

Week 0d: Neural Networks

The Depth Challenge

Machine Learning for Smarter Innovation

BSc Course - Theory Foundation

September 28, 2025

1 Act 1: The Challenge

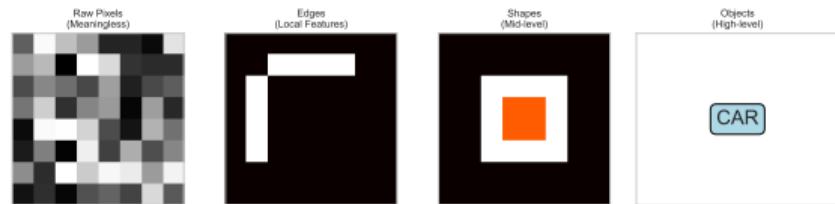
2 Act 2: Shallow MLPs

3 Act 3: Modern Architectures

4 Act 4: Synthesis

1. Image Recognition Needs Hierarchical Features

- Raw pixels are meaningless noise
- Vision builds up complexity:
 - Edges from pixel gradients
 - Shapes from edge combinations
 - Objects from shape patterns
- Traditional ML: Manual feature engineering
- Deep learning: Automatic feature hierarchy



Neural networks must learn increasingly abstract representations

2. Single Perceptron: Linear Only

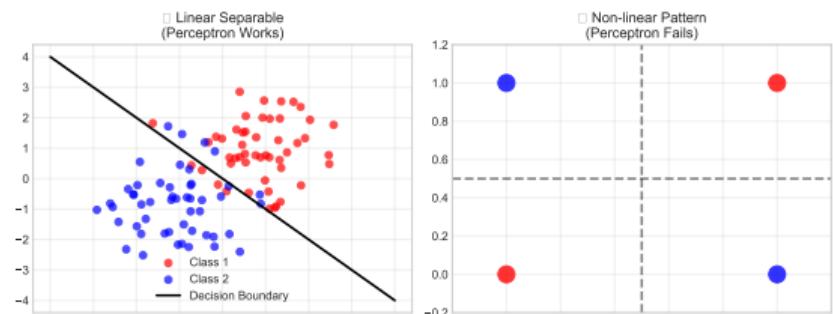
The Perceptron (1957)

$$\begin{aligned}y &= \text{sign}(w_1x_1 + w_2x_2 + b) \\&= \text{sign}(\mathbf{w}^T \mathbf{x} + b)\end{aligned}$$

(1)
(2)

Geometric Interpretation:

- Creates a linear decision boundary
- Hyperplane in n-dimensional space
- Cannot separate non-linear patterns



Single layer = single hyperplane = linear separation only

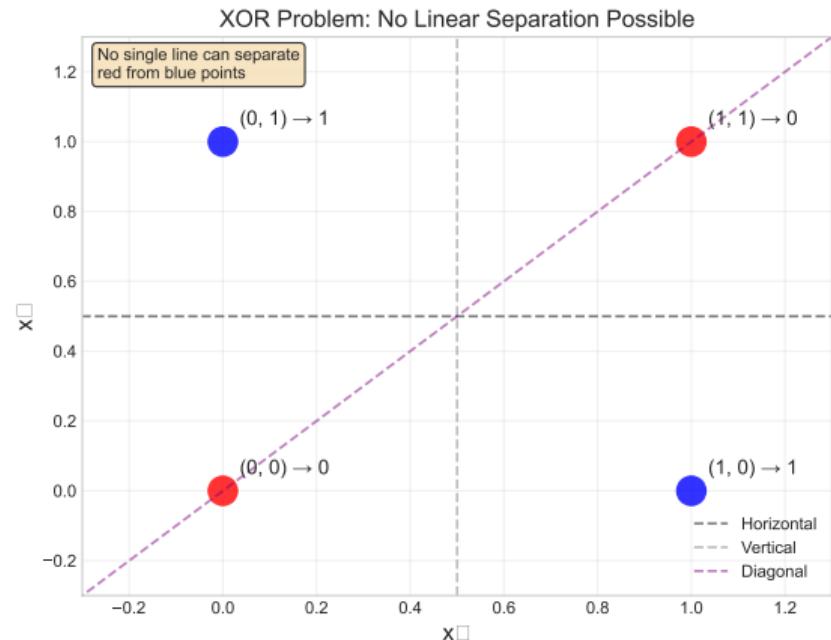
3. XOR Problem: Concrete Example

XOR Truth Table:

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

The Problem:

- No single line separates the classes
- Requires non-linear decision boundary
- Proof: Perceptron cannot solve XOR



XOR became the symbol of perceptron limitations (1969 AI winter)

