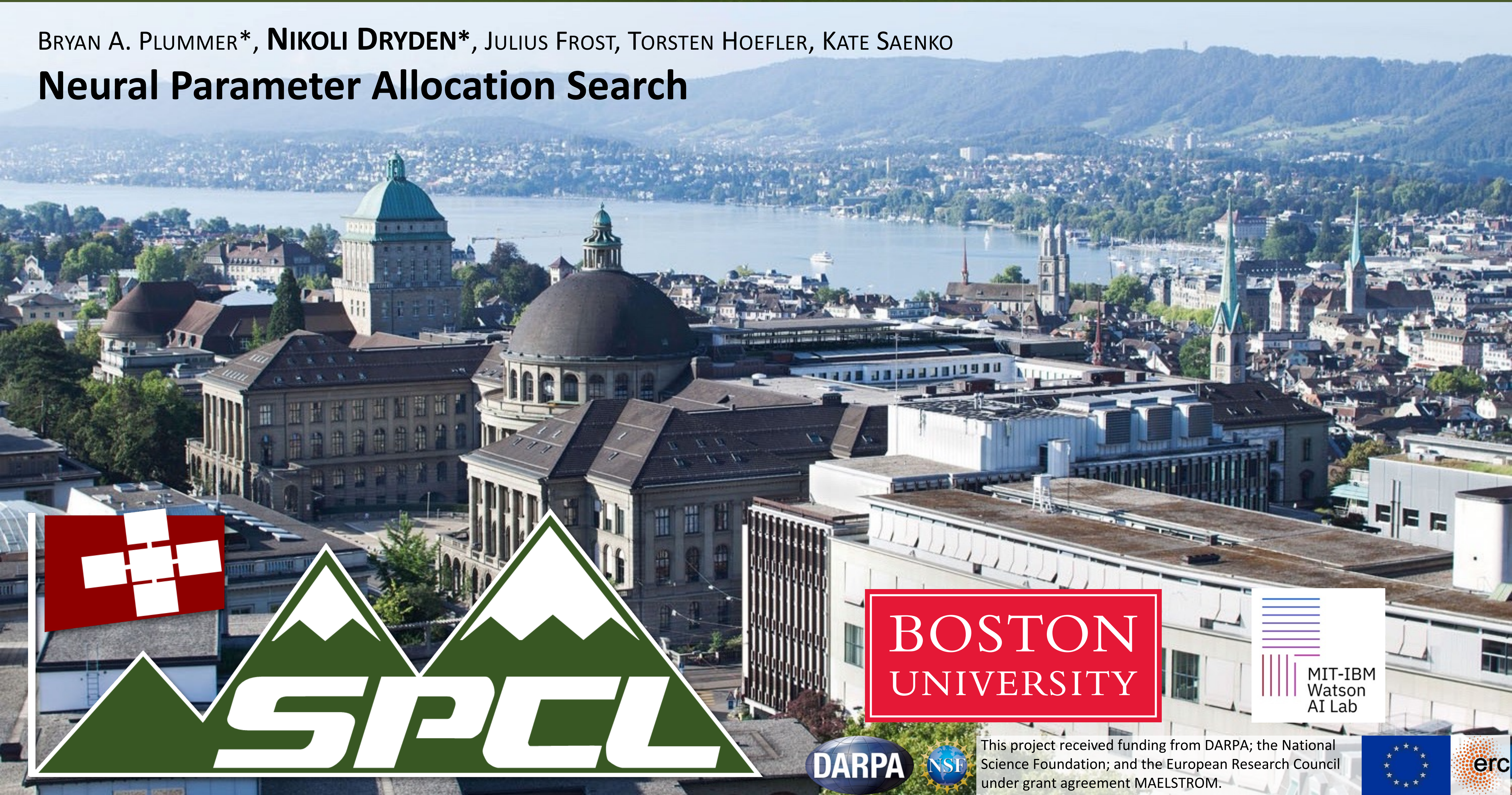


Bryan A. Plummer*, **Nikoli Dryden***, Julius Frost, Torsten Hoefler, Kate Saenko

Neural Parameter Allocation Search

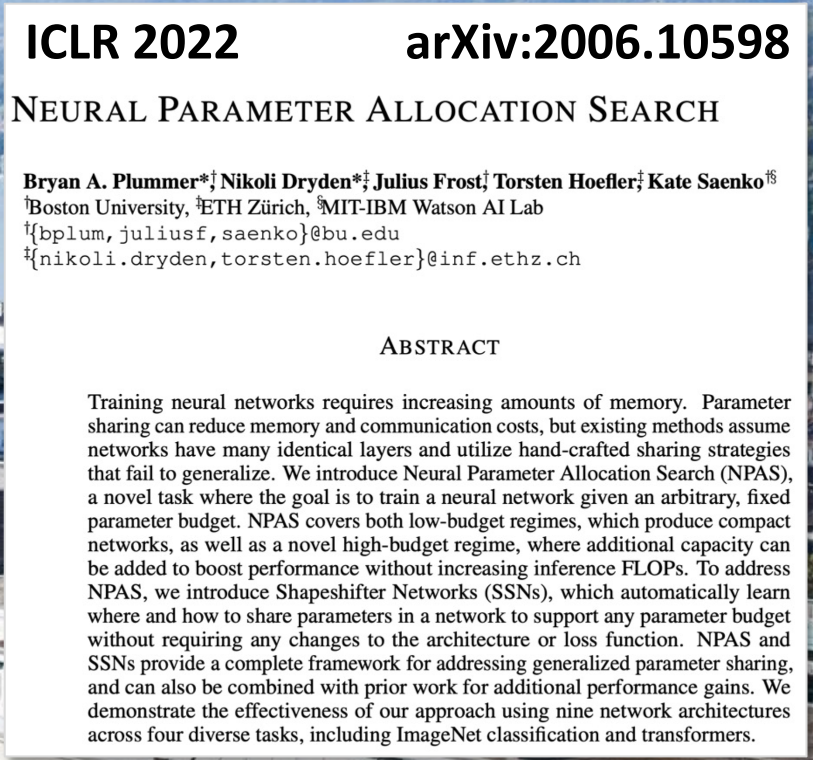
BOSTON UNIVERSITY

MIT-IBM Watson AI Lab

This project received funding from DARPA; the National Science Foundation; and the European Research Council under grant agreement MAELSTROM.

Bryan A. Plummer*, **Nikoli Dryden***, Julius Frost, Torsten Hoefler, Kate Saenko

Neural Parameter Allocation Search



ICLR 2022 **arXiv:2006.10598**

Neural Parameter Allocation Search

Bryan A. Plummer*[†], Nikoli Dryden*[‡], Julius Frost[†], Torsten Hoefler[‡], Kate Saenko[†§]
[†]Boston University, [‡]ETH Zürich, [§]MIT-IBM Watson AI Lab
[†]{bplum,juliusf,saenko}@bu.edu
[‡]{nikoli.dryden,torsten.hoefler}@inf.ethz.ch

Abstract

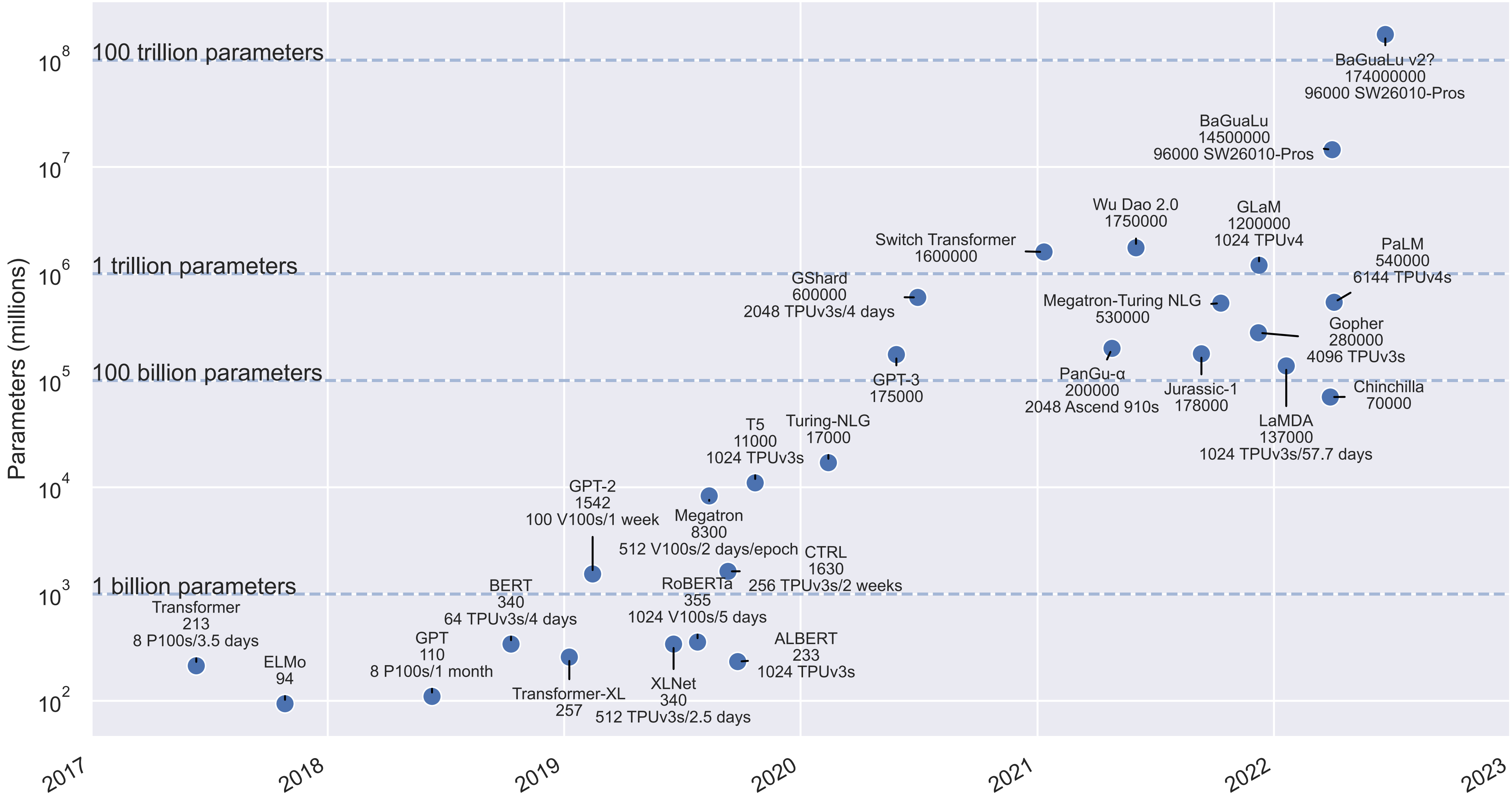
Training neural networks requires increasing amounts of memory. Parameter sharing can reduce memory and communication costs, but existing methods assume networks have many identical layers and utilize hand-crafted sharing strategies that fail to generalize. We introduce Neural Parameter Allocation Search (NPAS), a novel task where the goal is to train a neural network given an arbitrary, fixed parameter budget. NPAS covers both low-budget regimes, which produce compact networks, as well as a novel high-budget regime, where additional capacity can be added to boost performance without increasing inference FLOPs. To address NPAS, we introduce Shapeshifter Networks (SSNs), which automatically learn where and how to share parameters in a network to support any parameter budget without requiring any changes to the architecture or loss function. NPAS and SSNs provide a complete framework for addressing generalized parameter sharing, and can also be combined with prior work for additional performance gains. We demonstrate the effectiveness of our approach using nine network architectures across four diverse tasks, including ImageNet classification and transformers.

