Exploring Binarization and Pruning of Convolutional Neural Networks

Ameya Prabhu

Work under the guidance of:

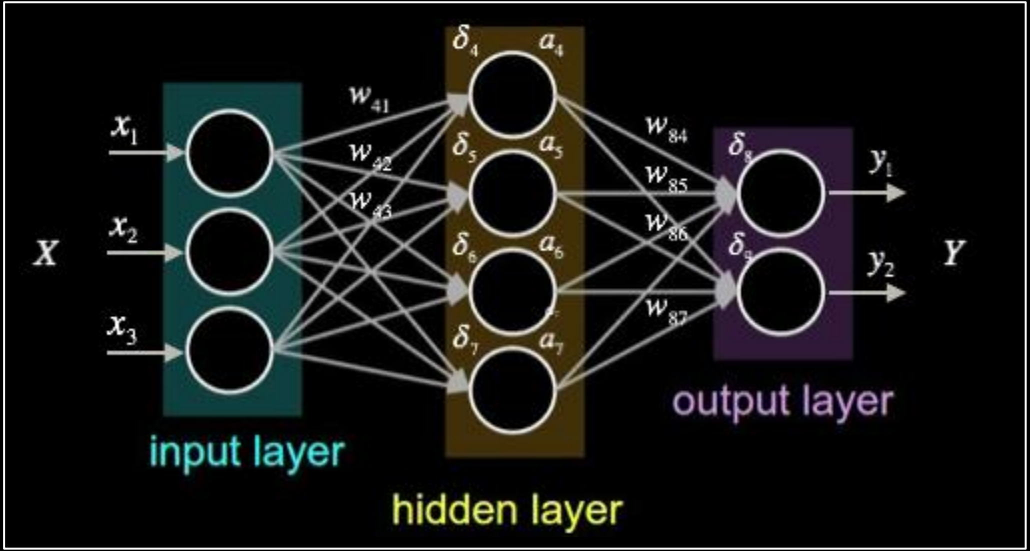
Prof. Anoop Namboodiri
Center for Visual Information Technology (CVIT)
IIIT Hyderabad

Thesis Panel:

Prof. Anoop Namboodiri & Prof. Avinash Sharma
Center for Visual Information Technology (CVIT)
IIIT Hyderabad

Background

Neural Network (MLP)



What we will mostly be dealing with:

(a) All neurons in layer are connected to all other neurons in that layer.
(b) The non-linear function has **floating point** weights (***w***) and activations (***a***).

Index

1. Introduction

- Motivation: Quantization & Pruning
- Applications
- Related Work

2. Distribution-aware Binarization

- Motivation for the Hypothesis
- Representational Power
- “General” Binary Representations
- Experiments & Results

3. Hybrid Binary Networks

- Motivation for the Hypothesis
- Metric for quantifying “where”?
- Experiments & Results

4. Deep Expander Networks

- Motivation for the Hypothesis
- What are expander graphs?
- What guarantees do we get?
- Experiments & Results

Motivation

Biological Perspective | Practical Perspective

Intuition: Brain is amazingly efficient in compute, memory & power, current ANNs are far worse.

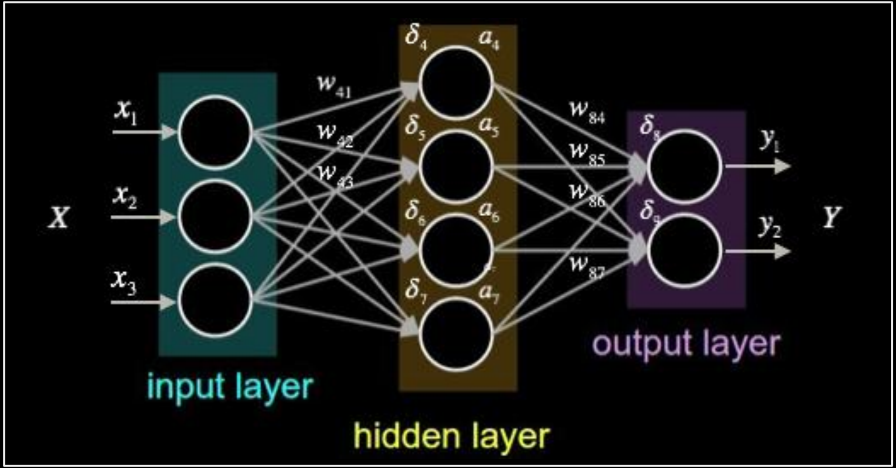
Hodgkin-Huxley Model (Leaky Integrate-and-Fire) ≈> Spiking Neural Networks (SNNs)

Binary Networks: When we contrast SNNs with current MLP models, we can observe[1]:

- SNNs have {0,1} activations (a_i) (fire or not), MLPs have activations (a_i) in $\mathbb{R}$
- SNNs have {0,1} weights (w_i) (exciting and inhibiting), MLPs have weights (w_i) in $\mathbb{R}$

Pruning: (Mocanu et. al)[2] argue that:

- SNNs: Sparse edges, MLPs: Fully connected edges.
- Why connect all neurons to all other neurons?



[1] Whetstone: A Method for Training Deep Artificial Neural Networks for Binary Communication, Nature, Machine Intelligence, 2019
[2] Scalable training of artificial neural network with adaptive sparse connectivity inspired by network science, Nature Communications, 2018

Motivation

Biological Perspective | Practical Perspective

Binarization

If 32-bit/64-bit float valued matrix *W* and *I* => boolean matrix *B* and I_B, we get 64x compression by bit-stuffing!

1	0	1	1	..	0	1	0	1	→	56	21	⟺ 64x gap ⟺	1.2	0.6	1.4	..	1.7	3.1	0.2

B → *B* ⟺ *W*

We similarly obtain 64x speedup by matrix multiplication (convolution) with bitwise XNOR-Popcount ops parallely.

56 | 21 ⊗ 24 | 69 → 67 | 54 ⟺ 64x gap

B ⊗ I_B → Out_B

1.2	0.6	1.4	..	1.7	3.1	0.2
⊛						
2.1	1.7	9.3	..	6.4	0.3	0.6
=						
1.2	0.6	1.4	..	1.7	3.1	0.2

W ⊛ *I* = *Out*

Pruning

We greatly reduce the dimensions of the weight matrix by selectively connecting important edges.

$W_{orig} \in \mathbb{R}^{n \times m} \rightarrow W_{pruned} \in \mathbb{R}^{n \times k}$ where $k \ll m$.

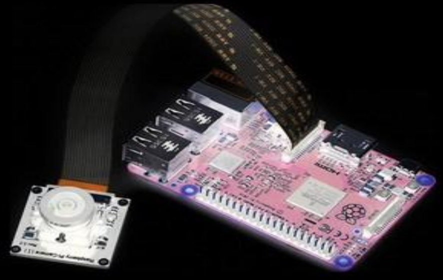
Applications

Practical Power

- 32x smaller memory (32GB VRAM models in 1GB for GPU)
- Forward-pass a month's worth data in 1 day (on a GPU)
- Consume ~100x less electricity

Applications

Eg: Mobiles ✗ 12GB GPU, ✗ 64GB RAM, ✗ High battery => "AI-powered apps"?



Apply this to your problem! If there is an ANN, it can be compressed!
Eg: Language modeling, text classification, semantic segmentation, object detection, 3D modelling, etc!

INDEX

1. Introduction

- Biological & practical purpose.
- Applications: Low memory, compute and power usage.

3. Hybrid Binary Networks

- Motivation for the Hypothesis
- Metric for quantifying “where”?
- Experiments & Results

2. Distribution-aware Binarization

- Related Works: Use 2-bit **w** and/or **a**
- Representational Power
- “General” Binary Representations
- Experiments & Results

4. Deep Expander Networks

- Motivation for the Hypothesis
- What are expander graphs?
- What guarantees do we get?
- Experiments & Results

Binary Nets: Survey

First modern binarization works: Hubara et. al.[1] and Rastegari et. al.[2]: Proposed "how to binarize".

<u>Achievements</u>: 1) Same accuracy at 64x => Binary weights & activations (on CIFAR10 using VGG7)
2) Same accuracy => Binary weights (on Imagenet using AlexNet)

<u>Failures</u>: 1) 12% drop in accuracy => Binary weights & activations (on Imagenet using AlexNet)
2) CIFAR10 results do not scale to subsequent networks (Resnets, etc)

Later works[3] state "binary weights and activations are insufficient" & move to 2-bit representations.

<u>Our chance</u>: 2-bit representations lose all speedups and compression (bool => int8)

<u>What do we do?</u>

- Are binary representations sufficient?
- Can we improve the binary approximation or make "clever" modifications to improve the accuracies?

[1] Hubara et. al. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations, JMLR 2018
[2] Rastegari et. al. XNOR-Net: ImageNet Classification Using BinaryConvolutional Neural Networks, ECCV 2016
[3] Zhu et. al. Trained Ternary Quantization, ICLR 2017

BINARY NETS: ENOUGH?

Problem: (Lin et al.)[1] consider the no. of neurons required p(**x**), a multivariate monomial, expressed as the product of *n* variables: $\prod_{i=1}^{n} x_i = \frac{1}{2^n} \sum_{\{s\}} s_1 ... s_n \sigma(s_1 x_1 + ... + s_n x_n).$

Result 1: Using a single hidden layer network requires exactly 2^n infinite precision neurons.

Our Result: There exists a construction of binary weighted network which requires exactly the same number (2^n) neurons.

Result 2: A *k* layered deep network, (Rolnick et. al.)[2] set an upper bound of #neurons as: $\mathcal{O}\left(n^{(k-1)/k} \cdot 2^{n^{1/k}}\right)$ and show experimental evidence for tightness of this upper bound.

Our Result: There exists a construction of binary weighted network which requires the same order of neurons. This is interesting because we reduce the infinite precision to a finite precision, which is *not a* constant factor.

Assumptions: The activation $\boldsymbol{\sigma}$ has nonzero Taylor coefficients up to degree *n*. Eg: ELU.

Summary: Binary weight networks could be as expressible as infinitely precise ones.

[1] Lin et. al, Why does deep and cheap learning work so well?, Journal of Statistical Physics, 2017
[2] Rolnick et. a. The power of deeper networks for expressing natural functions, ICLR 2018

Distribution-Aware Binary Nets

- **Current Approaches**: Binarize $\mathbf{x} \in \mathbb{R}^n$ are as follows:

  -> $\mathbf{b} \in \{-1,+1\}^n$ where binarization function is sign($\mathbf{x}$).
  -> $\mathbf{b} \in \{-\alpha,+\alpha\}^n$ where binarization function is sign($\mathbf{x}$) and $\alpha = |x|_1/n$ is a scaling factor

- **Our proposal**: A given $\alpha,\beta \in \mathbb{R}$ to form a 1 bit representation: binarized weight vector $\mathbf{b} \in \{\alpha,\beta\}^n$

- **Questions**: Some questions immediately arise about a "general" binary representation:

  1) Is there a optimal solution to this?
  Ans: Yes!

  2) Is it efficiently computable a.k.a. in O(n)?
  Ans: Yes, with dynamic programming!

  3) Does it improve results when trained?
  Ans: Yes, albeit it needs some stabilization.

OPTIMAL A, B AND e

Given weight vector **W**, binarize to get $\widetilde{\mathbf{W}}$ $\in$ {α,β}^{n} represented as α*(**e**)+ β*(**1**-**e**) where **e** $\in$ {0,1}^{n} , (**e** != 0 and **e** != 1)

($\widetilde{\mathbf{W}}$: [α α β β α … β α β] and **e** : [1 1 0 0 1 ... 0 1 0])

We formulate it as an optimization problem: $\widetilde{\mathbf{W}}^* = \underset{\widetilde{W}}{argmin} \, || \, \mathbf{W} - \widetilde{\mathbf{W}} \, ||^2$

The optimal α,β and **e** we obtain are as follows:

$$\alpha = \frac{\mathbf{W}^T\mathbf{e}}{K}, \; \beta = \frac{\mathbf{W}^T(\mathbf{1}-\mathbf{e})}{n-K} \qquad \mathbf{e}^* = \underset{e}{argmax}\left(\frac{||\,\mathbf{W}^T\mathbf{e}\,||^2}{K} + \frac{||\,\mathbf{W}^T(\mathbf{1}-\mathbf{e})\,||^2}{n-K}\right)$$

where K = $\mathbf{e}^T\mathbf{e}$ (K counts 1^s in **e**)

Beauty: Projecting **W** onto two vectors **e** and (**1**-**e**).

Problem: Picking the optimal **e** among all K requires O(n^2) time.

Solution: We propose a prefix-sum based dynamic programming subroutine which computes it in O(n log$n$) time. (Details in the thesis)

Datasets & Models

- **TU-Berlin**: A popular large sketch recognition benchmark consisting of 20,000 sketches over 250 classes.
- **Sketchy**: A popular SBIR benchmark consisting of 75,471 sketches over 125 classes.

