Title: PSI 2018/2019 - Machine Learning - Lecture 14

Speakers: Michael Albergo

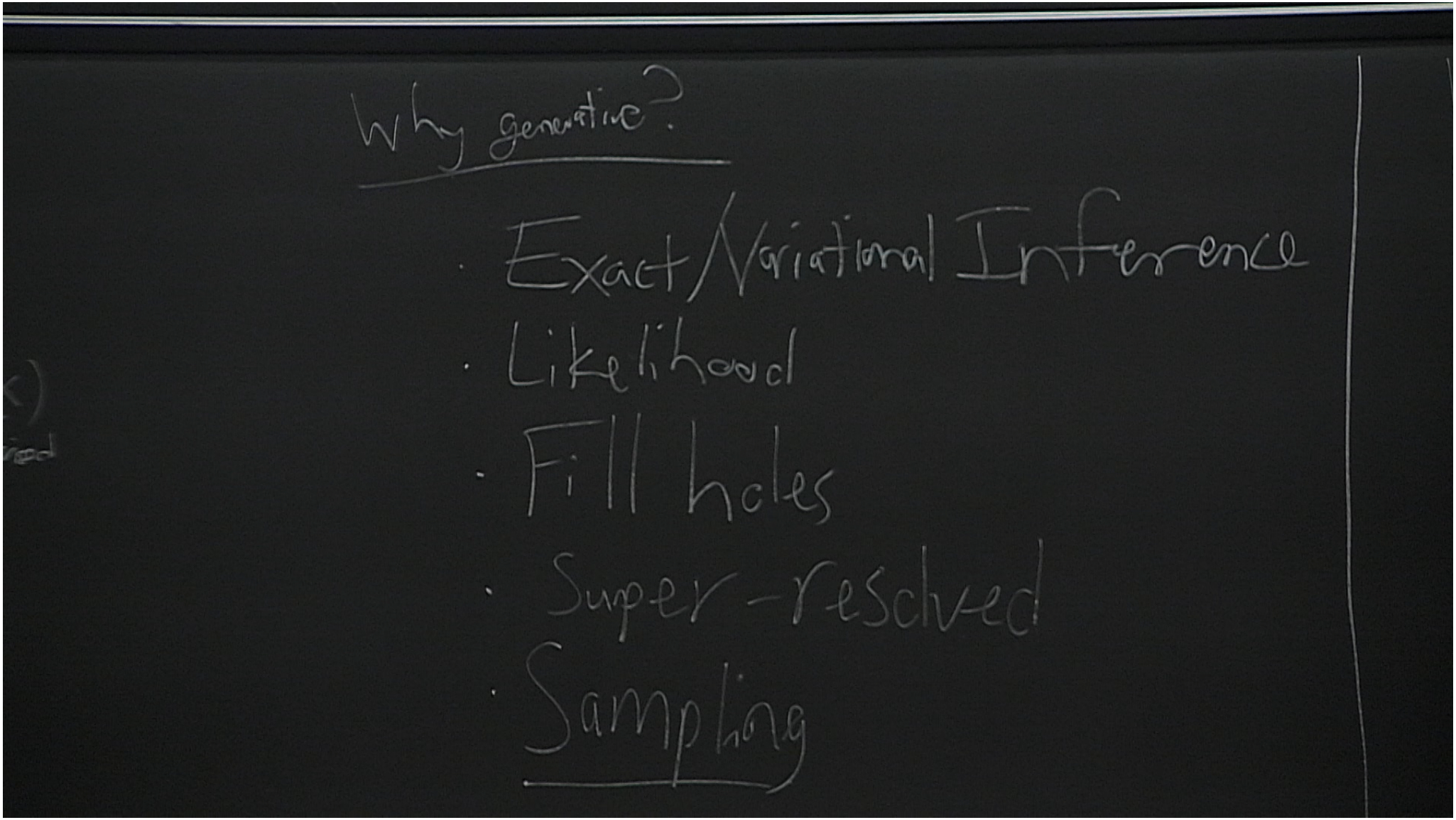
Collection: PSI 2018/2019 - Machine Learning (Hayward Sierens)