BOSTON UNIVERSITY

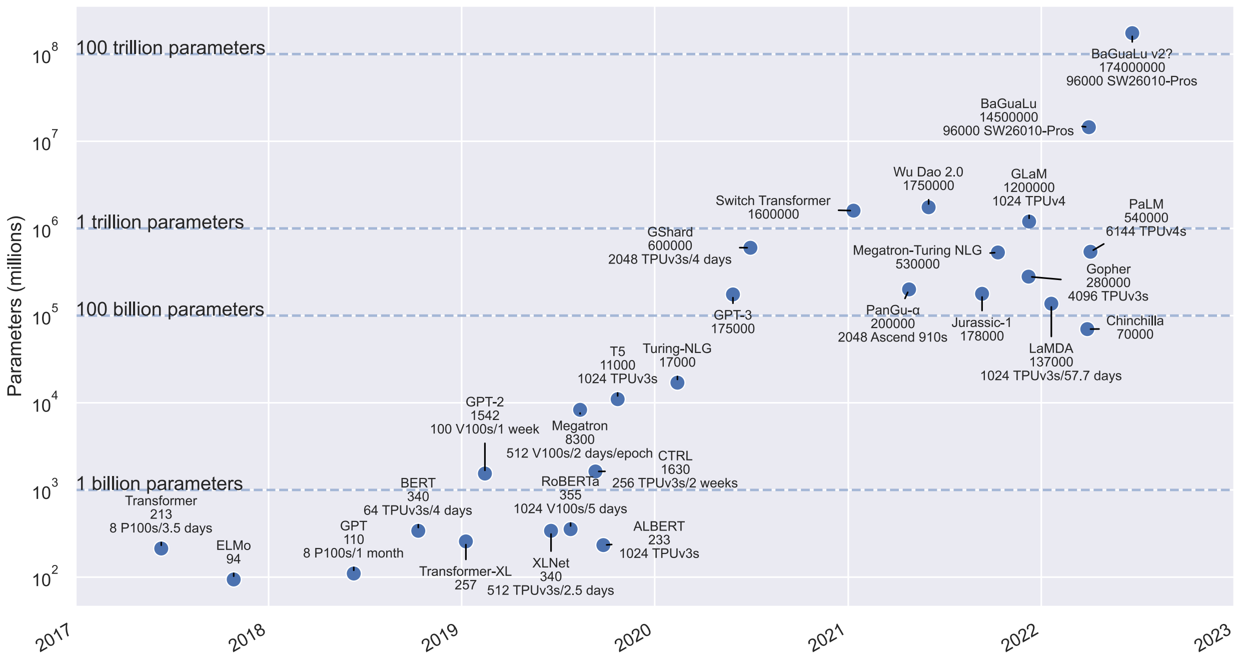
MIT-IBM Watson AI Lab

This project received funding from DARPA; the National Science Foundation; and the European Research Council under grant agreement MAELSTROM.

