الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Topic 3

التعلم العميق

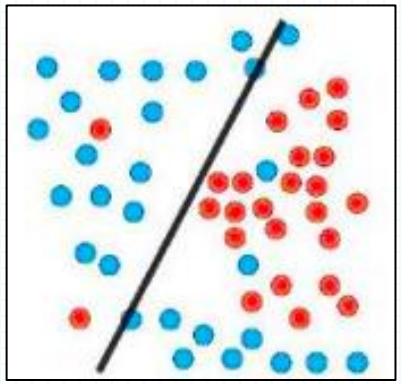
Generalization and Regularization Techniques in Deep Neural Networks

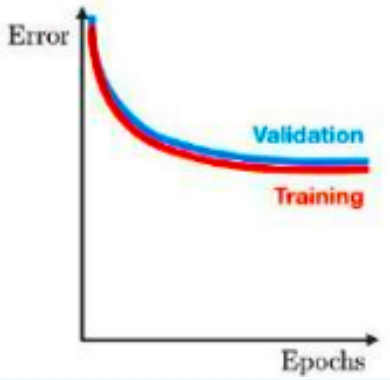
د. رياض سنبل

Generalization (Bias-Variance Tradeoff)

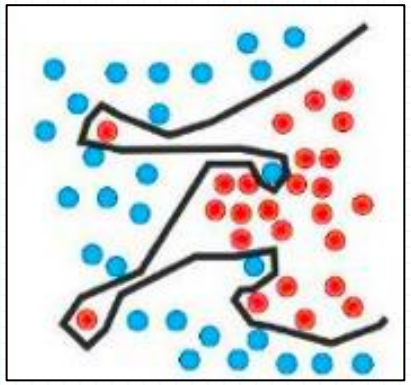
Generalization

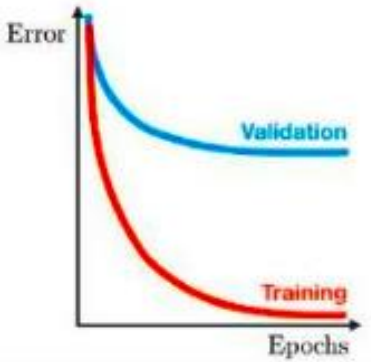
- ***Underfitting***
  - The model is too “simple” to represent all the relevant class characteristics
  - E.g., model with too few parameters produces high error on the training set and high error on the validation set



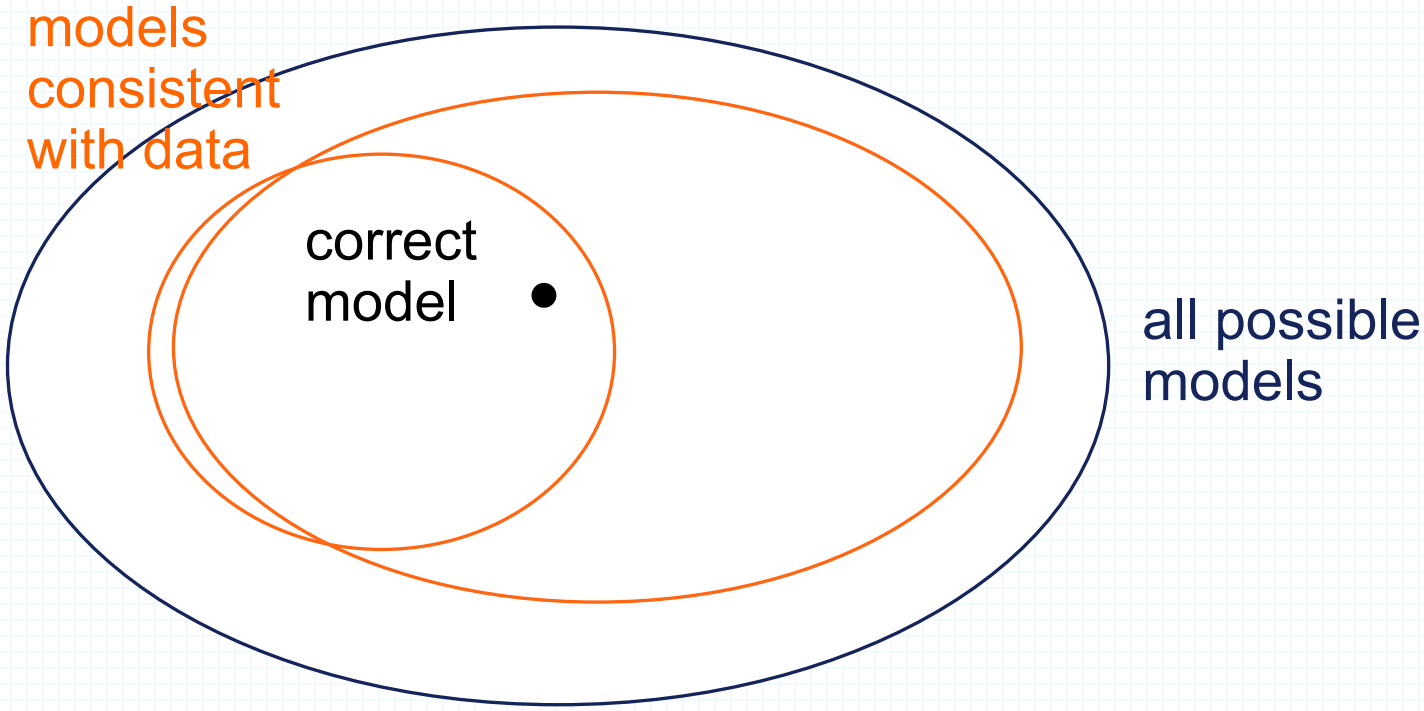


- ***Overfitting***
  - The model is too “complex” and fits irrelevant characteristics (noise) in the data
  - E.g., model with too many parameters produces low error on the training set and high error on the validation set



