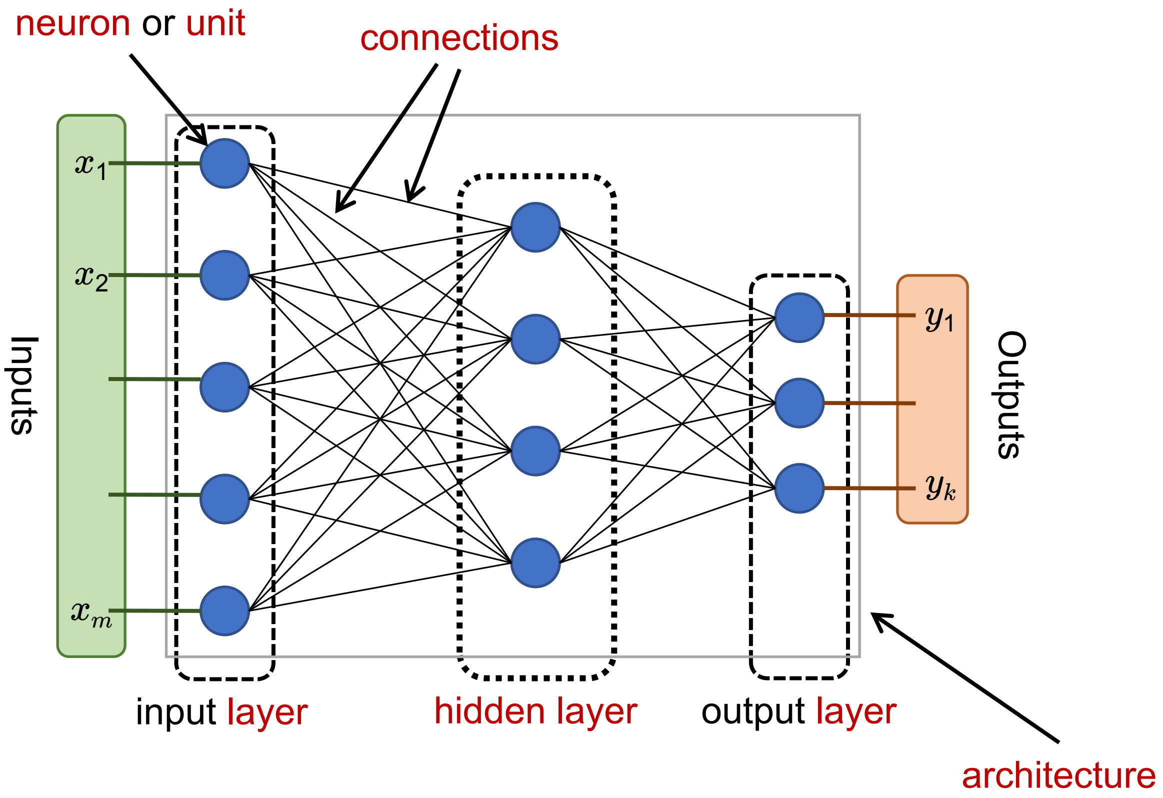
CAI 4104/6108 – Machine Learning Engineering:

Training Deep Neural Networks

Prof. Vincent Bindschaedler

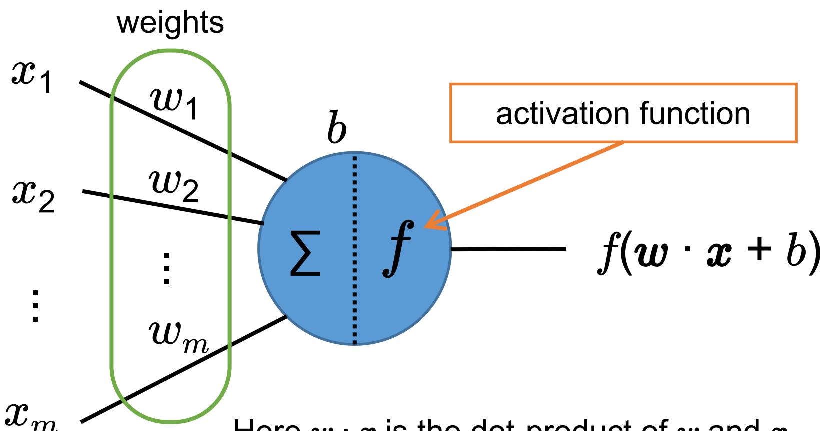
Spring 2024

Reminder: Neural Network Terminology



Reminder: A Simple Neural Network

- Consider a single neuron / unit
  - The model is $h_{\boldsymbol{w},b}(\boldsymbol{x}) = f(\boldsymbol{w} \cdot \boldsymbol{x} + b)$
    - What if we take f to be the identity function?
      - That is: $f(z) = z$
    - What if we take f to be the sigmoid / logistic function?
      - That is: $f(z) = 1/(1+e^{-z})$