The Memory Explosion

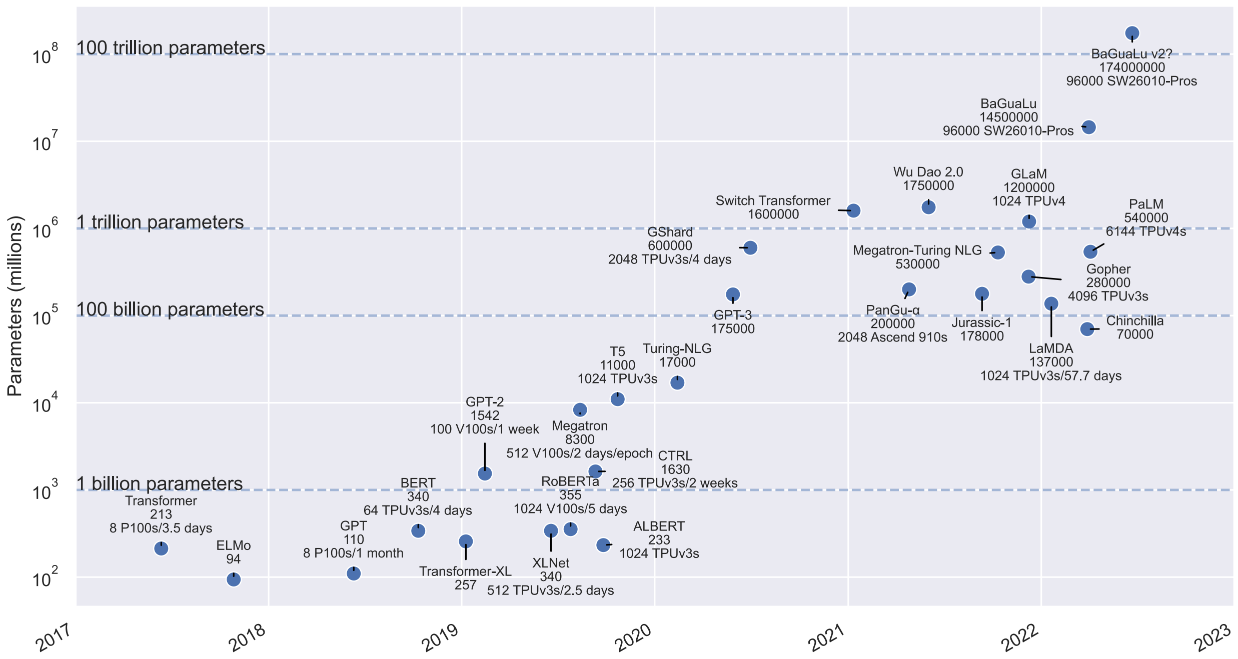


The Memory Explosion



Memory Usage

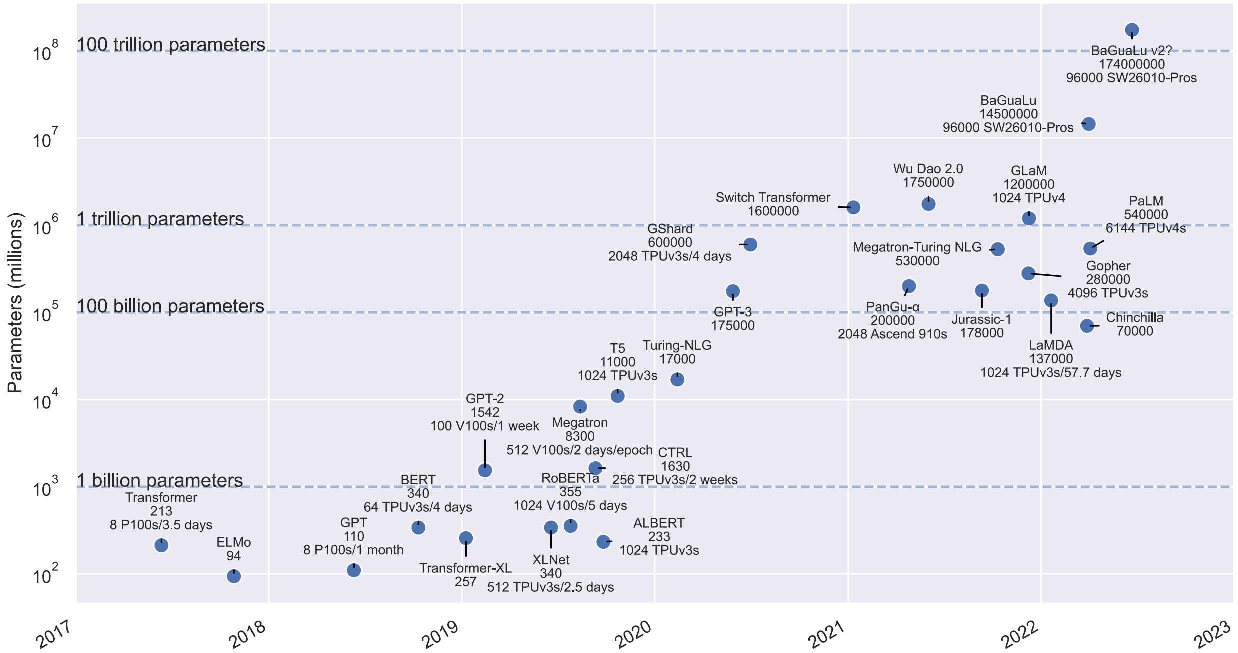
The Memory Explosion



Memory Usage

Parameters

The Memory Explosion

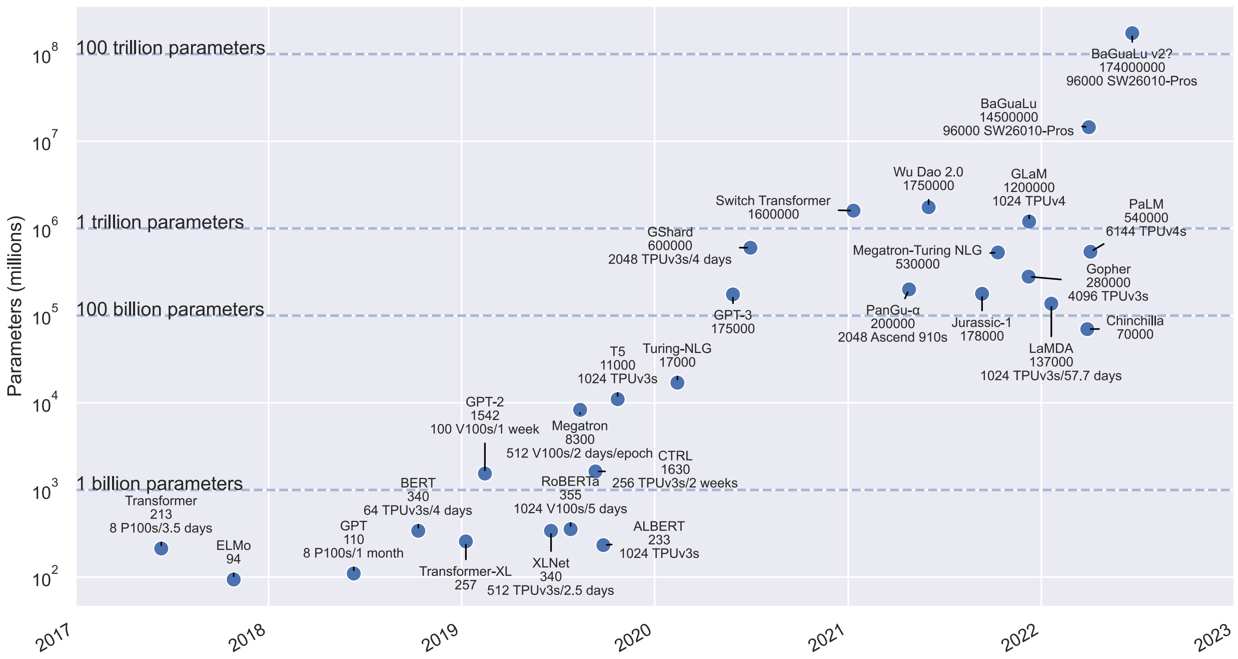


Memory Usage

Parameters

Activations

The Memory Explosion



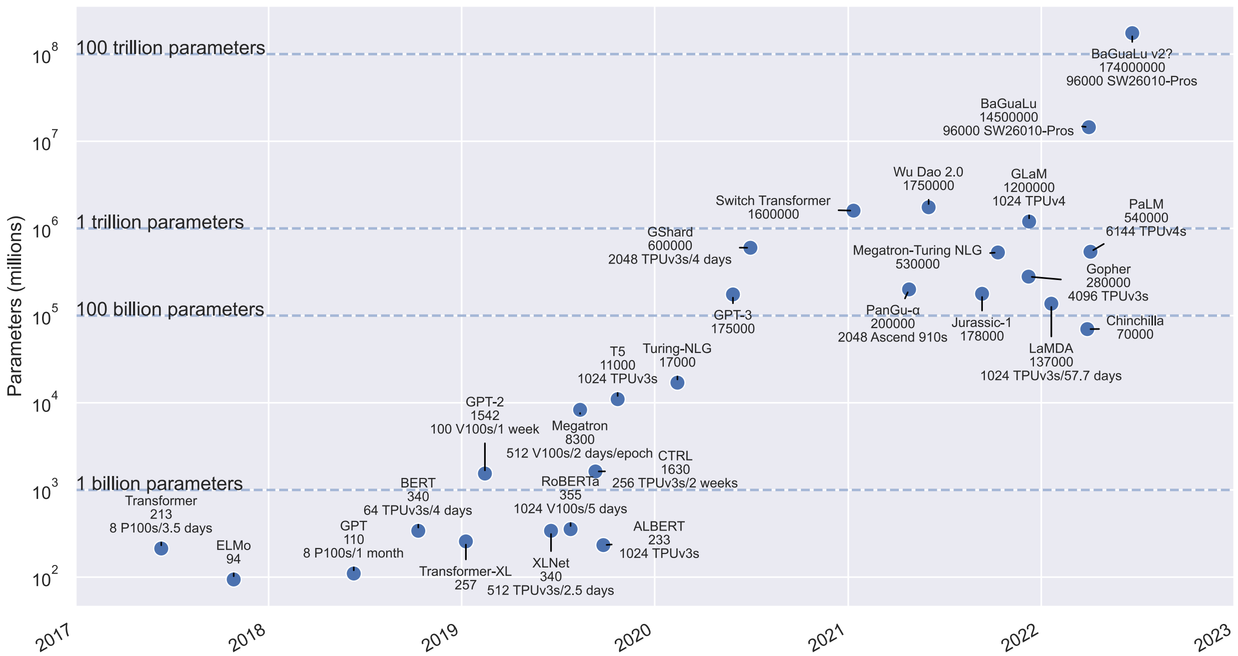
Memory Usage

Parameters

Activations

Gradients

The Memory Explosion



Memory Usage

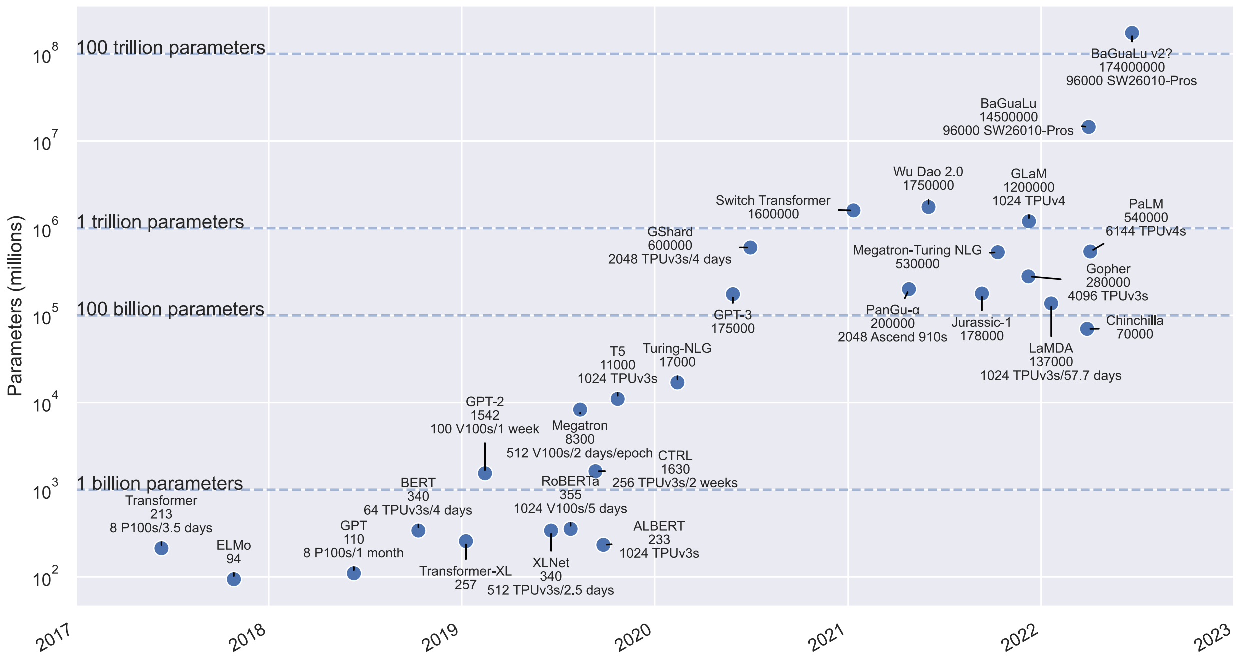
Parameters

Activations

Gradients

Optimizer state

The Memory Explosion



Memory Usage
100 trillion parameters, FP32, Adam

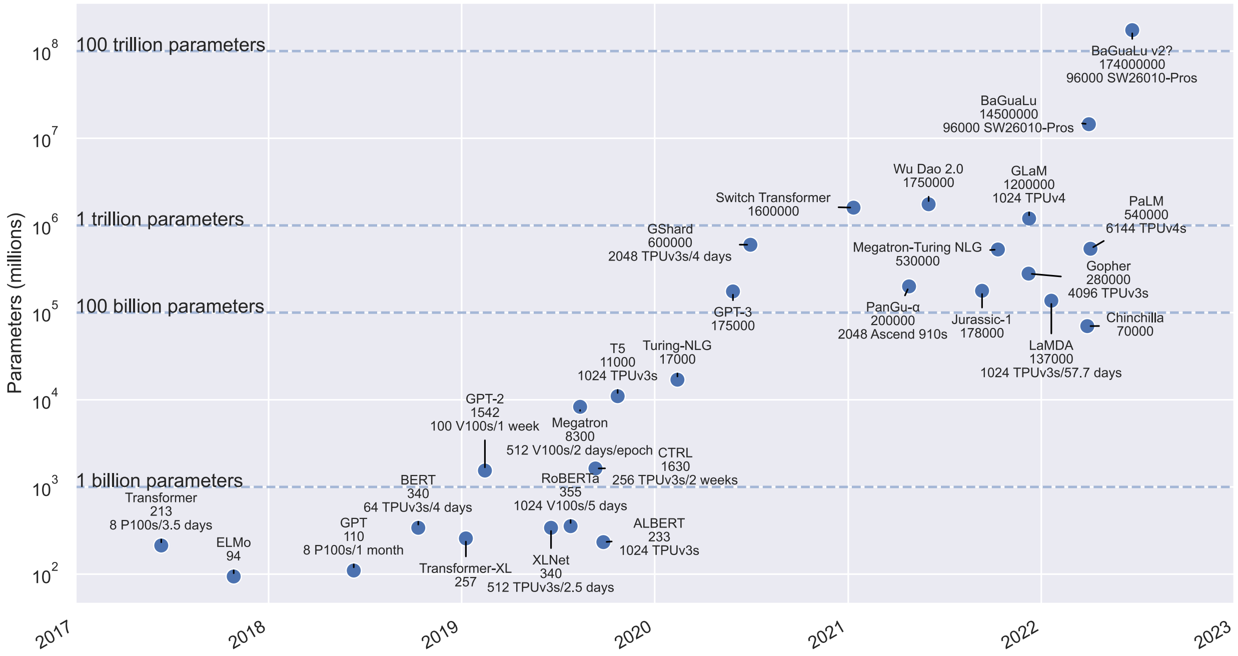
Parameters

Activations

Gradients

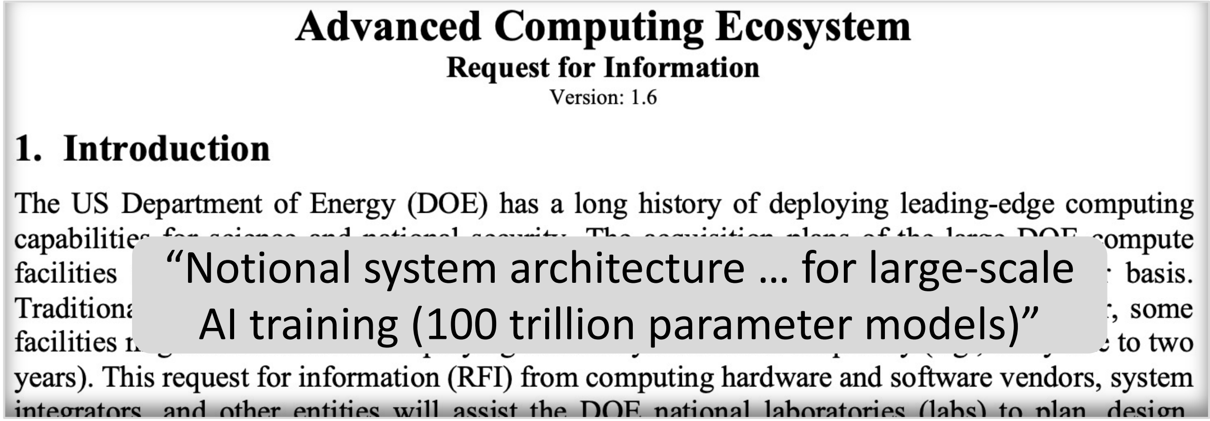
Optimizer state

The Memory Explosion



