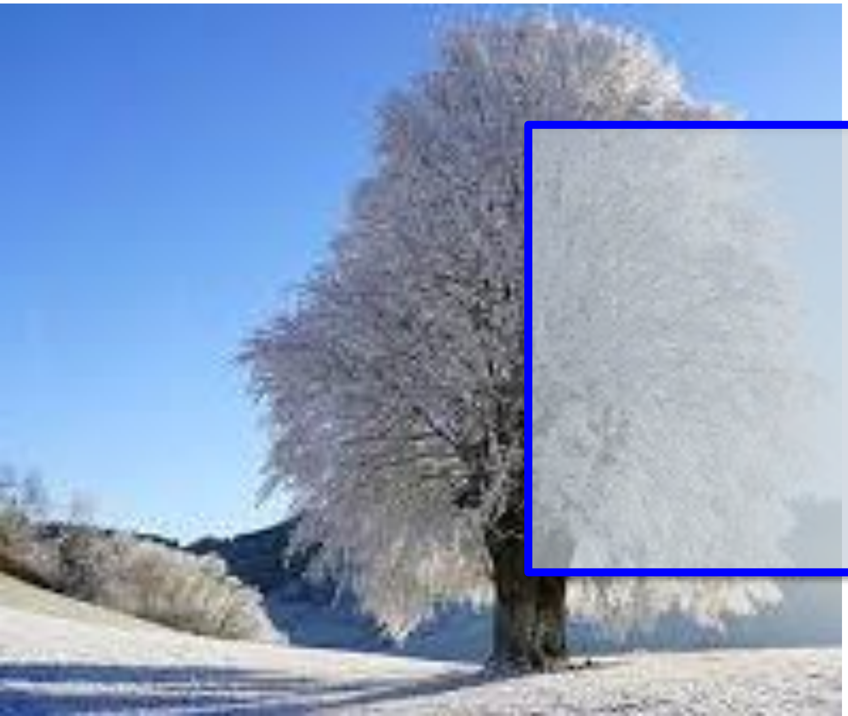
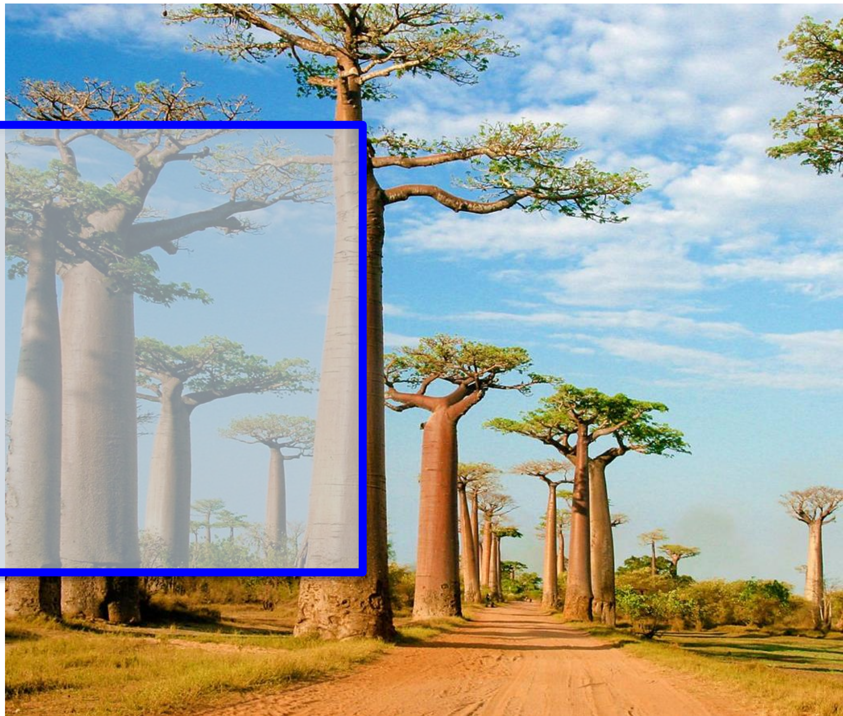




Diffusion

CS109, Stanford University

Learning Goals…

1. See **beautiful** math

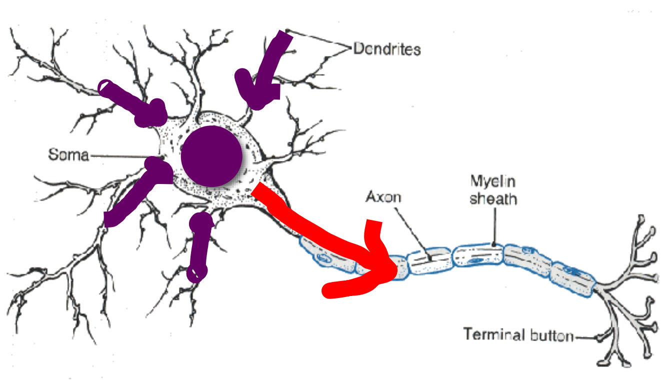
2. See **impactful** math

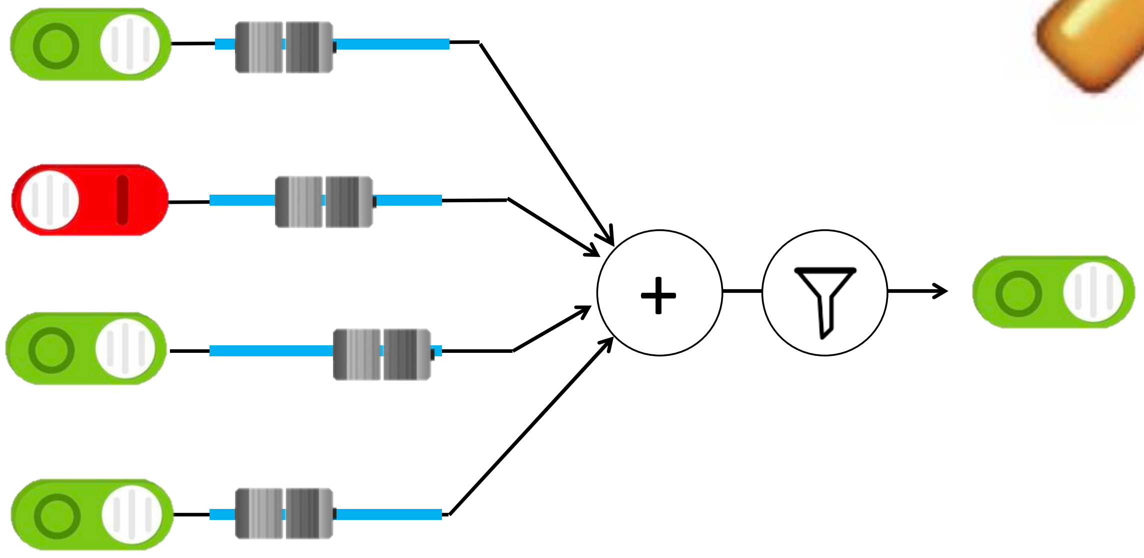
3. Review concepts

What would it take for **you** to invent diffusion?

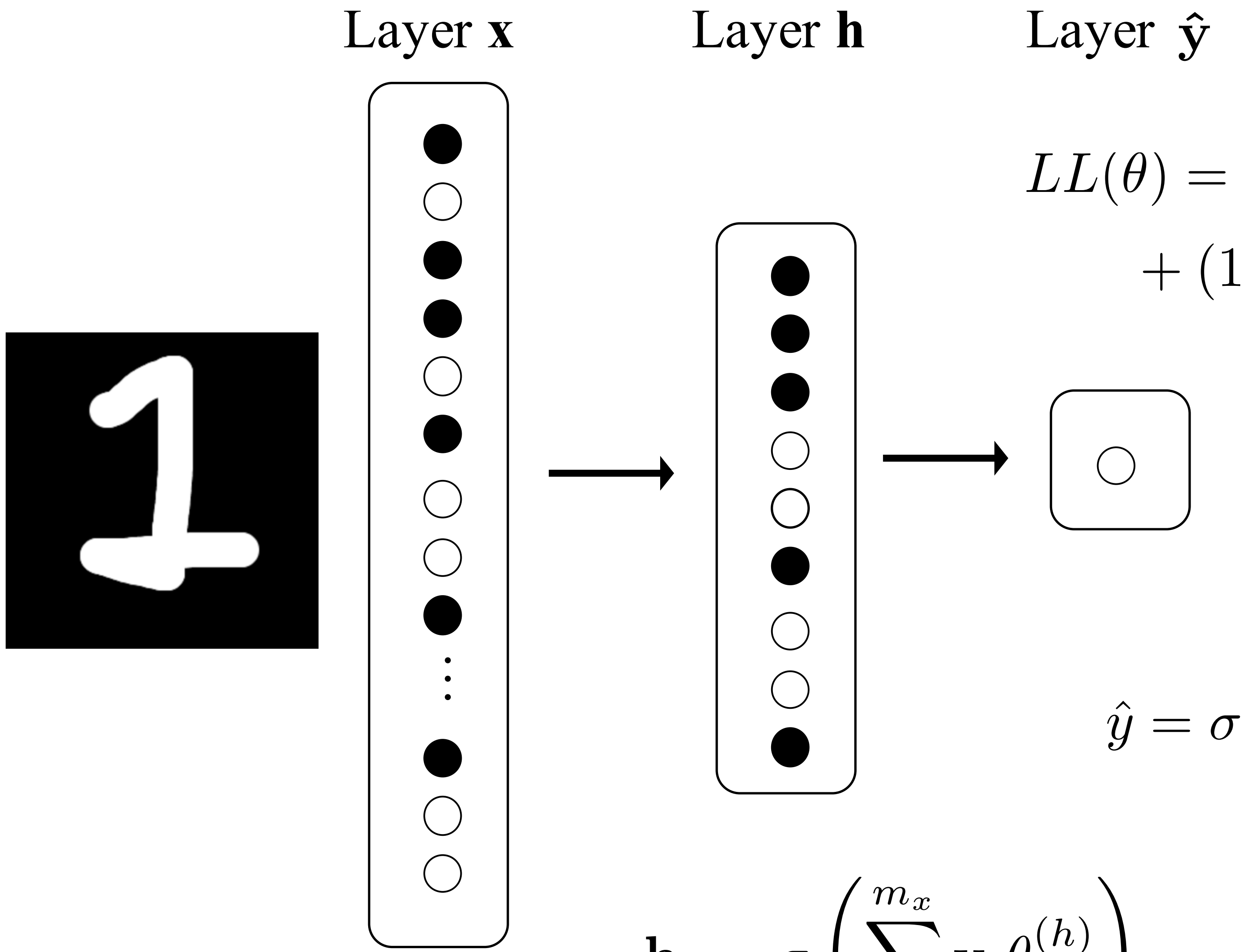
Review

Artificial Neurons





Deep Learning

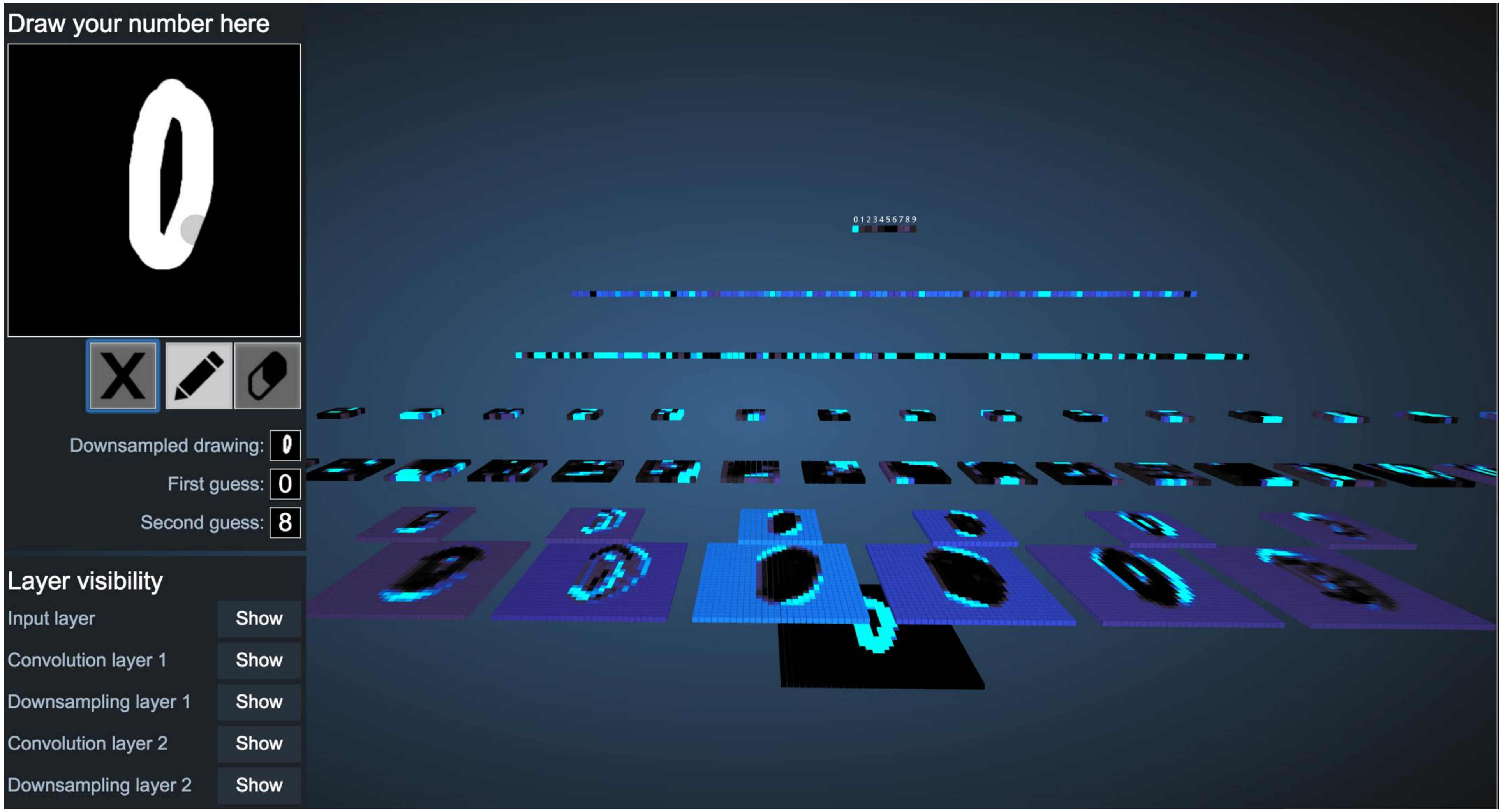


$$LL(\theta) = y \log \hat{y} + (1-y) \log[1-\hat{y}]$$

$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

$$\mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

Demonstration



https://web.archive.org/web/20211117115916/https://www.cs.ryerson.ca/~aharley/vis/conv/

