上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Lecture 17: Convolutional Neural Networks

Tao Huang

John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

https://taohuang.info/cs3317

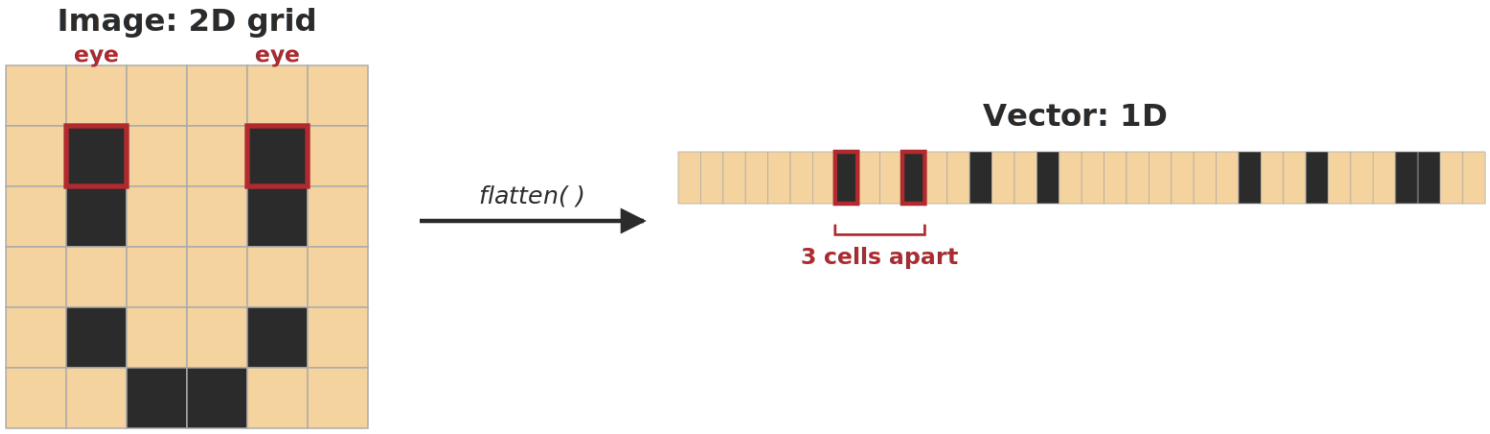
https://oc.sjtu.edu.cn/courses/89538

AI tools assisted in generating some figures in these slides. All such content has been reviewed, and the instructor is responsible for its accuracy.

From Optimization to Architecture

From “Can We Train Deep Nets?” to “What Should They Look Like?”

- Last week: init + norm + optimizer + regularization made deep MLPs trainable
- But an MLP on a 224×224 image is still disappointing. Why?

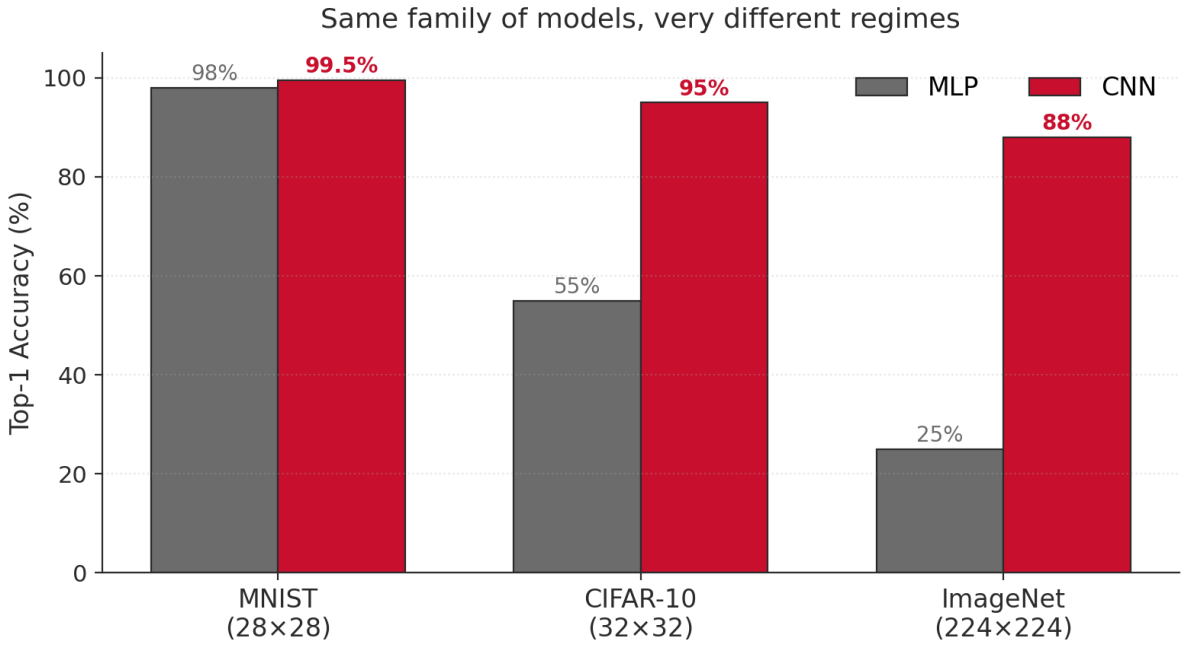


To an MLP, the two eyes and the chin are equally “close” — locality is destroyed.

- **Today: stop fixing optimization, start fixing architecture**

MNIST Lied to Us

- MLP on MNIST: ~98%. Problem solved?
- CIFAR-10 (32×32 color): MLP plateaus around 50–55%
- ImageNet (224×224): MLP is not a serious baseline
- The gap between MNIST and CIFAR is the gap between “pixel vector” and “image”



Objectives

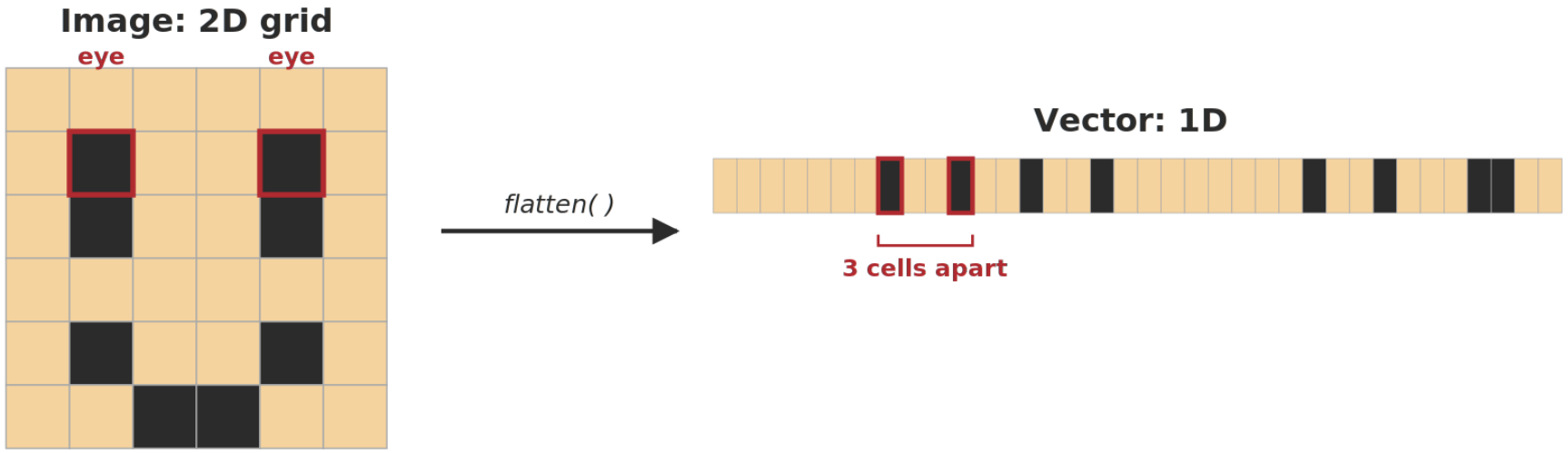
By the end of this lecture, you should be able to:

