Speeding up Deep Neural Networks with Adaptive Computation and Efficient Multi-Scale Architectures

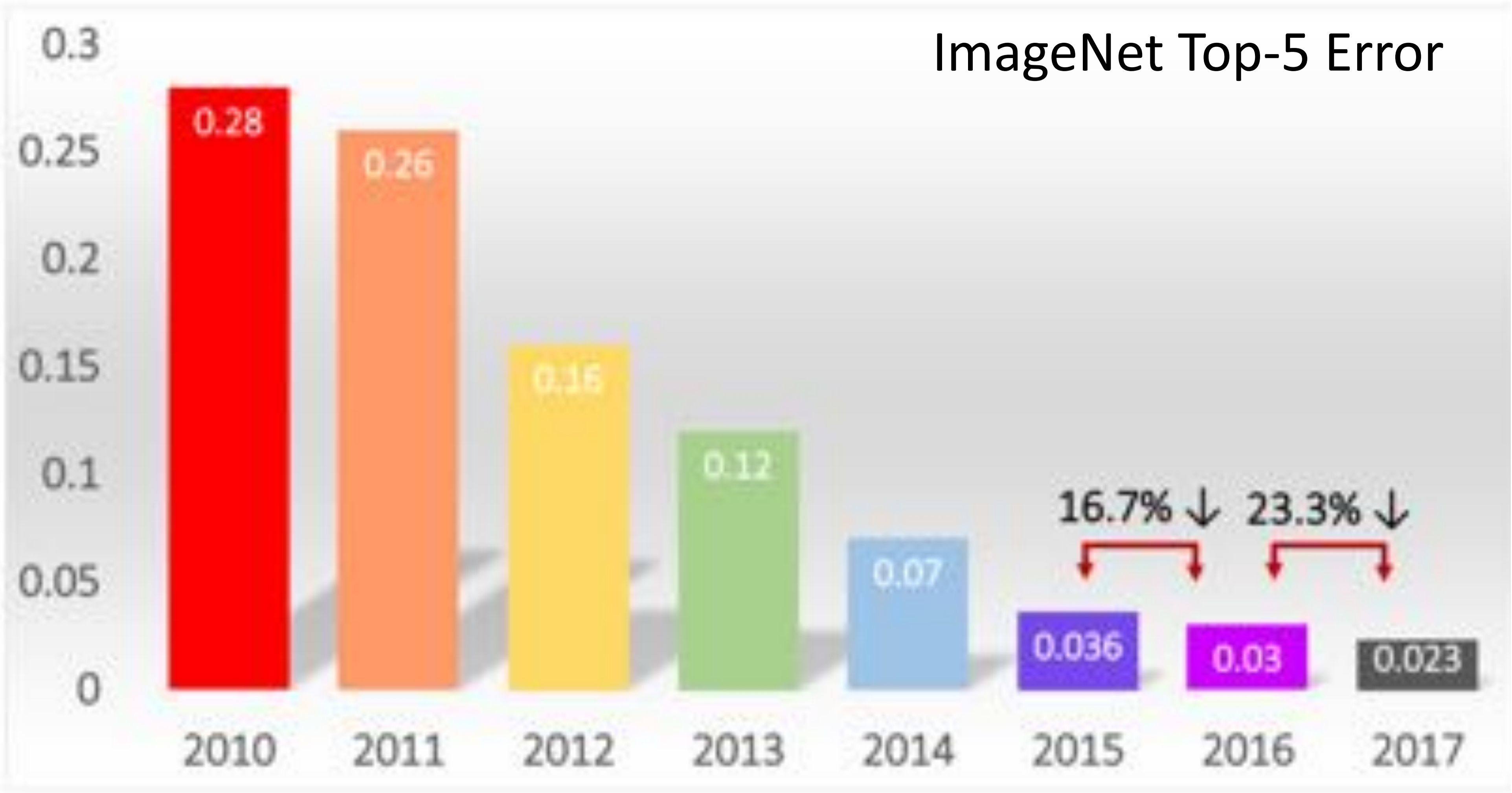
Rogerio Schmidt Feris

Principal Research Scientist and Manager

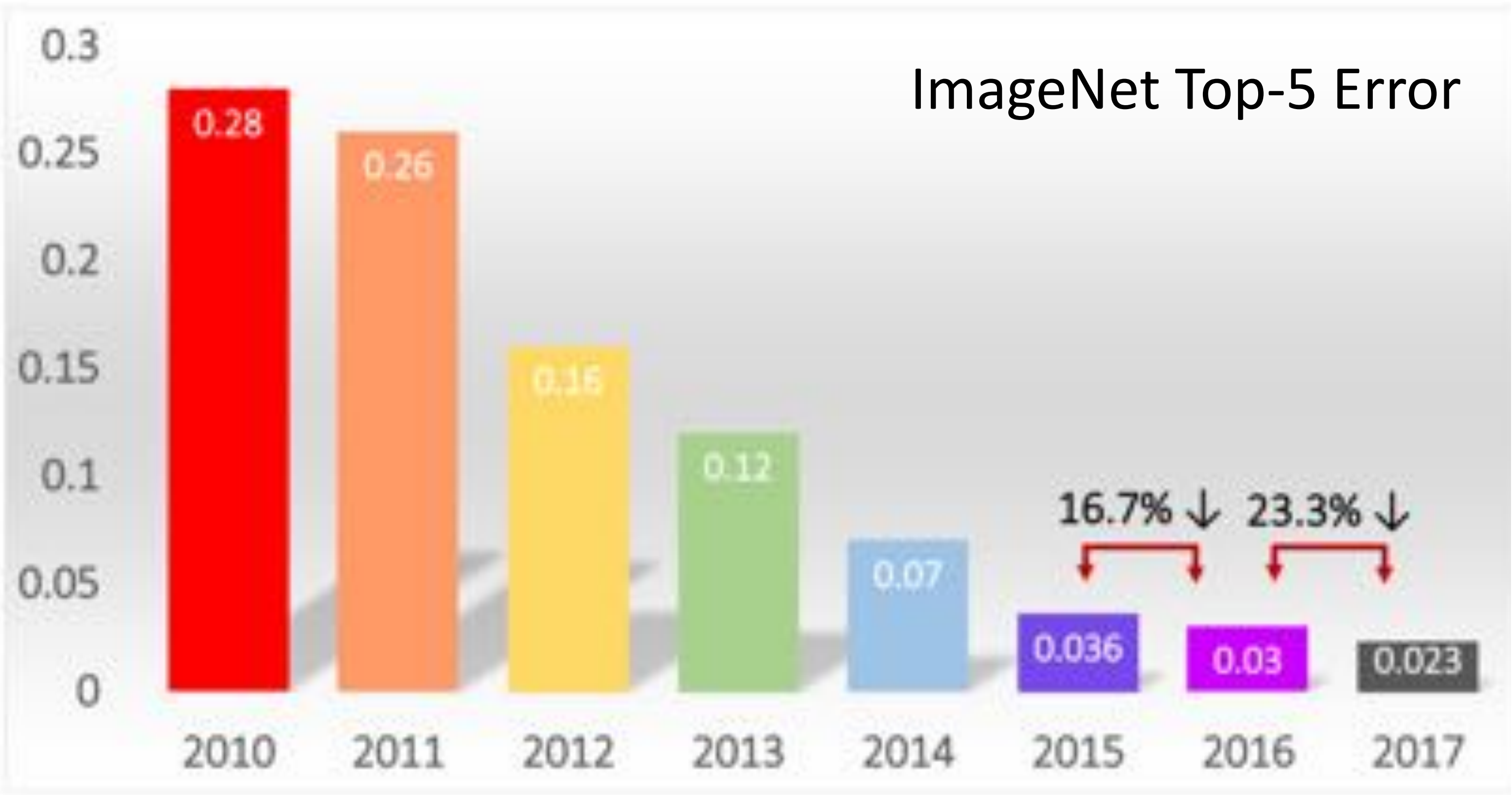
IBM Research & MIT-IBM Watson AI Lab

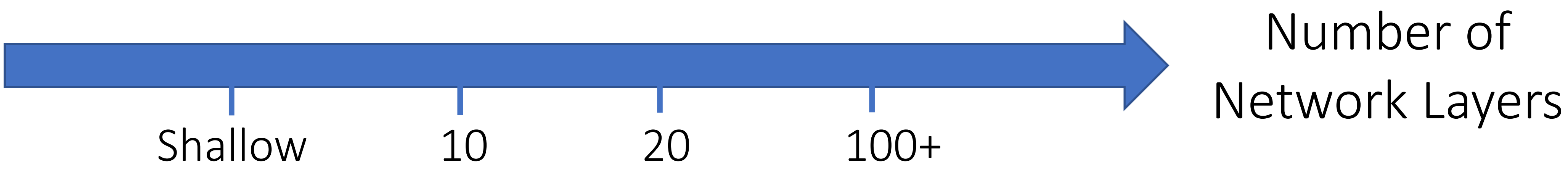
I thought I would die without seeing…

… these results!



Better Results → More Complexity

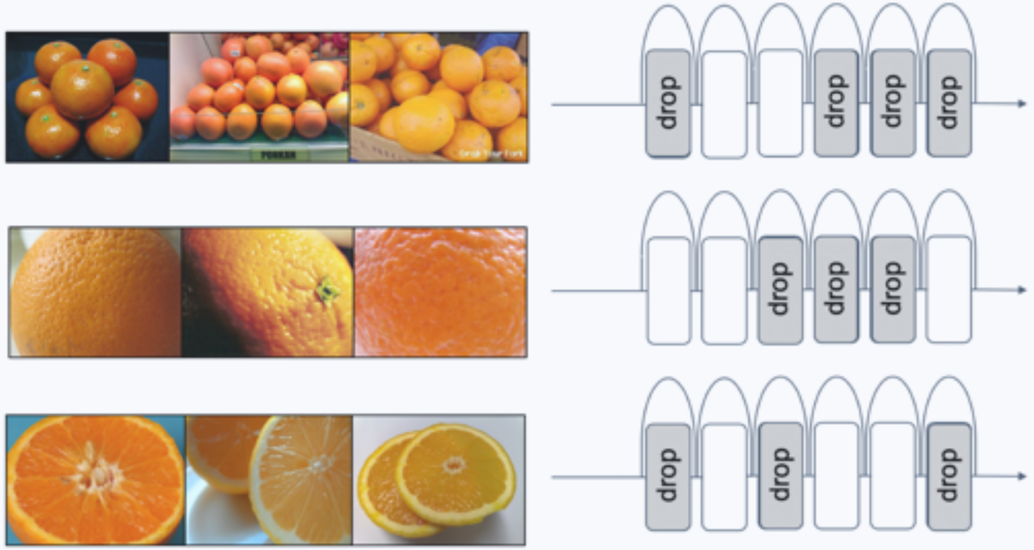




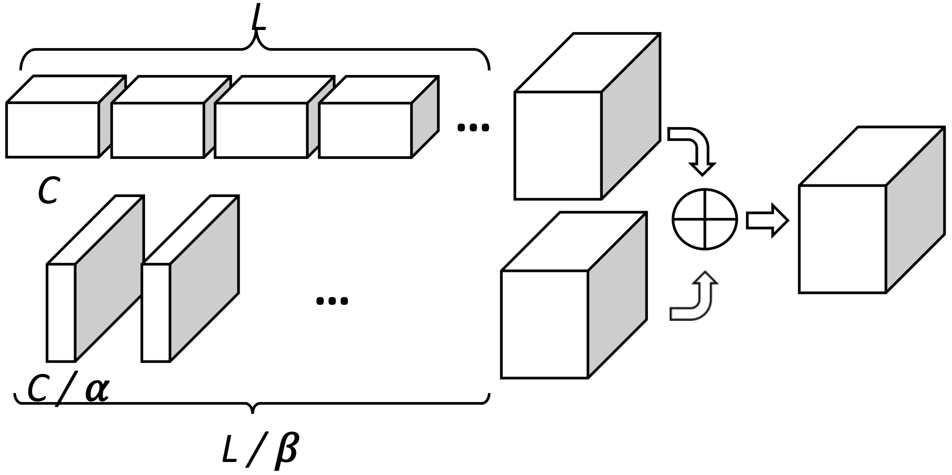
Many applications require real-time inferencing

This talk: Speeding up Deep Neural Networks

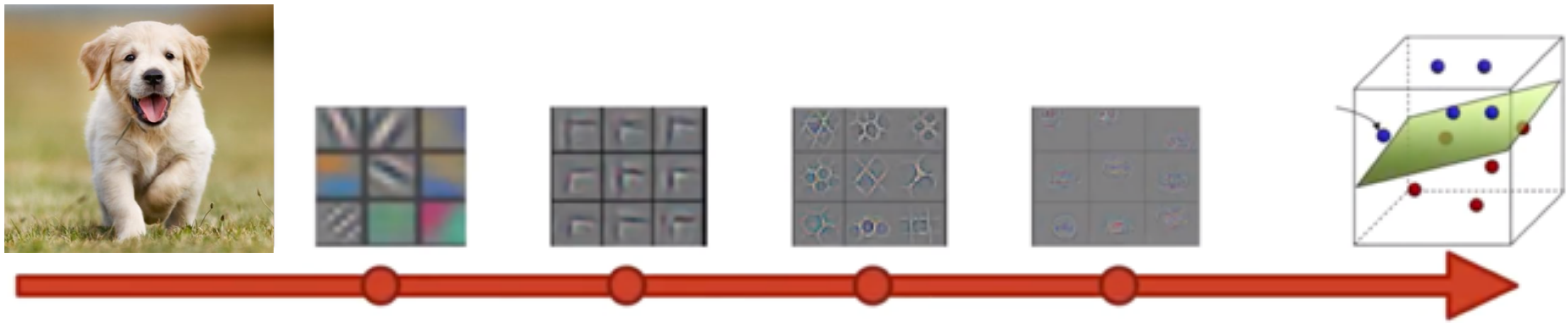
- Adaptive Computation



- Efficient Multi-Scale Architectures

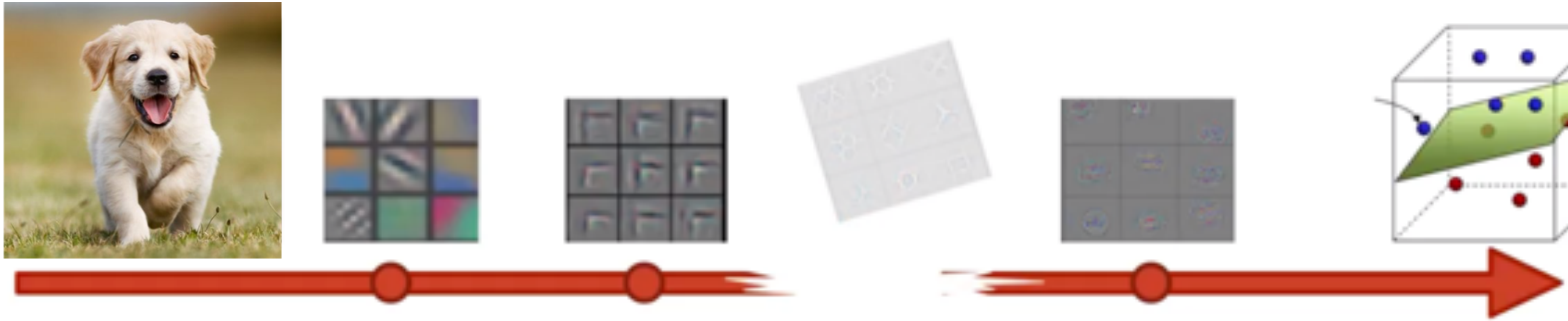


Feed-Forward Convolutional Neural Networks



Adapted from Veit et al

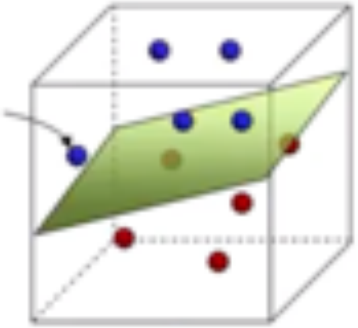
Feed-Forward Convolutional Neural Networks



What happens when we delete a step?