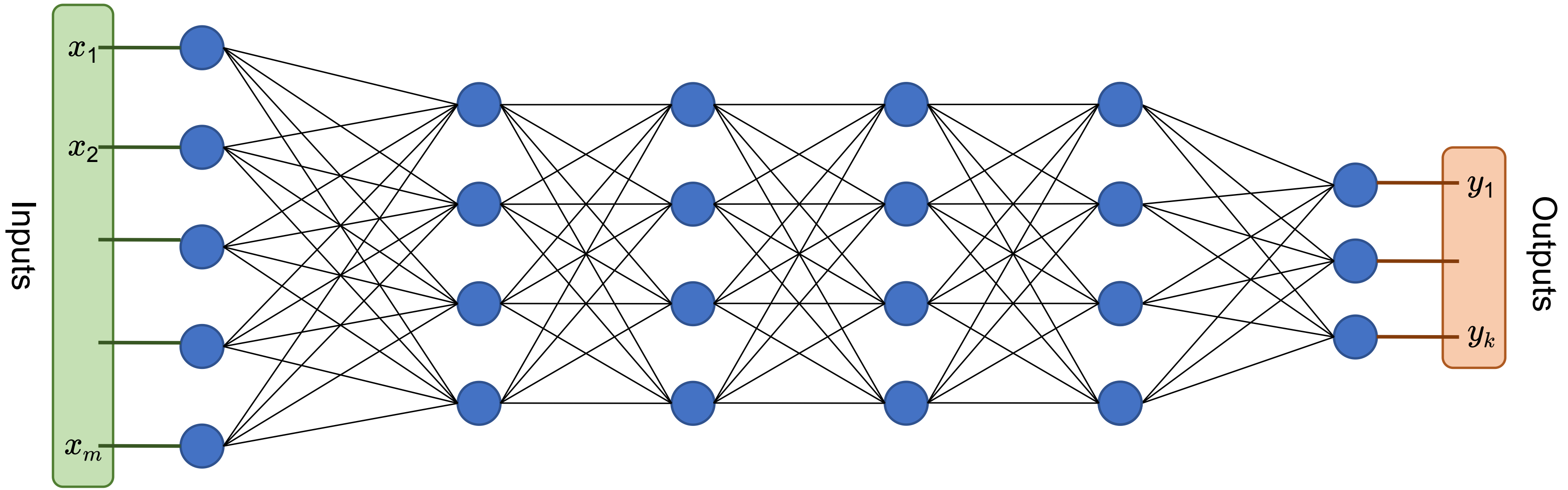


Here $\boldsymbol{w} \cdot \boldsymbol{x}$ is the dot-product of $\boldsymbol{w}$ and $\boldsymbol{x}$.
So $f(\boldsymbol{w} \cdot \boldsymbol{x} + b) = f(\sum w_i x_i + b)$

- The Perceptron
  - Invented by Frank Rosenblatt in 1957
    - "The Perceptron—a perceiving and recognizing automaton". Report 85-460-1. Cornell Aeronautical Laboratory
  - A different neuronal architecture called a threshold linear unit (TLU)
    - No bias term
    - With a step activation function. For example:
      - heaviside(z) = 0 if z ≤ 0, 1 otherwise (z ≥ 1) ; or sign(z)

Reminder: Deep Neural Networks

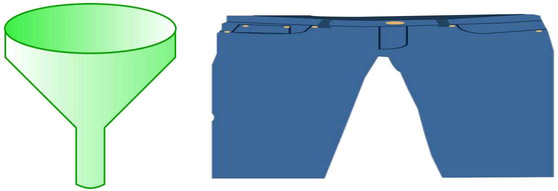
- What is a deep neural network?
  - Any neural network with two or more hidden layers
  - Nowadays, the best neural networks architectures for many applications & problems are deep
    - E.g.: AlexNet (2012) has 8 layers, ResNet18 has 18 layers, GPT-2 has 48 layers



