Intro to Artificial Intelligence

Ethan Hartley[1, 2]

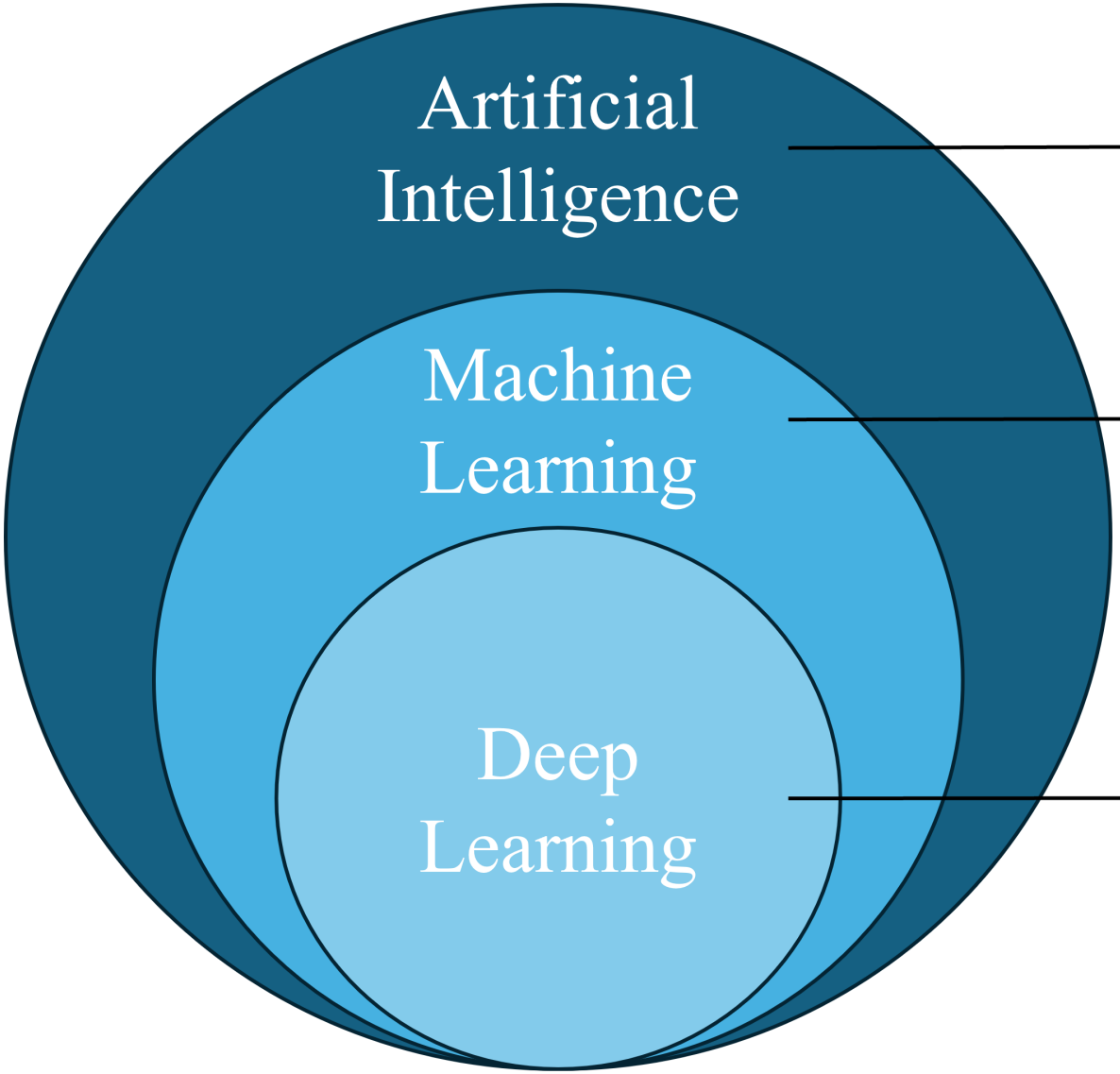
[1]University of Hawai'i at Mānoa

[2]U.S. Department of Energy
(Oak Ridge National Laboratory)

Outline

1. Fundamentals
2. Training AI & ML
3. Causal ML
4. Recent Advancements in AI

AI vs. ML vs. DL



Creating systems that attempt to imitate human intelligence.

Creating computational models that learn patterns and relationships without explicit programming.

Machine learning that enables the model to actively learn from the data.

Types of Learning

Supervised Learning – Learn from labeled data to make predictions.

- Classification – predict categories (e.g., treatment)
- Regression – predict numbers (e.g., price forecasting)

Unsupervised Learning – Find patterns in unlabeled data.

- Clustering – group similar items (e.g., customer segments)
- Dimensionality Reduction – reduce complexity (e.g., PCA)

When to Use Machine Learning (ML)?

Machine Learning (ML) and Deep Learning (DL) are powerful tools, but they are not universally applicable. These methods are most effective when:

- **Data volume is high:** ML models thrive on large datasets.
- **Data is complex:** Unstructured or high-dimensional data (e.g., images, text, time series).
- **Relationships are nonlinear:** Traditional models may fail to capture intricate dependencies and interactions.

ML is not a silver bullet: classical statistical principles still apply.

- For example, if your data-generating process is linear and assumptions hold, then by the Gauss-Markov theorem, the Ordinary Least Squares (OLS) estimator is BLUE (Best Linear Unbiased Estimator).

Learning by Minimizing Loss

The core goal of machine learning is to **learn patterns from data** to make predictions or decisions.

- Conceptually, this means we are trying to **minimize a loss function** — a way of quantifying how wrong the model's predictions are.
- The **loss function is highly customizable**, depending on the specific task:
  - Mean Squared Error (MSE) for regression tasks
    - Simple loss functions can easily be augmented with various regulatization penalties (L1 or L2 are very common)
  - Cross-Entropy Loss for classification tasks
  - Custom loss functions for specialized goals (e.g., fairness, robustness)

Choosing the right loss function is critical and defines what the model learns to optimize.

OLS: Two Views, One Goal

- **Statistical Modeling (OLS Closed-Form):**

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

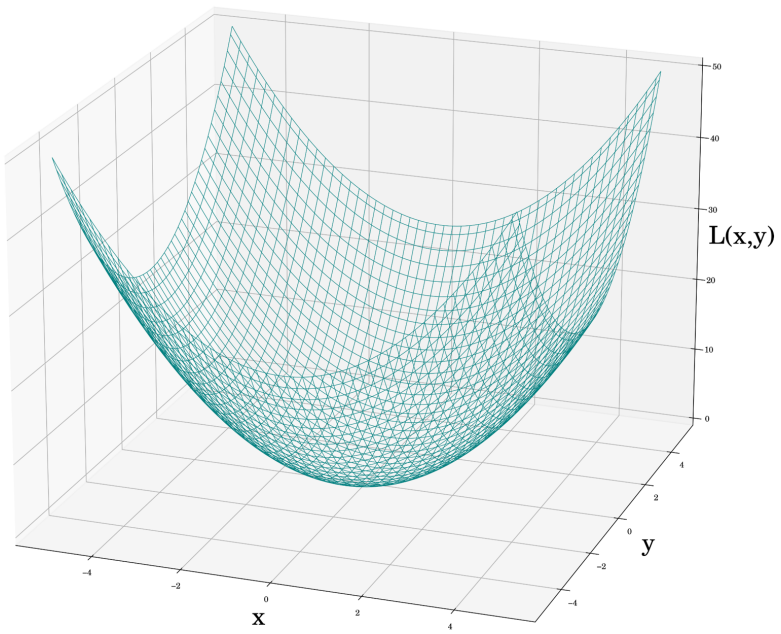
- **Machine Learning (Loss Minimization):**

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (y_i - X_i \beta)^2 \quad \text{(MSE Loss)}$$

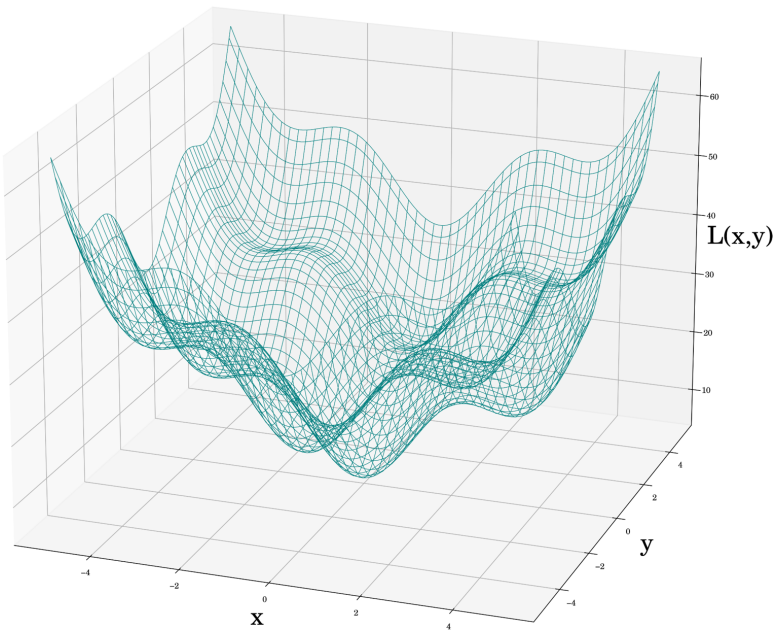
- **Key Insight:** Both approaches minimize the same objective — the sum of squared errors. The use of gradient descent with appropriately tuned hyperparameters in ML frameworks will result in an identical solution to the closed-form optimum in OLS.

Classical Optimization vs. Machine Learning

Convex Loss Function



Non-Convex Loss Function



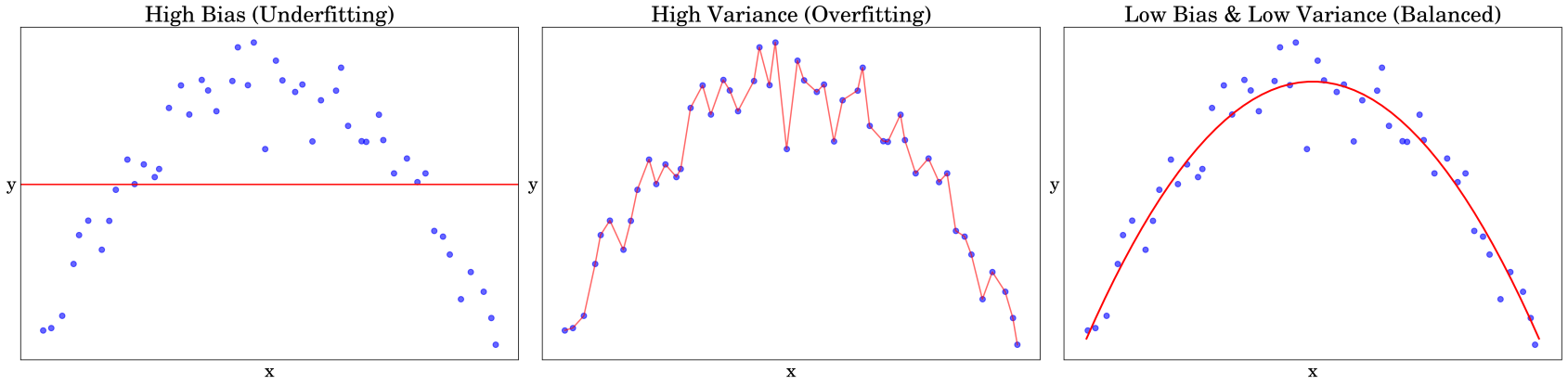
The Learning Process

- Learning typically happens through an **iterative optimization process**.
- Most models start with **random initial parameters**.
- Using the data, we:
  - Compute predictions based on current parameters.
  - Evaluate the **loss function** to see how far off those predictions are.
  - Use the **gradient of the loss** to update the parameters in a direction that reduces the loss.
- This process is often powered by **gradient descent** or its variants (e.g., Adam, RMSProp).
- Repeated over many iterations (epochs), this **minimizes the loss** and improves the model's performance.