Adapted from Veit et al

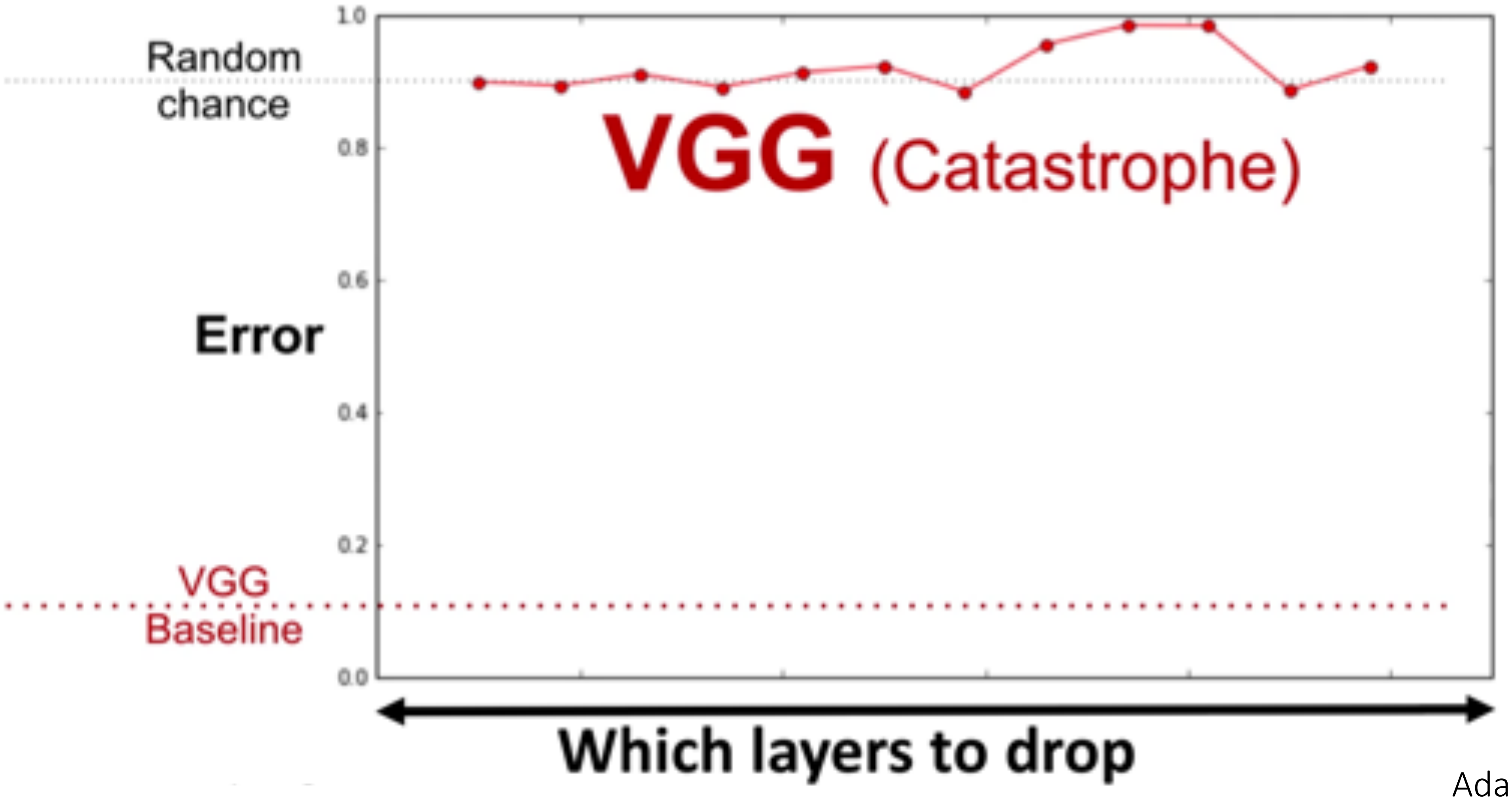
Feed-Forward Convolutional Neural Networks



Adapted from Veit et al

What happens if we delete a layer at test time?

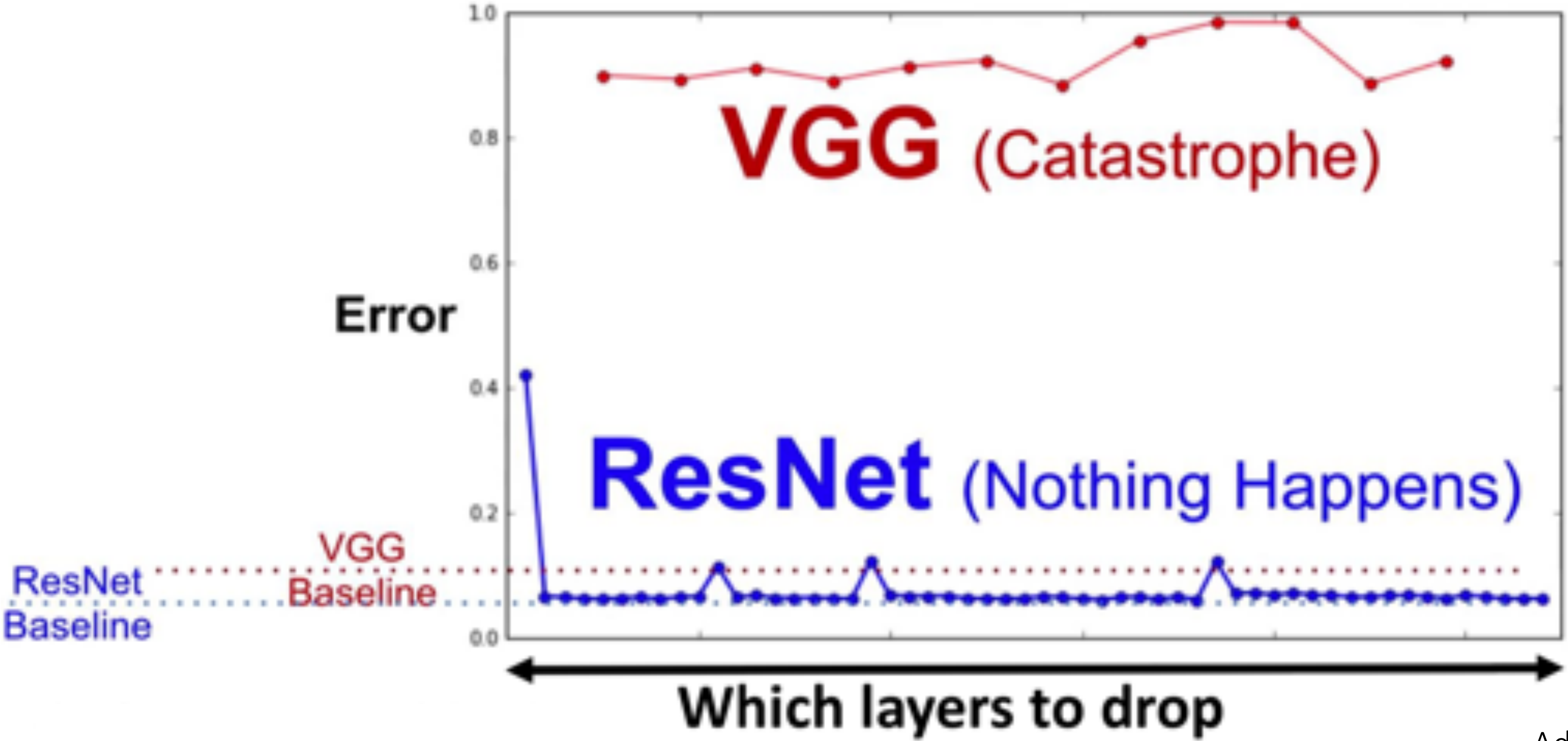
CIFAR-10



Adapted from Veit et al

What happens if we delete a layer at test time?

CIFAR-10



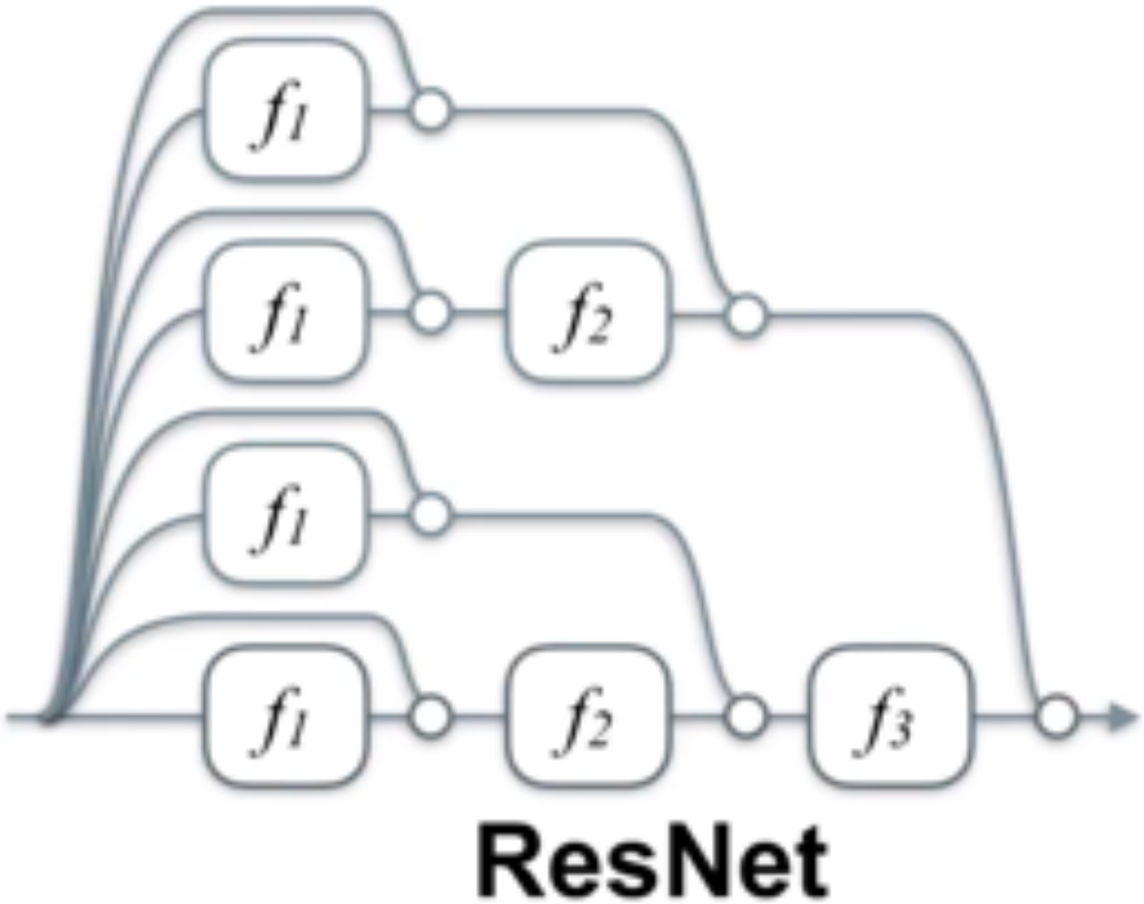
Adapted from Veit et al

Why does this happen?

Adapted from Veit et al

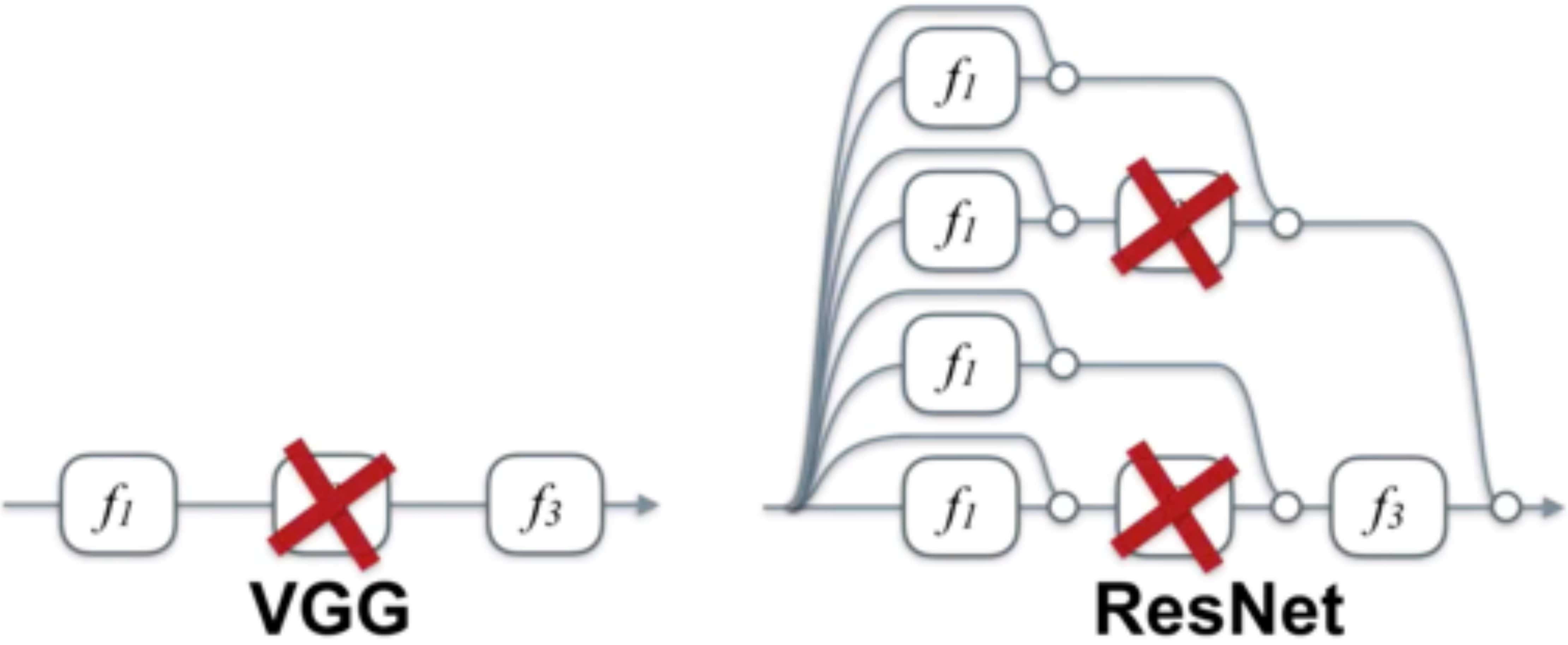
Why does this happen?

