上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Lecture 18: Modern CNN Architectures

Tao Huang

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

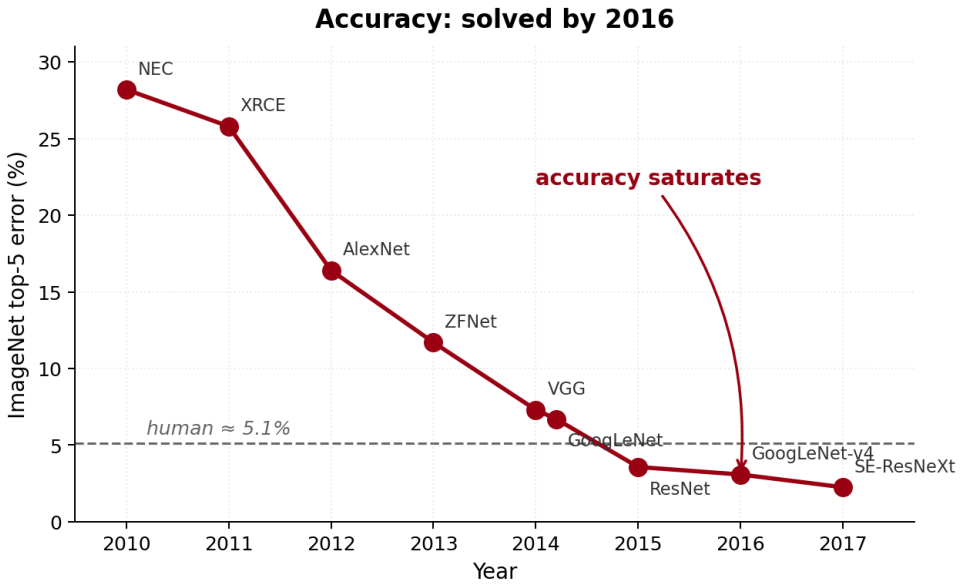
https://taohuang.info/cs3317

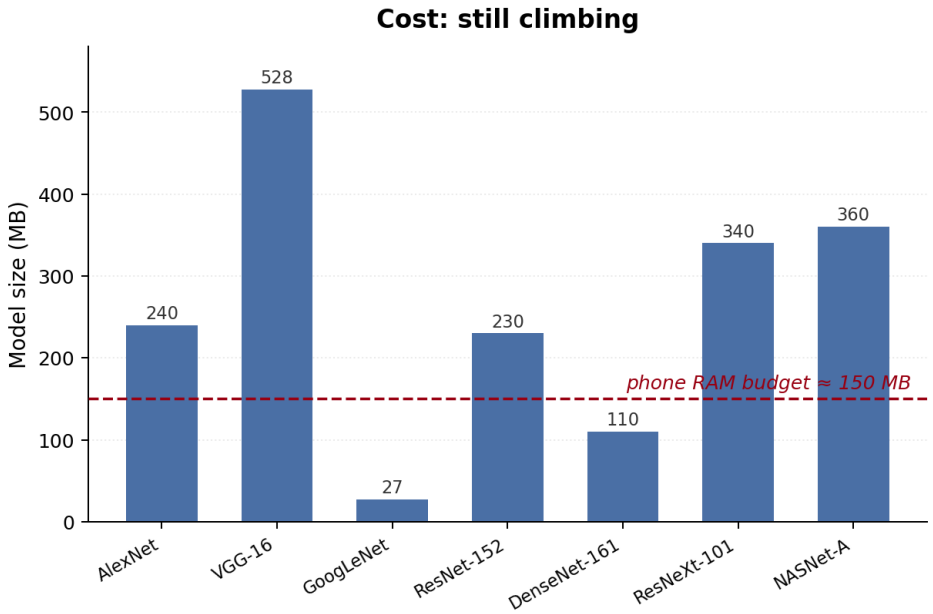
https://oc.sjtu.edu.cn/courses/89538

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

Where We Left Off

- L17 ended at ResNet-152: depth no longer the binding constraint.
- ImageNet top-5 error: 25% (2012) → 3.6% (2015). Below human by 2016.
- *So… are CNNs solved? If yes, why are we here for another 45 minutes?*





Plot twist: by 2016, the field's question changed. Accurate => Accurate & Efficient

A New Deployment Reality

Where models lived:

2012	Two GPUs in a basement
2015	A datacenter
2017+	**Your phone, your car, a $5 microcontroller**

ResNet won ImageNet. Accuracy is “good enough” for many tasks.

But: ResNet-152 = 11 GFLOPs/image. Phone budget = ~500 MFLOPs at 30 FPS.

Can we keep the accuracy and shed 95% of the compute?

Objectives

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

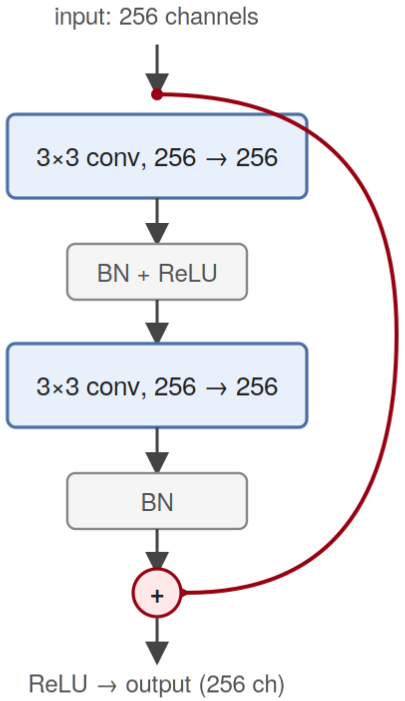
- Analyze a ResNet bottleneck block and explain why its three-conv structure is cheaper than two 3×3 convs at the same width.
- Decompose a standard convolution into depthwise + pointwise components, and derive the FLOP reduction factor algebraically.
- Evaluate the accuracy–latency–memory trade-offs of MobileNet, EfficientNet, and ConvNeXt on a given hardware budget.
- Critique Neural Architecture Search: when it's a breakthrough, when it's expensive curve-fitting.
- Defend or attack the claim: "After 2020, all CNN improvements are just borrowed Transformer ideas."

