Advanced Deep Learning

DATA.ML.230

Fitting models and regularization

From: Simon J. D. Prince, Chapters 6 & 9 – Understanding Deep Learning, MIT Press (19 May 2025)

Fitting models

Step 1. Compute the derivatives of the loss with respect to the parameters:

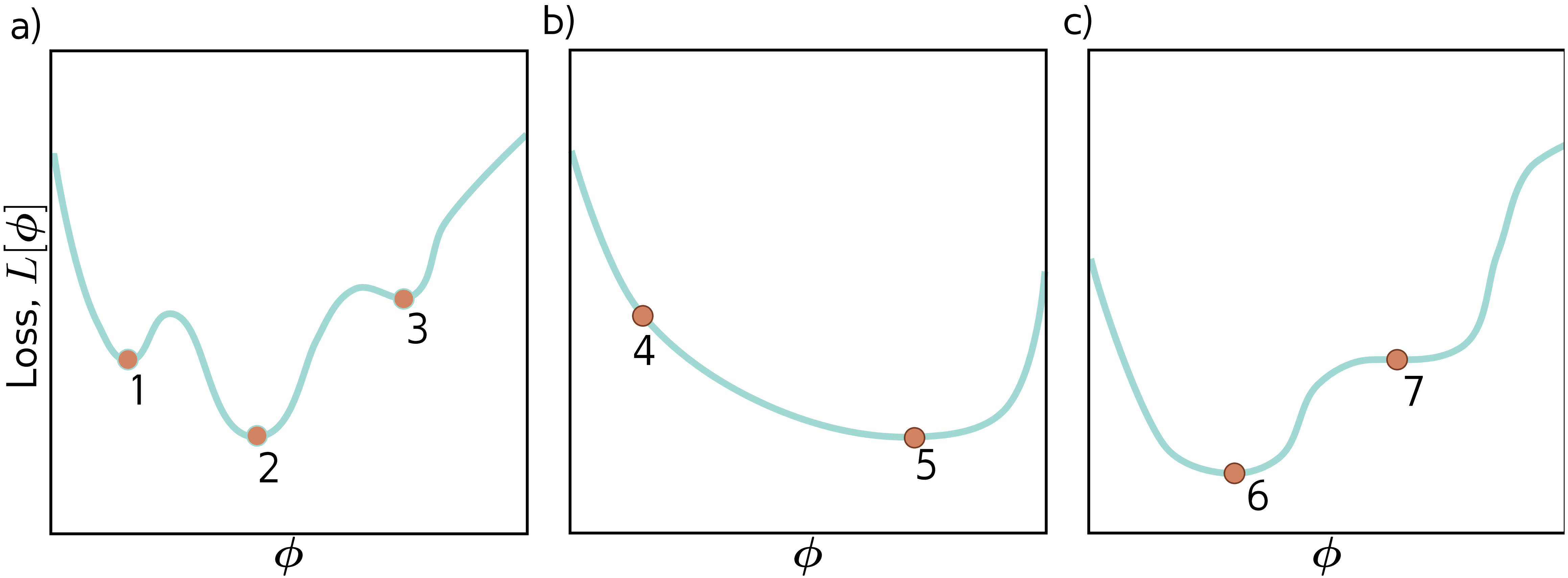
$$\frac{\partial L}{\partial \phi} = \begin{bmatrix} \frac{\partial L}{\partial \phi_0} \\ \frac{\partial L}{\partial \phi_1} \\ \vdots \\ \frac{\partial L}{\partial \phi_N} \end{bmatrix}.$$

Step 2. Update the parameters according to the rule:

$$\phi \longleftarrow \phi - \alpha \frac{\partial L}{\partial \phi},$$

where the positive scalar α determines the magnitude of the change.

Convexity



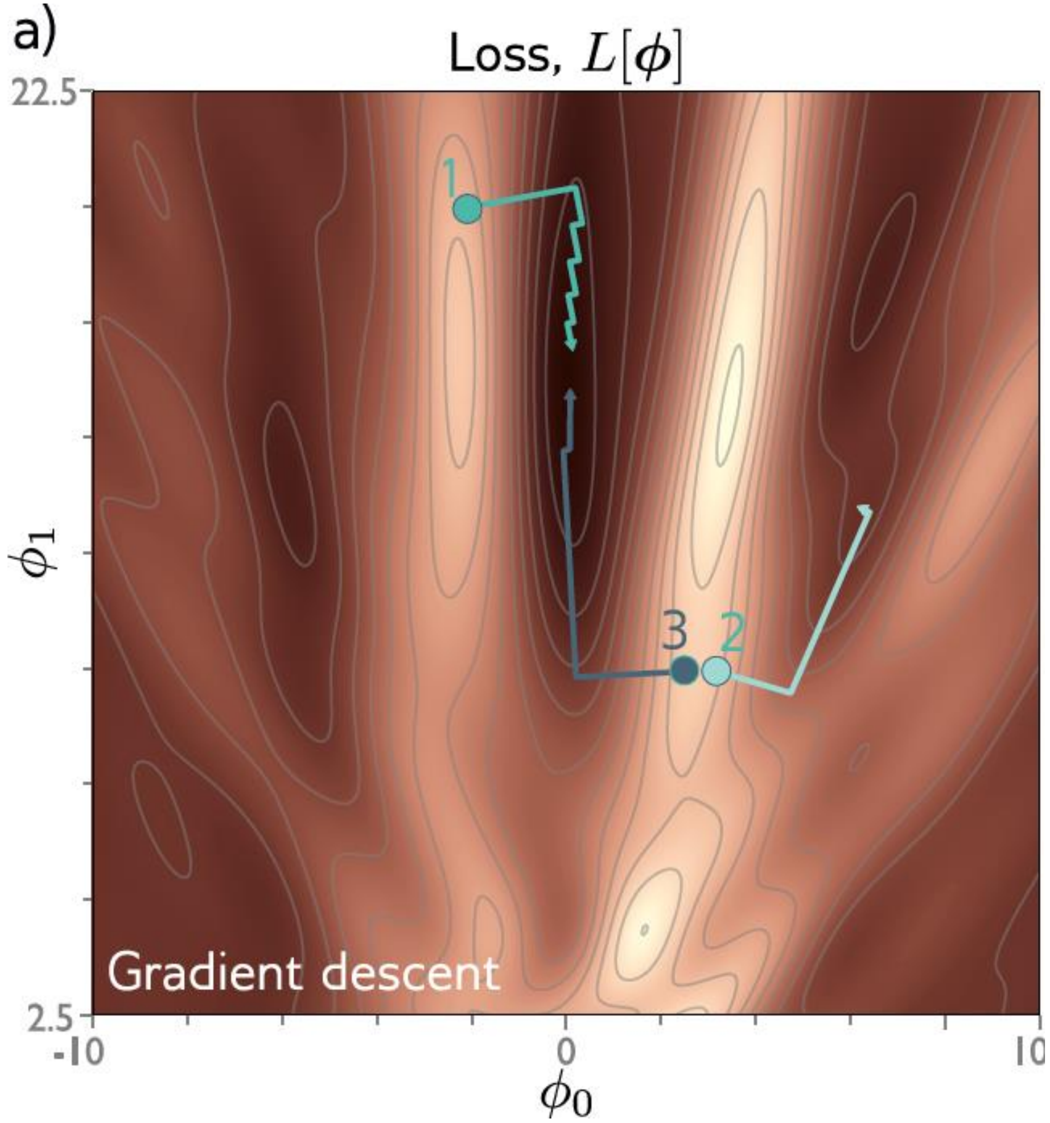
Non convex

Convex

Non-Convex

Test for convexity is that 2nd derivative is positive everywhere

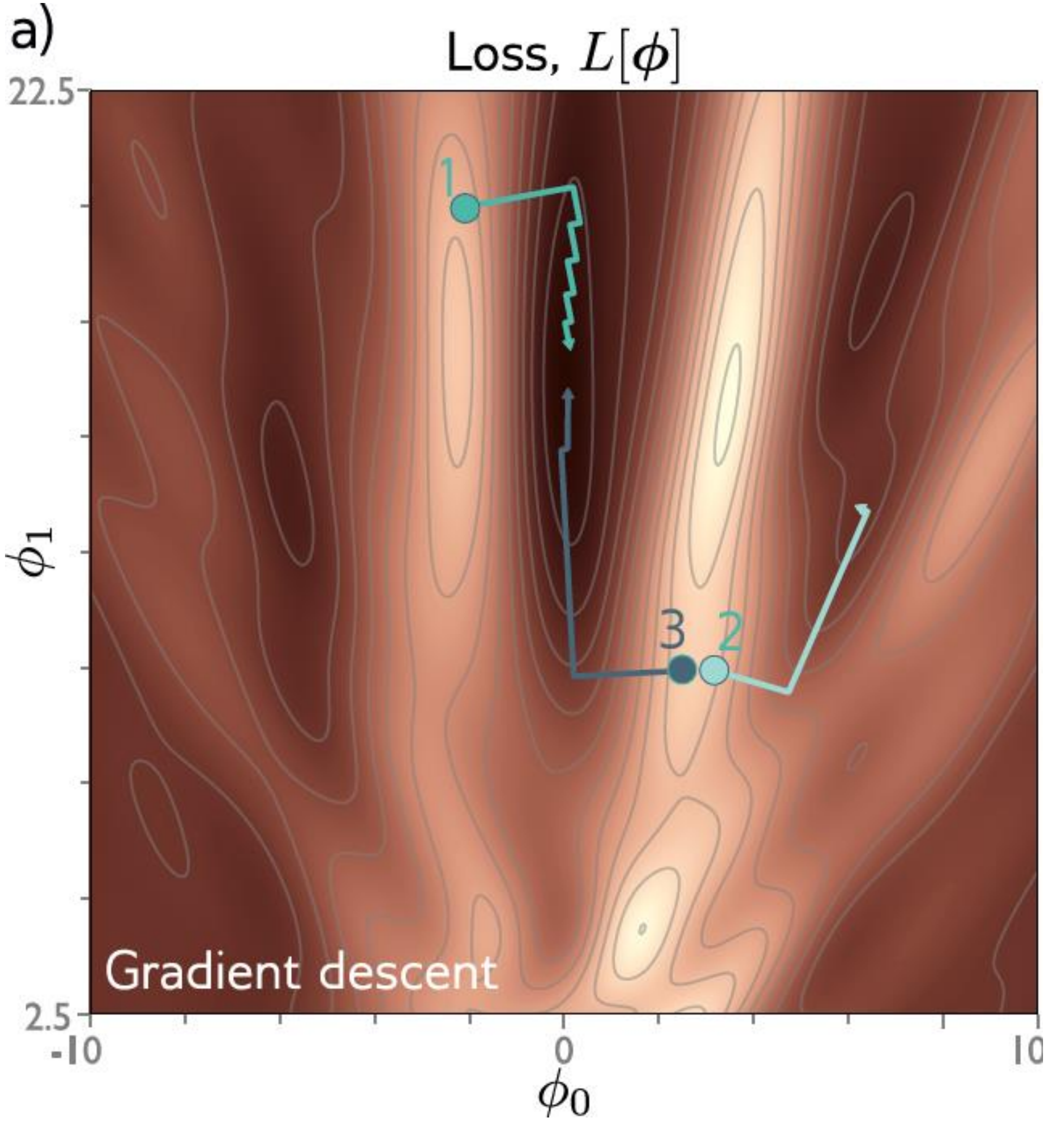
Stochastic Gradient Descent



IDEA: add noise

- Stochastic gradient descent
- Compute gradient based on only a subset of points – a mini-batch
- Work through dataset sampling without replacement
- One pass though the data is called an epoch

Stochastic Gradient Descent



Before (full batch descent)

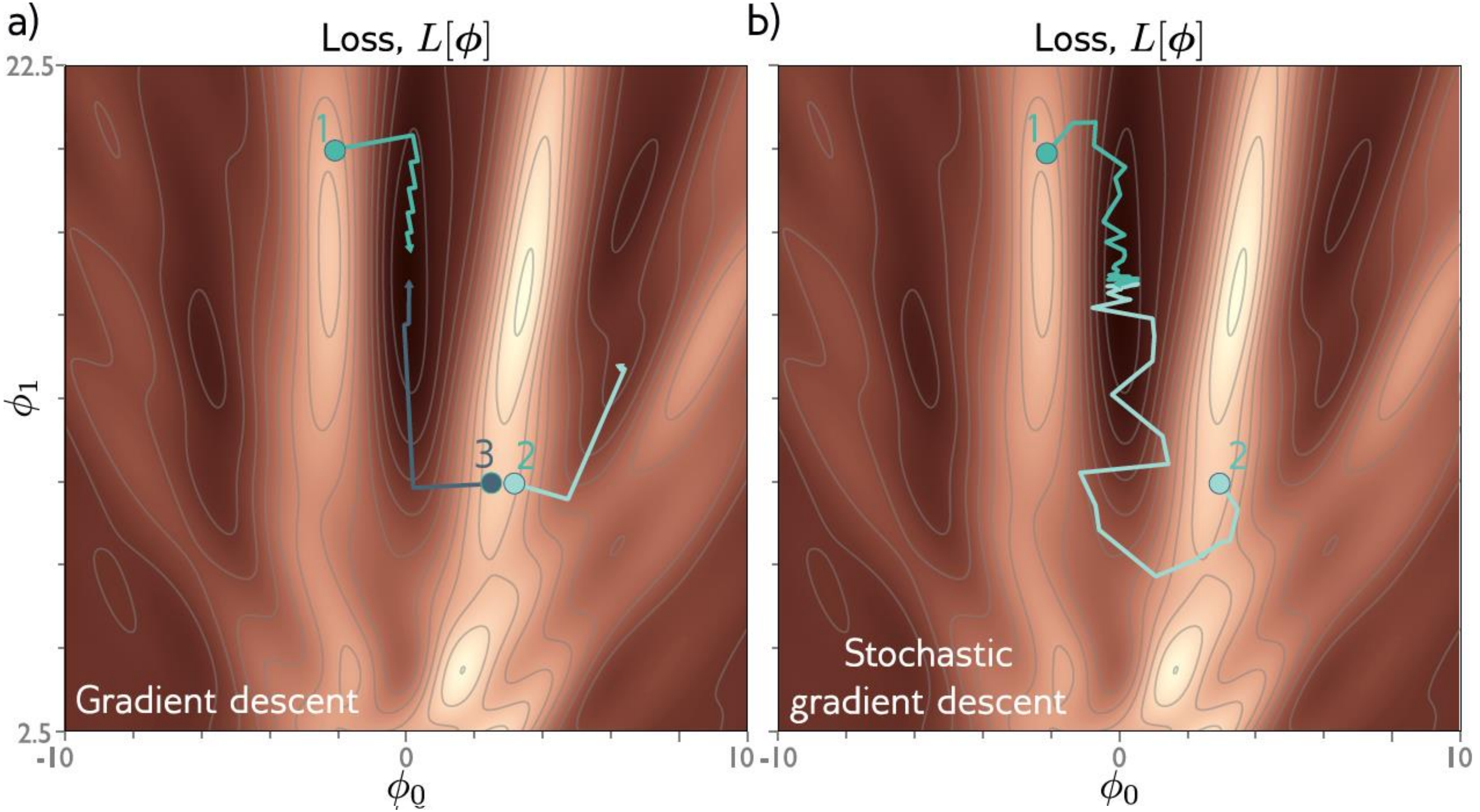
$$\phi_{t+1} \longleftarrow \phi_t - \alpha \sum_{i=1}^{I} \frac{\partial \ell_i[\phi_t]}{\partial \phi},$$

After (SGD)

$$\phi_{t+1} \longleftarrow \phi_t - \alpha \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\phi_t]}{\partial \phi},$$

Fixed learning rate α

Stochastic Gradient Descent



Properties of SGD

- Can escape from local minima
- Adds noise, but still sensible updates as based on part of data
- Uses all data equally
- Less computationally expensive
- Seems to find better solutions

- Doesn’t converge in traditional sense
- Learning rate schedule – decrease learning rate over time