Reminder: Hyperparameters Tuning

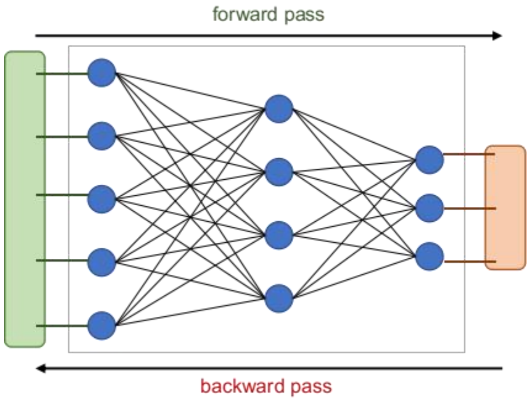
- Challenges:
  - Endless options for the network architecture/topology
    - E.g.: number of layers; units per layer; connections between units; activation functions; weight initialization method
  - Hyperparameters related to learning:
    - E.g.: optimizer, learning rate, decay/momentum, (mini)batch size, number of epochs, etc.

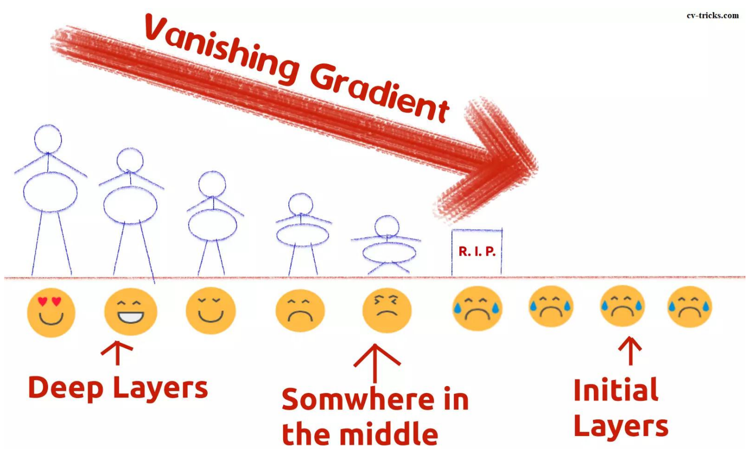
- Rules of Thumb:
  - Number of hidden layers
    - ***Deep > shallow***: For the same number of parameters, more hidden layers is better than wider layers
    - Why? Parameter efficiency
  - Number of units in each layer
    - Funnel approach: make the network look like a funnel
    - "Stretch pants" approach: make hidden layers wider than what you need and then regularize (e.g., dropout)
      - Ref: Vanhoucke' Udacity course on Deep Learning



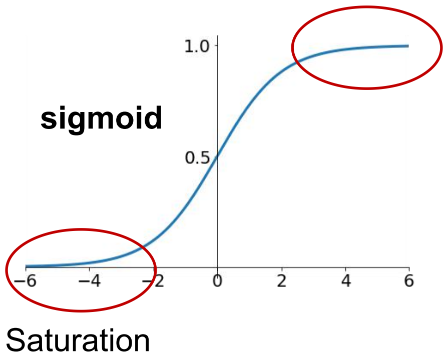
Reminder: Vanishing & Exploding Gradient

- Vanishing/Exploding Problems:
  - Gradient vector becomes very small (vanishing gradient) or very large (exploding gradient) during backpropogation
    - Difficult to update weights of lower/earlier layers => Training does not converge
  - Instance of a more general problem: unstable gradients
    - Layers (of a deep neural network) learn at very different rates