1. Going Deeper Properly — Beyond Vanilla ResNet

The Bottleneck Block — ResNet's Real Innovation

- ResNet-18/34 use the basic block: two 3 × 3 convs.
- ResNet-50/101/152 use the **bottleneck block**: 1 × 1 → 3 × 3 → 1 × 1. *Why?*

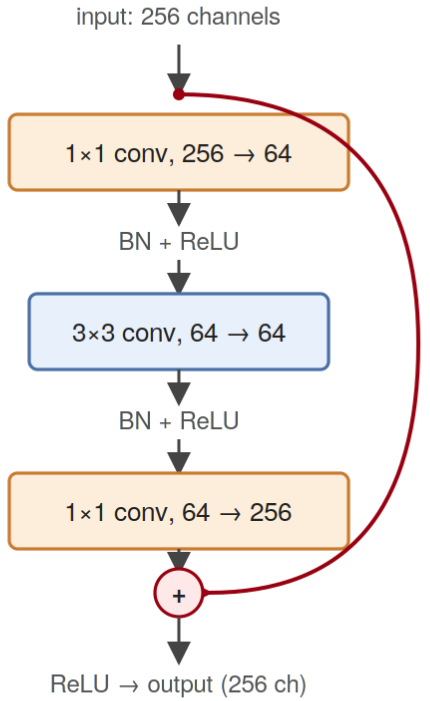
Basic block (ResNet-18/34)



~1.18M params

9 · 256 · 256 · 2 = 1,179,648

Bottleneck block (ResNet-50/101/152)



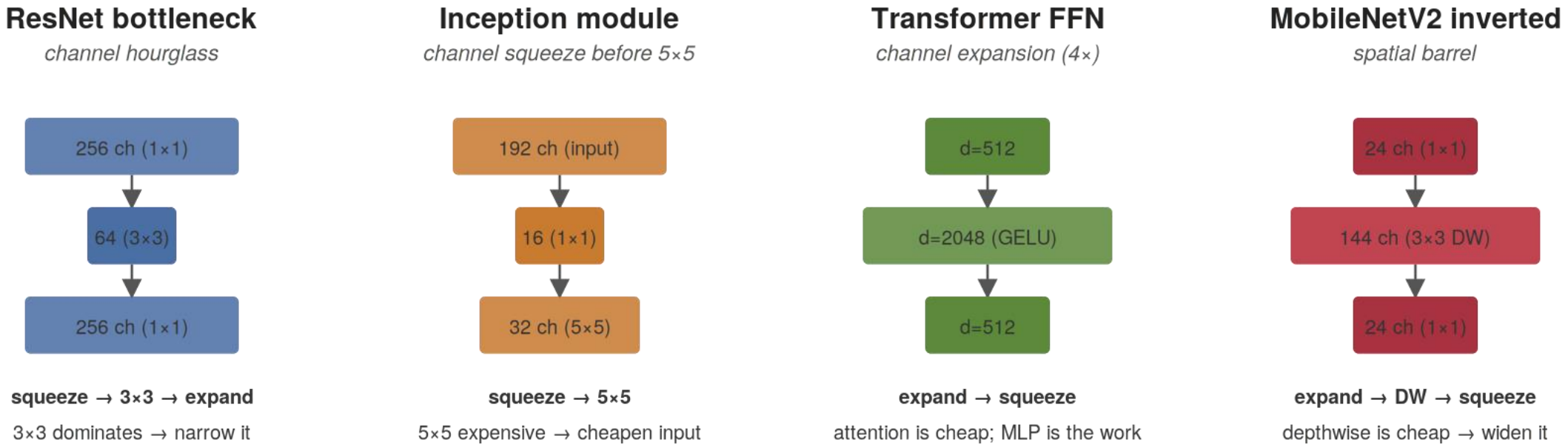
~70K params

256·64 + 9·64·64 + 64·256 ≈ 70,000

Same I/O shape, same expressive role, ~17× cheaper.

Squeeze → Process → Expand: A Recurring Pattern

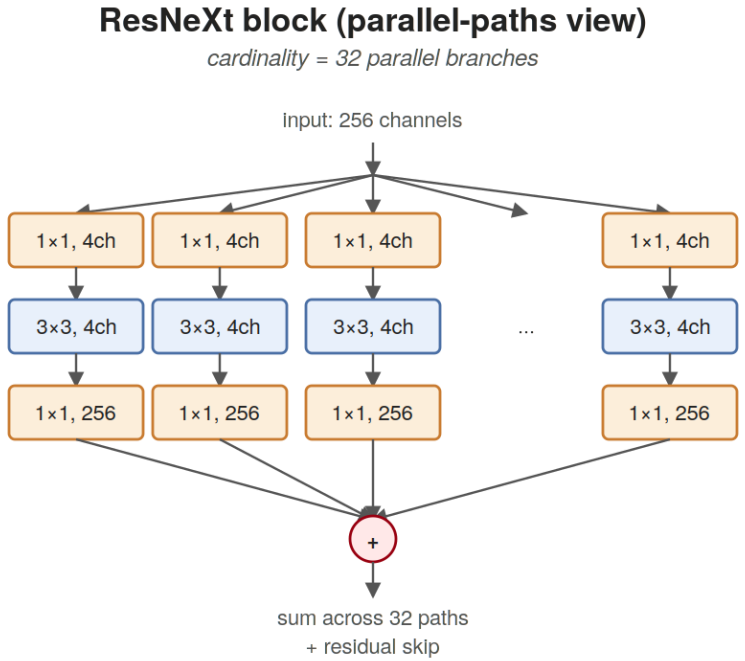
- The bottleneck is a recurring design pattern. You will see it everywhere:

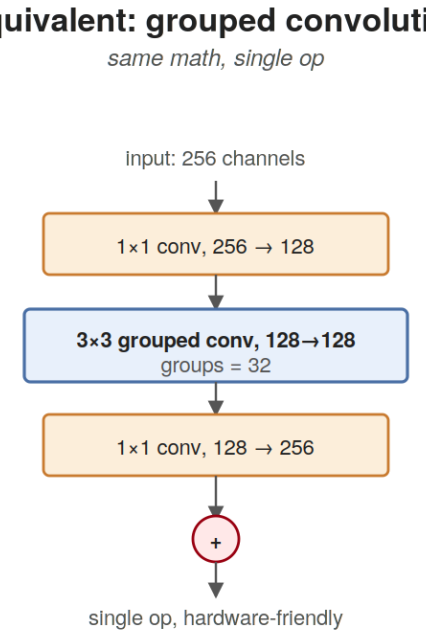


Why does the same primitive recur in inverse forms? Save that question — we will answer it on Slide 16.

