BOSTON UNIVERSITY

Measuring Performance

DL4DS – Spring 2026

Where we are

=== Foundational Concepts ===

- ✓ 04 -- Shallow networks and their representation capacity
- ✓ 05 -- Deep networks and depth efficiency
- ✓ 06 -- Loss function in terms of maximizing likelihoods
- ✓ 07 – Fitting models with different optimizers
- ✓ 08 – Gradients on deep models and backpropagation
- ✓ 09 – Initialization to avoid vanishing and exploding weights & gradients
- 10 – Measuring performance, test sets, overfitting and double descent
- 11 – Regularization to improve fitting on test sets and unseen data

=== Network Architectures and Applications ===

- 12 – Convolutional Networks
- 13 – Residual Networks
- 14 – Transformers
- Large Language and other Foundational Models
- Generative Models
- Graph Neural Networks
- …

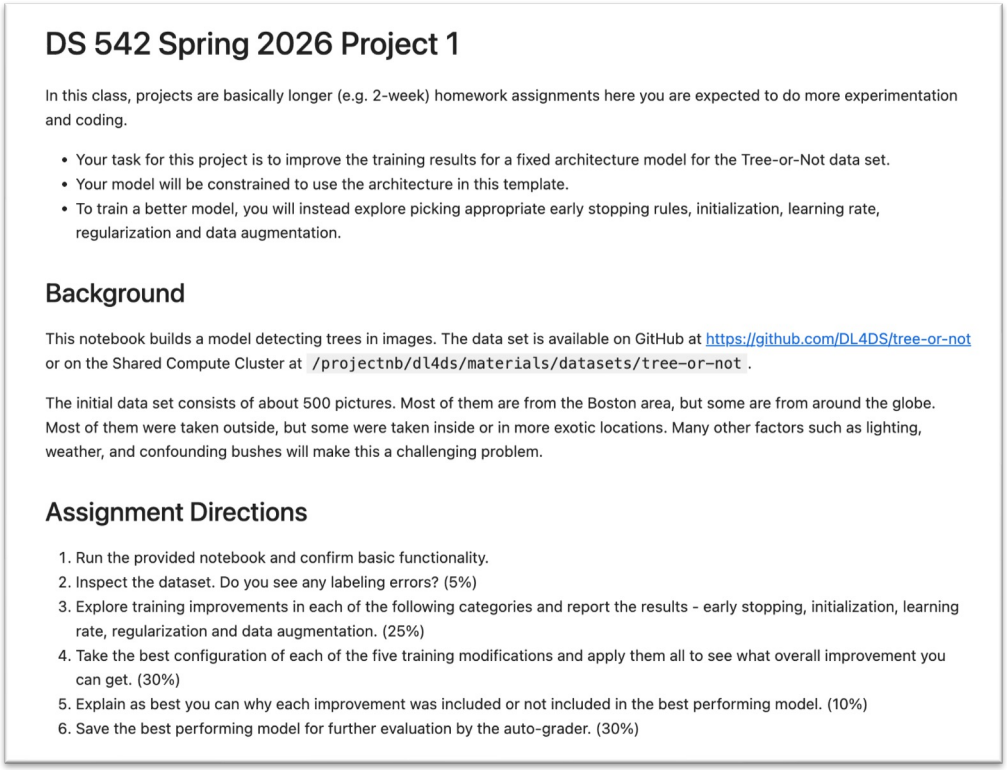
Measuring performance

- Project 1 Overview
- MNIST1D dataset model and performance
- Noise, bias, and variance
- Reducing variance
- Reducing bias & bias-variance trade-off
- Double descent
- Curse of dimensionality & weird properties of high dimensional space
- Choosing hyperparameters

Project 1 – Improving Model Training

~2-week assignment

- https://github.com/DL4DS/sp2026/blob/main/static_files/assignments/project1.ipynb



DS 542 Spring 2026 Project 1

In this class, projects are basically longer (e.g. 2-week) homework assignments here you are expected to do more experimentation and coding.

- Your task for this project is to improve the training results for a fixed architecture model for the Tree-or-Not data set.
- Your model will be constrained to use the architecture in this template.
- To train a better model, you will instead explore picking appropriate early stopping rules, initialization, learning rate, regularization and data augmentation.

Background

This notebook builds a model detecting trees in images. The data set is available on GitHub at https://github.com/DL4DS/tree-or-not or on the Shared Compute Cluster at `/projectnb/dl4ds/materials/datasets/tree-or-not`.

The initial data set consists of about 500 pictures. Most of them are from the Boston area, but some are from around the globe. Most of them were taken outside, but some were taken inside or in more exotic locations. Many other factors such as lighting, weather, and confounding bushes will make this a challenging problem.

Assignment Directions

1. Run the provided notebook and confirm basic functionality.
2. Inspect the dataset. Do you see any labeling errors? (5%)
3. Explore training improvements in each of the following categories and report the results - early stopping, initialization, learning rate, regularization and data augmentation. (25%)
4. Take the best configuration of each of the five training modifications and apply them all to see what overall improvement you can get. (30%)
5. Explain as best you can why each improvement was included or not included in the best performing model. (10%)
6. Save the best performing model for further evaluation by the auto-grader. (30%)

Measuring performance

- MNIST1D dataset model and performance
- Noise, bias, and variance
- Reducing variance
- Reducing bias & bias-variance trade-off
- Double descent
- Curse of dimensionality & weird properties of high dimensional space
- Choosing hyperparameters

MNIST1D

Scaling down Deep Learning

Sam Greydanus [1]

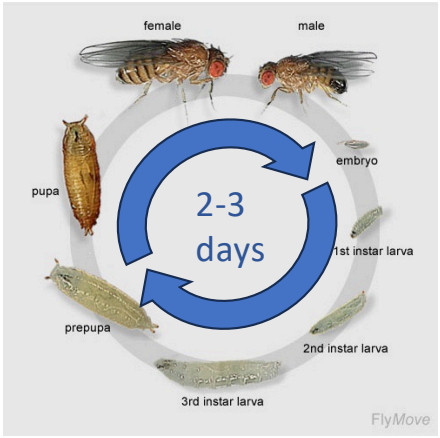
"A large number of deep learning innovations including dropout, Adam, convolutional networks, generative adversarial networks, and variational autoencoders began life as MNIST experiments. Once these innovations proved themselves on small-scale experiments, scientists found ways to scale them to larger and more impactful applications."

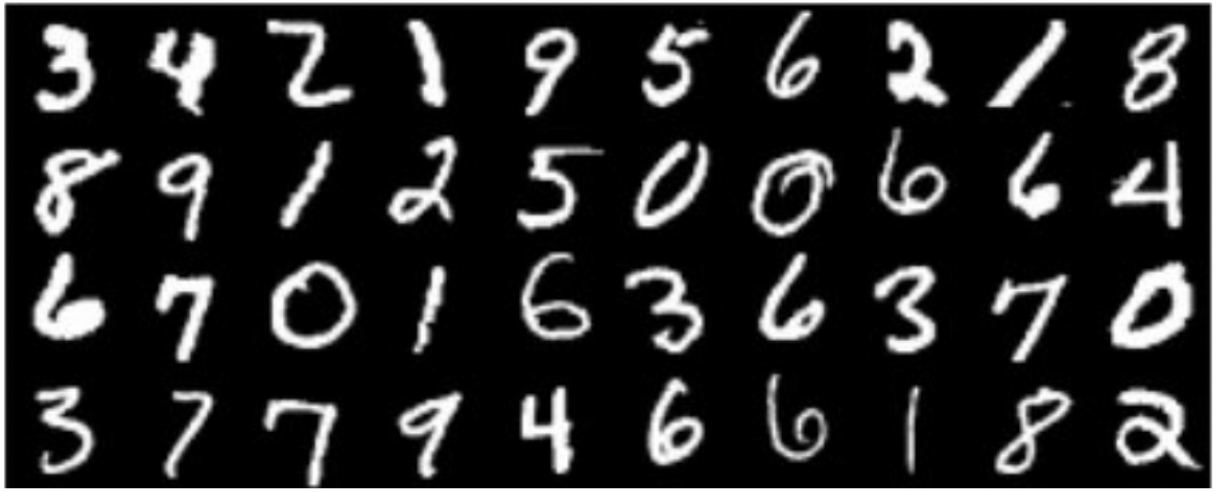
S. Greydanus, "Scaling down Deep Learning." arXiv, Dec. 04, 2020. doi: 10.48550/arXiv.2011.14439.

https://github.com/greydanus/mnist1d

MNIST Dataset

- 28x28x1 grayscale images
- 60K Training, 10K Test
- “Is to Deep Learning what fruit flies are to genetics research”





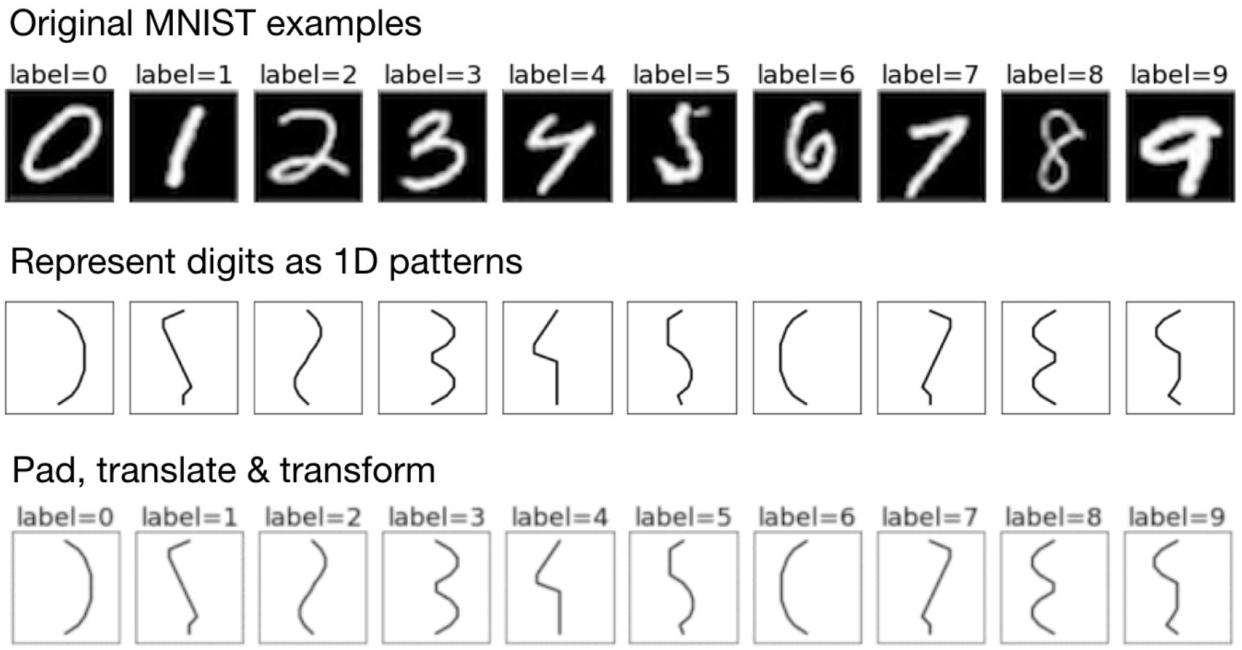
But poorly differentiates model performance:

Model Type	Accuracy
Logistic Regression	94%
MLP	99+%
CNN	99+%

MNIST 1D Dataset

Fixed, 1-D, length-12 templates for each of 10 digit classes

Generate dataset by programmatically applying 6 parametric transformations.



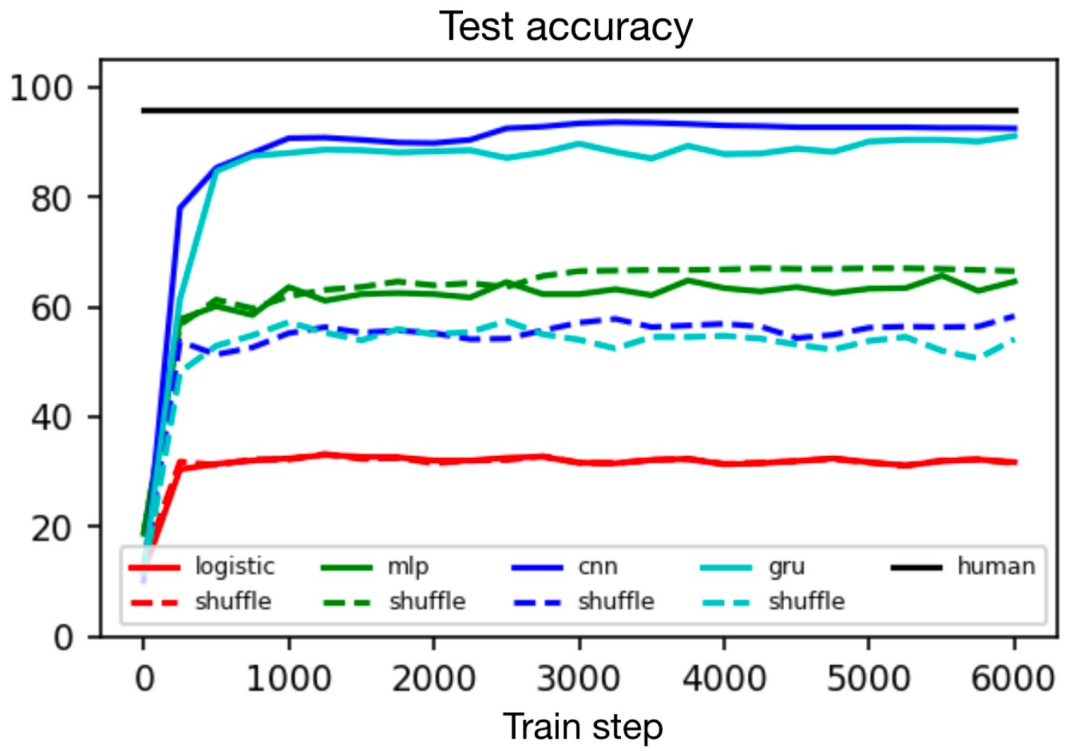
E.g. pad, shear, translate, correlated noise, i.i.d. noise, interpolation.