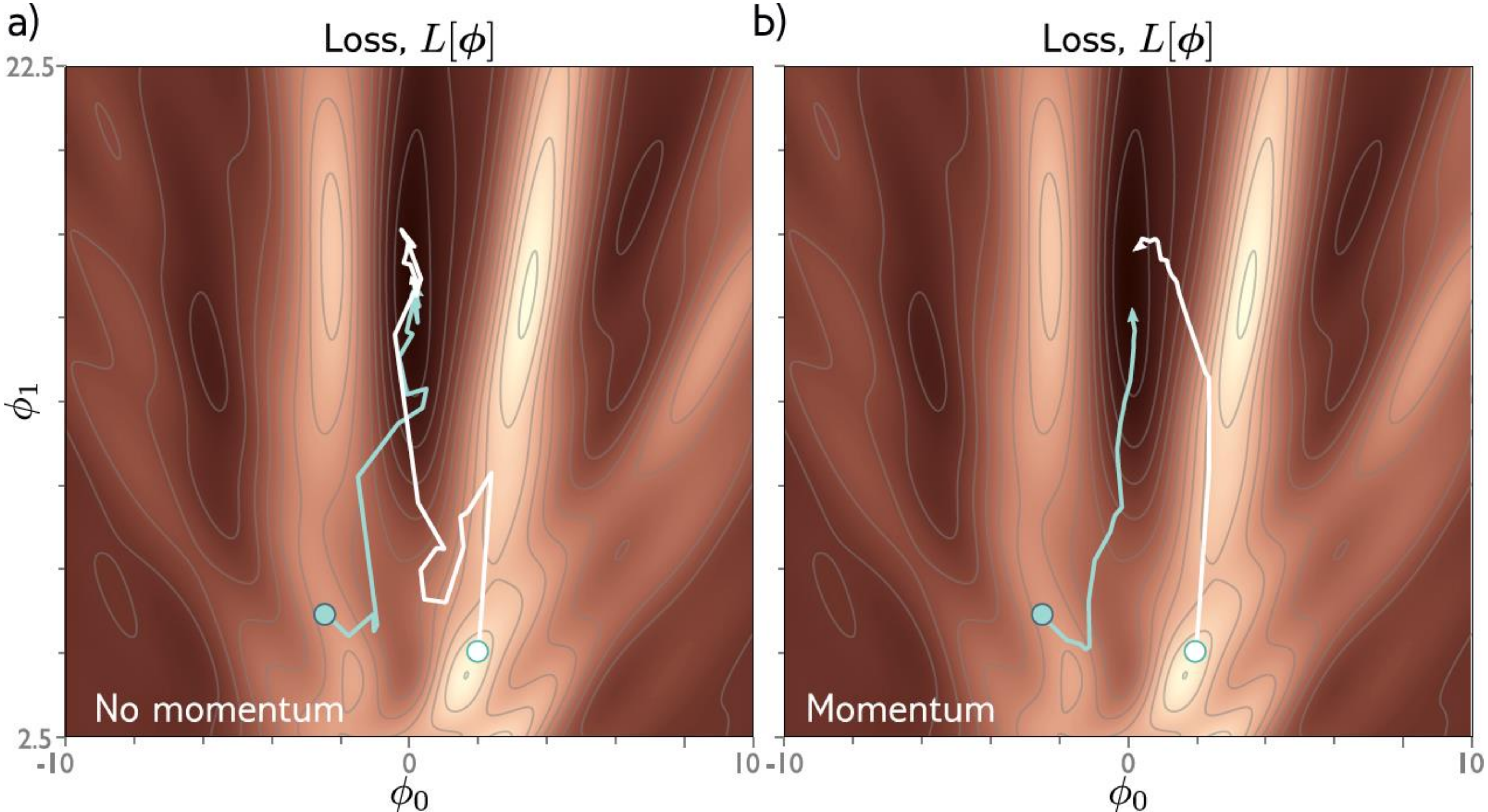
Momentum

- Weighted sum of the current gradient and the previous gradient:

$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1-\beta) \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \mathbf{m}_{t+1}$$

Momentum


a)
Loss, $L[\phi]$
22.5
2.5
ϕ_1
-10
0
10
ϕ_0
No momentum
b)
Loss, $L[\phi]$
-10
0
10
ϕ_0
Momentum

Nesterov accelerated momentum

- Momentum is kind of like a prediction of where we are going

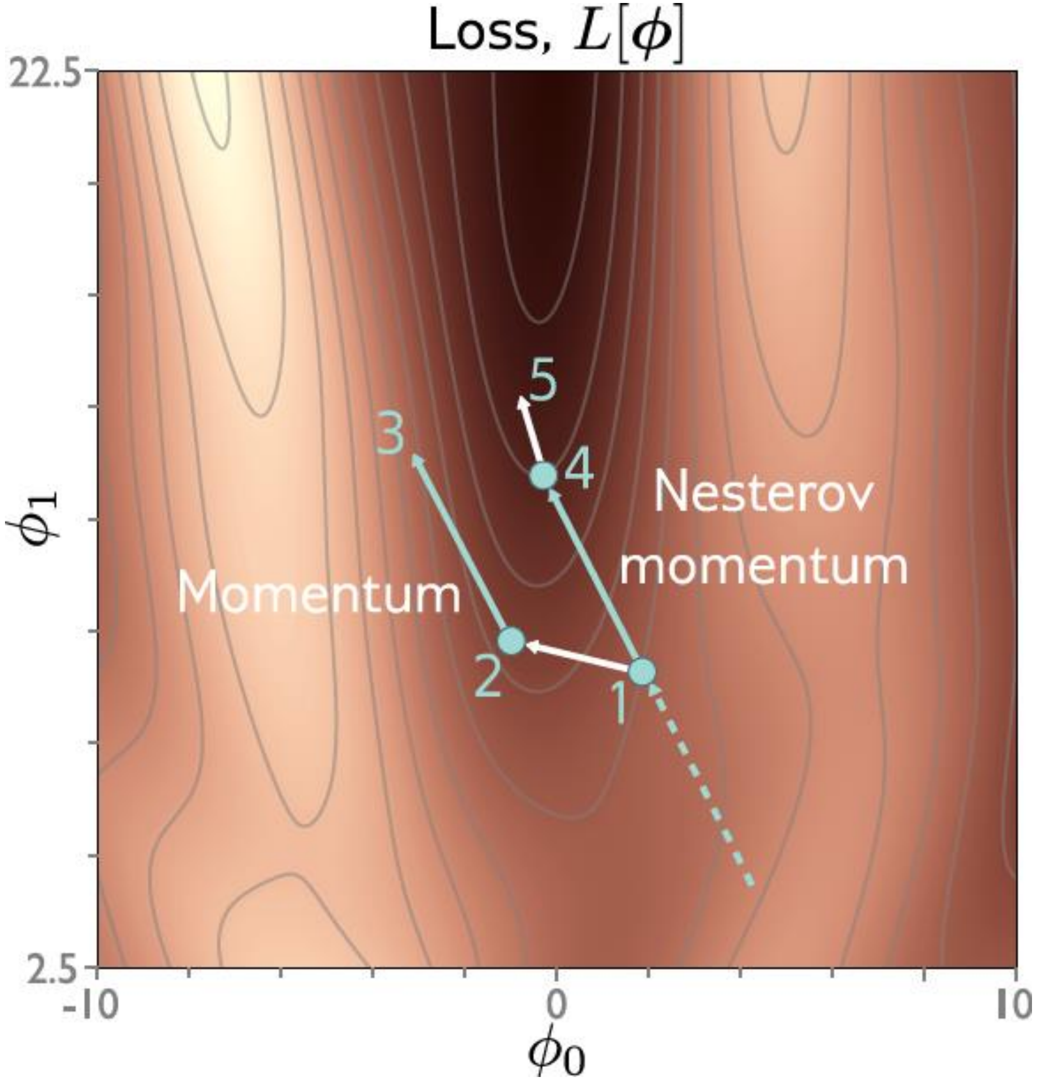
$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1-\beta) \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \mathbf{m}_{t+1}$$

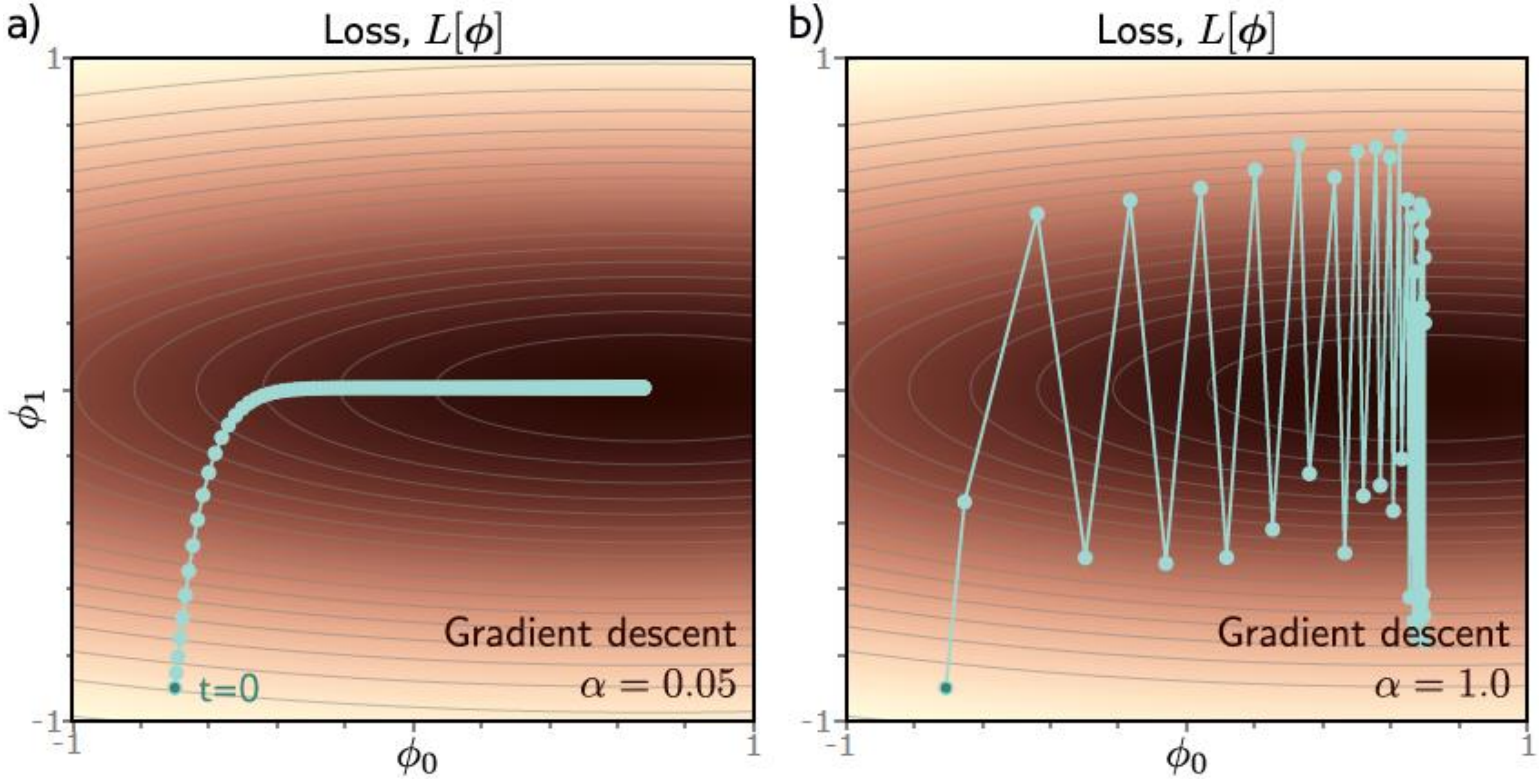
- Move in the predicted direction, THEN, measure the gradient

$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1-\beta) \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\boldsymbol{\phi}_t - \alpha \cdot \mathbf{m}_t]}{\partial \boldsymbol{\phi}}$$

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \mathbf{m}_{t+1}$$



Adaptive moment estimation (Adam)



Normalized gradients

- Measure mean and pointwise squared gradient

$$\mathbf{m}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\mathbf{v}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}^2$$

- Normalize:

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon}$$

Normalized gradients

- Measure mean and pointwise squared gradient

$$\mathbf{m}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\mathbf{v}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}^2$$

$$\mathbf{m}_{t+1} = \begin{bmatrix} 3.0 \\ -2.0 \\ 5.0 \end{bmatrix}$$

$$\mathbf{v}_{t+1} = \begin{bmatrix} 9.0 \\ 4.0 \\ 25.0 \end{bmatrix}$$

- Normalize:

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon}$$

$$\frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon} = \begin{bmatrix} 1.0 \\ -1.0 \\ 1.0 \end{bmatrix}$$

Normalized gradients

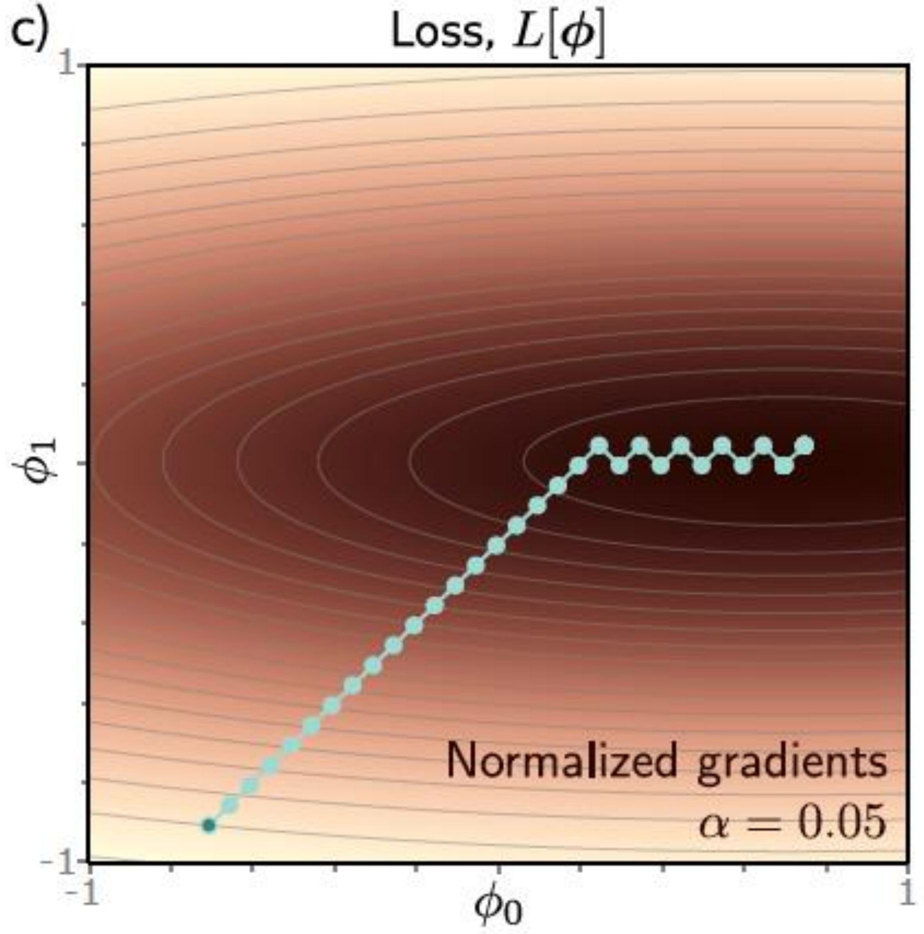
- Measure mean and pointwise squared gradient

$$\mathbf{m}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\mathbf{v}_{t+1} \leftarrow \frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}^2$$

- Normalize:

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon}$$



Adaptive moment estimation (Adam)

- Compute mean and pointwise squared gradients with momentum

$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1-\beta)\frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}$$

$$\mathbf{v}_{t+1} \leftarrow \gamma \cdot \mathbf{v}_t + (1-\gamma)\left(\frac{\partial L[\boldsymbol{\phi}_t]}{\partial \boldsymbol{\phi}}\right)^2$$

- Moderate near start of the sequence

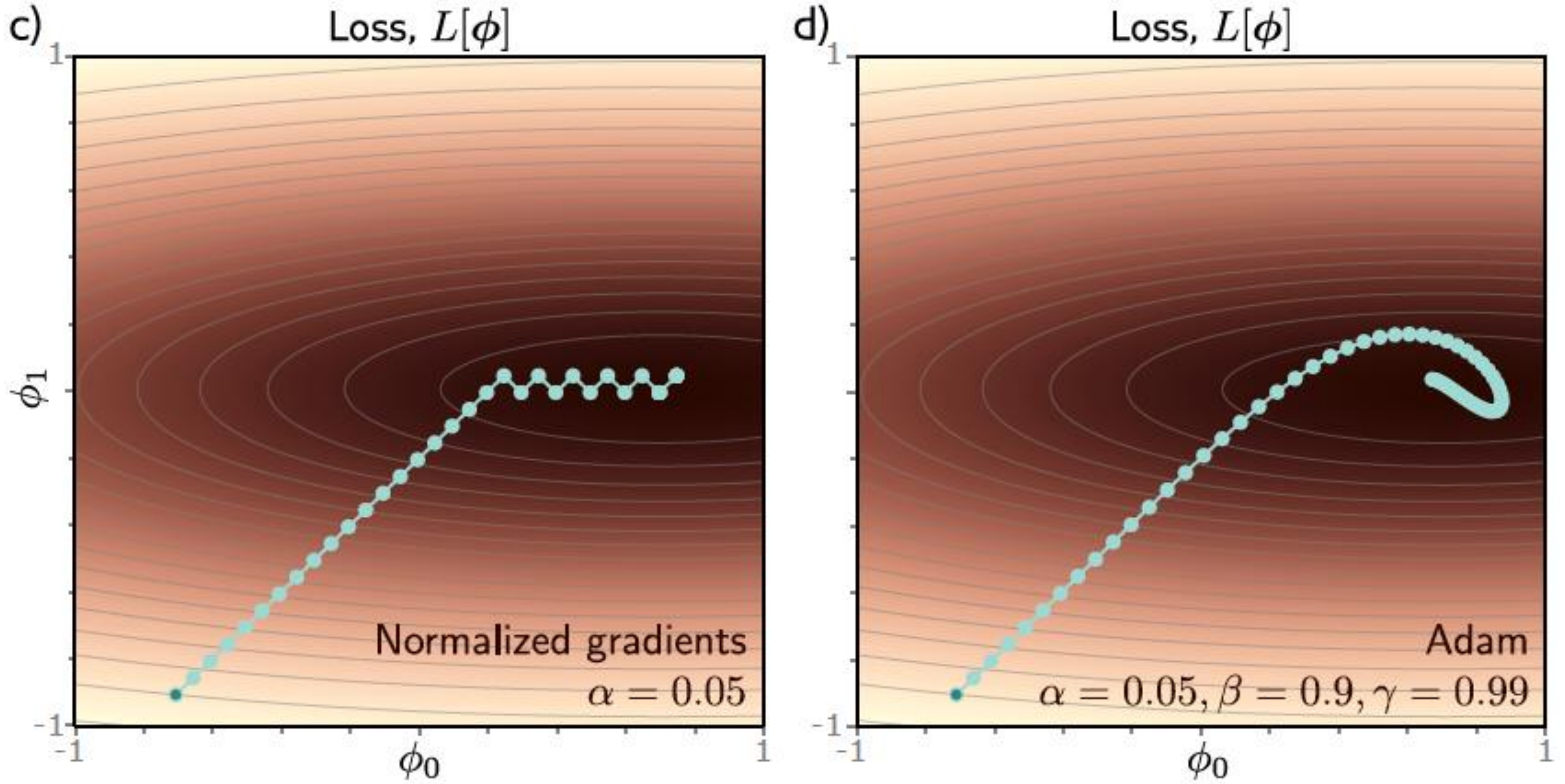
$$\tilde{\mathbf{m}}_{t+1} \leftarrow \frac{\mathbf{m}_{t+1}}{1-\beta^{t+1}}$$

$$\tilde{\mathbf{v}}_{t+1} \leftarrow \frac{\mathbf{v}_{t+1}}{1-\gamma^{t+1}}$$

- Update the parameters

$$\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \alpha \cdot \frac{\tilde{\mathbf{m}}_{t+1}}{\sqrt{\tilde{\mathbf{v}}_{t+1}} + \epsilon}$$

