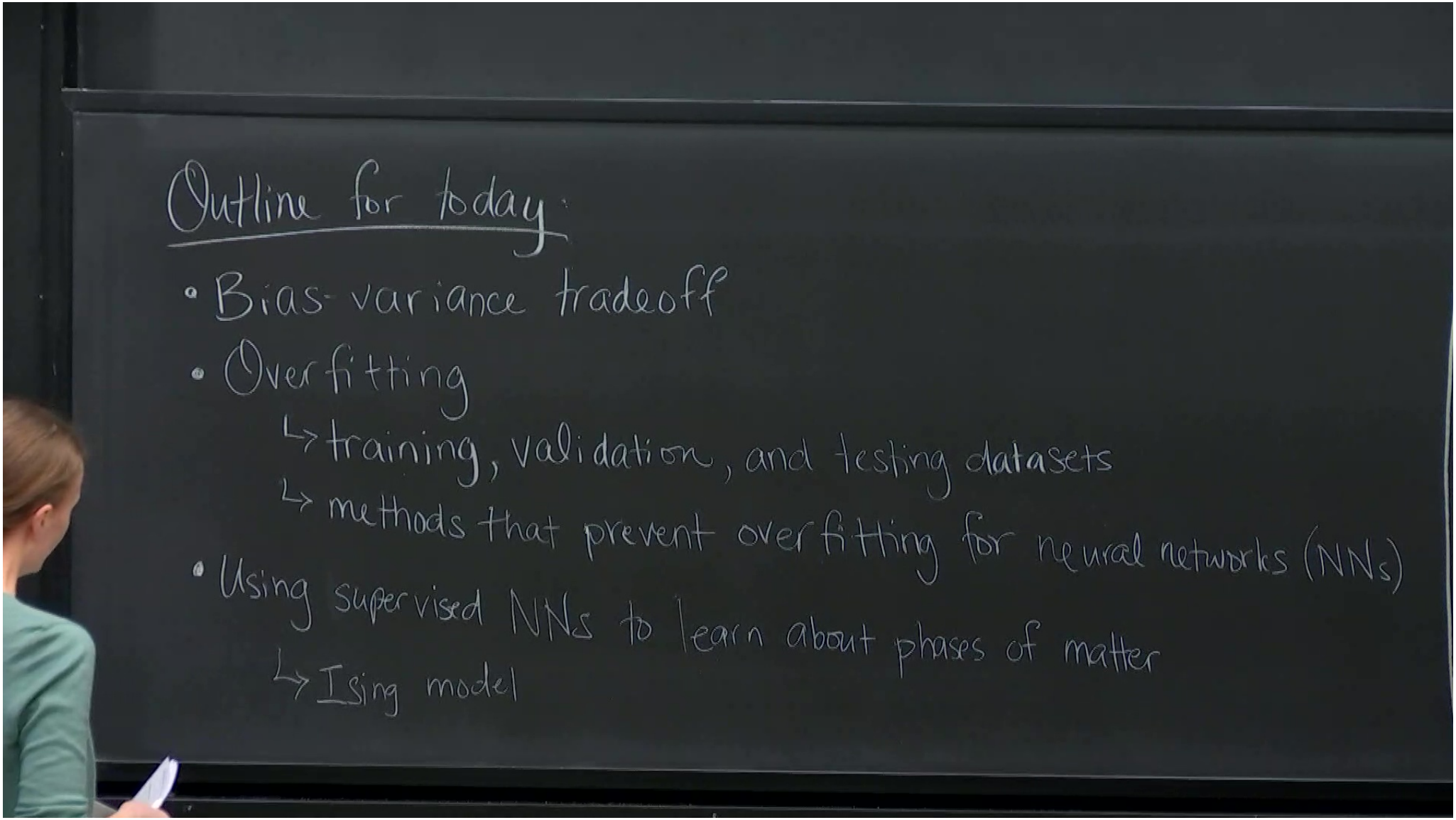
Title: Machine Learning Lecture - 230307

Speakers: Lauren Hayward

Collection: Machine Learning for Many-Body Physics (2022/2023)