See https://github.com/greydanus/mnist1d/blob/master/notebooks/building-mnist1d.ipynb
https://colab.research.google.com/github/greydanus/mnist1d/blob/master/notebooks/building-mnist1d.ipynb

MNIST 1D

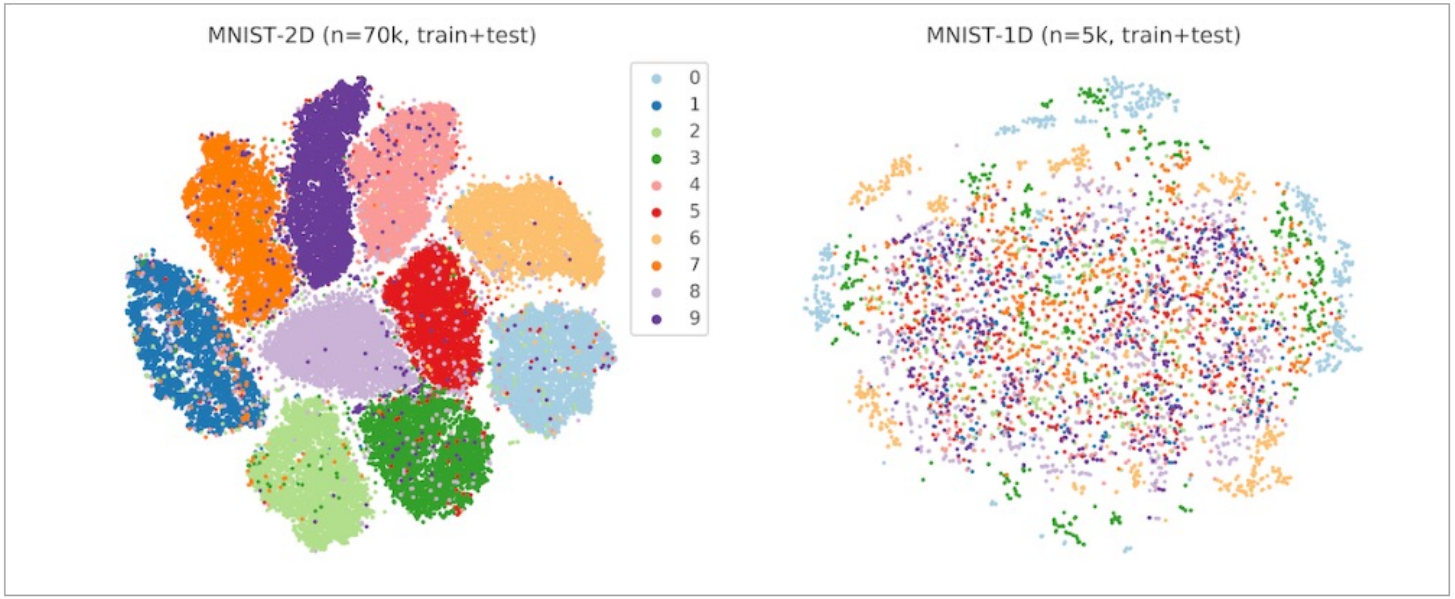
Differentiates performance of different model types much more than MNIST

Shuffle: dataset was permuted along the spatial dimension. This 'shuffled' version measured each of the models' performances in the absence of local spatial structure



Dataset	Logistic regression	Fully connected model	Convolutional model	GRU model	Human expert
MNIST	94 ± 0.5	> 99	> 99	> 99	> 99
MNIST-1D	32 ± 1	68 ± 2	94 ± 2	91 ± 2	96 ± 1
MNIST-1D (shuffled)	32 ± 1	68 ± 2	56 ± 2	57 ± 2	$\approx 30 \pm 10$

Visualizing MNIST and MNIST-1D with tSNE

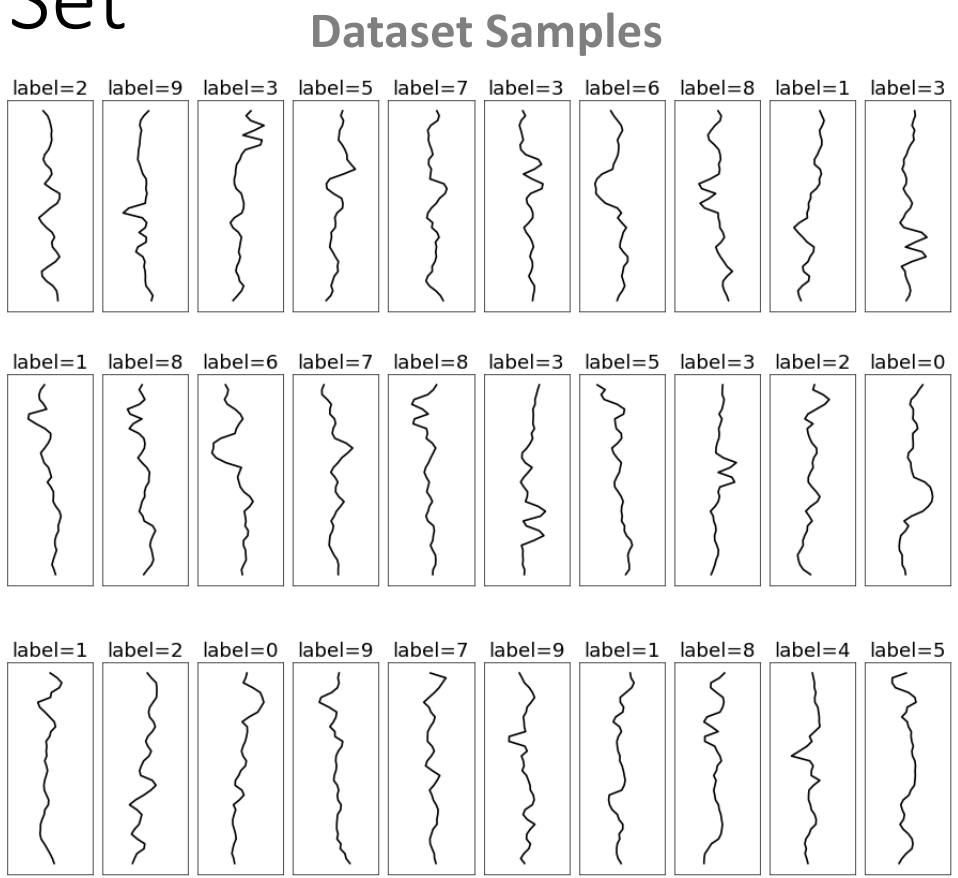


Visualizing the MNIST and MNIST-1D datasets with tSNE. The well-defined clusters in the MNIST plot indicate that the majority of the examples are separable via a kNN classifier in pixel space. The MNIST-1D plot, meanwhile, reveals a lack of well-defined clusters which suggests that learning a nonlinear representation of the data is much more important to achieve successful classification. Thanks to Dmitry Kobak for making this plot.

https://twitter.com/hippopedoid

MNIST1D Train and Test Set

- 1D, Length 40 samples
- 4,000 training samples
- 1,000 test samples (80/20 split)



Network

- 40 inputs
- 10 outputs
- Two hidden layers
  - 100 hidden units each
- SGD with batch size 100, learning rate 0.1
- Say 50 epochs, how many steps total?

```
model = torch.nn.Sequential(
    torch.nn.Linear(40, 100),
    torch.nn.ReLU(),
    torch.nn.Linear(100, 100),
    torch.nn.ReLU(),
    torch.nn.Linear(100, 10))
```

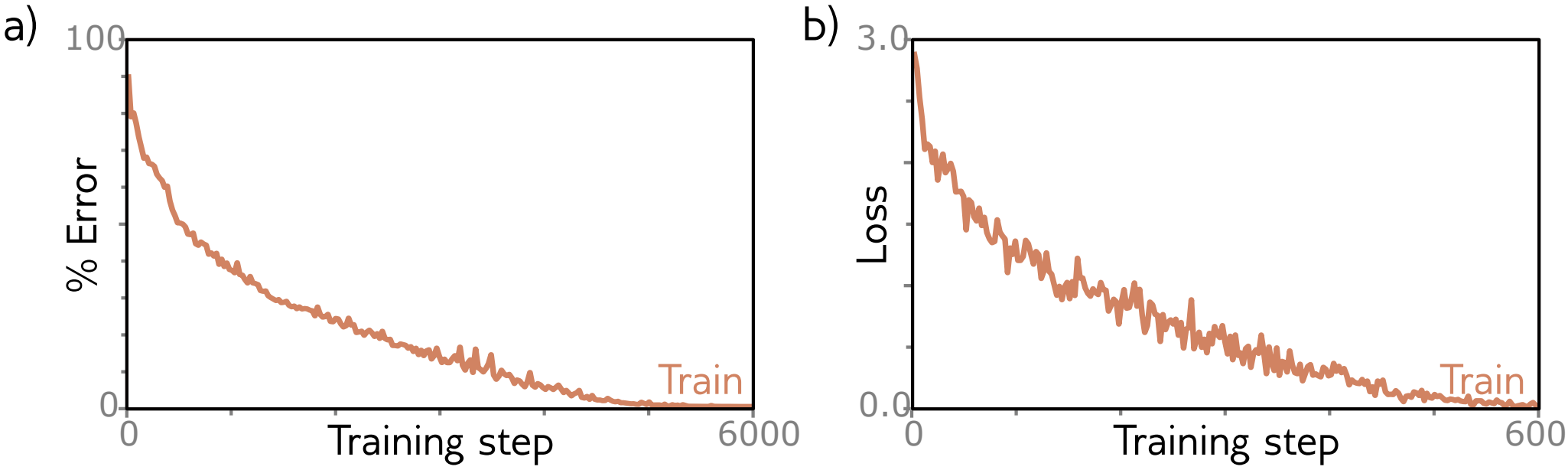
```
from torchinfo import summary
summary(model, input_size(100,40)) # batch: 100

==========================================
Layer (type:depth-idx) Output Shape  Param #
==========================================
Sequential             [1, 10]       -
├─Linear: 1-1          [1, 100]      4,100
├─ReLU: 1-2            [1, 100]      -
├─Linear: 1-3          [1, 100]      10,100
├─ReLU: 1-4            [1, 100]      -
├─Linear: 1-5          [1, 10]       1,010
==========================================
Total params: 15,210
Trainable params: 15,210
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 1.52
==========================================
Input size (MB): 0.02
Forward/backward pass size (MB): 0.17
Params size (MB): 0.06
Estimated Total Size (MB): 0.24
==========================================
```

```
# choose cross entropy loss function
loss_function = torch.nn.CrossEntropyLoss()

# construct SGD optimizer and initialize learning rate and momentum
optimizer = torch.optim.SGD(model.parameters(), lr = 0.05, momentum = 0.9)

# object that decreases learning rate by half every 10 epochs
scheduler = StepLR(optimizer, step_size=10, gamma=0.5)

# load the data into a class that creates the batches
data_loader = DataLoader(TensorDataset(x_train,y_train), batch_size=100, shuffle=True)
```

...

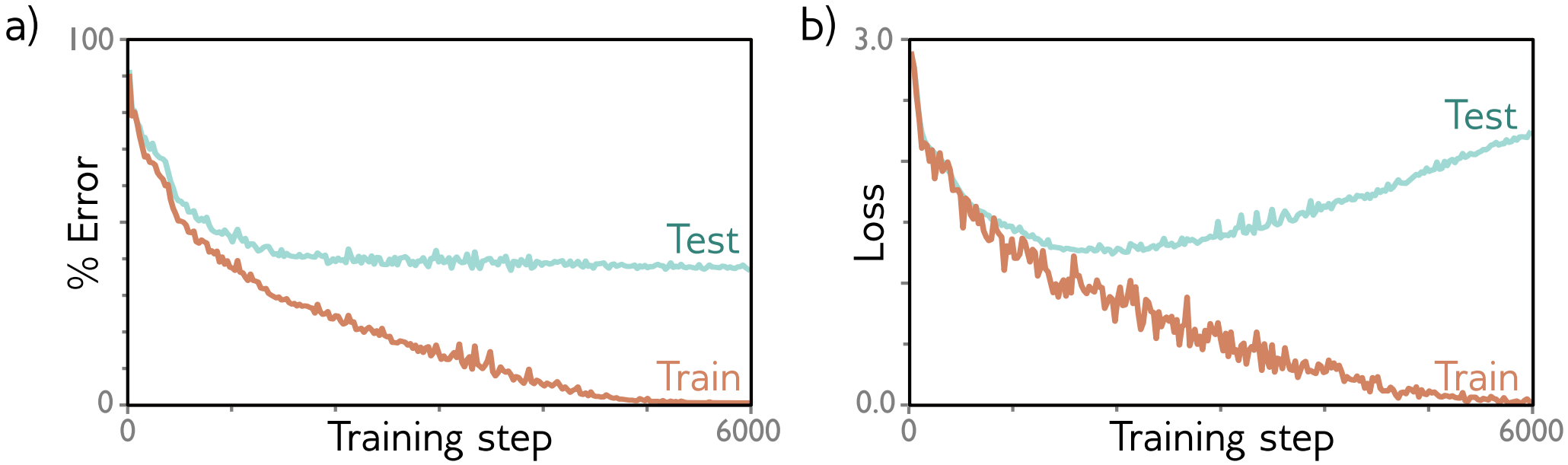
```
# inference – just choose the max
pred_train = model(x_train)
pred_test = model(x_test)
_, predicted_train_class = torch.max(pred_train.data, 1)
_, predicted_test_class = torch.max(pred_test.data, 1)
```

Results



You can try this yourself in [UDL Notebook 8.1 MNIST_1D_Performance]

Need to use separate test data



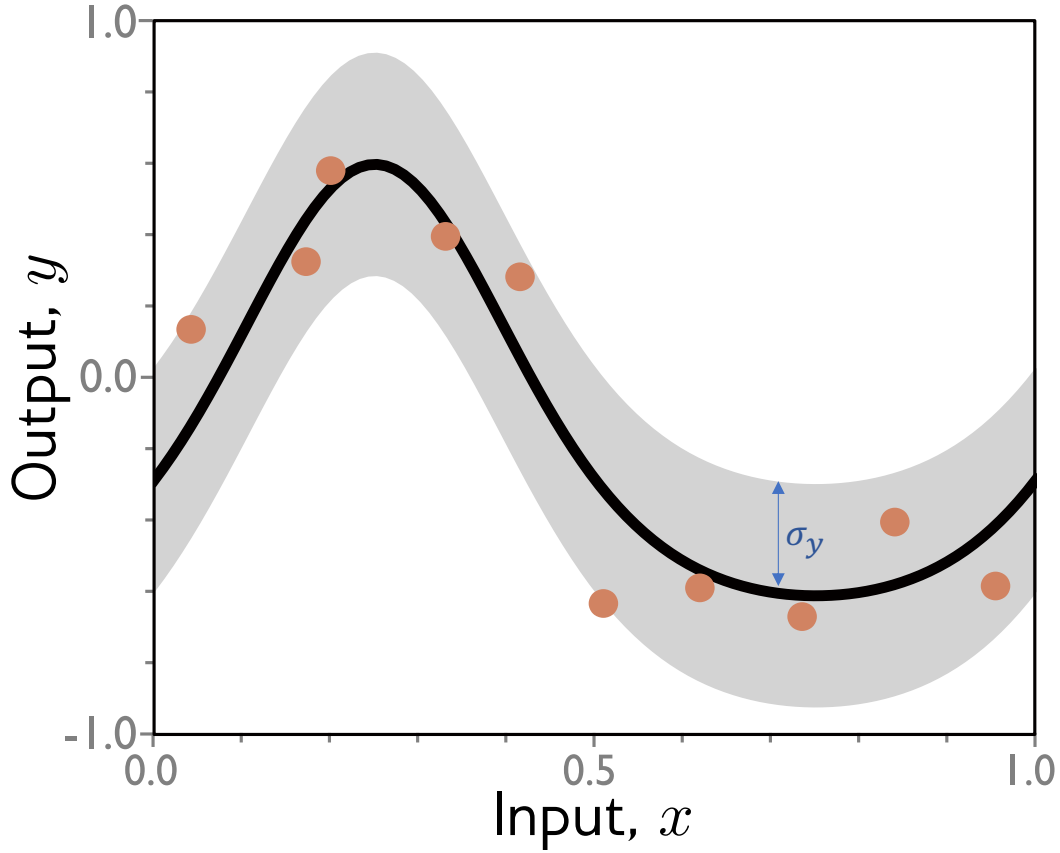
The model has not generalized well to the new data

Measuring performance

- MNIST1D dataset model and performance
- Noise, bias, and variance
- Reducing variance
- Reducing bias & bias-variance trade-off
- Double descent
- Curse of dimensionality & weird properties of high dimensional space
- Choosing hyperparameters

