Title: Machine Learning (2021/2022)

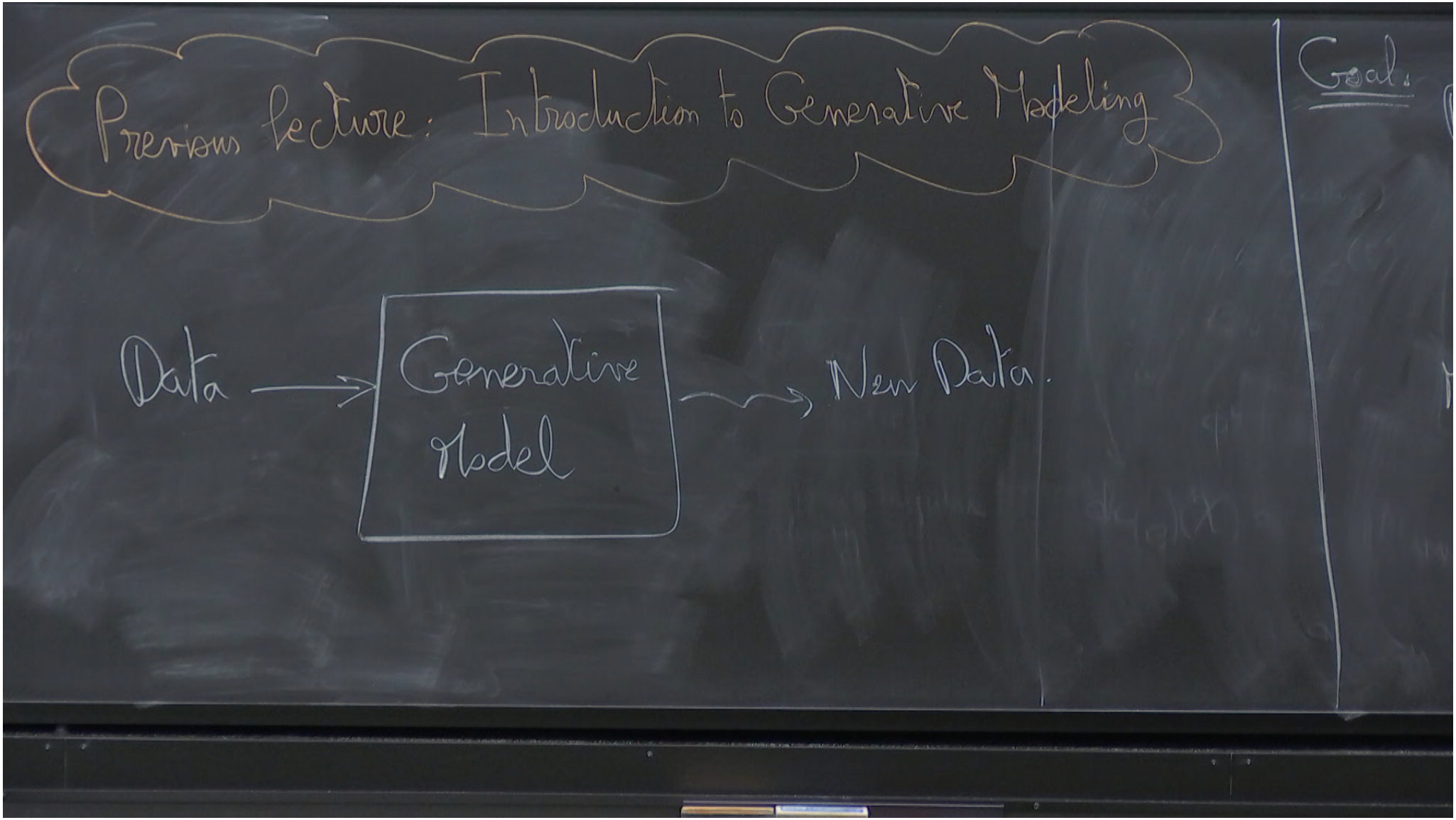
Speakers: Lauren Hayward

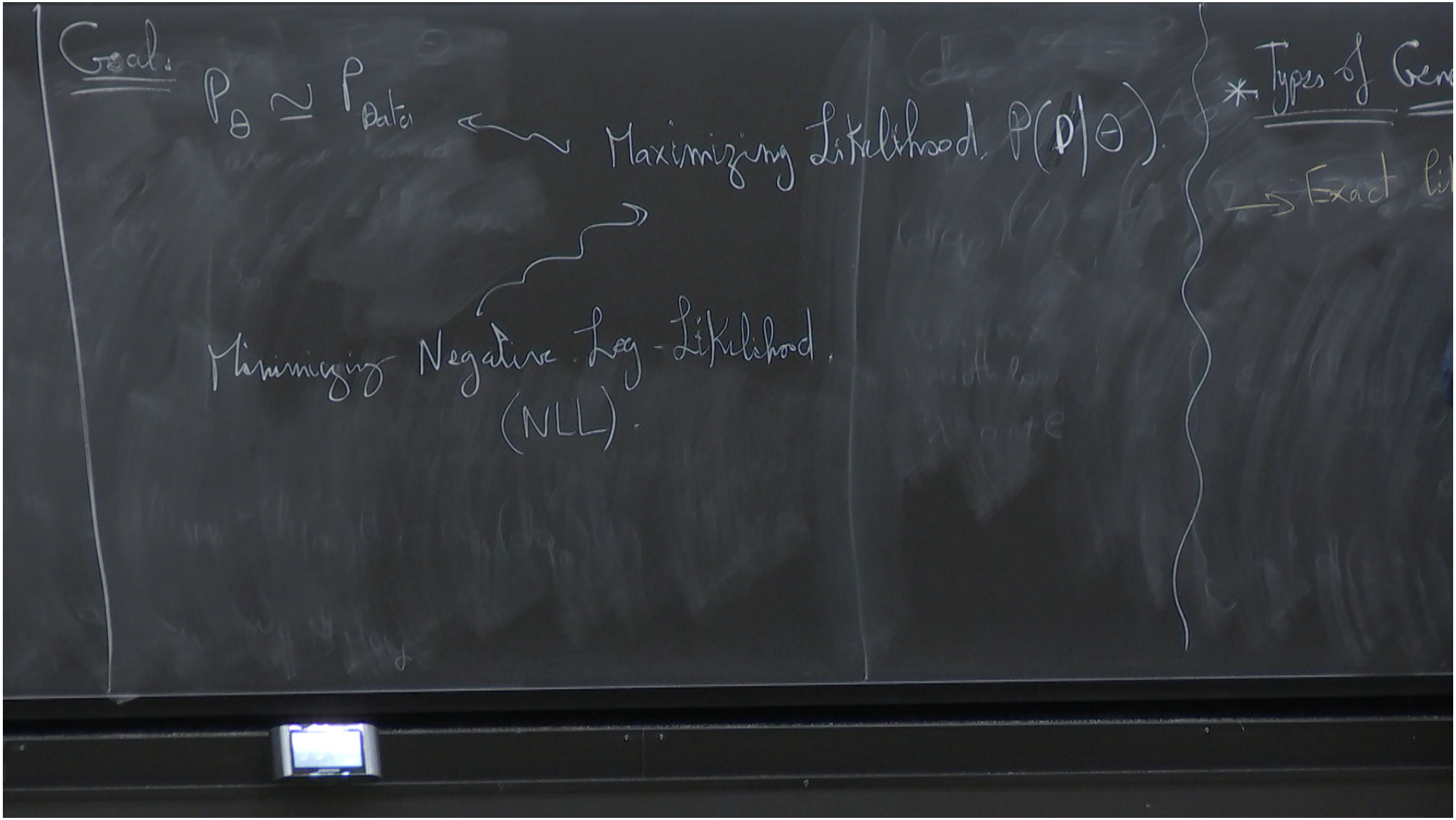
Collection: Machine Learning (2021/2022)

