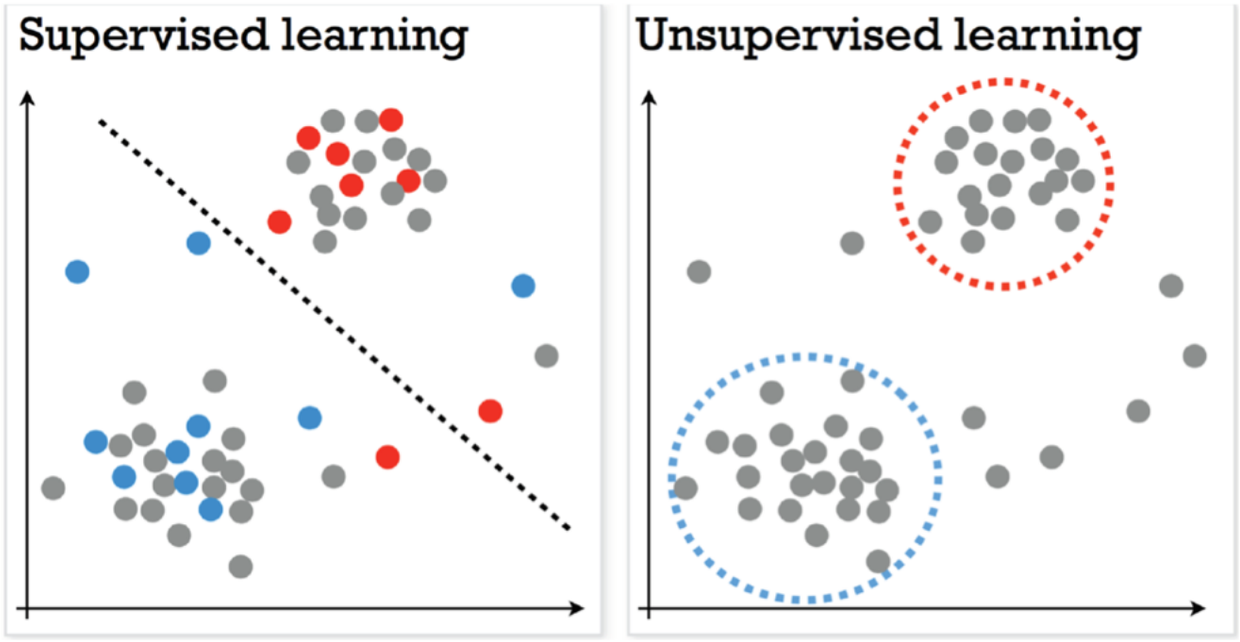
Quantum-enhanced unsupervised (generative) learning (with near-term devices)

Machine Learning: the **WHAT**




or


Learning *P(labels|data)* given samples from *P(data,labels)*
(also regression)


?

-generative models
-clustering (discriminative)
-feature extraction

Learning structure in *P(data)*
give samples from *P(data)*

Generative models

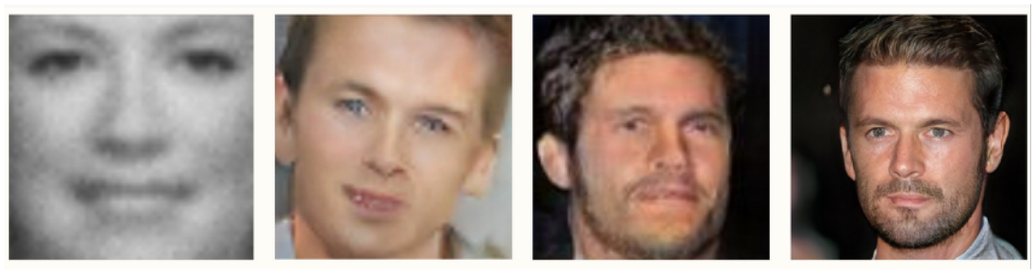
"What I cannot create, I do not understand"
R. Feynman

$\mathbf{x} \in S \subseteq \mathbb{R}^n$

$P(\mathbf{x})$

(algorithmically) generate new samples $\mathbf{x}$
(approximately) distributed according to P

$D = \{\mathbf{x}_i\} \sim P^{\times|D|}$



2014

2017

arXiv:1710.10196
arXiv:1802.07228

Name one thing in this photo

source: twitter

Why care about generative models

- generating new data; e.g. new **medicinal** drugs

- completing missing data; image recovery

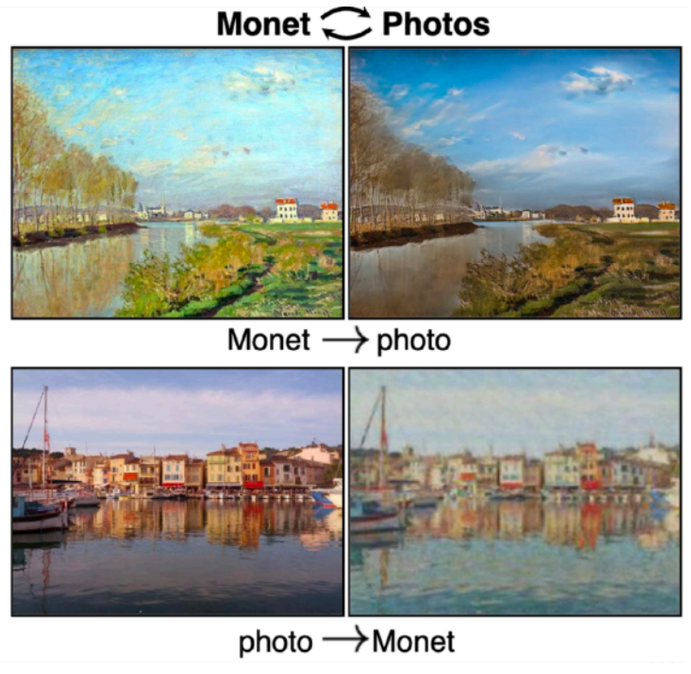
$P_\theta(x_1 \cdots x_M) \approx P(\vec{x})$ $\quad x = (x_1 \cdots x_M)$ ← EG. PIXEL VALUES

CORRUPT : $(\bar{x}_1 \cdots \bar{x}_k, ? \; ? \; ? \cdots ?)$

RECOVERY : $P_\theta(x_{k+1} \cdots x_M \mid \bar{x}_1, \bar{x}_2 \cdots \bar{x}_k)$

- modality translation

- …modelling reality…predict the future?



arxiv.org/pdf/1703.10593.pdf
arxiv.org/pdf/1804.07723.pdf

Full generative problems strictly more general than supervised learning

Let $Z = X \times Y$

SL: Given $D \sim P(X, Y)$, approximate $P(Y \mid X)$

UL: Given $D \sim P(X, Y)$, approximate $P(X, Y)$

Given access to $P(X, Y)$, the conditional can be derived….

(Also: *distributions generalize functions*)

$f: X \to Y \longrightarrow P(Y|X)$ (is FUNCTIONAL IF $P(Y|X)$ is DIRAC-DELTA $\forall$ X)

$P(Y|X) \xrightarrow{+P(X)} P(X,Y) = P(Y|X) \cdot P(X)$

$\hookrightarrow P(X) = \sum_{y} P(X,Y)$

Full generative problems strictly more general than supervised learning

Let $Z = X \times Y$

SL: Given $D \sim P(X, Y)$, approximate $P(Y \mid X)$

UL: Given $D \sim P(X, Y)$, approximate $P(X, Y)$

Given access to $P(X, Y)$, the conditional can be derived….

In generative models we find parameters that *explain ("cause") all of the data*

In discriminative models we find parameters that explain *only the (for SL) relevant aspects* of data

How do we do it?

$$\mathbf{x} \in S \subseteq \mathbb{R}^n$$

$P(\mathbf{x})$ (algorithmically) generate new samples $\mathbf{x}$ (approximately) distributed according to P

$$D = \{\mathbf{x}_i\} \sim P^{\times|D|}$$

How (1): parametrized distribution family $\{d^{\theta}(\mathbf{x})\}$

Find θ such that $d^{\theta}(\mathbf{x}) \approx P_{data}(\mathbf{x})$

Hard because we need to efficiently:

- represent
- sample from
- learn

complex and high dimensional probability distribution

How (2): the training and the metrics

- have parametrized distribution family $\{d^{\theta}(\mathbf{x})\}$

  - How do we find θ such that $d^{\theta}(\mathbf{x}) \approx P_{data}(\mathbf{x})$ $\qquad D = \{x_i\}$

Rather: *what does "≈" mean* given that we have samples only!

1) Maximum likelihood estimation (MLE):

Likelihood $L(\theta, D) = \prod_{x \in D} d^{\theta}(x)$

Find $\theta_{opt} = argmin_{\theta}\{-log\ L(\theta, D)\}$

*optimizing negative log-likelyhood is equivalent to optimizing the
so called Kullback–Leibler divergence (a.k.a. relative entropy, a measure of distinction between distributions)
between the source and target distribution

How (2): the training and the metrics

- have parametrized distribution family $\{d^{\theta}(\mathbf{x})\}$
  - How do we find θ such that $d^{\theta}(\mathbf{x}) \approx P_{data}(\mathbf{x})$ $\qquad D = \{x_i\}$

Rather: what does "≈" mean given that we have samples only!

1) Maximum likelihood estimation (MLE):

Likelihood $L(\theta, D) = \prod_{x \in D} d^{\theta}(x)$

Find $\theta_{opt} = argmin_{\theta}\{-log\ L(\theta, D)\}$

Updates can go via (approximate) gradient descent…
or derivative-free methods

How (2): the training and the metrics

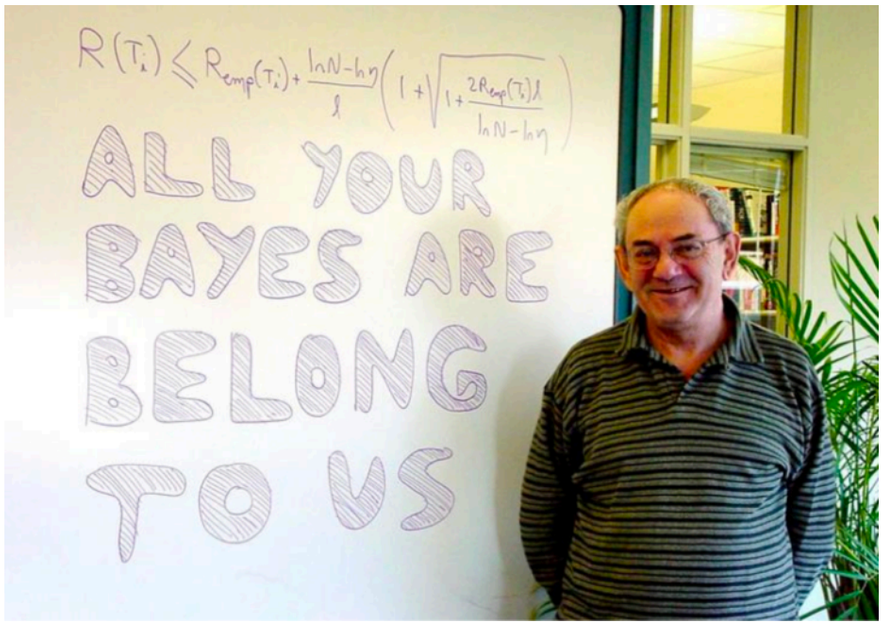
- have parametrized distribution family $\{d^{\theta}(\mathbf{x})\}$
  - How do we find θ such that $d^{\theta}(\mathbf{x}) \approx P_{data}(\mathbf{x})$ $\quad D = \{x_i\}$

Other options (2)

2) Maximum a-posteriori (MAP):

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)}$$

$$\theta_{MAP} = argmax_{\theta}\ P(D \mid \theta)P(\theta)$$
$$= argmax_{\theta}\ log\ P(D \mid \theta)P(\theta)$$
$$= argmax_{\theta}\ \underbrace{log\ P(D \mid \theta)}_{\text{MLE}} + \underbrace{\log P(\theta)}_{\Rightarrow \text{ MAP IS MLE UNDER UNIFORM PRIOR}}$$



Vladimir Vapnik

http://yann.lecun.com/ex/fun/index.html#allyourbayes

How (2): the training and the metrics

- have parametrized distribution family $\{d^{\theta}(\mathbf{x})\}$
- How do we find θ such that $d^{\theta}(\mathbf{x}) \approx P_{data}(\mathbf{x})$ $\quad D = \{x_i\}$

A number of options

3) Full Bayes:

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)}$$

4) Adversarial methods (can be MLE)

All in general intractable, exaclty

…some require values we may not be able to compute ($d^{\theta}(x)$)

more on this in a second….

But which parametrized distribution family $\{d^{\theta}(\mathbf{x})\}$?

How to specify distributions in general…

by characterizing the probabilities:

$$f : [N] \to \mathbb{R}^{+}; d(x) = \frac{f(x)}{\sum_x f(x)}$$ *c.f. sqashing functions in ML*

by characterizing a generating process

e.g. the stationary distribution of *a Markov chain*

$$P = \begin{bmatrix} P_{1,1} & P_{1,2} & \dots & P_{1,j} & \dots & P_{1,S} \\ P_{2,1} & P_{2,2} & \dots & P_{2,j} & \dots & P_{2,S} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{i,1} & P_{i,2} & \dots & P_{i,j} & \dots & P_{i,S} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{S,1} & P_{S,2} & \dots & P_{S,j} & \dots & P_{S,S} \end{bmatrix}.$$

But which parametrized distribution family $\{d^{\theta}(\mathbf{x})\}$?

