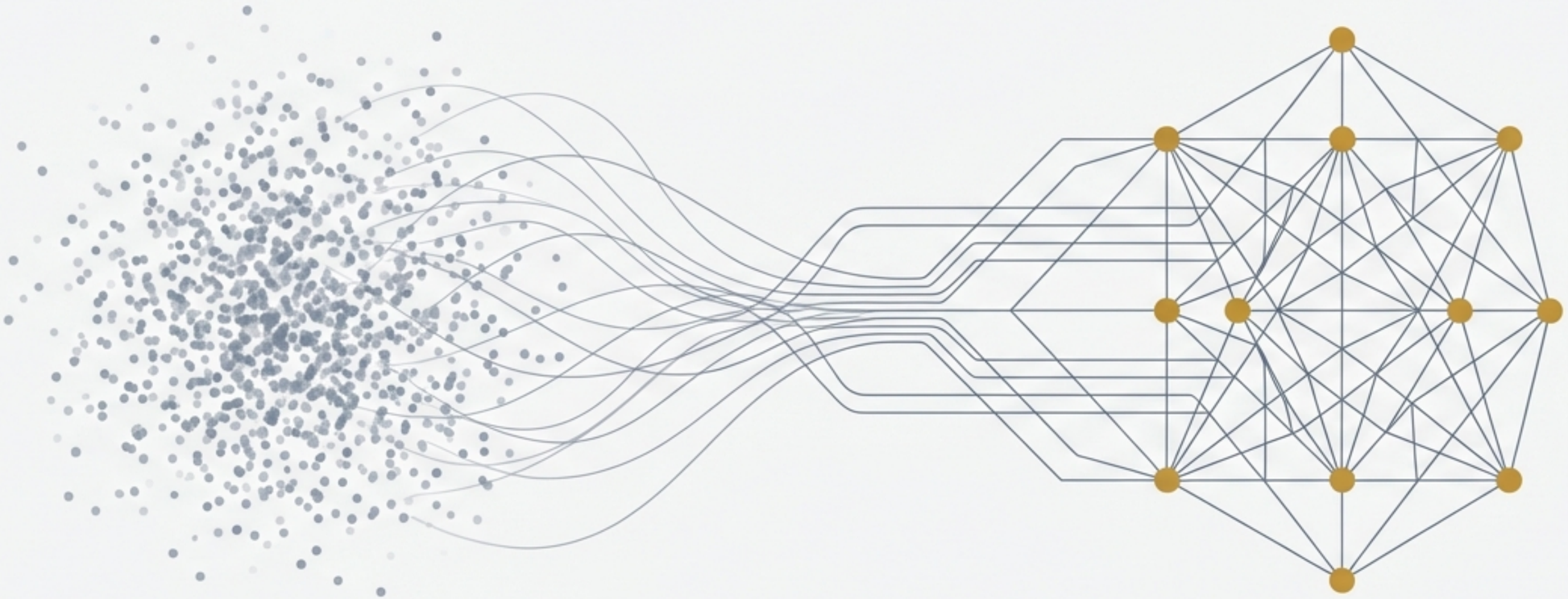
A Mathematical Toolkit for Machine Learning

From Linear Foundations to Neural Architectures



Visualizing the transformation from unstructured data to complex, structured knowledge through mathematical principles.

The Toolkit Agenda

1. **Module 1: Linear Regression**
   The Mathematics of Prediction
2. **Module 2: Logistic Regression**
   The Mathematics of Classification
3. **Module 3: K-Means Clustering**
   The Mathematics of Grouping
4. **Module 4: Multilayer Perceptron (MLP)**
   The Mathematics of Non-Linearity
5. **Module 5: Convolutional Neural Networks (CNN)**
   The Mathematics of Perception

Linear Regression: Modeling a Continuous Relationship

Core Concept

The fundamental goal is to model a linear relationship between input features **x** and a continuous target variable *y*.

Mathematical Formulation (Hypothesis)

The model's prediction, $\hat{\mathbf{y}}$, is a linear combination of input features, represented in vectorized form:

$$\hat{\mathbf{y}} = h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

where:

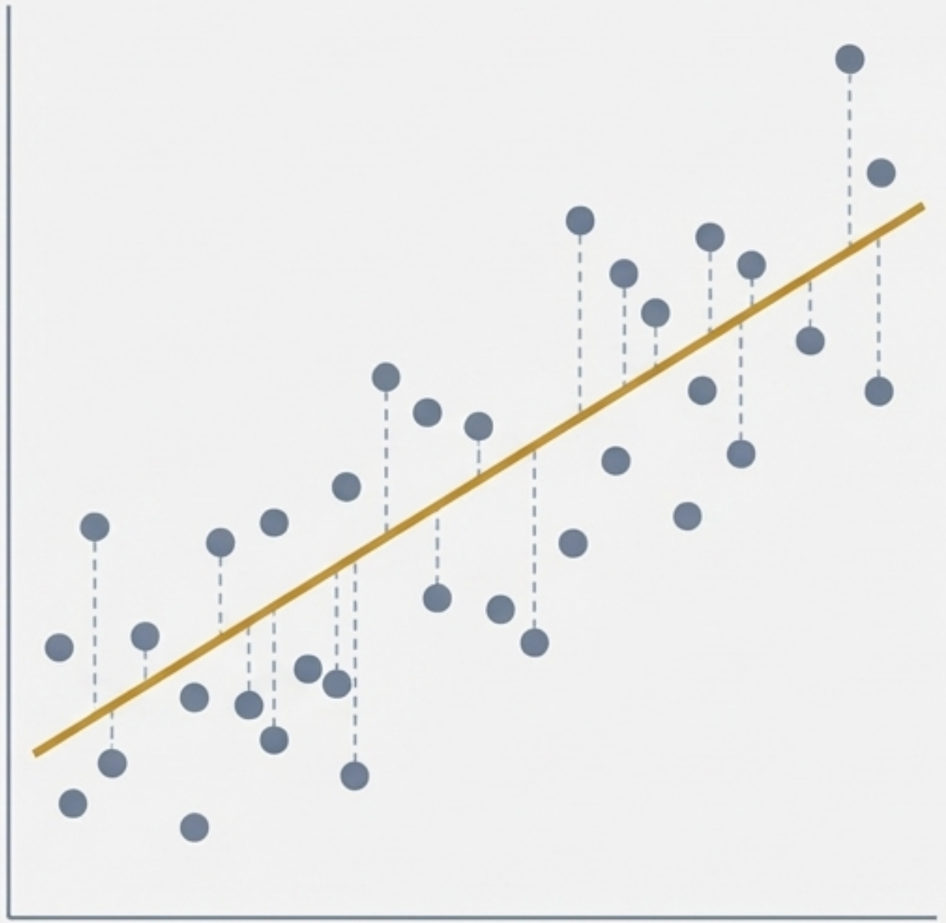
- **w** is the D-dimensional vector of model weights (parameters).
- **x** is the D-dimensional feature vector.

