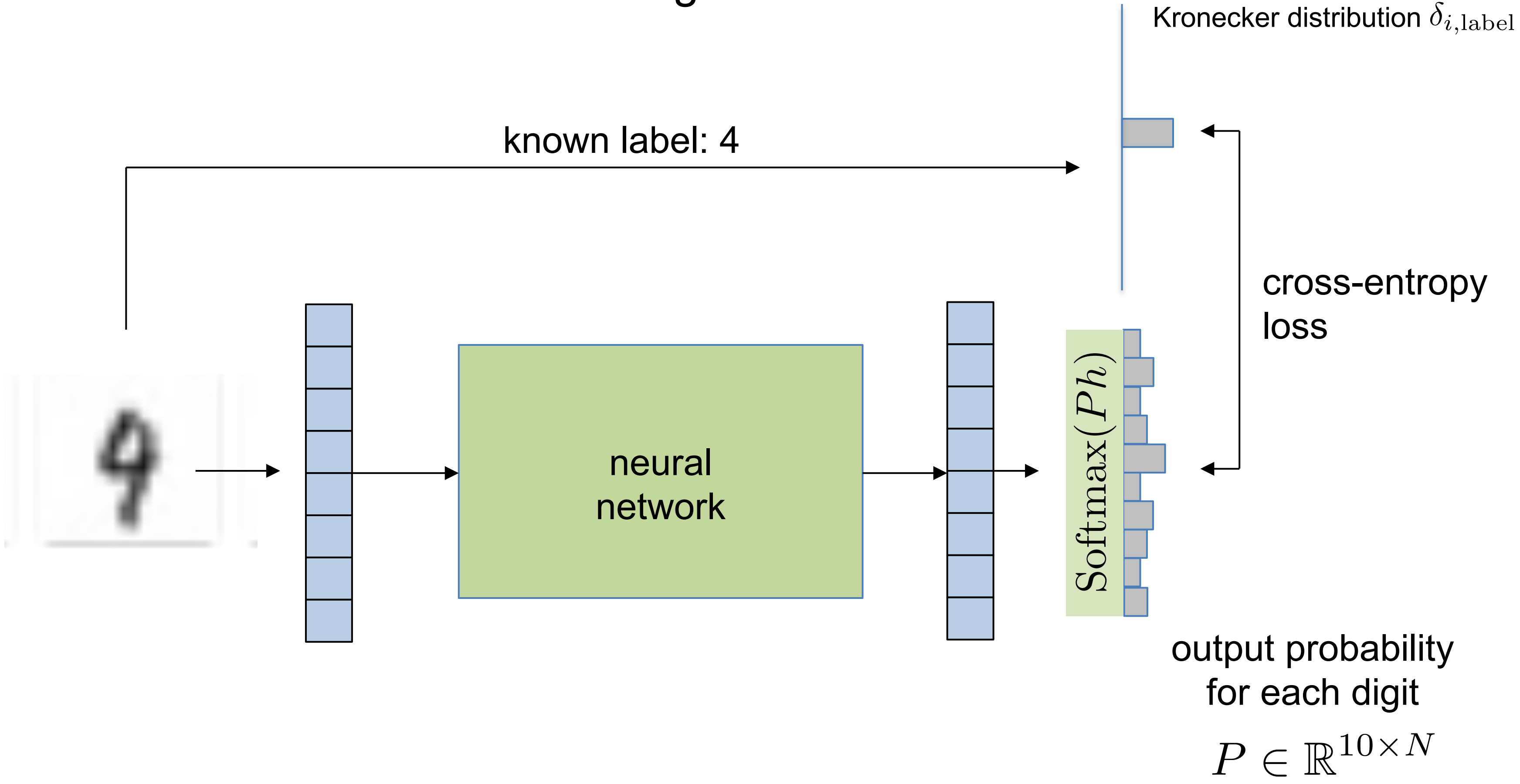
We can easily derive the following recursion relation

$$\boldsymbol{e}_{n+1} = \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \boldsymbol{f}(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi})$$

Song et al., Consistency Models, 2023, https://arxiv.org/pdf/2303.01469.pdf

Simple Example: MNIST

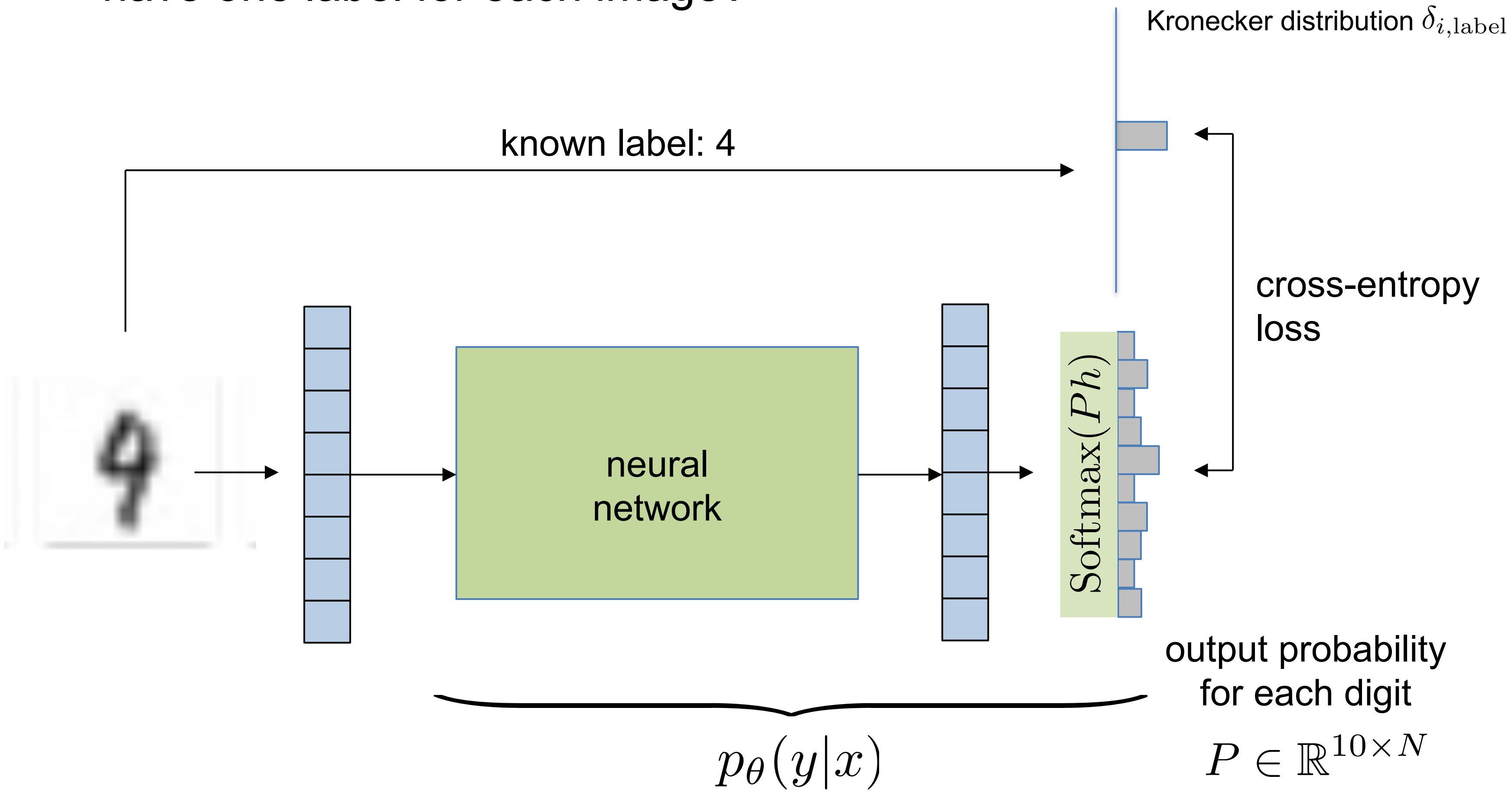
- Training: how can we train the probability distribution—we only have one label for each image?



Kronecker distribution $\delta_{i,\text{label}}$
known label: 4
cross-entropy loss
neural network
$\text{Softmax}(Ph)$
output probability for each digit
$P \in \mathbb{R}^{10\times N}$

Simple Example: MNIST

- Training: how can we train the probability distribution—we only have one label for each image?


Kronecker distribution $\delta_{i,\text{label}}$
known label: 4
cross-entropy loss
neural network
$\text{Softmax}(Ph)$
output probability for each digit
$p_\theta(y|x)$
$P \in \mathbb{R}^{10 \times N}$

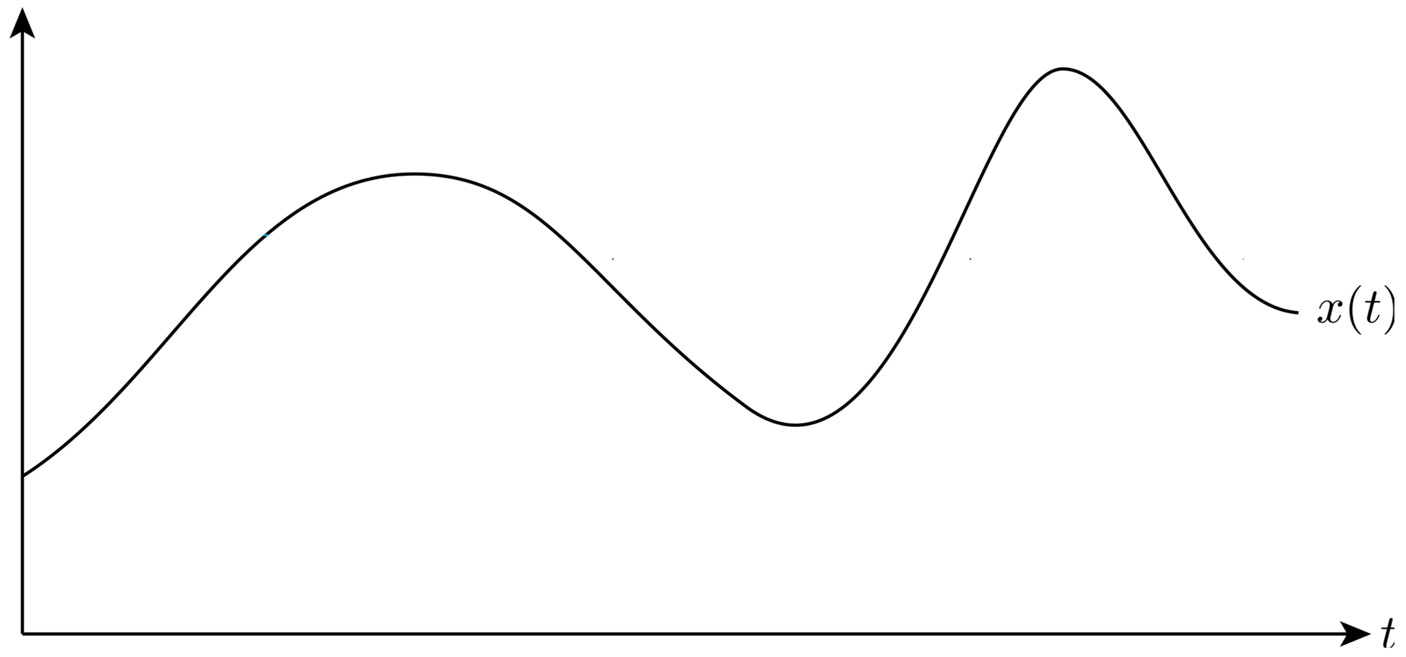
Simple Example: MNIST

- But how can we learn a non-degenerate distribution p when we we only have degenerate labels?

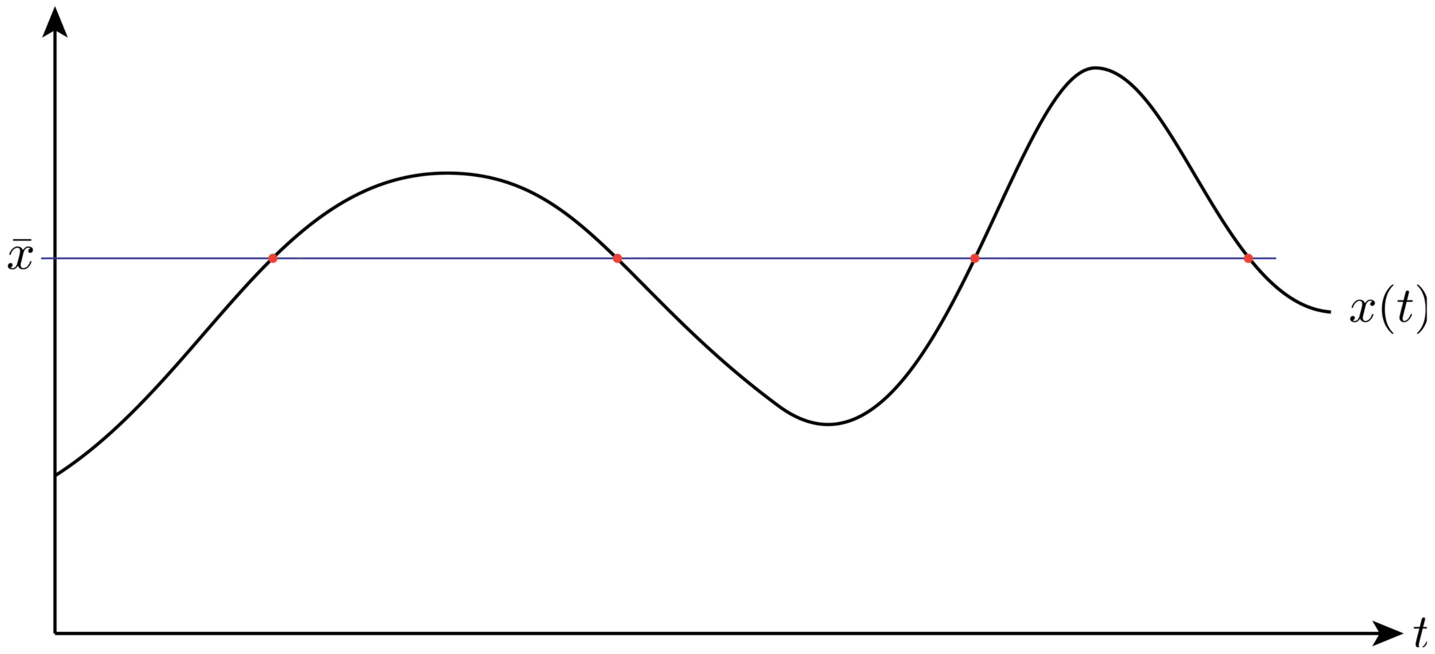
Simple Example: MNIST

- But how can we learn a non-degenerate distribution p when we we only have degenerate labels?
- Training:
  - Monte Carlo approximation of expected value over data distribution (as approximated by training data set)
  - Training data contain sufficient number of similar images with different labels. Overall loss/expectation is minimized by learning well calibrated probability distribution