Date: March 07, 2023 - 9:00 AM

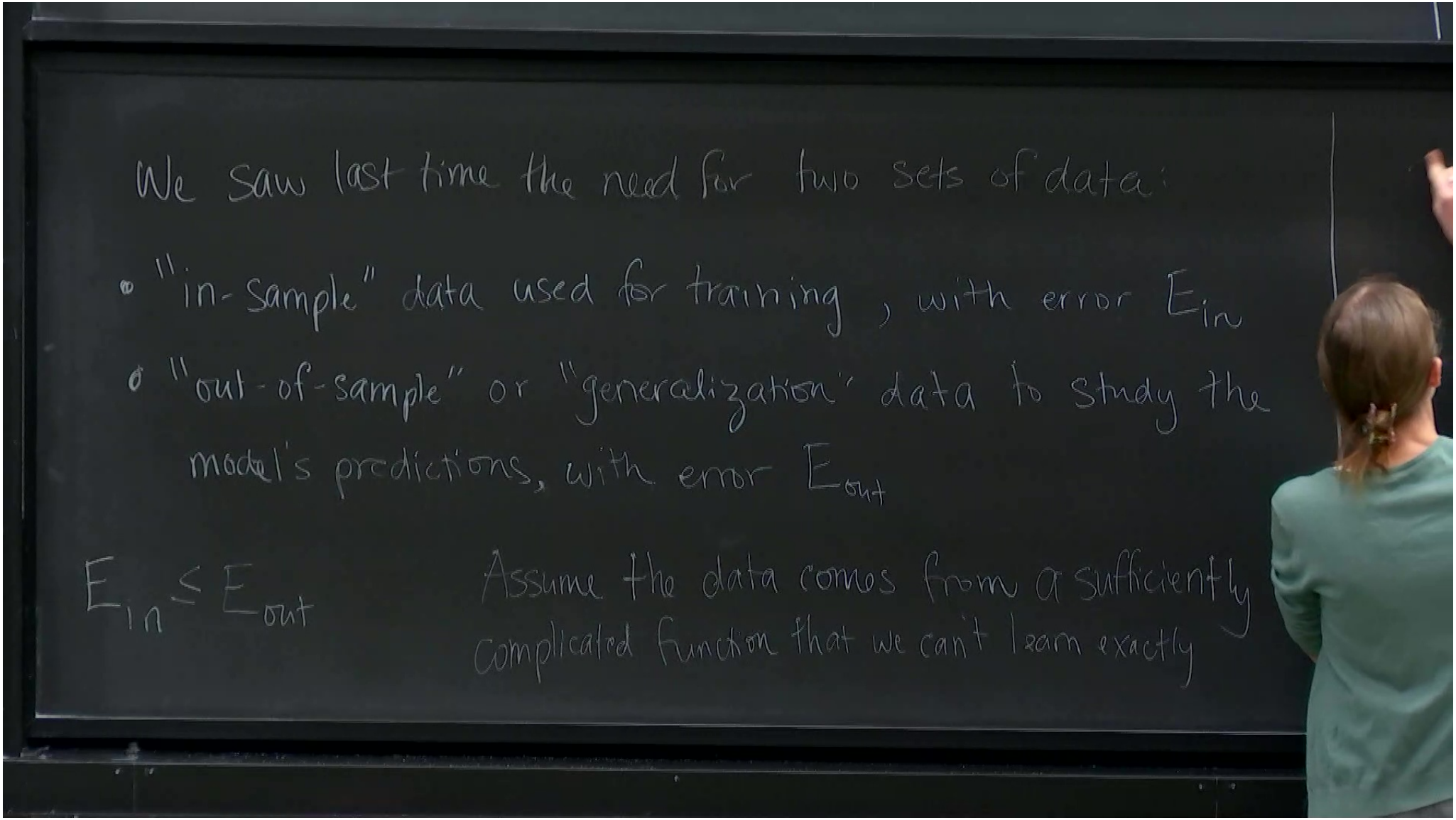
URL: https://pirsa.org/23030030

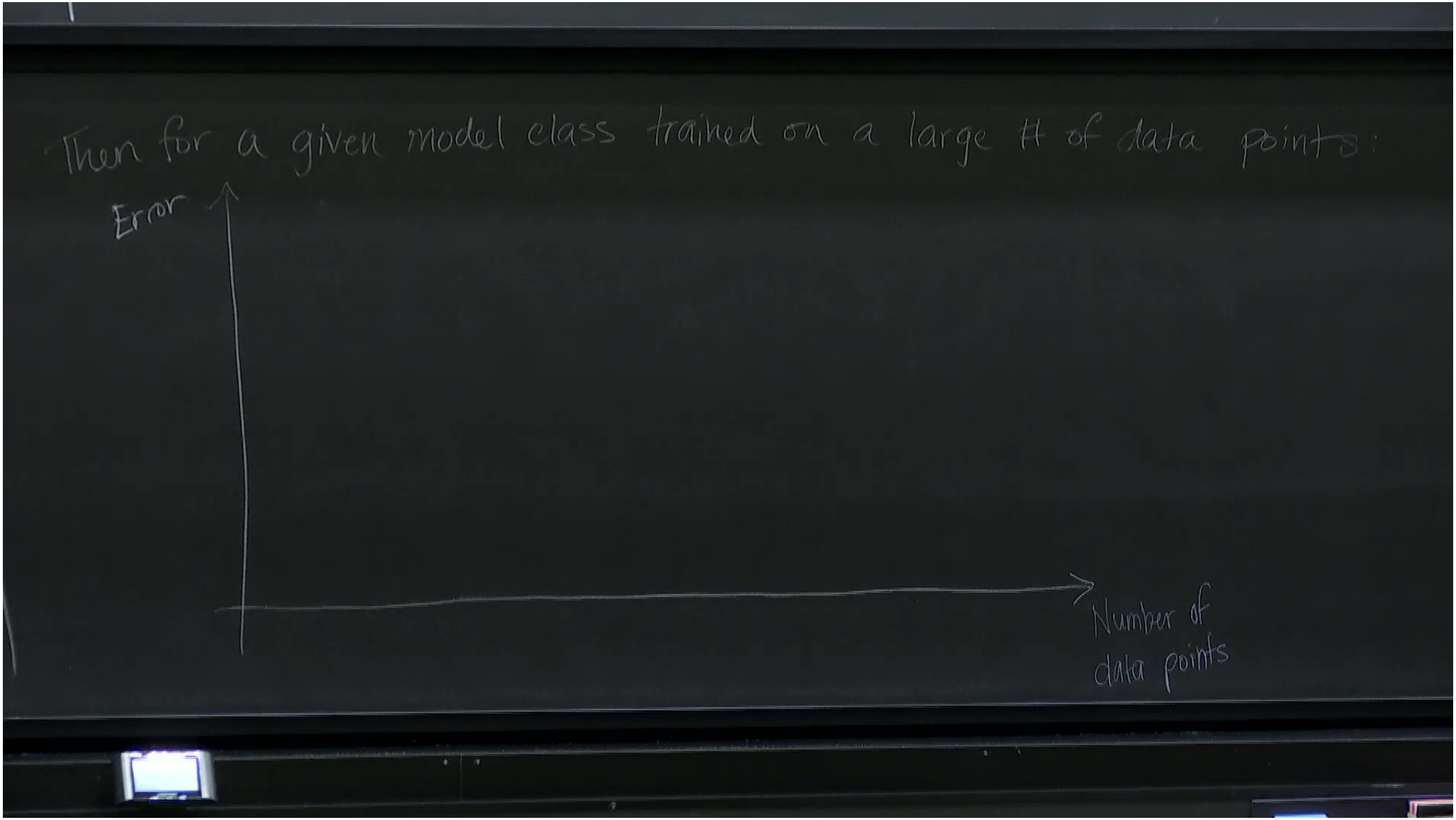

Outline for today:
• Bias-variance tradeoff
• Overfitting
↳ training, validation, and testing datasets
↳ methods that prevent overfitting for neural networks (NNs)
• Using supervised NNs to learn about phases of matter
↳ Ising model

Bias-variance tradeoff (Reference: Sec. III of arXiv: 1803.08823)

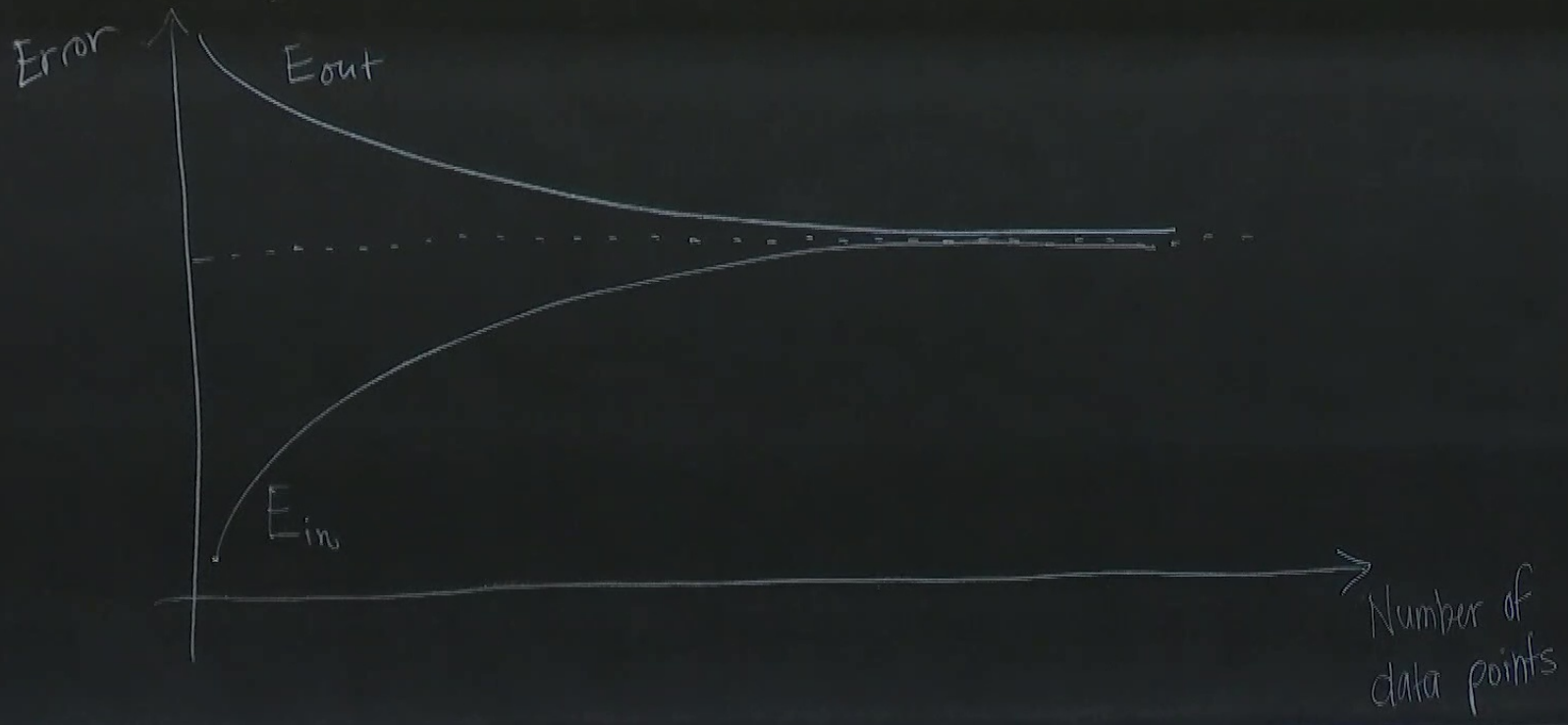
Recall the set-up of supervised learning: given a dataset of pairs $\mathcal{D} = \{(\vec{x}, \vec{y})\}$, find a model that approximates the function used to produce $\mathcal{D}$.