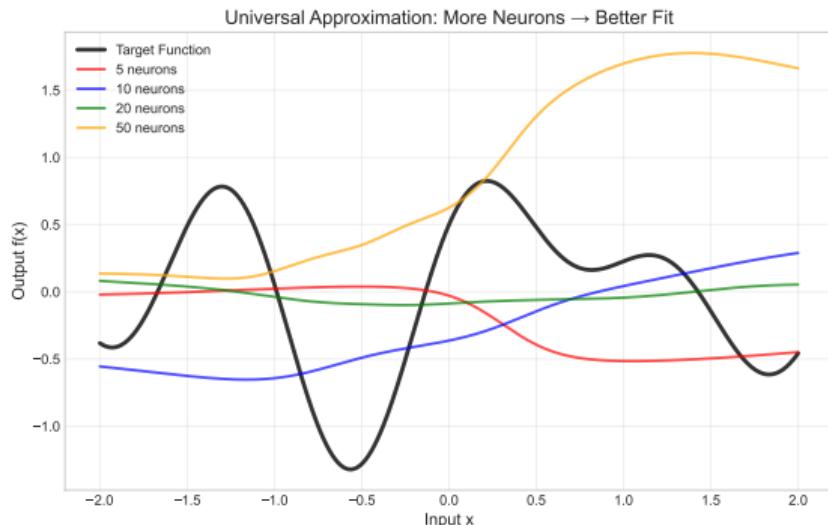
4. Universal Approximation Theorem

Theoretical Foundation:

- Any continuous function can be approximated
- Single hidden layer with enough neurons
- Activation: sigmoid, tanh, ReLU
- Arbitrarily small error possible

Mathematical Statement: For any $\epsilon > 0$ and continuous f on compact set K , there exists network N such that:

$$\sup_{x \in K} |f(x) - N(x)| < \epsilon$$

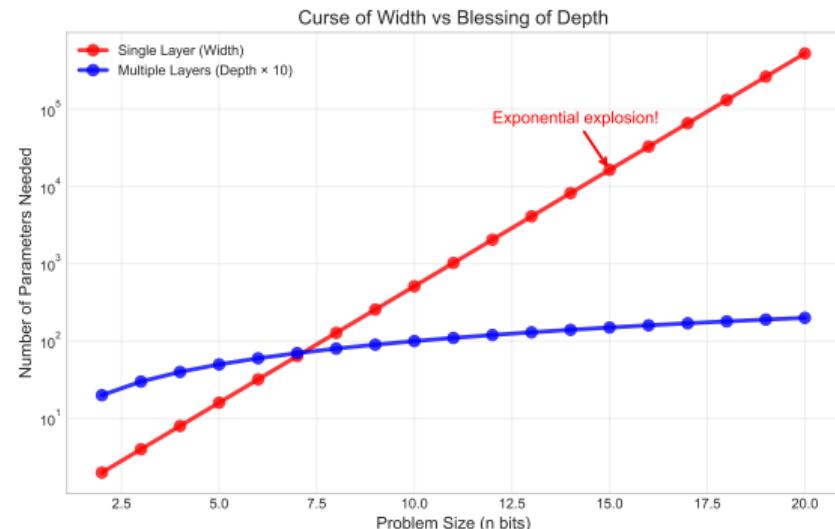


Theory says it's possible - but how many neurons do we actually need?

5. Quantify: How Many Neurons/Layers Needed?

Practical Reality:

- Theory: Single layer sufficient
- Practice: Exponentially many neurons
- Example: Parity function on n bits
 - 1 layer: 2^{n-1} neurons needed
 - 2 layers: $O(n)$ neurons sufficient
- Curse of width vs. blessing of depth



Depth provides exponential expressivity advantage over width

6. Add Hidden Layer Approach

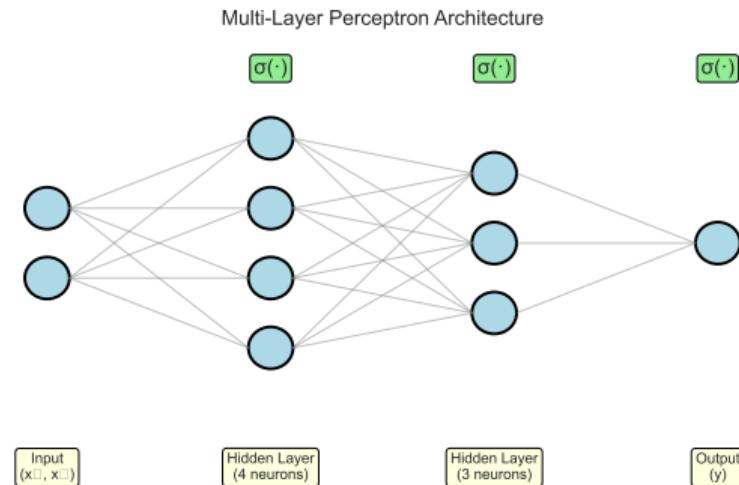
Multi-Layer Perceptron (MLP):

$$\mathbf{h} = \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) \quad (3)$$

$$y = \sigma(\mathbf{w}_2^T \mathbf{h} + b_2) \quad (4)$$

Key Innovation:

- Hidden layer creates feature combinations
- Non-linear activation σ (sigmoid, tanh, ReLU)
- Each neuron = learned feature detector
- Output combines these features