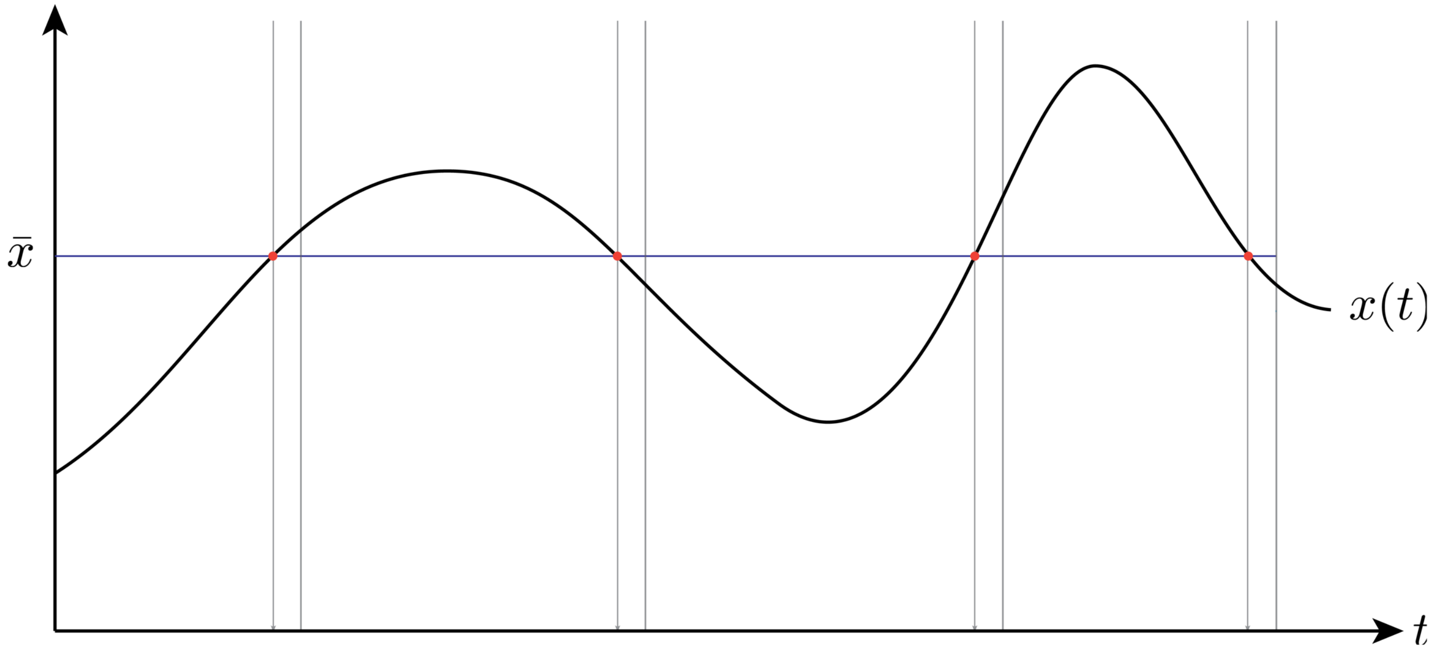
Simple Example: MNIST



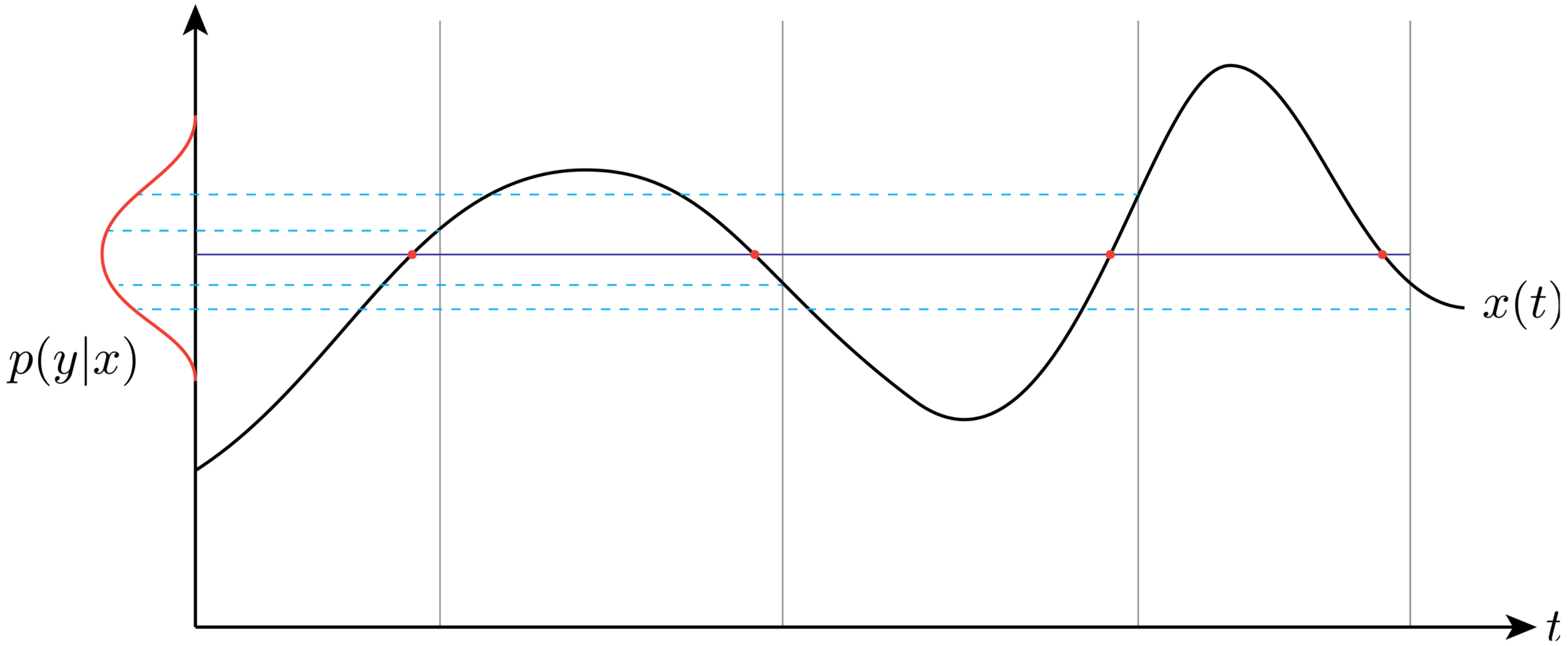
Simple Example: MNIST



Simple Example: MNIST

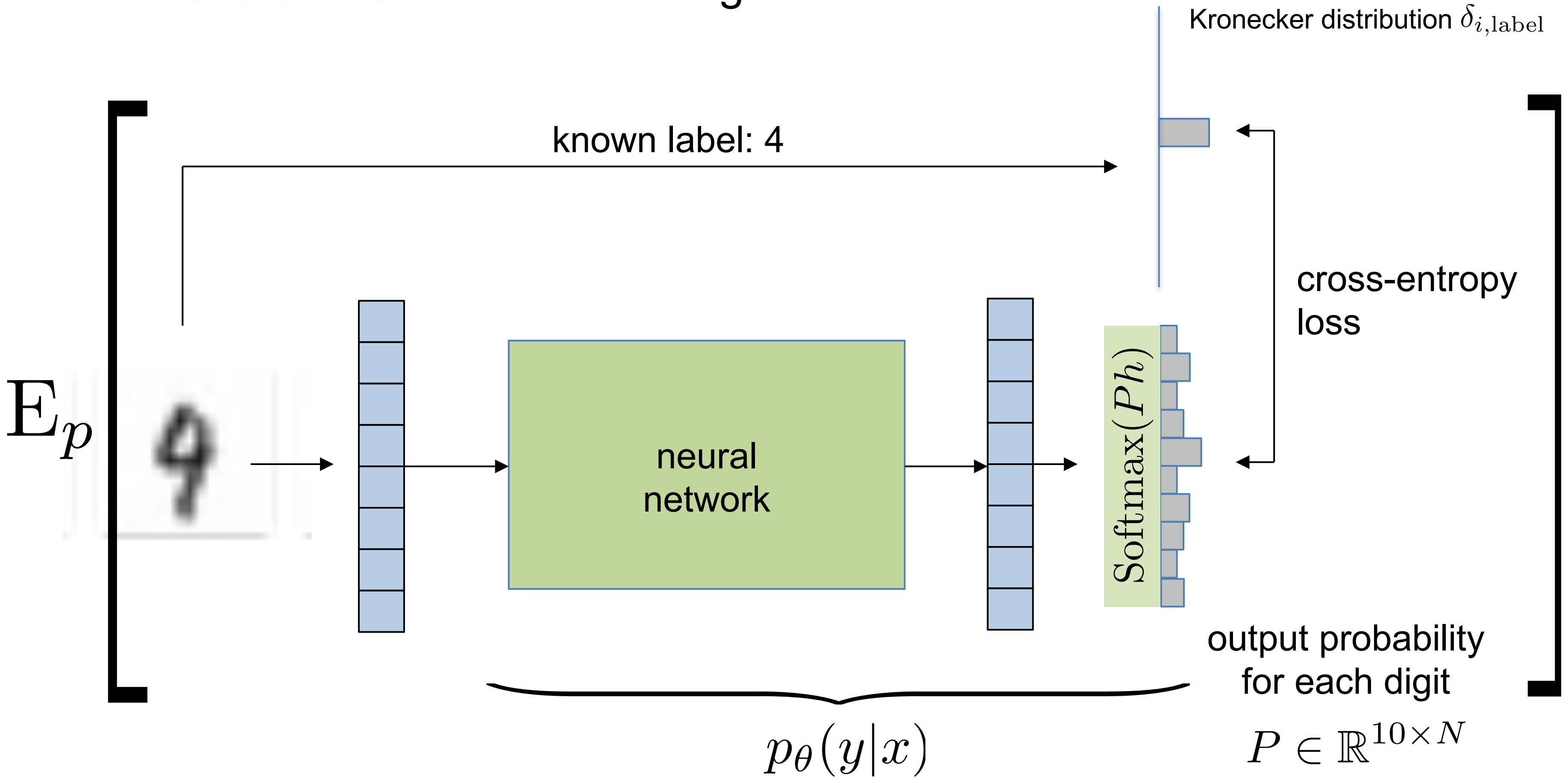


Simple Example: MNIST



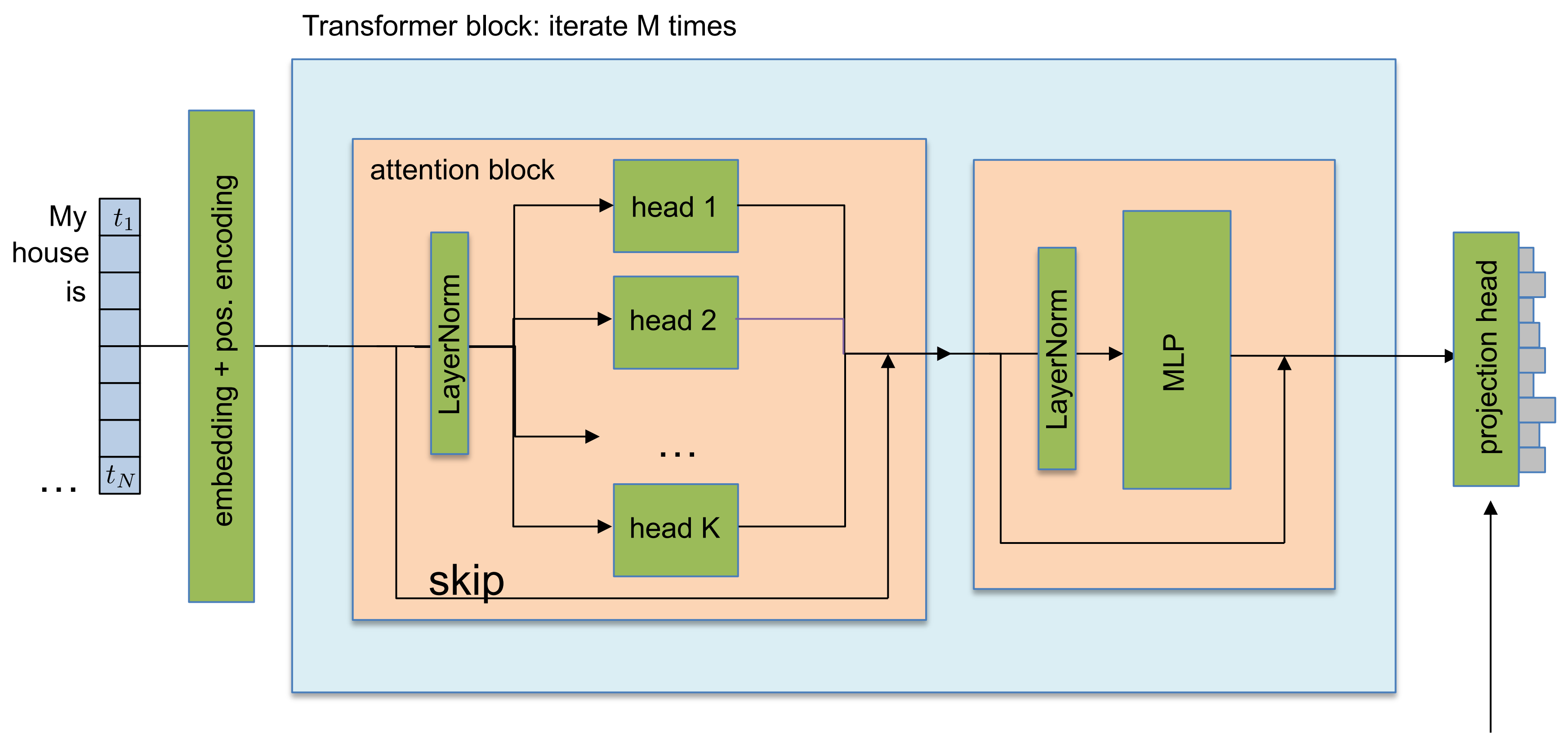
Simple Example: MNIST

- Training: how can we train the probability distribution—we only have one label for each image?