Deep Learning Code

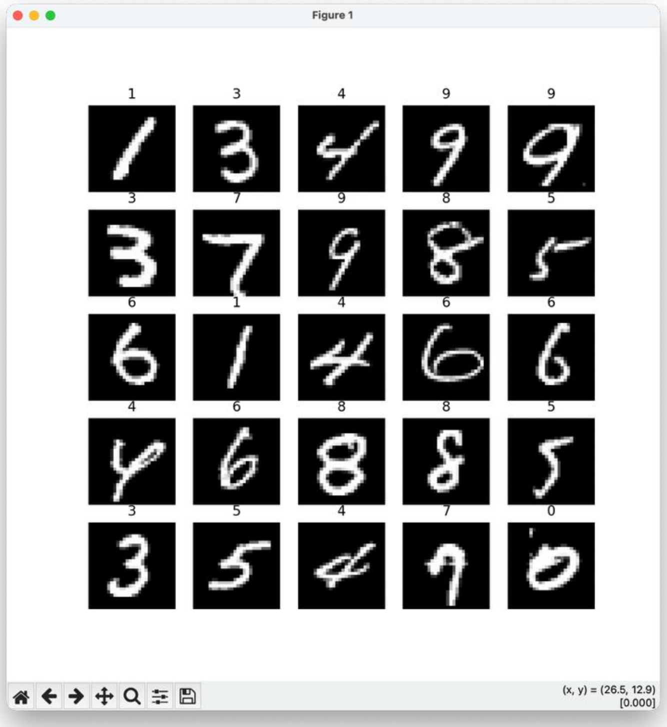
Build model

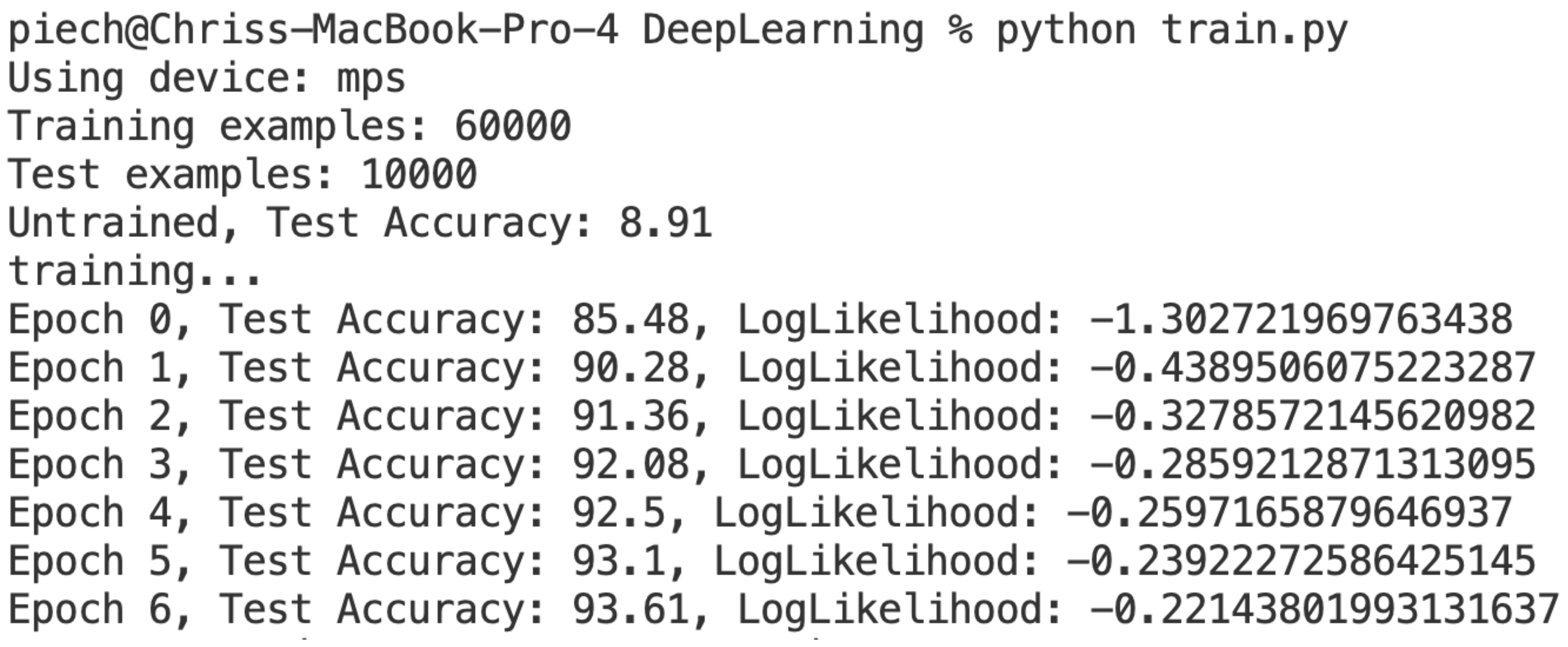
Train model

```
def run_train(train, model):
    model.train()
    loss_function = nn.NLLLoss(reduction="sum")
    optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)

    log_likelihoods = []

    for image_batch, truth_batch in train:
        # does one batch of images simultaneously
        optimizer.zero_grad()  # start with gradient zero
        pred = model(image_batch)  # predict the label
        loss = loss_function(pred, truth_batch) # calc loss
        loss.backward() # calculate gradients
        optimizer.step() # update parameters
        log_likelihoods.append(-loss.item()) # keep track of likelihoods

    return np.mean(log_likelihoods)
```

```
def main():
    # get the data
    train, test = download_data()
    print(f"Training examples: {len(train.dataset)}")
    print(f"Test examples: {len(test.dataset)}")

    # a very simple and fast nn
    model = nn.Sequential(
        nn.Flatten(),
        nn.Linear(28 * 28, 512),  # images are 28 x 28 pixels
        nn.Sigmoid(),
        nn.Linear(512, 512),
        nn.Sigmoid(),
        nn.Linear(512, 10),
        nn.LogSoftmax(dim=1),
    ).to(device)

    # see how bad it is before training
    accuracy = run_test(test, model, device)
    print(f"Untrained, Test Accuracy: {accuracy}")
```

Training our Classifier was fun!





```
piech@Chriss-MacBook-Pro-4 DeepLearning % python train.py
Using device: mps
Training examples: 60000
Test examples: 10000
Untrained, Test Accuracy: 8.91
training...
Epoch 0, Test Accuracy: 85.48, LogLikelihood: -1.302721969763438
Epoch 1, Test Accuracy: 90.28, LogLikelihood: -0.4389506075223287
Epoch 2, Test Accuracy: 91.36, LogLikelihood: -0.3278572145620982
Epoch 3, Test Accuracy: 92.08, LogLikelihood: -0.2859212871313095
Epoch 4, Test Accuracy: 92.5, LogLikelihood: -0.2597165879646937
Epoch 5, Test Accuracy: 93.1, LogLikelihood: -0.23922272586425145
Epoch 6, Test Accuracy: 93.61, LogLikelihood: -0.22143801993131637
```

Types of Machine Learning Tasks

Muli-Class Classification

Regression

Reinforcement Learning

Generation

End Review

I am curious…

How Does This Work?

create an image of a tree

Here's an image of a serene tree in a peaceful meadow.

another tree

Here's another beautiful tree, this time showcasing the vibrant colors of autumn.

Classification is easy, but sampling is hard?

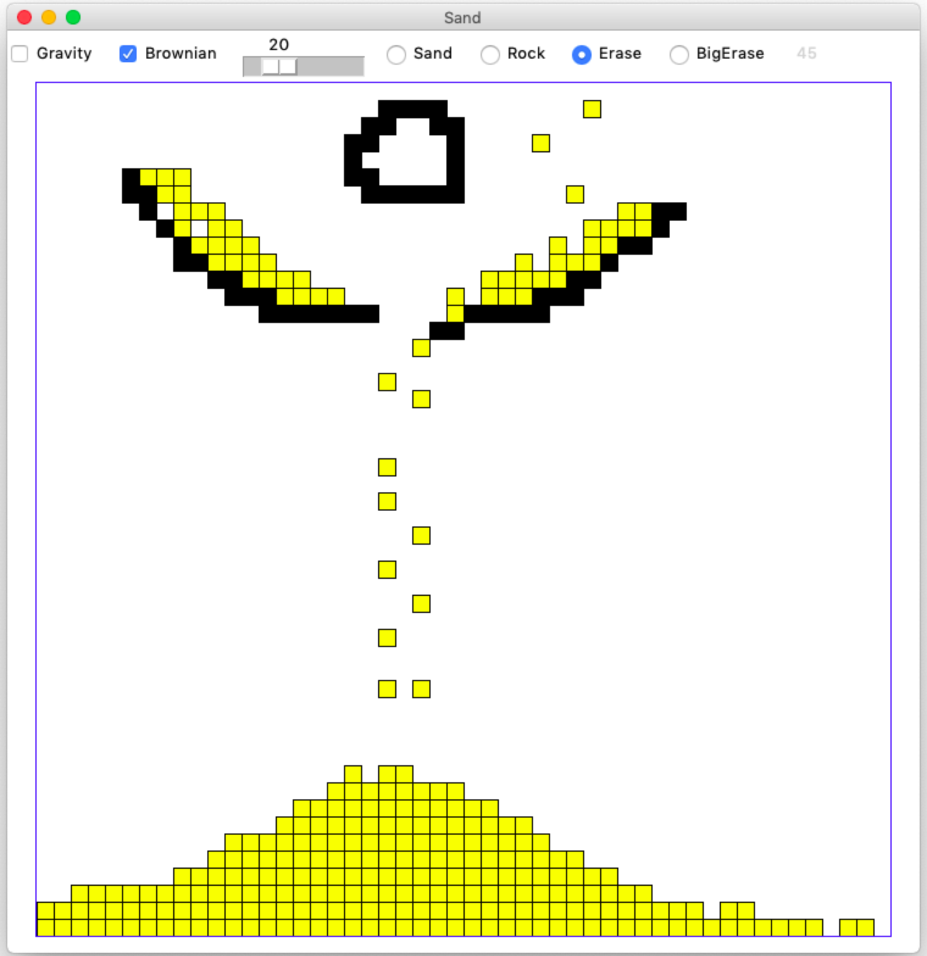
Goal: write an algorithm that can **generate** hand drawn number 8s

samples at step 236000 (class 8)

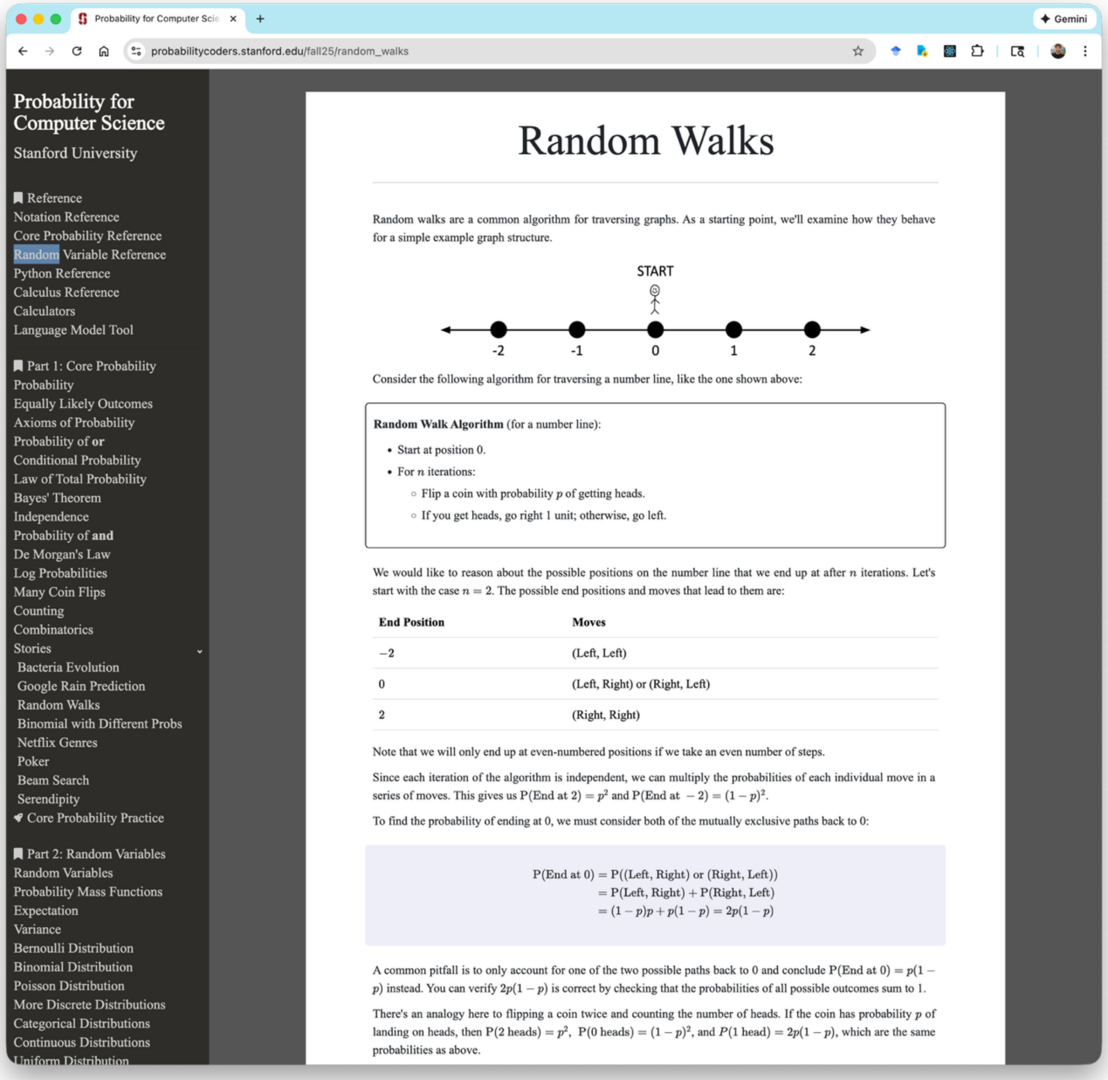
Challenge: can you come up with a (hacky, brute force) way to generate images if I give you a working classifier?

Stanford University

Random Walks



☐ Gravity [✓ Brownian] 20 ○ Sand ○ Rock [● Erase] ○ BigErase 45



Probability for Computer Scie
Gemini
probabilitycoders.stanford.edu/fall25/random_walks

Probability for Computer Science

Stanford University

Random Walks

Random walks are a common algorithm for traversing graphs. As a starting point, we'll examine how they behave for a simple example graph structure.

Consider the following algorithm for traversing a number line, like the one shown above:

Random Walk Algorithm (for a number line):

- Start at position 0.
- For n iterations:
  - Flip a coin with probability p of getting heads.
  - If you get heads, go right 1 unit; otherwise, go left.

We would like to reason about the possible positions on the number line that we end up at after n iterations. Let's start with the case $n = 2$. The possible end positions and moves that lead to them are:

End Position	Moves
-2	(Left, Left)
0	(Left, Right) or (Right, Left)
2	(Right, Right)

Note that we will only end up at even-numbered positions if we take an even number of steps.

Since each iteration of the algorithm is independent, we can multiply the probabilities of each individual move in a series of moves. This gives us $\text{P}(\text{End at } 2) = p^2$ and $\text{P}(\text{End at } -2) = (1-p)^2$.

To find the probability of ending at 0, we must consider both of the mutually exclusive paths back to 0:

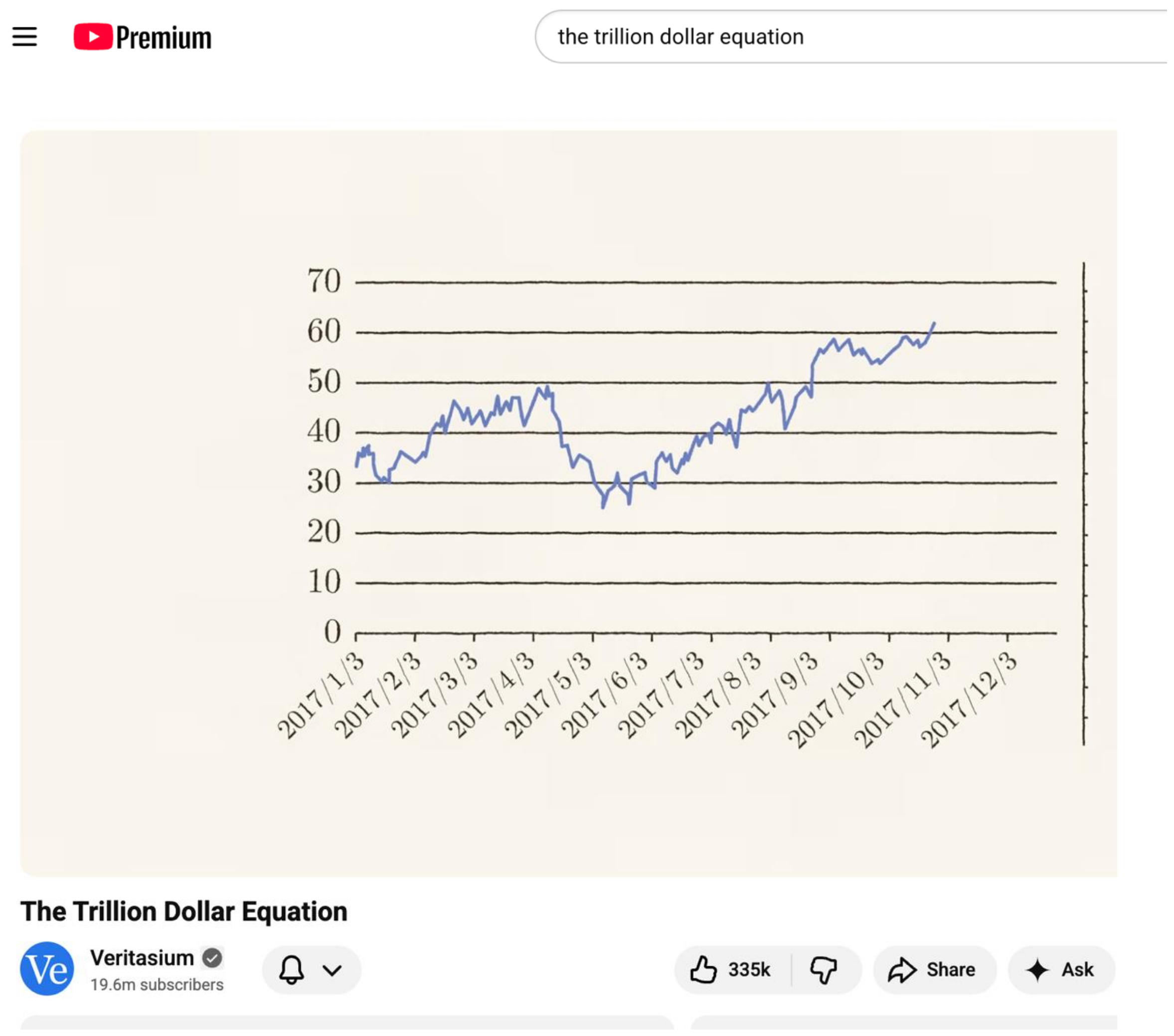
$$\begin{aligned}\text{P}(\text{End at } 0) &= \text{P}((\text{Left, Right}) \text{ or } (\text{Right, Left})) \\ &= \text{P}(\text{Left, Right}) + \text{P}(\text{Right, Left}) \\ &= (1-p)p + p(1-p) = 2p(1-p)\end{aligned}$$

A common pitfall is to only account for one of the two possible paths back to 0 and conclude $\text{P}(\text{End at } 0) = p(1-p)$ instead. You can verify $2p(1-p)$ is correct by checking that the probabilities of all possible outcomes sum to 1.

