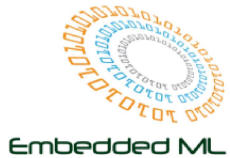


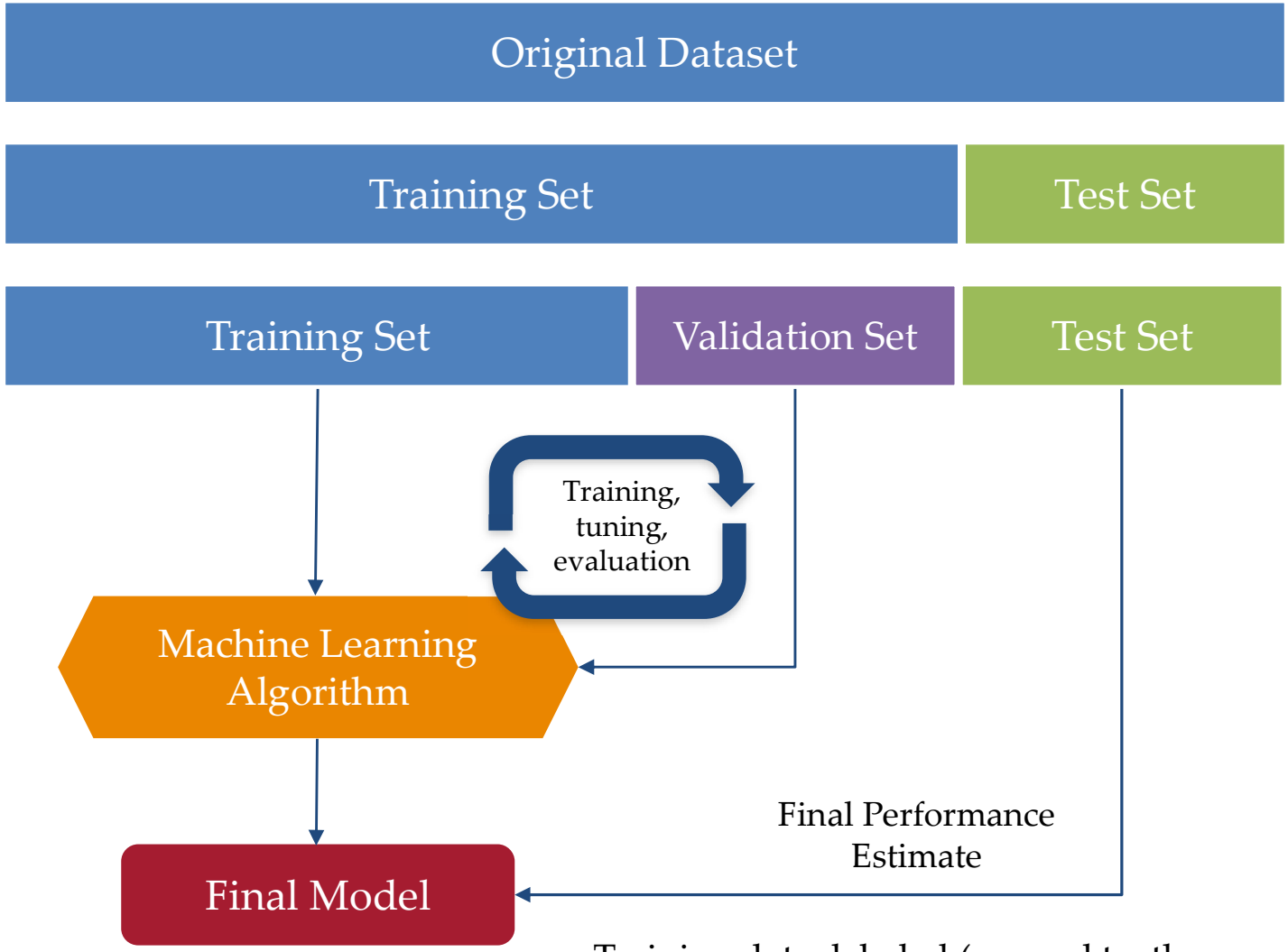
ML DATASET MODELING

ML Dataset Modeling

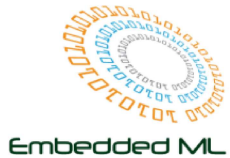
Embedded ML

- To build a machine learning models, data is split into two subsets - a training set and a test set.
- training set is used to train the model i.e. find the optimal parameters and fit the model to the data.
  - Training set is further split into training and validation set - Used for hyperparameter tuning and model selection. The model does not see this data during training.
- test set is used to evaluate the performance of the trained model on unseen data, to get an unbiased estimate of how well the model generalizes.
- relative sizes depend on the size of the original dataset, but a typical split would be 60% training, 20% validation and 20% test.
- three-way data split helps prevent overfitting, unbiased performance estimation and more robust model development.
- 10-fold cross-validation is a technique used with the training set to overcome overfitting and get a better measure of model performance.