The unraveled view is equivalent and showcases the many paths in ResNet.



Adapted from Veit et al

Deletion of a Layer

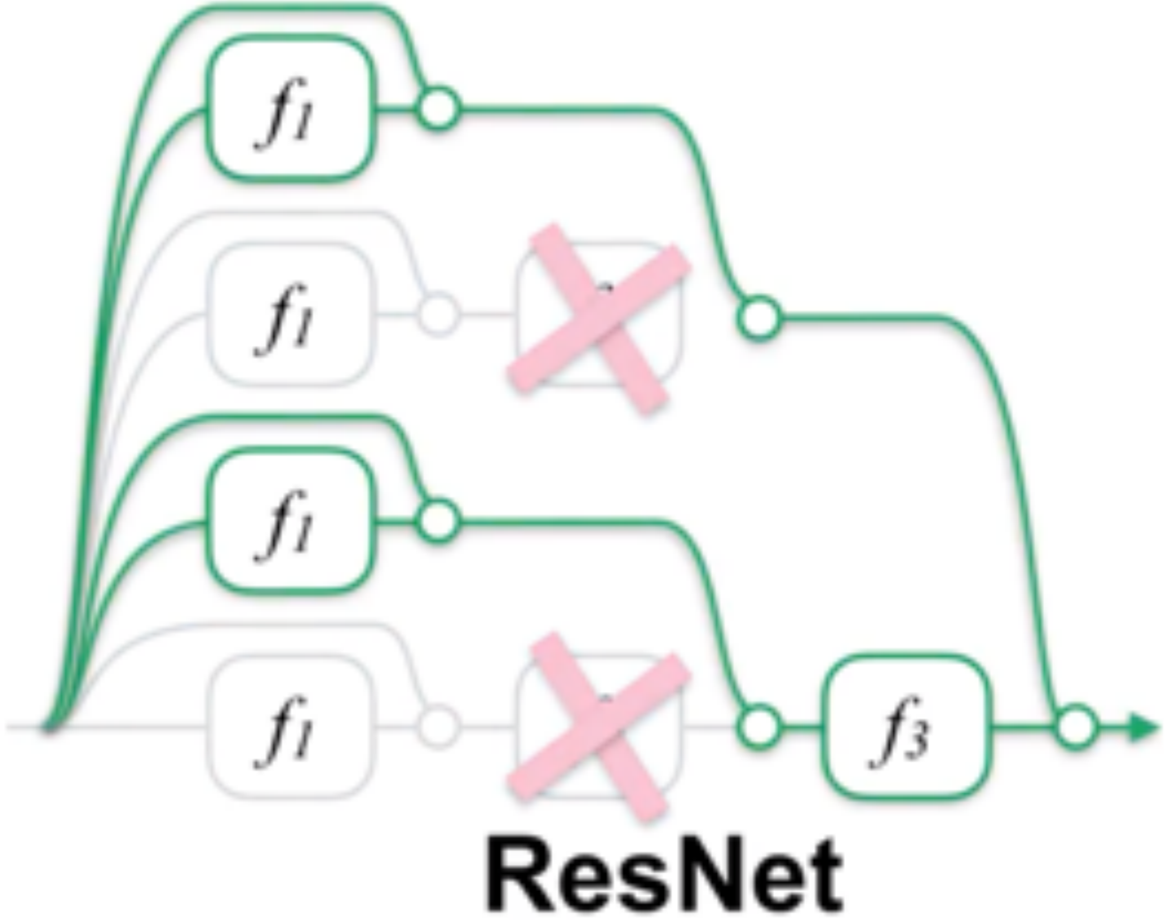


Adapted from Veit et al

Deletion of a Layer

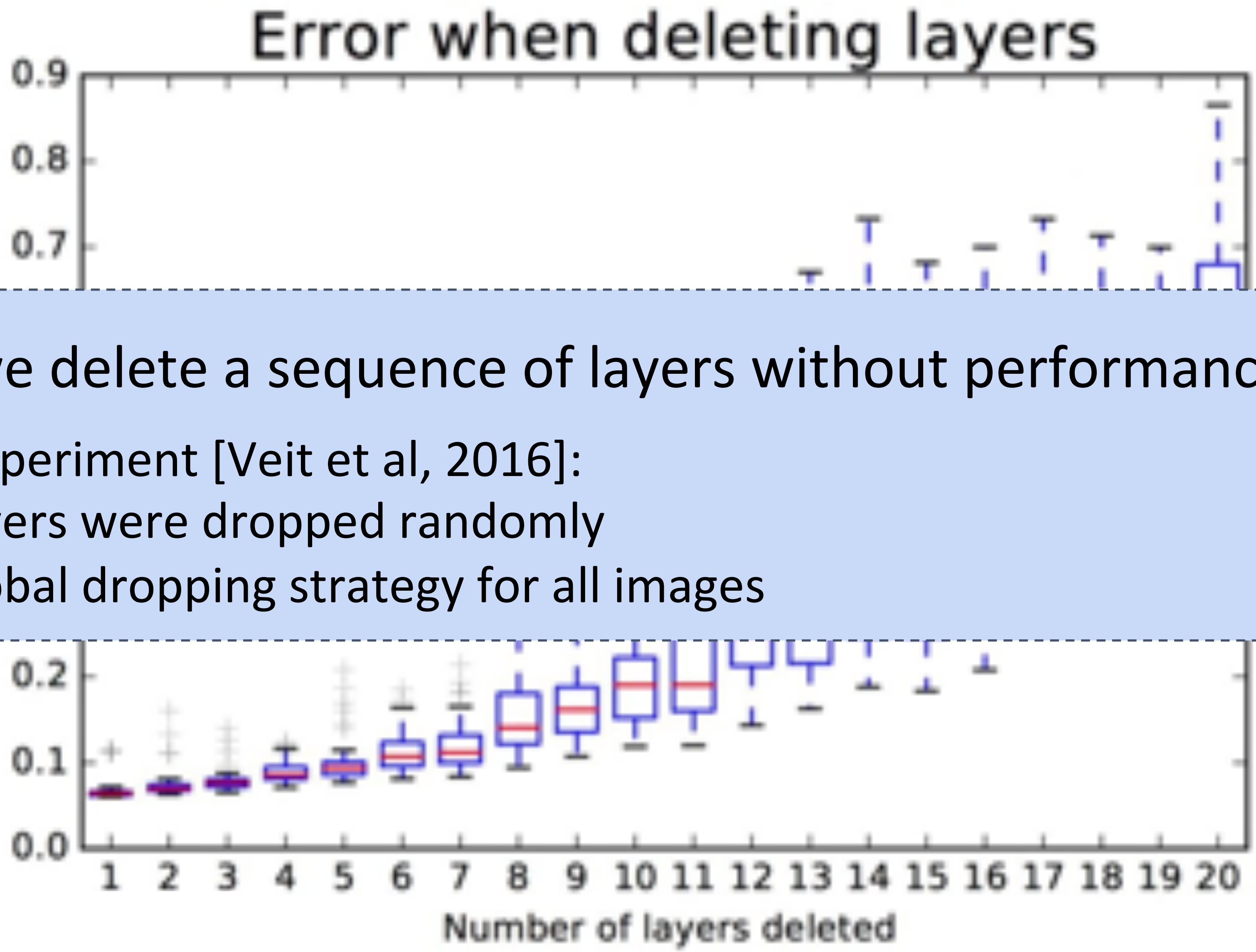
All paths are affected

Only half of the paths are affected



Adapted from Veit et al

Performance varies smoothly when deleting **several** layers.



Can we delete a sequence of layers without performance drop?

This experiment [Veit et al, 2016]:

- Layers were dropped randomly
- Global dropping strategy for all images

Adapted from Veit et al

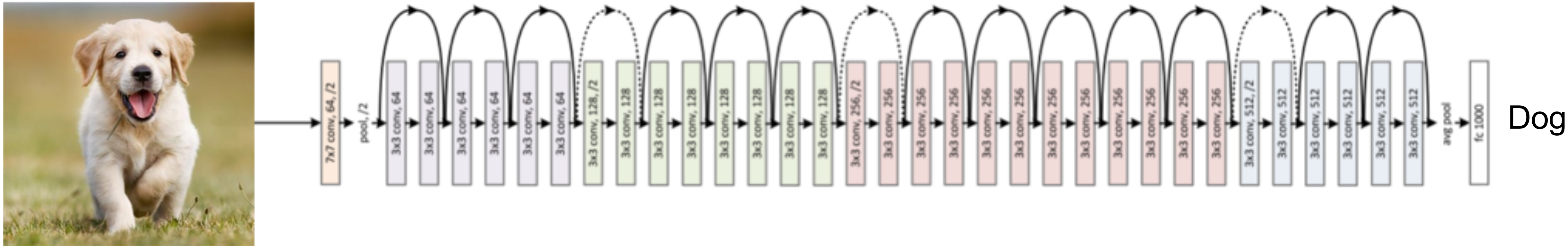
BlockDrop: Dynamic Inference Paths in Residual Networks

Zuxuan Wu*, Tushar Nagarajan*, Abhishek Kumar, Steven Rennie,
Larry S. Davis, Kristen Grauman, Rogerio Feris

CVPR 2018

* Authors contributed equally

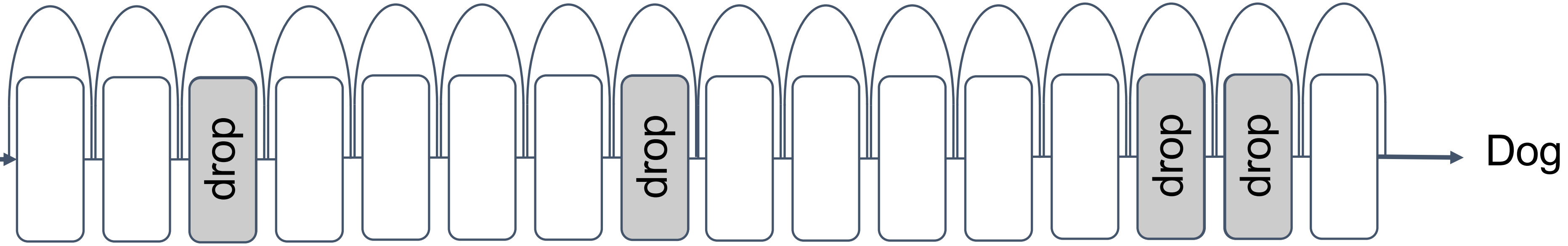
BlockDrop: Dynamic Inference Paths in Residual Networks



Do we really need to run 100+ layers / residual blocks of a neural network if we have an "easy" input image?

[Wu & Nagarajan et al, CVPR 2018]

BlockDrop: Dynamic Inference Paths in Residual Networks



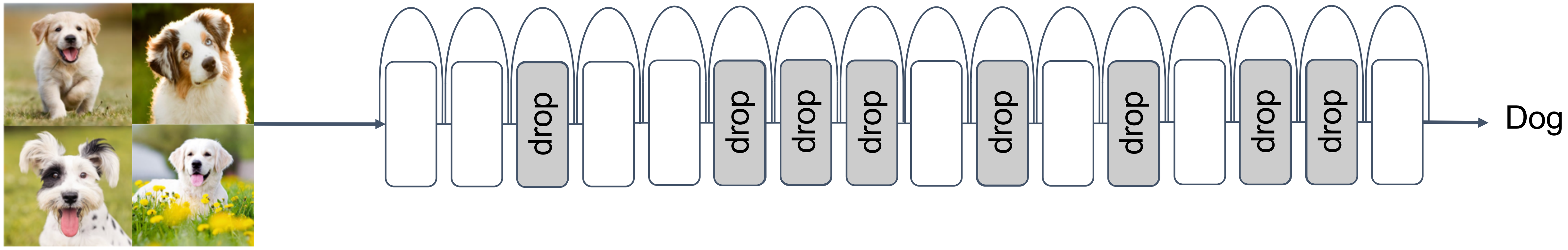
“Dropping some blocks during testing doesn’t hurt performance much”

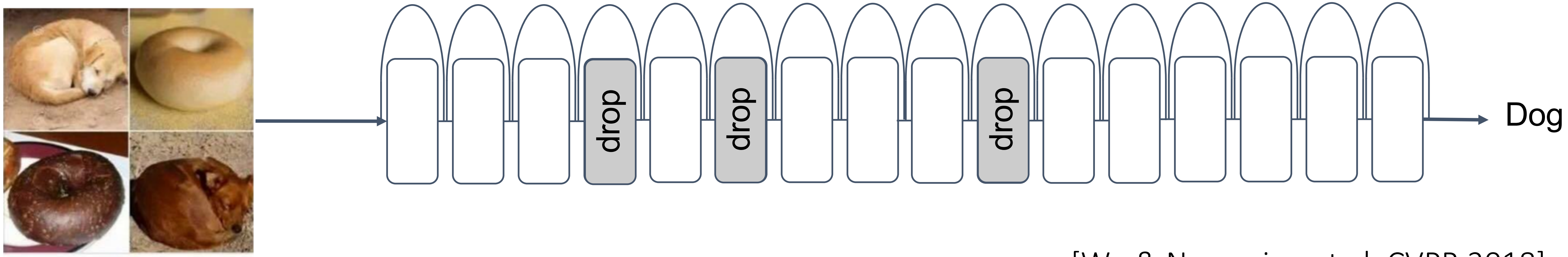
(Veit et al., NIPS 16)

[Wu & Nagarajan et al, CVPR 2018]

BlockDrop: Dynamic Inference Paths in Residual Networks [CVPR 2018]

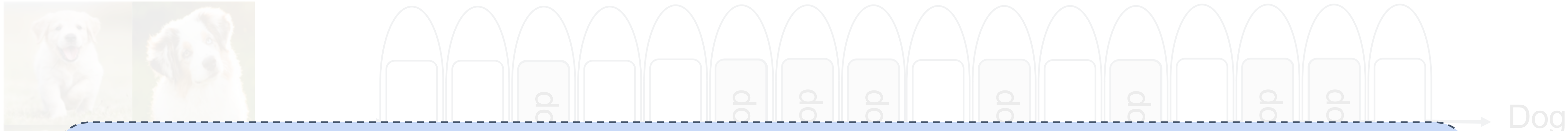
How to determine which blocks to drop depending on the input image?