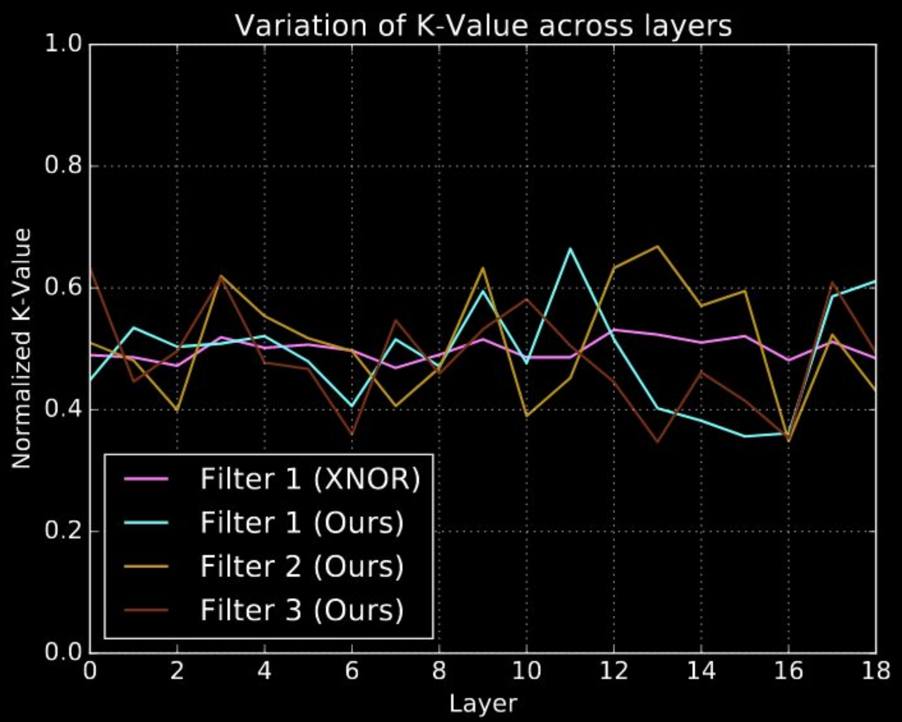
We tested the following models:

- **Sketch-A-Net**: (A popular Alexnet-like network), **ResNet-18** & **GoogleNet**

Results

Improvements on TU-Berlin and Sketchy respectively:

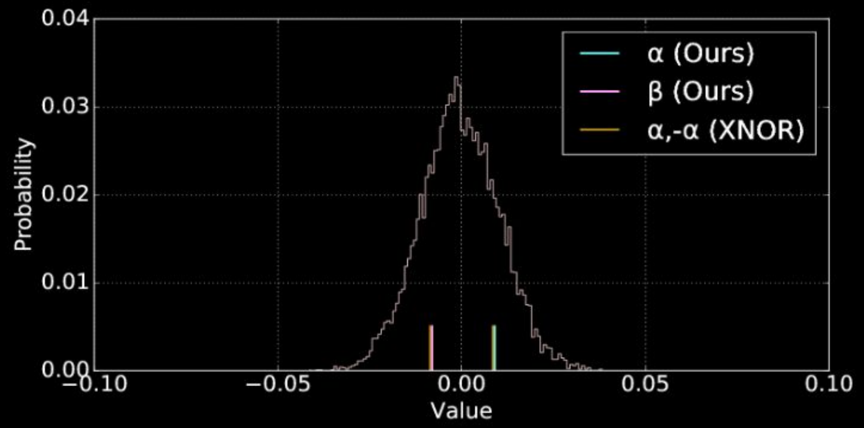
- Sketch-A-Net: 0.8% and 2% improvement
- ResNet-18: 2.5% and 1.4% improvement
- GoogleNet: 1.5% and 0.6% improvement



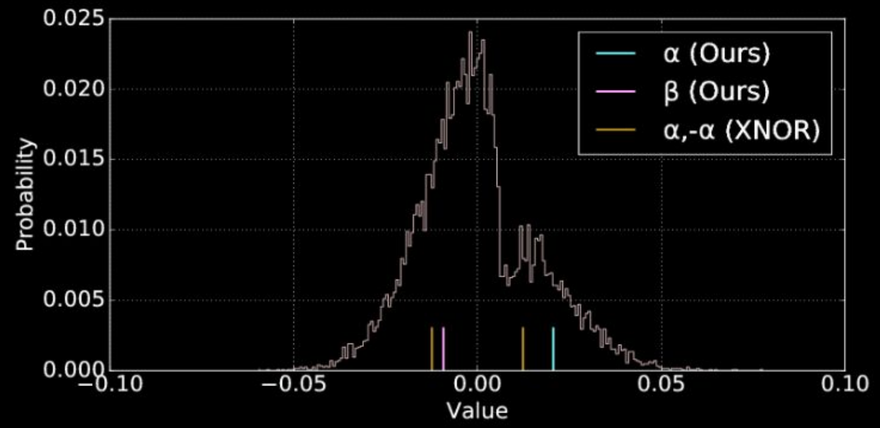
Models	Method	Accuracies	
		TU-Berlin	Sketchy
Sketch-A-Net	FPrec	72.9%	85.9%
	WBin (BWN)	73.0%	85.6%
	FBin (XNOR-Net)	59.6%	68.6%
	WBin DAB-Net	72.4%	84%
	FBin DAB-Net	60.4%	70.6%
Improvement	XNOR-Net s DAB-Net	+0.8%	+2.0%
ResNet-18	FPrec	74.1%	88.7%
	WBin (BWN)	73.4%	89.3%
	FBin (XNOR-Net)	68.8%	82.8%
	WBin DAB-Net	73.5%	88.8%
	FBin DAB-Net	71.3%	84.2%
Improvement	XNOR-Net s DAB-Net	+2.5%	+1.4%
GoogleNet	FPrec	75.0%	90.0%
	WBin (BWN)	74.8%	89.8%
	FBin (XNOR-Net)	72.2%	86.8%
	WBin DAB-Net	75.7%	90.1%
	FBin DAB-Net	73.7%	87.4%
Improvement	XNOR-Net s DAB-Net	+1.5%	+0.6%

ADDITIONAL ABLATION STUDIES

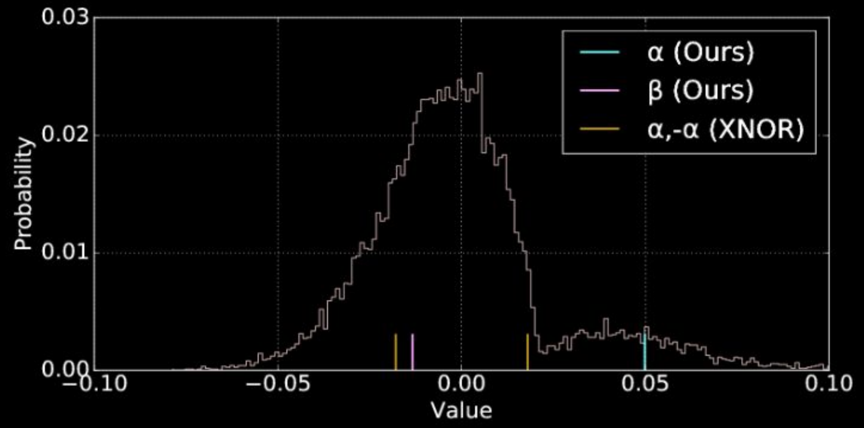
Variation of α and β across a filter's weights during training: ($T_1 < T_2 < T_3 < T_4$)



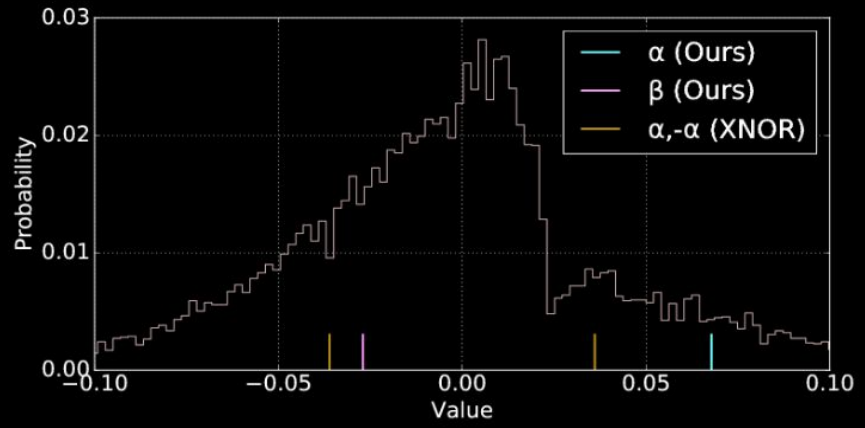
(1)



(2)



(3)



(4)

Future Directions

Code: Implementation available on github. Links provided in the thesis.

Nice Directions

- **Stabilization techniques**: Bayesian Binary Networks (ICCV 2019)

INDEX

1. Introduction

- Biological & practical purpose.
- Practical apps: Low memory, compute & power use.
- Related Works: Use 2-bit **w** and/or **a**

3. Hybrid Binary Networks

- Motivation for the Hypothesis
- Metric for quantifying “where”?
- Experiments & Results

2. Distribution-aware Binarization

- As expressive as infinite precision.
- Projects to 2 weighted || vectors.
- “Optimal” binary rep., efficiently computable, trainable via SGD.

4. Deep Expander Networks

- Motivation for the Hypothesis
- What are expander graphs?
- What guarantees do we get?
- Experiments & Results

THE NOVEL QUESTION

Recent works: “How” to binarize and (recently) more “binarizable” architectures.

We ask: “Where” to binarize?

“Intelligently” *not* binarizing some activations (**a**) recovers accuracy drops with minimal increase in FLOPs.

Intuitions:

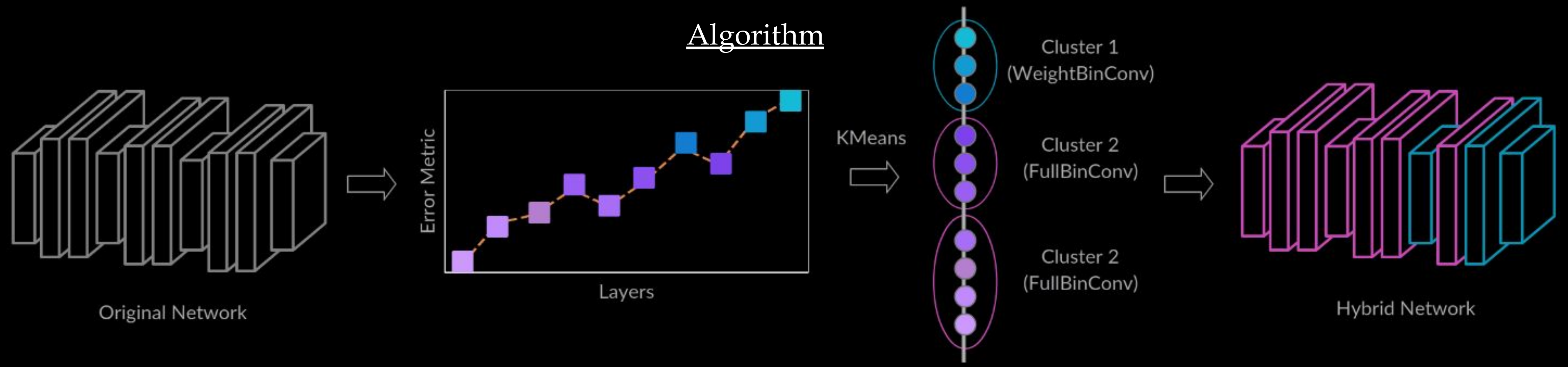
- Low-computations layers are amenable to hybridization.
- Layers with high binarization errors ($\mathbf{E} = \frac{\| \mathbf{I} - \mathbf{I_B} \|^2}{n}$) are amenable

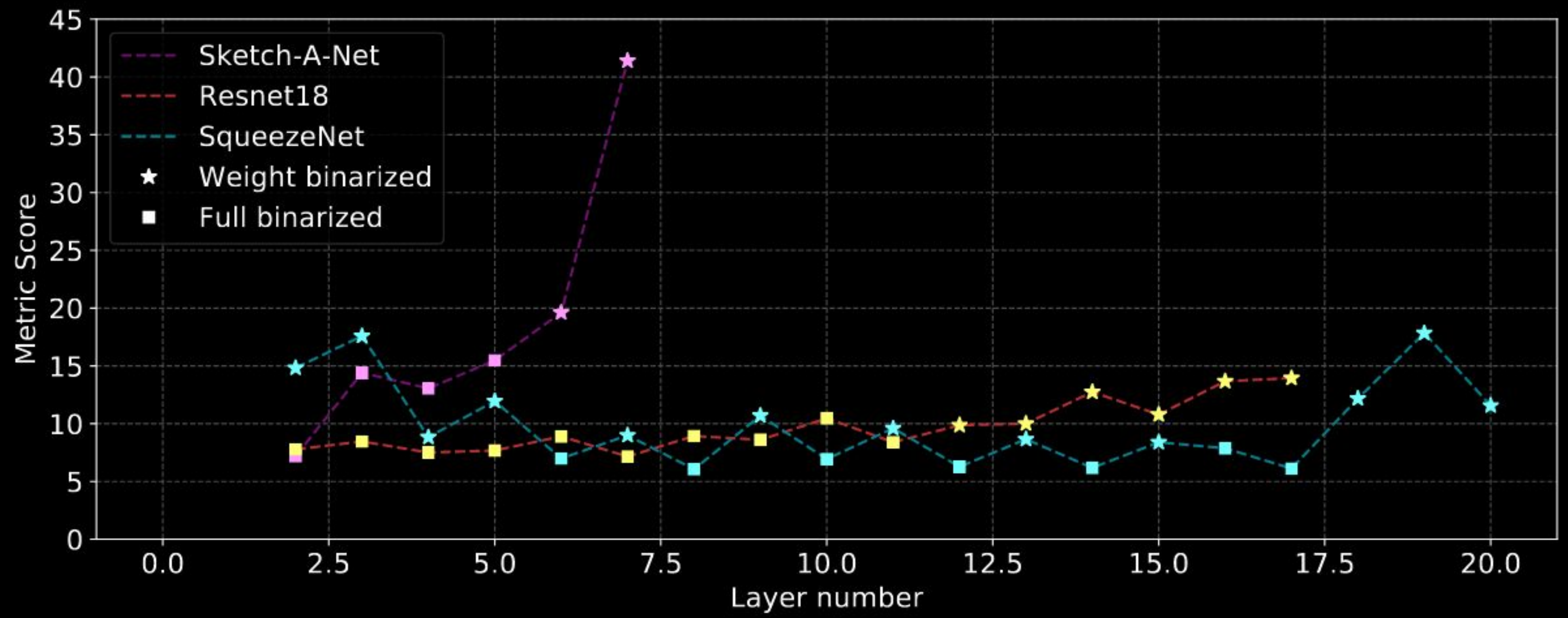
Discovery: Layers with low-computational cost (NF) turn out to have high binarization errors (E) on layerwise analysis.

We optimize a metric $\mathbf{M} = \mathbf{E} + \gamma \cdot \frac{1}{\mathbf{NF}}$ to create our “hybrid” binary network!