ML Dataset Modeling



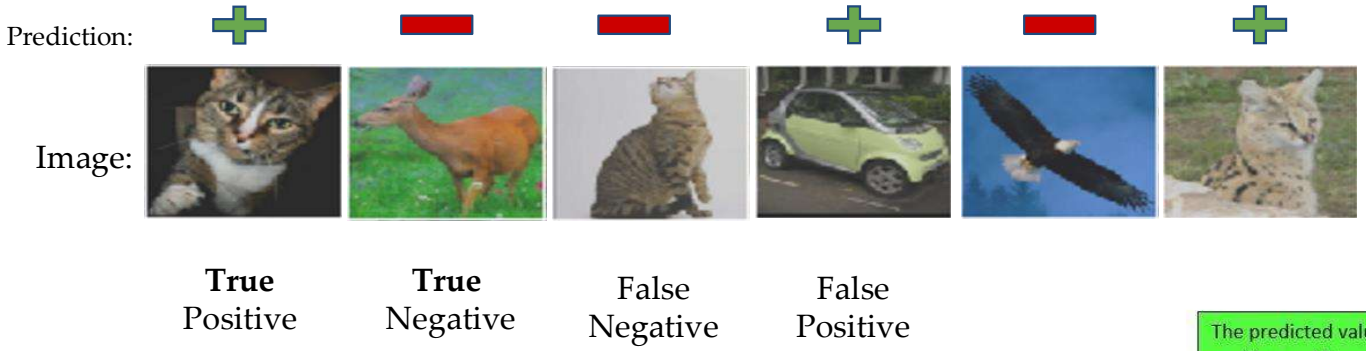
- Training data: labeled (ground truth - expected algorithm result; expensive!)
- Training: Algorithm uses labels to evaluate its accuracy on training data.
- Not much training data required. Risk of overfitting.



MACHINE LEARNING METRICS

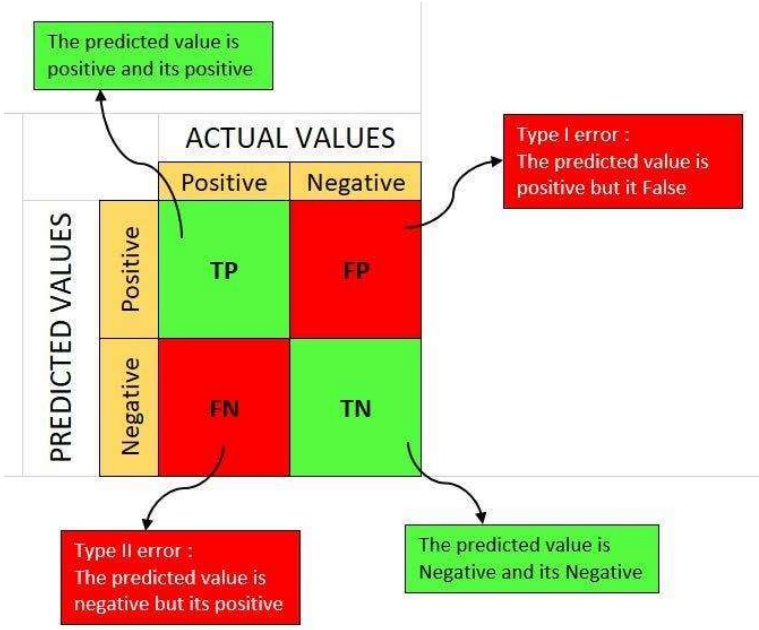
Measuring Success for Classification

- True Positive: Correctly identified as relevant
- True Negative: Correctly identified as not relevant
- False Positive: Incorrectly labeled as relevant (T1 Error)
- False Negative: Incorrectly labeled as not relevant (T2 Error)



Confusion matrix - Table showing correct and incorrect predictions by class.