Date: April 11, 2019 - 9:00 AM

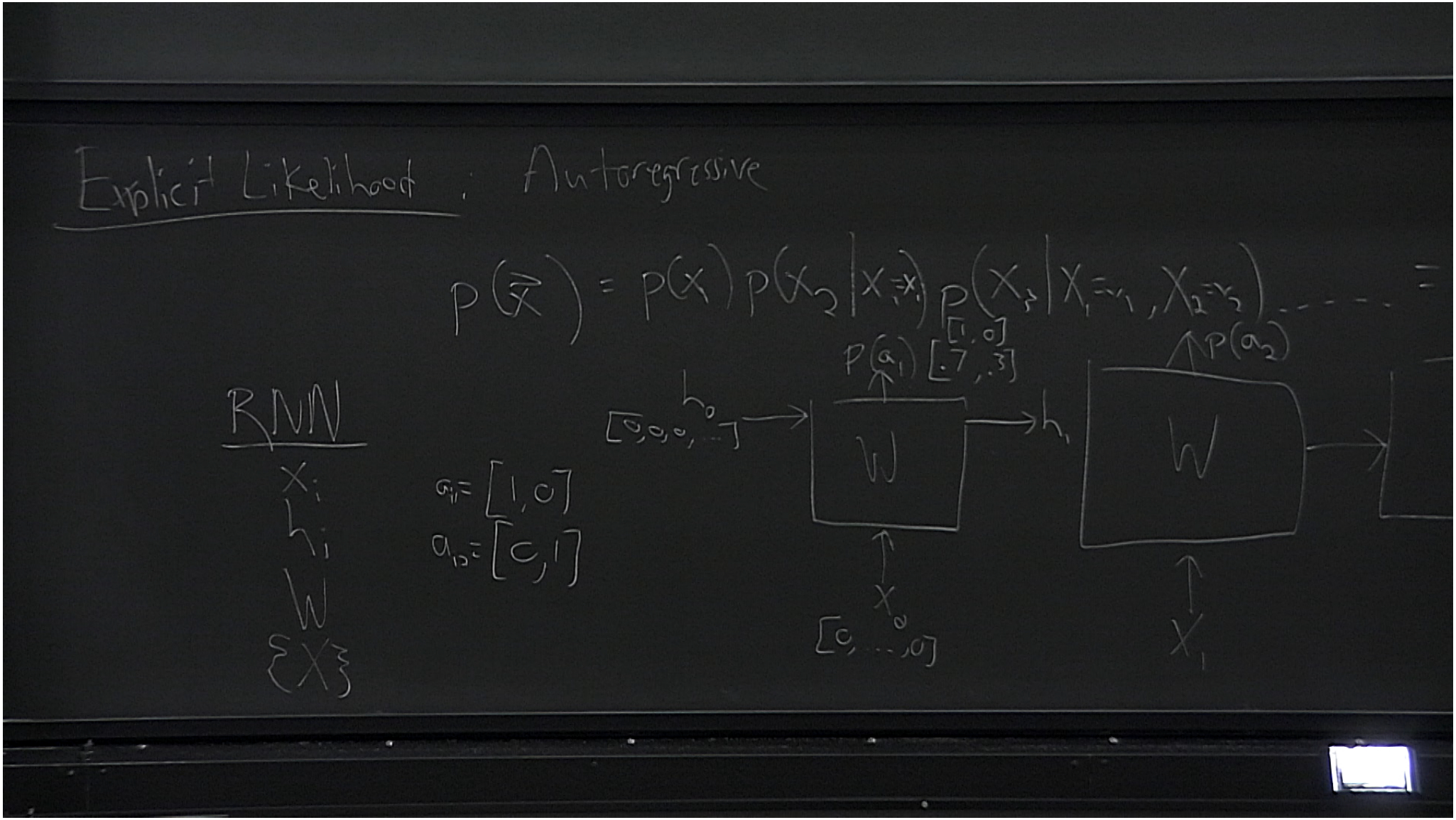
URL: http://pirsa.org/19040009

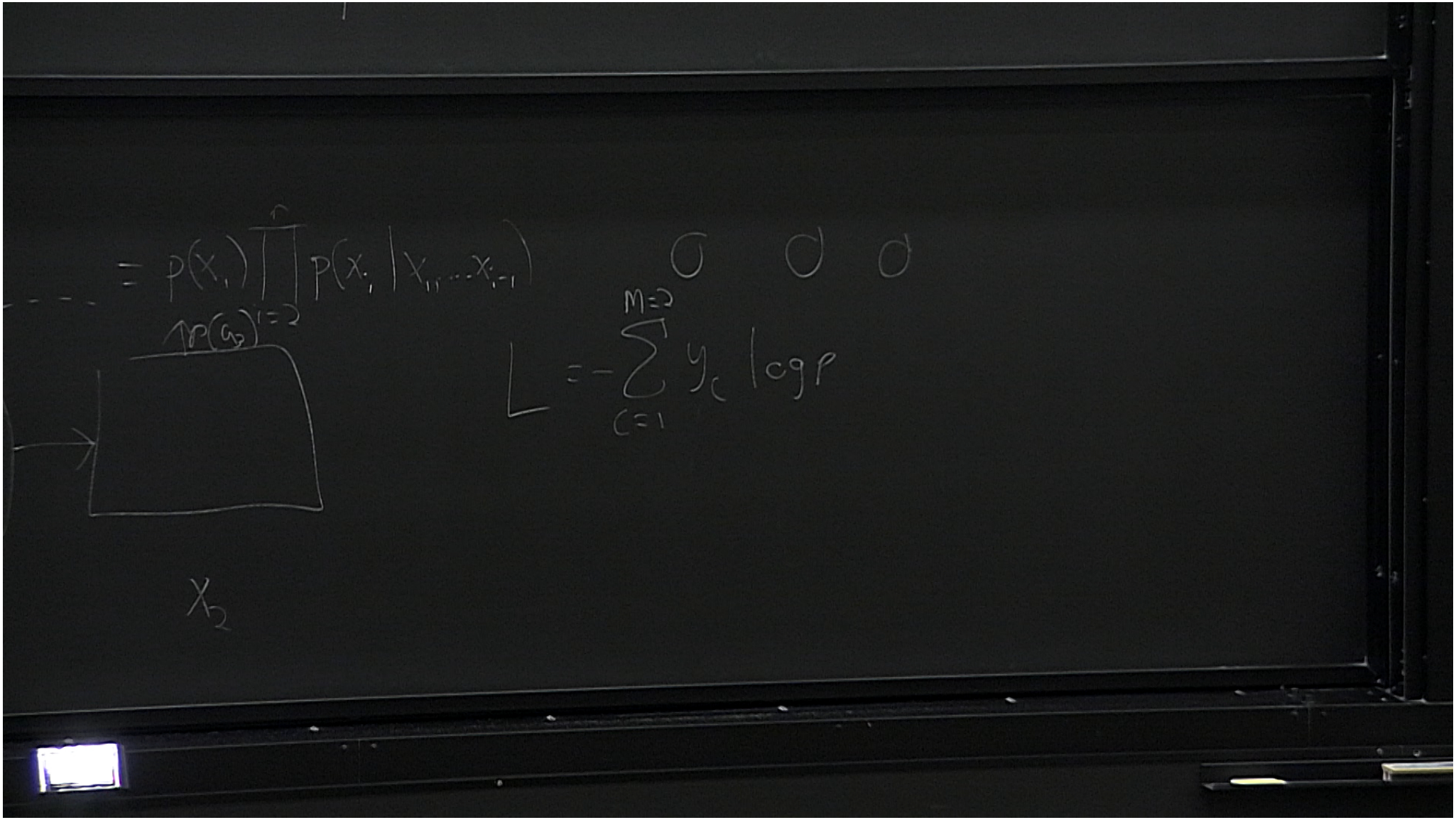