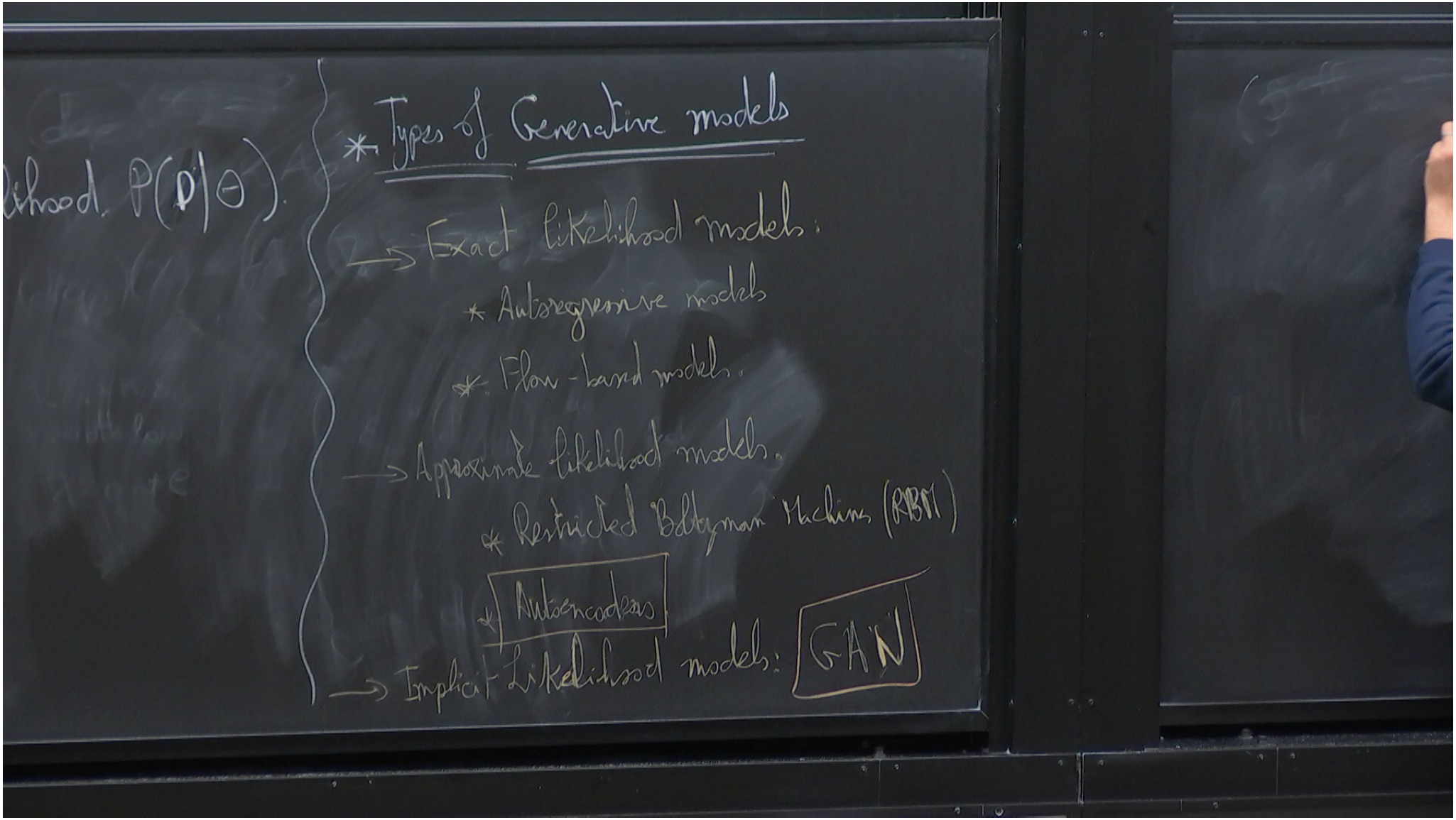
Date: April 28, 2022 - 11:30 AM

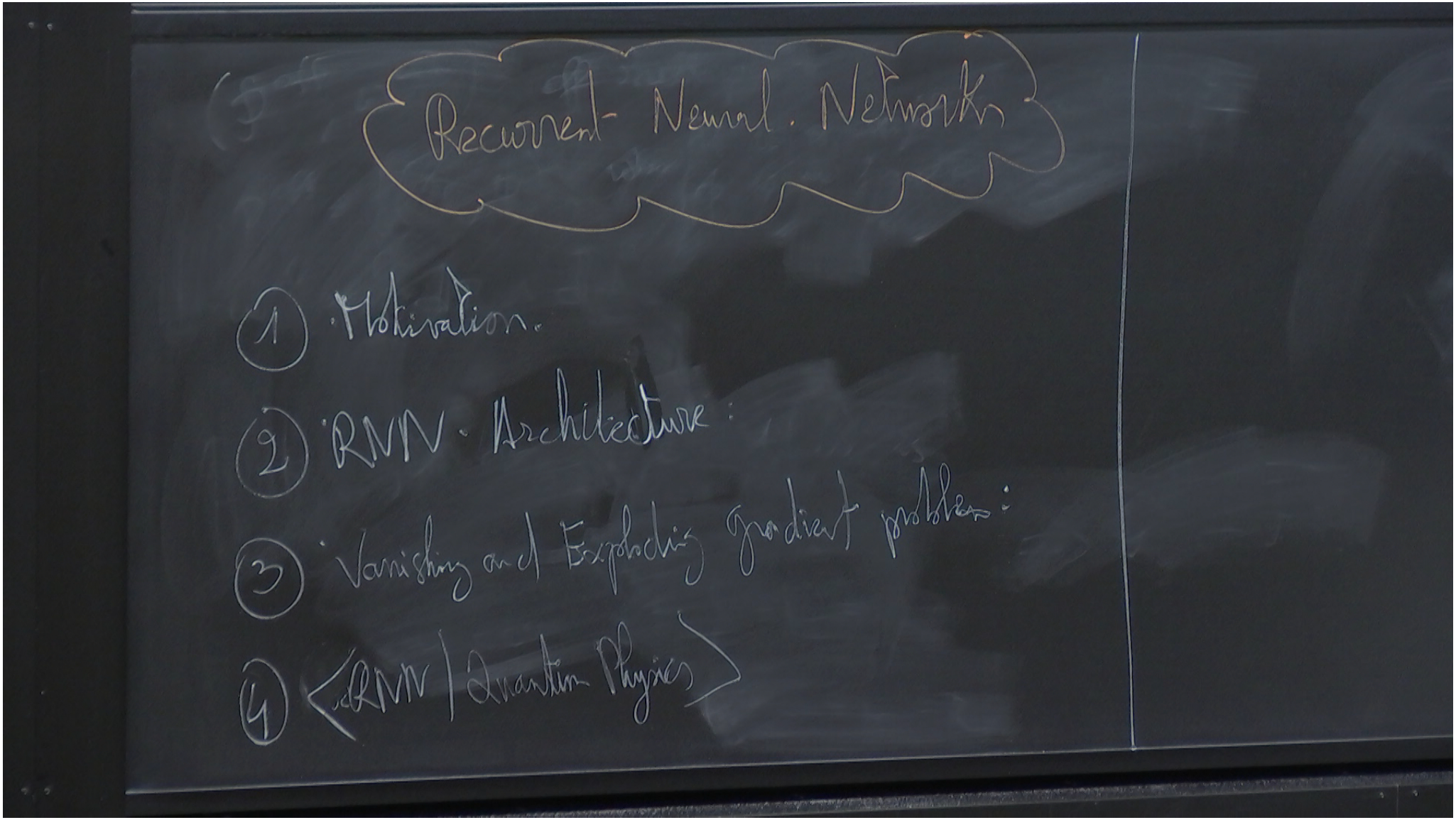
URL: https://pirsa.org/22040075

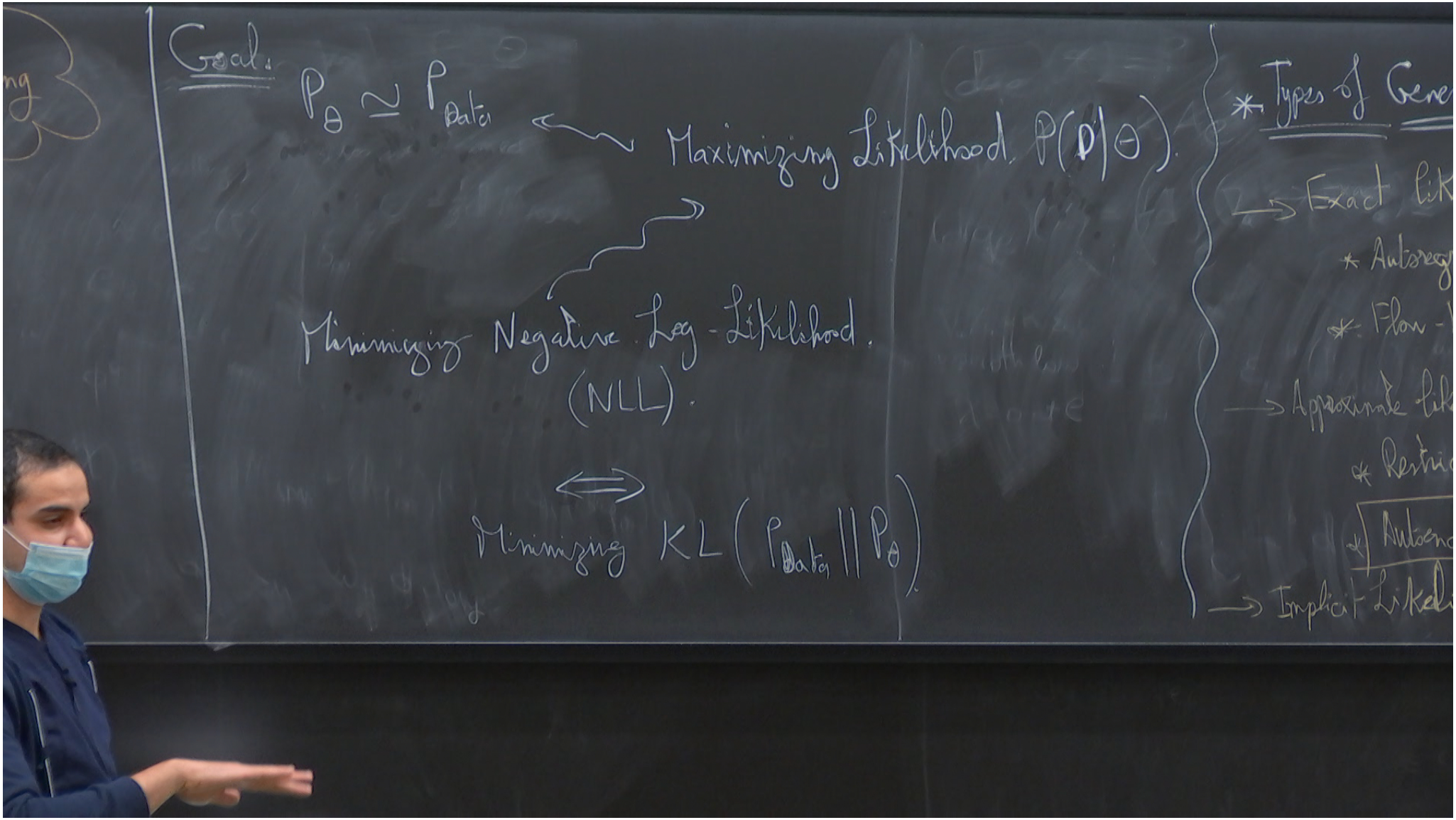
Abstract: This course is designed to introduce modern machine learning techniques for studying classical and quantum many-body problems encountered in condensed matter, quantum information, and related fields of physics. Lectures will focus on introducing machine learning algorithms and discussing how they can be applied to solve problem in statistical physics. Tutorials and homework assignments will concentrate on developing programming skills to study the problems presented in lecture.

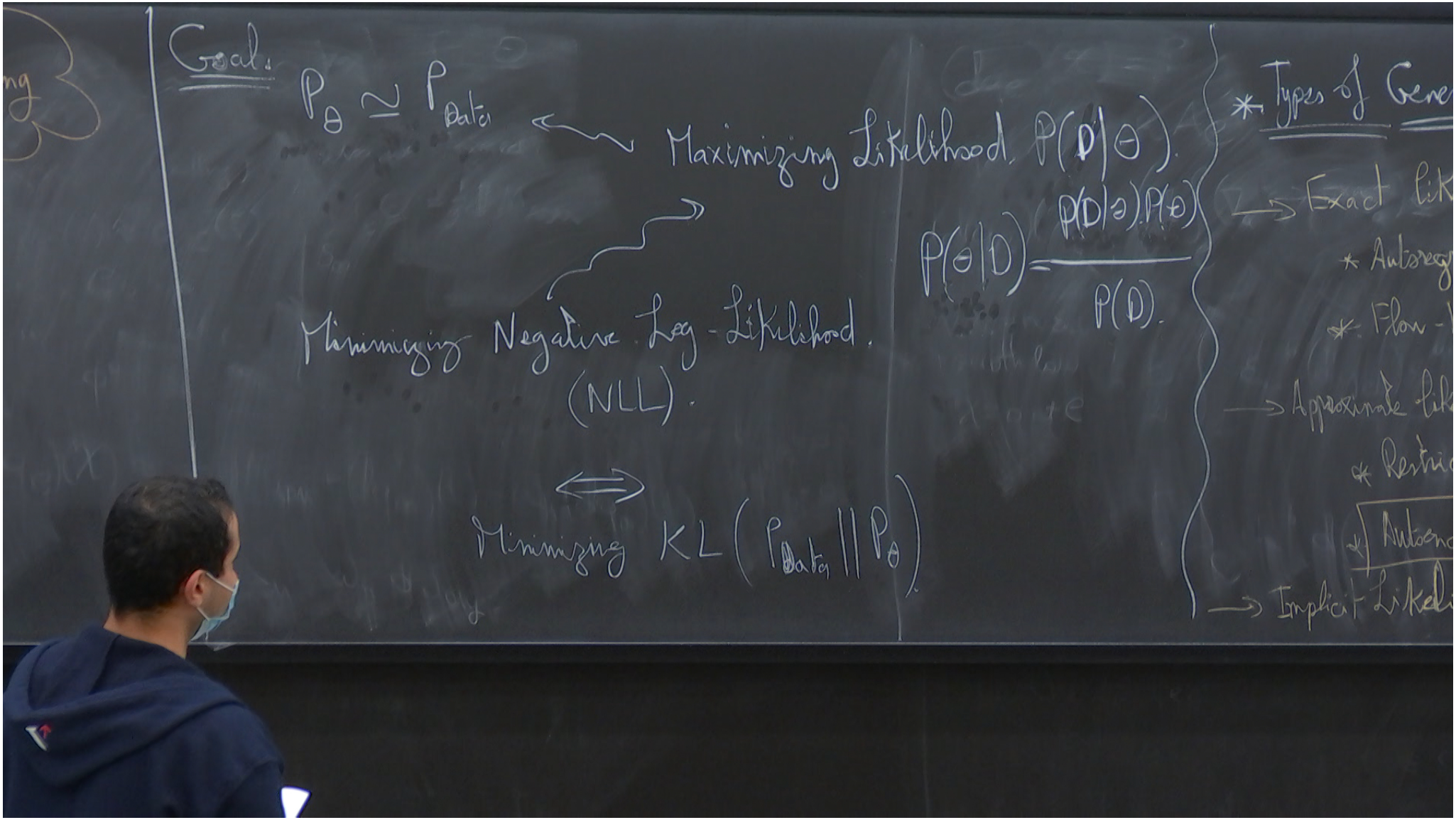

Previous lecture: Introduction to Generative Modeling
Data
Generative Model
New Data.
Goal:

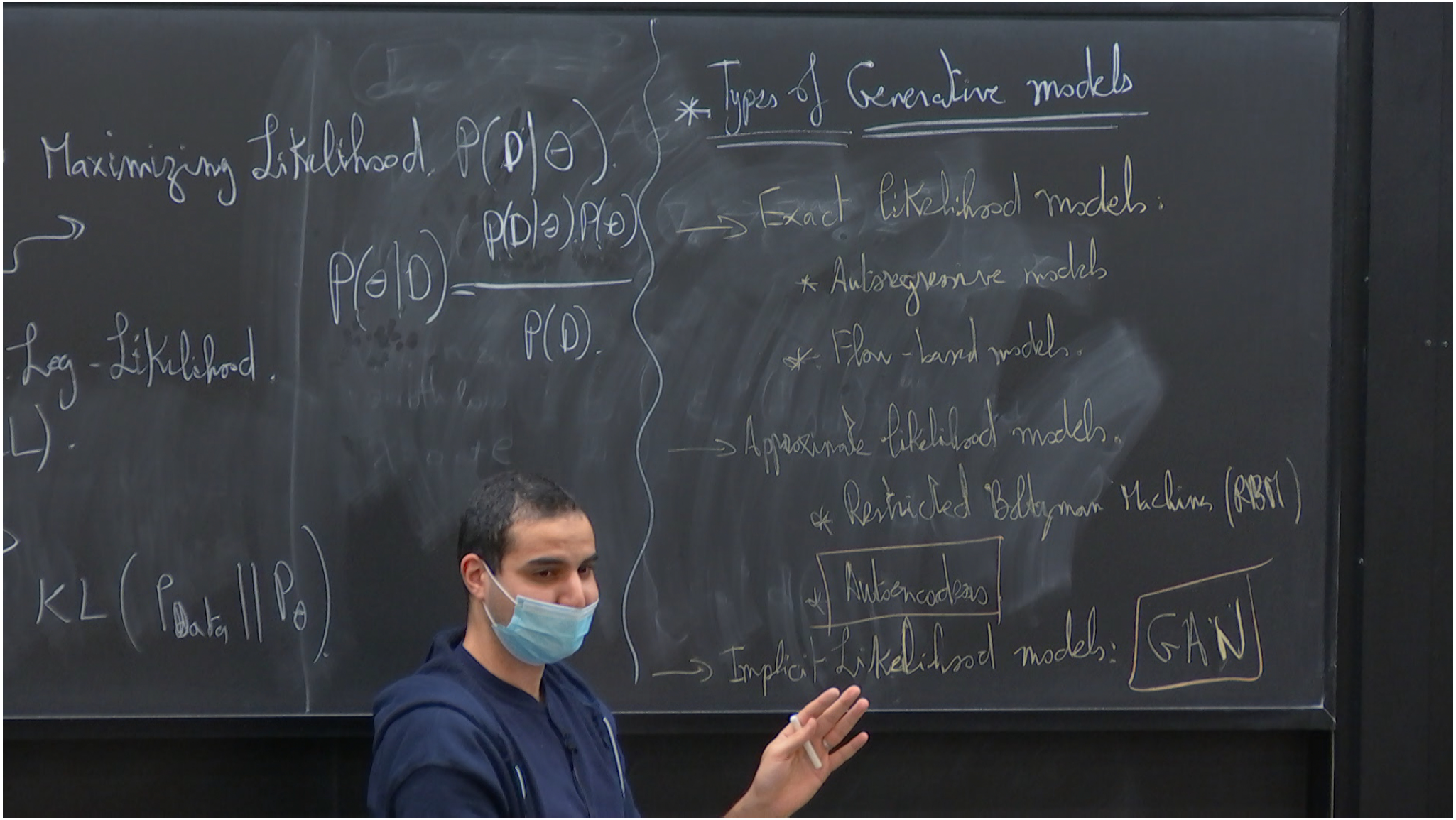

Goal:
$P_\theta \simeq P_{Data}$
Maximizing Likelihood $P(D|\theta)$.
Minimizing Negative Log-Likelihood (NLL).
* Types of Gen
→ Exact

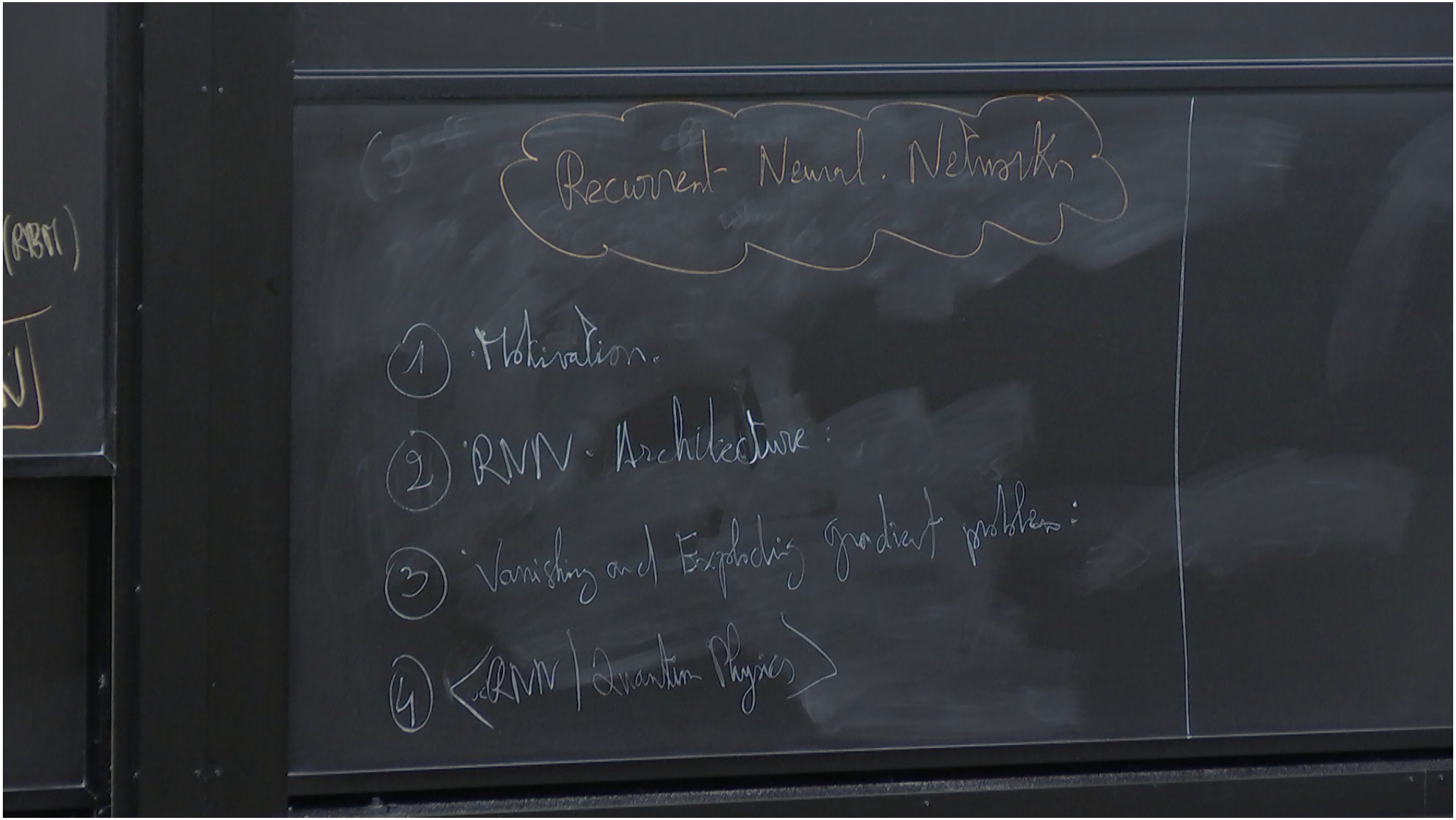

lihood. P(D|θ).
* Types of Generative models
→ Exact Likelihood models:
* Autoregressive models
* Flow-based models.
→ Approximate likelihood models.
* Restricted Boltzman Machine (RBM)
* Autoencoders
→ Implicit Likelihood models: GAN


Recurrent Neural Networks
1 Motivation.
2 RNN - Architecture:
3 Vanishing and Exploding gradient problem:
4 <RNN / Quantum Physics>


Goal:
$P_\theta \simeq P_{Data}$
Maximizing Likelihood. $P(D|\theta)$.
Minimizing Negative Log-Likelihood. (NLL).
$\Leftrightarrow$
Minimizing $KL(P_{Data} || P_\theta)$
* Types of Gene
→ Exact lik
* Autoreg
* Flow -
→ Approximate lik
* Restri
* Autoen
→ Implicit Likeli


Goal:
$P_\theta \simeq P_{Data}$
Maximizing Likelihood. $P(D|\theta)$.
$P(\theta|D) = \frac{P(D|\theta).P(\theta)}{P(D)}$
Minimizing Negative Log-Likelihood. (NLL).
$\Leftrightarrow$
Minimizing $KL(P_{Data} || P_\theta)$
* Types of Gene
→ Exact lik
* Autoreg
* Flow -
→ Approximate lik
* Restri
* Autoen
→ Implicit Likeli


Maximizing Likelihood. P(D|θ).
Log-Likelihood.
KL(P_data || P_θ)
P(θ|D) = P(D|θ).P(θ) / P(D).
* Types of Generative models
→ Exact likelihood models:
* Autoregressive models
* Flow-based models.
→ Approximate likelihood models.
* Restricted Boltzman Machins (RBM)
* Autoencoders
→ Implicit Likelihood models: GAN


Recurrent Neural Networks
1 Motivation.
2 RNN Architecture:
3 Vanishing and Exploding gradient problem:
4 <RNN / Quantum Physics>

$P_\theta \simeq P_{Data}$ ⟷ Maximizing Likelihood. $P(D|\theta)$.