Boltzmann machines

-energy based models
-stochastic recurrent NNs

→ GENERATE DISTRIBUTION OVER $\{-1,1\}^n$

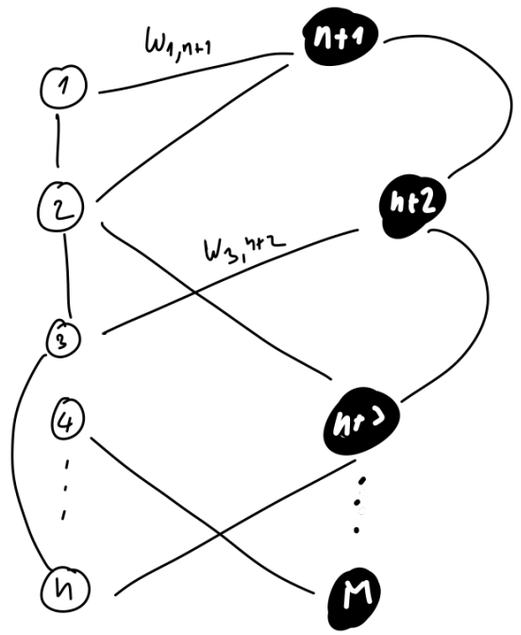
$$E(x_1 \dots x_n, x_{n+1} \dots x_M) = \sum_{i,j} w_{ij}\, x_i \cdot x_j$$

=> ISING MODEL!

THERMAL STATE: $P(\vec{x}) = \frac{\exp[\beta E(\vec{x})]}{Z}$ $\quad Z = \sum_x \exp(\beta E(x))$

"Gibbs state"
"Boltzmann distribution"

$\beta = \frac{1}{T}$ = "inverse temperature".



W_{ij}

But which parametrized distribution family $\{d^{\theta}(\mathbf{x})\}$?

Boltzmann machines

→ GENERATE DISTRIBUTION OVER $\{-1,1\}^n$

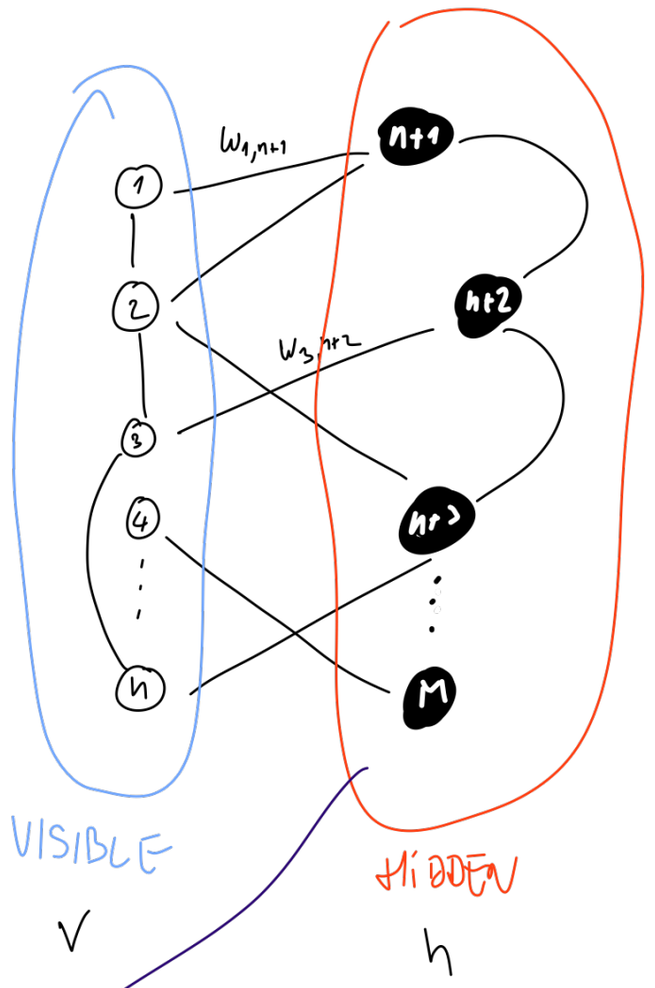
$$E(x_1 \dots x_n, x_{n+1} \dots x_M) = \sum_{i,j} w_{ij}\, x_i \cdot x_j$$

=> ISING MODEL!

THERMAL STATE: $P(\vec{x}) = \frac{\exp[\beta E(\vec{x})]}{Z}$

$P(v,h) \dots$

$$BM := P_w(v) = \sum_h P_w(v,h)$$



W_{ij}

POINT: CORRELATIONS!

But which parametrized distribution family $\{d^{\theta}(\mathbf{x})\}$?

Boltzmann machines

→ GENERATE DISTRIBUTION OVER $\{-1,1\}^n$

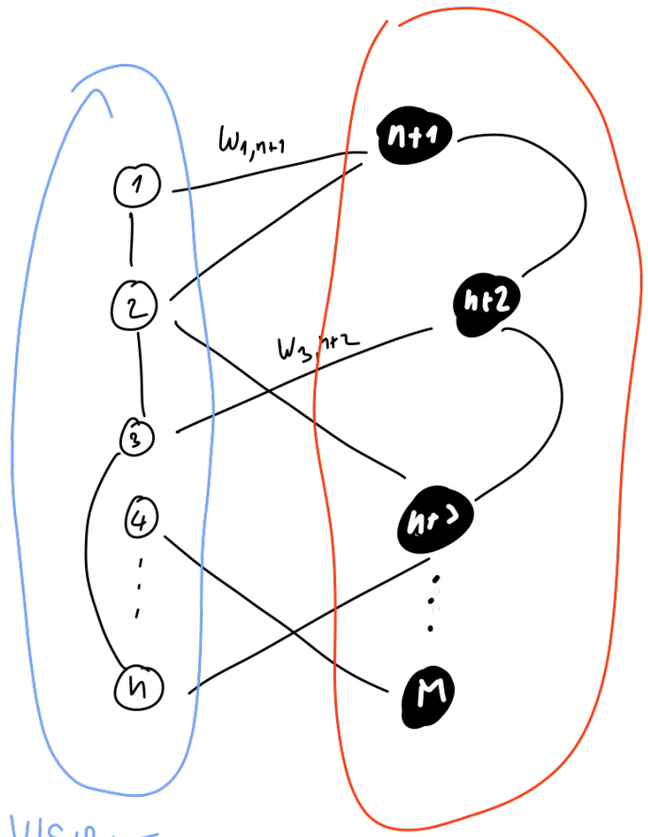
$$E(x_1 \cdots x_n, x_{n+1} \cdots x_M) = \sum_{i,j} w_{ij}\, x_i \cdot x_j$$

=> ISING MODEL!

THERMAL STATE: $P(\vec{x}) = \dfrac{\exp[\beta E(\vec{x})]}{Z}$

C.S. "SOFTMAX"

"SQUASHING FUNCTION"



W_{ij}

But which parametrized distribution family $\{d^{\theta}(\mathbf{x})\}$?

Boltzmann Machines (BM)

How do we use this? Run? Train?

"ISING MODEL"

=

"QUADRATIC UNCONSTRAINED BINARY OPTIMIZATION"

IS "NP-HARD"

- G.S. IS FNP-HARD
- LOW. TEMP. SAMPLING SOLVES G.S.

Powerful but heavily intractable

Cannot run:

CS: Low-temperature sampling NP-hard

Phys: Need to compute the partition function

Cannot train:

Training requires conditional Gibbs sampling: intractable

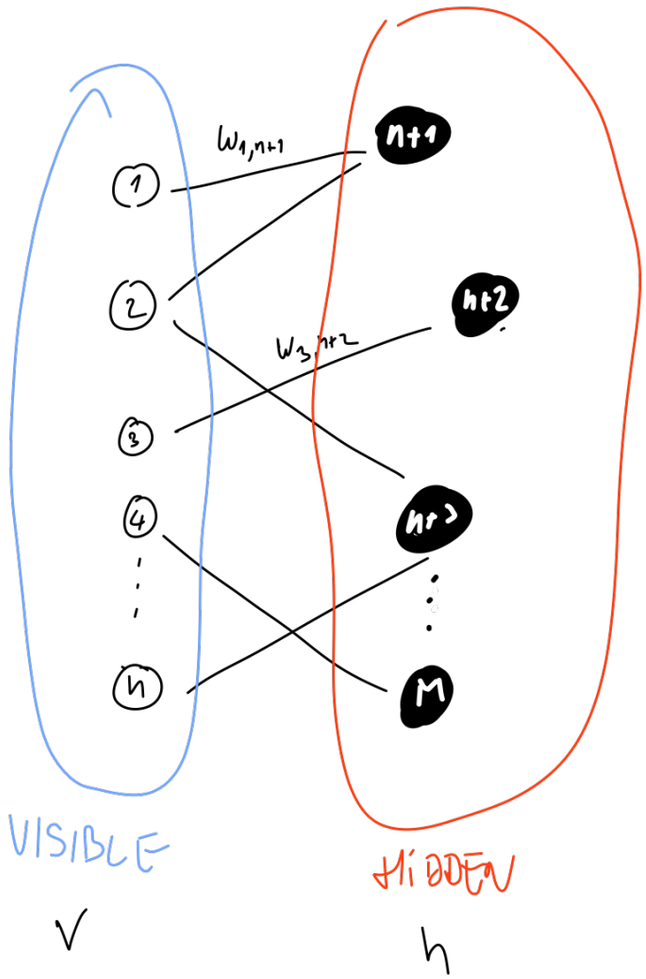
Even approximating log-likelihood too hard

But which parametrized distribution family $\{d^{\theta}(\mathbf{x})\}$?

Enter Restricted Boltzmann Machines (RBM) More tractable!

Bipartite nature makes things conditionally factorize, e.g.

$$P(v) = \frac{1}{Z}\prod_{j=1}^{m} e^{b_j v_j} \prod_{i=1}^{n}\left(1 + e^{c_i + \sum_{j=1}^{m} w_{ij} v_j}\right)$$



W_{ij}

This allows for more tractable training algorithms

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

Still "model " is hard. NB: Z is STILL HARD!
See "contrastive divergence"… approximation…

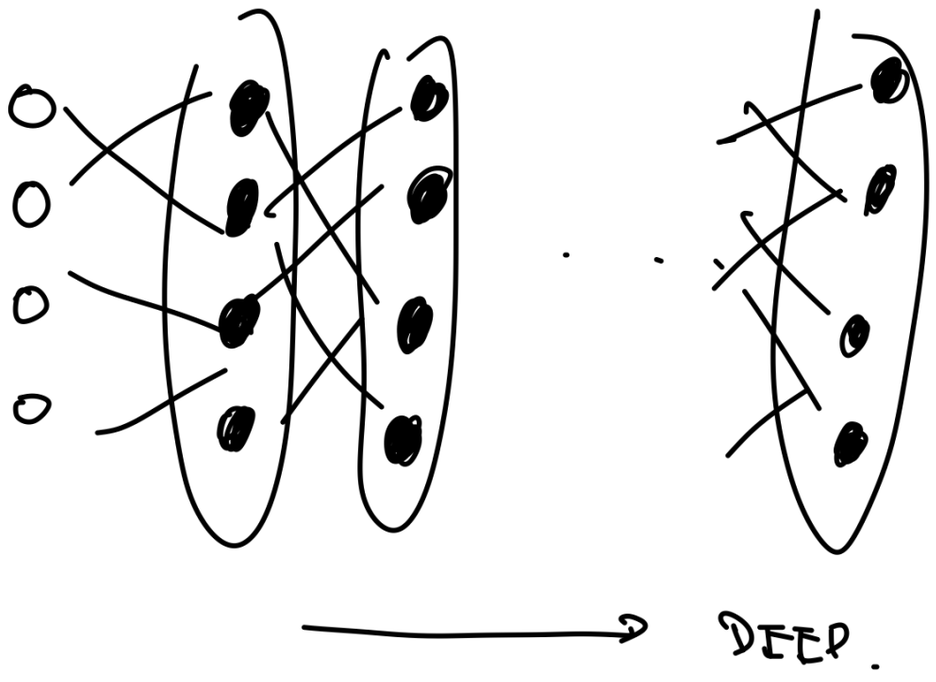
=> BIPARTITE GRAPH!

See "introduction to restricted boltzmann machines", 10.1007/978-3-642-33275-3_2

RBM is universal (Freund, Haussler, ’94) (analog of Cybenko theorem)

RBM vs BM is analogous to NN vs deep NN

What we mean by deep BM (DBM):



FIXUP

STILL RBM is LESS POWERFUL

DBM is ESSENTIALLY EQUIVALENT TO BM.

https://arxiv.org/pdf/1701.05039.pdf

RBM vs (D)BM (summary)

BM and RBM both energy models: distribution specified via an (bipartite) Ising model over "visibe" and "hidden" units, by marginalizing over the hidden units

Both universal, but BM significantly more expressive for same number of units
(has to do with the independence of hidden variables)

Training much less costly in RBM case (but still very expensive)

**In both cases, to even *run* the model, you need to use a sampler,
e.g. Markov Chain Monte Carlo**

QC/QM meets (R)BM

Quantum-applied BMs:

(R)BMs can be used to parametrize quantum states (wavefunctions) [Carleo, Troyer ’16]
(weights allowed to be complex-valued)

Quantum-enhanced BMs:

Quantum computers **can help train (R)BMs**: key step in training is (approximate) sampling from Gibbs (Boltzmann, $P(v,h)$) distribution; QCs help get those: Quantum walks; Q. semidefinite programs; Annealing. Quantum Approximate Optimization algorithms

QM can generalize (R)BM: move away from classical Ising

$$H = -\sum_a \Gamma_a \sigma_a^x - \sum_a b_a \sigma_a^z - \sum_{a,b} w_{ab} \sigma_a^z \sigma_b^z$$

arxiv:1606.02318
arxiv:1601.02036

QC/QM meets (R)BM

Quantum-enhanced BMs:

Quantum computers **can help train (R)BMs**: key step in training is (approximate) sampling from Gibbs (Boltzmann, $P(v,h)$) distribution; QCs help get those: Quantum walks; Q. semidefinite programs; Annealing. Quantum Approximate Optimization algorithms

QM can generalize (R)BM: move away from classical Ising

$$H = -\sum_a \Gamma_a \sigma_a^x - \sum_a b_a \sigma_a^z - \sum_{a,b} w_{ab} \sigma_a^z \sigma_b^z$$

In all cases. The physical intuition is: encode distribution in density matrices: mixed states.

