Dynamic Neural Networks for Efficient Image and Video Classification

Rogerio Schmidt Feris

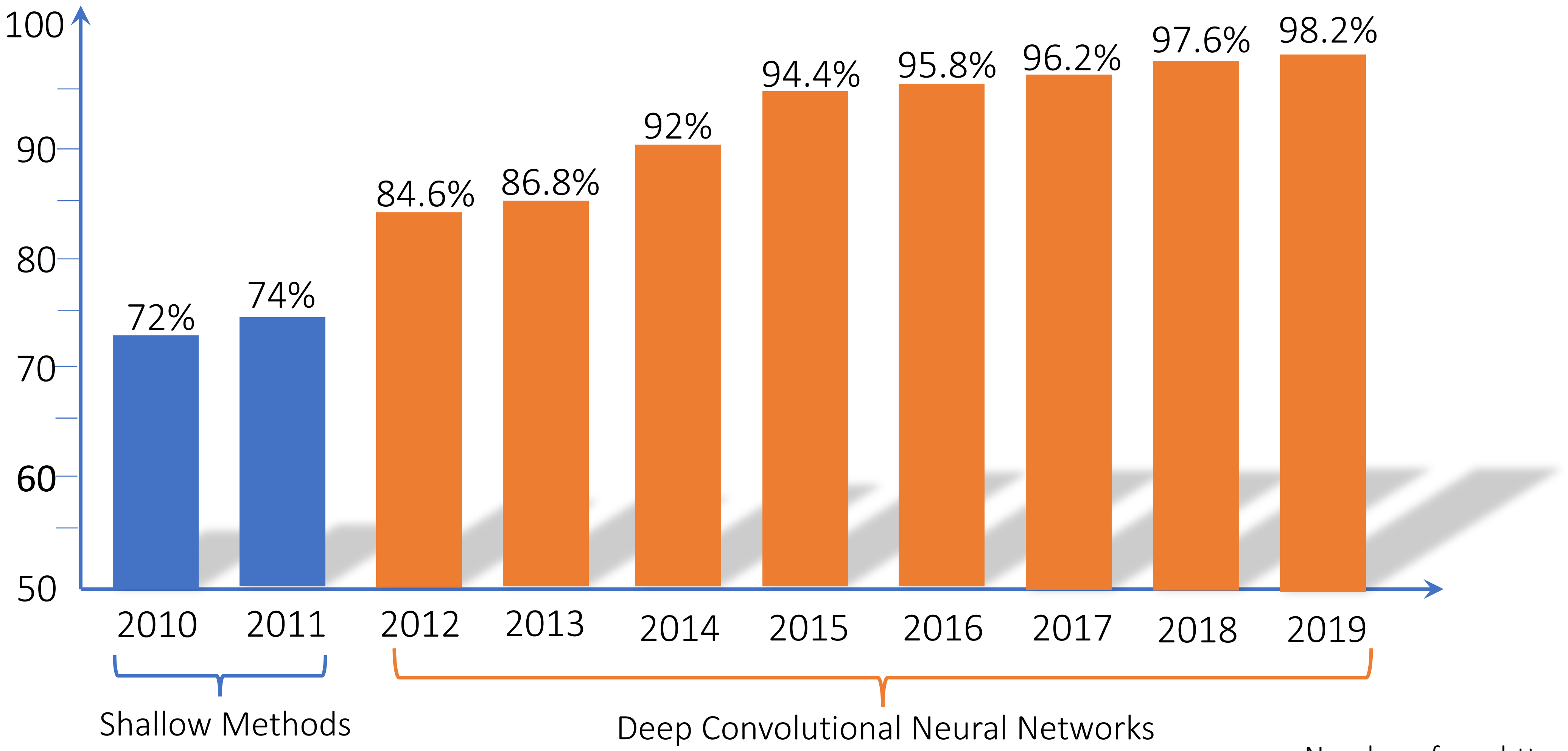
MIT-IBM Watson AI Lab

http://rogerioferis.com

LatinX in AI Research at ICML 2020

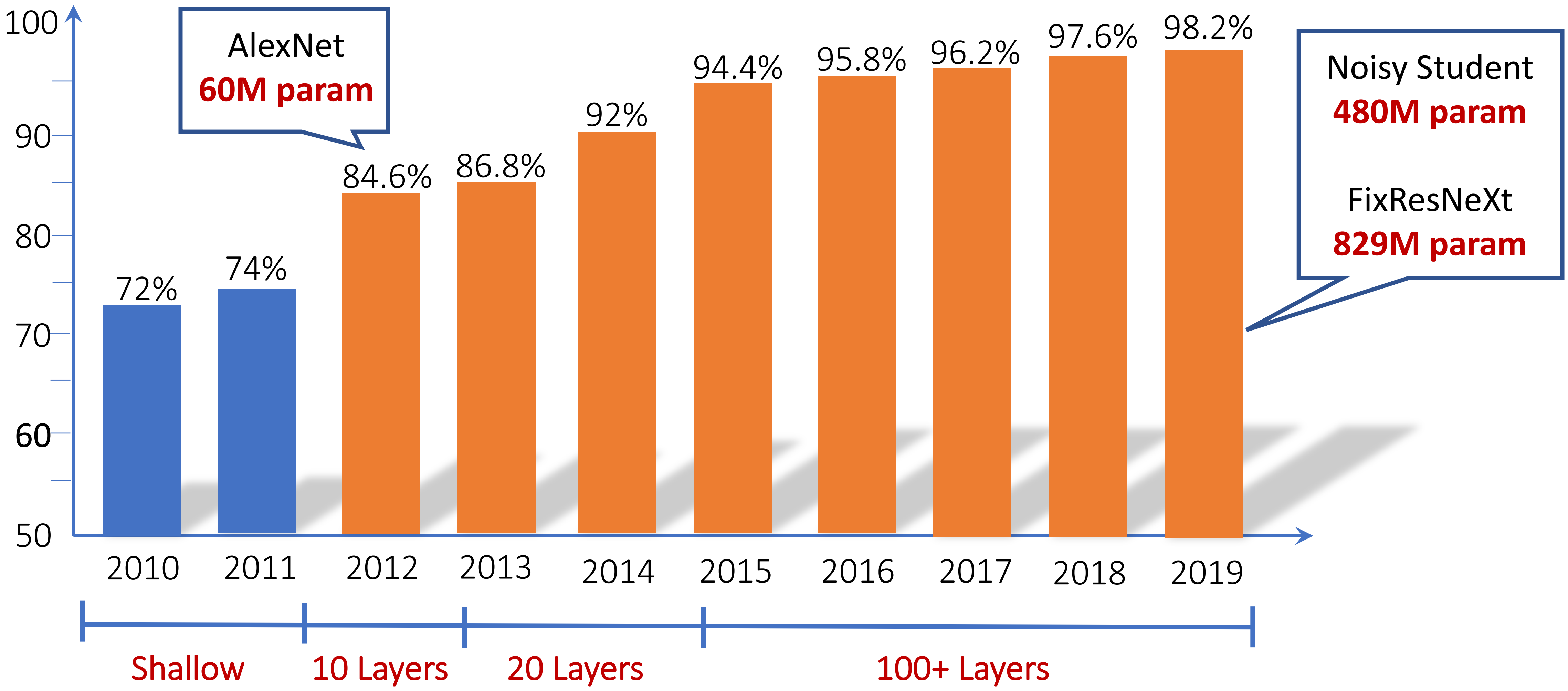
Rio Grande - RS

ImageNet Classification (top-5 accuracy)

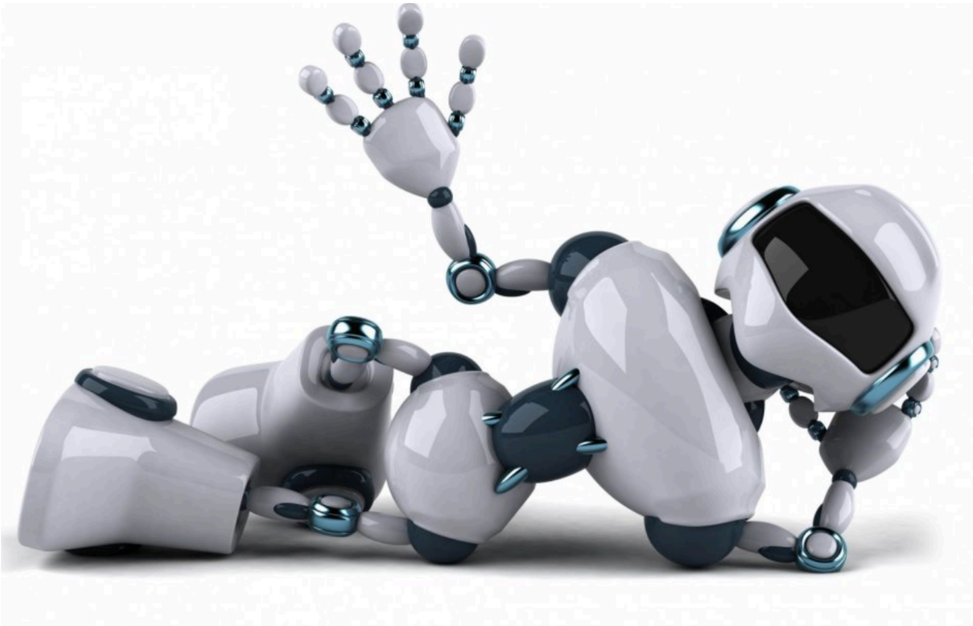


Numbers from http://paperswithcode.com

Better Results → More Complexity



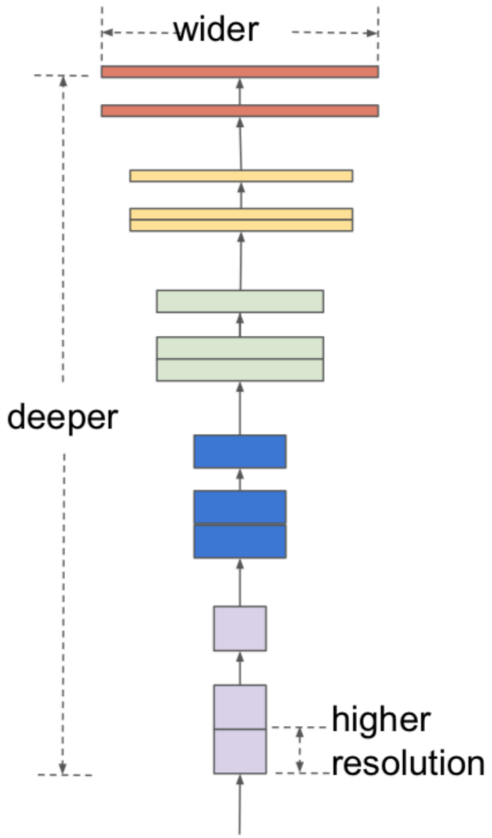
Many applications require real-time inferencing



Model Compression and Acceleration

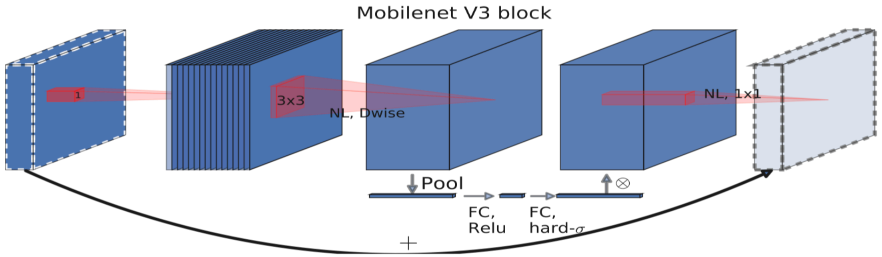
- Low-rank factorization, Knowledge Distillation, Pruning, Quantization, Neural Architecture Search, etc.

EfficientNet [Tan & Le, 2019]

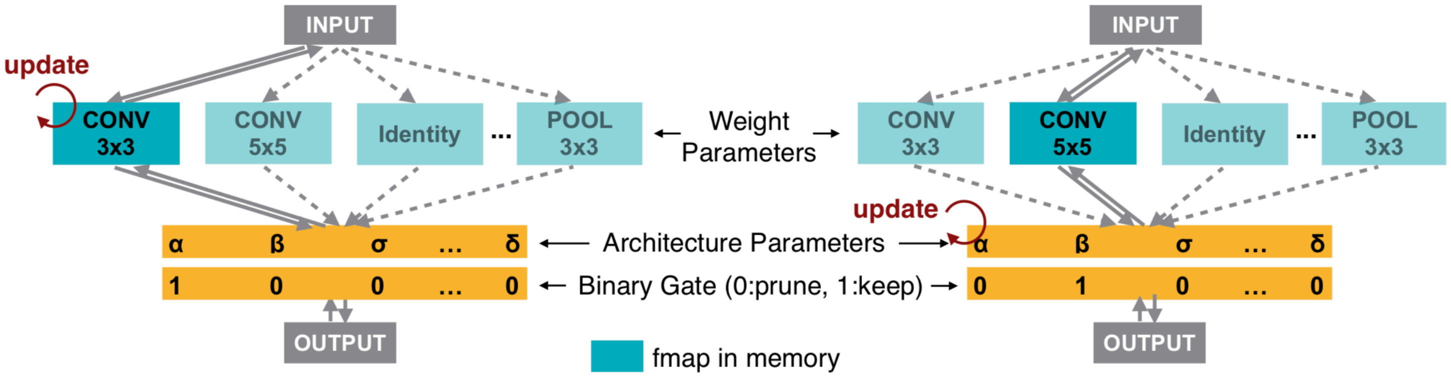


(e) compound scaling

MobileNet V3 [Howard et al, 2019]



ProxylessNAS [Cai et al, 2019]



