Title: Machine Learning (2021/2022)

Speakers: Lauren Hayward

Collection: Machine Learning (2021/2022)

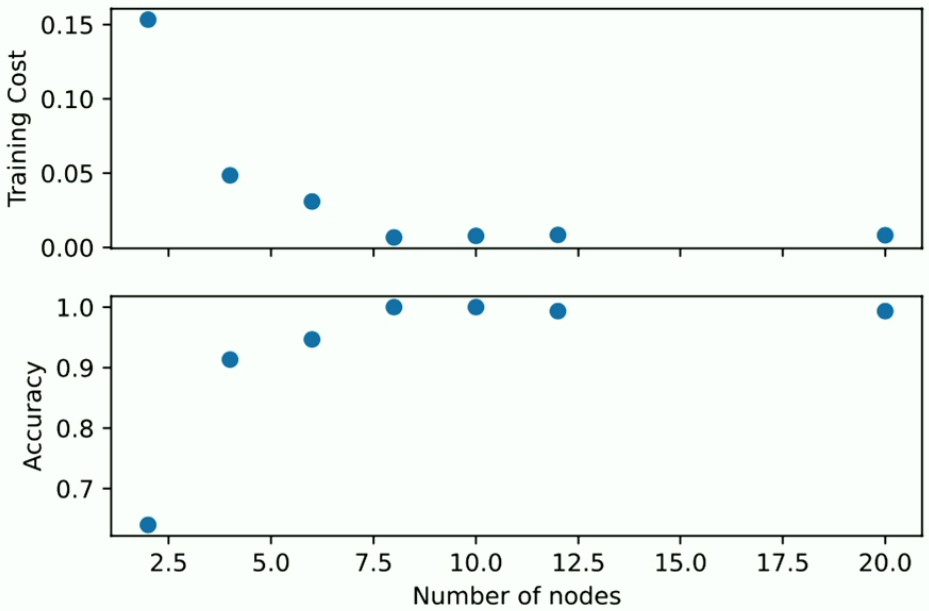
Date: April 14, 2022 - 11:30 AM

URL: https://pirsa.org/22040069

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.

Tutorial 2, Problem 2

Accuracy vs. the number of neurons in the hidden layer

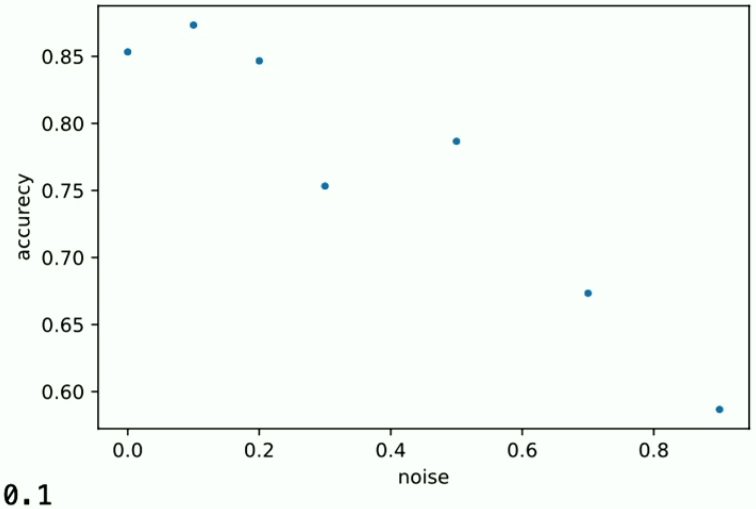


- sigmoid and softmax activation functions
- learning rate 1.5
- SGD optimizer
- MSE cost function
- 20000 epochs

PERIMETER SCHOLARS INTERNATIONAL

Tutorial 2, Problem 2

Accuracy vs. the magnitude of noise in the data

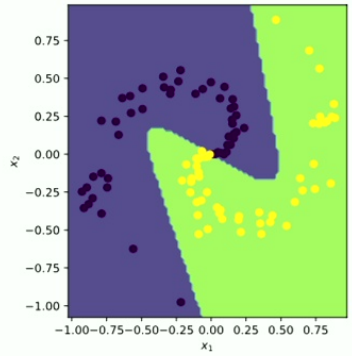


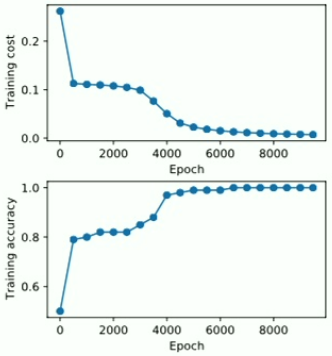
- learning_rate = 0.1
- cost_func = torch.nn.MSELoss()
- activation_function = sigmoid
- number_of_layer and neurons: 1, 4
- N_epochs = 100000

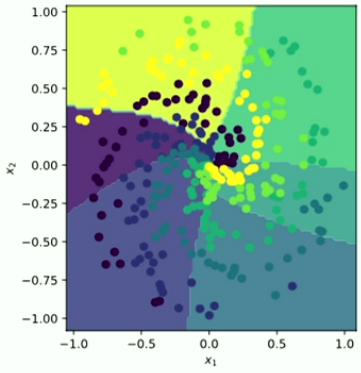
PERIMETER SCHOLARS INTERNATIONAL

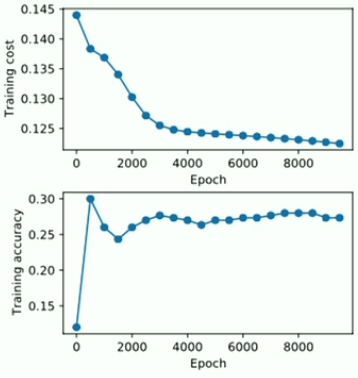
Tutorial 2, Problem 2

Varying the number of different labels (branches)





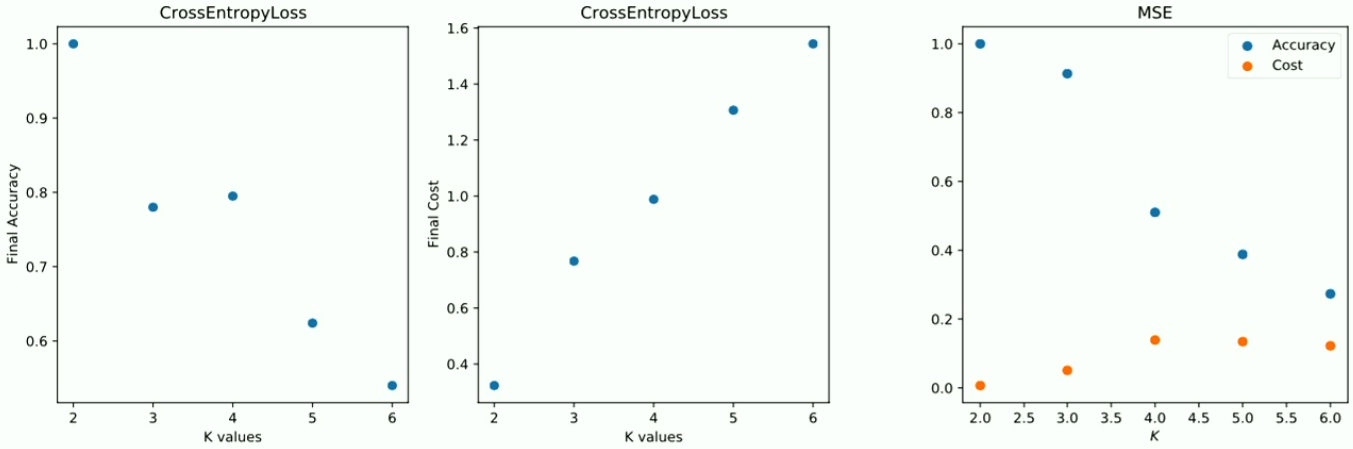




PERIMETER SCHOLARS INTERNATIONAL

Tutorial 2, Problem 2

Accuracy vs. the number of different labels (branches)