In the quantum world… already **pure states** encode (many) distributions!

arxiv:1606.02318
arxiv:1601.02036

Next: quantum circuit Born machine

Boltzmann machine ➡ distribution is a (marginalized) Boltzmann distribution

(distr which maximizes the entropy, subject to mean energy condition)

statistical mechanics

Born machine ➡ distribution governed by the ***Born rule*** of quantum mechanics

$$P_O(i\,||\psi\rangle) = |\langle i|\psi\rangle|^2 = Tr[\,|i\rangle\langle i|\,||\psi\rangle\langle\psi|\,]$$

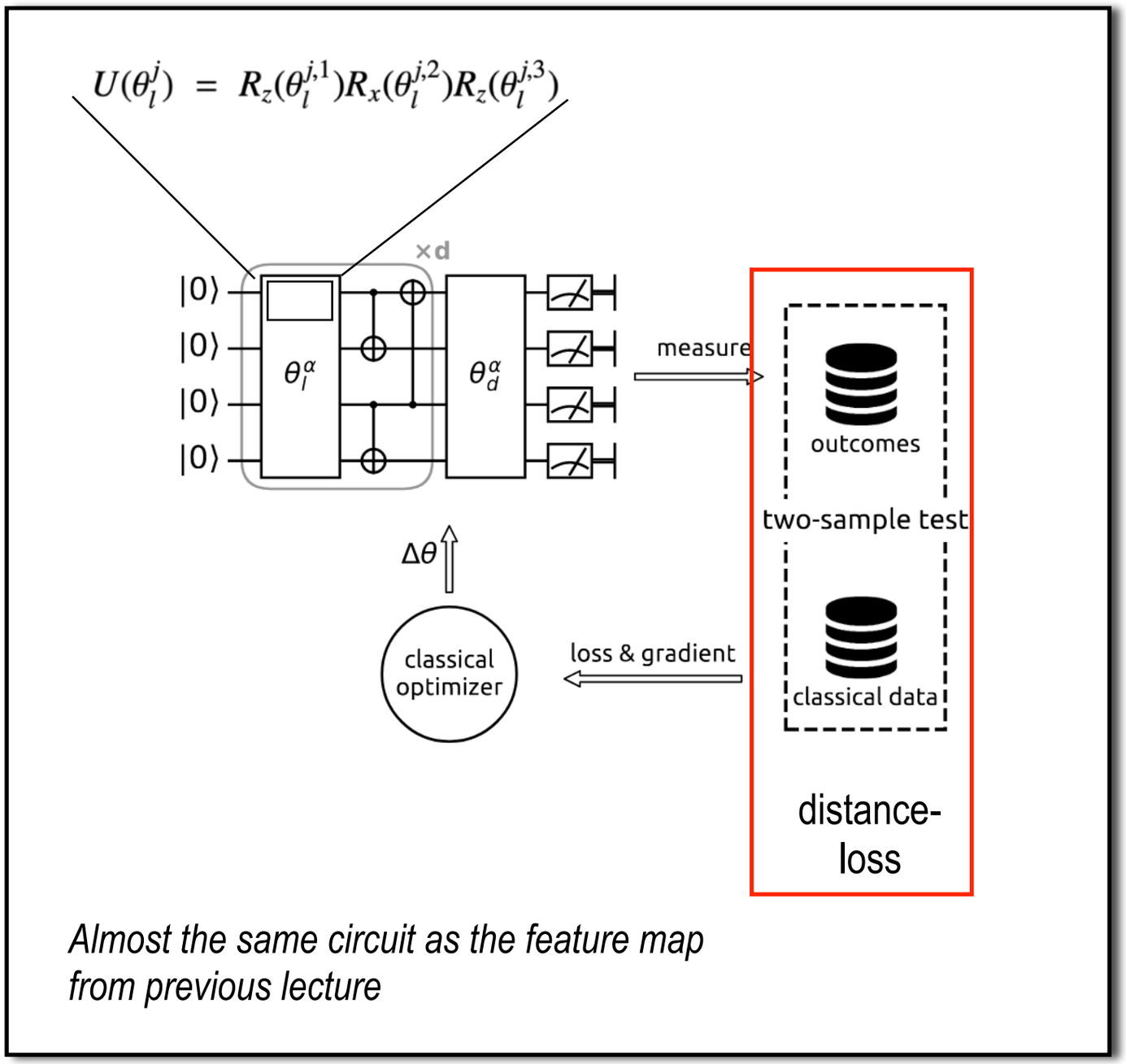
$$P_O(i\,|\rho) = Tr[P_i\rho]$$

Advantage: sampling (inference) is (quantum) easy (no nasty MCMC or similar)

Hypothesis family: parametrized circuit+ computational basis measurement

Plus: evaluation is easy, does not involve hard sampling procedures

“Minus”: “implicit” model. No direct access to likeihood, no way to compute $d_\theta(x)$, for a chosen x



arxiv.org:1804.04168

How to measure distances between distributions based on sampling… deep waters.

MLE optimization/KL optimization?

$$\mathcal{L}_{\mathsf{KL}} = -\sum_{\mathbf{z}} \pi(\mathbf{z}) \log\left(p_{\boldsymbol{\theta}}(\mathbf{z})\right) = -\mathbb{E}_{\mathbf{z}\sim\pi}(\log(p_{\boldsymbol{\theta}}(\mathbf{z}))$$

$$\frac{\partial \mathcal{L}_{\mathsf{KL}}}{\partial \boldsymbol{\theta}} \sim \sum_{\mathbf{z}} \frac{\pi(\mathbf{z})}{p_{\theta}(\mathbf{z})} \left(p^{-}_{\boldsymbol{\theta}}(\mathbf{z}) - p^{+}_{\boldsymbol{\theta}}(\mathbf{z})\right)$$

Intractable

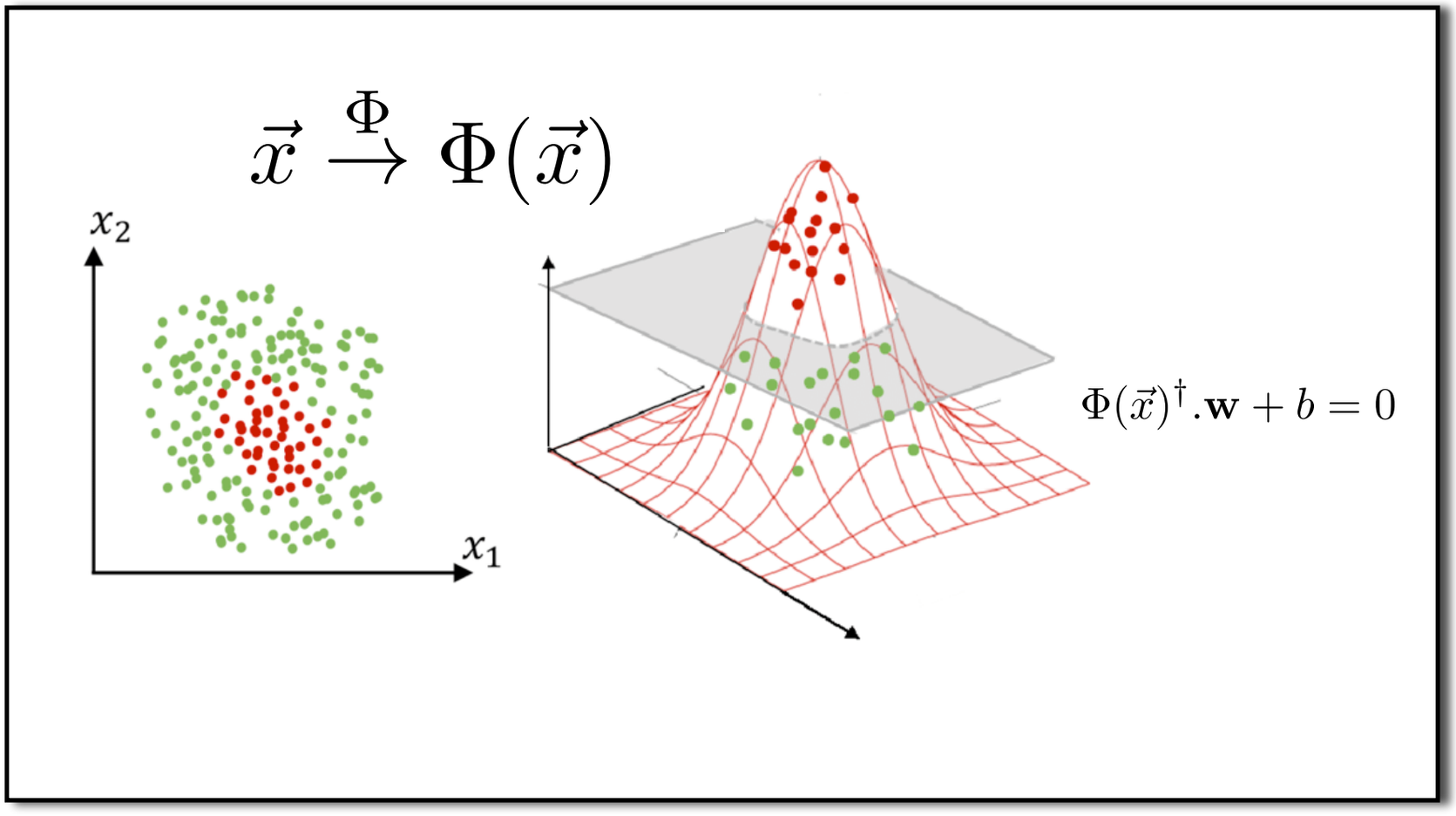
Here: Squared maximum mean discrepancy (MMD)

$$\mathcal{L} = \left\| \sum_x p_\theta(x)\phi(x) - \sum_x \pi(x)\phi(x) \right\|^2 = \mathop{\mathbb{E}}_{x\sim p_\theta, y\sim p_\theta}[K(x,y)] - 2\mathop{\mathbb{E}}_{x\sim p_\theta, y\sim \pi}[K(x,y)] + \mathop{\mathbb{E}}_{x\sim \pi, y\sim \pi}[K(x,y)].$$

arxiv.org:1804.04168

Squared maximum mean discrepancy (MMD)

$$\mathrm{MMD}(P, Q) = \|\mathbb{E}_{X\sim P}[\varphi(X)] - \mathbb{E}_{Y\sim Q}[\varphi(Y)]\|_{\mathcal{H}}.$$



Feature map

Key point: express differences of distributions JUST in terms of expected values (first moment)

Easier to measure. If Φ *is powerful enough… zero iff same distribution.*

$$\mathcal{L} = \left\| \sum_x p_\theta(x)\phi(x) - \sum_x \pi(x)\phi(x) \right\|^2 = \mathop{\mathbb{E}}_{x\sim p_\theta, y\sim p_\theta}[K(x,y)] - 2\mathop{\mathbb{E}}_{x\sim p_\theta, y\sim \pi}[K(x,y)] + \mathop{\mathbb{E}}_{x\sim \pi, y\sim \pi}[K(x,y)].$$

Everything boils down to sampling and estimating the mean…

$$K(x,y) = \phi(x)^T \phi(y)$$

$$k(x, x') = \exp(-\|x - x'\|^2/(2\,\sigma^2))$$

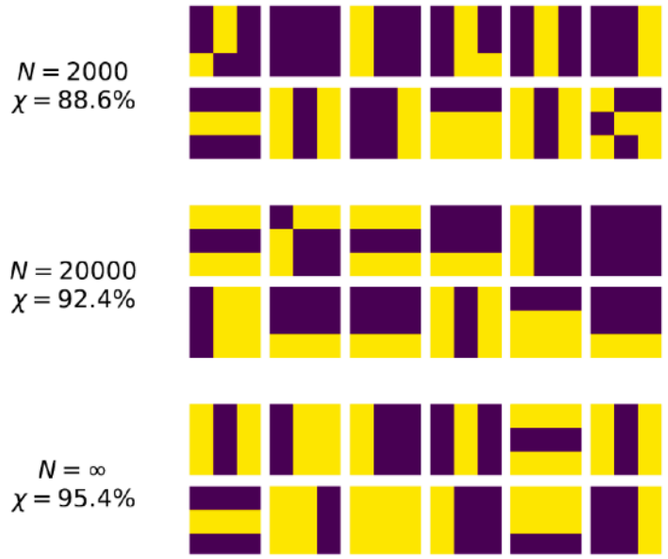
Gradients also can be expressed in terms of samples + parameter shift rule.