Bias-Variance Tradeoff

A central challenge in ML is managing the **bias-variance tradeoff**, which affects model generalization:

- **Bias:** Difference between average prediction and the true value.
- **Variance:** Variability of predicted values for an input and model.



Goal: Achieve low generalization error by finding a sweet spot between bias and variance.

The Curse of Dimensionality

The **curse of dimensionality** refers to various problems that arise as the number of input features (d) increases:

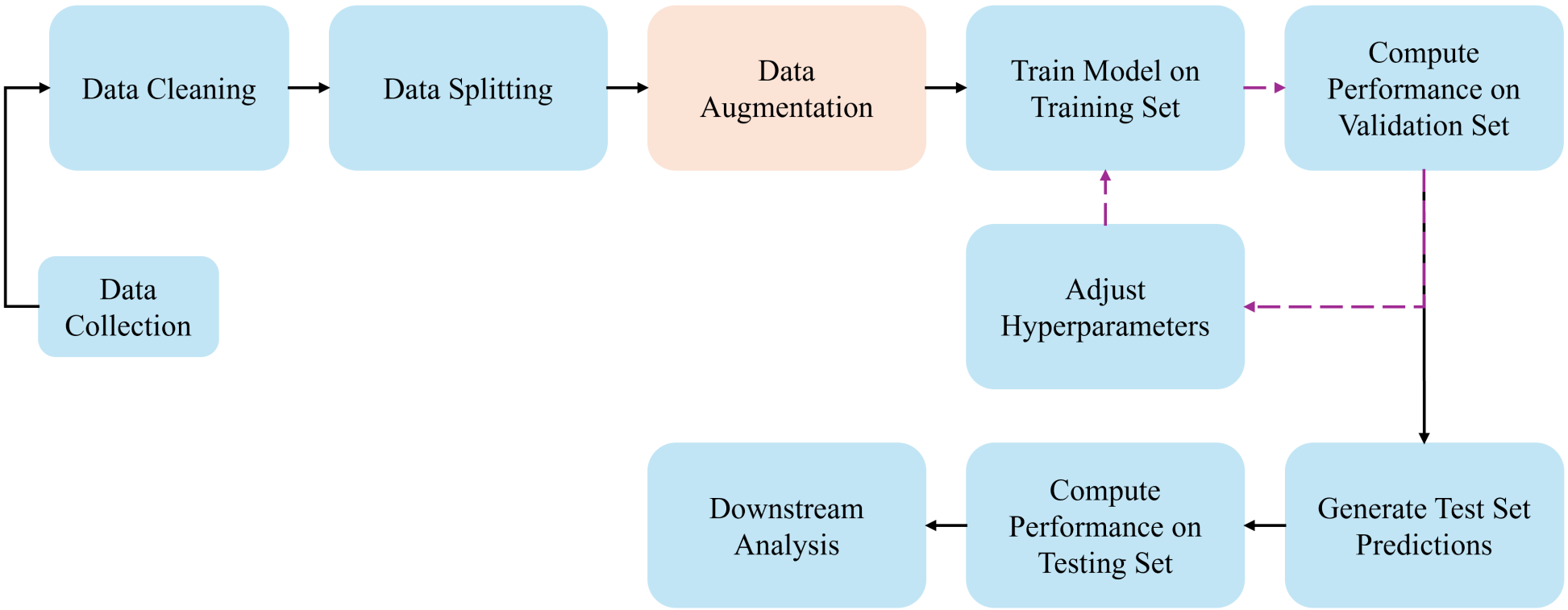
- Data becomes sparse in high-dimensional space.

In traditional statistical methods, good performance typically requires $N \gg d$. However, modern ML and DL often operate in the opposite regime ($d \gg N$).
Why it works now:

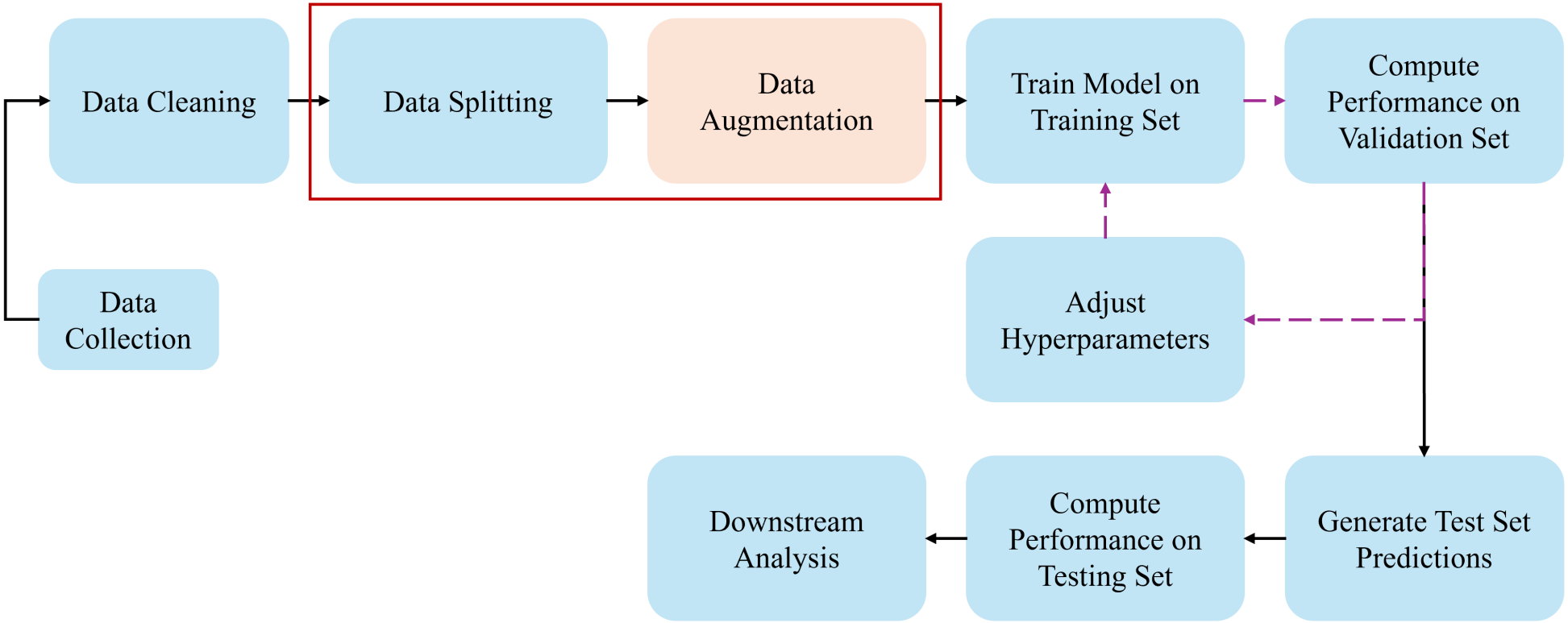
- Deep architectures can exploit hierarchical representations.
- Regularization techniques and computationally efficient hyperparameter tuning help prevent overfitting.

Overparameterization, counterintuitively, can help generalization in neural networks due to implicit regularization.

Model Development



Data Splitting & Cross Validation



Train-Validation-Test Splits

Splitting Data:

- **Training Set:** The portion of the dataset used to train the model. It allows the model to learn the patterns and relationships in the data.
- **Validation Set:** A separate portion of the dataset used to tune hyperparameters and evaluate the model during the training process. It helps to avoid overfitting and guides model selection.
- **Test Set:** The portion of the dataset that is held out and only used for final evaluation after the model has been trained. It gives an unbiased estimate of the model's performance on unseen data.

General Guidelines

General Guidelines for Splitting:

- Common split ratios: 60%-80% for training, 10%-20% for validation, and 10%-20% for testing.
- Ensure random sampling to minimize bias and representativeness in each set.
  - Remember: setting your seed allows for replicable randomness.
- If your data is time-series or has sequential dependencies, be sure to split chronologically to preserve temporal ordering.
  - You may need to condition your splits on other fundamental attributes depending on the structure of the underlying data.

Cross Validation

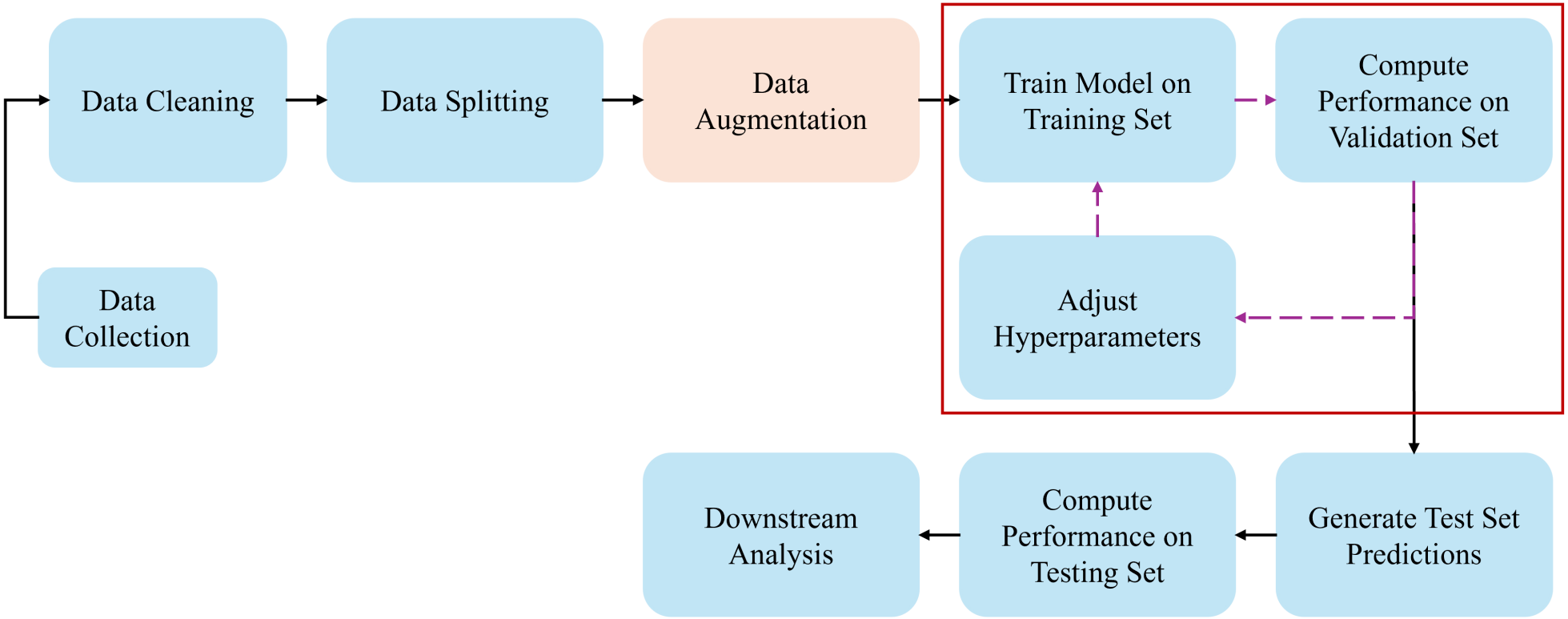
- **K-Fold Cross Validation:** The dataset is split into K subsets (or folds). The model is trained K times, each time using $K-1$ folds for training and the remaining fold for validation.
  - If your data has a temporal relationship, it is best to structure these folds across time. For example, leaving one year out at a time and training on all other years.
- **Stratified K-Fold:** When dealing with imbalanced data, stratified k-fold ensures that each fold maintains the same distribution of classes as the original dataset.
- **Note**: Cross-validation can provide a more reliable estimate of model performance by averaging the results over multiple splits, reducing the risk of overfitting to a particular training-validation split.
  - In both instances, you either need to hold out two folds, one for each validation and testing, or retain a held-out test set. Models across each fit are then averaged to create your final optimal model.