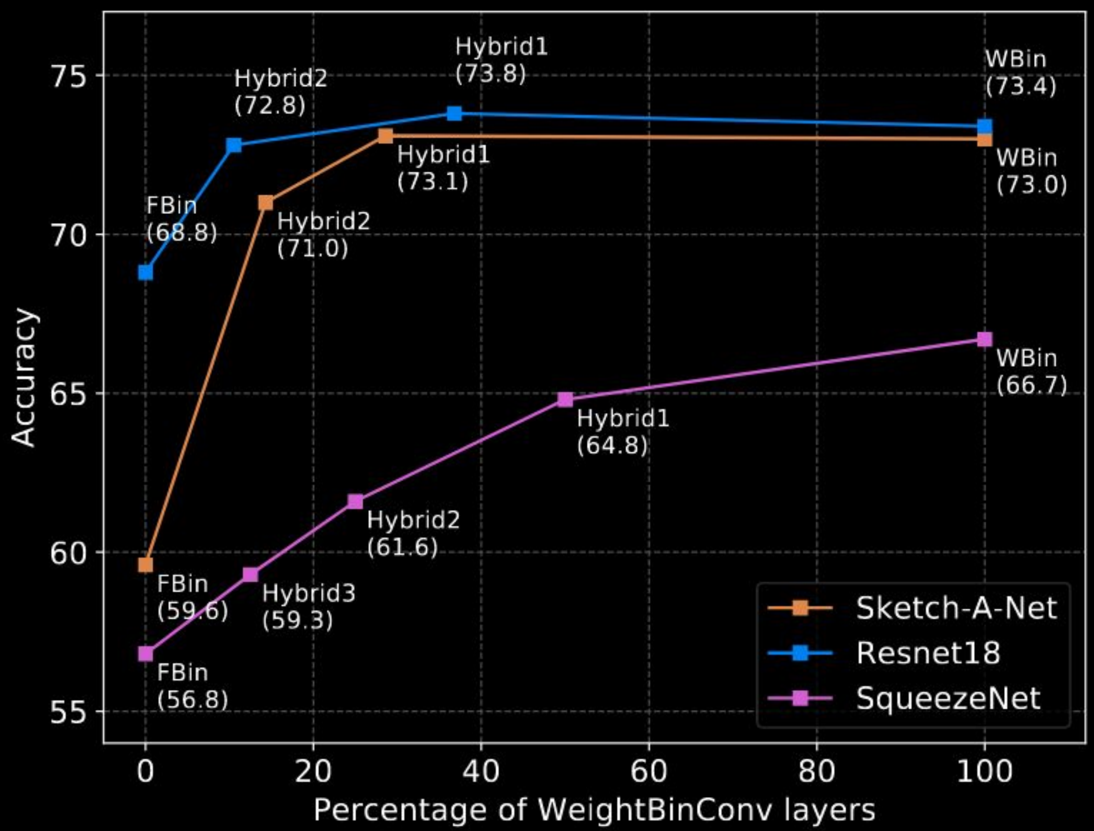
How to "Hybridize" Network?

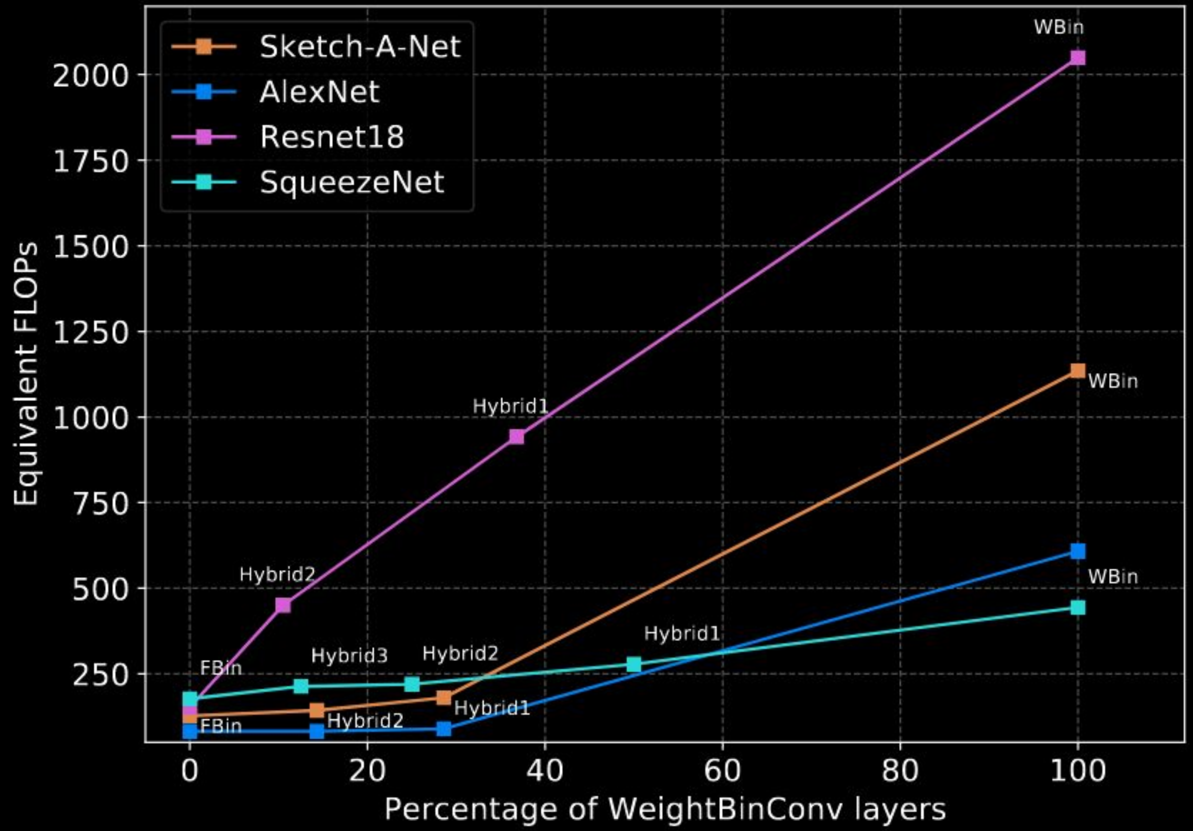
Algorithm





Results: Can we trade-off Effectively?





Recover most of the accuracy with minimal addition in #FLOPs

Comparisons: TU-Berlin and Sketchy datasets

Improvements on TU-Berlin and Sketchy:

- Sketch-A-Net improves 13.5% and 15%.
  - ResNet-18 improves 5% and 5.1%.
  - Squeezenet improves 8% and 13%.

Model	Method	Accuracy		Memory Savings	FLOPs (in M)
		TU-Berlin	Sketchy		
Sketch-A-Net	FPrec	72.9%	85.9%	1x	608 (7.8x)
	WBin (BWN)	73%	85.6%	29.2x	406 (5.2x)
	FBin (XNOR)	59.6%	68.6%	19.7x	**78 (1x)**
	Hybrid	**73.1%**	**83.6%**	**29.2x**	**85 (1.1x)**
Increase	Hybrid vs FBin	+13.5%	+15.0%	+9.5%	+7 (+0.1x)
ResNet-18	FPrec	74.1%	88.7%	1x	1814 (13.5x)
	WBin (BWN)	73.4%	89.3%	31.2x	1030 (7.7x)
	FBin (XNOR)	68.8%	82.8%	31.2x	**134 (1x)**
	Hybrid	**73.8%**	**87.9%**	**31.2x**	359 (2.7x)
Increase	Hybrid vs FBin	+5.0%	+5.1%	-	+225 (+1.7x)
Sketch-A-Net	FPrec	72.9%	85.9%	1x	1135 (12.3x)
Squeezenet	FPrec	71.2%	86.5%	8x	610 (6.6x)
Squeezenet	WBin	66.7%	81.1%	23.7x	412 (4.5x)
Squeezenet	FBin	56.8%	66.0%	23.7x	**92 (1x)**
Squeezenet	Hybrid	**64.8%**	**79.6%**	**23.7x**	164 (1.8x)
Improvement	Hybrid vs FBin	+8.0%	+13.6%	-	+72 (+0.8x)

Comparisons: Imagenet

Imagenet: Accuracy improvements

- Hybrid AlexNet: 4.9%
- Hybrid ResNet-18: 3.6%

Model	Method	Accuracy		Mem. Savings	FLOPs
		Top-1	Top-5		
AlexNet	FPrec	57.1%	80.2%	1x	1135 (9.4x)
	WBin (BWN)	56.8%	79.4%	10.4x	780 (6.4x)
	FBin (XNOR)	43.3%	68.4%	10.4x	**121 (1x)**
	Hybrid-1	48.6%	72.1%	10.4x	174 (1.4x)
	Hybrid-2	**48.2%**	**71.9%**	**31.6x**	174 (1.4x)
Increase	Hyb. vs FBin	+4.9%	+3.5%	+21.2x	+53 (+0.4x)
ResNet-18	FPrec	69.3%	89.2%	1x	1814 (13.5x)
	WBin (BWN)	60.8%	83.0%	13.4x	1030 (7.7x)
	FBin (XNOR)	51.2%	73.2%	13.4x	**134 (1x)**
	Hybrid-1	54.9%	77.9%	13.4x	359 (2.7x)
	Hybrid-2	**54.8%**	**77.7%**	**31.2x**	359 (2.7x)
Increase	Hyb. vs FBin	+3.6%	+4.5%	+17.8x	+225(+1.7x)

Comparison with other approaches: Imagenet

- Outperforms all binary/ternary networks, while preserving maximal compression.

AlexNet

- 1.2% increase over DoReFa-Net (2-bit a)
- 1.6% higher accuracy than HTCBN

ResNet-18

- 1.2% higher than HTCBN

While preserving maximal compression & speedup rates!

Technique	Acc-Top1	Acc-Top5	W/I	Mem	FLOPs
AlexNet					
BNN	39.5%	63.6%	1/1	32x	121 (1x)
XNOR	43.3%	68.4%	1/1	10.4x	**121 (1x)**
Hybrid-1	48.6%	71.7%	1/1	10.4x	174 (1.4x)
Hybrid-2	**48.2%**	**71.5%**	1/1	**31.6x**	174 (1.4x)
HTCBN	46.6%	71.1%	1/2	31.6x	780 (6.4x)
DoReFa-Net	47.7%	-	1/2	10.4x	780 (6.4x)
Res-Net 18					
BNN	42.1%	67.1%	1/1	32x	134 (1x)
XNOR	51.2%	73.2%	1/1	13.4x	**134 (1x)**
Hybrid-1	54.9%	77.9%	1/1	13.4x	359 (2.7x)
Hybrid-2	**54.8%**	**77.7%**	1/1	**31.2x**	359 (2.7x)
HTCBN	53.6%	-	1/2	31.2x	1030 (7.7x)

Other Experiments

Last Layer Binarization (For maximal compression)

On Sketch-A-Net and ResNet-18:

XNOR-Net: 11% and 1%

Ours: 1% and 0.1%

Model	BinType	Last Bin?	Acc	Mem
Sketch-A-Net	FBin (XNOR)	No	59.6%	19.7x
		Yes	48.3%	**29.2x**
Sketch-A-Net	Hybrid	No	73.1%	19.7x
		Yes	72.0%	**29.2x**
Resnet-18	FBin (XNOR)	No	69.9%	13.4x
		Yes	68.8%	**31.2x**
Resnet-18	Hybrid	No	73.9%	13.4x
		Yes	73.8%	**31.2x**

INDEX

1. Introduction

- Biological & practical purpose.
- Practical apps: Low memory, compute & power use.
- Related Works: Use 2-bit **w** and/or **a**

3. Hybrid Binary Networks

- Asked a nice question: "Where to quantize"
- Badly quantized layers <=> low compute layers!
- Hence, greatly reduce acc. drops at minimal cost

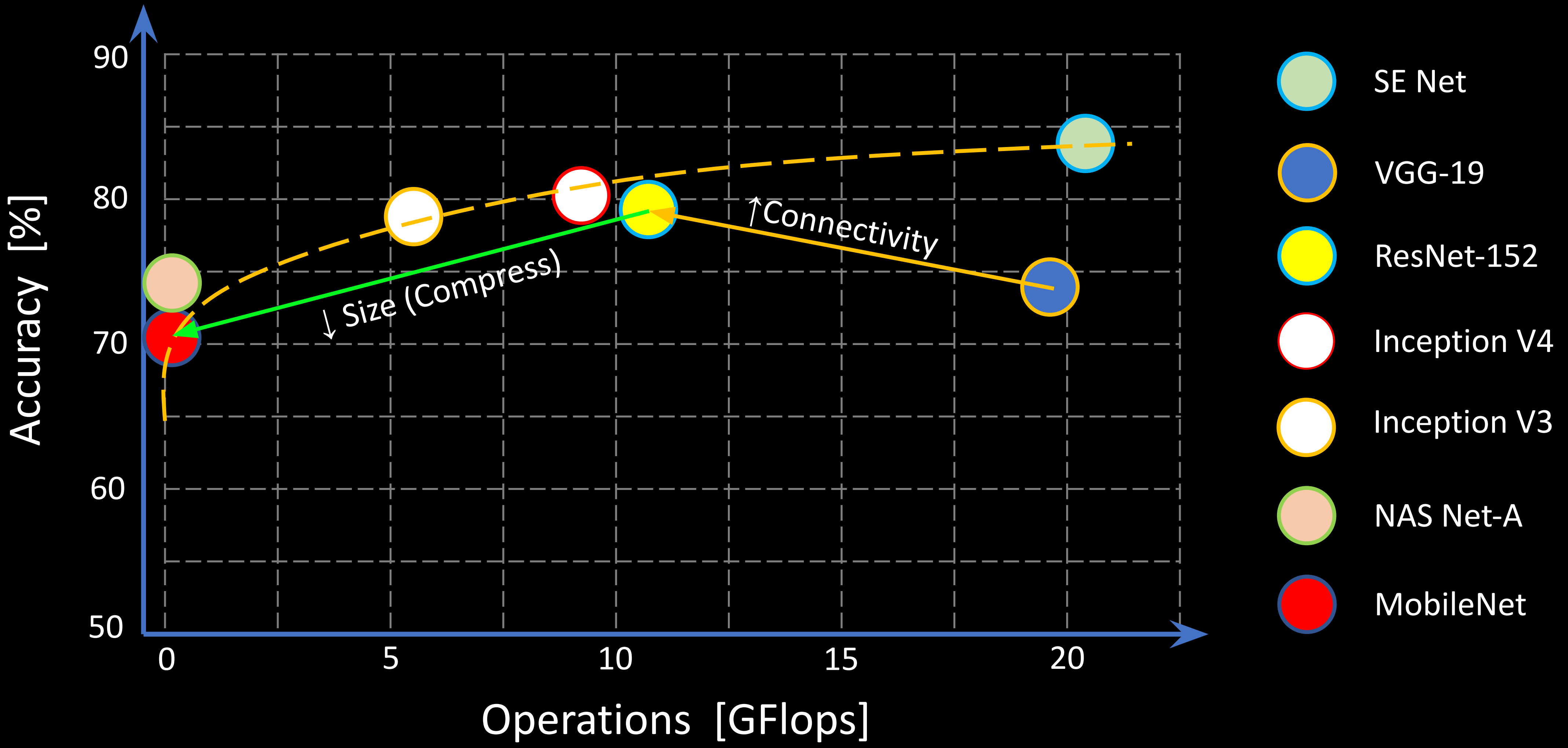
2. Distribution-aware Binarization

- As expressive as infinite precision.
- Projects to 2 weighted || vectors.
- "Optimal" binary rep., efficiently computable, trainable via SGD.

