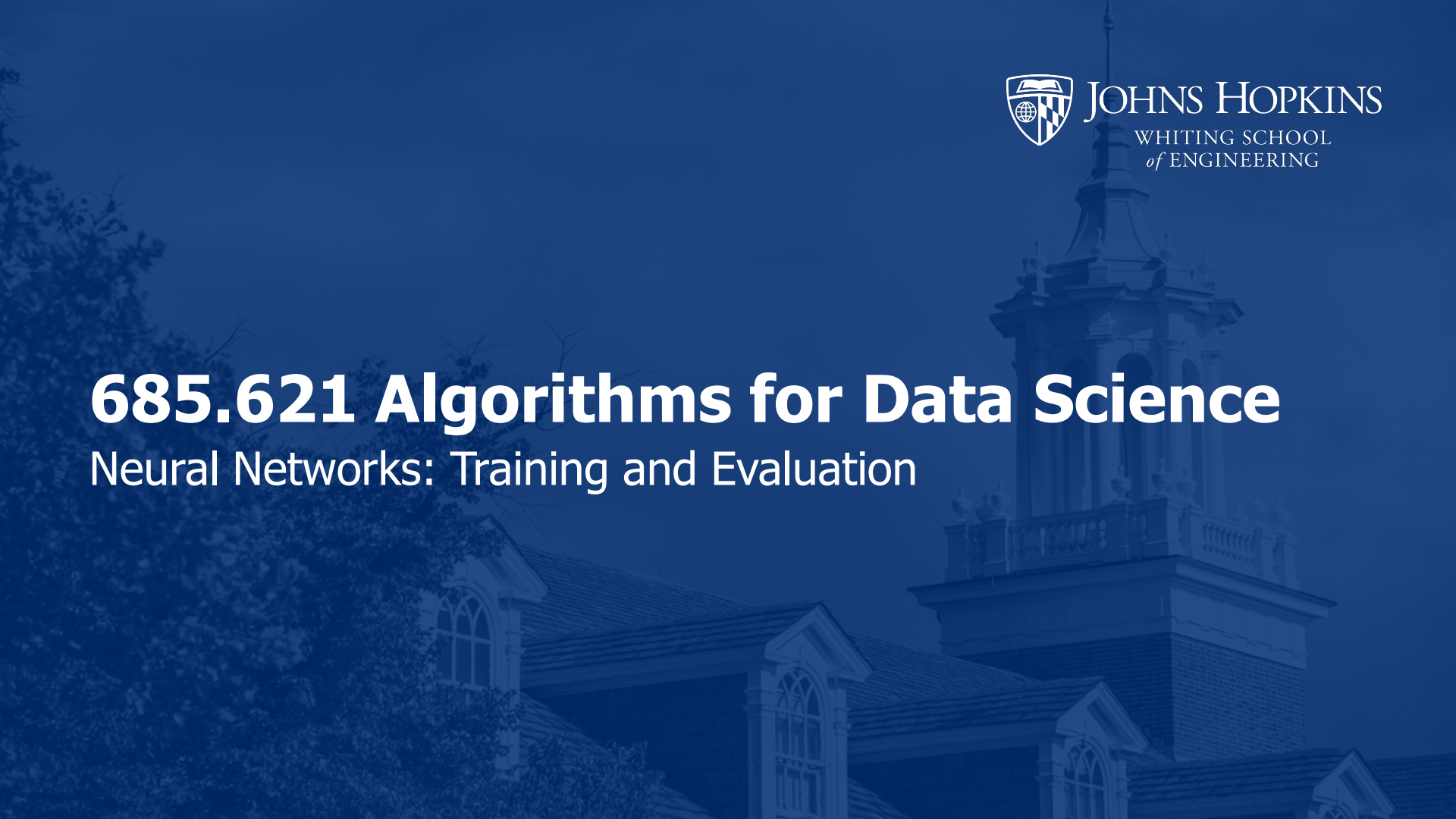
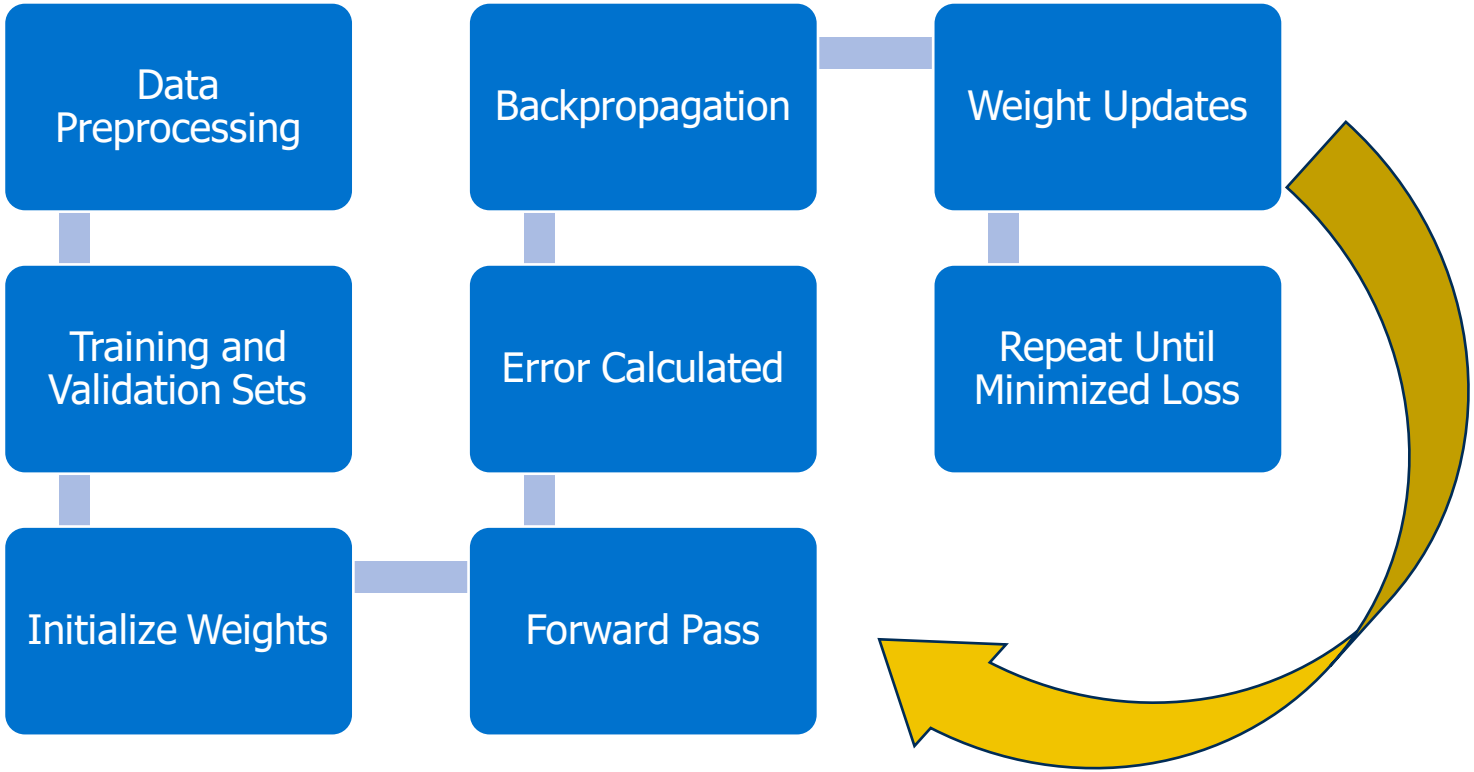