- Explain why fully-connected layers are structurally wrong for images
  - in terms of parameter count, translation, and locality
- Derive convolution as the unique linear operator satisfying locality + translation equivariance
- Compute output shapes, receptive fields, parameter counts for arbitrary conv layers
- Analyze the lineage LeNet → AlexNet → VGG → GoogLeNet → ResNet as responses to bottlenecks
- Evaluate when a CNN's inductive bias is an asset vs. a liability (foreshadowing ViT)

1. Why MLPs Break on Images

Flattening Throws Away Everything

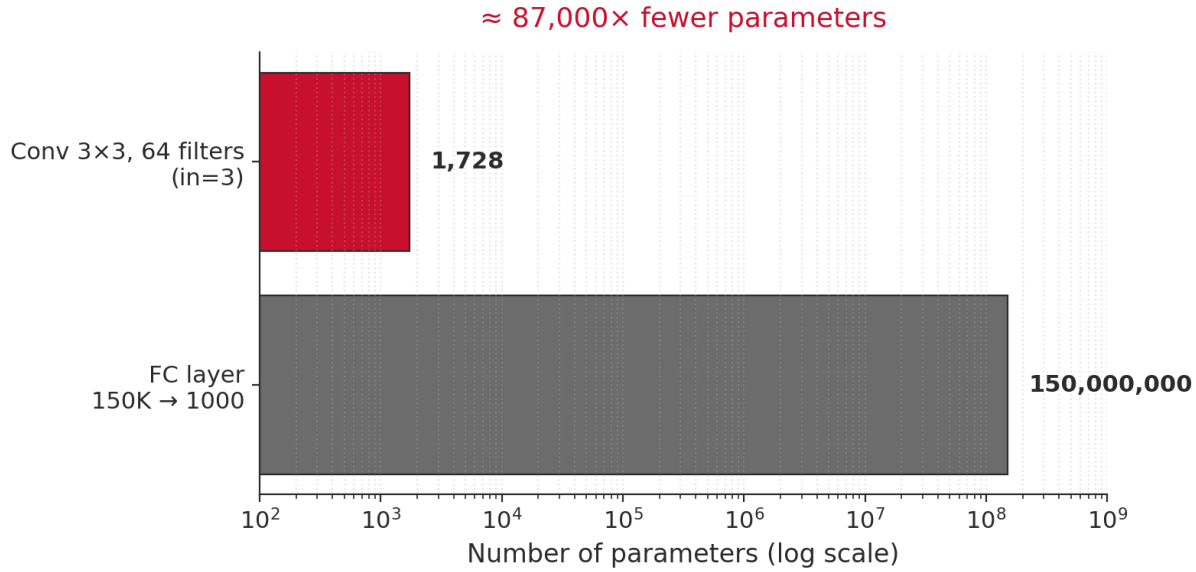
- An image is not x ∈ ℝ^(HW) — it is x ∈ ℝ^(H×W), a signal on a 2D grid with three structural properties:
  - Locality — meaning lives in pixel neighborhoods (an edge is 3–5 pixels wide)
  - Stationarity — a cat patch is a cat patch anywhere in the image
  - Compositionality — edges → textures → parts → objects



To an MLP, the two eyes and the chin are equally "close" — locality is destroyed.

Do the Math on a 224×224 Image

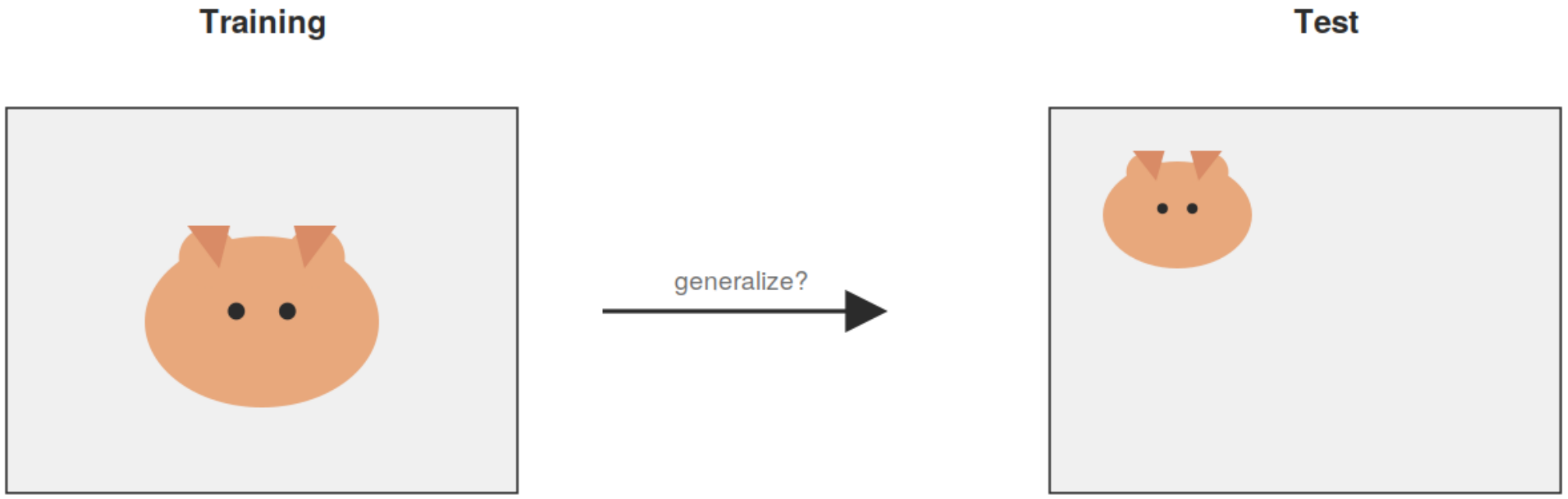
- Input: 224 × 224 × 3 = 150,528 inputs
- **First hidden layer with 1000 units: ~150 million params**
- **Three problems:**
  - Memory — won‘t fit on GPUs in 2012
  - Data — can't constrain 150M params from 1.2M labels
  - Relearning the same feature at every position



CNN's first conv layer: ~2K parameters. Four orders of magnitude less, more expressive.

A Thought Experiment (30 seconds)

- You train an MLP on “cat vs. dog” where cats are always centered.
- At test time, cats appear in the top-left corner. What happens?



MLPs learn position-bound weights — "cat at center" ≠ "cat at corner"

What Linear Operator Should Replace the FC Layer?

We want a layer y = f(x) that is:

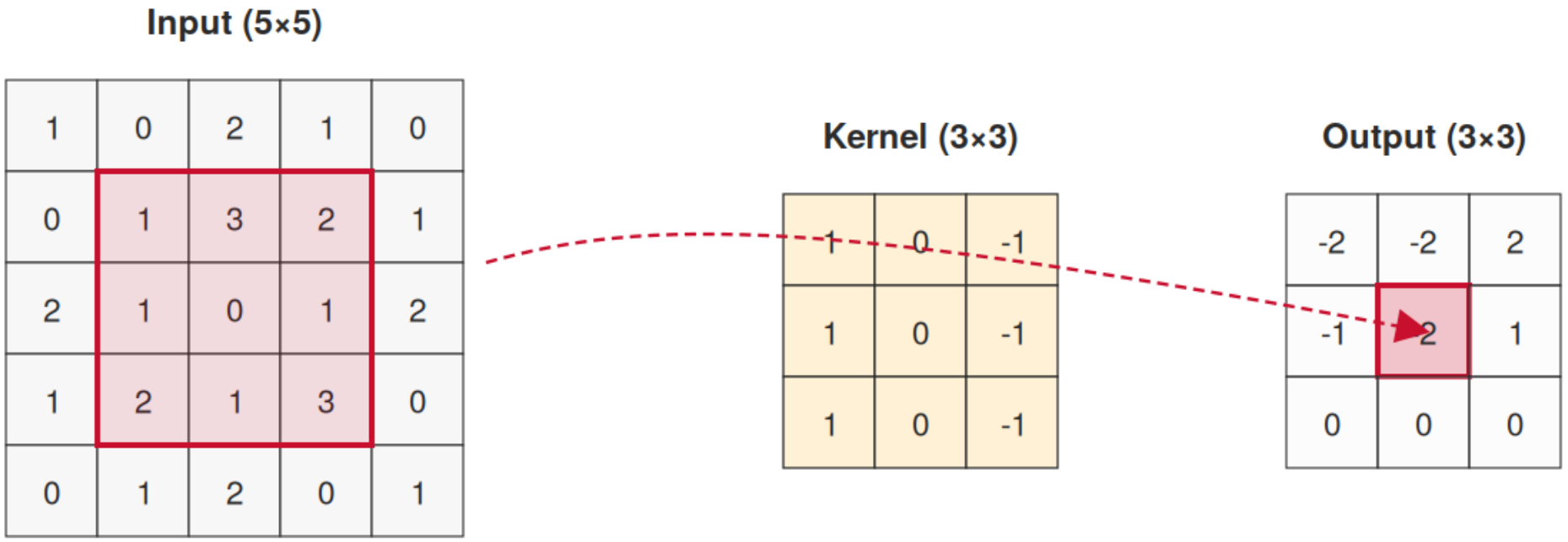
- Linear — composes with activations like FC
- Local — each output depends on a small input neighborhood
- Translation-equivariant — f(shift(x)) = shift(f(x))
- Parameter-shared — same detector applied everywhere

Claim: up to boundary effects, there is essentially one such operator — **convolution.**

2. Convolution from First Principles

Slide a Small Window, Compute a Weighted Sum

- $Y[i, j] = \Sigma \Sigma K[u, v] \cdot X[i+u, j+v]$



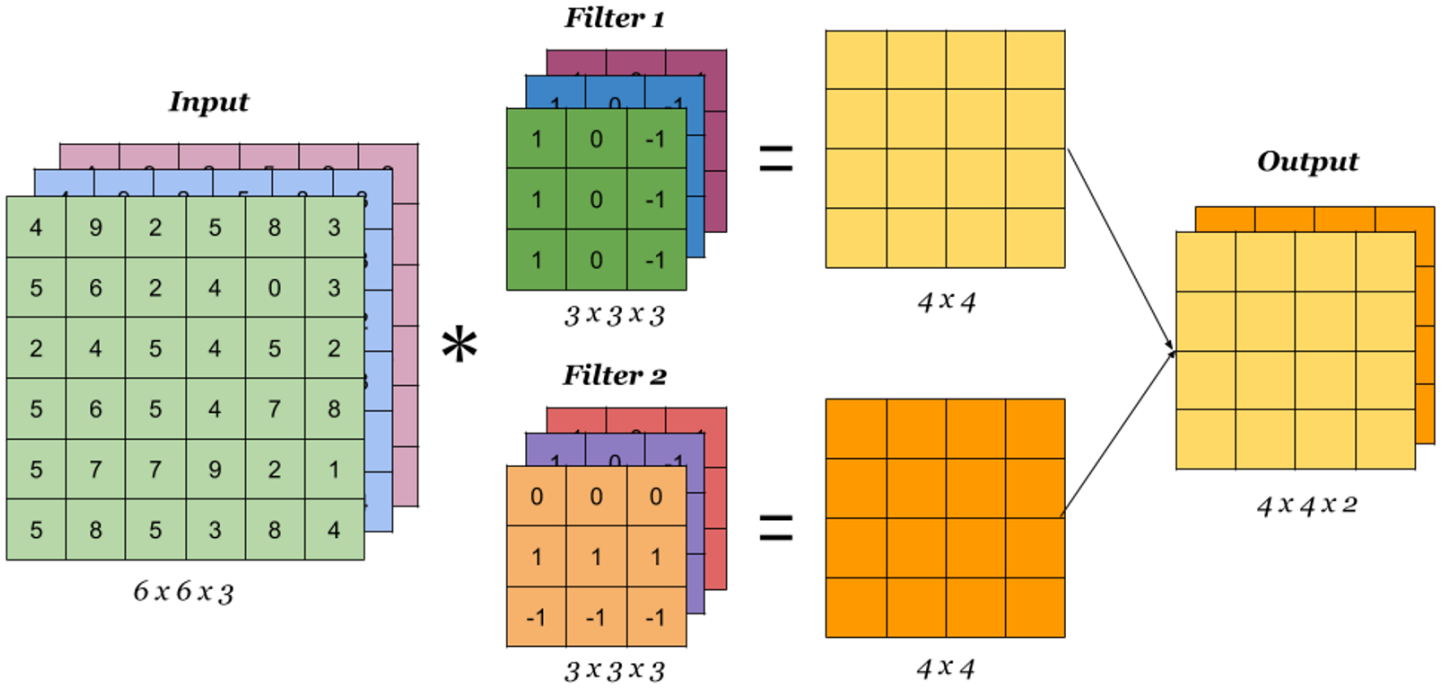
One kernel, reused at every spatial position — that is parameter sharing

Locality + Translation Equivariance ⇒ Convolution

- Theorem (informal): any linear, translation-equivariant operator on a grid is a convolution
- Proof is short (signal processing textbooks); we won't derive it here
- Convolution is not a clever trick — it is forced by our assumptions about images
- Architecture as encoded assumption about data — the single most important idea in this lecture

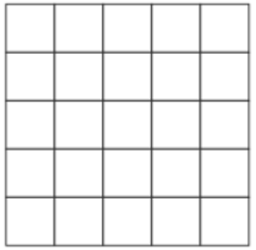
Channels: The Third Dimension

- Input: H × W × C_in (RGB → C_in = 3; hidden layers → 64, 256, 512…)
- Each filter: k × k × C_in — integrates across all input channels per spatial location
- C_out filters → output H' × W' × C_out