4. Deep Expander Networks

- Motivation for the Hypothesis
- What are expander graphs?
- What guarantees do we get?
- Experiments & Results

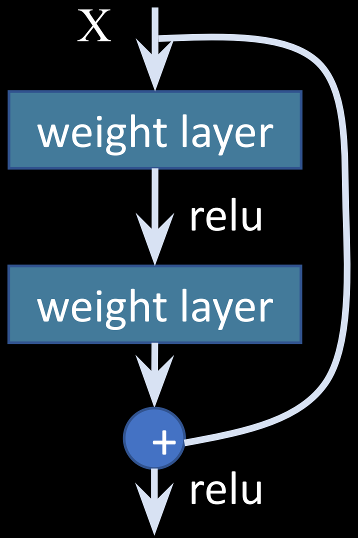
Efficient CNNs

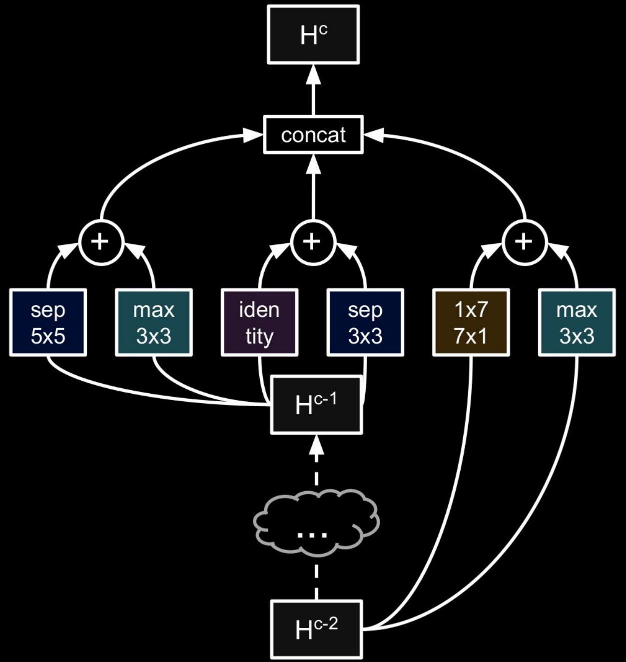


Approaches towards Efficiency

Better Network Design

- Inception Net
- NAS Net
- ResNet
- P-NAS Net

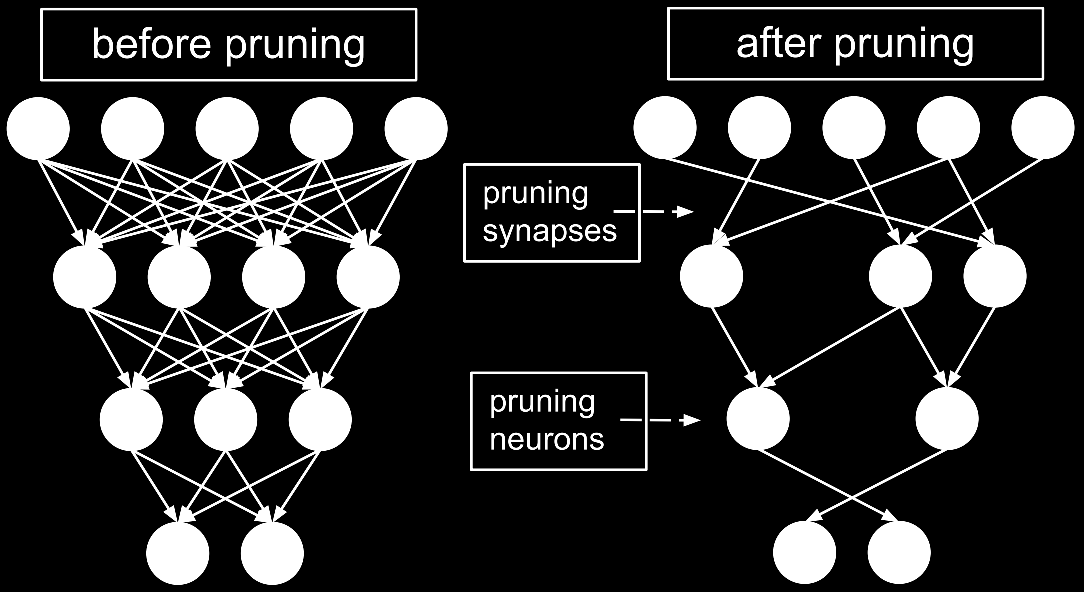




↑ Connectivity

Efficient Layer Modification

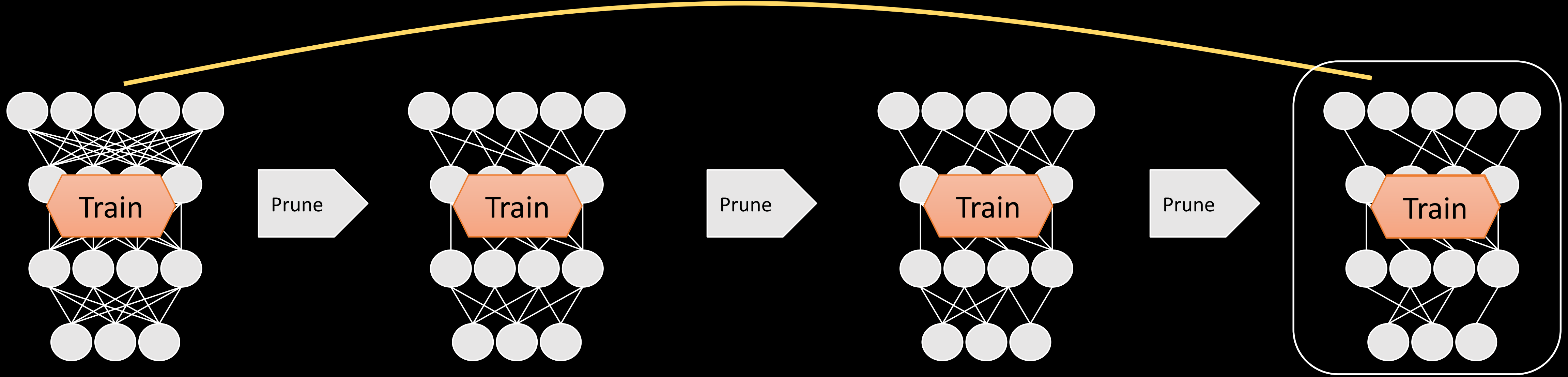
- Group Convolutions
- Pruning



↓ Size

Between Layer Connections: Train → Prune

Can We Prune → Train



Train → Prune

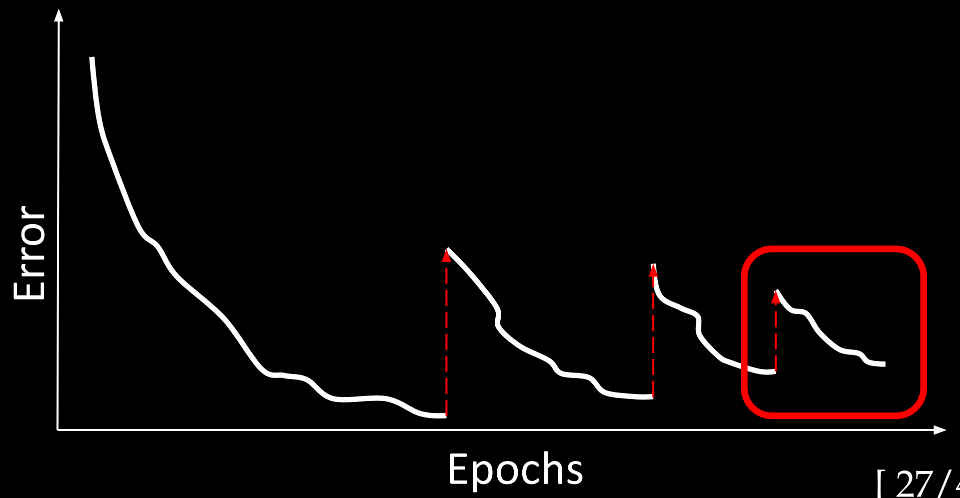
- ✘ Need to train full network
- ✘ Need Multiple trainings
- ✘ Layer structure specific to given data

Not Transferrable

Prune → Train

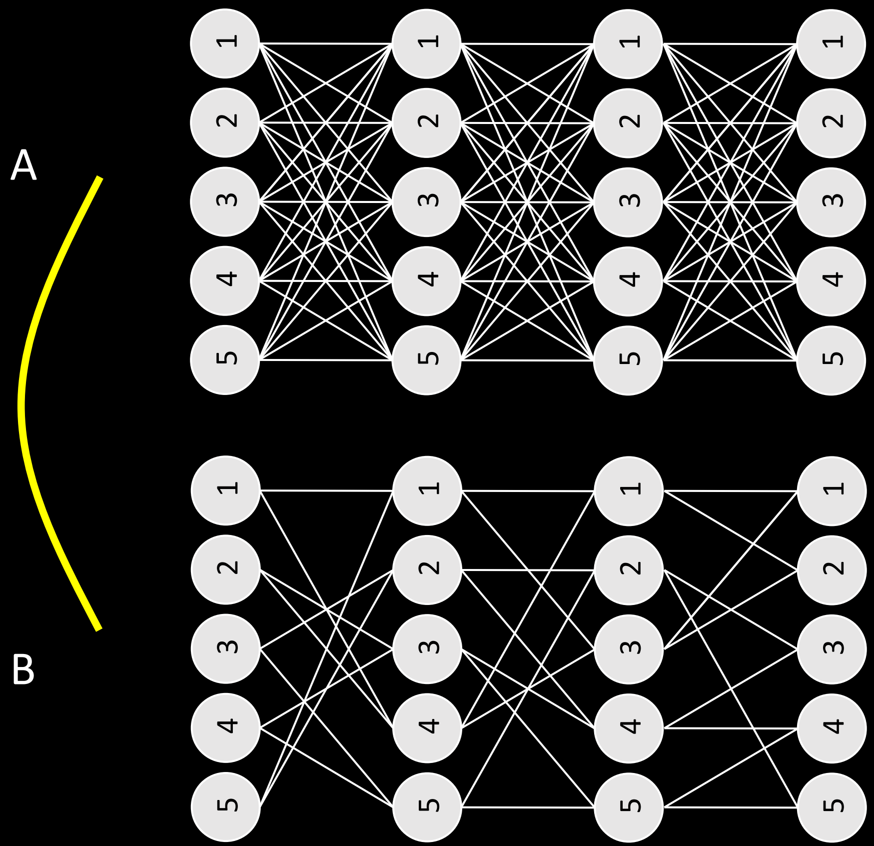
- ✓ Train a compact network
- ✓ Single training
- ✓ Generic layer structure, independent of data

Transferrable



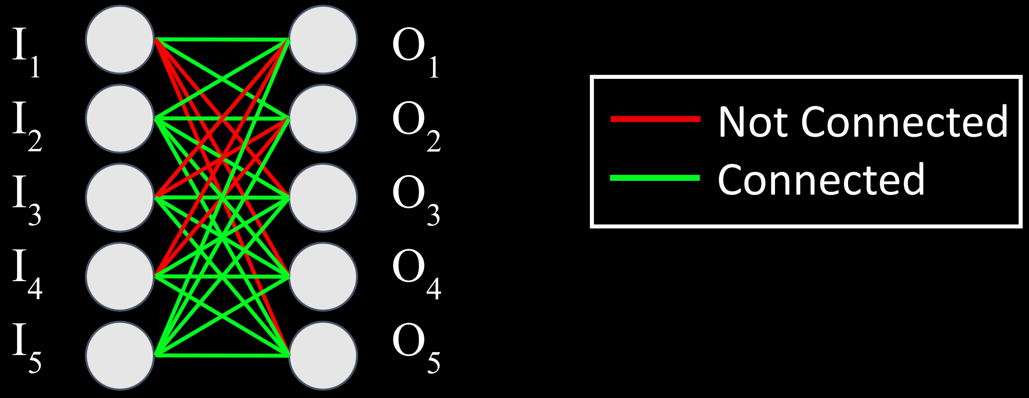
Pruning Without Data

- Need to sparsify connections
- Need to ensure multi-layer connectivity



Regular Pruning

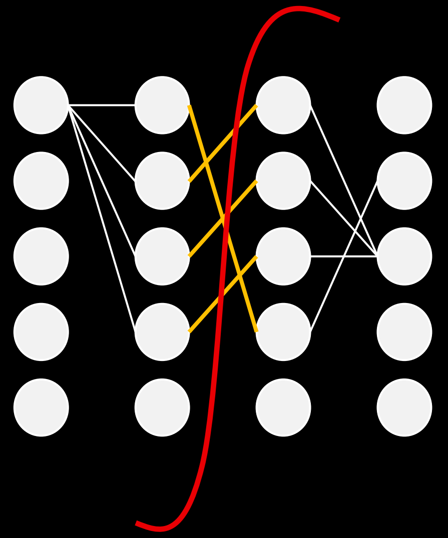
- Not well connected
- Retraining does not help



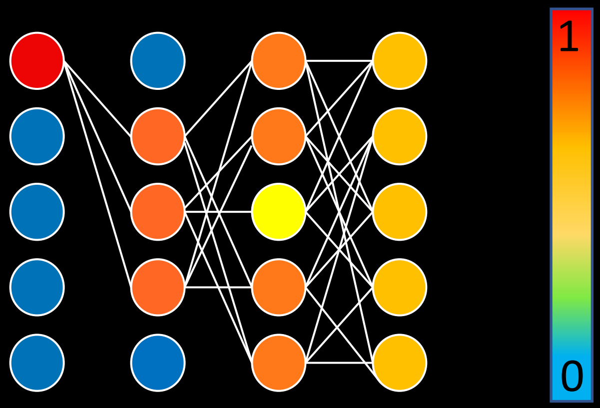
All this to be done without data!!