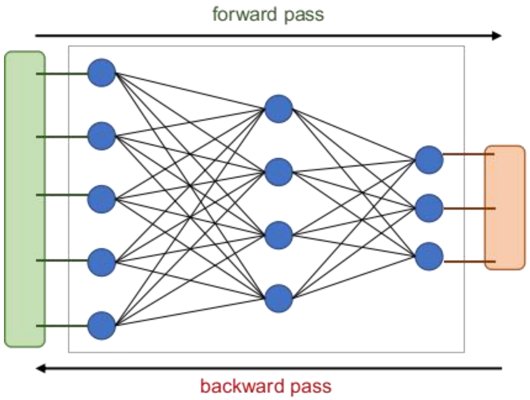


Source: https://cv-tricks.com/keras/understand-implement-resnets/

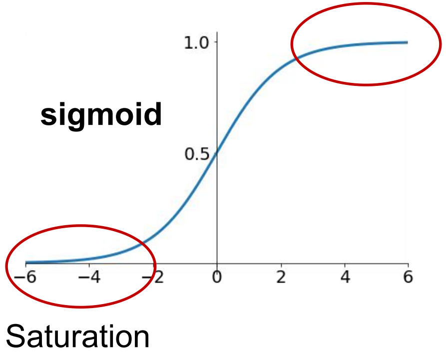


Reminder: Vanishing & Exploding Gradient

- Vanishing/Exploding Problems:
  - Gradient vector becomes very small (vanishing gradient) or very large (exploding gradient) during backpropogation
    - Difficult to update weights of lower/earlier layers => Training does not converge
  - Instance of a more general problem: unstable gradients
    - Layers (of a deep neural network) learn at very different rates



- Mitigations:
  - Weight initialization method
  - Non-saturating activation functions
  - Batch normalization
  - Gradient clipping (for exploding gradient)
  - Skip-connections (CNNs)
  - ...