ResNeXt — Cardinality, a Third Scaling Axis

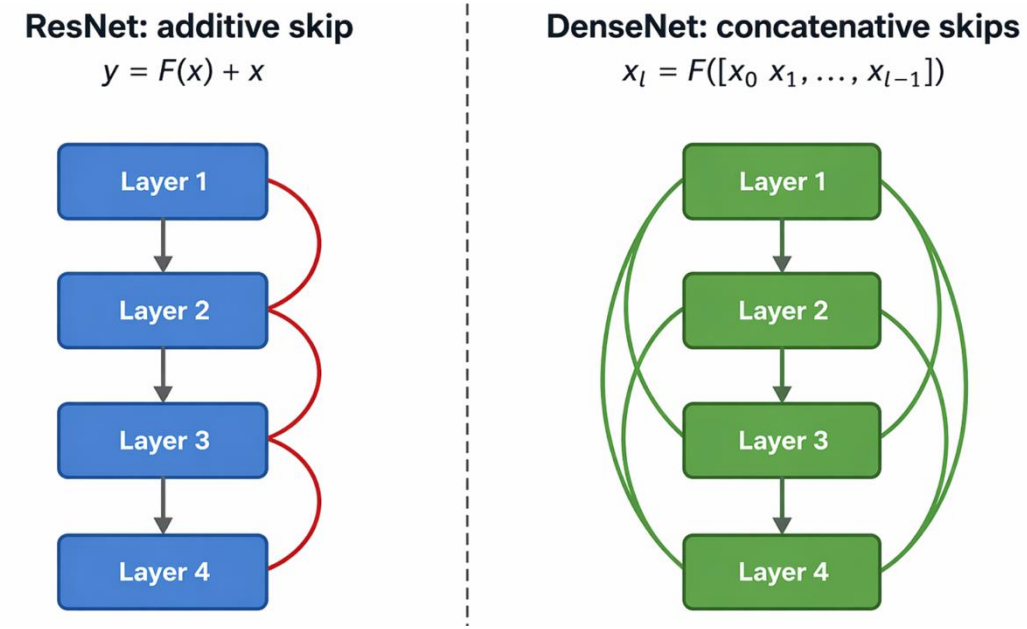
- Two knobs to scale a network: **deeper** (ResNet) and **wider** (Wide ResNet).
- ResNeXt (2017) adds a third: **cardinality** = number of parallel paths in a block.
- Same params and FLOPs as ResNet-50. **~1% better top-1.**
- **Why this matters:** *cardinality = grouped convolution = first step toward depthwise (slide 13). Foreshadowing.*





DenseNet (2017) — Maximum Information Reuse

- ResNet: y = F(x) + x (additive skip)
- DenseNet: x_l = F([x_0, x_1, …, x_{l-1}]) — concatenative skip from every earlier layer



Win: feature reuse + implicit deep supervision → fewer params.

Catch: concatenation grows feature maps — memory-hungry.

Interactive Check

A ResNet-50 bottleneck has the topology: **1×1 (256→64), 3×3 (64→64), 1×1 (64→256).**
Q1: What is the parameter count? (Ignore bias.)
Q2: If you doubled the bottleneck width (64 → 128) keeping I/O at 256, what happens to params? FLOPs?

Answers:
Q1: 256×64 + 9×64×64 + 64×256 = 16,384 + 36,864 + 16,384 = **69,632 params**
Q2: Params and FLOPs both grow by ~3.5×. **The 3×3 middle conv dominates — its cost scales as C_mid^2.**

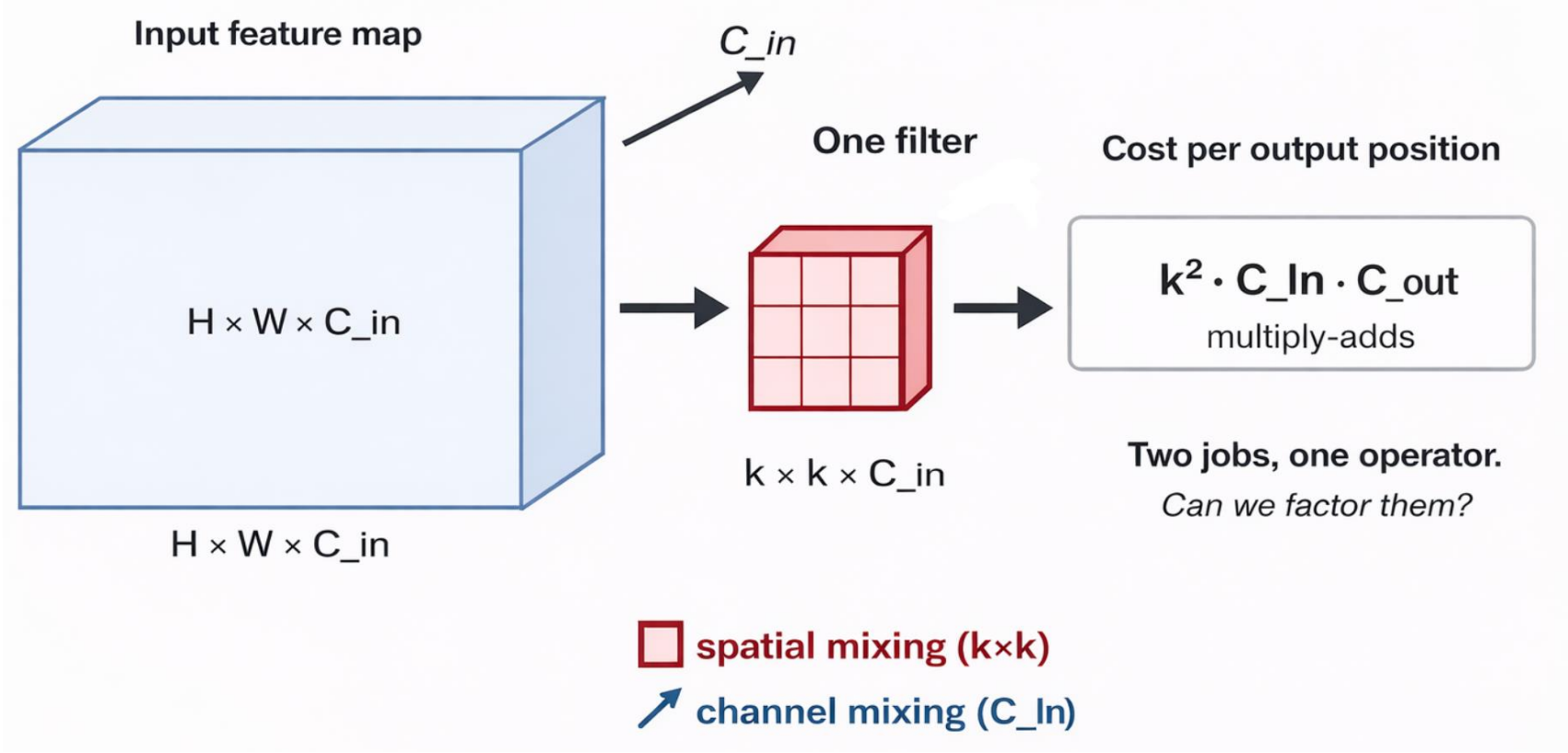
Takeaway: the middle width is the key knob. EfficientNet (slide 20) will turn exactly this lever.