Generative Modeling

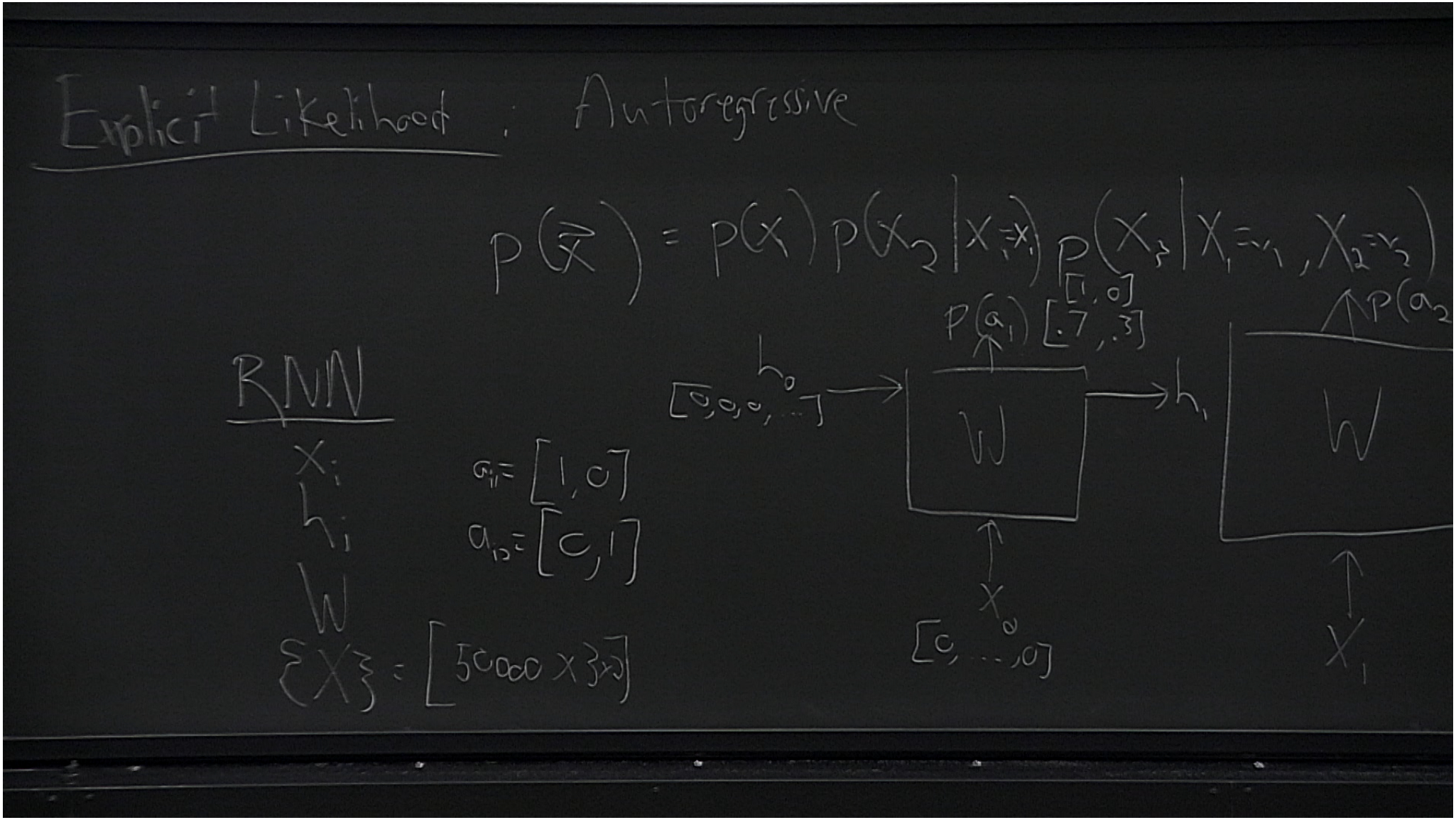
- Discriminative: Learn $p(y|x)$
- Generative: Learn $p(x,y)$, $p(x)$ unsupervised