Regression example with Toy Model

Dataset



”True” function:

$$y = e^{\sin(2\pi x)}$$

Add small uniform noise to x:

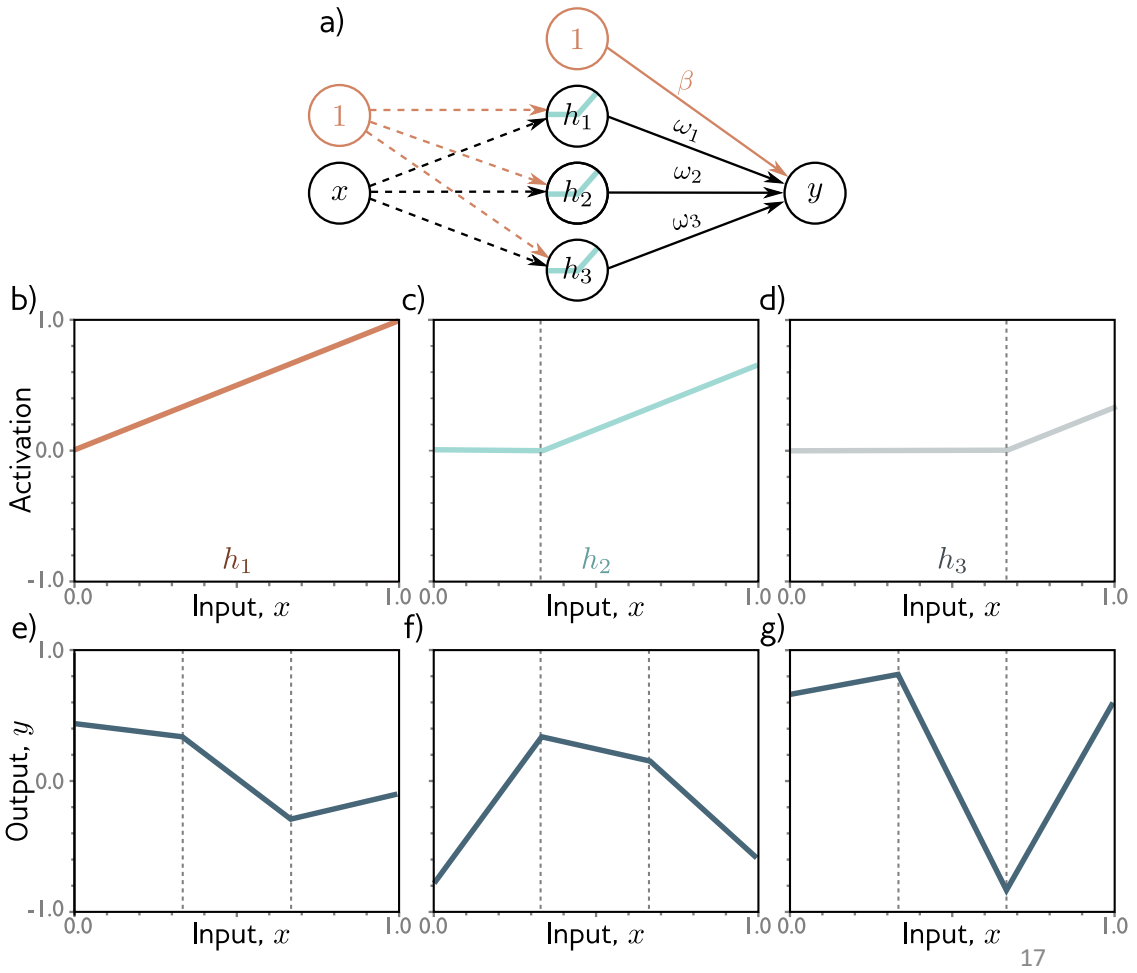
$$x = x + \mathcal{U}(\pm 1/\text{num\_data})$$

Add small Gaussian noise to y:

$$y = y + \mathcal{N}(0, \sigma_y)$$

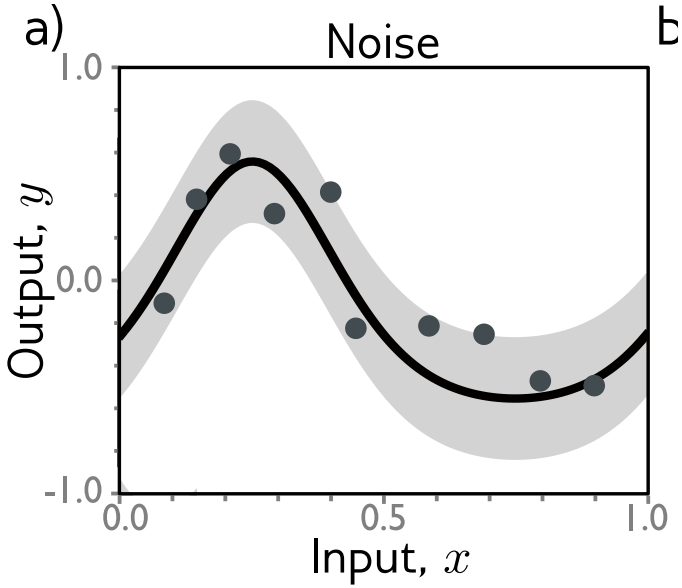
Toy model

- D hidden units
- First layer fixed so "joints" divide interval evenly, e.g. $0, 1/D, 2/D, \ldots, (D\text{-}1)/D$
- Second layer trained
- But… now linear in **h**
  - so convex cost function
  - can find best soln in closed-form
- A piecewise linear model with D regions.

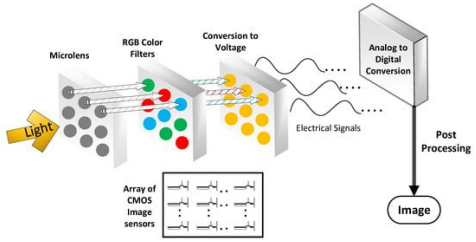


Three possible sources of error: *noise*, *bias* and *variance*

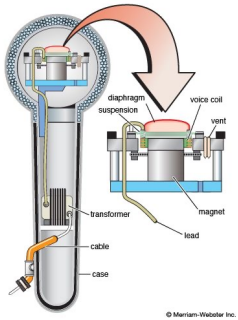
Noise, bias, and variance



- Genuine stochastic nature of the underlying model
- Noise in measurements, e.g. from sensors



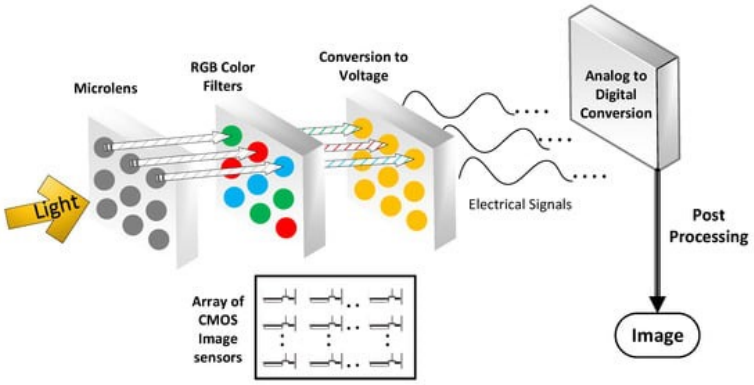
https://images.app.goo.gl/2PuBhaFpfdL9Pyjb8



https://images.app.goo.gl/CMDaXSCdX4pqN8Yx7

- Some variables not observed
- Data mislabeled

Noise in Digital Image Capture



https://images.app.goo.gl/2PuBhaFpfdL9Pyjb8

A. **A. Photon Shot Noise (The Quantum Limit)**
   - **Source:** The quantum nature of light. Photons arrive at the sensor pixel independently and randomly.
   - **Distribution: Poisson**.
   - **Key Characteristic:** It is **signal-dependent**. The variance of the noise scales with the signal intensity. Brighter regions have more absolute noise (though a higher Signal-to-Noise Ratio).
   - **DL Relevance:** This is why simply adding Gaussian noise to normalized images during data augmentation is physically inaccurate. For low-light imaging tasks (e.g., night vision models), modeling Poisson noise is critical.

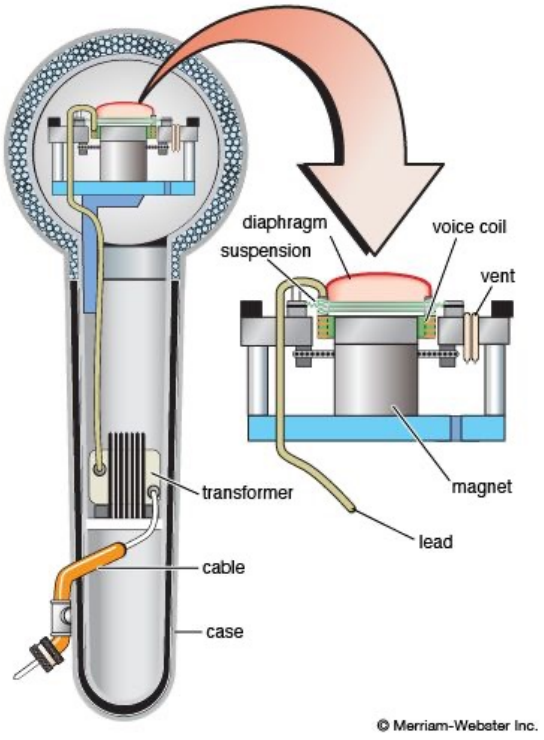
B. **B. Dark Current / Thermal Noise**
   - **Source:** Heat causes electrons to be thermally generated within the silicon substrate, even when no photons are hitting the sensor. These "phantom" electrons accumulate in the potential well.
   - **Distribution:** Roughly **Poisson** (for the generation rate) but often approximated as **Gaussian** in high-flux regimes.
   - **Key Characteristic:** It is essentially temperature-dependent and exposure-time dependent.
   - **DL Relevance:** This introduces a baseline "fog" or grain, particularly visible in the shadows of an image where the signal is low.

C. **C. Read Noise**
   - **Source:** Imperfections in the on-chip amplifiers (source follower) and the analog-to-digital converter (ADC) circuitry as the charge is read out.
   - **Distribution:** Generally **Gaussian**.
   - **Key Characteristic:** Independent of the signal level. It sets the "noise floor" of the sensor—the minimum signal required to distinguish a pixel from total darkness.

D. **D. Fixed Pattern Noise (FPN)**
   - **Source:** Variations in the manufacturing process cause some pixels to have higher dark currents ("hot pixels") or different sensitivities (gain) than their neighbors.
   - **Key Characteristic: Non-stochastic** (spatial). It is the same "noise" pattern in every frame taken at the same settings.
   - **DL Relevance:** Unlike random noise, a deep learning model *can* overfit to FPN if the training set comes from a single camera, leading to poor generalization on other cameras (a classic "camera identity" bias).

E. **E. Quantization Noise**
   - **Source:** Mapping the continuous voltage values to discrete integer levels (e.g., 0-255 for 8-bit).
   - **Distribution:** Uniform.
   - **Key Characteristic:** Loss of information is deterministic based on bit-depth.

Noise in Digital Audio Capture



https://images.app.goo.gl/CMDaXSCdX4pqN8Yx7

A. **A. Thermal (Johnson-Nyquist) Noise**
   - **Source:** The random Brownian motion of electrons in the microphone's resistor elements and the pre-amplifier.
   - **Key Characteristic:** It produces "white noise" (flat power spectral density). It is present in all electrical circuits and increases with temperature and resistance.
   - **DL Relevance:** This is the constant background "hiss" found in nearly all raw audio recordings.

B. **B. 1/f Noise (Flicker Noise)**
   - **Source:** Imperfections in the semiconductor materials of the amplifier or microphone capsule.
   - **Key Characteristic:** The power density is inversely proportional to frequency $(1/f)$.
   - **DL Relevance:** This dominates at low frequencies, often appearing as "rumble" or low-end drift that can confuse models trained on spectrograms, looking like low-frequency features.

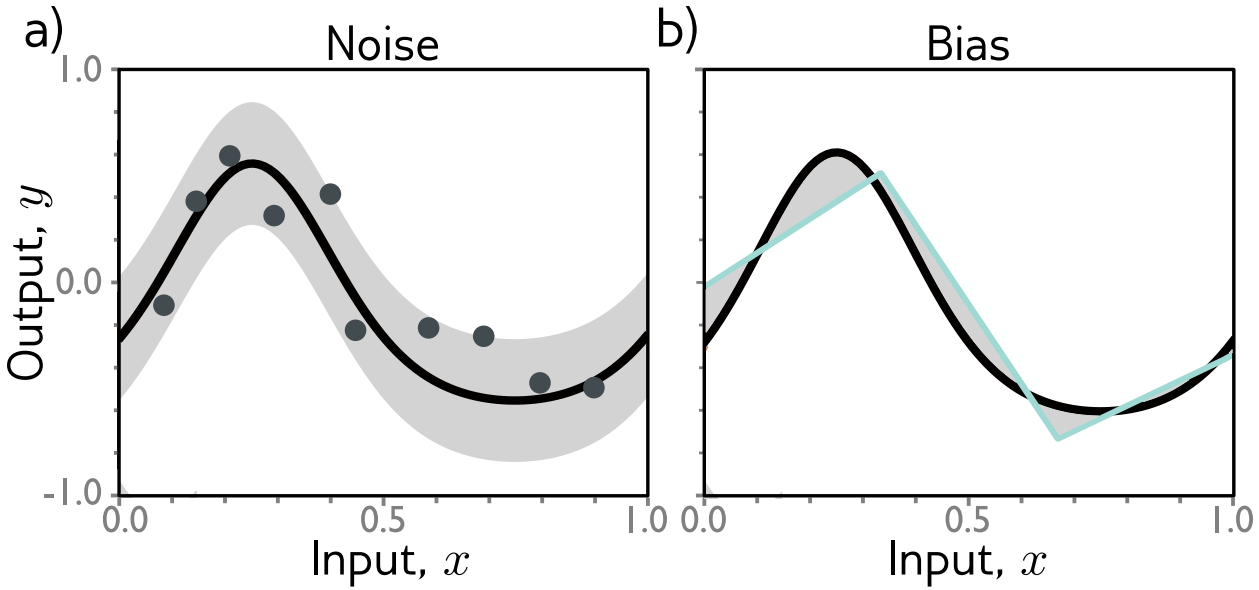
C. **C. Shot Noise (Electronic)**
   - **Source:** Similar to photon shot noise, but caused by the discrete nature of electrons crossing potential barriers (like p-n junctions) in the amplifier.
   - **Key Characteristic:** Appears as random popping or crackling in extreme cases, though usually just contributes to the broadband noise floor.

D. **D. Quantization Noise**
   - **Source:** The error introduced when converting analog voltage to discrete samples (e.g., 16-bit vs. 24-bit).
   - **DL Relevance:** In audio, aggressive quantization (low bit-depth) manifests as a correlation between the signal and the quantization error, perceived as "distortion" rather than hiss. This is why techniques like dithering (adding noise intentionally) are used to decorrelate the error.

