# What is the State of Neural Network Pruning?

Davis Blalock*
Jose Javier Gonzalez*
Jonathan Frankle
John V. Guttag

*equal contribution

# Overview

**Meta-analysis of neural network pruning**

We aggregated results across 81 pruning papers and pruned hundreds of networks in controlled conditions

- Some surprising findings…

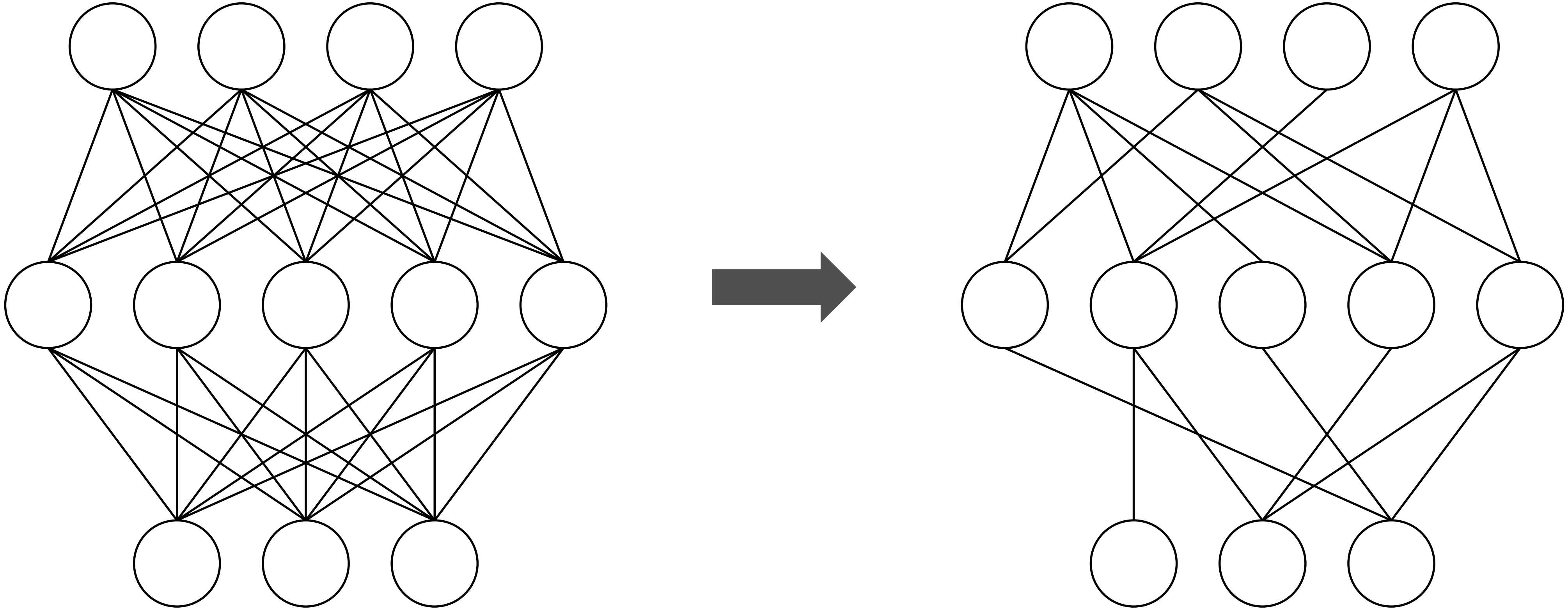
**ShrinkBench**

Open source library to facilitate development and standardized evaluation of neural network pruning methods

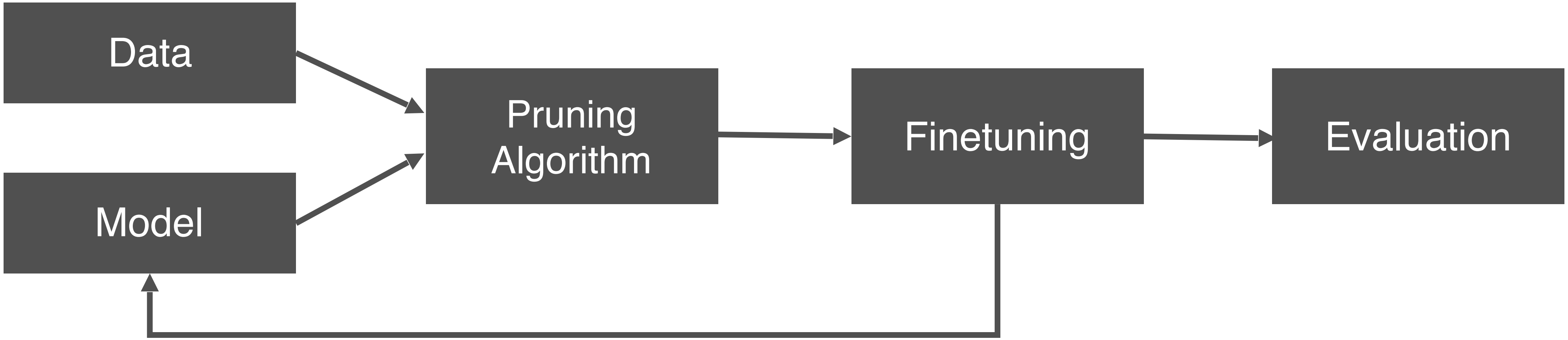
# Part 0: Background

# Neural Network Pruning

- Neural networks are often accurate but large
- **Pruning**: Systematically removing parameters from a network