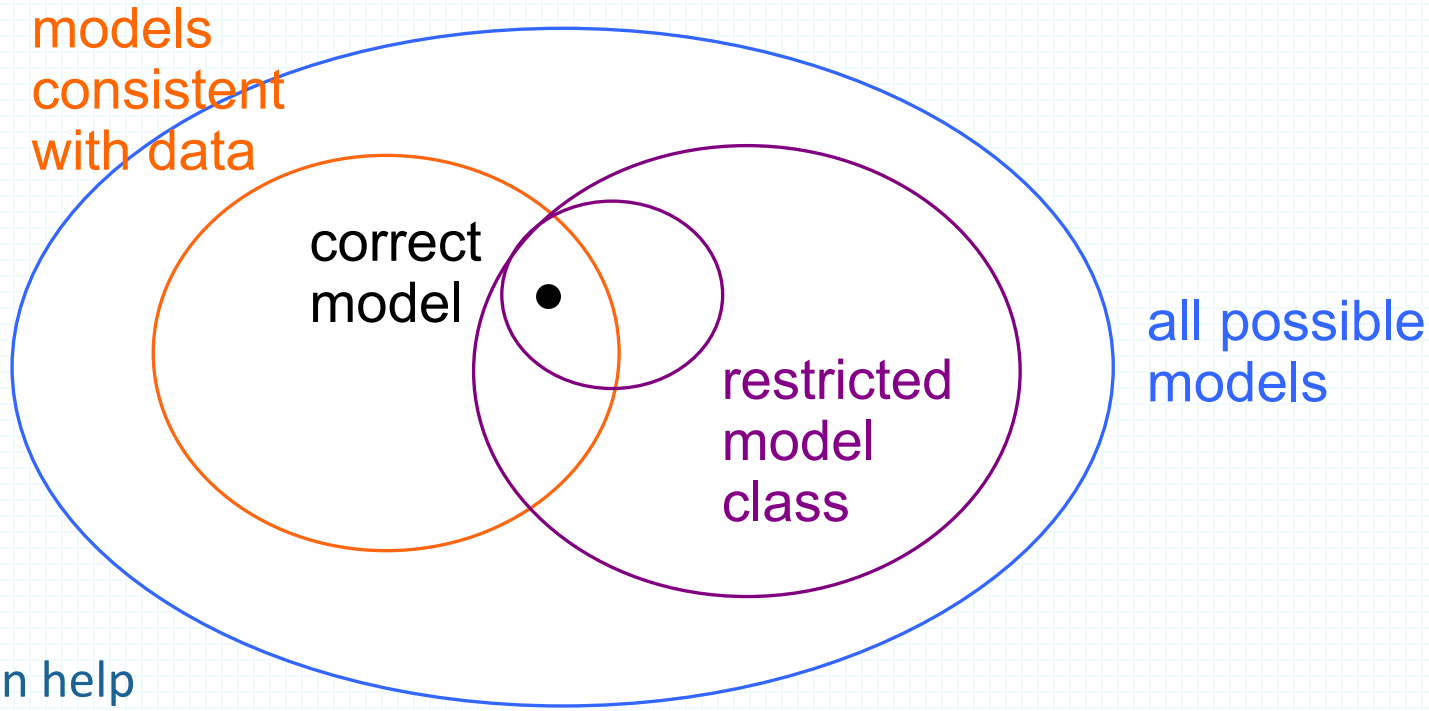


Back to the core idea in ML☺



More data can help!