Memory Usage
100 trillion parameters, FP32, Adam

Parameters | **Activations** | **Gradients** | **Optimizer state**



Advanced Computing Ecosystem

Request for Information

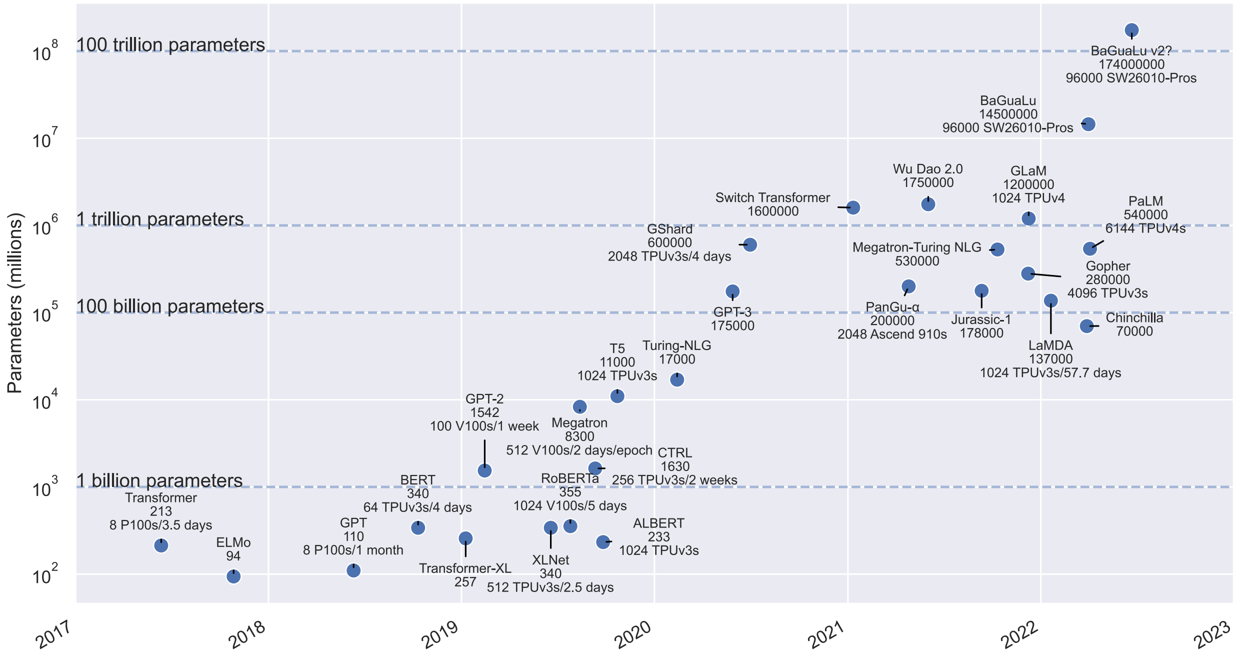
Version: 1.6

1. Introduction

The US Department of Energy (DOE) has a long history of deploying leading-edge computing capabilities for science and national security. The acquisition plans of the large DOE compute facilities … basis. Traditiona… , some facilities … to two years). This request for information (RFI) from computing hardware and software vendors, system integrators, and other entities will assist the DOE national laboratories (labs) to plan, design

“Notional system architecture … for large-scale AI training (100 trillion parameter models)”