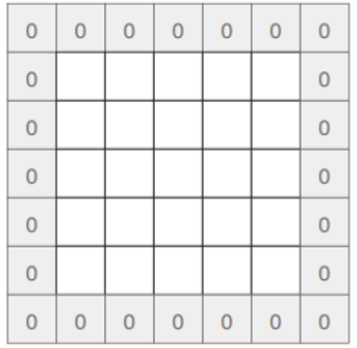
Stride, Padding & Output Shape

k=3, s=1, p=0



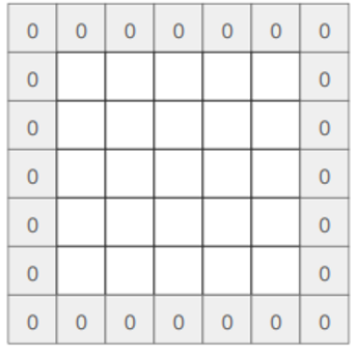
H' = (5 + 2·0 − 3) / 1 + 1 = 3

k=3, s=1, p=1



H' = (5 + 2·1 − 3) / 1 + 1 = 5

k=3, s=2, p=1



H' = (5 + 2·1 − 3) / 2 + 1 = 3

Quick check — 60 seconds with your neighbor

Input 32×32×3 → conv with 64 filters of 5×5, stride 1, padding 2

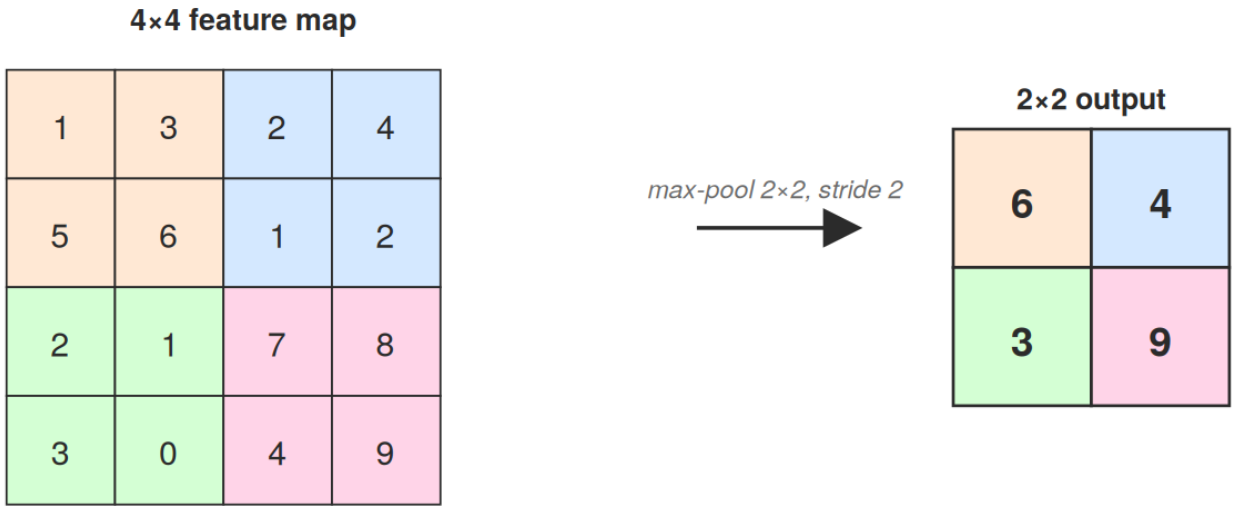
Output shape? Parameters? Multiply-adds?

Answer: 32×32×64 · 4,864 params · ~4.9M MACs

CNNs are compute-heavy, parameter-light — the opposite of MLPs

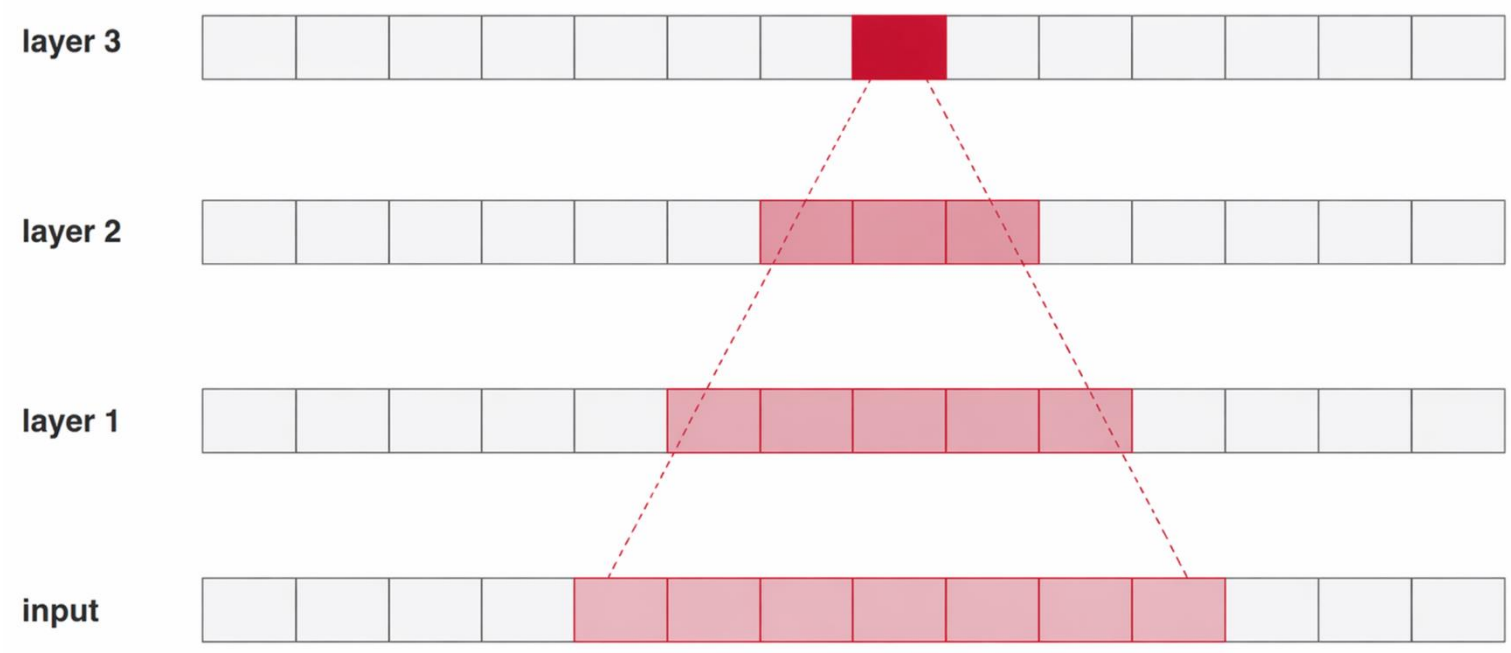
Pooling: Controlled Information Loss

- Max-pool 2 × 2, stride 2: keep the max in each window
- **Why?**
  - Downsample → larger receptive field for later layers
  - Small translation invariance
  - No parameters, cheap
- *Modern nets often replace it with strided convolutions*



Receptive Field: Depth Is How CNNs See Globally

- L stacked 3×3 convs → receptive field (2L+1)×(2L+1) — linear growth
- Add stride / pool → receptive field grows exponentially
- Early layers see edges; deep layers see whole objects — the compositional hierarchy



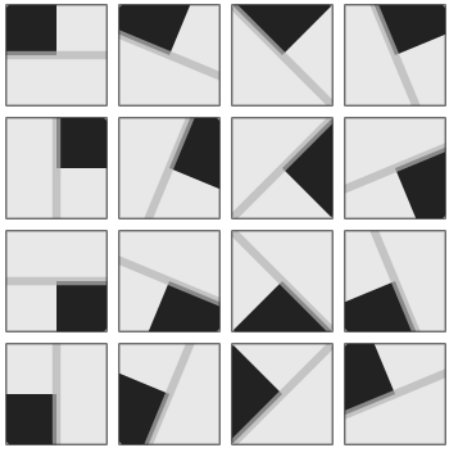
Receptive field grows with depth → deep layers see global structure

Why CNNs Train More Easily Than Deep MLPs

Issue	MLP	CNN
Parameters	150M+ (bloated)	~5K (shared, sparse)
Overfitting	needs heavy dropout/WD	architecture already regularizes
Vanishing gradients	long backward paths	short paths for early features
Data efficiency	relearns at every pixel	one shared filter, full image

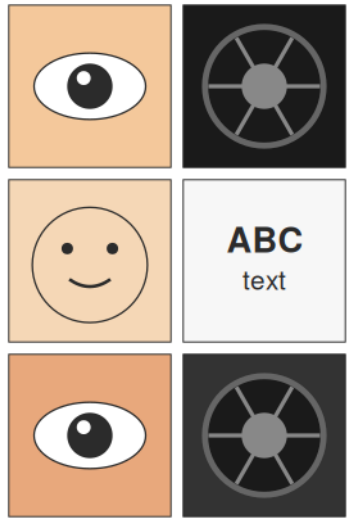
What Do the Filters Actually Learn?