2. The Efficiency Revolution — Rethinking the Convolution

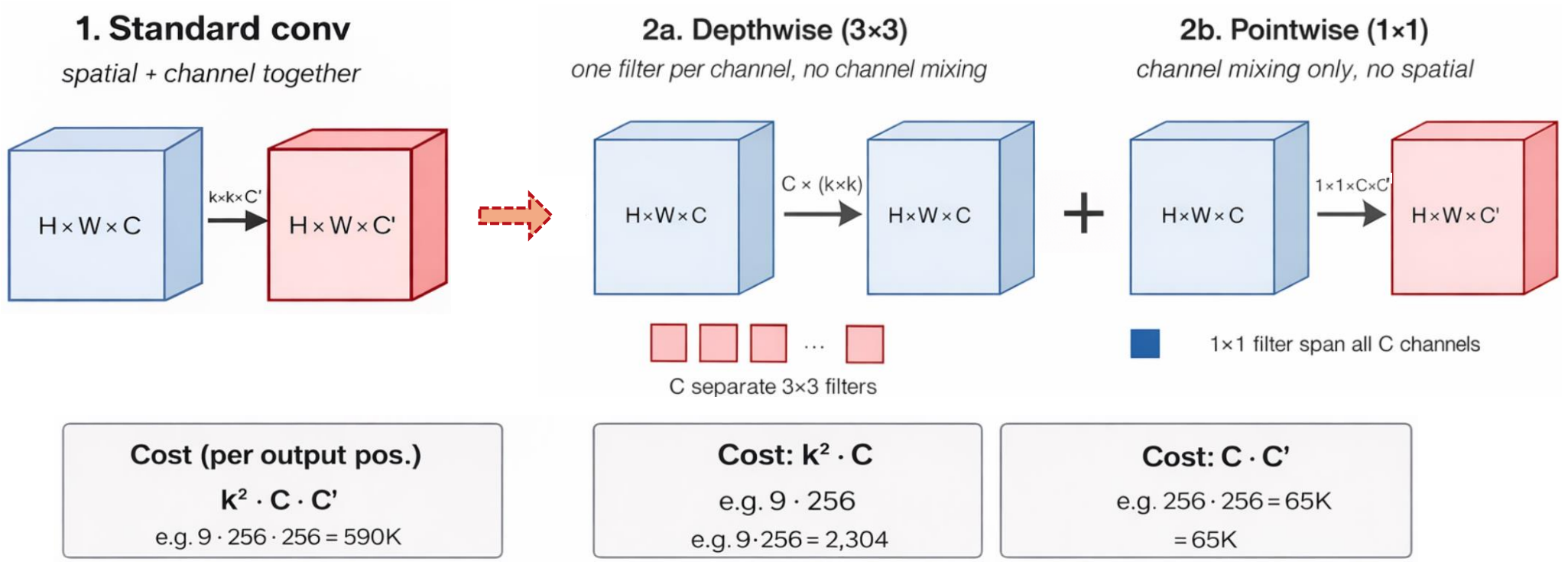
Standard Conv: Two Jobs in One Operator

A standard k × k conv layer with C_in → C_out channels does **both**:

- **Spatial mixing:** combines nearby pixels (the k × k part)
- **Channel mixing:** combines all input channels at every position (the C_in part)



Depthwise Separable Conv — Two Cheap Ops Win



- Step 1: apply one k × k filter per input channel (no cross-channel mixing).
- Step 2: 1 × 1 pointwise mixes channels at every spatial position (no spatial mixing).
- For k=3, C_out=256: ~9× cheaper.

A Familiar Pattern

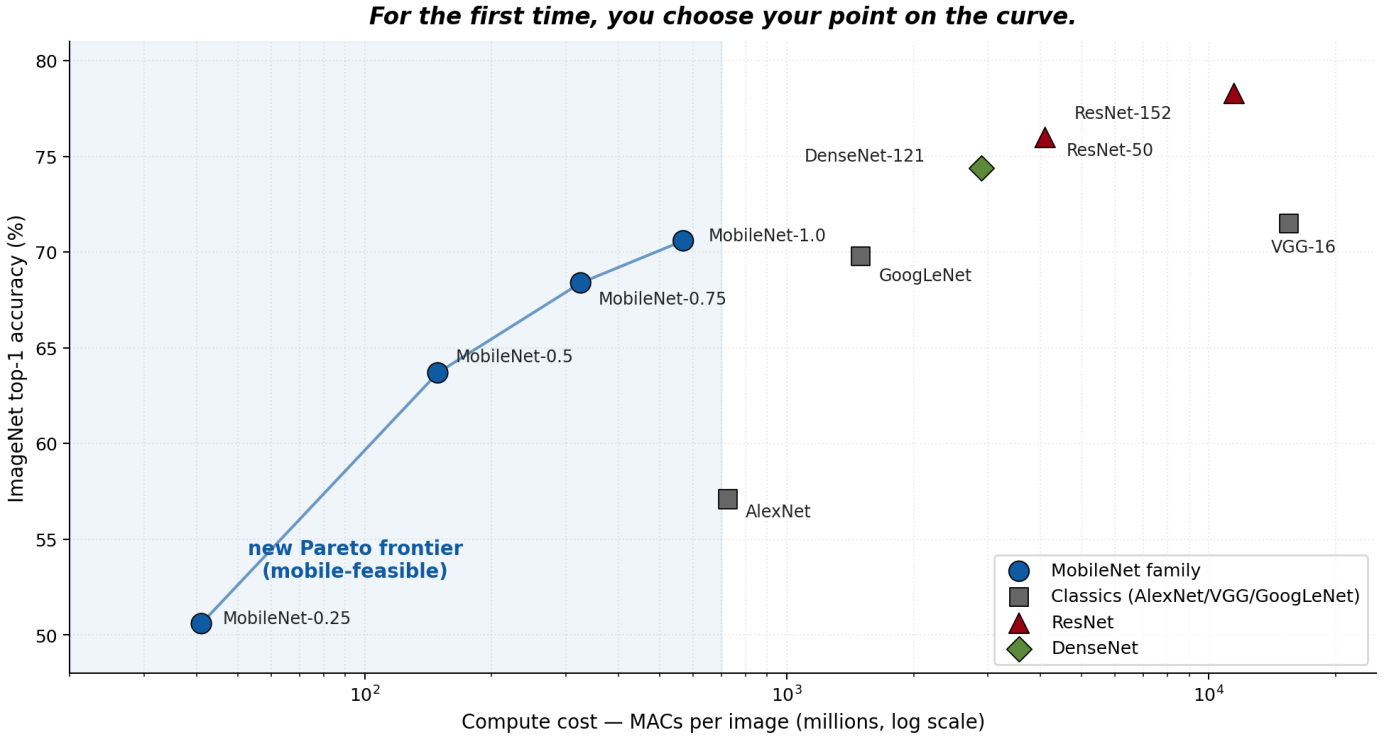
Where have we seen decoupling before?