685.621 Algorithms for Data Science

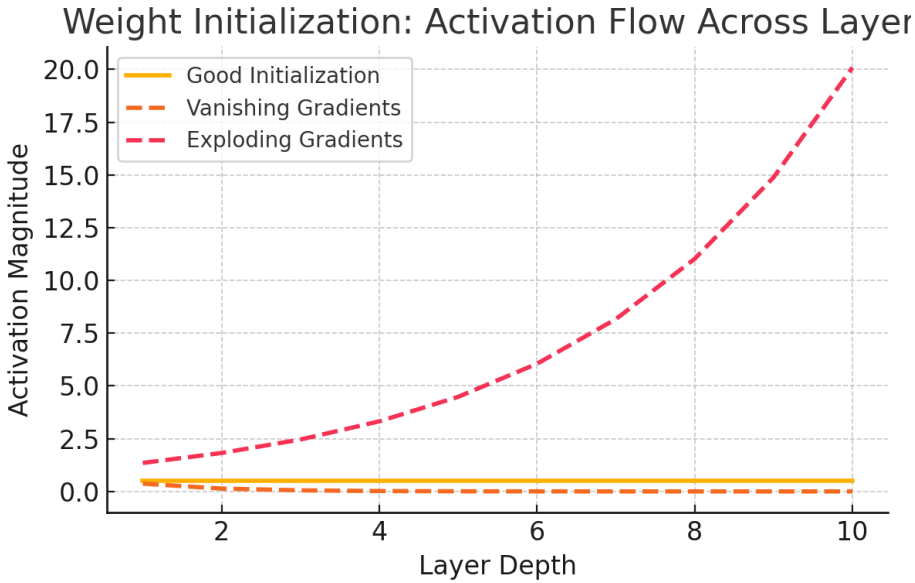
Neural Networks: Training and Evaluation