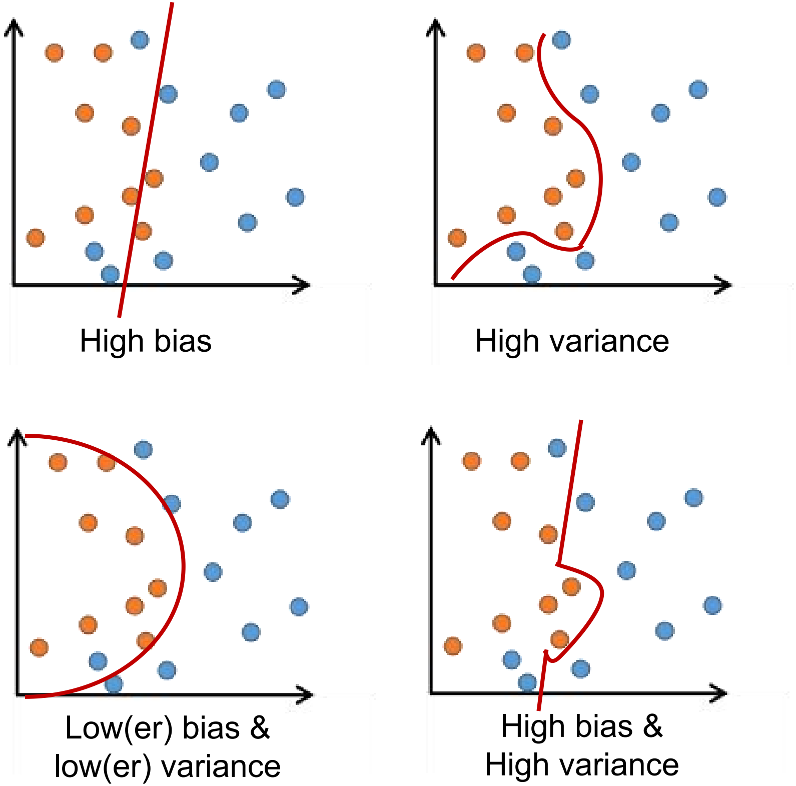


Regularizing Neural Networks

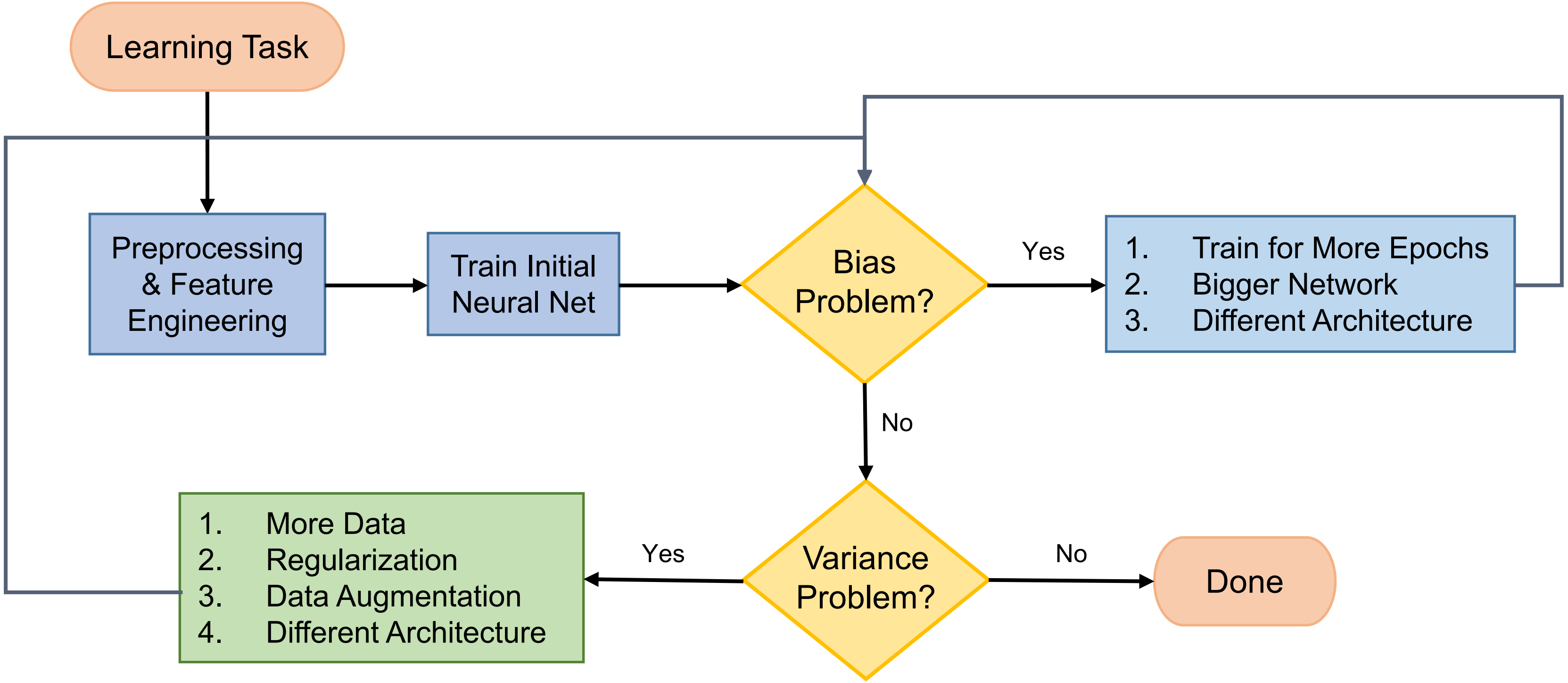
- Early Stopping
  - Stop once the validation loss is at its minimum (before it starts to go back up)
- L_1 or L_2 regularization (or both)
  - L_1: penalty term $\lambda \sum_i |w_i|$
  - L_2: penalty term $\lambda \sum_i |w_i|^2$
- Max-norm regularization
  - The norm of weights incoming to each neuron is at most r (hyperparameter): $||\boldsymbol{w}||_2 < r$
  - Note: this is **not** added to the loss; after each training step, the weights are rescaled to ensure $||\boldsymbol{w}||_2 < r$
- Dropout
  - Seminal paper: Geoffrey E. Hinton et al. "*Improving neural networks by preventing co-adaptation of feature detectors.*" arXiv preprint, 2012.
  - Idea: during training each neuron has a probability p of being *dropped out* (it will be ignored for this step)
  - Hyperparameter p is called the dropout rate
  - After training: we do not drop any neurons anymore (but we need to adjust connection weights)

Reminder: Bias and Variance

- Bias
  - Error due to incorrect assumptions in the model
  - *Inability to capture the true relationship*
- Variance
  - Sensitivity to small variations in the training data
- Ideally, we want: low bias **and** low variance
  - Strategies to lower bias:
    - Increase model complexity
    - Use more features
  - Strategies to lower variance:
    - Reduce model complexity
    - Use more training data