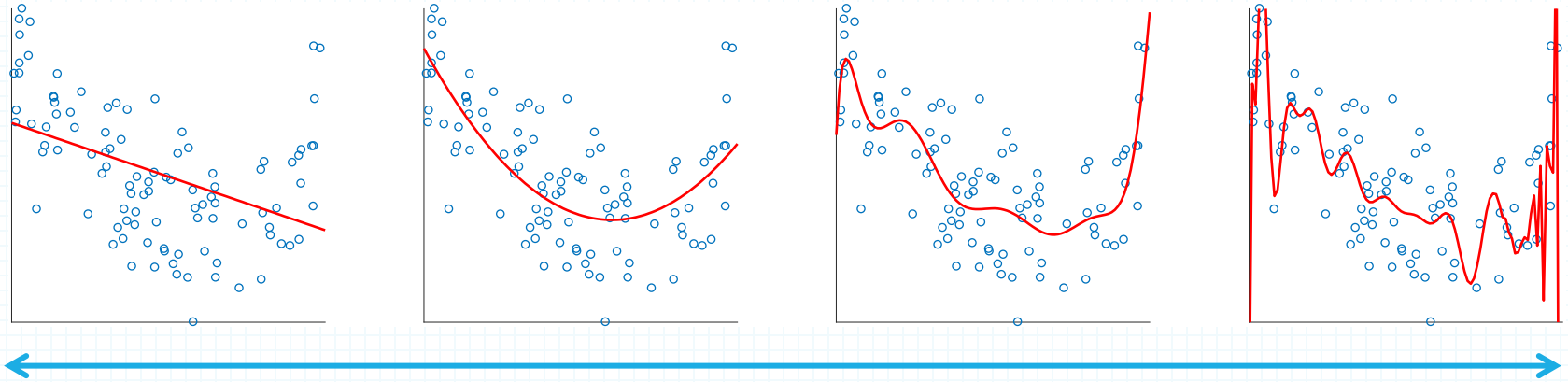
Back to the core idea in ML☺



- Restricting model class can help
- Or it can hurt
- Depends on whether restrictions are domain appropriate

Restricting Models

- Models range in their flexibility to fit arbitrary data



simple model

constrained

small capacity may prevent it from representing all structure in data

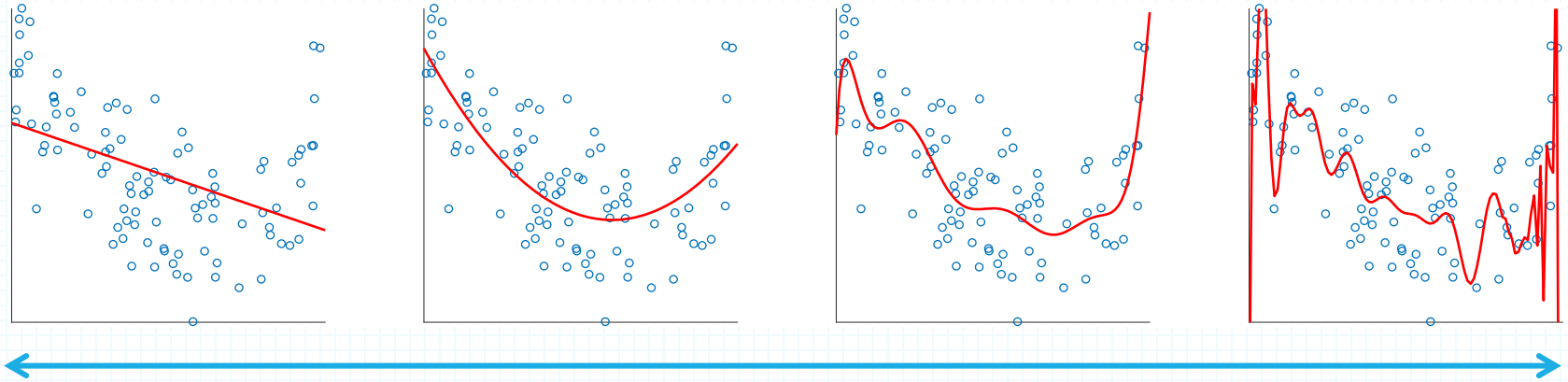
complex model

unconstrained

large capacity may allow it to fit quirks in data and fail to capture regularities