Expander Graphs

Combinatorics: Highly connected; Sever many edges to disconnect any large part of the graph



Probability: Random walk on these converges to its limiting distribution as rapidly as possible.

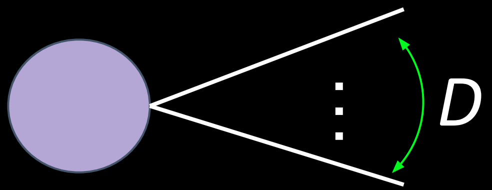


Algebra: First positive eigenvalue of their laplace operator is bounded away from zero.

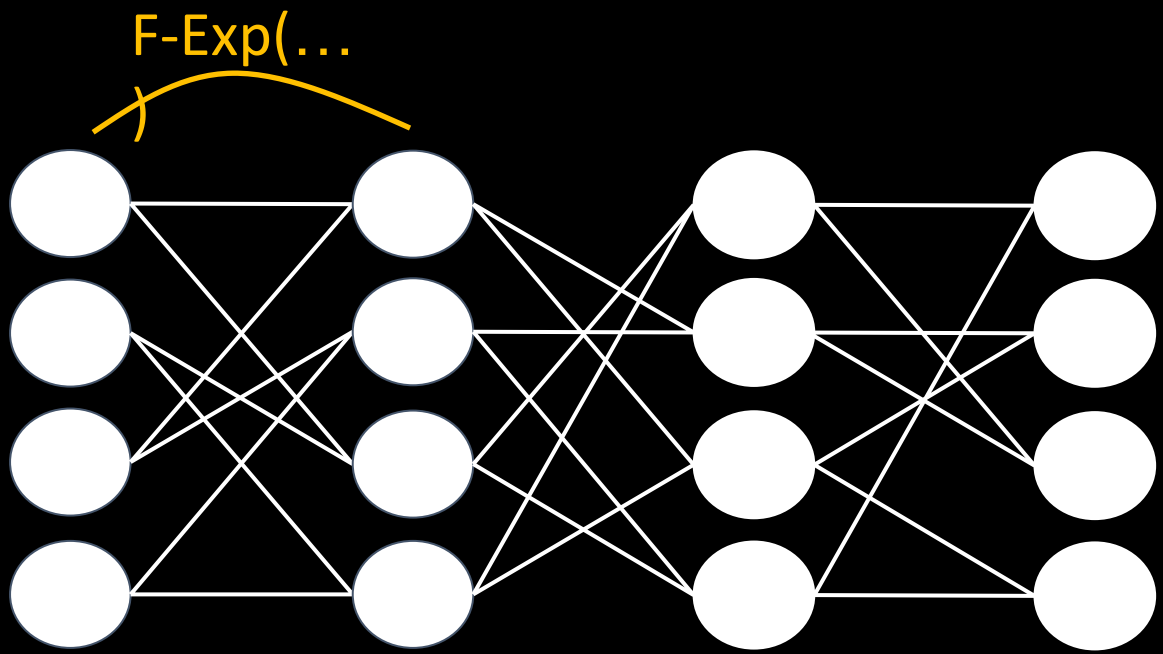
Large expansion →Large spectral gap

Expander Graph are are simultaneously sparse and highly connected.

Constructing Expander Layers

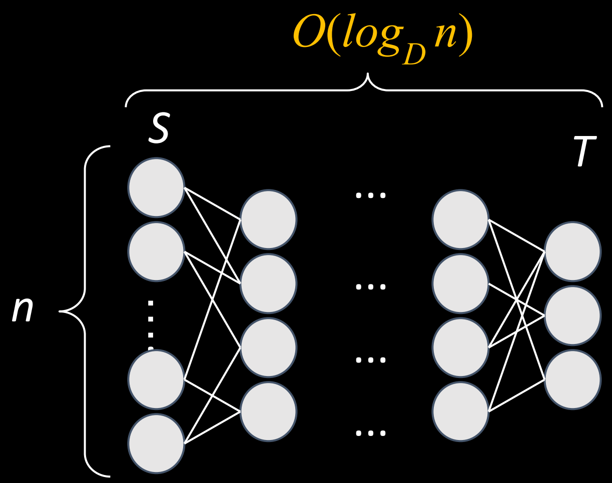


- Pick an input node
- Connect it to D random outputs
- Repeat for every input node
- Repeat for every layer

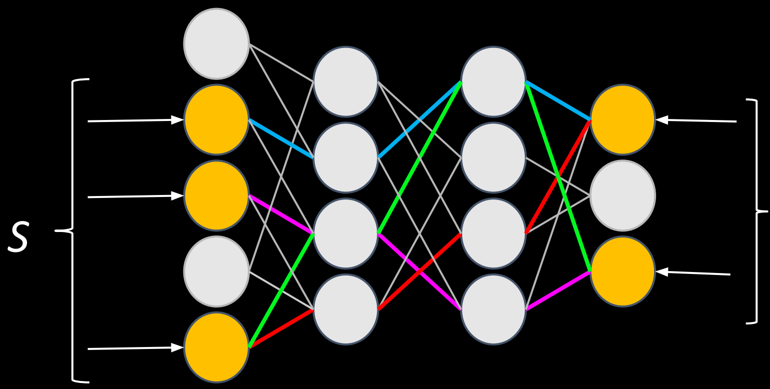


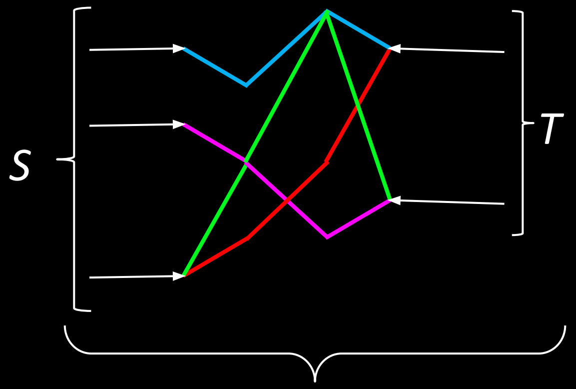
Guarantees About X-Nets

Theorem 1 (Sensitivity):
Let n be the number of input as well as output nodes in the network and G1 , G2 ,⋯ , Gt be D-regular bipartite expander graphs with n nodes on both sides. Then every output neuron is sensitive to every input in a Deep X-Net defined by G i 's with depth t = $O(\log_D n)$.



Theorem 2 (Rich Connectivity):
Let n be the number of input as well as output nodes in the network and G be D regular bipartite expander graph with n nodes on both sides. Let S,T be subsets of input and output nodes in the X-Net layer defined by G. The number of edges between S and T $|E| \approx \frac{D|S||T|}{n}$