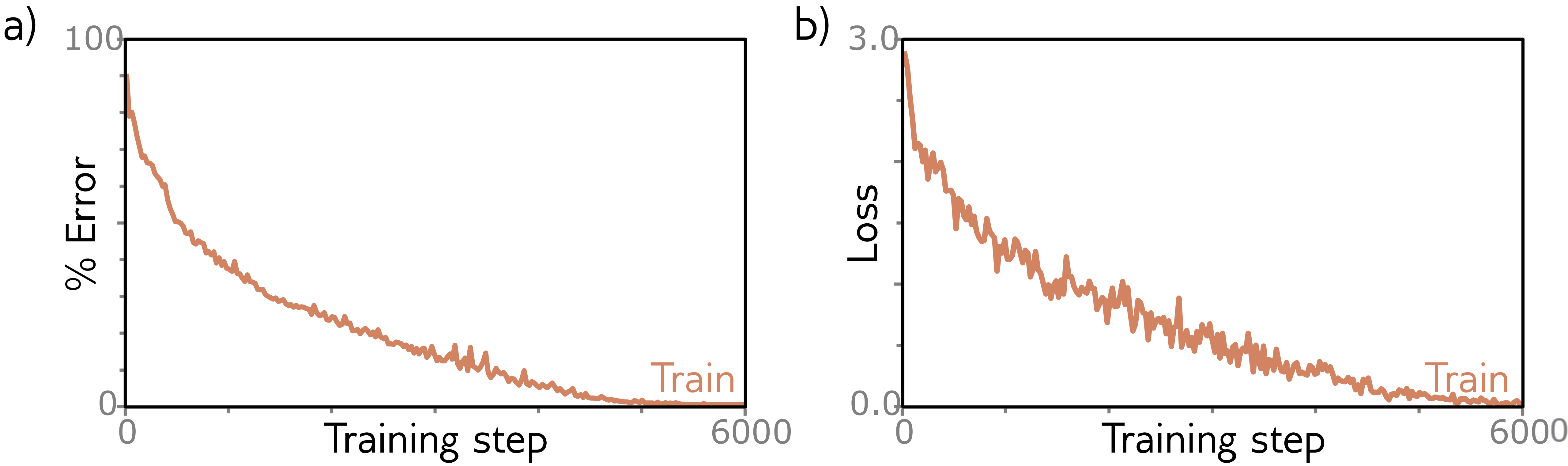
Adaptive moment estimation (Adam)



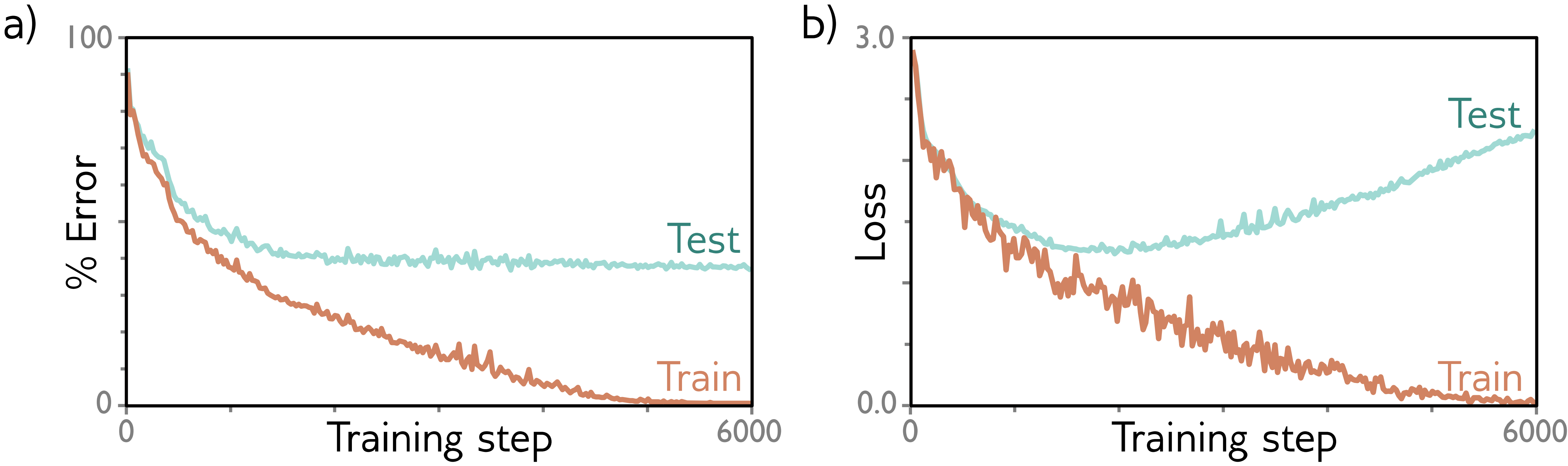
Hyperparameters

- Choice of learning algorithm
- Learning rate
- Momentum
- Neural network structure

Measuring performance

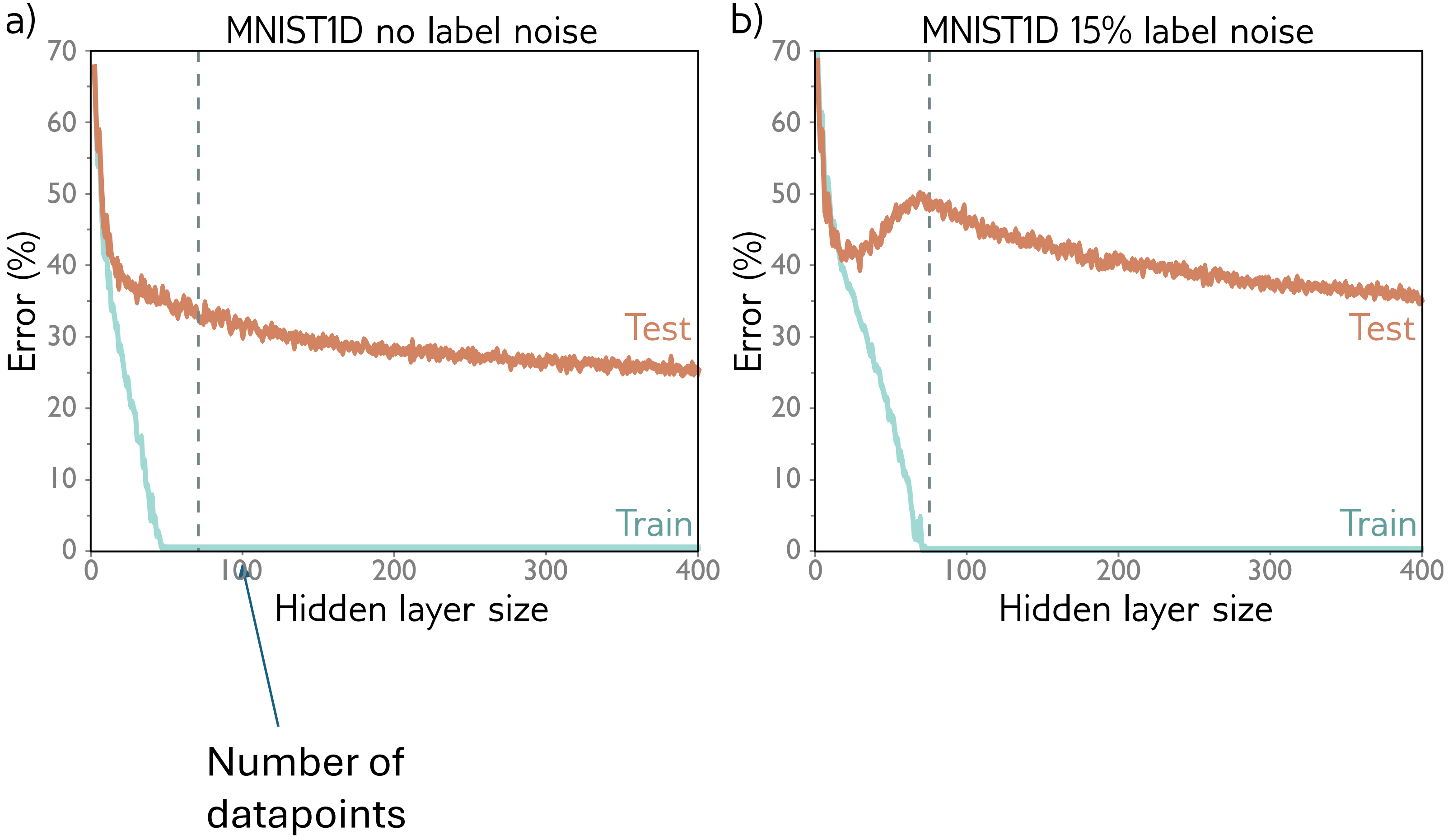


Measuring performance

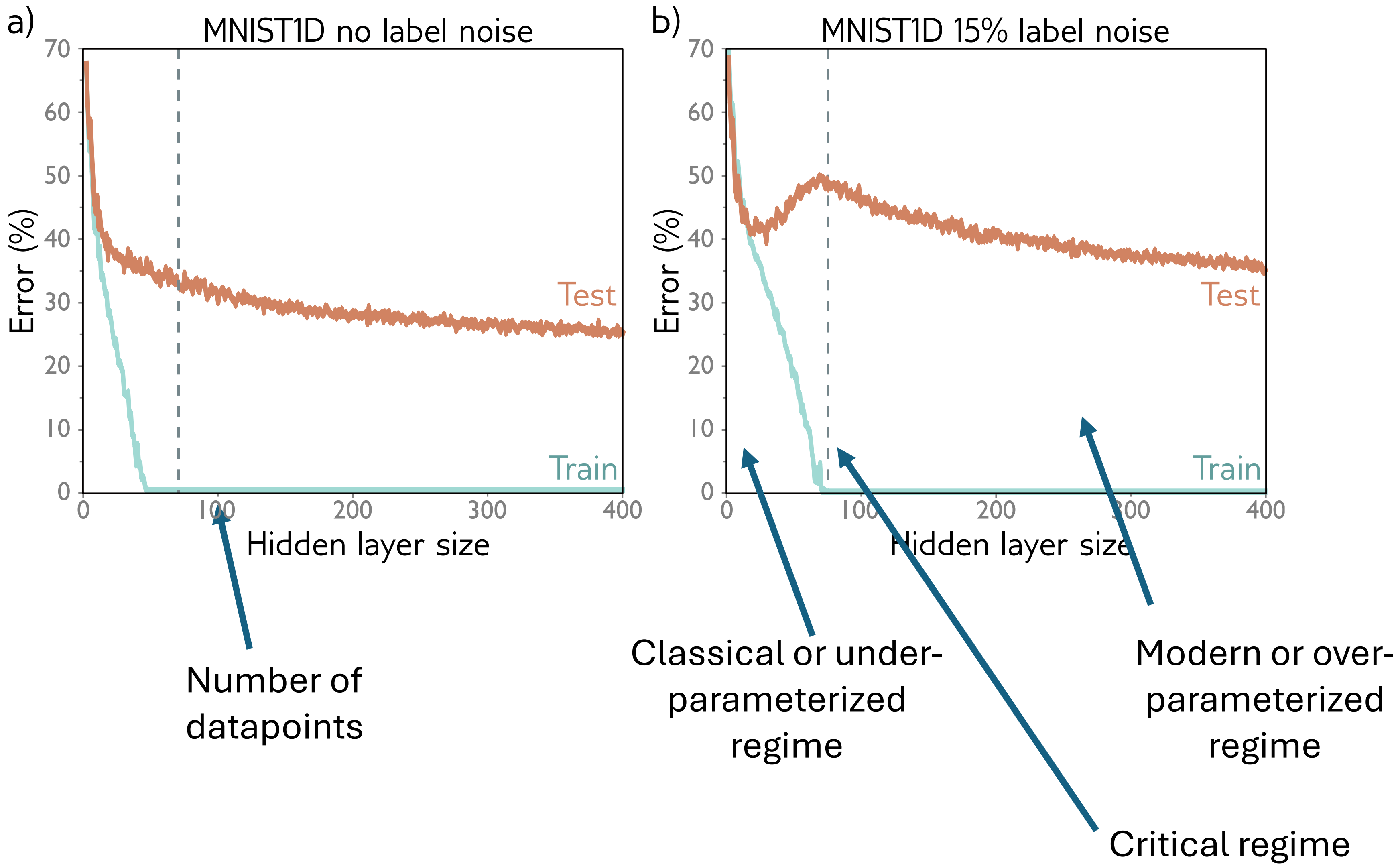


Need to use separate test data

Double descent

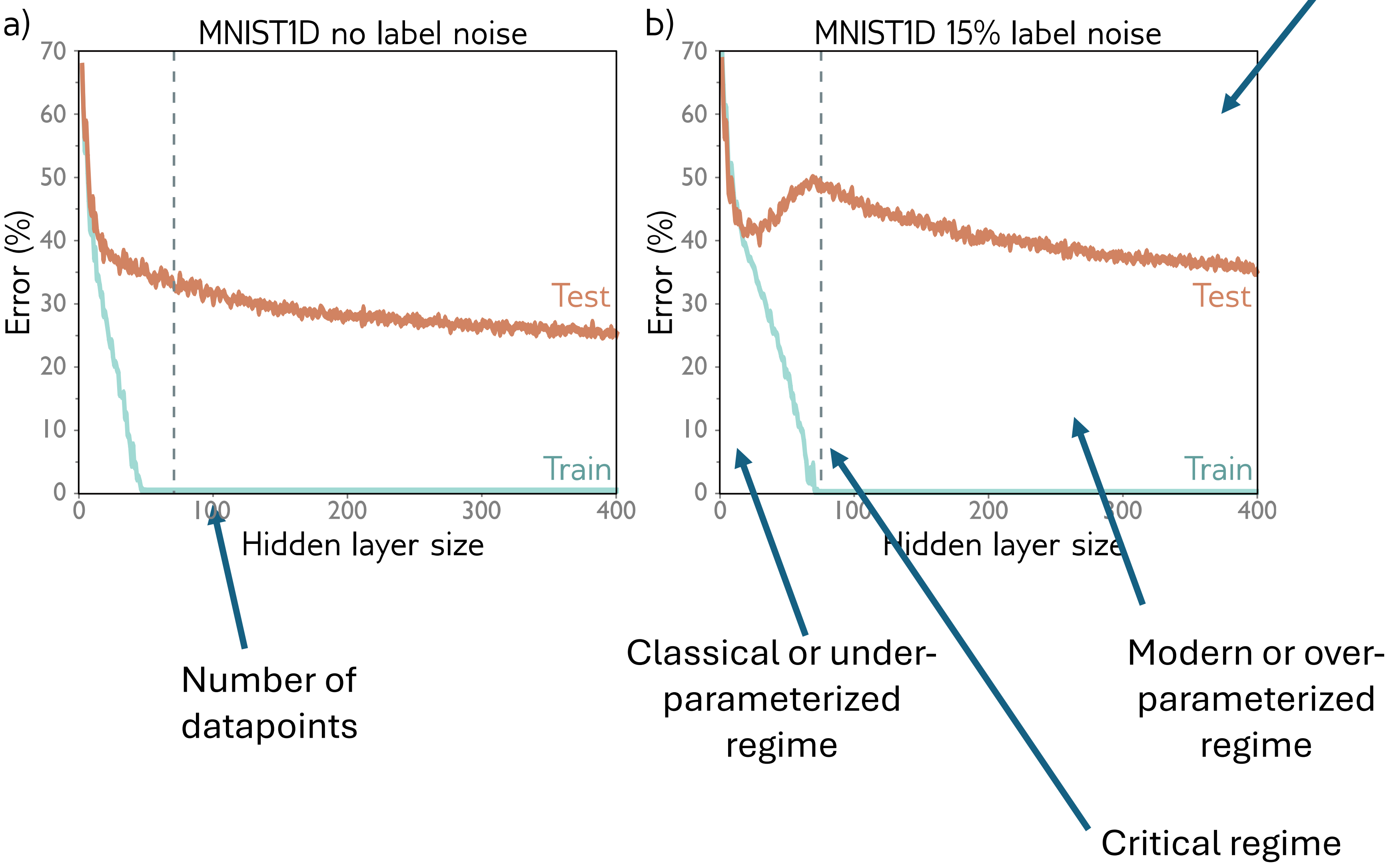


Double descent



Double descent

- Note that errors for the train data is very close to zero.
- Whatever is happening isn't happening at training data points
- Must be happening between the data points??