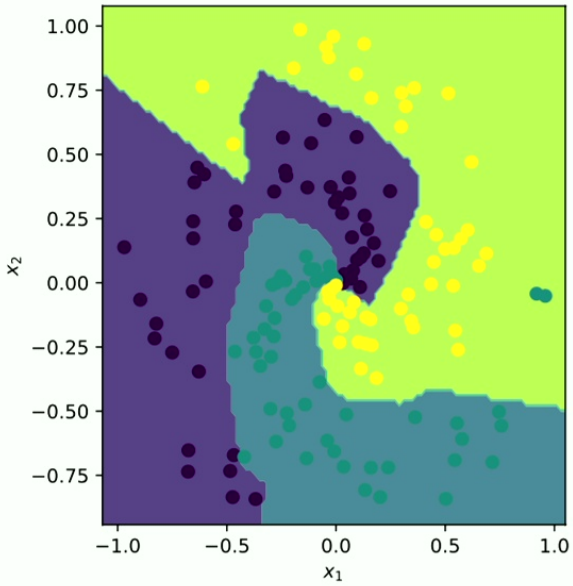


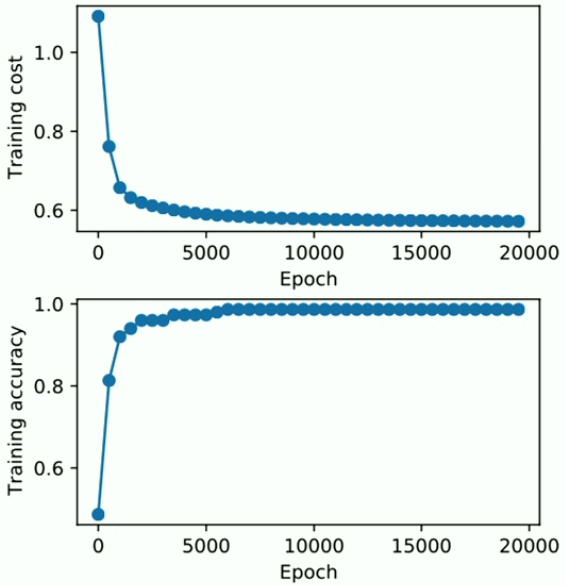
- learning rate = 1
- 4 neurons in the hidden layer
- first layer uses sigmoid
- outer layer uses softmax

PERIMETER SCHOLARS INTERNATIONAL

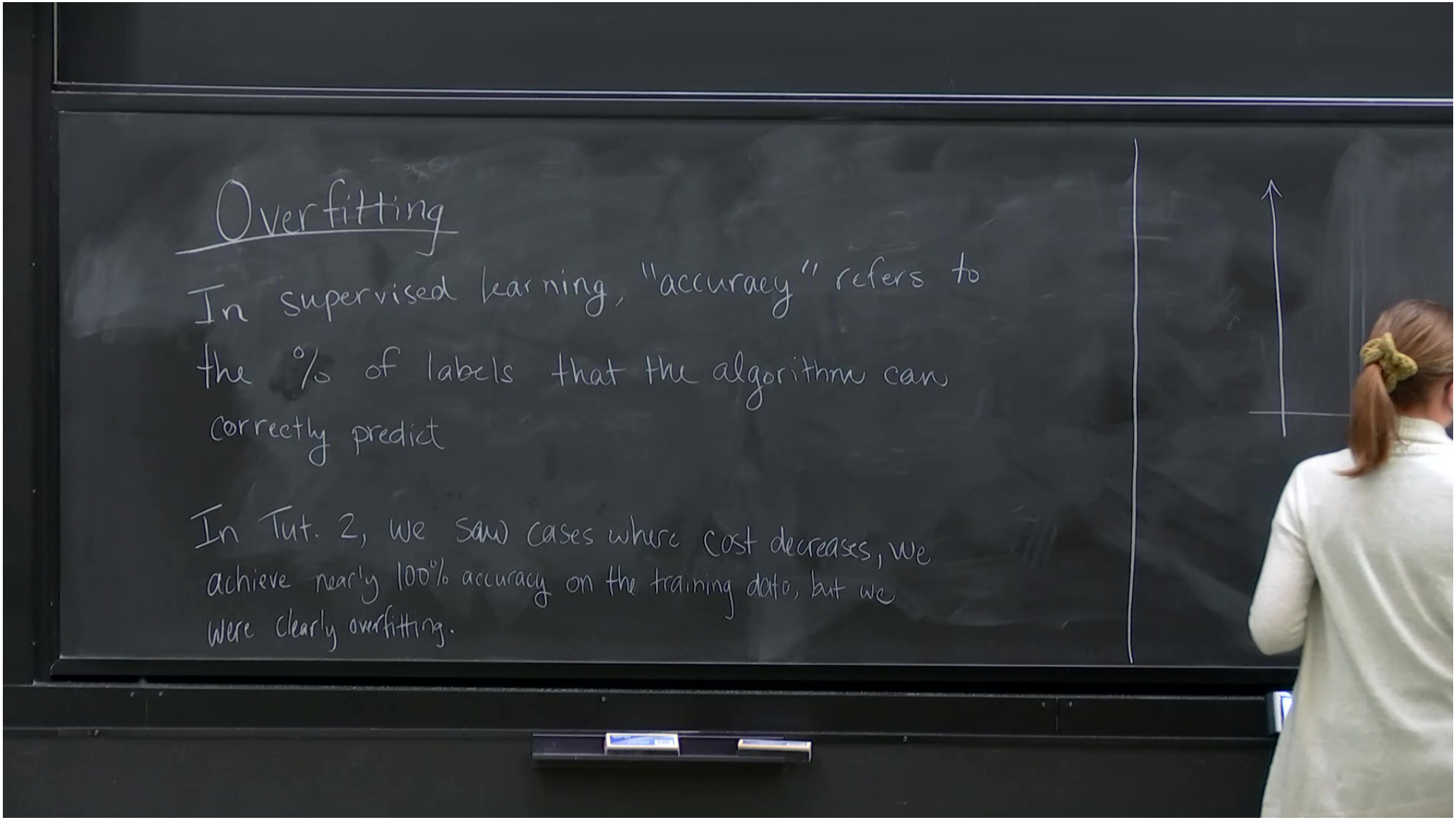
Overfitting

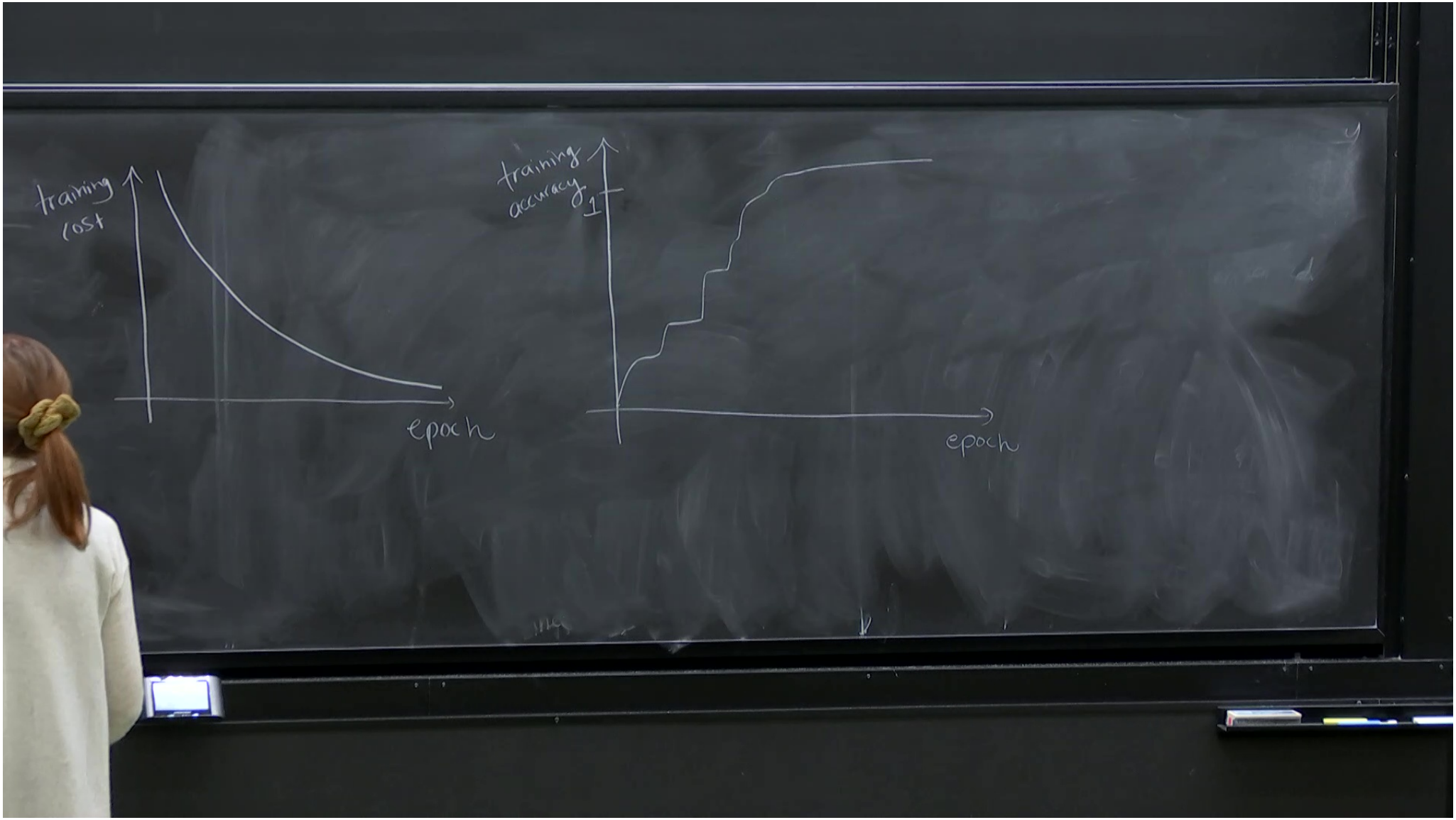
```
mag_noise   = 0.5
hidden_size = 100
```