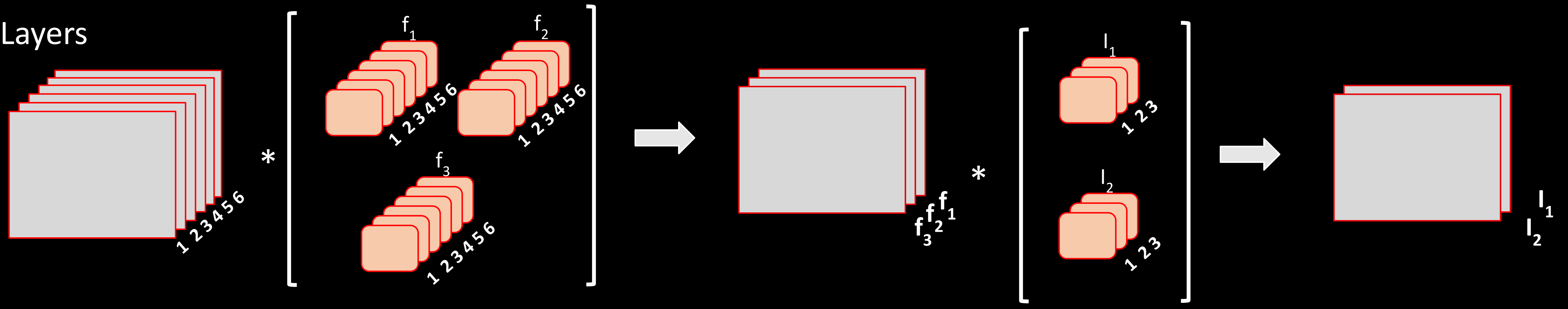




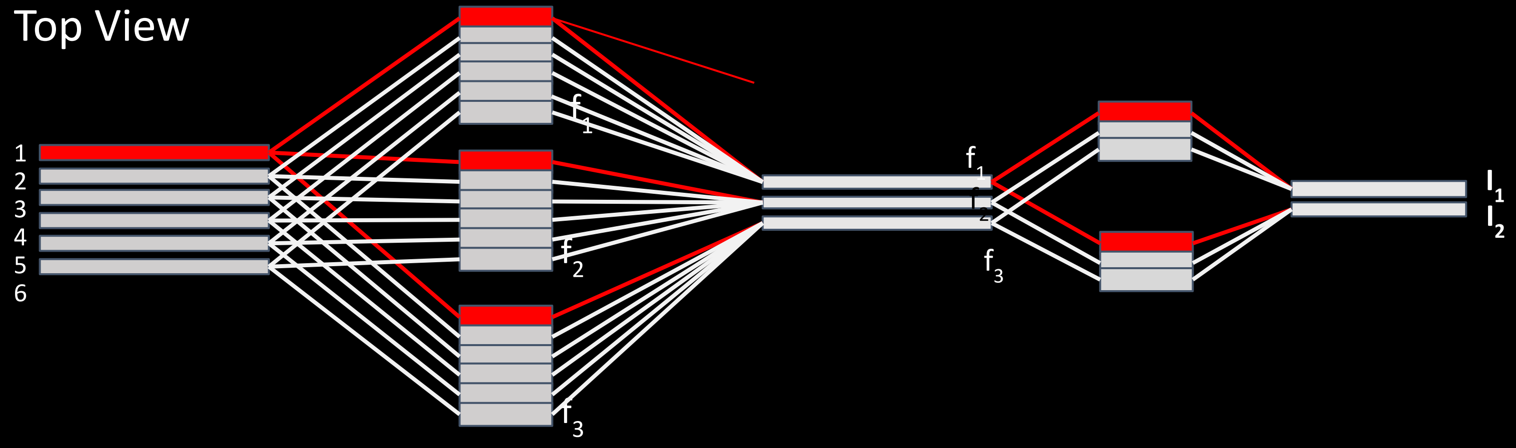
Lots of paths between any S and T

Note: Connectivity Graph of Convolutions

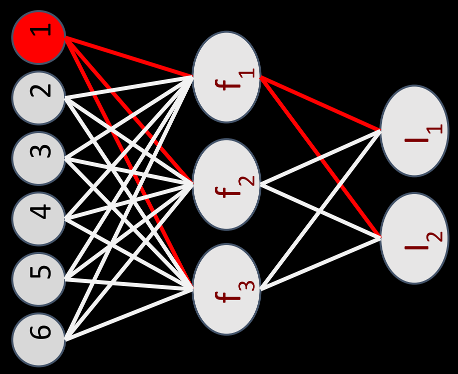
Conv Layers



Top View

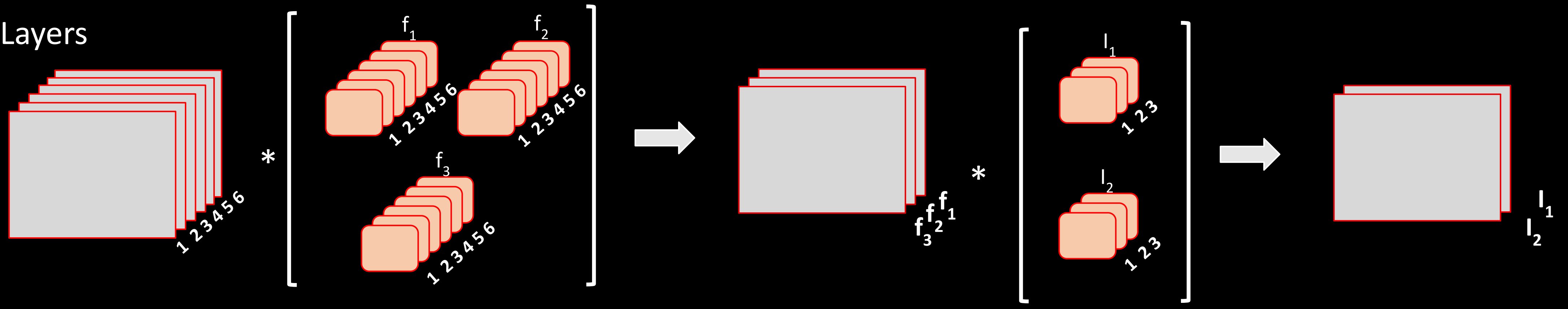


Connectivity Graph

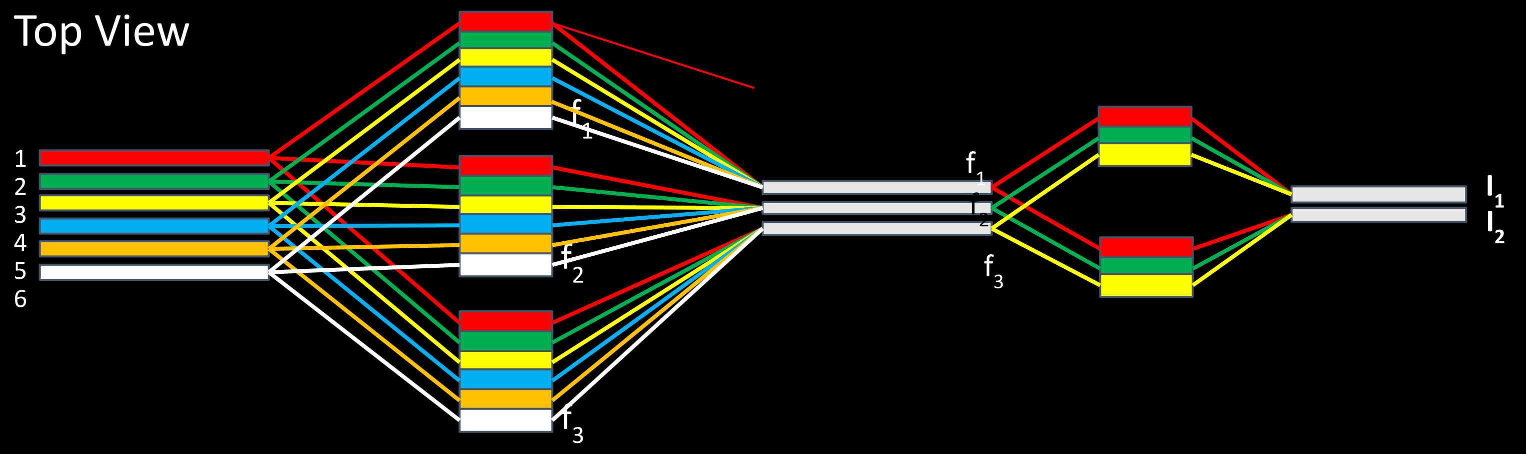


Note: Connectivity Graph of Convolutions

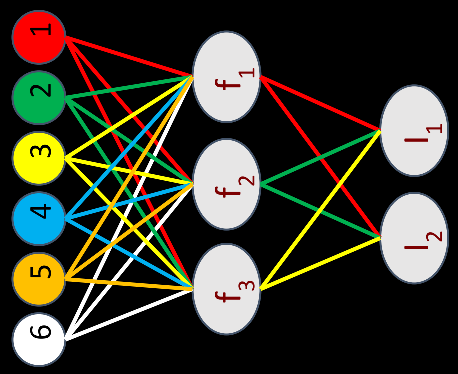
Conv Layers



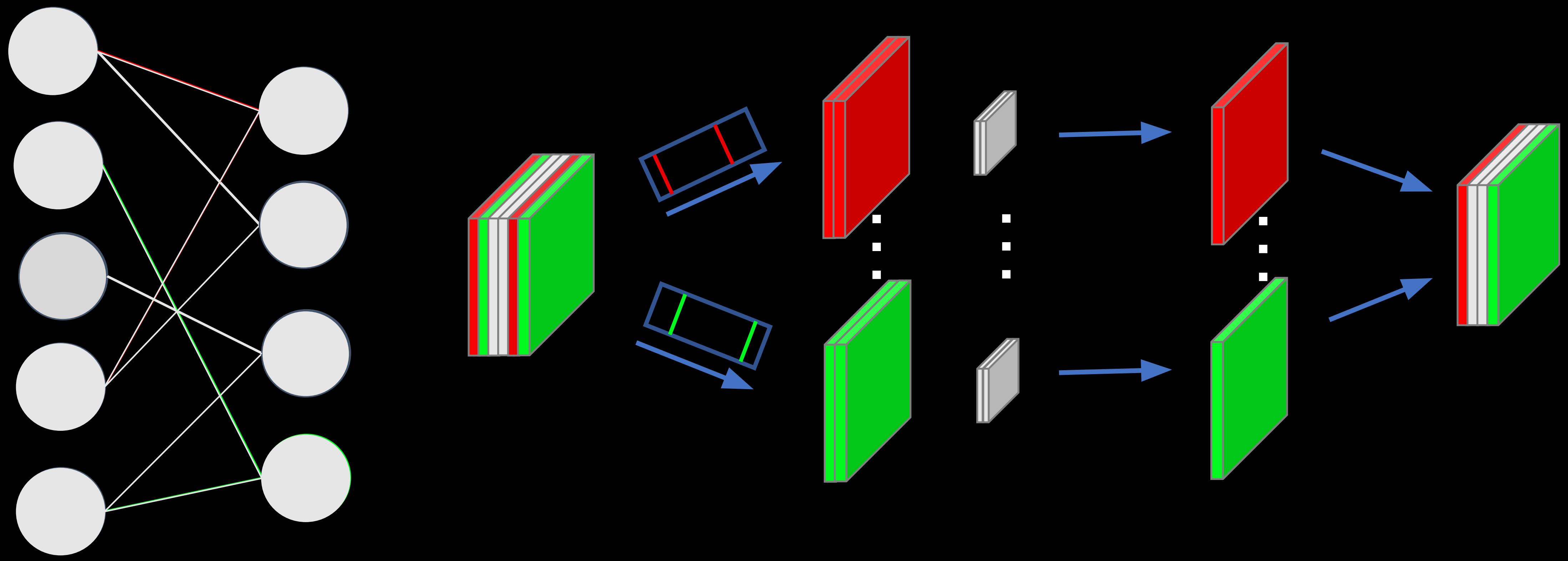
Top View



Connectivity Graph

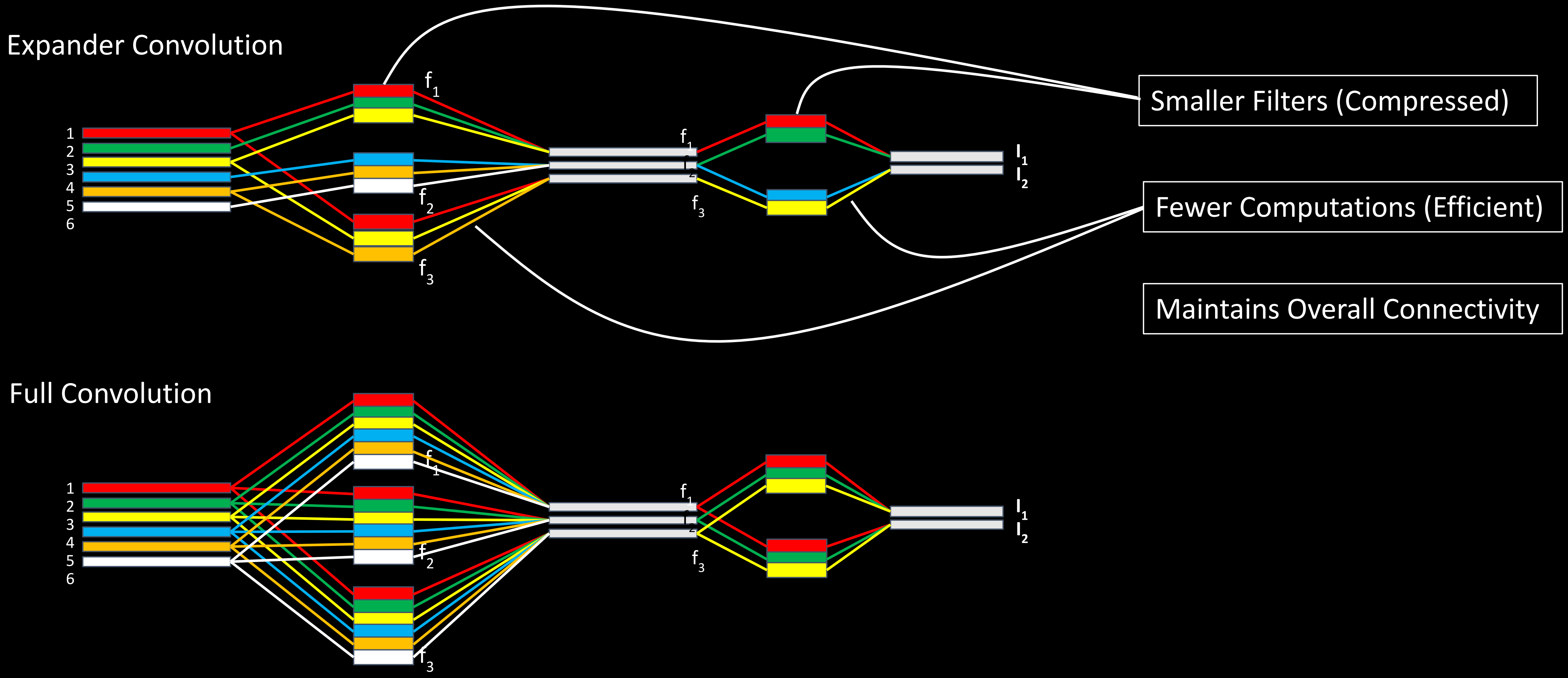


Our Convolutional Layer



Red and green represent the subsets that are connected

EXPANDER VS. FULL CONVOLUTION



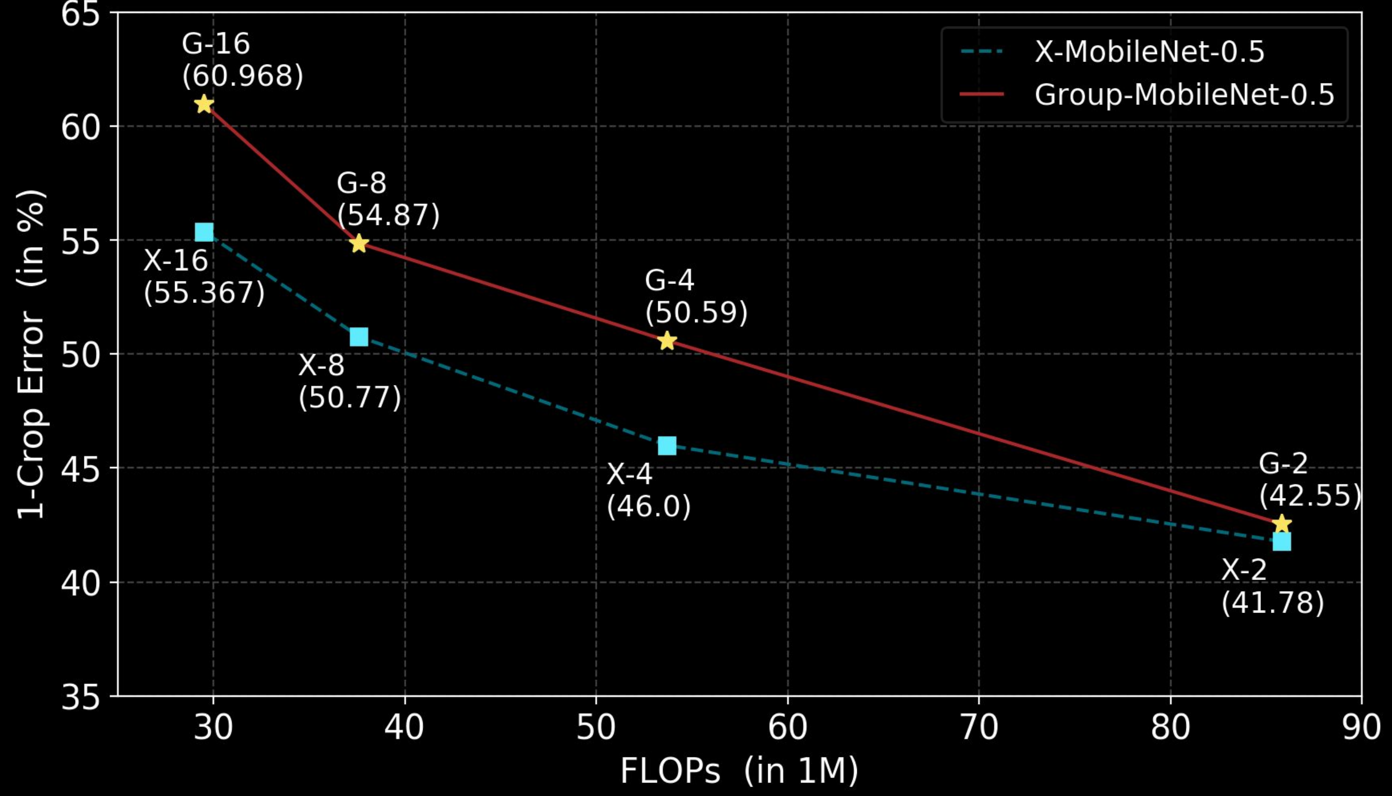
Experimental Results

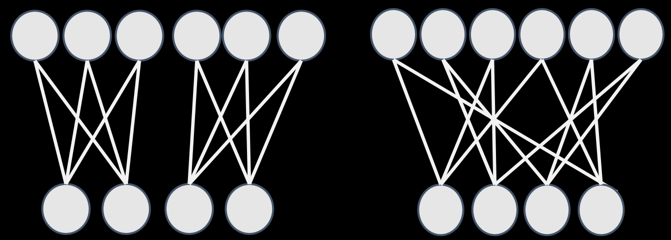
Comparisons with:

- Layer Connectivity Graphs: Group Convolution
- Network Compression: Pruning
- Efficient Architectures: ResNet and DenseNet

Benchmarking with Group Convolution

X-Conv beats G-Conv by ~ 4-5%
on a compact MobileNet-0.5 on Imagenet





Compression	G-Conv	X-Conv (Ours)	Err. Red.
2x	42.55%	41.78%	**0.8%**
4x	50.59%	46.00%	**4.6%**
8x	54.87%	50.77%	**4.1%**
16x	60.97%	55.37%	**5.6%**

Comparison with Pruning

VGG-16 on CIFAR-10

Method	Accuracy	# Params
Li et al.	93.4 %	5.4 M (2.8x)
NW Slimming	93.8 %	2.3 M (6.5x)
X-VGG 16-1	**93.4 %**	**1.65 M (9x)**
X-VGG 16-2	**93.0 %**	**1.15 M (13x)**
VGG-16 Orig	94.0 %	15.0 M (1.0x)