The Objective Function: Mean Squared Error (MSE)

To find the optimal weights **w**, we must minimize the difference between our predictions $\hat{\mathbf{y}}$ and the true values y. The objective is to minimize the Mean Squared Error (MSE) loss function, L(**w**):

$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(y_n - \mathbf{w}^T\mathbf{x}_n)^2$$

where N is the number of training examples.



Optimization: Finding the Optimal Weights

Method 1: Iterative Optimization via Gradient Descent

We iteratively adjust the weights **w** in the direction opposite to the gradient of the loss function.

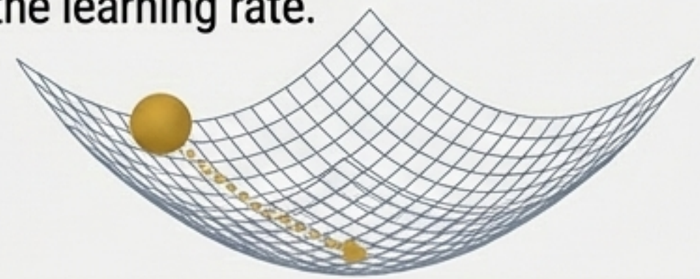
$$\nabla_w L(\mathbf{w}) = -\frac{2}{N}\sum_{n=1}^{N}(y_n - \mathbf{w}^T\mathbf{x}_n)\cdot\mathbf{x}_n$$

Gradient Descent Update Rule:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta\cdot\nabla_w L(\mathbf{w}(t))$$

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta\frac{2}{N}\sum_{n=1}^{N}(y_n - \mathbf{w}(t)^T\mathbf{x}_n)\cdot\mathbf{x}_n$$

where η is the learning rate.

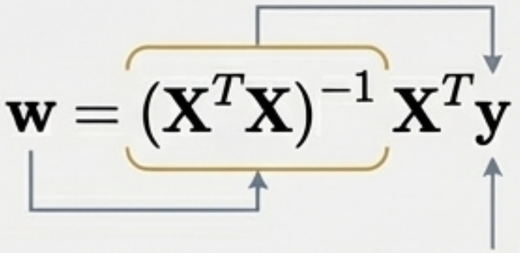


Method 2: Closed-Form Analytical Solution (The Normal Equation)

For smaller datasets where matrix inversion is feasible, the optimal **w** can be solved for directly without iteration. By setting the gradient to zero ($\nabla_w L(\mathbf{w}) = 0$), we derive the Normal Equation:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

where **X** is the N × D design matrix and **y** is the vector of true values.



Trade-off: Gradient Descent is iterative but avoids the expensive $(X^TX)^{-1}$ matrix inversion, making it suitable for large datasets and high-dimensional features. The Normal Equation is direct but computationally prohibitive for large D.

Evaluation and Model Summary

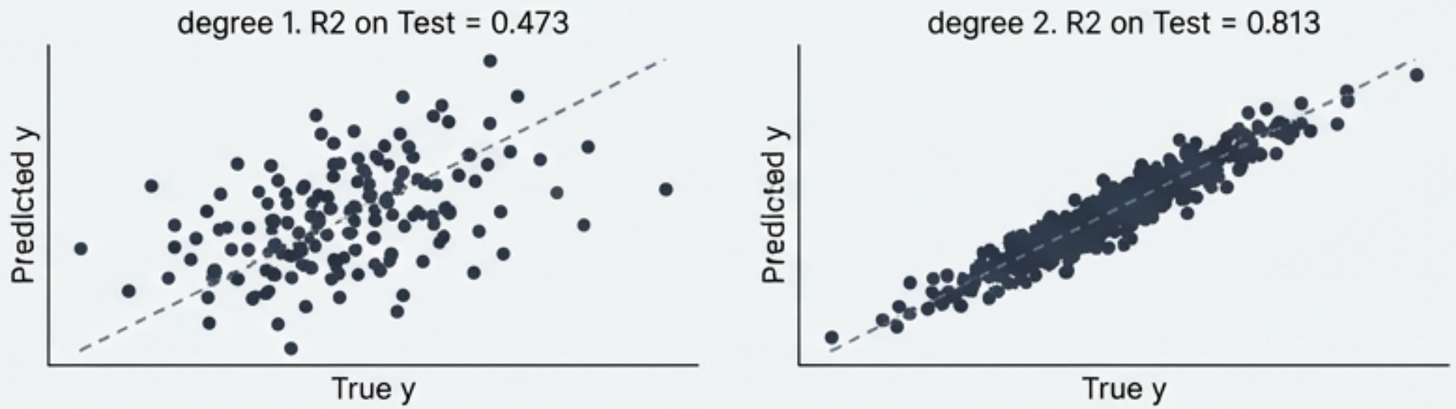
Measuring Performance: Coefficient of Determination (R^2)

R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It provides a relative measure of fit.

$$R^2 = 1 - \frac{\sum(y_n - \hat{y}_n)^2}{\sum(y_n - \bar{y})^2}$$

where $\bar{y}$ is the mean of the true y values.