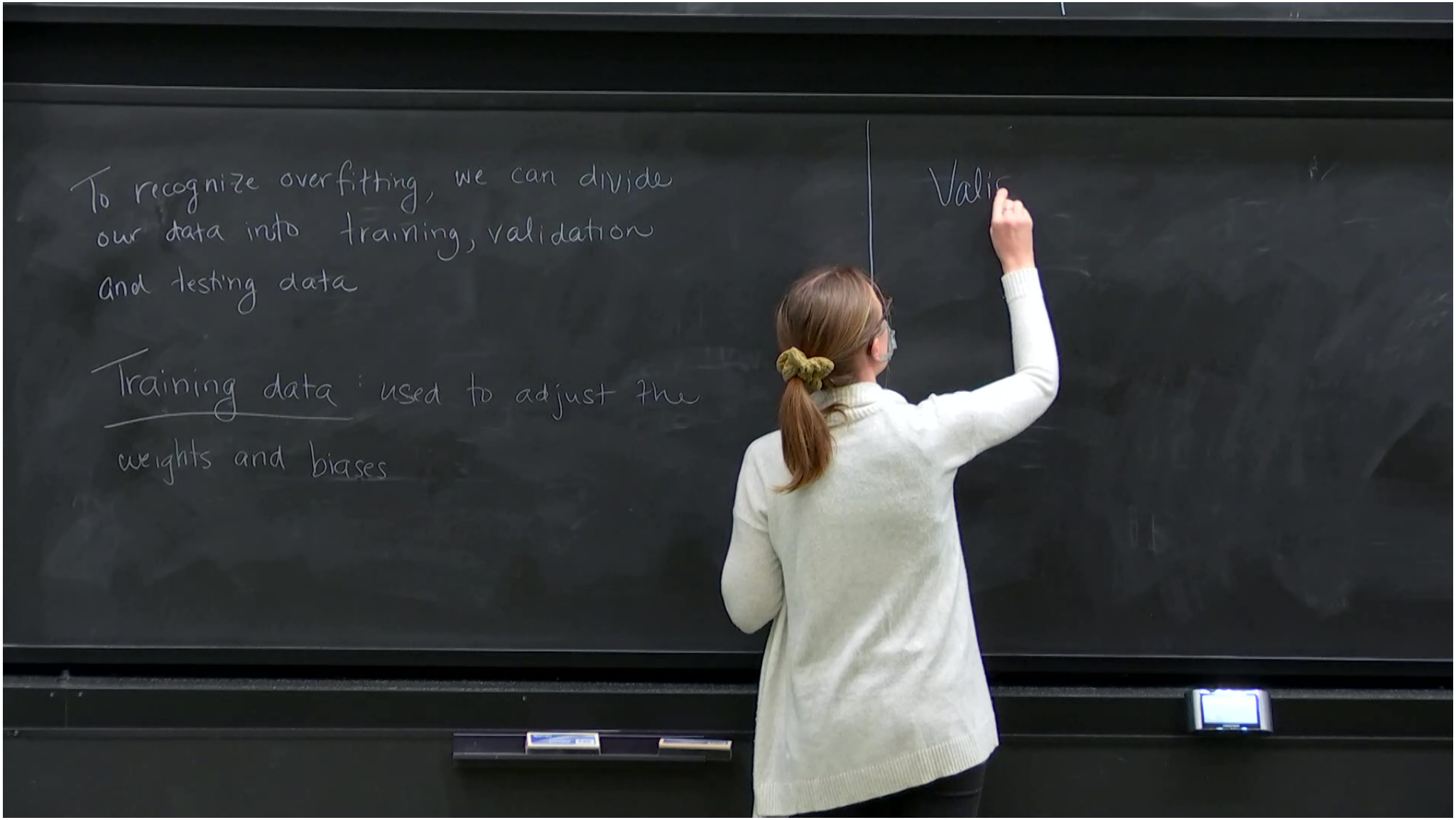


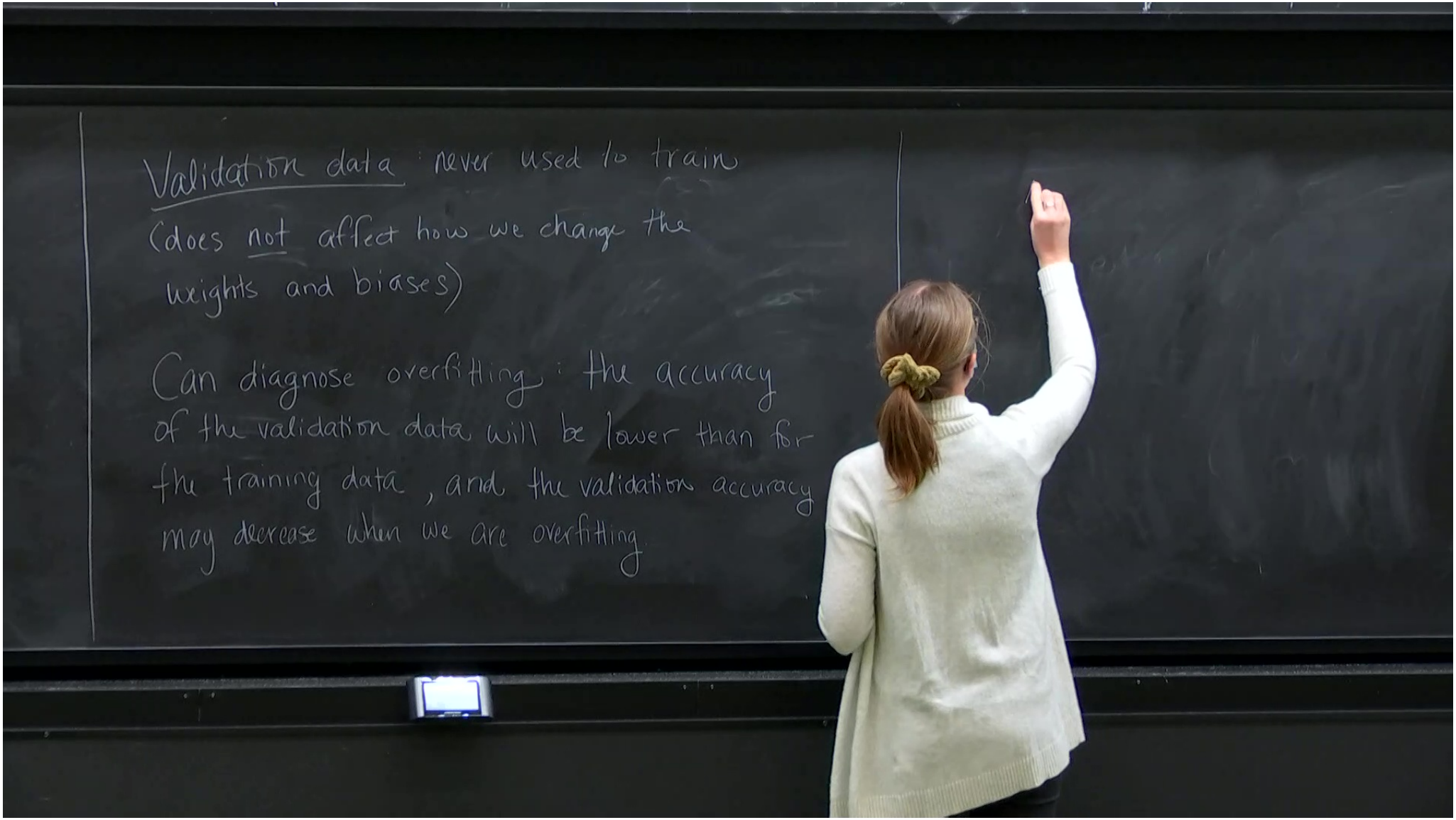


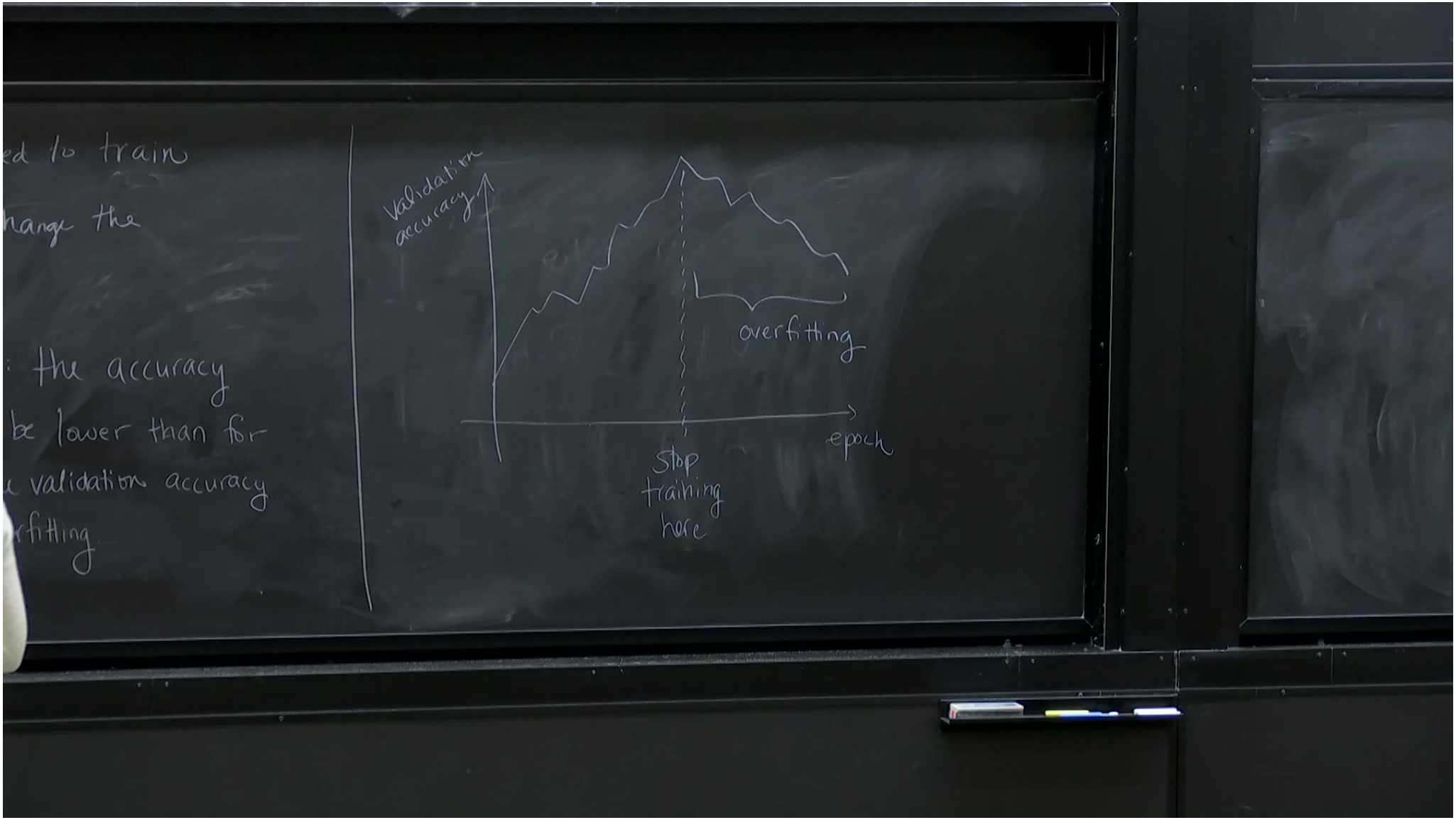
PERIMETER SCHOLARS INTERNATIONAL


Overfitting
In supervised learning, "accuracy" refers to the % of labels that the algorithm can correctly predict
In Tut. 2, we saw cases where cost decreases, we achieve nearly 100% accuracy on the training data, but we were clearly overfitting.


training cost
epoch
training accuracy
1
epoch


To recognize overfitting, we can divide our data into training, validation and testing data