- Diagonal shows correct predictions, off-diagonals show errors.
- Sums can be used to calculate various metrics like accuracy, precision, recall.



Precision, Recall, and Accuracy

Embedded ML

- Precision (predictive accuracy)
  - Percentage of positive labels that are correct
  - Precision = (# true positives) / (# true positives + # false positives)
- Recall (sensitivity)
  - Percentage of positive examples that are correctly labeled
  - Recall = (# true positives) / (# true positives + # false negatives)
- Accuracy
  - Percentage of correct labels
  - Accuracy = (# true positives + # true negatives) / (# of samples)
- F1-score
  - Harmonic mean of precision and recall. Provides a balance between the two.

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

Bias & Variance

- Bias & Variance
  - bias and variance both indicate how well a model is likely to perform on new unseen data.
- Bias
  - expected difference between model's prediction and truth
- Variance
  - how much the model differs among training sets

- Model Scenarios
  - High Bias (underfitting): Model makes inaccurate predictions on training data
  - High Variance (overfitting): Model does not generalize to new datasets
  - Low Bias: Model makes accurate predictions on training data
  - Low Variance: Model generalizes to new datasets

Optimal Measures

Lower bias and variance, along with high accuracy, precision and recall indicate a robust, generalized model.

Precision <> Recall

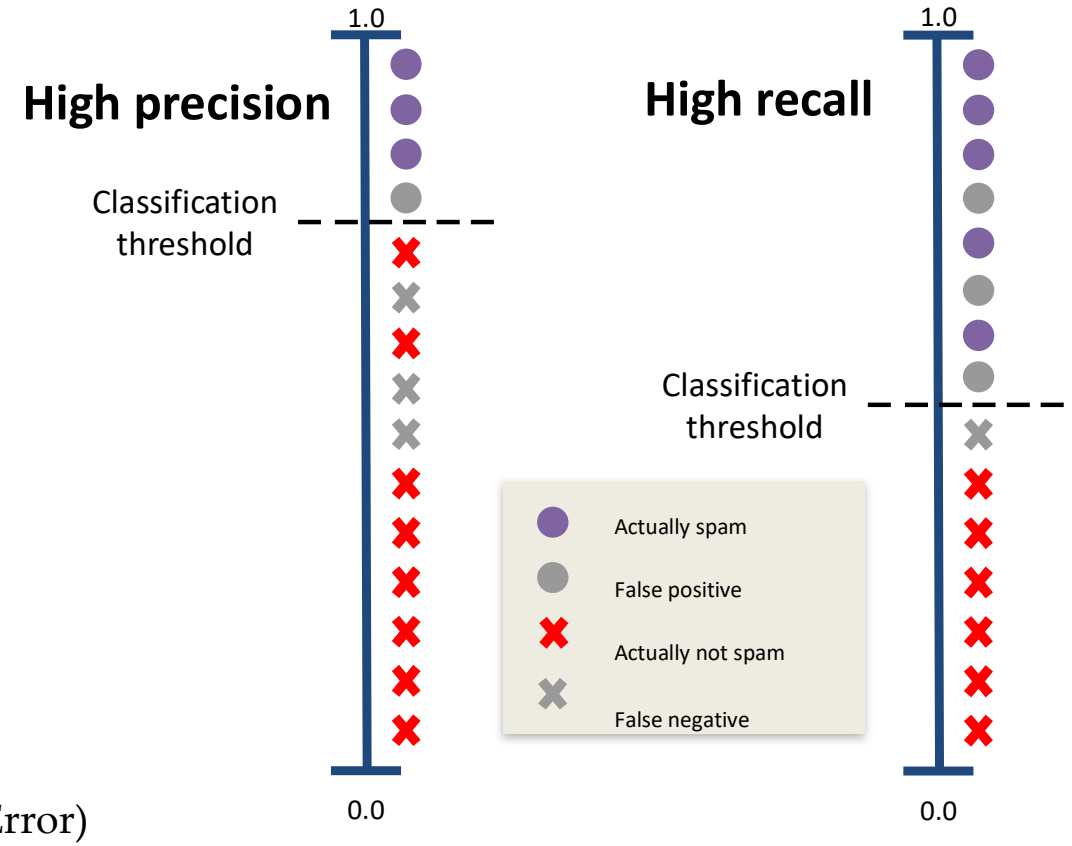
- If we optimize our model for high precision
  - minimize *Type 1* Errors (fewer false positives)
  - increase *Type 2* Errors (more false negatives)