- An R^2 of 1 indicates a perfect fit.
- An R^2 of 0 indicates the model performs no better than the mean $\bar{y}$.
- A negative R^2 indicates the model performs worse than the mean.



Strengths	**Weaknesses & Limitations**	**Core Assumptions**
Highly interpretable	Prone to underfitting complex data	Linear relationship between features and target
Computationally efficient	Sensitive to outliers	Homoscedasticity (constant variance of errors)
Provides a good baseline	Assumes features are independent	Independence of errors

Logistic Regression: The Probability of a Class

Formulation

To solve **binary classification** problems ($y \in \{0, 1\}$), we model the probability of an instance belonging to the positive class.

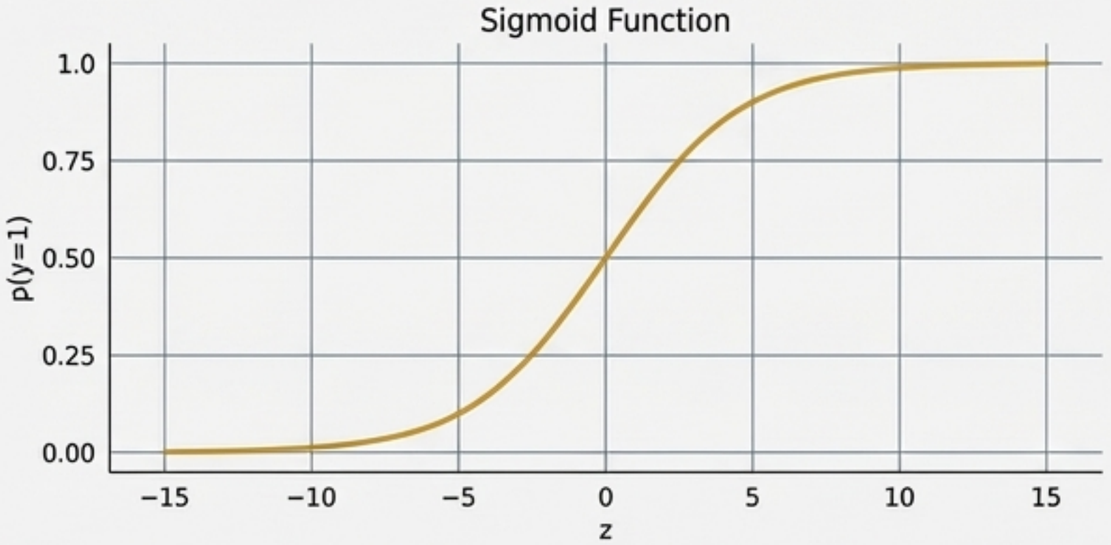
Mathematical Formulation (Hypothesis): A linear score $\mathbf{z} = \mathbf{w}^T\mathbf{x}$ is passed through the sigmoid (logistic) function $\sigma(\mathbf{z})$ to map it to a probability between 0 and 1.

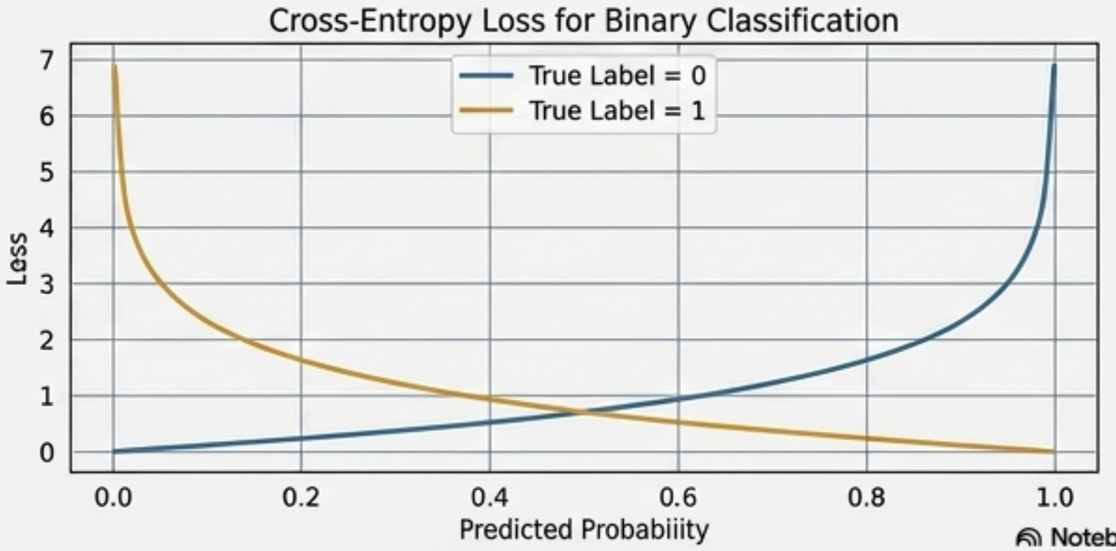
$$\mu = p(y{=}1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

Objective Function: Binary Cross-Entropy (BCE): The goal is to minimize the Negative Log-Likelihood of the training data. This loss function, known as **Binary Cross-Entropy**, heavily penalizes predictions that are confidentiy wrong.

$$L(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}[y_n \cdot \log(\mu_n) + (1-y_n) \cdot \log(1-\mu_n)]$$

Visuals





Optimization and Multi-Class Extension

Optimization via Gradient Descent

We find the optimal weights **w** by iteratively minimizing the BCE loss.