$$\frac{\partial \mathcal{L}}{\partial \theta_l^\alpha} = \mathop{\mathbb{E}}_{x\sim p_{\theta^+}, y\sim p_\theta}[K(x,y)] - \mathop{\mathbb{E}}_{x\sim p_{\theta^-}, y\sim p_\theta}[K(x,y)] - \mathop{\mathbb{E}}_{x\sim p_{\theta^+}, y\sim \pi}[K(x,y)] + \mathop{\mathbb{E}}_{x\sim p_{\theta^-}, y\sim \pi}[K(x,y)].$$

$$\frac{\partial f_{\mathcal{P}}(\vec{\theta})}{\partial \vec{\theta}_j} = \frac{f_{\mathcal{P}}(\vec{\theta} + \frac{\pi}{2} e_j) + f_{\mathcal{P}}(\vec{\theta} - \frac{\pi}{2} e_j)}{2}$$

For parameter shift rule, recall 7.3.1 of
http://liacs.leidenuniv.nl/~dunjkov/aQa/aQa-Lecture-3-vqe1.pdf

Some results (bottom line it works)



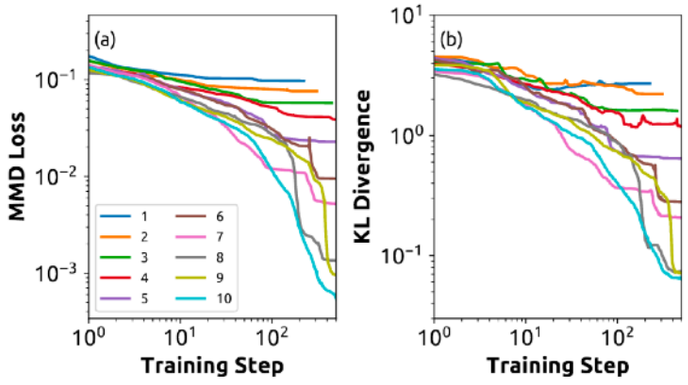


Figure 5. Losses as a function of training step for circuit depth $d = 1, \ldots, 10$. (a) The MMD loss Eq. (1), and (b) the corresponding KL divergence. Here, we use L-BFGS-B optimizer with exact gradient.

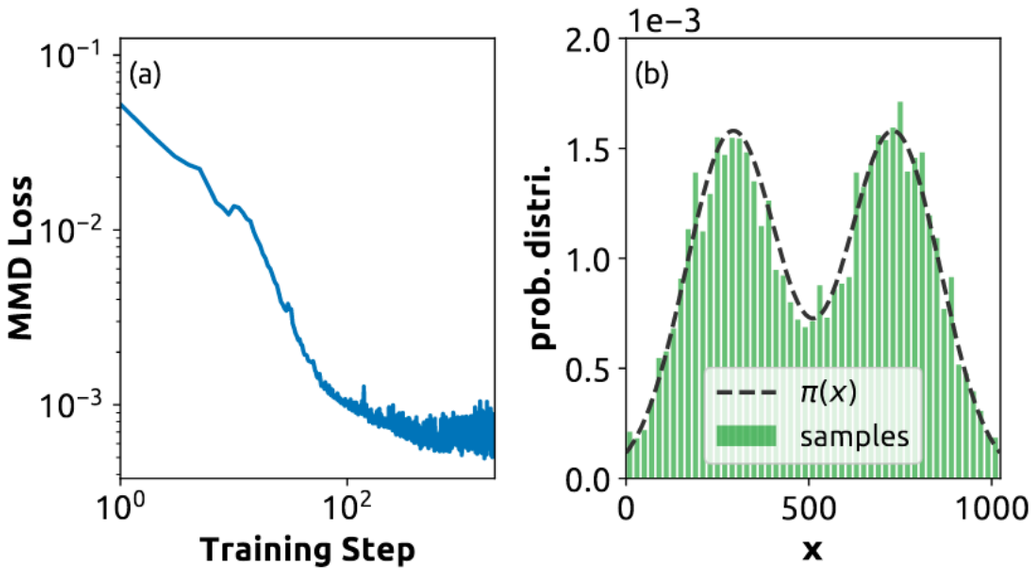


Figure 6. (a) The MMD loss as a function of Adam training step. (b) Histogram for samples generated by a trained QCBM with a bin width 20 (green bars), in comparison with the exact probability density function (black dashed line).

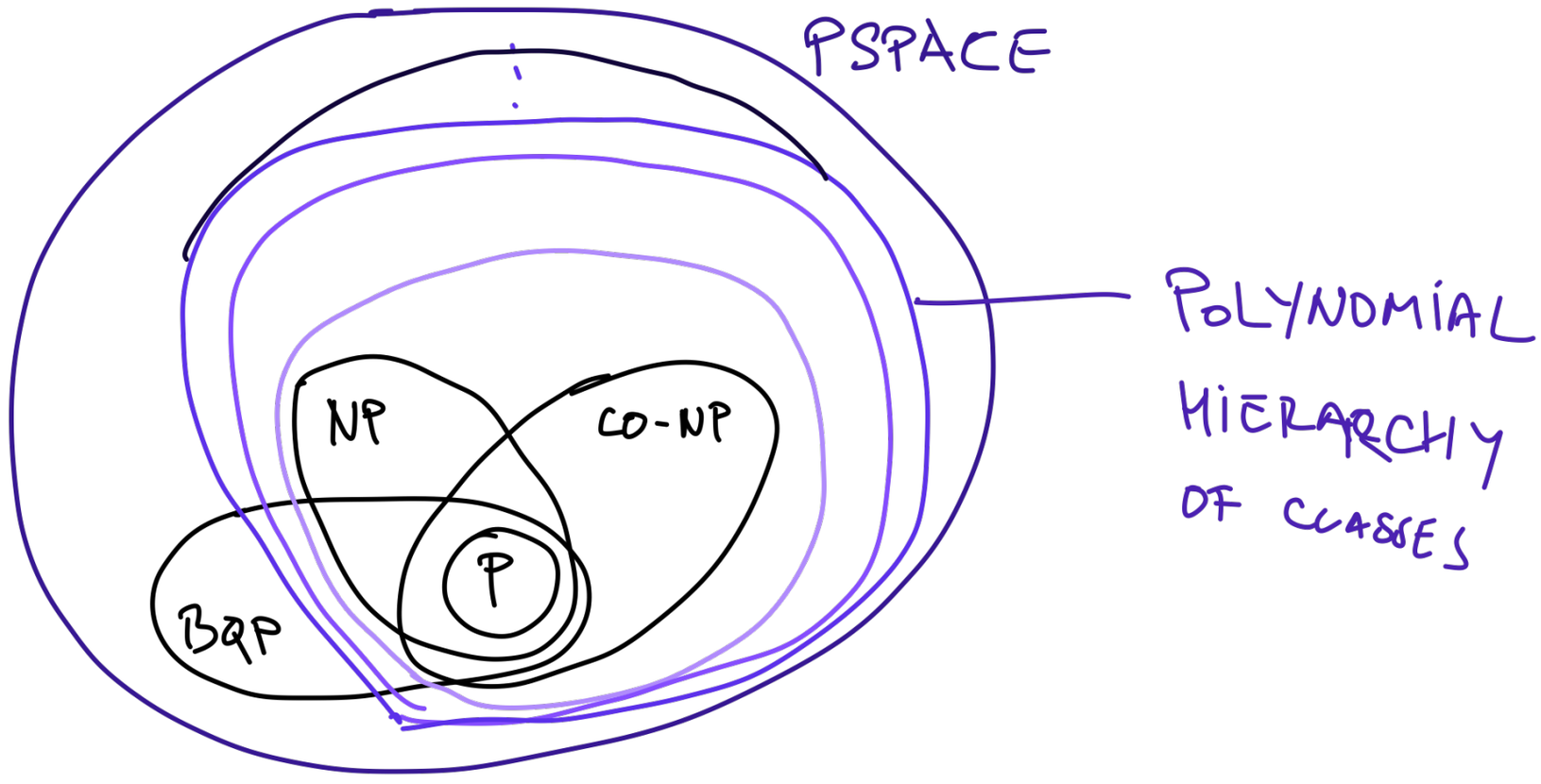
https://arxiv.org/pdf/1804.04168.pdf

Quantum supremacy of IQP sampling

Theorem 3.2 ([35]). *Assume it is # P-hard to approximate $|\mathcal{Z}|^2$ up to a relative error $1/4+o(1)$ for $1/24$ fraction of instances over the choice of the weights and biases, J_{ij}, b_k. If it is possible to classically sample from the output probability distribution of any IQP circuit in polynomial time, up to an additive error of $1/384$ in total variation distance, then there is a BPP^{NP} algorithm to solve any problem in $P^{\#P}$, and hence the polynomial hierarchy collapses to the third level)*

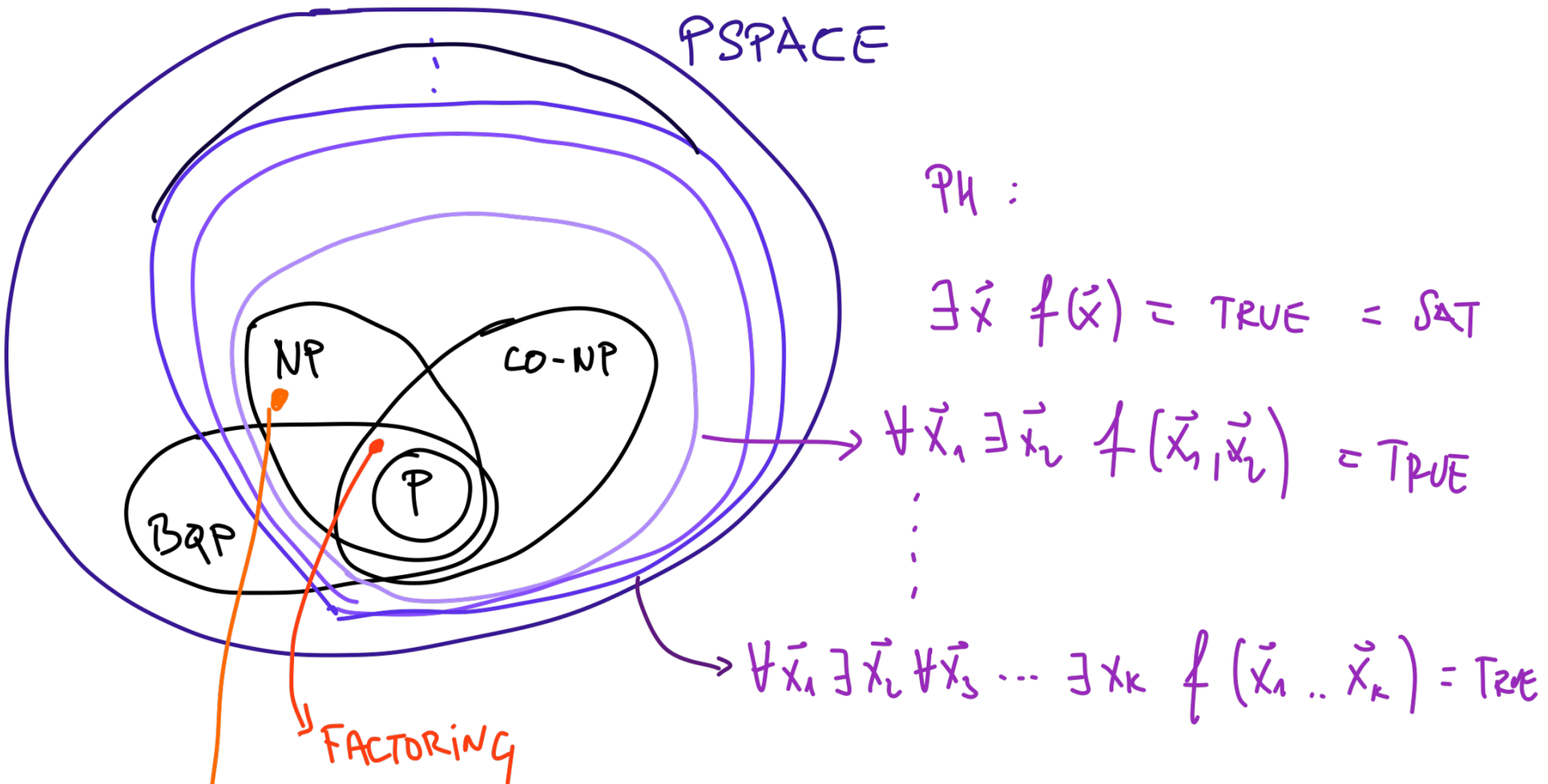
The "quantum supremacy" business..