- If we optimize our model for high recall
  - minimize *Type 2* Errors (fewer false negatives)
  - increase *Type 1* Errors (more false positives)

True Positive: Correctly identified as relevant
True Negative: Correctly identified as not relevant
False Positive: Incorrectly labeled as relevant (T1 Error)
False Negative: Incorrectly labeled as not relevant (T2 Error)

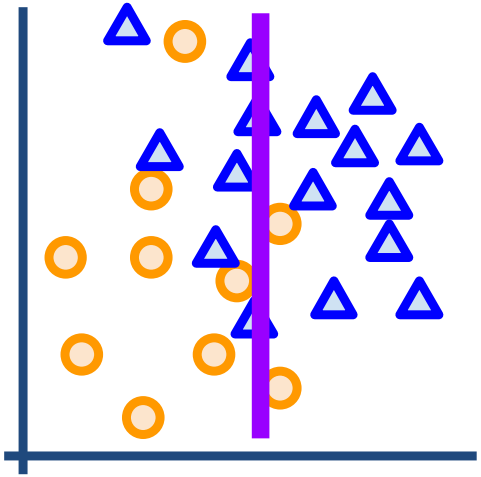
Precision <> Recall

- If *Type 1* errors are most harmful, then precision is more important
- If *Type 2* errors are most harmful, then recall is more important

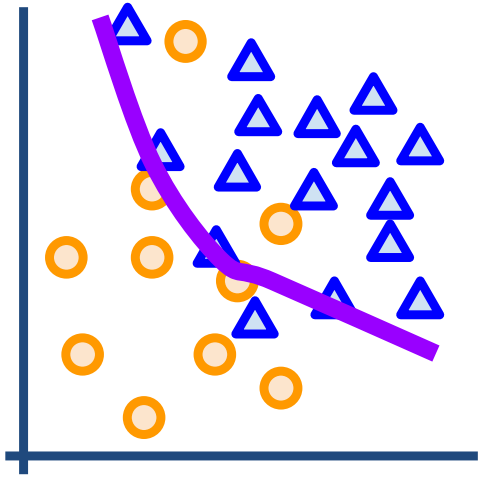


True Positive: Correctly identified as relevant
True Negative: Correctly identified as not relevant
False Positive: Incorrectly labeled as relevant (T1 Error)
False Negative: Incorrectly labeled as not relevant (T2 Error)

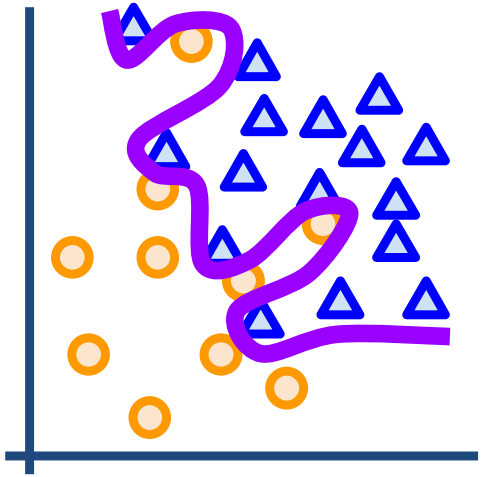
Training: Underfitting vs. Overfitting



Underfit: Model fails to capture trends in the data



Good fit: Model captures trends and can generalize to unseen data



Overfit: Model captures training data trends but fails on unseen data

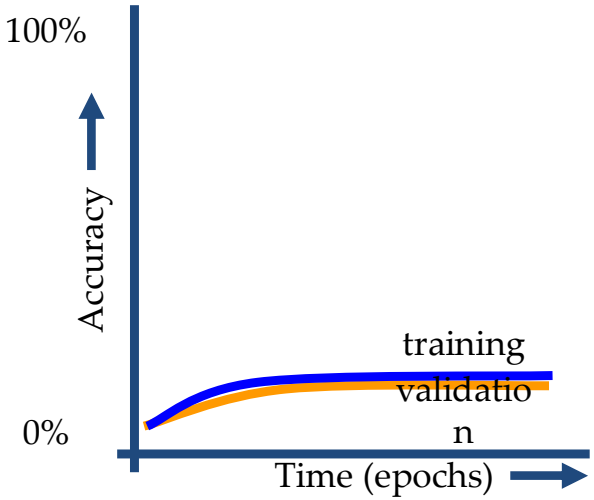
Training Time

- Epoch refers to the number of passes of the full training dataset made during model training.