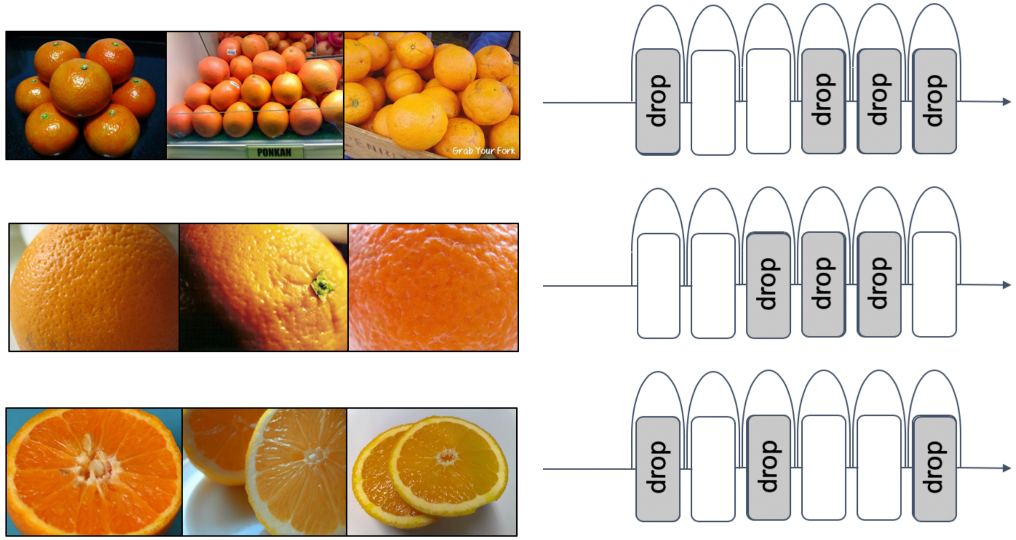
(1) Update weight parameters

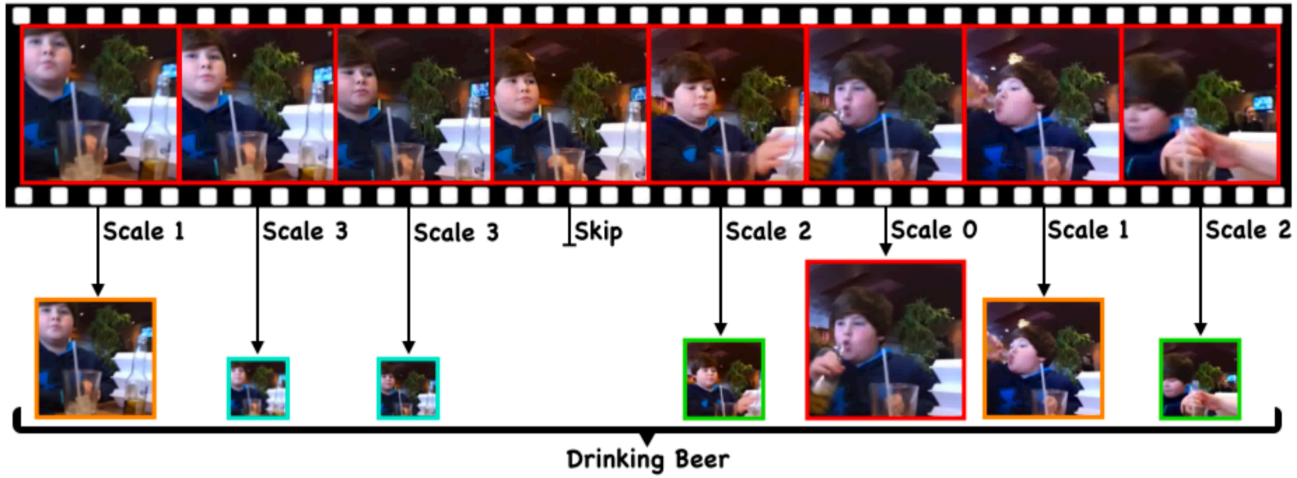
(2) Update architecture parameters

Most methods rely on one-size-fits-all networks that require the same fixed set of features to be extracted for all inputs, no matter their complexity

This talk: Dynamic (Adaptive) Neural Networks for Efficient Image and Video Classification

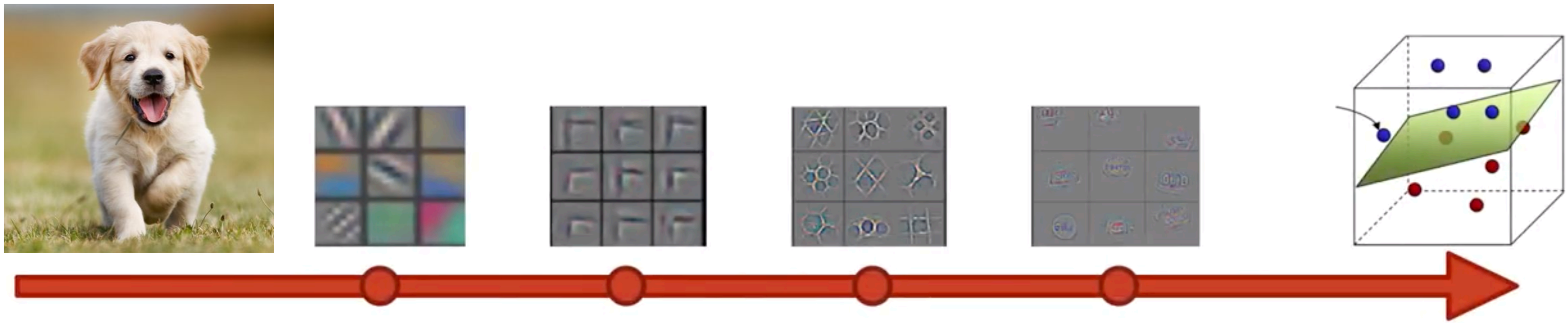
- Networks models that are dynamically reconfigured depending on the input





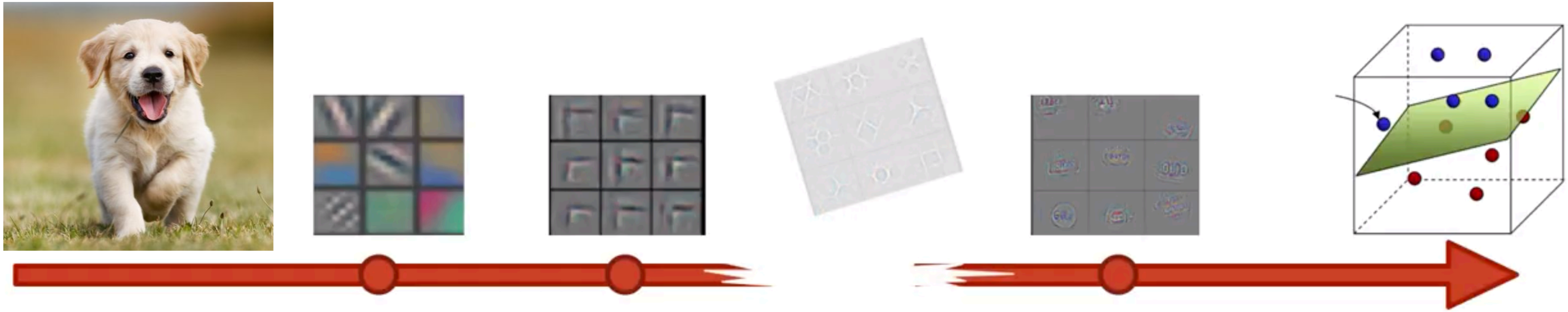
- Conditional Computation [Bengio et al, 2013/2016]

Feed-Forward Convolutional Neural Networks



Adapted from Veit et al

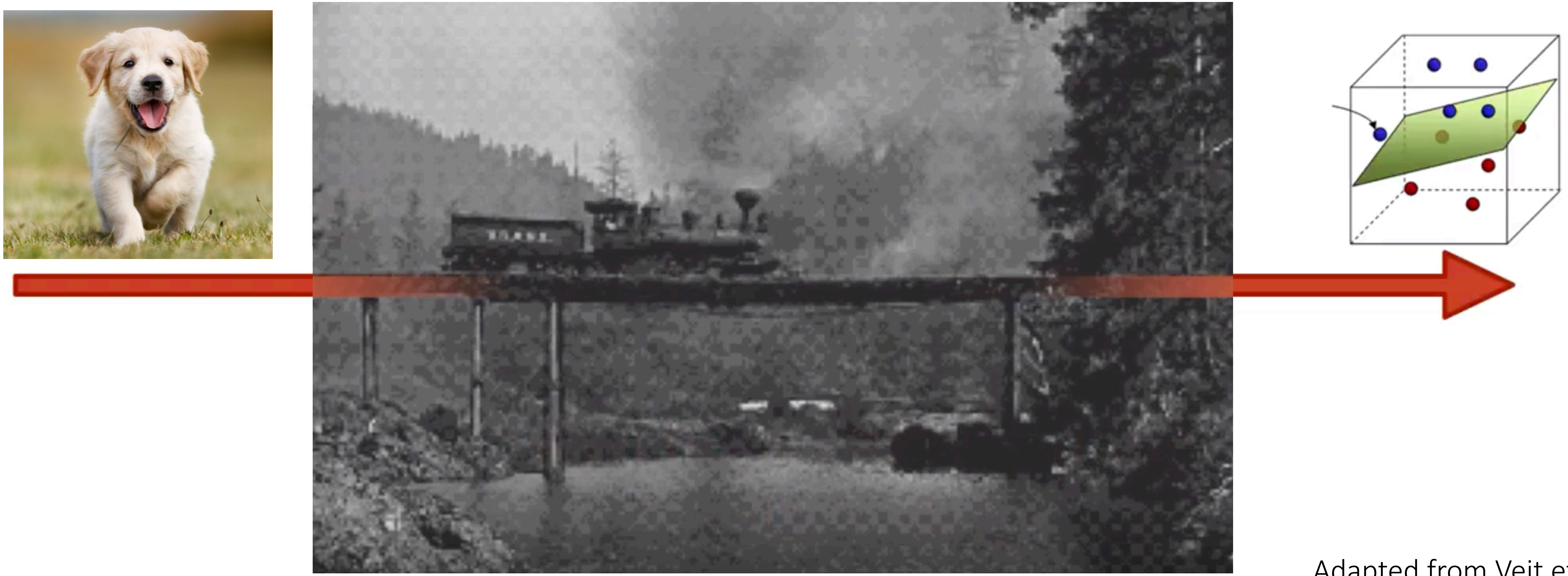
Feed-Forward Convolutional Neural Networks



What happens when we delete a step?