Noise in Digital Image and Audio Capture

Noise Type	Domain	Source	Distribution	Signal Dependent?
Photon Shot	Image	Quantum nature of light	Poisson	Yes
Dark Current	Image	Heat in Silicon	Poisson/Gaussian	No (Temp/Time dependent)
Read Noise	Image	Electronics/Amplifiers	Gaussian	No
FPN	Image	Manufacturing defects	Constant (Spatial)	No
Thermal	Audio	Electron motion	Gaussian (White)	No
1/f (Flicker)	Audio	Material impurities	Pink Noise ($1/f$)	No

Noise, bias, and variance



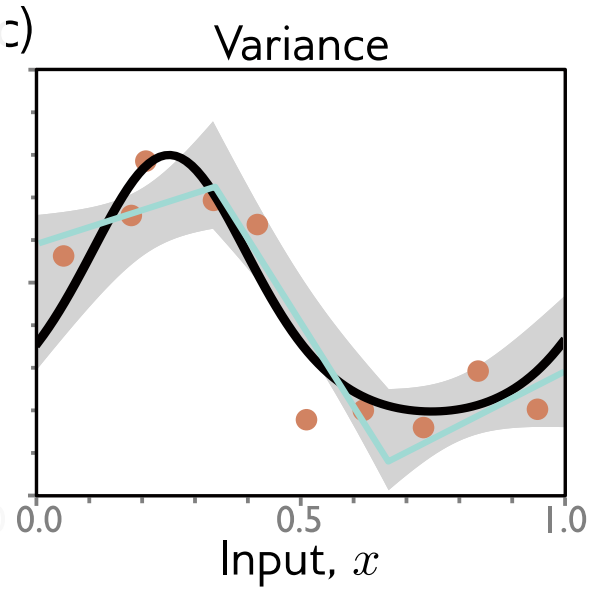
Bias occurs because the model lacks precision or capacity to accurately match the underlying function.

E.g. optimal fit with 3 hidden units and 3 line segments

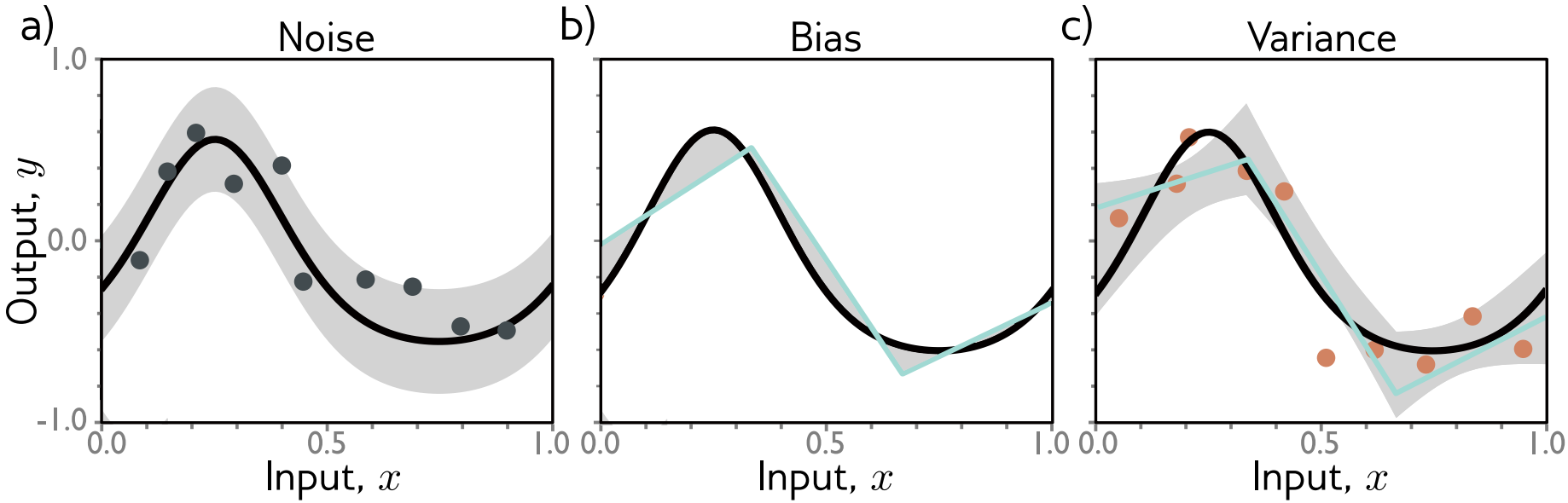
Noise, bias, and variance

In practice, we have limited noisy training data (orange points)

Variability every time we capture training data and also from stochastic learning algorithms.

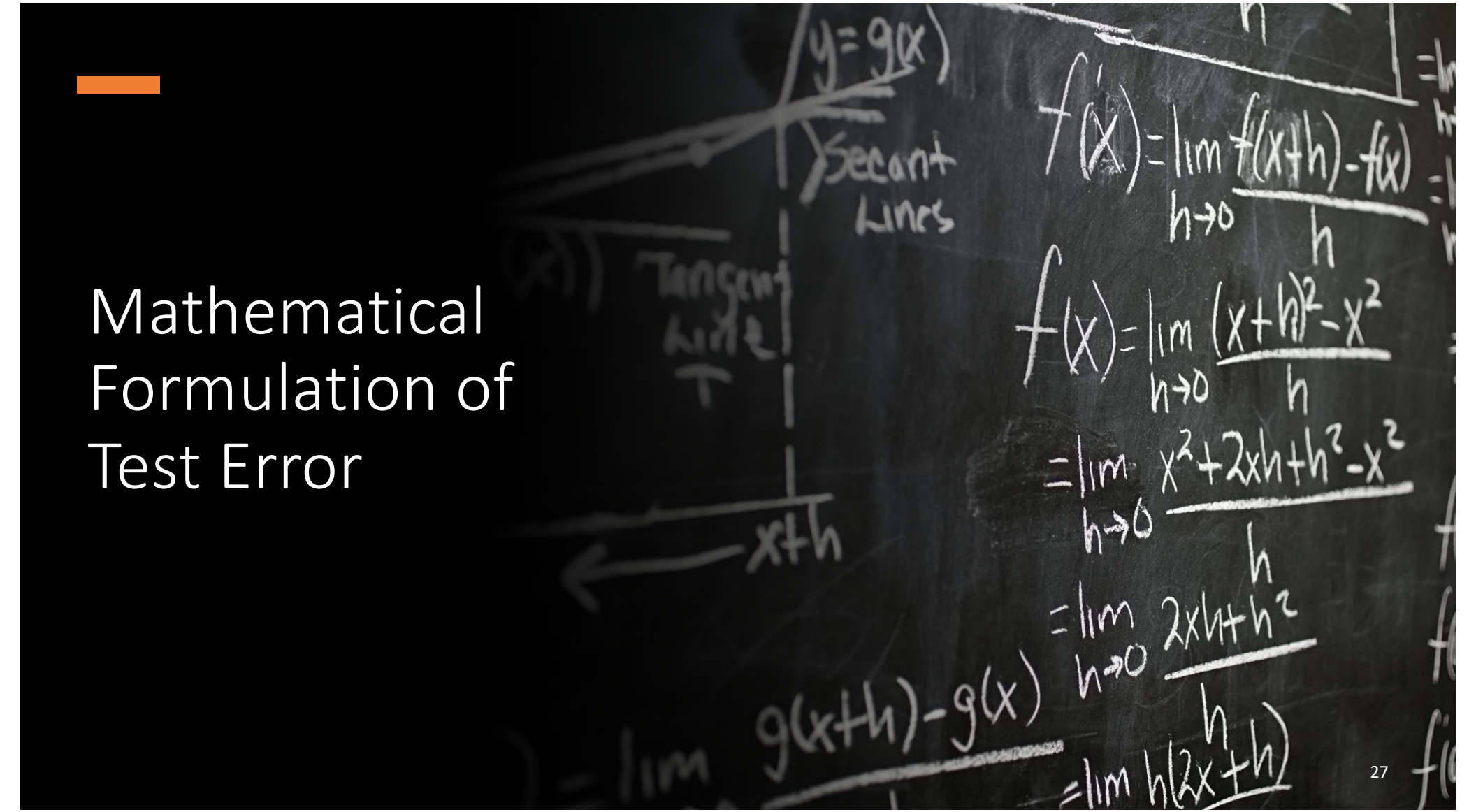


Noise, bias, and variance



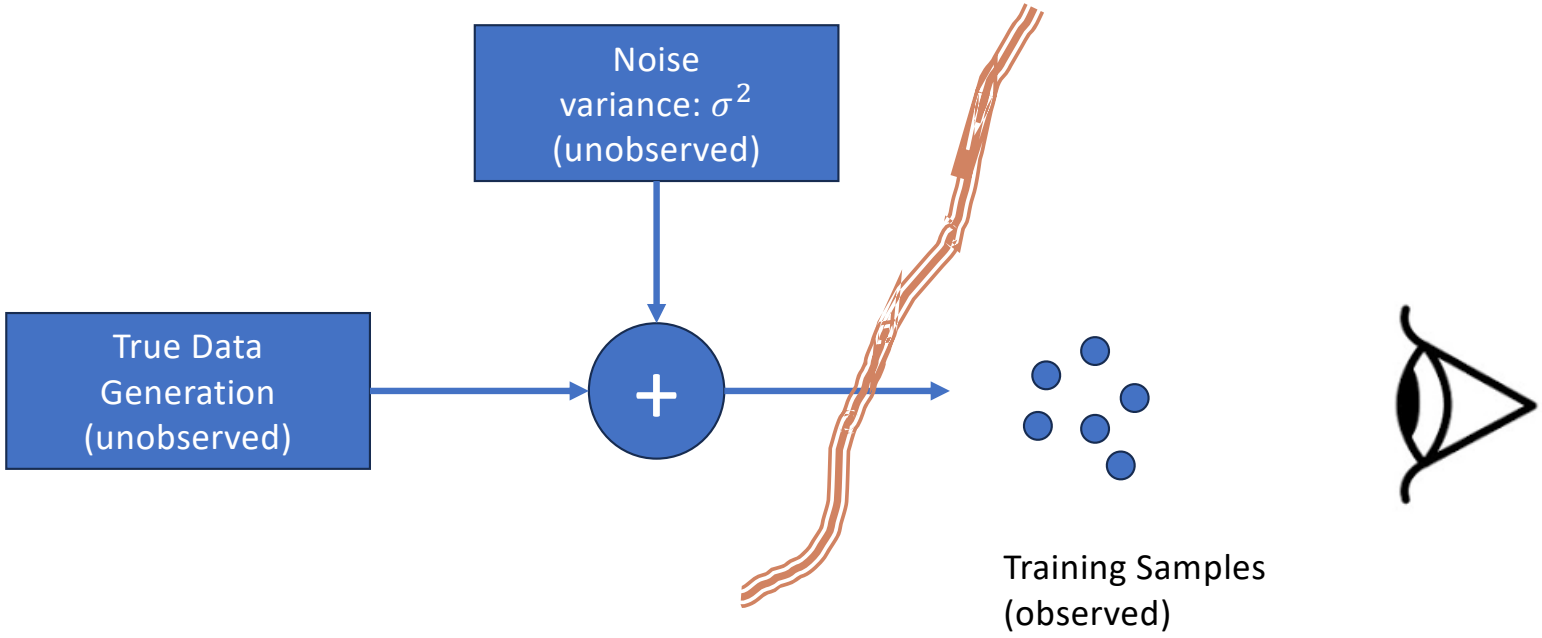
A researcher trains a neural network on a small dataset of 50 examples. The model achieves 99% training accuracy but only 61% test accuracy. She then doubles the dataset to 100 examples and retrains — this time achieving 98% training accuracy and 74% test accuracy. Which of the following best explains this improvement?

Mathematical Formulation of Test Error



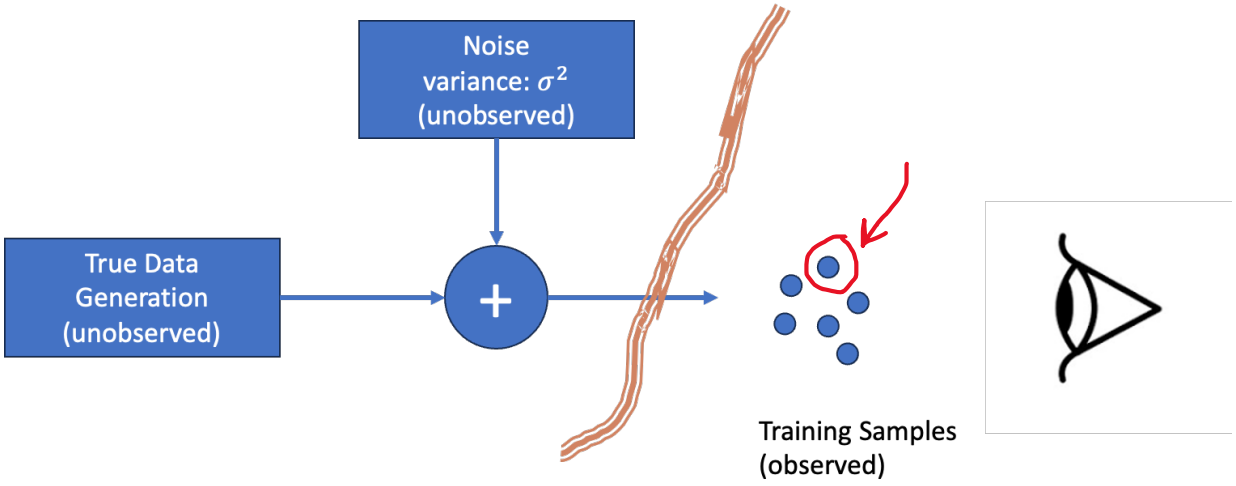
Mathematical Formulation of Test Error

- 1-D regression where underlying data generation process (unobservable) has additive noise with variance σ^2.



Mathematical Formulation of Test Error

- 1-D regression where underlying data generation process (unobservable) has additive noise with variance σ^2.
- For each x there is a distribution $P(y|x)$



Mathematical Formulation of Test Error

- 1-D regression where underlying data generation process (unobservable) has additive noise with variance σ^2.
- For each x there is a distribution of $y[x]$ which is $P(y|x)$
- We can calculate the expected value (i.e. mean), $\mu[x]$:

$$\mu[x] = \mathbb{E}_y[y[x]] = \int y[x] Pr(y|x) dy,$$

The name is a bit nuanced and maybe a little confusing:
$\mu[x]$ is the expected value of y for a given x.
Maybe $\mu_y[x]$ would be better??

Mathematical Formulation of Test Error

- 1-D regression where underlying data generation process (unobservable) has additive noise with variance σ^2.
- For each x there is a distribution of $y[x]$ which is $P(y|x)$
- We can calculate the expected value (i.e. mean), $\mu[x]$:

$$\mu[x] = \mathbb{E}_y[y[x]] = \int y[x] Pr(y|x) dy,$$

Definition of Expectation.

- We can express the noise variance as

$$\sigma^2 = \mathbb{E}_y\left[(\mu[x] - y[x])^2\right]$$

Definition of variance in terms of expectation.