Let E be an error func. (eg: MSE) to measure performance


We saw last time the need for two sets of data:
• "in-sample" data used for training, with error E_{in}
• "out-of-sample" or "generalization" data to study the model's predictions, with error E_{out}
$E_{in} \leq E_{out}$
Assume the data comes from a sufficiently complicated function that we can't learn exactly

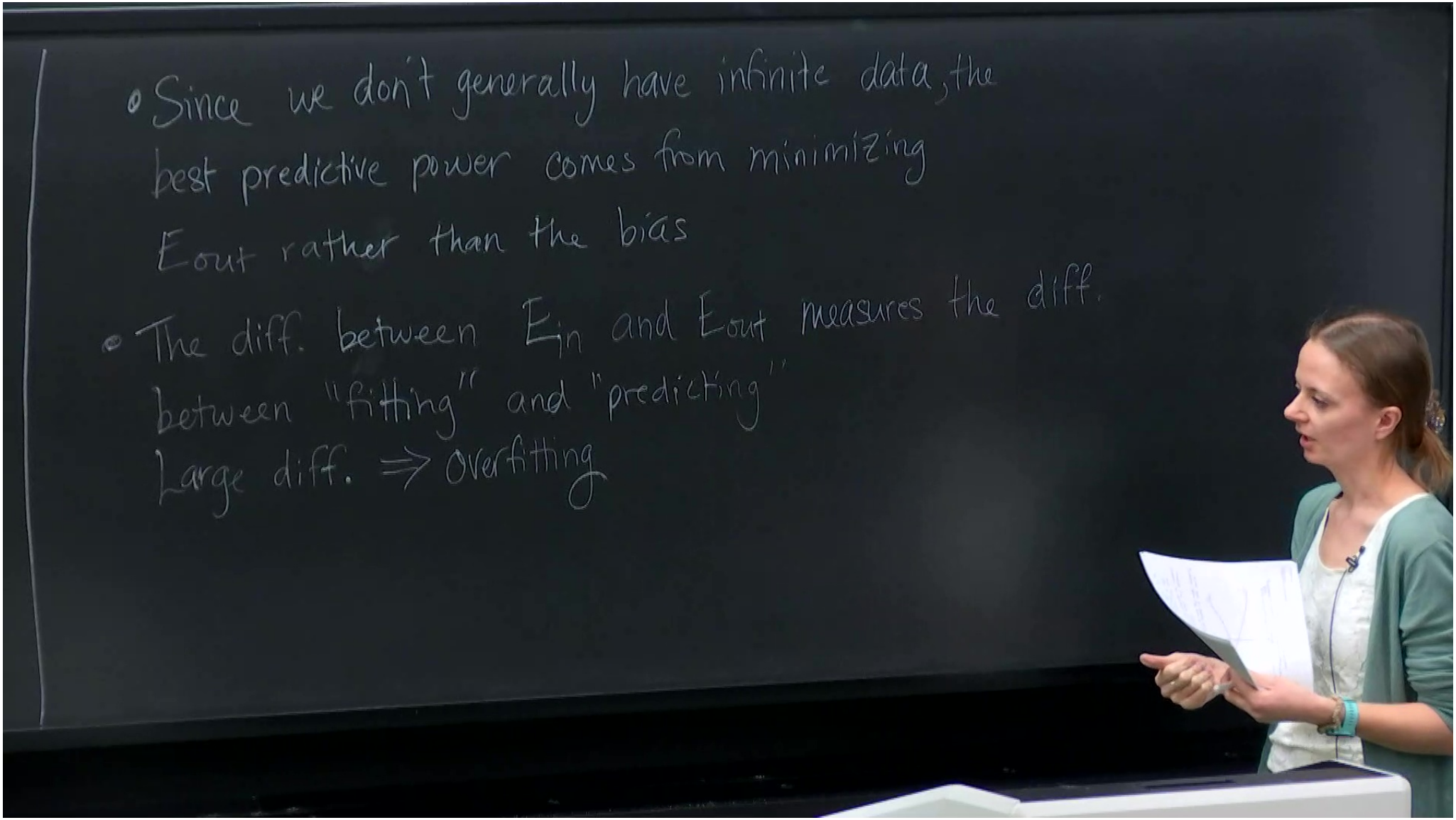

Then for a given model class trained on a large # of data points:
Error
Number of data points

Then for a given model class trained on a large # of data points:

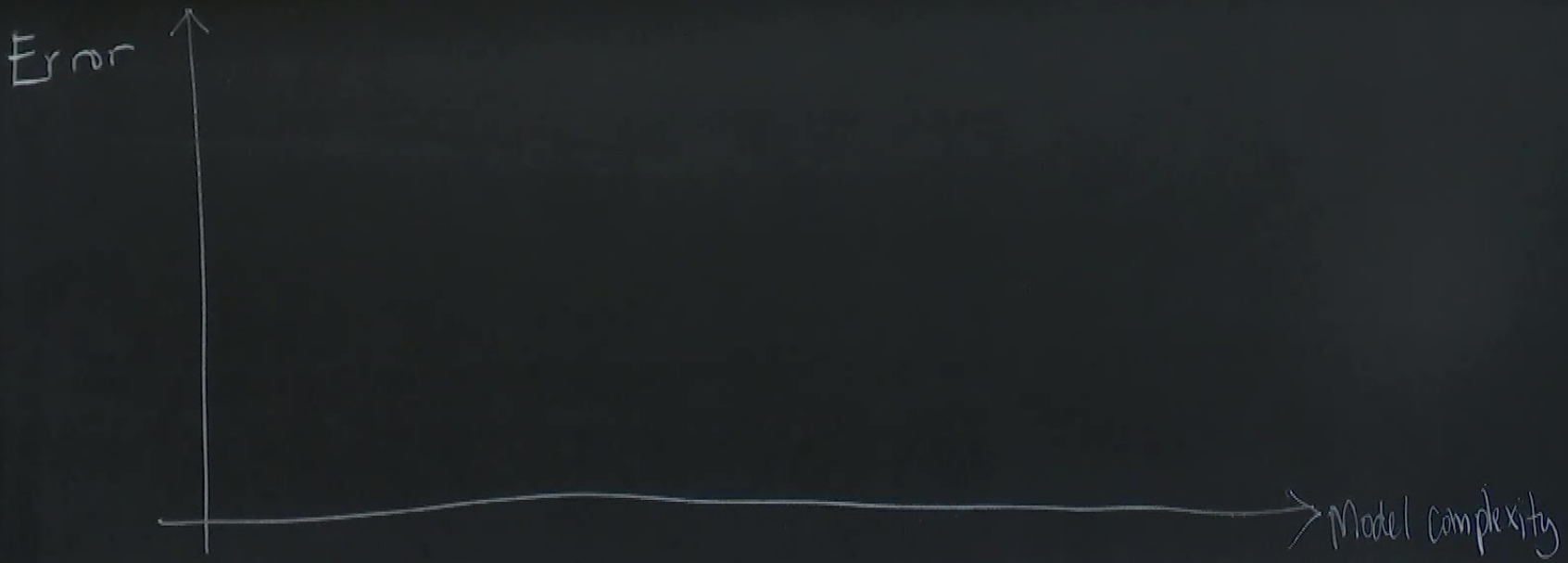


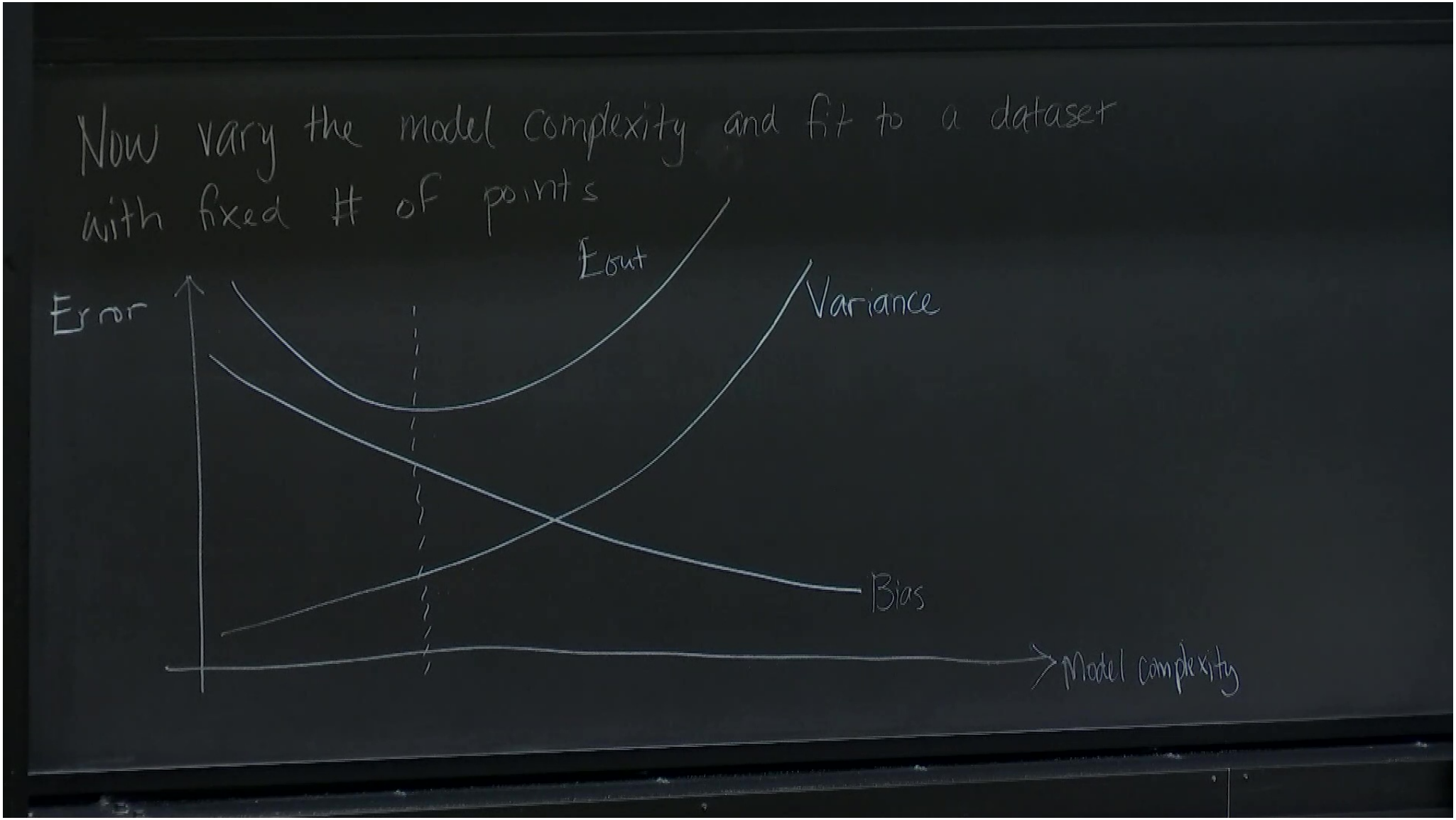
- E_{in} increases because the model is not powerful enough
- As the # of data points increases, the training set becomes more representative of the data's true dist.
- Bias:
  - ↳ a property of the model
    - ↳ how well the model does with infinite data

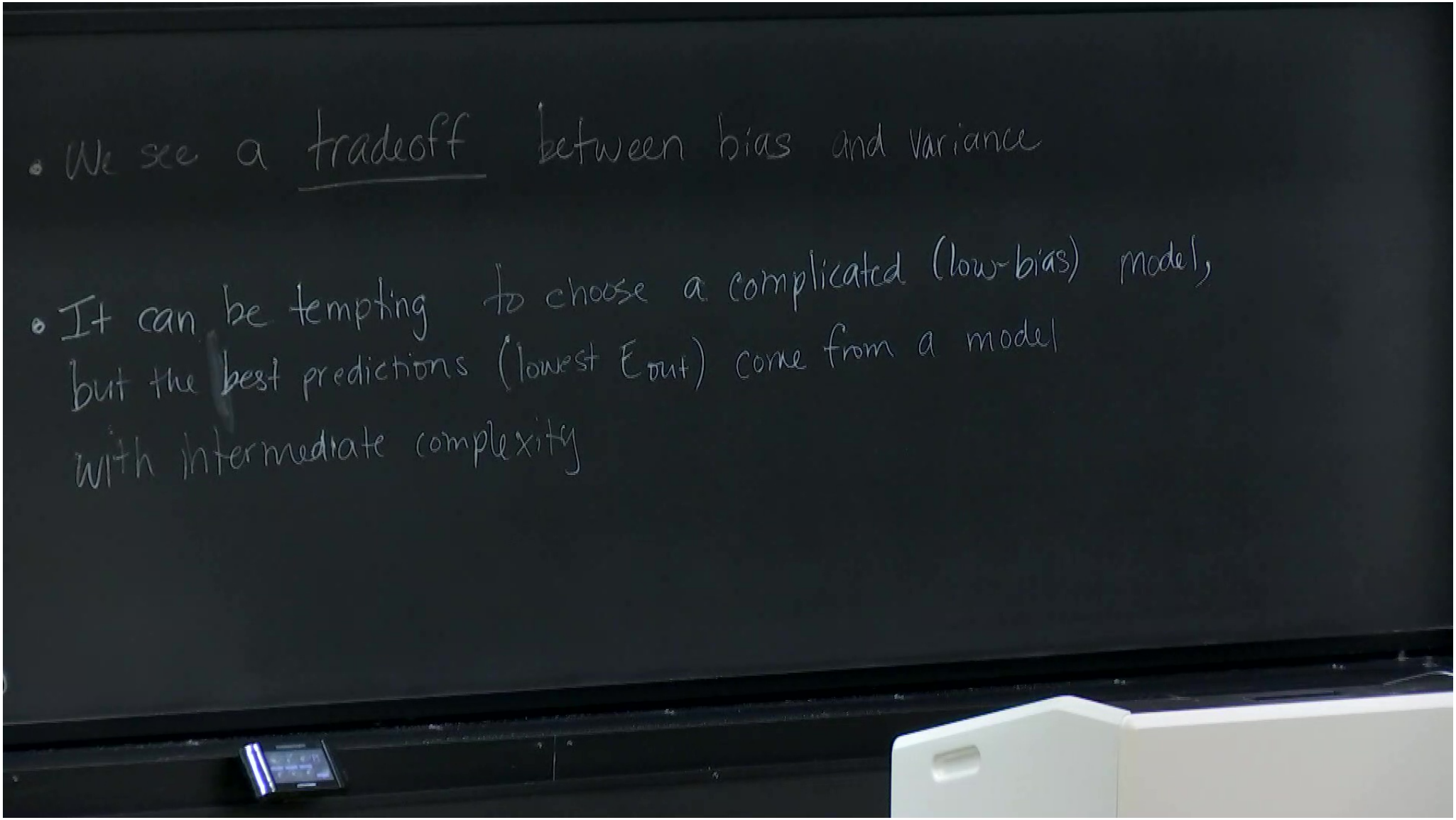
- E_{in} increases because the model is not powerful enough
- As the # of data points increases, the training set becomes more representative of the data's true dist.
- Bias:
  - ↳ a property of the model
  - ↳ how well the model does with infinite data
- Variance: measures how dependent the fit is on the particular realization of data


• Since we don't generally have infinite data, the best predictive power comes from minimizing E_{out} rather than the bias
• The diff. between E_{in} and E_{out} measures the diff. between "fitting" and "predicting"
Large diff. ⇒ Overfitting