X-Nets are as compressible as the best pruning techniques

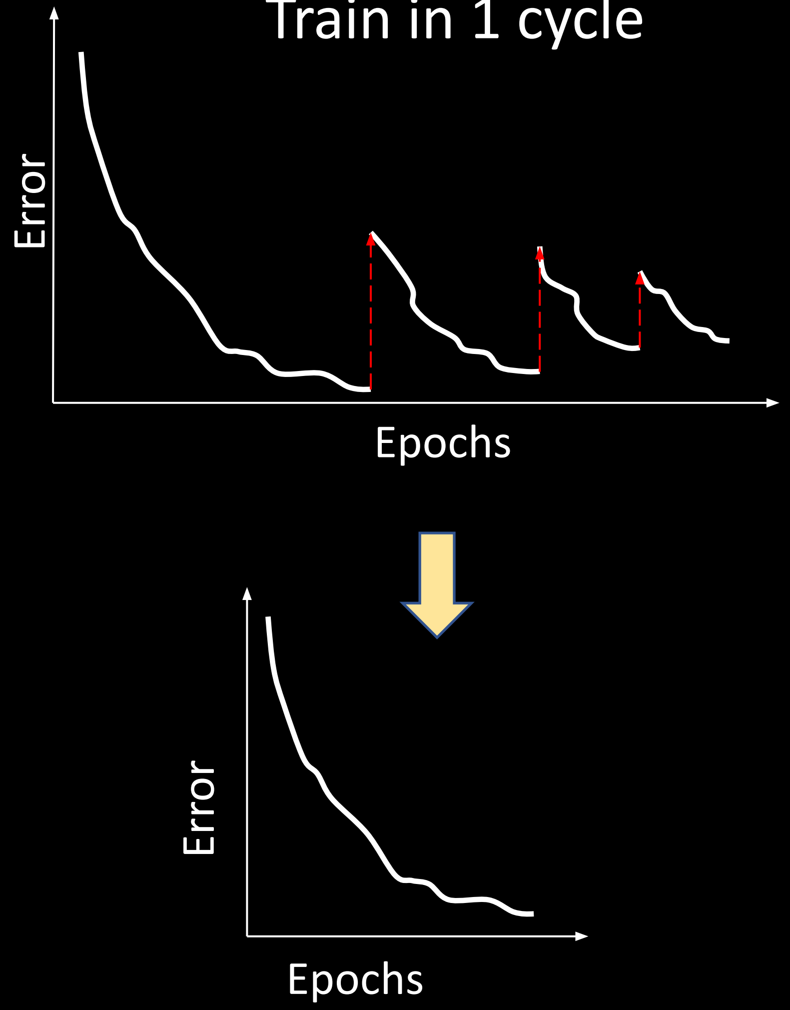
AlexNet on ImageNet

Method	Accuracy	# Params
Collins et al.	55.1 %	15.2 M (4x)
Zhou et al.	54.4 %	14.1 M (4.3x)
Han et al.	57.2 %	6.7 M (9.1x)
Srinivas et al.	56.9 %	5.9 M (10.3x)
Guo et al.	**56.9 %**	**3.4 M (18x)**
X-AlexNet-1	**55.2 %**	**7.6 M (8x)**
X-AlexNet-2	**56.2 %**	**9.7 M (6.3x)**
AlexNet-Orig	57.2 %	61 M (1.0x)

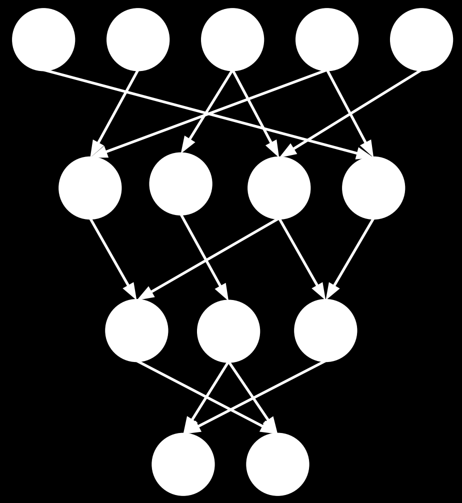
Failure Case ?

Advantages over Pruning

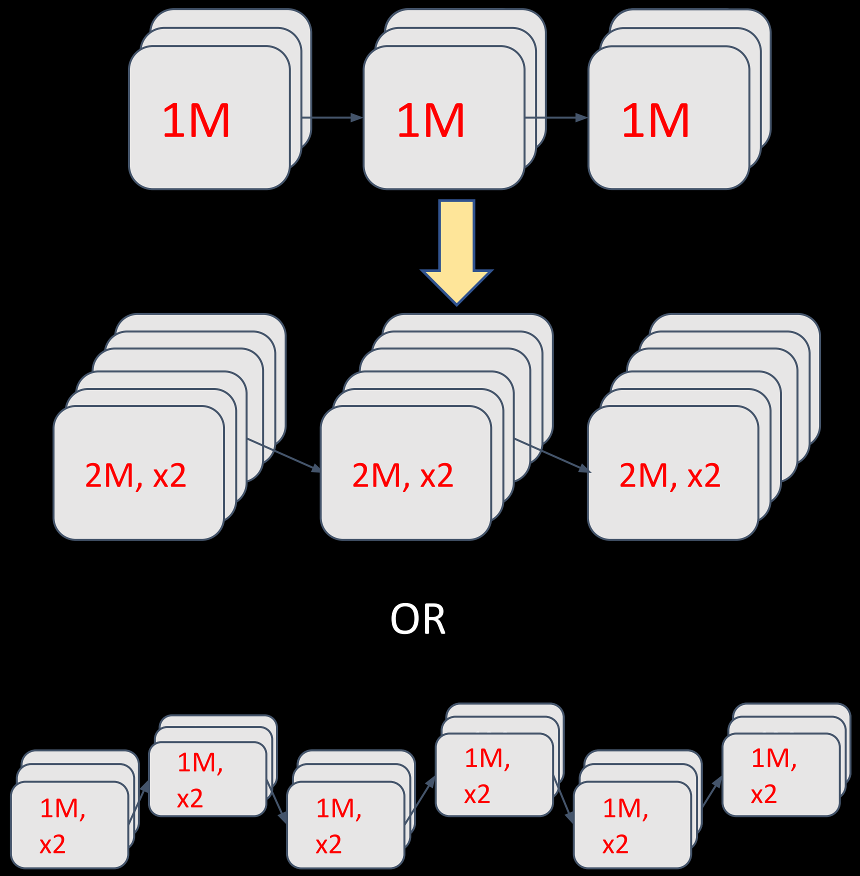
Train in 1 cycle



Transferable Architectures

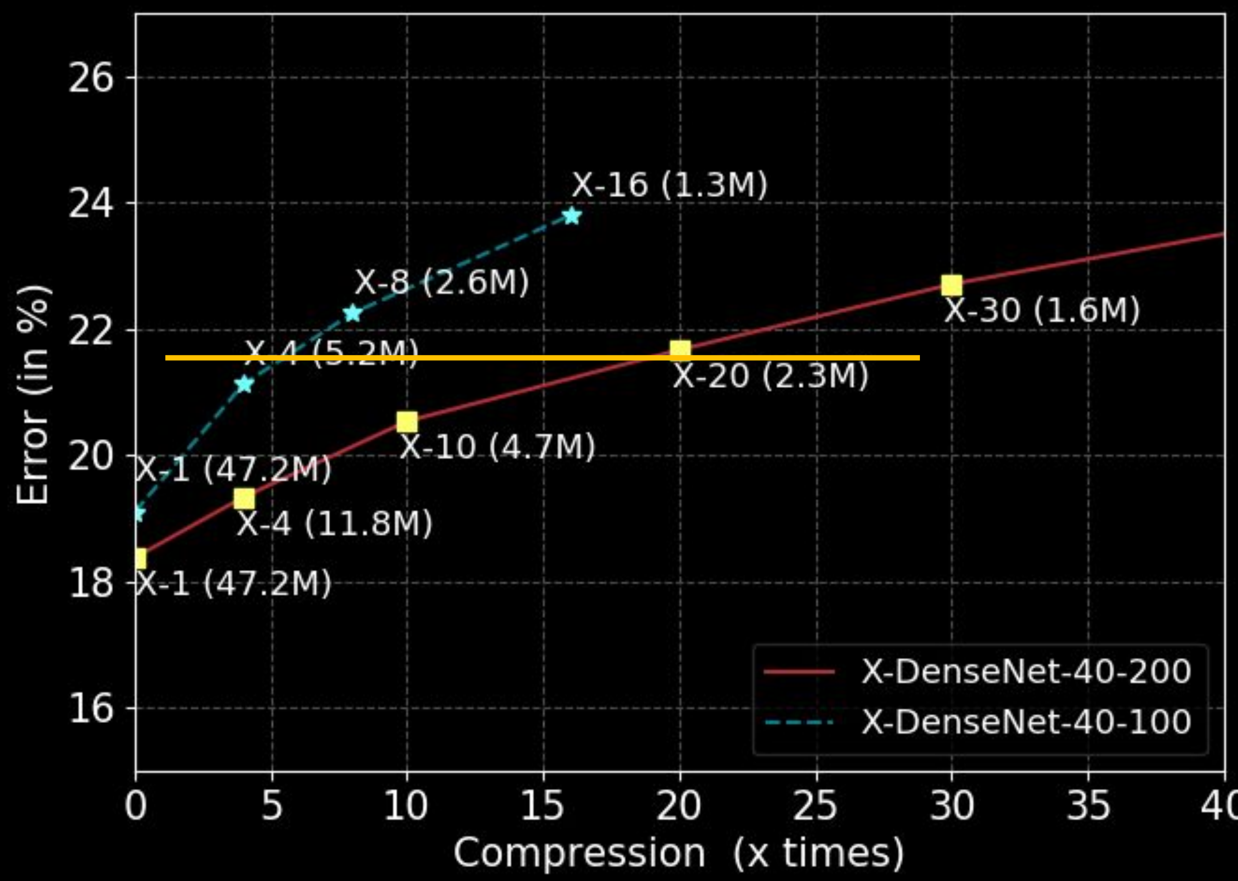


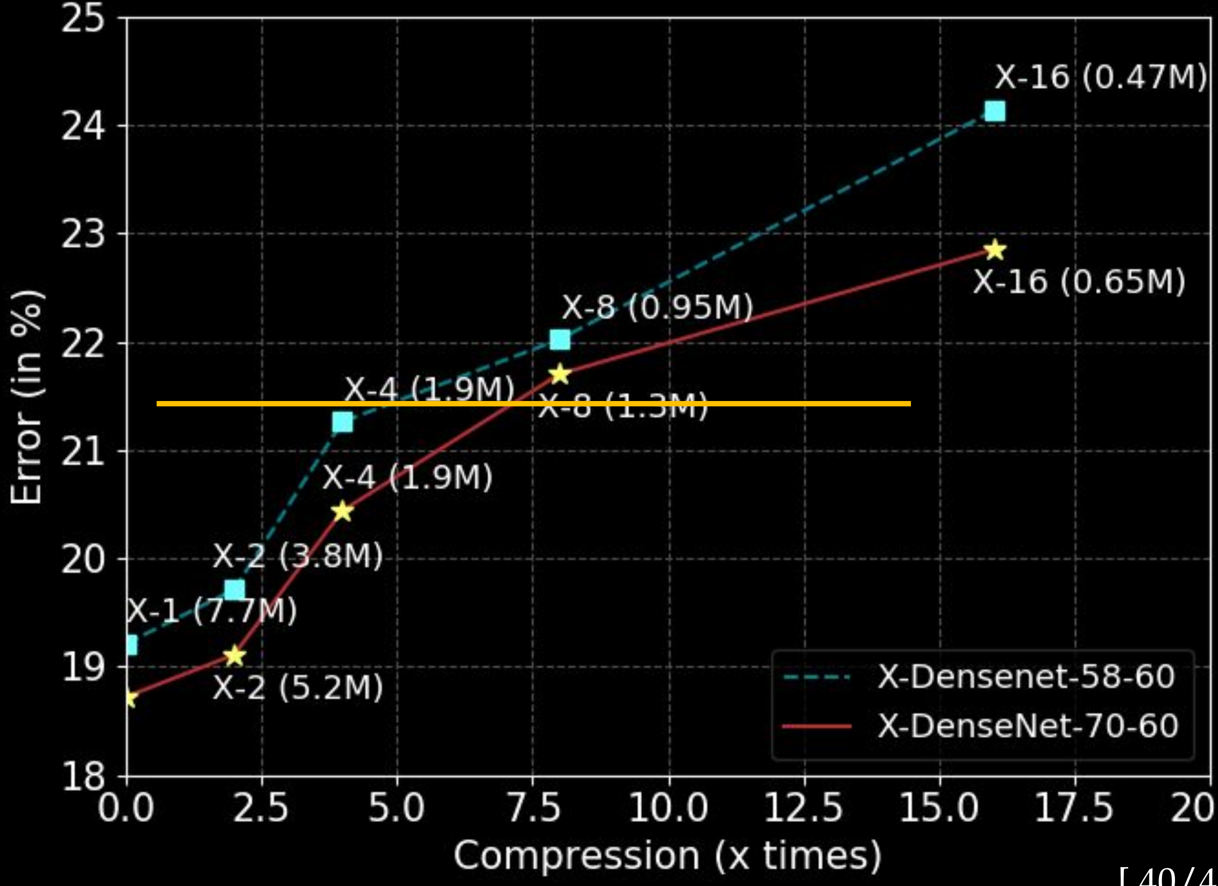
Go Wider / Deeper



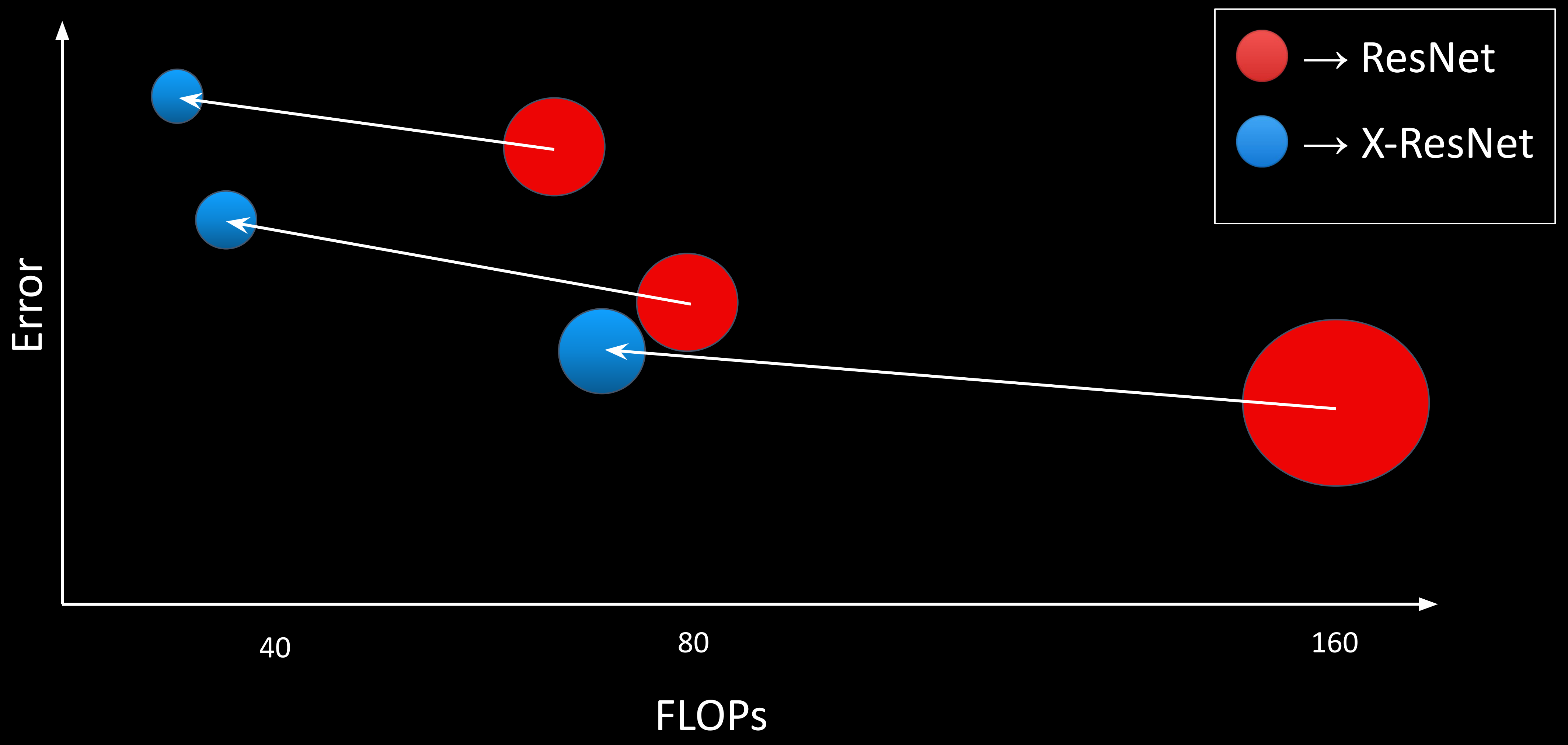
Going Wider and Deeper

Wider/Deeper networks with higher compression achieves same error rate with fewer parameters