Gradient of BCE Loss

The gradient has a simple and intuitive form, representing the error ($\boldsymbol{\mu}_n - \mathbf{y}_n$) scaled by the input $\mathbf{x}_n$.

$$\nabla_w L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\mu_n - y_n)\cdot \mathbf{x}_n$$

Gradient Descent Update Rule

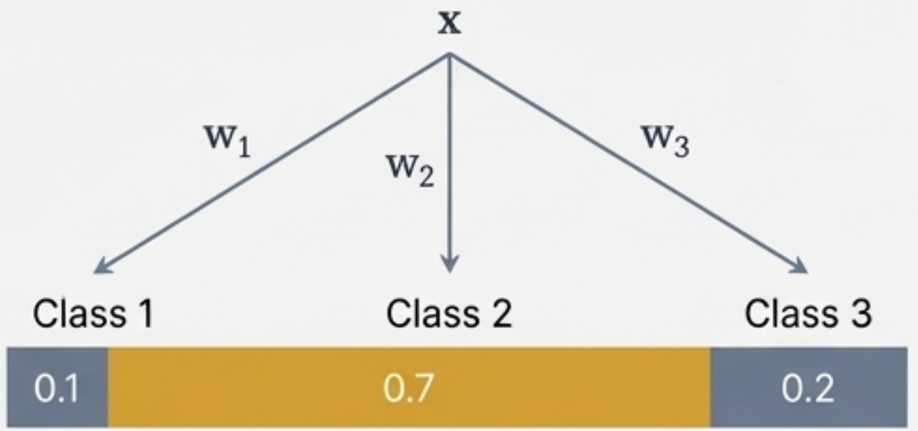
$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \cdot \frac{1}{N}\sum_{n=1}^{N}(\sigma(\mathbf{w}(t)^T\mathbf{x}_n) - y_n)\cdot \mathbf{x}_n$$

Deep Dive: Multi-Class Classification with Softmax

For C > 2 classes, the model uses C weight vectors and the Softmax function to compute a probability distribution over the classes.

Softmax Function: $p(y = i \mid \mathbf{x}; \mathbf{W}) = \frac{\exp(\mathbf{w}_i^T\mathbf{x})}{\sum_{j=1}^{C}\exp(\mathbf{w}_j^T\mathbf{x})}$

The corresponding loss function is the Categorical Cross-Entropy.



Evaluation and Model Summary

Measuring Performance for Classification

Unlike regression, we evaluate performance based on correct and incorrect classifications.

The **Confusion Matrix:** A table that visualizes the performance of an algorithm.

	Predicted: 1	**Predicted: 0**
Actual: 1	True Positive (TP)	False Negative (FN)
Actual: 0	False Positive (FP)	True Negative (TN)

Key Metrics Derived from the Matrix

- **Accuracy:** (TP + TN) / (TP + TN + FP + FN)
- **Precision:** TP / (TP + FP)
- **Recall (Sensitivity):** TP / (TP + FN)
- **AUC-ROC:** A plot of True Positive Rate vs. False Positive Rate, measuring performance across all classification thresholds.

Strengths	Weaknesses & Limitations	Core Assumptions
• Provides probabilistic outputs • Highly interpretable weights • Efficient to train	• Assumes a linear decision boundary • Can be outperformed by more complex models • Can be sensitive to multicollinearity	• Features are linearly related to the log-odds • Meaningful probabilities require well-calibrated model • Independence of irrelevant alternatives (for Softmax)

K-Means: Discovering Structure in Unlabeled Data

Core Concept

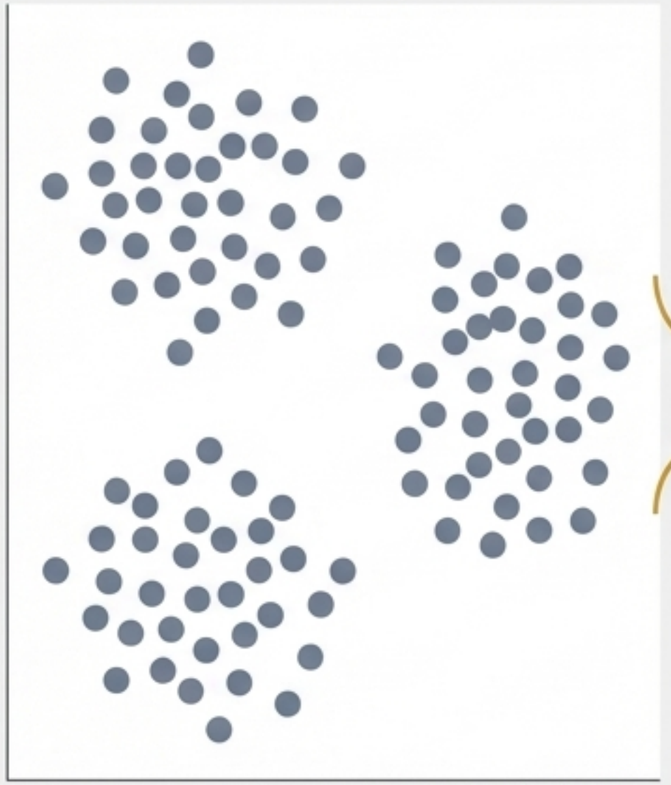
An unsupervised algorithm that partitions a dataset of N points into K distinct, non-overlapping clusters based on feature similarity.