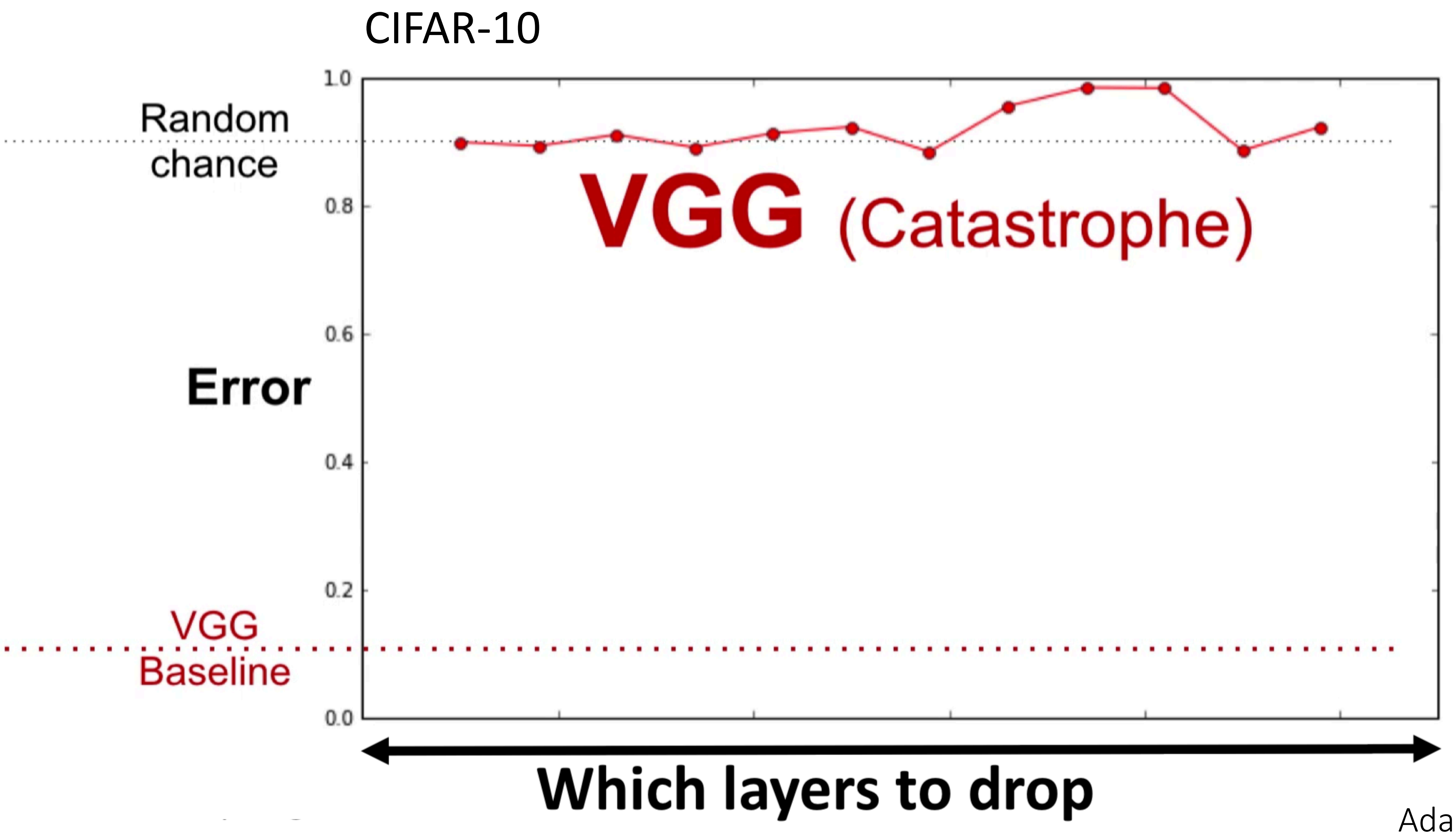
Adapted from Veit et al

Feed-Forward Convolutional Neural Networks



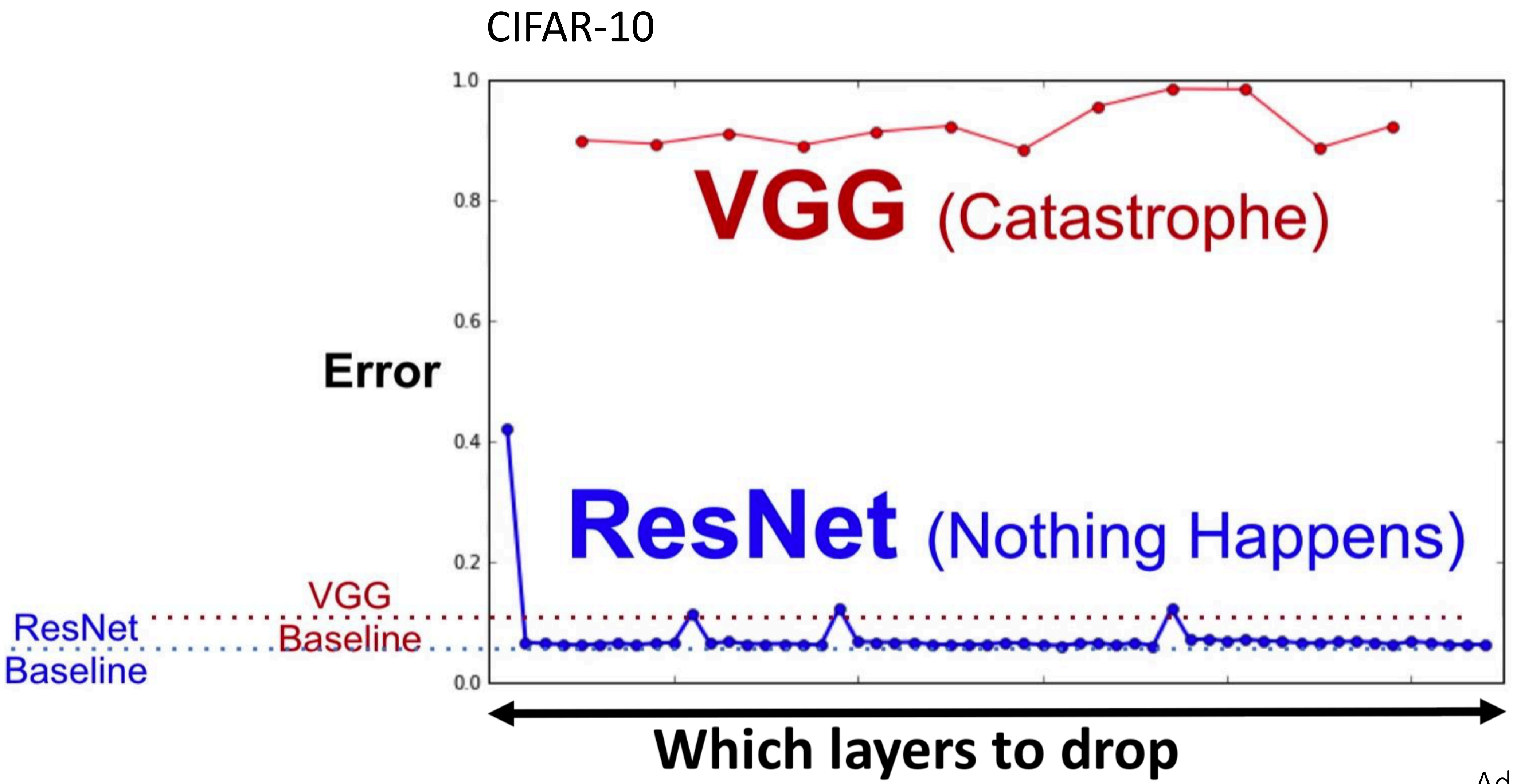
Adapted from Veit et al

What happens if we delete a layer at test time?



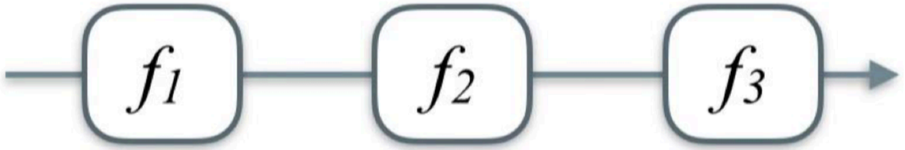
Adapted from Veit et al

What happens if we delete a layer at test time?

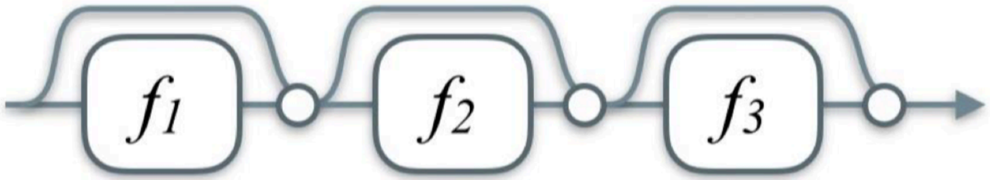


Adapted from Veit et al

Why does this happen?



VGG

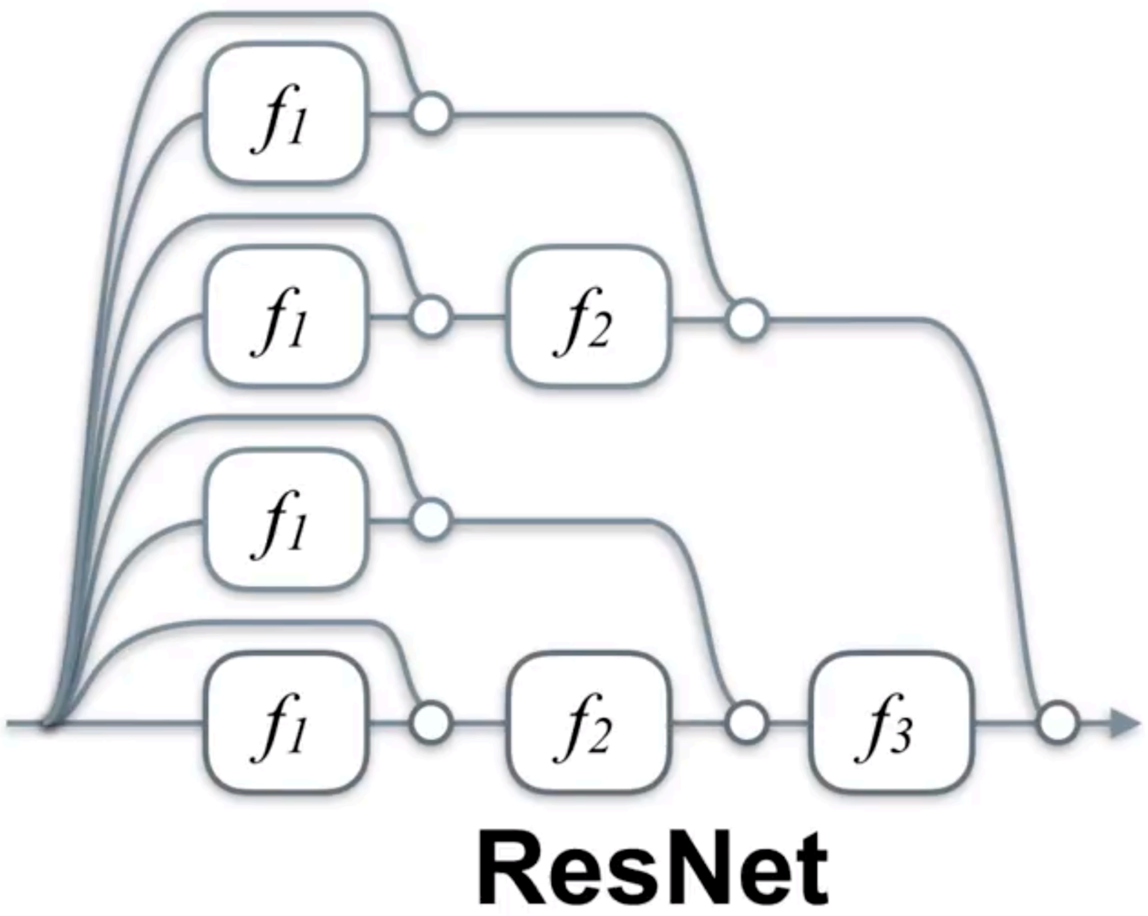


ResNet

Adapted from Veit et al

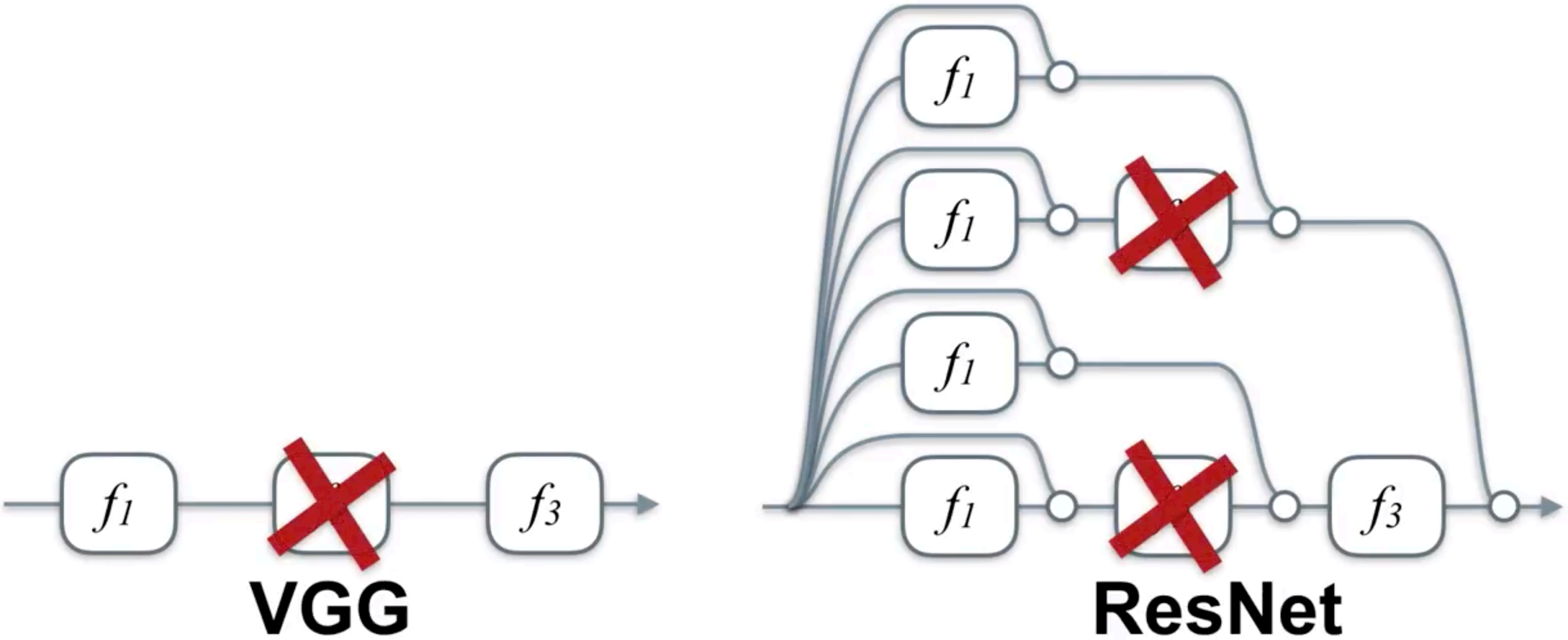
Why does this happen?