[Wu & Nagarajan et al, CVPR 2018]

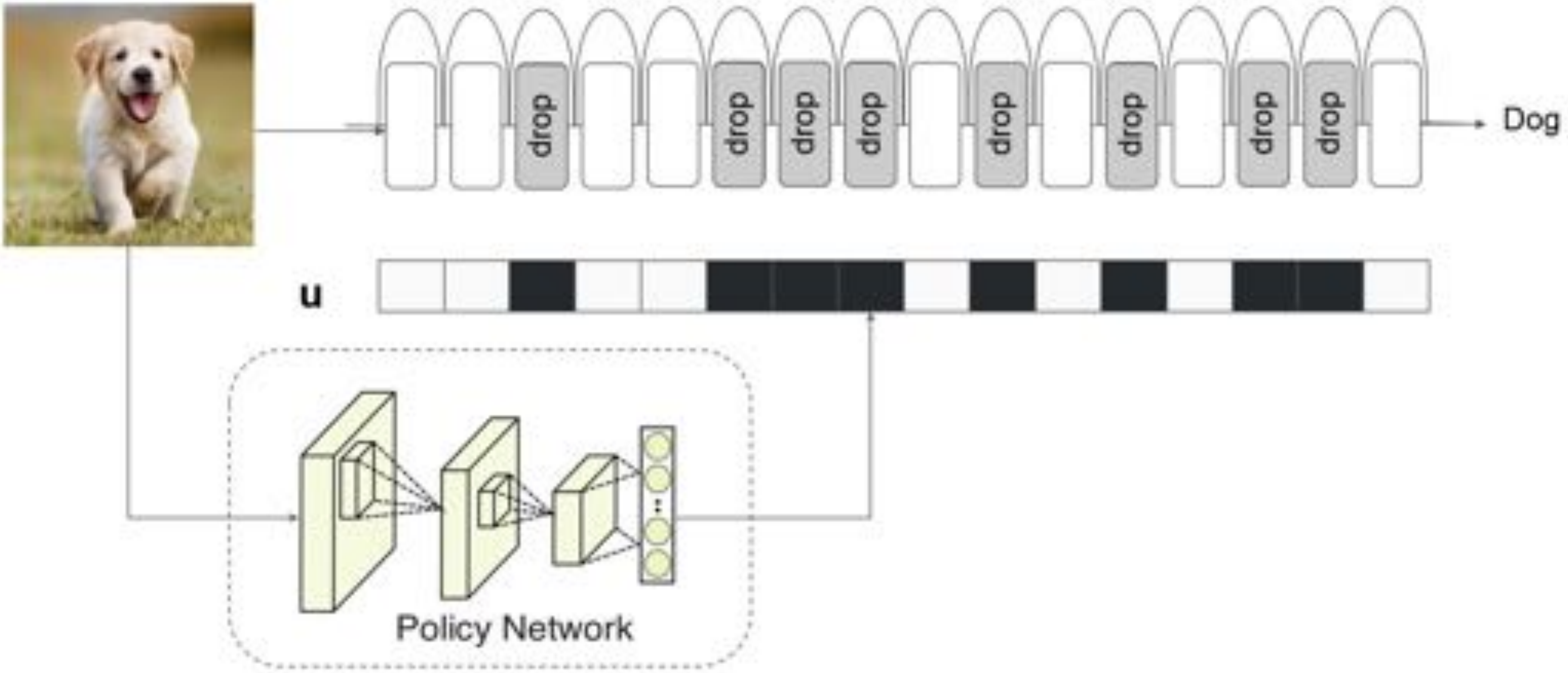
Our Idea: BlockDrop



"Predict which blocks to drop conditioned on the input image, in one shot, without compromising accuracy"

[Wu & Nagarajan et al, CVPR 2018]

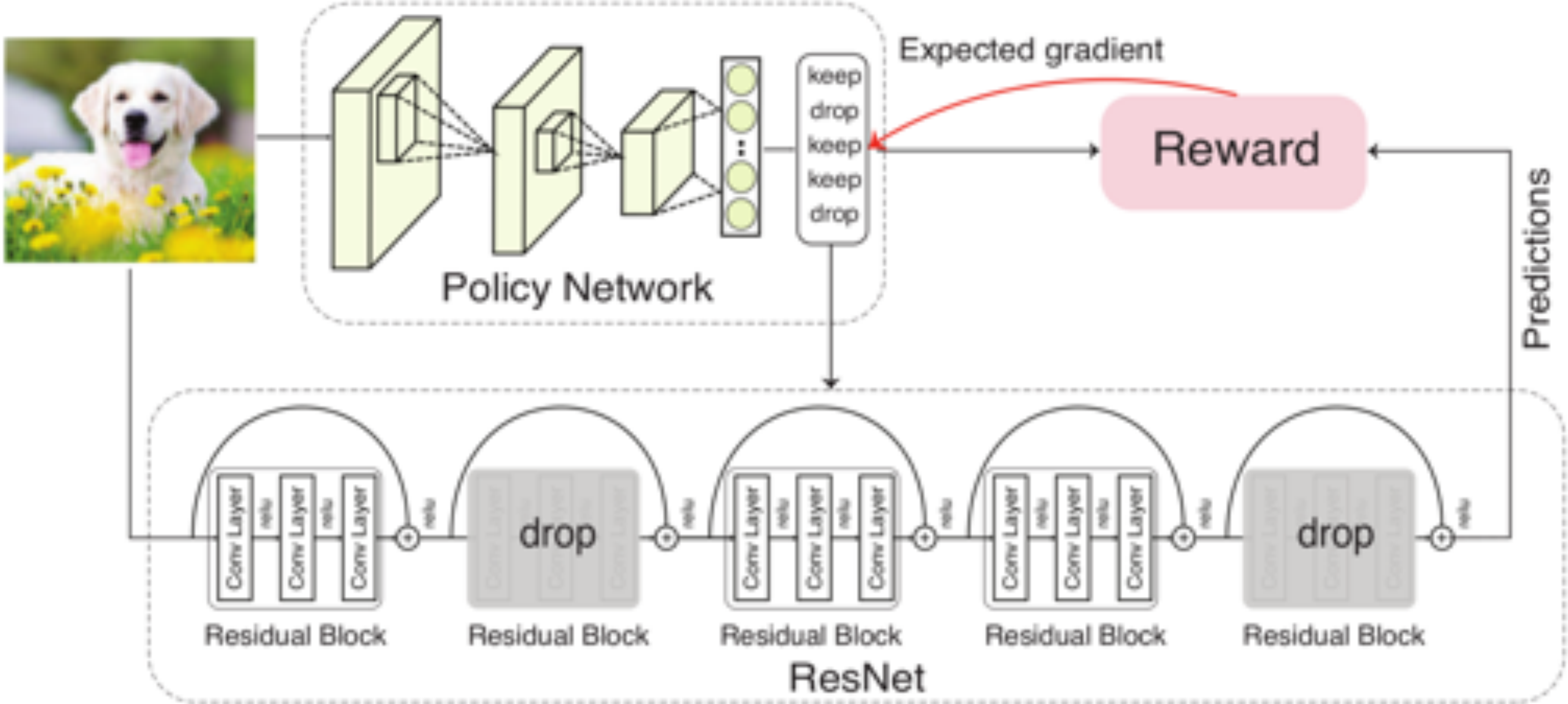
BlockDrop: Dynamic Inference Paths in Residual Networks



[Wu & Nagarajan et al, CVPR 2018]

BlockDrop: Dynamic Inference Paths in Residual Networks [CVPR 2018]

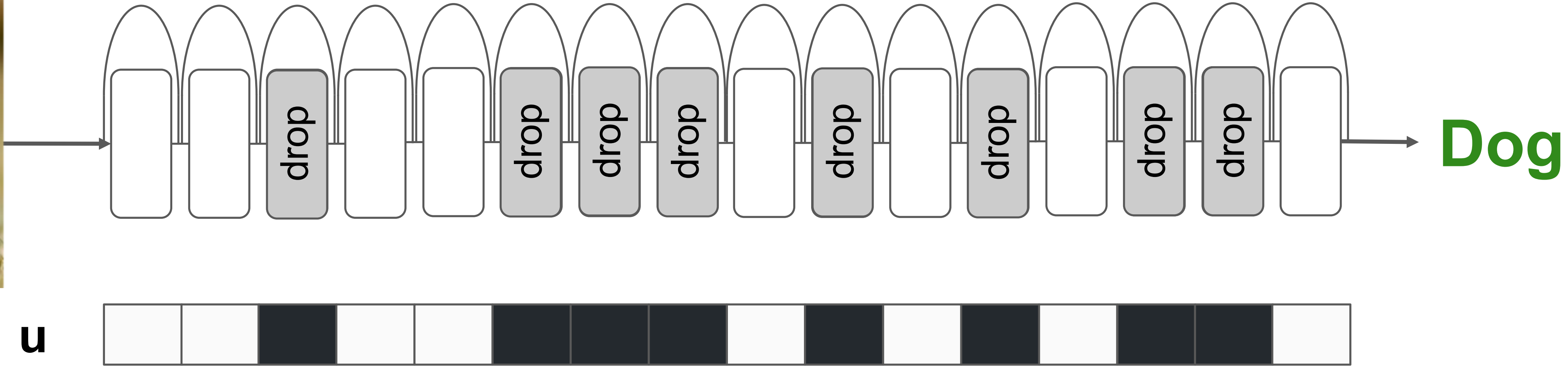
Policy Network Training through Reinforcement Learning



[Wu & Nagarajan et al, CVPR 2018]

BlockDrop: Dynamic Inference Paths in Residual Networks

- Reward function takes into account both accuracy and block usage

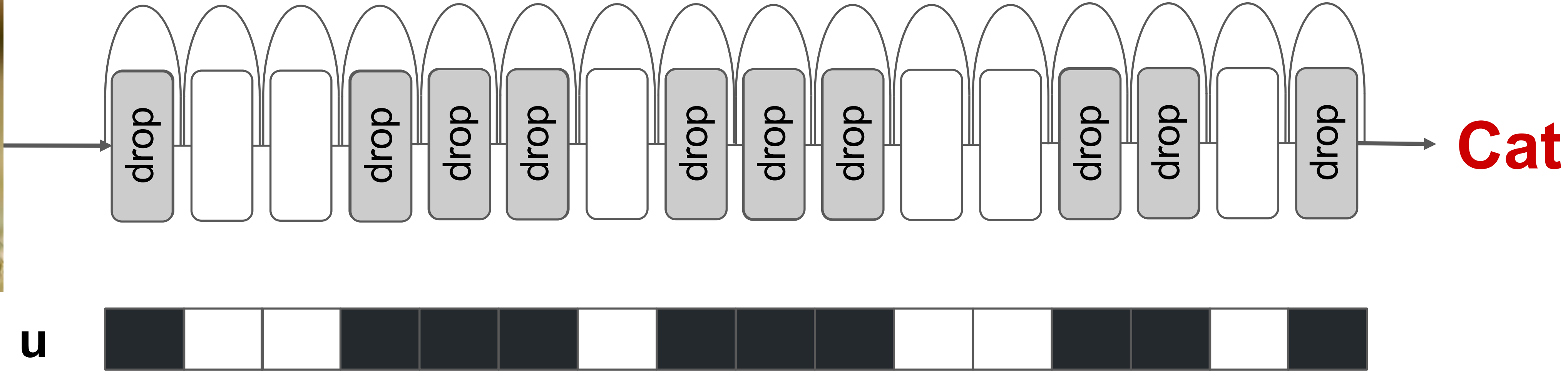


$$R(\mathbf{u}) = \begin{cases} 1 - \left(\frac{|\mathbf{u}|_0}{K}\right)^2 & \text{if correct} \\ -\gamma & \text{otherwise.} \end{cases}$$

$$R(\mathbf{u}) = \mathbf{1} - \left(\frac{\mathbf{8}}{\mathbf{16}}\right)^{\mathbf{2}} = \mathbf{0.75}$$

[Wu & Nagarajan et al, CVPR 2018]