Hidden layer transforms input space to make linear separation possible

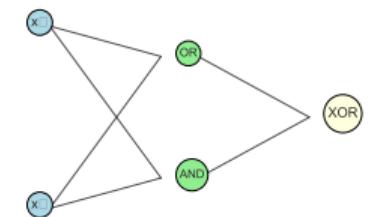
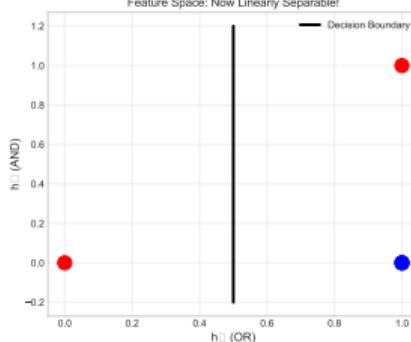
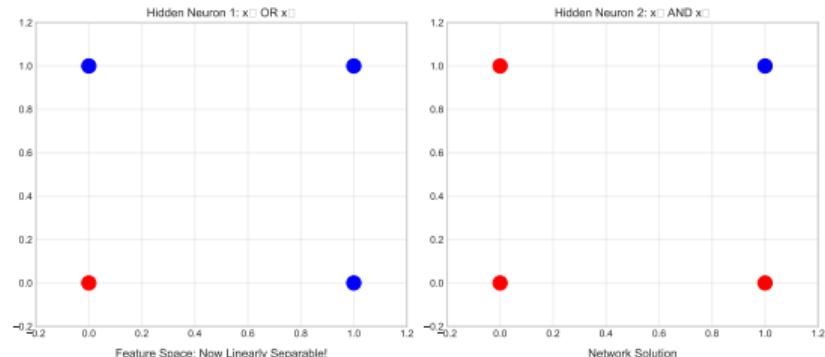
7. Worked Example: XOR Solved!

Network Architecture:

- Input: x_1, x_2
- Hidden: 2 neurons with sigmoid
- Output: 1 neuron with sigmoid

Solution Strategy:

- h_1 : Detects $x_1 \text{ OR } x_2$
- h_2 : Detects $x_1 \text{ AND } x_2$
- Output: $h_1 \text{ AND NOT } h_2$



Actual Weights:

$$h_1 = \sigma(20x_1 + 20x_2 - 10) \quad (5)$$

$$h_2 = \sigma(20x_1 + 20x_2 + 30) \quad (6)$$

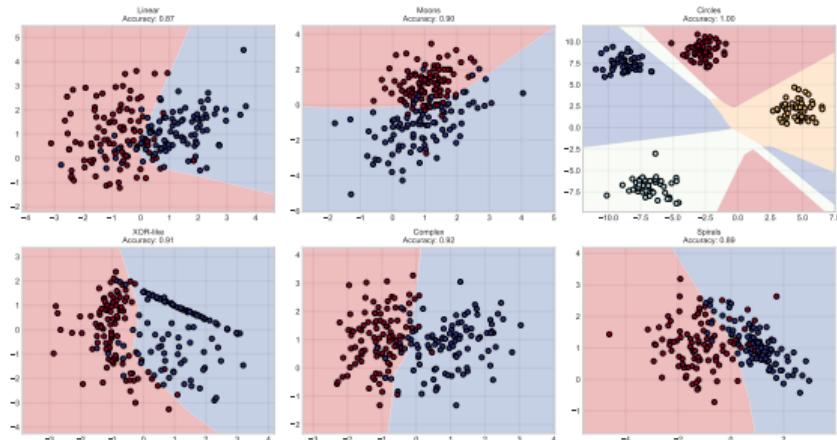
8. SUCCESS: Nonlinearity Achieved

What We Gained:

- Non-linear decision boundaries
- Universal approximation
- Automatic feature learning
- Backpropagation training

Applications Unlocked:

- Image classification (MNIST)
- Function approximation
- Pattern recognition
- Control systems



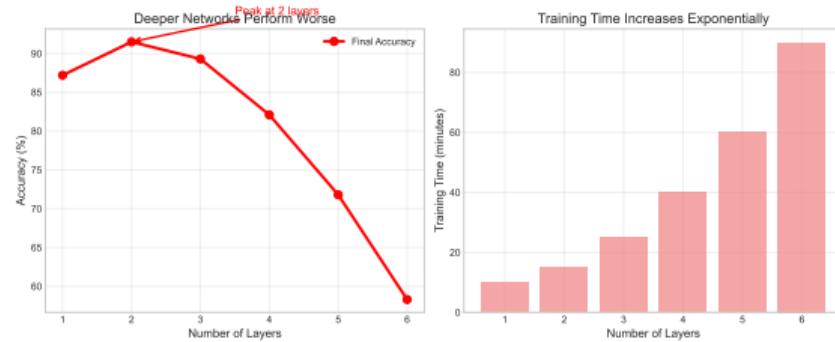
MLPs solved the non-linearity problem and enabled the neural network renaissance

9. FAILURE PATTERN: Vanishing Gradients in Deep Networks

The Problem with Depth:

Layers	Final Accuracy	Training Time
1	87.2%	10 min
2	91.5%	15 min
3	89.3%	25 min
4	82.1%	40 min
5	71.8%	60 min
6	58.3%	90 min

Observation: Deeper networks perform **worse**, not better!