Express Loss Equation

- We can write the loss at input x, $L[x]$, between the model prediction $\mathrm{f}[x, \phi]$ and the output at x, $y[x]$:

$$L[x] = \left(\mathrm{f}[x, \boldsymbol{\phi}] - y[x]\right)^2$$

Subtract and add $\mu[x]$

Mathematical "trick"

- We can write the loss at input x, $L[x]$, between the model prediction $\mathrm{f}[x, \phi]$ and the output at x, $y[x]$:

$$\begin{aligned} L[x] &= \big(\mathrm{f}[x, \phi] - y[x]\big)^2 \\ &= \Big(\big(\mathrm{f}[x, \phi] - \mu[x]\big) + \big(\mu[x] - y[x]\big)\Big)^2 \\ &= \big(\mathrm{f}[x, \phi] - \mu[x]\big)^2 + 2\big(\mathrm{f}[x, \phi] - \mu[x]\big)\big(\mu[x] - y[x]\big) + \big(\mu[x] - y[x]\big)^2, \end{aligned}$$

Subtract and add $\mu[x]$, group then multiply out

Take expectation

- We are treating $y[x]$ as a random variable, so we can take the expectation of $L[x]$ with respect to y.

$$\mathbb{E}_y\left[L[x]\right] = \mathbb{E}_y\left[\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\left(\mu[x]-y[x]\right)+\left(\mu[x]-y[x]\right)^2\right]$$

Use properties of expectation

- We are treating $y[x]$ as a random variable, so we can take the expectation of $L[x]$ with respect to y.

$$\begin{aligned}\mathbb{E}_y\left[L[x]\right] &= \mathbb{E}_y\left[\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\left(\mu[x]-y[x]\right)+\left(\mu[x]-y[x]\right)^2\right]\\ &= \left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\left(\mu[x]-\mathbb{E}_y\left[y[x]\right]\right)+\mathbb{E}_y\left[\left(\mu[x]-y[x]\right)^2\right]\end{aligned}$$

Using the linear properties of expectation, we can move it into the sum.

Use properties of expectation

- We are treating $y[x]$ as a random variable, so we can take the expectation of $L[x]$ with respect to y.

$$\begin{aligned}
\mathbb{E}_y\left[L[x]\right] &= \mathbb{E}_y\left[\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\left(\mu[x]-y[x]\right)+\left(\mu[x]-y[x]\right)^2\right] \\
&= \left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\left(\mu[x]-\mathbb{E}_y\left[y[x]\right]\right)+\mathbb{E}_y\left[\left(\mu[x]-y[x]\right)^2\right] \\
&= \left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\cdot 0+\mathbb{E}_y\left[\left(\mu[x]-y[x]\right)^2\right]
\end{aligned}$$

Middle term becomes zero.

To get two terms…

- We are treating $y[x]$ as a random variable, so we can take the expectation of $L[x]$ with respect to y.

$$\begin{aligned}
\mathbb{E}_y\left[L[x]\right] &= \mathbb{E}_y\left[\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\left(\mu[x]-y[x]\right)+\left(\mu[x]-y[x]\right)^2\right] \\
&= \left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\left(\mu[x]-\mathbb{E}_y\left[y[x]\right]\right)+\mathbb{E}_y\left[\left(\mu[x]-y[x]\right)^2\right] \\
&= \left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+2\left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)\cdot 0+\mathbb{E}_y\left[\left(\mu[x]-y[x]\right)^2\right] \\
&= \left(\mathrm{f}[x,\boldsymbol{\phi}]-\mu[x]\right)^2+\sigma^2, \qquad (8.3)
\end{aligned}$$

Squared deviation of model from true mean.

Standard deviation of noise.

Expectation w.r.t. dataset sample

- The first term can be further split into bias and variance.
- Parameters ϕ of the model $f[x, \phi]$ depend on the training dataset $\mathcal{D} = \{x_i, y_i\}$, e.g. $f[x, \phi[\mathcal{D}]]$
- And training dataset is a noisy sample of the true data
- So we can write the expected model output $f_\mu[x]$ w.r.t. all possible datasets $\mathcal{D}$

$$\mathrm{f}_\mu[x] = \mathbb{E}_{\mathcal{D}}\Big[\mathrm{f}\big[x, \phi[\mathcal{D}]\big]\Big]$$

Add and subtract $f_\mu[x]$

- We can expand that first term by subtracting and adding $f_\mu[x]$ and multiply

$$\begin{aligned}
&\big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - \mu[x]\big)^2 \\
&\quad = \Big(\big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - \mathrm{f}_\mu[x]\big) + \big(\mathrm{f}_\mu[x] - \mu[x]\big)\Big)^2 \\
&\quad = \big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - \mathrm{f}_\mu[x]\big)^2 + 2\big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - \mathrm{f}_\mu[x]\big)\big(\mathrm{f}_\mu[x] - \mu[x]\big) + \big(\mathrm{f}_\mu[x] - \mu[x]\big)^2.
\end{aligned}$$

Mathematical Formulation of Test Error

- Then take expectation w.r.t. training dataset $\mathcal{D}$:

Middle term on previous slide goes to zero. Check!

$$\mathbb{E}_{\mathcal{D}}\left[\left(\mathrm{f}[x,\boldsymbol{\phi}[\mathcal{D}]]-\mu[x]\right)^2\right]=\mathbb{E}_{\mathcal{D}}\left[\left(\mathrm{f}[x,\boldsymbol{\phi}[\mathcal{D}]]-\mathrm{f}_{\mu}[x]\right)^2\right]+\left(\mathrm{f}_{\mu}[x]-\mu[x]\right)^2,$$

- We can then $\mathbb{E}_{\mathcal{D}}\Big[\mathbb{E}_{y}[L[x]]\Big]$

$$\mathbb{E}_{\mathcal{D}}\Big[\mathbb{E}_{y}[L[x]]\Big]=\underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(\mathrm{f}[x,\boldsymbol{\phi}[\mathcal{D}]]-\mathrm{f}_{\mu}[x]\right)^2\right]}_{\text{variance}}+\underbrace{\left(\mathrm{f}_{\mu}[x]-\mu[x]\right)^2}_{\text{bias}}+\underbrace{\sigma^2}_{\text{noise}}.$$

Summary for Regression with Least Squares Error

$$L[x] = \big(\mathrm{f}[x, \boldsymbol{\phi}] - y[x]\big)^2$$

- We can show that:

$$\mathbb{E}_y\big[L[x]\big] = \big(\mathrm{f}[x, \boldsymbol{\phi}] - \mu[x]\big)^2 + \sigma^2$$

- And then:

$$\mathbb{E}_{\mathcal{D}}\Big[\mathbb{E}_y[L[x]]\Big] = \underbrace{\mathbb{E}_{\mathcal{D}}\Big[\big(\mathrm{f}[x, \boldsymbol{\phi}[\mathcal{D}]] - f_\mu[x]\big)^2\Big]}_{\text{variance}} + \underbrace{\big(f_\mu[x] - \mu[x]\big)^2}_{\text{bias}} + \underbrace{\sigma^2}_{\text{noise}}$$

Expectation over noise in training data

Expectation over noise in test data

Actual model

Best possible model if we had infinite data

True function

More complex interactions between noise, bias and variance in more complex models.

After all that algebra, which term in the decomposition can we actually do something about by collecting more data?

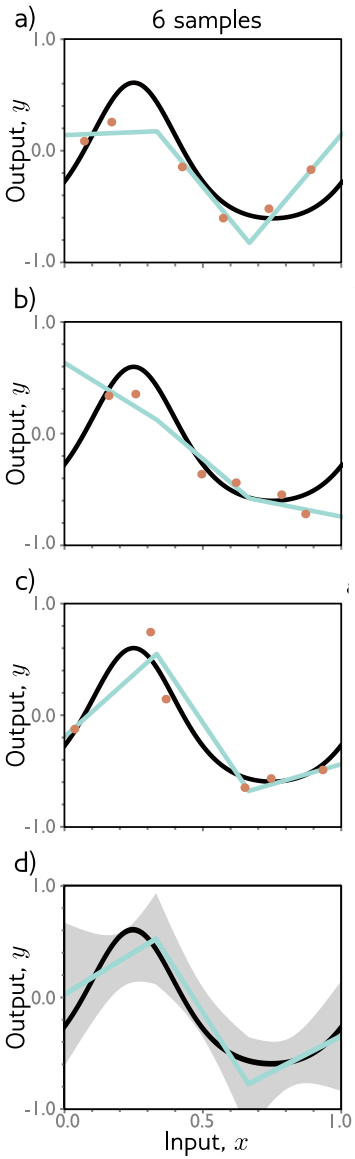
Warning

- The interactions between variance, bias and noise is more complex for other loss functions, e.g. cross entropy loss in classification.
- Adding variance might counteract bias and improve classification result by pushing samples across classification boundaries…

Measuring performance

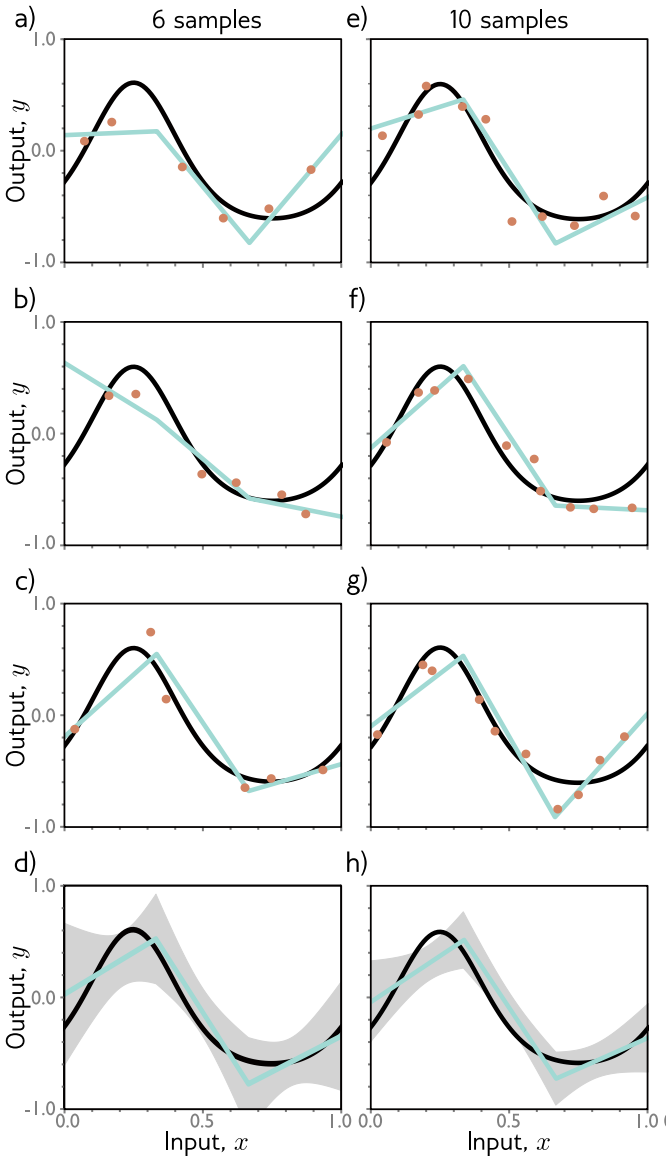
- MNIST1D dataset model and performance
- Noise, bias, and variance
- Reducing variance
- Reducing bias & bias-variance trade-off
- Double descent
- Curse of dimensionality & weird properties of high dimensional space
- Choosing hyperparameters

Variance



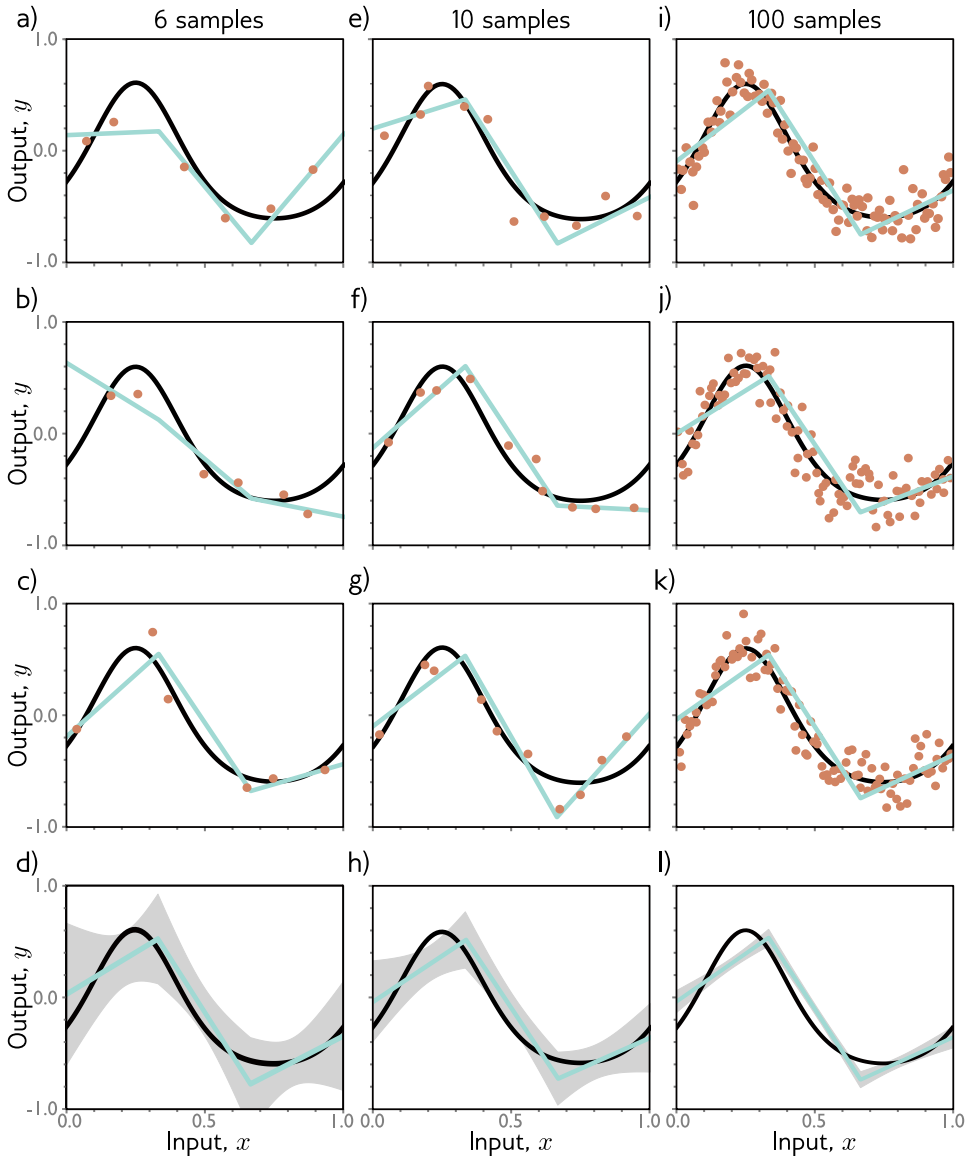
When measuring (capturing) 6 different data samples with a fixed model (e.g. 3 hidden units), we get different optimal fits every time.