Complexity theory (of decision problems)



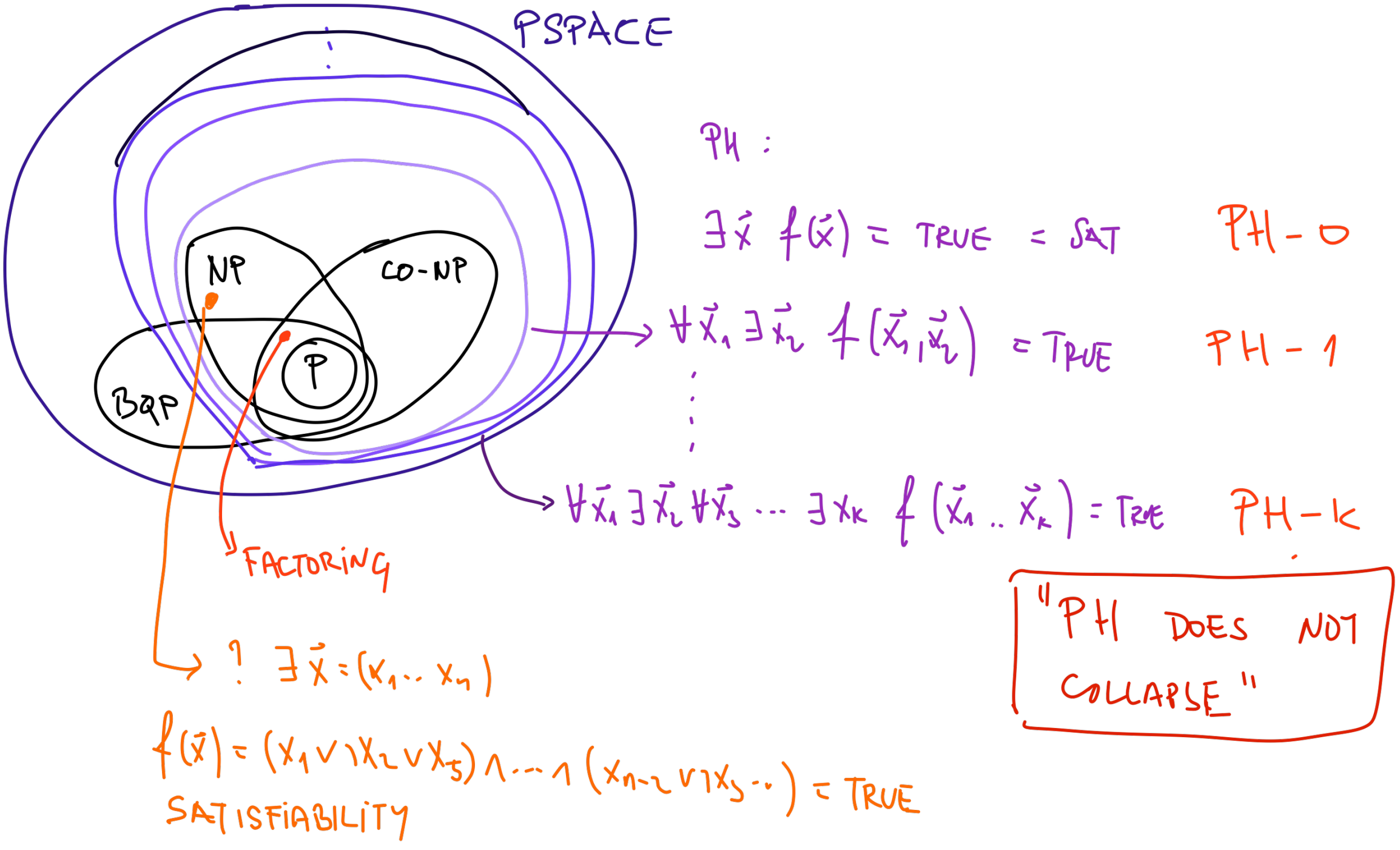
The "quantum supremacy" business..

Complexity theory (of decision problems)



The "quantum supremacy" business..

Complexity theory (of decision problems)



NOTE ... PH

HAS NOTHING TO DO WITH QC PER-SE

- IT CAN BE: $PH\text{-}1 = PH\text{-}n \quad \forall n$ & $BQP \neq P$
- IT CAN BE $BQP = P$ & $PH\text{-}k \subsetneq PH\text{-}k+1 \quad \forall k$

The "quantum supremacy" business..

Sampling problems v.s. *decision problems*

BQP contains Factoring. Assume BQP in P.
Factoring is then in P. Unlikely but no other consequences

The "quantum supremacy" business..

Sampling problems v.s. *decision problems*

BQP contains Factoring. Assume BQP in P.
Factoring is then in P. Unlikely but no other consequences

<u>Surprisingly: If we can sample from IQP circuits, PH collapses to 3rd level</u>

Even though Sampling is not decision stuff! and PH is about that…

The "quantum supremacy" business..

Sampling problems v.s. *decision problems*

BQP contains Factoring. Assume BQP in P.
Factoring is then in P. Unlikely but no other consequences

Surprisingly: If we can sample from IQP circuits, PH collapses to 3rd level

Even though Sampling is not decision stuff! and PH is about that…

Surprisingly: Even if the sampling is with multiplicative error and even additive error (worst case)

Open: Average case hardness for additive error; has unproven conjectures atop of PH collapse

Instantaneous Quantum Polynomial-time

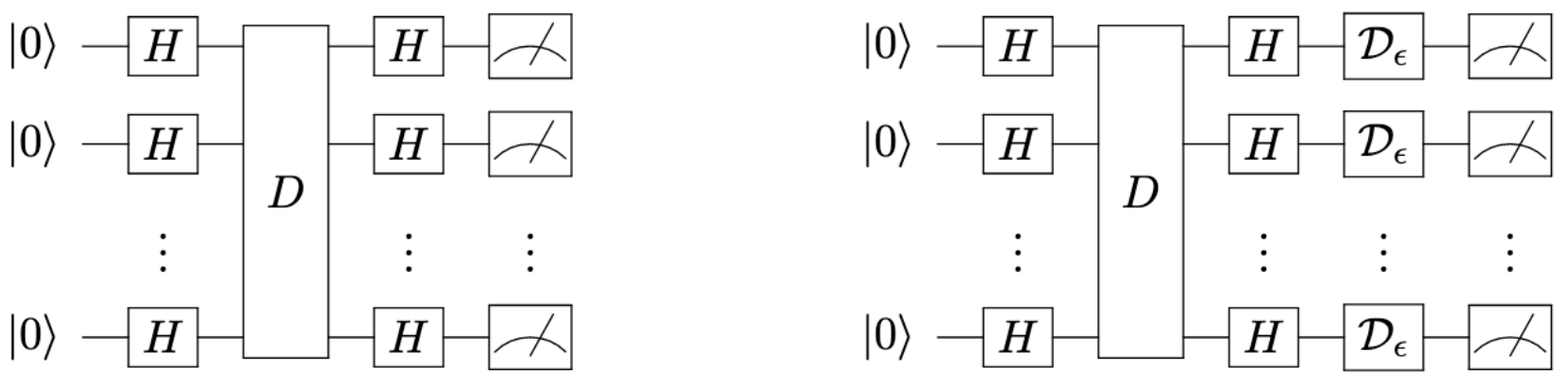


Figure 1: A standard IQP circuit, and an IQP circuit with depolarising noise. D is a circuit made up of $\mathrm{poly}(n)$ diagonal gates.

These are our QML circuits…

https://arxiv.org/pdf/1610.01808.pdf

The framework of Generative Adversarial Nets

Elements of Generative models:

a) model
(something that can generate samples from distributions / specification of distribution)

b) distance measure (metric, divergence)
(something to tell you how “far we are from real data”, to define the *loss*)

c) computational method to to minimize loss

Goodfellow et al. NIPS, ‘14

The framework of Generative Adversarial Nets

Elements of Generative models:

a) model
(something that can generate samples from distributions / specification of distribution)

b) distance measure (metric, divergence)
(something to tell you how “far we are from real data”, to define the *loss*)

c) computational method to to minimize loss

Both b) and c) can be very problematic (involving likelihoods).
How to choose metric…

Goodfellow et al. NIPS, ‘14

The framework of Generative Adversarial Nets

Generative models about distribution $P(X)$…
when are we *faking it well*?

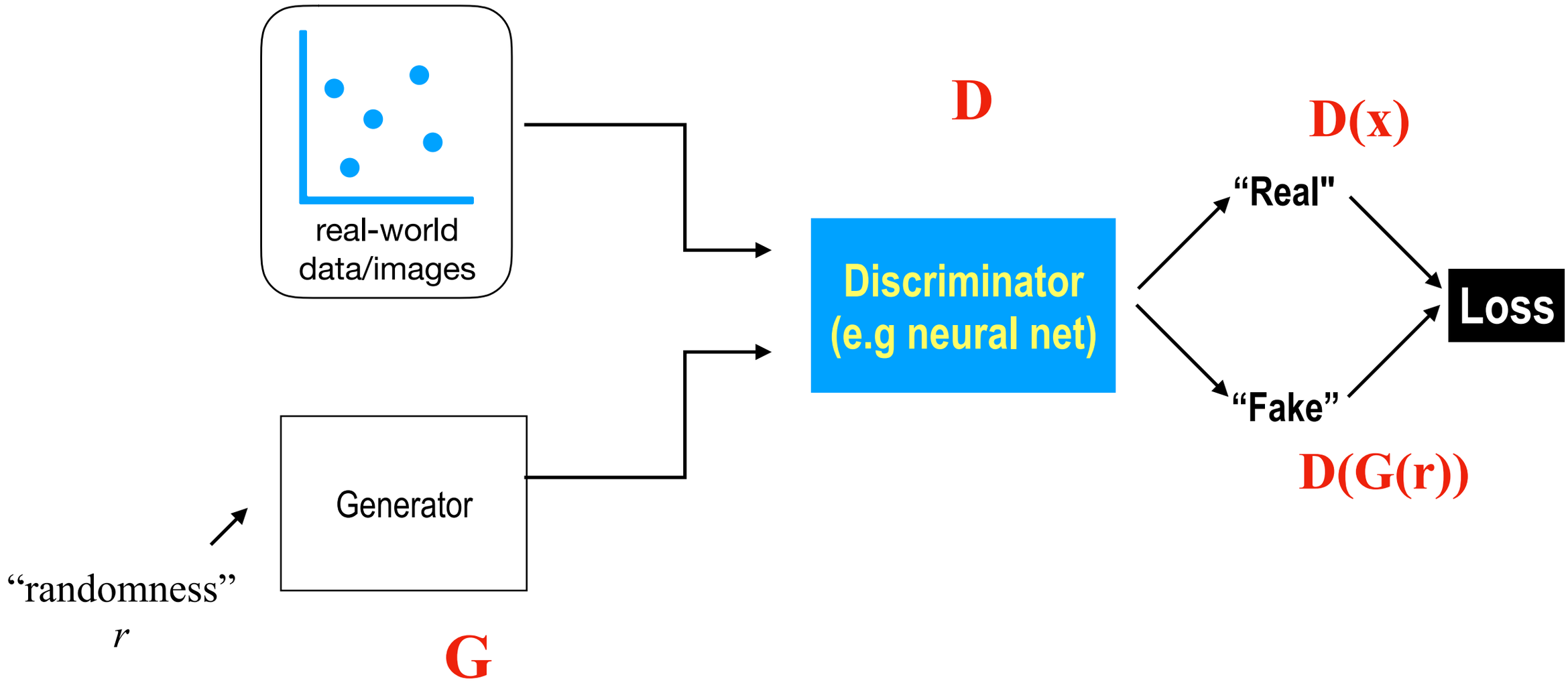
Mebbe: When an expert cannot tell real from fake…

Who is the relevant expert?
How about ML itself?

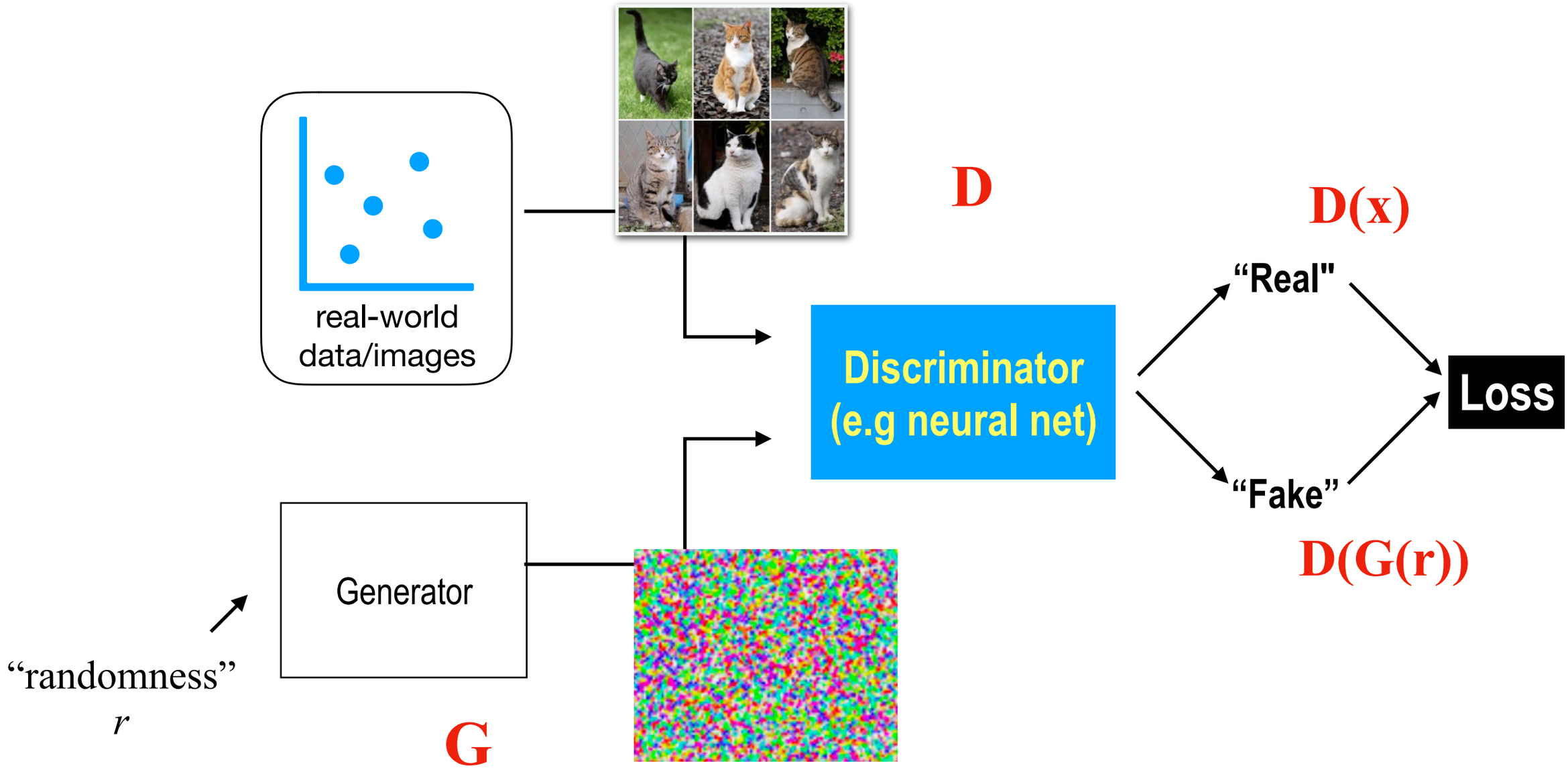
Lost Van Gogh?