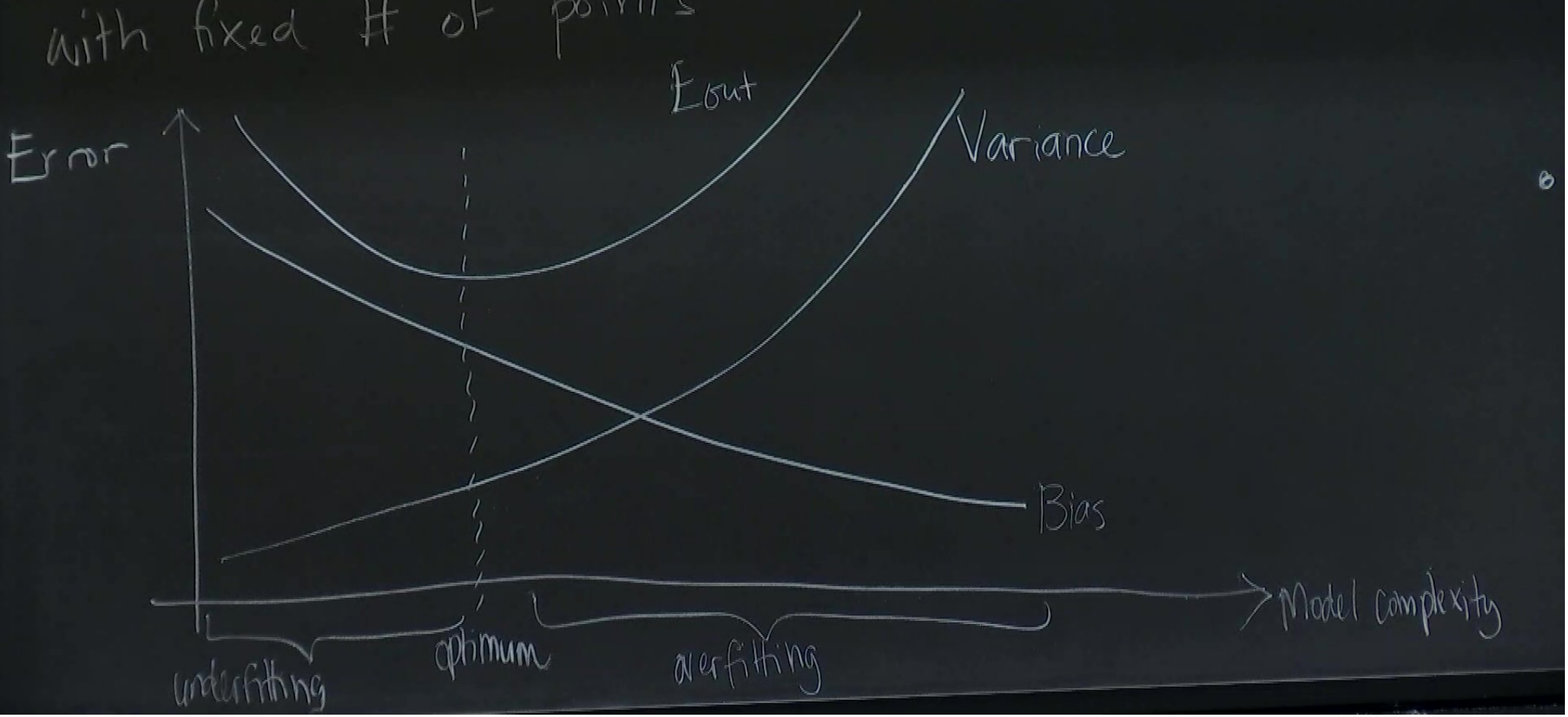
Now vary the model complexity and fit to a dataset with fixed # of points




Now vary the model complexity and fit to a dataset with fixed # of points
Error
Eout
Variance
Bias
Model complexity


• We see a tradeoff between bias and variance
• It can be tempting to choose a complicated (low-bias) model,
but the best predictions (lowest E_{out}) come from a model
with intermediate complexity

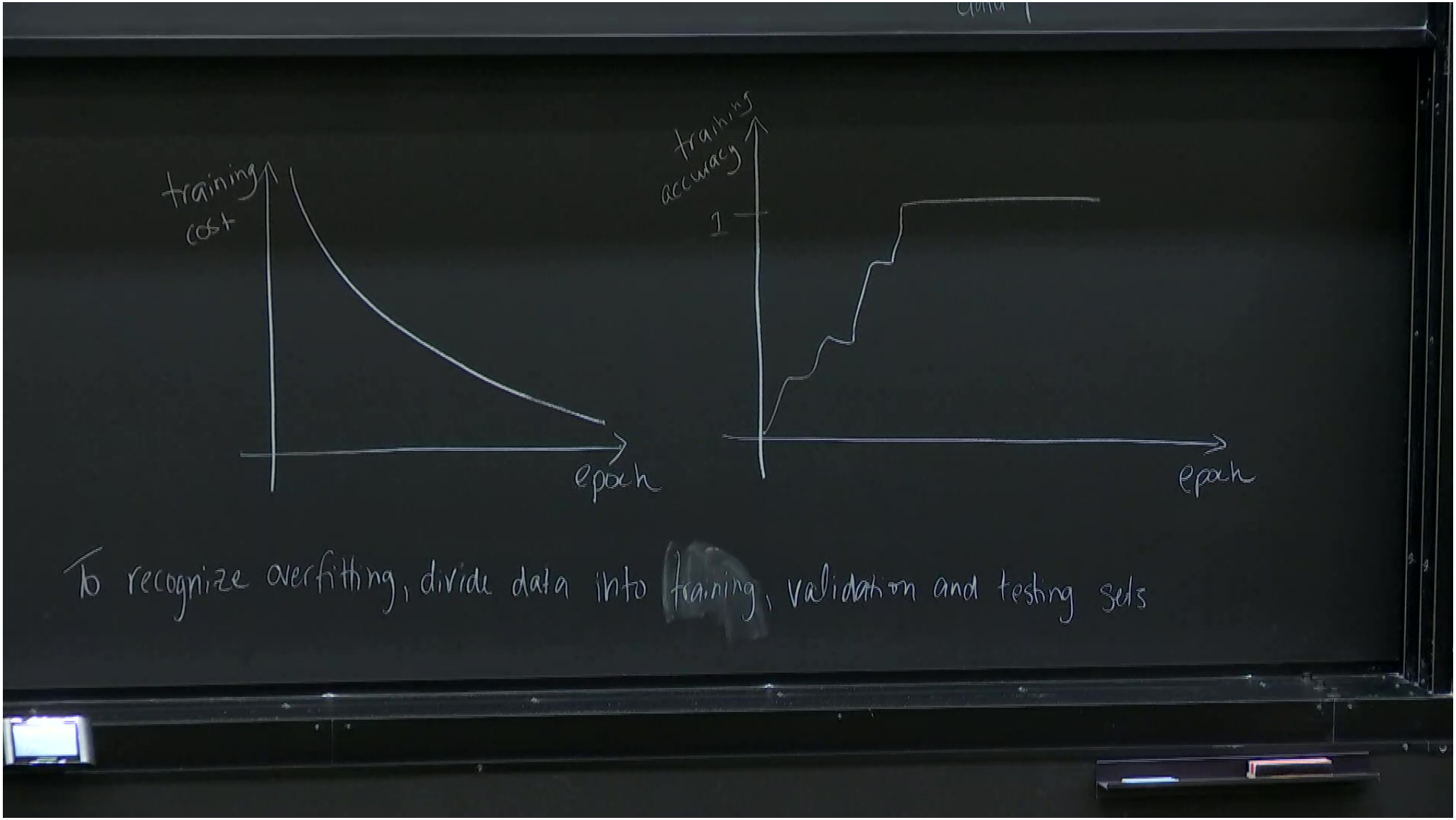
Now vary the model complexity and fit to a dataset with fixed # of points



Overfitting

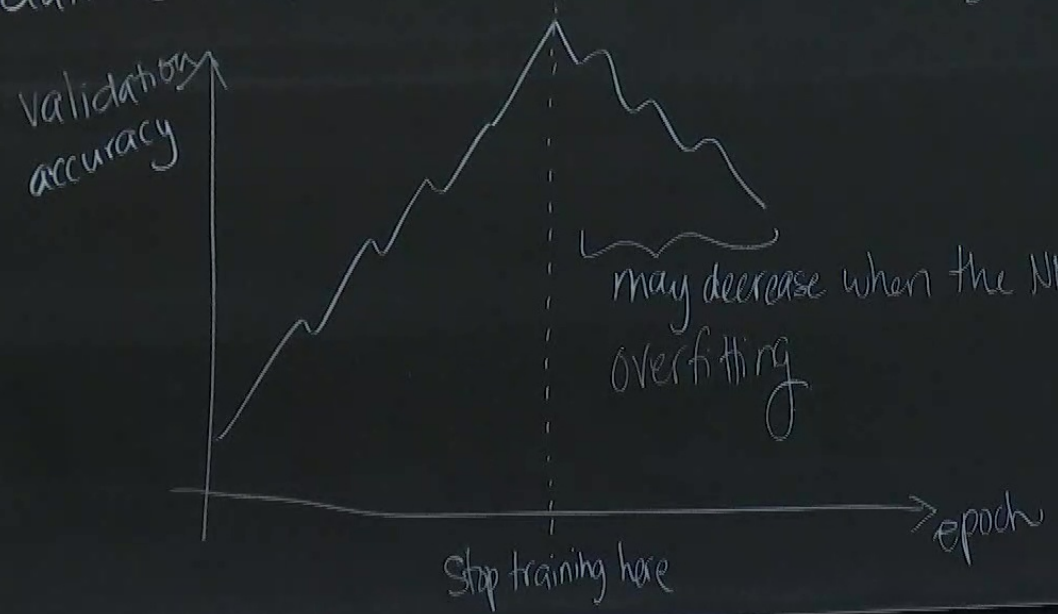
In SL, the "accuracy" refers percentage of labels the algorithms correctly predicts