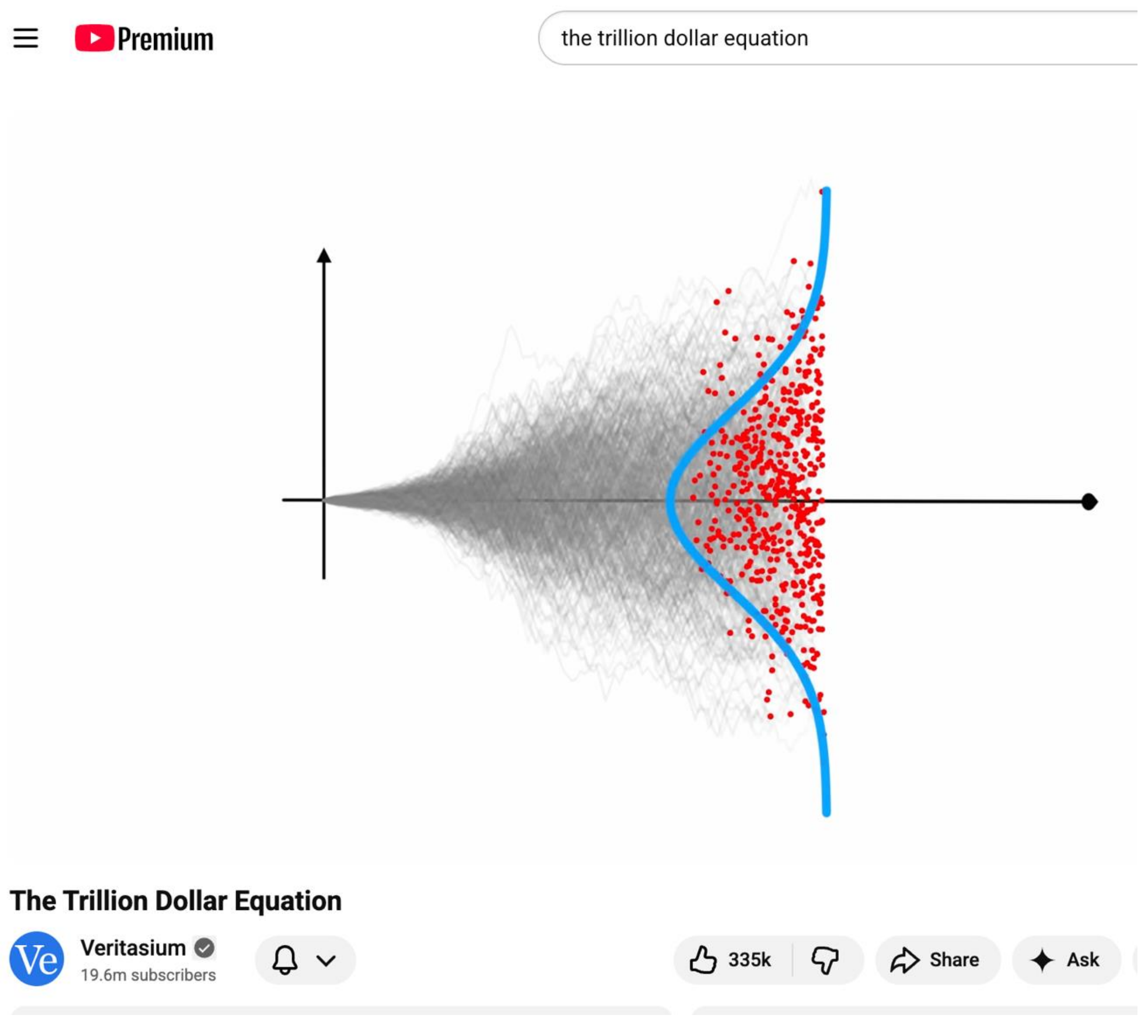
There's an analogy here to flipping a coin twice and counting the number of heads. If the coin has probability p of landing on heads, then $\text{P}(2 \text{ heads}) = p^2$, $\text{P}(0 \text{ heads}) = (1-p)^2$, and $P(1 \text{ head}) = 2p(1-p)$, which are the same probabilities as above.

The "brownian" parameter is a number in the range 0..100 inclusive. When brownian is 20, that means there is a 20% chance that each sand will randomly try to move one square left or right each turn.

Random Walks -> Trillion Dollars



The Trillion Dollar Equation



The Trillion Dollar Equation

Black-Sholes Equation is based off thinking of stocks as random walks

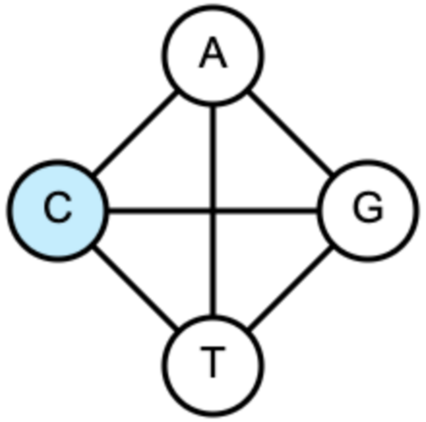
DNA Mutation Clocks

Now let's look at a more complex graph to analyze a problem about DNA sequence mutations.

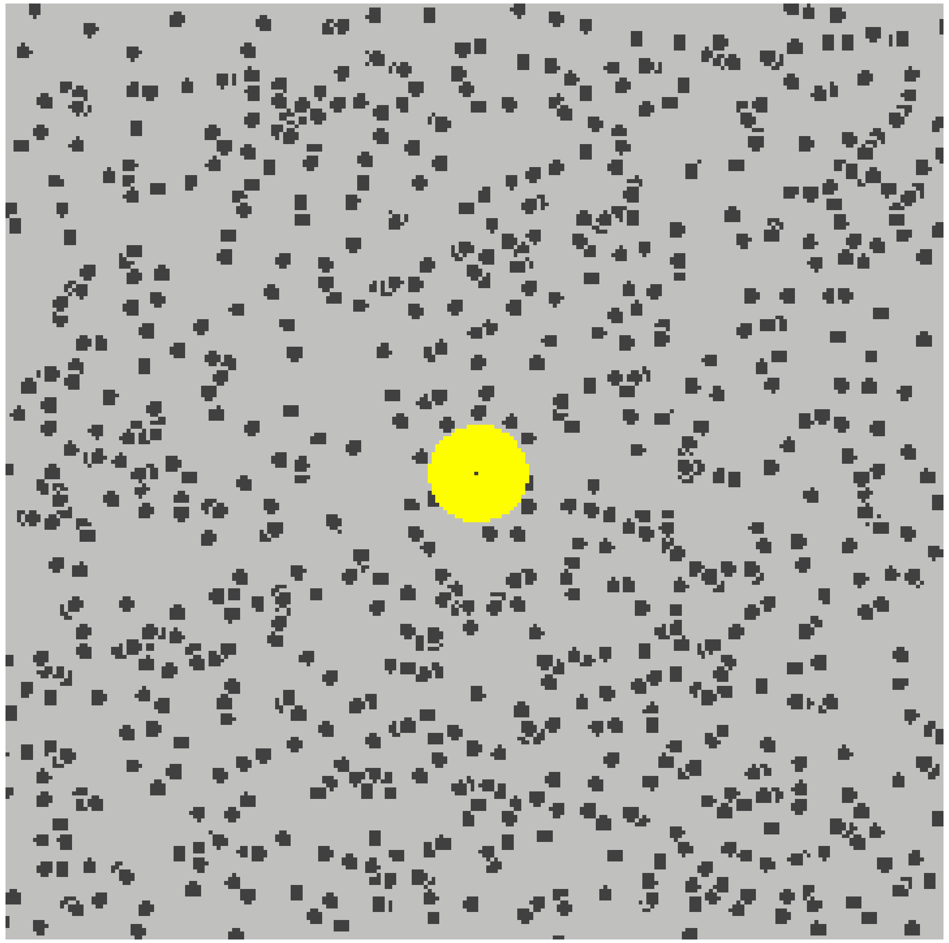
A species of bacteria is rapidly multiplying inside a dish. Each time a bacterium clones itself, it copies its DNA sequence (a very long ordered list made up of the letters A, C, G, and T). DNA copying is an error-prone process, so mutations sometimes occur -- i.e., at a certain position in the DNA sequence, the letter A changes to, say, a C. Each time a mutation occurs, the original letter changes to one of the three other letters with equal likelihood.

Assume that before any mutations occur, there is an A at a particular position in the DNA sequence. What is the probability that after n mutations, there is an A at this position again?

This problem might not sound like it's about a random walk at first, but it can be represented as one!



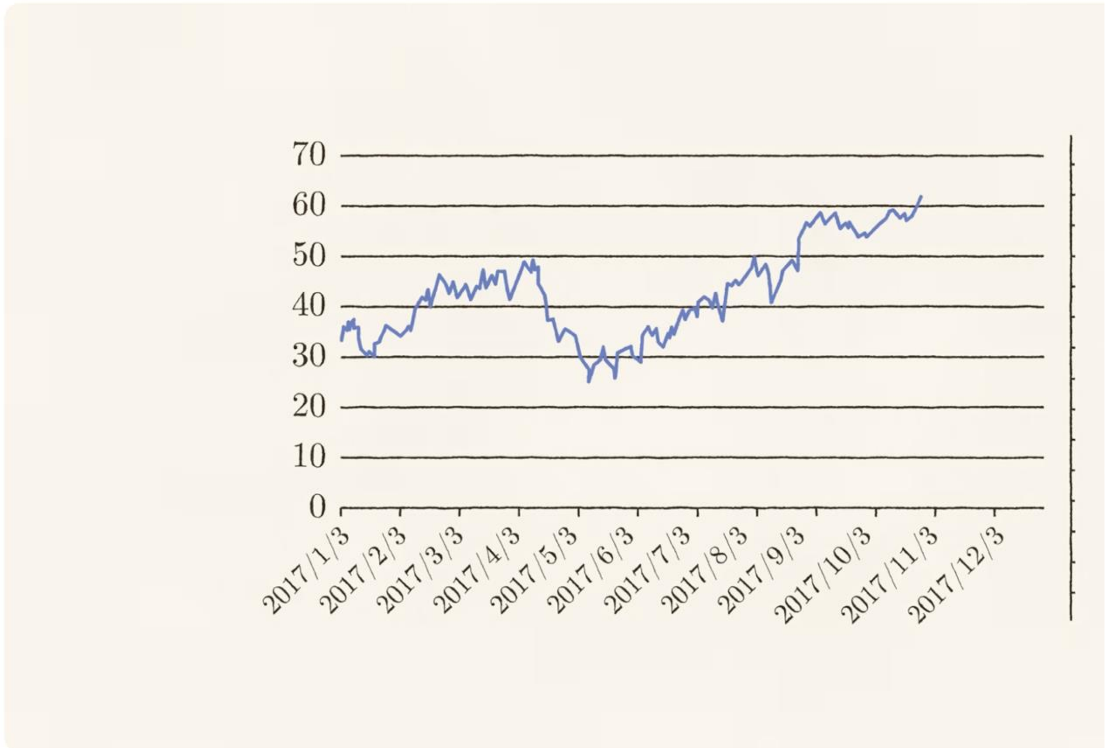
Diffusion in Physics!



Many **cool problems** related to random walks…

Today’s mystery: Can you infer where you came from?

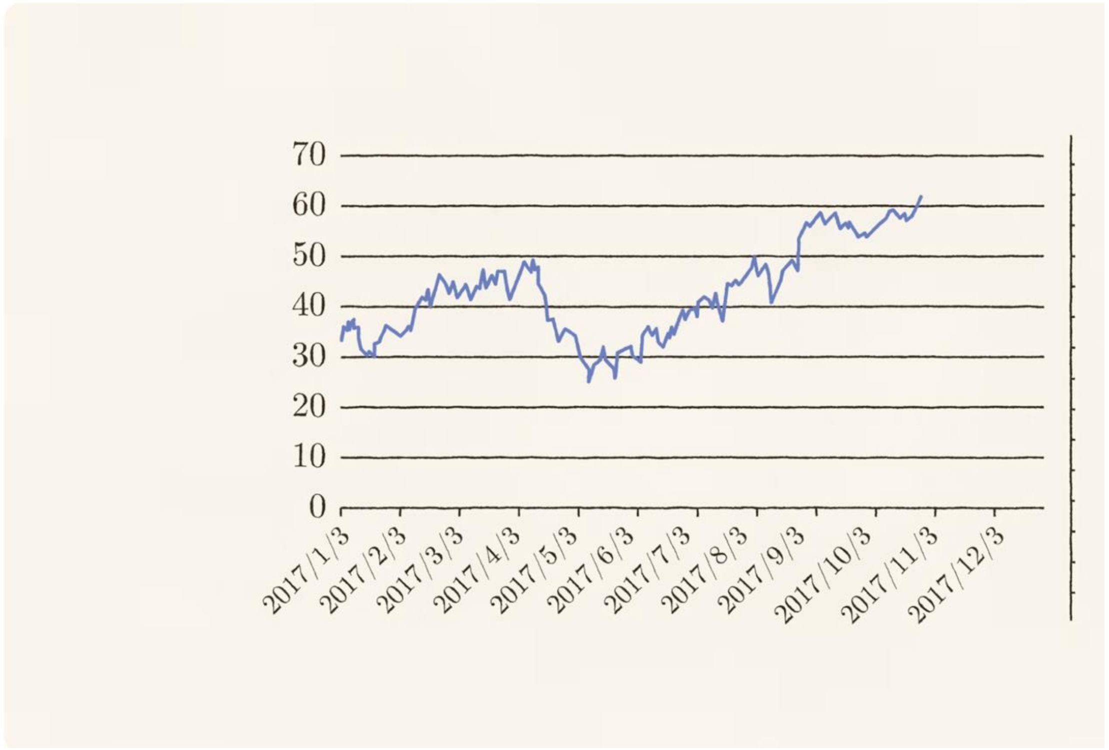
Random Walks



$$X_0 = 30$$

$$X_{t+1} = X_t + N_t \qquad N_i \sim N(\mu = 0, \overset{0.01}{\overset{\downarrow}{\sigma^2}})$$

Random Walks

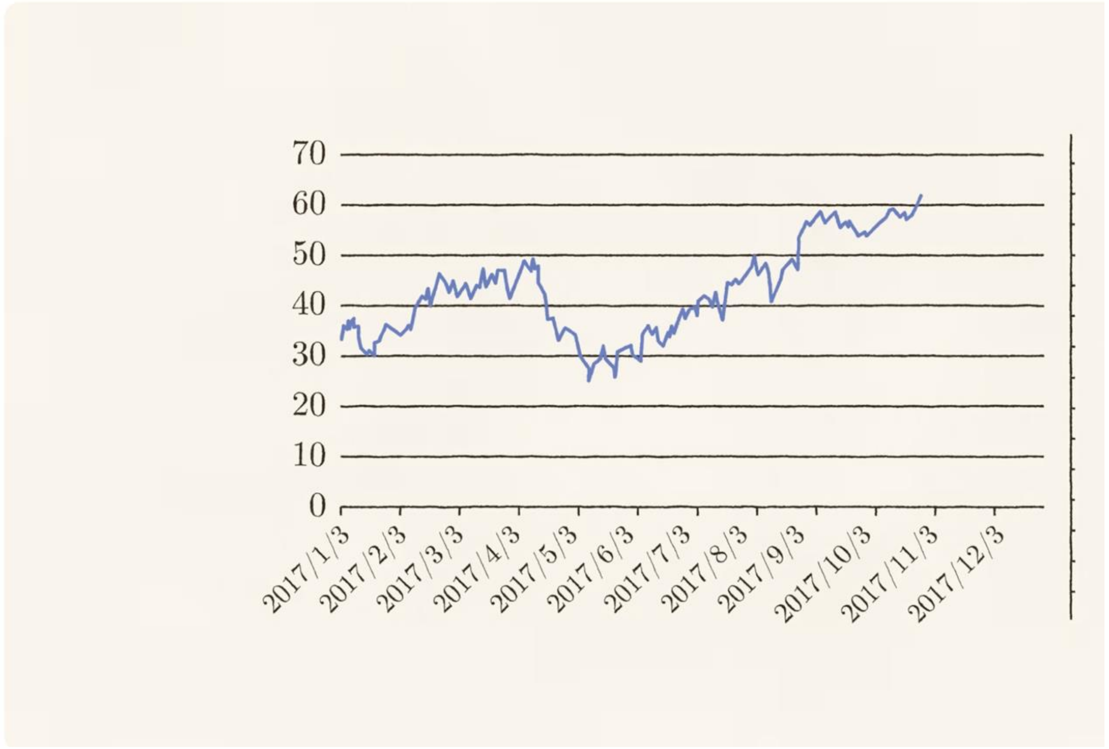


$$X_0 = 30$$

$$X_{t+1} = X_t + N_t \qquad N_i \sim N(\mu = 0, \overset{0.01}{\overset{\downarrow}{\sigma^2}})$$

Warmup: What is the distribution of X1 ?