# Typical Pruning Pipeline

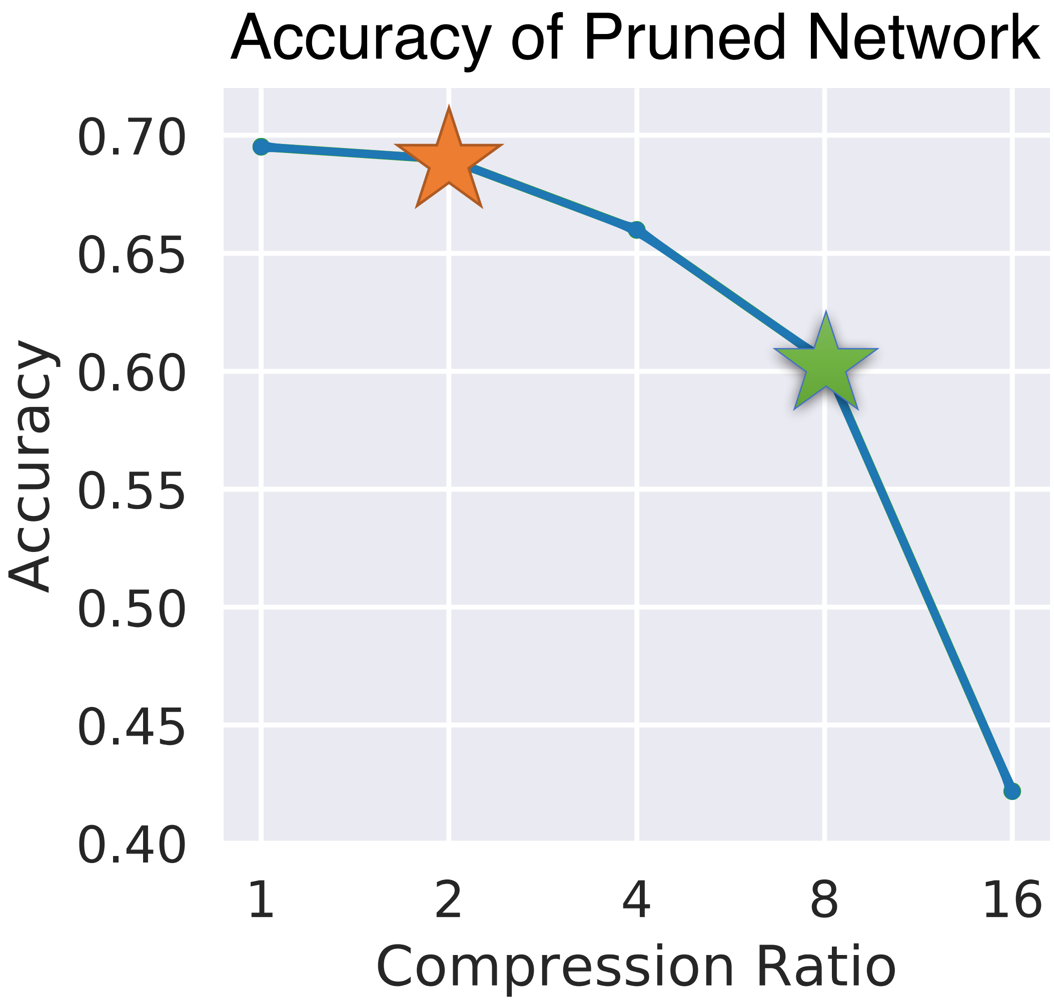



Many design choices:

- **Scoring** importance of parameters
- **Schedule** of pruning, training / finetuning
- **Structure** of induced sparsity
- **Finetuning** details — optimizer, duration, hyperparameters

# Evaluating Neural Network Pruning

- **Goal**: Increase efficiency of network as much as possible with minimal drop in quality
- Metrics
  - Quality = Accuracy
  - Efficiency = FLOPs, compression, latency…
- Must use comparable tradeoffs