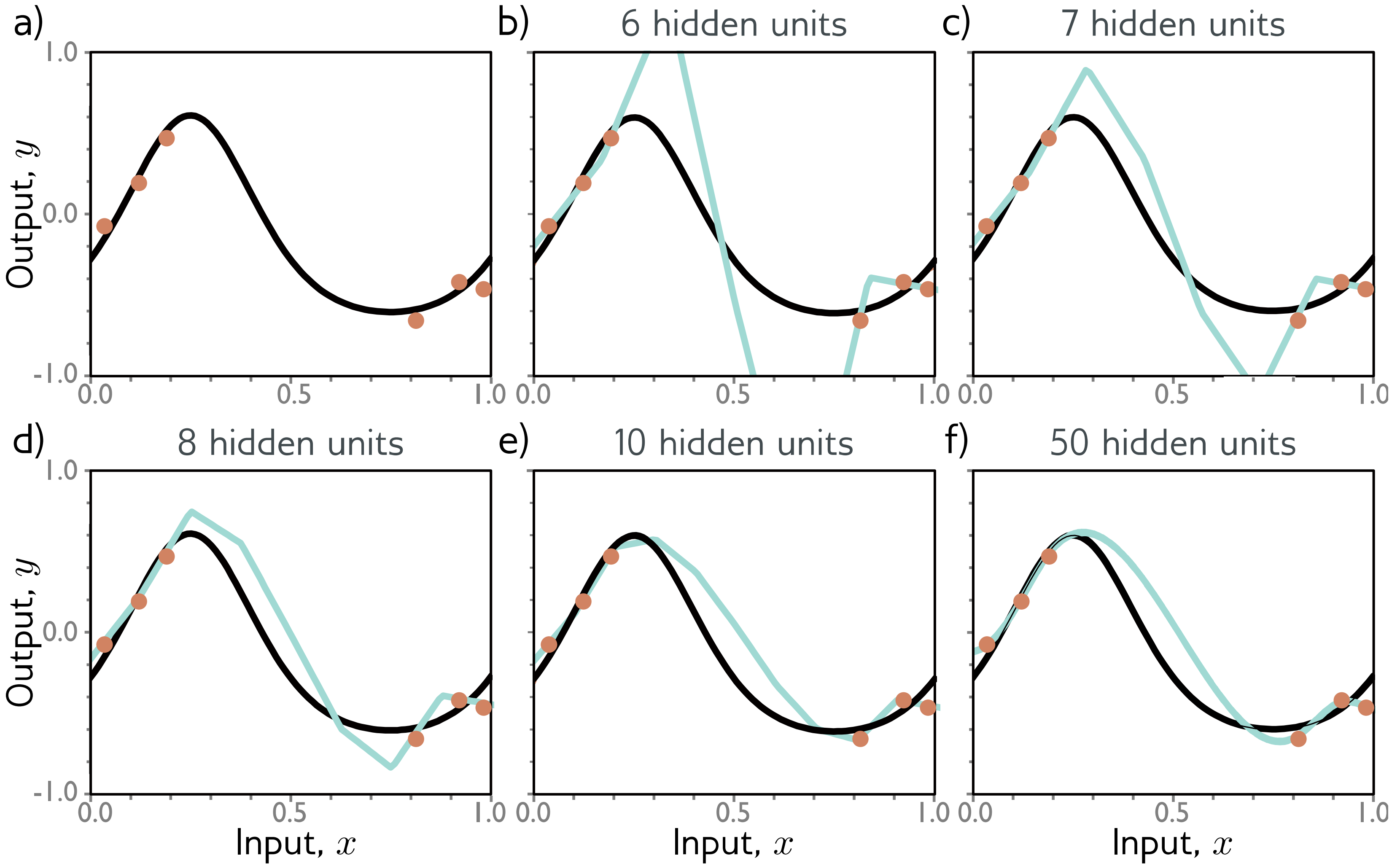


Double descent

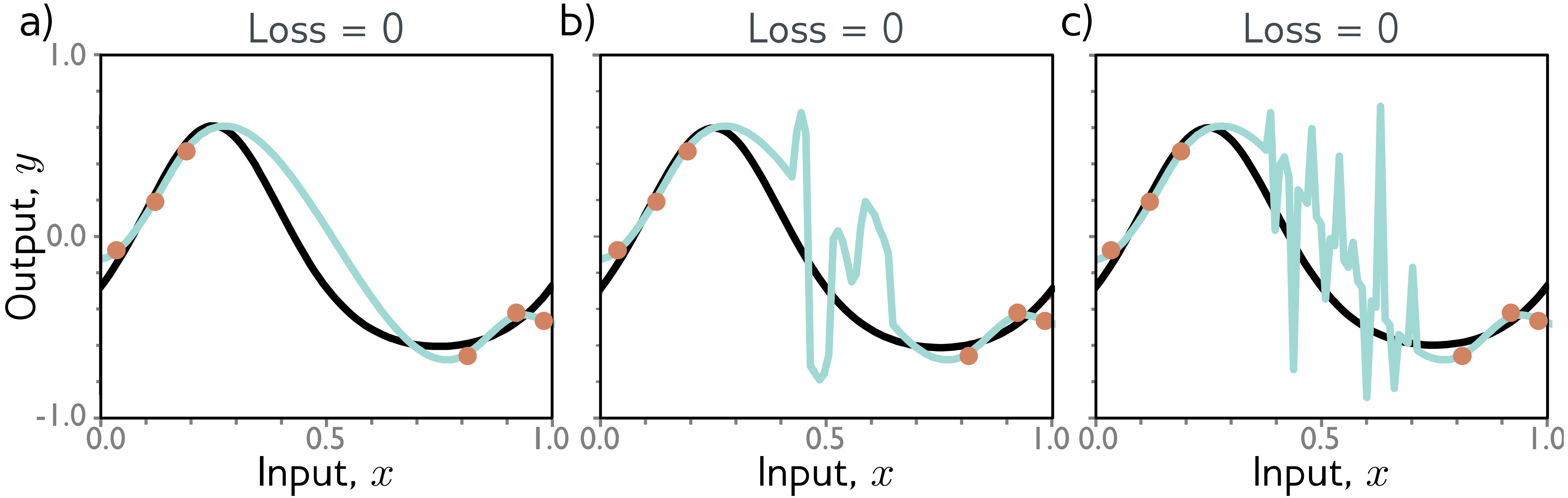
Potential explanation:

- can make smoother functions with more hidden units
- being smooth between the datapoints is a reasonable thing to do

But why?



Double descent



- All of these solutions are equivalent in terms of loss.
- Why should the model choose the smooth solution?
- Tendency of model to choose one solution over another is inductive bias
- Any factor that biases a solution toward a subset of equivalent solutions is known as a regularizer

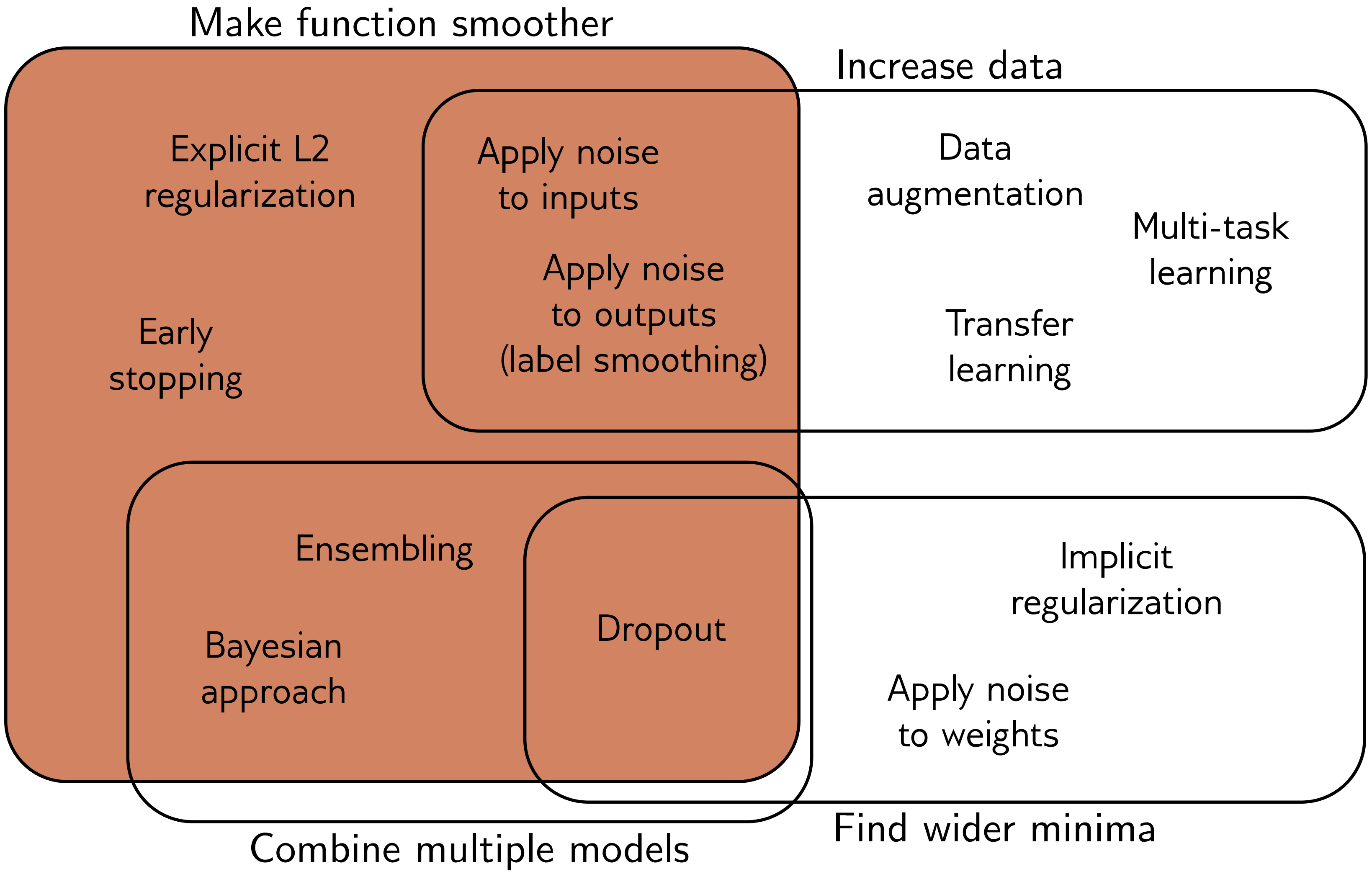
Choosing hyperparameters

- We don’t know the bias or the variance
- We don’t know how much capacity to add
- How do we choose capacity in practice?
  - Or model structure
  - Or training algorithm
  - Or learning rate
- Third data set – validation set
  - Train models with different hyperparameters on training set
  - Choose best hyperparameters with validation set
  - Test once with test set

Regularization

- Why is there a generalization gap between training and test data?
  - Overfitting (model describes statistical peculiarities)
  - Model unconstrained in areas where there are no training examples
- Regularization = methods to reduce the generalization gap
- Technically means adding terms to loss function
- But colloquially means any method (hack) to reduce gap

Regularization overview



Explicit regularization

- Standard loss function:

$$
\begin{aligned}
\hat{\boldsymbol{\phi}} &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\Big[\mathrm{L}[\boldsymbol{\phi}]\Big] \\
&= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[\sum_{i=1}^{I}\ell_i[\mathbf{x}_i,\mathbf{y}_i]\right]
\end{aligned}
$$

Explicit regularization

- Standard loss function:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\mathrm{argmin}}\Big[\mathrm{L}[\boldsymbol{\phi}]\Big] = \underset{\boldsymbol{\phi}}{\mathrm{argmin}}\left[\sum_{i=1}^{I} \ell_i[\mathbf{x}_i, \mathbf{y}_i]\right]$$

- Regularization adds an extra term

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\mathrm{argmin}}\left[\sum_{i=1}^{I} \ell_i[\mathbf{x}_i, \mathbf{y}_i] + \lambda \cdot \mathrm{g}[\boldsymbol{\phi}]\right]$$

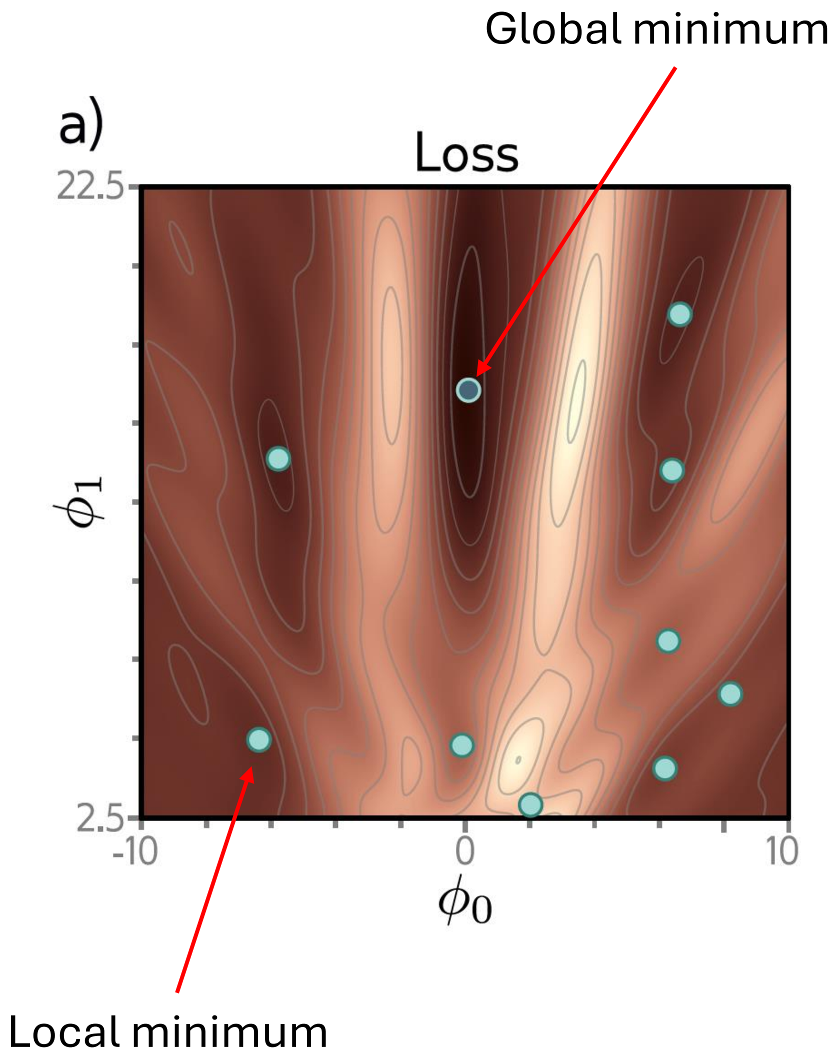
Explicit regularization

- Standard loss function: $$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\Big[\mathrm{L}[\boldsymbol{\phi}]\Big] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[\sum_{i=1}^{I}\ell_i[\mathbf{x}_i, \mathbf{y}_i]\right]$$
- Regularization adds an extra term