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



Adapted from Veit et al

Deletion of a Layer

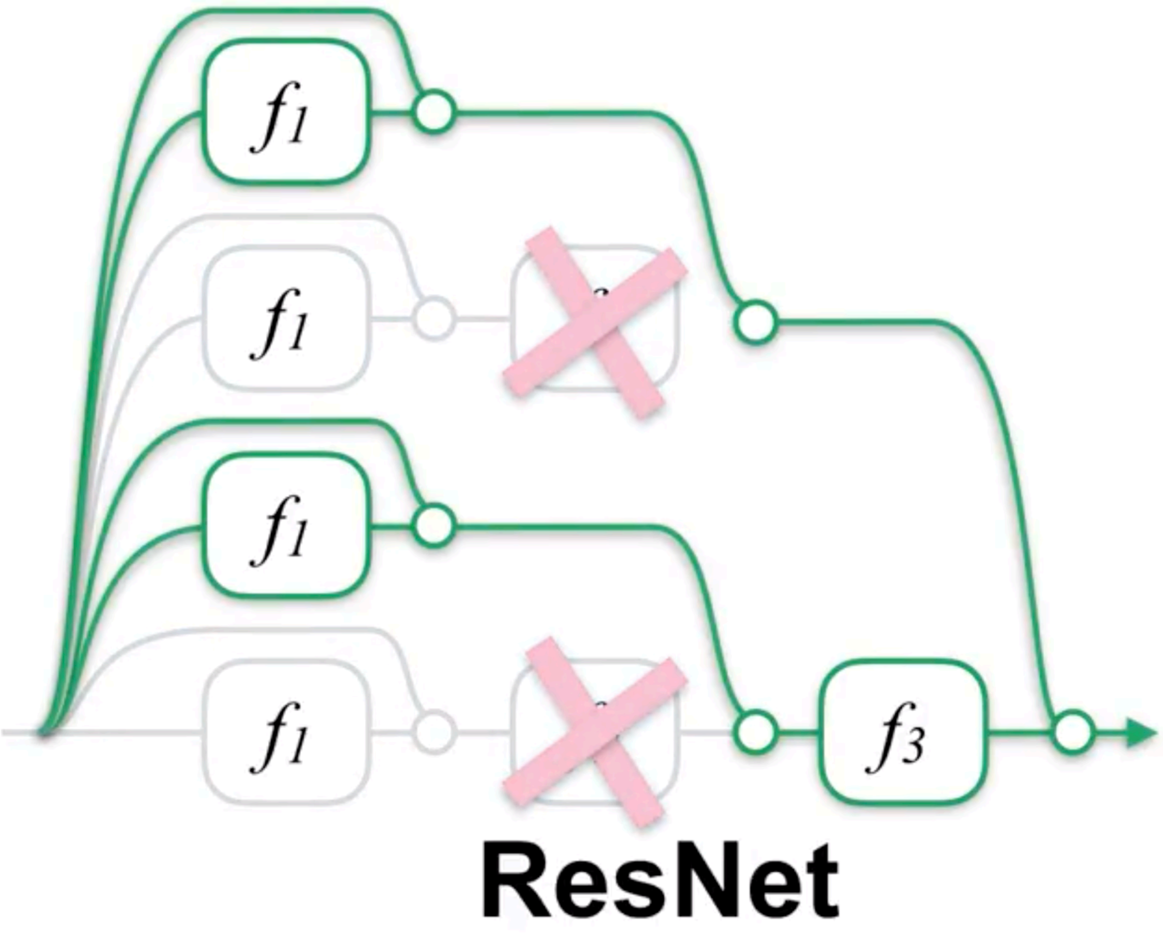


Adapted from Veit et al

Deletion of a Layer

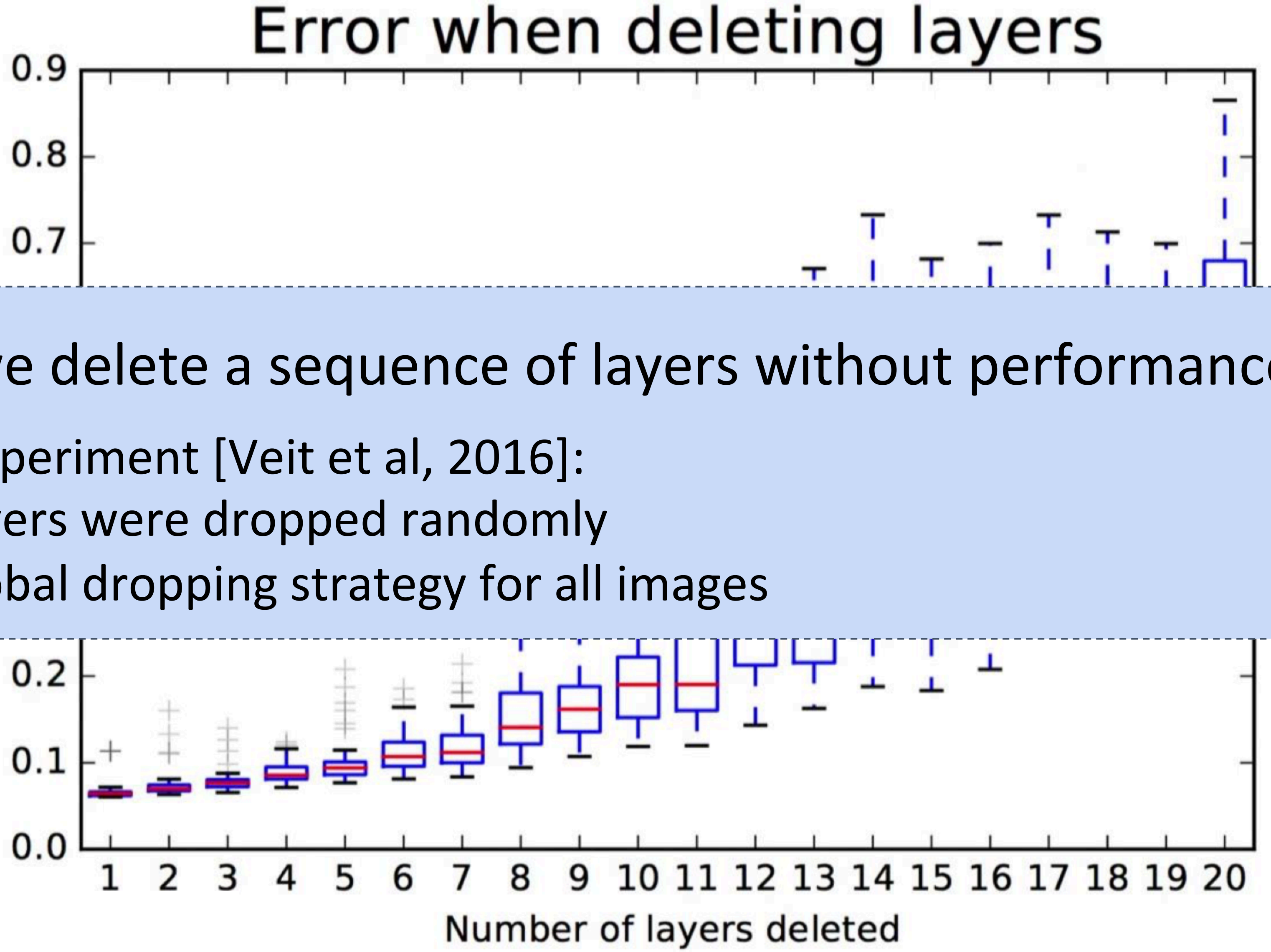
Only half of the paths are affected

All 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

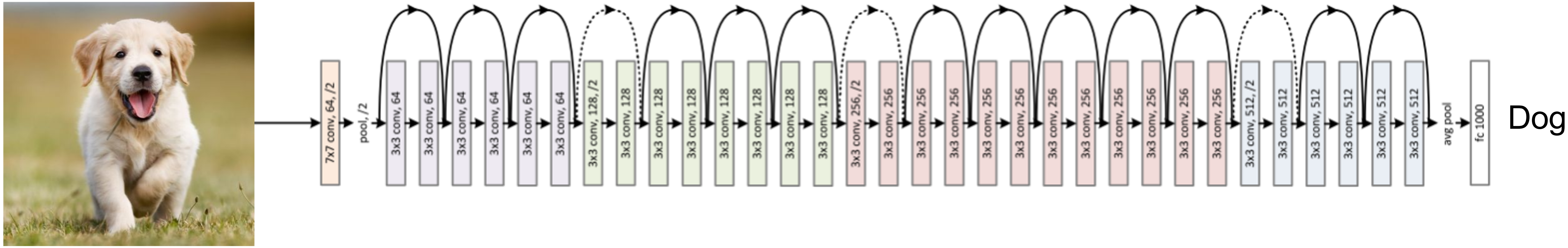
BlockDrop: Dynamic Inference Paths in Residual Networks

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

CVPR 2018

* Authors contributed equally

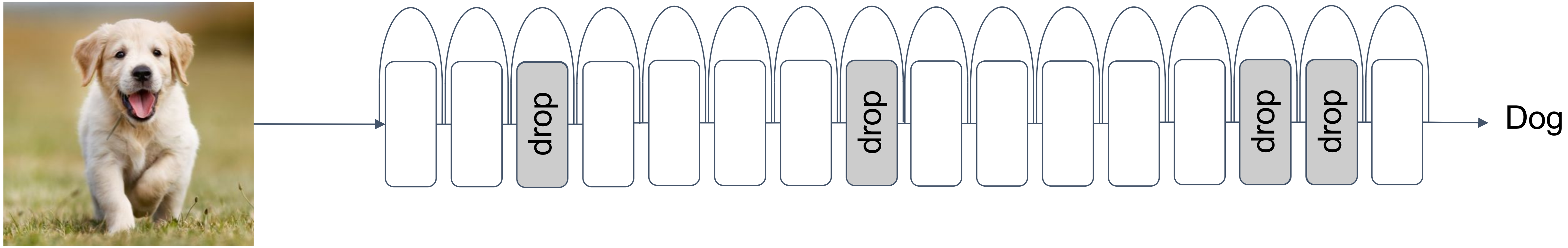
BlockDrop: Dynamic Inference Paths in Residual Networks [CVPR 2018]



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 [CVPR 2018]



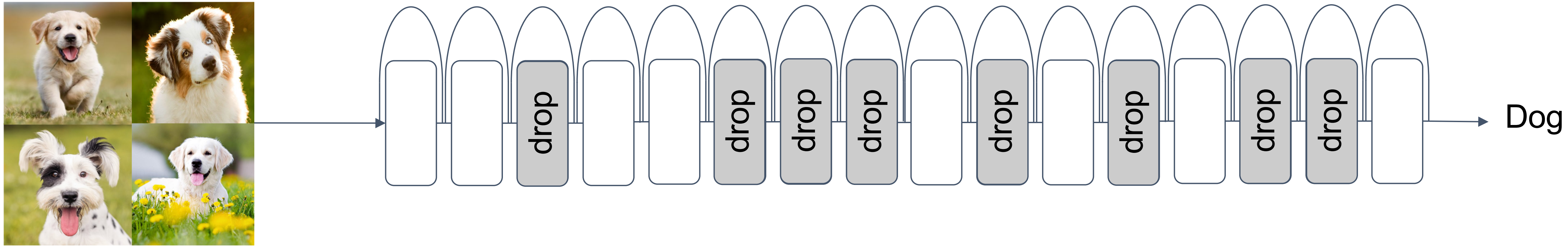
"Dropping some blocks during testing doesn't hurt performance much"

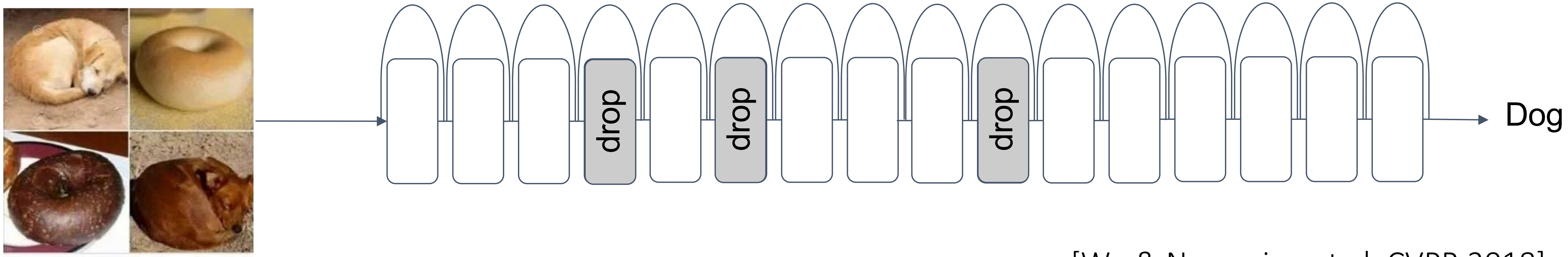
(Veit et al., NIPS 16)

[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?





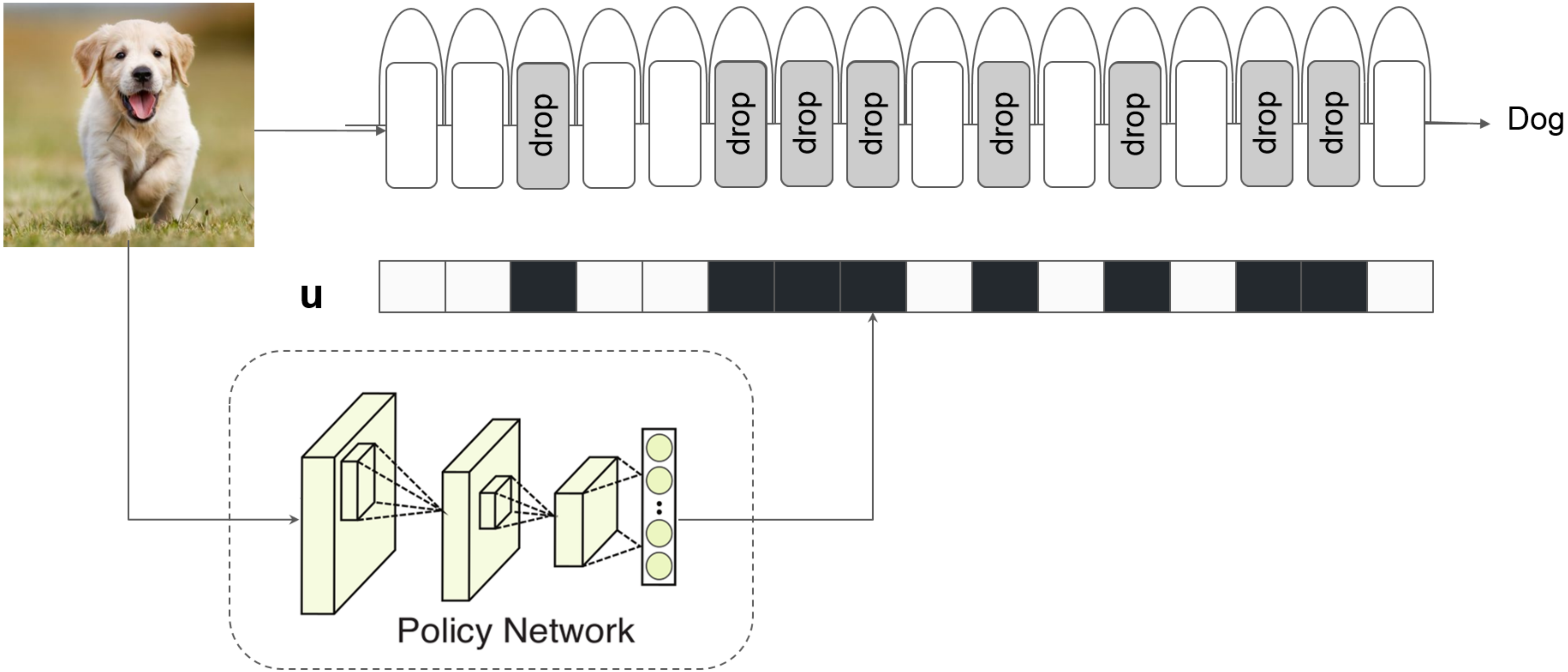
[Wu & Nagarajan et al, CVPR 2018]

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]

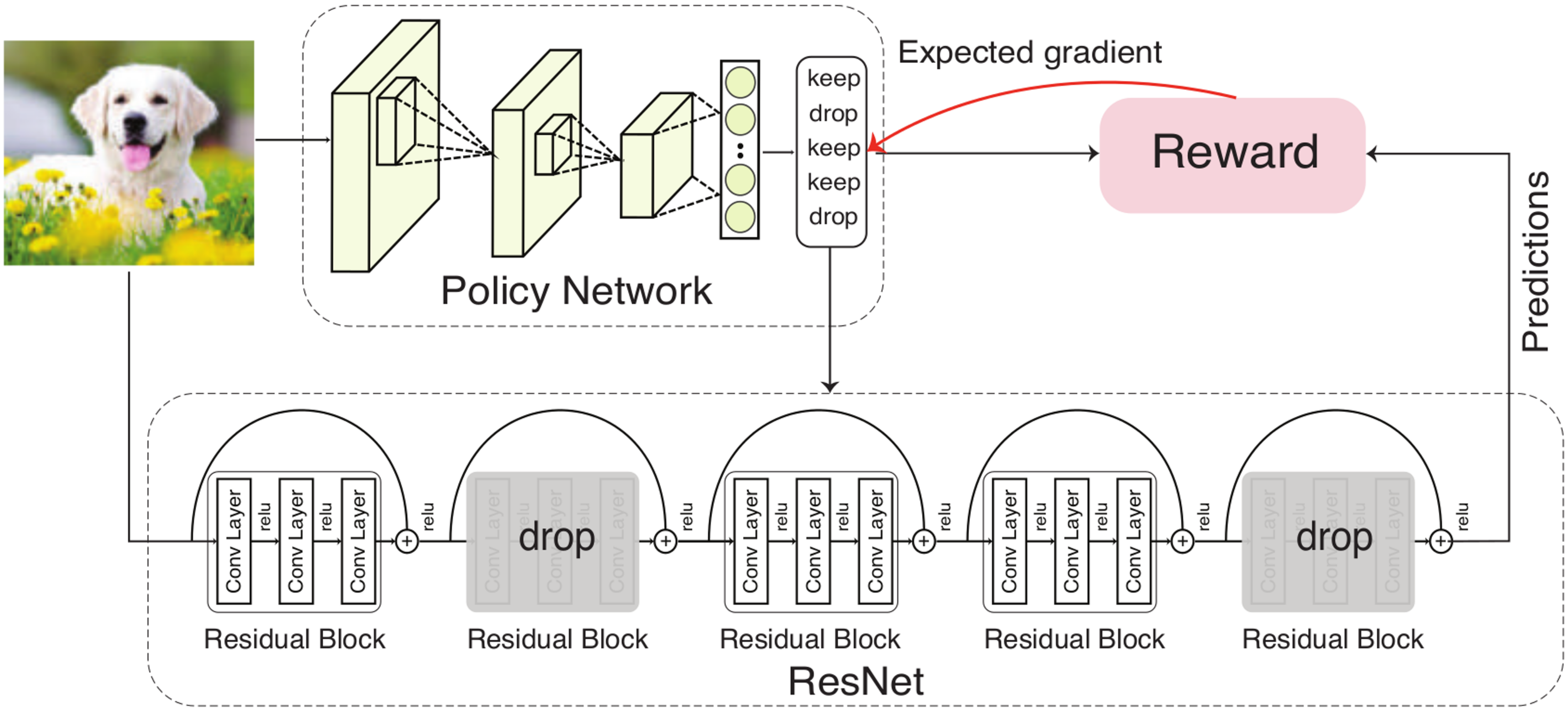
BlockDrop: Dynamic Inference Paths in Residual Networks