Operation	What it factors
1×1 conv (L19)	channel-mixing without spatial-mixing
Depthwise conv	spatial-mixing without channel-mixing
Self-attention (L23)	content-based token-mixing without position-mixing
MLP block in Transformers	channel-mixing only

The recurring idea: when one operator does two jobs, factor it. The factored version is usually cheaper and easier to interpret.
Lecture 17 hinted: "Convolution is forced by our assumptions about images." **Modern architectures ask: which assumptions can we relax, and what do we save?**

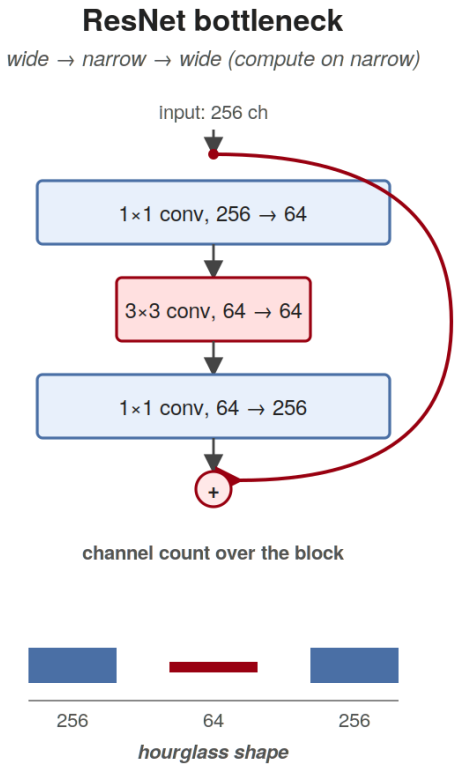
MobileNetV1 (2017) — The First Mobile-First CNN

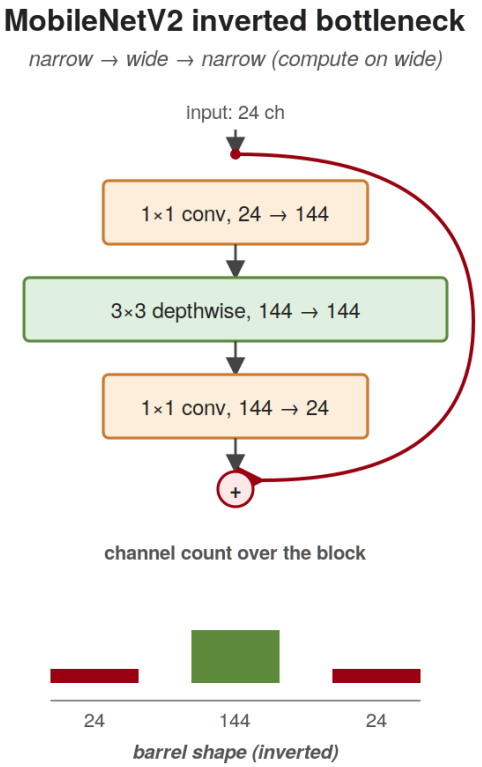
- Replace every standard 3×3 conv in a VGG-style stack with a **depthwise + pointwise** pair.
- Add two hyperparameters: **width multiplier α** (shrinks all channel counts) and **resolution multiplier ρ** (shrinks H, W).
- Result: 70.6% top-1 with ~4.2M params, ~570M MACs. **~9× cheaper than VGG-16, comparable accuracy.**