Variance



Can reduce variance by adding more samples

Variance



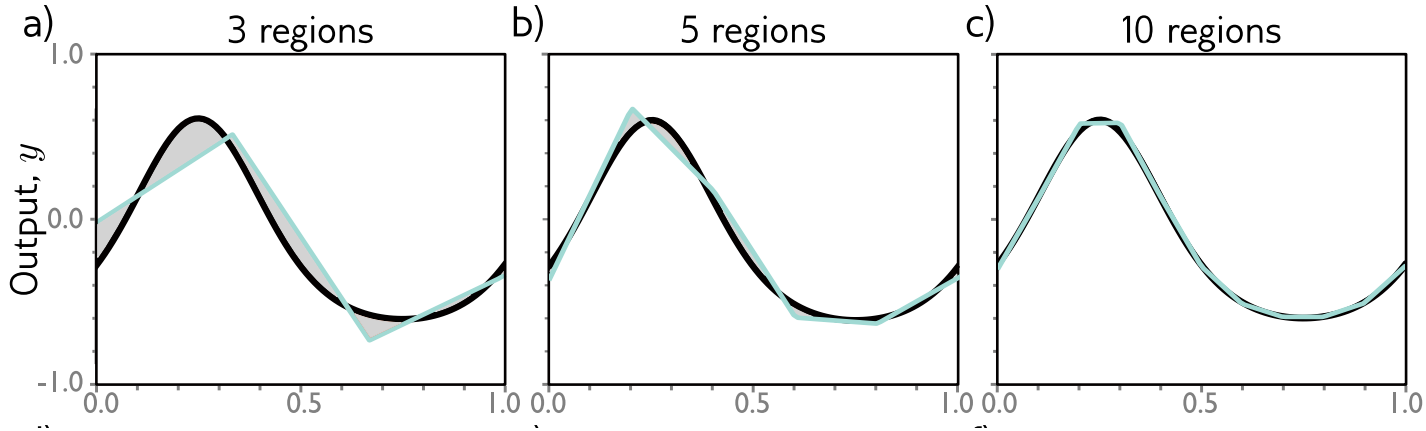
Can reduce variance by adding more samples

Measuring performance

- MNIST1D dataset model and performance
- Noise, bias, and variance
- Reducing variance
- Reducing bias & bias-variance trade-off
- Double descent
- Curse of dimensionality & weird properties of high dimensional space
- Choosing hyperparameters

Reducing bias
(example with the true function)

Bias



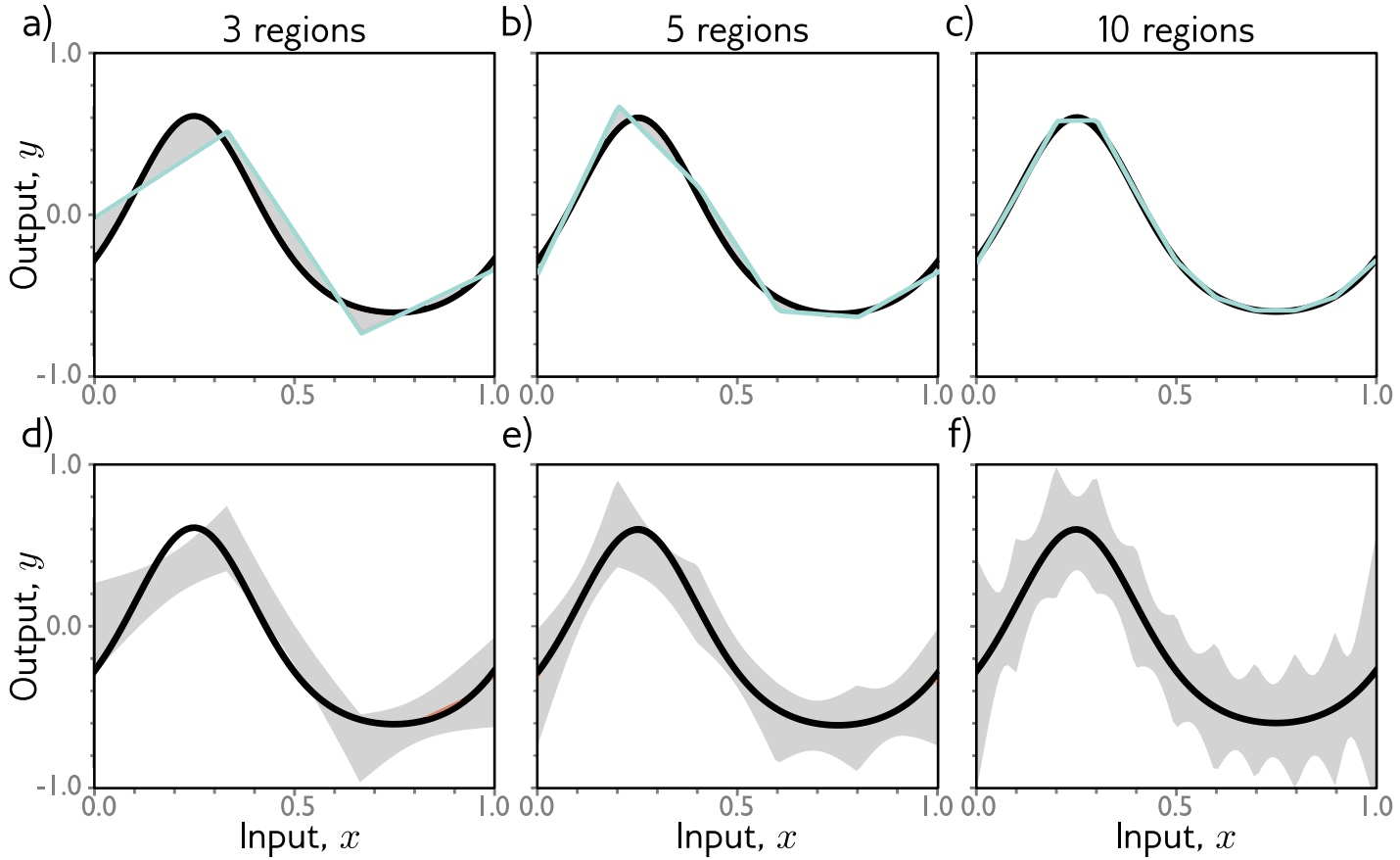
We can reduce bias by adding more model capacity.

In this case, adding more hidden units.

Reducing bias ➔ Increases variance!!

Bias

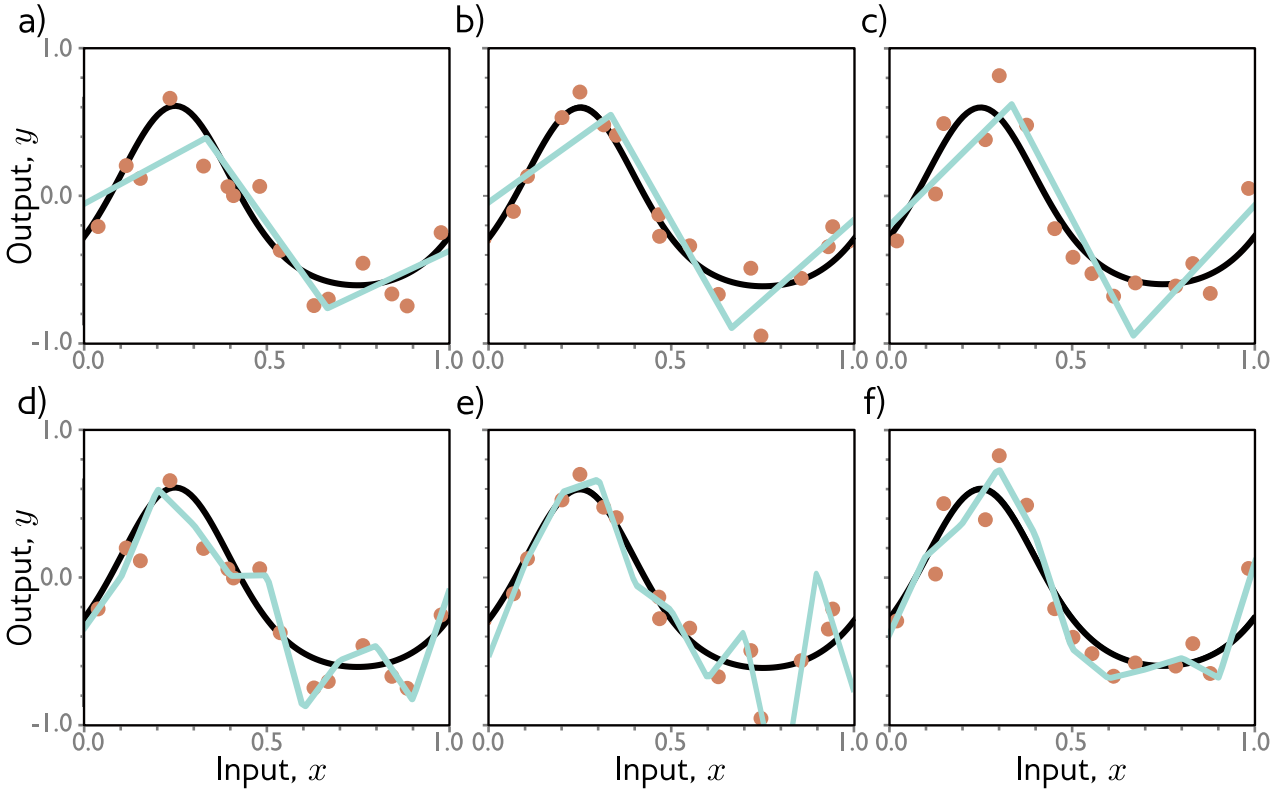
Variance



Why does variance increase? Overfitting

3 Regions

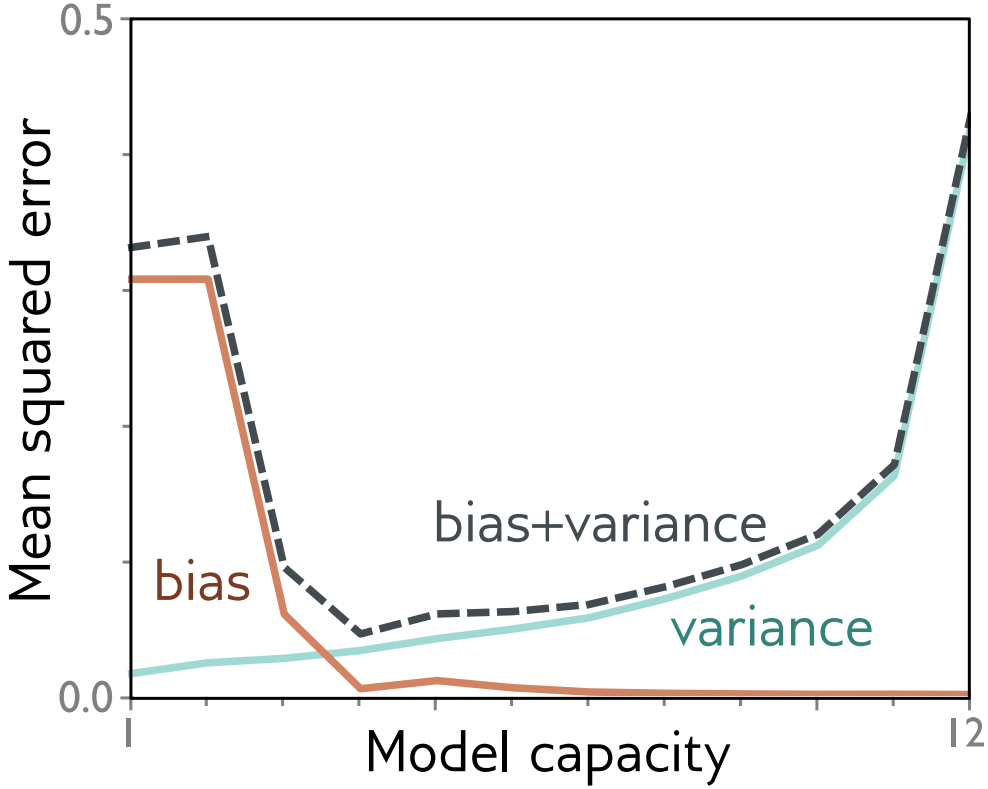
10 Regions

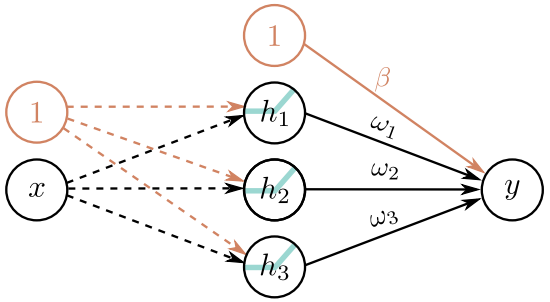


Describes the training data better, but not the true underlying function (black curve)

Many ways to fit a sample of 15 data points

Bias and variance trade-off for the simple linear model





$$\mathbb{E}_{\mathcal{D}}\Big[\mathbb{E}_y[L[x]]\Big] = \underbrace{\mathbb{E}_{\mathcal{D}}\Big[\big(\mathrm{f}[x,\boldsymbol{\phi}[\mathcal{D}]] - f_\mu[x]\big)^2\Big]}_{\text{variance}} + \underbrace{\big(f_\mu[x] - \mu[x]\big)^2}_{\text{bias}} + \underbrace{\sigma^2}_{\text{noise}}$$

Number of hidden units

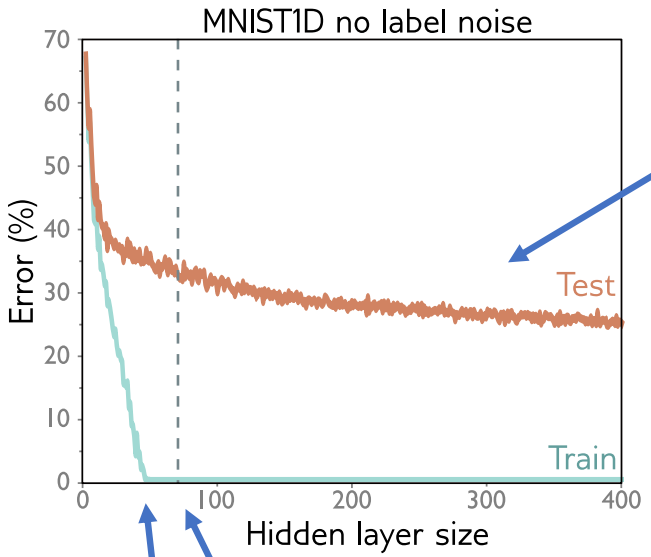
But does picking model capacity to minimize bias & variance hold for more complex data and models?

Measuring performance

- MNIST1D dataset model and performance
- Noise, bias, and variance
- Reducing variance
- Reducing bias & bias-variance trade-off
- Double descent
- Curse of dimensionality & weird properties of high dimensional space
- Choosing hyperparameters

Train and Test Error versus # of Hidden Layers

- 10,000 training examples
- 5,000 test examples
- Two hidden layers
- Adam optimizer
- Step size of 0.005
- Full batch
- 4000 training steps



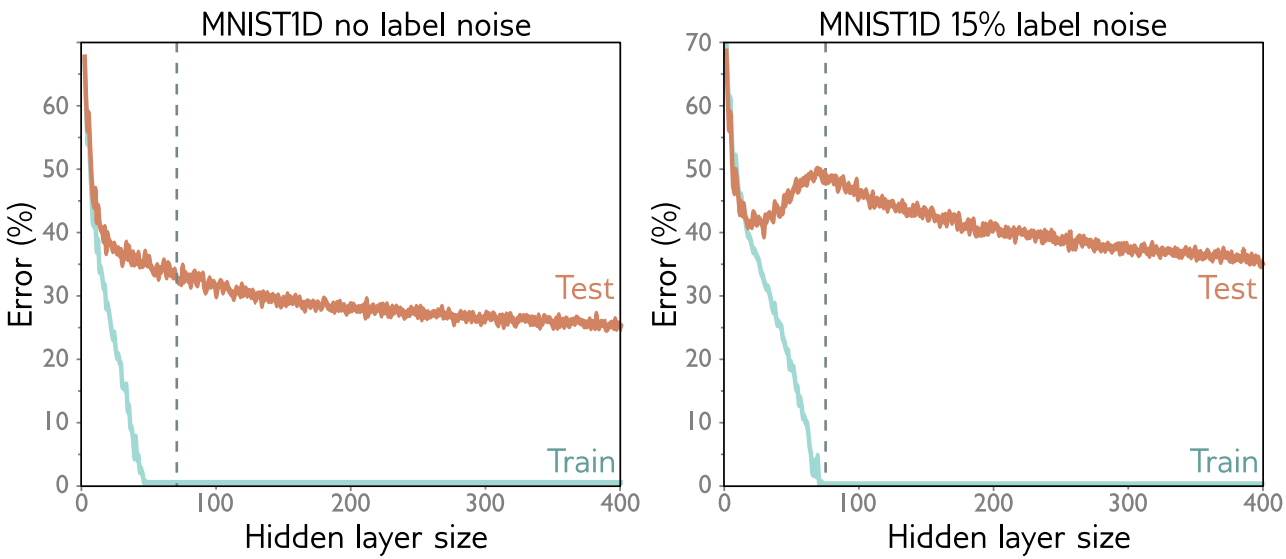
Test error keep decreasing even as we keep increasing model capacity!