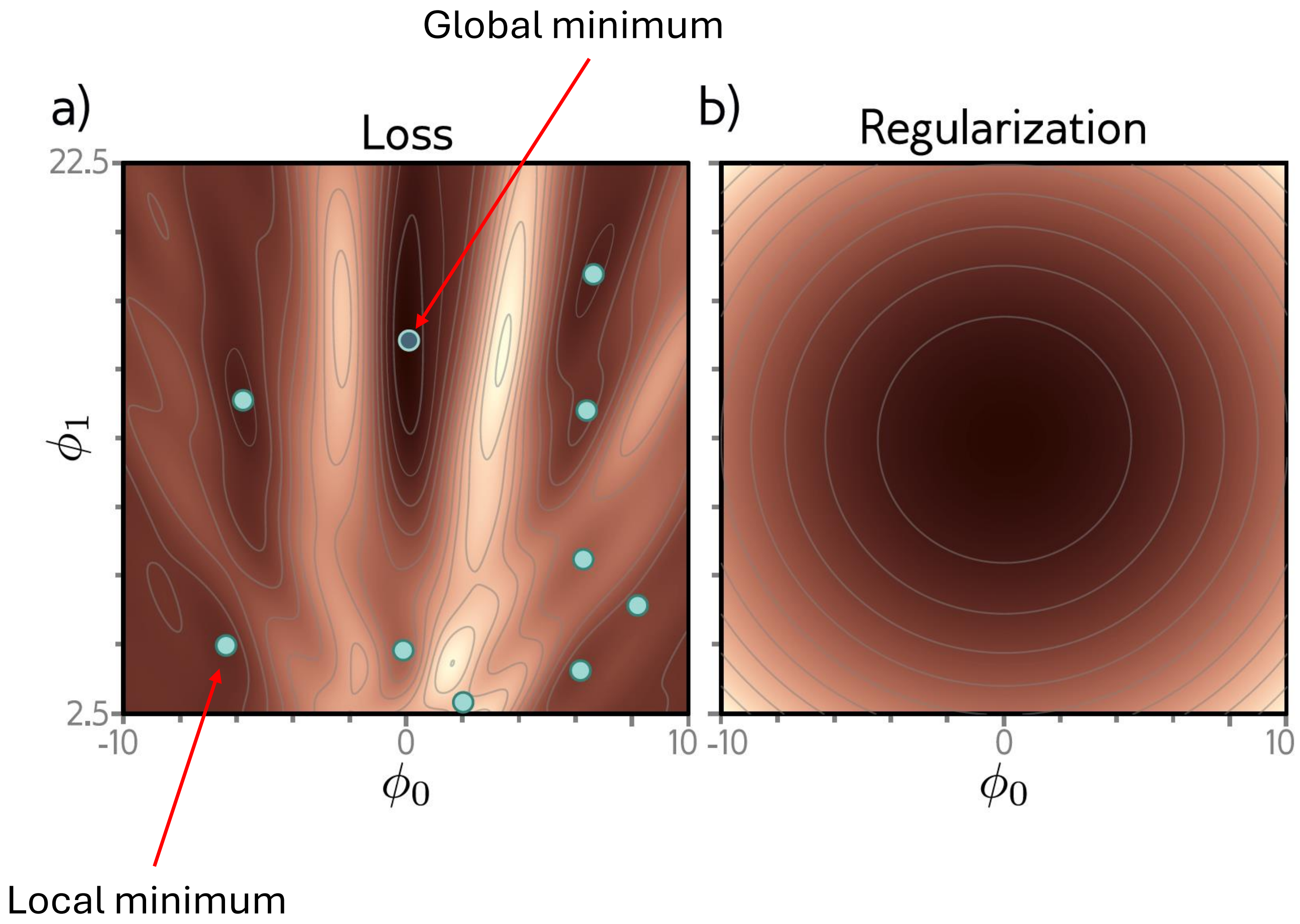
$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[\sum_{i=1}^{I}\ell_i[\mathbf{x}_i, \mathbf{y}_i] + \lambda \cdot \mathrm{g}[\boldsymbol{\phi}]\right]$$

- Favors some parameters, disfavors others.
- λ>0 controls the strength

Explicit regularization



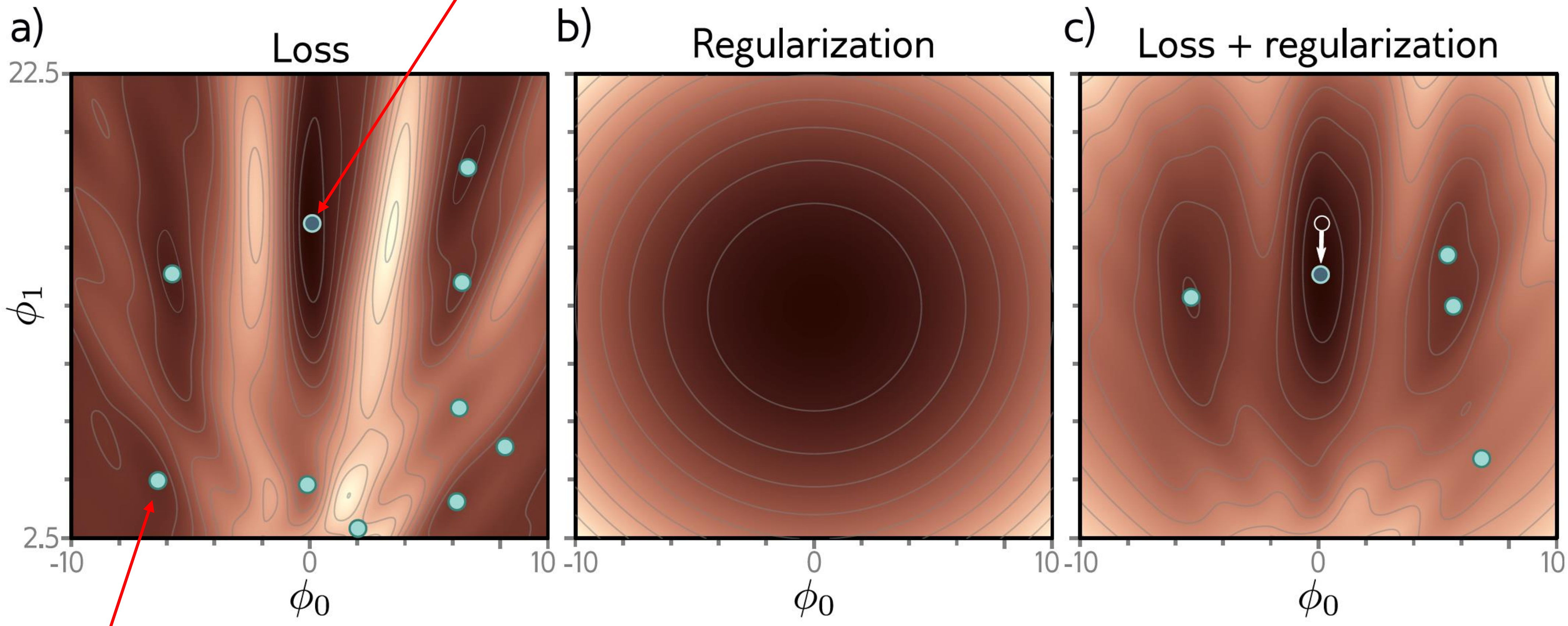
Explicit regularization



Explicit regularization

The combined loss has fewer local minima, and the global minimum has moved to a different position

Global minimum



Local minimum

Probabilistic interpretation

- Maximum likelihood:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i | \mathbf{x}_i, \boldsymbol{\phi}) \right]$$

Probabilistic interpretation

- Maximum likelihood:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\phi}) \right]$$

- Regularization is equivalent to a adding a prior over parameters

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\phi}) Pr(\boldsymbol{\phi}) \right]$$

Maximum a posteriori (MAP) criterion

... what you know about parameters *before* seeing the data

Equivalence

- Explicit regularization:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \ell_i[\mathbf{x}_i, \mathbf{y}_i] + \lambda \cdot \mathrm{g}[\boldsymbol{\phi}] \right]$$

- Probabilistic interpretation:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i | \mathbf{x}_i, \boldsymbol{\phi}) Pr(\boldsymbol{\phi}) \right]$$

Equivalence

- Explicit regularization:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \ell_i[\mathbf{x}_i, \mathbf{y}_i] + \lambda \cdot \mathrm{g}[\boldsymbol{\phi}] \right]$$

- Probabilistic interpretation:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\phi}) Pr(\boldsymbol{\phi}) \right]$$

- Mapping: $\lambda \cdot \mathrm{g}[\boldsymbol{\phi}] = -\log[Pr(\boldsymbol{\phi})]$

L2 Regularization

- Can only use very general terms
- Most common is L2 regularization
- Favors smaller parameters

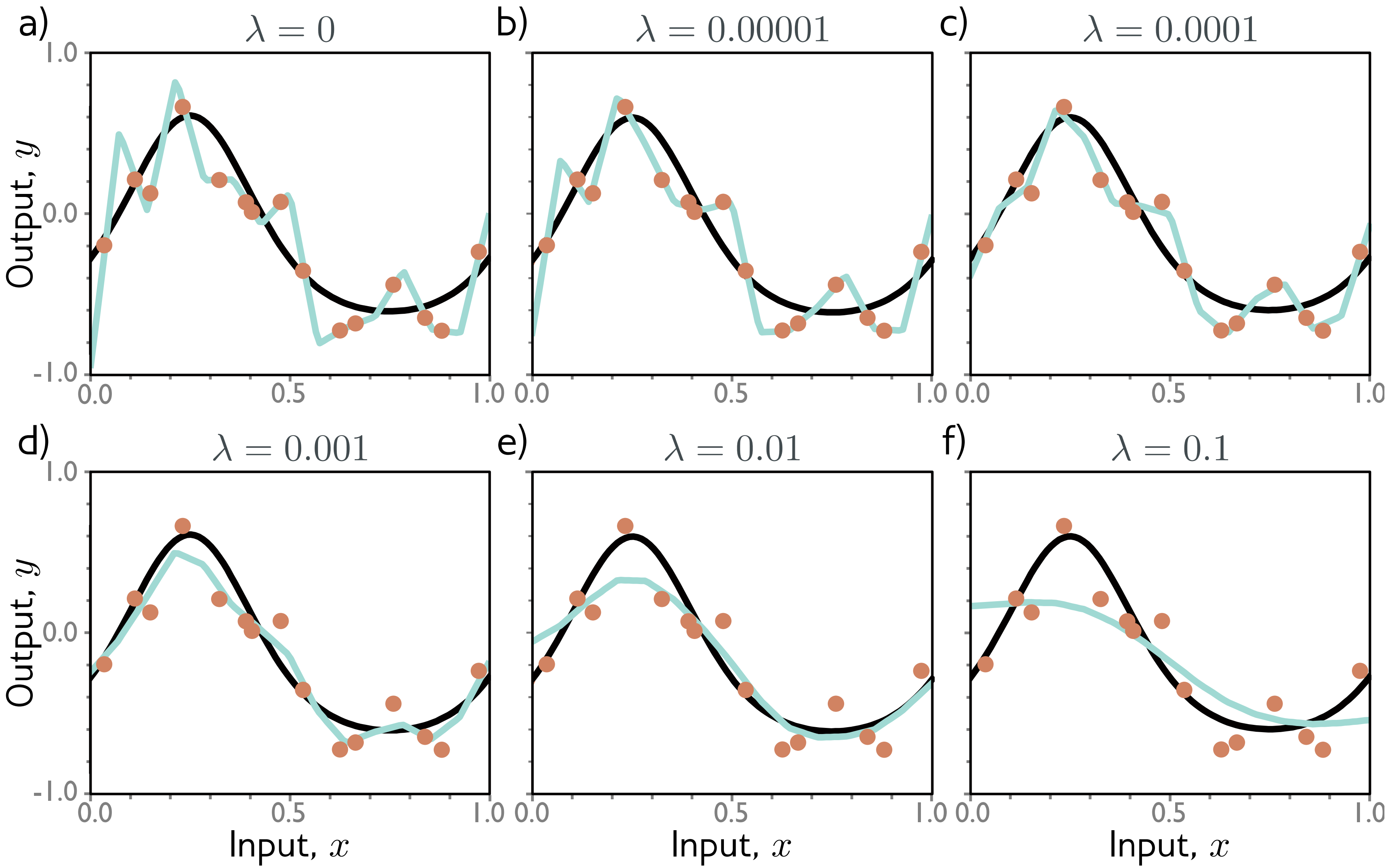
$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\mathrm{L}[\boldsymbol{\phi}, \{\mathbf{x}_i, \mathbf{y}_i\}] + \lambda \sum_j \phi_j^2 \right]$$

- Also called Tikhonov regularization, ridge regression, or Frobenius norm regularization
- In neural networks, usually just for weights (not for the biases) and called weight decay

Why does L2 regularization help?

- Discourages memorizing the data (overfitting)
- Encourages smoothness between datapoints

L2 regularization

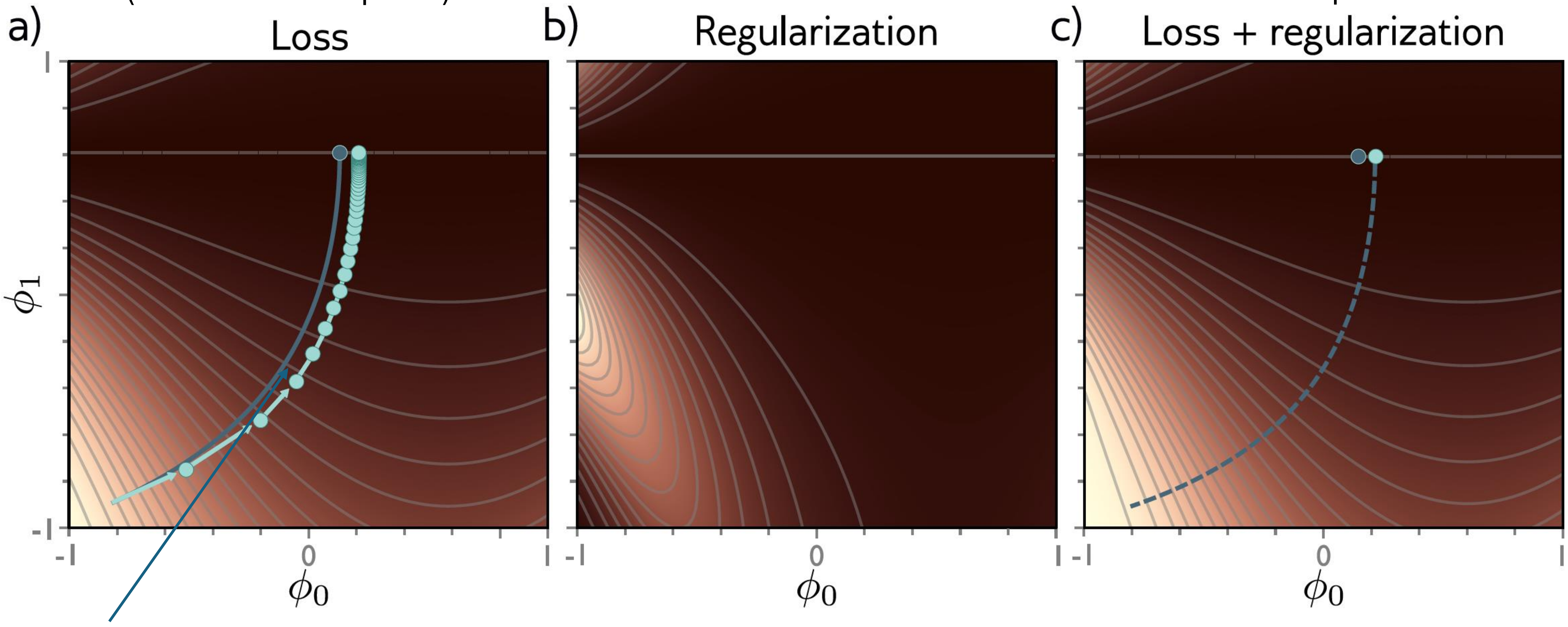


Implicit regularization

Gradient descent approximates a differential equation (infinitesimal step size)

Finite step size equivalent to regularization

Add in that regularization and differential equation converges to same place



Continuous gradient descent path. The finite step size causes reaching a different final position

Implicit regularization

- Gradient descent disfavors areas where gradients are steep

$$\tilde{\mathrm{L}}_{GD}[\boldsymbol{\phi}] = \mathrm{L}[\boldsymbol{\phi}] + \frac{\alpha}{4}\left\|\frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2$$

Implicit regularization

- Gradient descent disfavors areas where gradients are steep

$$\tilde{\mathrm{L}}_{GD}[\boldsymbol{\phi}] = \mathrm{L}[\boldsymbol{\phi}] + \frac{\alpha}{4}\left\|\frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2$$