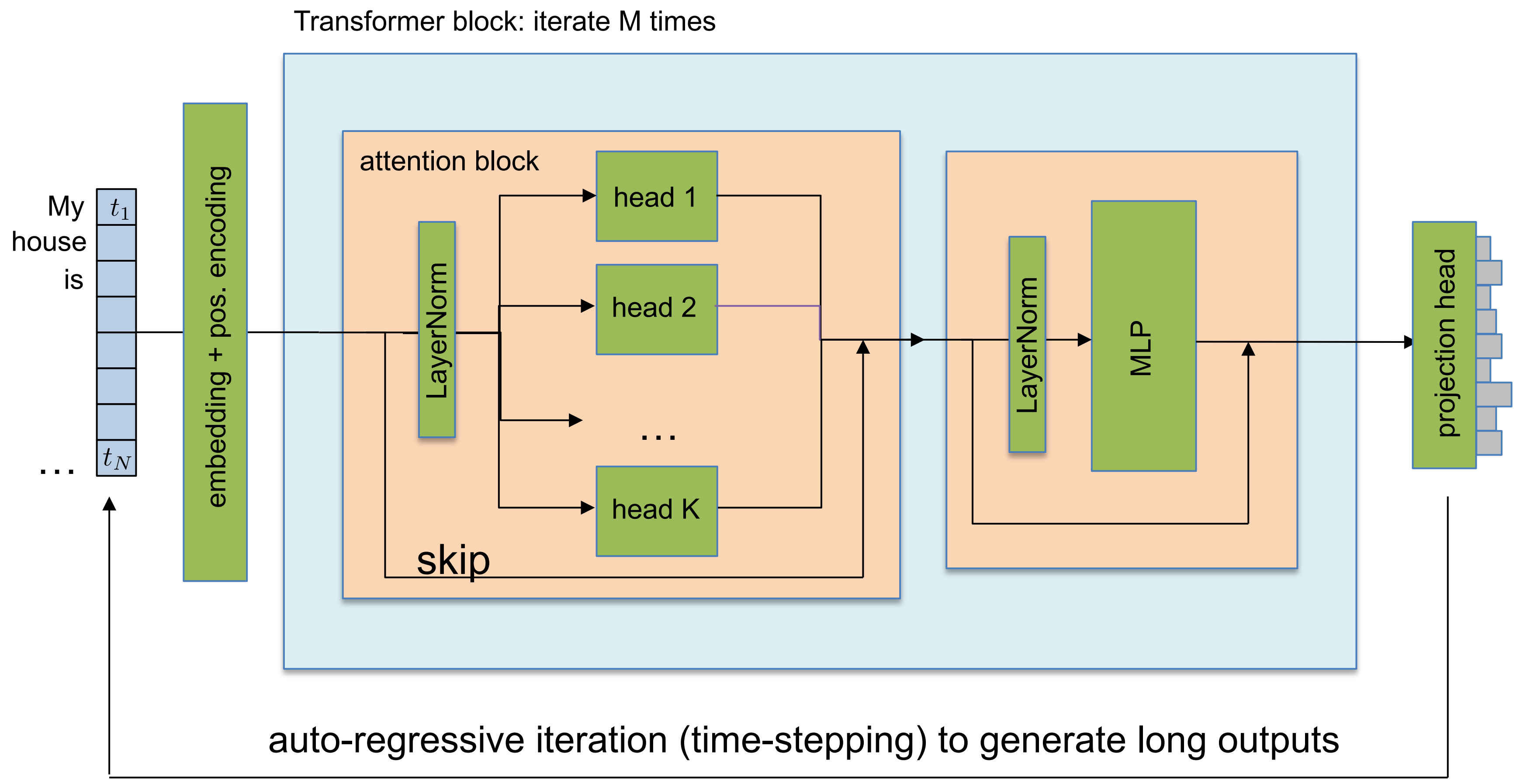
Kronecker distribution $\delta_{i,\mathrm{label}}$
known label: 4
cross-entropy loss
E_p
neural network
$\mathrm{Softmax}(Ph)$
output probability for each digit
$p_\theta(y|x)$
$P \in \mathbb{R}^{10\times N}$

Large language models

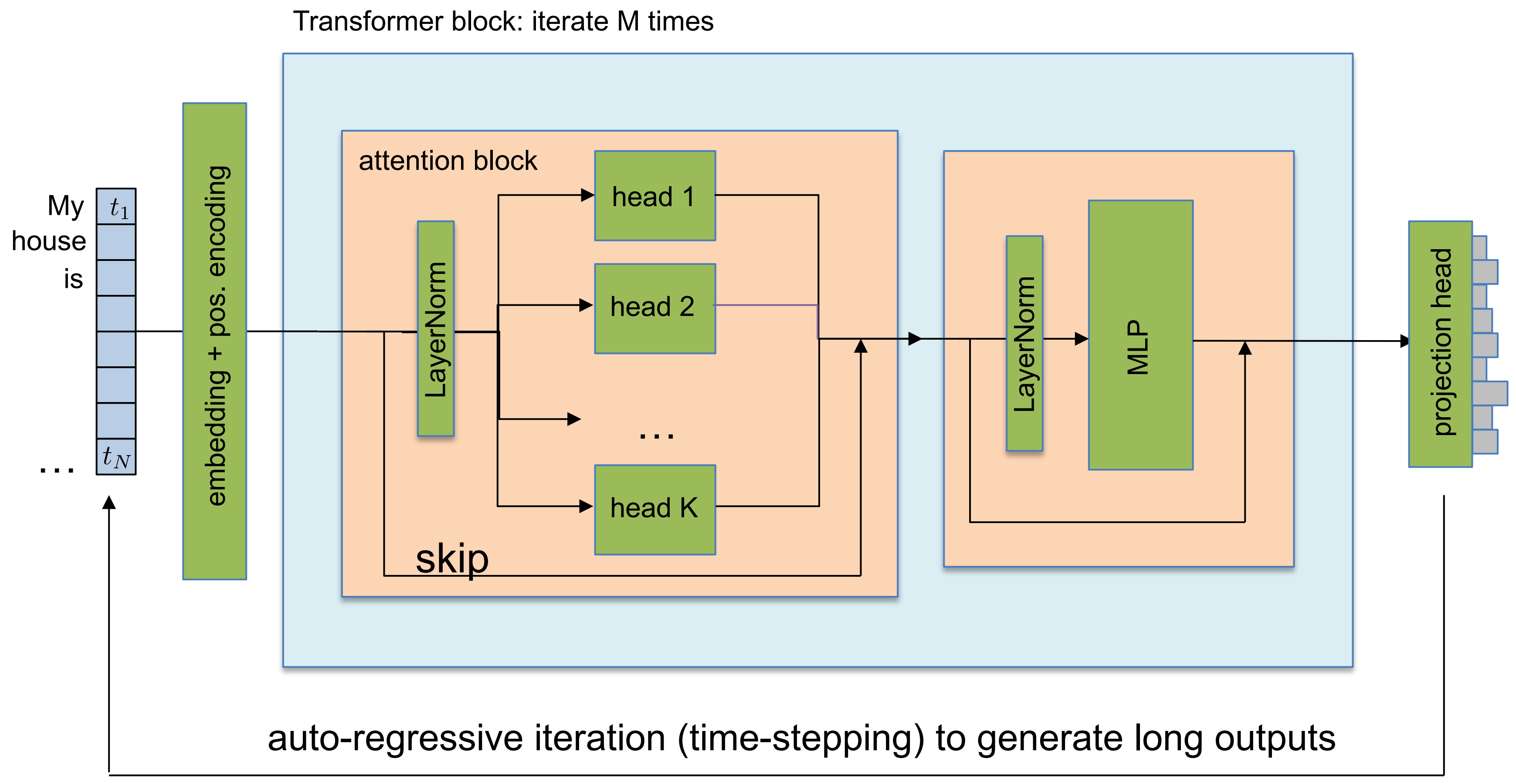


output probability for each word in the corpus (100k in state-of-the-art models)

Large language models


Transformer block: iterate M times
My
house
is
...
t_1
t_N
embedding + pos. encoding
attention block
LayerNorm
head 1
head 2
...
head K
skip
LayerNorm
MLP
projection head
auto-regressive iteration (time-stepping) to generate long outputs

Large language models



auto-regressive iteration (time-stepping) to generate long outputs

fine-tuning (e.g. human feedback reinforcement learning) to obtain well calibrated probability distribution

But why probabilistic modeling?

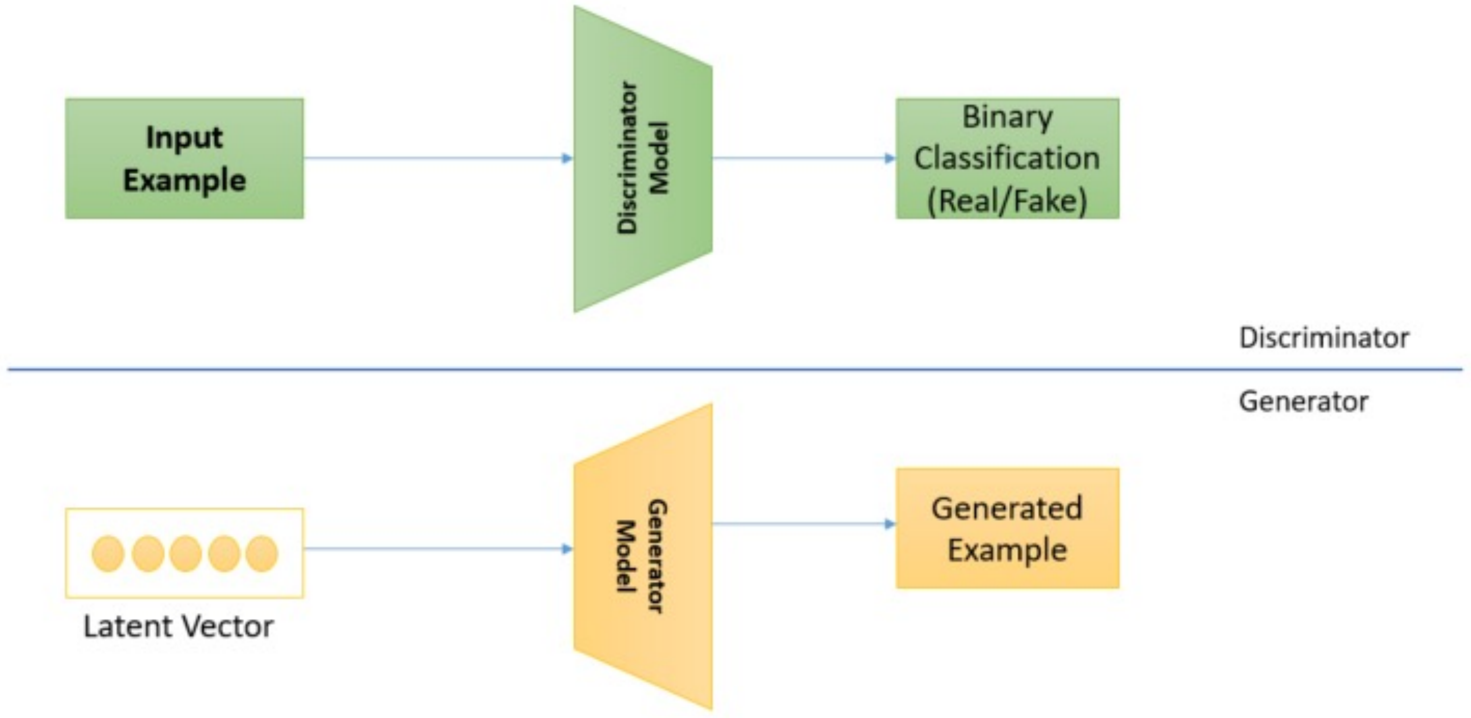
- Essentially all poster child applications of deep learning are probabilistic models
    - Natural language processing and computer vision are theories without useful conventional models
- Probabilistic models are well suited for problems where ambiguity or uncertainty or stochasticity play a role
    - Can be made rigorous using Bayesian modeling and expectation maximization

Generative adversarial networks (GANs)

Generative adversarial networks (GANs)

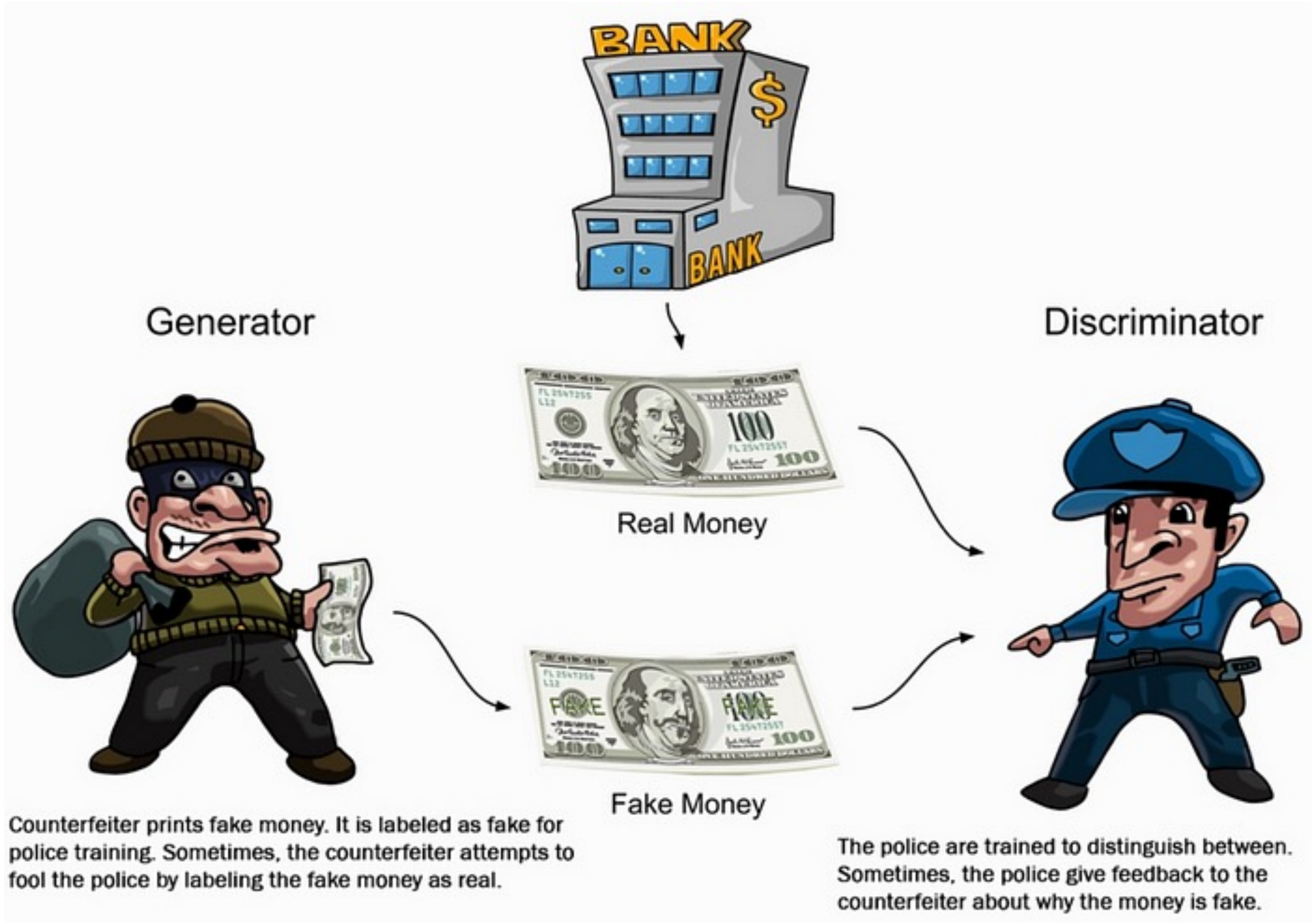
Deep learning model employing unsupervised learning with the aim of discovering hidden patterns within input data and generate realistic output.

Composed of two competing sub-models:
- the **Generator** (G) learns to generate realistic-looking (fake) samples
- the **Discriminator** (D) learns to distinguish fakes from real samples



https://towardsai.net/p/l/the-intuition-behind-gans-for-beginners

Generative adversarial networks (GANs)


BANK
BANK
Generator
Discriminator
Real Money
Fake Money
FAKE
FAKE
Counterfeiter prints fake money. It is labeled as fake for police training. Sometimes, the counterfeiter attempts to fool the police by labeling the fake money as real.
The police are trained to distinguish between. Sometimes, the police give feedback to the counterfeiter about why the money is fake.

https://towardsdatascience.com/the-math-behind-gans-generative-adversarial-networks-3828f3469d9c

GAN training: It’s all about competition !

Two-player **minimax game** with a value function V(D, G):

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

G aims to minimize V(D,G) while D tries to maximize it. The game stops when Nash Equilibrium is reached.