The Memory Explosion



Memory Usage
100 trillion parameters, FP32, Adam

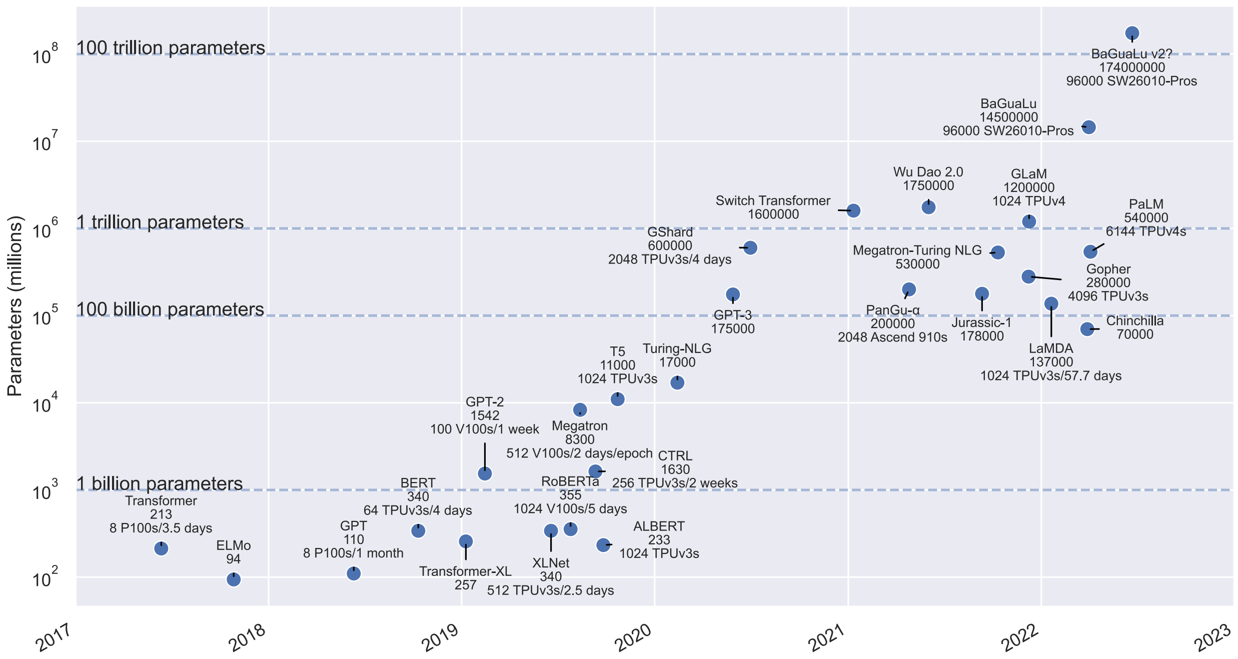
Parameters

Activations

Gradients

Optimizer state

The Memory Explosion

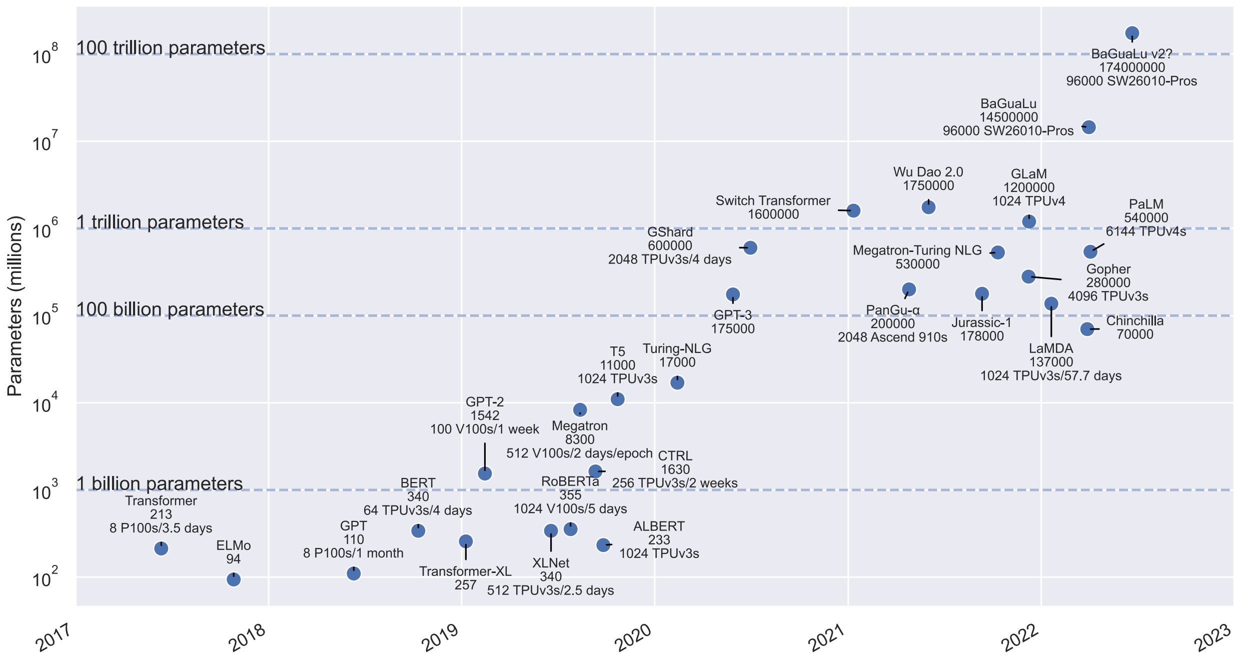


Memory Usage
100 trillion parameters, FP32, Adam

Parameters | **Activations** | **Gradients** | **Optimizer state**

400 TB

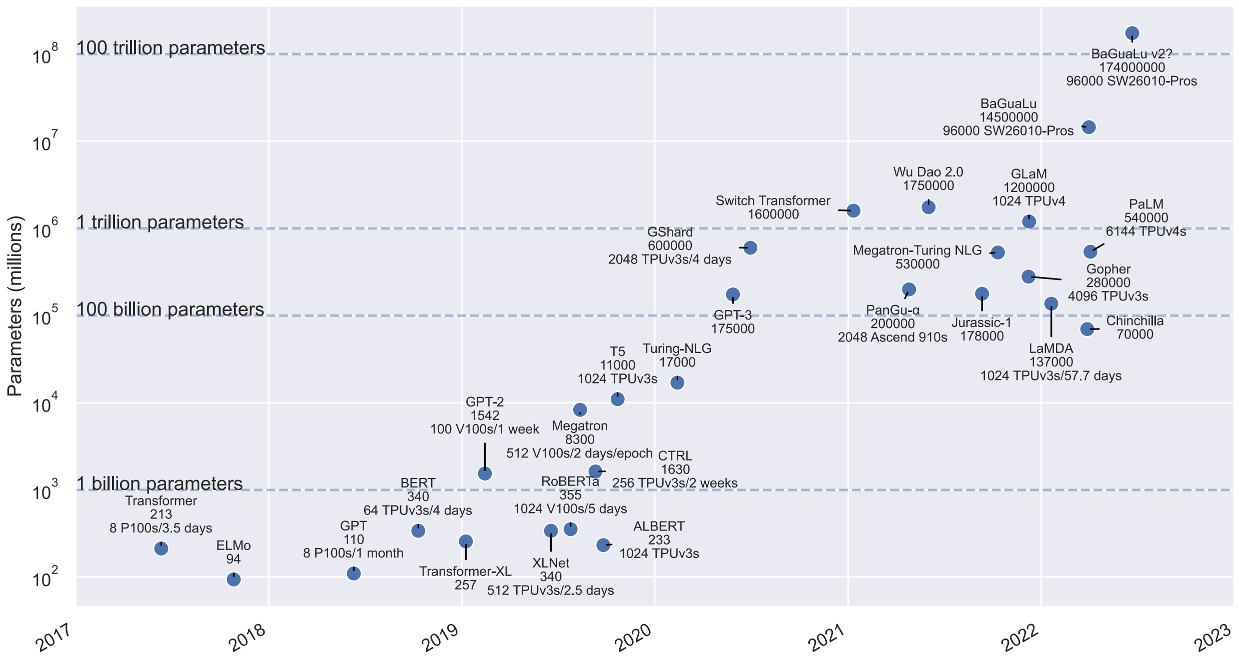
The Memory Explosion



Memory Usage
100 trillion parameters, FP32, Adam

Parameters	Activations	Gradients	Optimizer state
400 TB	(n/a)		