Training data: used to adjust the weights and biases
Vali

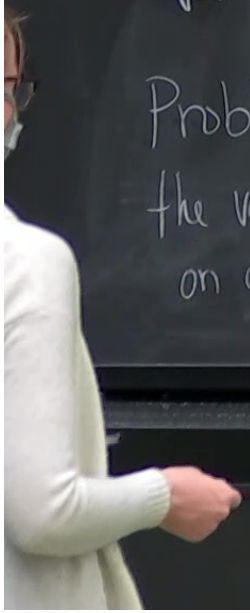

Validation data: never used to train
(does not affect how we change the
weights and biases)
Can diagnose overfitting: the accuracy
of the validation data will be lower than for
the training data, and the validation accuracy
may decrease when we are overfitting.

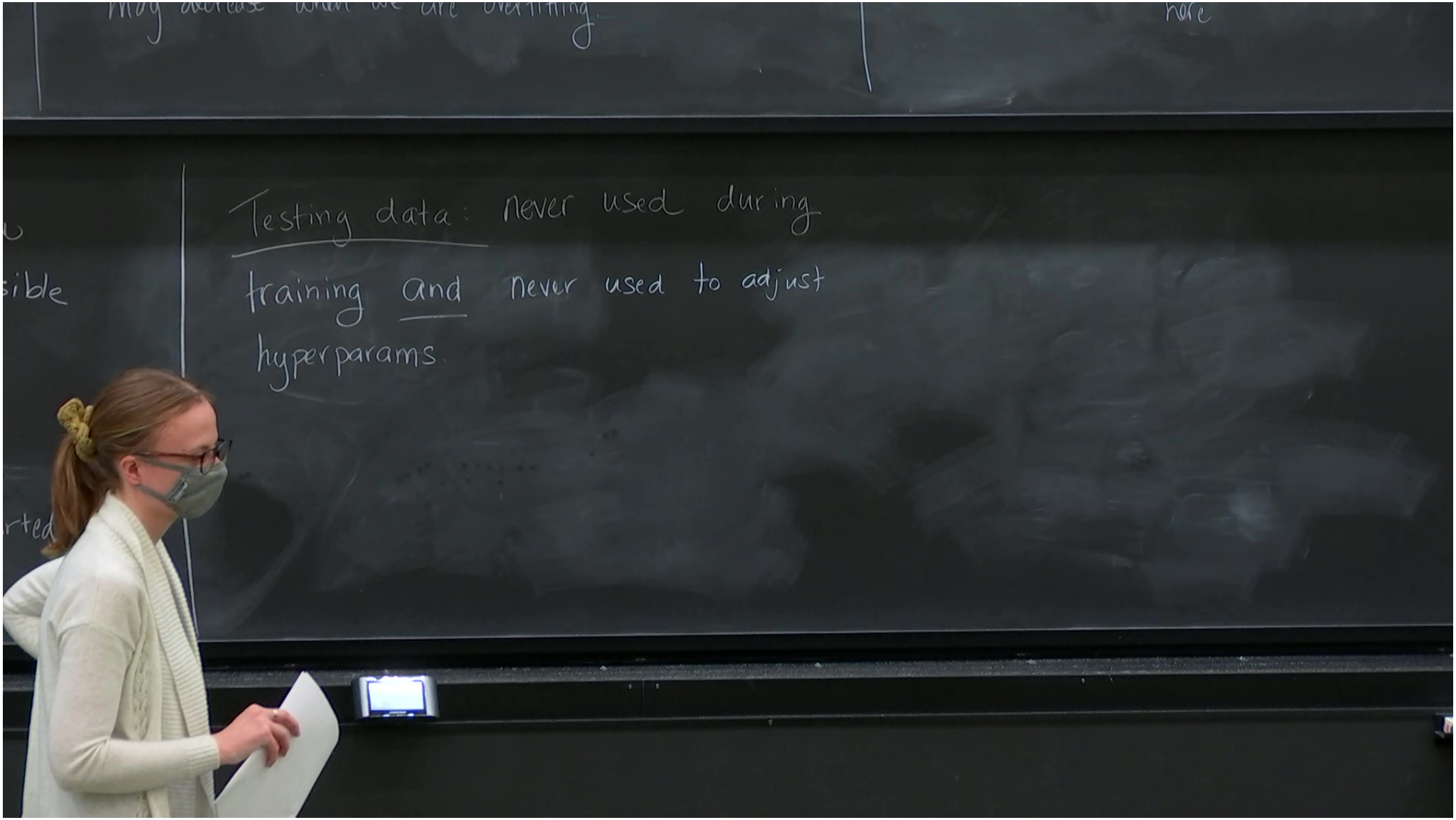

ed to train
hange the
: the accuracy
be lower than for
validation accuracy
rfitting
Validation accuracy
overfitting
epoch
Stop training here