https://www.groundai.com/project/automated-deep-photo-style-transfer/1

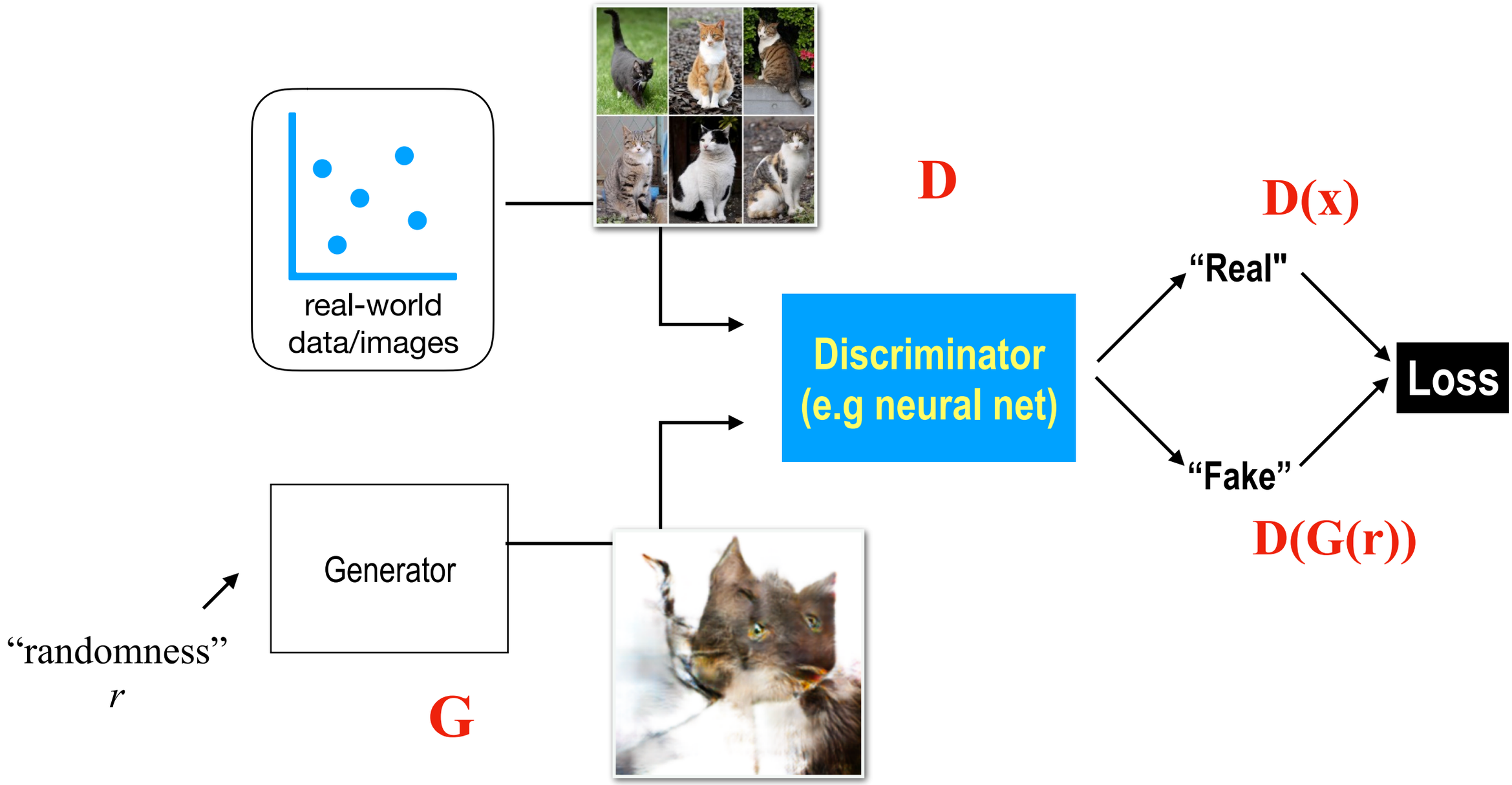
The framework of Generative Adversarial Nets



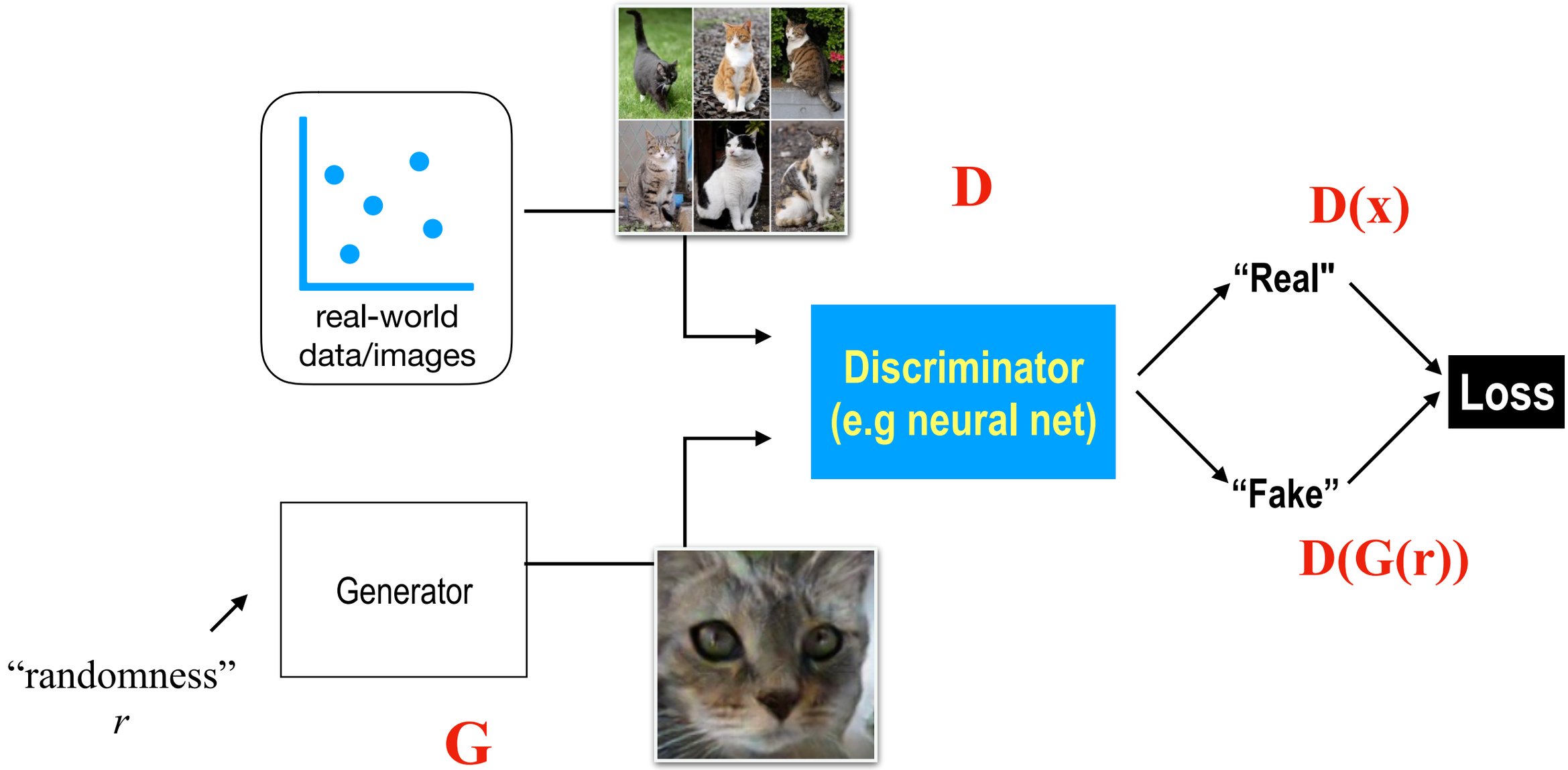
The framework of Generative Adversarial Nets



The framework of Generative Adversarial Nets



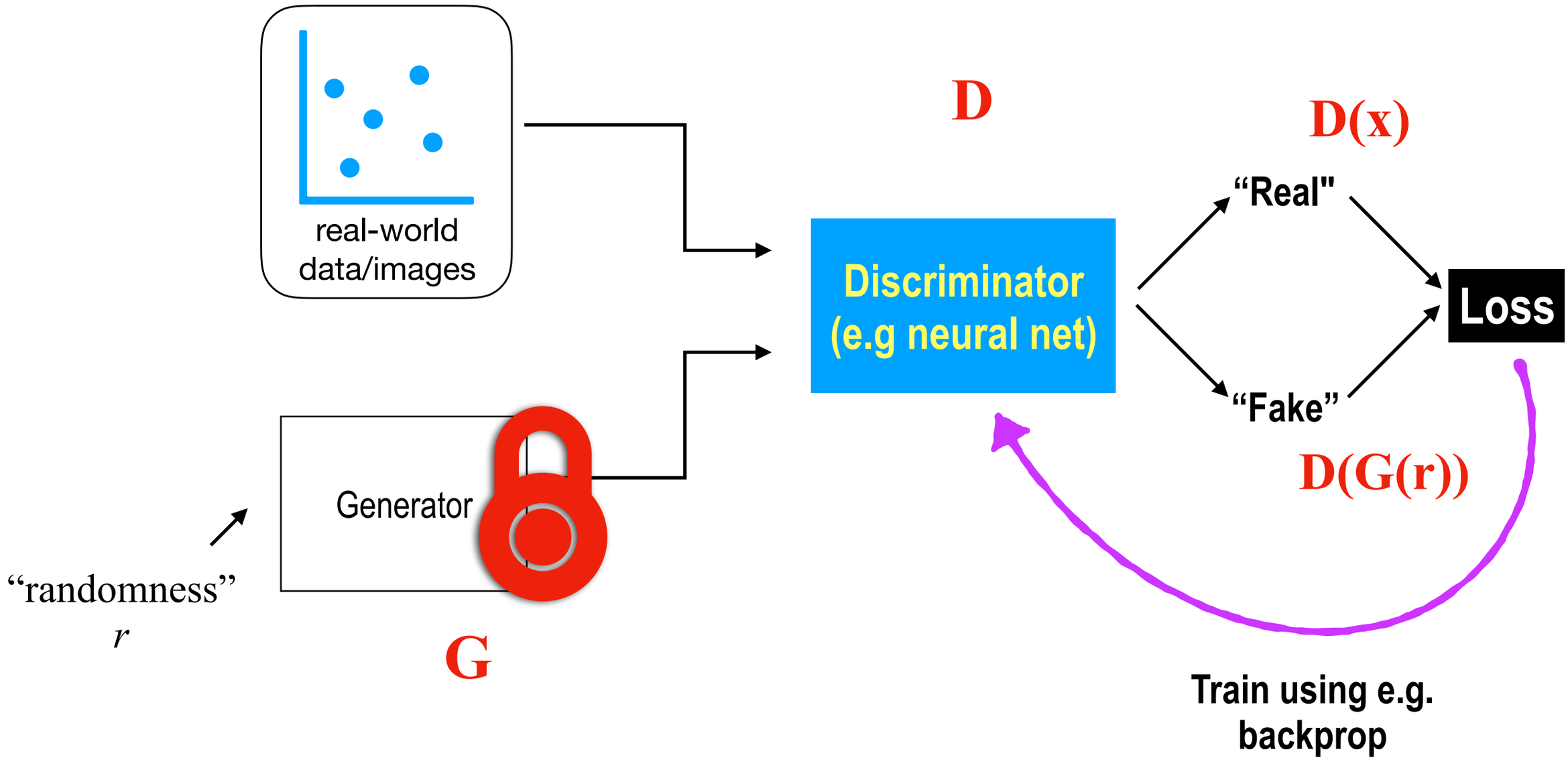
The framework of Generative Adversarial Nets