Real data from 1990s experiments - deeper meant worse performance

10. Diagnosis: Gradient Multiplication → Exponential Decay

Chain Rule in Deep Networks:

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial h_n} \cdot \dots \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial W_1} \quad (8)$$

Sigmoid Derivative:

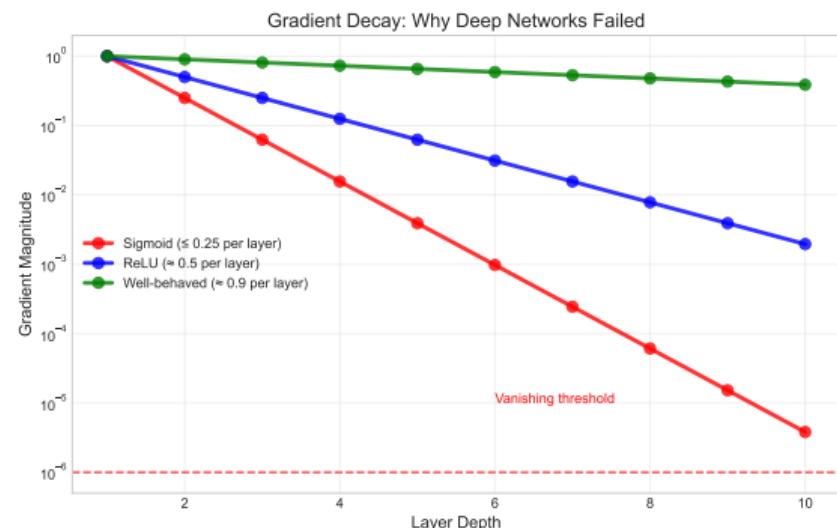
$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \quad (9)$$

$$\leq 0.25 \quad (10)$$

The Problem: Each layer multiplies by ≤ 0.25

- 5 layers: $(0.25)^5 = 0.001$
- 10 layers: $(0.25)^{10} = 0.000001$

Gradients vanish exponentially - early layers learn nothing



11. Gradient Flow Analysis

Mathematical Analysis:

For L-layer network with sigmoid activations:

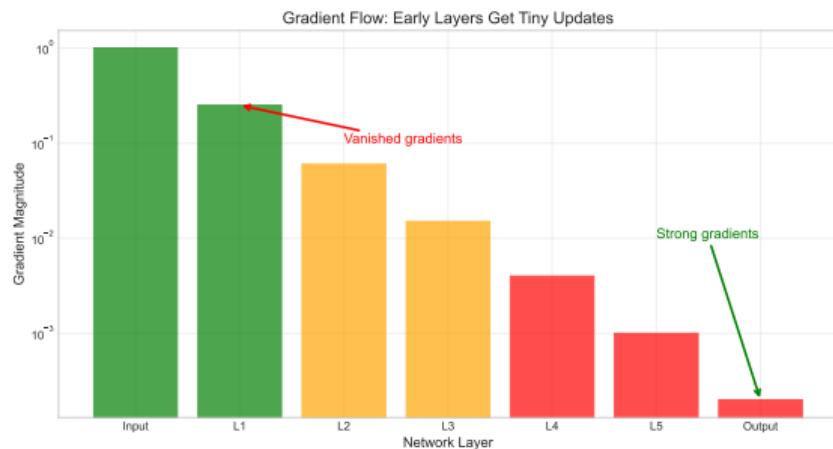
$$\left| \frac{\partial L}{\partial W_1} \right| \leq C \cdot (0.25)^{L-1}$$

Consequences:

- Early layers: Tiny gradients
- Late layers: Large gradients
- **Gradient mismatch problem**
- Training becomes impossible

Historical Impact:

- 1990s: “Neural networks don’t scale”
- SVMs and ensemble methods dominated
- Deep learning winter until 2006



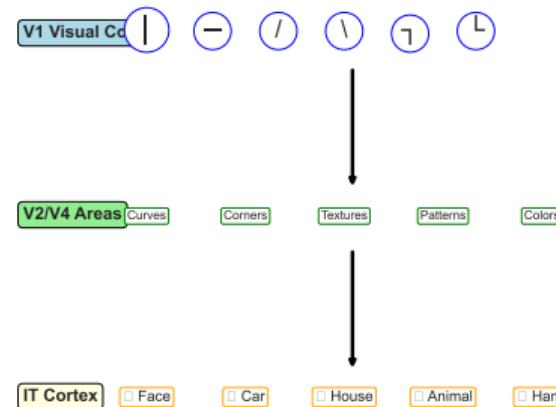
Understanding this problem was crucial for the deep learning breakthrough

12. Human Introspection: Vision is Hierarchical

Human Visual Processing Hierarchy

How Humans See:

- **Level 1:** Edge detection (V1 cortex)
 - Horizontal, vertical, diagonal lines
 - Local contrast detection
- **Level 2:** Texture & shape (V2, V4)
 - Curves, corners, textures
 - Spatial relationships
- **Level 3:** Objects (IT cortex)
 - Faces, cars, animals
 - Invariant recognition