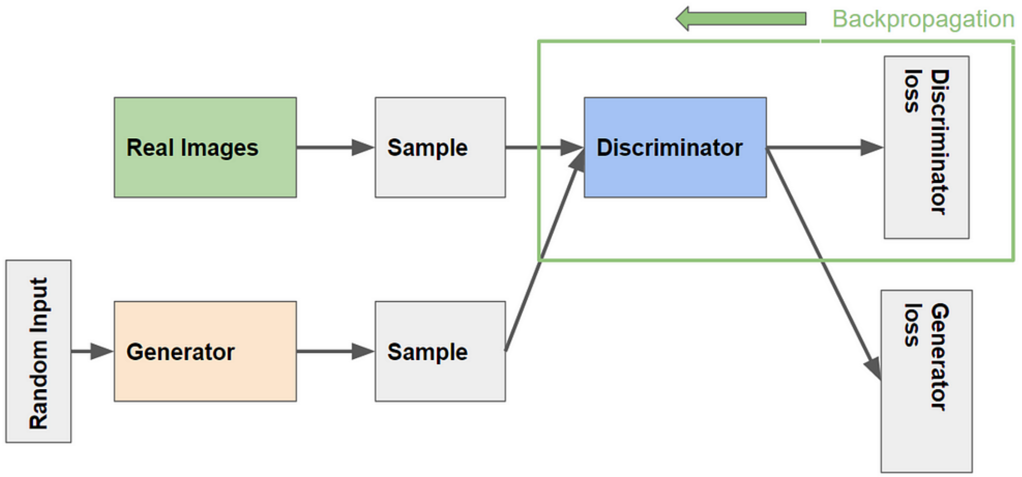
Some notations ...

$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}$	Expected value over all real instances.
$D(\boldsymbol{x})$	Discriminator score on the real instance.
$\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}$	Expected value over all fake instances.
$G(\boldsymbol{z})$	Generator output given a random noise.
$D(G(\boldsymbol{z}))$	Discriminator score on the fake instance.

https://arxiv.org/abs/1406.2661

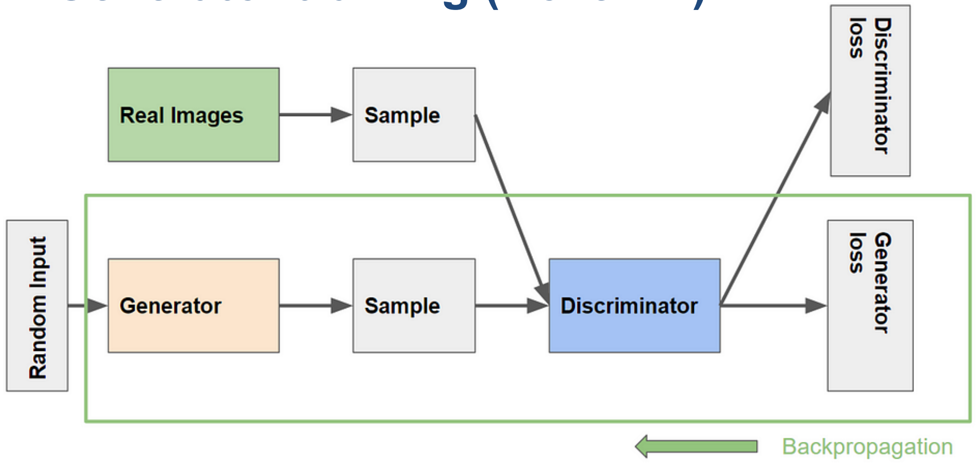
GAN training: It’s all about competition! (but also, collaboration)

Discriminator training (frozen G)



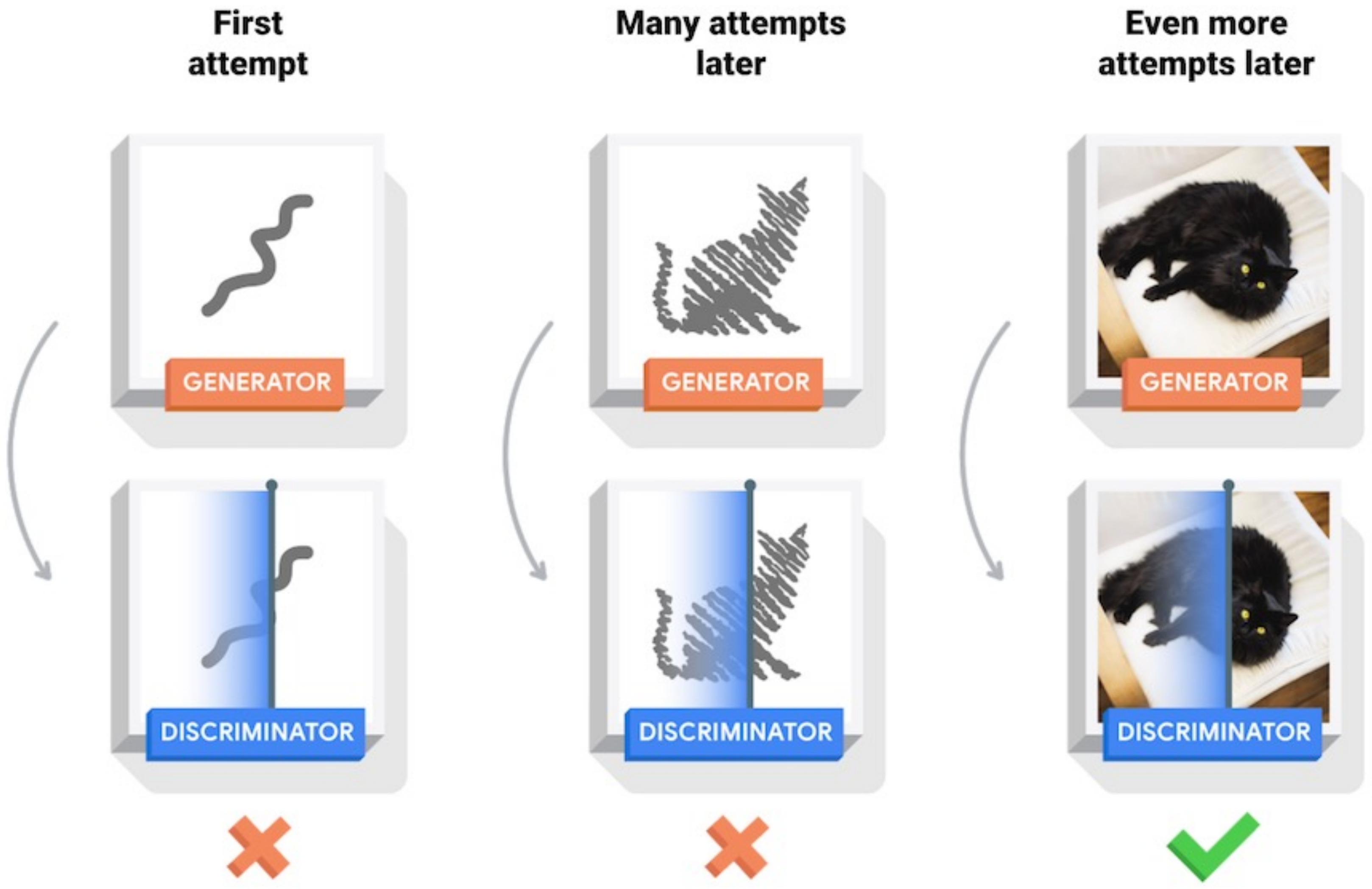
- G is frozen during training step of D.
- D classifies both real and fake data and is penalized for incorrect classification.

Generator training (frozen D)



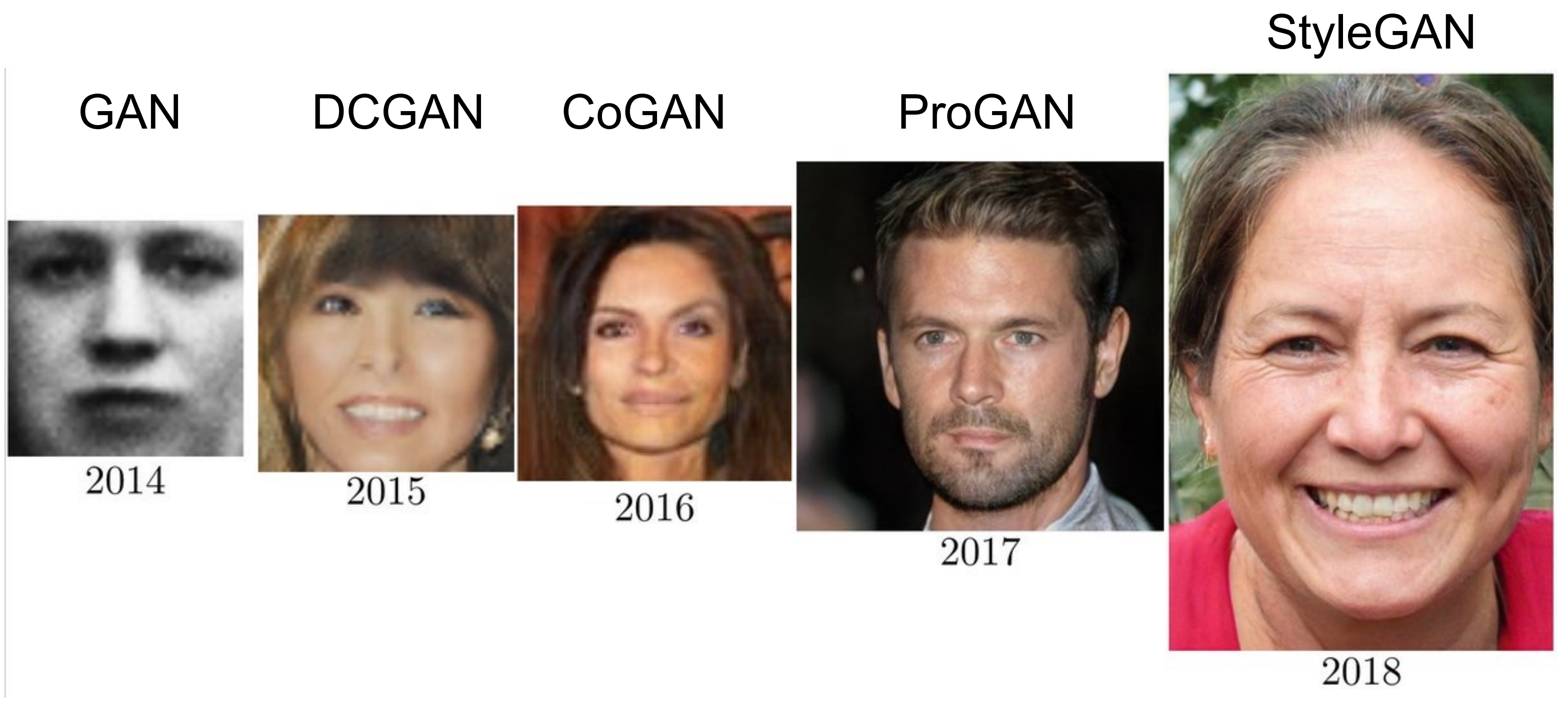
- D is frozen during the generator's G training step.
- Gradients from the discriminator are still passed to the generator.
- Tight collaboration ensures G learns from D's feedback.
- The generator is penalized for producing samples correctly classified by D.

https://arxiv.org/abs/1406.2661
https://towardsai.net/p/l/the-intuition-behind-gans-for-beginners