BlockDrop: Dynamic Inference Paths in Residual Networks



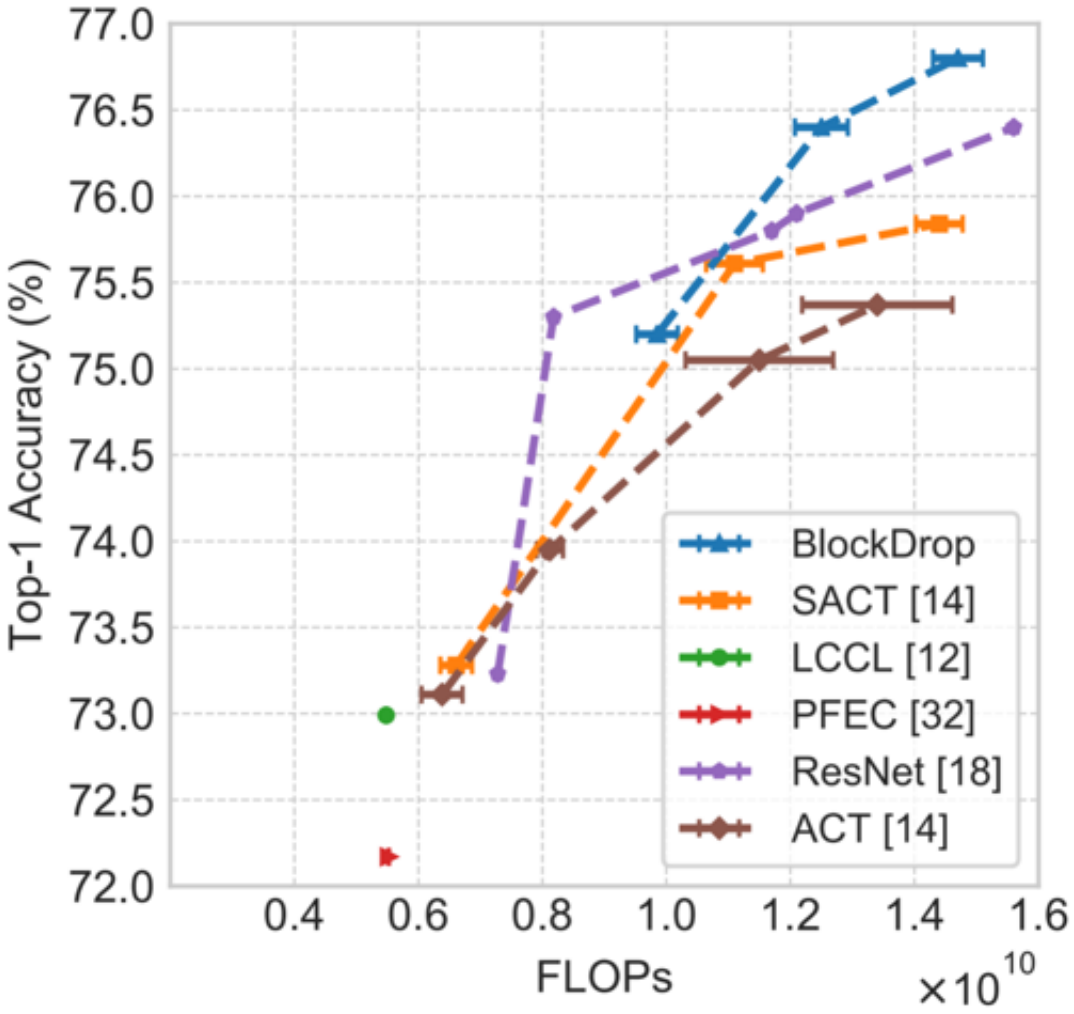
$$R(\mathbf{u}) = \begin{cases} 1 - \left(\frac{|\mathbf{u}|_0}{K}\right)^2 & \text{if correct} \\ -\gamma & \text{otherwise.} \end{cases}$$

$$R(\mathbf{u}) = 1 - \left(\frac{8}{16}\right)^2 = 0.75$$

$$R(\mathbf{u}) = -10$$

[Wu & Nagarajan et al, CVPR 2018]

BlockDrop: Dynamic Inference Paths in Residual Networks

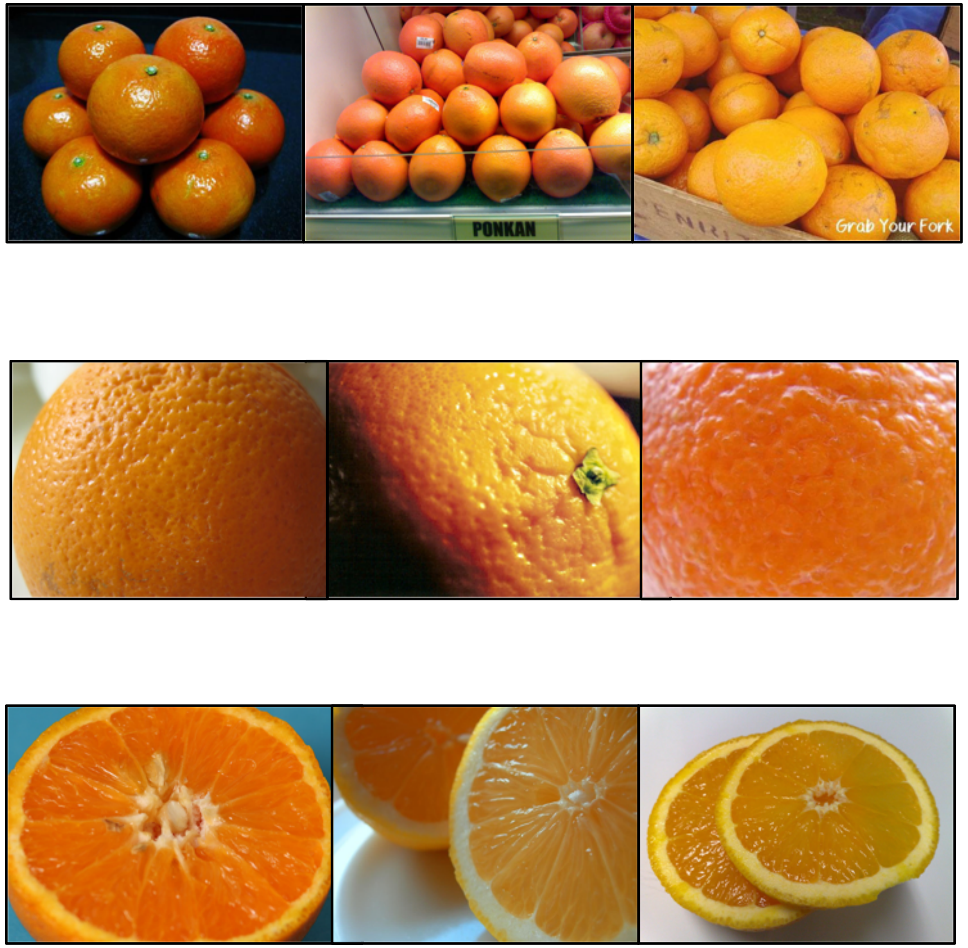


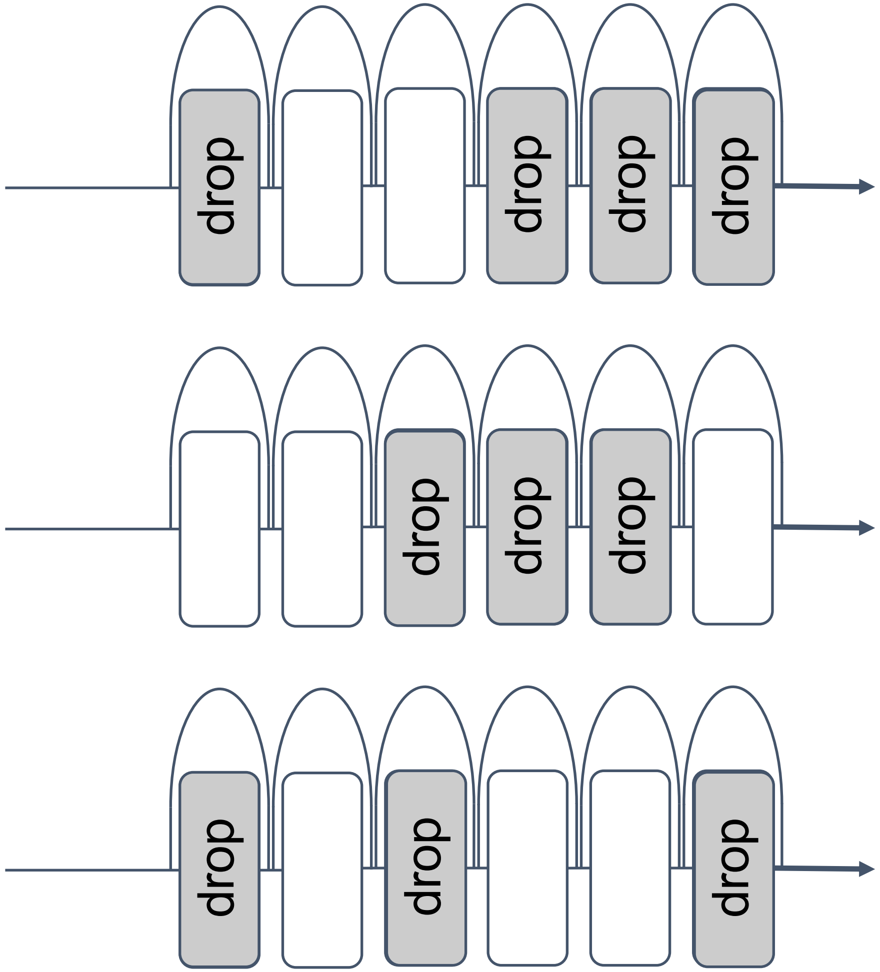
Results on ImageNet:

20% - 36% computational savings (FLOPs)

[Wu & Nagarajan et al, CVPR 2018]

BlockDrop: Dynamic Inference Paths in Residual Networks





Config 1

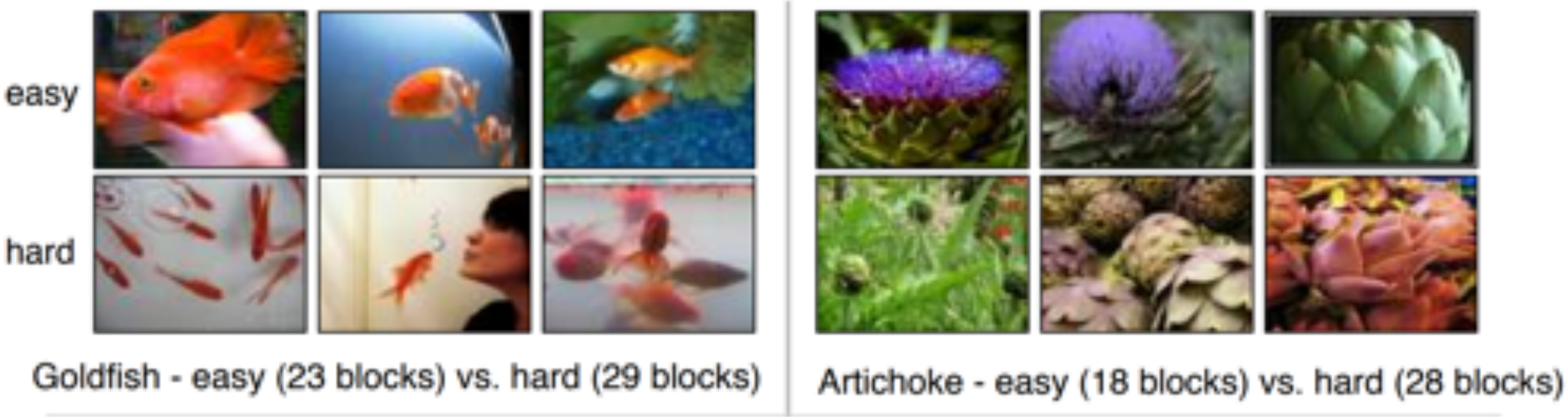
Config 2

Config 3

orange

[Wu & Nagarajan et al, CVPR 2018]

BlockDrop: Dynamic Inference Paths in Residual Networks



Block usage in neural networks agrees with our perception of *difficulty*

[Wu & Nagarajan et al, CVPR 2018]

Extension of BlockDrop: Adaptive Computation for Transfer Learning

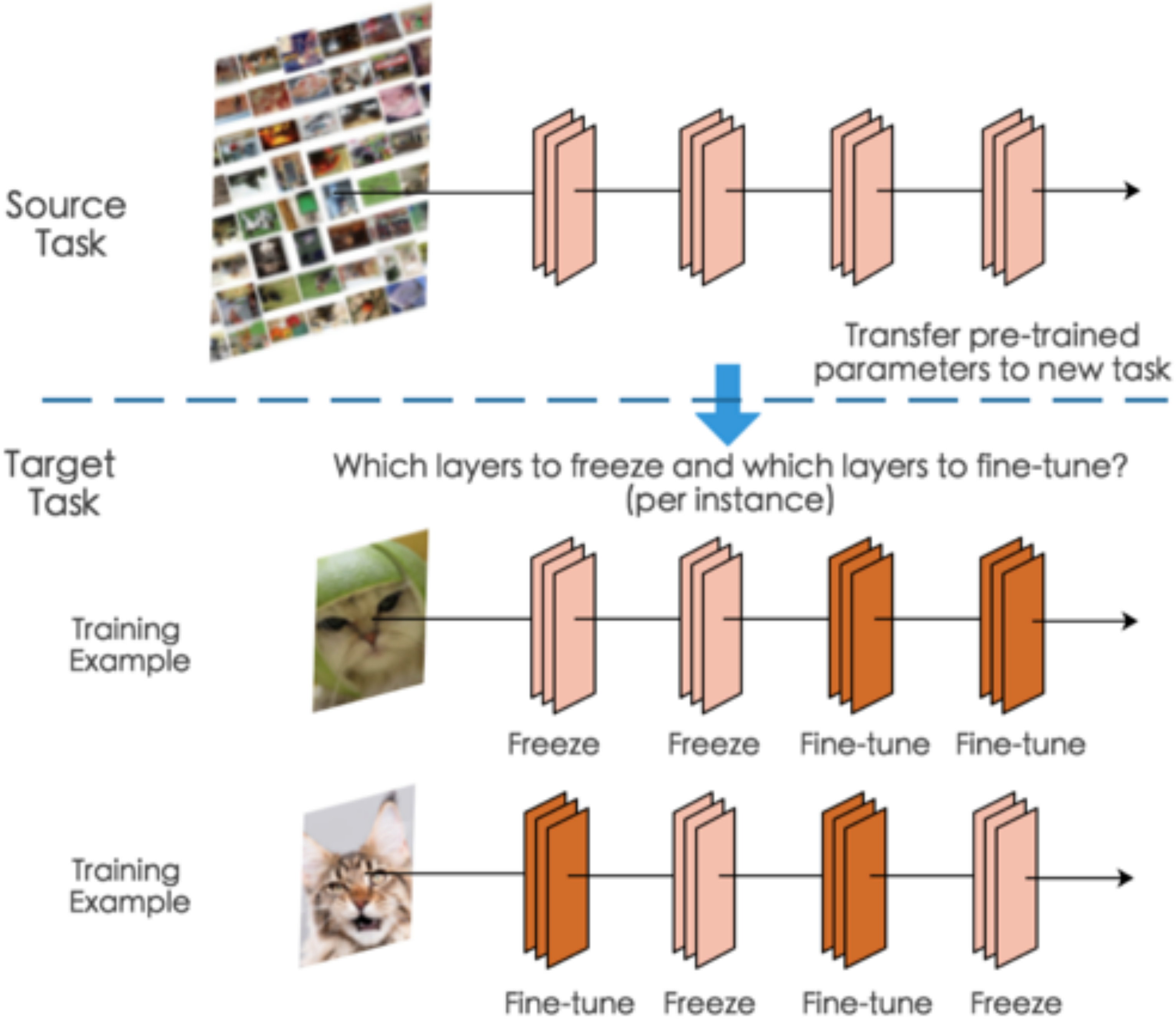
Data Efficiency: Transfer Learning

- Fine-tuning is arguably the most widely used approach for transfer learning

- Existing methods are ad-hoc in terms of determining ***where to fine-tune*** in a deep neural network (*e.g.*, fine-tuning last k layers)