# Part 1: Meta-Analysis

# Overview of Meta-Analysis

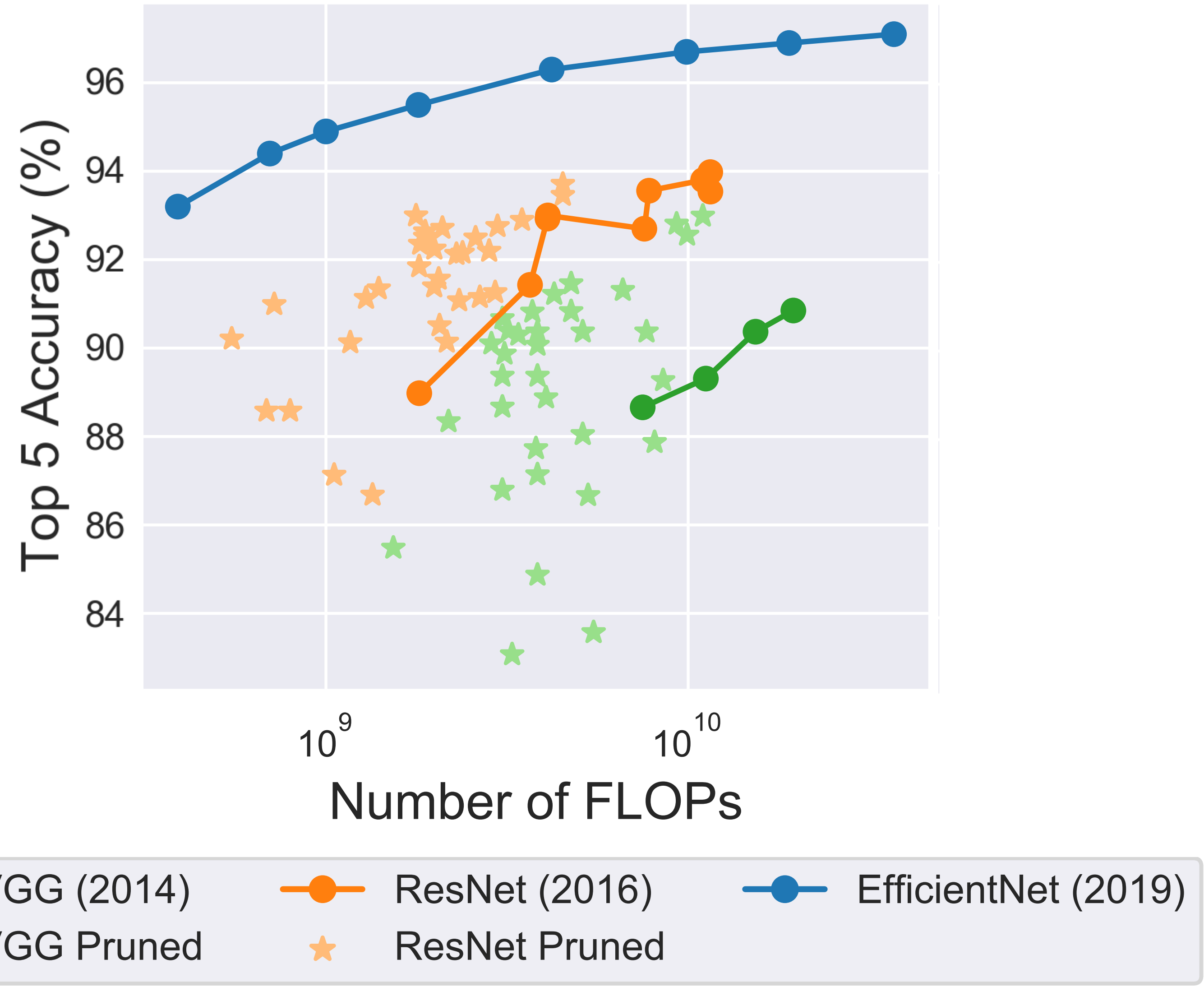
- We aggregated results across 81 pruning papers
- Mostly published in top venues
- Corpus closed under experimental comparison

| Venue | # of Papers |
|---|---|
| arXiv only | 22 |
| NeurIPS | 16 |
| ICLR | 11 |
| CVPR | 9 |
| ICML | 4 |
| ECCV | 4 |
| BMVC | 3 |
| IEEE Access | 2 |
| Other | 10 |