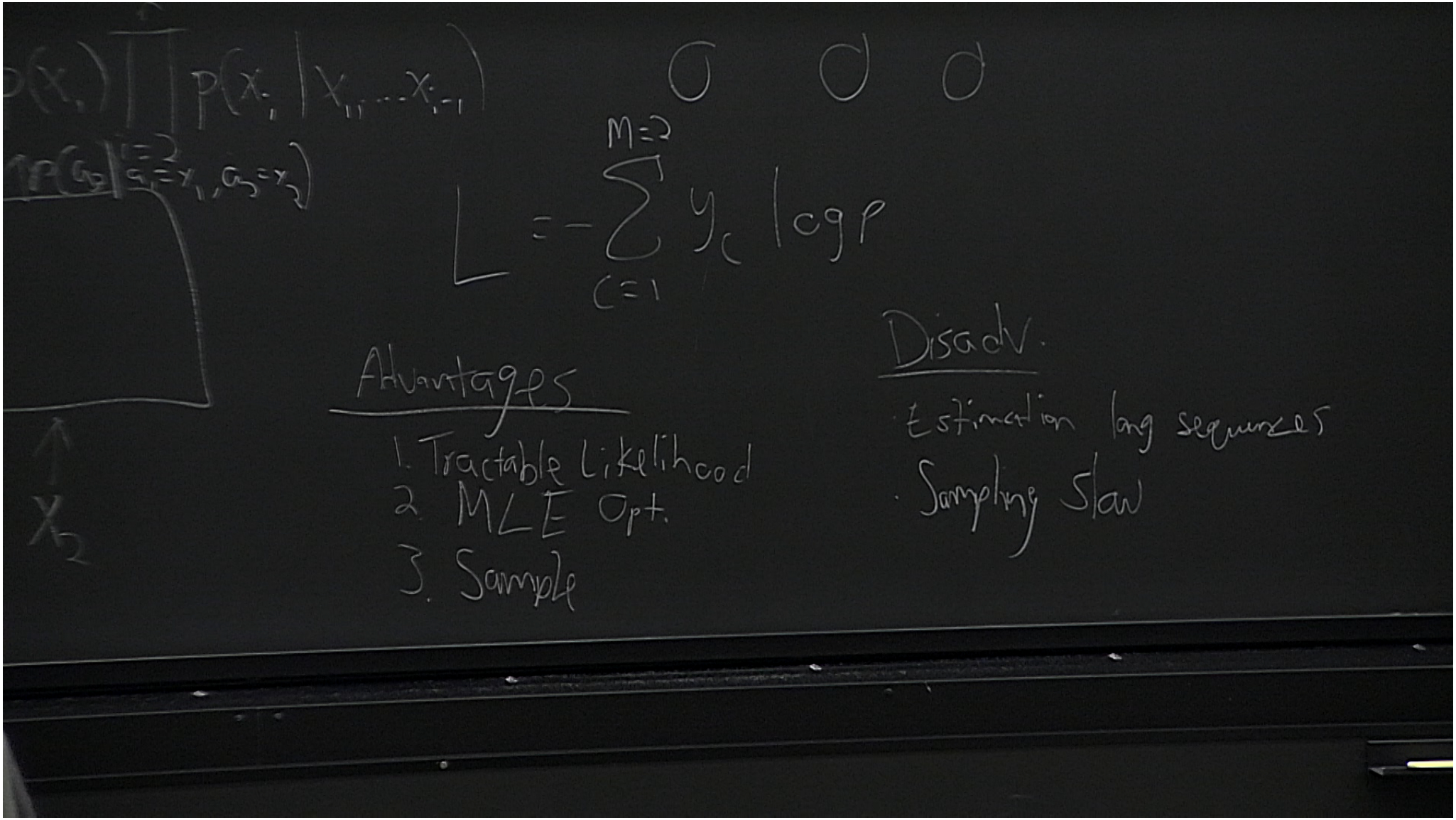

Why generative?
· Exact/Variational Inference
· Likelihood
· Fill holes
· Super-resolved
· Sampling

1. Explicit and Tractable Likelihood $p(x)$
   - Autoregressive, normalizing flows
2. Explicat Approximate Likelihood (bound)
   - VAE
3. Implicit Likelihood
   - GAN (Adversarial Learning)


Explicit Likelihood : Autoregressive
$p(\vec{x}) = p(x_1)\, p(x_2 \mid x_1 = x_1)\, p(x_3 \mid x_1 = x_1, x_2 = x_2) \ldots =$
RNN
x_i
h_i
W
$\{x\}$
$a_1 = [1, 0]$
$a_{12} = [0, 1]$
h_0
$[0, 0, \ldots]$
$p(a_1)$
$[1, 0]$
$[.7, .3]$
$p(a_2)$
W
W
h_1
x_0
$[0, \ldots, 0]$
x_1


$= p(x_1) \prod_{i=2} p(x_i | x_1, \ldots x_{i-1})$
$p(G_2)^{i=2}$
$L = -\sum_{c=1}^{M=2} y_c \log p$
x_2


Explicit Likelihood : Autoregressive
$p(\vec{x}) = p(x_1)\, p(x_2 \mid x_1 = x_1)\, p(x_3 \mid x_1 = x_1, x_2 = x_2)$
RNN
x_i
h_i
W
$\{x\} = [50000 \times 3 \times 3]$
$a_{11} = [1, 0]$
$a_{12} = [0, 1]$
h_0
$[0, 0, \ldots]$
W
h_1
x_0
$[0, \ldots, 0]$
$p(a_1)$ $[1, 0]$
$[.7, .3]$
$p(a_2$
W
x_1


$p(x_1)\prod p(x_i \mid x_1, \ldots x_{i-1})$
$p(G_2 \mid G_1 = x_1, G_3 = x_2)$
$L = -\sum_{c=1}^{M=2} y_c \log p$
Advantages
1. Tractable Likelihood
2. MLE Opt.
3. Sample
Disadv.
· Estimation long sequences
· Sampling slow
X_2