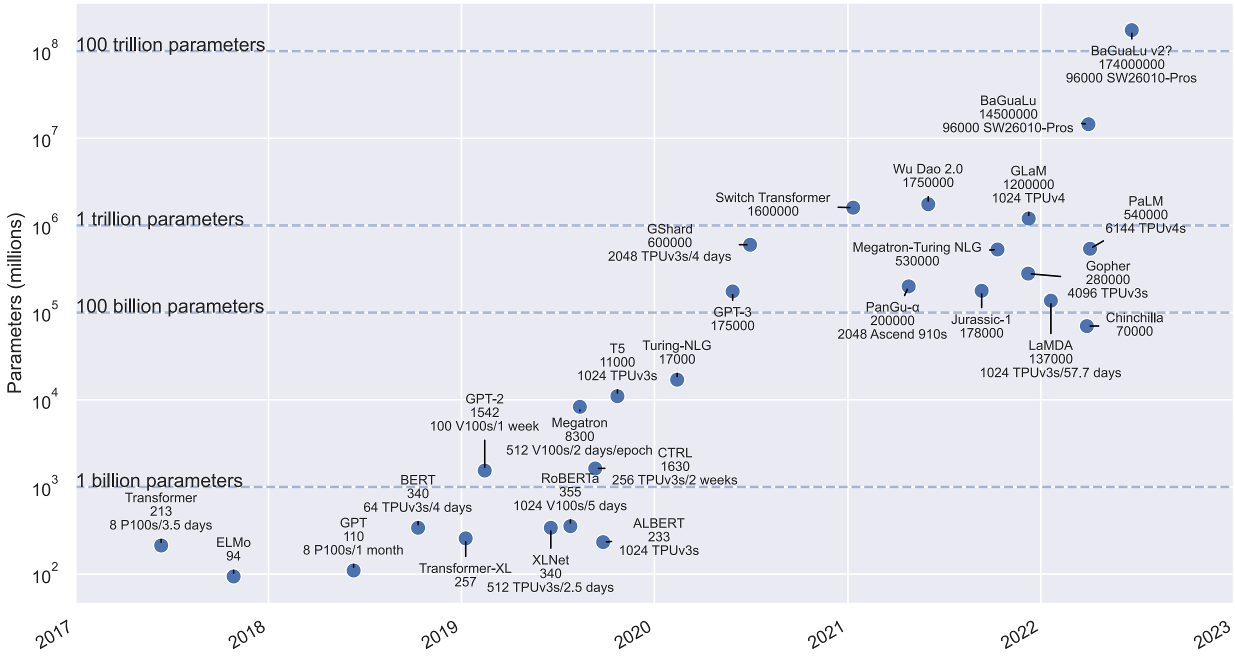
The Memory Explosion



Memory Usage
100 trillion parameters, FP32, Adam

Parameters	Activations	Gradients	Optimizer state
400 TB	(n/a)	400 TB	

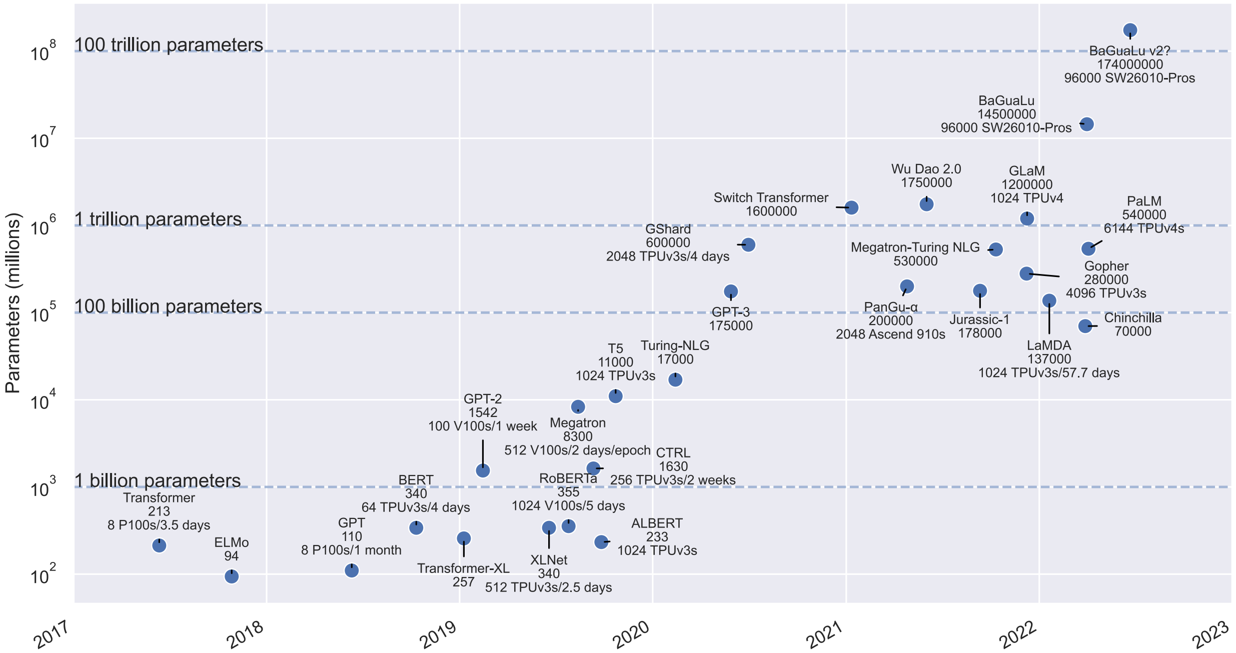
The Memory Explosion



Memory Usage
100 trillion parameters, FP32, Adam

Parameters	Activations	Gradients	Optimizer state
400 TB	(n/a)	400 TB	800 TB

The Memory Explosion

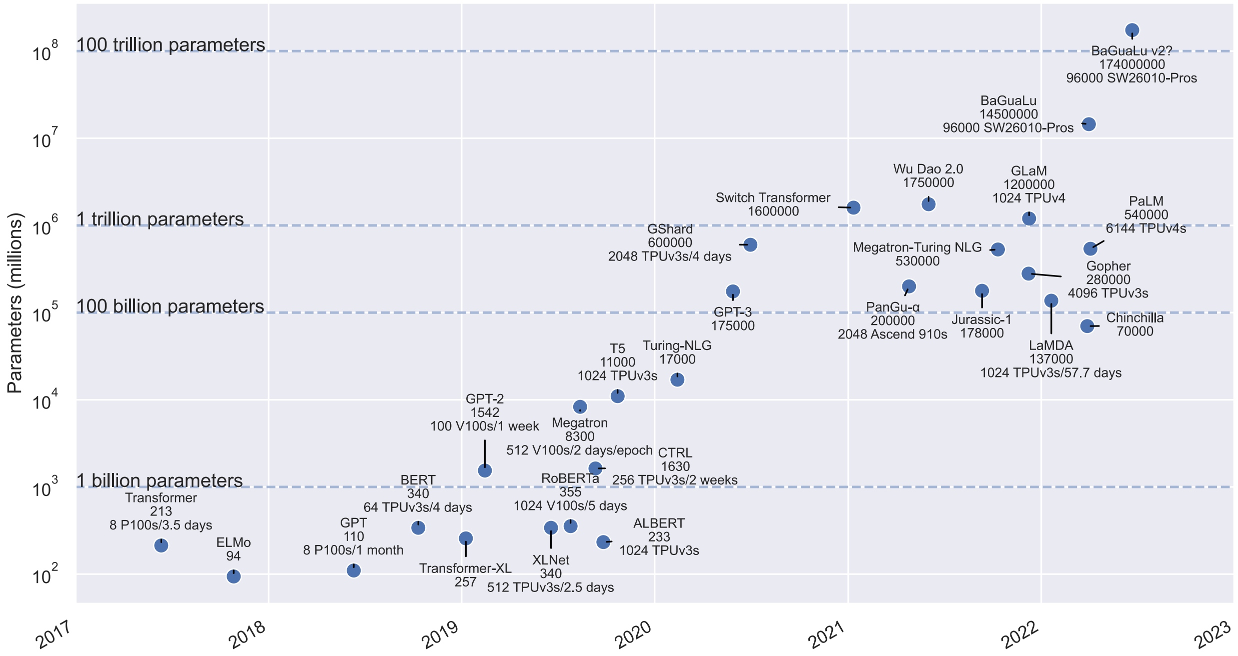


Memory Usage
100 trillion parameters, FP32, Adam

Parameters	Activations	Gradients	Optimizer state
400 TB	(n/a)	400 TB	800 TB

= 1.6 PB

The Memory Explosion

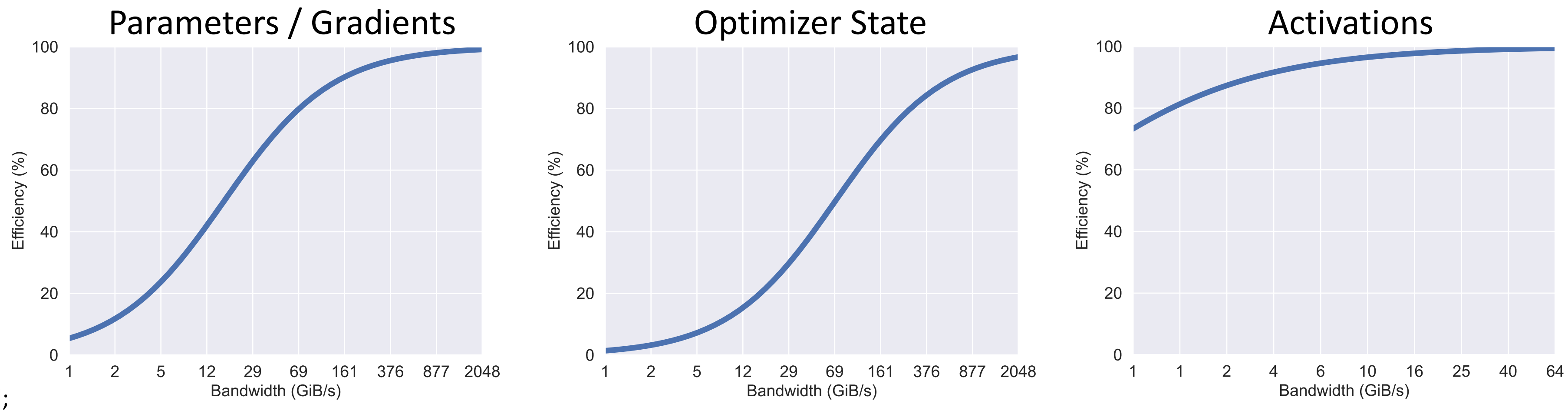


Memory Usage
100 trillion parameters, FP32, Adam

Parameters	Activations	Gradients	Optimizer state
400 TB	(n/a)	400 TB	800 TB

= 1.6 PB

Memory Bandwidth:


Parameters / Gradients

Optimizer State

Activations

(Adapted from ZeRO-Infinity [Rajbhandari et al., 2021]; batch size 4, seqlen 1024, hidden dim 8K, 70 Tflop peak)