The Objective Function: Within-Cluster Sum of Squares (WCSS)

The algorithm aims to find K cluster centroids $\{\mu_1, \ldots, \mu_K\}$ that minimize the WCSS (also called inertia). This is the sum of squared Euclidean distances between each data point $\mathbf{x}_n$ and the centroid μ_k of its assigned cluster C_k.

$$J = \sum_{k=1}^{K} \sum_{x_n \in C_k} \|\mathbf{x}_n - \mu_k\|^2.$$

Input Data



Clustered Output (K=3)

The K-Means Algorithm and Smarter Initialization

The Optimization Algorithm (Lloyd's Algorithm)

K-Means finds a local minimum of the WCSS objective through a two-step iterative process:

1. **Assignment Step**: Assign each data point x_n to the cluster C_k whose centroid μ_k is closest.

$$C_k^{(t)} = \{x_n : \|x_n - \mu_k^{(t-1)}\|^2 \leq \|x_n - \mu_j^{(t-1)}\|^2 \text{ for all } j\}$$

2. **Update Step**: Recalculate the centroid μ_k as the mean of all data points assigned to cluster C_k.

$$\mu_k^{(t)} = \frac{1}{|C_k^{(t)}|} \sum_{x_n \in C_k^{(t)}} x_n$$

The steps are repeated until the cluster assignments no longer change.

Deep Dive: K-Means++ for Better Initialization

Randomly choosing initial centroids can lead to poor results. K-Means++ is a smarter initialization that improves convergence and quality.

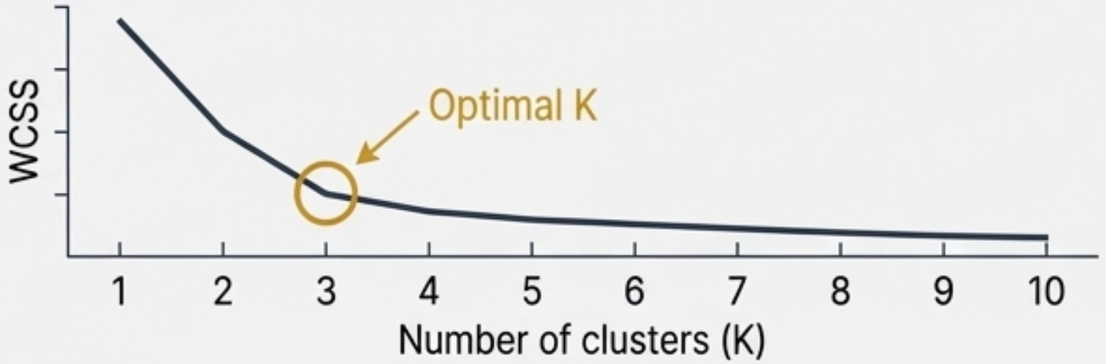
1. Choose the first centroid μ_1 uniformly at random from the data points.
2. For each data point x_n, compute $D(x_n)^2$, its squared distance to the *nearest* existing centroid.
3. Choose the next centroid from the data points with a probability proportional to $D(x_n)^2$.
4. Repeat steps 2 and 3 until K centroids have been chosen. This spreads out the initial centroids, leading to better solutions.

Evaluation and Model Summary

Measuring Performance for Clustering

The Elbow Method

To find an optimal value for K, plot WCSS as a function of K. The "elbow" of the curve—the point of diminishing returns—is a good estimate for the optimal K.



Silhouette Score

Measures how similar a point is to its own cluster compared to other clusters.

$$s(n) = \frac{b(n) - a(n)}{\max(a(n), b(n))}$$

where a(n) is the mean intra-cluster distance and b(n) is the mean nearest-cluster distance for point n.

Scores range from -1 to 1, with higher values indicating better-defined clusters.

Strengths	Weaknesses & Limitations	Core Assumptions
Simple and easy to implement	Requires the number of clusters K to be specified	Clusters are spherical, isotropic (equal variance)
Computationally efficient and scalable	Sensitive to initial centroid placement	All clusters have similar variance
Good for discovering basic group structures	Struggles with clusters of non-convex shapes	Hard assignment (each point to one cluster)

Multilayer Perceptron: Learning Non-Linear Functions

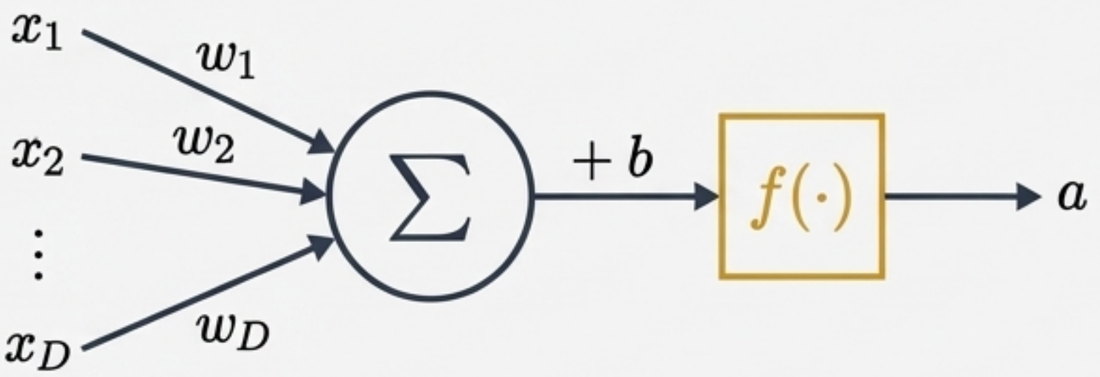
Core Concept

An MLP is a class of feedforward artificial neural network that can learn complex, non-linear relationships between inputs and outputs. It is a universal function approximator.

Mathematical Formulation (Single Neuron)

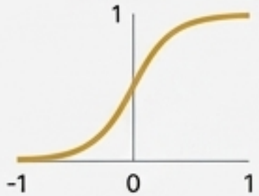
Each neuron computes a weighted sum of its inputs, adds a bias, and applies a non-linear activation function $f(\cdot)$.

$$a = f(z) = f\left(\sum_{i=1}^{D} w_i \cdot x_i + b\right)$$



Common Activation Functions $f(z)$

Sigmoid



$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Tanh

$$\tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$

ReLU

$$f(z) = \max(0, z)$$

Architecture

By stacking layers of these neurons, an MLP builds a complex, nested function. An L-layer MLP can be written as:

$$\hat{y} = f^{(L)}(W^{(L)} \cdot f^{(L-1)}(\dots f^{(1)}(W^{(1)} \cdot x + b^{(1)})\dots) + b^{(L)})$$

Objective

Minimize a loss function (e.g., MSE for regression, Cross-Entropy for classification) with respect to all weights W and biases b.