Normalizing Flows

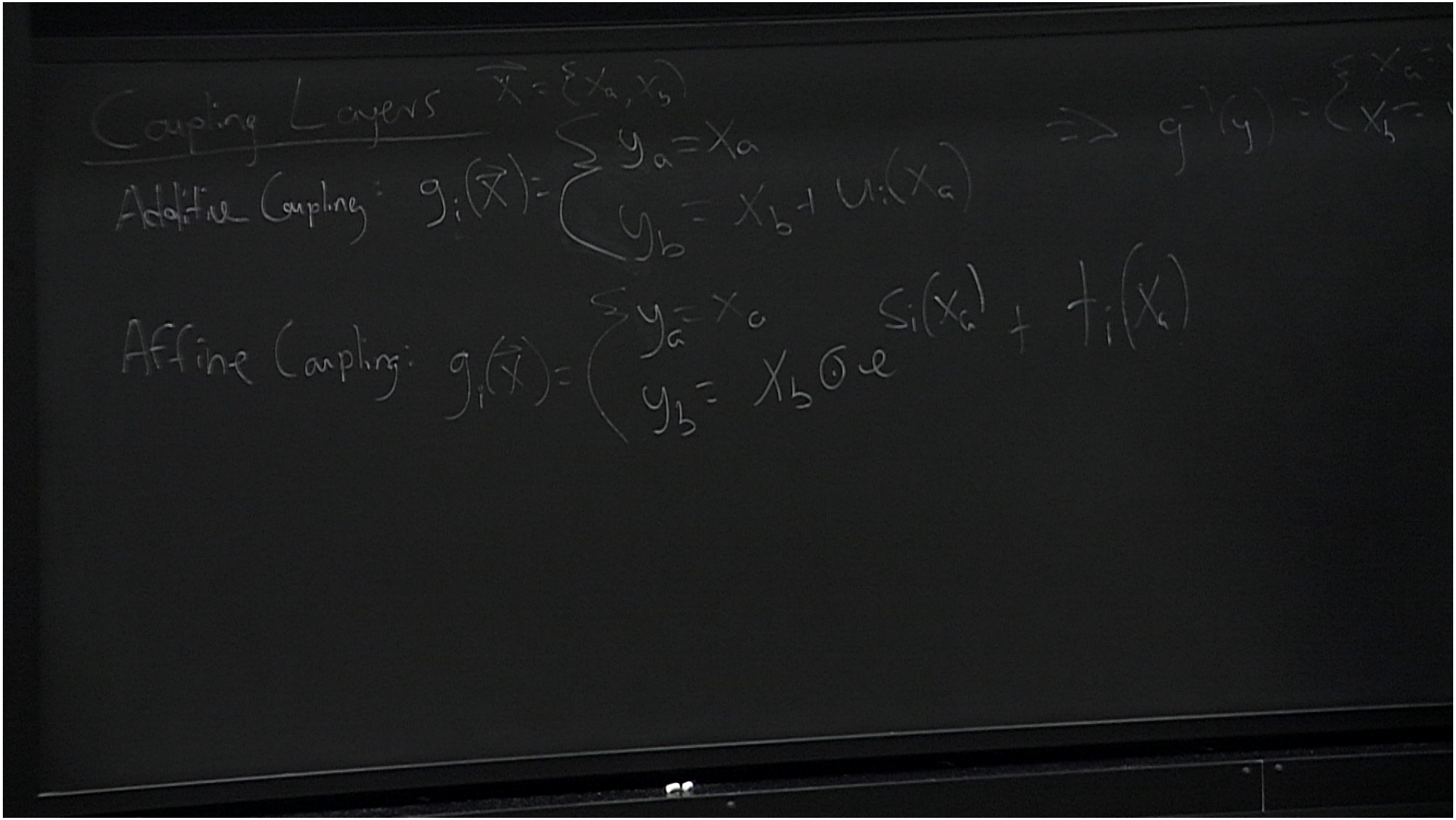
$x \sim p(\vec{x})$ intractable

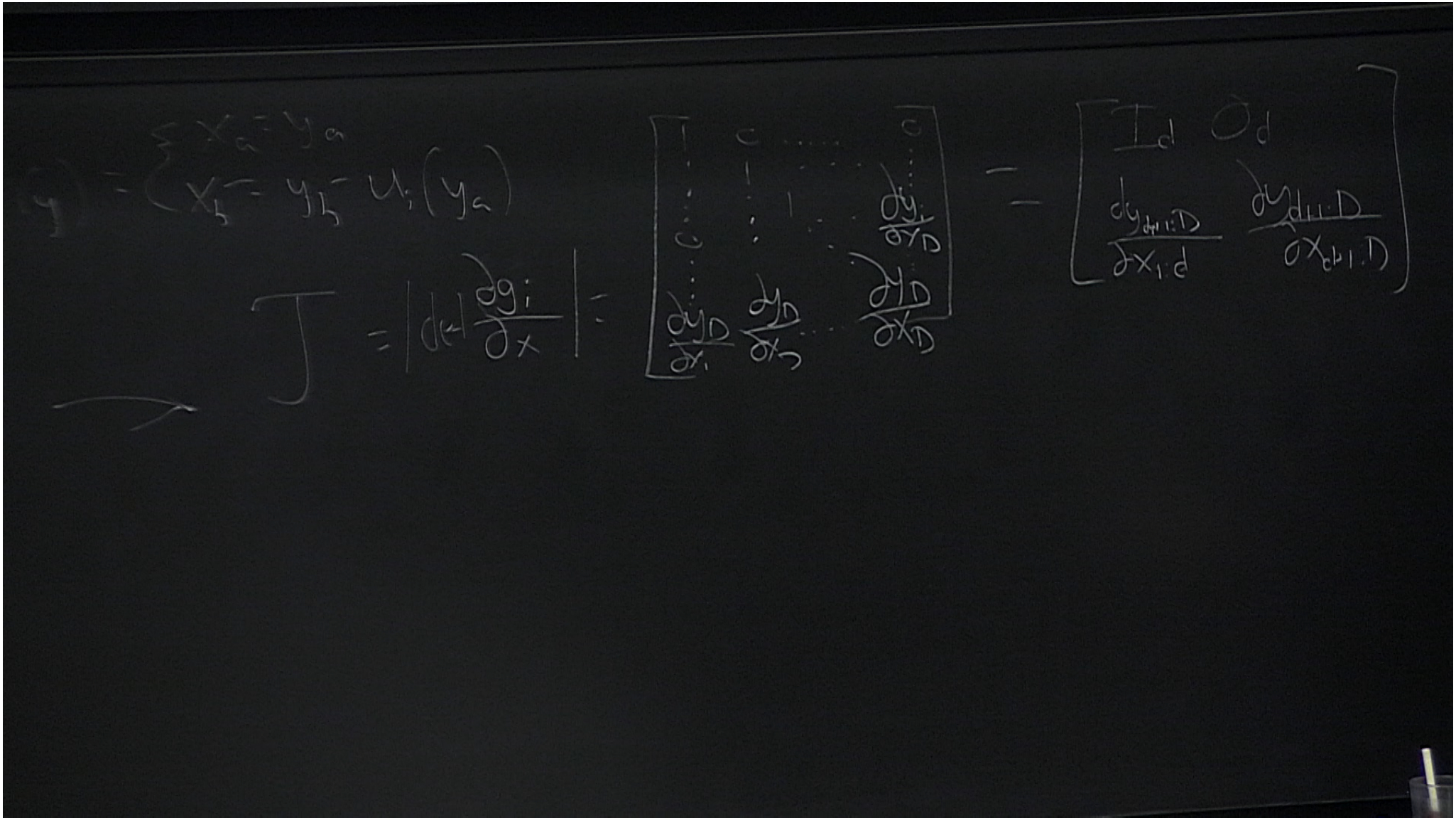
$z \sim N(0, I)$ $p_z(\vec{z})$

$f(x) = z$

$p(x) = p_z(f(x)) \left| \det \frac{\partial f(x)}{\partial x^T} \right|$

1. f invertible
2. Jacobian computable


Coupling Layers $\vec{x} = (x_a, x_b)$
Additive Coupling: $g_i(\vec{x}) = \begin{cases} y_a = x_a \\ y_b = x_b + u_i(x_a) \end{cases}$ $\Rightarrow g^{-1}(y) = \begin{cases} x_a = \\ x_b = \end{cases}$
Affine Coupling: $g_i(\vec{x}) = \begin{cases} y_a = x_a \\ y_b = x_b \odot e^{s_i(x_a)} + t_i(x_a) \end{cases}$