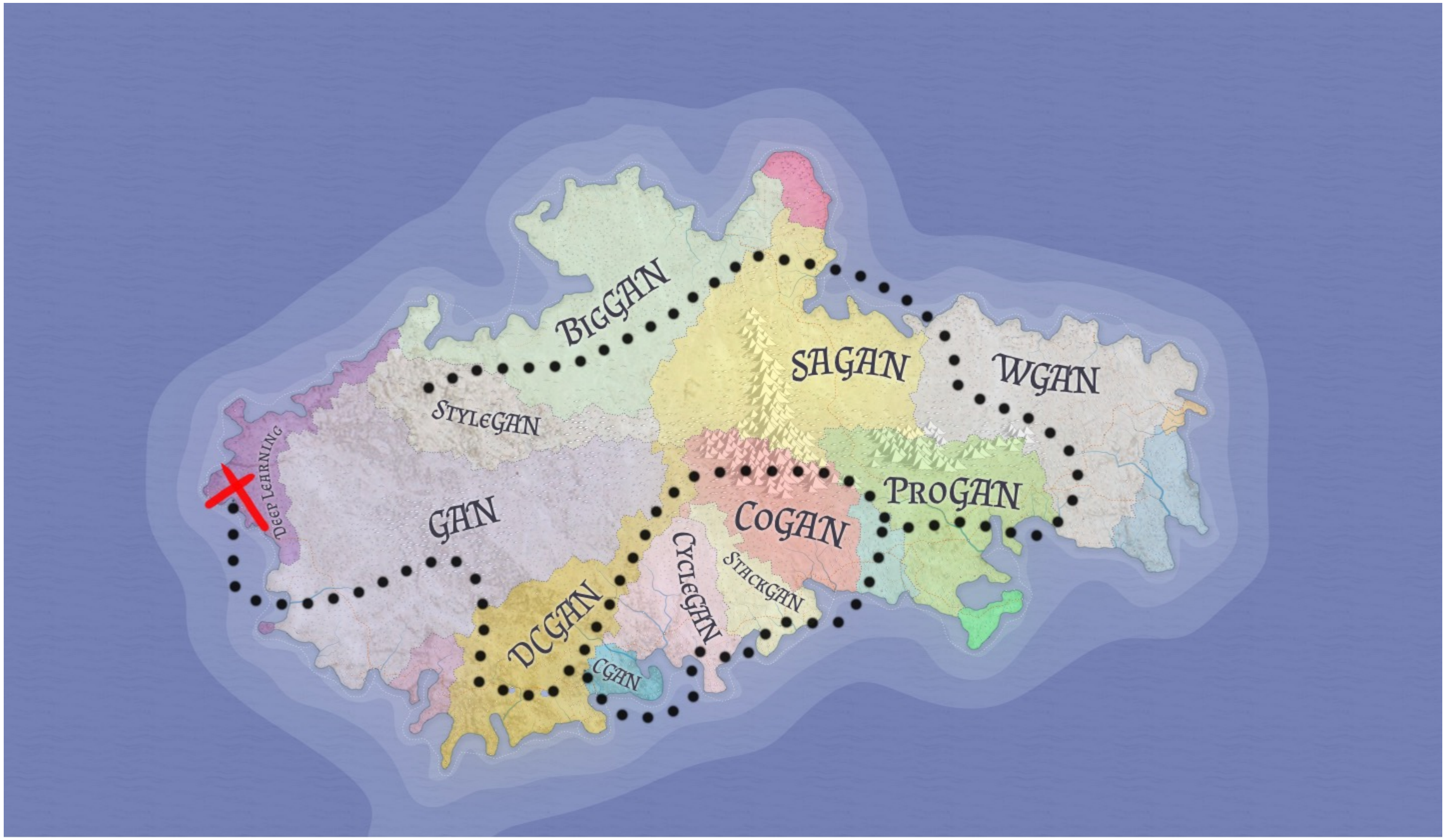


https://www.tensorflow.org/tutorials/generative/dcgan

Rapid evolution in image quality



A (brief) GAN roadmap…



https://blog.floydhub.com/gans-story-so-far/

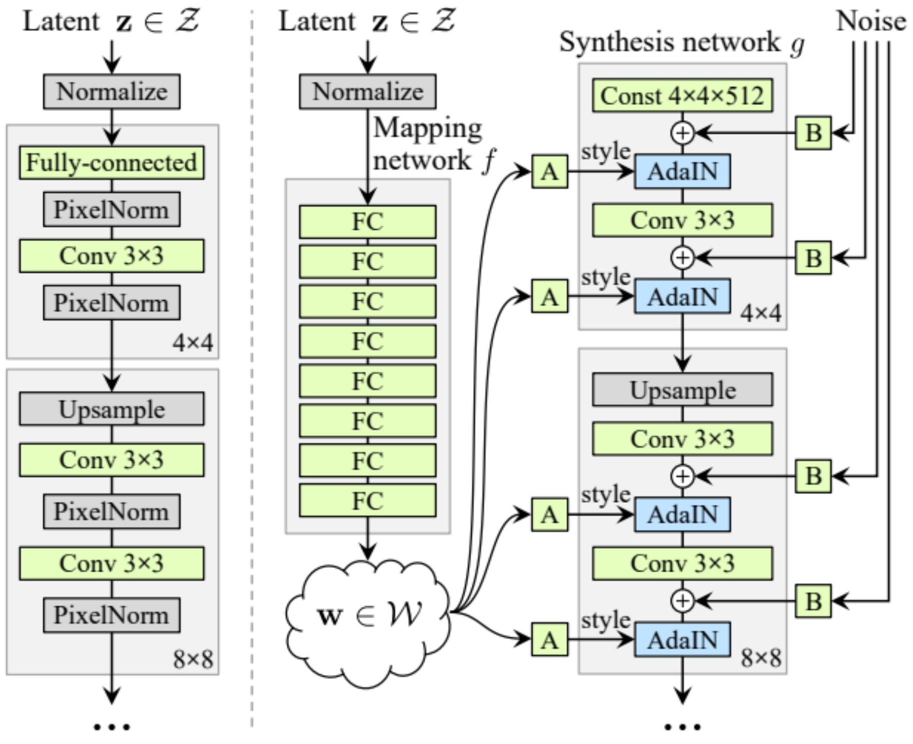
Developments in the first years …

- **2014:** Introduction by Ian Goodfellow and colleagues **Goodfellow et al. 2014**
- **2014:** Introduction of the **Conditional GAN** by **Mirza et al 2014**
- **2015:** Introduction of deep convolutional networks for both generator and discriminator by **Radford et al. 2015**
- **2017:** Introduction of **Wasserstein GAN (WGAN)** by **Arjovsky et al. 2017**
  - Introduction of the Wasserstein Loss Function based on the **Wasserstein distance**, also known as Earth Mover's distance (EM distance), as **a measure of the difference between the generated and real data distributions.**
  - Introduction of **Lipschitz continuity constraint on Discriminator** with Gradient Penalty. Alternative weight normalisation for Lipschitz constraint explored by **Miyato et al. 2018** in SNGAN using spectral normalization.
  - Greater training stability and improved Generator convergence
- **2017:** Introduction of **CycleGAN** for image-to-image translation by **Zhu et al. 2017**
- **2018:** Introduction of **BigGAN** by **Brock et al. 2018**
  - **Massive scale**, both in terms of **model size** and **dataset** used for training (ImageNet). Increasing the scale of GANs could lead to improved image quality and diversity.

StyleGAN Series (SOTA)

- **2017:** Introduction of **ProGAN** by Karras et al. 2018.
  - Progressive growing

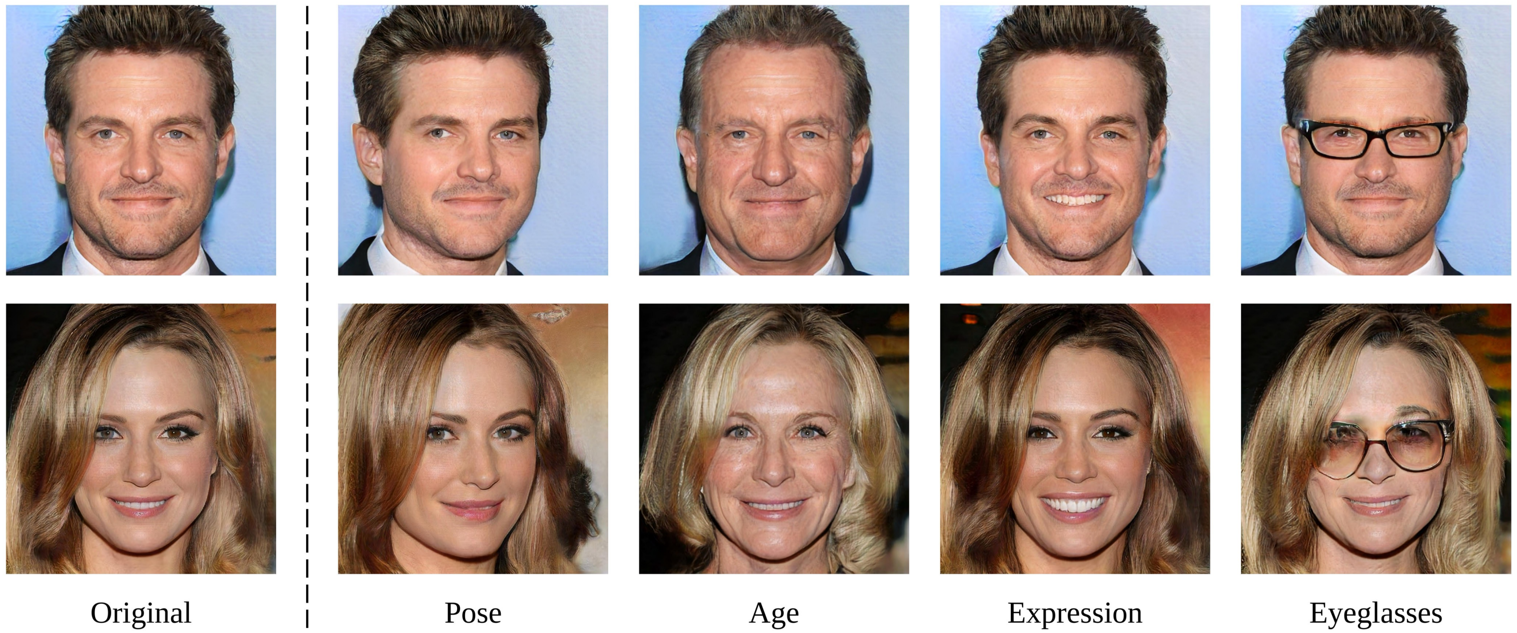
- **2018 - 2019 - 2021:** Introduction of **StyleGAN**, **StyleGAN2** and **StyleGAN3** by Karras et al. 2018, Karras et al. 2019 and Karras et al. 2021 .
  - Style transfer at different resolution levels
  - **Automatically learned, unsupervised separation of high-level attributes**
  - Introduction of an **intermediate latent space 'W'** that allows better **disentanglement of the latent factors of variation.**



(a) Traditional (b) Style-based generator

StyleGAN Series (SOTA)

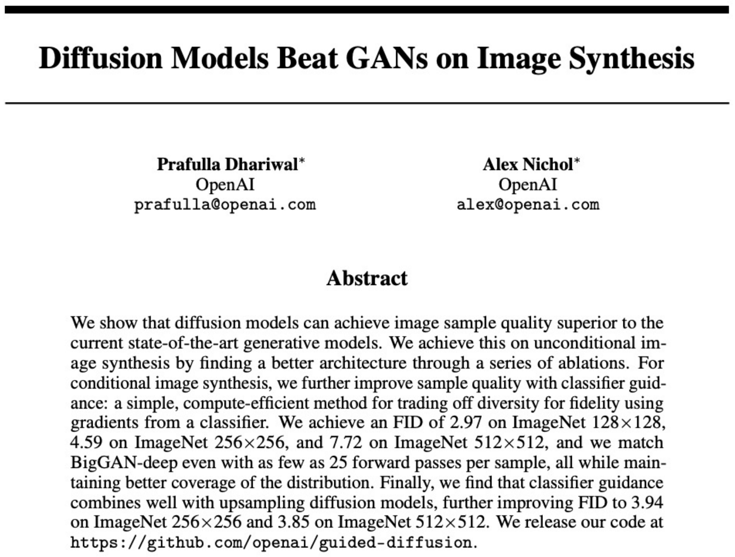
- The separation of different resolution attributes and the disentanglement of the latent factors of variation enables impressive image-editing applications:



InterFaceGAN by [Shen et al. 2020]

Check also the works of [Alaluf et al. 2022] and previous works by the same authors

Bad news for GANs in 2021…



Diffusion Models Beat GANs on Image Synthesis

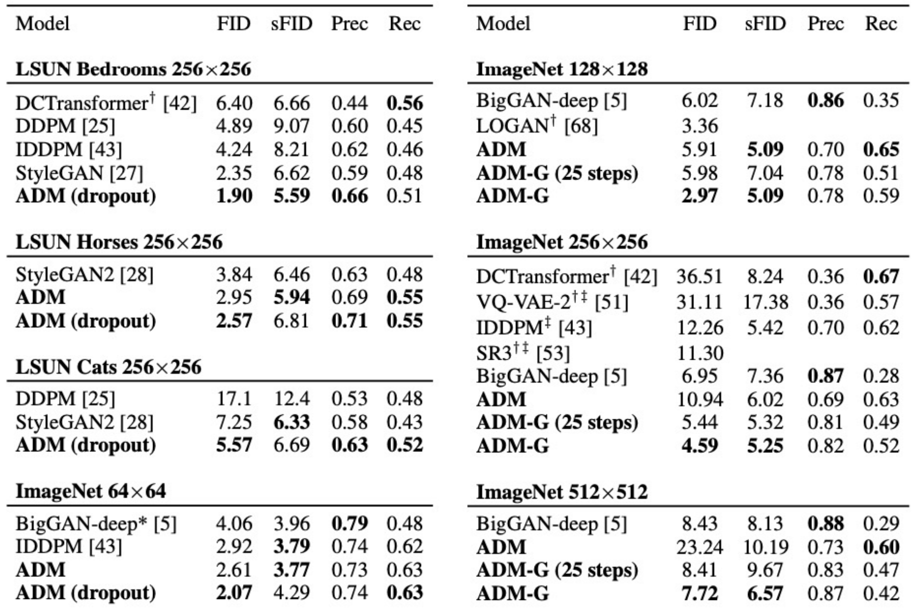
Prafulla Dhariwal* OpenAI prafulla@openai.com

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.

Pretty convincing results…



Model	FID	sFID	Prec	Rec
LSUN Bedrooms 256×256				
DCTransformer† [42]	6.40	6.66	0.44	**0.56**
DDPM [25]	4.89	9.07	0.60	0.45
IDDPM [43]	4.24	8.21	0.62	0.46
StyleGAN [27]	2.35	6.62	0.59	0.48
ADM (dropout)	**1.90**	**5.59**	**0.66**	0.51
LSUN Horses 256×256				
StyleGAN2 [28]	3.84	6.46	0.63	0.48
ADM	2.95	**5.94**	0.69	**0.55**
ADM (dropout)	**2.57**	6.81	**0.71**	**0.55**
LSUN Cats 256×256				
DDPM [25]	17.1	12.4	0.53	0.48
StyleGAN2 [28]	7.25	**6.33**	0.58	0.43
ADM (dropout)	**5.57**	6.69	**0.63**	**0.52**
ImageNet 64×64				
BigGAN-deep* [5]	4.06	3.96	**0.79**	0.48
IDDPM [43]	2.92	**3.79**	0.74	0.62
ADM	2.61	**3.77**	0.73	0.63
ADM (dropout)	**2.07**	4.29	0.74	**0.63**

Model	FID	sFID	Prec	Rec
ImageNet 128×128				
BigGAN-deep [5]	6.02	7.18	**0.86**	0.35
LOGAN† [68]	3.36			
ADM	5.91	**5.09**	0.70	**0.65**
ADM-G (25 steps)	5.98	7.04	0.78	0.51
ADM-G	**2.97**	**5.09**	0.78	0.59
ImageNet 256×256				
DCTransformer† [42]	36.51	8.24	0.36	**0.67**
VQ-VAE-2†‡ [51]	31.11	17.38	0.36	0.57
IDDPM‡ [43]	12.26	5.42	0.70	0.62
SR3†‡ [53]	11.30			
BigGAN-deep [5]	6.95	7.36	**0.87**	0.28
ADM	10.94	6.02	0.69	0.63
ADM-G (25 steps)	5.44	5.32	0.81	0.49
ADM-G	**4.59**	**5.25**	0.82	0.52
ImageNet 512×512				
BigGAN-deep [5]	8.43	8.13	**0.88**	0.29
ADM	23.24	10.19	0.73	**0.60**
ADM-G (25 steps)	8.41	9.67	0.83	0.47
ADM-G	**7.72**	**6.57**	0.87	0.42

Key takeaways

'StyleGAN sets new standards for generative modeling regarding image quality and controllability. However, StyleGAN's performance severely degrades on large unstructured datasets such as ImageNet.' [1]

2022: Introduction of **StyleGAN-XL** by [1]

- Scaled StyleGAN3 architecture to ImageNET
- Competitive scores on ImageNet against Diffusion Models

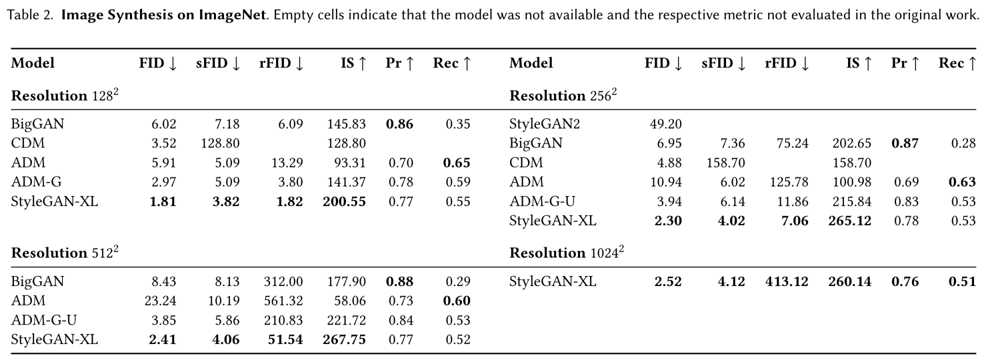


Table 2. **Image Synthesis on ImageNet**. Empty cells indicate that the model was not available and the respective metric not evaluated in the original work.

Model	FID ↓	sFID ↓	rFID ↓	IS ↑	Pr ↑	Rec ↑
Resolution 128^2						
BigGAN	6.02	7.18	6.09	145.83	**0.86**	0.35
CDM	3.52	128.80		128.80		
ADM	5.91	5.09	13.29	93.31	0.70	**0.65**
ADM-G	2.97	5.09	3.80	141.37	0.78	0.59
StyleGAN-XL	**1.81**	**3.82**	**1.82**	**200.55**	0.77	0.55
Resolution 512^2						
BigGAN	8.43	8.13	312.00	177.90	**0.88**	0.29
ADM	23.24	10.19	561.32	58.06	0.73	**0.60**
ADM-G-U	3.85	5.86	210.83	221.72	0.84	0.53
StyleGAN-XL	**2.41**	**4.06**	**51.54**	**267.75**	0.77	0.52

Model	FID ↓	sFID ↓	rFID ↓	IS ↑	Pr ↑	Rec ↑
Resolution 256^2						
StyleGAN2	49.20					
BigGAN	6.95	7.36	75.24	202.65	**0.87**	0.28
CDM	4.88	158.70		158.70		
ADM	10.94	6.02	125.78	100.98	0.69	**0.63**
ADM-G-U	3.94	6.14	11.86	215.84	0.83	0.53
StyleGAN-XL	**2.30**	**4.02**	**7.06**	**265.12**	0.78	0.53
Resolution 1024^2						
StyleGAN-XL	**2.52**	**4.12**	**413.12**	**260.14**	**0.76**	**0.51**

[1] StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets Sauer et al. 2022

Diffusion Models vs GANs in 2024

- Nowadays, **GANs** have fell out of favour due to the incredible success of **Diffusion Models** in **high-quality image synthesis based on text prompts.**
- **Diffusion Models** have been more successful on training against **unstructured and massively diverse, large-scale datasets.**
- **State-of-the-art GANs** can be trained with smaller structured datasets and still deliver good results. Furthermore, the **latent space control** that StyleGANs offer can be an intriguing application.
- **Diffusion Models** offer a **more interpretable and explicit training process.**
- GAN training involves a minmax game between two sub-models and stability and converge can be an issue (depending on dataset).
- GANs main advantage is inference speed (one network pass to generate an image vs multiple needed for denoising). This is changing fast in the favour of Diffusion Models.