How to Break the Memory Wall

How to Break the Memory Wall

Systems

How to Break the Memory Wall

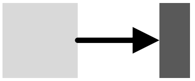
Systems

Recomputation

How to Break the Memory Wall

Systems

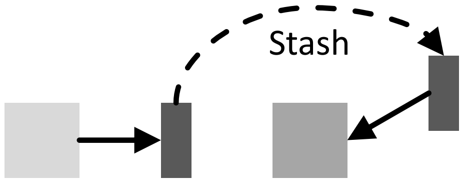
Recomputation



How to Break the Memory Wall

Systems

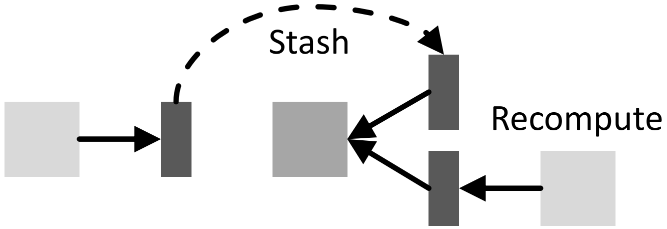
Recomputation



How to Break the Memory Wall

Systems

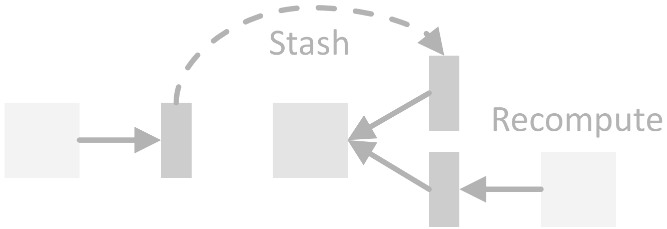
Recomputation



How to Break the Memory Wall

Systems

Recomputation

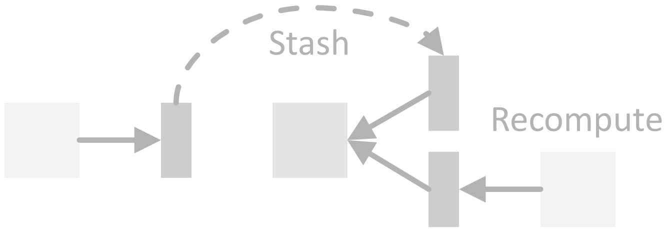


Out-of-core

How to Break the Memory Wall

Systems

Recomputation

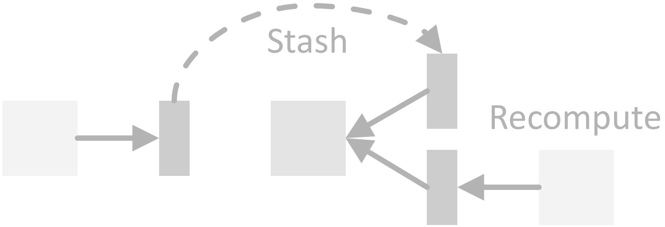


Out-of-core

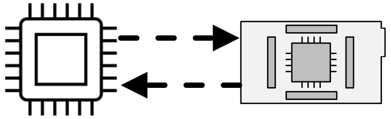
How to Break the Memory Wall

Systems

Recomputation



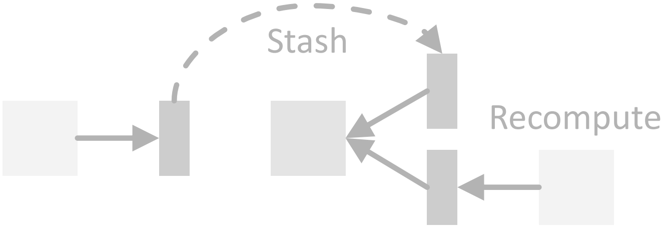
Out-of-core



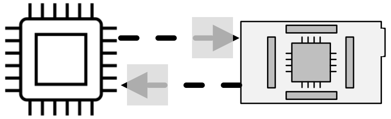
How to Break the Memory Wall

Systems

Recomputation



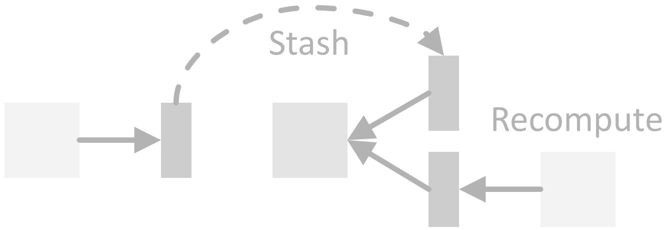
Out-of-core



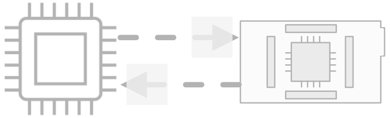
How to Break the Memory Wall

Systems

Recomputation



Out-of-core

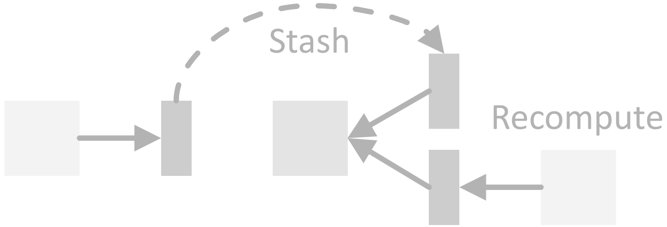


Model parallelism

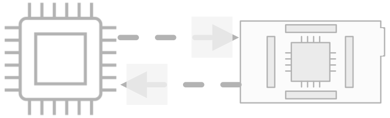
How to Break the Memory Wall

Systems

Recomputation



Out-of-core

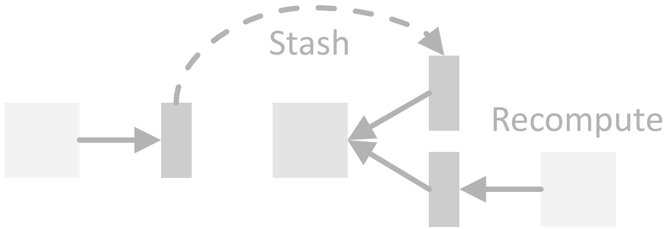


Model parallelism

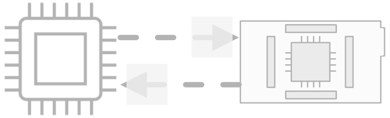
How to Break the Memory Wall

Systems

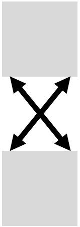
Recomputation



Out-of-core



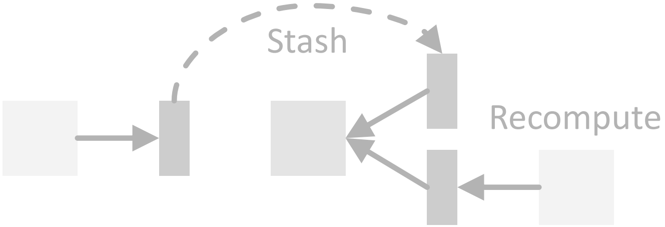
Model parallelism



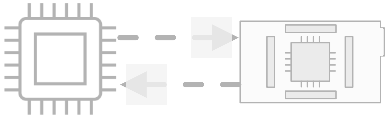
How to Break the Memory Wall

Systems

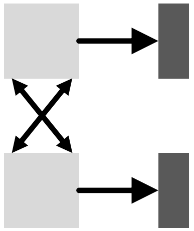
Recomputation



Out-of-core



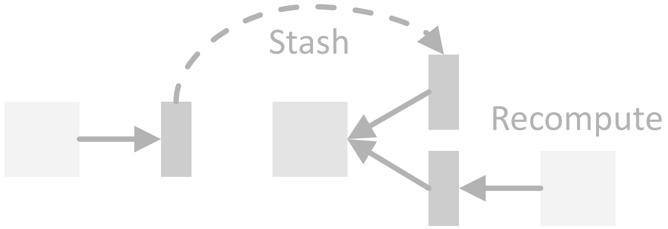
Model parallelism



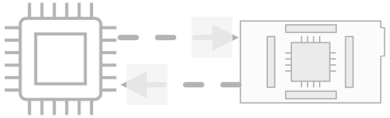
How to Break the Memory Wall

Systems

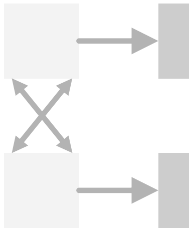
Recomputation



Out-of-core



Model parallelism

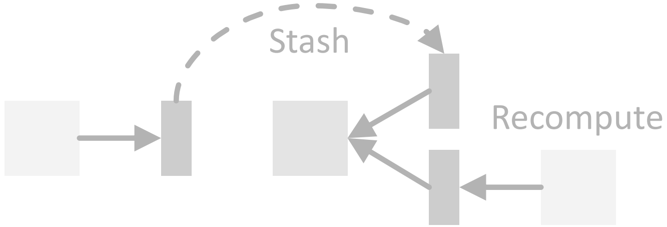


Models

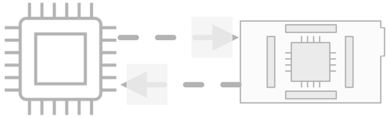
How to Break the Memory Wall

Systems

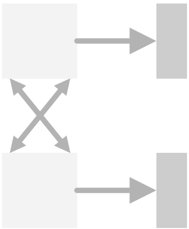
Recomputation



Out-of-core



Model parallelism



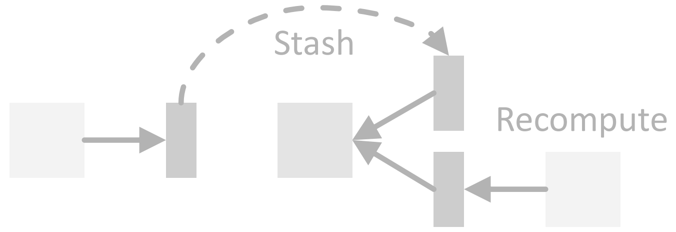
Models

Quantization / Pruning / Low-rank / Distillation

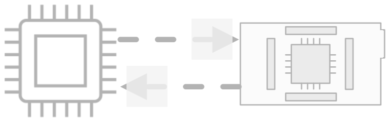
How to Break the Memory Wall

Systems

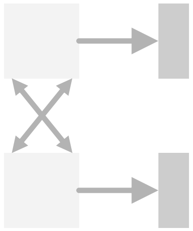
Recomputation



Out-of-core

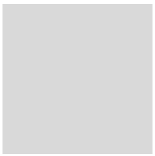


Model parallelism



Models

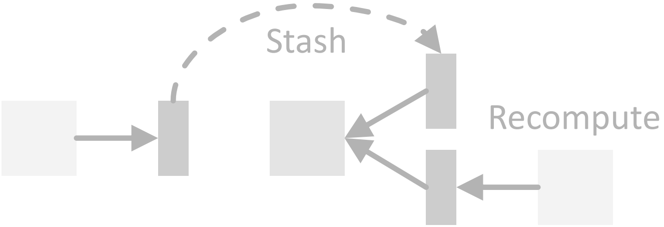
Quantization / Pruning / Low-rank / Distillation



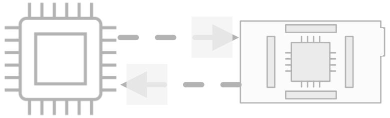
How to Break the Memory Wall

Systems

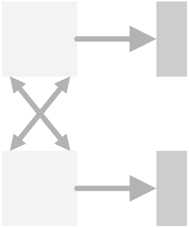
Recomputation



Out-of-core

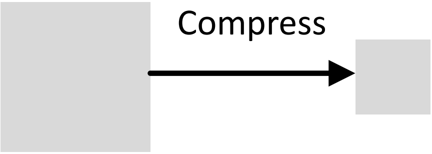


Model parallelism



Models

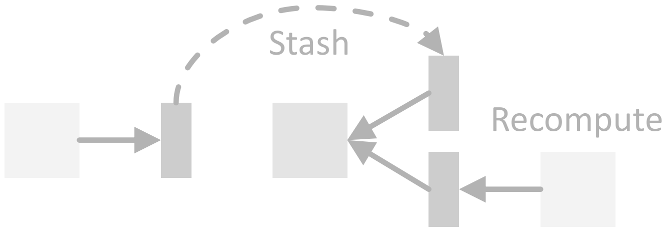
Quantization / Pruning / Low-rank / Distillation