Normalizing Flows

$x \sim p(x)$ intractable

$z \sim N(0, I)$ $p_z(z)$

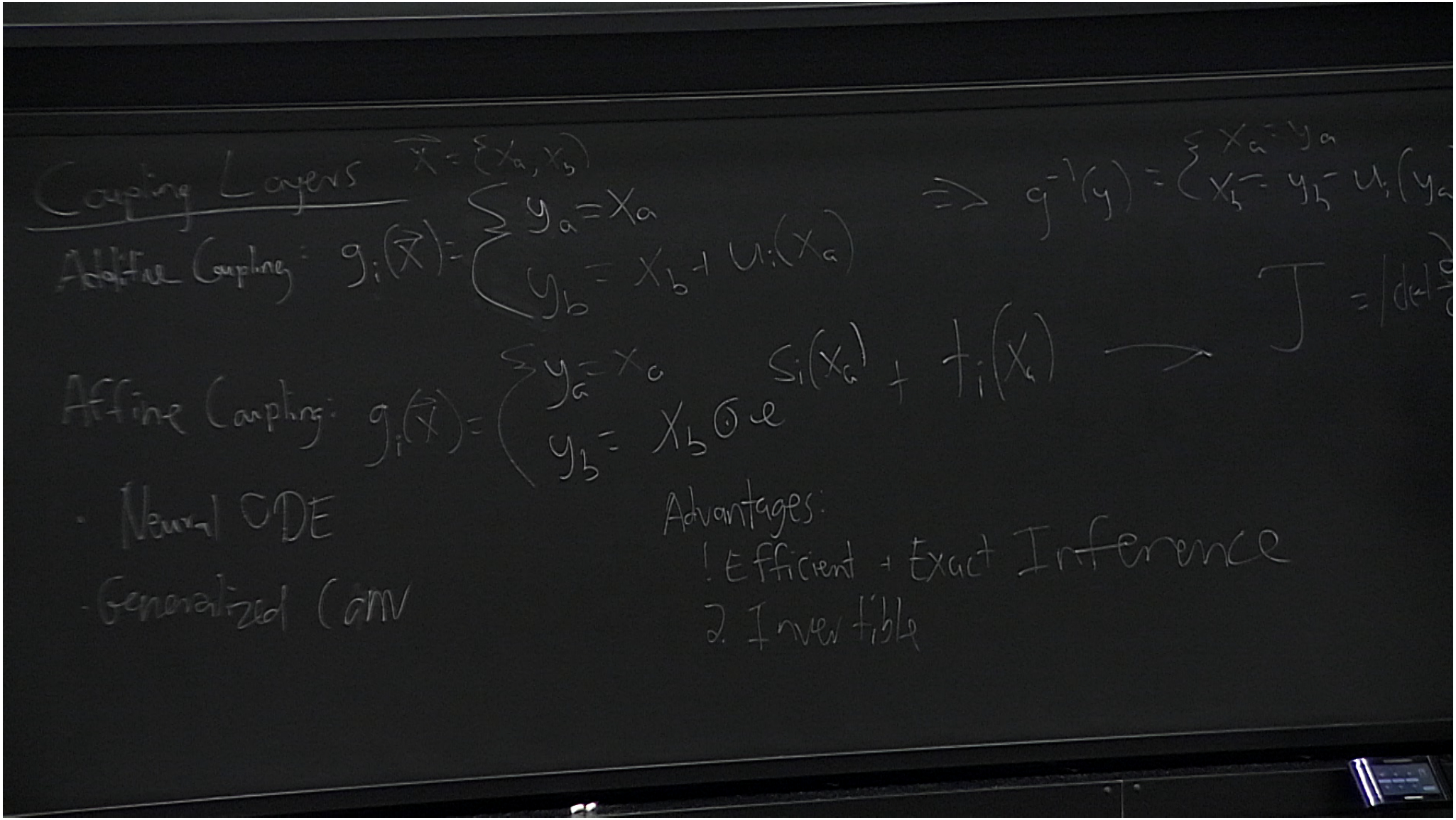
$f(x) = z$

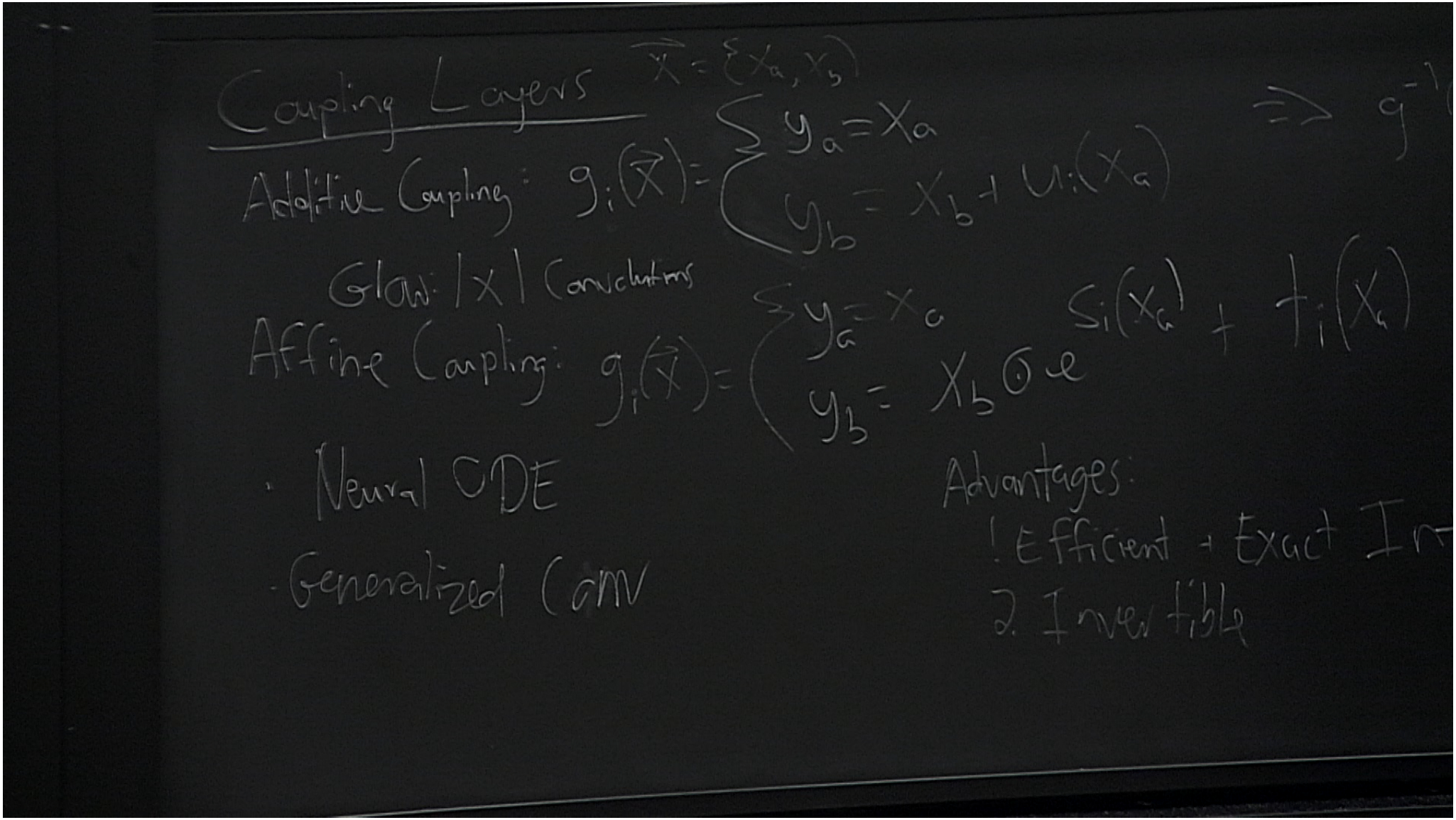
$p(x) = p_z(f(x)) \left| \det \frac{\partial f(x)}{\partial x^T} \right|$

$\log p(x) = \log p_z(f(x)) + \log \left(\left| \det \frac{\partial f}{\partial x^T} \right| \right)$

1. f invertible
2. Jacobian computable

Learn bijective


Coupling Layers
Additive Coupling:
Affine Coupling:
Neural ODE
Generalized Conv
Advantages:
1. Efficient + Exact Inference
2. Invertible


Coupling Layers
$\vec{x} = (x_a, x_b)$
Additive Coupling: $g_i(\vec{x}) = \begin{cases} y_a = x_a \\ y_b = x_b + u_i(x_a) \end{cases} \Rightarrow g^{-1}$
Glow: 1x1 Convolutions
Affine Coupling: $g_i(\vec{x}) = \begin{cases} y_a = x_a \\ y_b = x_b \odot e^{s_i(x_a)} + t_i(x_a) \end{cases}$
· Neural ODE
· Generalized Conv
Advantages:
1. Efficient + Exact In
2. Invertible

```
%matplotlib inline
import torch
import numpy as np
from torch import nn, optim
from torch.nn import functional as F
import numpy as np
import os
import datetime
import matplotlib
from matplotlib import pyplot as plt
from matplotlib.colors import ListedColormap