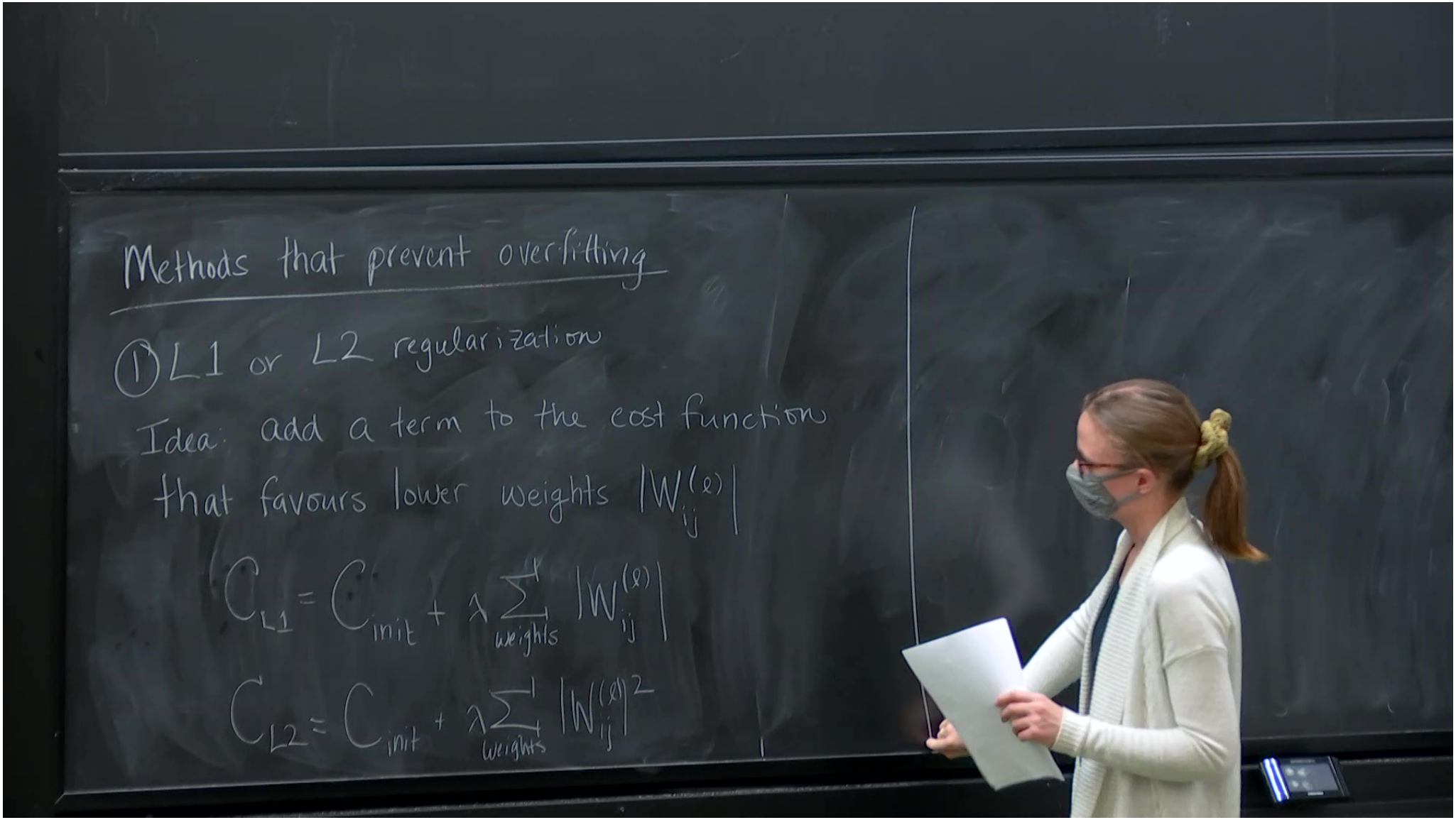
may decrease when we are overfitting

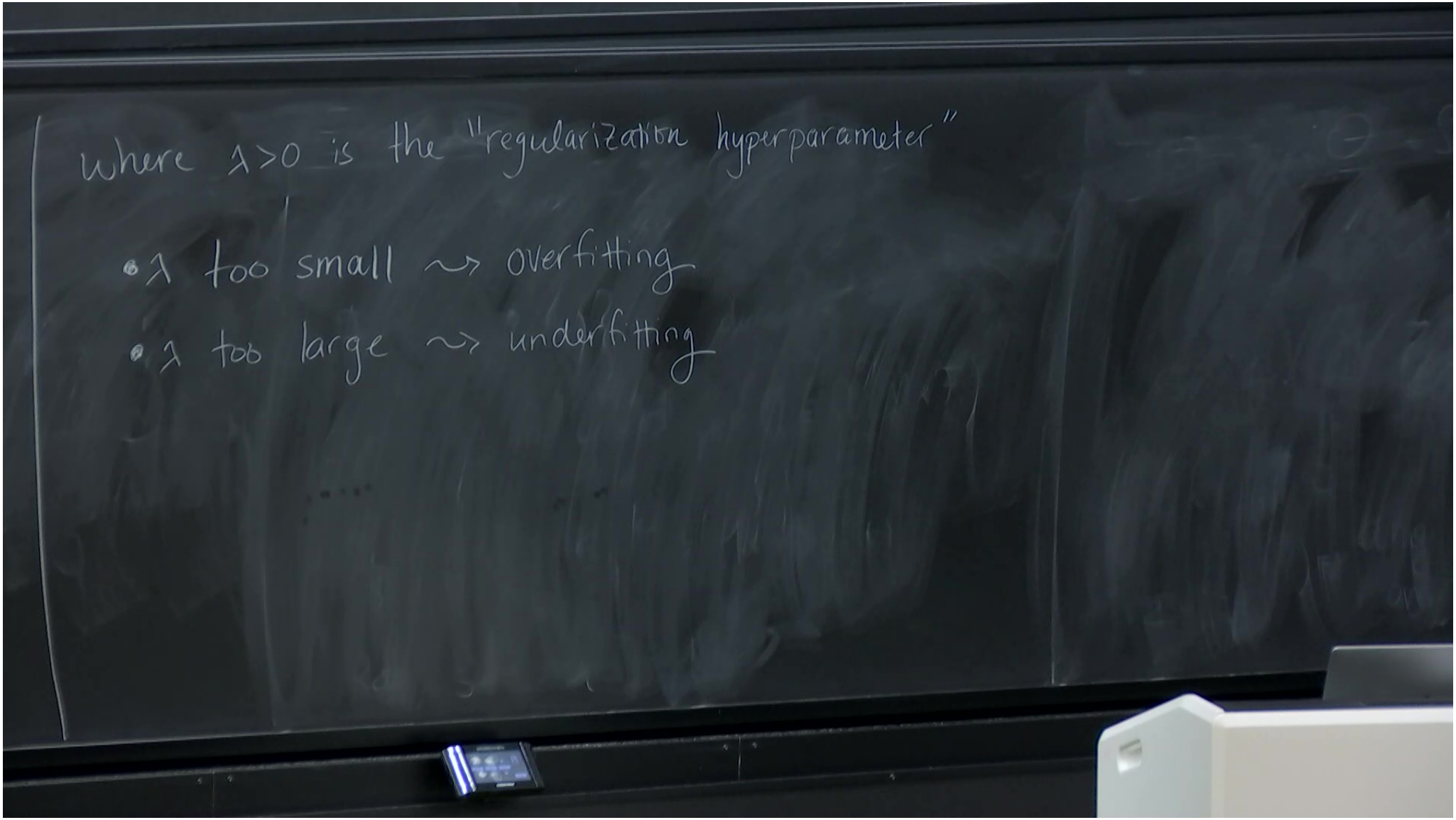
Could then adjust the hyperparams. and train again, with goal of achieving the highest possible validation accuracy.