Optimization: The Backpropagation Algorithm

Core Concept

Backpropagation is the algorithm used to efficiently compute the gradients of the loss function with respect to every weight and bias in the network. These gradients are then used by an optimization algorithm like Gradient Descent.

The Mathematics: Chain Rule in Action

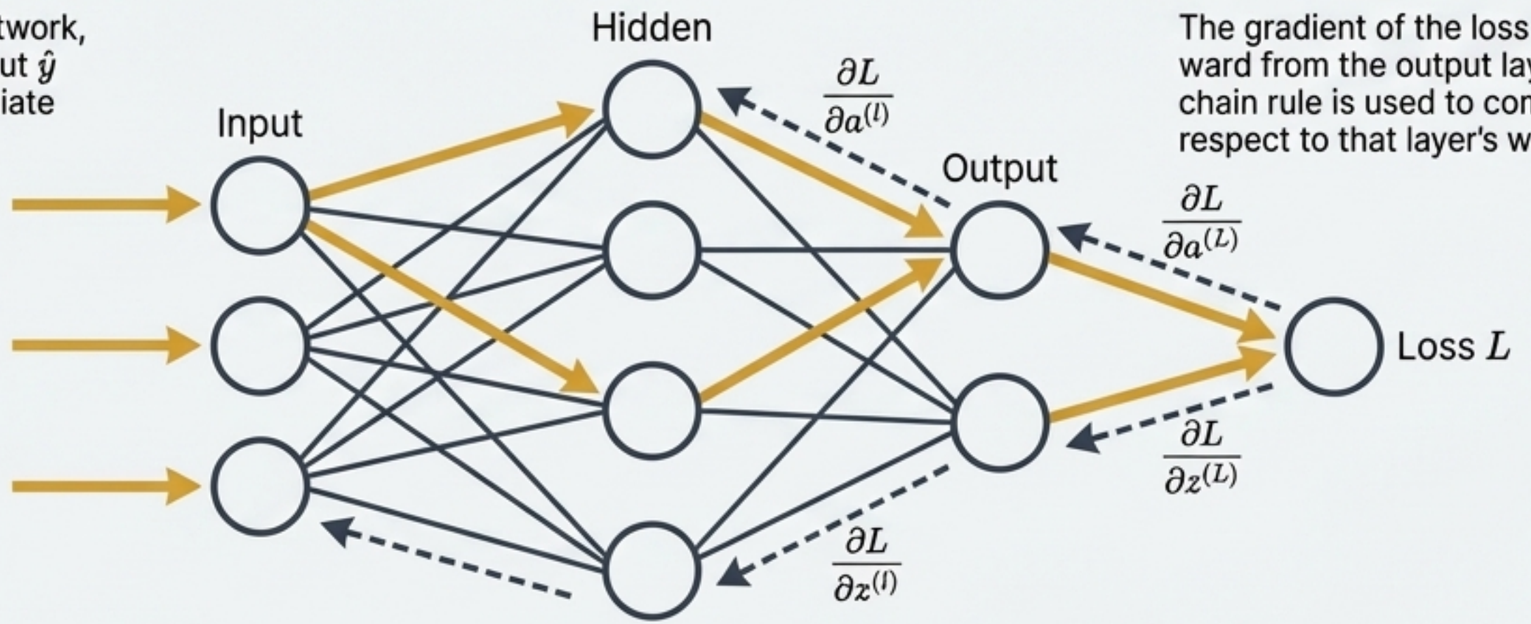
Backpropagation is an application of the chain rule for derivatives. It computes gradients by starting from the output layer and moving backward through the network.

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial a^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdot \dots \cdot \frac{\partial a^{(l)}}{\partial z^{(l)}} \cdot \frac{\partial z^{(l)}}{\partial w_{ij}^{(l)}}$$

The Algorithm in Two Passes

1. Forward Pass

Input x propagates through the network, layer by layer, to compute the output $\hat{y}$ and the final loss $L(y, \hat{y})$. Intermediate activations $a^{(l)}$ are stored.



2. Backward Pass

The gradient of the loss is propagated backward from the output layer. At each layer, the chain rule is used to compute gradients with respect to that layer's weights and biases.

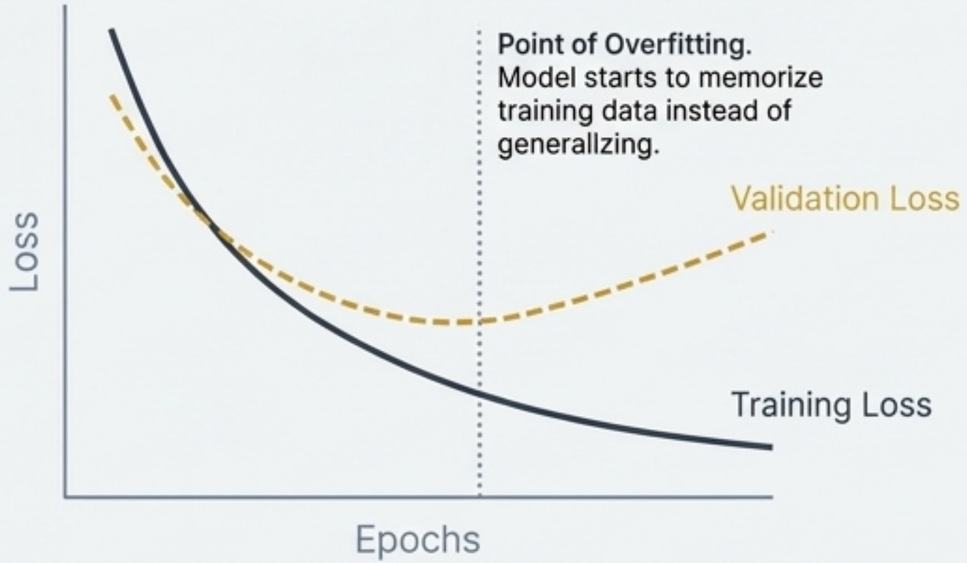
Evaluation and Model Summary

Measuring Performance

Evaluation depends on the task and follows standard practices:

- **Regression**: MSE, R^2
- **Classification**: Cross-Entropy Loss, Accuracy, Precision, Recall, F1-Score, Confusion Matrix.