#import palettable
import seaborn as sns
```

```
input_size = 2 # dimension
n_layers = 10 # how many coupling layers
```

Beni Yoshida

Below is how we generate some pretty 2D distributions to model.

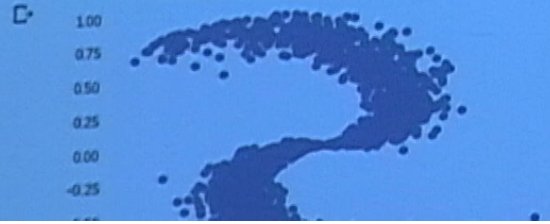
```
def real_sample(N = 100, K = 3):
    N_train = N*K # total number of points in the training set
    x_train = np.zeros((N_train,2)) # matrix containing the 2-dimensional datapoints
    y_train = np.zeros(N_train, dtype='uint8') # labels (not in one-hot representation)

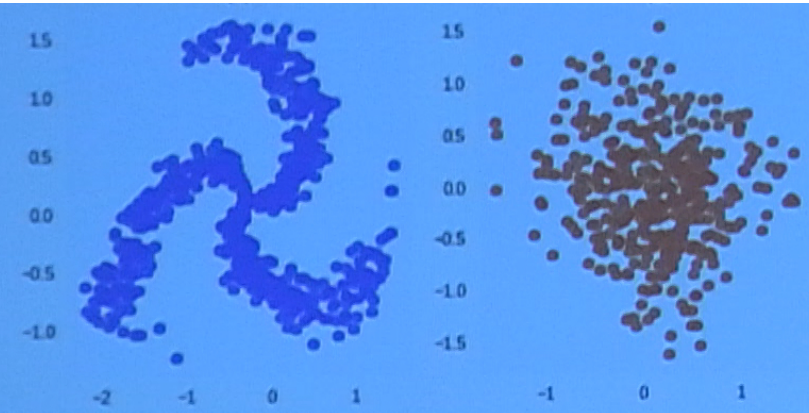
    mag_noise = 0.2 # controls how much noise gets added to the data
    dTheta    = 2   # difference in theta in each branch

    ### Data generation: ###
    for j in range(K):
        ix = range(N*j,N*(j+1))
        r = np.linspace(0.01,1,N) # radius
        t = np.linspace(j*(2*np.pi)/K,j*(2*np.pi)/K + dTheta,N) + np.random.randn(N)*mag_noise # theta
        x_train[ix] = np.c_[r*np.cos(t), r*np.sin(t)]

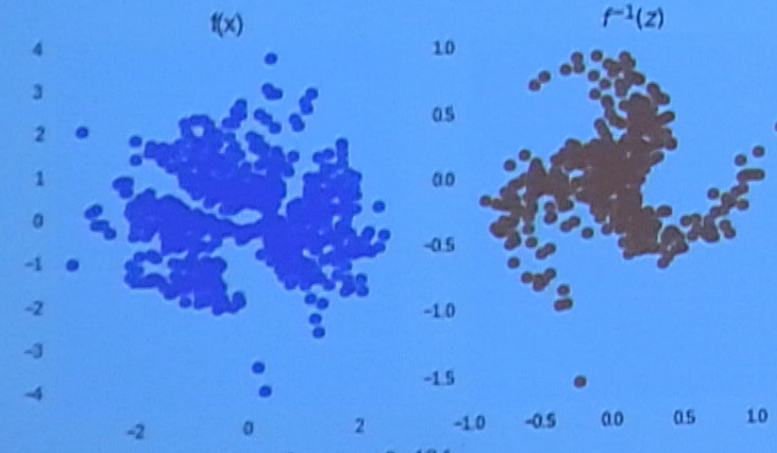
    return torch.from_numpy(x_train).double()

test = real_sample(N=1000)
plt.scatter(test[:, 0], test[:, 1], s=40)
plt.show()
```