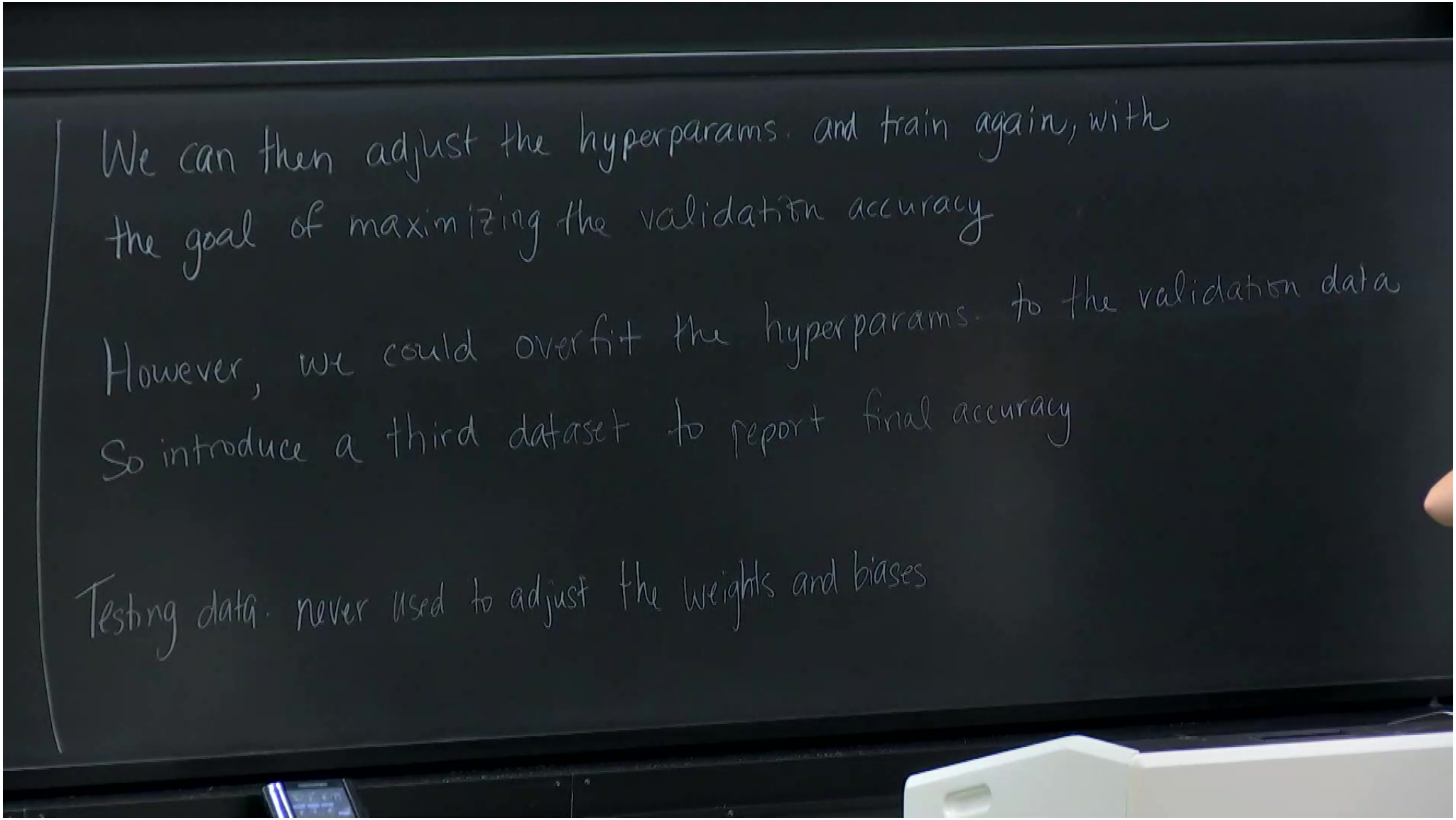
In Tutorial #2, we will see cases where the cost is decreasing and we get ~100% accuracy but we are overfitting


training cost
epoch
training accuracy
1
epoch
To recognize overfitting, divide data into training, validation and testing sets

Training data: used to adjust the weights and biases

Validation data: never used for training




We can then adjust the hyperparams. and train again, with
the goal of maximizing the validation accuracy
However, we could overfit the hyperparams. to the validation data
So introduce a third dataset to report final accuracy
Testing data: never used to adjust the weights and biases

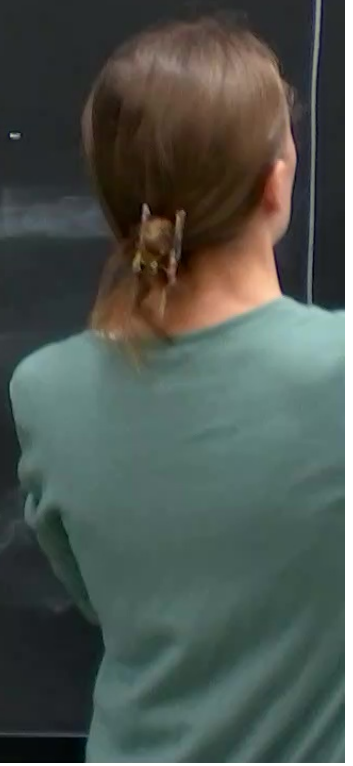
Methods that prevent overfitting for NNs

(1) L1 or L2 regularization

Idea: add an extra term to the cost func. that depends on the weights

$$C_{L1} = C_{init} + \lambda \sum_{weights} \left| W_{ij}^{(\ell)} \right|$$

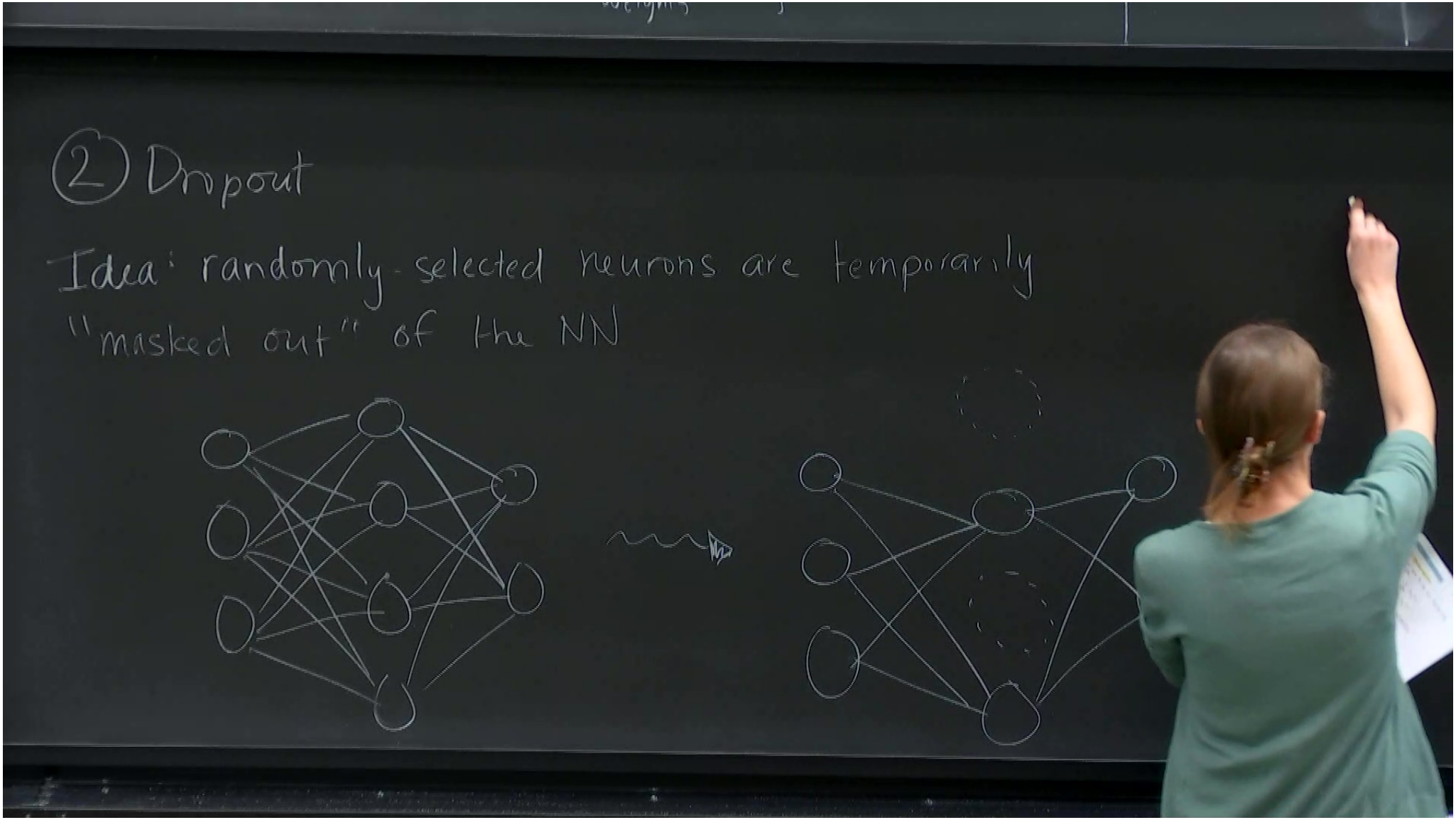
$$C_{L2} = C_{init} + \lambda \sum_{weights} \left| W_{ij}^{(\ell)} \right|^2$$