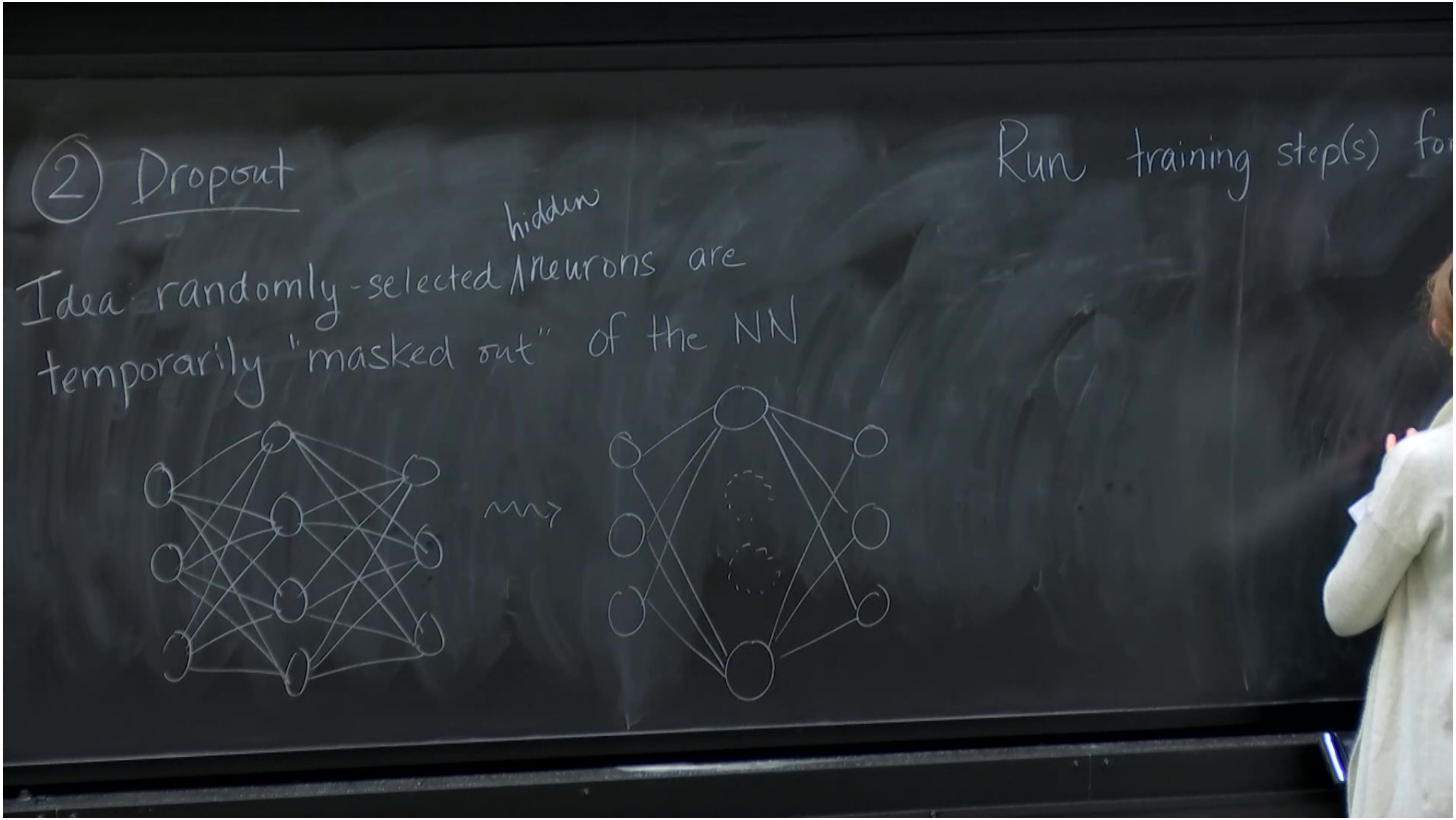
Problem: we could overfit the hyperparams. to the validation data. The final accuracy should be reported on a third set of data: testing data