$\downarrow$

Minimizing Negative Log-Likelihood. (NLL).

$\Leftrightarrow$

Minimizing $KL(P_{Data} \,||\, P_\theta)$.

$P(\theta|D) = \frac{P(D|\theta).P(\theta)}{P(D)}$.

$\frac{P(\sigma)}{P(\sigma)}$

* Types of Ge

→ Exact li

* Autor

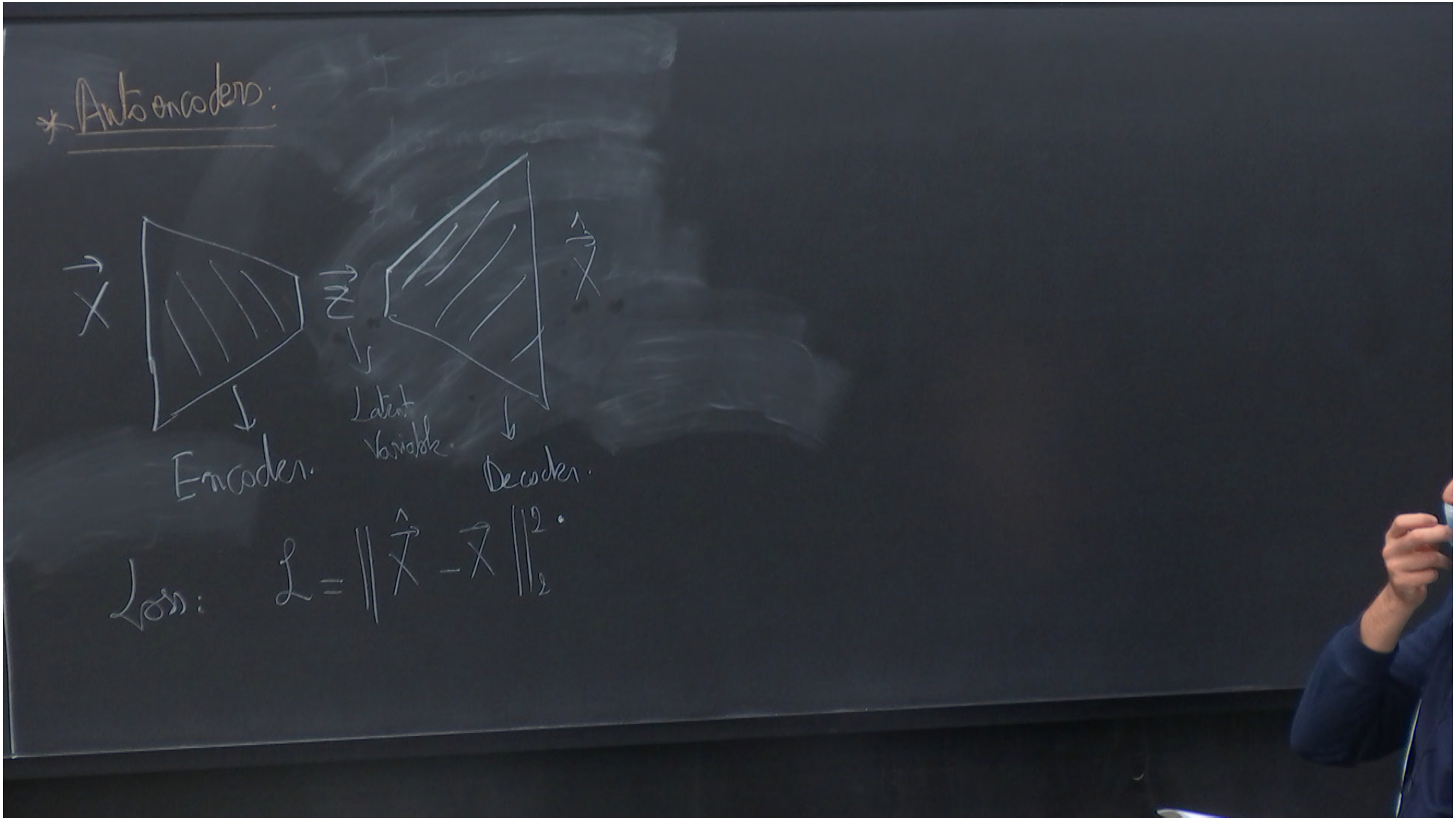
* Flow

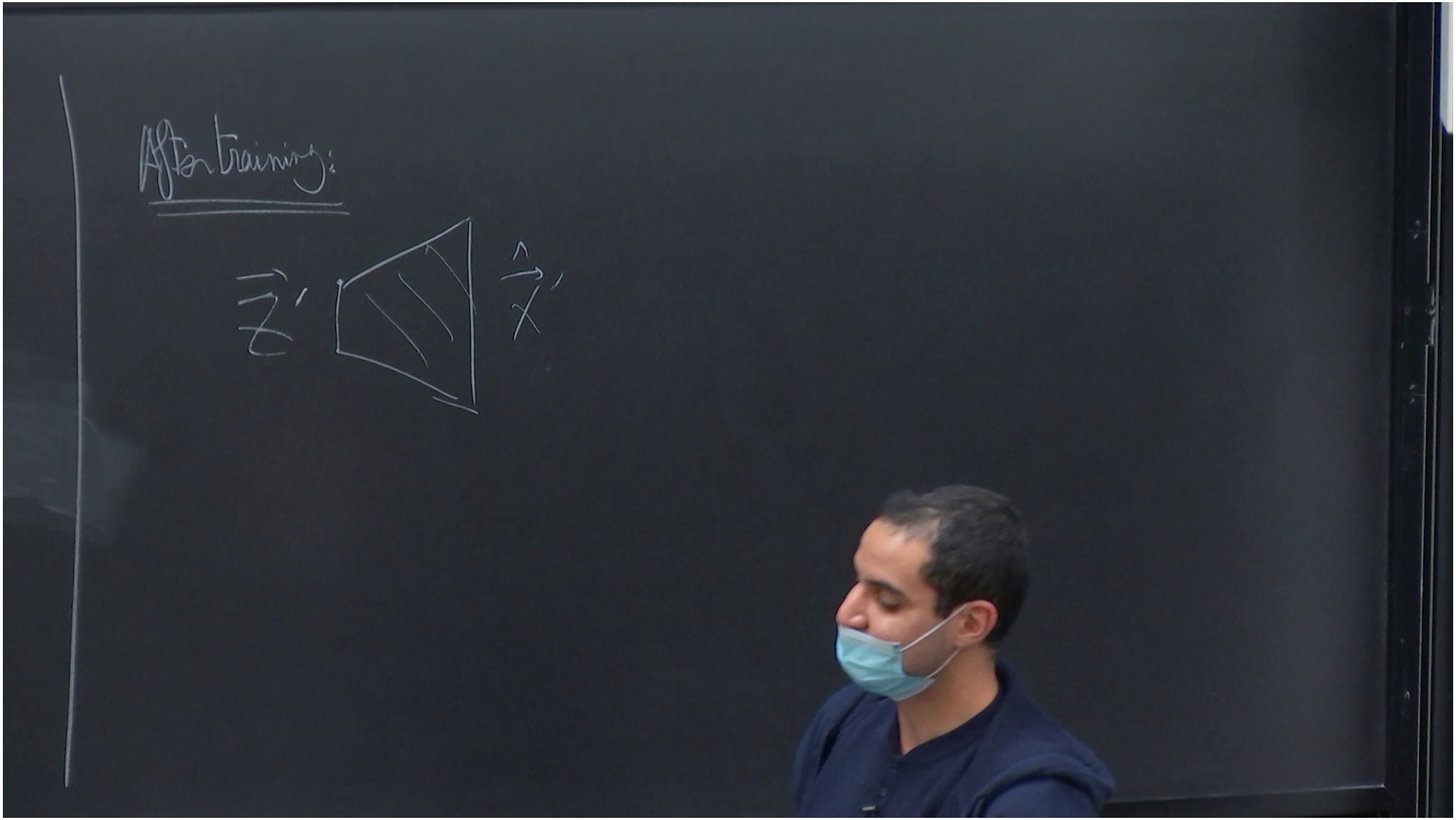
→ Approximate

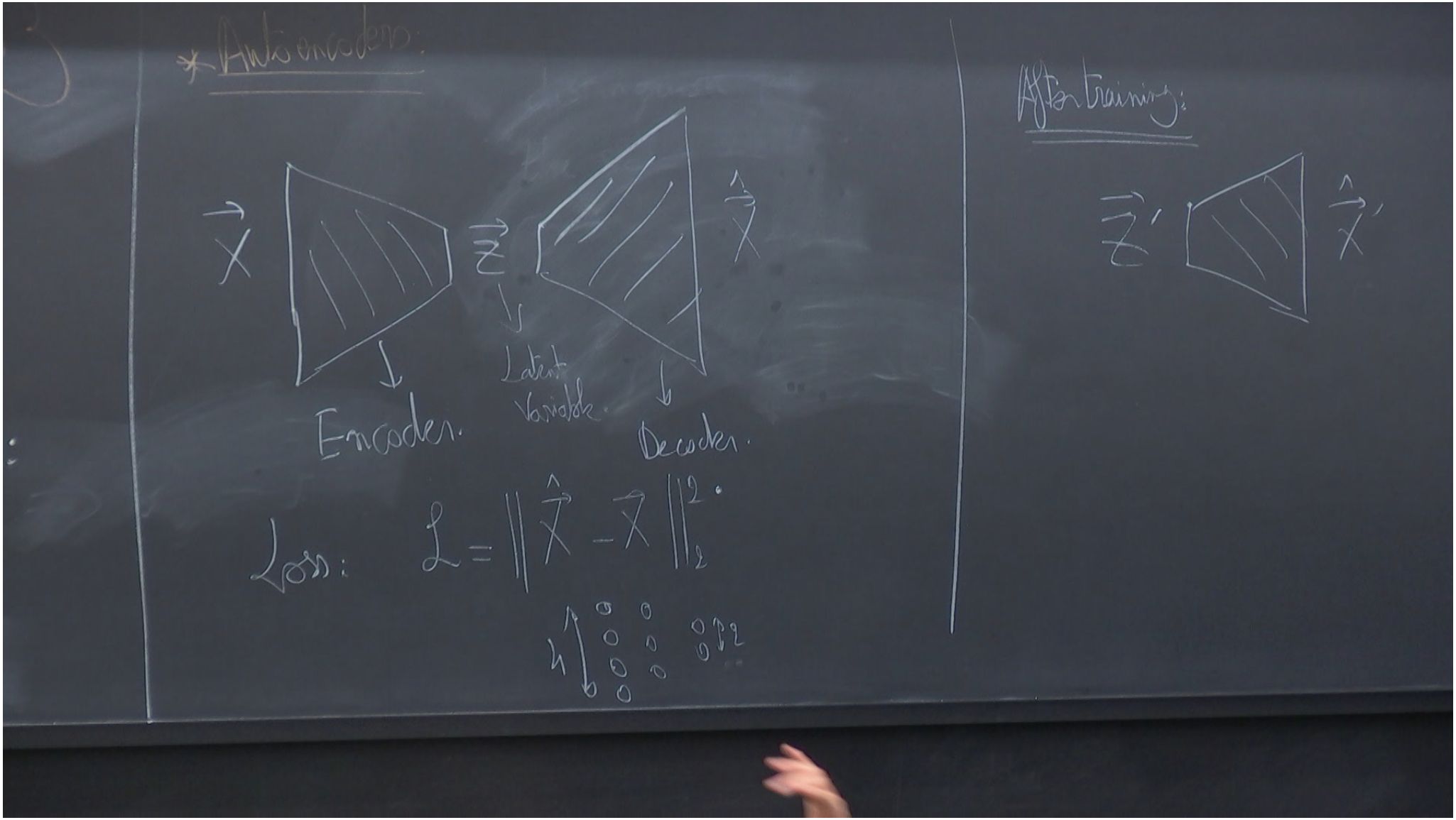