A key challenge is monitoring for **overfitting**, where the model performs well on training data but poorly on unseen test data. This is often tracked by plotting training and validation loss curves over training epochs.



Strengths	Weaknesses & Limitations	Key Considerations
Universal Function Approximator: Can model any continuous function.	**"Black Box":** Difficult to interpret how it makes decisions.	**Hyperparameter Tuning:** Performance is highly sensitive.
Learns feature hierarchies automatically.	**Data Hungry:** Requires large amounts of labeled data.	**Regularization:** Crucial to prevent overfitting (e.g., L2, Dropout).
Foundation for more complex deep learning models.	**Computationally Expensive:** Requires significant hardware (GPUs).	**Vanishing/Exploding Gradients:** Addressed by ReLU, careful initialization.

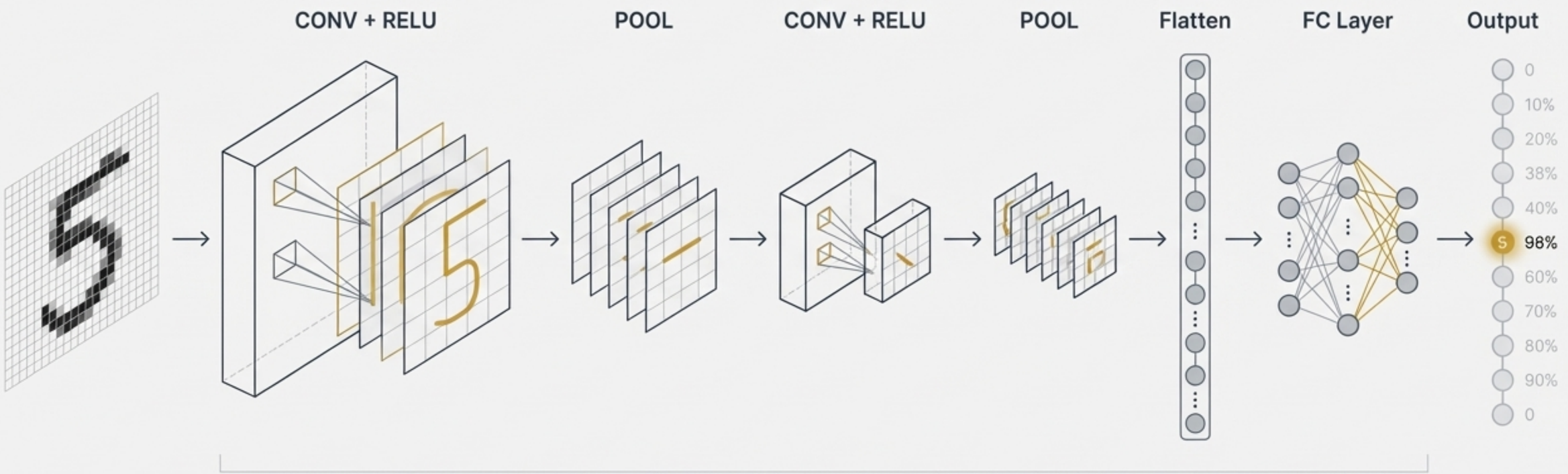
CNNs: The Mathematics of Perception

Core Concept

Convolutional Neural Networks (CNNs) are a class of deep neural networks designed to process data with a grid-like topology, such as images. They use specialized layers to learn a hierarchy of spatial features.

The Core Building Blocks

1. **Convolutional Layer:** Applies a set of learnable filters (kernels) to the input. Each filter detects a specific feature (e.g., an edge, a texture).
2. **Pooling (Subsampling) Layer:** Reduces the spatial dimensions of the feature maps, making the representation more compact and invariant to small translations.
3. **Fully Connected Layer:** A standard MLP that performs classification based on the high-level features extracted by the convolutional and pooling layers.



Flow of data through a Convolutional Neural Network, illustrating feature extraction and classification.

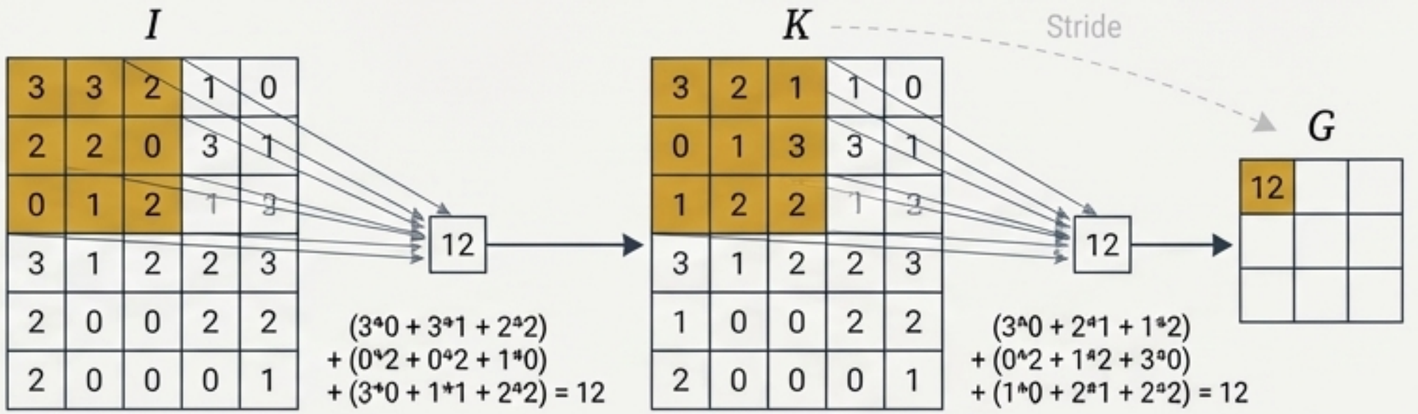
Mathematical Deep Dive: Convolution and Pooling