Random Walks



$$X_0 = 30$$

$$X_{t+1} = X_t + N_t \qquad N_i \sim N(\mu = 0, \overset{0.01}{\sigma^2})$$

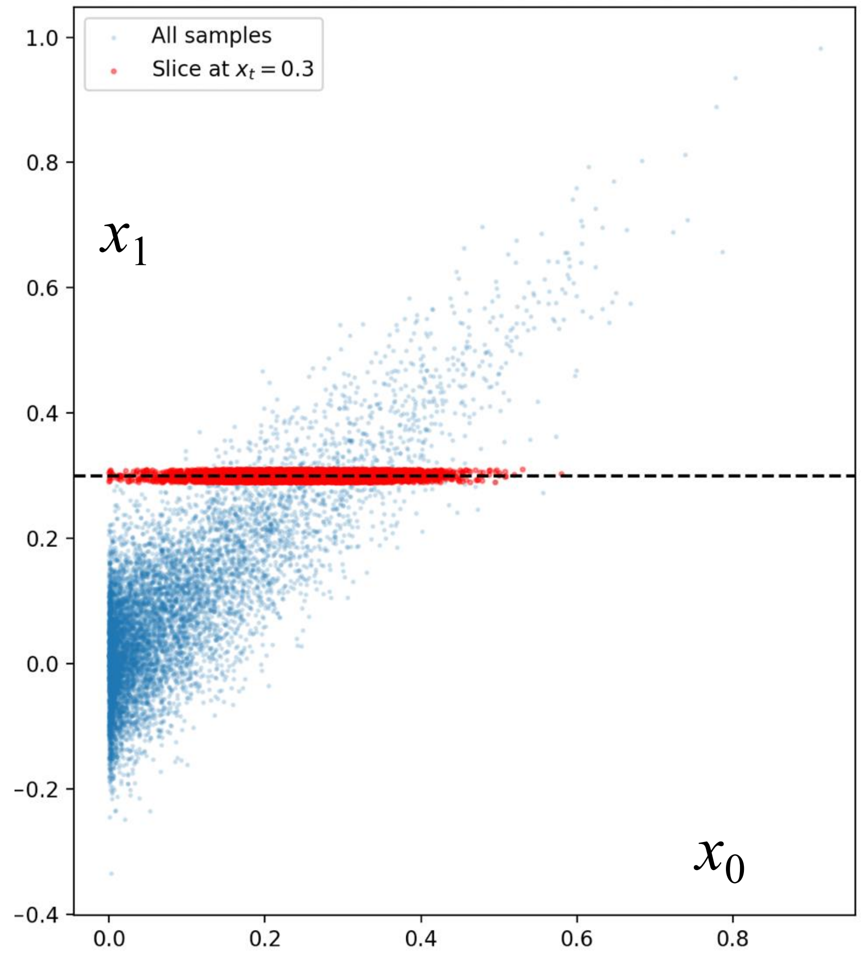
What is the probability distribution of X_{100}?

You observe $X_{101} = 10$

What is the probability density of X_{100}?

Random Walks

$$P(X_0 = x_0 \,, X_1 = x_1\,)$$



$$X_0 \sim \text{Beta}(2,\ 7)$$

0.01

$$X_{t+1} = X_t + N_t \qquad N_i \sim N(\mu = 0, \sigma^2)$$

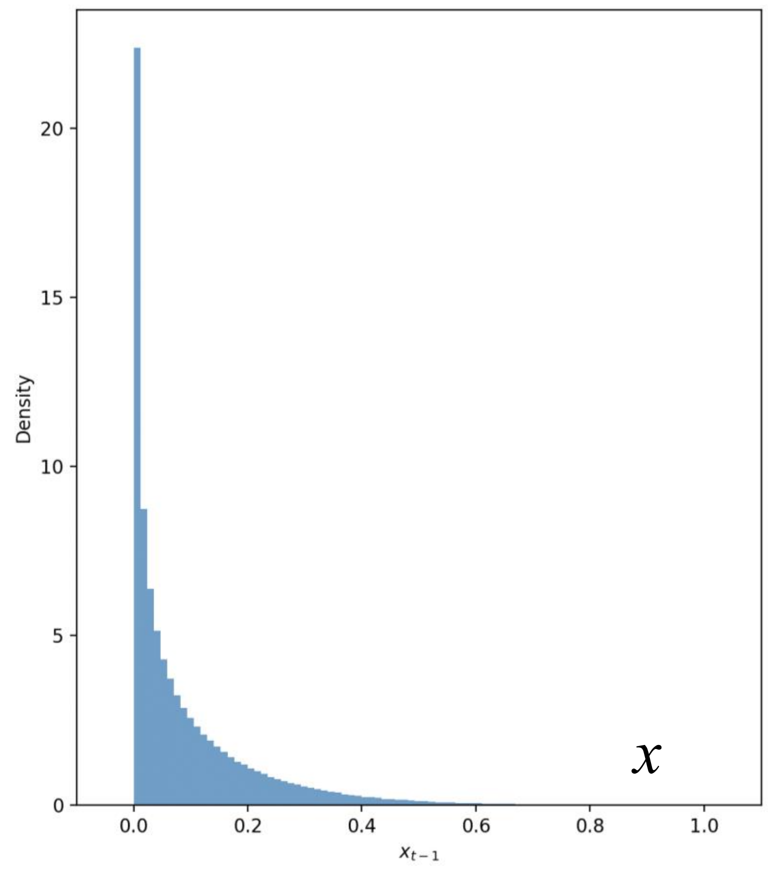
You observe $X_1 = 0.3$

What is the probability density of X_0?

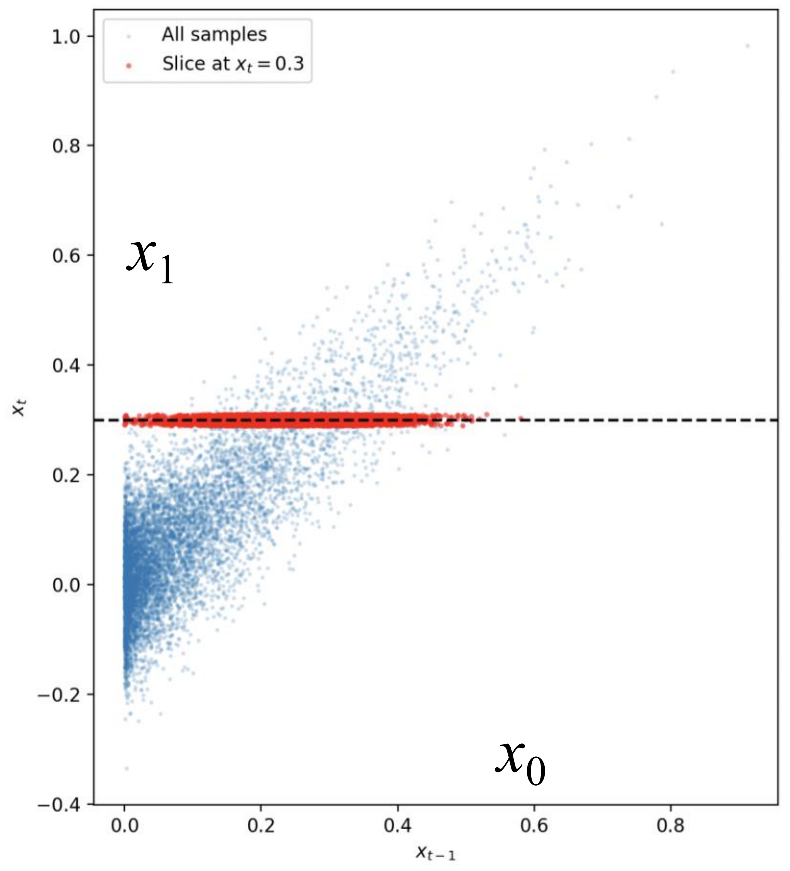
To the course reader!

This One Strange Fact

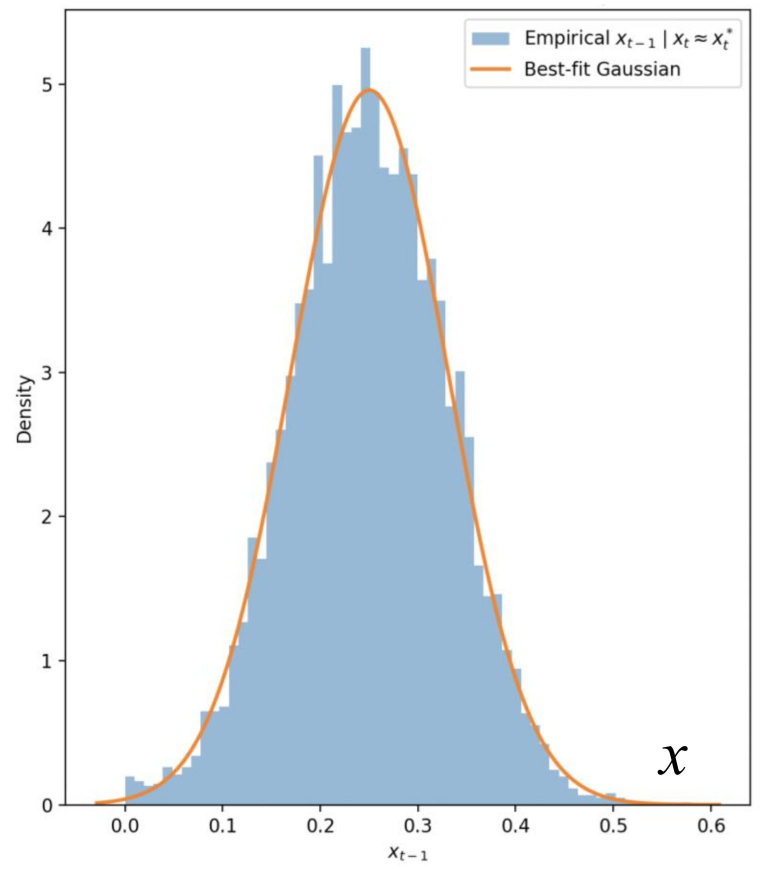
$P(X_0 = x\ )$



$P(X_0 = x_0\ ,\ X_1 = x_1\ )$

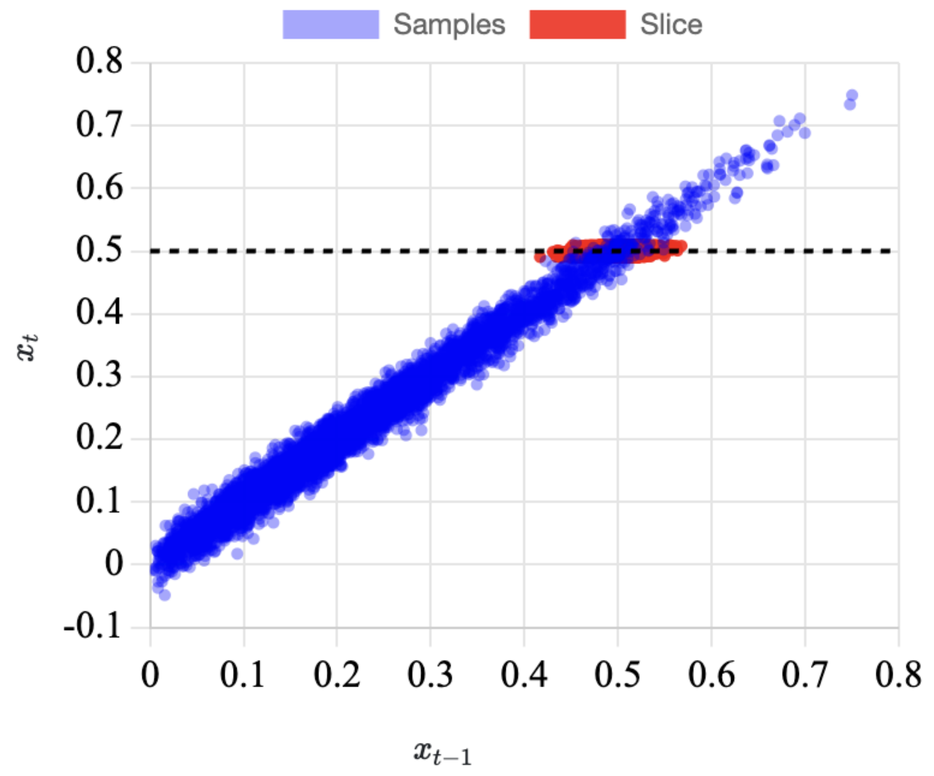


$P(X_0 = x \mid X_1 = 0.3)$

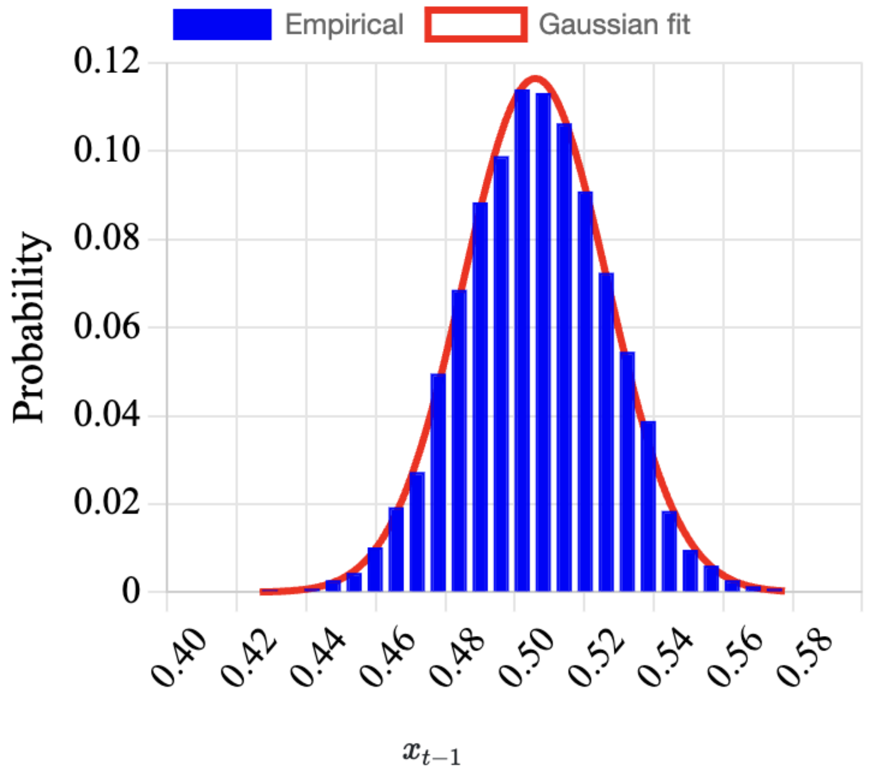


This One Strange Fact: Generalizes

Joint $P(X_{t-1} = x_{t-1}, X_t = x_t)$



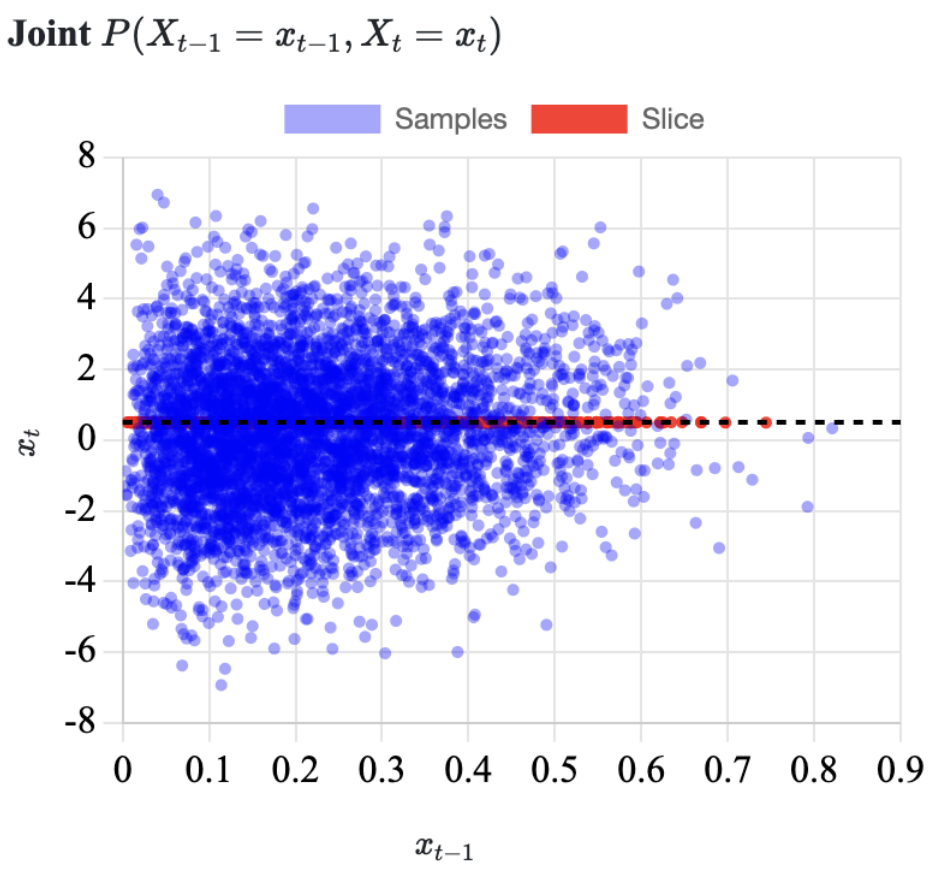
Posterior $P(X_{t-1} = x_{t-1} \mid X_t \approx k)$