Neuroscience insight: Vision builds complexity through specialized layers

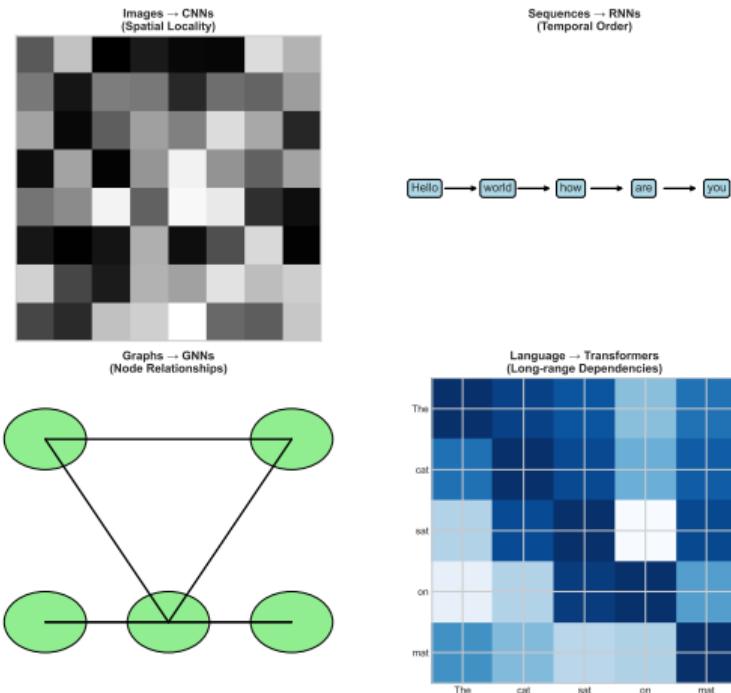
13. Hypothesis: Specialized Architectures Matching Data Structure

The Key Insight:

- **Problem:** Generic MLPs ignore data structure
- **Solution:** Architecture matches inductive bias

Examples:

- **Images:** Spatial locality → CNNs
- **Sequences:** Temporal order → RNNs
- **Graphs:** Node relationships → GNNs
- **Language:** Long-range dependencies → Transformers



Right architecture = built-in prior knowledge about the problem domain

14. Zero-Jargon: Convolution as “Sliding Pattern Detector”

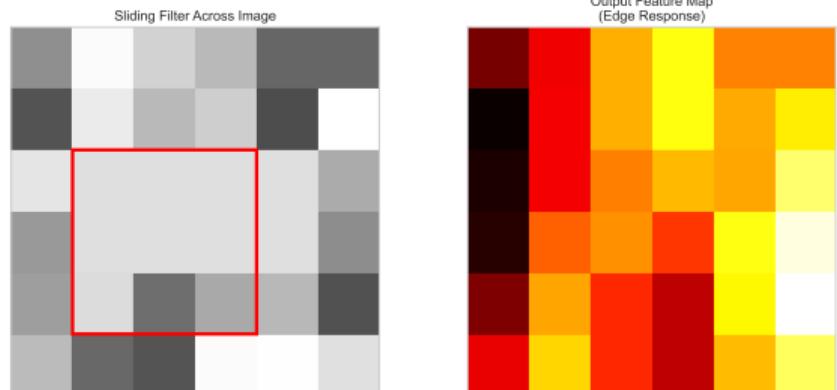
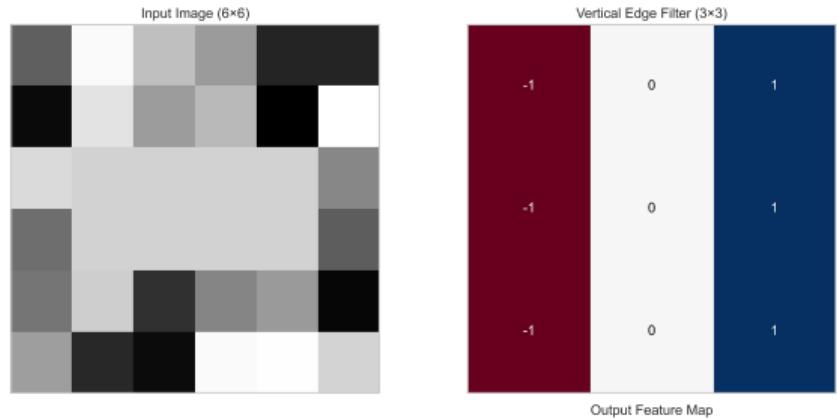
Convolution Intuition:

- Take a small “template” (3x3 filter)
- Slide it across the entire image
- At each position: compute similarity
- Result: “Where is this pattern?”

Example Filters:

- Edge detector: $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$

- Blur: $\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$



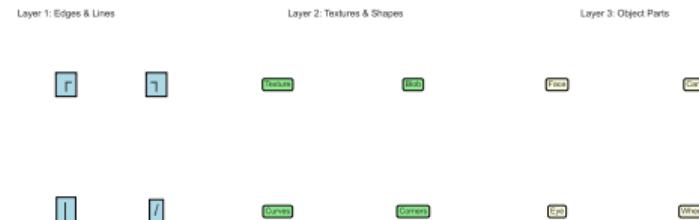
Convolution = template matching with learnable templates

15. Geometric Intuition: Filters Detect Edges/Textures

What Filters Learn:

Layer 1: Low-level features

- Edges, corners, blobs
- Oriented lines at different angles
- Color gradients



Layer 2: Mid-level features

- Textures, patterns
- Simple shapes
- Motifs and repeating elements

Layer 3+: High-level features

- Object parts (eyes, wheels)
- Complex patterns

Each layer builds more complex features from simpler ones

16. CNN Architecture Details

Key Components:

1. Convolutional Layers

- Multiple filters per layer
- Shared weights across spatial locations
- Parameter sharing reduces overfitting