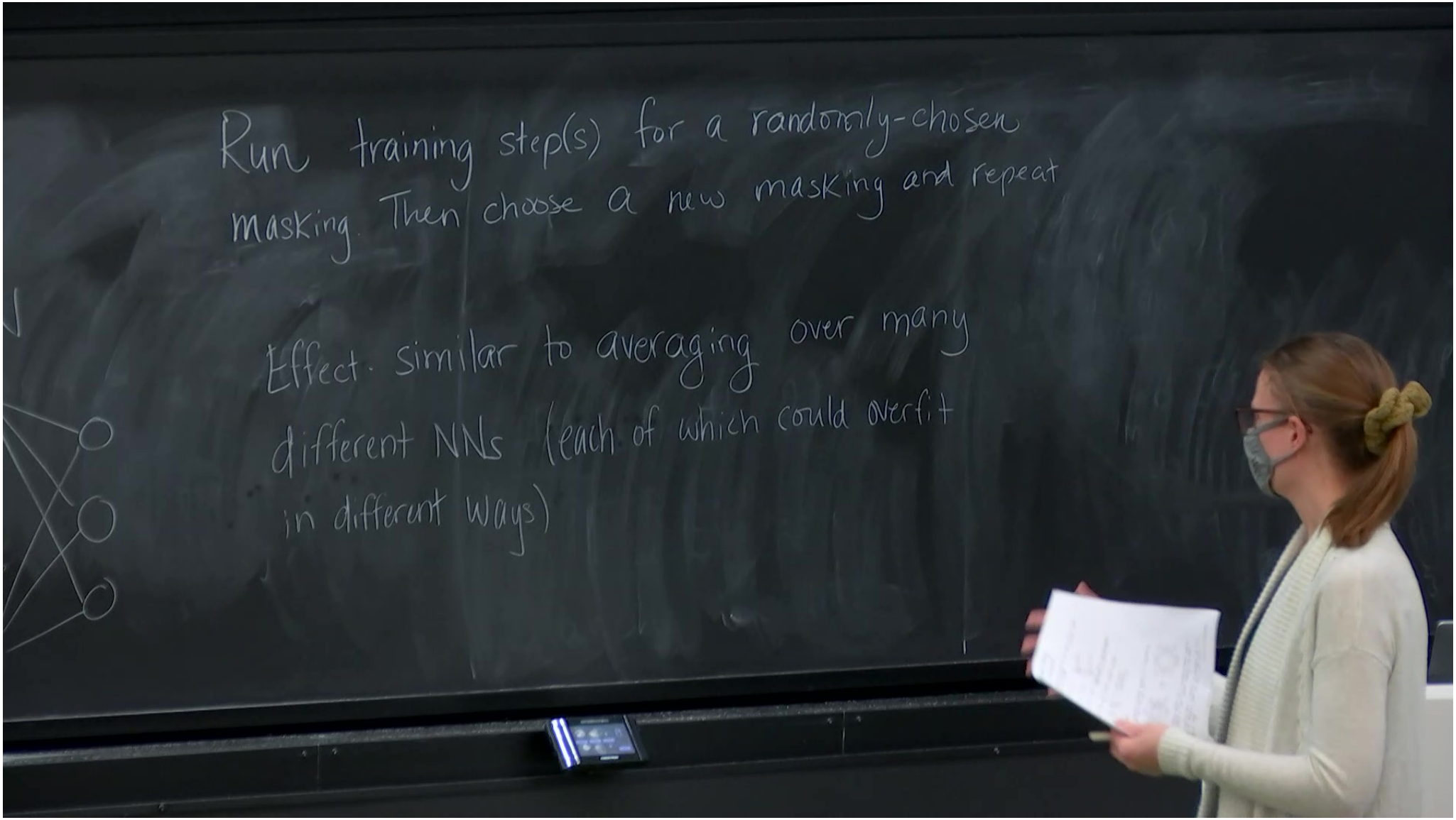


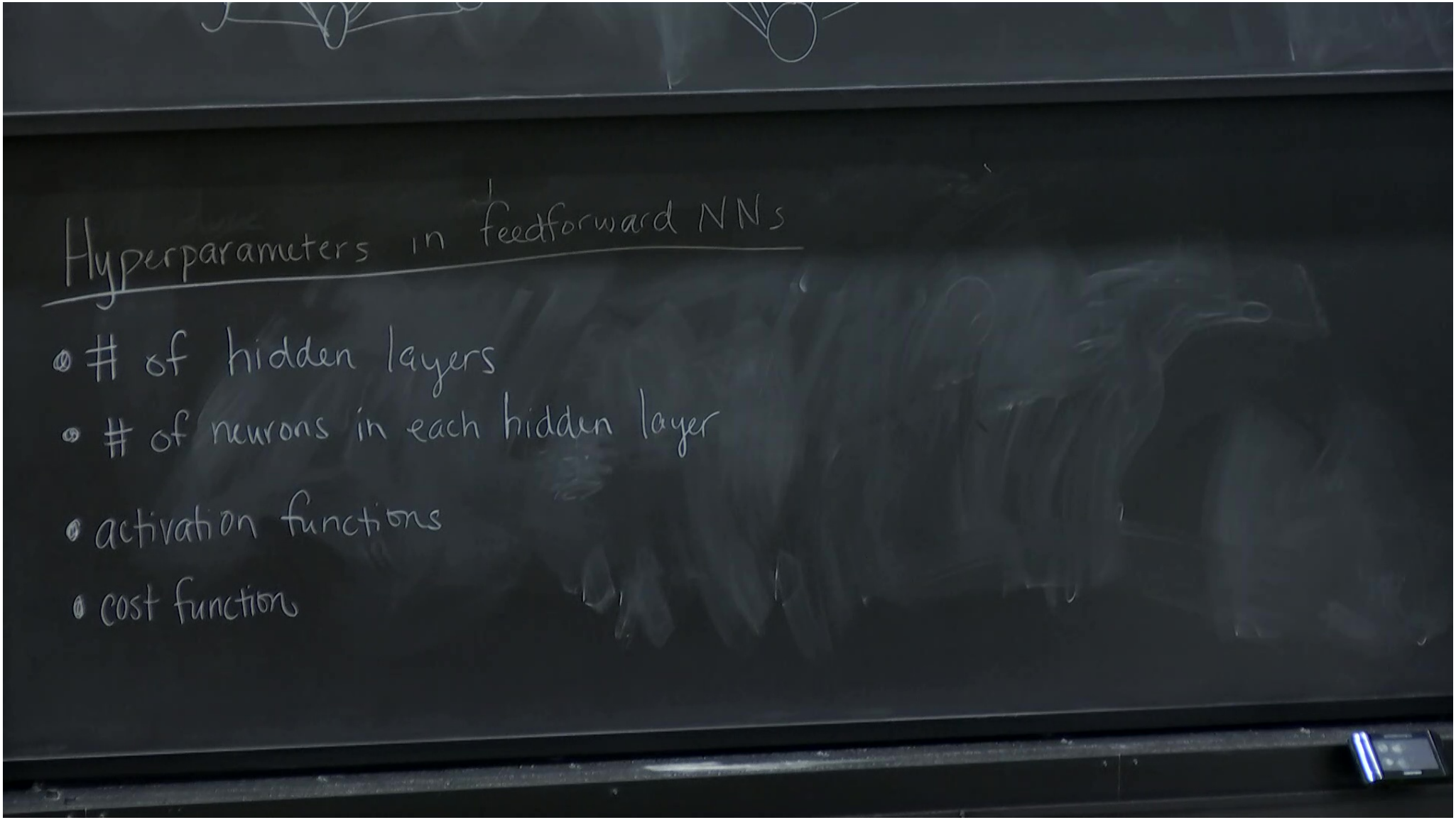

Testing data: never used during
training and never used to adjust
hyperparams.

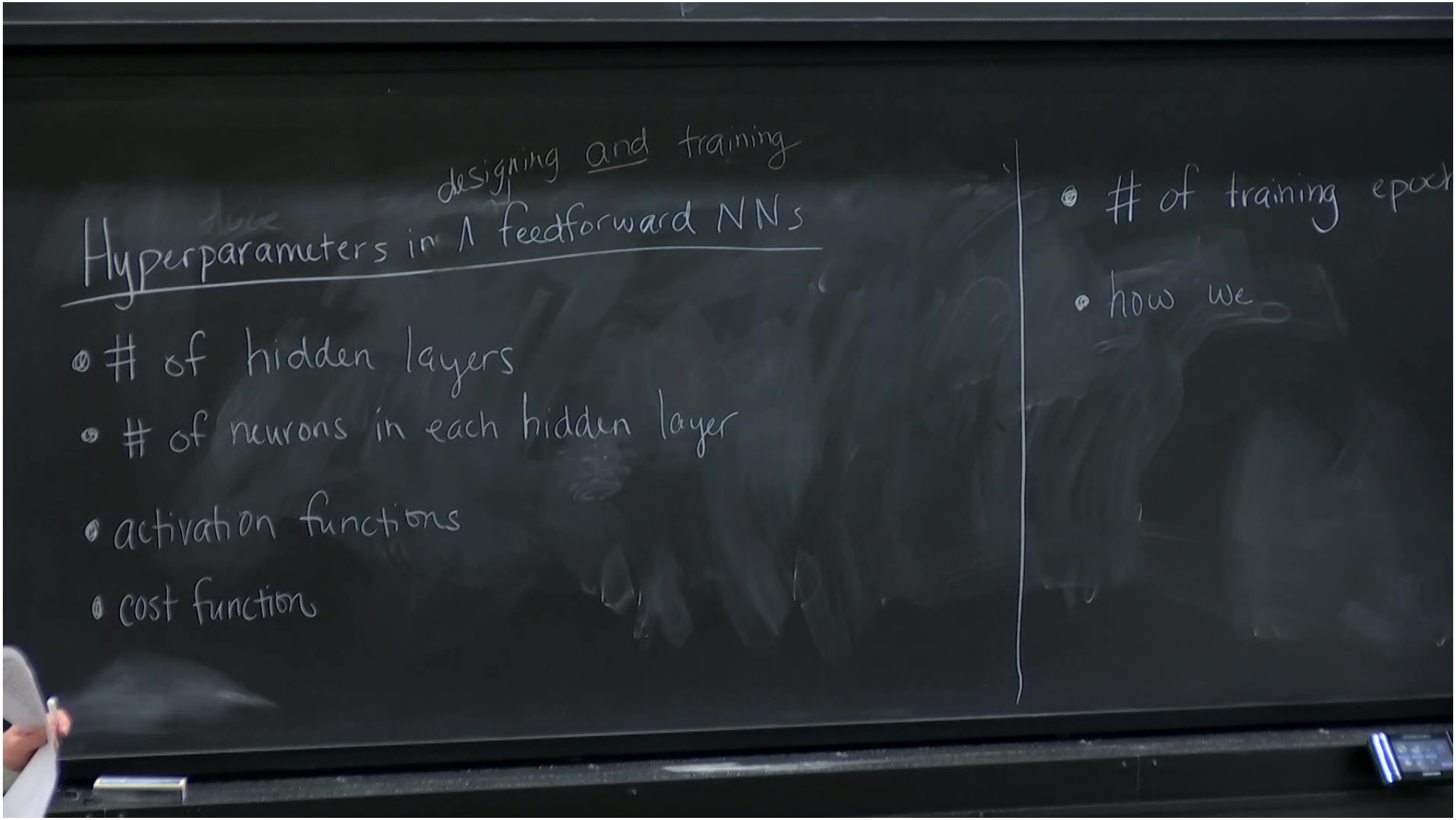

Methods that prevent overfitting
① L1 or L2 regularization
Idea: add a term to the cost function that favours lower weights $|W_{ij}^{(\ell)}|$
$C_{L1} = C_{init} + \lambda \sum_{weights} |W_{ij}^{(\ell)}|$
$C_{L2} = C_{init} + \lambda \sum_{weights} |W_{ij}^{(\ell)}|^2$