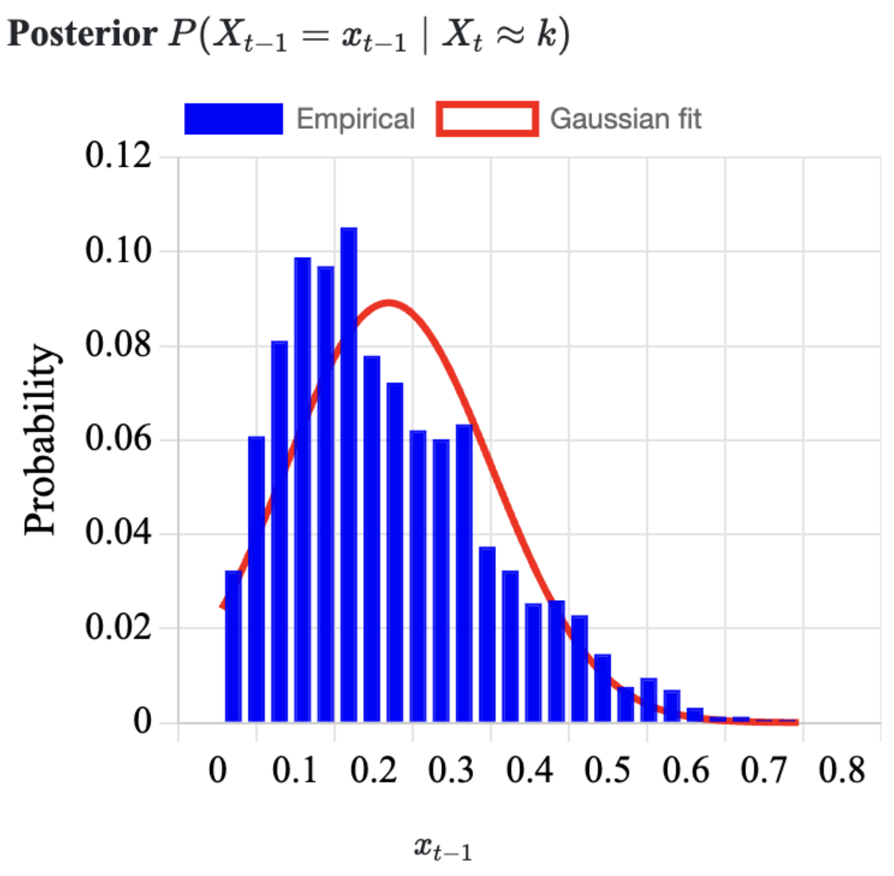


k: 0.5 σ: 0.02 Slice width: 0.01 Samples in slice: **3631**

https://probabilitycoders.stanford.edu/fall25/diffusion

This One Strange Fact: Generalizes (if σ is small)





k: 0.5 σ: 2 Slice width: 0.01 Samples in slice: **1579**

https://probabilitycoders.stanford.edu/fall25/diffusion

Diffusion Critical Fact

If

$$X_{t+1} = X_t + N_t \qquad N_i \sim N(\mu = 0, \sigma^2)$$

This must be a small known value

Then

$$X_t | X_{t+1} \sim N(\mu, \sigma^2)$$

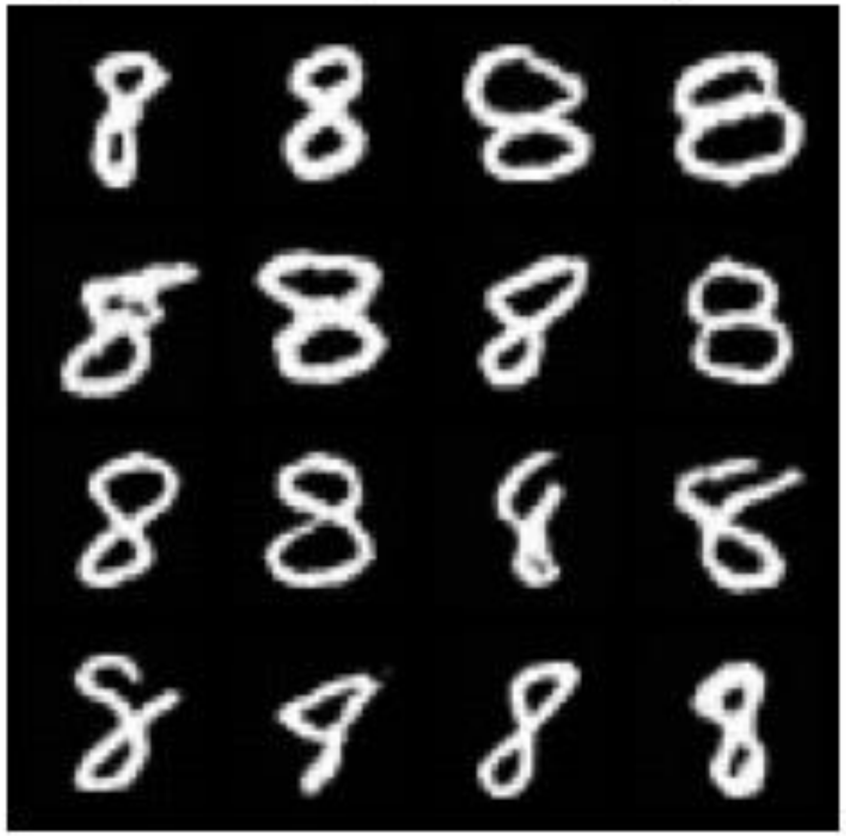
This value is unknown. Must figure it out

This holds regardless of the prior distribution of X_t

TLDR: We just need a neural network that can predict the mean of the previous timestep (and we get a backwards process we can sample from)

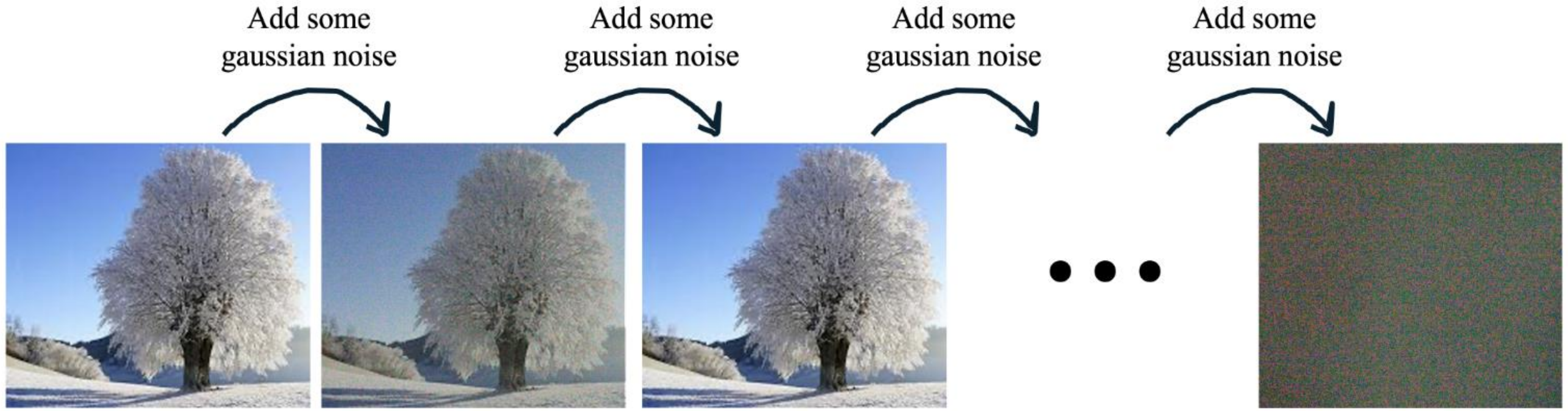
I generated this 8s this morning!

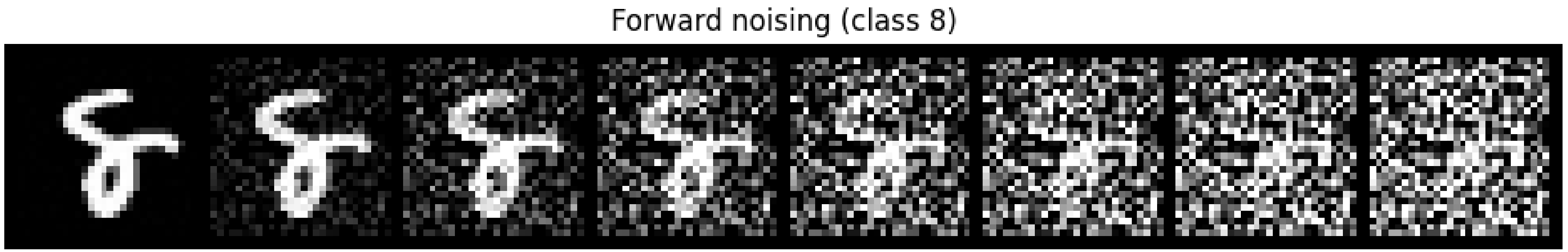
samples at step 236000 (class 8)



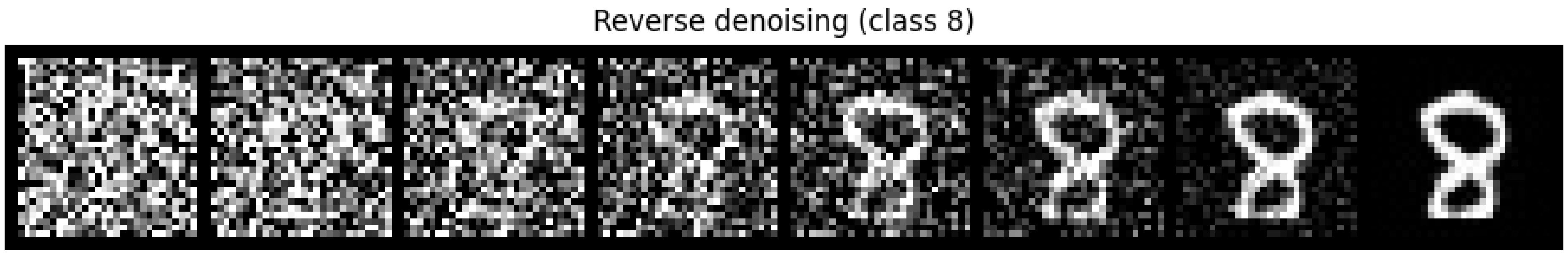
How?

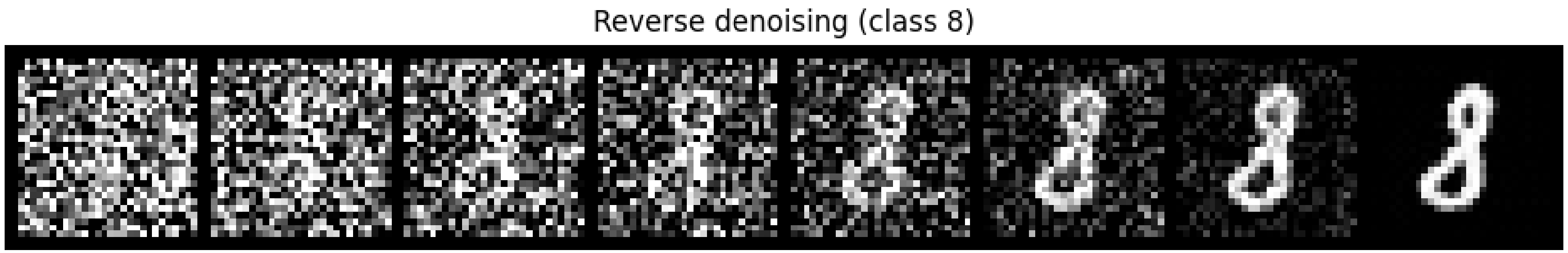
Take Your Images on a Walk!

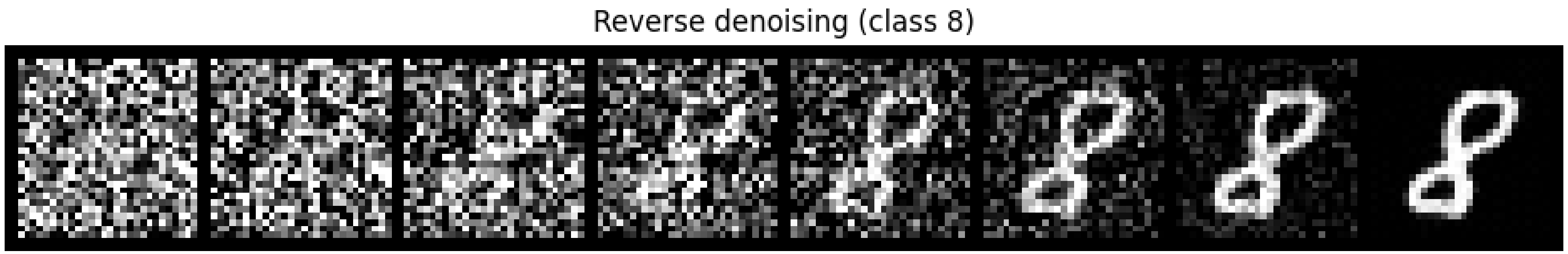


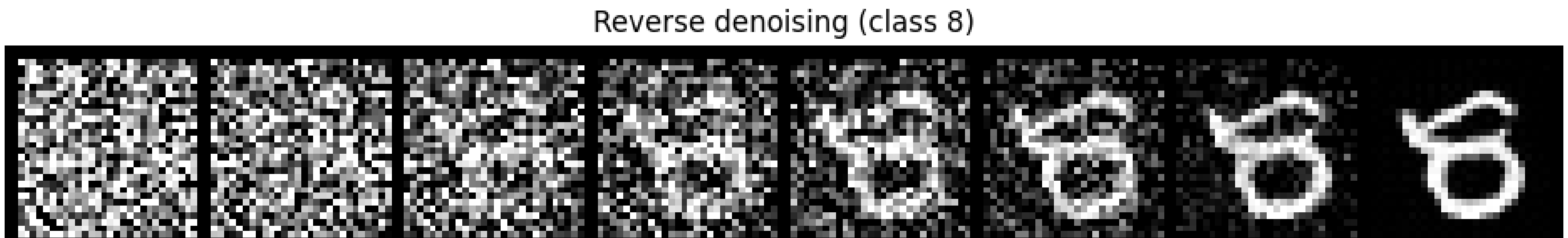


Goal: Learn how to remove noise!