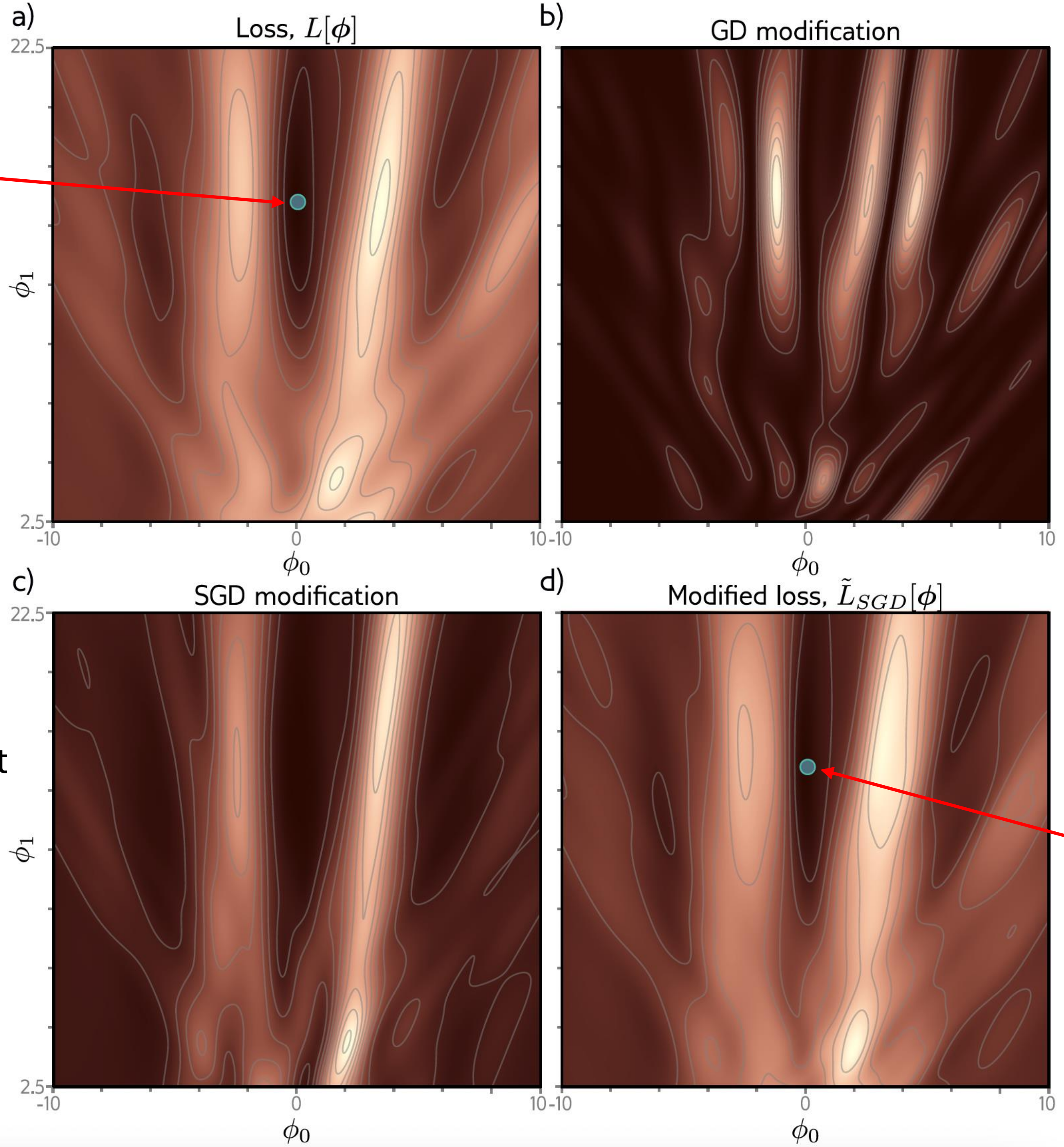
- SGD likes all batches to have similar gradients

$$\begin{aligned}\tilde{\mathrm{L}}_{SGD}[\boldsymbol{\phi}] &= \tilde{\mathrm{L}}_{GD}[\boldsymbol{\phi}] + \frac{\alpha}{4B}\sum_{b=1}^{B}\left\|\frac{\partial L_b}{\partial \boldsymbol{\phi}} - \frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2 \\ &= \mathrm{L}[\boldsymbol{\phi}] + \frac{\alpha}{4}\left\|\frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2 + \frac{\alpha}{4B}\sum_{b=1}^{B}\left\|\frac{\partial L_b}{\partial \boldsymbol{\phi}} - \frac{\partial L}{\partial \boldsymbol{\phi}}\right\|^2\end{aligned}$$

- Depends on learning rate – perhaps why larger learning rates generalize better.

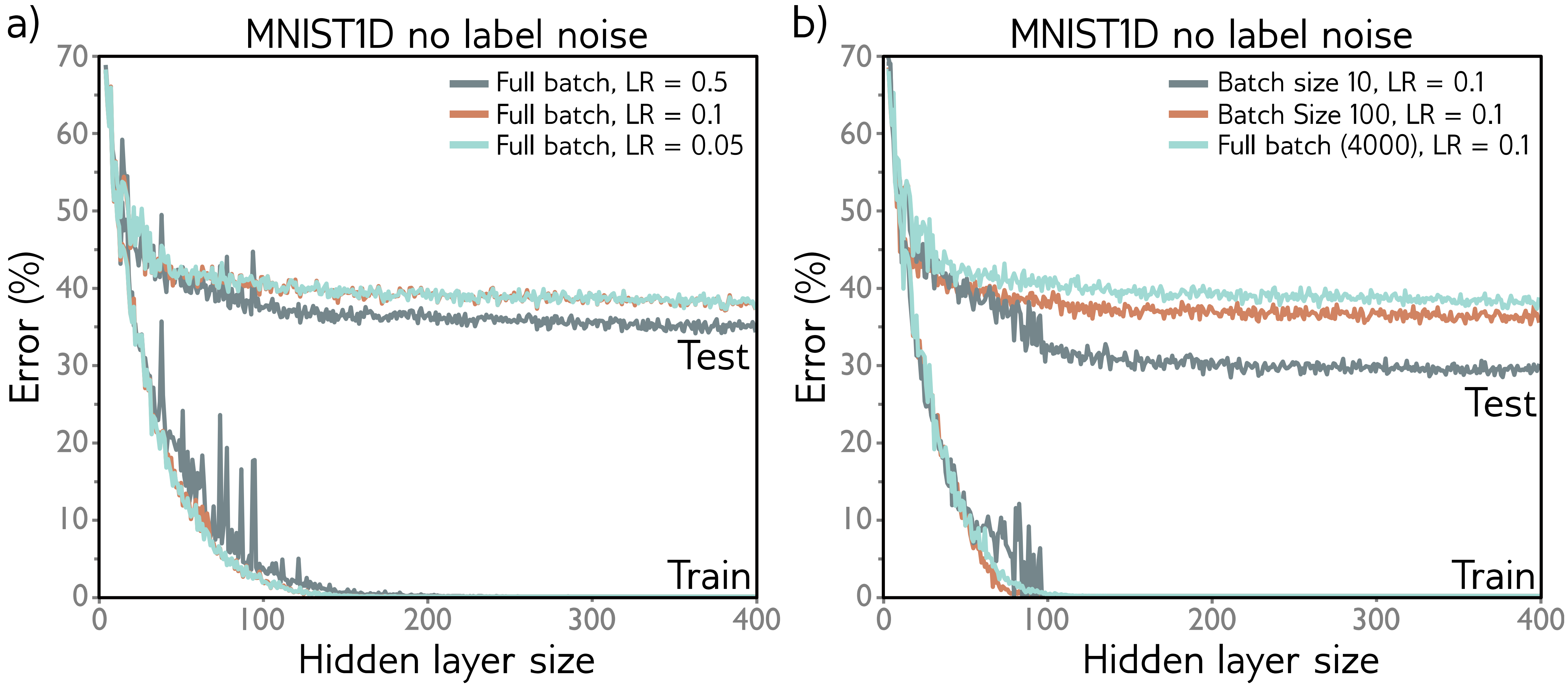
Global minimum

Implicit regularization term from gradient descent penalizes the squared gradient magnitude.



Additional implicit regularization from stochastic gradient descent penalizes the variance of the batch gradients

Global minimum can be different from that of the original loss function $L[\varphi]$



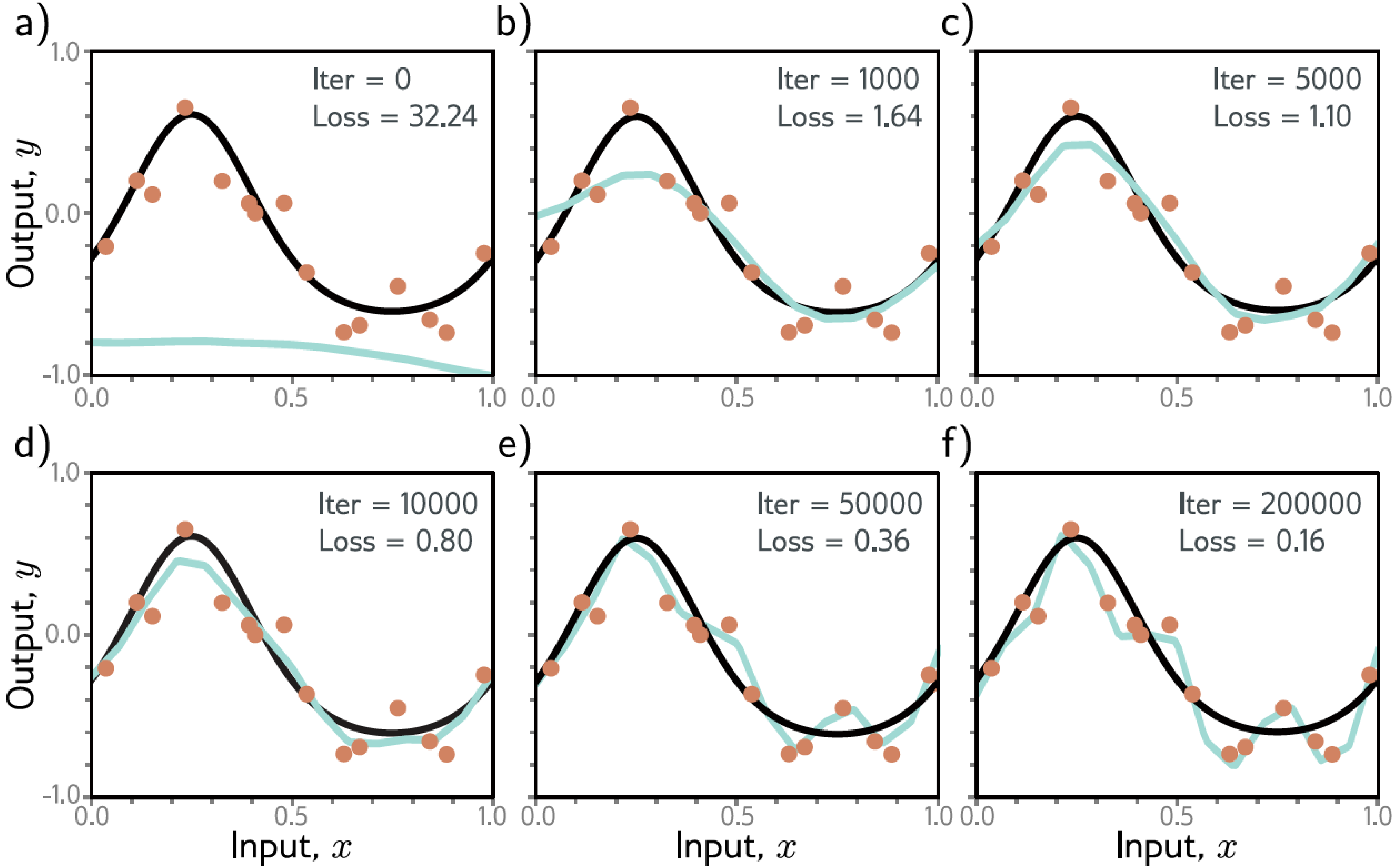
Effect of learning rate (LR) and batch size for 4000 training and 4000 test examples from MNIST-1D for a NN with two hidden layers

Generally, performance is:

- best for larger learning rates
- best with smaller batches

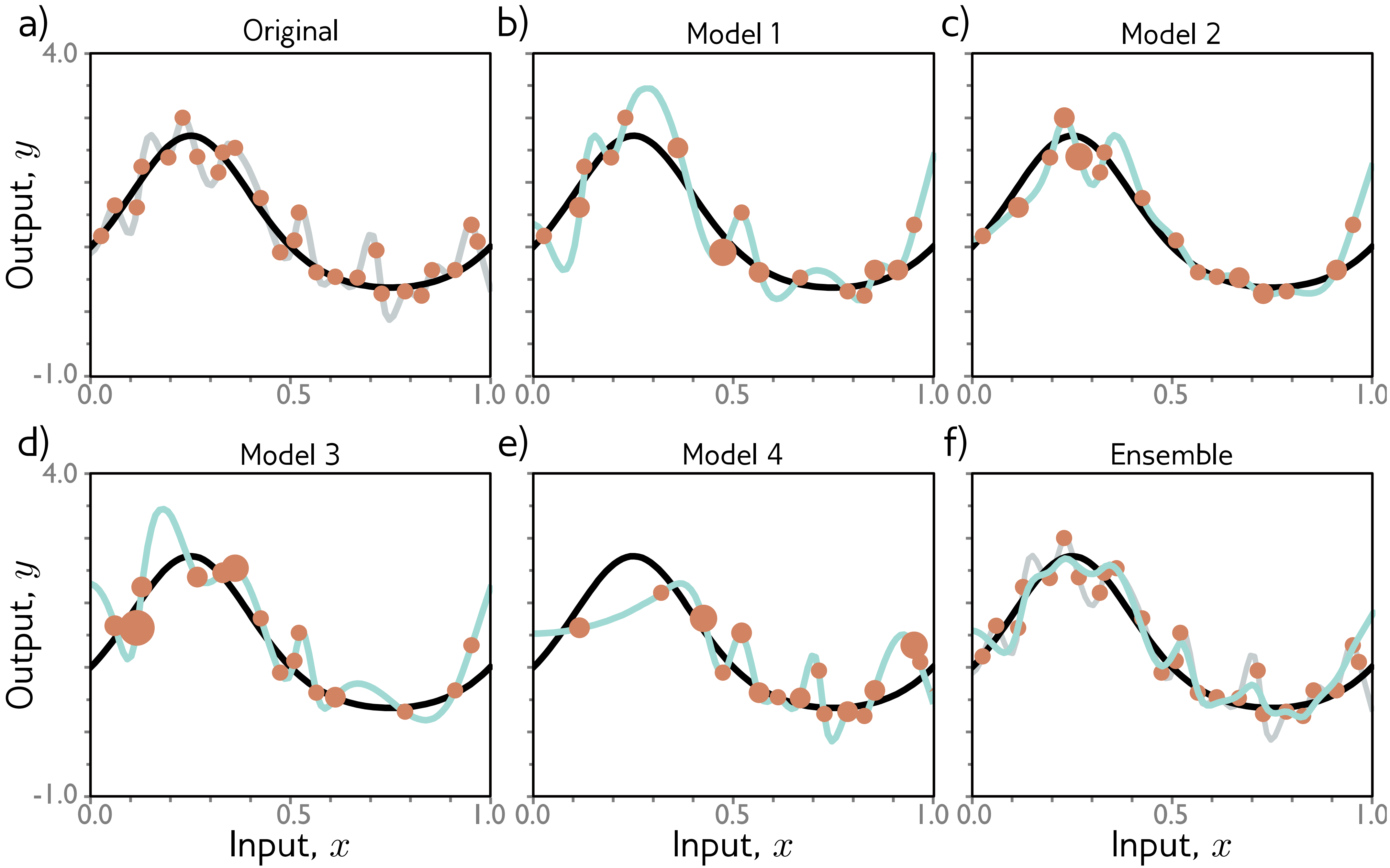
Early stopping

- If we stop training early, weights don't have time to overfit to noise
- Weights start small, don't have time to get large
- Reduces effective model complexity
- Known as early stopping
- Don't have to re-train


a)
b)
c)
d)
e)
f)
Iter = 0
Loss = 32.24
Iter = 1000
Loss = 1.64
Iter = 5000
Loss = 1.10
Iter = 10000
Loss = 0.80
Iter = 50000
Loss = 0.36
Iter = 200000
Loss = 0.16
Output, y
Input, x
1.0
0.0
-1.0
0.0
0.5
1.0