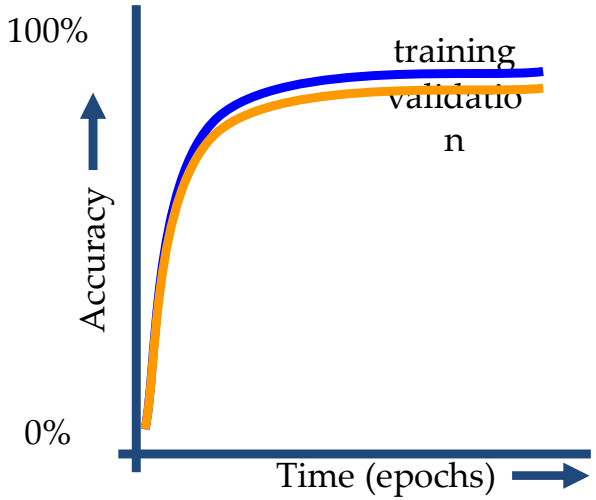
- More Epoch
  - Generally, improves accuracy, can lead to over fitting.
  - Precision and recall may also artificially increase, no model improvement.

- Optimal Epoch
  - validation accuracy, precision and recall hit their peak before overfitting begins.
  - greatly based on model and data complexity.

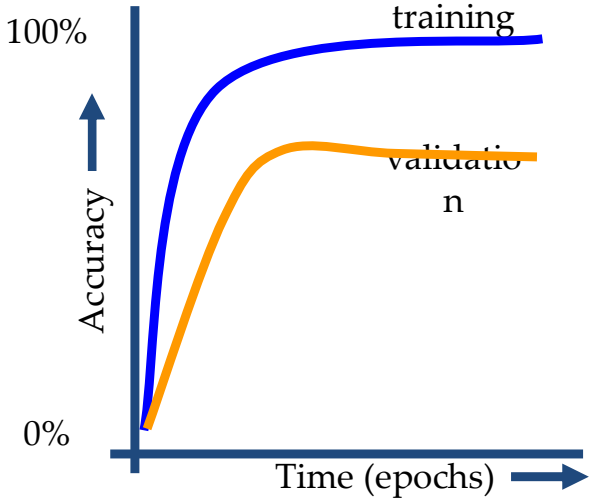
Spotting Underfitting and Overfitting



Underfit: Model performs poorly on training and validation data

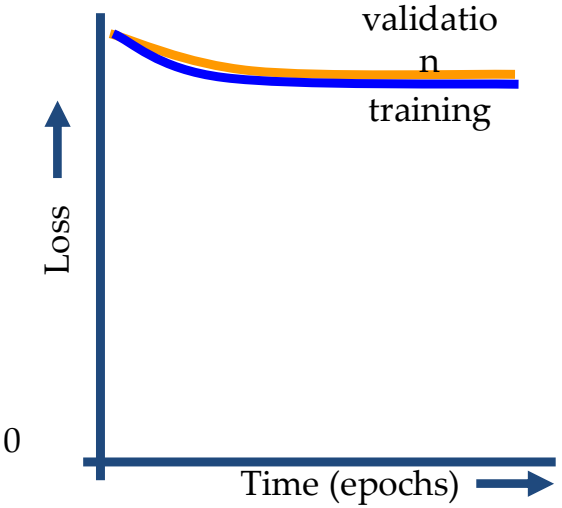


Good fit: Model generalizes well from training to validation data

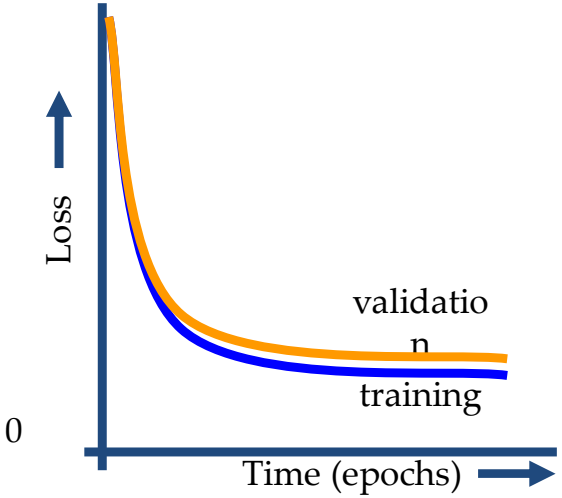


Overfit: Model predicts training data well but fails to generalize to validation data