The Convolution Operation

The core operation is a 2D discrete convolution between an input I and a kernel K. The output feature map G is computed as:

$$G[i,j] = (I * K)[i,j] = \sum_m \sum_n I[i-m, j-n] * K[m,n]$$

Key Parameters: Kernel Size, Stride, Padding



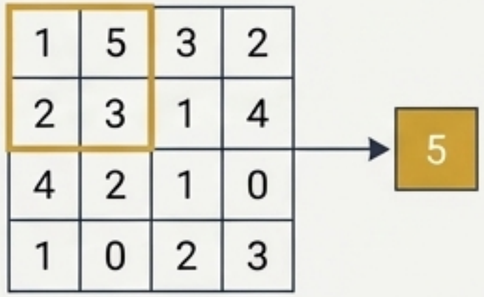
The Pooling Operation

Pooling summarizes a neighborhood of features. For a p x p region R:

Max Pooling

$y = \max(x_i)$ for $x_i \in R$

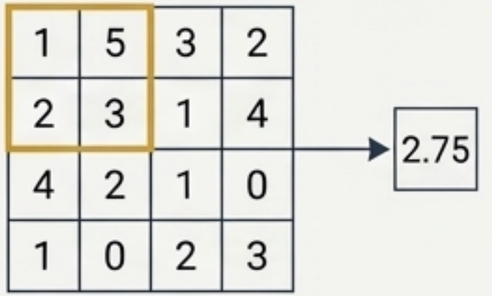
Selects the most activated feature, providing robustness to small translations.



Average Pooling

$y = \frac{1}{p^2} \sum (x_i)$ for $x_i \in R$

Provides a smoothed summary of the features.



Optimization for the entire network is performed using **Backpropagation**.

Evaluation and Model Summary

Performance & Principles

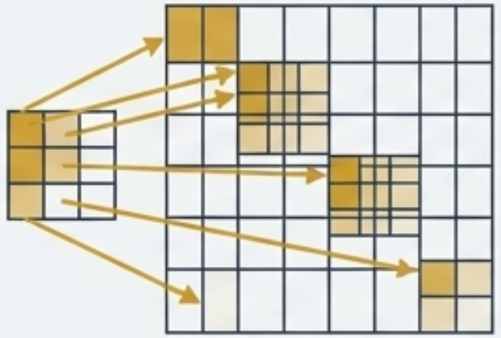
Measuring Performance

Standard classification metrics apply: Accuracy, Top-5 Accuracy, Confusion Matrix.

Why CNNs Excel: The Core Principles

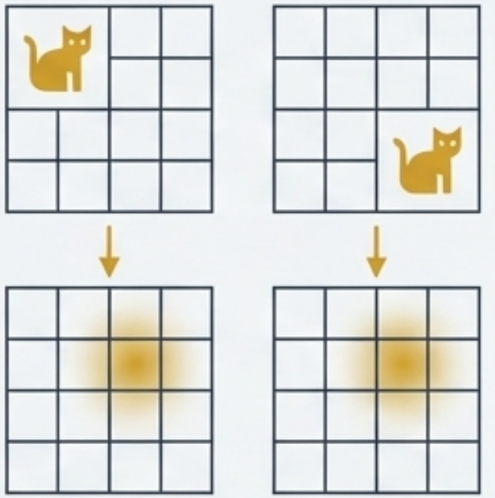
Principle 1: Parameter Sharing

The same kernel is applied across the entire image, drastically reducing parameters and allowing position-independent feature detection.



Principle 2: Translation Invariance

Convolution and pooling make feature detection robust to small shifts of an object in the image.



Strengths	Weaknesses & Limitations	Key Architectures
State-of-the-art on spatial data	Extremely data hungry	**LeNet-5**: Pioneering digit recognition.
Learns translation-invariant features	Computationally intensive	**AlexNet, VGG, ResNet**: Deeper architectures.
Highly scalable through depth	Less effective on non-grid data	**U-Net, YOLO**: Segmentation & object detection.

Synthesis: A Unified View and Future Directions

The Machine Learning Toolkit Recapped

This presentation has constructed a toolkit progressing in complexity and capability:

Linear Regression

Form the foundation for modeling structured data with clear, interpretable relationships.

Logistic Regression

Form the foundation for modeling structured data with clear, interpretable relationships.

K-Means Clustering

Provides a fundamental tool for unsupervised discovery of latent groups in data.

MLP (Multilayer Perceptron)

CNNs (Convolutional Neural Networks)

Represent the power of deep learning to model highly complex, non-linear patterns in both structured (MLP) and unstructured, spatial data (CNN).

Next Steps and Further Study

The principles of loss functions, optimization, and hierarchical feature learning extend to other advanced domains:

- **Recurrent Neural Networks (RNNs) & LSTMs**: For modeling sequential data like time series and natural language.
- **Transformers**: The current state-of-the-art architecture for natural language processing and, increasingly, computer vision.
- **Generative Models (GANs, VAEs, Diffusion Models)**: For creating new data rather than just making predictions on existing data.