* Rest

* Autoc

→ Implicit Like

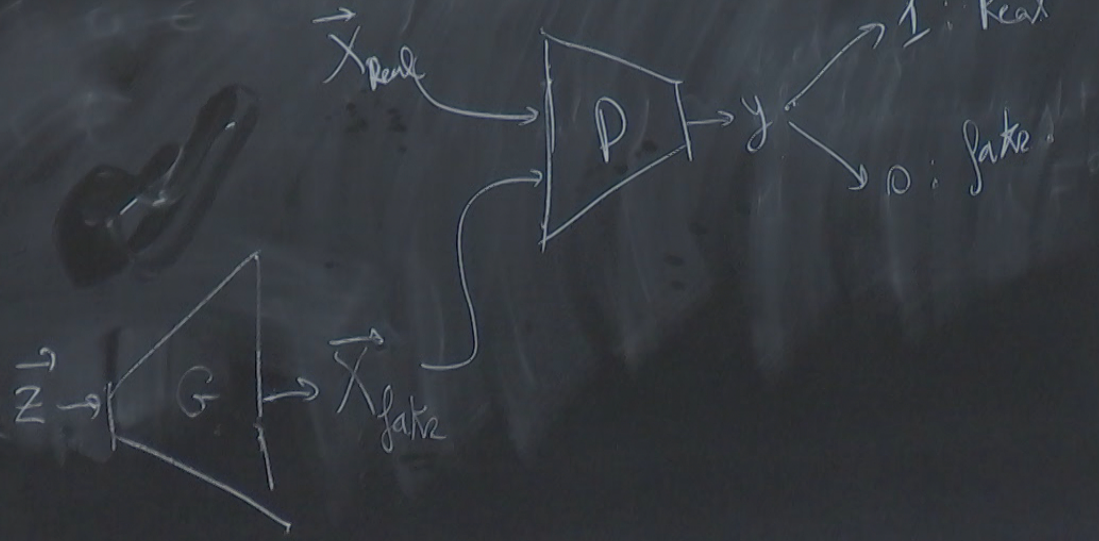

* Autoencoders:
$\vec{x}$
$\vec{z}$
$\hat{\vec{x}}$
Latent Variable.
Encoder.
Decoder.
Loss: $\mathcal{L} = \|\hat{\vec{x}} - \vec{x}\|_2^2$.

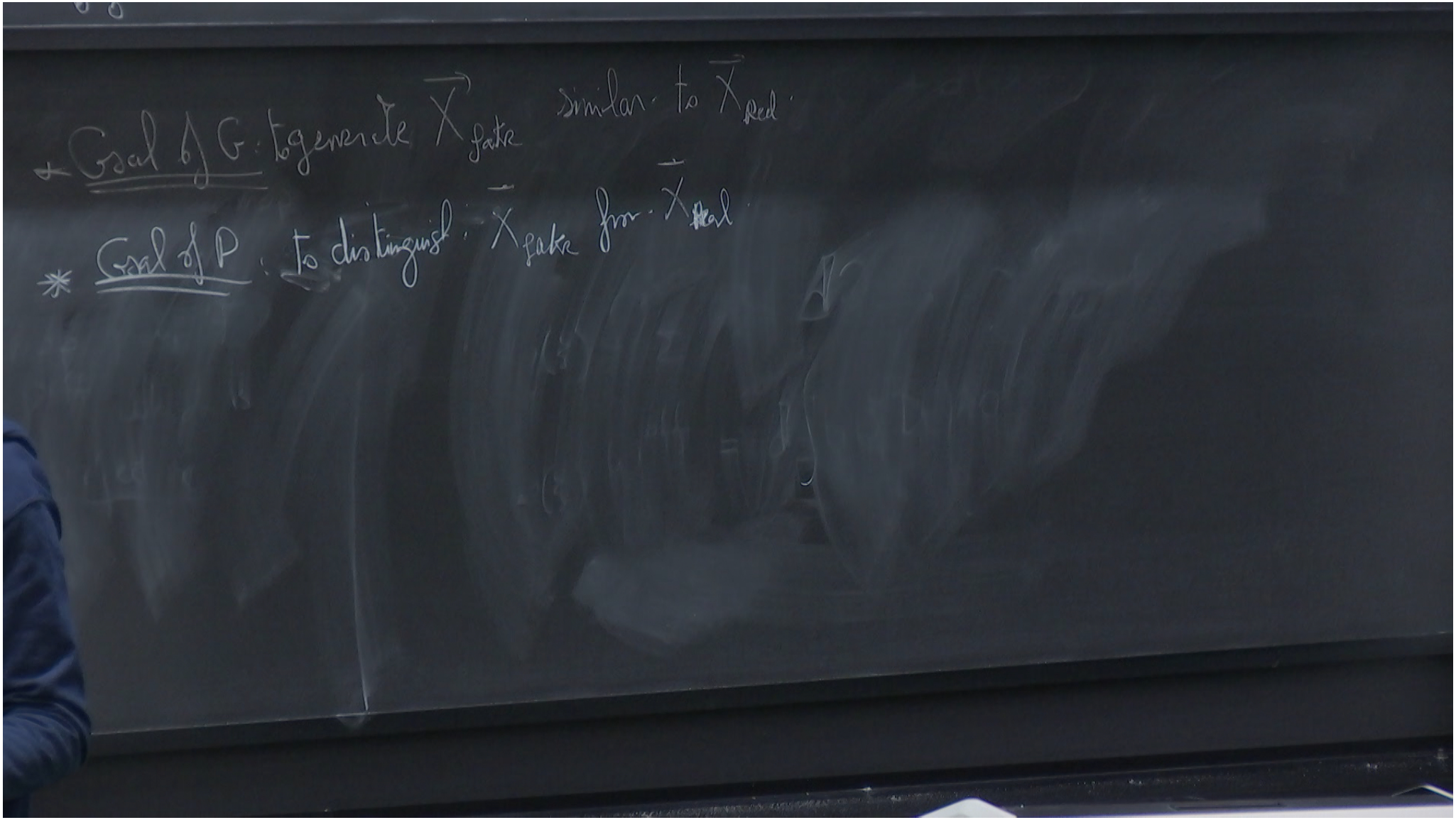

After training:

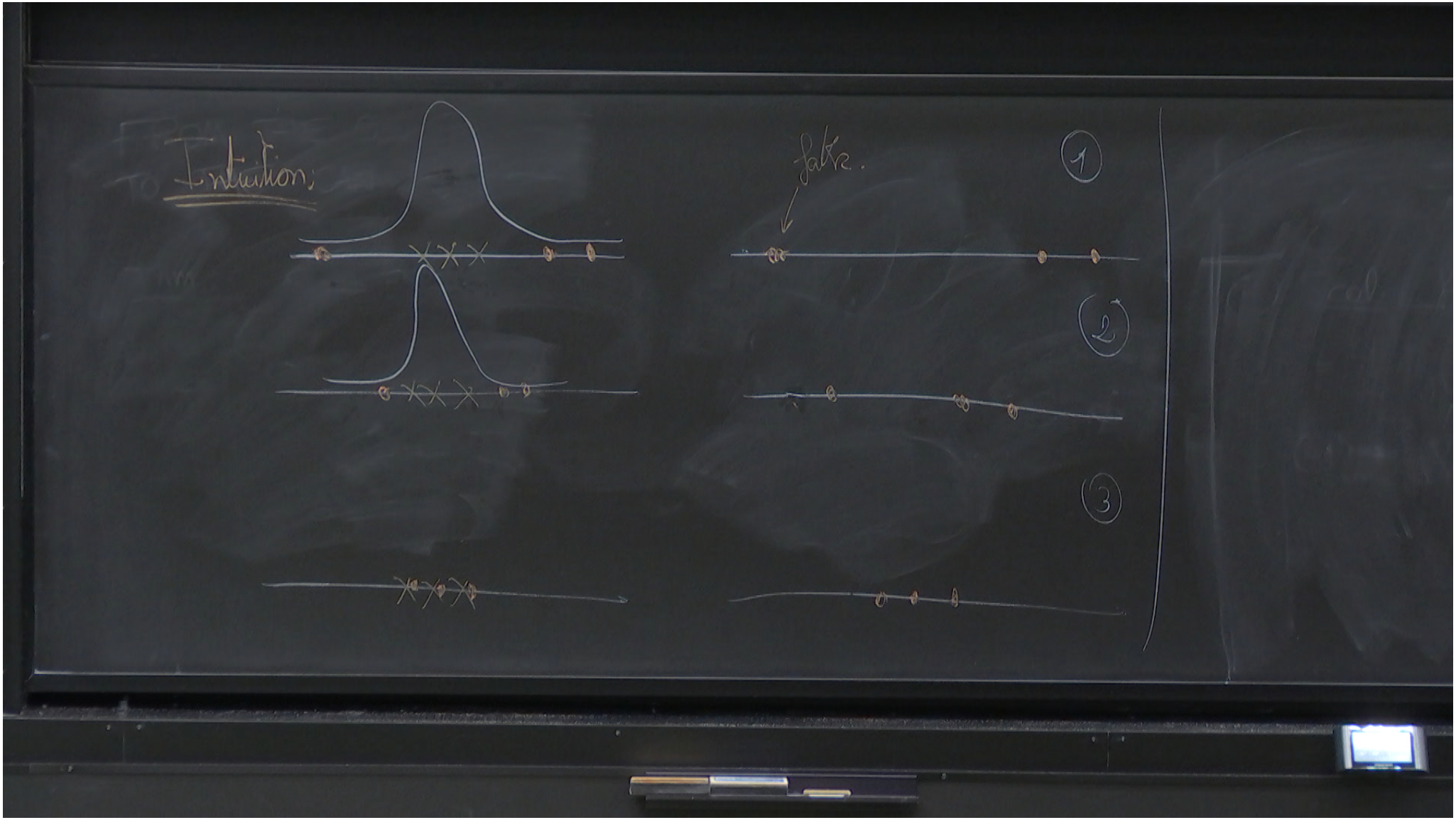

* Auto encoders:
Encoder.
Latent Variable.
Decoder.
Loss:
After training:

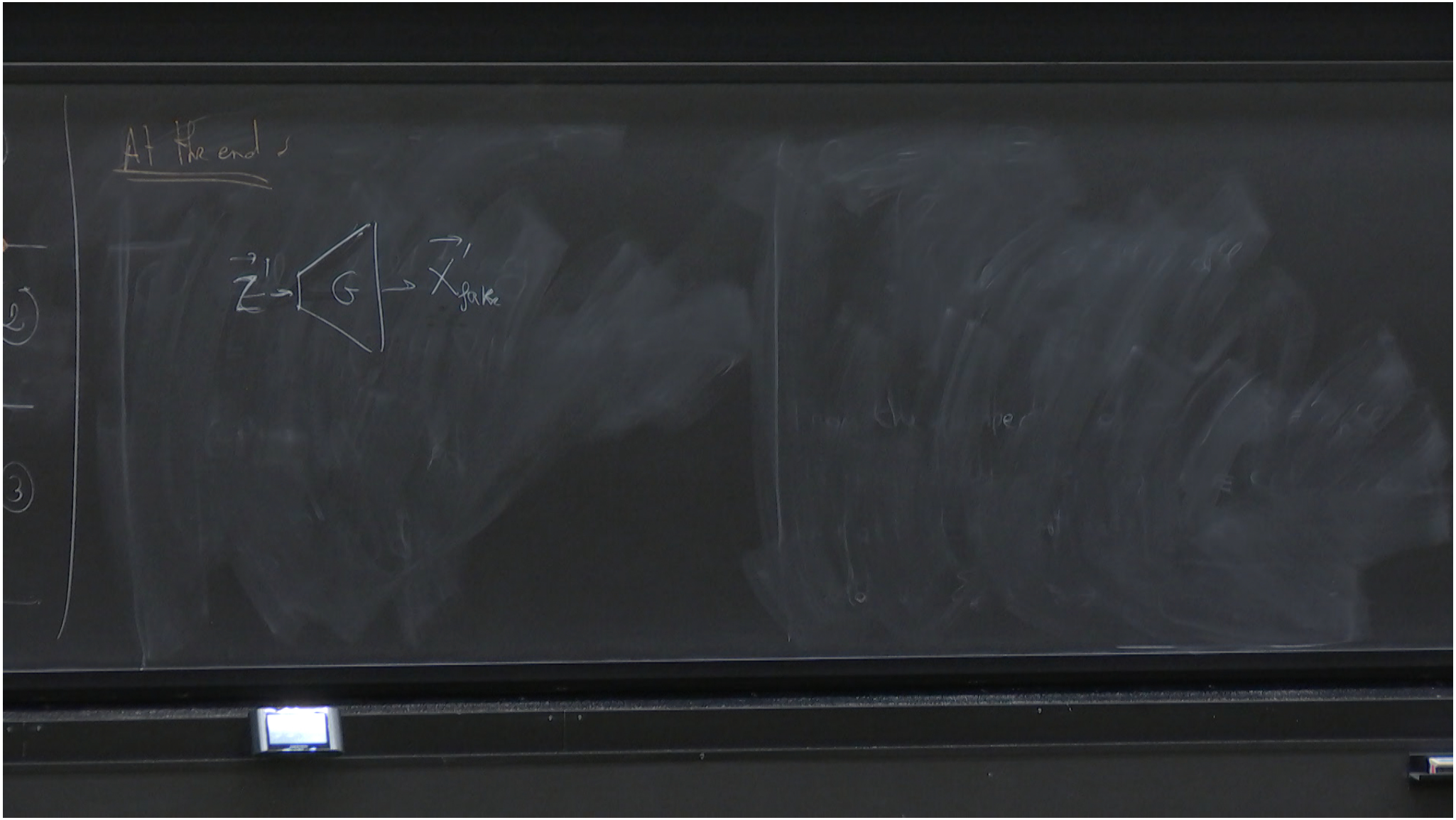
Generative Adversarial Networks (GAN)

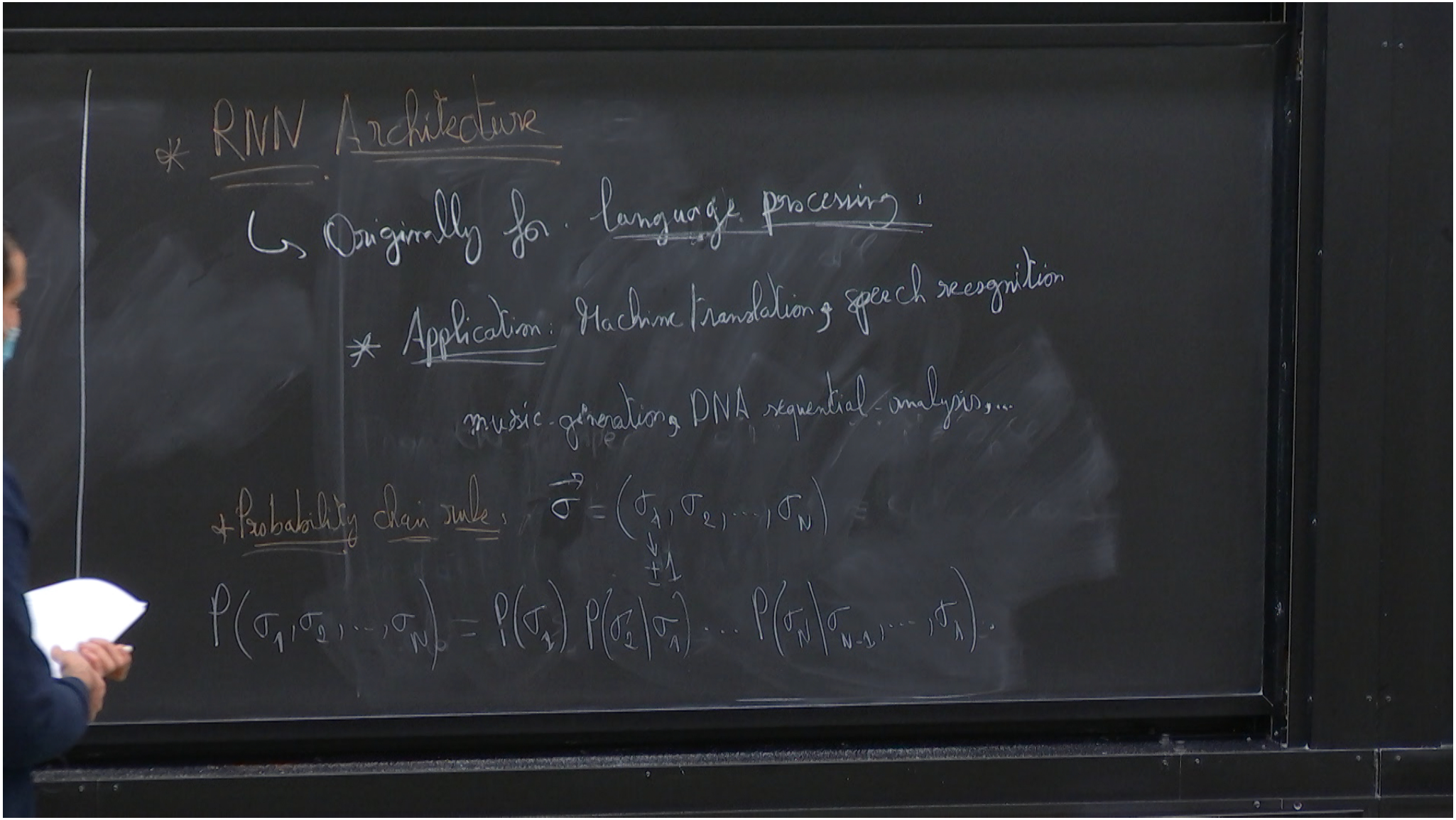
↳ We just want to sample, we don't want to compute $\mathcal{L}_\theta(\vec{x})$