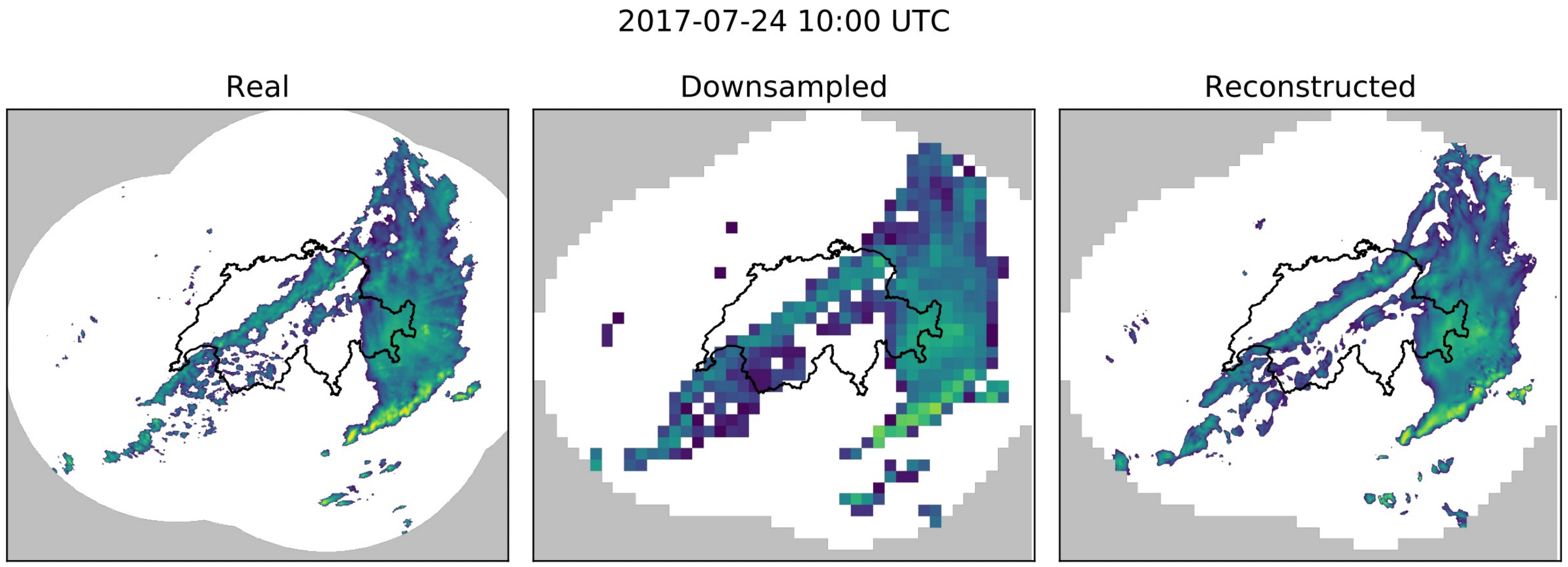
GANs: Applications

Downscaling / Super-Resolution

Stochastic Super-Resolution for Downscaling Time-Evolving Atmospheric Fields with a Generative Adversarial Network

Jussi Leinonen, Daniele Nerini, Alexis Berne in 2019

- *Stochastic super-resolution GAN to generate ensembles of time-evolving high-resolution atmospheric fields for an input consisting of a low-resolution sequence of images of the same field.*



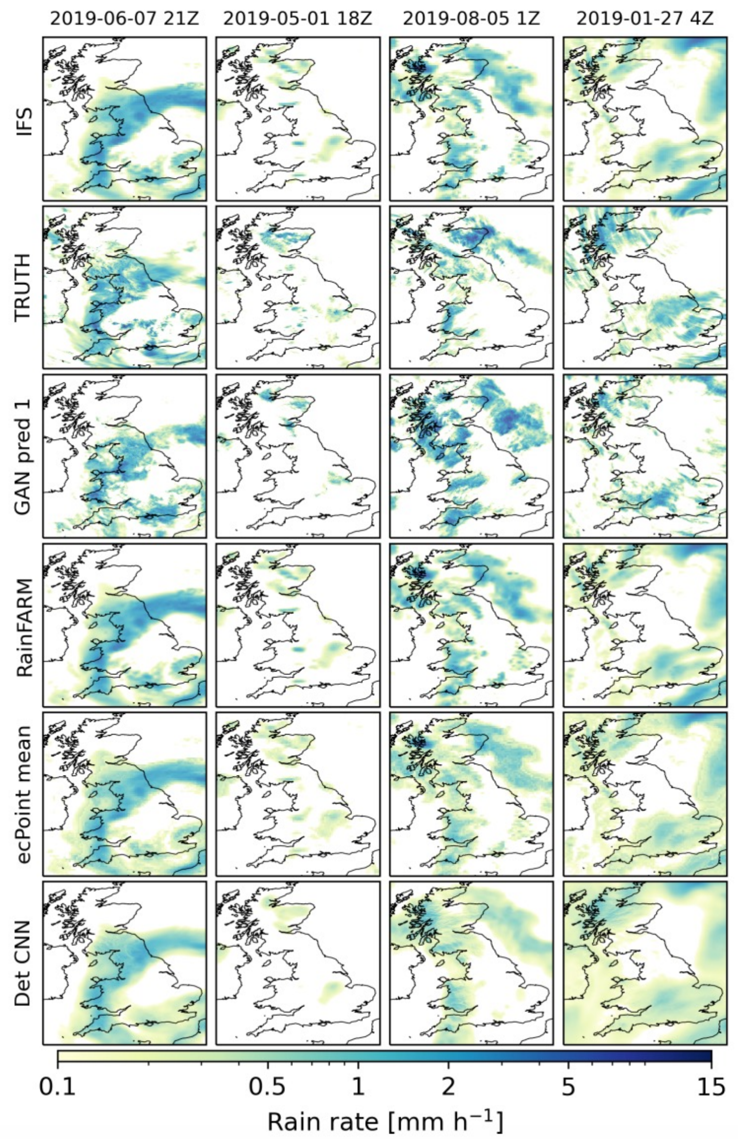
Downscaling / Super-Resolution

A Generative Deep Learning Approach to Stochastic Downscalinç of Precipitation Forecasts

Lucy Harris, Andrew T. T. McRae, Matthew Chantry, Peter D., Tim N. Palmer 2022

- Used GANs to produce ensembles of reconstructed high-resolution precipitation field: increasing the accuracy and resolution of comparatively low-resolution input from a weather forecasting model, using high-resolutic radar measurements as a "ground truth".

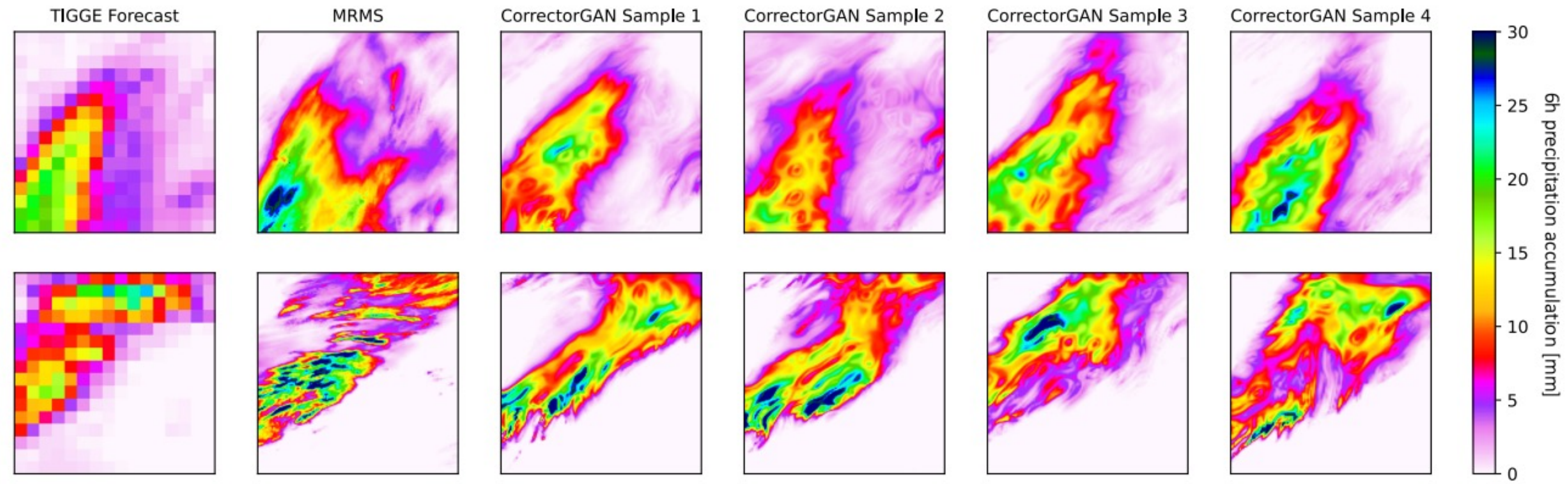
Example predictions for different input conditions



Downscaling / Super-Resolution

Increasing the accuracy and resolution of precipitation forecasts using deep generative models
Ilan Price, Stephan Rasp 2022

- *Using fine-grained radar observations as ground truth, a conditional Generative Adversarial Network -- coined CorrectorGAN – is trained via a custom training procedure and augmented loss function, to produce ensembles of high-resolution, bias-corrected forecasts based on coarse, global precipitation forecasts in addition to other relevant meteorological fields*

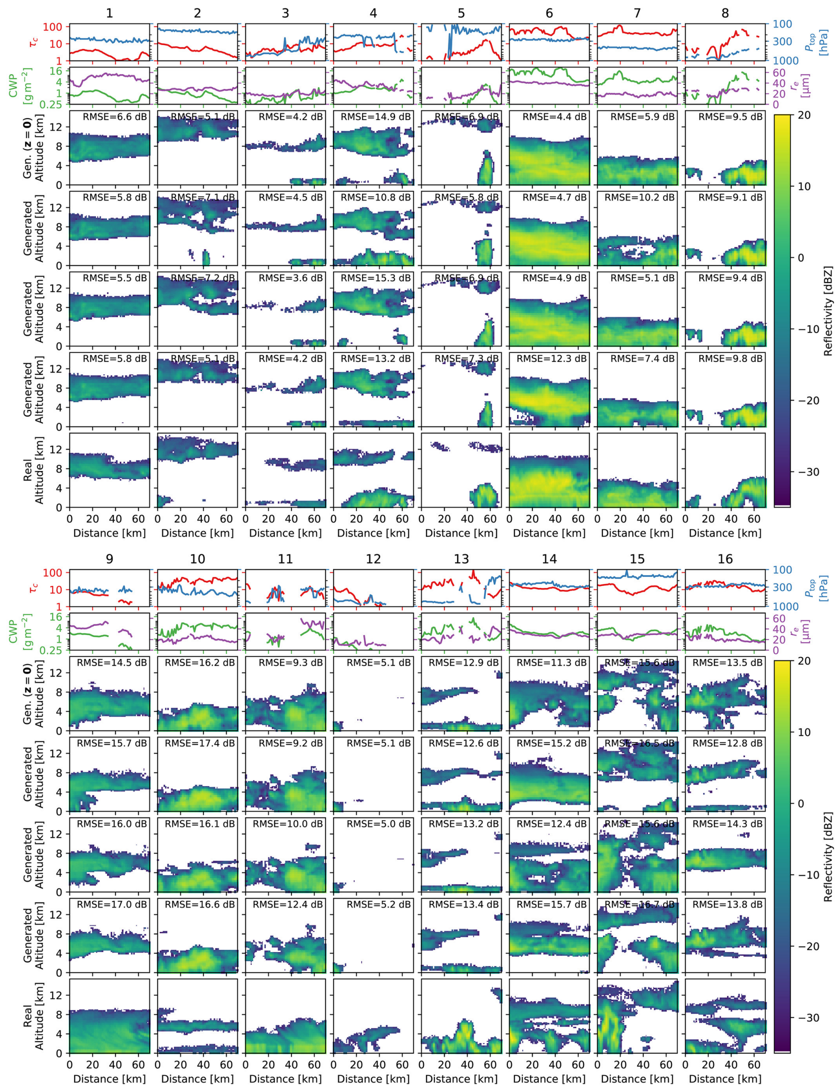


Reconstruction of physical fields

[Reconstruction of Cloud Vertical Structure With a Generative Adversarial Network](#)

Jussi Leinonen, Alexandre Guillaume, Tianle Yuan in 2019

- *CGANs were used to generate two-dimensional cloud vertical structures that would be observed by the CloudSat satellite-based radar, using only the collocated Moderate-Resolution Imaging Spectrometer measurements as input.*