Bias vs Variance

- Models range in their flexibility to fit arbitrary data



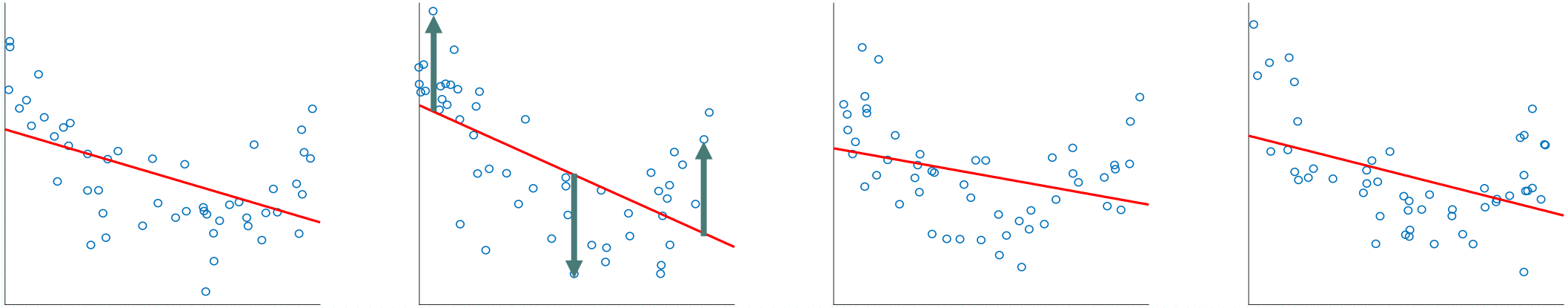
high bias

low variance

low bias

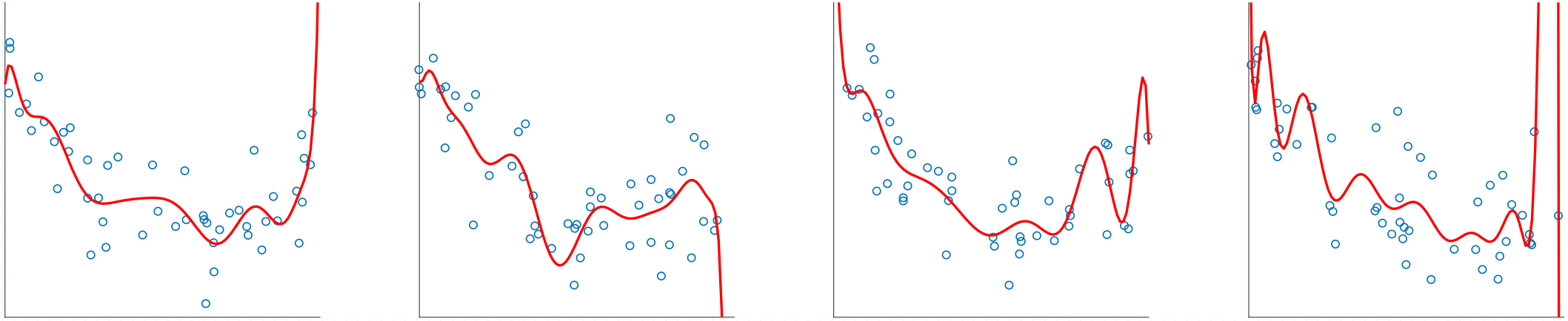
high variance

Bias



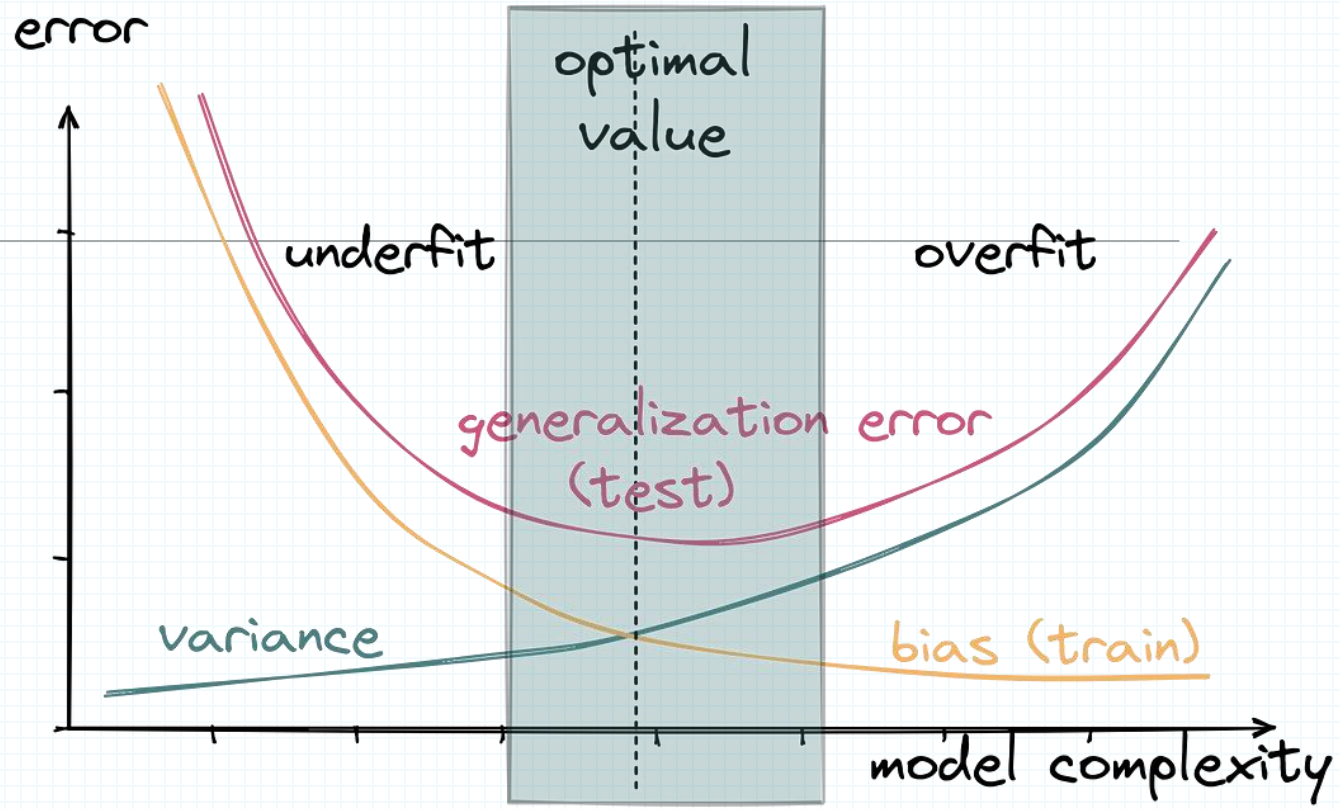
- Bias is the amount that a model's prediction differs from the target value, compared to the **training data**.
- Bias error **results from** simplifying the assumptions used in a model so the target functions are easier to approximate.
- Regardless of training sample, or size of training sample, model will produce consistent errors