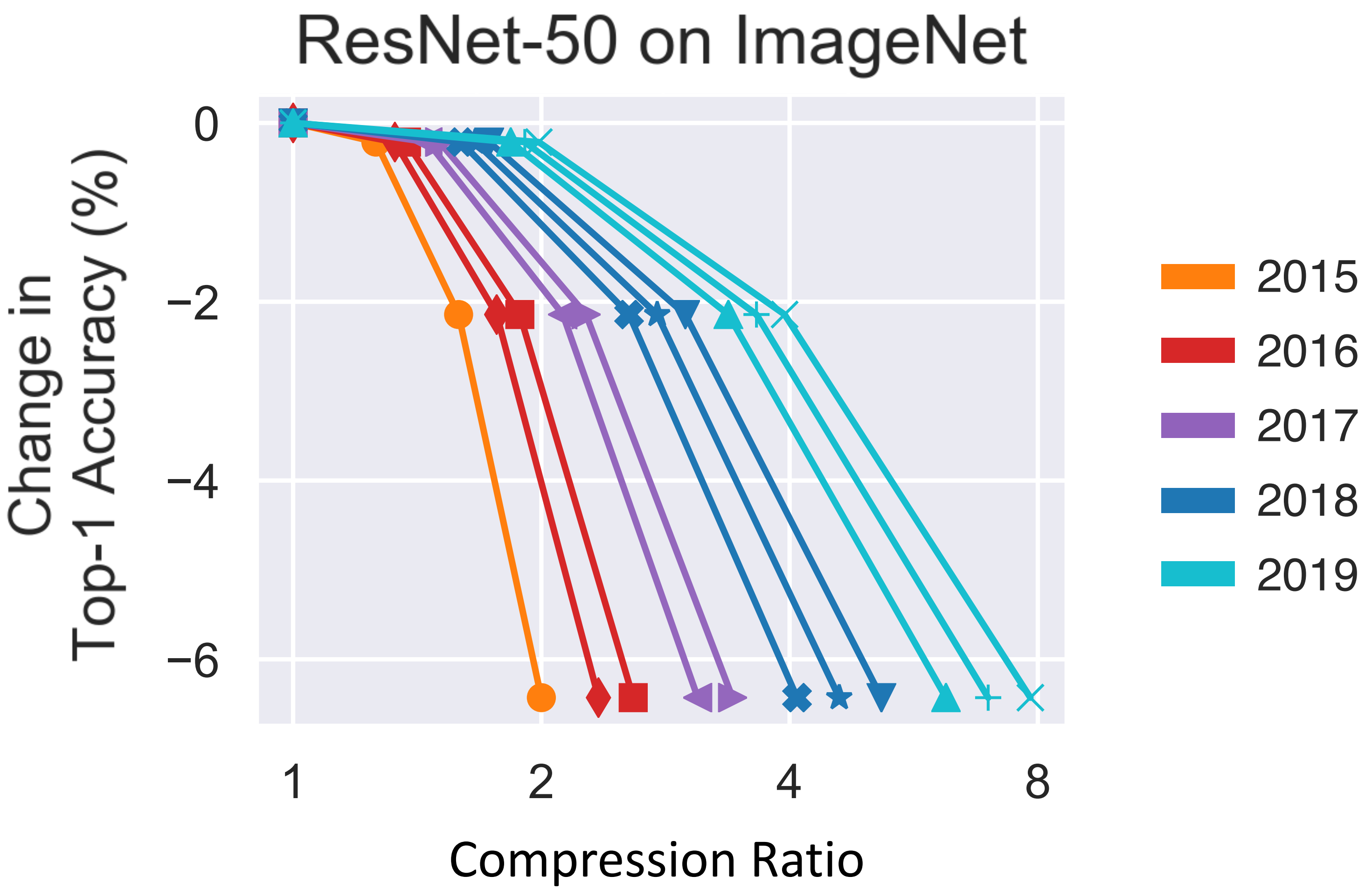
# Robust Findings

- **Pruning works**
  - Almost any heuristic improves efficiency with little performance drop
  - Many methods better than random pruning
- Don’t prune all layers **uniformly**
- **Sparse models better** for fixed # of parameters