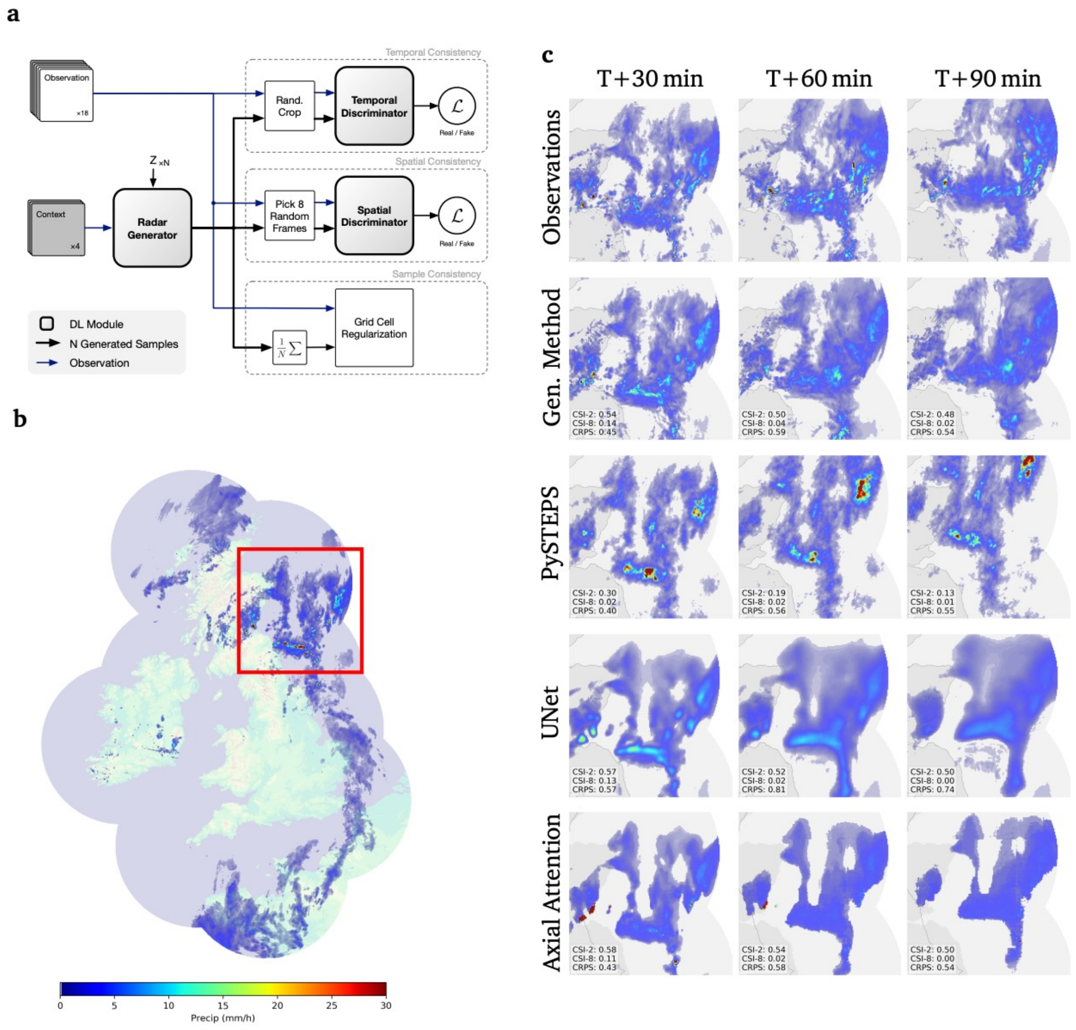


Precipitation nowcasting

Skillful Precipitation Nowcasting using Deep Generative Models of Radar

Ravuri et al. 2021

- *Deep Generative Model for the probabilistic nowcasting of precipitatio from radar. realistic and spatio-temporally consistent predictions over regions up to 1536 km x 1280 km and with lead times from 5-90 min ahead.*

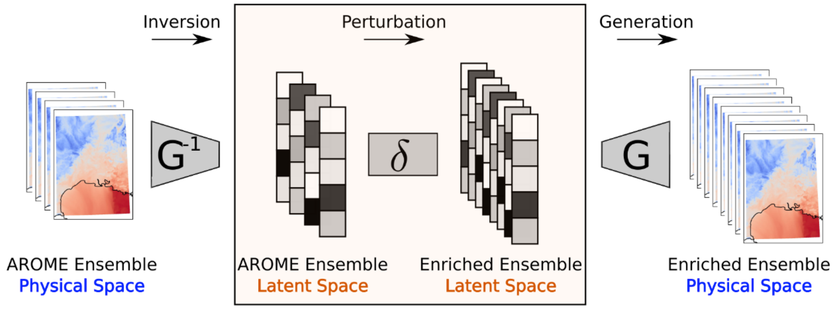

a
Observation
×18
Context
×4
Z ×N
Radar Generator
Temporal Consistency
Rand. Crop
Temporal Discriminator
$\mathcal{L}$
Real / Fake
Spatial Consistency
Pick 8 Random Frames
Spatial Discriminator
Sample Consistency
$\frac{1}{N}\sum$
Grid Cell Regularization
DL Module
N Generated Samples
Observation
b
0
5
10
15
20
25
30
Precip (mm/h)
c
T+30 min
T+60 min
T+90 min
Observations
Gen. Method
PySTEPS
UNet
Axial Attention
CSI-2: 0.54 CSI-8: 0.14 CRPS: 0.45
CSI-2: 0.50 CSI-8: 0.04 CRPS: 0.59
CSI-2: 0.48 CSI-8: 0.02 CRPS: 0.54
CSI-2: 0.30 CSI-8: 0.02 CRPS: 0.40
CSI-2: 0.19 CSI-8: 0.02 CRPS: 0.56
CSI-2: 0.13 CSI-8: 0.01 CRPS: 0.55
CSI-2: 0.57 CSI-8: 0.13 CRPS: 0.57
CSI-2: 0.52 CSI-8: 0.02 CRPS: 0.81
CSI-2: 0.50 CSI-8: 0.00 CRPS: 0.74
CSI-2: 0.58 CSI-8: 0.11 CRPS: 0.43
CSI-2: 0.54 CSI-8: 0.02 CRPS: 0.58
CSI-2: 0.50 CSI-8: 0.00 CRPS: 0.54

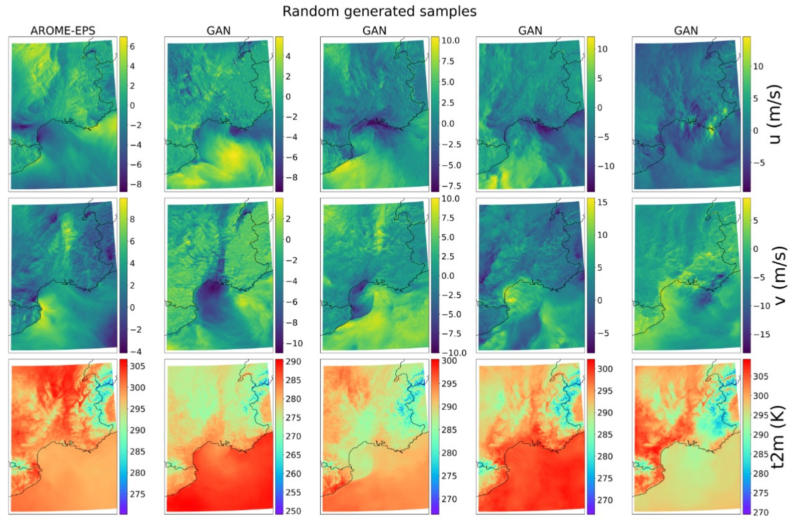
Ensemble members Generation

- Producing realistic climate data with generative adversarial networks
  Camille Besombes, Olivier Pannekoucke, Corentin Lapeyre, Benjamin Sanderson, and Olivier Thual

- Multivariate Emulation of Kilometer-Scale Numerical Weather Predictions with Generative Adversarial Networks: A Proof of Concept
  Clément Brochet, Laure Raynaud, Nicolas Thome, Matthieu Plu, and Clément Rambour 2023

- **Enriching Atmospheric Ensemble Forecasts using Conditioned State-of-the-art Generative Models.**
  Clément Brochet, Gabriel Moldovan, Laure Raynaud (work in progress)





Diffusion models

Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it's a training protocol and not a model (network)

Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it’s a training protocol and not a model (network)
- Idea:

$$p(x, y)$$

Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it's a training protocol and not a model (network)
- Idea:

$$\mathcal{N}(0,1) \qquad\qquad\qquad\qquad p(x,y)$$

Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it’s a training protocol and not a model (network)
- Idea:

$$\begin{tikzcd}\mathcal{N}(0,1) \arrow[rrr, bend left, "\text{map from Gaussian to target distribution}"] & & & p(x,y)\end{tikzcd}$$

Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it's a training protocol and not a model (network)
- Idea:

$$\begin{tikzcd}
\mathcal{N}(0,1) \arrow[rrr, bend left, "\text{map from Gaussian to target distribution}"] & & & p(x,y) \arrow[lll, bend left, "\text{forward flow}"]
\end{tikzcd}$$

Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it’s a training protocol and not a model (network)
- Idea:

map from Gaussian to target distribution

$$\begin{tikzcd}
\mathcal{N}(0,1) \arrow[rrr, bend left, "\text{backward flow}"] & & & p(x,y) \arrow[lll, bend left, "\text{forward flow}"]
\end{tikzcd}$$

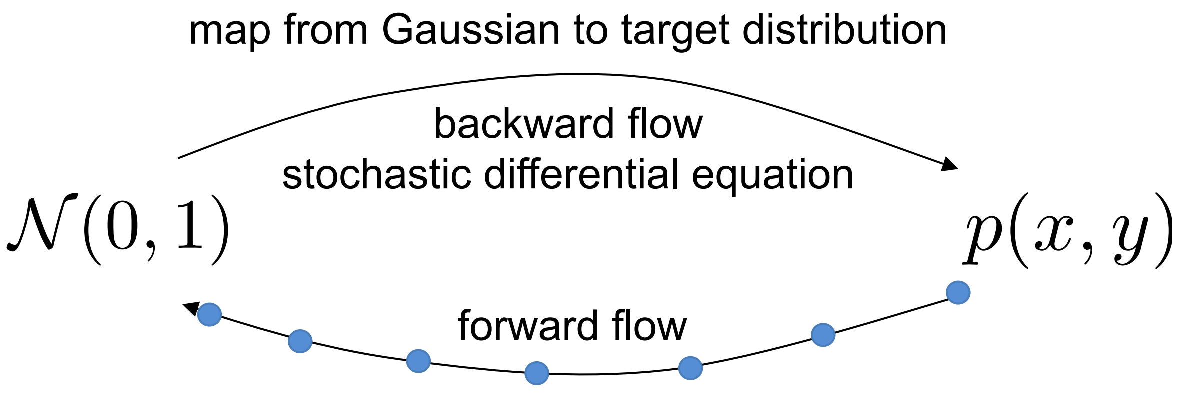
Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it’s a training protocol and not a model (network)
- Idea:

$$\begin{tikzcd}[column sep=huge]
\mathcal{N}(0,1) \arrow[r, bend left, "\text{map from Gaussian to target distribution}", "\substack{\text{backward flow}\\ \text{stochastic differential equation}}"'] & p(x,y) \arrow[l, bend left, "\text{forward flow}"']
\end{tikzcd}$$

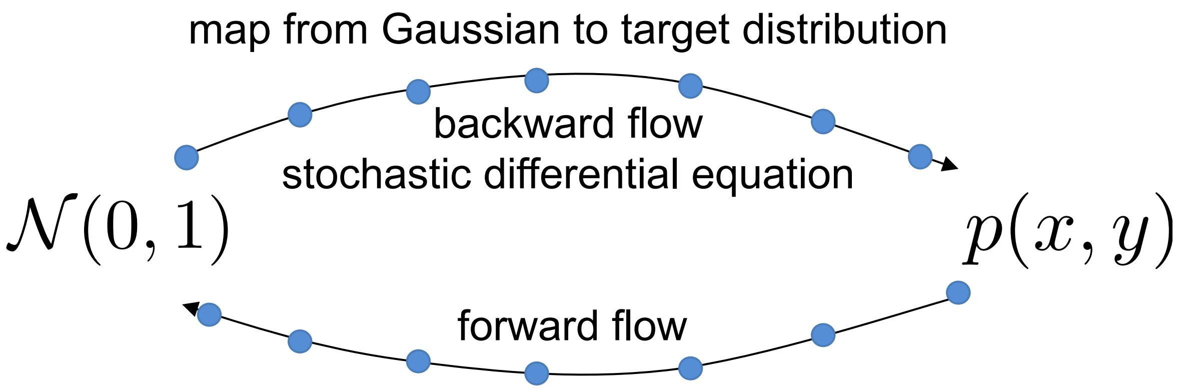
Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it's a training protocol and not a model (network)
- Idea:


map from Gaussian to target distribution
backward flow
stochastic differential equation
$\mathcal{N}(0,1)$
$p(x,y)$
forward flow

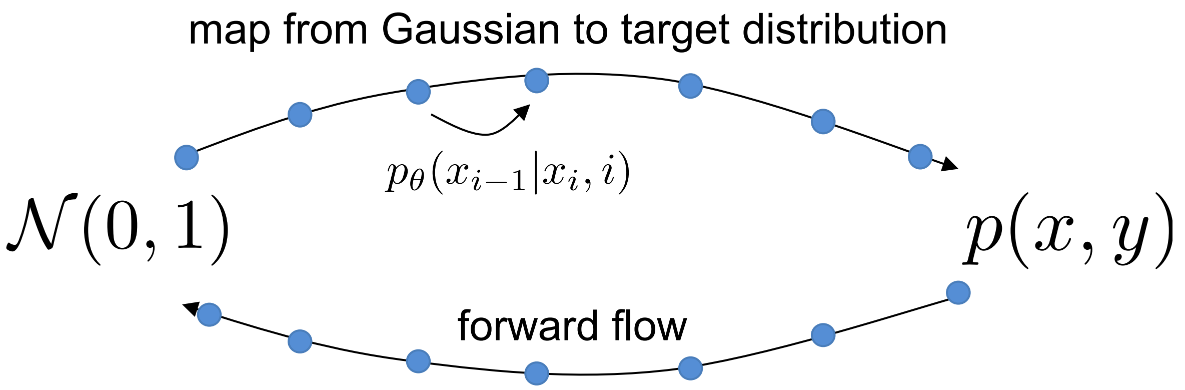
Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it's a training protocol and not a model (network)
- Idea:


map from Gaussian to target distribution
backward flow
stochastic differential equation
$\mathcal{N}(0,1)$
$p(x,y)$
forward flow

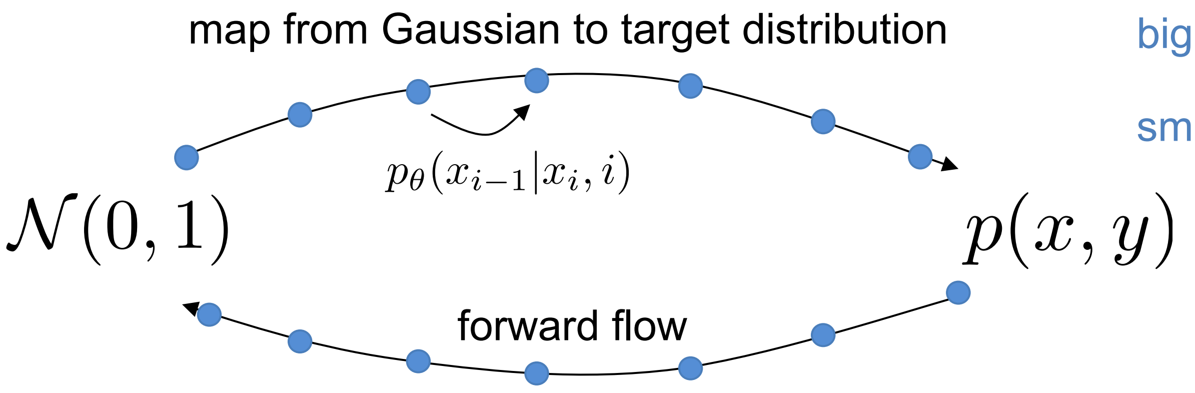
Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it’s a training protocol and not a model (network)
- Idea:


map from Gaussian to target distribution
$p_\theta(x_{i-1}|x_i, i)$
$\mathcal{N}(0,1)$
$p(x,y)$
forward flow

Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it's a training protocol and not a model (network)
- Idea:

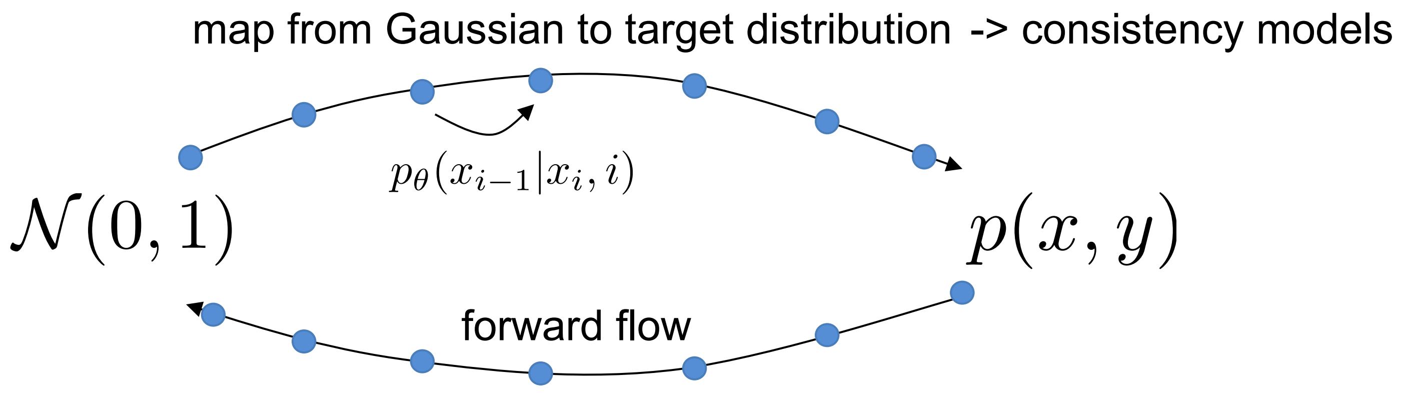

map from Gaussian to target distribution
$p_\theta(x_{i-1}|x_i, i)$
$\mathcal{N}(0,1)$
$p(x,y)$
forward flow

big step: very hard

small step: easy

Diffusion models

- State-of-the-art for generative image and video models
  - GenCast demonstrates potential for forecasting
  - Misnomer: it's a training protocol and not a model (network)
- Idea:



map from Gaussian to target distribution -> consistency models
$p_\theta(x_{i-1}|x_i, i)$
$\mathcal{N}(0,1)$
$p(x,y)$
forward flow

Song et al., Consistency Models, https://arxiv.org/abs/2303.01469

Diffusion models

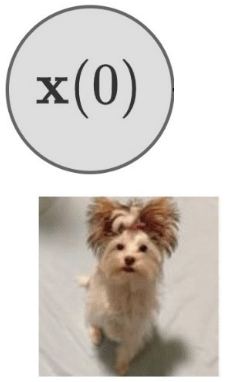


Figure from https://theaisummer.com/static/d007d60f773b61f4585cbec3869490d5/a878e/score-sde.png

Diffusion models

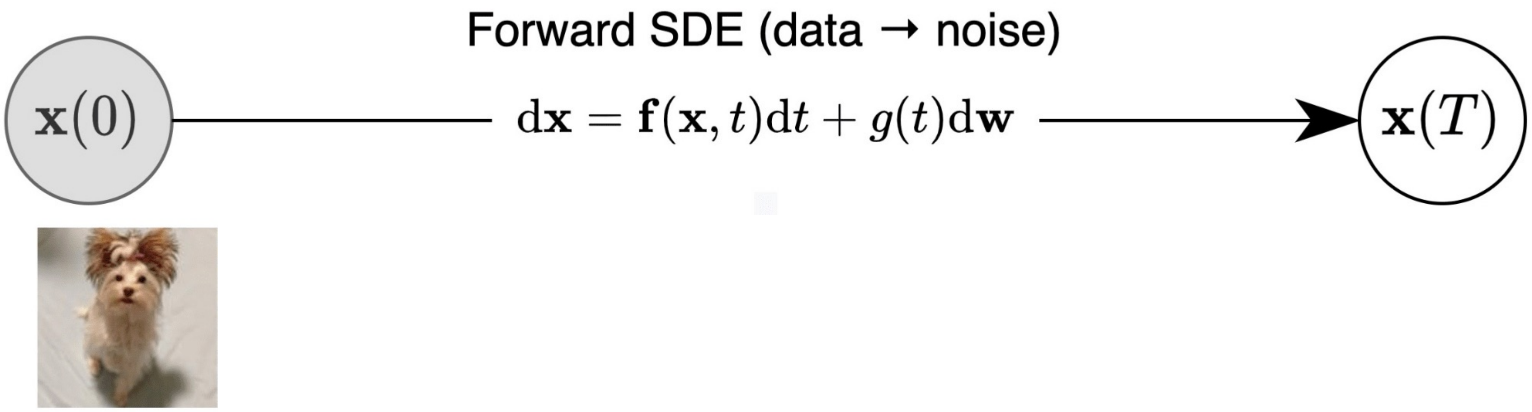

Forward SDE (data → noise)
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$

Figure from https://theaisummer.com/static/d007d60f773b61f4585cbec3869490d5/a878e/score-sde.png

Diffusion models

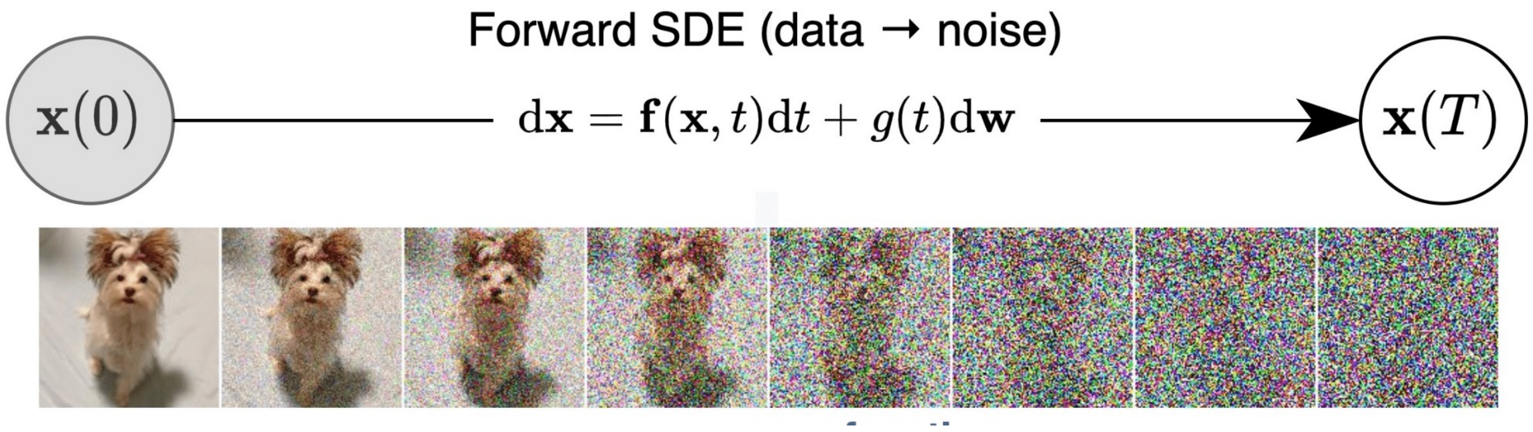

Forward SDE (data → noise)
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$

Figure from https://theaisummer.com/static/d007d60f773b61f4585cbec3869490d5/a878e/score-sde.png

Diffusion models

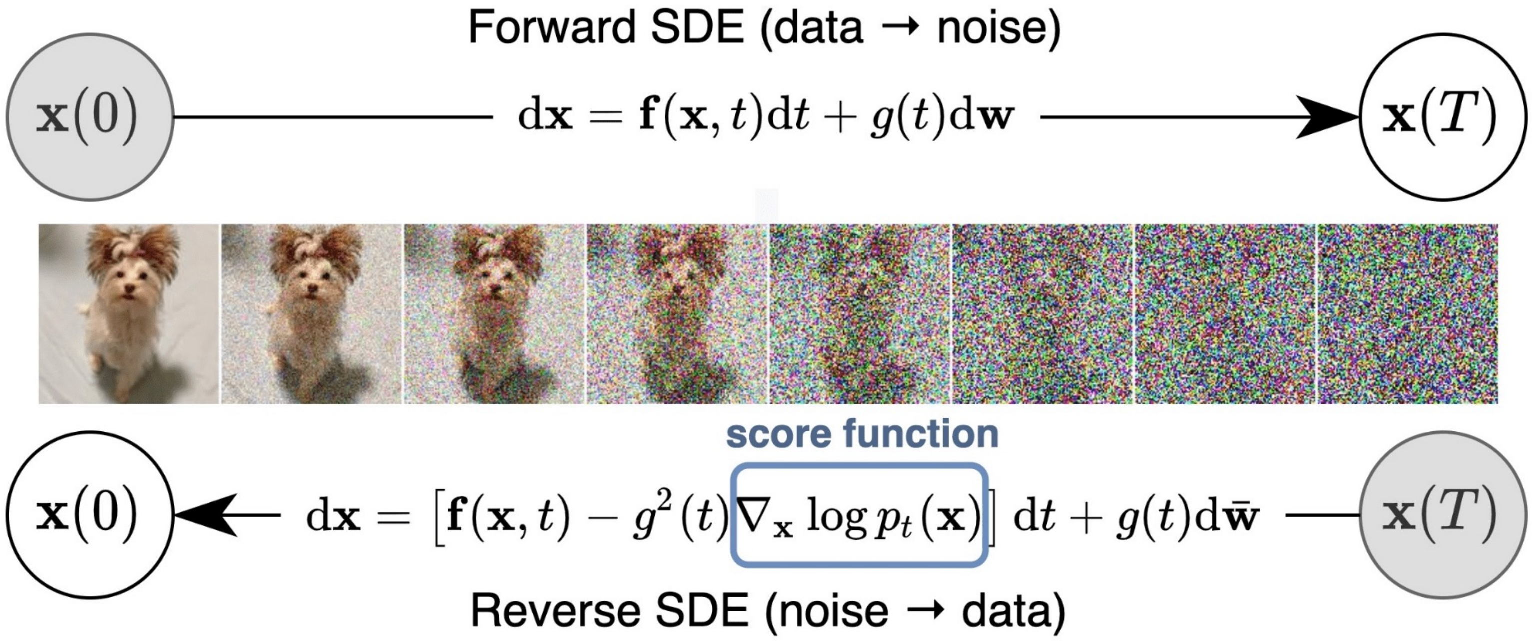
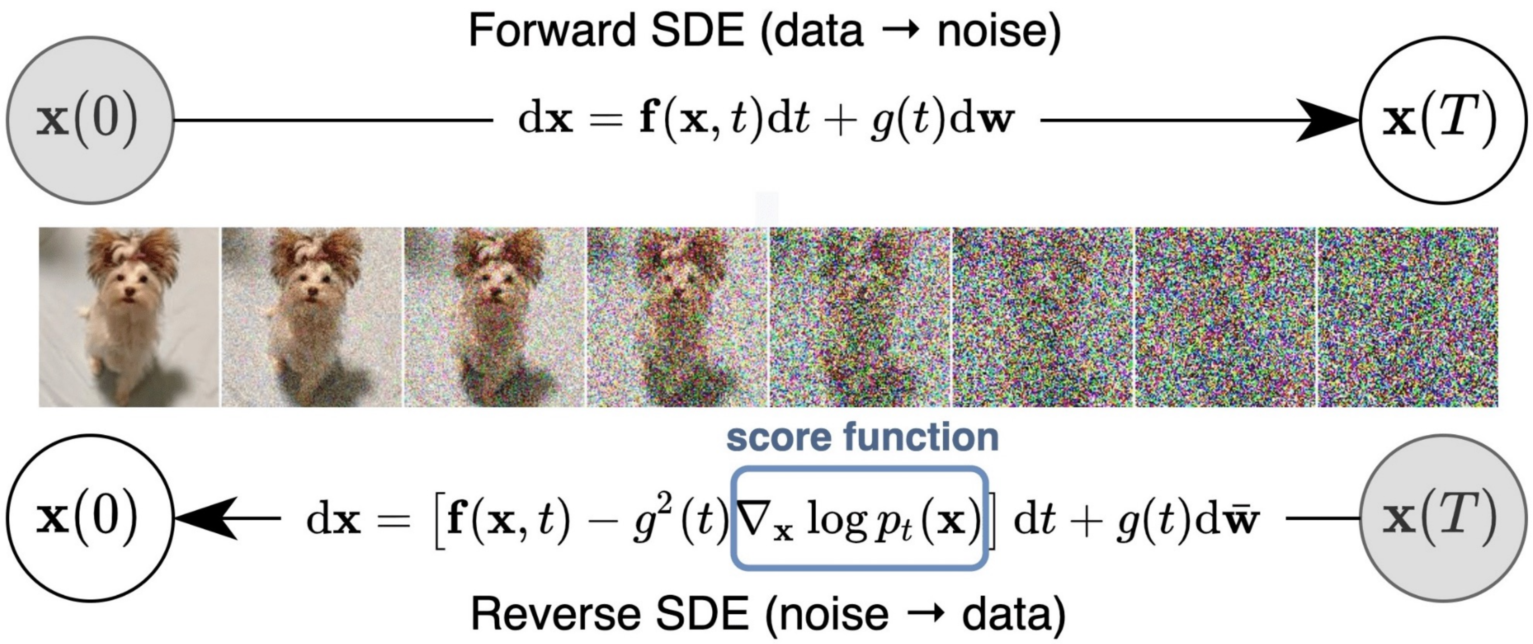

Forward SDE (data → noise)
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$
score function
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$
$\mathbf{x}(T)$
Reverse SDE (noise → data)

Figure from https://theaisummer.com/static/d007d60f773b61f4585cbec3869490d5/a878e/score-sde.png

Diffusion models


Forward SDE (data → noise)
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$
score function
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$
$\mathbf{x}(T)$
Reverse SDE (noise → data)

diffusion models learn time stepping for reverse SDE

Figure from https://theaisummer.com/static/d007d60f773b61f4585cbec3869490d5/a878e/score-sde.png

Training Diffusion models

1. Forward process up to step i: generate noised sample
   - Can be computed in closed form since standard Gaussian noise

Training Diffusion models

1. Forward process up to step i: generate noised sample
   - Can be computed in closed form since standard Gaussian noise
2. Predict x_{i-1} from x_i
   - Equivalent to learning change in noise level

Training Diffusion models

1. Forward process up to step i: generate noised sample
   - Can be computed in closed form since standard Gaussian noise
2. Predict x_{i-1} from x_i
   - Equivalent to learning change in noise level

Inference: use conventional time stepping scheme to go from pure noise to noise-free sample p(y,x)

- Standard time stepping schemes (Euler, Runge-Kutta)
- 10s – 100s of steps needed => expensive

Training Diffusion models

- Which neural network should one use?

Training Diffusion models

- Which neural network should one use?
  - Any sensible neural network will work
  - U-Net is the historical defaul
  - Transformers are quickly becoming standard (e.g. OpenAI’s Sora, Stable Diffusion 3)

Summary

- Generative models learn probability distribution

$$p(y,x) \approx p_\theta(y,x)$$

  - Neural network is numerical model for prob. distribution
  - No assumption about the form of the distribution or relationship x, y
  - Natural fit for problems where ambiguity or uncertainty or stochasticity plays a role
    - We typically do not have strong conventional models in these cases or cannot solve them efficiently

Literature

- Murphy, Probabilistic Machine Learning, Cambridge University Press, 2023, https://github.com/probml/pml-book?tab=readme-ov-file
- Karras et al., Elucidating the Design Space of Diffusion-Based Generative Models, 2022, https://openreview.net/pdf?id=k7FuTOWMOc7
- Price et al., GenCast: Diffusion-based ensemble forecasting for medium-range weather, 2023, https://arxiv.org/abs/2312.15796