Variance



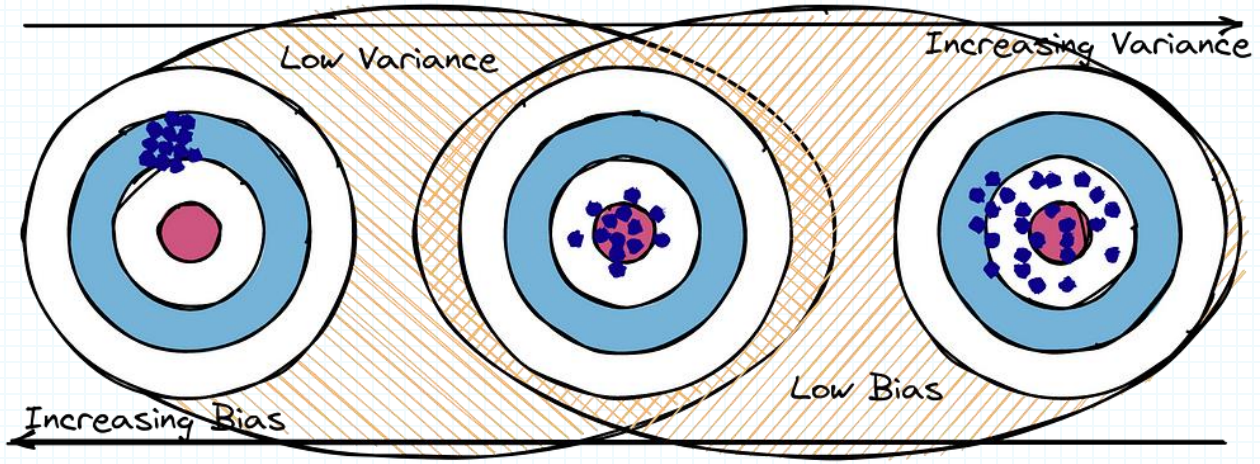
- Variance refers to **the changes in the model** when using different portions of the training data set.
- Simply stated, variance is the **variability in the model prediction**
- Different samples of training data yield different model fits

Bias-Variance Trade off



What does tons of training data do for us in terms of bias-variance trade off?

- ✓ Doesn't affect bias
- ✓ Reduces variance
- ✓ Allows more complex model to be used [shift of optimal complexity to the right]



Bias-Variance Trade off

Training set Error	1%	15%	15%	0.5%
Testing set Error	11%	16%	30%	1%
Bias				
Variance				

DOG vs. CAT

Bias-Variance Trade off

Training set Error	1%	15%	15%	0.5%
Testing set Error	11%	16%	30%	1%
Bias	Low	High	High	Low
Variance	High	Low	High	Low

DOG vs. CAT

Error = Irreducible Error + Bias + Variance

Solutions!

- **High Bias:**
  - Bigger Network?
  - Train Longer?
- **High Variance:**
  - More Data
  - Regularization.

$$Err(x) = E\left[(Y - \hat{f}(x))^2\right]$$

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Regularization Techniques in Deep Learning

Regularization Techniques in Deep Learning

- Regularization is a set of techniques that can prevent overfitting in neural.
- How to introduce regularization in deep learning models?
  - Weight Decay
  - Dropout
  - Data Augmentation
  - Early Stop

(1) Weight Decay

- **Overfitting may occur when you train a neural network too long.**
- Neural networks that have been over-trained are often characterized by having weights that are large.
  - A good NN might have weight values that range between -5.0 to +5.0 but a NN that is overfitted might have some weight values such as 25.0 or -32.0.
- So, one approach for discouraging overfitting is to prevent weight values from getting large in magnitude.
- Weight decay can be implemented by modifying the update rule for the weights such that the gradient is **not only** based on the training data **but also** on the weight decay term.