[Wu & Nagarajan et al, CVPR 2018]

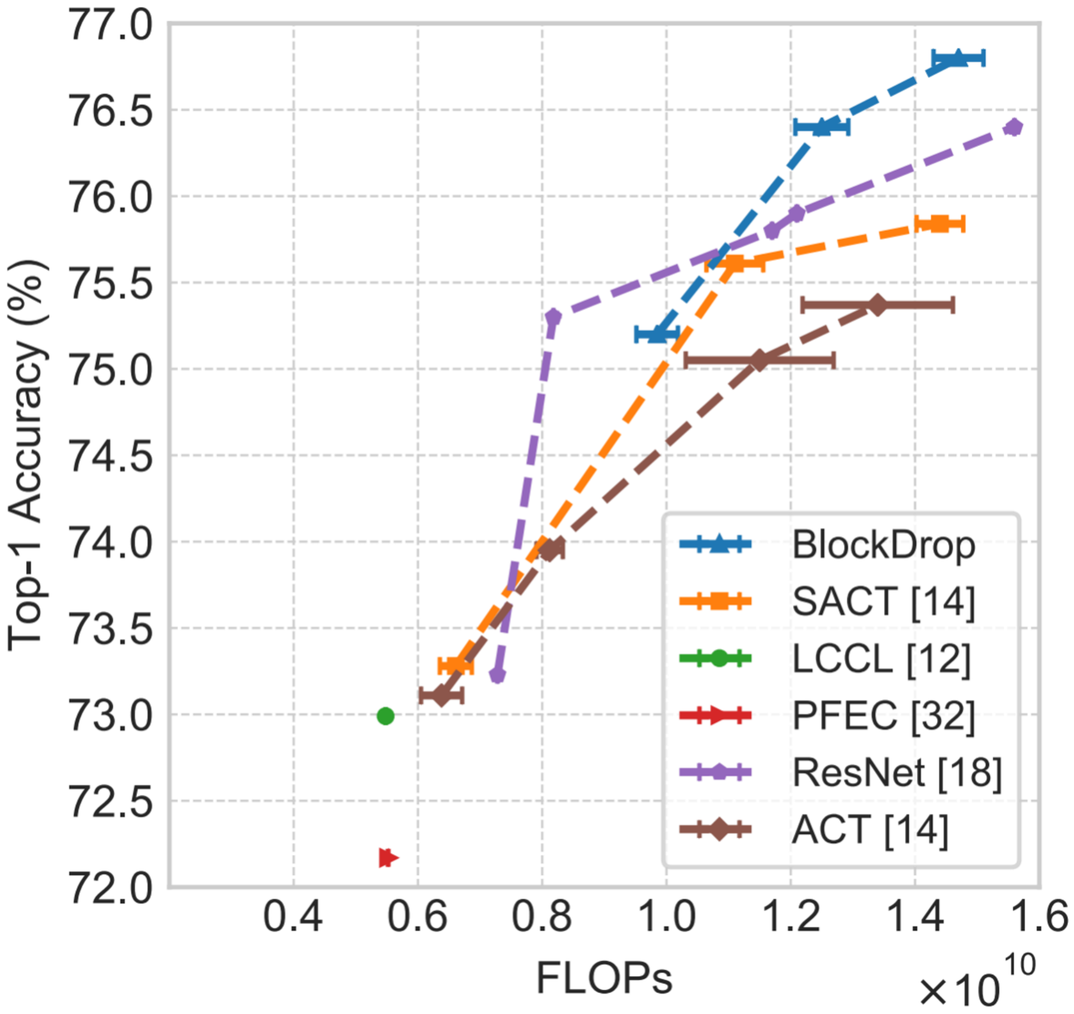
BlockDrop: Dynamic Inference Paths in Residual Networks [CVPR 2018]

Policy Network Training through Reinforcement Learning



[Wu & Nagarajan et al, CVPR 2018]

BlockDrop: Dynamic Inference Paths in Residual Networks



Results on ImageNet:

20% - 36% computational savings (FLOPs)

Complementary to other model compression techniques

[Wu & Nagarajan et al, CVPR 2018]

SpotTune: Transfer Learning through Adaptive Fine-Tuning

Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman,
Tajana Rosing, Rogerio Feris

CVPR 2019

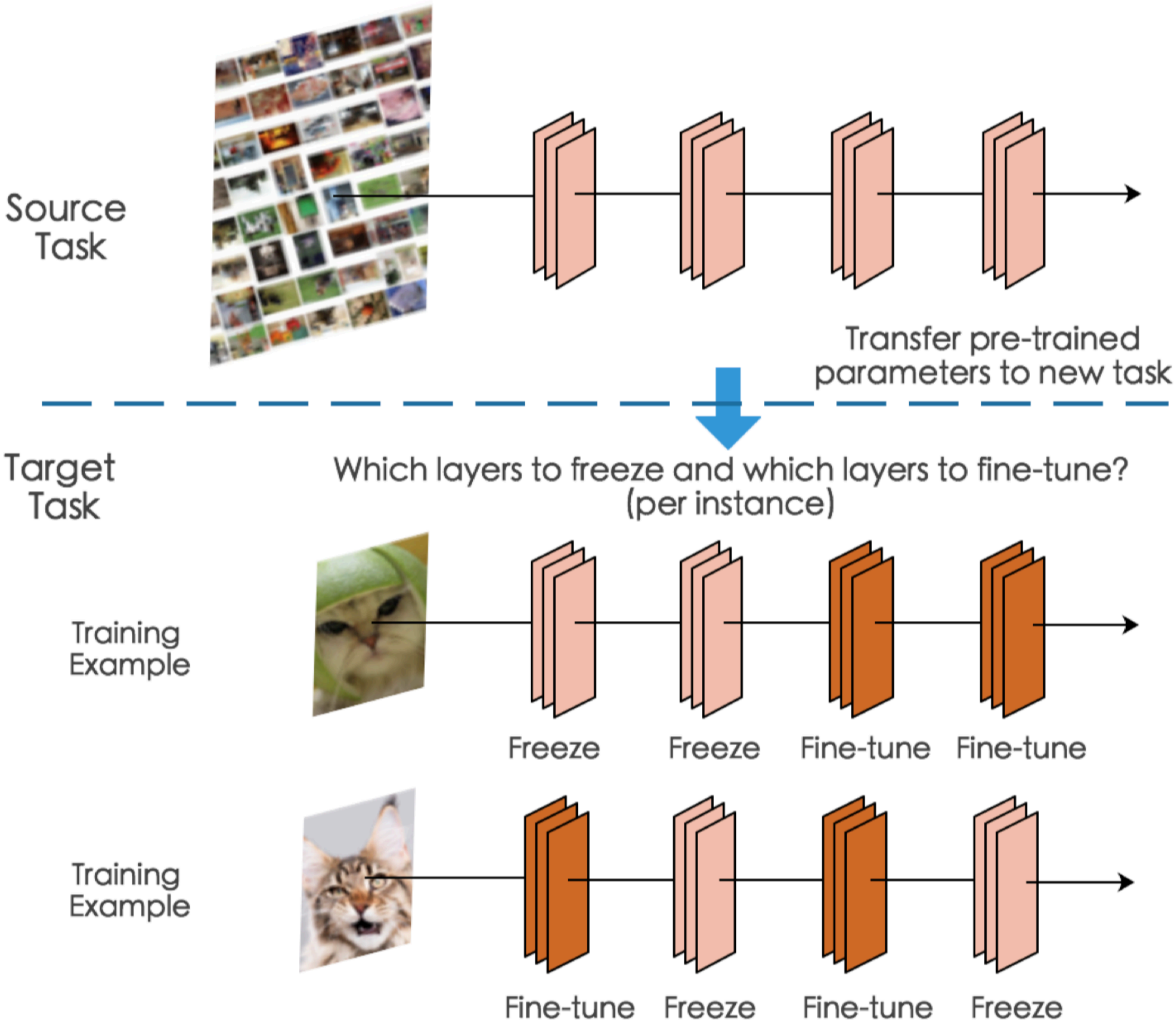
Data Efficiency: Transfer Learning

- 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)

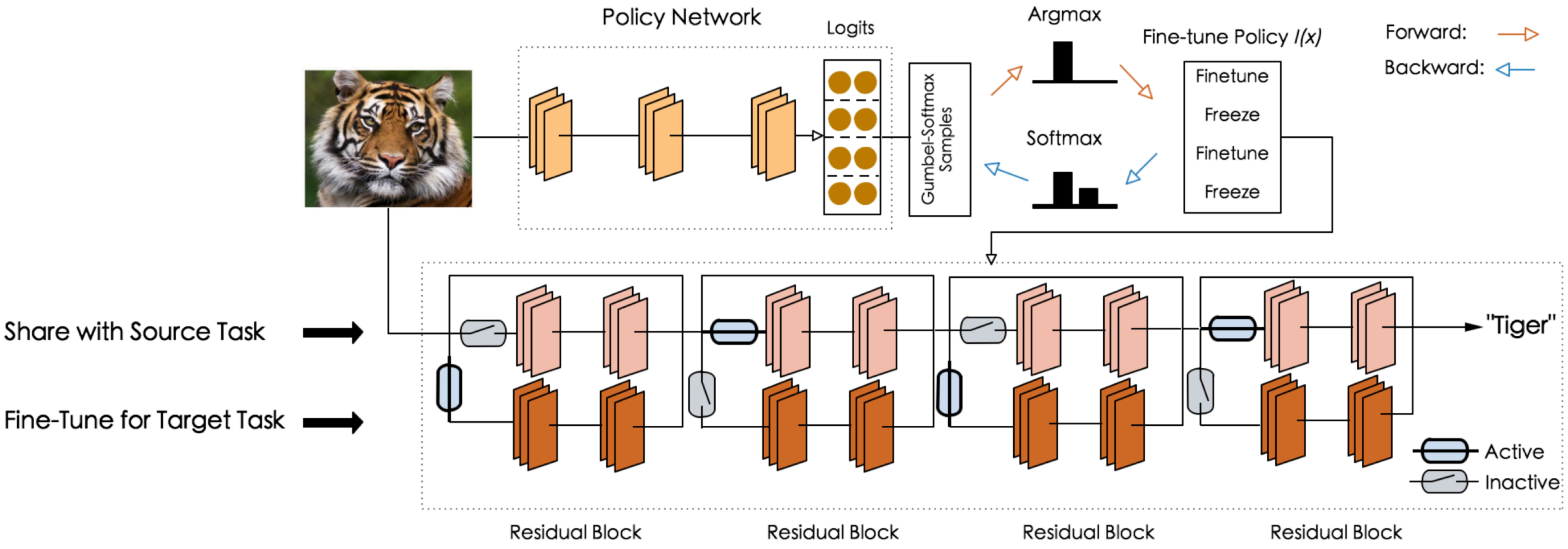
- We propose *SpotTune*, a method that automatically decides, per training example, which layers of a pre-trained model should have their parameters frozen (shared with the source domain) or fine-tuned (adapted to the target domain)

[Guo et al, CVPR 2019]



[Guo et al, CVPR 2019]

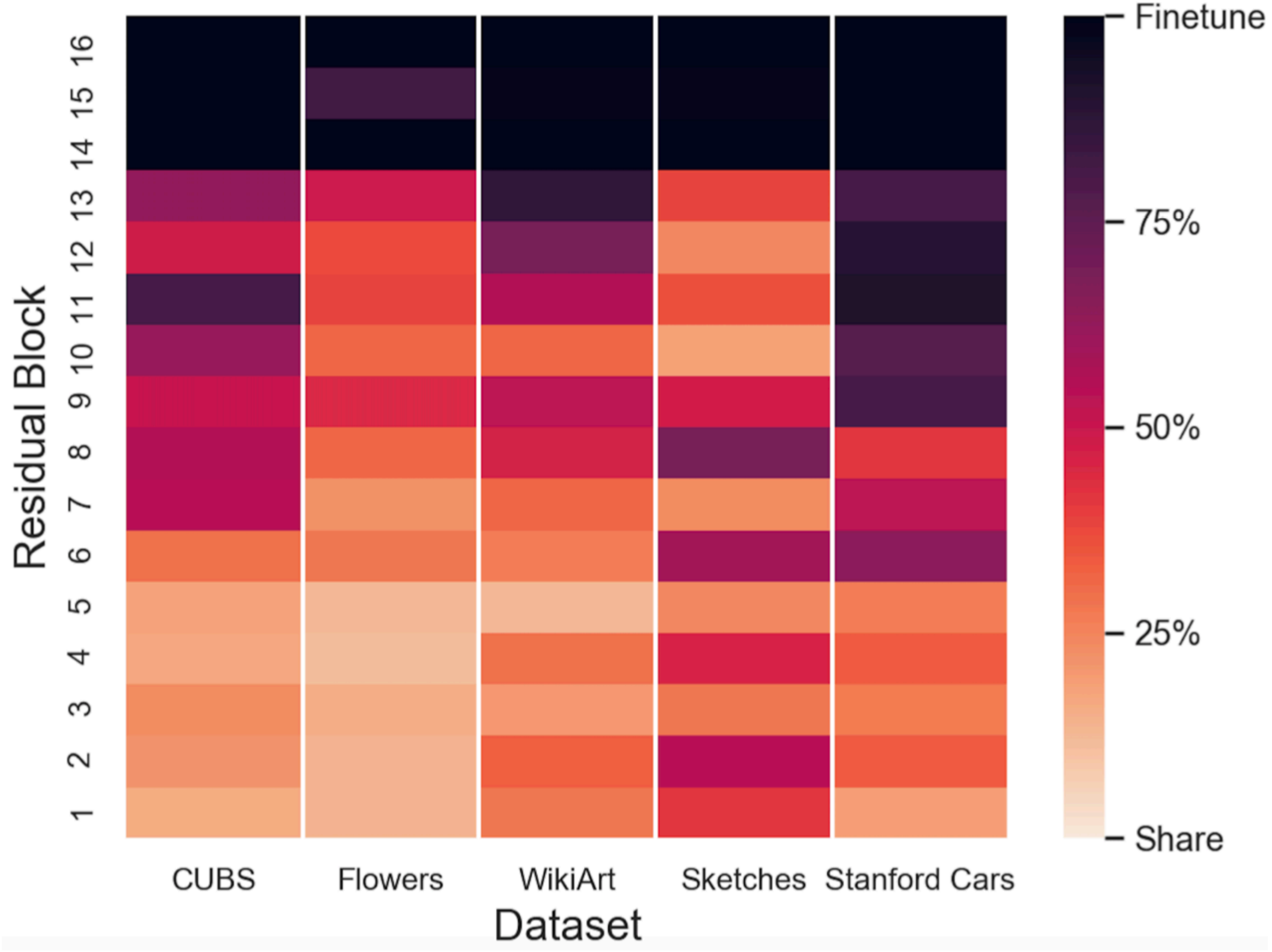
SpotTune: Transfer Learning through Adaptive Fine-Tuning