Problem Solving with Neural Networks

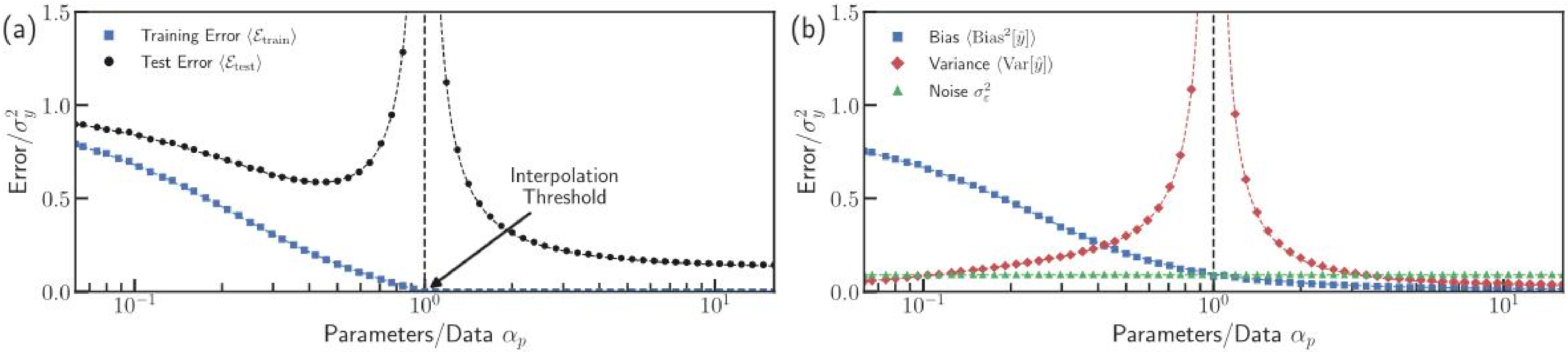


Reminder: Bias-Variance Tradeoff

- Generalization error (aka out-of-sample error or risk)
  - Prediction error on *unseen* data
  - Related to overfitting
    - If the model overfits, then the generalization error will be large
- Bias-Variance Tradeoff
  - Generalization error: $\text{bias}^2 + \text{variance} + \text{irreducible error}$
    - For more details:
      - Geman et al. "*Neural networks and the bias/variance dilemma.*" Neural computation (1992)
      - Kohavi et al. "Bias plus variance decomposition for zero-one loss functions." ICML, 1996.
  - Why is it a tradeoff?
    - Increasing model complexity => lower bias
    - Decreasing model complexity => lower variance
    - Note: may not apply to neural networks in the same way!
      - E.g.: see Neal et al. "A modern take on the bias-variance tradeoff in neural networks." arXiv, 2018.

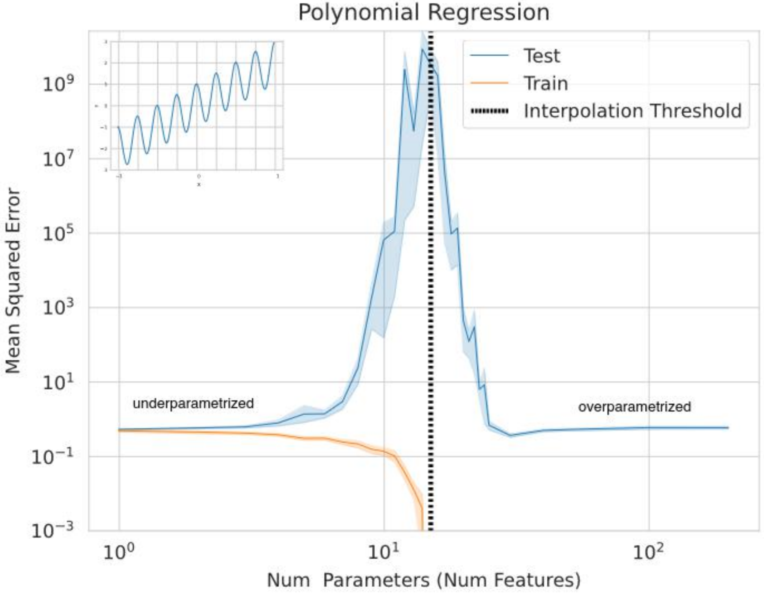
Phenomenon: Double Descent

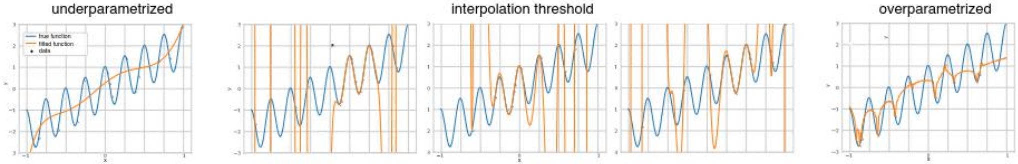
- Double Descent:
  - **Unexpected** and **sudden** change in **test error** as a function of model complexity (# of parameters)
  - Note: not all settings/models/data yield exactly two descents (e.g., sometimes there are three or more)
  - Ref for intuition: https://mlu-explain.github.io/double-descent/