# Better Pruning vs Better Architecture

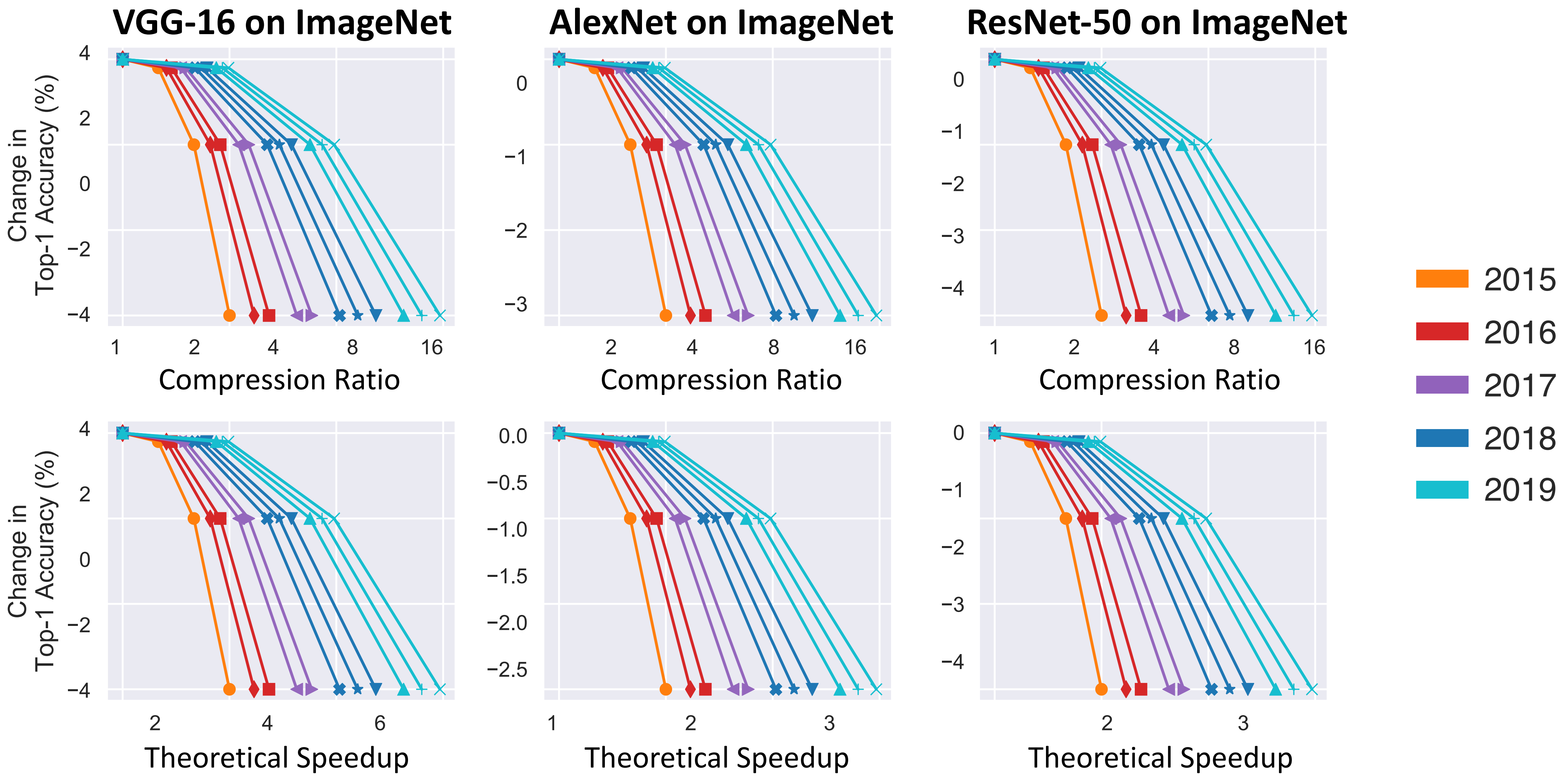


# Ideal Results Over Time

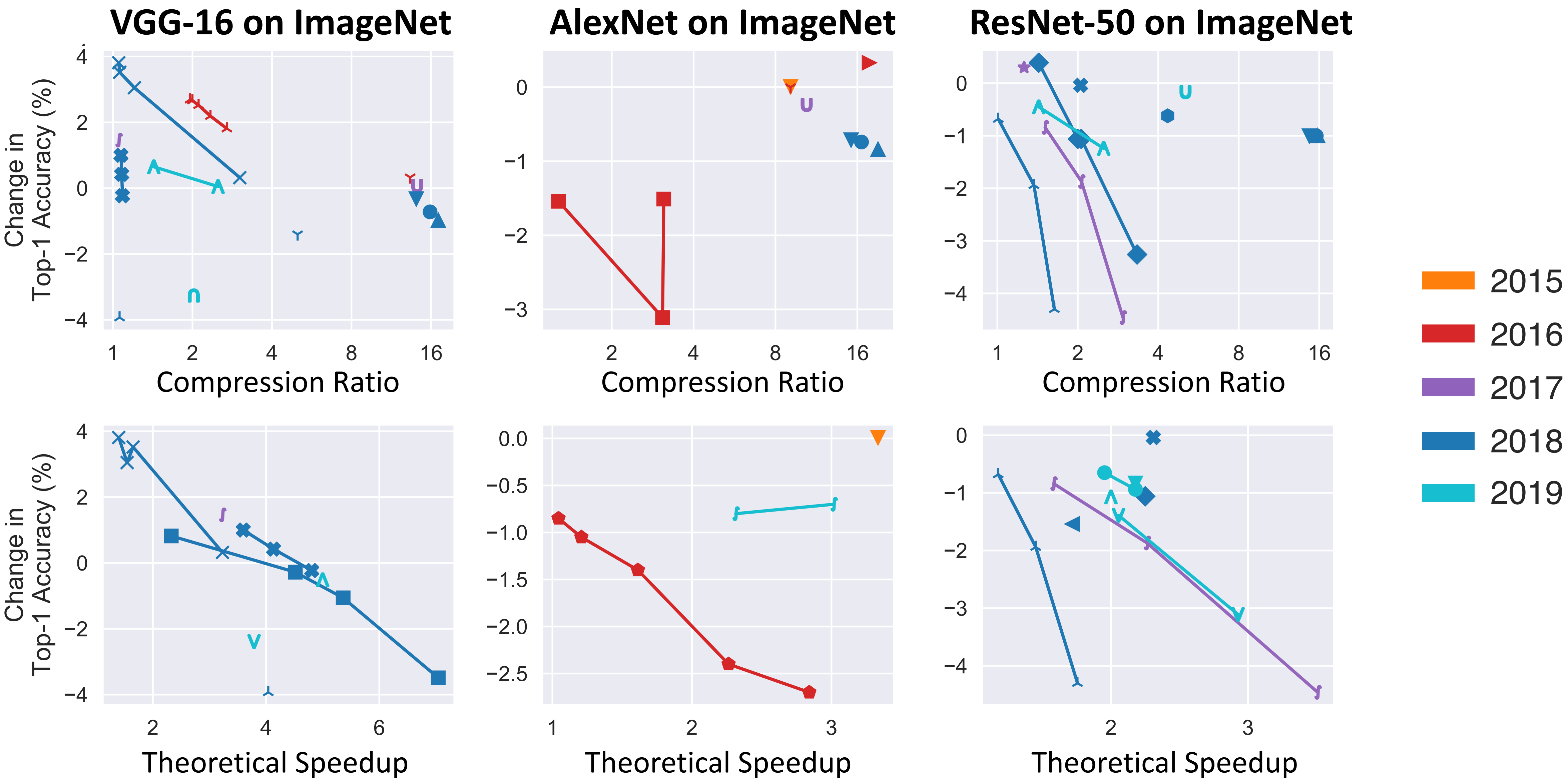



(Dataset, Architecture, X metric, Y metric, Hyperparameters) → Curve

# Ideal Results Over Time



# Actual Results Over Time



# Quantifying the Problem

- Among 81 papers:
  - 49 datasets
  - 132 architectures
  - 195 (dataset, architecture) pairs

- Vicious cycle: extreme burden to compare to existing methods

**All (dataset, architecture) pairs used in at least 4 papers**

| Dataset | Architecture | # of Papers Using Pair |
|---|---|---|
| ImageNet | VGG-16 | $\boxed{22}$ |
| CIFAR-10 | ResNet-56 | 14 |
| ImageNet | ResNet-50 | 14 |
| ImageNet | CaffeNet | 11 |
| ImageNet | AlexNet | 9 |
| CIFAR-10 | CIFAR-VGG | 8 |
| ImageNet | ResNet-34 | 6 |
| ImageNet | ResNet-18 | 6 |
| CIFAR-10 | ResNet-110 | 5 |
| CIFAR-10 | PreResNet-164 | 4 |
| CIFAR-10 | ResNet-32 | 4 |

# Dearth of Reported Comparisons

- **Presence of comparisons:**
  - Most papers compare to at most 1 other method
  - 40% papers have never been compared to
  - Pre-2010s methods almost completely ignored
- **Reinventing the wheel:**
  - Magnitude-based pruning: *Janowsky (1989)*
  - Gradient times magnitude: *Mozer & Smolensky (1989)*
  - "Reviving" pruned weights: *Tresp et al. (1997)*

# Pop quiz!

- Alice's network has 10 million parameters. She prunes 8 million of them. What compression ratio might she report in her paper?

  A. 80%

  B. 20%

  C. 5x

  D. No reported compression ratio

# Pop quiz!

- Alice's network has 10 million parameters. She prunes 8 million of them. What compression ratio might she report in her paper?

  **A. 80%**

  **B. 20%**

  **C. 5x**

  **D. No reported compression ratio**

# Pop quiz!

- According to the literature, how many FLOPs does it take to run inference using AlexNet on ImageNet?