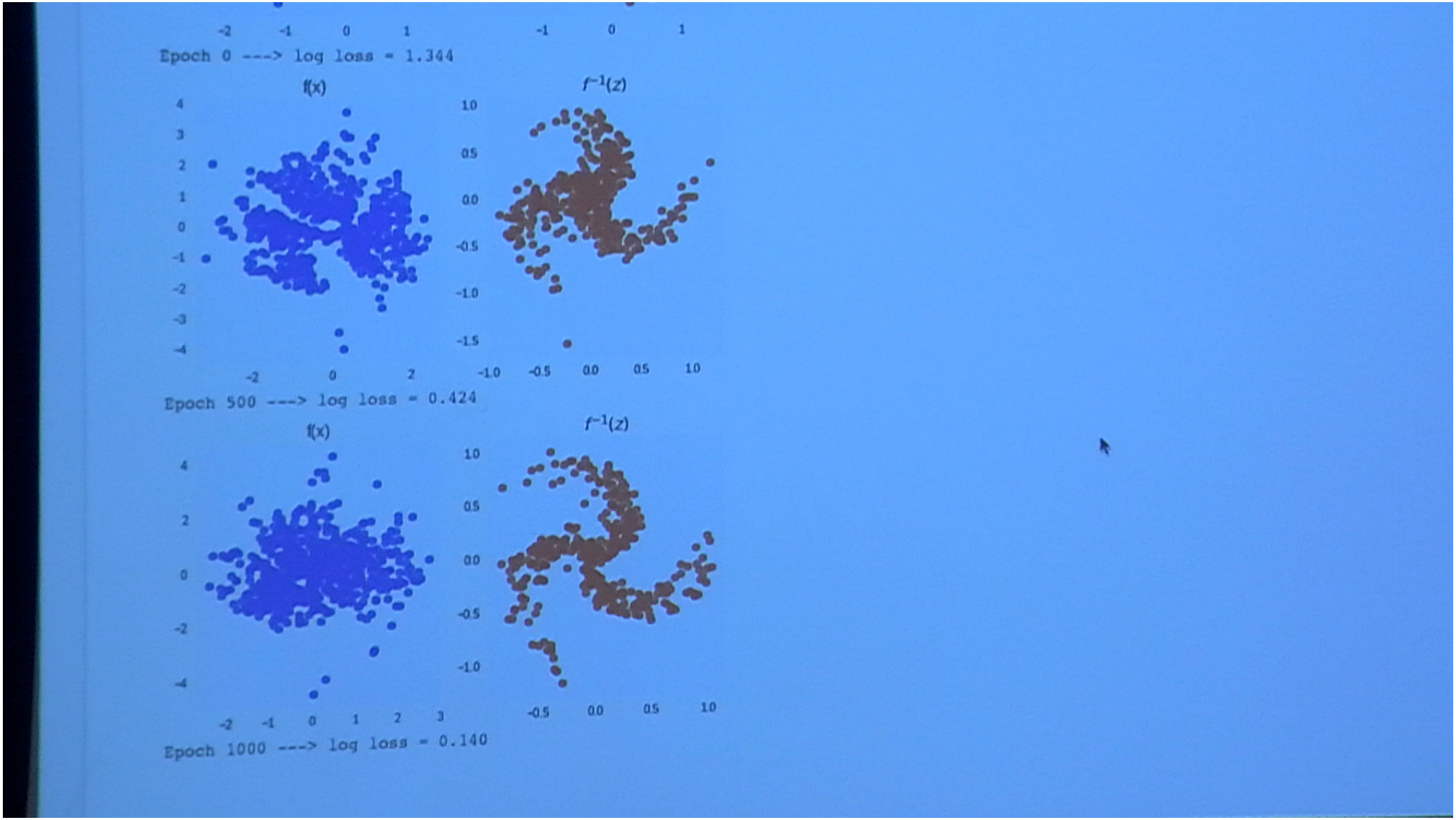




Epoch 0 ---> log loss = 1.344



Epoch 500 ---> log loss = 0.424


Epoch 0 ---> log loss = 1.344
f(x)
$f^{-1}(z)$
Epoch 500 ---> log loss = 0.424
f(x)
$f^{-1}(z)$
Epoch 1000 ---> log loss = 0.140