Training parameters = Training examples

Model has *memorized* the training set

Why do we say that?

Now randomize 15% of the training labels

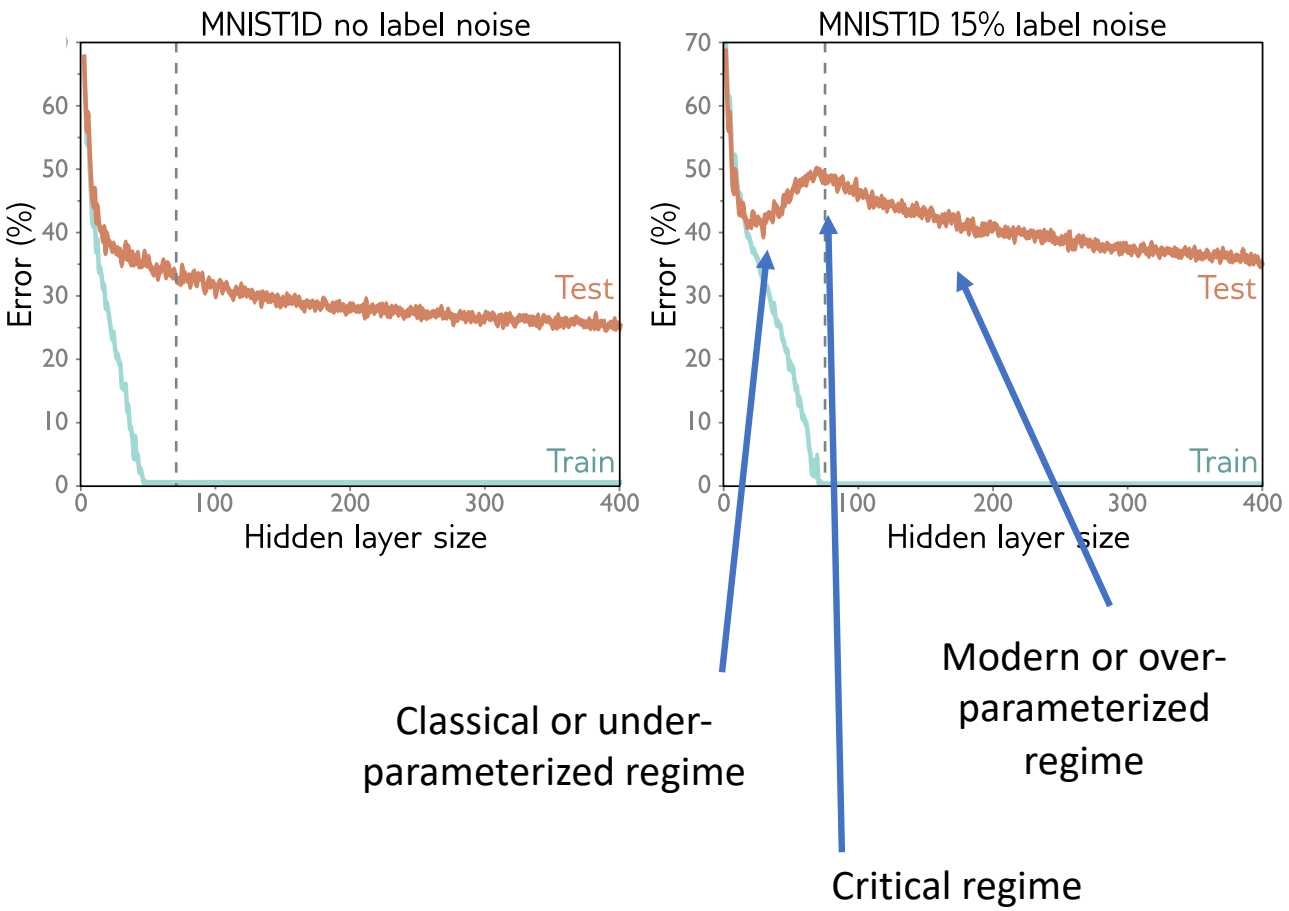


Now we see what looks like bias-variance trade-off as we increase capacity to the point where the model fits training data.

But then???

Reminder: vertical dashed line is where:
training parameters = # training samples

Double Descent

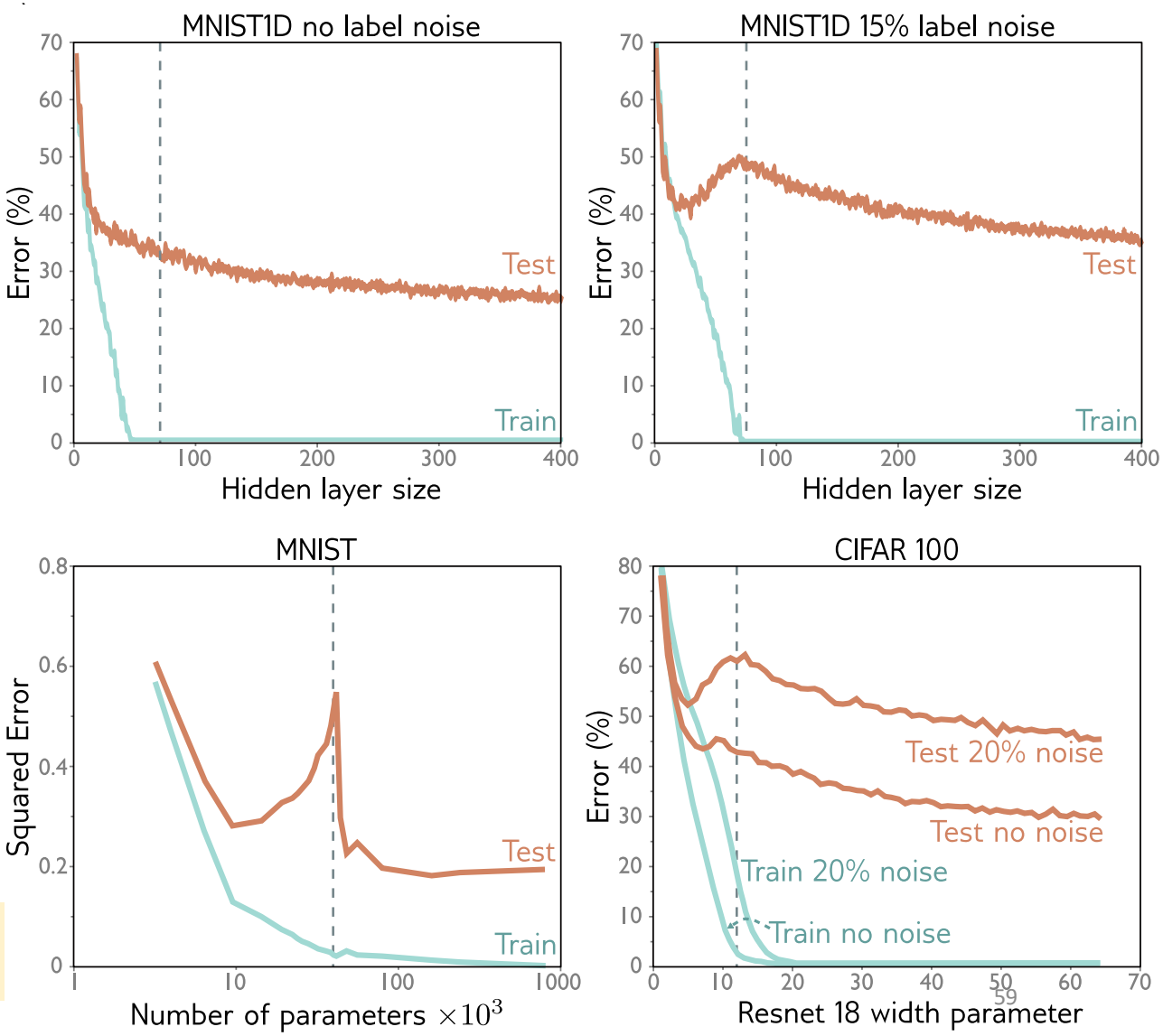


Reminder: vertical dashed line is where:
training parameters = # training samples

A model is trained with 10,000 examples. At hidden layer size 100 the test error spikes to its highest point, then starts falling again as hidden layer size increases further. What is most likely true about the model at hidden layer size 100?

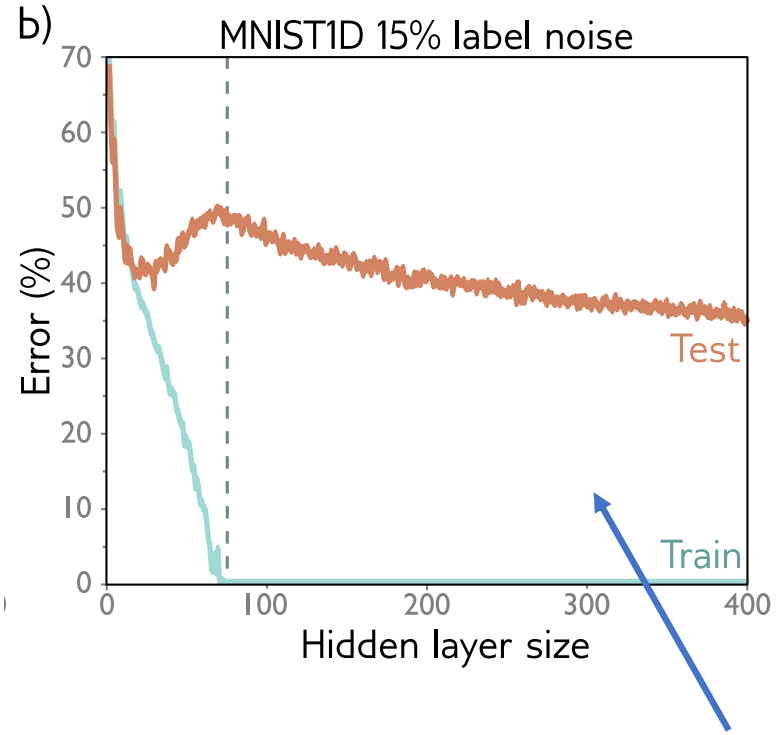
Same phenomenon shows up on MNIST and CIFAR100

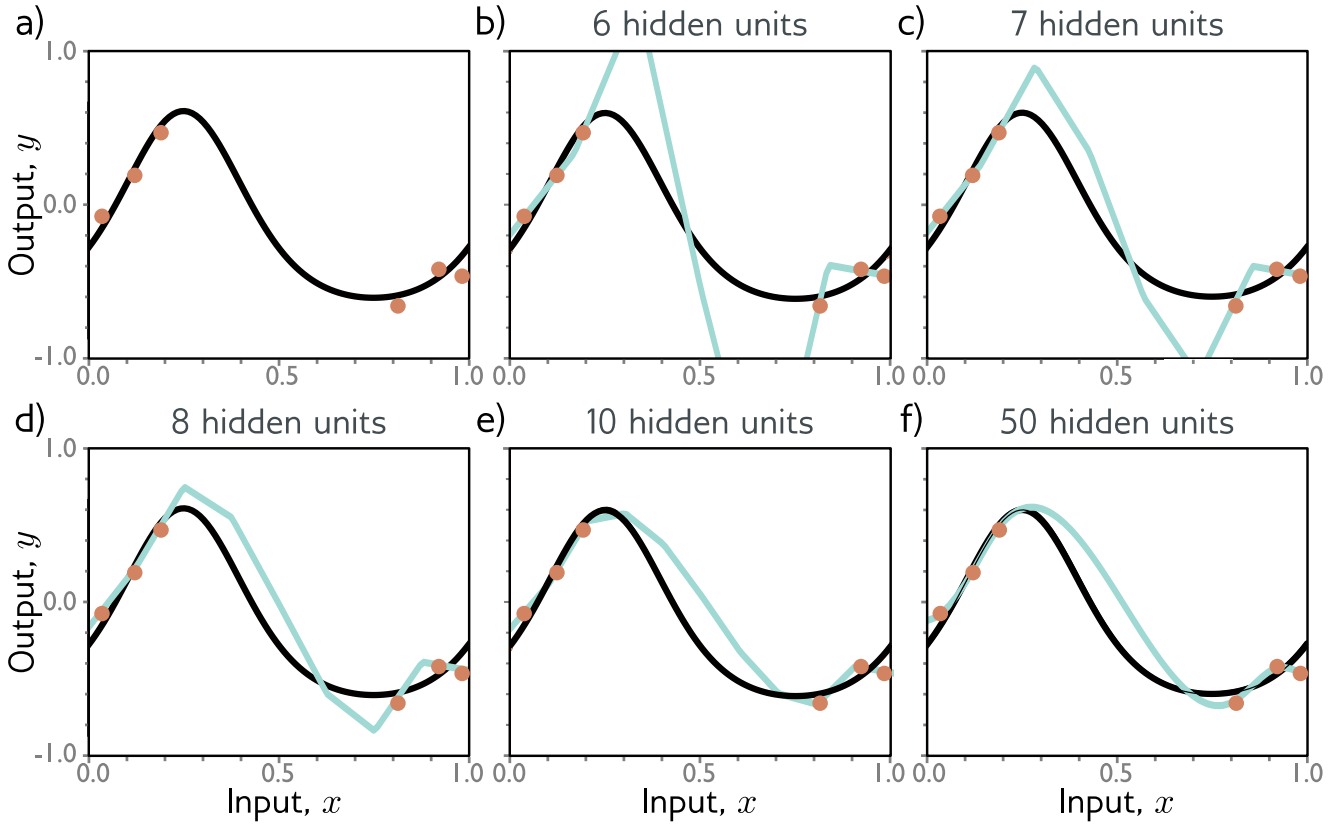
Reminder: vertical dashed line is where:
training parameters = # training samples



Double Descent

- Note that training loss is very close to zero.
- Whatever is happening isn't happening at training data points
- Model never sees test set during training
- Must be happening between the data points??