[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]

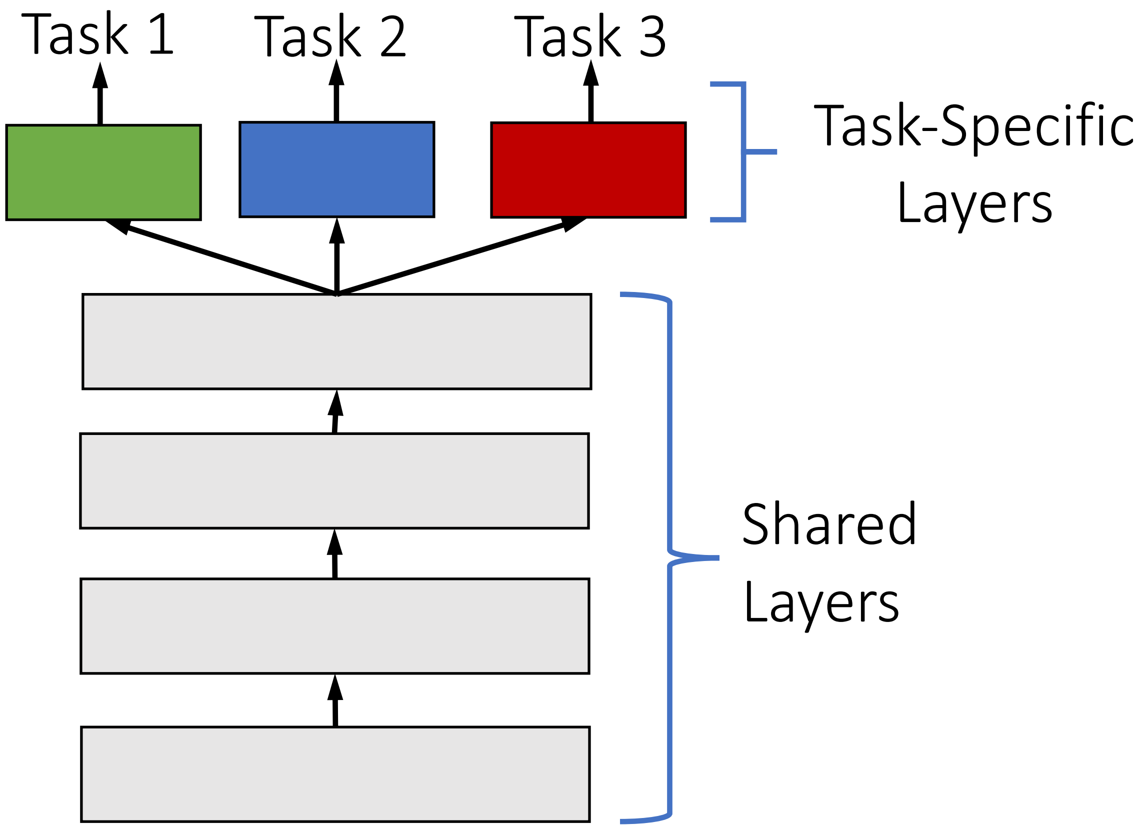
AdaShare: Learning What to Share for Efficient Multi-Task Learning

Ximeng Sun, Rameswar Panda, Rogerio Feris, Kate Saenko

NeurIPS 2020

Hard Parameter Sharing

- Hand-designed architectures composed of base layers that are shared across tasks and specialized branches that learn task-specific features.

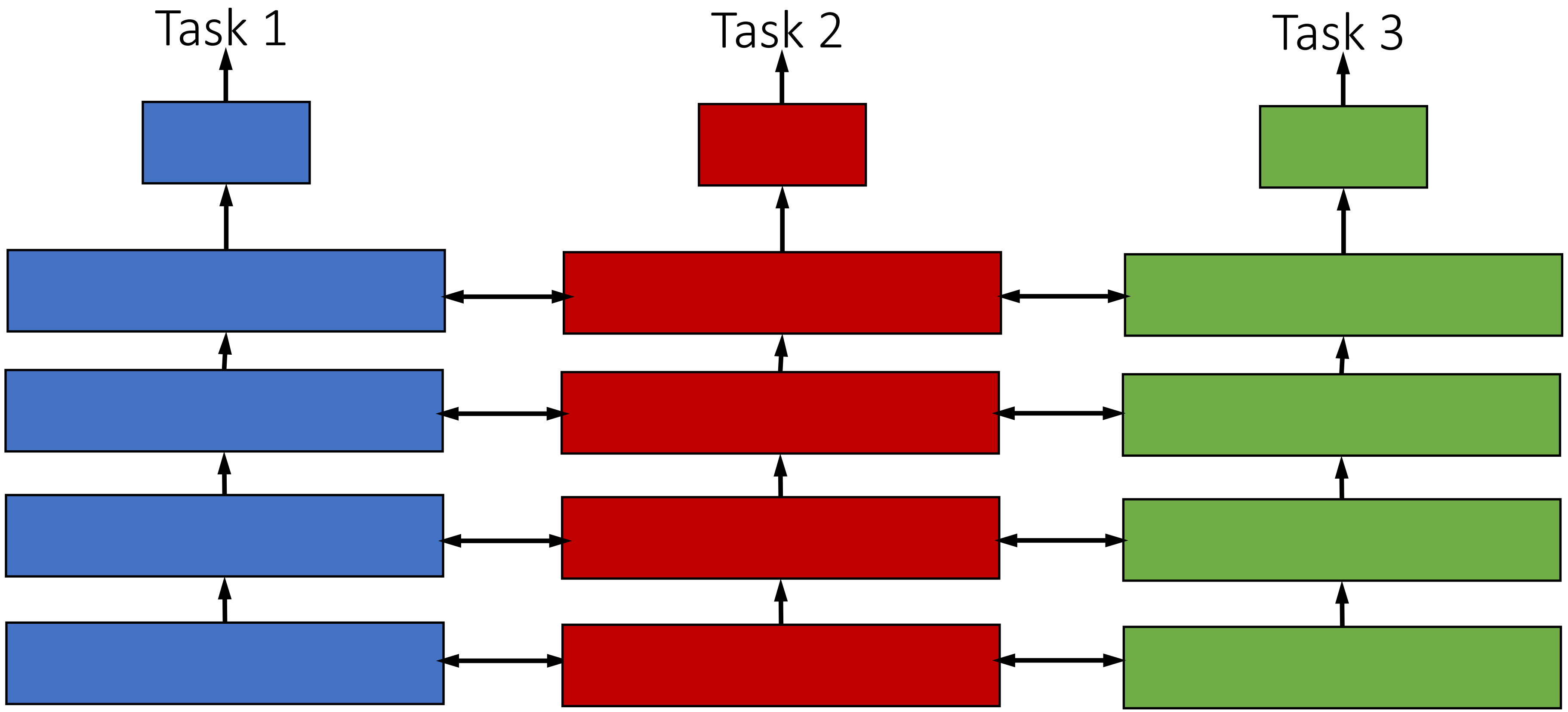


- Performance depends on "where to branch" in the network [Misra et al, 2016]
- The space of possible branching architectures is combinatorially large !

Soft Parameter Sharing

- Network column for each task and a mechanism for feature sharing between columns.

Number of parameters grow linearly with the number of tasks !

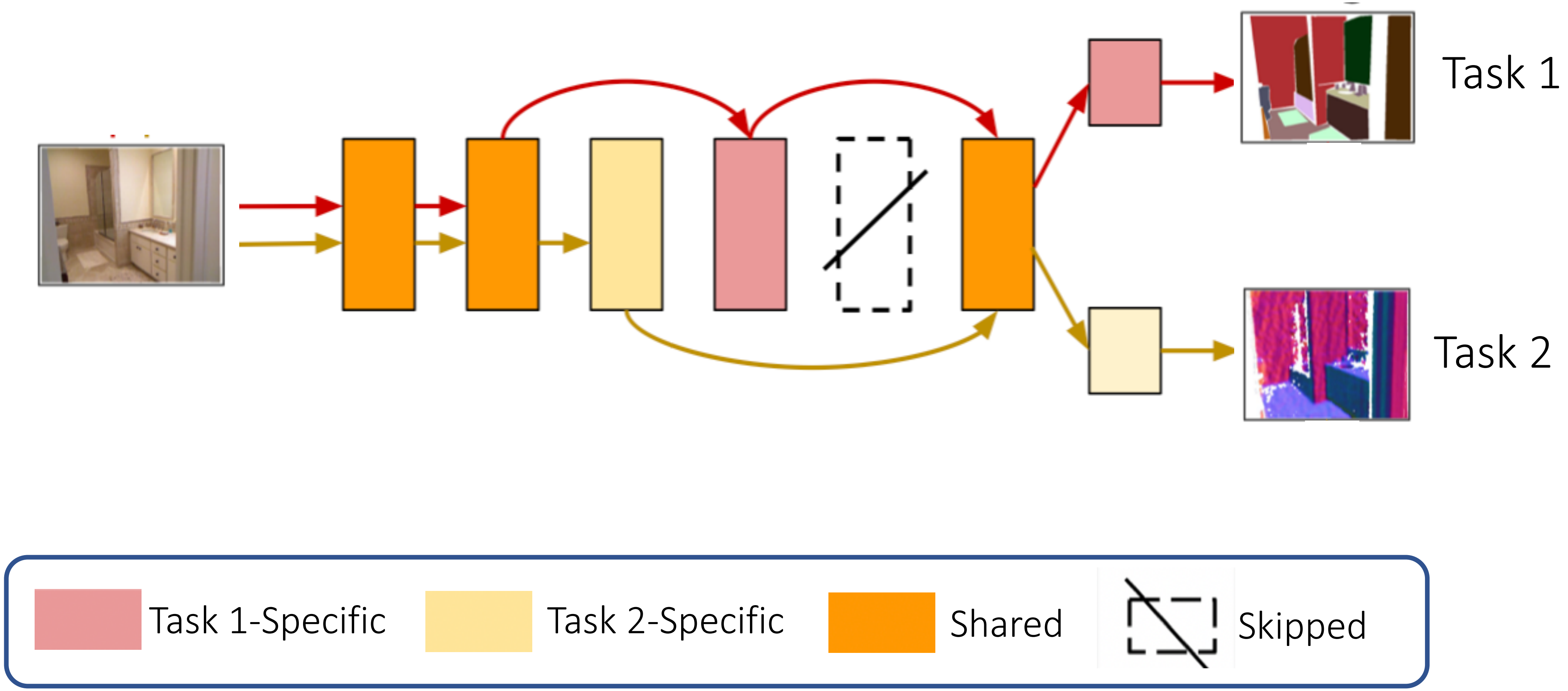


Problem

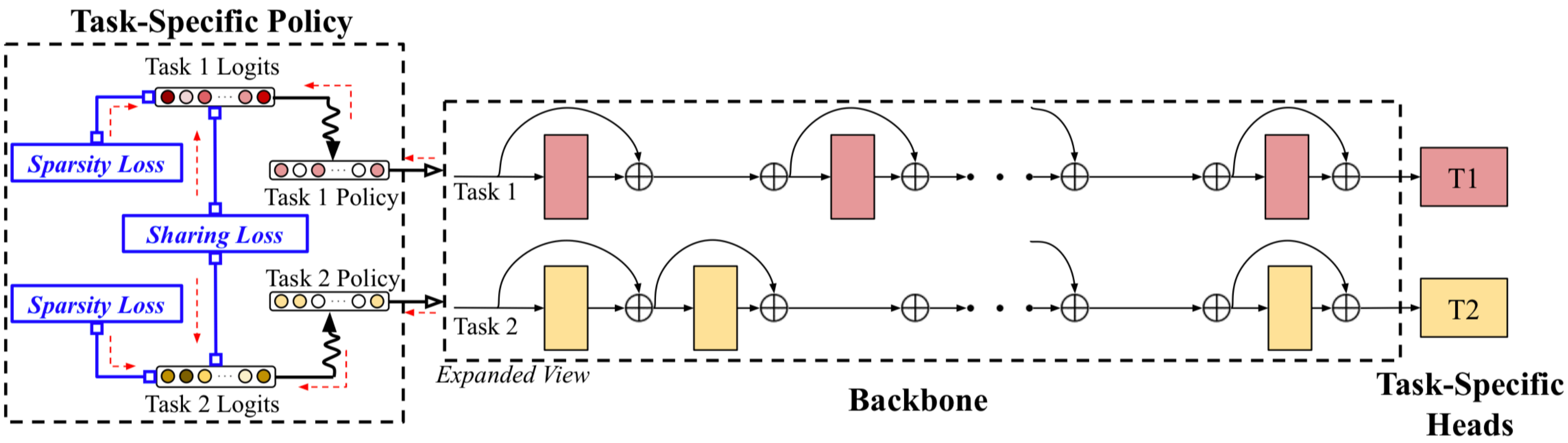
Can we determine which layers in the network should be shared across which tasks and which layers should be task-specific to achieve the best accuracy/memory footprint trade-off for scalable and efficient multi-task learning?