Data Augemntations

Data Augmentation:

- In data-constrained environments, augmenting the dataset (e.g., through transformations like rotations, flipping, scaling in image tasks) can help create more diverse training examples.
- Augmentation helps prevent overfitting and improves the model's ability to generalize to new, unseen data.

Training Models & Tuning Hyperparameters



Hyperparameters

What Are Hyperparameters?

- **Hyperparameters** are configuration settings defined *before* training begins.
- Unlike model parameters (which are learned from the data), hyperparameters control how the model learns.
- They affect training speed, model complexity, and generalization ability.

Why Tuning Matters

- Proper tuning can dramatically improve performance on unseen data.
- Prevents underfitting and overfitting by finding the right tradeoffs.
- Typically done using a validation set or cross-validation.
- Common tuning approaches: **grid search**, **random search**, and **Bayesian optimization**.

What Hyperparameters Do We Tune?

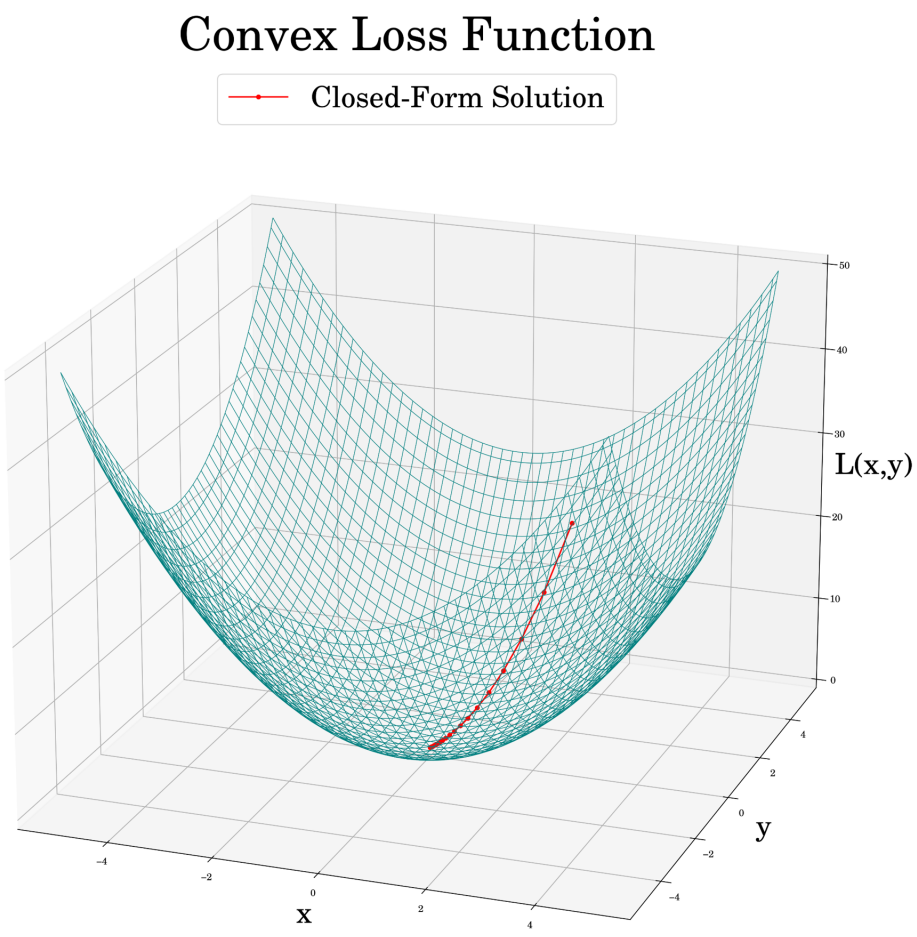
Common Hyperparameters (Deep Learning):

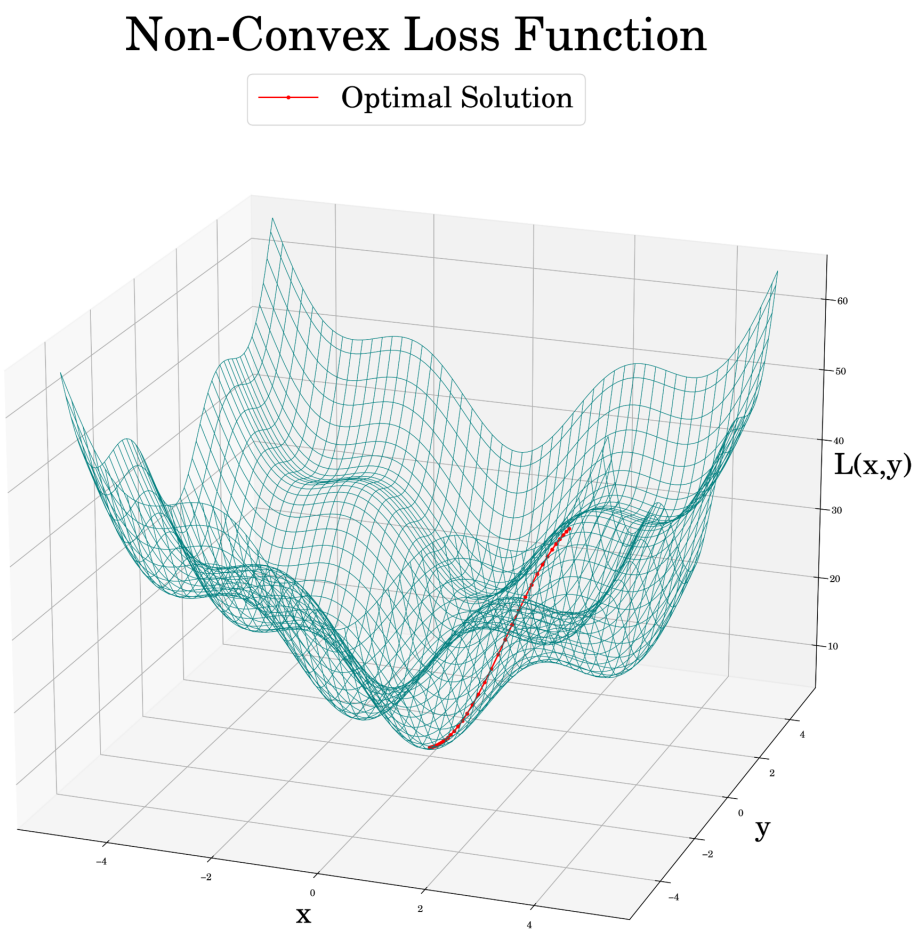
- **Learning rate:** Step size in optimization; too high causes instability, too low slows learning.
- **Regularization strength (e.g., L1, L2):** Controls model complexity and prevents overfitting.
- **Model complexity:** Number of layers, hidden units, activation functions.
- **Batch size / Epochs:** Influence training stability and convergence.

Hyperparameters are **model-dependent** — tuning for a neural net is different from a random forest.

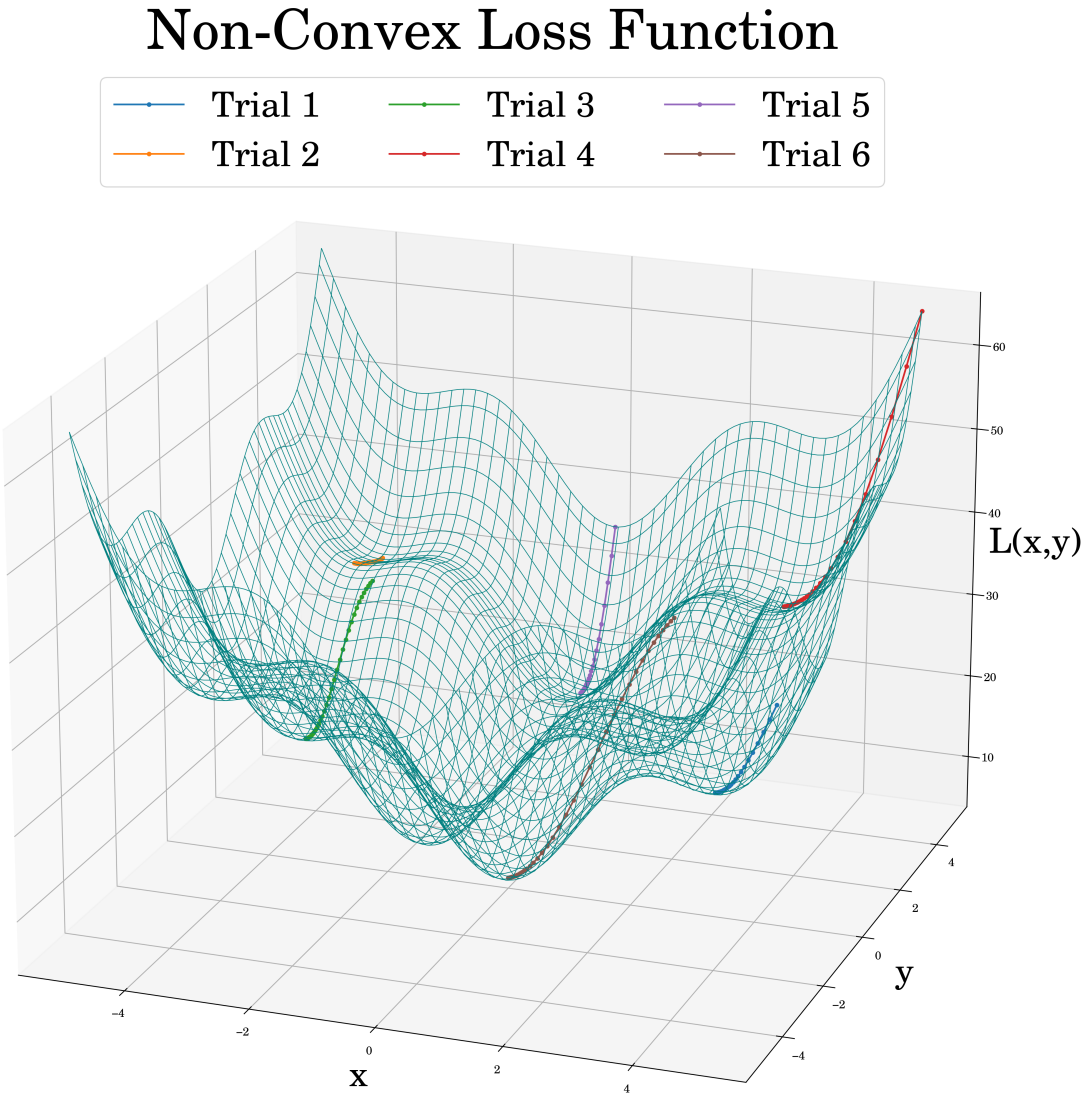
- There is no one-size-fits-all set of hyperparameters — experimentation is key.
  - **Decision Trees:** max depth, min samples split
  - **KNN:** number of neighbors, distance metric
  - **SVM:** kernel type, regularization parameter