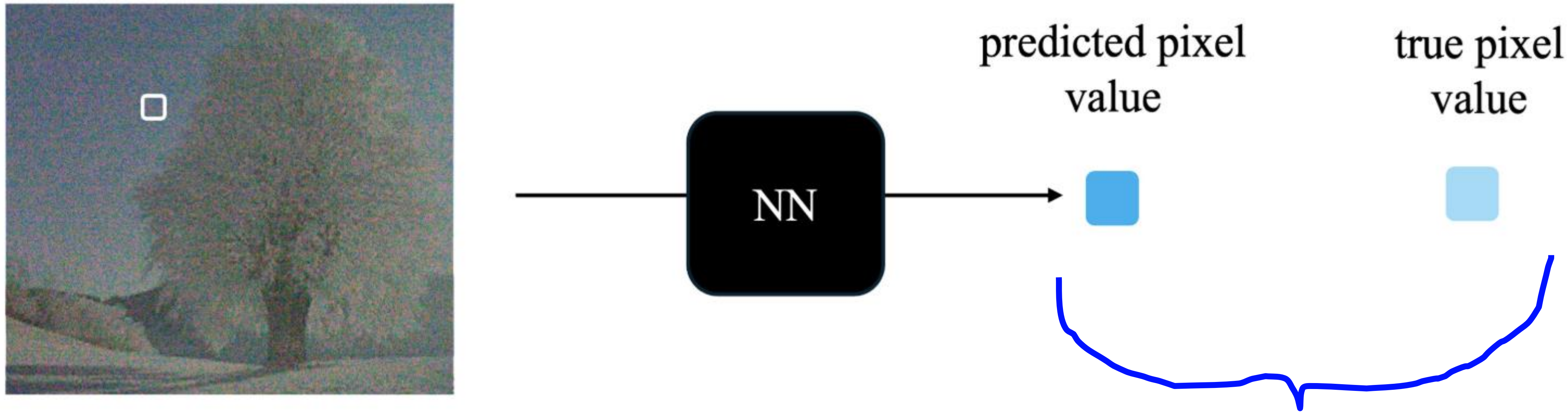






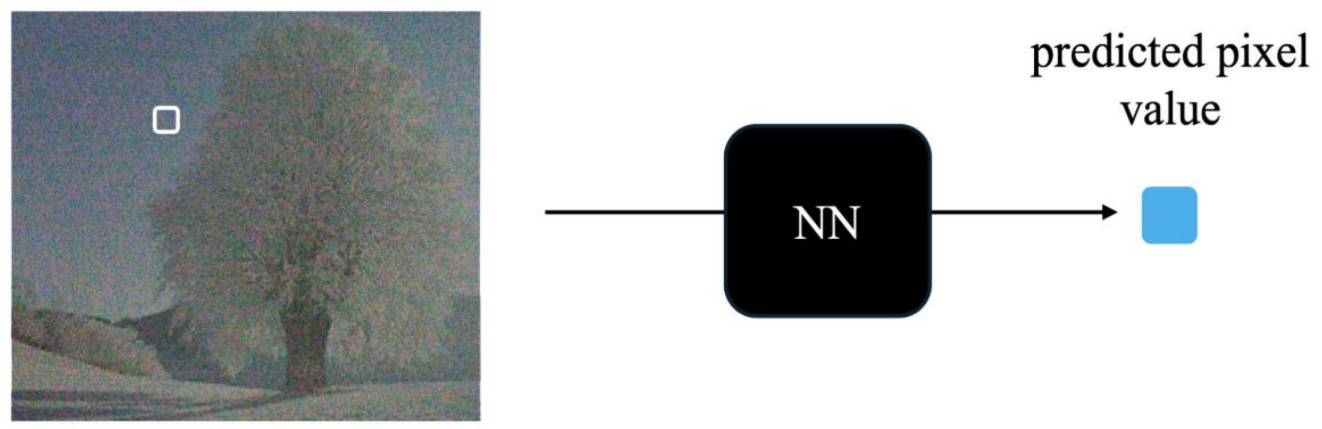


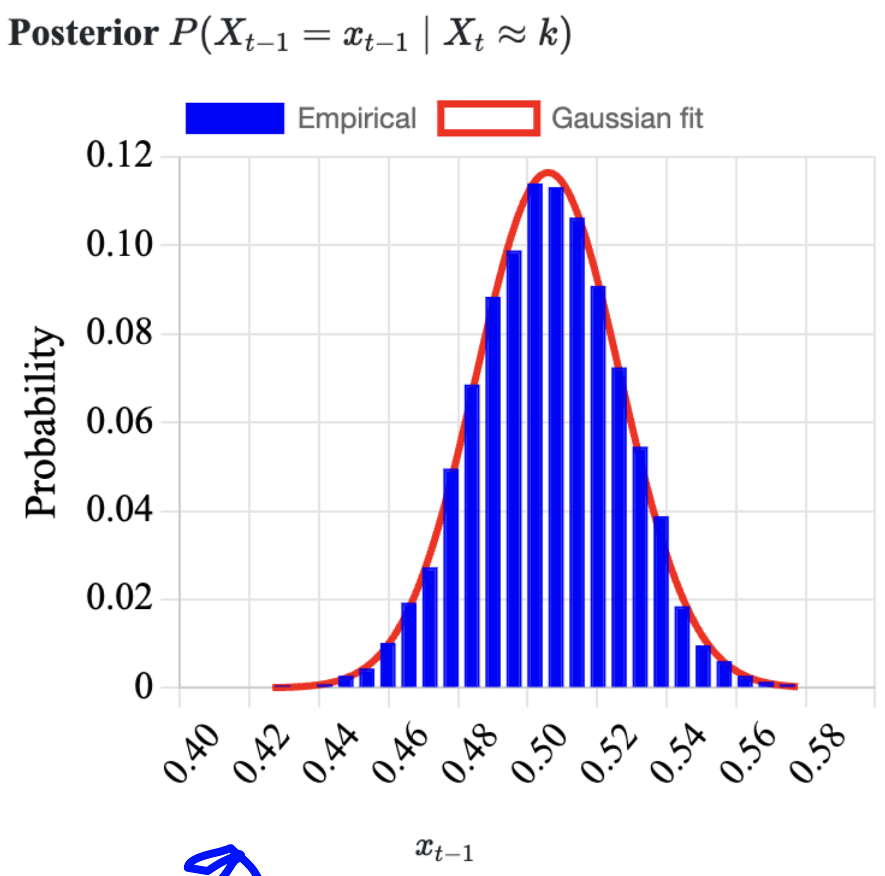
We Need a Neural Network that Denoises



Used KL divergence to show you just need to predict the squared error

Learning the mean is all we need!



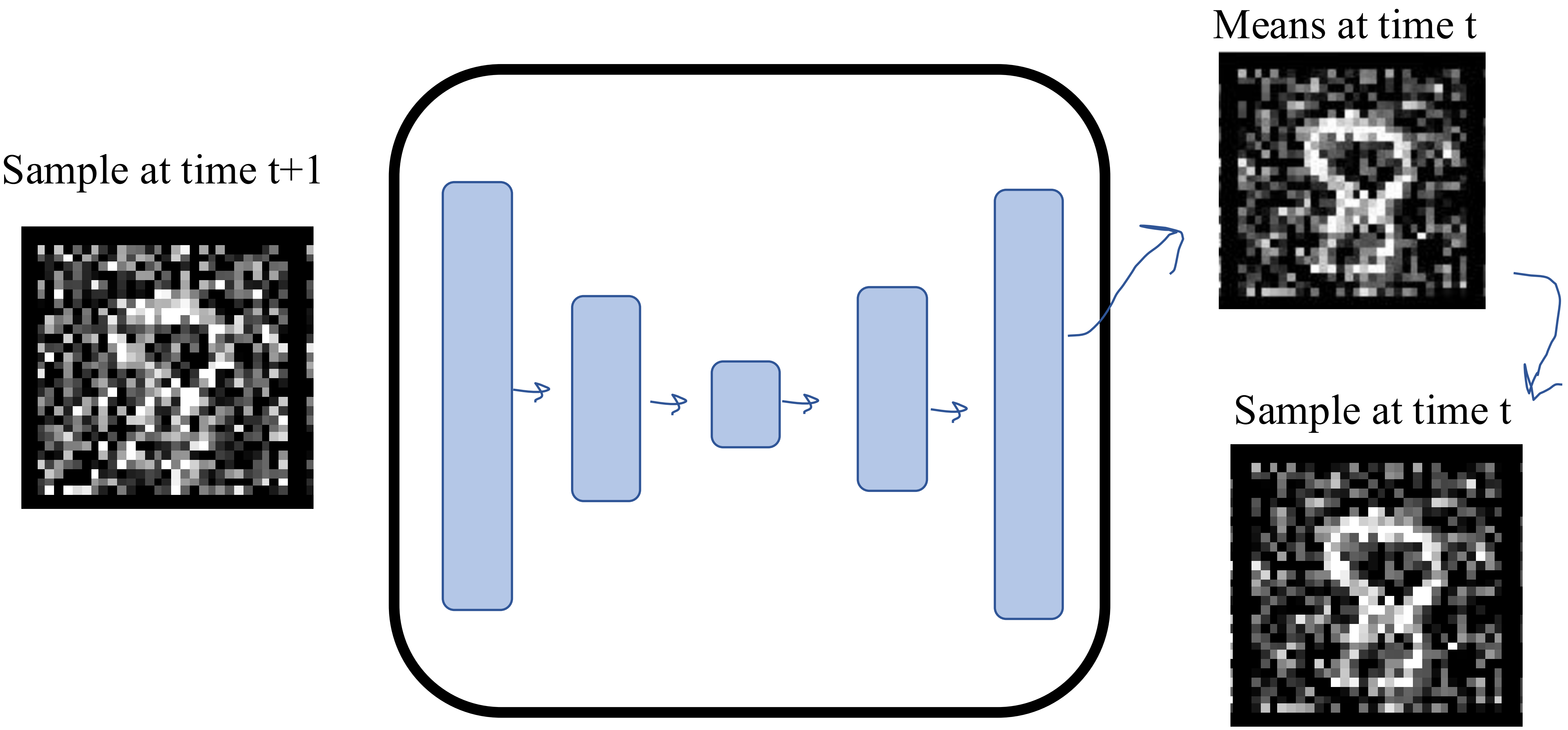


This is the mean of the posterior gaussian

Now you have a distribution for the reverse process

https://probabilitycoders.stanford.edu/fall25/diffusion

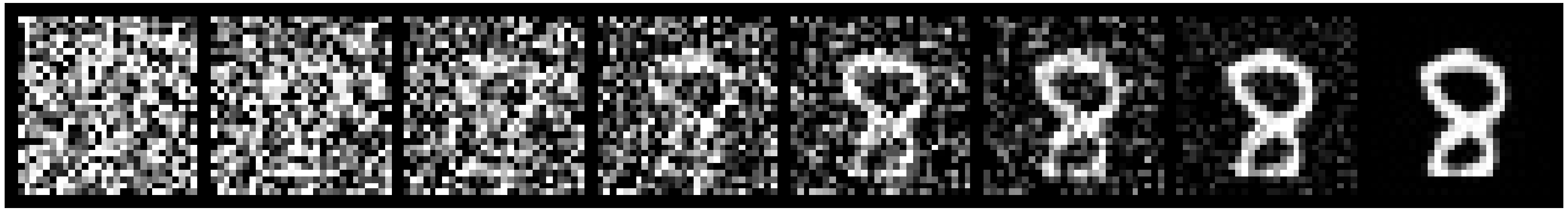
Inside the Neural Network



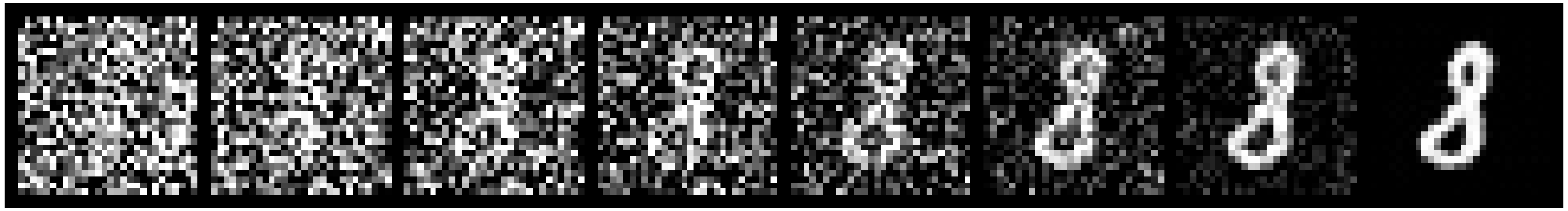
Neural Network

We learn how to remove noise!

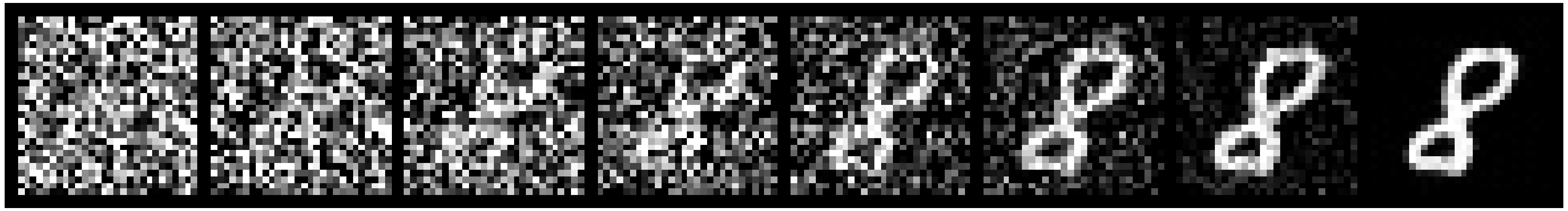
Reverse denoising (class 8)



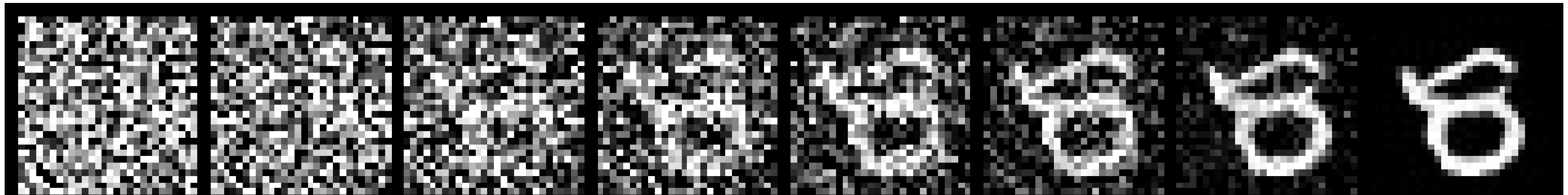
Reverse denoising (class 8)



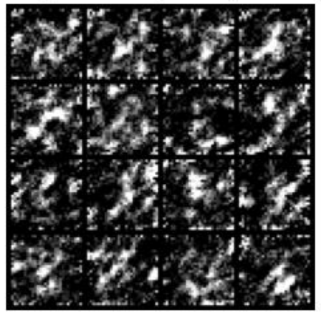
Reverse denoising (class 8)

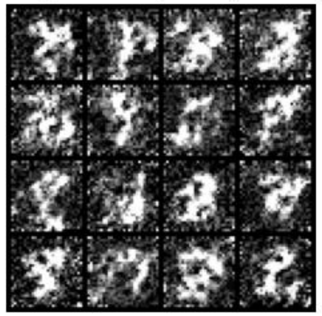


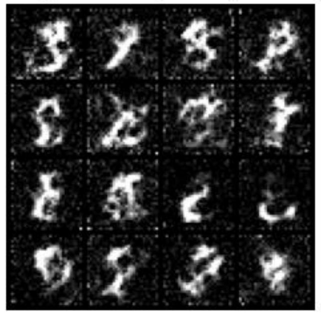
Reverse denoising (class 8)

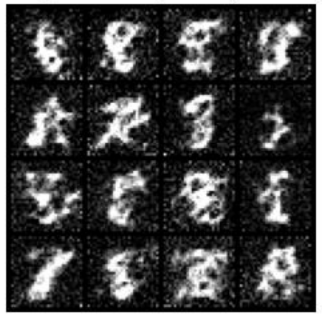


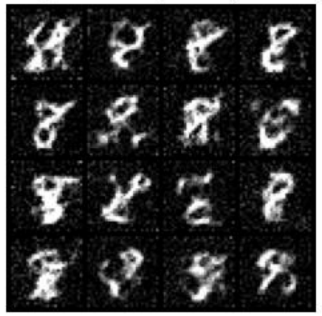
Training a Neural Network

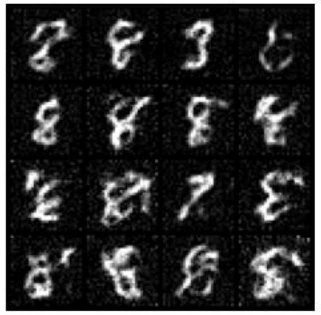
samples at step 100 (class 8)


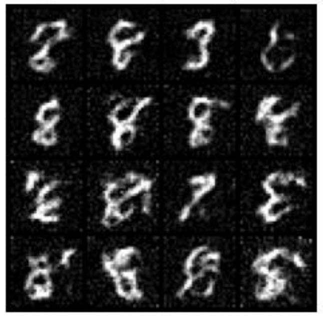
samples at step 200 (class 8)


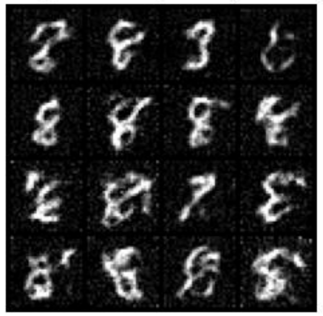
samples at step 300 (class 8)


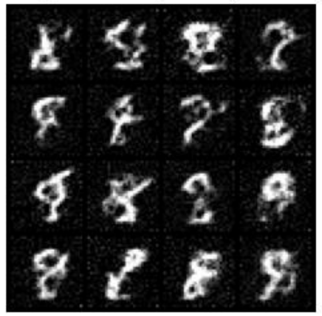
samples at step 400 (class 8)


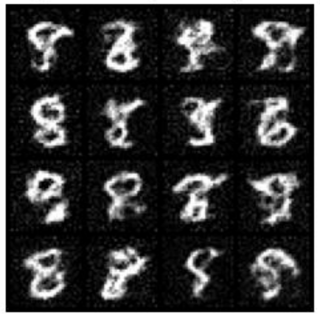
samples at step 500 (class 8)


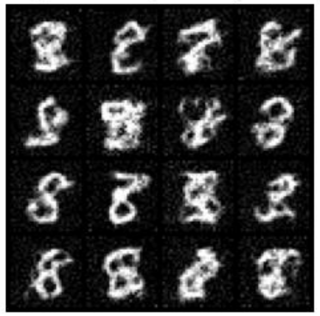
samples at step 600 (class 8)


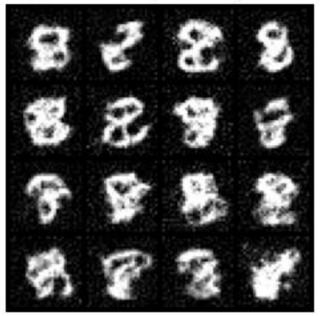
samples at step 600 (class 8)


samples at step 600 (class 8)


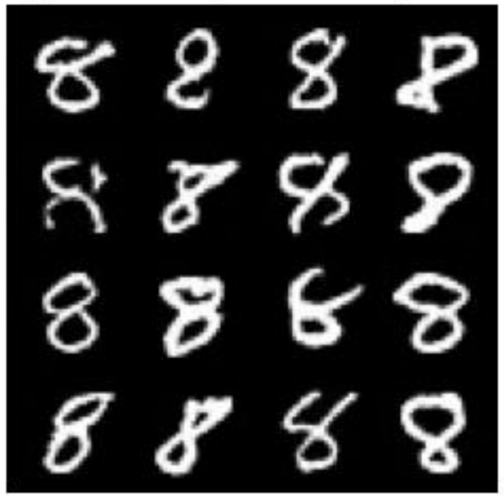
samples at step 900 (class 8)


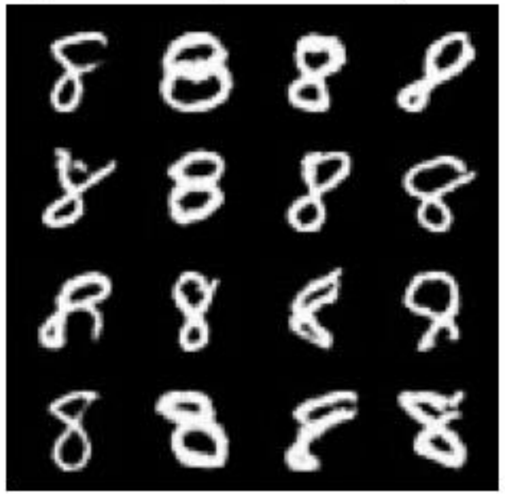
samples at step 1000 (class 8)