Google "Meow generator"…

The framework of Generative Adversarial Nets

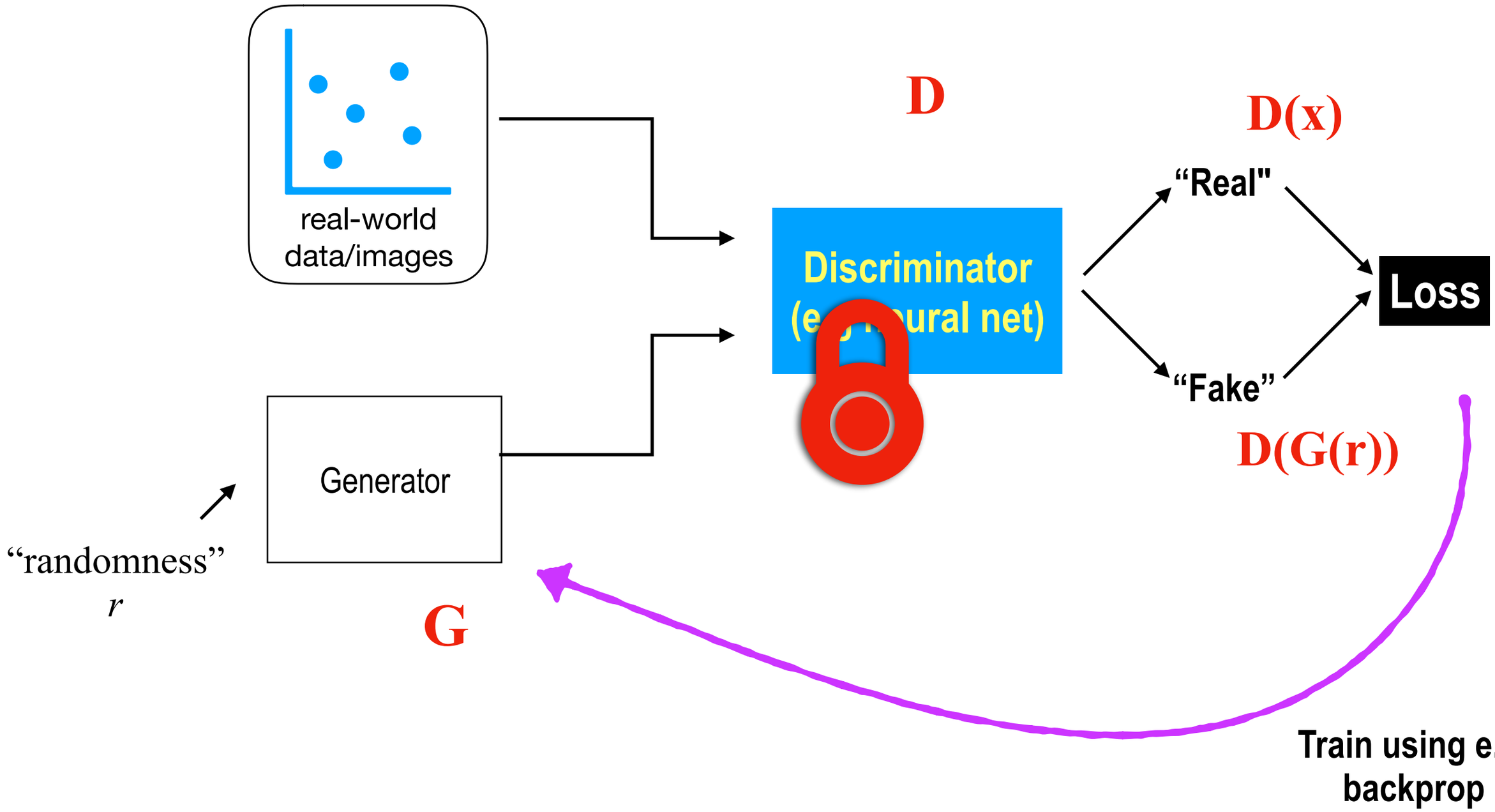
Training
Phase: training discriminator



Makes *D* *an expert counterfeit sensor against current G*

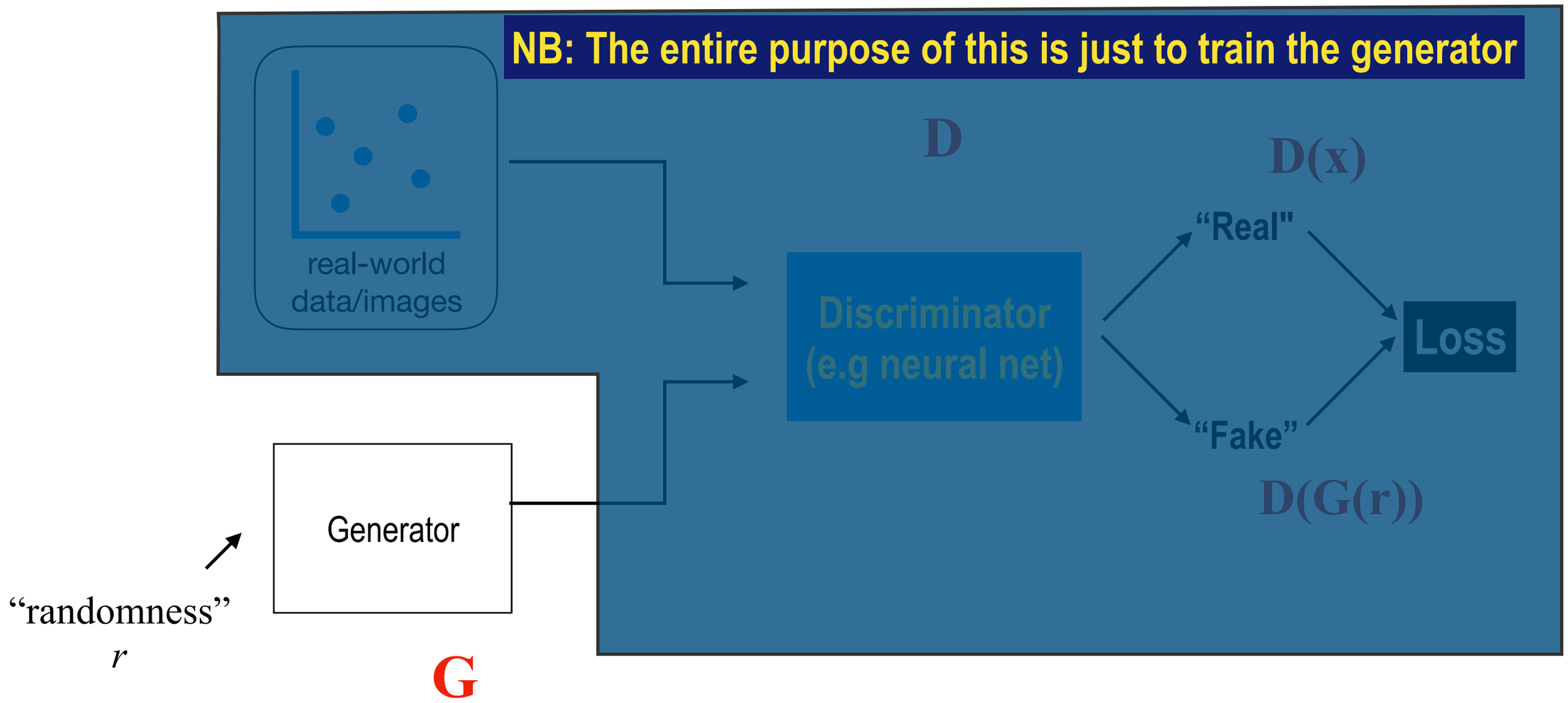
The framework of Generative Adversarial Nets

Training
Phase: training generator



Makes *G* *an expert counterfeiter against current D*

The framework of Generative Adversarial Nets



The framework of Generative Adversarial Nets

Phases are iterated

Sequential solution to a *minimax game; V(D,G) is success of D*

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

Train D

Train G

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**

  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

  - 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

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

The framework of Generative Adversarial Nets

Theory says it works:

The Nash equilibrium is achieved at:

- $p_{data}(x) = p_G(x)$
- *D(x) = 1/2 (random guess)*

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

$$C(G) = \max_D V(G,D)$$

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data}

The framework of Generative Adversarial Nets

Practice says it sometimes works.

Advantages:

- Works with models where sampling is easy (to train feed-forward NN with noise input)
- No need to compute Maximum Likelihood estimation
- Could be robust to overfitting
- Empirical success

	Deep directed graphical models	Deep undirected graphical models	Generative autoencoders	Adversarial models
Training	Inference needed during training.	Inference needed during training. MCMC needed to approximate partition function gradient.	Enforced tradeoff between mixing and power of reconstruction generation	Synchronizing the discriminator with the generator. Helvetica.
Inference	Learned approximate inference	Variational inference	MCMC-based inference	Learned approximate inference
Sampling	No difficulties	Requires Markov chain	Requires Markov chain	No difficulties
Evaluating $p(x)$	Intractable, may be approximated with AIS	Intractable, may be approximated with AIS	Not explicitly represented, may be approximated with Parzen density estimation	Not explicitly represented, may be approximated with Parzen density estimation
Model design	Models need to be designed to work with the desired inference scheme — some inference schemes support similar model families as GANs	Careful design needed to ensure multiple properties	Any differentiable function is theoretically permitted	Any differentiable function is theoretically permitted

Table 2: Challenges in generative modeling: a summary of the difficulties encountered by different approaches to deep generative modeling for each of the major operations involving a model.

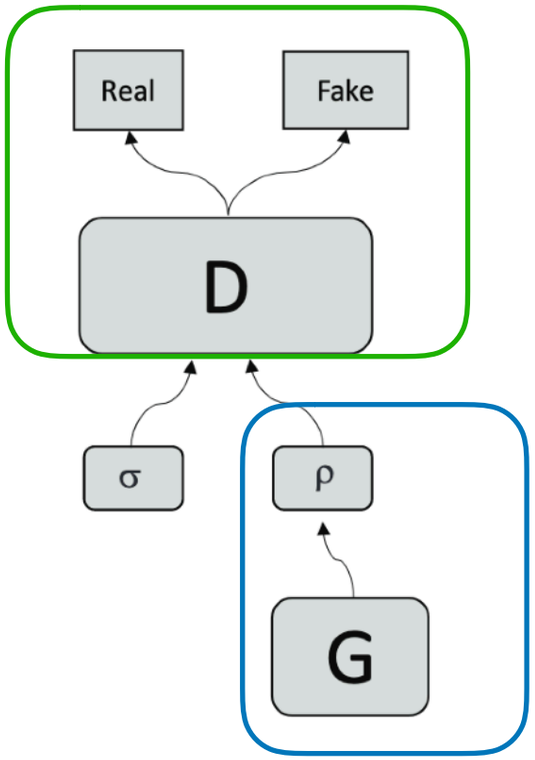
https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

From GANs to Q. GANS

Note GANs is a framework which can in principle work with (almost) any pair of supervised learner and a generator.

Any part can be made quantum

The entire formalism generalizes to quantum systems



measurement…of density operators (mixed states)

specification of density operator (mixed states)…

Fixed-point theorems can be proven in a vastly generalized setting, and they are actually simpler.
(theory of optimal measurements was developed by Helstrom in ’76)

Quantum cases allow speed-ups, applications in many-body physics and chemistry

Lloyd, Dallaire-Demers, Killoran
https://arxiv.org/abs/1804.09139
https://arxiv.org/pdf/1804.08641