  A. 371 million
  B. 500 million
  C. 724 million
  D. 1.5 billion

# Pop quiz!

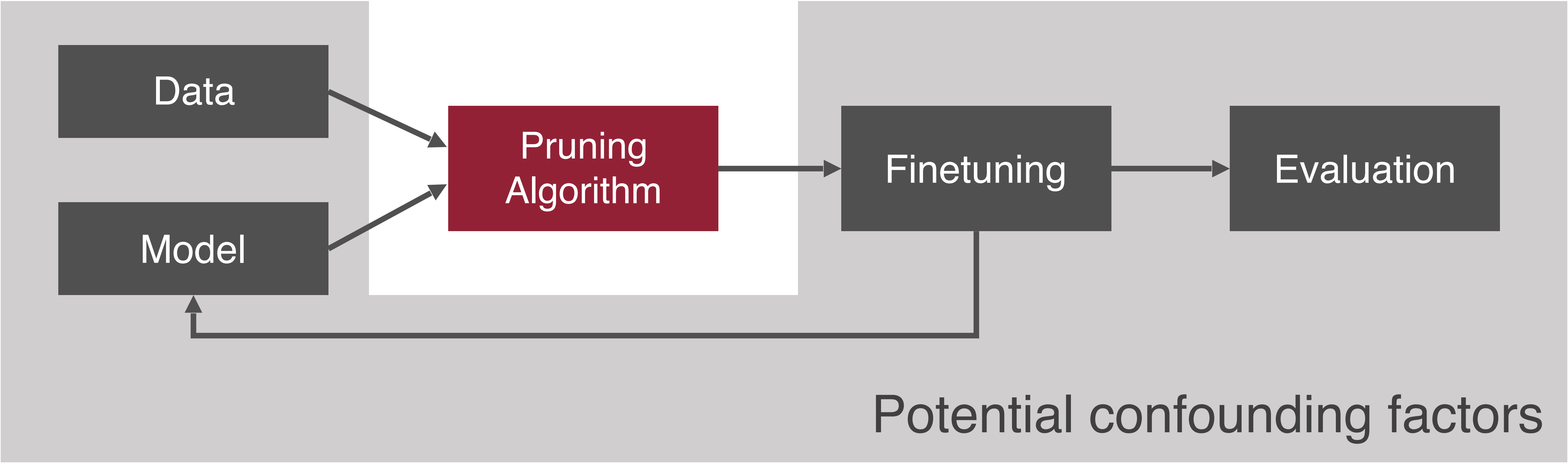
- According to the literature, how many FLOPs does it take to run inference using AlexNet on ImageNet?

  A. **371 million**
  B. **500 million**
  C. **724 million**
  D. **1.5 billion**

# Part 2: ShrinkBench

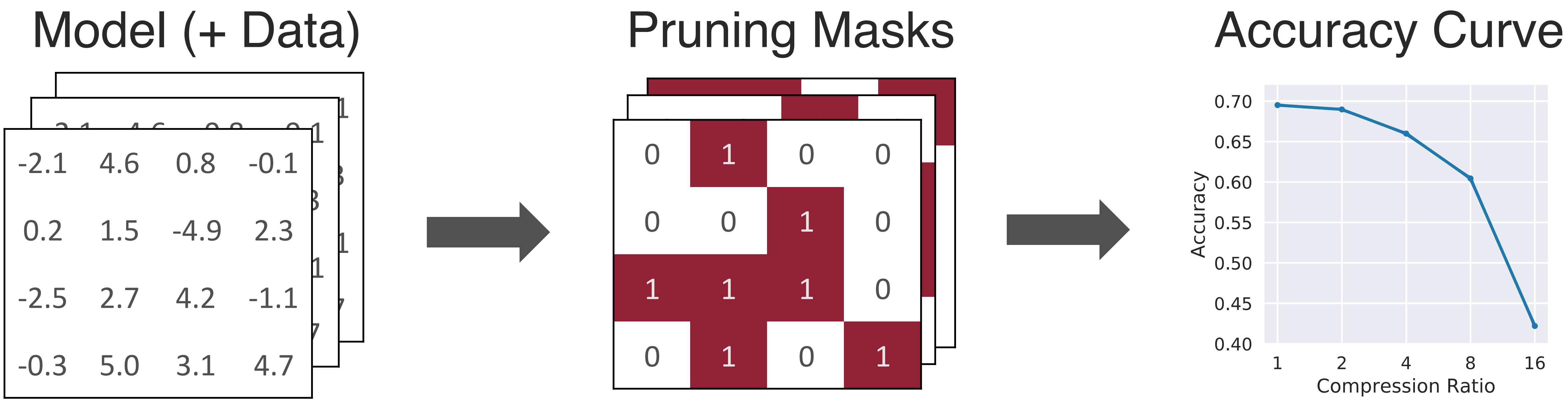
# Why ShrinkBench?

- Want to hold everything but pruning algorithm constant
  - Improved rigor, development time