Proposed Approach: AdaShare

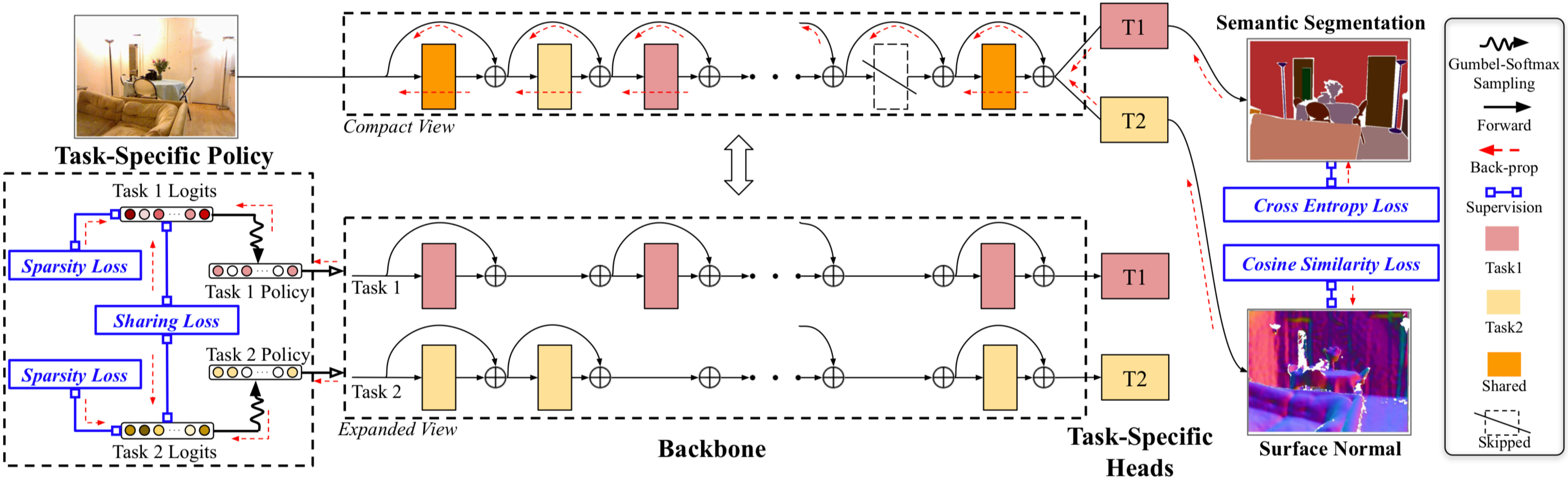
- Single network that supports separate execution paths for different tasks



AdaShare: Learning what to Share in Multi-Task Learning



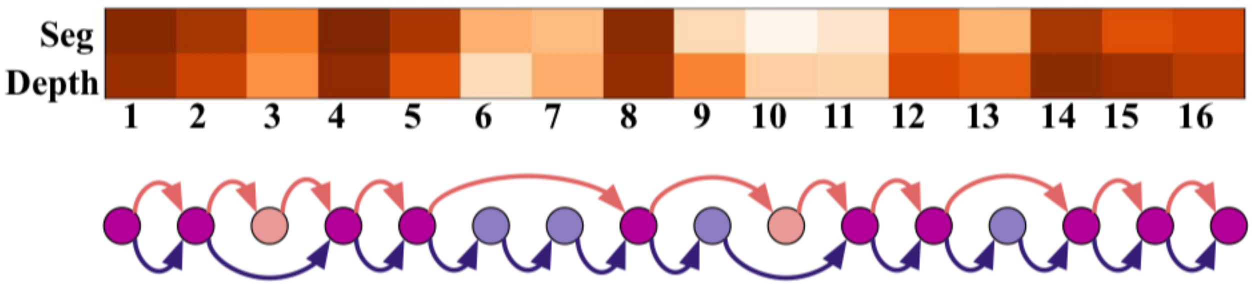
AdaShare: Learning what to Share in Multi-Task Learning



AdaShare: Experimental Results

- **CityScapes [2 tasks].** *AdaShare* achieves the best performance on 5 out of 7 metrics using less than 1/2 parameters of most baselines.

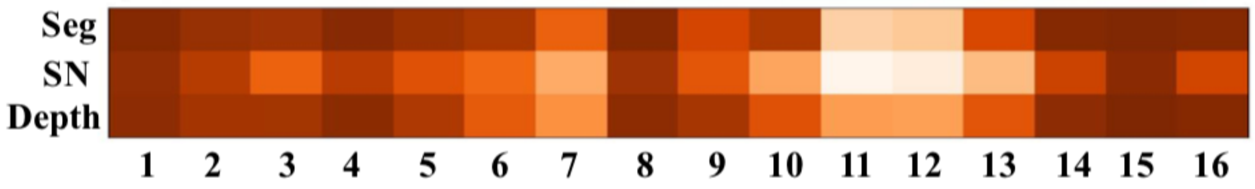
Model	# Params ↓	Semantic Seg.		Depth Prediction				
		mIoU ↑	Pixel Acc ↑	Error↓		δ, within ↑		
				Abs	Rel	1.25	1.25^2	1.25^3
Single-Task	2	40.2	74.7	0.017	0.33	70.3	86.3	93.3
Multi-Task	**1**	37.7	73.8	0.018	0.34	72.4	88.3	94.2
Cross-Stitch	2	40.3	74.3	**0.015**	**0.30**	74.2	89.3	**94.9**
Sluice	2	39.8	74.2	0.016	0.31	73.0	88.8	94.6
NDDR-CNN	2.07	**41.5**	74.2	0.017	0.31	74.0	89.3	94.8
MTAN	2.41	40.8	74.3	**0.015**	0.32	75.1	89.3	94.6
AdaShare	**1**	**41.5**	**74.9**	0.016	0.33	**75.5**	**89.8**	**94.9**

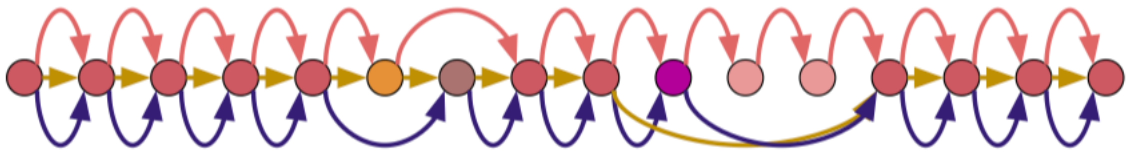


AdaShare: Experimental Results

- **NYU v2 [3 tasks].** AdaShare achieves the best performance on 10 out of 12 metrics using less than 1/3 parameters of most baselines.

Model	# Params ↓	Semantic Seg.		Surface Normal Prediction					Depth Prediction				
				Error ↓		θ, within ↑			Error ↓		δ, within ↑		
		mIoU ↑	Pixel Acc ↑	Mean	Median	11.25°	22.5°	30°	Abs	Rel	1.25	1.25^2	1.25^3
Single-Task	3	27.5	58.9	17.5	15.2	34.9	73.3	85.7	0.62	0.25	57.9	85.8	95.7
Multi-Task	**1**	24.1	57.2	**16.6**	13.4	42.5	73.2	84.6	0.58	0.23	62.4	88.2	96.5
Cross-Stitch	3	25.4	57.6	17.2	14.0	41.4	70.5	82.9	0.58	0.23	61.4	88.4	95.5
Sluice	3	23.8	56.9	17.2	14.4	38.9	71.8	83.9	0.58	0.24	61.9	88.1	96.3
NDDR-CNN	3.15	21.6	53.9	17.1	14.5	37.4	**73.7**	**85.6**	0.66	0.26	55.7	83.7	94.8
MTAN	3.11	26.0	57.2	**16.6**	13.0	43.7	73.3	84.4	0.57	0.25	62.7	87.7	95.9
AdaShare	**1**	**30.2**	**62.4**	**16.6**	**12.9**	**45.0**	71.7	83.0	**0.55**	**0.20**	**64.5**	**90.5**	**97.8**

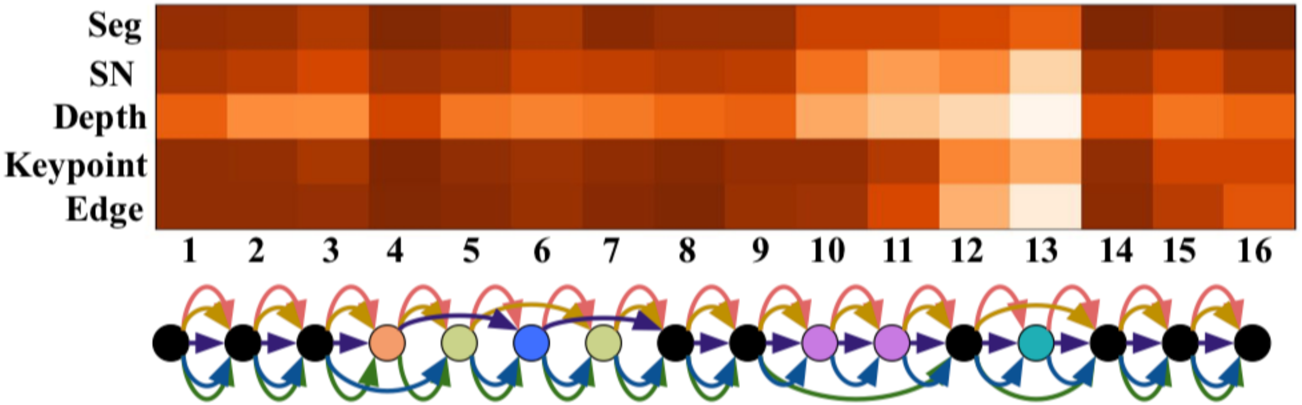




AdaShare: Experimental Results

- **Tiny-Taskonomy [5 Tasks].** AdaShare outperforms the baselines on 3 out of 5 tasks using less than 1/5 parameters of most baselines.

Models	# Params ↓	Seg ↓	SN ↑	Depth ↓	Keypoint ↓	Edge ↓
Single-Task	5	0.575	**0.707**	**0.022**	0.197	0.212
Multi-Task	**1**	0.587	0.702	0.024	0.194	0.201
Cross-Stitch	5	0.560	0.684	**0.022**	0.202	0.219
Sluice	5	0.610	0.702	0.023	**0.192**	0.198
NDDR-CNN	5.41	**0.539**	0.705	0.024	0.194	0.206
MTAN	4.51	0.637	0.702	0.023	0.193	0.203
AdaShare	**1**	0.566	**0.707**	0.025	**0.192**	**0.193**



Dynamic Neural Networks for Video Classification

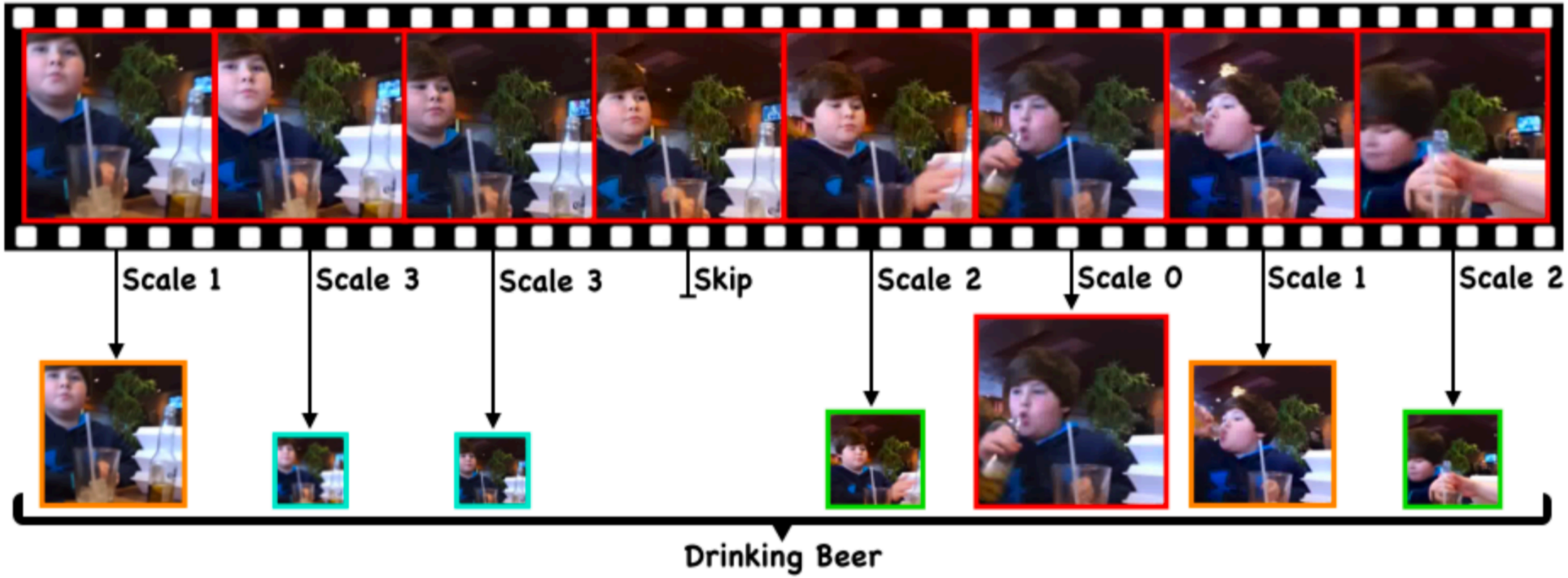
MIT: Bowen Pan, Camilo Fosco, Alex Andonian, Aude Oliva

BU & IBM: Ximeng Sun and Kate Saenko

IBM: Yue Meng, Rameswar Panda, Chung-Ching Lin, Richard Chen, Quanfu Fan, Prasanna Sattigeri, Leonid Karlinsky, Rogerio Feris