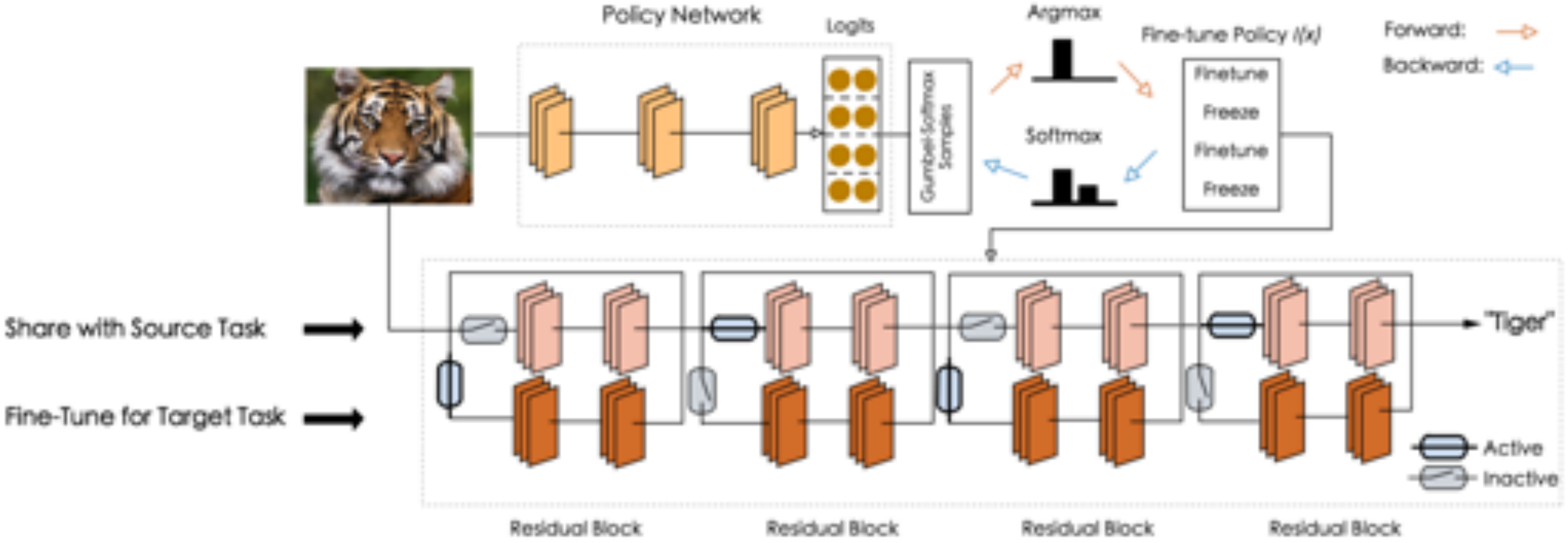
- We propose *SpotTune*, a method that automatically decides, per training example, which layers of a pre-trained model should have their parameters frozen (shared with the source domain) or fine-tuned (adapted to the target domain)

[Guo et al, CVPR 2019]



[Guo et al, CVPR 2019]

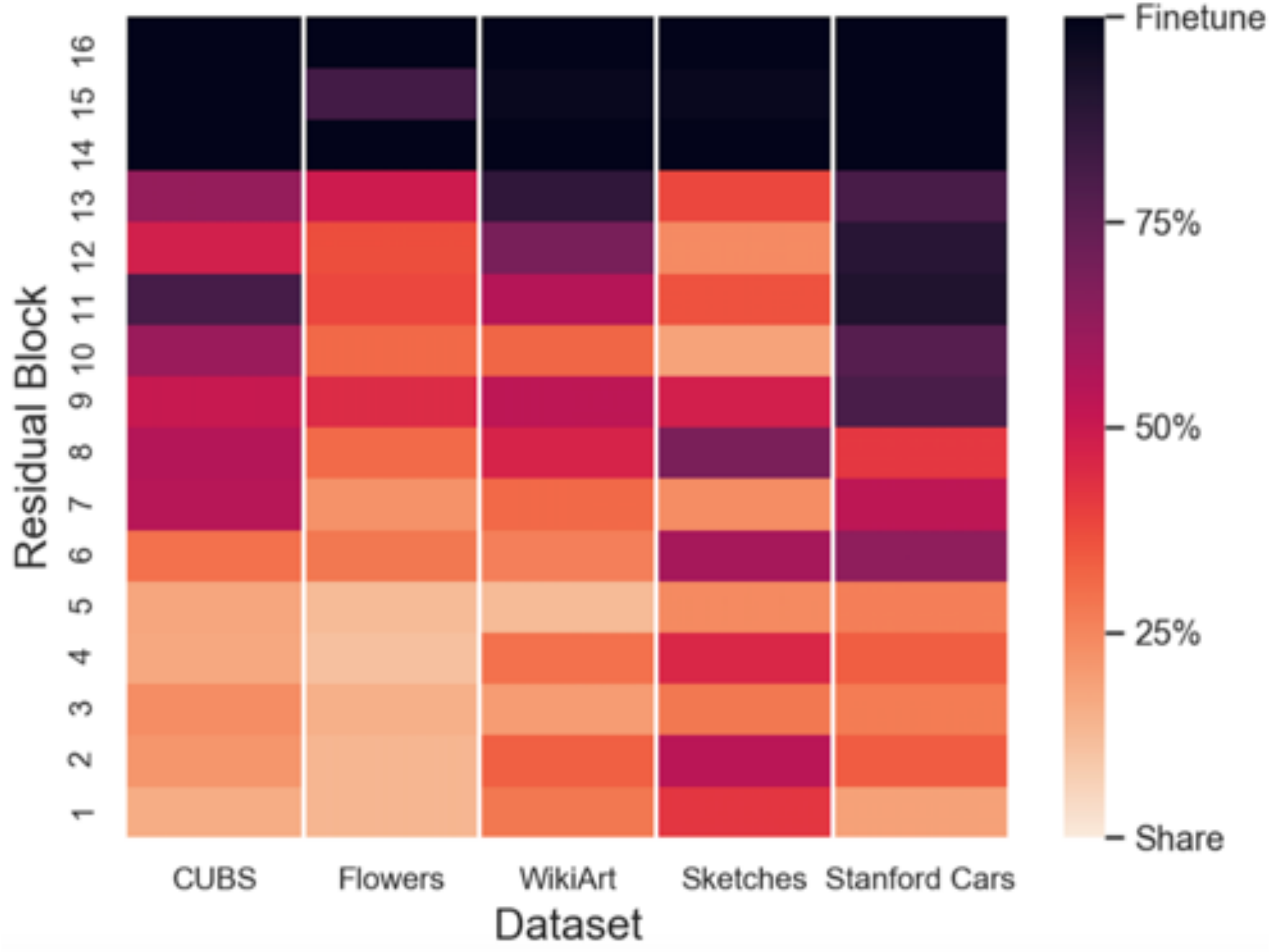
SpotTune: Transfer Learning through Adaptive Fine-Tuning



[Guo et al, CVPR 2019]

SpotTune: Transfer Learning through Adaptive Fine-Tuning

Fine-Tuning Policy Visualization



SpotTune automatically identifies the right fine-tuning policy for each dataset, for each training example.

[Guo et al, CVPR 2019]

SpotTune: Transfer Learning through Adaptive Fine-Tuning

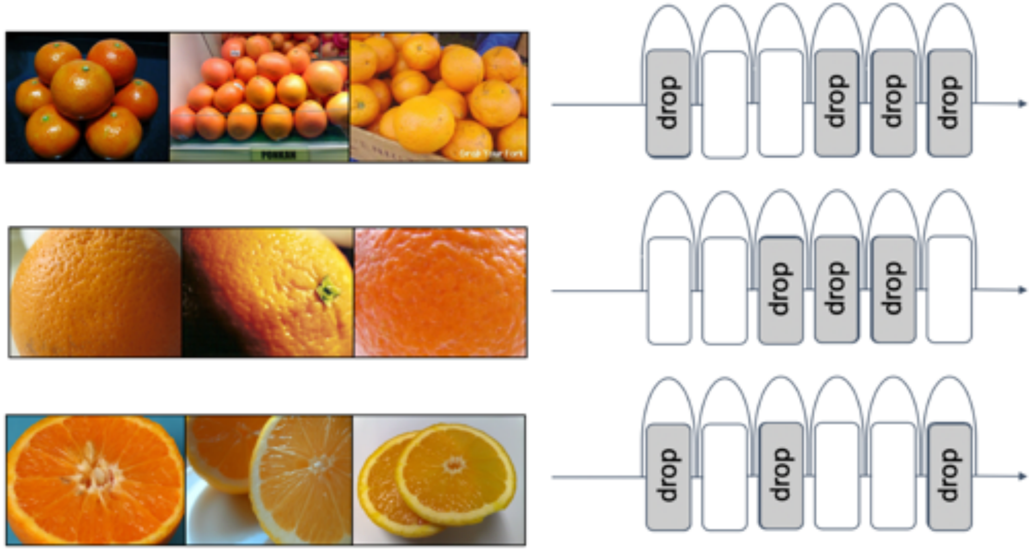
	#par	ImNet	Airc.	C100	DPed	DTD	GTSR	Flwr	OGlt	SVHN	UCF	Score
Scratch	10x	59.87	57.10	75.73	91.20	37.77	96.55	56.30	88.74	96.63	43.27	1625
Scratch+ [37]	11x	59.67	59.59	76.08	92.45	39.63	96.90	56.66	88.74	96.78	44.17	1826
Feature Extractor	1x	59.67	23.31	63.11	80.33	55.53	68.18	73.69	58.79	43.54	26.80	544
Fine-tuning [38]	10x	60.32	61.87	82.12	92.82	55.53	99.42	81.41	89.12	96.55	51.20	3096
BN Adapt. [5]	1x	59.87	43.05	78.62	92.07	51.60	95.82	74.14	84.83	94.10	43.51	1353
LwF [26]	10x	59.87	61.15	82.23	92.34	58.83	97.57	83.05	88.08	96.10	50.04	2515
Series Res. adapt. [37]	2x	60.32	61.87	81.22	93.88	57.13	99.27	81.67	89.62	96.57	50.12	3159
Parallel Res. adapt. [38]	2x	60.32	64.21	81.92	94.73	58.83	99.38	84.68	89.21	96.54	50.94	3412
Res. adapt. (large) [37]	12x	67.00	67.69	84.69	94.28	59.41	97.43	84.86	89.92	96.59	52.39	3131
Res. adapt. decay [37]	2x	59.67	61.87	81.20	93.88	57.13	97.57	81.67	89.62	96.13	50.12	2621
Res. adapt. finetune all [37]	2x	59.23	63.73	81.31	93.30	57.02	97.47	83.43	89.82	96.17	50.28	2643
DAN [39]	2x	57.74	64.12	80.07	91.30	56.54	98.46	86.05	89.67	96.77	49.48	2851
PiggyBack [31]	1.28x	57.69	65.29	79.87	96.99	57.45	97.27	79.09	87.63	97.24	47.48	2838
SpotTune	11x	60.32	63.91	80.48	96.49	57.13	99.52	85.22	88.84	96.72	52.34	**3612**

SpotTune sets the new state of the art on the Visual Decathlon Challenge

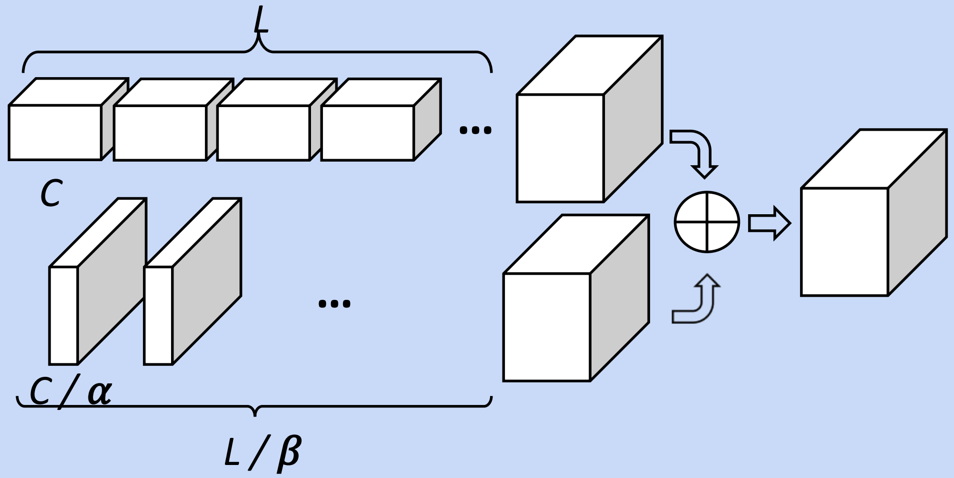
[Guo et al, CVPR 2019]

This talk: Speeding up Deep Neural Networks

- Adaptive Computation

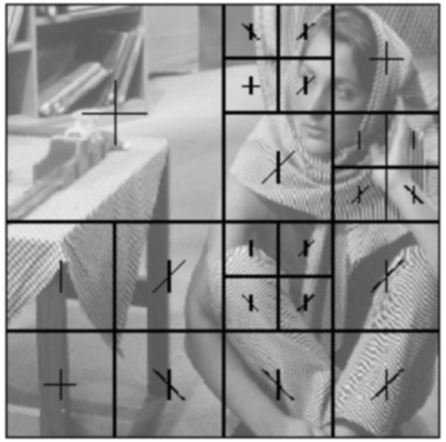


- Efficient Multi-Scale Architectures

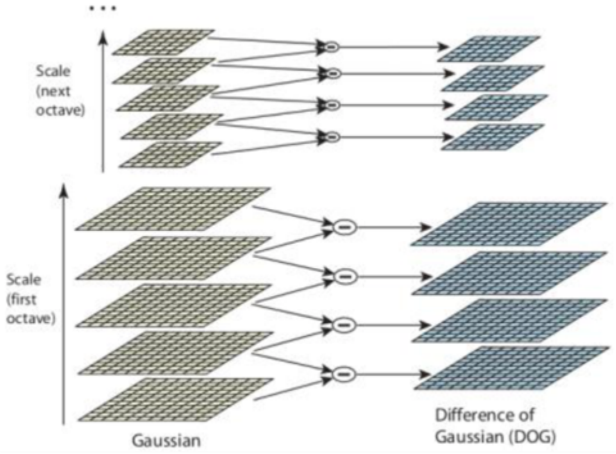


Multi-Scale Feature Representations

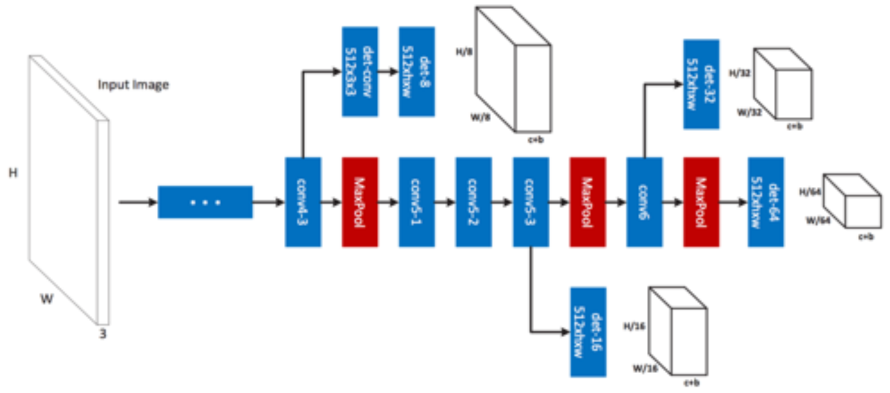
Wavelets [Daubechies/Mallat/etc. 90s]