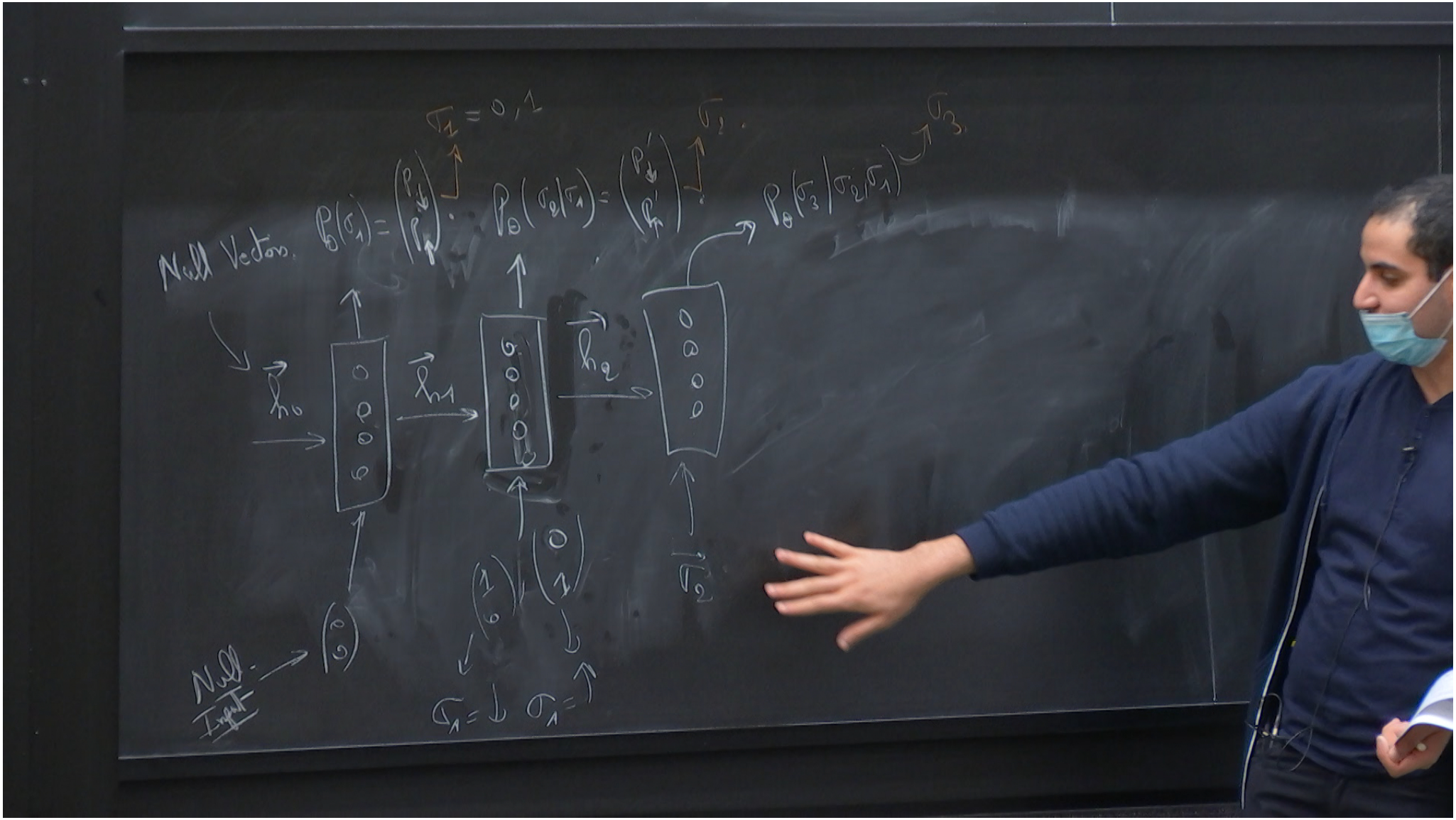


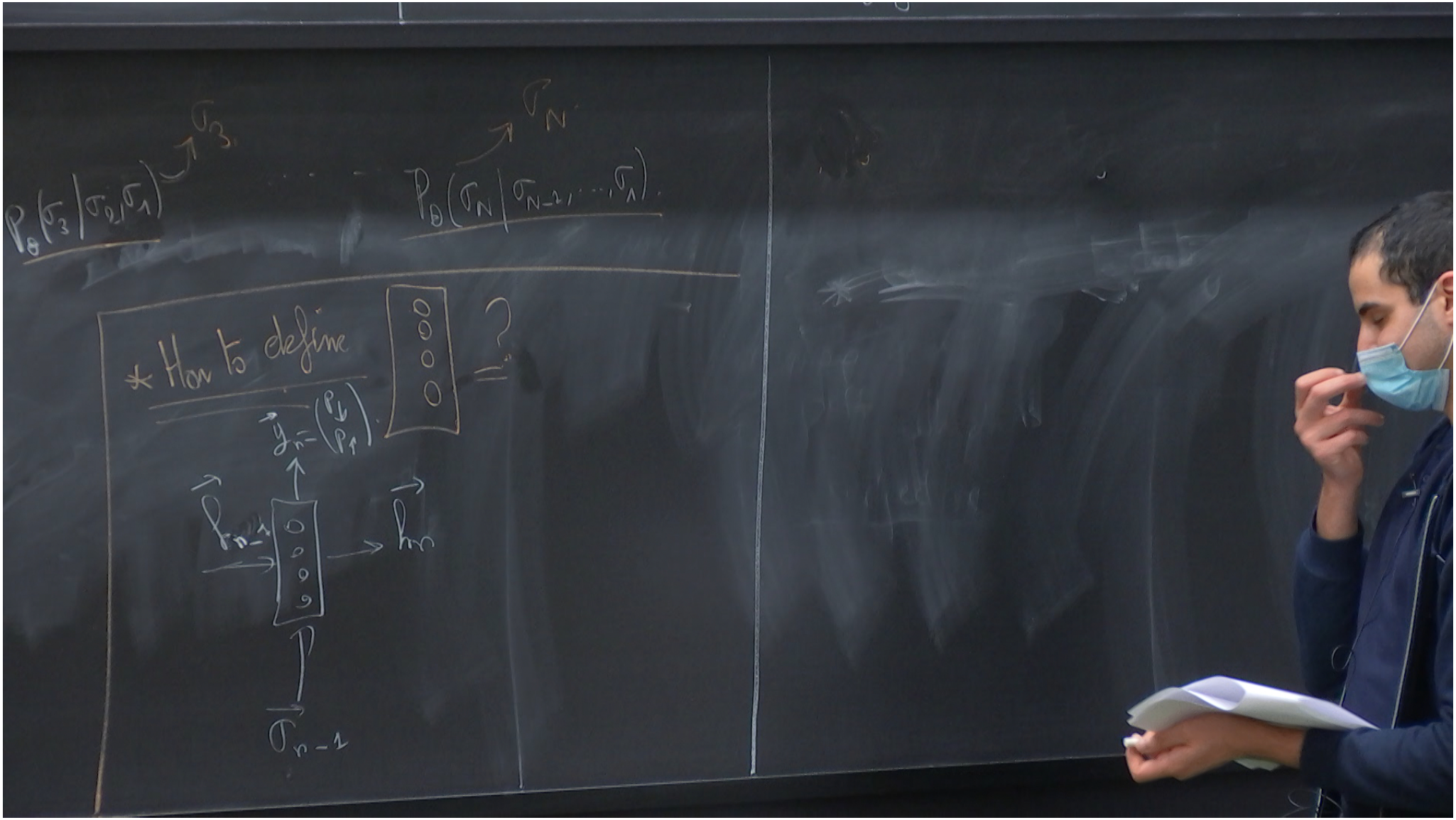

* Goal of G: to generate $\vec{X}_{fake}$ similar to $\vec{X}_{real}$
* Goal of D: To distinguish $\vec{X}_{fake}$ from $\vec{X}_{real}$