Layer 1: oriented edges



Middle layers: textures & parts

Deep layers: objects

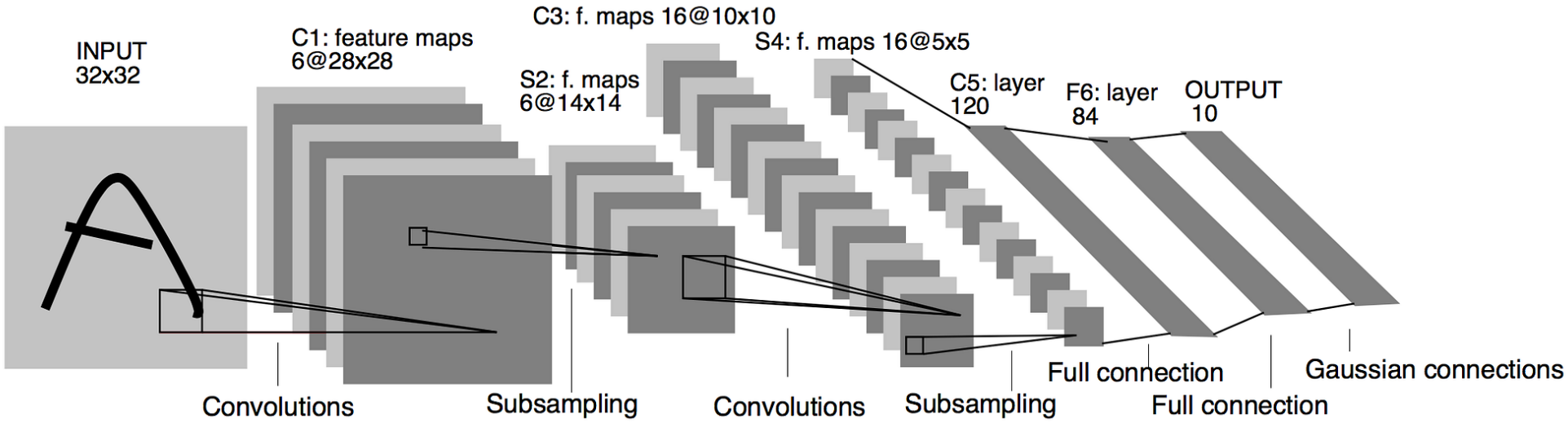


Layer-1 filters are oriented-edge detectors — nearly identical to V1 neurons in mammalian cortex

Five Architectures, Five Bottlenecks Solved

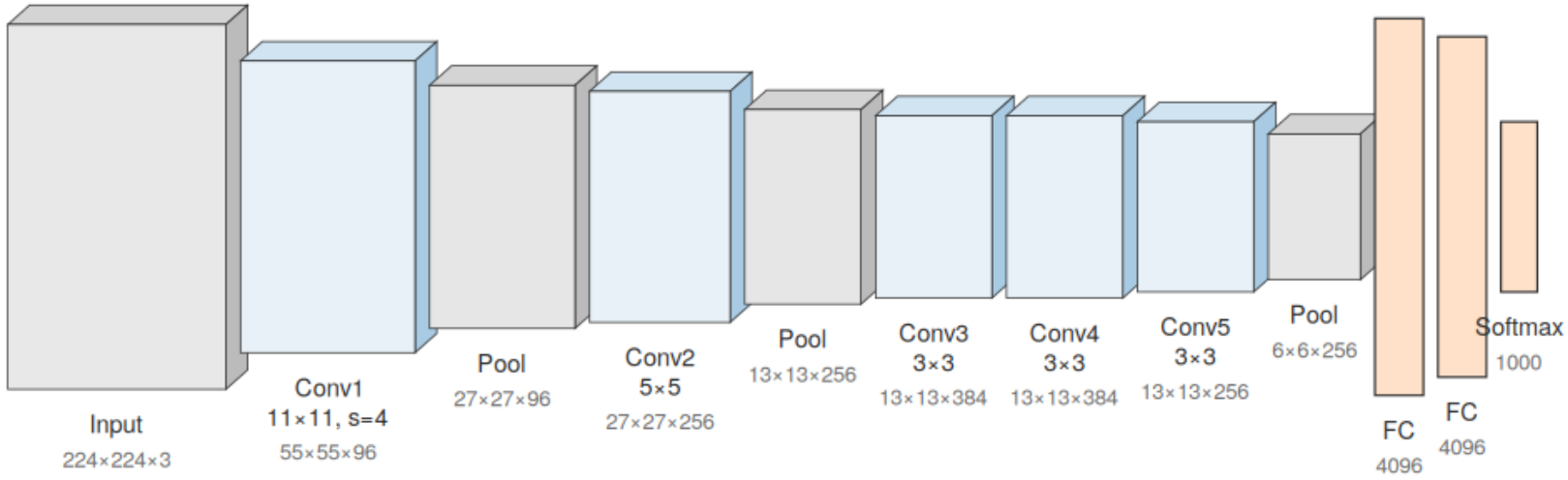
Year	Model	Key idea	Bottleneck solved
1998	LeNet-5	Conv + pool + FC	Hand-crafted features
2012	AlexNet	ReLU + GPU + dropout	Scaling to ImageNet
2014	VGG	Stacked 3×3 convs	Design regularity
2014	GoogLeNet	Multi-scale + 1×1 bottlenecks	Compute cost of depth
2015	ResNet	Skip connections F(x) + x	Optimization at depth

LeNet-5 (1998)



- LeCun et al.: digit recognition on bank checks and postal envelopes
- Full template already here: conv → pool → conv → pool → FC → FC. Sigmoid/tanh, ~60K params
- Dormant for 14 years: no ImageNet, no GPUs, community focused on SVMs/kernels
- Pattern: good ideas wait for their infrastructure

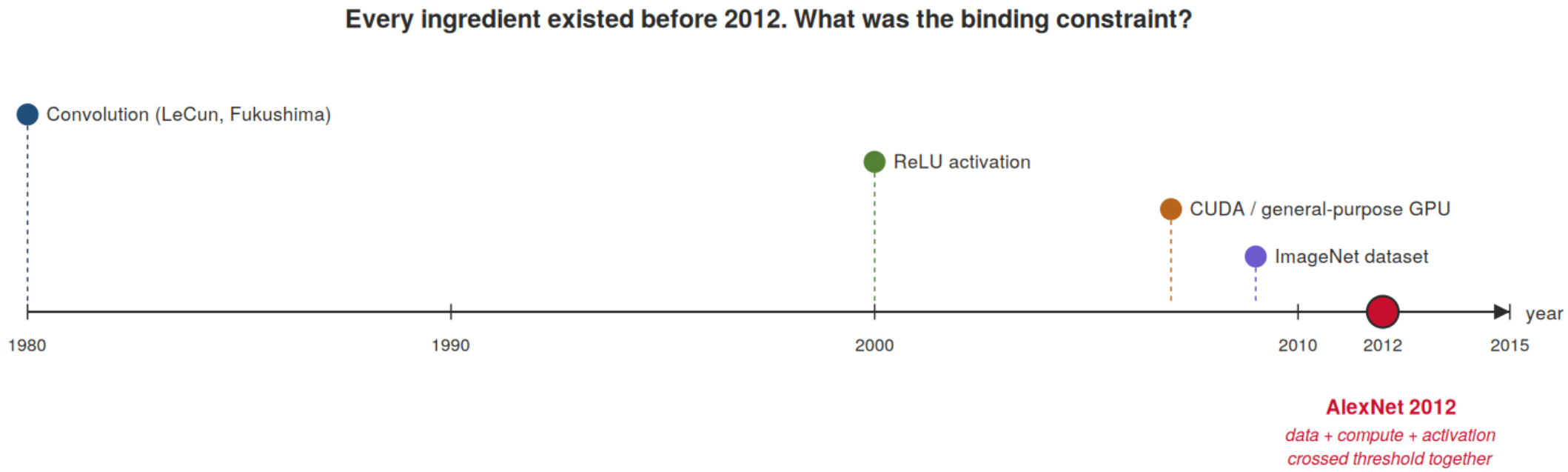
AlexNet (2012)