MobileNetV2 (2018) — The Inverted Bottleneck

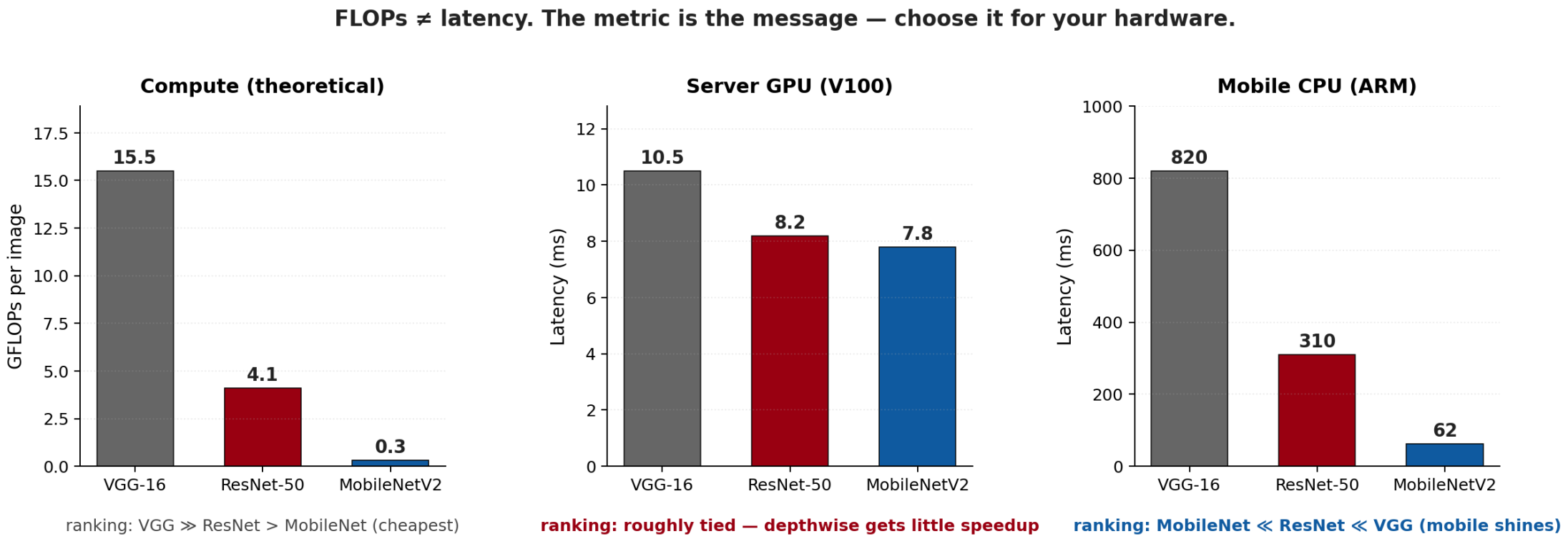
- Why this works: (1) depthwise is already cheap — wider channels don't blow up cost; (2) info lives better in higher dim, ReLU collapses narrow channels; (3) skip on narrow ends → memory-efficient.
- Returning to slide 7: bottleneck direction is set by which operator dominates cost.





A Brief Word on Hardware

- MobileNetV1 has 9× fewer FLOPs than VGG-16. **It is not 9× faster on a GPU.**
- Why? Depthwise convs have **low arithmetic intensity** (FLOPs per memory access). GPUs are memory-bandwidth-limited on these.



3. Scaling Laws and the Search for the Right Network

How Should We Scale a CNN? Three Knobs.

Depth d

more layers

Exemplar:

VGG

Width w

more channels per layer

Exemplar:

Wide ResNet

Resolution r

larger input image

Exemplar:

ImageNet @ 384×384

Conventional wisdom (pre-2019): pick one knob and crank it.

The problem: each axis has diminishing returns alone. Doubling depth gives ~0.3% past a point.

Question: *where do you spend your compute budget?*

EfficientNet (2019) — Compound Scaling

- Claim: depth, width, and resolution should be scaled together, in fixed ratios.
- Given compute budget ϕ: $d = \alpha^{\phi}$, $w = \beta^{\phi}$, $r = \gamma^{\phi}$, subject to $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$

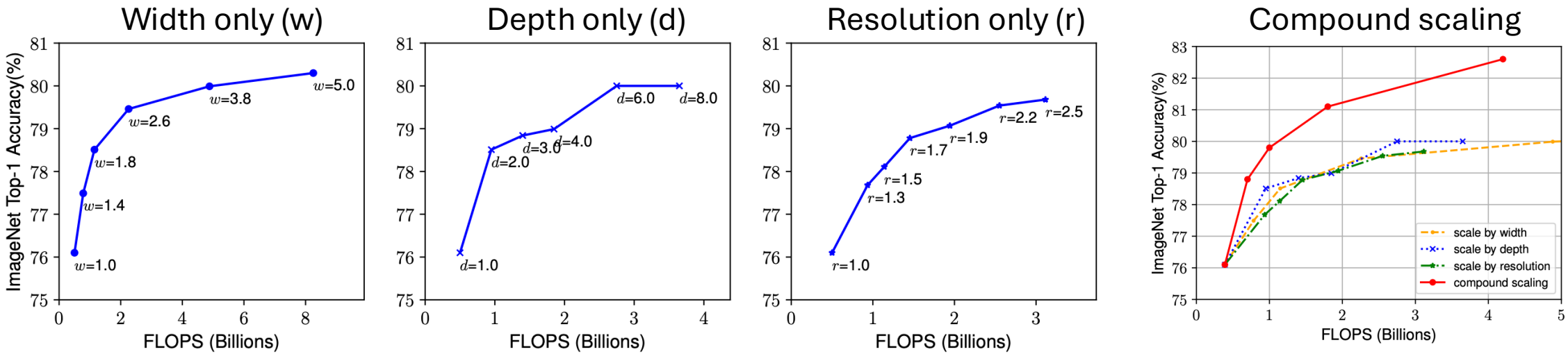


Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.**

EfficientNet-B7 hits 84.3% ImageNet top-1 with 8.4× fewer params than the previous SOTA at the same accuracy.