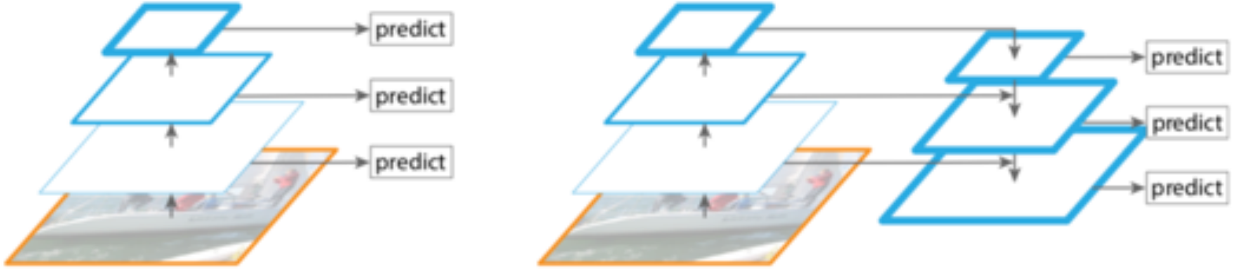
SIFT Features [Lowe, 1996]



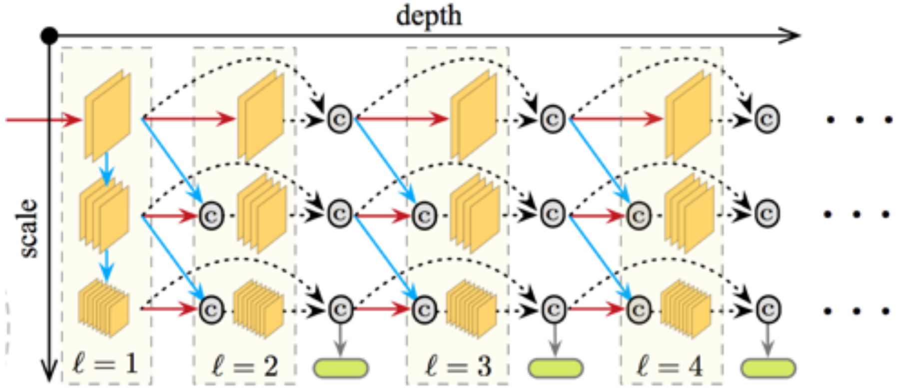
MS-CNN [Cai et al, 2016]



Feature Pyramid Networks [Lin et al, 2017



MSDNet [Huang et al, 2018]



Many more!

Problem

- Image processing at multiple resolutions usually leads to additional computational time

→ How to design an efficient multi-scale network architecture?

- Goal: Speed up inferencing while maintaining accuracy

Big-Little Net

- A multi-branch network that:

  1) has different computation complexities for each branch/scale

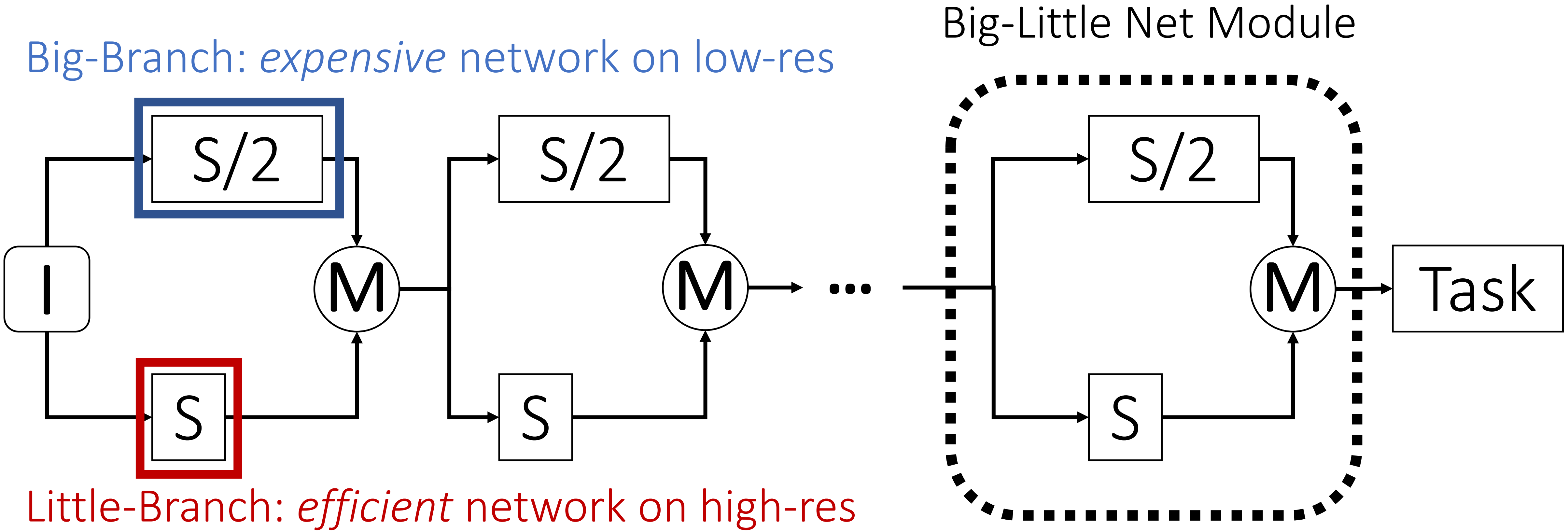
  2) fuses different scales at multiple levels of the network

  in order to achieve the best accuracy-efficiency trade-off

[Chen et al, ICLR 2019]

Big-Little Net

Big-Branch: *expensive* network on low-res



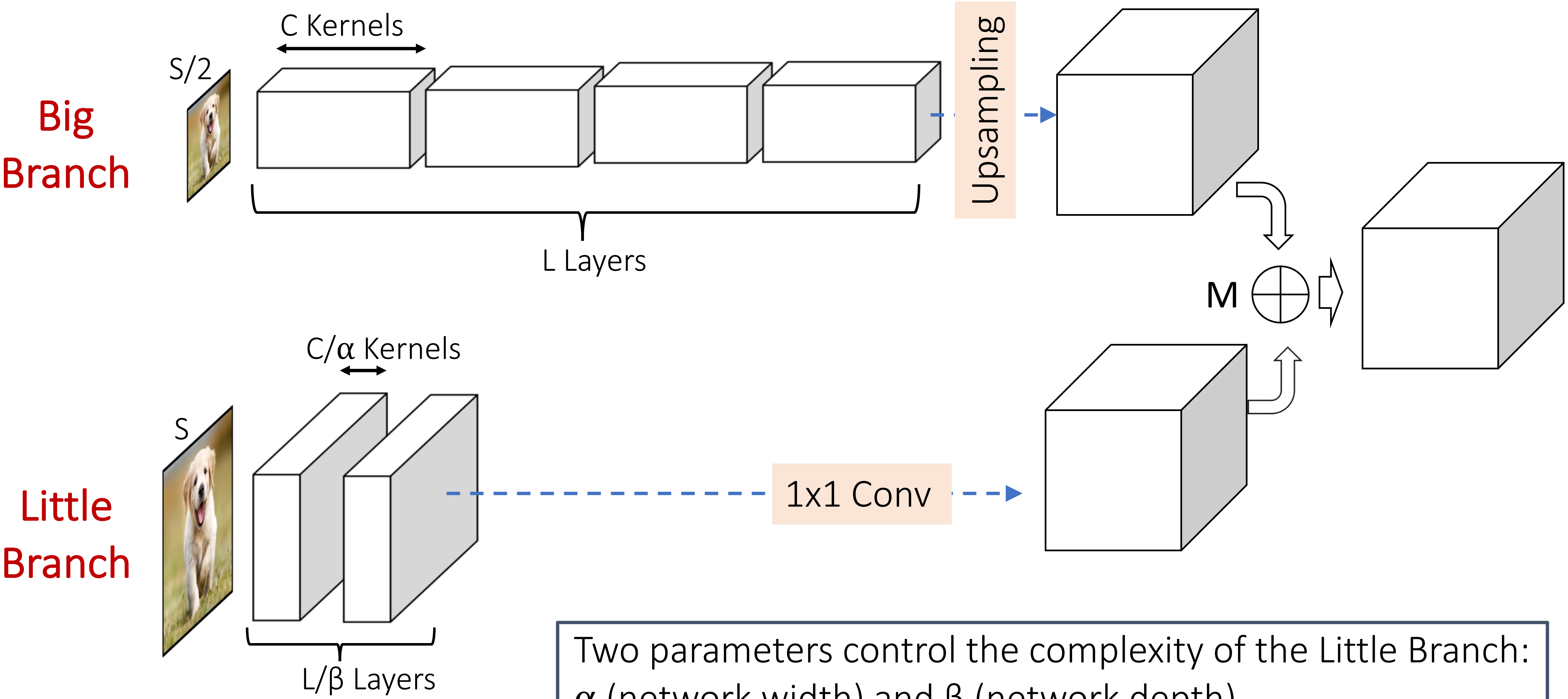
Little-Branch: *efficient* network on high-res

I: input; **M**: merge operator

S: original resolution of input, **S/2**: half resolution of input

[Chen et al, ICLR 2019]

Big-Little Net Module



Two parameters control the complexity of the Little Branch: α (network width) and β (network depth)

Experimental Results

- Image Classification:
  - Dataset: ImageNet-1K
  - Backbone network: ResNet or ResNeXt
- Speech Recognition:
  - Dataset: Switchboard
  - Backbone network: ResNet

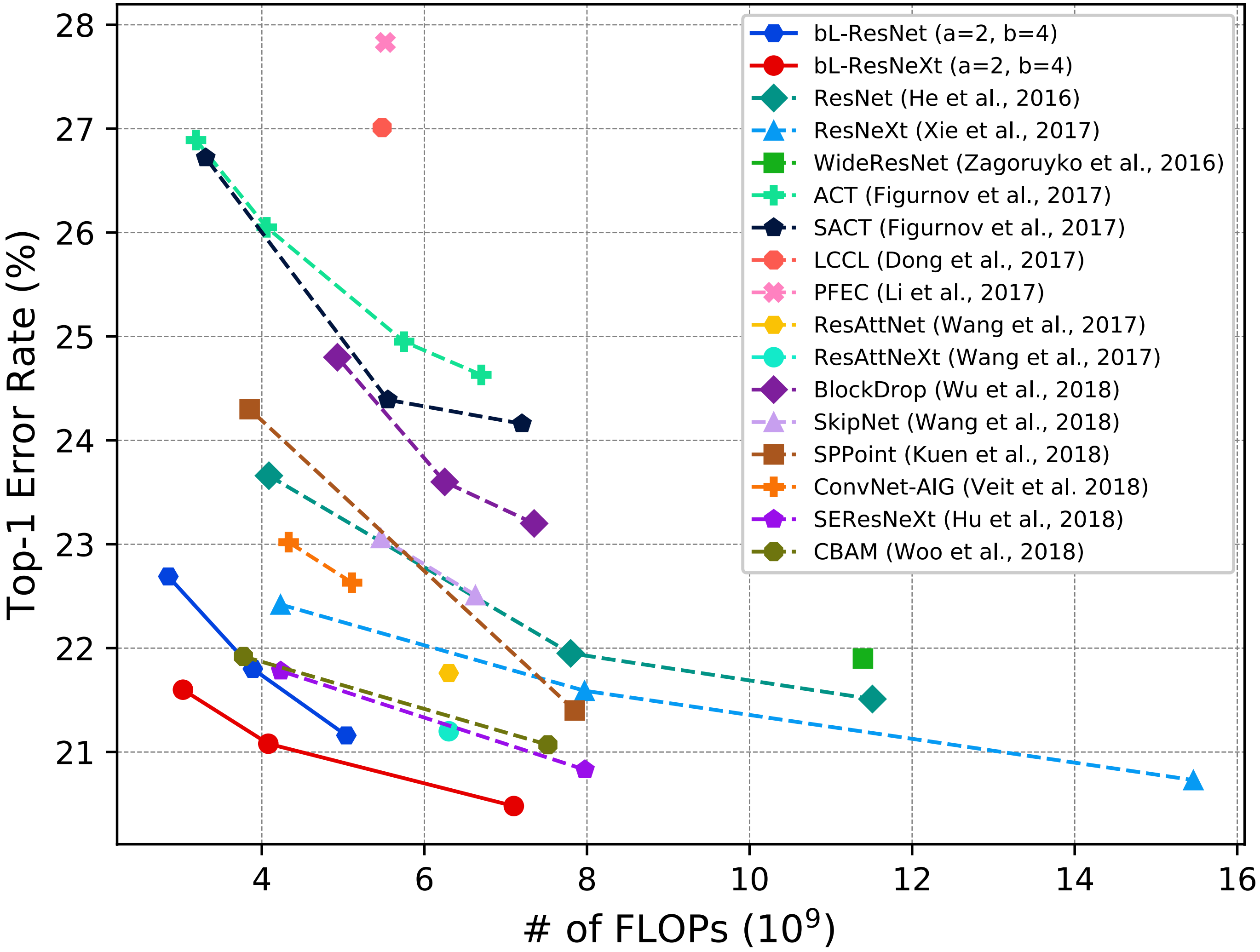
[Chen et al, ICLR 2019]