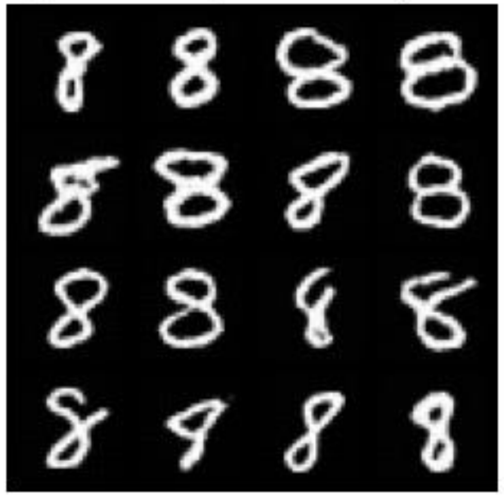
samples at step 1100 (class 8)


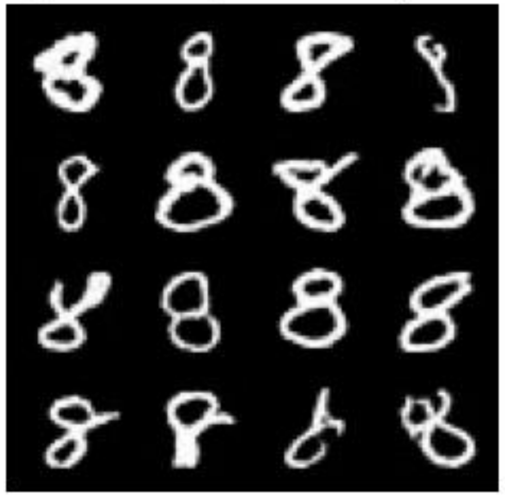
samples at step 1200 (class 8)


Trained for 20 mins. Works fine!


samples at step 160000 (class 8)


samples at step 222000 (class 8)

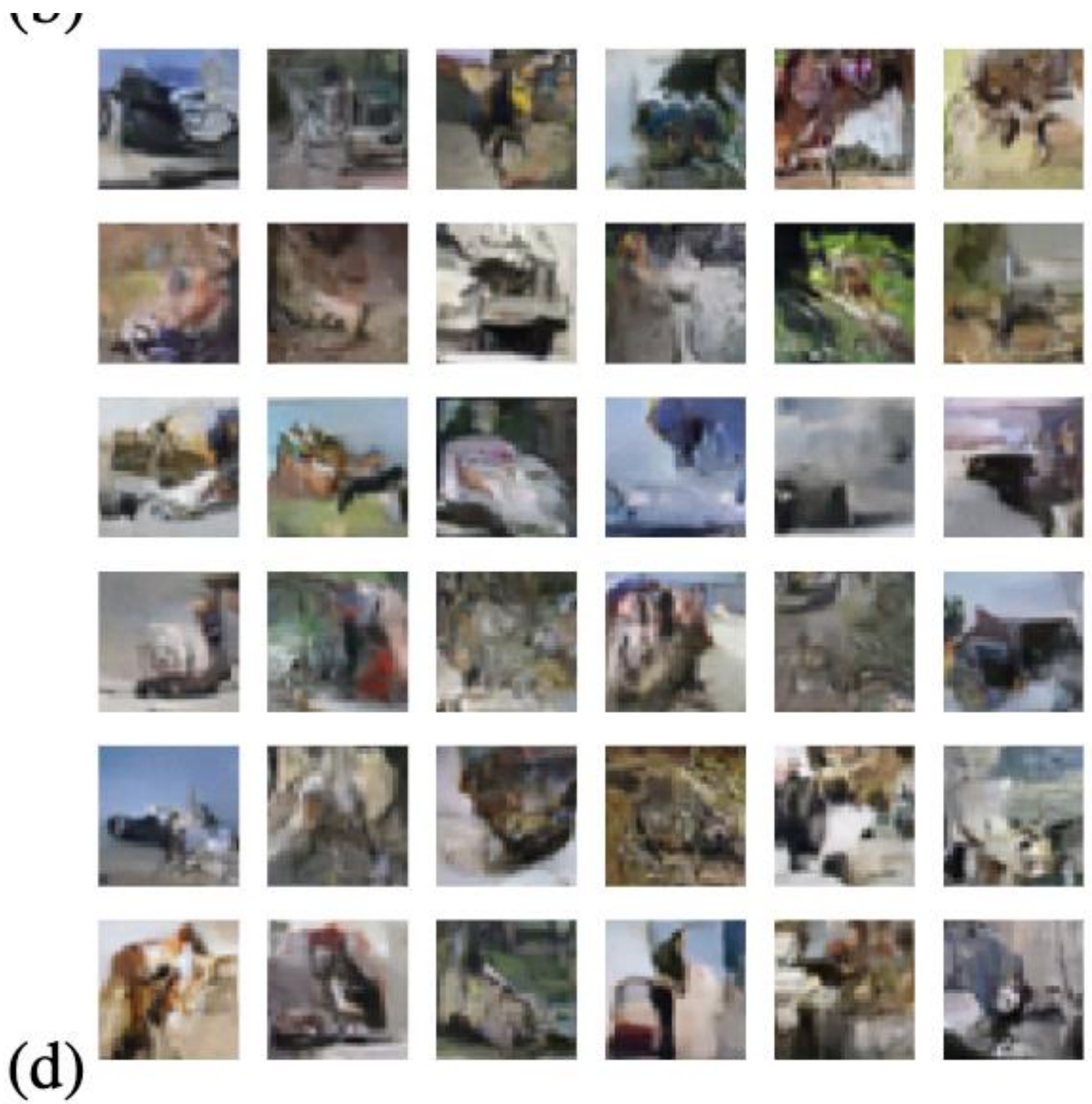

samples at step 236000 (class 8)


samples at step 228000 (class 8)

History at Stanford: Original Diffusion Paper

History at Stanford: Original Diffusion Paper

(a)

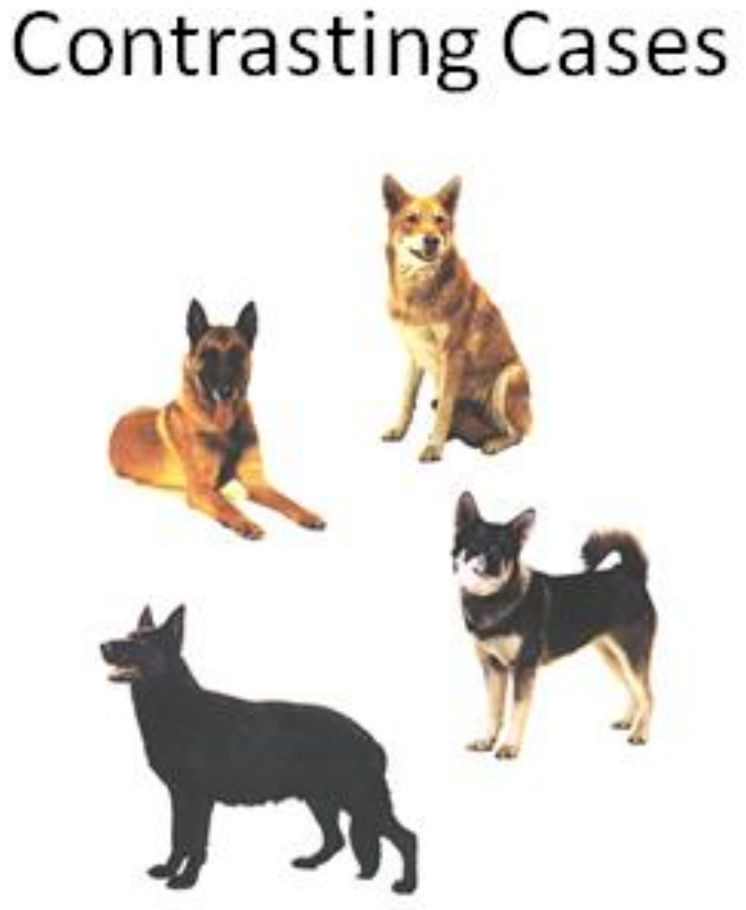

(d)

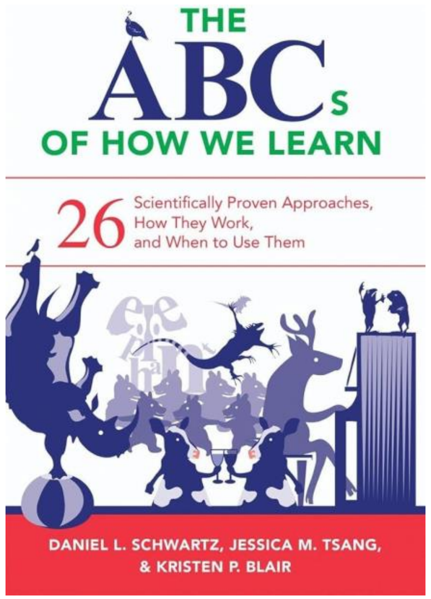
How do you use text to guide the generation of images?

What are people doing in Diffusion today?

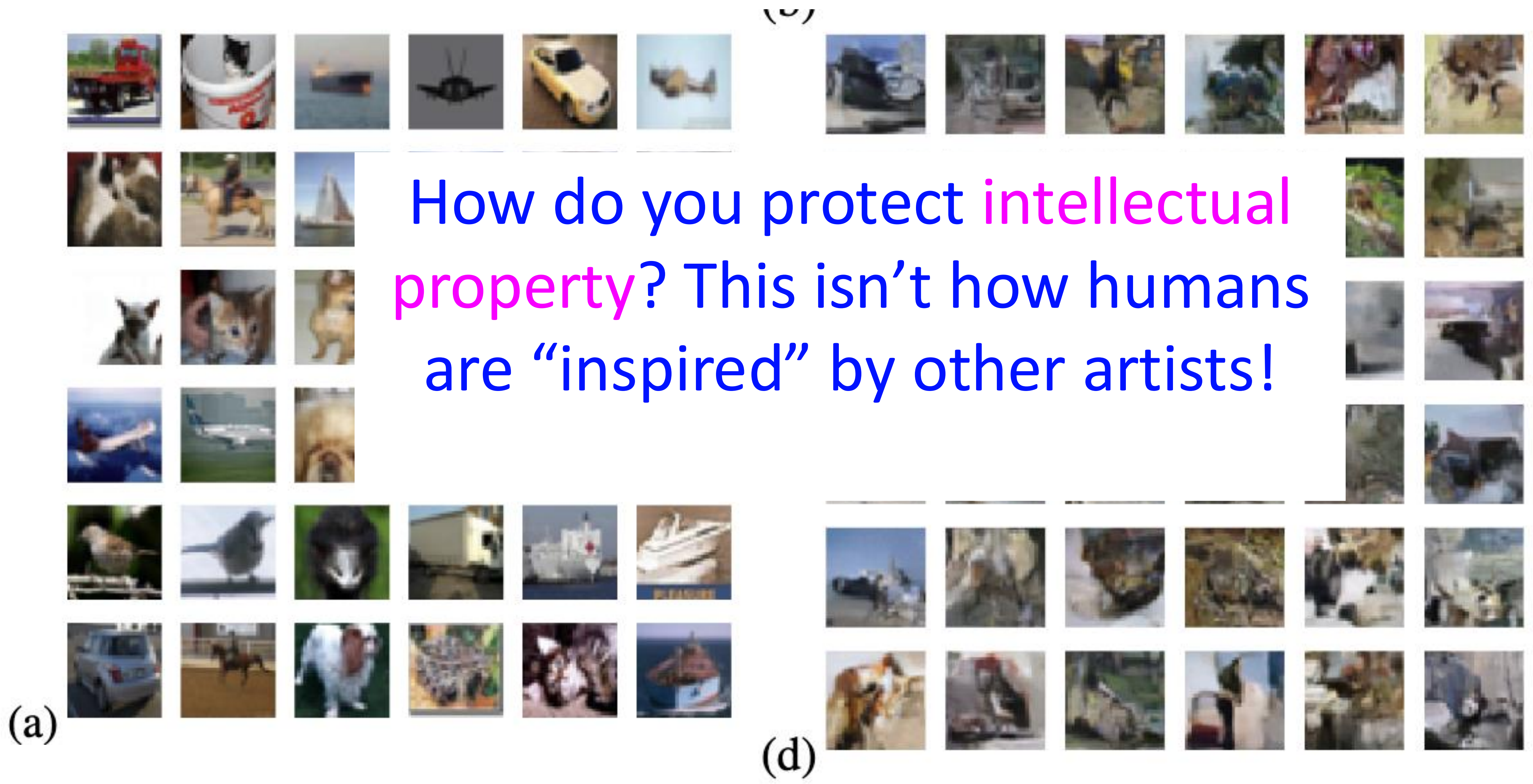
Can you use a classifier to guide generation of an image?

Can you Generate Contrasting Cases?



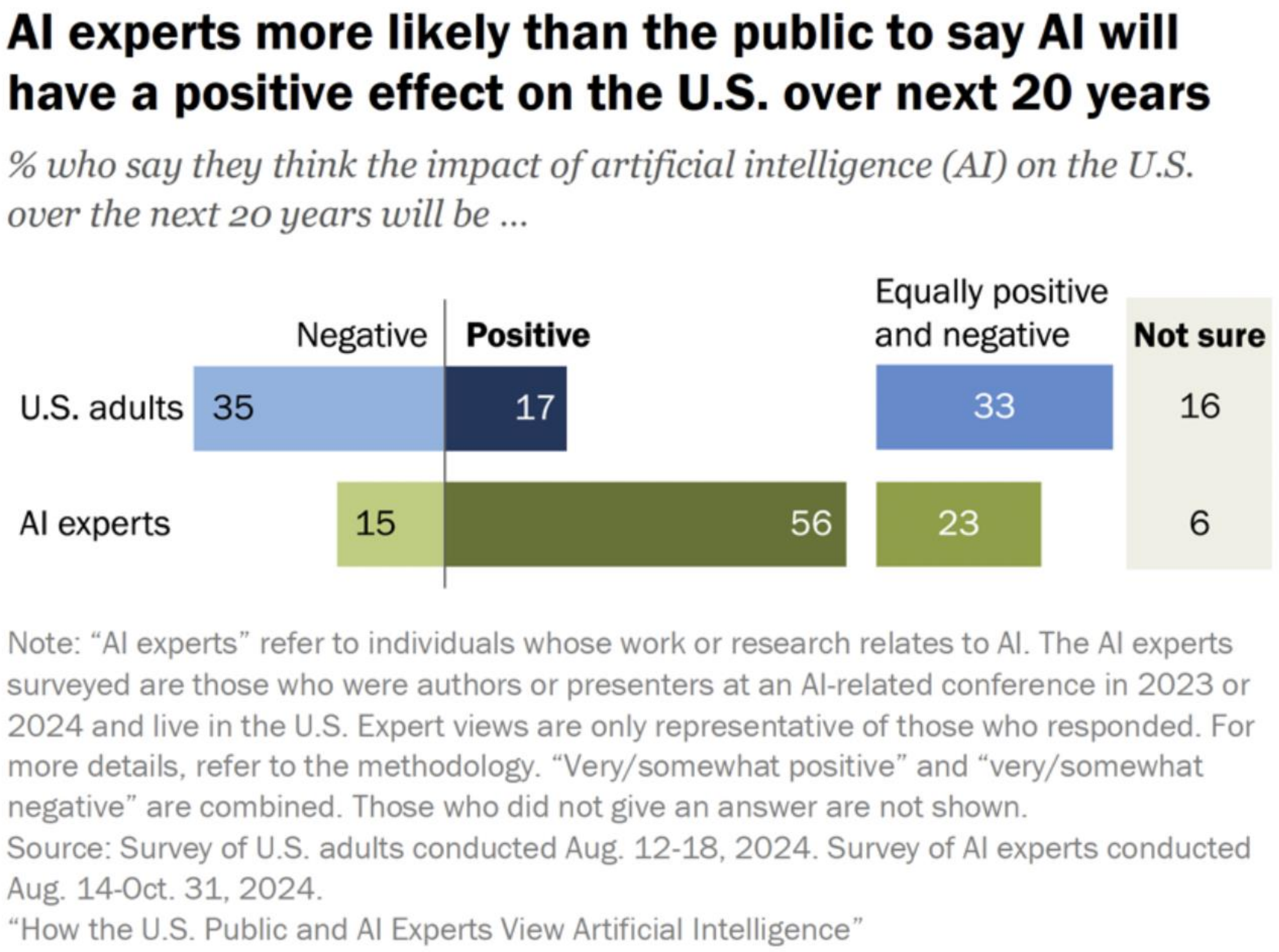

THE ABCs OF HOW WE LEARN
26 Scientifically Proven Approaches, How They Work, and When to Use Them
DANIEL L. SCHWARTZ, JESSICA M. TSANG, & KRISTEN P. BLAIR

History at Stanford: Original Diffusion Paper



How do you protect intellectual property? This isn't how humans are "inspired" by other artists!

Impact of AI is Really Wild!