Optimality: Closed-Form Solutions & Gradient Descent

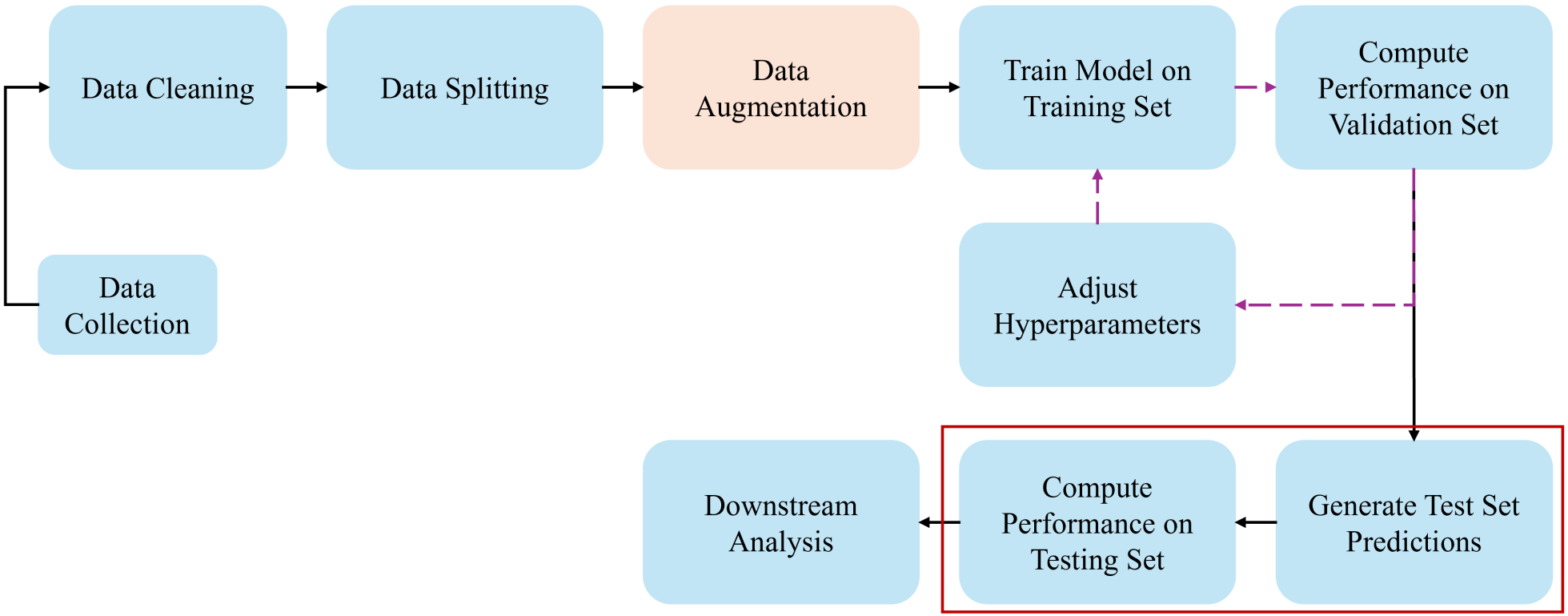




Reality: Hyperparameter Tuning by Trials



Performance Evaluation



Strategies for Evaluating Model Performance

Regression

- **MSE (Mean Squared Error)** – Penalizes larger errors more heavily; sensitive to outliers.
- **MAE (Mean Absolute Error)** – More robust to outliers; treats all errors equally.
- **RMSE (Root Mean Squared Error)** – Same units as target variable; emphasizes larger errors.
- **$\mathbf{R^2}$** – **Opinion: Almost never use R^2 in ML contexts.**

Classification

- **AUROC** – Measures ability to distinguish classes across all thresholds.
- **Accuracy** – Proportion of correct predictions; can be misleading.
- **Precision** – Proportion of predicted positives that are true positives.
- **Recall (Sensitivity)** – Proportion of actual positives correctly identified.
- **F1 Score** – Harmonic mean of precision and recall; balances false positives and false negatives.
- **Confusion Matrix** – Breaks predictions into TP, FP, TN, FN for deeper insight.

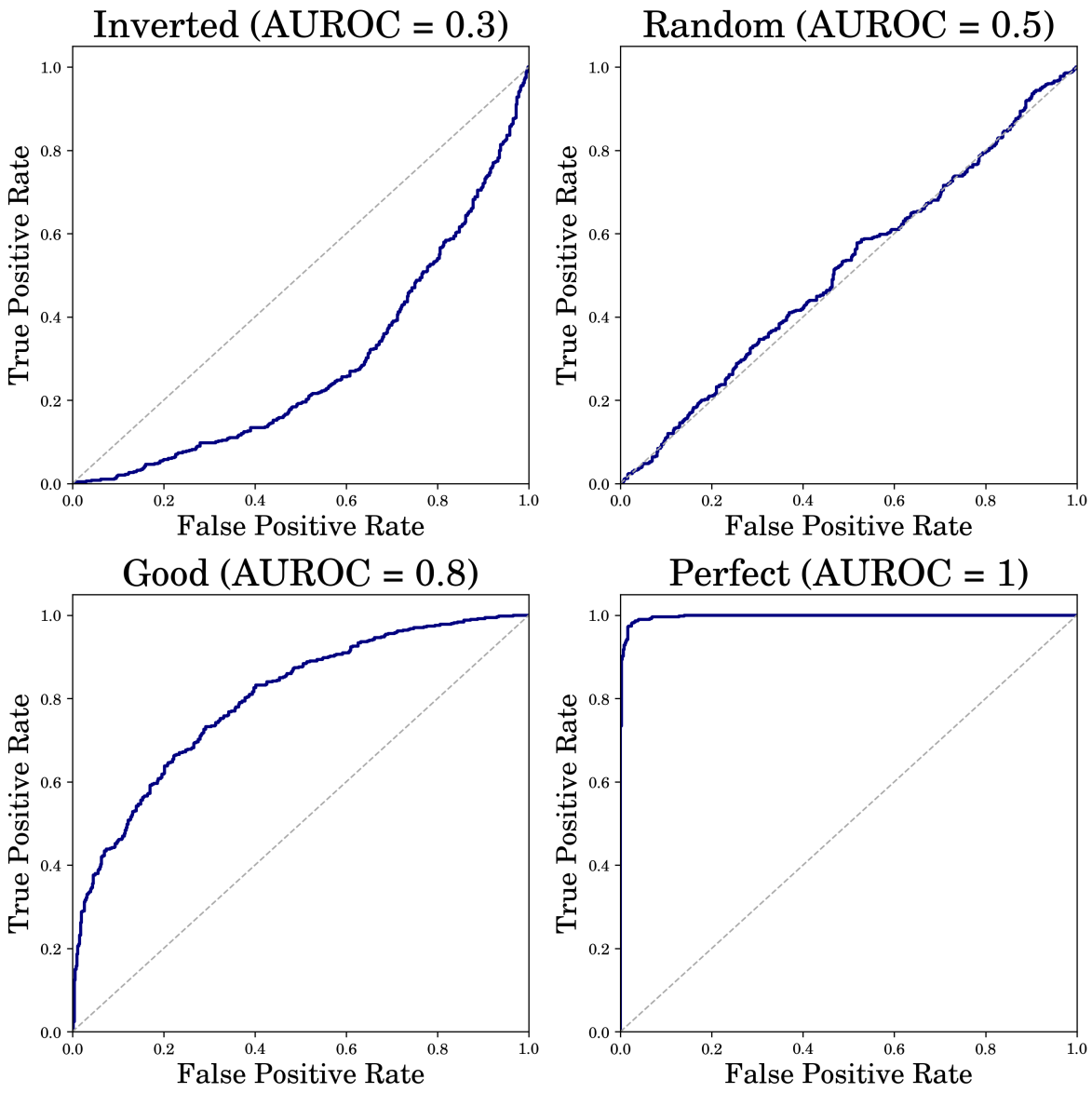
Understanding AUROC for Model Evaluation

AUROC (Area Under the Receiver Operating Characteristic Curve):

- Measures a classifier's ability to distinguish between positive and negative classes by plotting the True Positive Rate (TPR) vs. False Positive Rate (FPR) across all decision thresholds.
- **Interpretation:**
  - 1.0 = Perfect classification.
  - 0.5 = Random guessing.
  - < 0.5 = Worse than random (model may be inverted).
- **Why use AUROC?**
  - Especially useful for imbalanced datasets.
  - Captures overall ranking quality, unlike loss functions that depend on chosen threshold.
  - Threshold-independent — evaluates model performance across all possible cutoffs.

Bottom line: AUROC is a powerful tool for comparing classification models when class imbalance or threshold sensitivity is a concern.

AUROC Curves



Key Tools for Developing Models: Python

- **scikit-learn** – General-purpose ML toolkit for classification, regression, clustering, and pipelines.
  - Contains built-in support for hyperparameter tuning.
- **TensorFlow** – End-to-end deep learning framework with support for deployment and scalability.
- **Keras** – High-level API running on top of TensorFlow for fast prototyping.
- **PyTorch** – Dynamic, flexible DL library widely adopted in both research and industry.
- **PyTorch Lightning** – Abstraction layer over PyTorch for cleaner, more reproducible training loops.
- **Optuna** – **Powerful hyperparameter optimization framework**; integrates seamlessly with scikit-learn, PyTorch, and more.

Key Tools for Developing Models: R

- **caret** – Classic ML framework for training and tuning models with a unified API.
- **mlr3** – Modern, modular ML toolbox for benchmarking and automating workflows.
- **tidymodels** – Tidyverse-aligned ecosystem for modeling, including:
  - `parsnip` (model interface),
  - `recipes` (preprocessing),
  - `tune` (hyperparameter tuning),
  - `workflows`, etc.
- **keras** – R interface to Keras and TensorFlow for deep learning models.
- **torch** – Native R bindings for PyTorch for building and training neural networks.

Formalizing & Scrutinizing ML Methods

1. Introduce the problem.
2. Introduce the dataset.
3. Introduce the model.
4. Specify features and pre-processing.
5. Specify how the data is used to evaluate the model.
6. Specify the hyperparameter search space.
7. Explain how hyperparameters were optimized.
8. Evaluate the model on a clean test set.
9. Comment on the risk of Distribution Shift.

Causal Machine Learning

- Traditionally, machine learning and causal inference have been seen as distinct, with ML focused on predictive accuracy and causal inference focused on identifying cause-and-effect relationships.
- However, recent advances show that ML can **support causal identification strategies**, especially in complex or high-dimensional settings.
- Use **Causal ML** when:
  - You care about **treatment effects**, not just predictions.
  - You have **high-dimensional covariates** or complex interactions.
  - Treatment effects may be **heterogeneous**.

Common Causal ML Methods

- **Causal Forests:** Estimate heterogeneous treatment effects using tree-based ensembles.
- **Doubly Robust Learner:** Combines outcome and treatment models to provide unbiased estimates if either model is correctly specified.
  - **Doubly Robust IV:** Extends doubly robust methods to instrumental variable settings.
  - **Double Machine Learning (DML):** Uses ML models to flexibly estimate nuisance functions and debias causal parameter estimates.
- **Uplift Modeling:** Directly models the difference in outcomes between treated and control groups.
- **Meta-Learners (e.g., S-, T-, X-, and R-learners):** Algorithms that combine ML predictions to estimate conditional average treatment effects.

Note: Many methods qualify as causal ML when machine learning aids causal identification, such as using high-dimensional ML-based propensity scores or tree-based methods for segmenting heterogeneous effects.

Causal Forests: Estimating Heterogeneous Effects

What is a Random Forest?

- An ensemble of decision trees trained on bootstrapped samples.
- Each tree makes a prediction, and the forest averages them (for regression) or takes a majority vote (for classification).
- Randomness in feature selection and data sampling helps reduce variance and avoid overfitting.

Causal forests are a **nonparametric method** for estimating heterogeneous treatment effects. They help avoid overfitting or cherry-picking by automating subgroup discovery.

Key Assumptions:

- **Honesty:** Trees are built using separate data for splitting and estimation.
- **Unconfoundedness:** Treatment assignment is independent of potential outcomes given covariates.

Why Use Causal Forests?