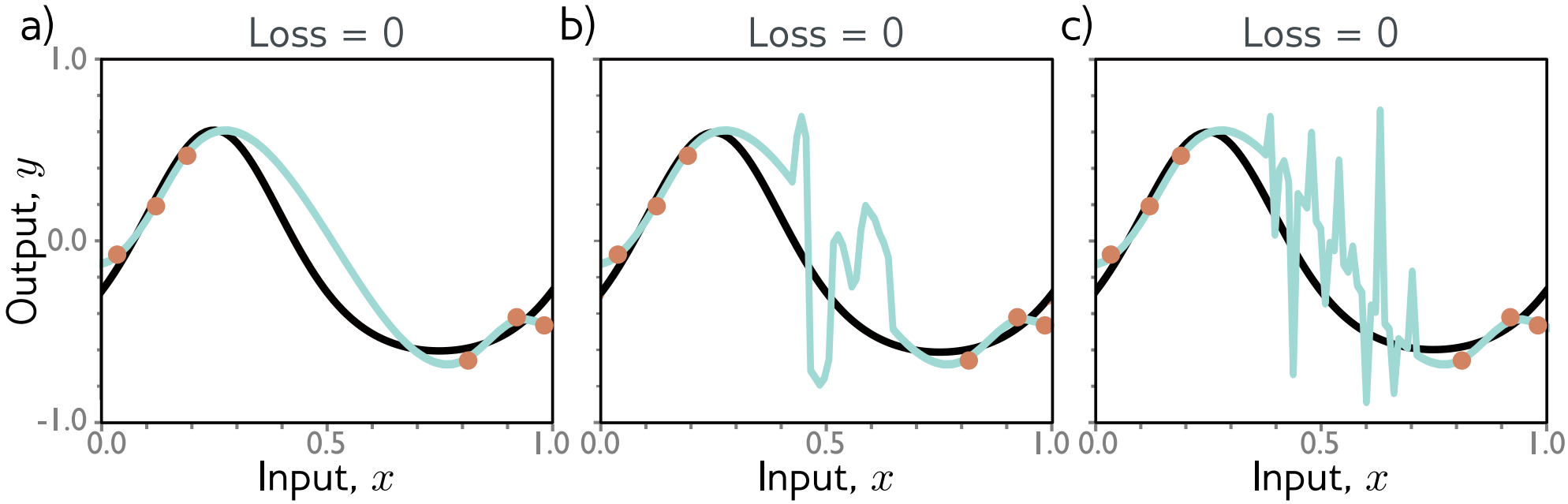




Potential explanation:

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

But why?



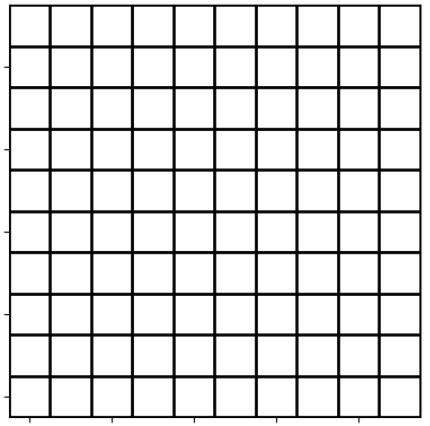
- All of these solutions are equivalent in terms of loss.
- Why should the model choose the smooth solution?

Measuring performance

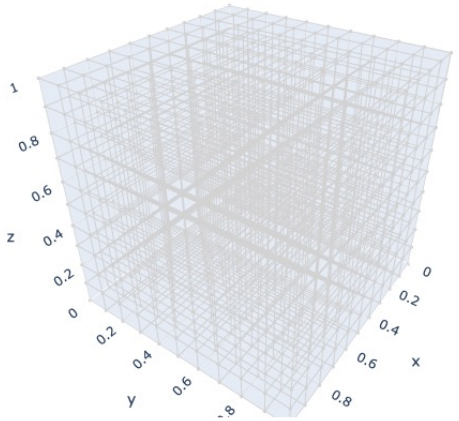
- MNIST1D dataset model and performance
- Noise, bias, and variance
- Reducing variance
- Reducing bias & bias-variance trade-off
- Double descent
- Curse of dimensionality & weird properties of high dimensional space
- Choosing hyperparameters

Curse of dimensionality

- 40-dimensional data
- 10,000 data points
- Consider quantizing each dimension into 10 bins
- 10^{40} bins
- 1 data point per 10^{35} bins
- *So most of the space has to be interpolated*
- The tendency of high-dimensional space to overwhelm the number of data points is called the curse of dimensionality



2D: 10x10=100 bins

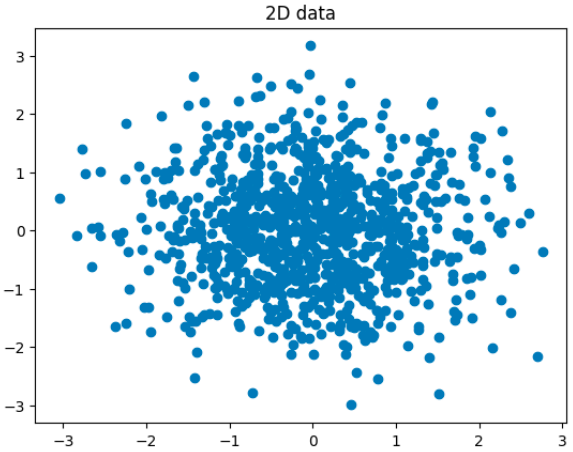


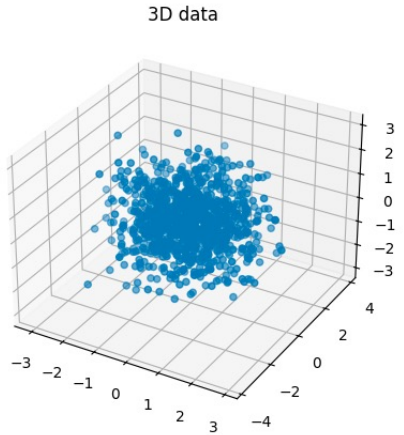
3D: 10x10x10=1000 bins

An Aside on Curse: Distances collapse

Generate 1,000 normally distributed samples in:

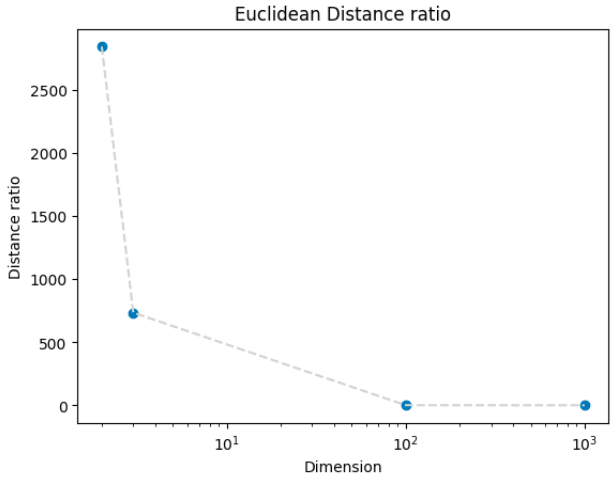
- 2D
- 3D
- 100D
- 1000D





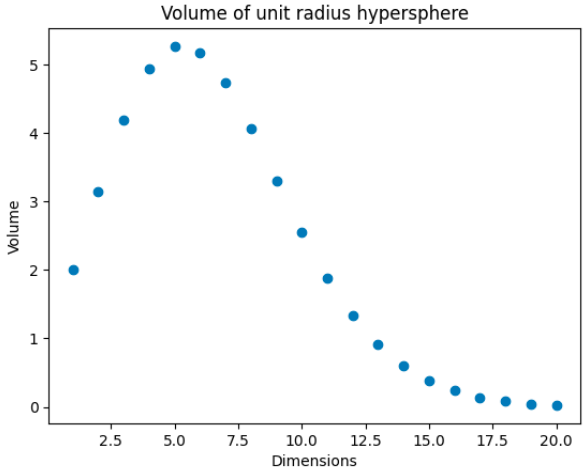
...

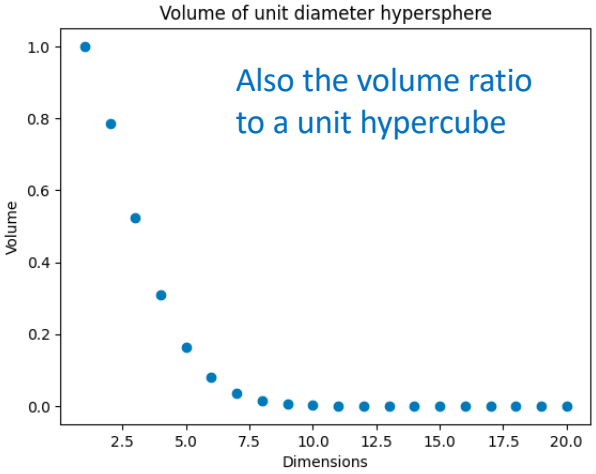
Calculate the ratio of distances between the farthest and closest points.

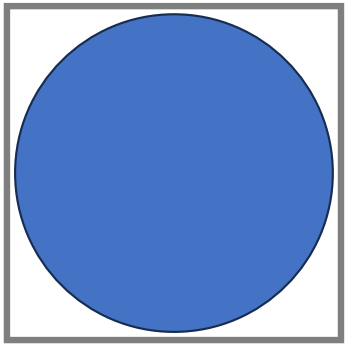


Approaches 1!!

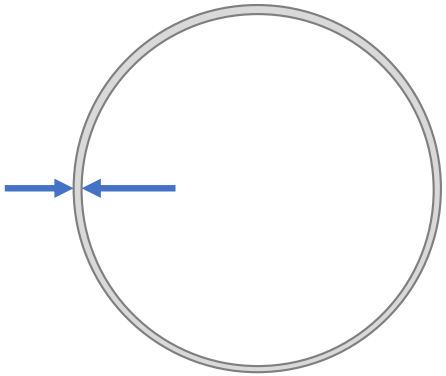
An Aside on Curse: Volumes of a hyperspheres

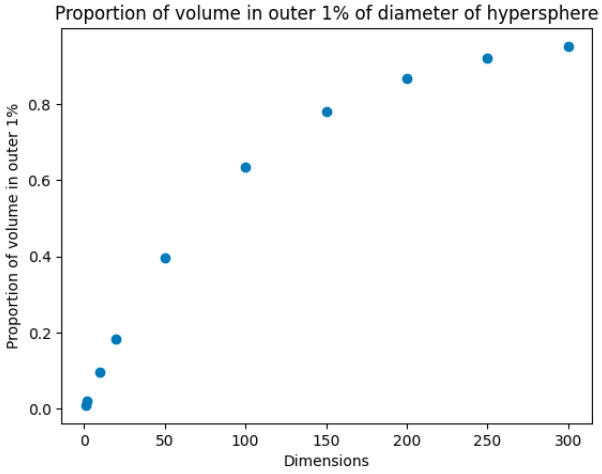






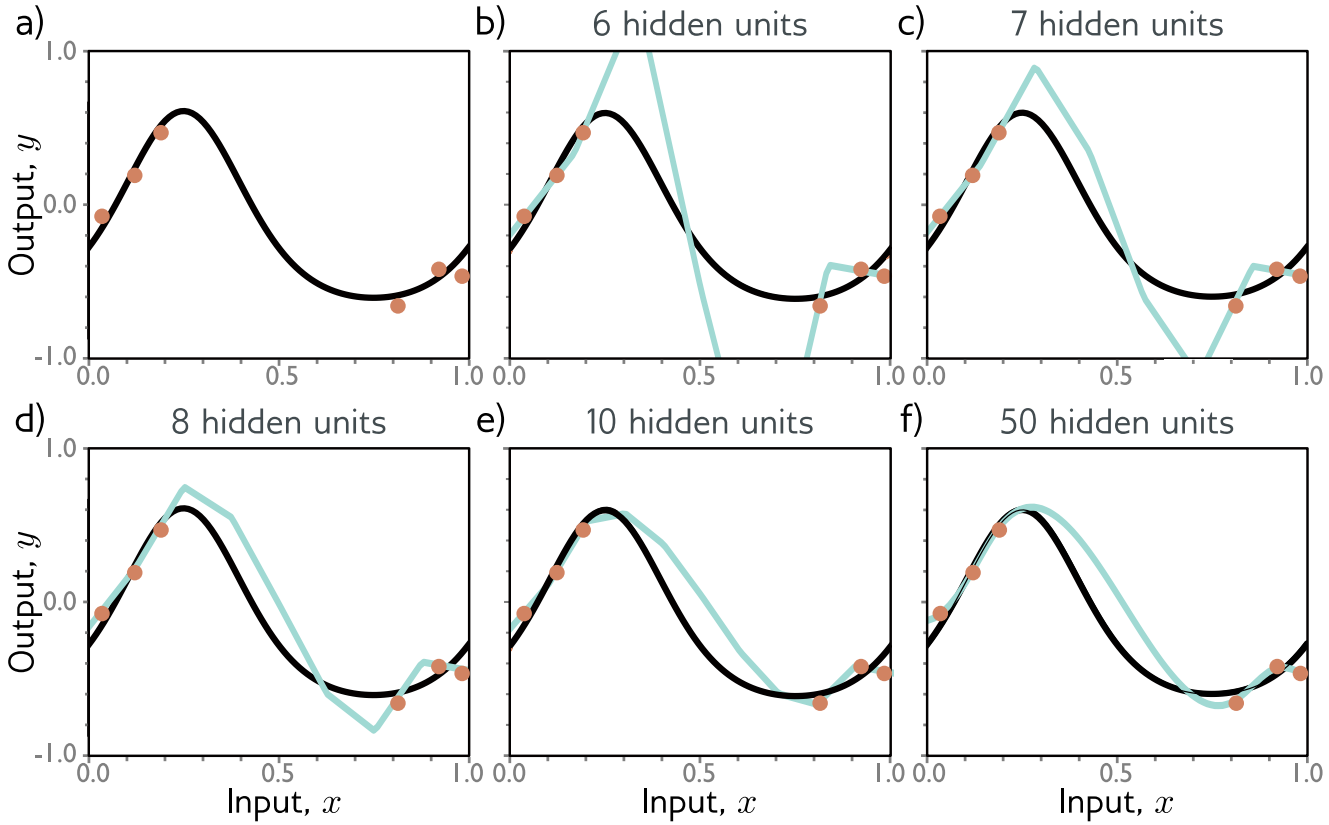
Unit diameter hypersphere in a unit hypercube.





"All the volume goes to the peel of the orange, not the pulp."

See also "An Adventure in the Nth Dimension", American Scientist



Potential explanations:

- Network initialization may encourage smoothness and model doesn't depart much during training
- It seems that through implicit and explicit regularization (next lecture!) the (well trained) model tends to interpolate smoothly between training data points.

Measuring performance

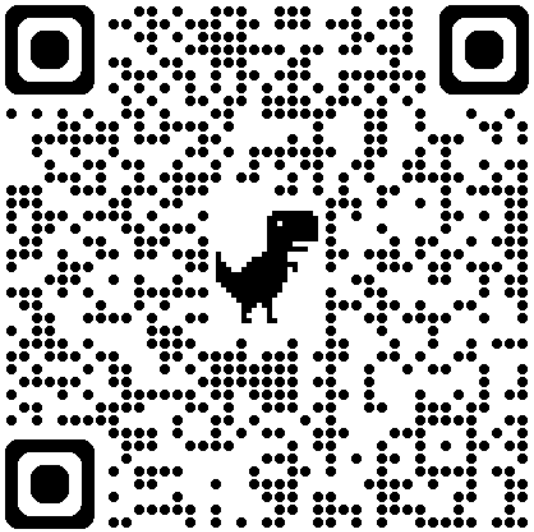
- MNIST1D dataset model and performance
- Noise, bias, and variance
- Reducing variance
- Reducing bias & bias-variance trade-off
- Double descent
- Curse of dimensionality & weird properties of high dimensional space
- Choosing hyperparameters

Choosing hyperparameters

- Don’t know bias or variance
- 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

You've trained 20 models with different hyperparameters and selected the one with the best performance on your validation set. You now report that model's validation accuracy as your estimate of real-world performance. What is wrong with this?

Feedback?



https://forms.gle/pXHM5nx1Ti9aFmpw6