$\lambda > 0$ is the regularization hyperparam.

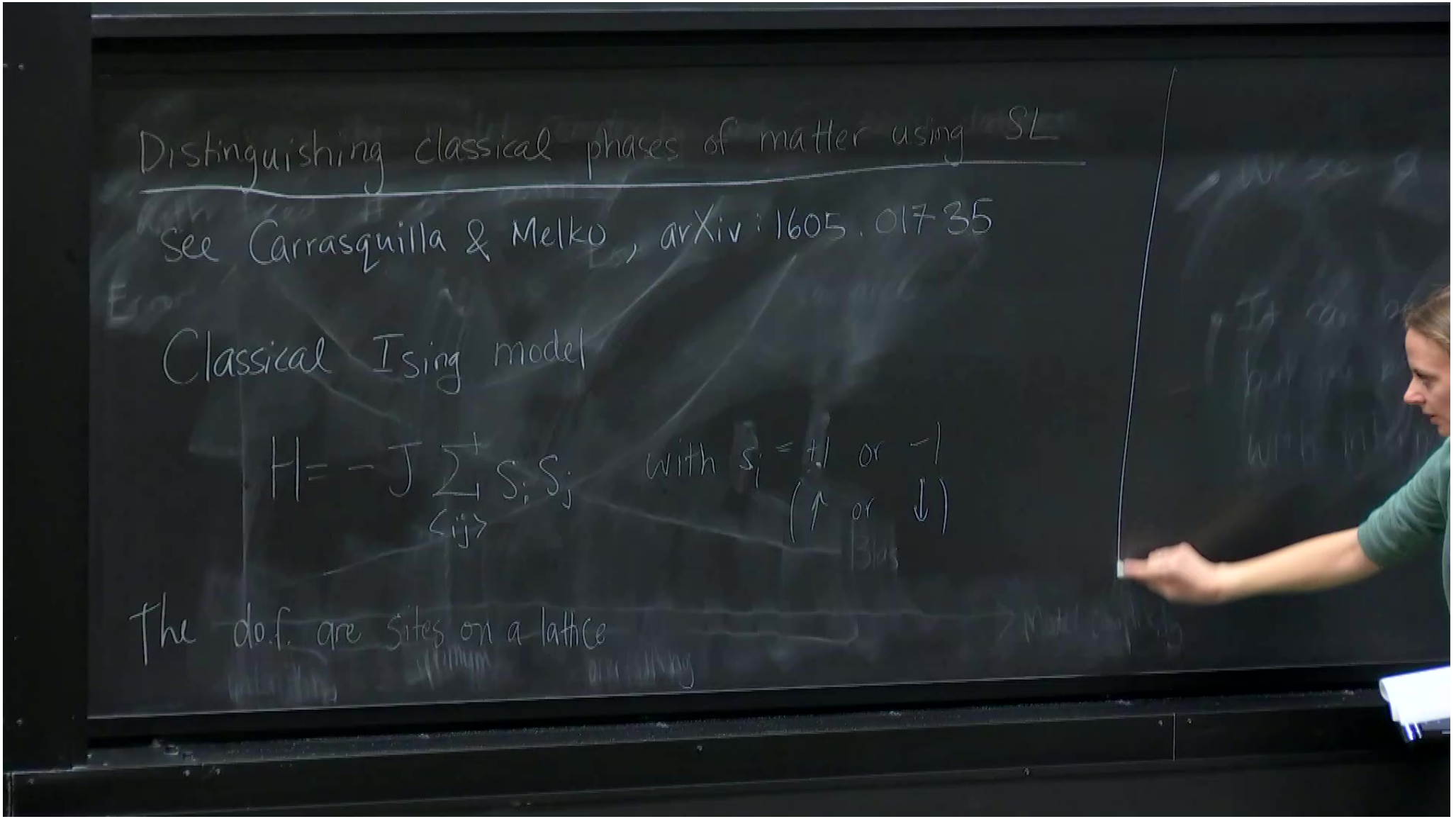
Effect: NN favours smaller weights

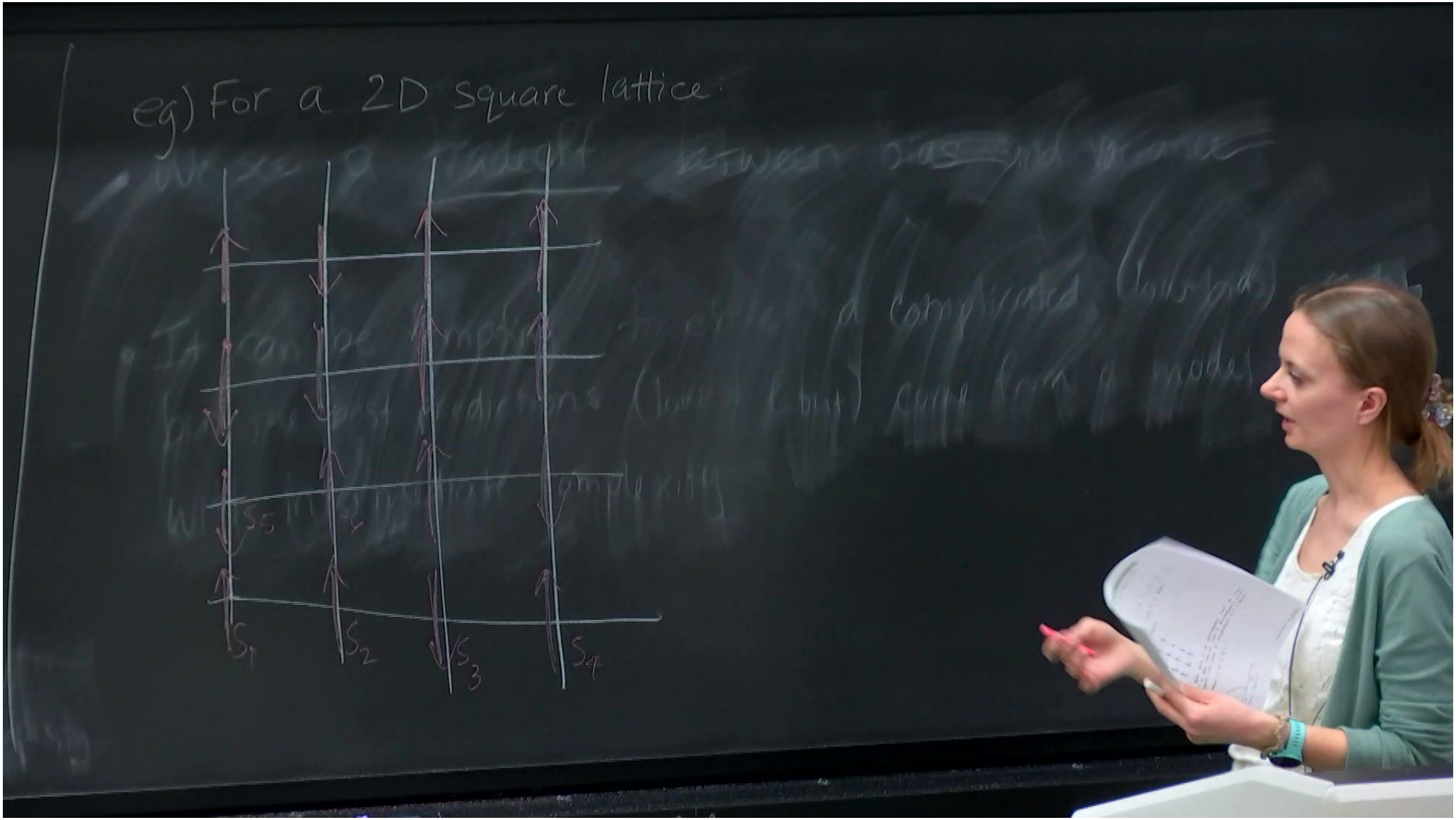
- λ too small $\longrightarrow$ overfitting
- λ too large $\longrightarrow$ underfitting