- Flexible modeling of treatment effect heterogeneity
- Avoids manual subgroup analysis & maintains strong theoretical guarantees:
  - Consistency
  - Asymptotic normality
  - Sometimes better prediction

Tools: `grf` (R), `CausalForest` (Python)

Causal Forests: Measuring Treatment Effects

For a collection of independent samples n indexed by i with vectors of features X_i, treatment indicators W_i, and response Y_i:

$$\tau(x) = \mathbb{E}\left[Y_i^1 - Y_i^0 | X_i = x\right]$$

In the context of causal trees, we want to partition the data into a set of leaves L such that the $L(x)$ is small enough that all (Y_i, W_i) pairs within the leaf act as though they had come from a random experiment (identical among all other dimensions).

$$\hat{\tau}_{\text{leaf}} = \frac{1}{n_{treated}} \sum_{i \in \text{treated}} Y_i - \frac{1}{n_{control}} \sum_{i \in \text{control}} Y_i$$

Causal forests will generate an ensemble of T such trees that the average effect is recovered as:

$$\hat{\tau} = \frac{1}{T} \sum_{t=1}^{T} \hat{\tau}_t(x)$$

How Causal Forests Work: Honesty and Tree Construction

Double-sample trees and "honesty":

- Each tree randomly selects a subsample of the data and splits the sample into two halves, A and B:
  - A: Used to determine the tree structure (splitting rules)
  - B: Used to estimate treatment effects in the leaves
- This separation of roles is key to avoiding overfitting and achieving valid inference.
- The honest sampling procedure is repeated independently in each tree of the forest which ensures full coverage of the data.

Split Criterion:

- Trees are built to maximize the heterogeneity of treatment effects:

$$\max \mathrm{Var}(\hat{\tau}(X_i)) \text{ for } i \in A$$

- An alternative criterion is to predict treatment assignment W_i using classification trees (e.g., for propensity scores).

The Challenge: High-Dimensional Causal Inference

We consider settings with a high-dimensional covariate space X, where traditional estimation methods break down. Using machine learning (ML) to model components of the system can help manage this complexity—but naive ML approaches can introduce bias due to regularization and overfitting.

$$Y = W\theta_0 + g_0(X) + \varepsilon$$

$$W = m_0(X) + \nu$$

Goal: Estimate the causal parameter θ_0, despite X being high-dimensional and potentially confounded with both W and Y.

Challenges:

- High-dimensional covariates X
- Potential omitted variable bias (OVB) & Simultaneity / endogeneity in W
- Regularization bias from ML estimators

Doubly Robust Double Machine Learning (DML)

Key idea: Combine Neyman-orthogonal moments with cross-fitting to obtain root-n consistency and valid confidence intervals, even when nuisance functions are estimated using ML.

$$Y = W\theta_0 + g_0(X) + \varepsilon$$

$$W = m_0(X) + \nu$$

Solution: Double Machine Learning

- Estimate nuisance functions $g_0(X)$ and $m_0(X)$ using ML
- Use **Neyman-orthogonal moments** to remove regularization bias
- Use **cross-fitting** to mitigate overfitting bias

What does "Doubly Robust" mean?

- The estimator remains consistent if <u>either</u> $g_0(X)$ or $m_0(X)$ is estimated correctly.

DML: A Quick Overview

The Frisch-Waugh-Lovell theorem can be generalized to an ML framework such that orthogonalization overcomes the regularization bias.

$$Y = W\theta_0 + g_0(X) + \varepsilon$$

$$W = m_0(X) + \nu$$

Steps:

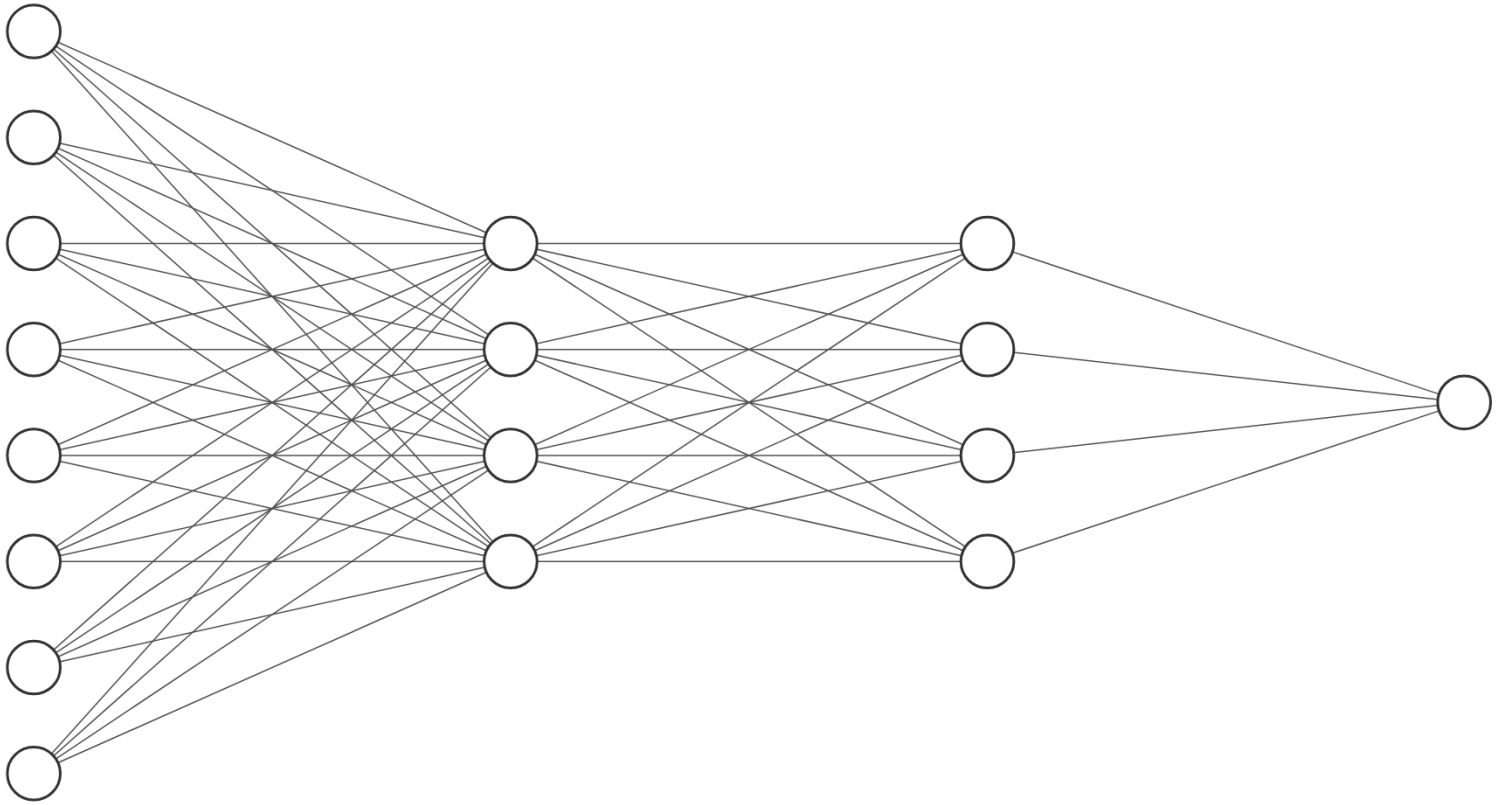
1. Cross-fitting: split your data randomly into two groups (50-50)
   - This naturally overcomes concerns of overfitting.
2. Estimate nuisance functions using ML on each split s.
   - $W^s = m_0^s(X^s) + \nu^s$
     - $\tilde{W}^s = W^s - \hat{W}^s$
   - $Y^s = g_0^s(X^s) + \varepsilon^s$
     - $\tilde{Y}^s = Y^s - \hat{Y}^s$
3. Regress the residuals from step (2) using the split not used for ML model development.
   - $\tilde{Y}^s = \theta_0^s \tilde{W}^s + \psi^s$
4. Average θ_0^1 and θ_0^2 to recover θ_0

Challenges in Causal ML

- **Confounding:** ML doesn't magically solve omitted variable bias — causal assumptions still matter.
- **Interpretability:** Complex models can make it harder to interpret treatment effects.
- **Overfitting:** Estimating treatment effects requires more caution than predictive modeling.
- **Assumptions still rule:** Unconfoundedness and consistency must be satisfied.

Reminder: ML is a tool — not a substitute for careful causal reasoning.

Artificial Neural Networks & Multi-Layer Perceptrons



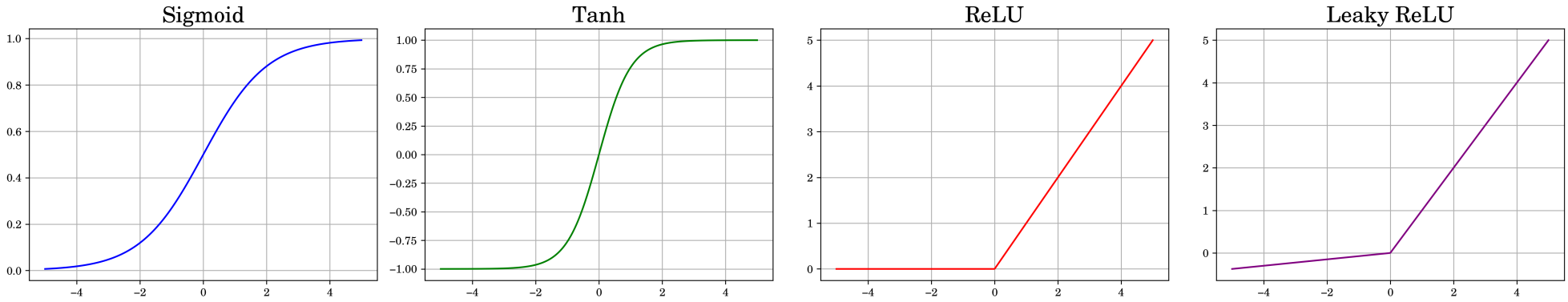
Input Layer $\in \mathbb{R}^8$ Hidden Layer $\in \mathbb{R}^4$ Hidden Layer $\in \mathbb{R}^4$ Output Layer $\in \mathbb{R}^1$

- *A Logical Calculus of the Ideas Immanent in Nervous Activity* (Warren S. McCulloch, Walter Pitts (1943))
- *Multilayer Feedforward Networks are Universal Approximators* (Kurt Hornik, Maxwell Stinchcombe, Halbert White (1989))

Activation Functions

Activation functions are key components in neural networks that introduce nonlinearity, enabling the network to learn complex patterns and model intricate relationships in data. Without this source of nonlinearity, a neural network would behave like a simple linear model, regardless of its depth.