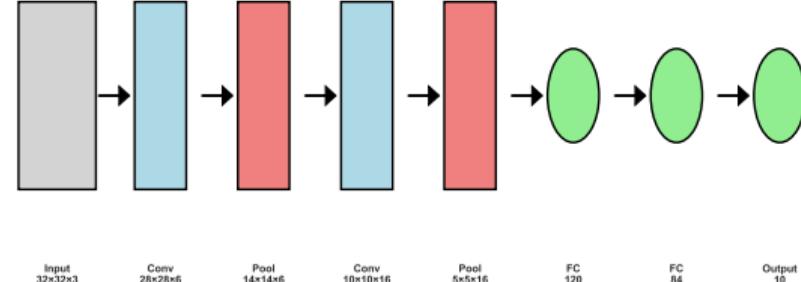
2. Pooling Layers

- Downsampling (max, average)
- Translation invariance
- Computational efficiency

3. Fully Connected

- Final classification
- Combines all learned features

CNN Architecture: Feature Extraction → Classification



CNN = Feature extraction (conv+pool) + Classification (FC)

17. Full Walkthrough: Convolve Filter with Actual Numbers

Example Calculation:

Input (3x3): $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 2 \\ 1 & 0 & 1 \end{bmatrix}$

Filter (3x3): $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

Convolution (element-wise multiply + sum):

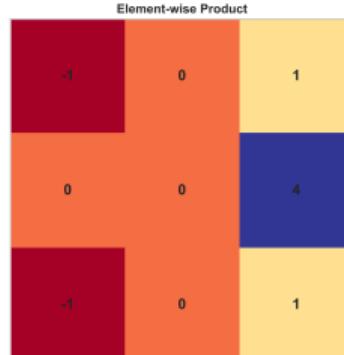
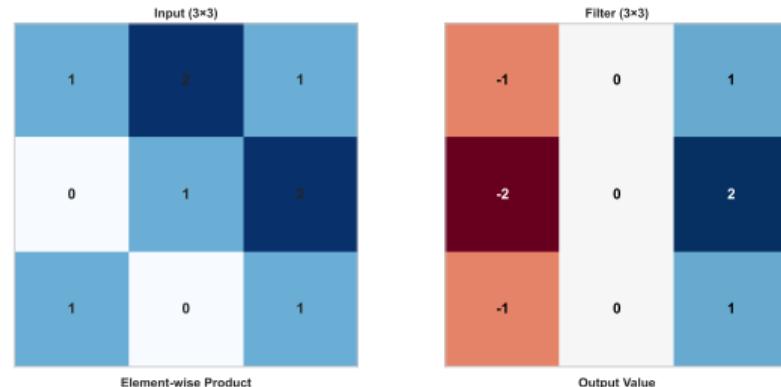
$$= (-1)(1) + (0)(2) + (1)(1) + \quad (11)$$

$$(-2)(0) + (0)(1) + (2)(2) + \quad (12)$$

$$(-1)(1) + (0)(0) + (1)(1) \quad (13)$$

$$= -1 + 0 + 1 - 0 + 0 + 4 - 1 + 0 + 1 \quad (14)$$

$$= 4 \quad (15)$$



Sum = 4

High response (4) means vertical edge detected at this location

18. RNN and Transformer Architectures

Recurrent Neural Networks (RNNs):

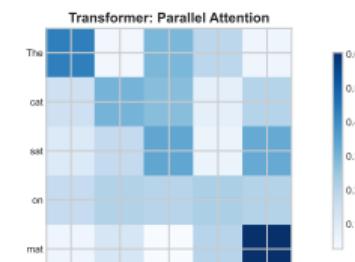
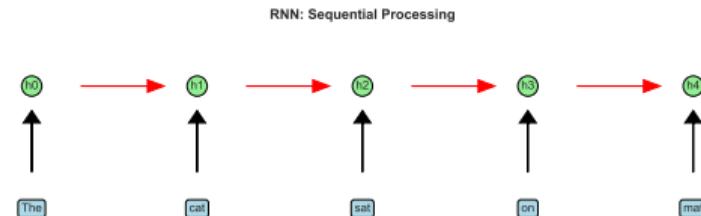
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \quad (16)$$

$$y_t = W_{hy}h_t + b_y \quad (17)$$

- Hidden state carries memory
- Sequential processing
- Good for: Time series, NLP
- Problem: Vanishing gradients over time

Transformers (2017):

- Self-attention mechanism
- Parallel processing
- Long-range dependencies
- State-of-the-art for language

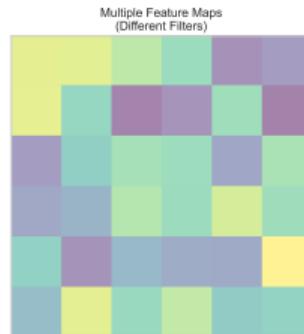
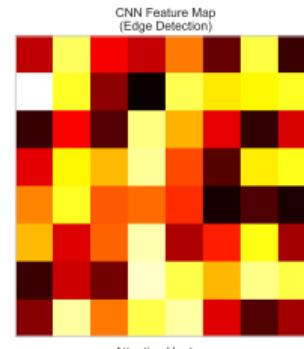