(1) Weight Decay

- ***ℓ_2 weight decay***

$$\mathcal{L}_{reg}(\theta) = \mathcal{L}(\theta) + \lambda \sum_k \theta_k^2$$

- ***ℓ_1 weight decay***

$$\mathcal{L}_{reg}(\theta) = \mathcal{L}(\theta) + \lambda \sum_k |\theta_k|$$

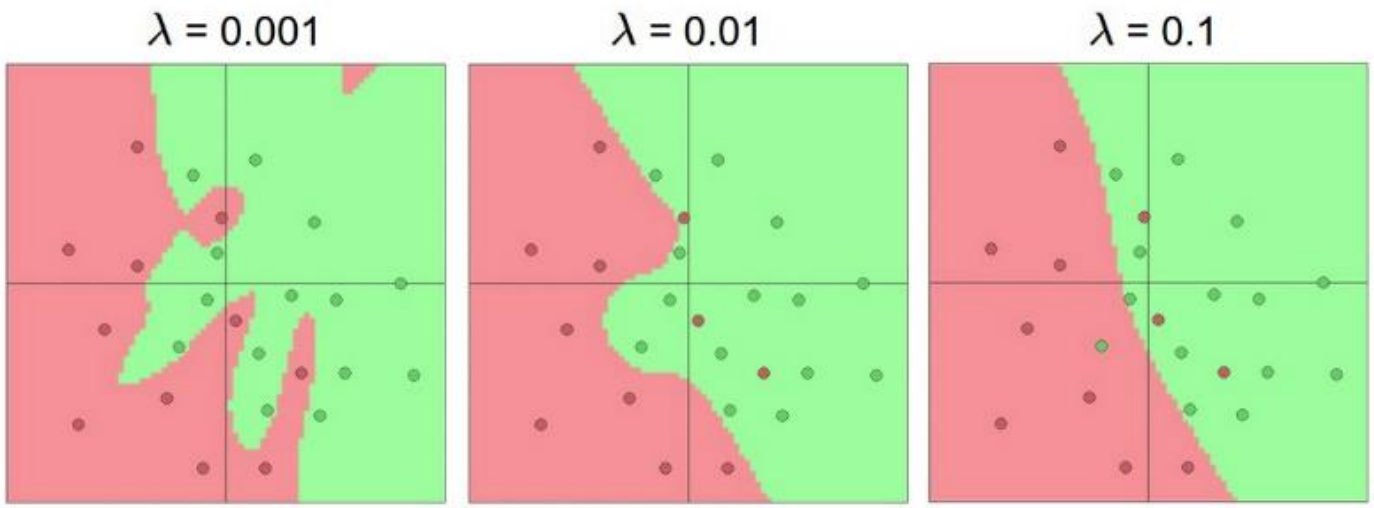
- ℓ_1 weight decay is less common with NN
  - Often performs worse than ℓ_2 weight decay
- It is also possible to combine ℓ_1 and ℓ_2 regularization
  - Called elastic net regularization

$$\mathcal{L}_{reg}(\theta) = \mathcal{L}(\theta) + \lambda_1 \sum_k |\theta_k| + \lambda_2 \sum_k \theta_k^2$$

- The weight decay coefficient λ determines how dominant the regularization is during the gradient computation

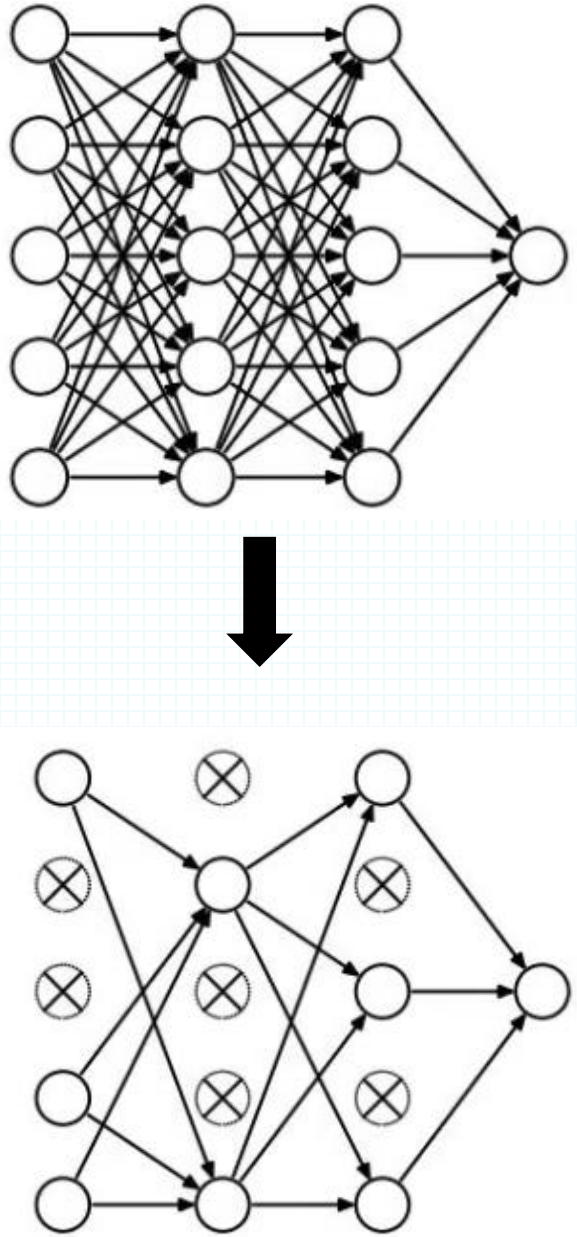
(1) Weight Decay

- Effect of the decay coefficient λ
  - Large weight decay coefficient $\longrightarrow$ penalty for weights with large values