Training Pipeline Overview



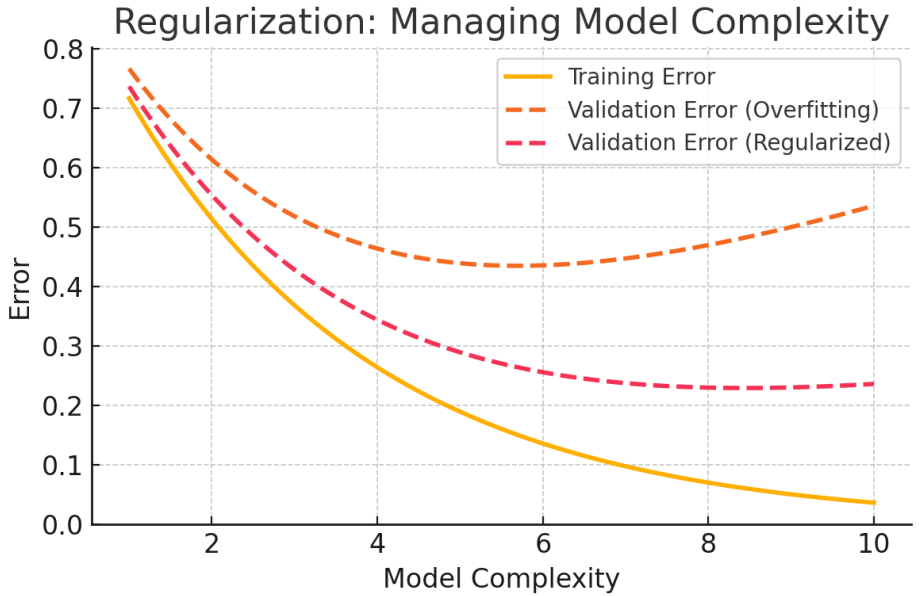
Weight Initialization

- To aid in the convergence of networks using popular frameworks, in both PyTorch and TensorFlow, **weight initialization is usually randomized by default.**
- The strategy depends on the following:
  - The layer type
  - The activation function
  - The library defaults (He initialization for ReLU by default in PyTorch)



Regularization Techniques

- **L2 Regularization**: Penalizes large weights
- **Dropout**: Randomly zeroes out neuron activations, forcing more robust representations.
- **Batch Normalization**: Normalizes activations within each mini-batch, stabilizing training and improving convergence.



Hyperparameter Tuning

- **Batch Size**
  - **Typical Start**: 32, 64, or 128
    - **Rule of thumb**: Smaller batch sizes give noisier gradient estimates but generalize better.
    - Larger batch sizes give smoother gradients and faster computation but risk poor generalization.
- **Neurons per Layer**
  - **Typical Start**: Same order of magnitude as input features.
    - If you have 100 input features, try hidden layers with 64–128 neurons.
    - **Deeper layers often shrink**: funnel architecture (e.g., 128 → 64 → 32).
- **Learning Rate**
  - **Typical Start**: 0.001 (especially with Adam optimizer)

Training Challenges

Challenge	What Happens	Why It Happens	Solutions
Vanishing Gradients	Training stalls, early layers don't learn	Gradients shrink exponentially in deep networks	ReLU activations, He initialization, skip connections (ResNet)
Exploding Gradients	Loss blows up, training unstable	Gradients grow exponentially in deep networks	Gradient clipping, proper initialization, normalized architectures
Overfitting	Great training accuracy, poor test accuracy	Model memorizes training data, lacks generalization	Regularization (L1/L2), dropout, data augmentation, early stopping
Underfitting	Poor accuracy on both training and test sets	Model too simple for the data complexity	Increase model capacity (layers/neurons), train longer, better features
Slow Convergence	Training takes too long	Poor learning rate, bad initialization, noisy gradients	Learning rate scheduling, batch normalization, adaptive optimizers (Adam)