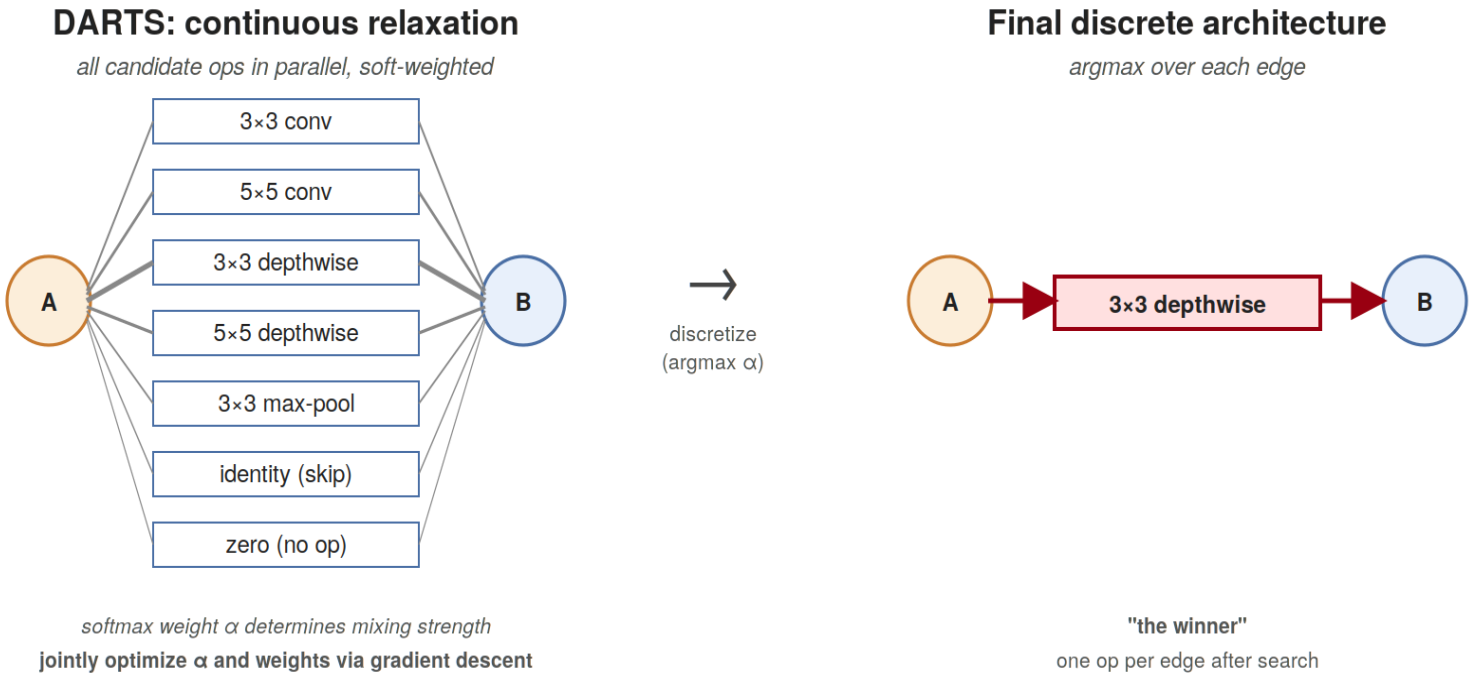
Be Careful What You Believe

Was EfficientNet "the right way to scale" — or just lucky?

- RegNet (2020): EfficientNet's compound coefficients are not optimal — they were over-fit to the B0 baseline.
- ResNet Strikes Back (2021): plain ResNet-50 + good training recipe = 80.4%, beating EfficientNet-B4.
- Most of the "architectural" gains 2017–2021 were actually training-recipe gains.

Neural Architecture Search

- **Idea:** define a search space of possible architectures; use an outer-loop optimizer (RL, evolution, gradient) to find the best one.
- **NASNet (2018):** RL controller, **800 GPUs for 28 days**. Beat human designs by 0.5%.
- **DARTS (2019):** continuous relaxation → train arch and weights jointly with gradient descent. **One GPU, one day.** A practical revolution.



NAS — A Critical Look (2026 hindsight)

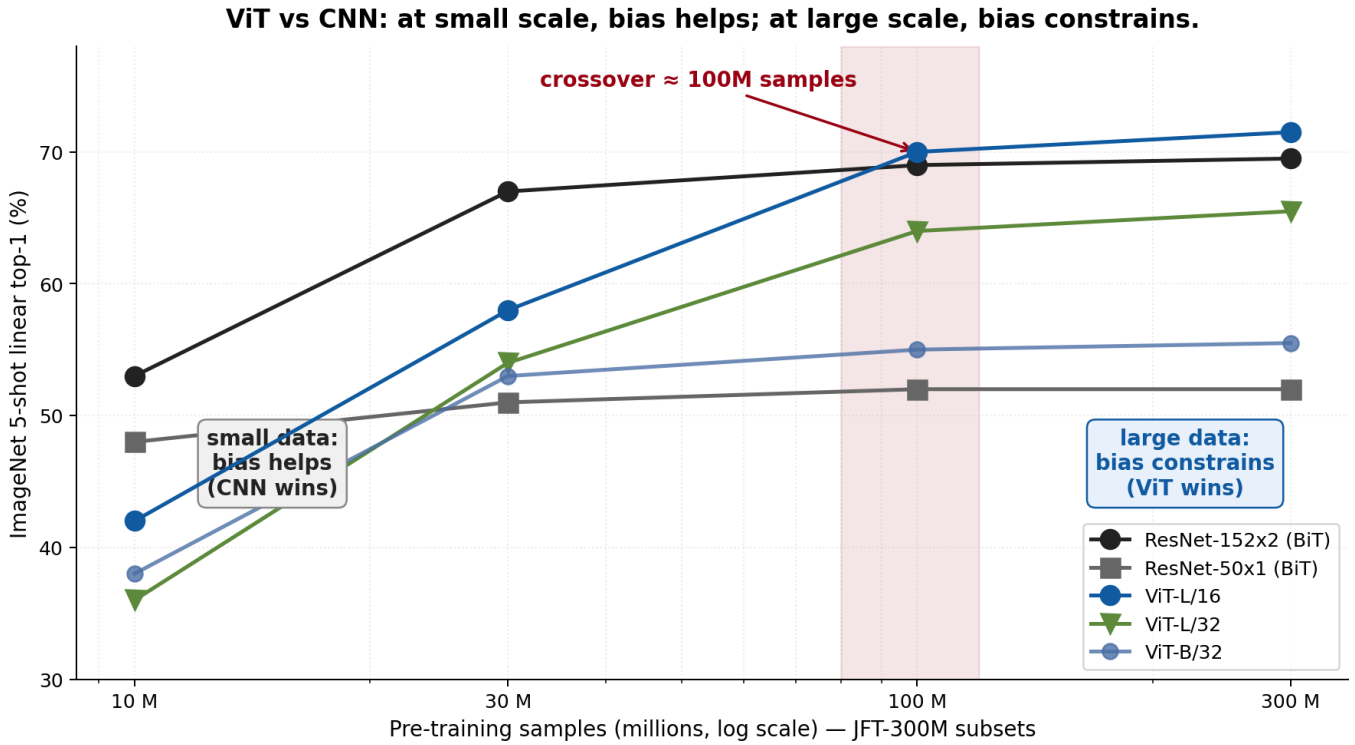
Did NAS actually discover anything?