Experimental Results: ImageNet

Model (bL-model, α=2, β=4)	Top-1 Error	FLOPs (10^9)	Params (10^6)	GPU speedup
ResNet-101	21.95%	7.80	44.54	-
bL-ResNet-101	**21.80%**	**3.89 (2.01×)**	41.85	1.33×
ResNet-152	21.51%	11.51	60.19	-
bL-ResNet-152	**21.16%**	**5.04 (2.28×)**	57.36	1.49×
ResNeXt-50 (32×4d)	22.20%	4.23	25.03	-
bL-ResNeXt-50 (32×4d)	**21.60%**	**3.03 (1.40×)**	25.03	1.26×
ResNeXt-101 (32×4d)	21.20%	7.97	44.17	-
bL-ResNeXt-101 (32×4d)	**21.08%**	**4.08 (1.95×)**	41.51	1.59×
ResNeXt-101 (64×4d)	20.73%	15.46	83.46	-
bL-ResNeXt-101 (64×4d)	**20.48%**	**7.14 (2.17×)**	77.36	1.98×
SEResNeXt-50 (32×4d)	21.78%	4.23	27.56	-
bL-SEResNeXt-50 (32×4d)	**21.44%**	**3.03 (1.40×)**	28.77	1.33×
SEResNeXt-101 (32×4d)	21.00%	7.97	48.96	-
bL-SEResNeXt-101 (32×4d)	**20.87%**	**4.08 (1.95×)**	45.88	1.60×

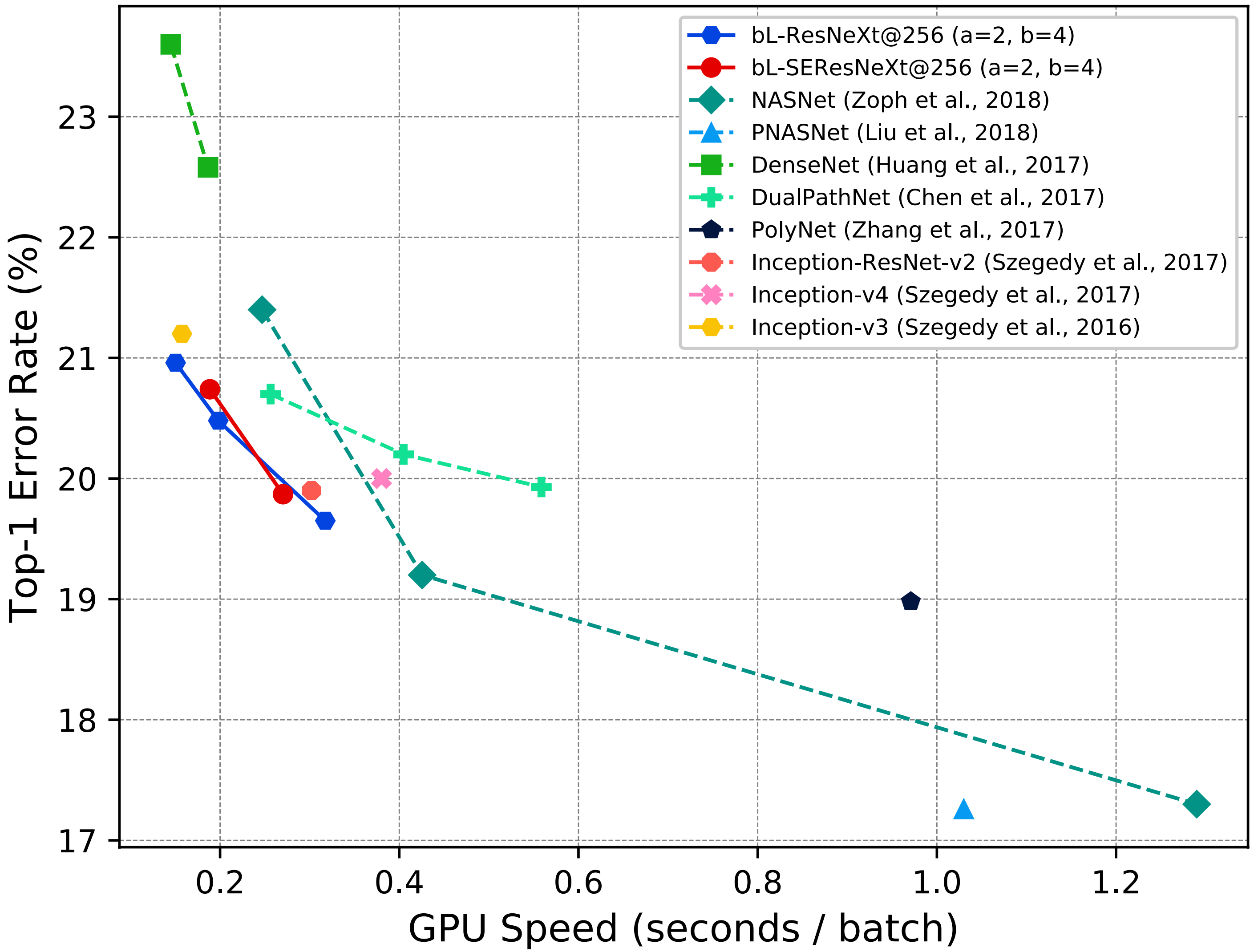
[Chen et al, ICLR 2019]

Experimental Results: Comparison with CNNs based on ResNet and ResNeXt on ImageNet



[Chen et al, ICLR 2019]

Experimental Results: Comparison with SOTA networks in accuracy and GPU runtime on ImageNet



[Chen et al, ICLR 2019]

Experimental Results: Speech Recognition

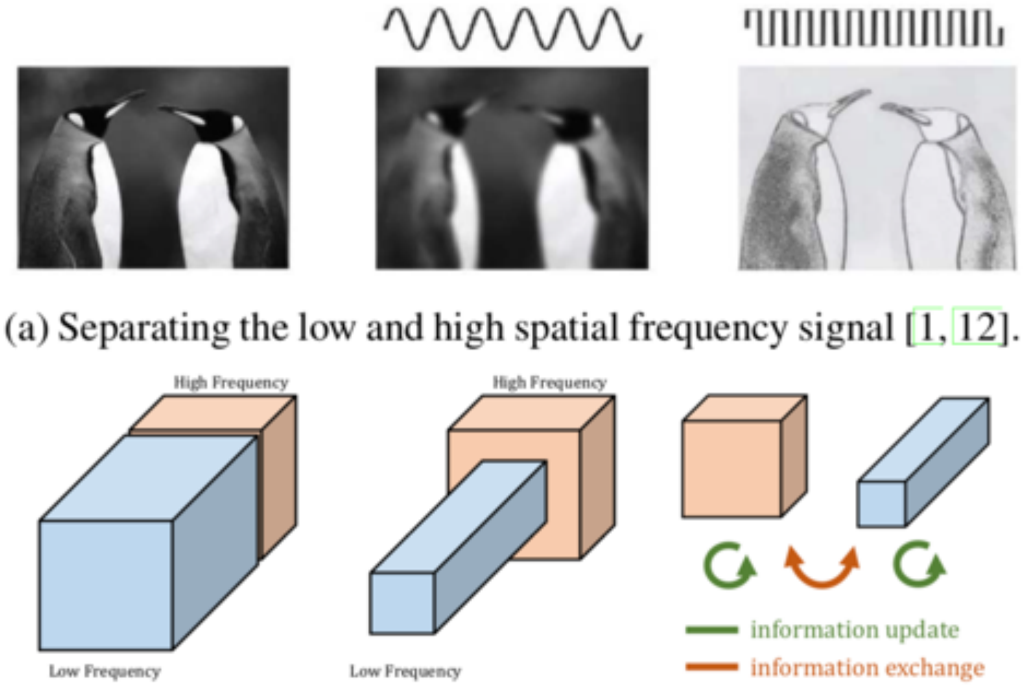
Dataset: Switchboard

Model	FLOPs (10^9)	Params (10^6)	WER Avg	Hub5	Hub5 CH
ResNet-22	1.11	3.02	14.67%	11.15%	18.17%
bL-ResNet-22 (α=4, β=1)	0.68	3.15	14.72%	11.24%	18.18%
bL-ResNet-22 (α=4, β=2)	0.66	3.11	14.47%	**10.95%**	17.95%
bL-ResNet-22 (α=4, β=3)	**0.65**	3.10	14.66%	11.25%	18.05%
bL-ResNet-22 (α=2, β=3)	0.77	3.07	**14.46%**	11.10%	**17.80%**

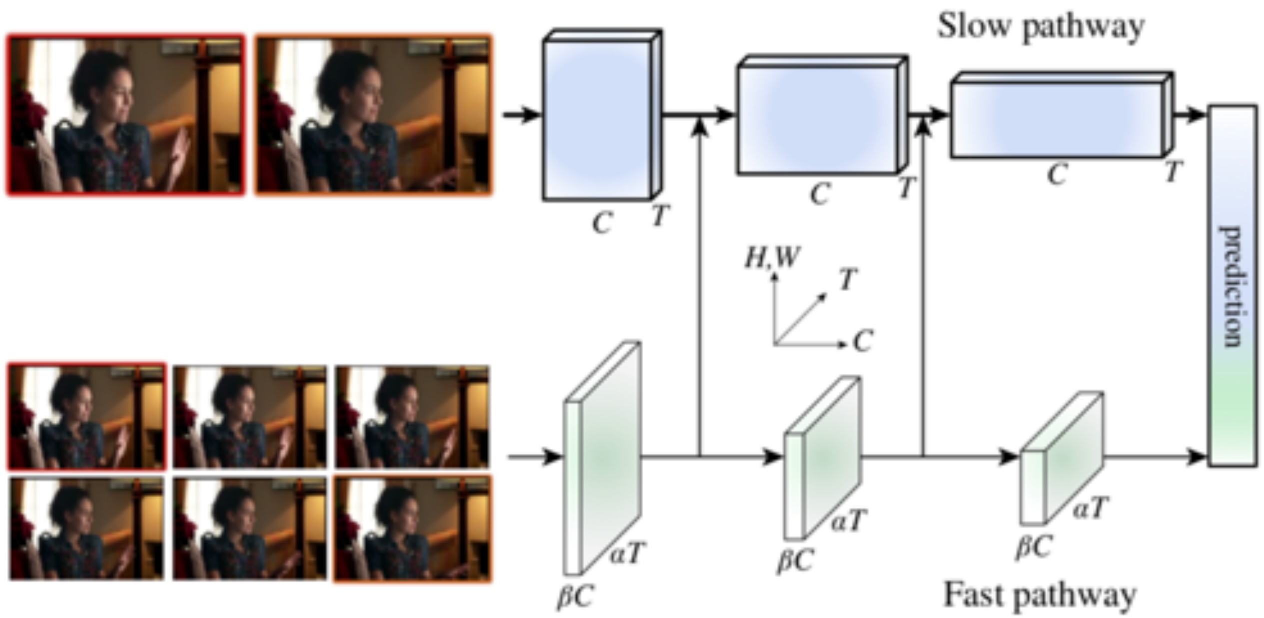
[Chen et al, ICLR 2019]

Recent work related to Big-Little Net

Drop an Octave [Chen et al, CVPR 2019]

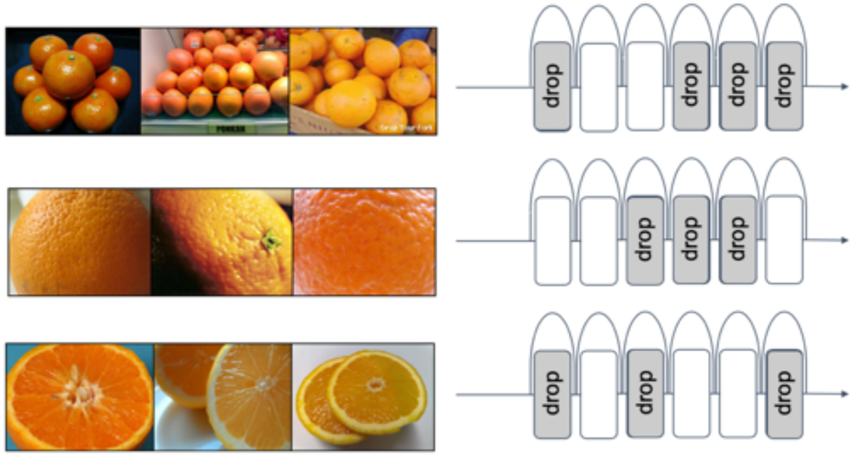


SlowFast Networks [Feichtenhofer et al, 2019]

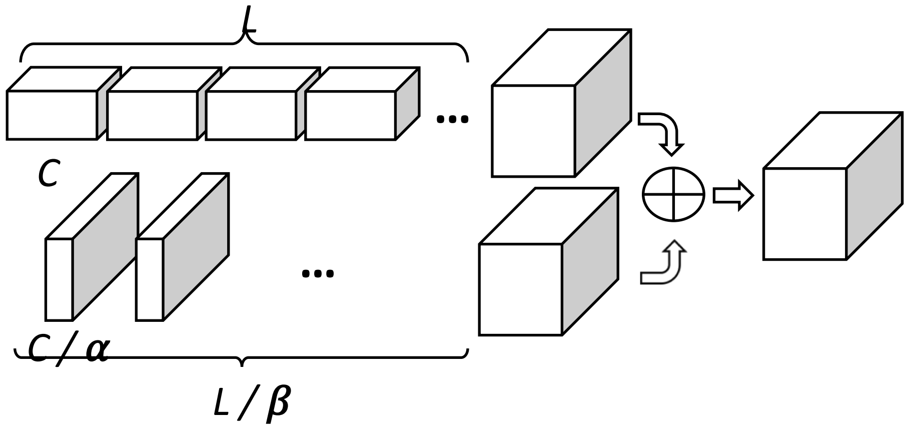


Summary

- Adaptive Computation: BlockDrop



- Efficient Multi-Scale Architectures: Big-Little Net



What’s Next?

- Big-Little Net with dynamic scale selection
- Neural architecture search: compact multi-task networks using Gumbel-Softmax
- Extension to Video Understanding

Thank you !

- C. Chen, Q. Fan, N. Mallinar, T. Sercu and R. S. Feris. "Big-Little Net: An Efficient Multi-Scale Feature Representation for Visual and Speech Recognition."ICLR 2019
- Z. Wu*, T. Nagarajan*, A. Kumar, S. Rennie, L. Davis, K. Grauman and R. S. Feris. "BlockDrop: Dynamic Inference Paths in Residual Networks." CVPR 2018, Spotlight (* equal contribution)
- Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing and R. S. Feris. “SpotTune: Transfer Learning Through Adaptive Fine-Tuning” CVPR 2019