AI experts more likely than the public to say AI will have a positive effect on the U.S. over next 20 years

% who say they think the impact of artificial intelligence (AI) on the U.S. over the next 20 years will be …

	Negative	**Positive**	Equally positive and negative	**Not sure**
U.S. adults	35	17	33	16
AI experts	15	56	23	6

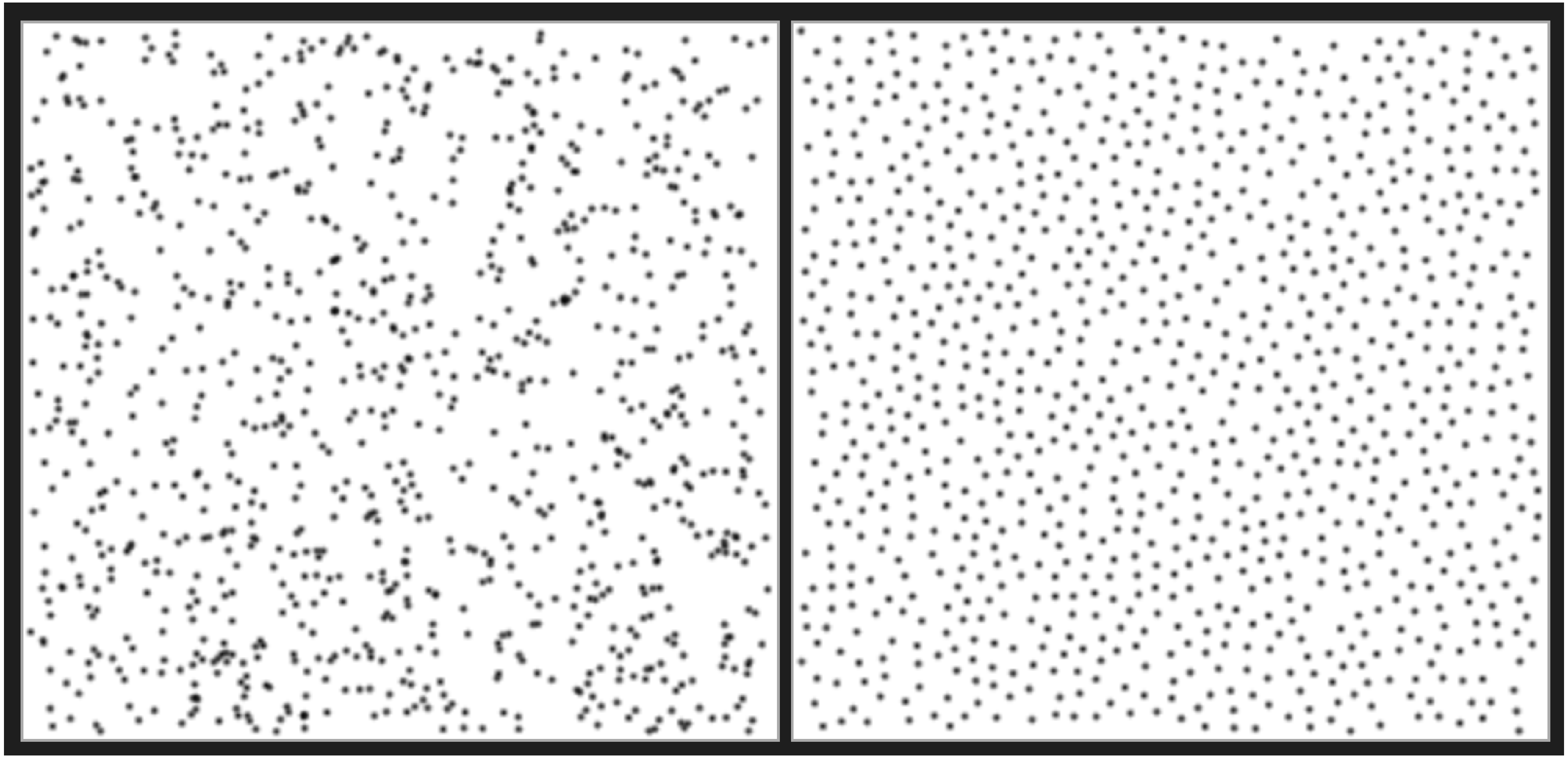
Note: "AI experts" refer to individuals whose work or research relates to AI. The AI experts surveyed are those who were authors or presenters at an AI-related conference in 2023 or 2024 and live in the U.S. Expert views are only representative of those who responded. For more details, refer to the methodology. "Very/somewhat positive" and "very/somewhat negative" are combined. Those who did not give an answer are not shown.
Source: Survey of U.S. adults conducted Aug. 12-18, 2024. Survey of AI experts conducted Aug. 14-Oct. 31, 2024.
"How the U.S. Public and AI Experts View Artificial Intelligence"

On Wed!

Second: Another practice problem

Which One Is Poisson?

Which one is Poisson? Justify with Math…

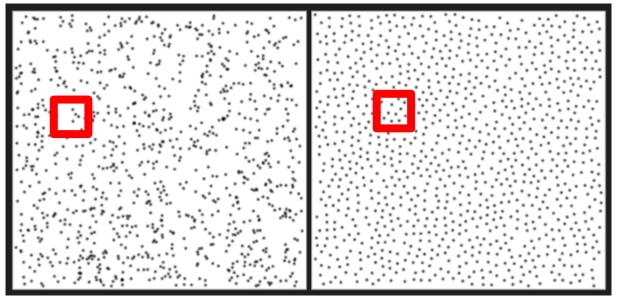


Same number of points…

Stanford University

Which one is Poisson? Sampling Solution

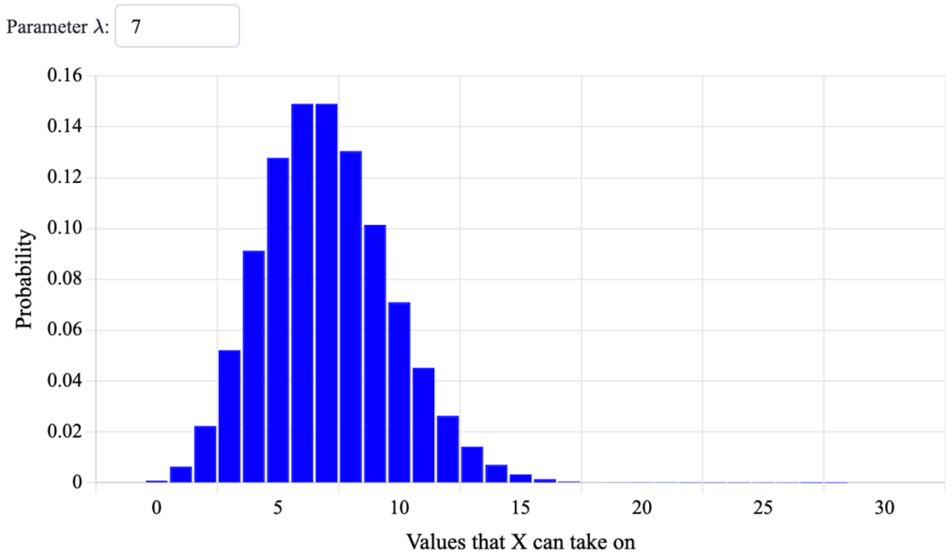
Squares that are 1x1

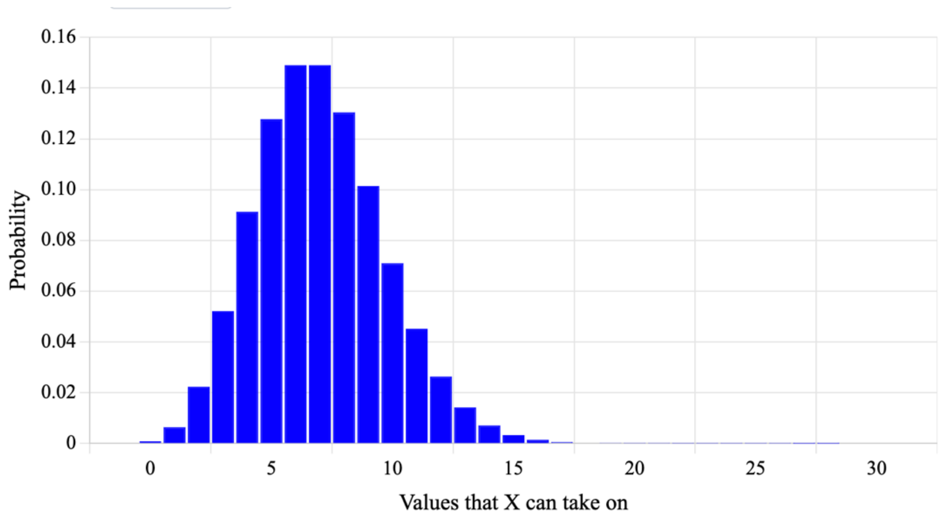


True rate = 7 points per unit size

Theory $\mathrm{P}(X = x) = \dfrac{\lambda^x e^{-\lambda}}{x!}$

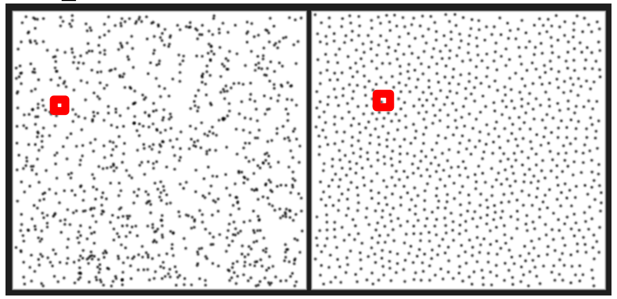
Observed (for both)





Which one is Poisson? Sampling Solution

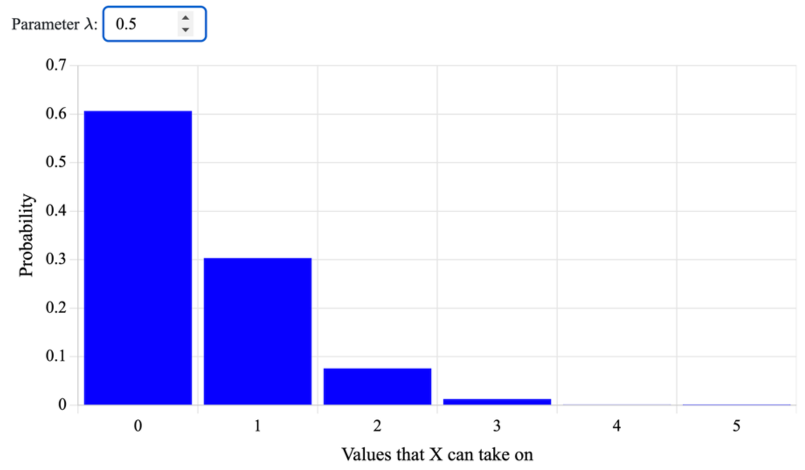
Squares that are .01 x .01

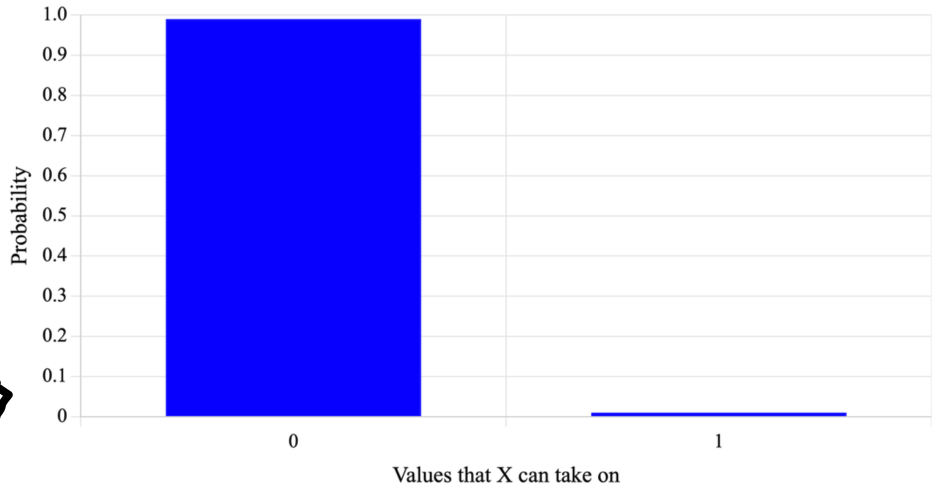


True rate = 7 points per unit size

Theory $\mathrm{P}(X = x) = \dfrac{\lambda^x e^{-\lambda}}{x!}$

Observed (for right)





Bonus, calculate the KL divergence between these two distributions

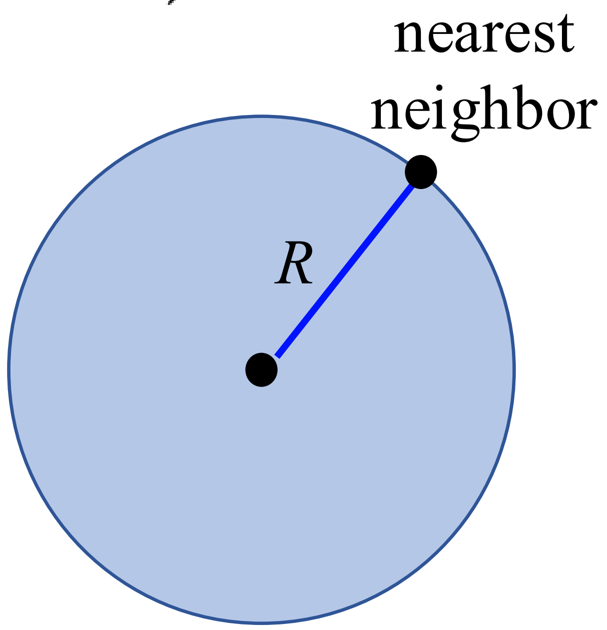
Which one is Poisson? Nearest Neighbor Distance

Let R be the distance to the nearest neighbor of a point

$$P(R < r) = 1 - P(\text{Zero points in radius } R)$$
$$= 1 - P(X = 0)$$

Let X be the number of points in radius R. $X \sim \text{Poi}(\lambda = 7 \cdot \pi \cdot r^2)$

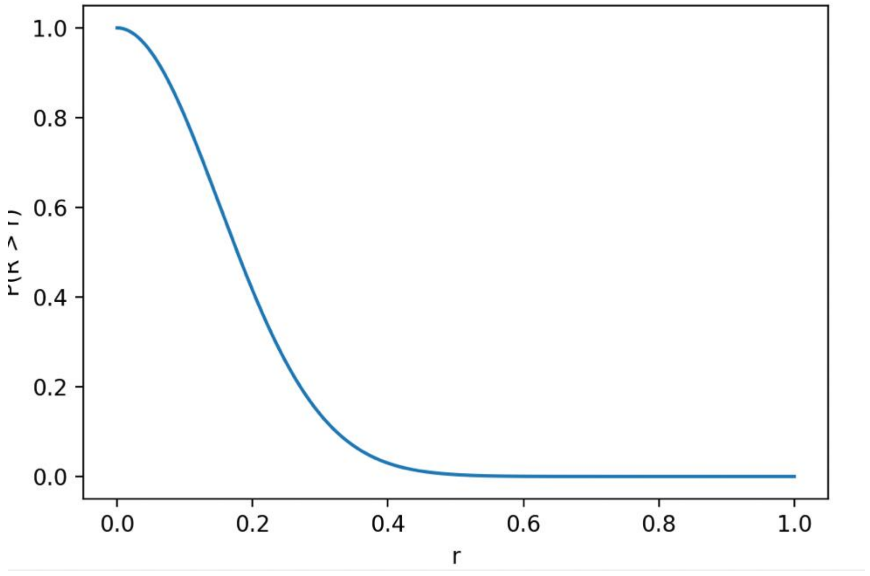
$$= 1 - \frac{(7\pi r^2)^0 e^{-7\pi r^2}}{0!}$$
$$= 1 - e^{7\pi r^2}$$



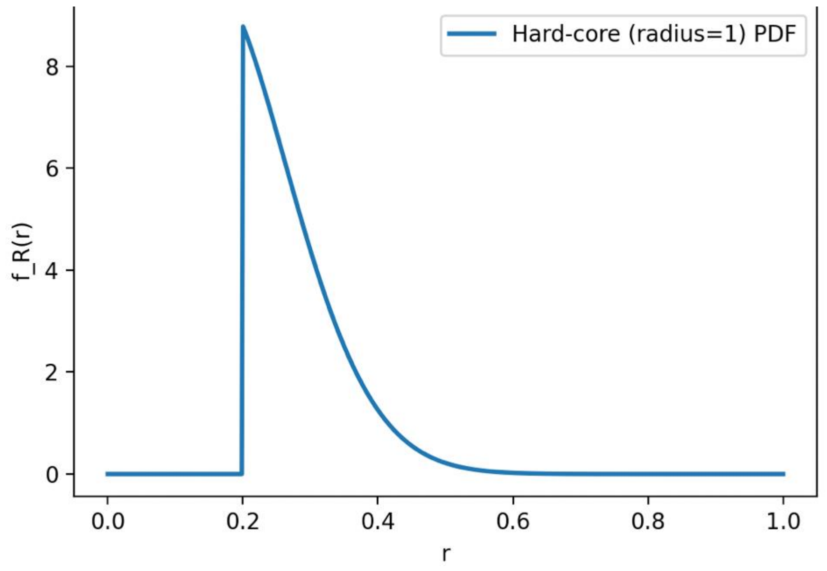
Which one is Poisson? Nearest Neighbor Distance

$$P(R > r) = 1 - e^{7\pi r^2}$$

Theory and picture on the left:



Picture on the right:



On Wed!