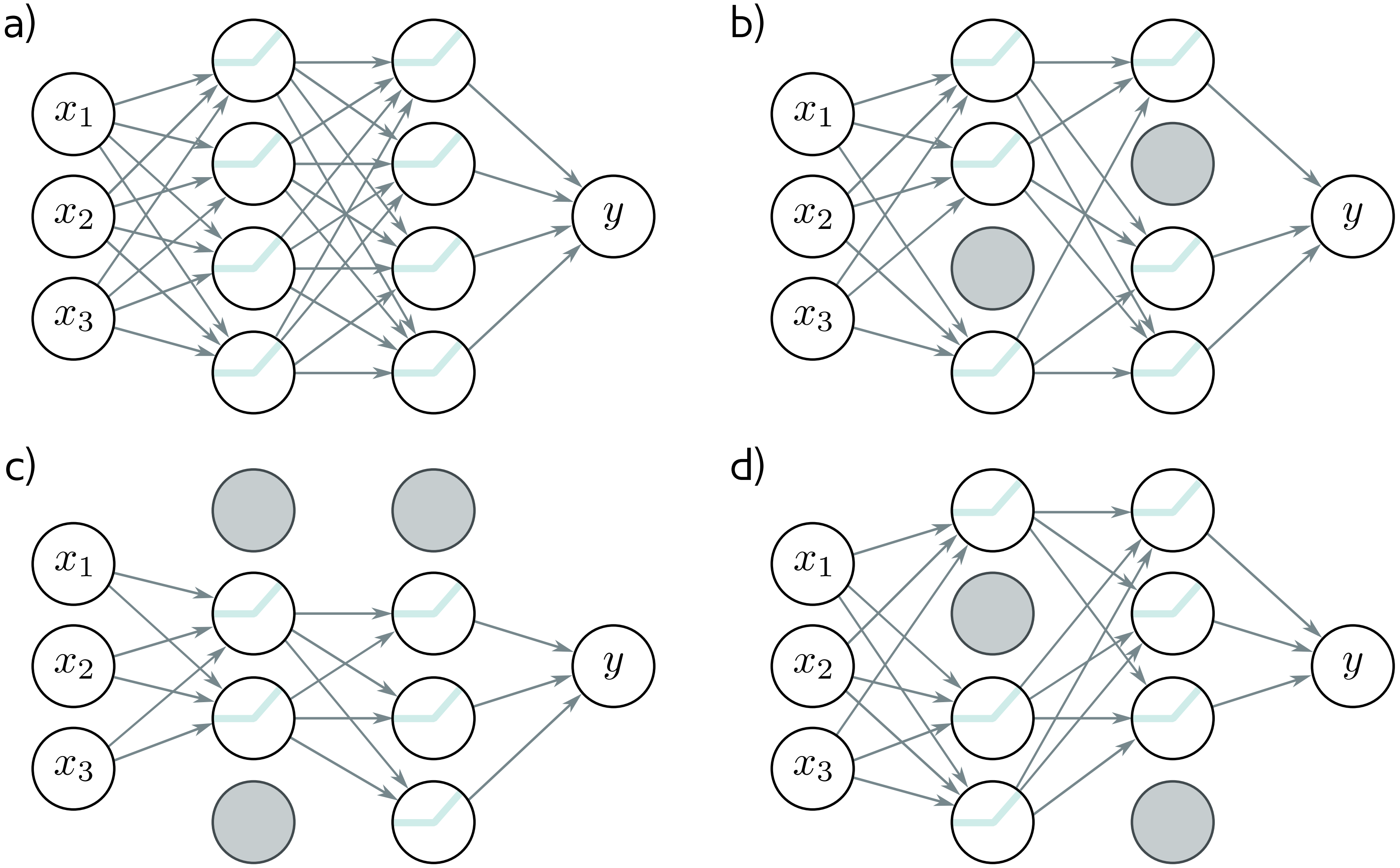
Ensembling

- Average together the predictions of several models – an ensemble
- Can take mean or median
- Different initializations / different models
- Different subsets of the data resampled with replacements – bagging
- The assumption is that model errors are independent and will cancel out



a)
Original
b)
Model 1
c)
Model 2
d)
Model 3
e)
Model 4
f)
Ensemble
4.0
-1.0
Output, y
0.0
0.5
1.0
Input, x

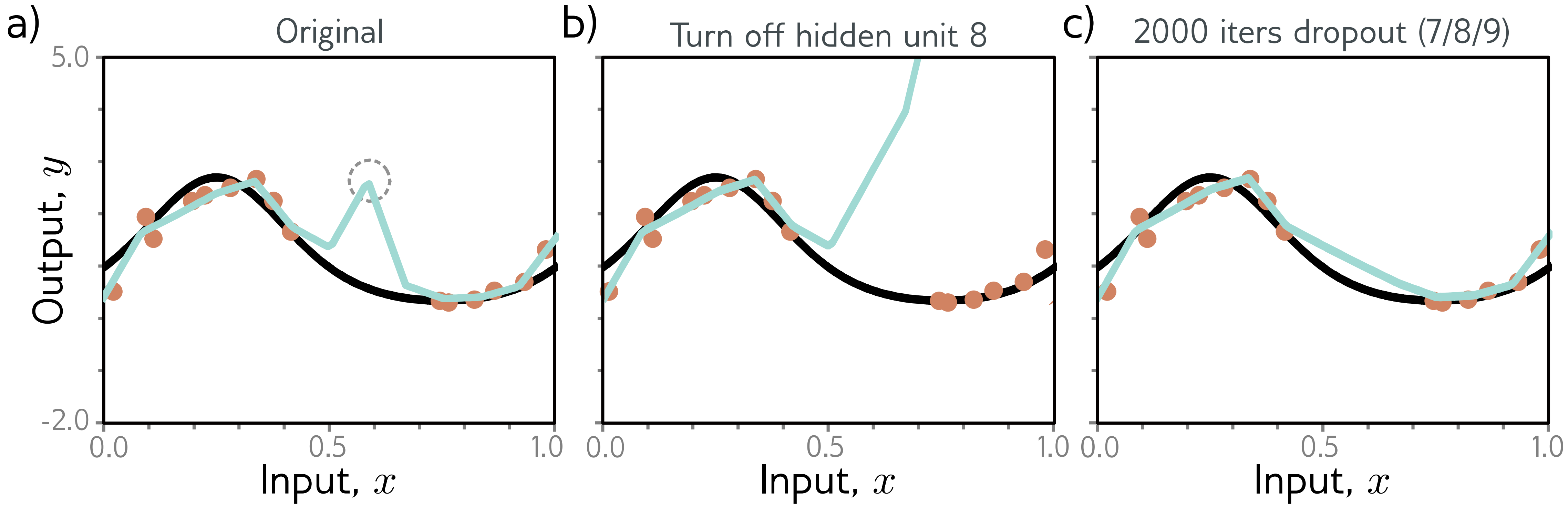
Dropout



Training:
- Drop units with probability p

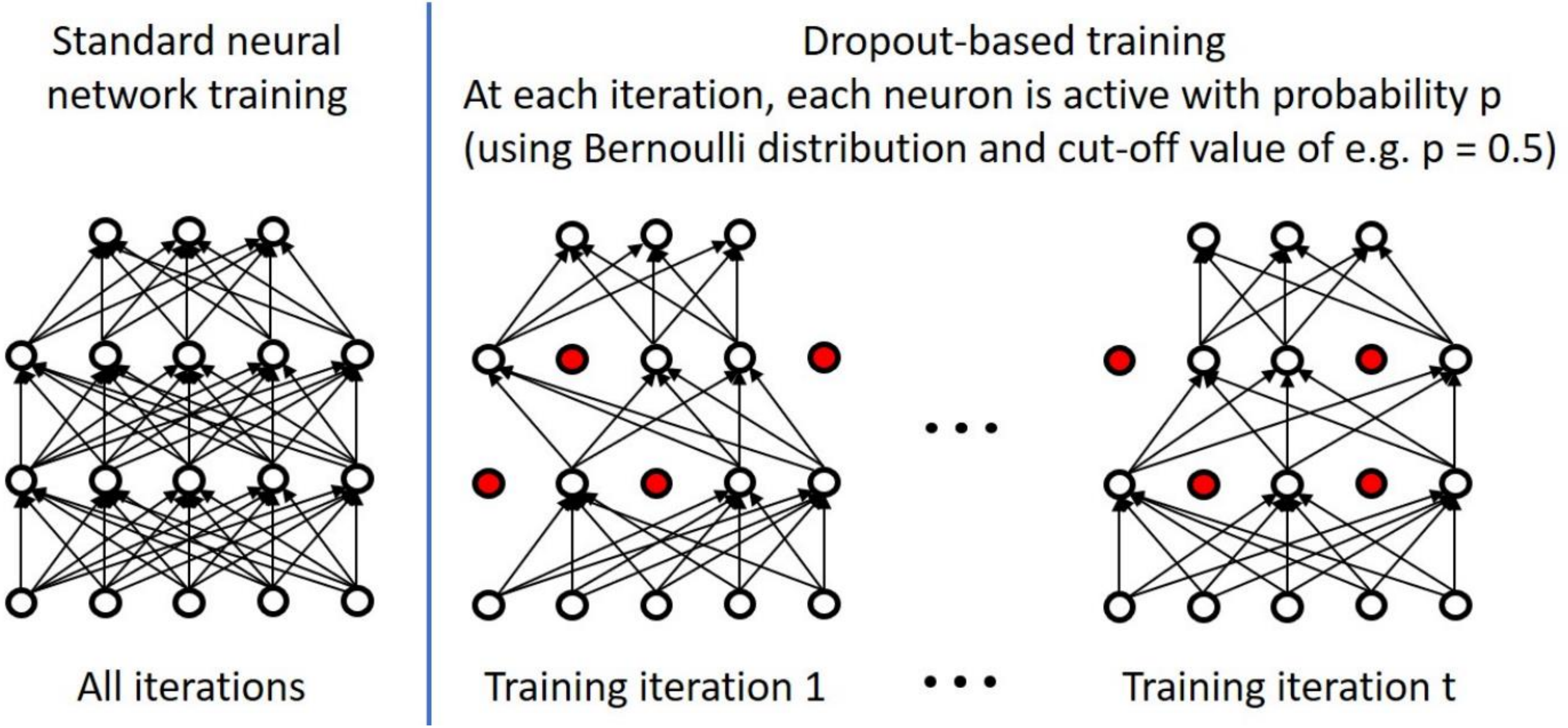
Testing:
- Do not drop any unit
- Scale the output of each unit with the value p

Dropout

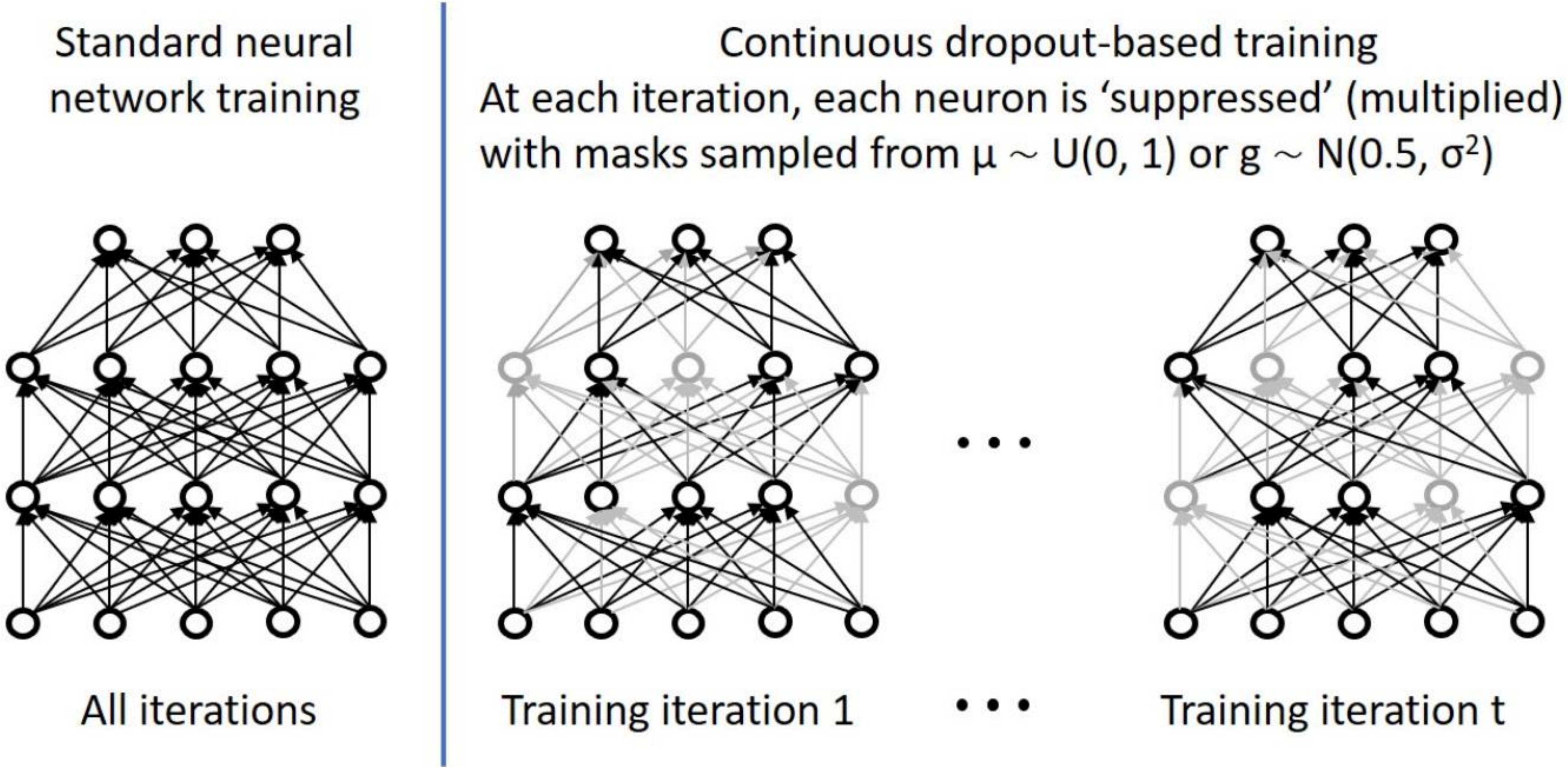


Can eliminate sudden changes in function that are far from data and don't contribute to training loss

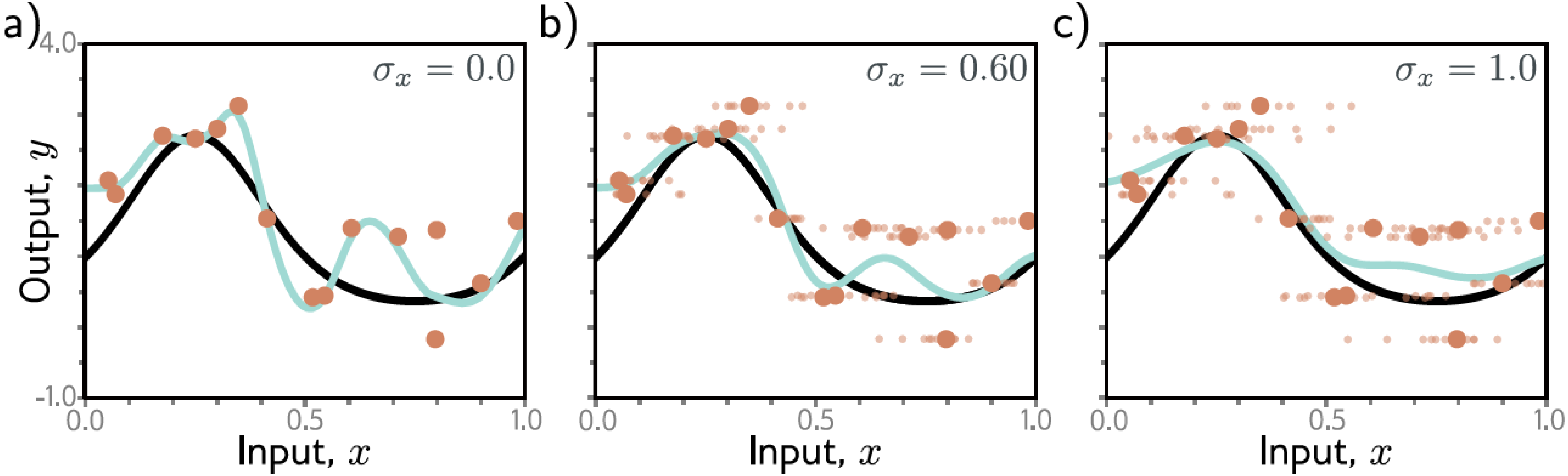
Dropout in iterative optimization: A probabilistic process to 'augment' the training set during iterative training and increase invariance:



Continuous Dropout-based iterative optimization:



Adding noise



- to inputs
- to weights
- to outputs (labels)

Bayesian approaches

- There are many parameters compatible with the data
- Can find a probability distribution over them

Prior info about parameters

$$Pr(\phi|\{\mathbf{x}_i, \mathbf{y}_i\}) = \frac{\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i, \phi) Pr(\phi)}{\int \prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i, \phi) Pr(\phi) d\phi}$$

Bayesian approaches

- There are many parameters compatible with the data
- Can find a probability distribution over them