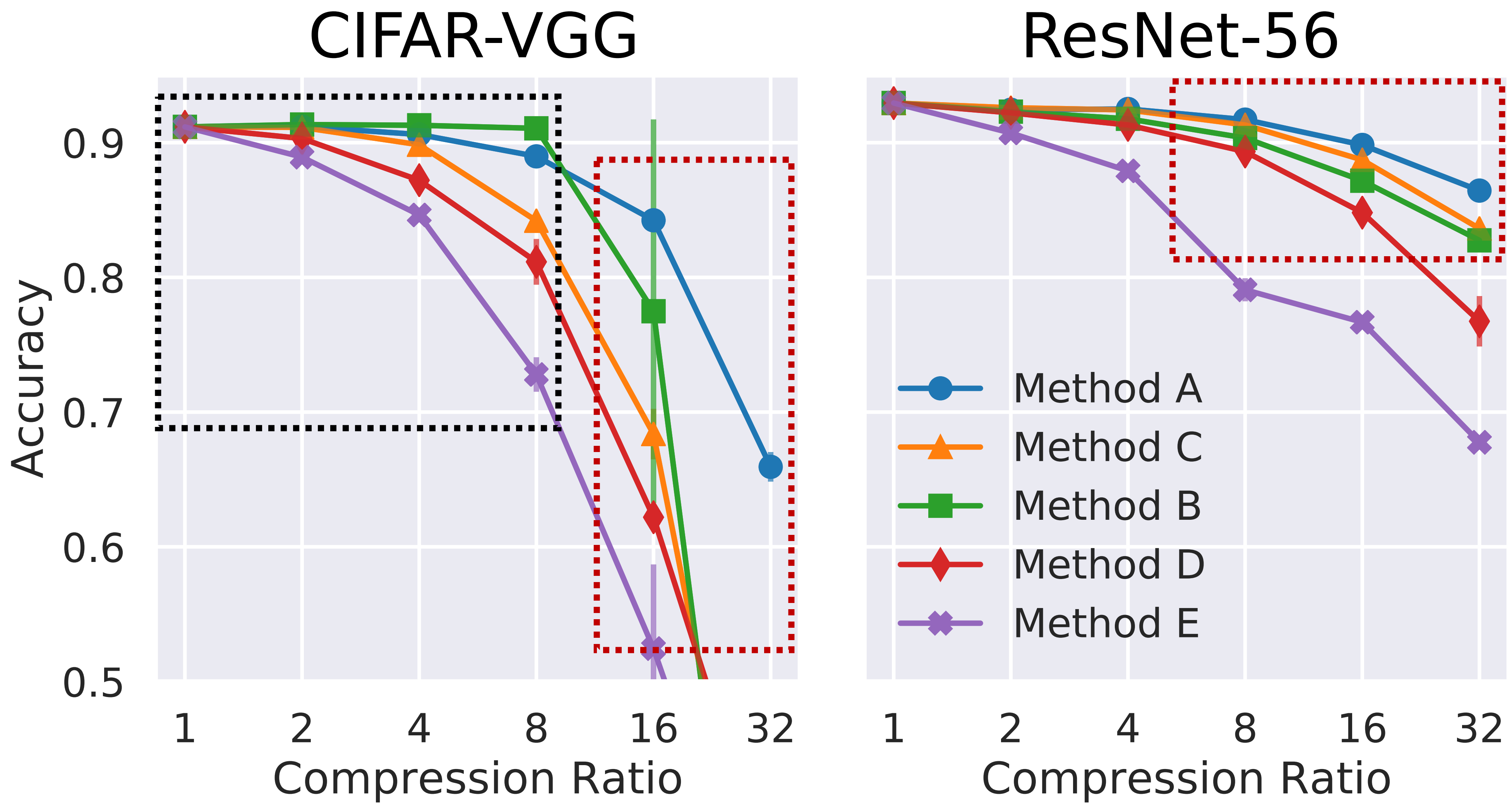


# Masking API

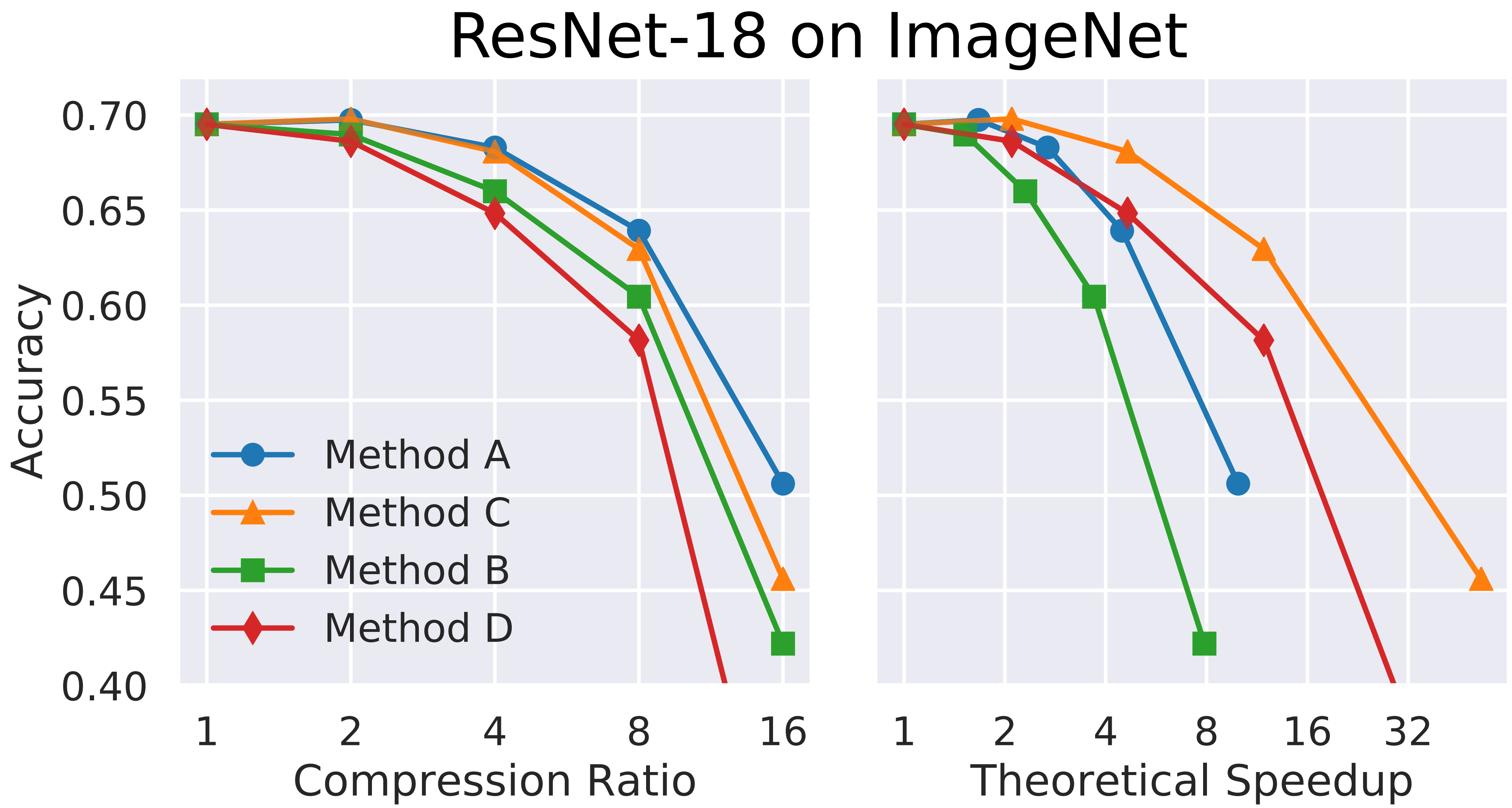
- Lets algorithm return arbitrary masks for weight tensors
- Standardizes all other aspects of training and evaluation