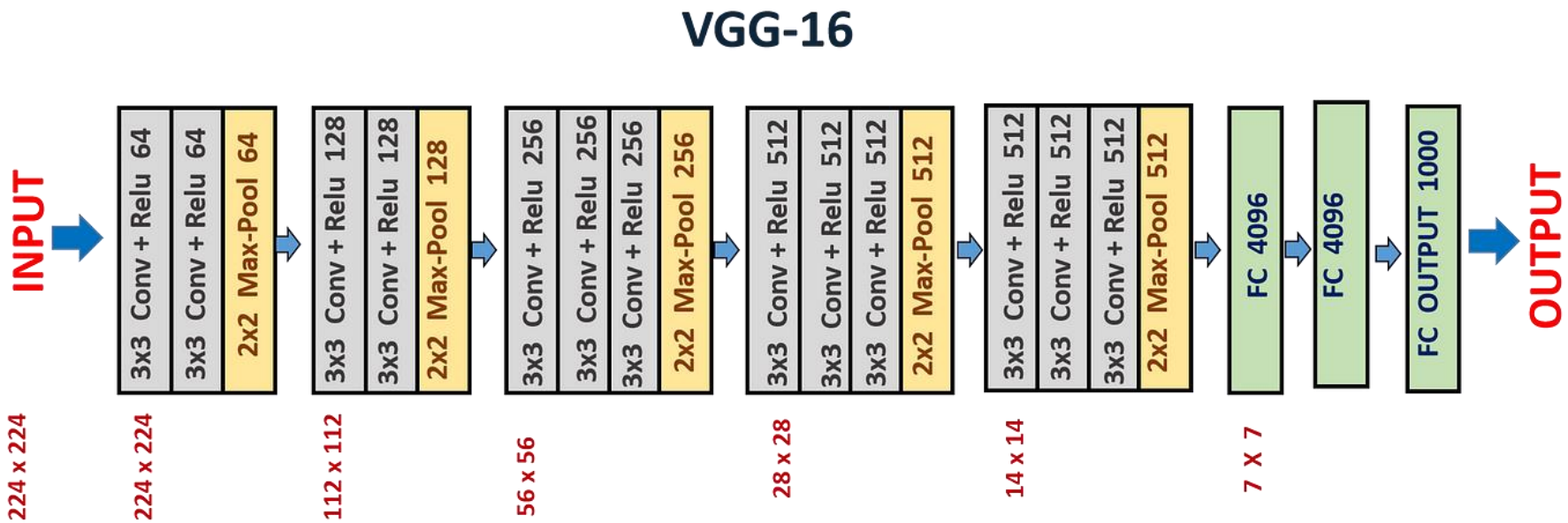
- ImageNet 2012: top-5 error 25% → 16% overnight
- Ingredients: ReLU (no saturation, trainable depth) + 2 GPUs (60M params on 1.2M images)
- Dropout + data augmentation for regularization (LRN mostly vestigial in hindsight)
- No single new idea — a co-tuned system. "Training is a systems problem" at architecture scale

Why Did AlexNet Work in 2012 and Not 2002?

- Data and compute crossed a threshold at roughly the same time.
- Transformers (2017), GPT-3 (2020), diffusion (2021) — all had antecedents that "didn't work" at smaller scale.
- The word to watch for in papers: scale.



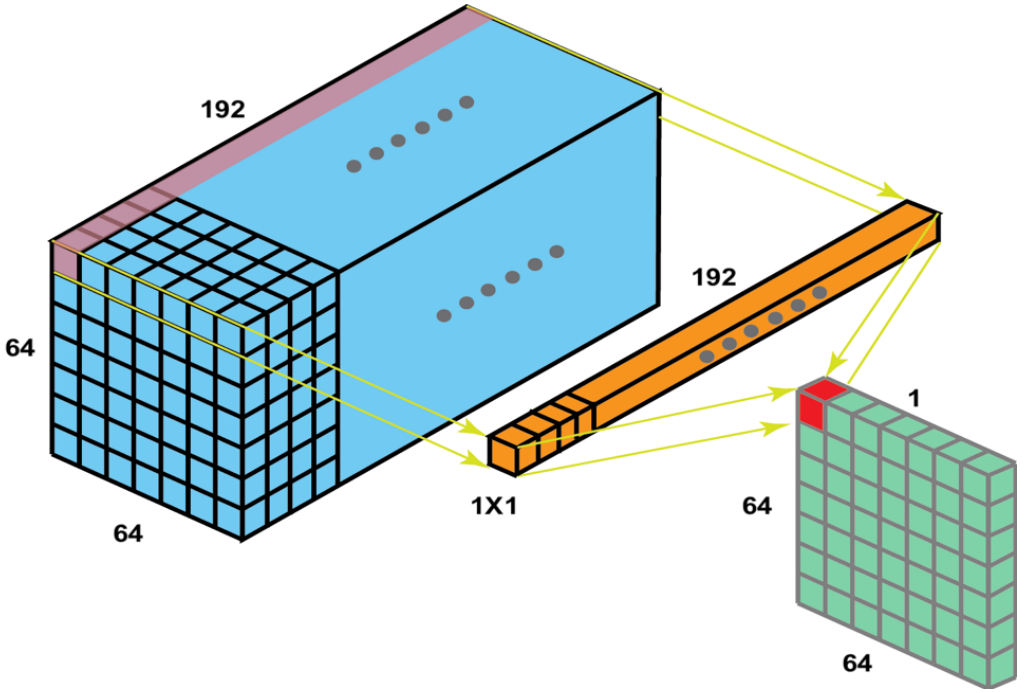
VGG (2014) — Depth Over Cleverness



- Principle: every conv is 3×3, every downsample is 2×2 max-pool, stack uniformly
- Two 3×3 > one 5×5: fewer params, extra nonlinearity, same receptive field. After VGG, 3×3 is the default for a decade
- *VGG-16 has 138M params — most in final FC layers. That bloat is the next architecture's target*

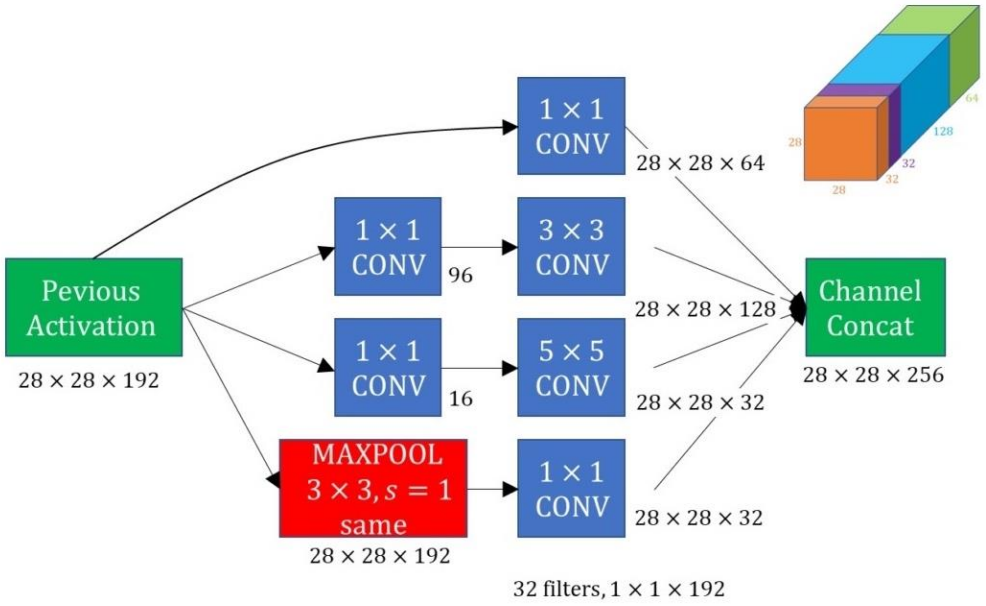
Aside: What Does a 1×1 Convolution Even Do?