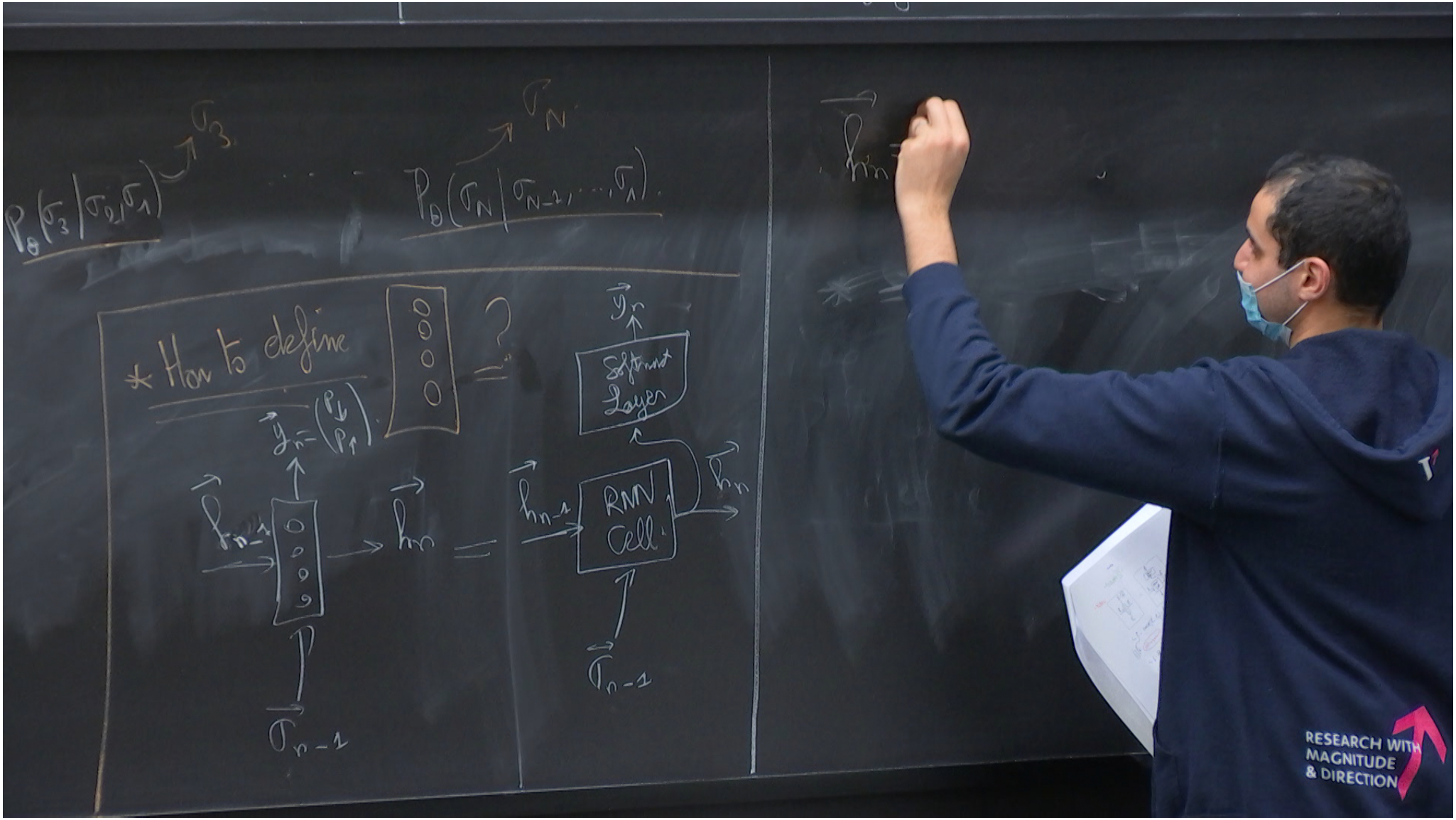

Intuition:
later.
1
2
3

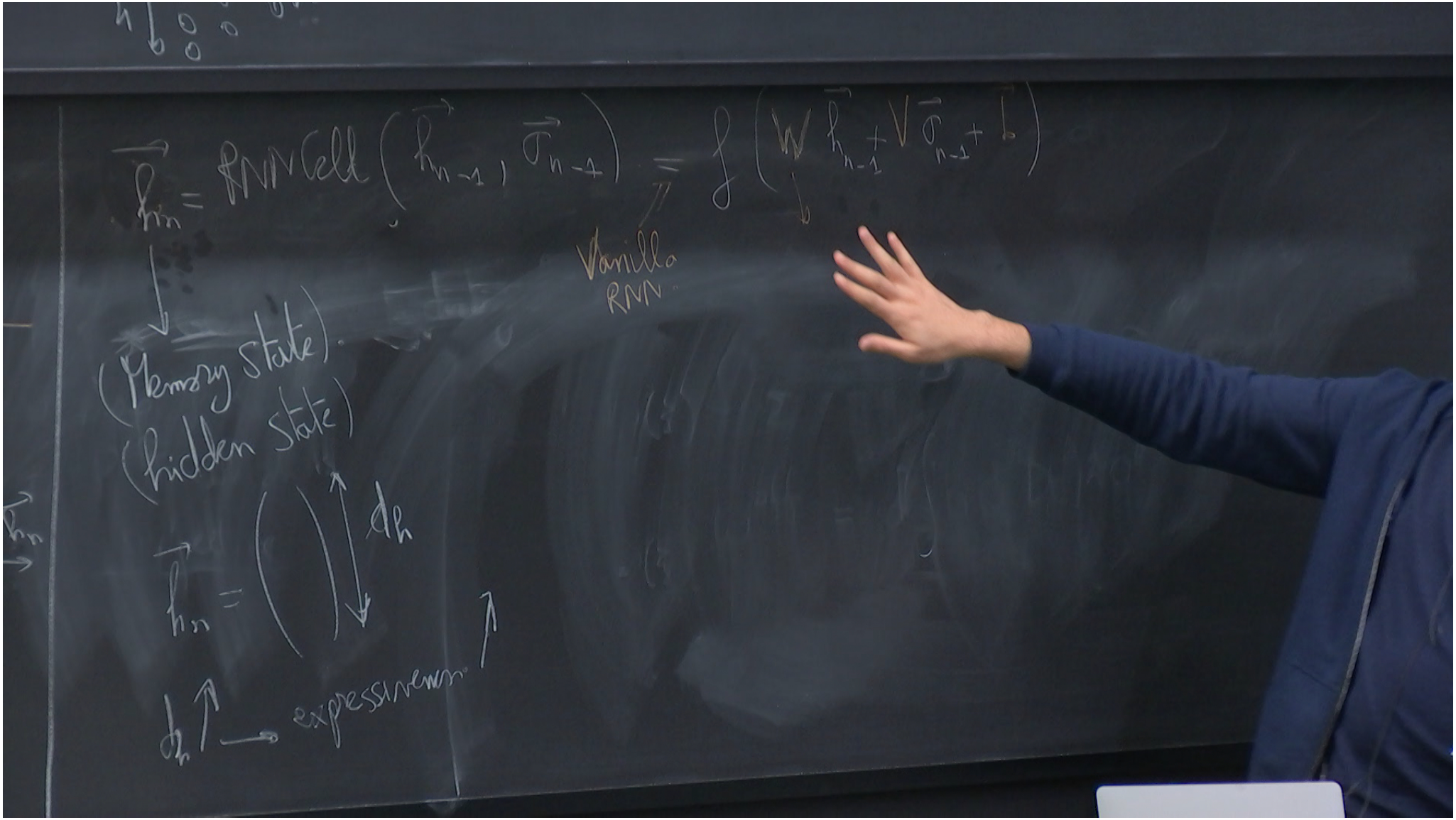

At the end :
$\vec{z}'$
G
$\vec{x}'_{fake}$

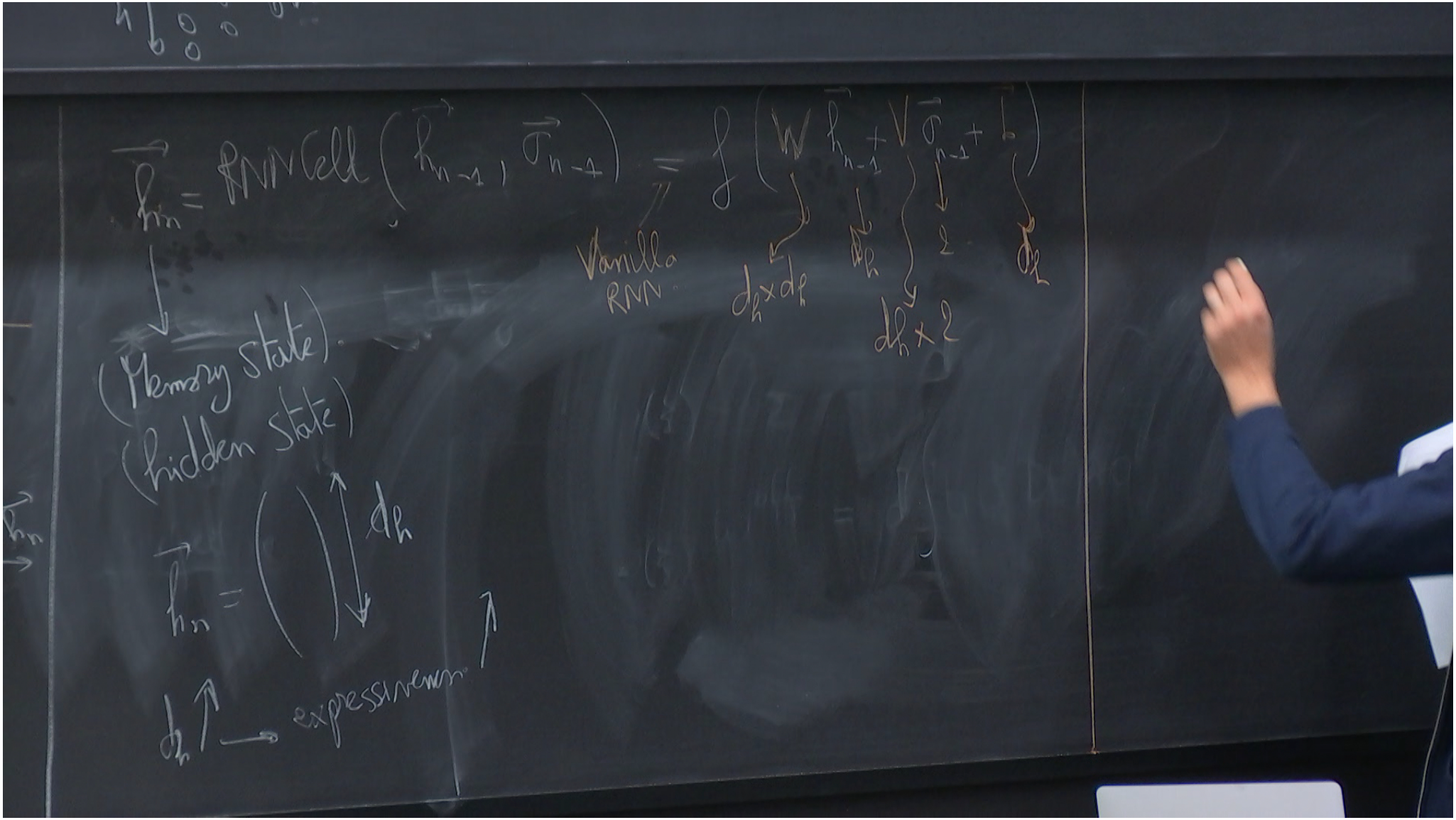

* RNN Architecture
↳ Originally for language processing:
* Application: Machine translation, speech recognition
music generation, DNA sequential analysis, ...
* Probability chain rule: $\vec{\sigma} = (\sigma_1, \sigma_2, \cdots, \sigma_N)$
$\sigma_i = \pm 1$
$P(\sigma_1, \sigma_2, \cdots, \sigma_N) = P(\sigma_1)\, P(\sigma_2|\sigma_1) \cdots P(\sigma_N|\sigma_{N-1}, \cdots, \sigma_1)$.

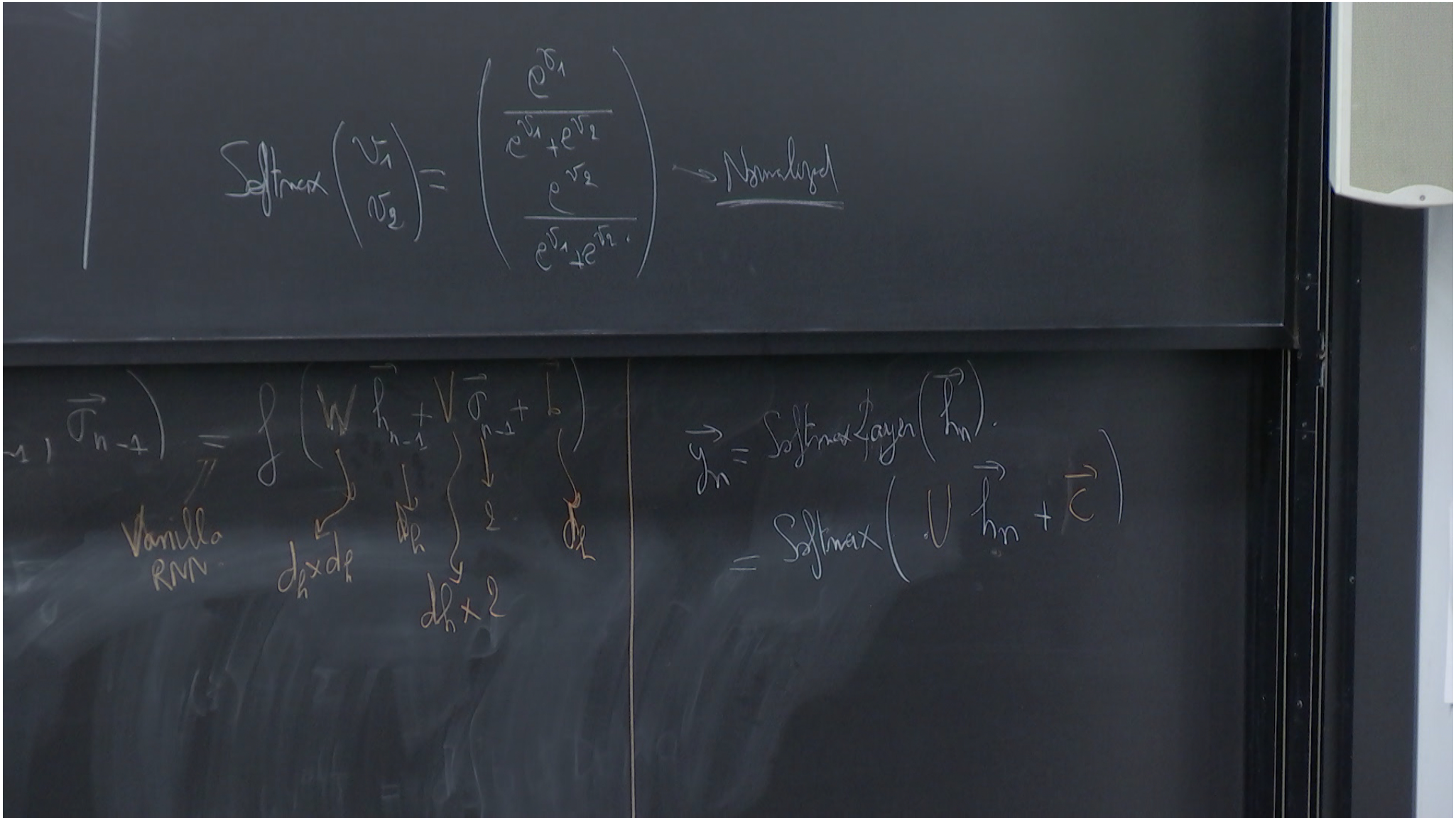

Null Vectors.
$P_\theta(\sigma_1) = \begin{pmatrix} p_\downarrow \\ p_\uparrow \end{pmatrix}$
$\sigma_1 = 0, 1$
$P_\theta(\sigma_2|\sigma_1) = \begin{pmatrix} p_\downarrow \\ p_\uparrow \end{pmatrix}$
σ_2
$P_\theta(\sigma_3|\sigma_2\sigma_1)$
σ_3
$\vec{h}_0$
$\vec{h}_1$
$\vec{h}_2$
Null Input
$\sigma_1 = \downarrow$
$\sigma_1 = \uparrow$
$\vec{\sigma}_2$


$P_\theta(\sigma_3 | \sigma_2, \sigma_1)$
σ_3
$P_\theta(\sigma_N | \sigma_{N-1}, \dots, \sigma_1)$
σ_N
* How to define
?
$\vec{y}_n = \begin{pmatrix} p_0 \\ p_1 \end{pmatrix}$
$\vec{h}_{n-1}$
$\vec{h}_n$
P
$\vec{\sigma}_{n-1}$


$P_\theta(\sigma_3|\sigma_2,\sigma_1)$
$P_\theta(\sigma_N|\sigma_{N-1},\dots,\sigma_1)$
* How to define
Softmax Layer
RNN Cell
RESEARCH WITH MAGNITUDE & DIRECTION


Vanilla RNN
(Memory state)
(hidden state)
expressiveness


(Memory state)
(hidden state)
expressiveness
Vanilla RNN


Softmax
Normalized
Vanilla RNN
Softmax layer

$\vec{z} \rightarrow \vec{x}'_{fake}$

* RNN Architecture

↳ Originally for language processing :

* Application : Machine translation, speech recognition, music generation, DNA sequential analysis, ...

* Probability chain rule : $\vec{\sigma} = (\sigma_1, \sigma_2, \cdots, \sigma_N)$ $\quad \sigma_i = \pm 1$

$$|\psi_{RNN}|^2 = P_\theta(\sigma_1, \sigma_2, \cdots, \sigma_N) = P_\theta(\sigma_1)\, P_\theta(\sigma_2|\sigma_1) \cdots P_\theta(\sigma_N|\sigma_{N-1}, \cdots, \sigma_1).$$