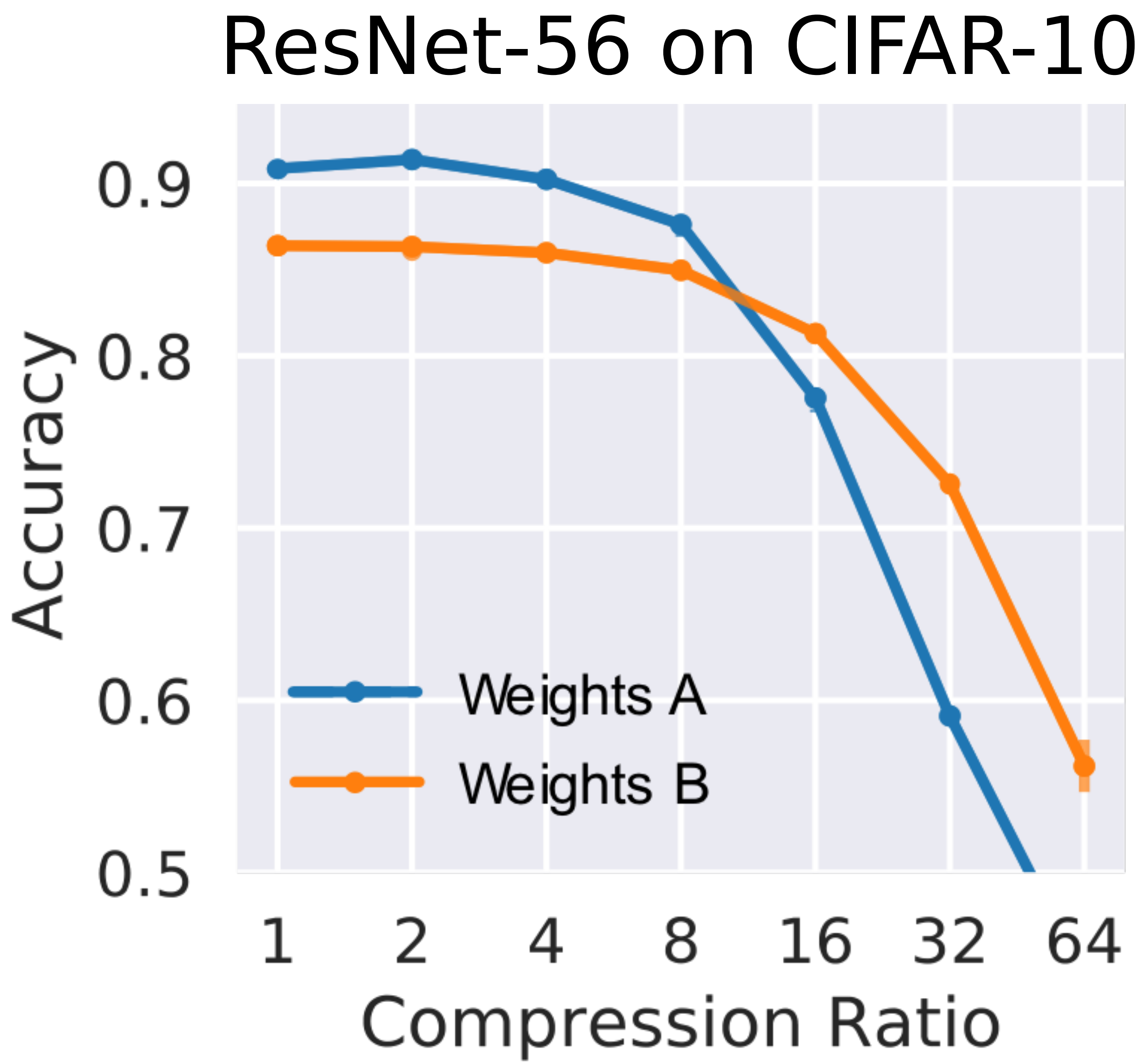
# Crucial to Vary Amount of Pruning & Architecture



# Compression and Speedup are not Interchangeable



# Using Identical Initial Weights is essential



# Conclusion

- **Pruning works**
  - But not as well as improving architecture
- **But we have no idea what methods work the best**
  - Field suffers from extreme fragmentation in experimental setups
- **We introduce a library/benchmark to address this**
  - Faster progress in the future, interesting findings already

[https://github.com/jjgo/shrinkbench](https://github.com/jjgo/shrinkbench)

# Questions?