Prior info about parameters

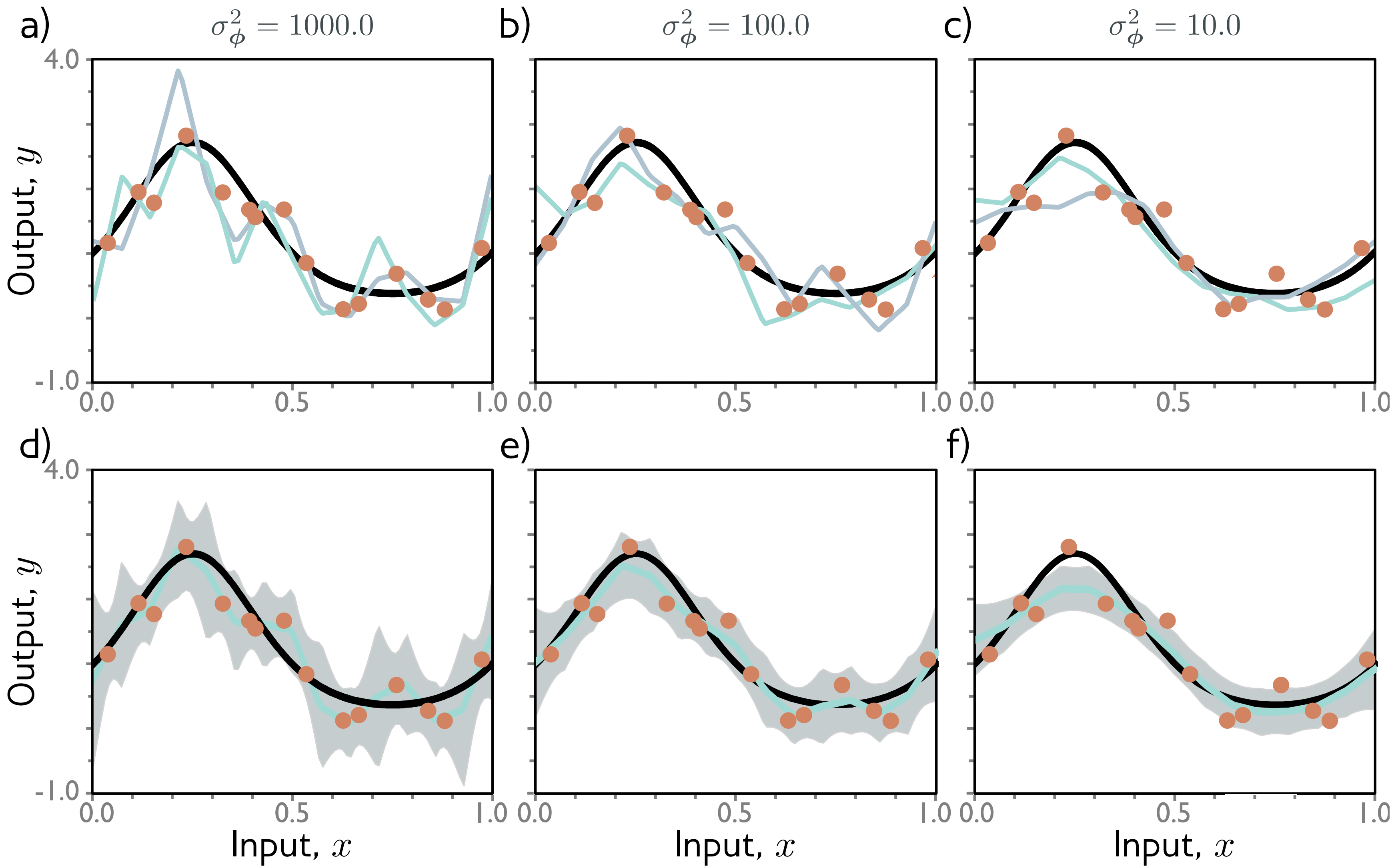
$$Pr(\phi|\{\mathbf{x}_i, \mathbf{y}_i\}) = \frac{\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i, \phi) Pr(\phi)}{\int \prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i, \phi) Pr(\phi) d\phi}$$

- Take all possible parameters into account when make prediction

$$Pr(\mathbf{y}|\mathbf{x}, \{\mathbf{x}_i, \mathbf{y}_i\}) = \int Pr(\mathbf{y}|\mathbf{x}, \phi) Pr(\phi|\{\mathbf{x}_i, \mathbf{y}_i\}) d\phi$$

Infinite weighted ensemble

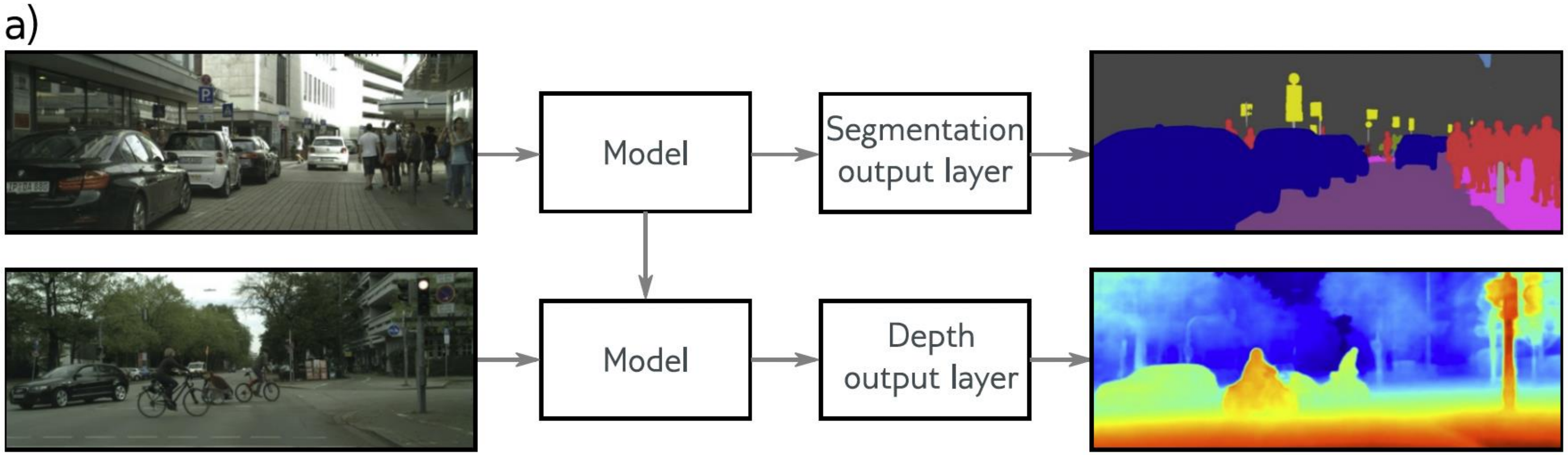
Bayesian approaches



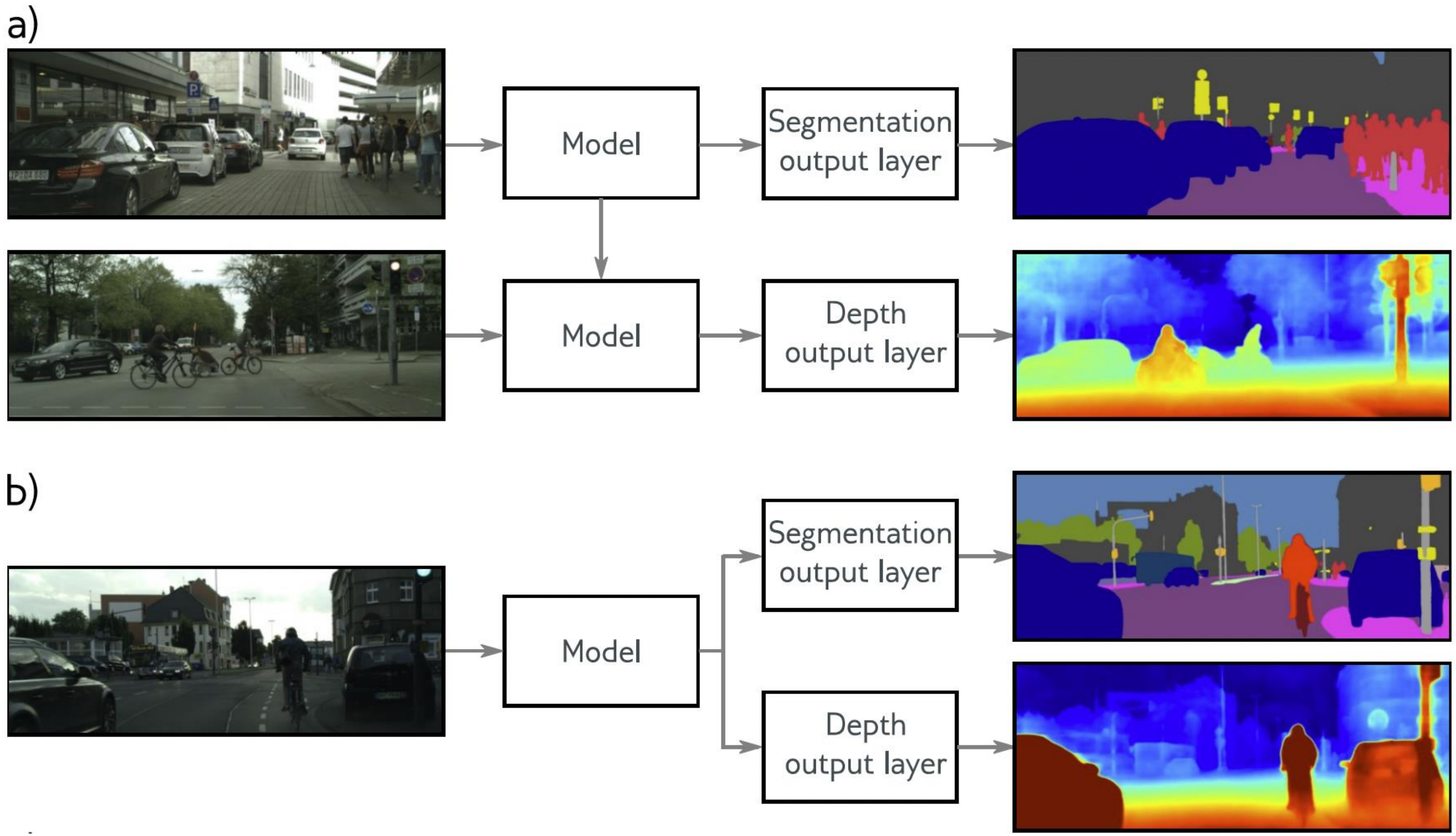
a–c) Two sets of parameters (cyan and gray curves) sampled from the posterior using normally distributed priors with mean zero and three variances.

d–f) Inference proceeds by taking a weighted sum over all possible parameter values where the weights are the posterior probabilities. This produces both a prediction of the mean (cyan curves) and the associated uncertainty (gray region is two standard deviations).

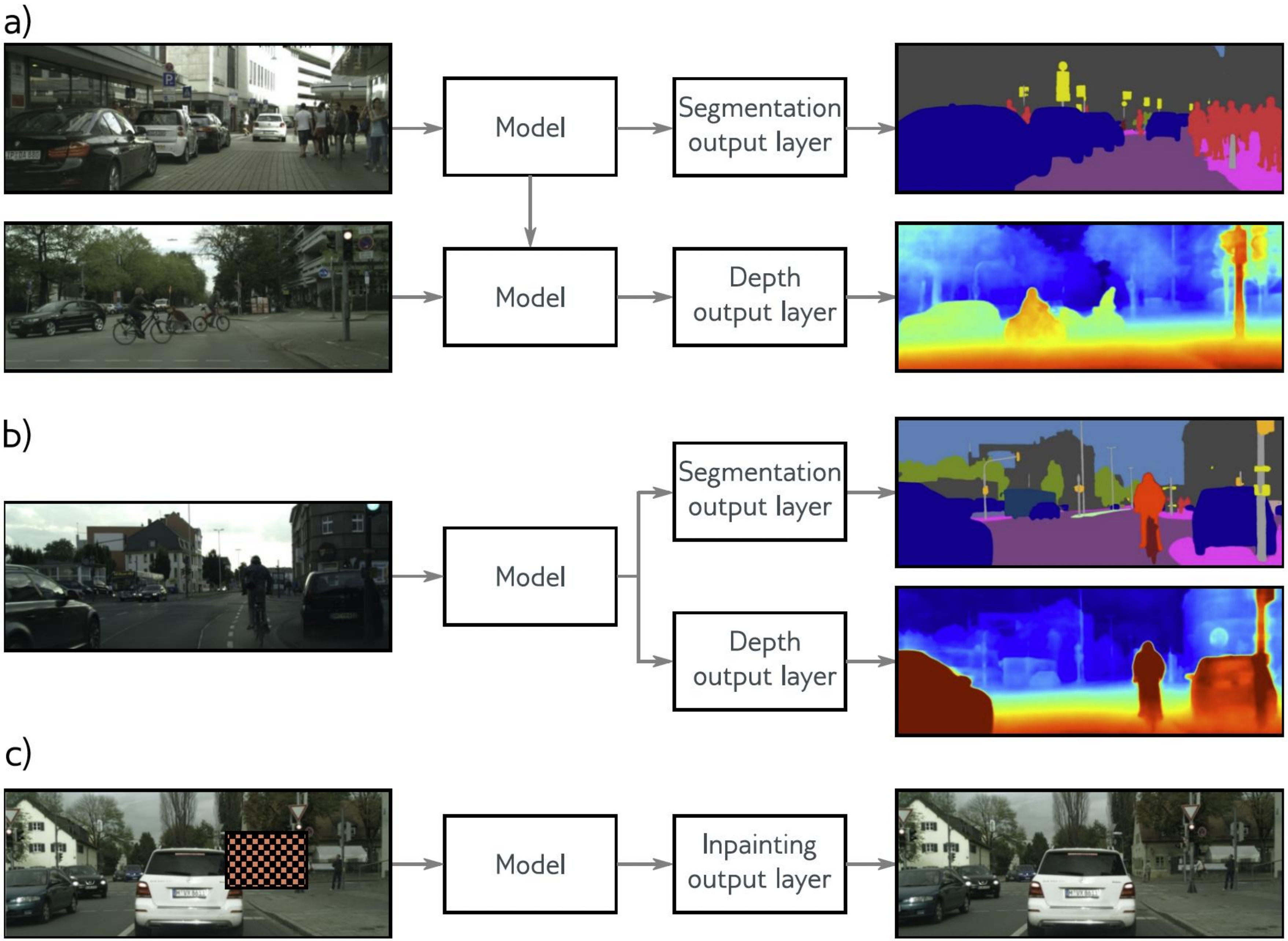
- Transfer learning



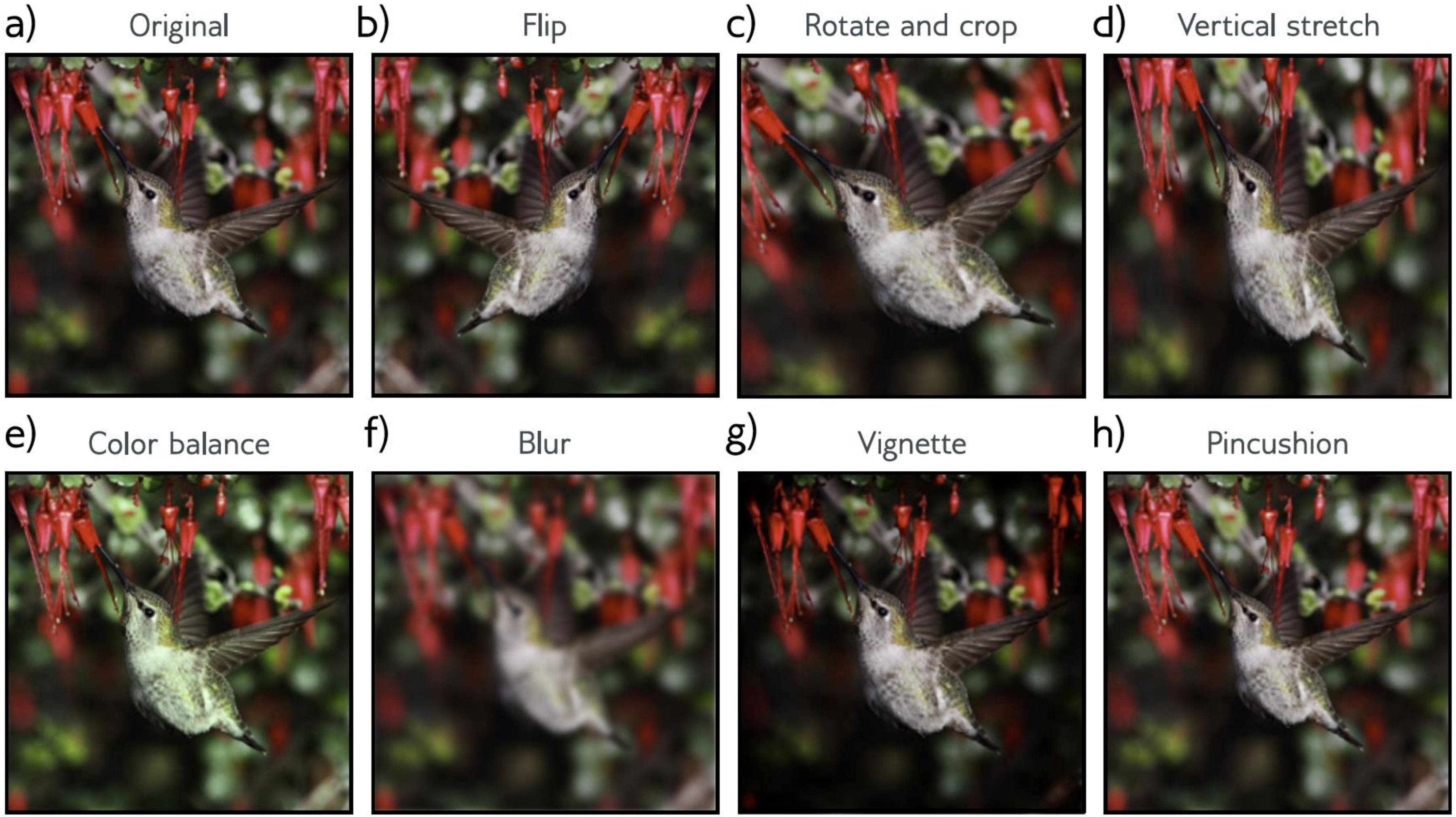
- Transfer learning
- Multi-task learning



- Transfer learning
- Multi-task learning
- Self-supervised learning



Data augmentation



Regularization overview