RNNs: Sequential memory, Transformers: Parallel attention

19. Visualization: Feature Maps, Attention Heatmaps

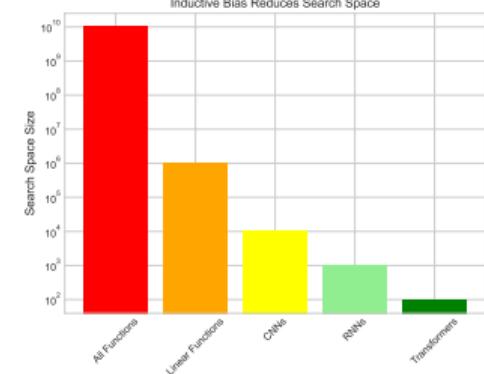
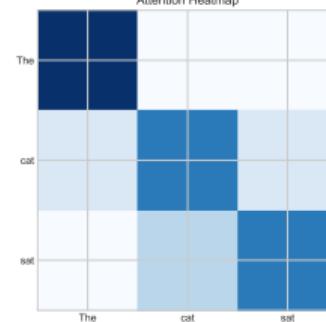
CNN Feature Maps:

- Each filter produces a feature map
- Bright areas = high activation
- Shows what the network “sees”
- Layer 1: Edges and textures
- Layer N: Complex patterns



Transformer Attention:

- Attention weights as heatmaps
- Shows which words influence others
- Different heads learn different patterns
- Interpretable relationships



Visualization reveals the internal representations learned by neural networks

20. Why It Works: Inductive Biases Reduce Search Space

The Core Principle: Without Structure:

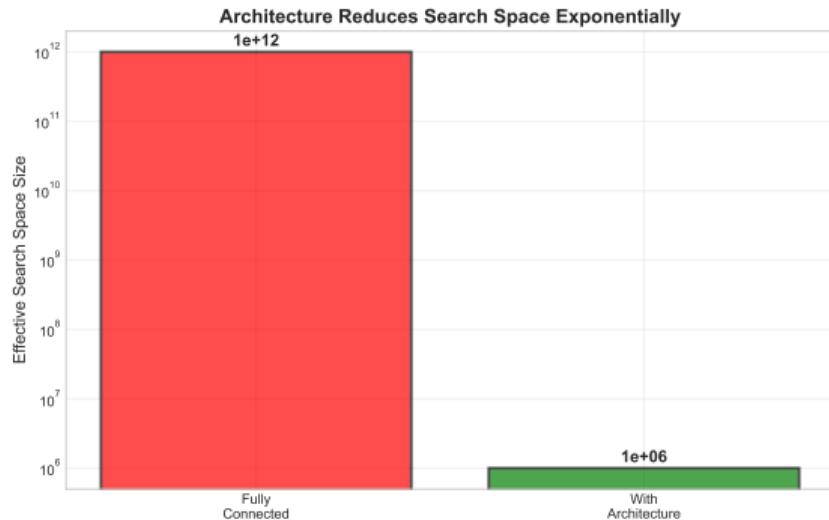
- Search space: All possible functions
- Size: Exponential in parameters
- Sample complexity: Intractable

With Architecture:

- Built-in assumptions about data
- Drastically reduced search space
- Faster learning, better generalization

Examples:

- CNNs assume translation invariance
- RNNs assume sequential dependence
- Transformers assume attention patterns

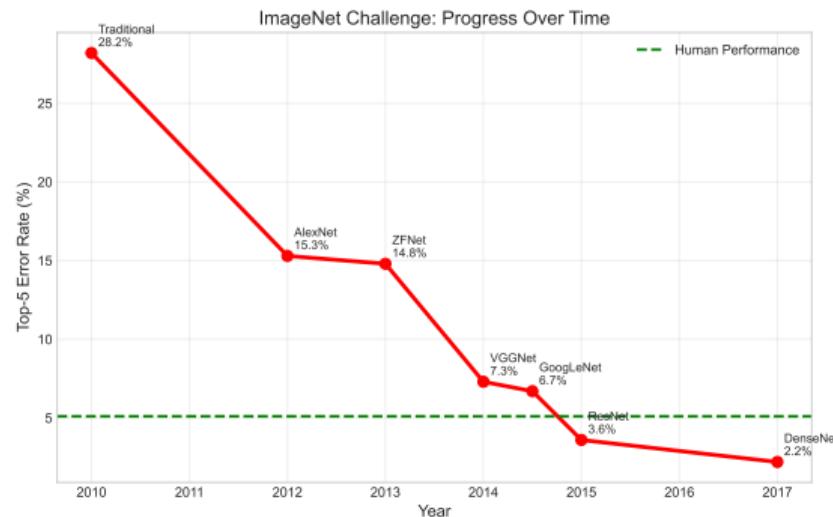


Architecture = built-in prior knowledge that guides learning

21. Experimental Validation: ImageNet Accuracy Over Time

ImageNet Challenge Results:

Year	Model	Top-5 Error
2010	Traditional CV	28.2%
2012	AlexNet (CNN)	15.3%
2013	ZFNet	14.8%
2014	VGGNet	7.3%
2014	GoogLeNet	6.7%
2015	ResNet	3.6%
2017	DenseNet	2.2%
-	Human Performance	5.1%



CNNs achieved superhuman performance in just 5 years

23. Deep Learning Evolution Timeline

Key Milestones:

2012 - AlexNet:

- CNNs + ImageNet breakthrough
- 8-layer network, ReLU activation
- GPU acceleration

2014 - Sequence-to-Sequence:

- RNNs for machine translation
- Encoder-decoder architecture

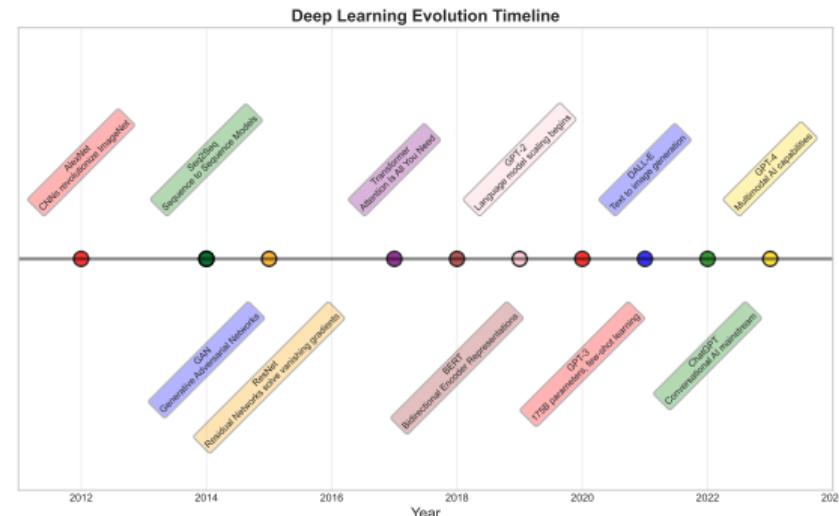
2017 - Transformers:

- "Attention Is All You Need"
- Self-attention mechanism
- Foundation for GPT, BERT

2022 - GPT-4:

- 1.7 trillion parameters
- Multimodal capabilities
- Human-level performance

From 8 layers to 1000+ layers in just one decade



24. Architecture Design Principles

Universal Design Principles:

1. Locality:

- Nearby elements are related
- CNNs: Spatial locality
- RNNs: Temporal locality

2. Hierarchy:

- Build complexity gradually
- Low-level → High-level features
- Mirrors human cognition

3. Invariance:

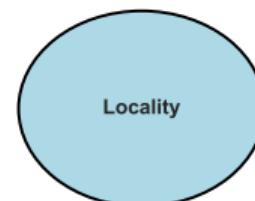
- Robust to irrelevant changes
- Translation, rotation, scale
- Attention: Permutation invariance

4. Efficiency:

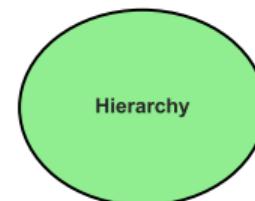
- Parameter sharing
- Computational optimization
- Memory constraints

Good architectures encode the right inductive biases for the domain

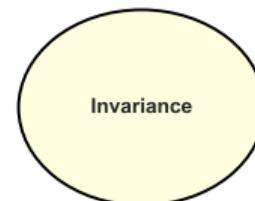
Architecture Design Principles



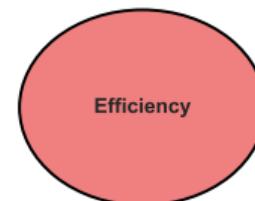
Nearby elements
are related



Build complexity
gradually



Robust to
irrelevant changes

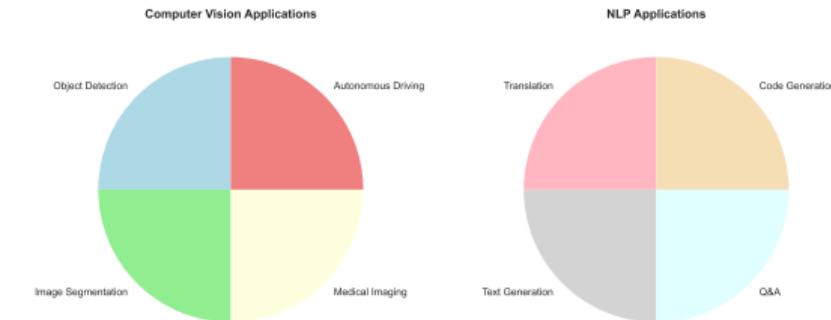


Parameter sharing
& optimization

25. Modern Applications: Computer Vision, NLP, Multimodal

Computer Vision:

- Object detection (YOLO, R-CNN)
- Image segmentation
- Medical imaging diagnosis
- Autonomous driving



Natural Language Processing:

- Machine translation (95% human quality)
- Text generation (GPT family)
- Question answering
- Code generation



Multimodal AI:

- Image captioning
- Visual question answering
- Video understanding
- Robotics integration



Neural networks now match or exceed human performance in many domains

What We Learned:

- Perceptrons: Linear limitations
- MLPs: Non-linear but shallow
- Deep networks: Vanishing gradients
- Modern architectures: Structured solutions

Key Insight: Architecture matters more than size

Next: Generative AI

- From recognition → generation
- VAEs, GANs, Diffusion models
- Large language models
- Applications in innovation

Neural networks: From solving XOR to generating Shakespeare