- Sigmoid: $\frac{1}{1+e^{-x}}$
- tanh: $\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- Relu: $\max(0, x)$
- Leaky ReLU: $\max(\alpha x, x)$, where $\alpha \in \mathbb{R},\ \alpha > 0$

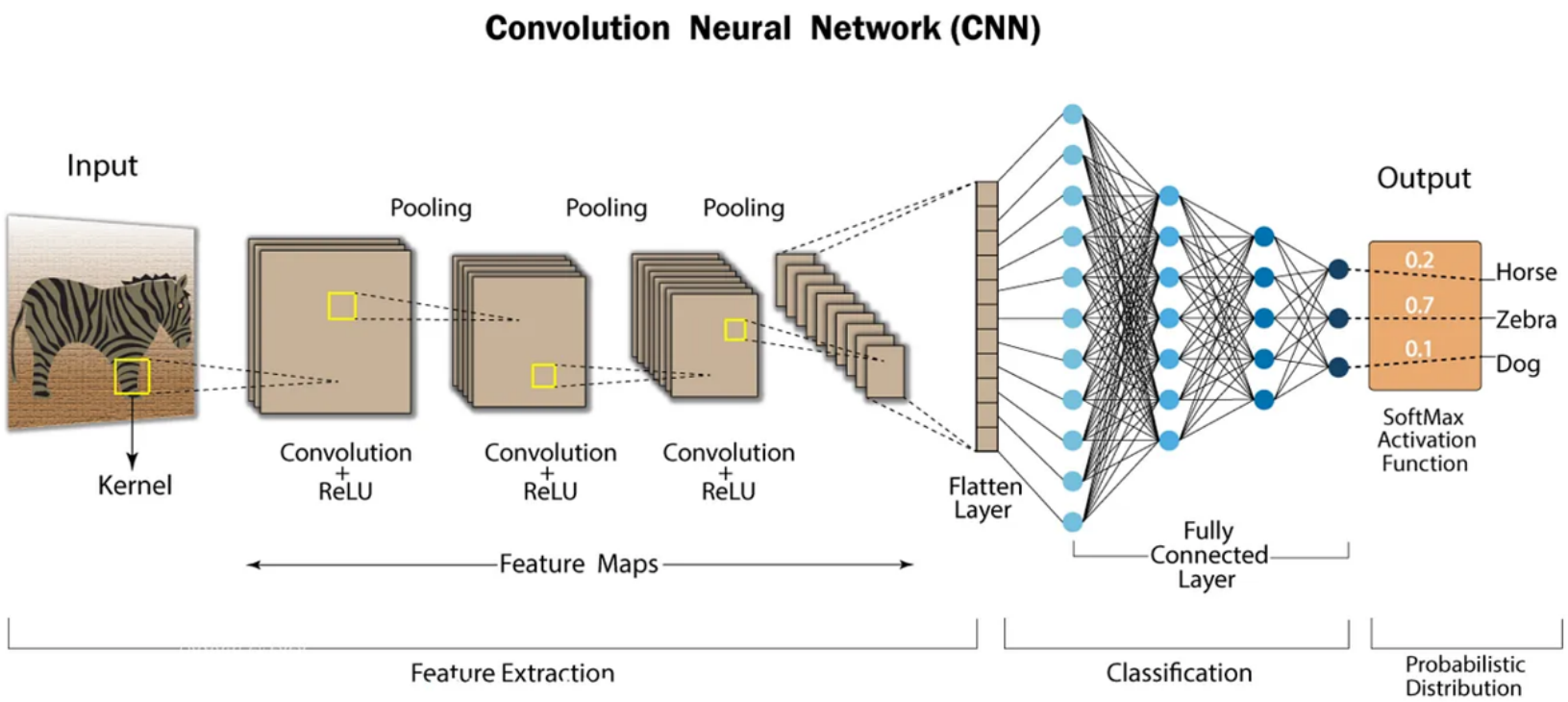


Moving Past MLPs

The architecture itself—how we design and connect layers—plays a critical role in shaping the functional form of the model.

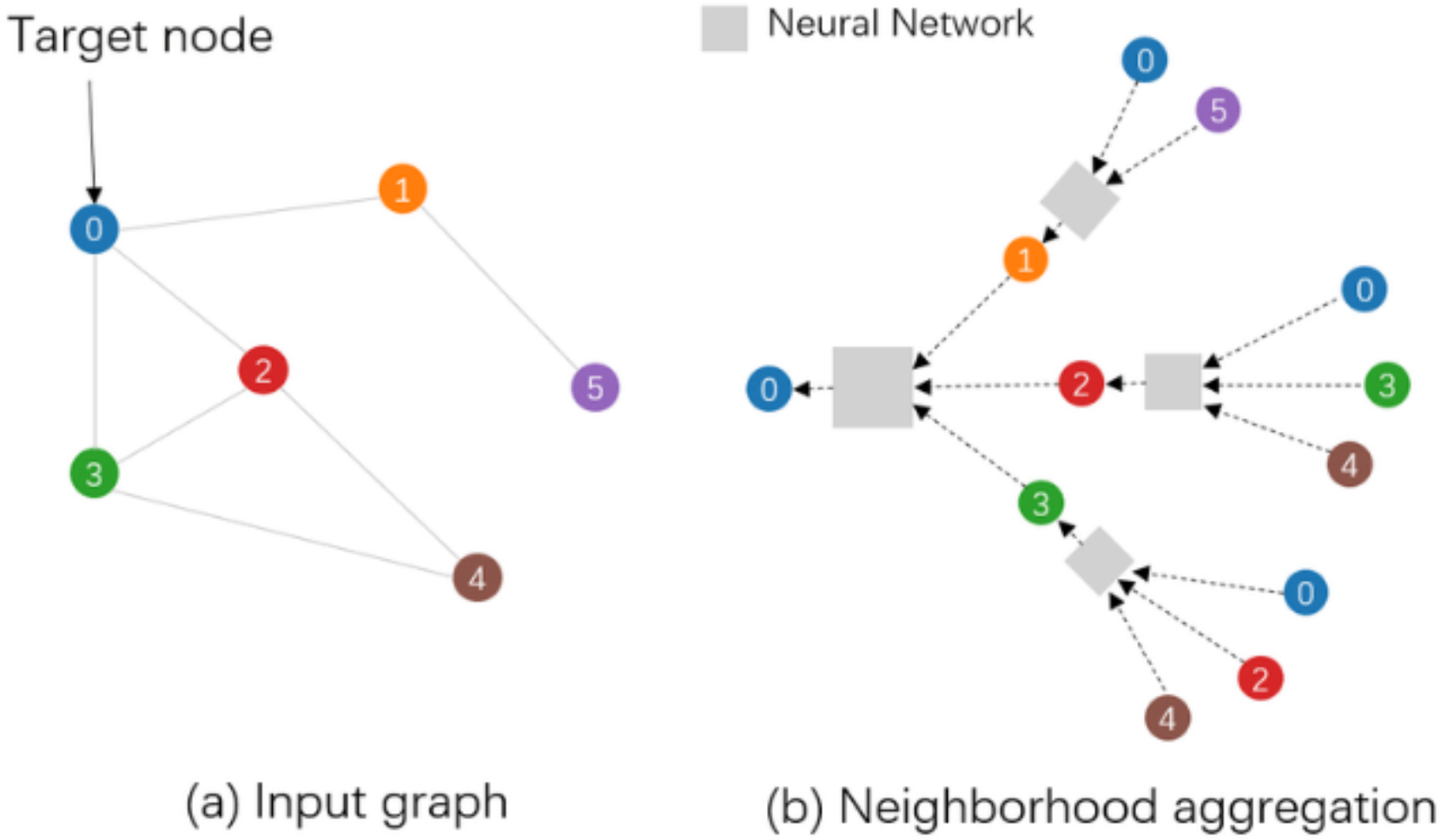
- **Flexible Architectures:**
  - $f(x)$: Basic feedforward MLPs with direct input-output mapping—no internal structure or context.
  - $f(h(x) + g(x))$: Modular networks with parallel transformations. Useful for combining different feature paths or representations.
  - $f(x, h(x, y))$: Networks that condition on auxiliary or contextual data—common in multimodal models or architectures with skip connections.
  - $f(x_t, g(x_{t-1}, \ldots, x_0))$: Recurrent or sequence-aware models, where outputs depend on current input and accumulated past. Captures temporal dependencies and internal memory.
  - $f(x_t, \text{context}_t)$: Attention-based models that dynamically focus on relevant input components—context-aware and often non-sequential.
  - $f(x, y) \to g(x, y)$: Convolutional architectures impose spatial locality and translational invariance, ideal for image and spatial data.
- **Functional Restrictions:** Network design dictates the functional form $f(x)$ the model can estimate. For example, a recurrent network restricts f to functions with temporal dependencies, while a CNN imposes locality through convolution, shaping the transformation the model performs.

Convolutional Neural Networks



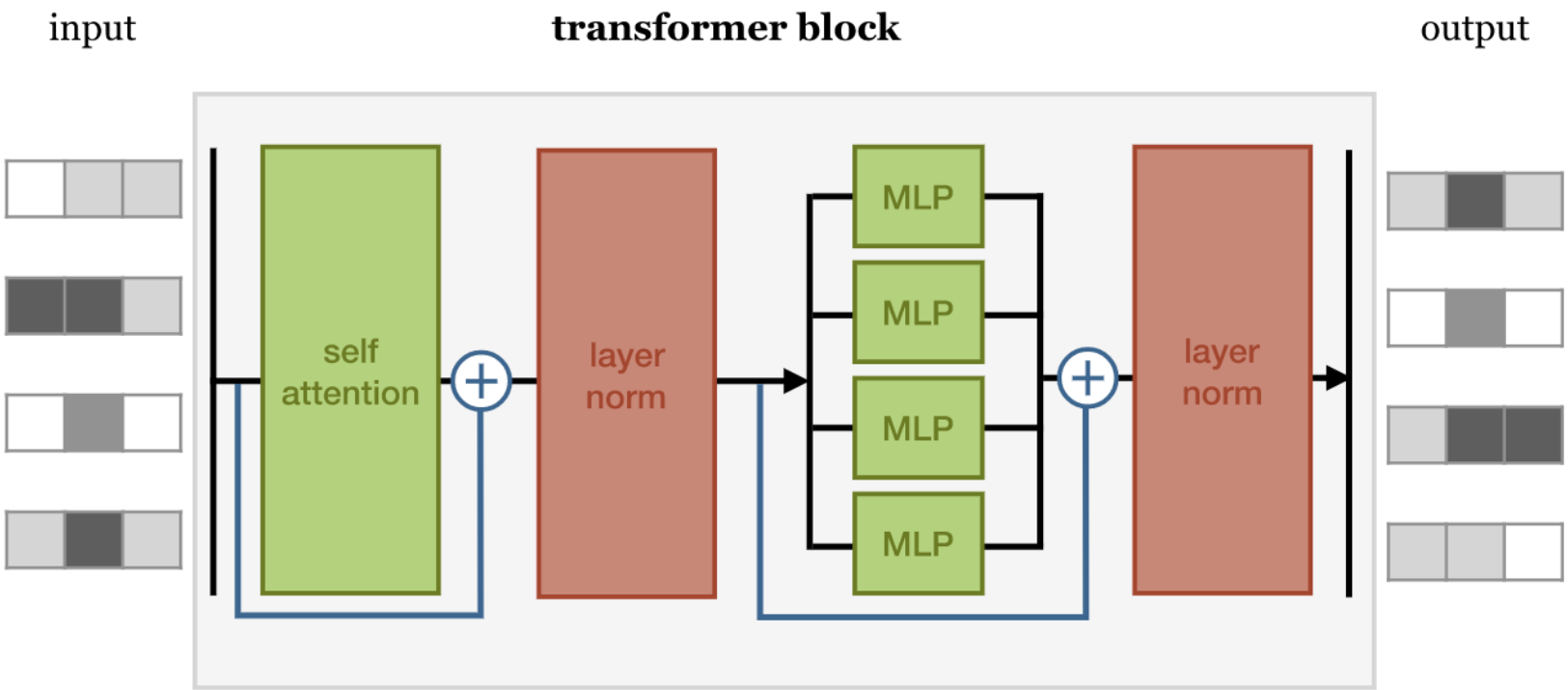
- *Using Neural Networks to Predict Microspatial Economic Growth* (Khachiyan et al. (2022))
- *A Regional Industrial Economic Forecasting Model Based on a Deep Convolutional Neural Network and Big Data* (Tuo et al. (2021))
- *Efficient OCR for Building a Diverse Digital History* (Carlson et al. (2023))