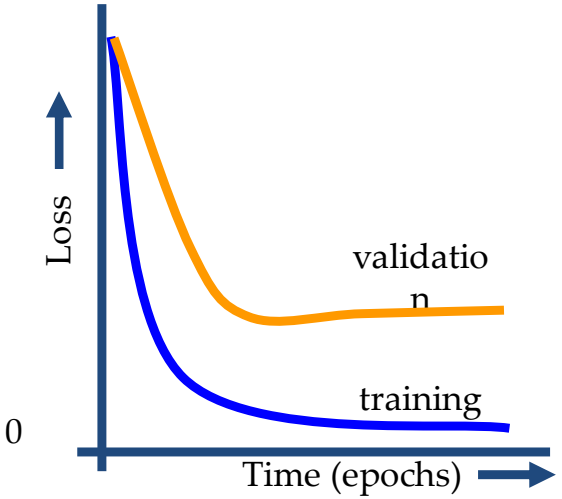
Spotting Underfitting and Overfitting



Underfit: Model performs poorly on training and validation data

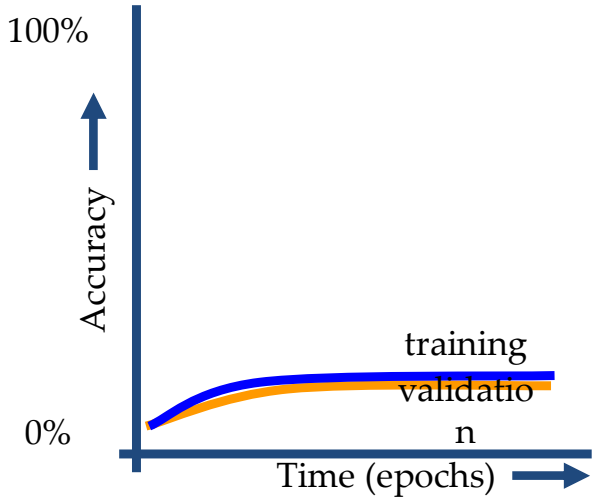


Good fit: Model generalizes well from training to validation data



Overfit: Model predicts training data well but fails to generalize to validation data

Fixing Underfitting

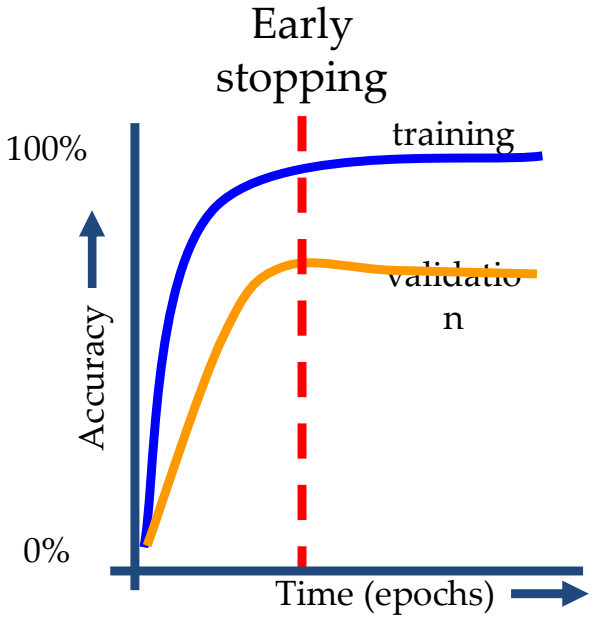


Underfit: Model performs poorly on training and validation data

- Get more data
- Try different features or more features
- Train for longer
- Try a more complex model (more layers, more nodes, etc.)

Fixing Overfitting

- Get more data
- Early stopping
- Reduce model complexity
- Add regularization terms
- Add dropout layers (for neural networks)



Overfit: Model predicts training data well but fails to generalize to validation data