Source: Rocks, J. W., & Mehta, P. (2022). Memorizing without overfitting: Bias, variance, and interpolation in overparameterized models. Physical review research, 4(1), 013201.

Phenomenon: Double Descent

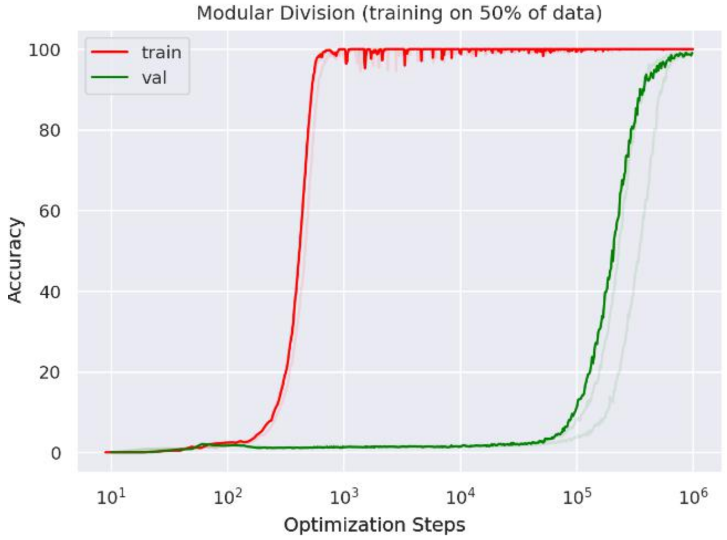




Source: "Double Descent Demystified: Identifying, Interpreting & Ablating the Sources of a Deep Learning Puzzle." Schaeffer et al.

Phenomenon: Grokking

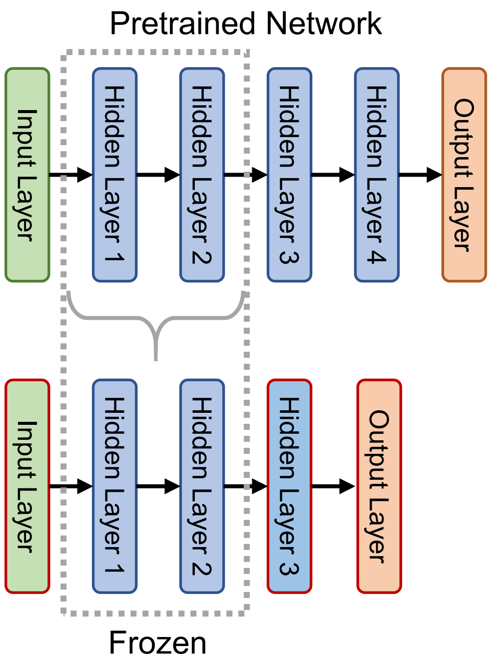
- Grokking:
  - Neural network **learns** to generalize suddenly well after the network has overfitted
  - Note: this occurs in very specific settings (specific tasks, small “algorithmic” datasets, complex neural networks)



Source: Power, A., Burda, Y., Edwards, H., Babuschkin, I., & Misra, V. (2022). Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets. arXiv e-prints, arXiv-2201.

Transfer Learning

- Should you train a deep neural network from scratch?
  - Not always. When possible you should use transfer learning:
    - Pick a pre-trained deep neural network in the same or related domain
    - Then fine-tune on the task you care about
- Reusing an existing deep neural network
  1. Pick some layers to reuse (typically the earlier layers)
  2. Freeze these layers
     - This will set the corresponding parameters as non-trainable
       - Optimization: you can actually ***cache*** the outputs of frozen layers for every input
  3. Add your own layers hidden layer(s)
  4. Replace or discard upper layers
     - You should always discard the existing output layer and use your own



Next Time

- Wednesday (3/6): Midterm Review

- Upcoming:
  - Homework 3 will be out soon