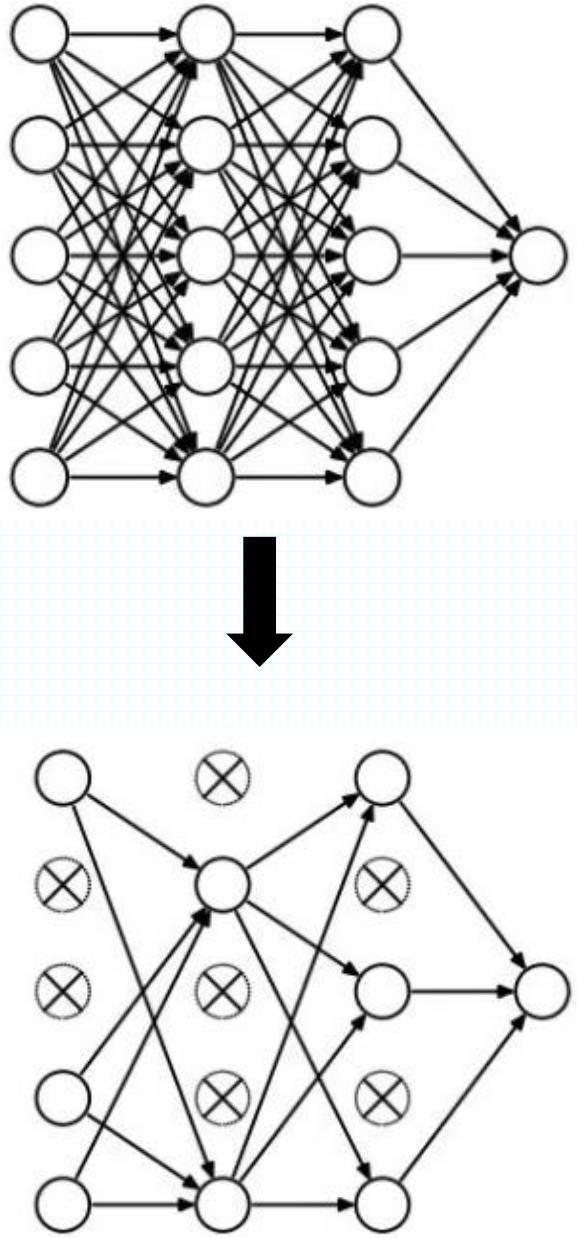
(2) Dropout

- **Overfitting may occur when Network layers co-adapt to correct mistakes from prior layers, in turn making the model more robust.**
  - units may change in a way that they fix up the mistakes of the other units.
  - This may lead to complex co-adaptations.
- At every iteration, we randomly selects some nodes and (temporary) removes them along with all of their incoming and outgoing connections as shown below.
- It increases the sparsity of the network and in general, encourages sparse representations!