Graph Neural Networks



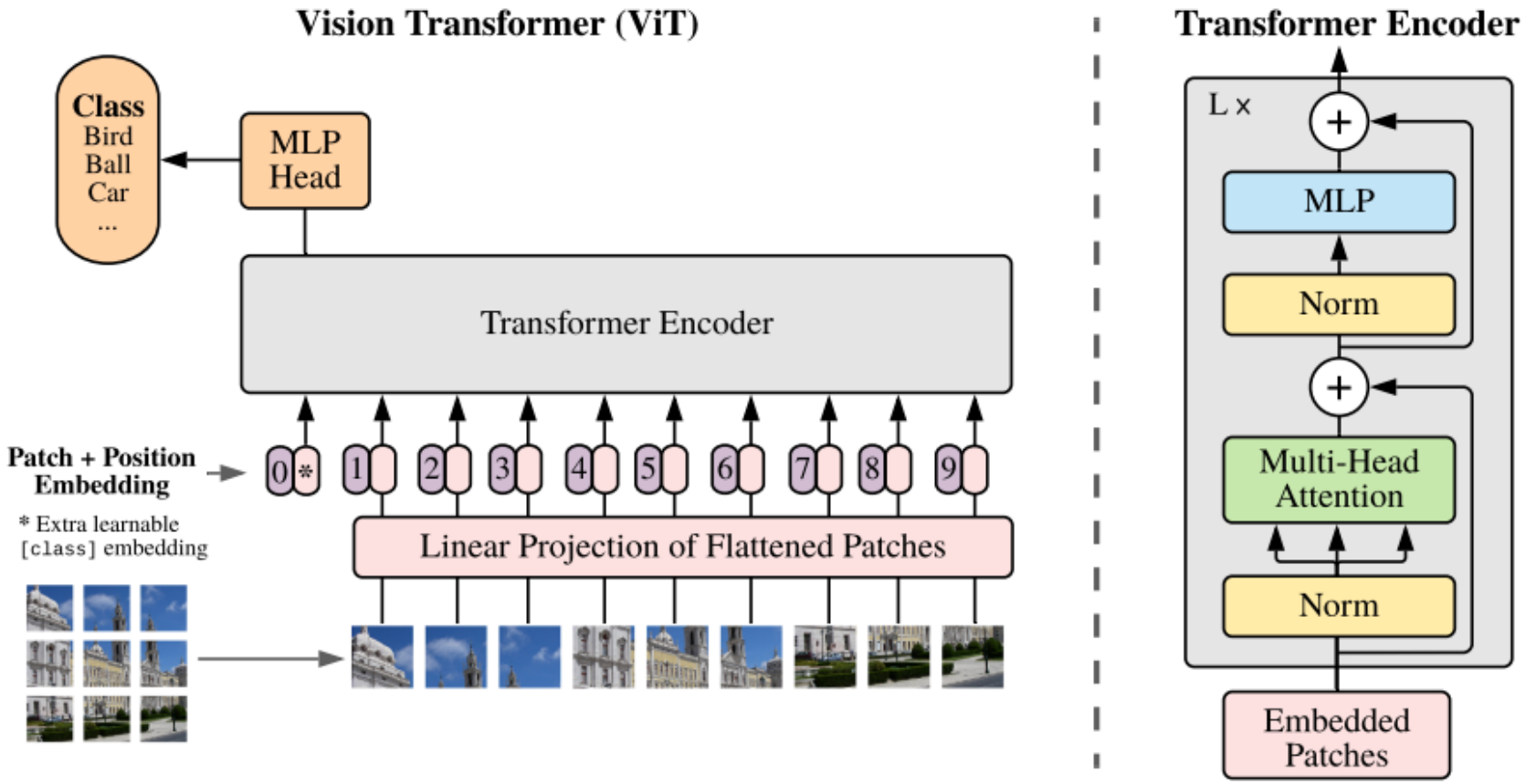
- *The Graph Neural Network Model* (Scarselli et al. (2008))
- *Semi-Supervised Classification with Graph Convolutional Networks* (Kipf et al. (2016))

Transformers



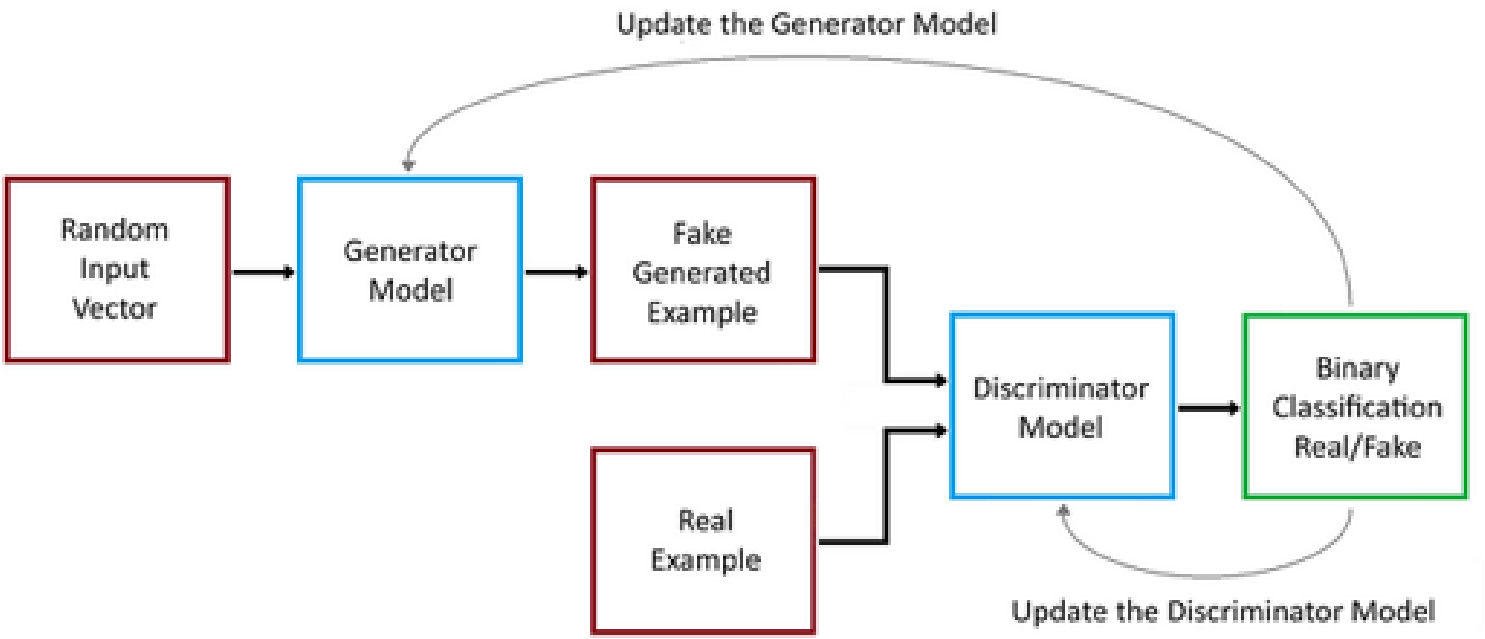
- Recurrent Neural Networks and Long-Short Term Memory were the key predecessors to transformers. Today, you need a really good reason to use either of these over a transformer.
- *Attention Is All You Need* (Vaswani et al. (2017))
- *CAREER: A Foundation Model for Labor Sequence Data* (Vafa et al. (2022))
- *Transformer Choice Net: A Transformer Neural Network for Choice Prediction* (Wang et al. (2023))

Vision Transformers



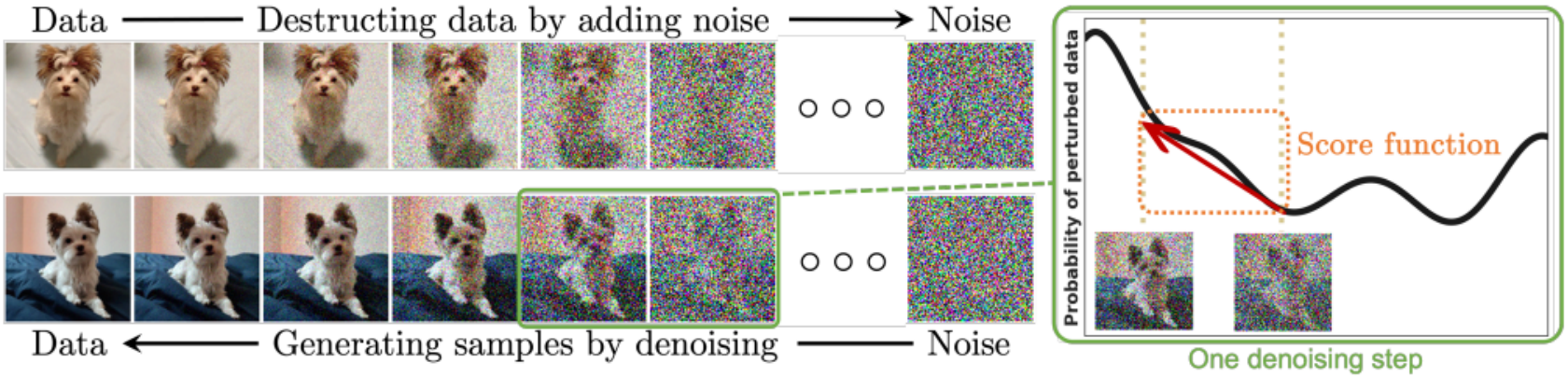
- *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale* (Dosovitskiy et al. (2021))
- *Pixel Perfect: Using Vision Transformers to Improve Road Quality Predictions from Medium Resolution and Heterogeneous Satellite Imagery* (Muhebwa et al. (2023))

Generative Adversarial Networks



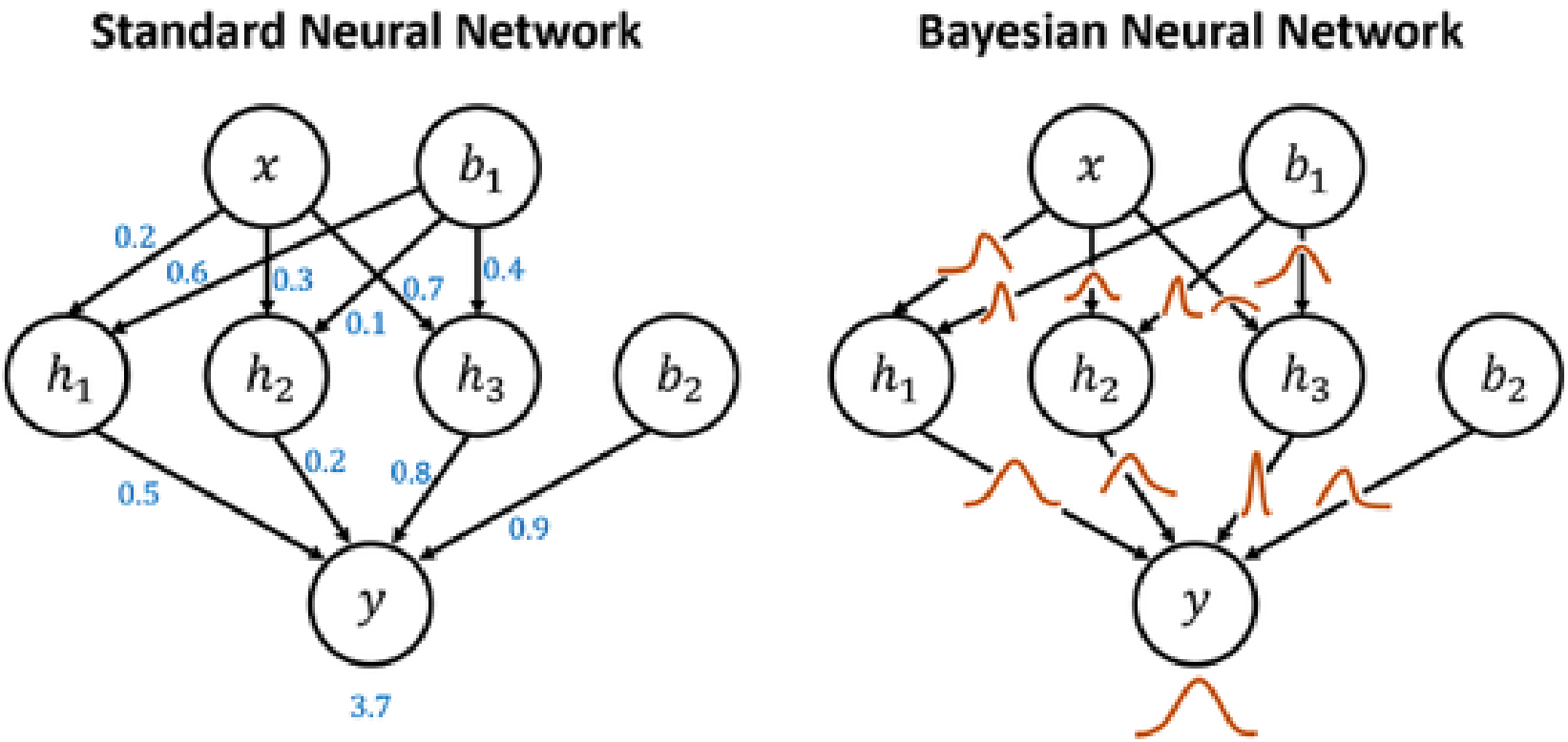
- *Generative Adversarial Networks* (Goodfellow et al. (2014))
- *Using Wasserstein Generative Adversarial Networks for the Design of Monte Carlo Simulations* (Athey et al. (2024))
- *Semi-Supervised Multitask Learning on Multispectral Satellite Images Using Wasserstein Generative Adversarial Networks (GANs) for Predicting Poverty* (Perez et al. (2019))