- One spatial position, all channels → linear projection across channels (an MLP per pixel)
- **Unlocks: bottleneck designs + decoupled channel-mixing from spatial-mixing**
- *Reappears in depthwise separable convs, Inception, and essentially in self-attention*



GoogLeNet / Inception (2014) — Stop Choosing Kernel Sizes

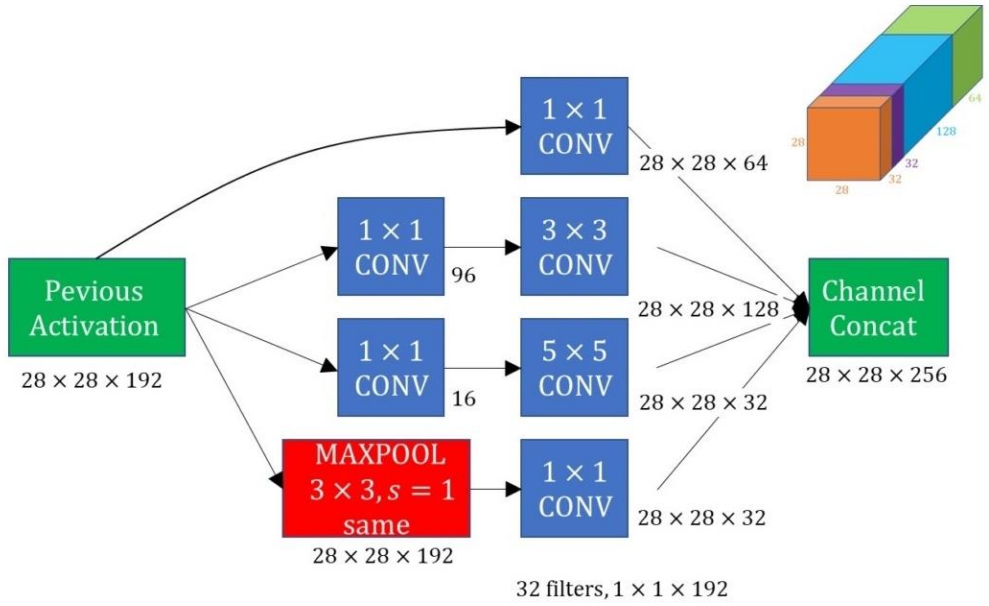
- **Motivation: what kernel size should we use?**
- VGG says 3×3 always. But different features live at different scales — a whole face might need 5×5, an edge needs 3×3
- **Inception's answer: stop choosing. Run 1×1, 3×3, 5×5, and pool in parallel. Concatenate outputs.**
- Let the next layer's weights decide how much to weight each scale



The 1×1 Bottleneck: How Inception Stays Cheap

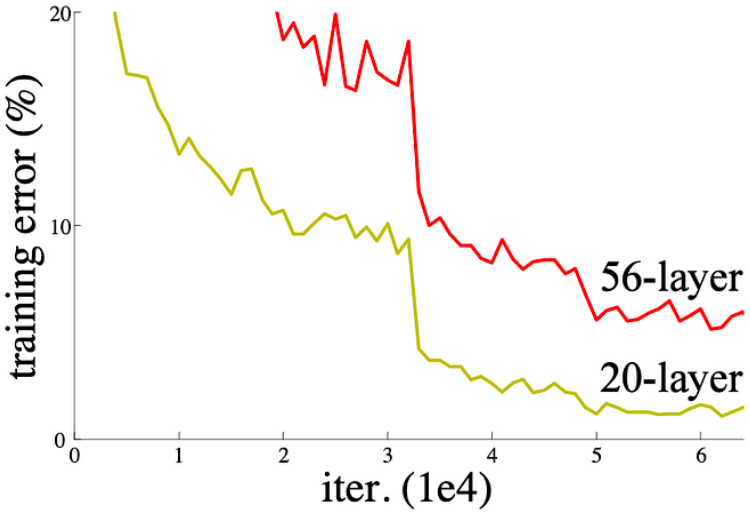
Problem: stacking 5×5 convs across many channels is expensive. Inception solves this with 1×1 bottlenecks.

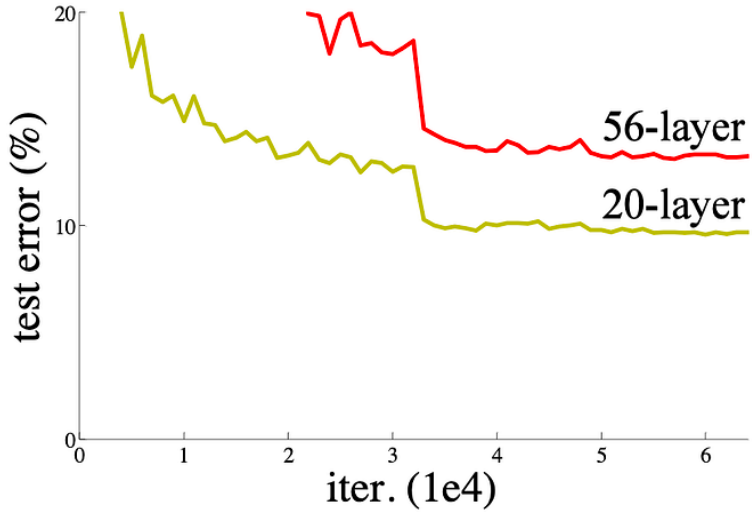
- Before expensive 3×3/5×5: insert 1×1 conv to reduce channels
- Example:
  - 256 ch → 5×5 → 128 ch costs ~820K params.
  - **With 1×1 (256→32) first: ~110K params. 7× cheaper.**
- **12× fewer params than VGG at similar accuracy**
- *Compute efficiency as a first-class design goal*



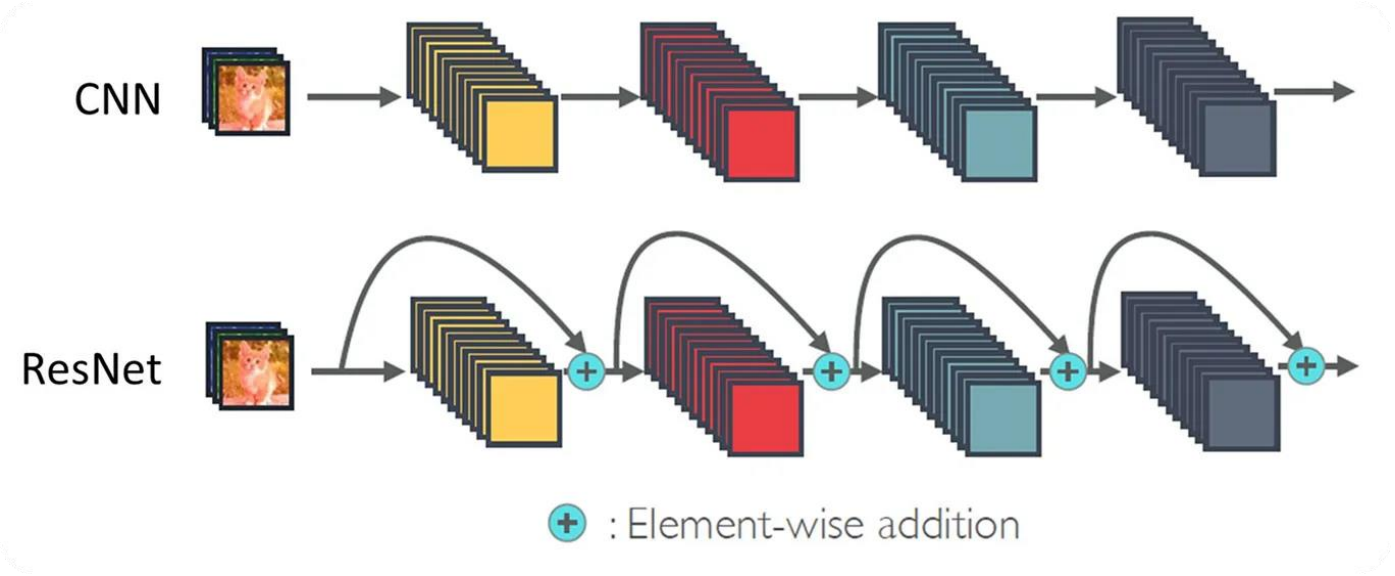
The Depth Problem — Why Didn't Everyone Just Go Deeper?