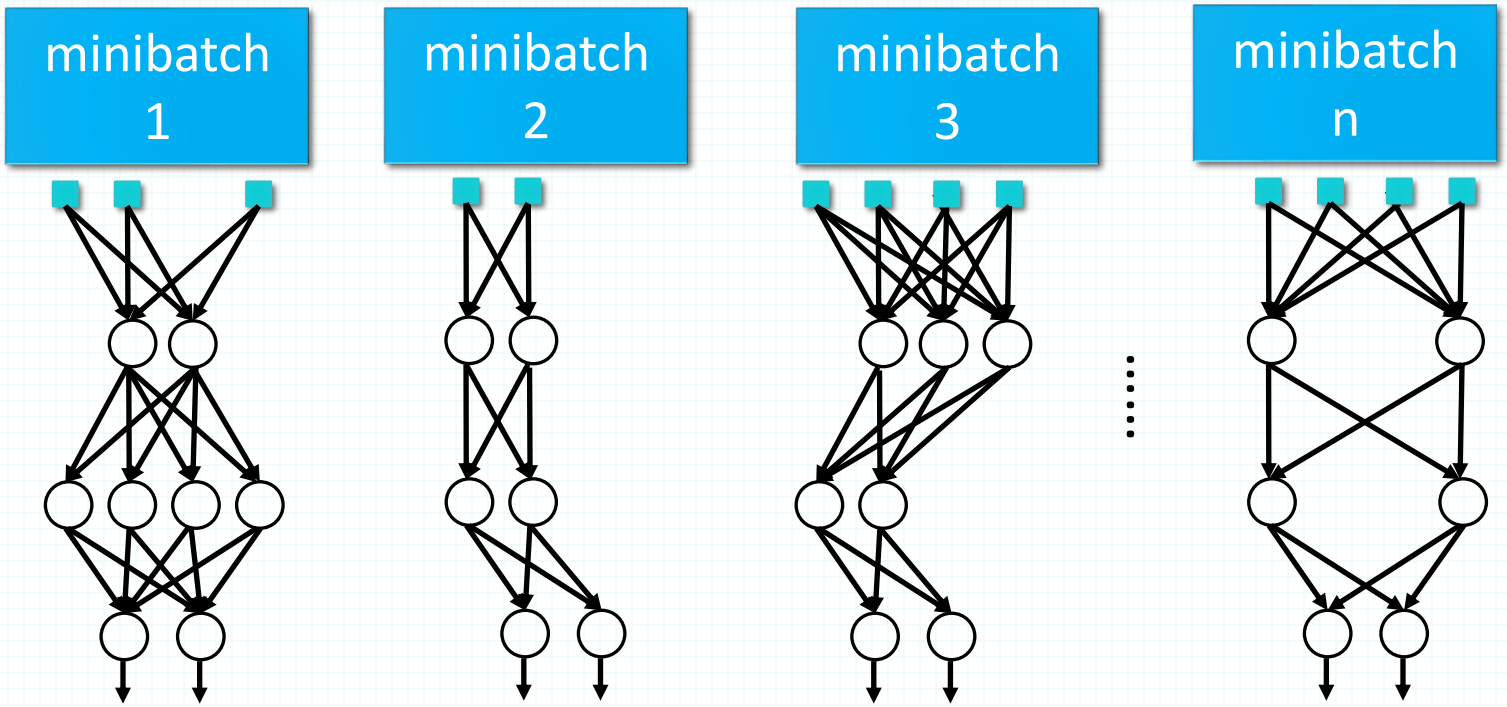
(2) Dropout

- Each unit is retained with a fixed dropout rate *p*, independent of other units
- The hyper-parameter *p* needs to be chosen (tuned)
  - Often, between 20% and 50% of the units are dropped
- **During test time, all units are present**



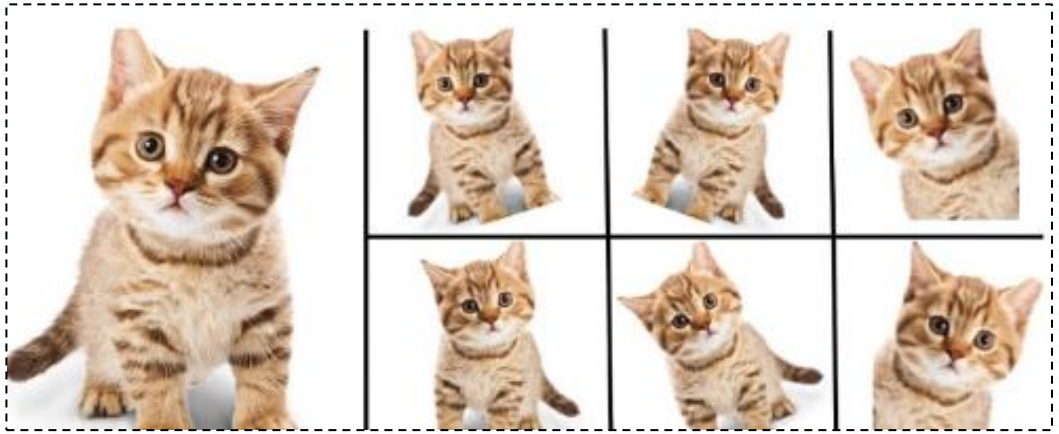
(2) Dropout

- Dropout is a kind of **ensemble** learning
  - Using one mini-batch to train one network with a slightly different architecture



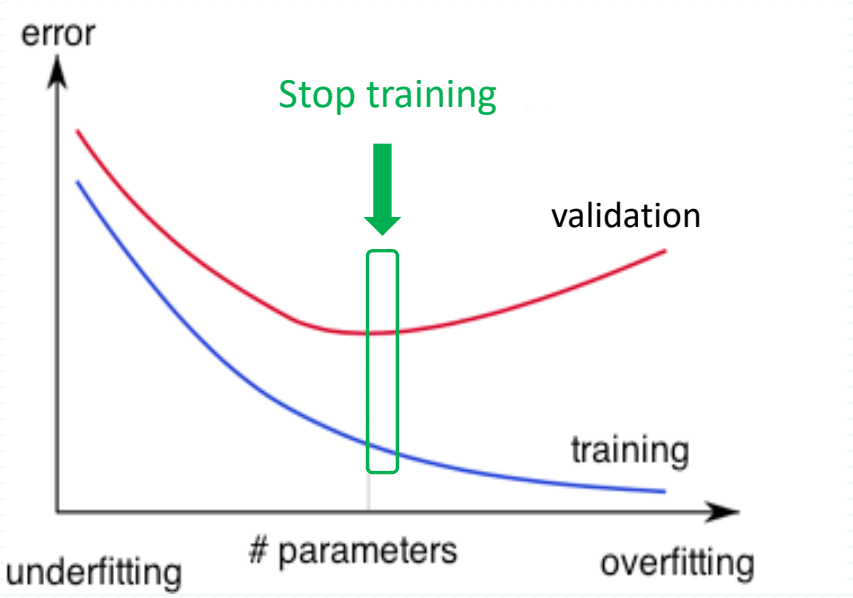
(3) Data Augmentation

- The simplest way to reduce overfitting is to increase the size of the training data.
- In machine learning, we were not able to increase the size of training data as the labeled data was too costly.
- **Solution:** creating new training examples by applying random transformations to your existing data,
  - Example: in computer vision => rotating, cropping, or flipping images.



(4) Early Stopping

- ***Early-stopping***
  - During model training, use a validation set
  - Stop when the validation accuracy (or loss) has not improved after *n* epochs
    - The parameter *n* is called patience



Hyper-parameter Tuning

- Training NNs can involve setting many ***hyper-parameters***
- The most common hyper-parameters include:
  - Number of layers, and number of neurons per layer
  - Initial learning rate
  - Learning rate decay schedule (e.g., decay constant)
  - Optimizer type
- Other hyper-parameters may include:
  - Regularization parameters (ℓ_2 penalty, dropout rate)
  - Batch size
  - Activation functions
  - Loss function
- Hyper-parameter tuning can be time-consuming for larger NNs