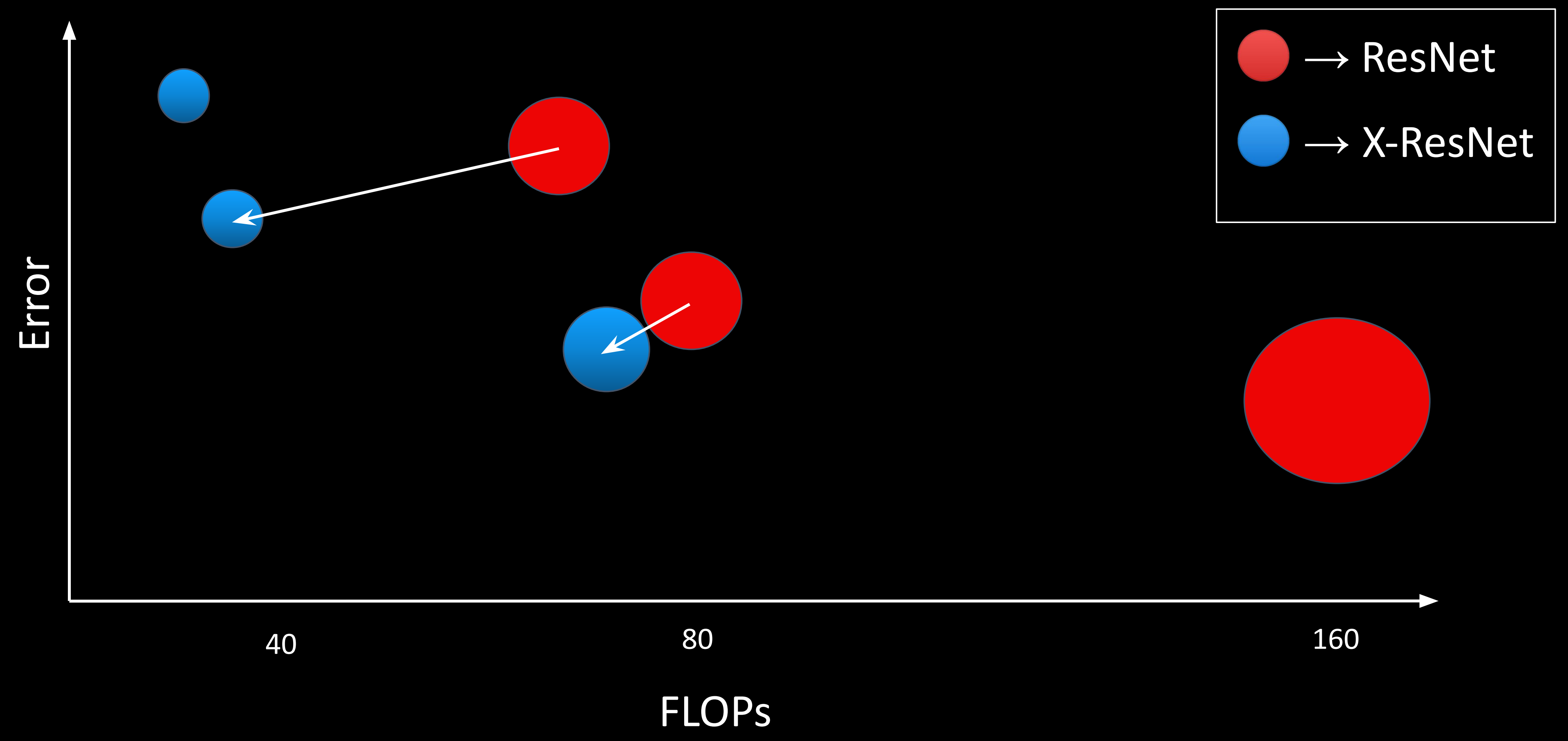




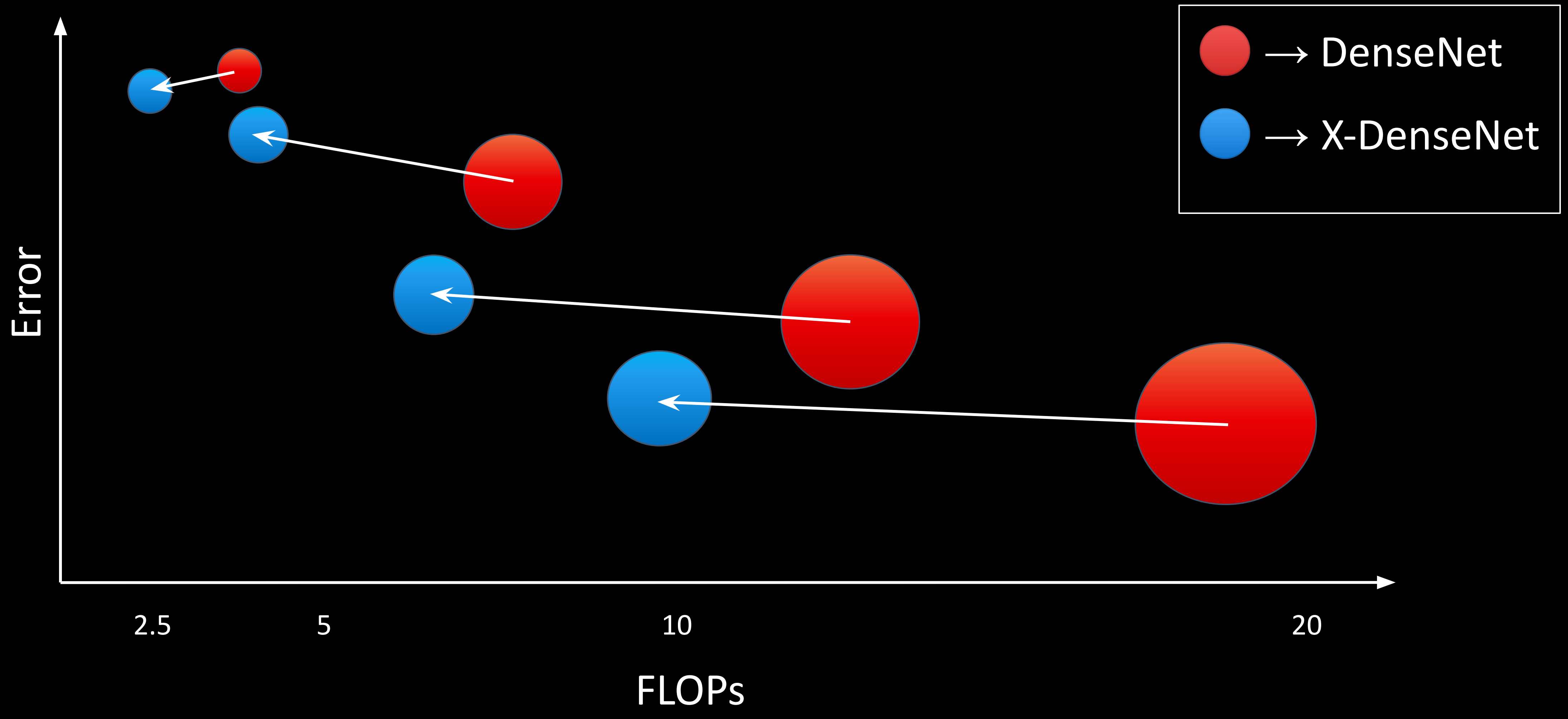
ResNet vs X-ResNet on ImageNet



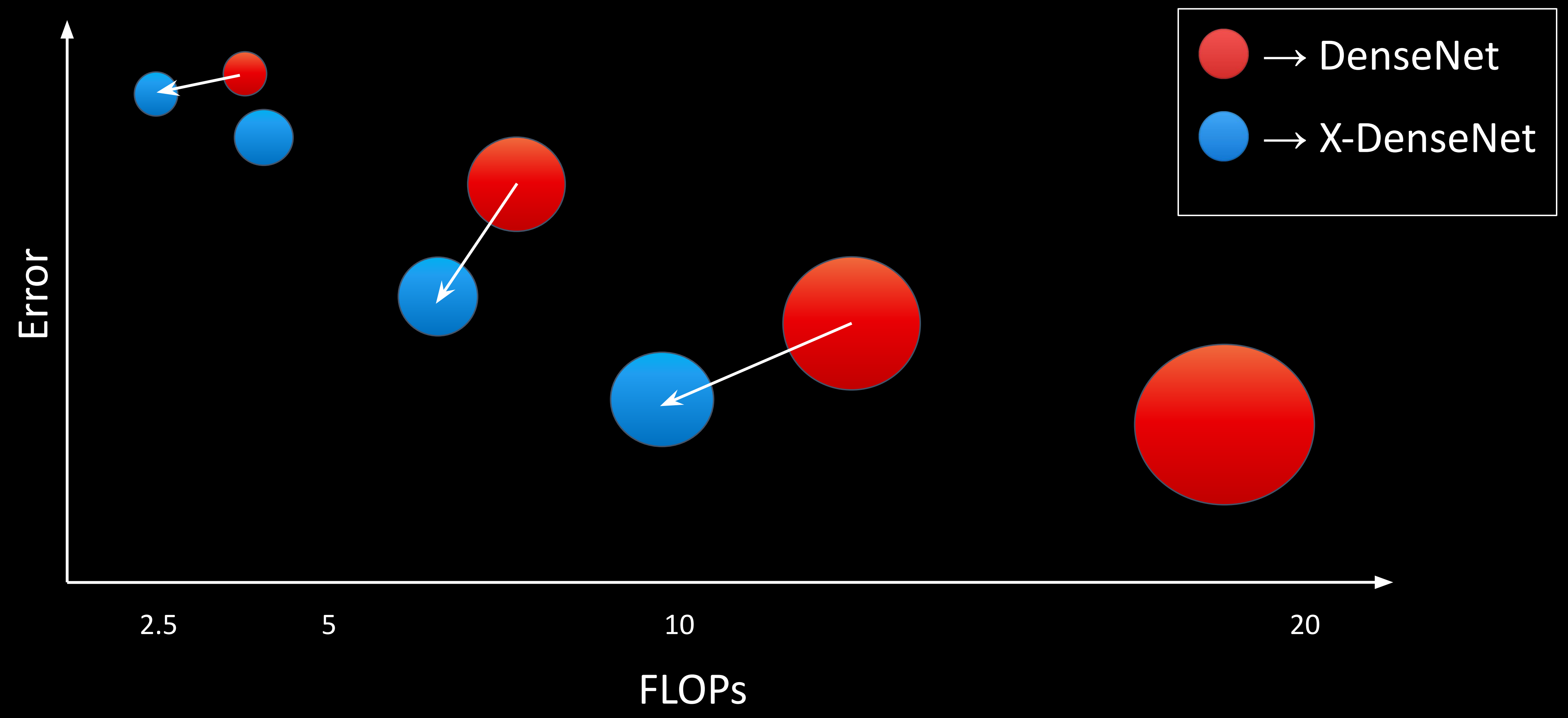
ResNet vs X-ResNet on ImageNet



DenseNet vs X-DenseNet on CIFAR-10



DenseNet vs X-DenseNet on CIFAR-10



SUMMARY

1. Introduction

- Biological & practical purpose.
- Practical apps: Low memory, compute & power use.
- Related Works: Use 2-bit **w** and/or **a**

3. Hybrid Binary Networks

- Asked a nice question: “Where to quantize”
- Badly quantized layers <=> low compute layers!
- Greatly reduce acc. drops at minimal cost.

2. Distribution-aware Binarization

- As expressive as infinite precision.
- Projects to 2 weighted || vectors.
- “Optimal” binary rep., efficiently computable, trainable via SGD.

4. Deep Expander Networks

- Principled way to prune deep networks.
- Allows training of wider and deeper networks.
- Highlights the use of global connectivity analysis in network architecture design

Thank You

Using our pytorch code to reproduce all results:

```
from layers import WBinConv2d, WBinLinear, …,  ExpanderLinear, ExpanderConv2d
```

nn.Conv2d(...) $\longrightarrow$ WBinConv2d(...)

nn.Conv2d(...) $\longrightarrow$ FBinConv2d(...)

nn.Conv2d(...) $\longrightarrow$ ExpanderConv2d(...)

nn.Linear(...) $\longrightarrow$ WBinLinear(...)

nn.Linear(...) $\longrightarrow$ FBinLinear(...)

nn.Linear(...) $\longrightarrow$ ExpanderLinear(...)

GitHub Repos: https://github.com/DrImpossible/Deep-Expander-Networks
https://github.com/Einsteino/Binary-Compression (Coming very soon)