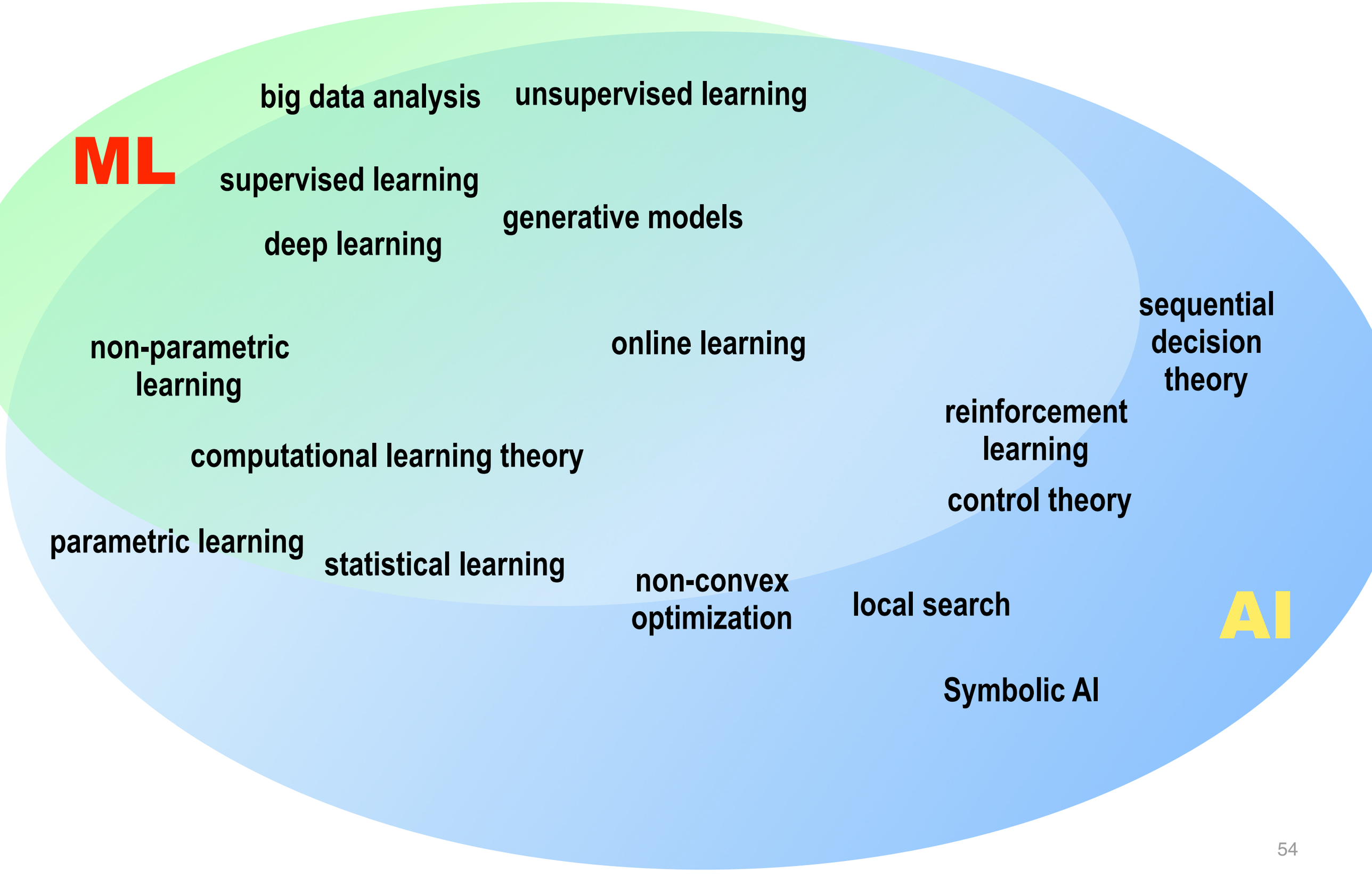
QML — the rest

Fun with variational circuits

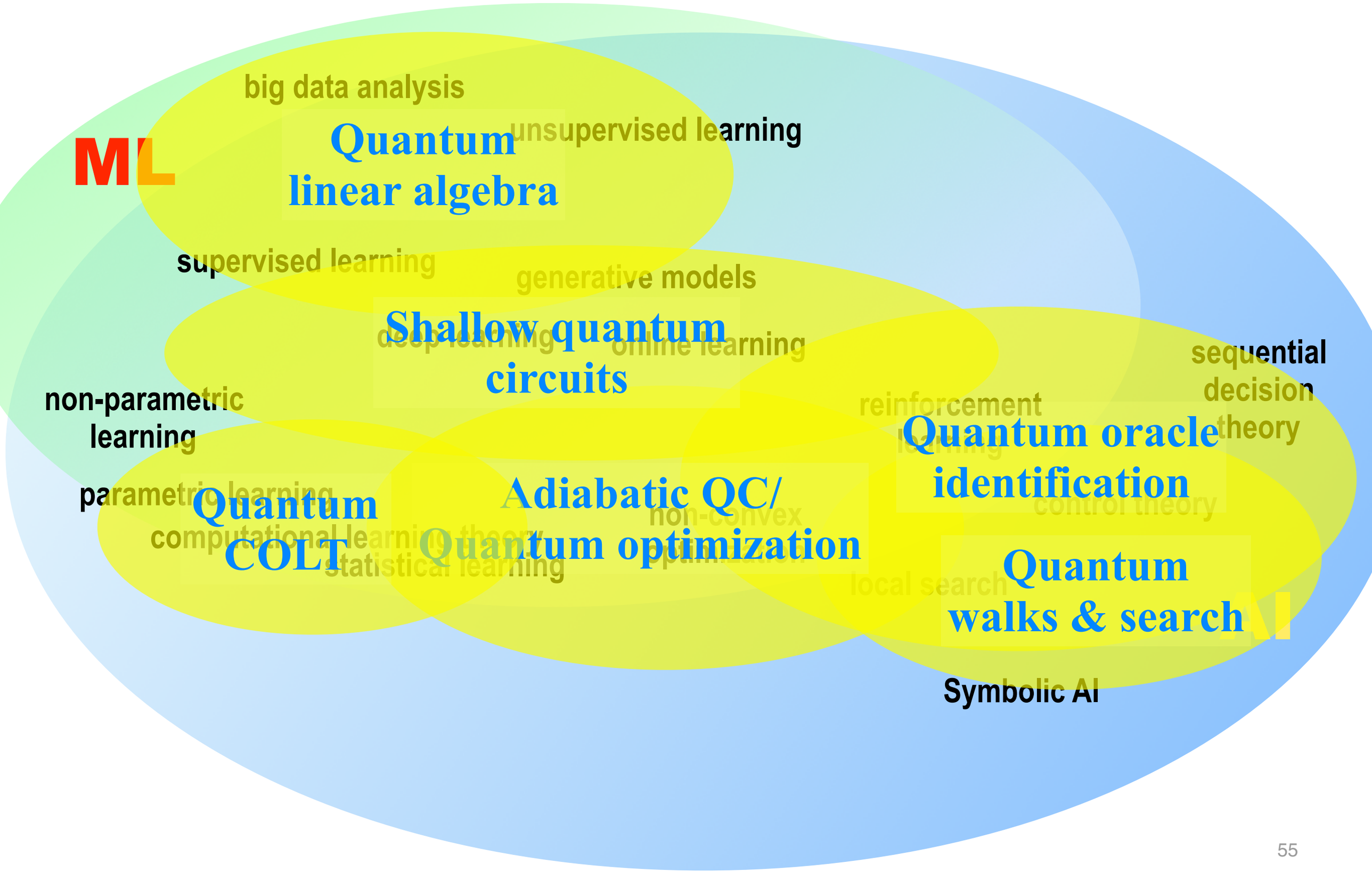
quantum convolutional NNs
data reuploading
quantum transfer learning
quantum reinforcement learning

Then there is all the rest…

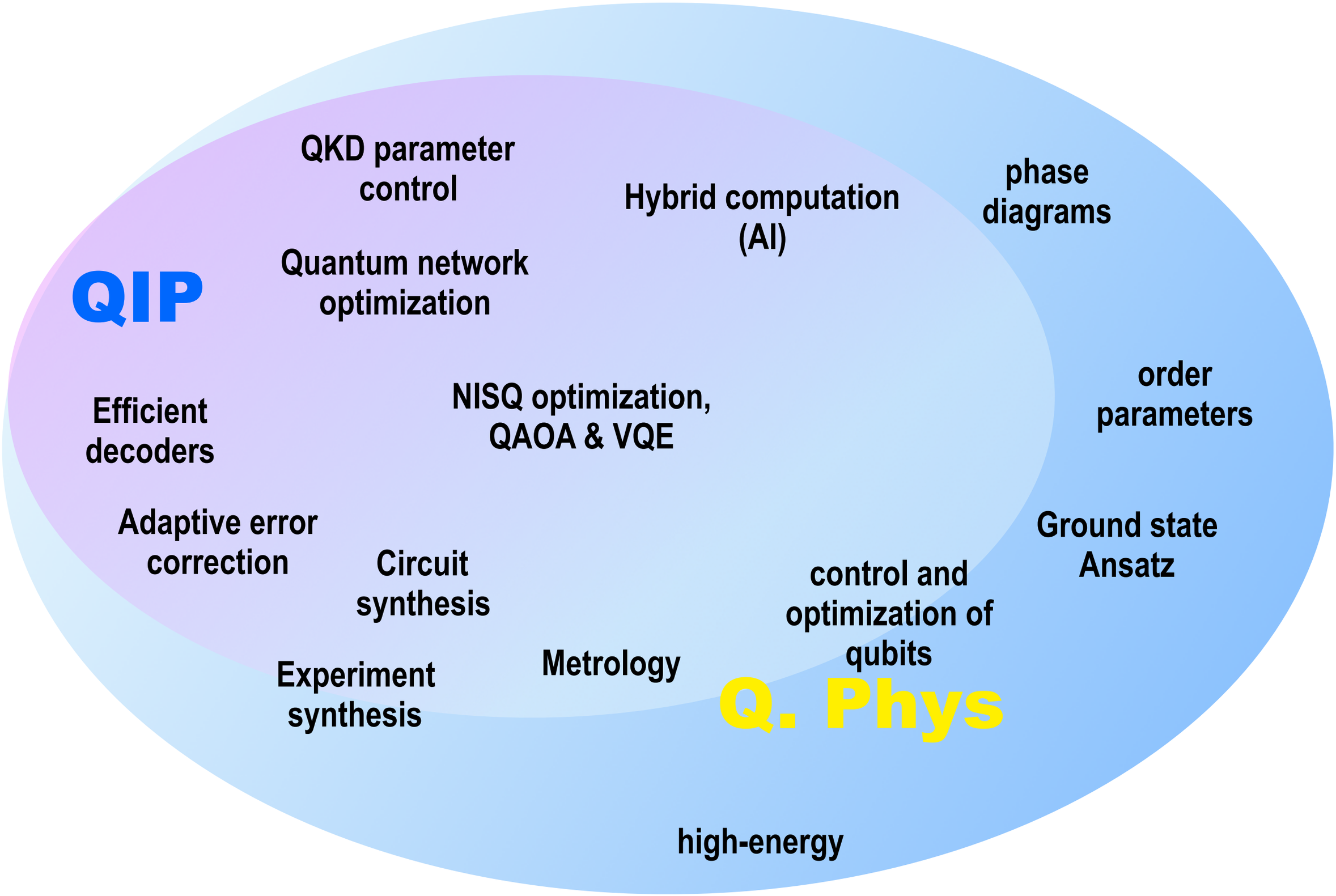
Machine learning is not one thing.
AI is not even a few things.

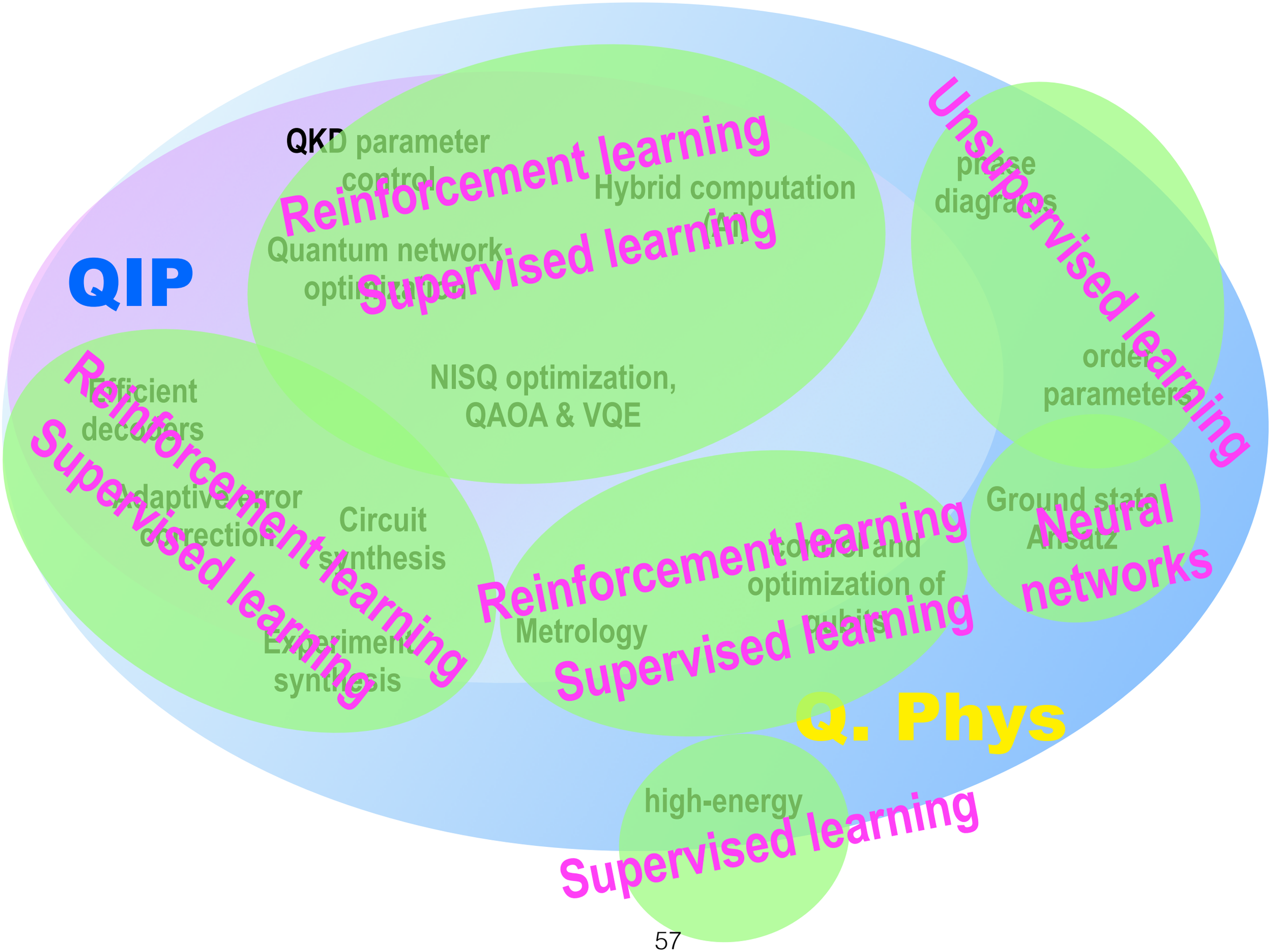


Quantum-enhanced ML



And then there's Quantum-applied ML!




QIP
QKD
QKD parameter control
Quantum network optimization
Reinforcement learning
Supervised learning
Hybrid computation (HQ)
NISQ optimization, QAOA & VQE
Unsupervised learning
phase diagrams
order parameters
Reinforcement learning
Supervised learning
Efficient decoders
Adaptive error correction
Circuit synthesis
Experiment synthesis
Reinforcement learning
Supervised learning
control and optimization of qubits
Metrology
Neural networks
Ground state Ansatz
Q. Phys
high-energy
Supervised learning