- NAS-found architectures rarely transfer outside their search domain.
- Random search inside the same search space often performs **comparably** (Li & Talwalkar, 2019).
- **Most "NAS gains" came from the search space designers' priors, not the search itself.**

4. After ResNet — The Convergence with Transformers

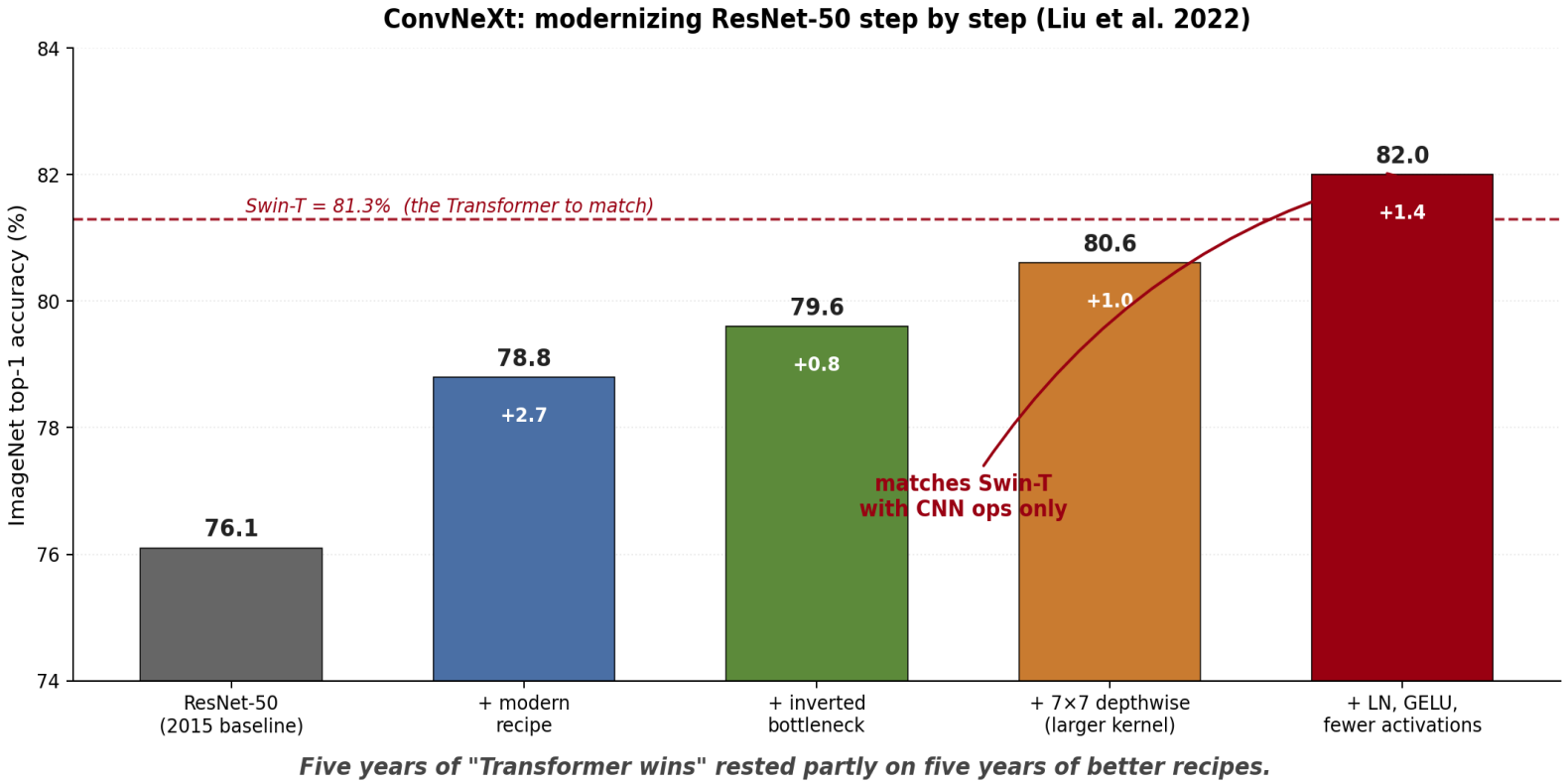
The Vision Transformer Disrupted Everything (2020)

- Take an image, cut into 16 × 16 patches, treat each patch as a token, run a Transformer.
- **No convolutions. No spatial inductive bias.**
- On ImageNet (1.2M images): worse than ResNet. **On JFT-300M: beats every CNN.**



The CNN's Reply: ConvNeXt (2022)

- Start with ResNet-50. Modernize step by step, borrowing only *design choices* (not operators) from Swin Transformer.



What this proves: most of ViT's gains were from training recipes and macro-design, not from attention itself.

What Each Era Bought Us

Era	Architecture	Bottleneck Solved	Cost
1998	LeNet	Hand-crafted features	Required infrastructure (waited 14 yrs)
2012	AlexNet	Scaling to ImageNet	60M params, 2 GPUs
2014	VGG	Design regularity	138M params (mostly FC)
2014	GoogLeNet	Compute at depth	Training complexity
2015	**ResNet**	**Optimization at depth**	**None — sets new baseline**
2017	MobileNet	Inference cost	~1% accuracy
2019	EfficientNet	Scaling principle	Recipe-coupled (overstated)
2018–20	NAS	Automation of design	Compute, transferability
2022	**ConvNeXt**	**"Did we need ViT?"**	**None — proves recipe matters**

When NOT to Reach for a CNN (2026 Edition)

Task	Right tool (today)	Why
Image classification, ≤10M training images	**CNN (ConvNeXt, EfficientNet)**	Inductive bias still wins data-efficiency
Image classification, ≥100M training images	**ViT / Hybrid**	Bias becomes a constraint
Mobile / on-device inference	**MobileNet / EfficientNet**	Hardware-aware design
Dense prediction (segmentation, detection)	**CNN backbone or hybrid**	Multi-scale features matter
Generation (diffusion)	**U-Net (CNN) + attention**	Best of both
Tokens that aren't pixels (text, audio)	**Transformer**	No translation equivariance to exploit

Summary

1. Modern CNNs solved cost, not accuracy.
2. Decoupling is the dominant motif.
   – *Depthwise, bottlenecks, compound scaling…*
3. Recipe ≠ architecture.
   – *Most claimed architectural gains 2017–2022 were partly recipe gains.*
4. Inductive bias is a double-edged sword.
   – *Regularizer at small scale; constraint at large scale. ViT and ConvNeXt are two sides of this trade-off.*

Next Lecture

L19. Self-Supervised Learning

- Contrastive learning (SimCLR, MoCo)
- The representation learning paradigm
- Foreshadowing: the pretraining recipe that makes both ConvNeXt and ViT actually work.