Diffusion Models



- *Denoising Diffusion Probabilistic Models* (Ho et al. (2020))
- *Score-Based Generative Modeling through Stochastic Differential Equations* (Song et al. (2021))
- *Predict, Refine, Synthesize: Self-Guiding Diffusion Models for Probabilistic Time Series Forecasting* (Kollovieh et al. (2023))

Bayesian Neural Networks

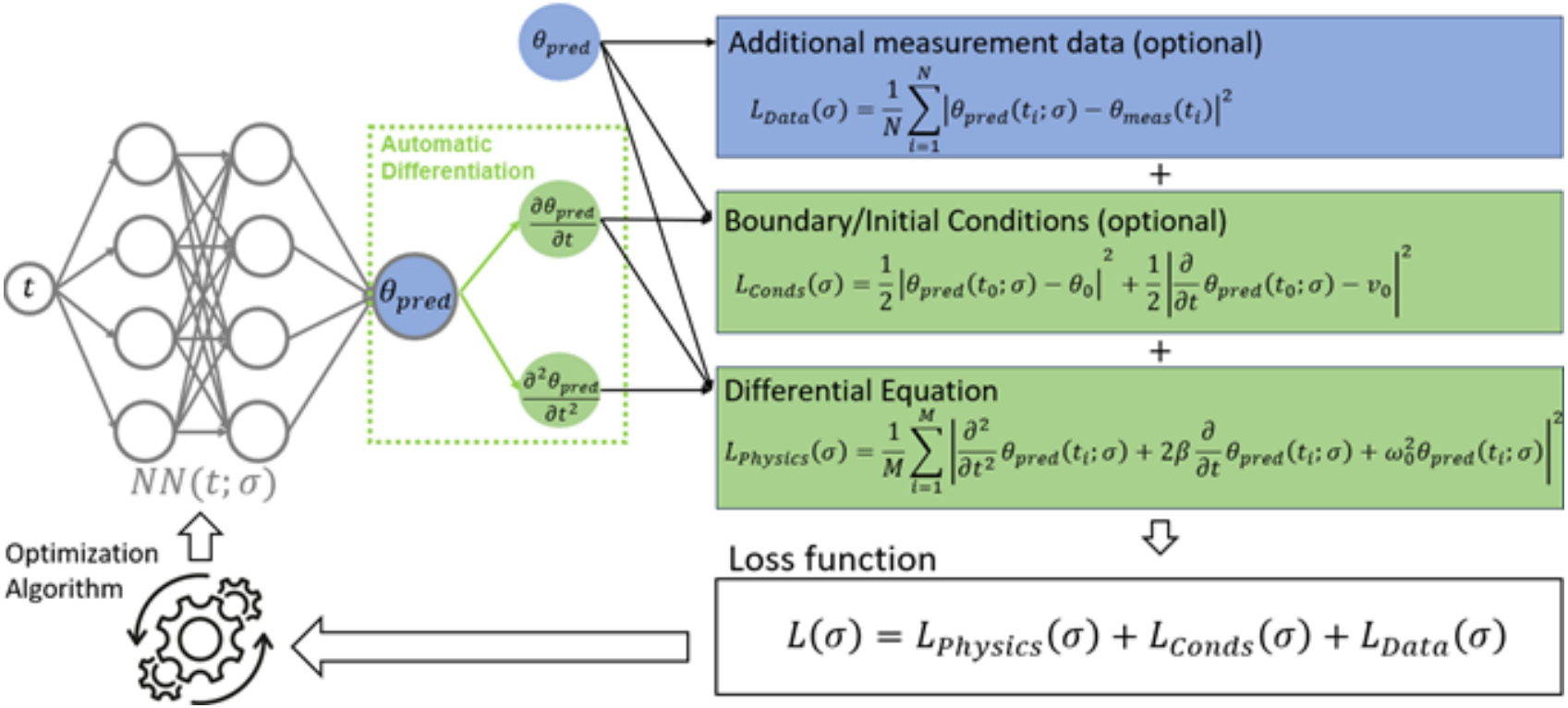


- *Bayesian Neural Networks: An Introduction and Survey* (Goan et al. (2020))
- *Bayesian Neural Networks for Macroeconomic Analysis* (Hauzenberger et al. (2024))

Informed Neural Networks

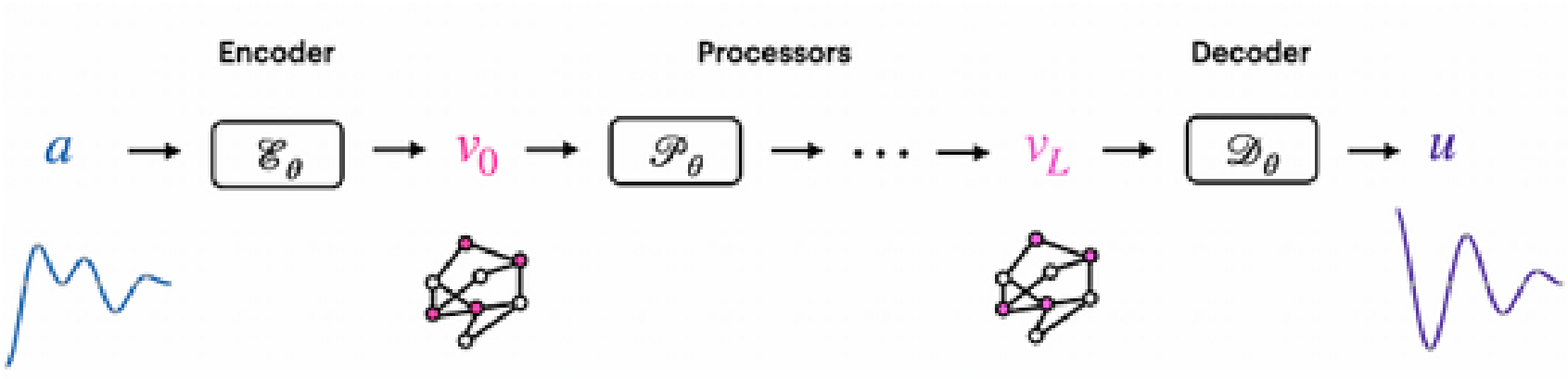
$$\theta''(t) + 2\beta\theta'(t) + \omega_0^2\theta(t) = 0 \quad \text{Pendulum equation}$$

$$\theta(t_0) = \theta_0, \quad \theta'(t_0) = v_0 \quad \text{Initial conditions}$$



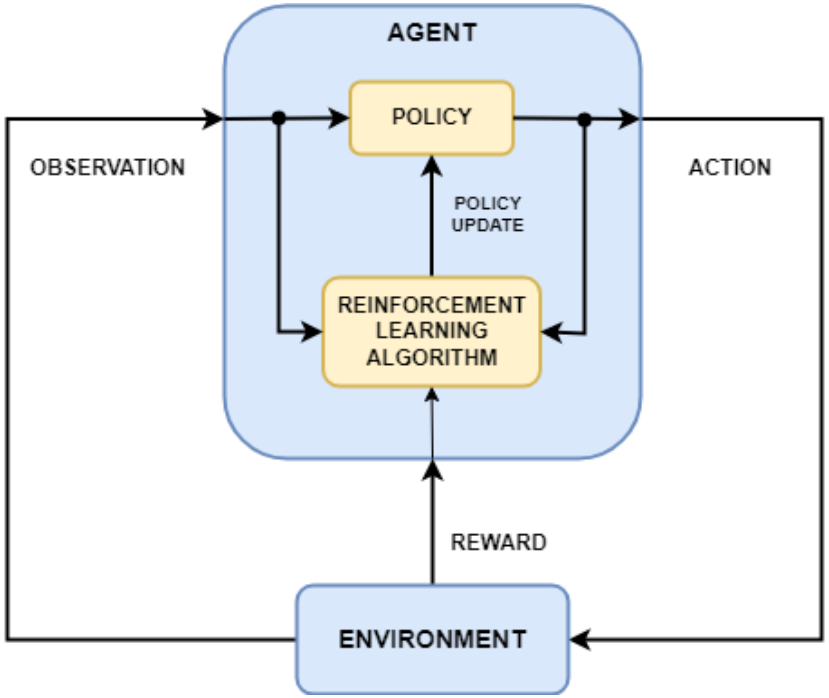
- *Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations* (Raissi et al. (2019))
- *Deep-MacroFin: Informed Equilibrium Neural Network for Continuous Time Economic Models* (Wu et al. (2024))

Neural Operators



- *Neural Operator: Learning Maps Between Function Spaces* (Kovachki et al. (2023))
- *Diffeomorphism Neural Operator for Various Domains and Parameters of Partial Differential Equations* (Zhao et al. (2025))

Reinforcement Learning



- *Reinforcement Learning in Economics and Finance* (Charpentier et al. (2022))
- *Simulating the Economic Impact of Rationality through Reinforcement Learning and Agent-Based Modelling* (Brusatin et al. (2024))
- *Reinforcement Learning in a Prisoner's Dilemma* (Dolgopolov (2024))
- *Game Theory and Multi-Agent Reinforcement Learning: A Mathematical Overview* (Alonso and Njupoun (2024))
- *Contextual Bandits in a Survey Experiment on Charitable Giving: Within-Experiment Outcomes versus Policy Learning* (Athey et al. (2022))

Conclusion

Be skeptical. Be curious. Be careful.

- AI and machine learning can be incredibly powerful—but only when used with care and clarity. Without transparency in development and careful methodology, these tools can easily lead us astray.

Resources:

- Probabilistic Machine Learning – A comprehensive guide to the theory behind ML
- Papers with Code – Explore state-of-the-art ML models and their implementations
- Hugging Face – Access pre-trained models, data, and AI tools

Tips:

- Always treat your data with care. Garbage in, garbage out—the validity of your ML methods depends entirely on clean data and rigorous train-test splits.
- Causal ML is a powerful framework, but only use it when appropriate and avoid overfitting causal narratives to observational data.
- Never implement a method you don't understand. Strong foundations lead to strong results.