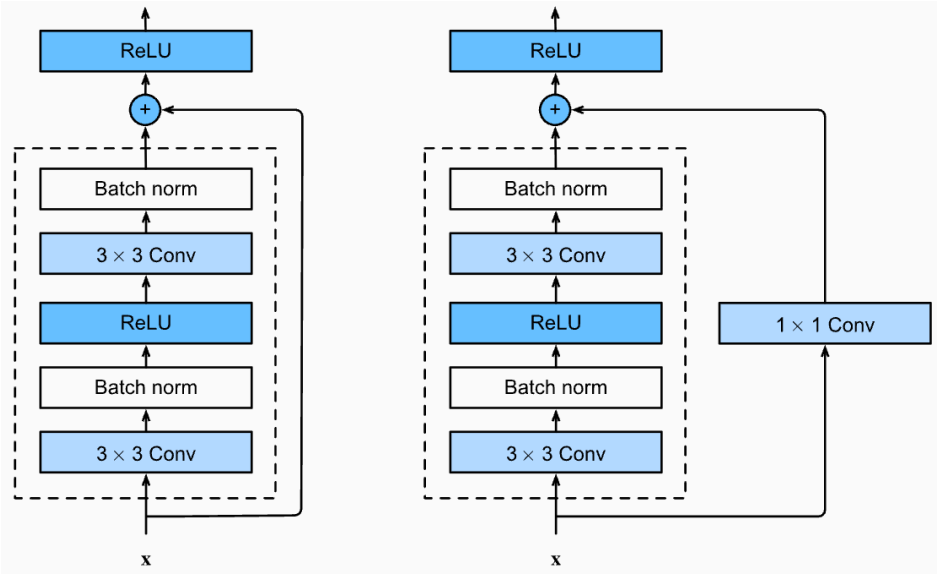
- **56-layer plain CNN has HIGHER training error than 20-layer — not overfitting, an optimization failure**
- Even with BN and good init. Something in plain stacking actively prevents deep optimization
- *L16's toolkit was necessary but not sufficient*





ResNet (2015) — If Identity Works, Make It Easy to Learn

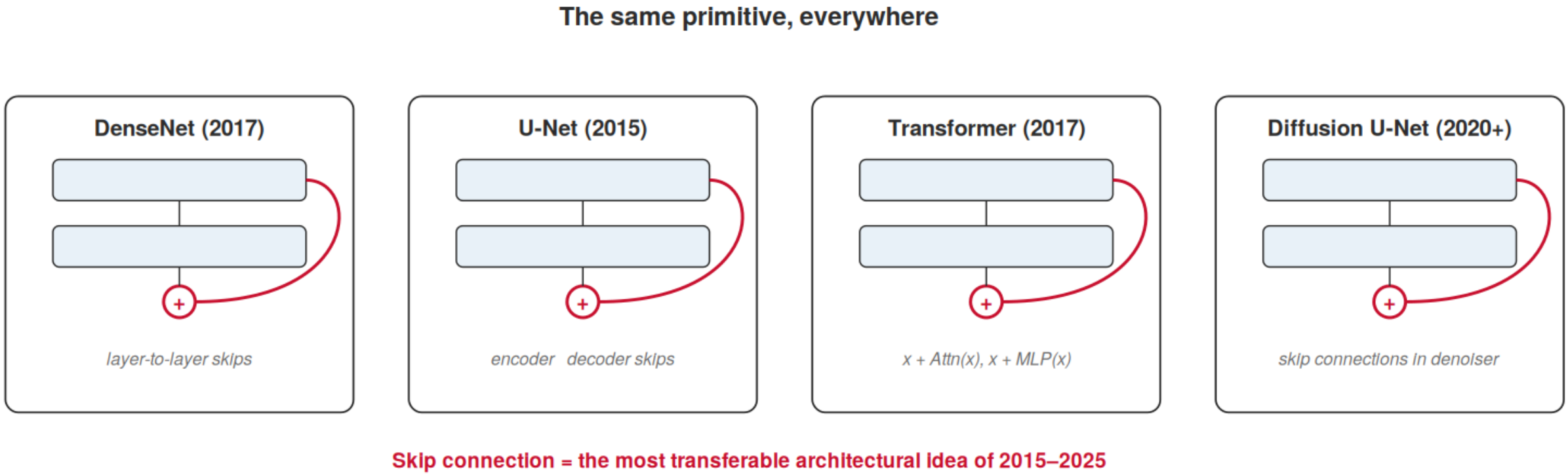




- Reparameterize: y = F(x) + x. Identity becomes the easy default (F ≈ 0)
- Gradient highway: ∂y/∂x has a +1 term through the skip — vanishing gradients structurally suppressed
- **ResNet-152 trains stably, wins ImageNet 2015. Three years after AlexNet: 8 → 152 layers**

4. Synthesis & Foreshadowing

Skip Connections Outlived CNNs



Without residual connections, modern Transformers and diffusion models don't train at scale.

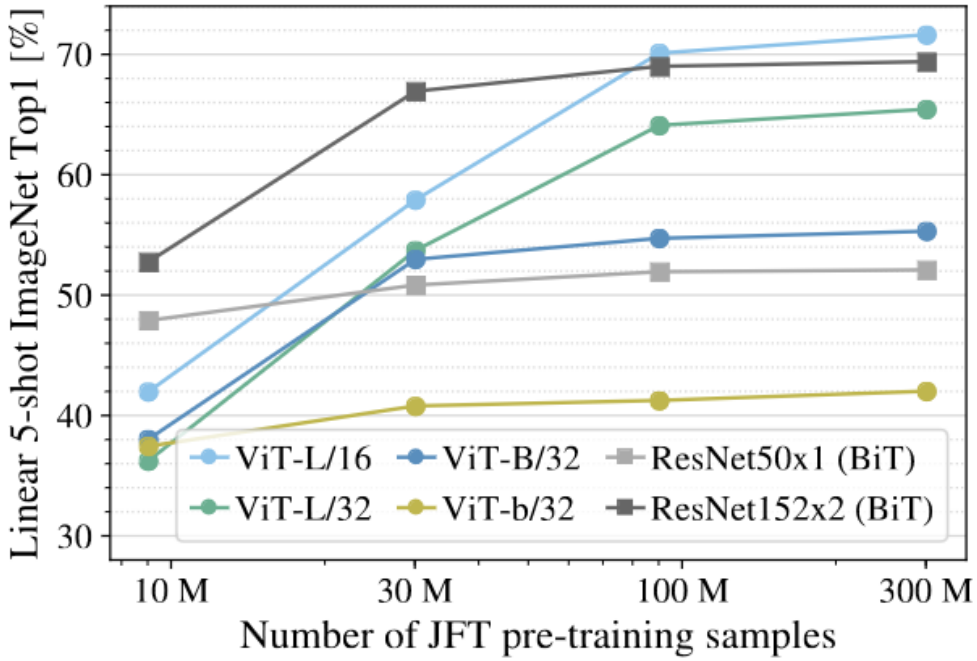
What Was the Architecture Community Actually Doing?

Transition	Problem being solved
LeNet → AlexNet	**Scale (data, compute, activation)**
AlexNet → VGG	**Design regularity**
VGG → Inception	**Compute efficiency at a given depth**
Inception → ResNet	**Optimization at extreme depth**

Meta-lesson: architecture research is rarely about new primitives.
It's about identifying which constraint is currently binding, and relaxing it.

When Is a CNN the Wrong Tool?

- Long-range dependencies
  - Conv receptive field grows linearly with depth; attention gets there in one operation
- Non-grid data
  - Graphs, point clouds, sets — no translation to be equivariant to
- Very large data regimes
  - Inductive biases can be learned rather than imposed



Summary

1. Convolution is forced, not clever.
   – Linearity + locality + translation equivariance uniquely determine it.
2. Weight sharing = regularizer + statistical efficiency.
   – CNNs train on less data than equivalent MLPs.
3. Every major architecture solved a specific bottleneck.
   – Depth, regularity, compute, optimization — read architectures as responses to binding constraints.
4. Skip connections outlived CNNs.
   – You will see them every week for the rest of the course.

Next Lecture

- **Modern CNN Architectures & Efficient Design**
  - ResNet variants, DenseNet, MobileNet / EfficientNet, depthwise separable convs, NAS