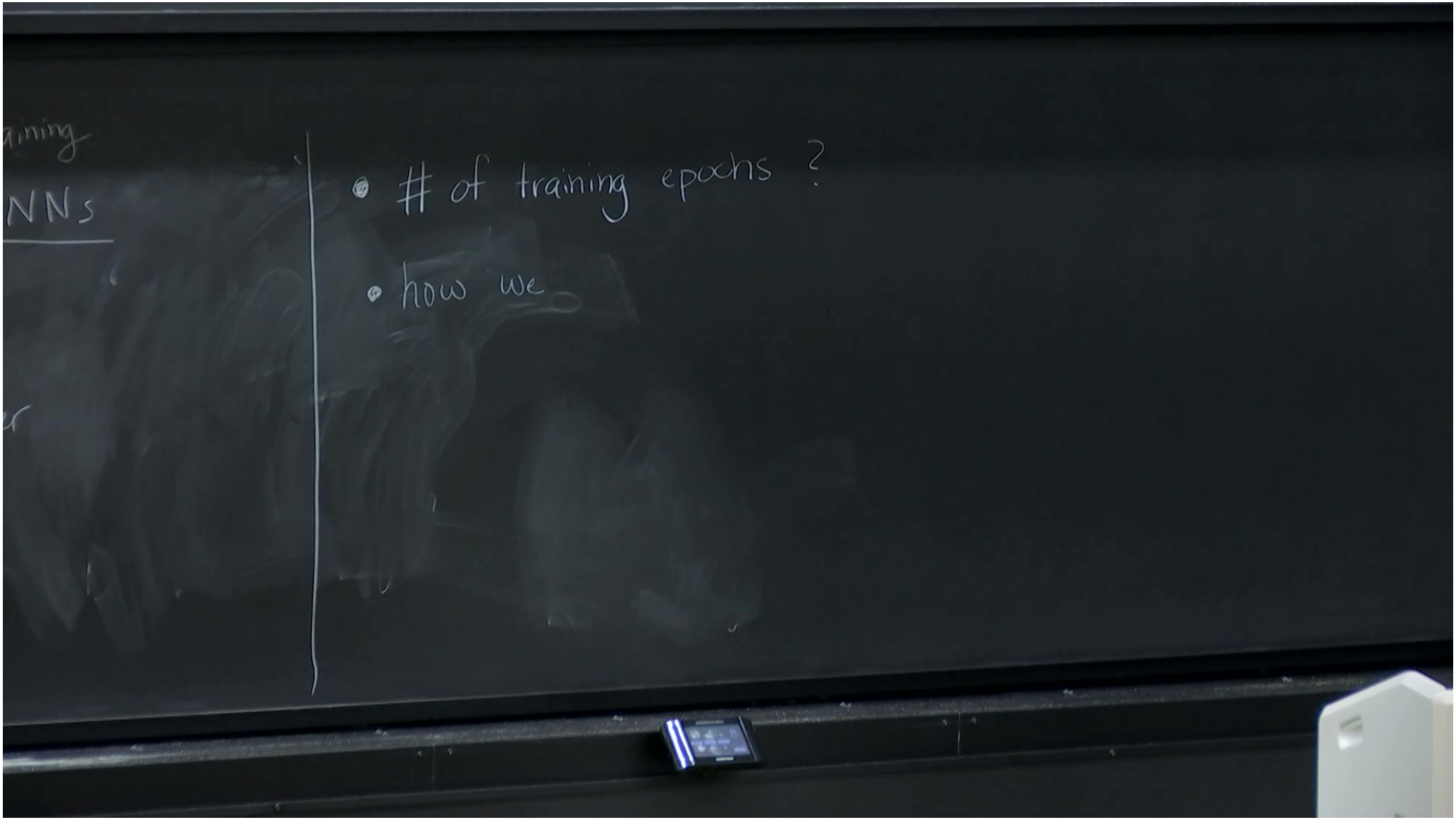

where $\lambda > 0$ is the "regularization hyperparameter"
• λ too small $\rightsquigarrow$ overfitting
• λ too large $\rightsquigarrow$ underfitting

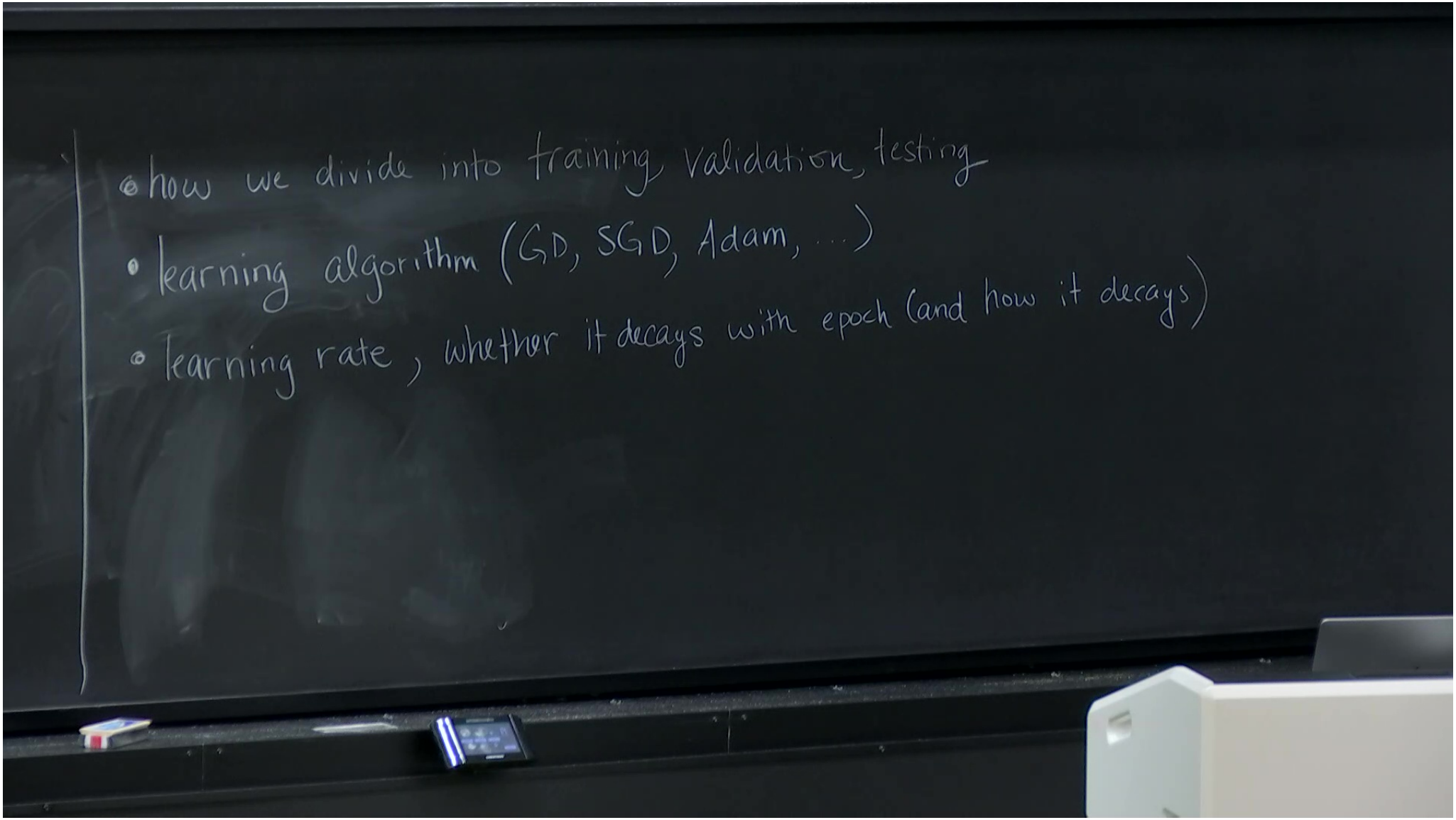

② Dropout
Idea: randomly-selected hidden neurons are temporarily "masked out" of the NN
Run training step(s) fo


Run training step(s) for a randomly-chosen masking. Then choose a new masking and repeat
Effect: similar to averaging over many different NNs (each of which could overfit in different ways)


Hyperparameters in feedforward NNs
# of hidden layers
# of neurons in each hidden layer
activation functions
cost function


designing and training
Hyperparameters in ∧ feedforward NNs
• # of hidden layers
• # of neurons in each hidden layer
• activation functions
• cost function
• # of training epoch
• how we


aining
NNs
er
• # of training epochs ?
• how we


how we divide into training, validation, testing
learning algorithm (GD, SGD, Adam, ...)
learning rate, whether it decays with epoch (and how it decays)