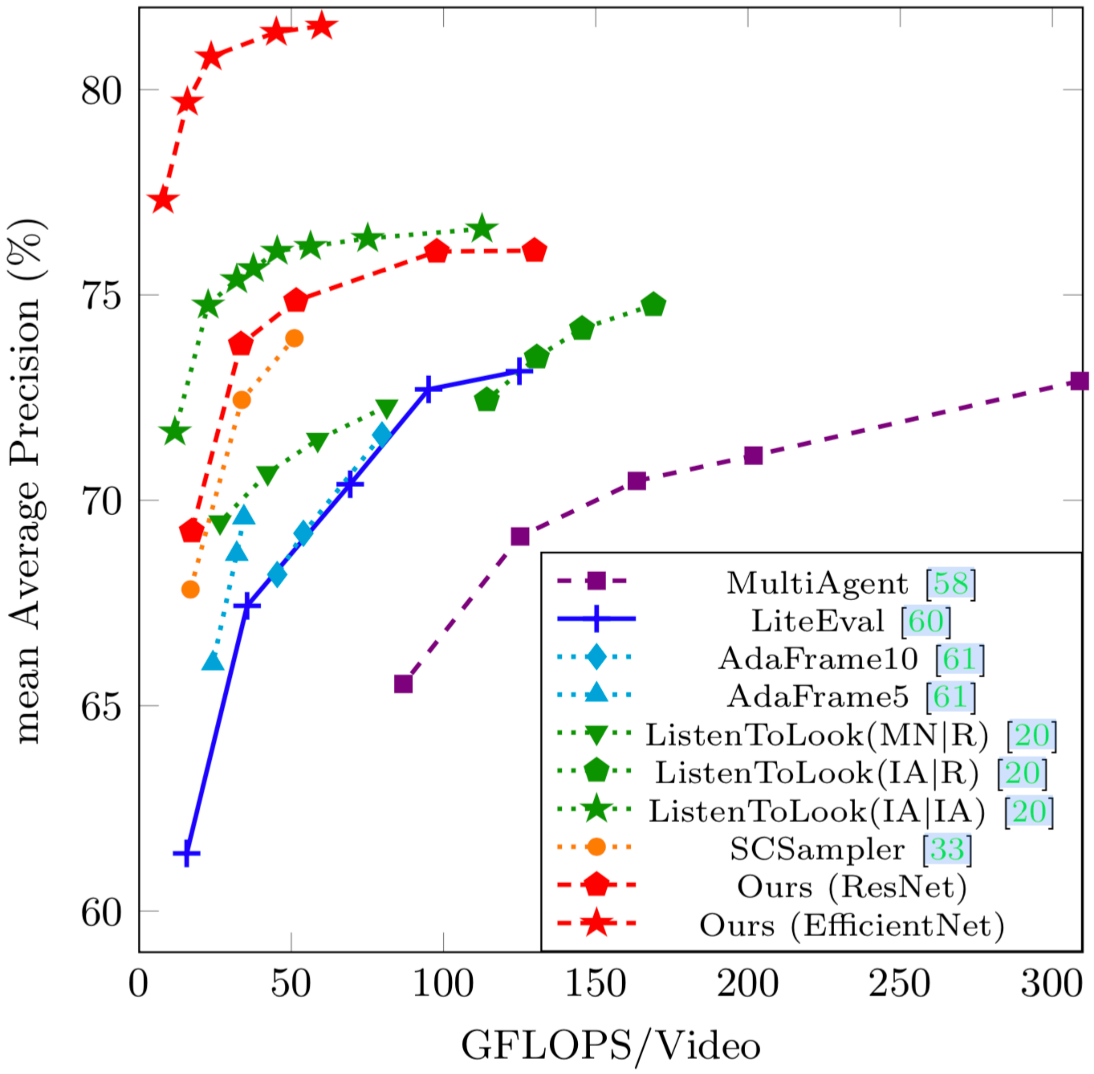
AR-Net: Adaptive frame resolution for efficient action recognition [ECCV 2020]

- Key idea is to select the resolution of each frame on-the-fly to achieve the best accuracy/efficiency trade-off in video classification



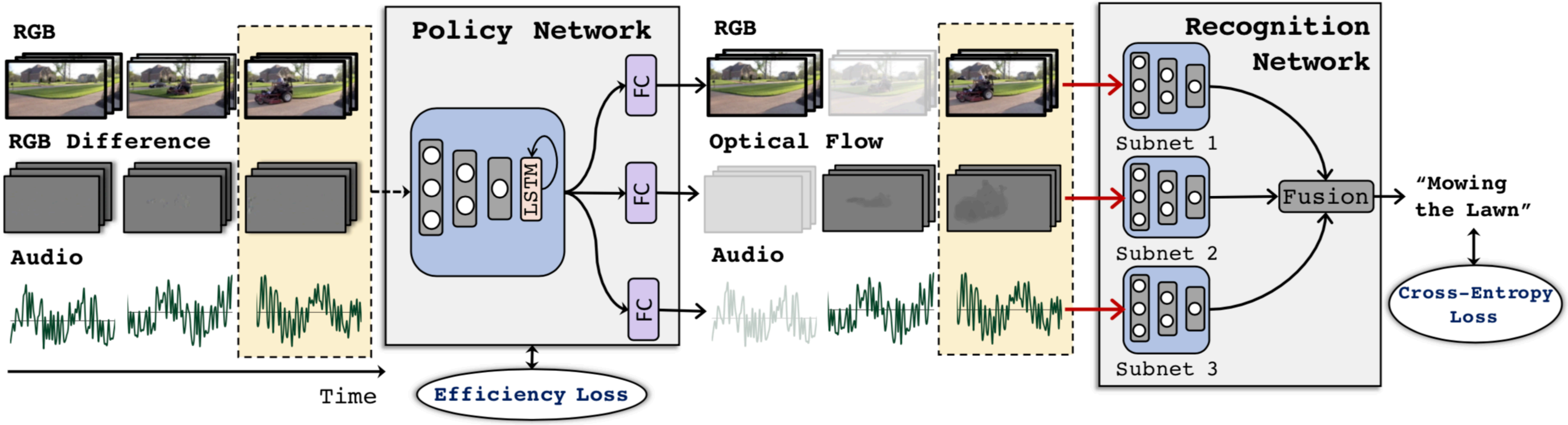
AR-Net: Experimental Results

ActivityNet dataset



AdaMML: Adaptive multimodal learning for efficient video recognition

- Key idea is to select on-the-fly the optimal modalities for each video segment conditioned on the input for efficient video recognition

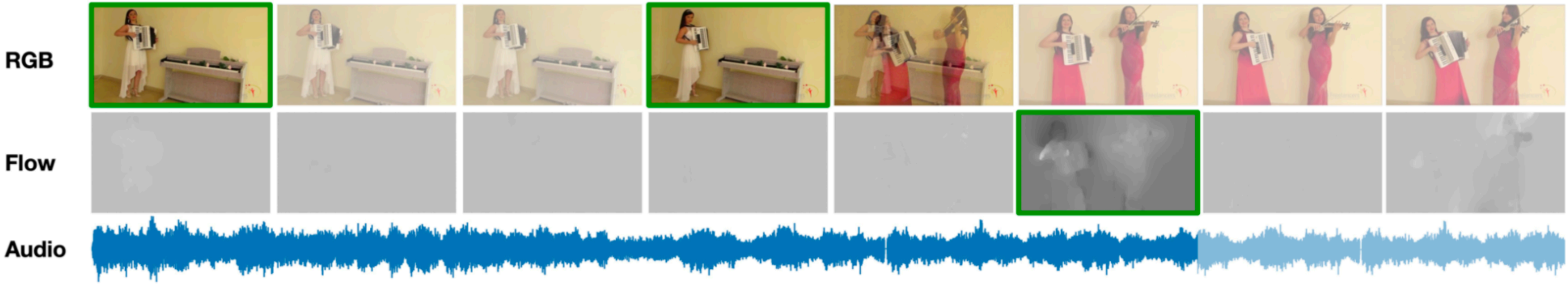


AdaMML: Experimental Results

RGB+Flow+Audio Performance on Kinetics-Sounds dataset

Method	Acc. (%)	Selection Rate (%) RGB	Flow	Audio	GFLOPs
RGB	82.85	100	–	–	141.36
Flow	75.73	–	100	–	163.39
Audio	65.49	–	–	100	3.82
Naïve	82.81	100	100	100	308.56
AdaMML-Flow	88.54	56.13	20.31	97.49	132.94
AdaMML-RGBDiff	**89.06**	55.06	26.82	95.12	141.97

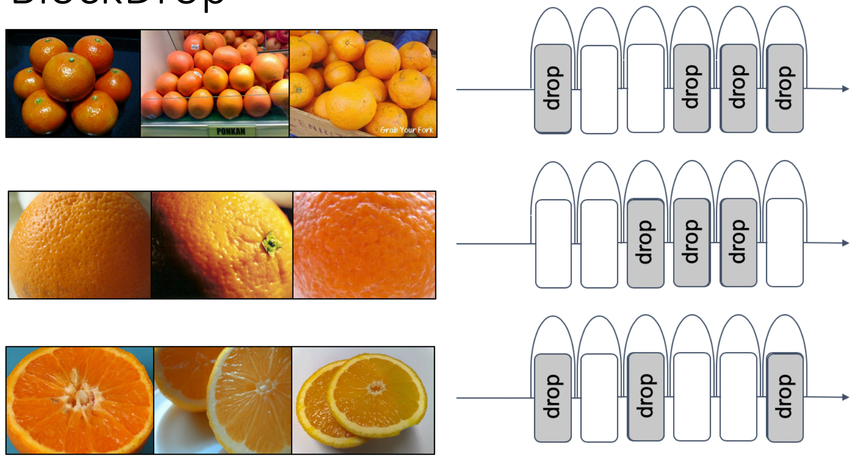
Action: Playing Accordion



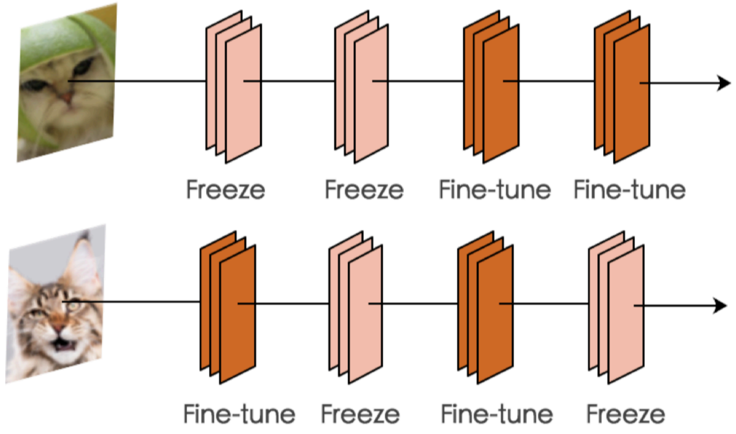
Summary

- Adaptive (dynamic) neural networks for efficient image and video classification

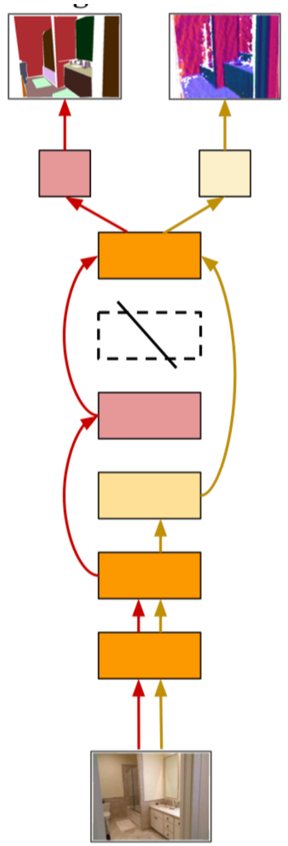
BlockDrop



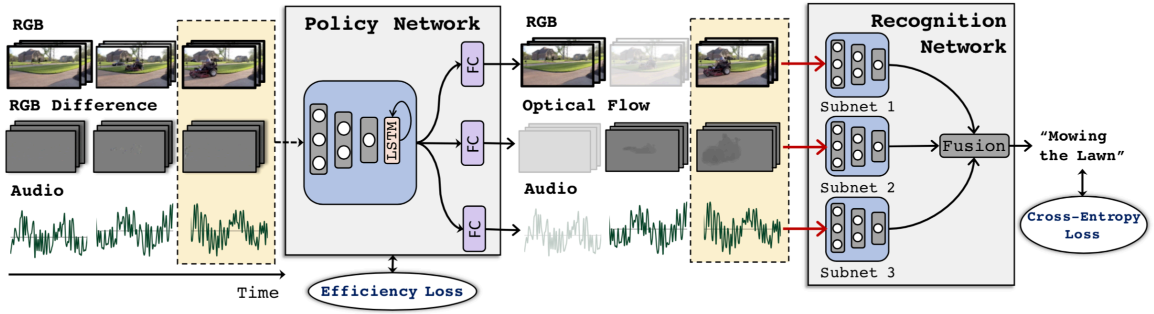
SpotTune



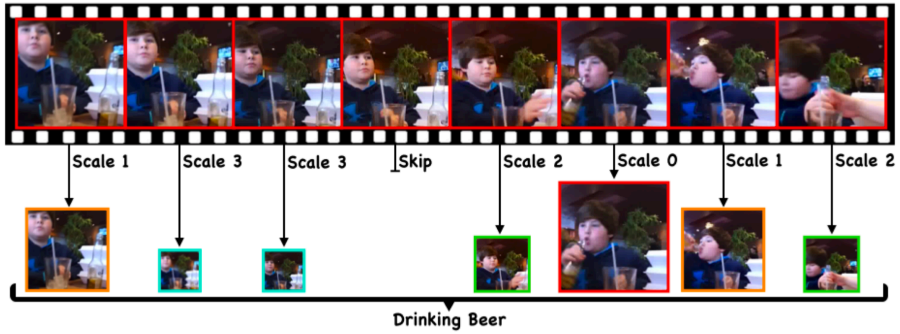
Adashare



AdaMML



AR-Net



References

- 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
- Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing and R. S. Feris. “SpotTune: Transfer Learning Through Adaptive Fine-Tuning” CVPR 2019
- Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, R. S. Feris. “Fully-adaptive Feature Sharing in Multi-Task Networks with Applications in Person Attribute Classification” CVPR 2017
- X. Sun, R. Panda and R. S. Feris. “AdaShare: Learning What to Share for Efficient Deep Multi-Task Learning” NeurIPS 2020
- Y. Meng, C. Lin, R. Panda, P. Sattigeri, L. Karlinsky, A. Oliva, K. Saenko, R. S. Feris. “AR-Net: Adaptive Frame Resolution for Efficient Action Recognition” ECCV 2020
- C. Chen*, R. Panda*, Q. Fan, X. Sun, K. Saenko, A. Oliva, R. S. Feris. “AdaMML: Adaptive Multi-Modal Learning for Efficient Video Recognition” (under submission)

(* equal contribution)