② Dropout
Idea: randomly selected neurons are temporarily "masked out" of the NN

Run training step(s) for a randomly-chosen masking.
Then choose a new masking and repeat

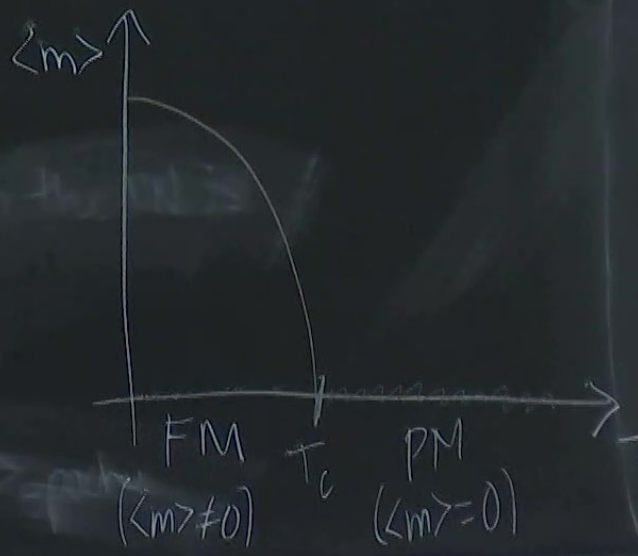
Effect: similar to averaging over many different networks


Distinguishing classical phases of matter using SL
see Carrasquilla & Melko, arXiv: 1605.01735
Classical Ising model
$H = -J \sum_{\langle ij \rangle} S_i S_j$ with $S_i = +1$ or -1
($\uparrow$ or $\downarrow$)
The d.o.f. are sites on a lattice


eg) For a 2D square lattice:
S_1
S_2
S_3
S_4

For a 2D square lattice in the thermodynamic limit, there is a PT from a ferromagnetic (FM) to a paramagnetic phase at temperature $T_c \approx 2.269J$

The order param. is $m = \frac{1}{N}\sum_i S_i$

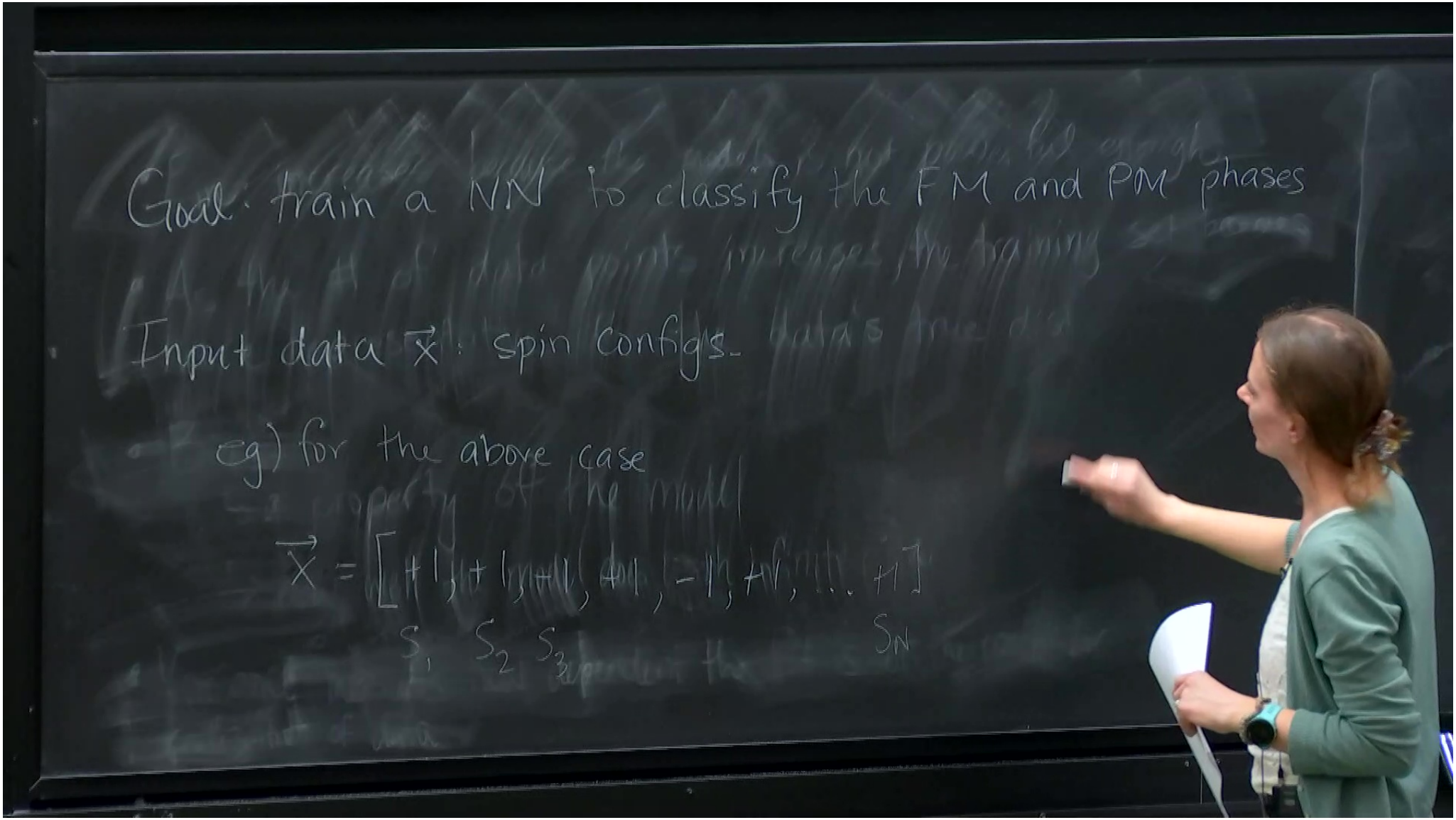


Goal: train a NN to classify the FM and PM phases

Input data $\vec{x}$: spin configs.

eg) for the above case

$\vec{x}$


Goal: train a NN to classify the FM and PM phases
Input data $\vec{x}$: spin configs.
eg) for the above case
$\vec{x} = [+1, +1, \ldots, -1, \ldots, +1]$
S_1 S_2 S_3 S_N

Goal: train a NN to classify the FM and PM phases

Input data $\vec{x}$: spin configs.

eg) for the above case

$\vec{x} = [+1, +1, +1, +1, -1, +1, \ldots, +1]$

$S_1\ S_2\ S_3 \qquad S_N$

Labels: 0 or 1

↑ FM ↑ PM

See Homework #1