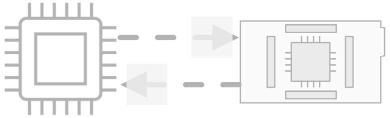
How to Break the Memory Wall

Systems

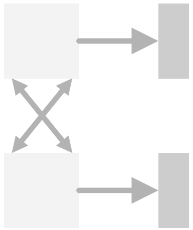
Recomputation



Out-of-core

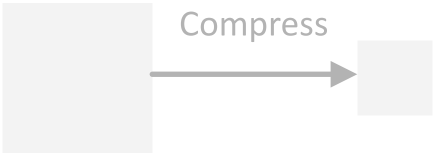


Model parallelism



Models

Quantization / Pruning / Low-rank / Distillation

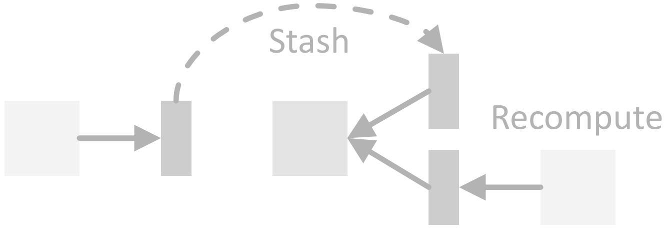


Parameter sharing

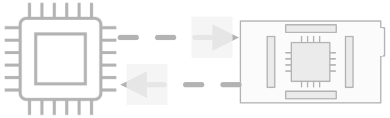
How to Break the Memory Wall

Systems

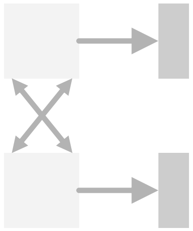
Recomputation



Out-of-core

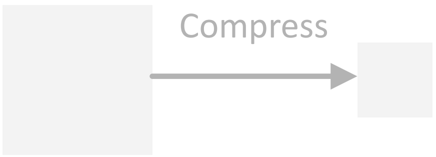


Model parallelism

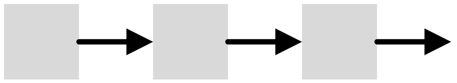


Models

Quantization / Pruning / Low-rank / Distillation



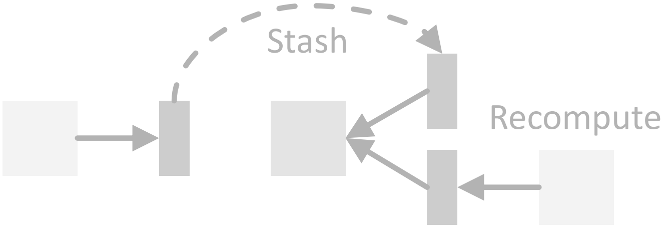
Parameter sharing



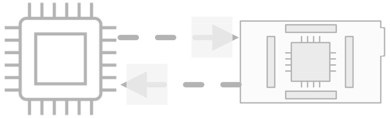
How to Break the Memory Wall

Systems

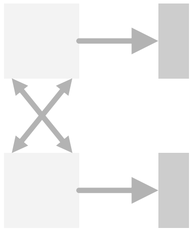
Recomputation



Out-of-core

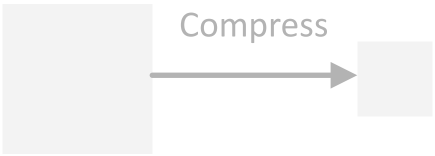


Model parallelism

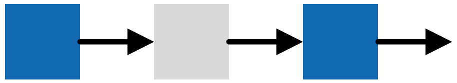


Models

Quantization / Pruning / Low-rank / Distillation



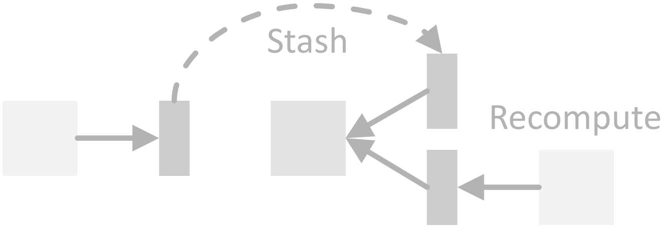
Parameter sharing



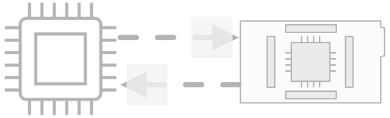
How to Break the Memory Wall

Systems

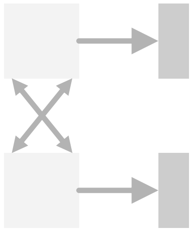
Recomputation



Out-of-core

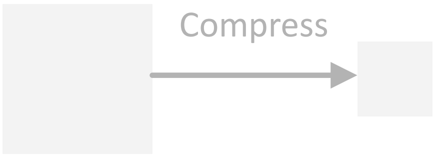


Model parallelism

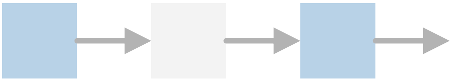


Models

Quantization / Pruning / Low-rank / Distillation



Parameter sharing

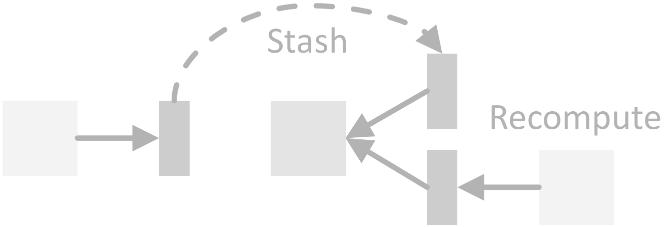


All of the above

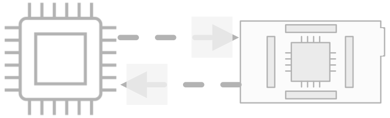
How to Break the Memory Wall

Systems

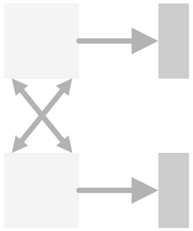
Recomputation



Out-of-core

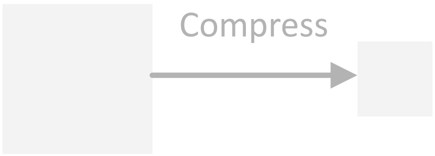


Model parallelism

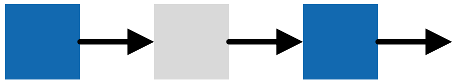


Models

Quantization / Pruning / Low-rank / Distillation



Parameter sharing

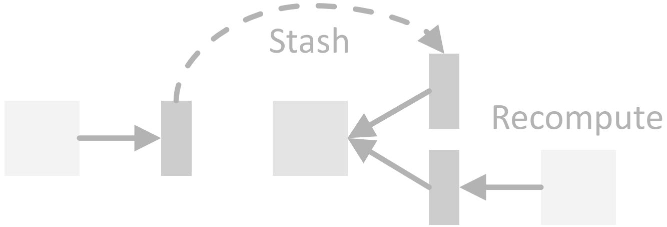


All of the above

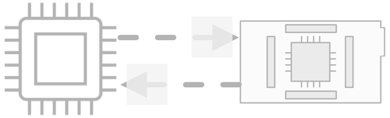
How to Break the Memory Wall

Systems

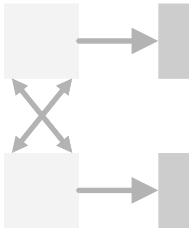
Recomputation



Out-of-core

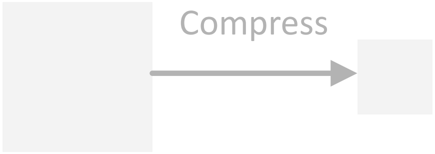


Model parallelism

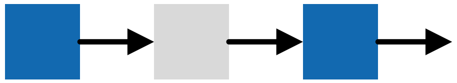


Models

Quantization / Pruning / Low-rank / Distillation



Parameter sharing



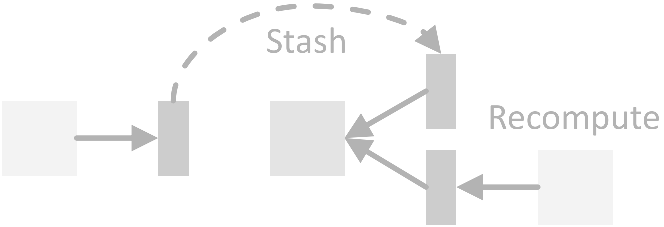
All of the above

Limitations of Standard Parameter Sharing

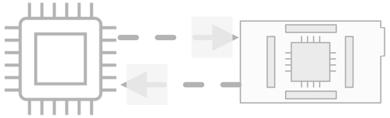
How to Break the Memory Wall

Systems

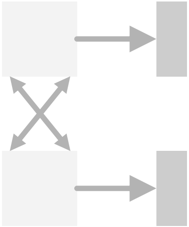
Recomputation



Out-of-core

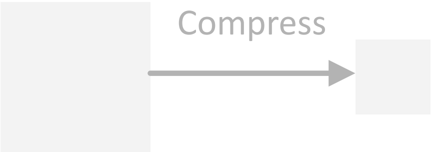


Model parallelism

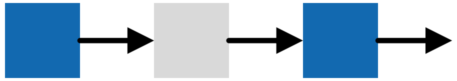


Models

Quantization / Pruning / Low-rank / Distillation



Parameter sharing



All of the above

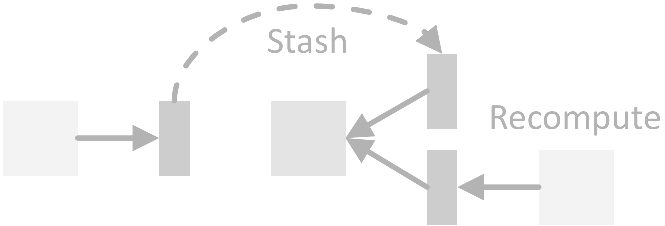
Limitations of Standard Parameter Sharing

Only share between identical layers

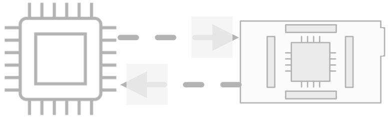
How to Break the Memory Wall

Systems

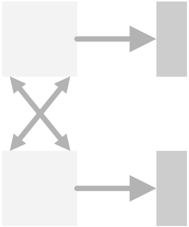
Recomputation



Out-of-core

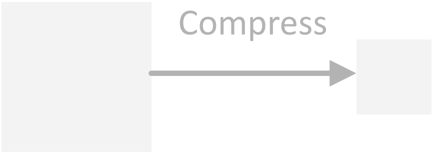


Model parallelism

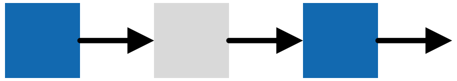


Models

Quantization / Pruning / Low-rank / Distillation



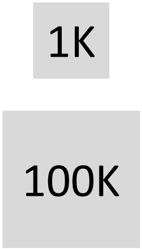
Parameter sharing



All of the above

Limitations of Standard Parameter Sharing

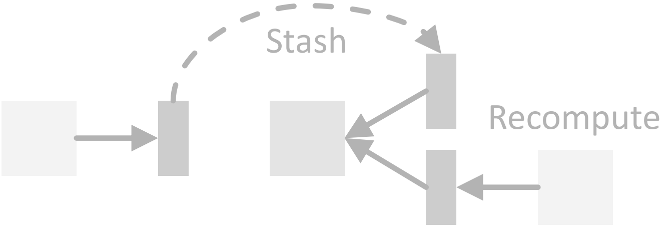
Only share between identical layers



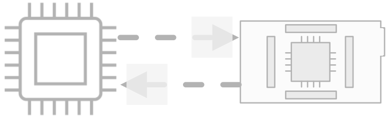
How to Break the Memory Wall

Systems

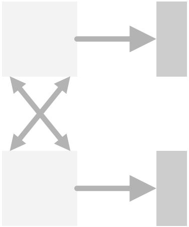
Recomputation



Out-of-core

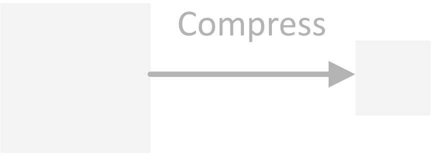


Model parallelism

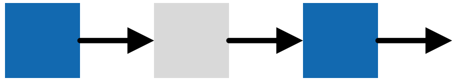


Models

Quantization / Pruning / Low-rank / Distillation



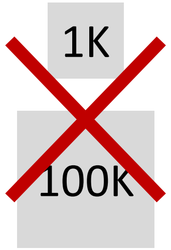
Parameter sharing



All of the above

Limitations of Standard Parameter Sharing

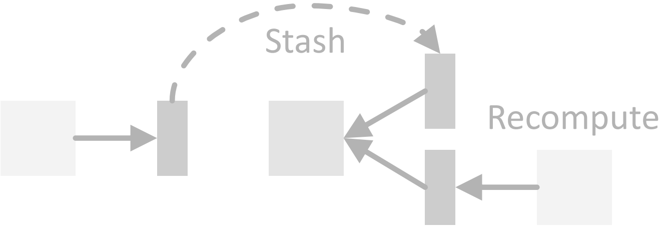
Only share between identical layers



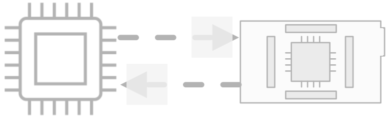
How to Break the Memory Wall

Systems

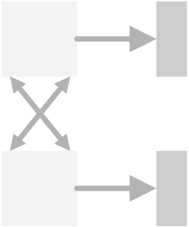
Recomputation



Out-of-core

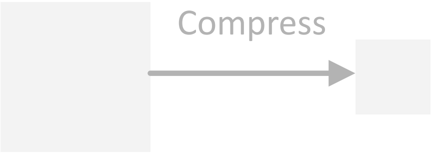


Model parallelism

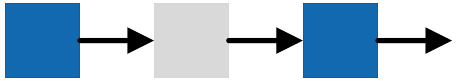


Models

Quantization / Pruning / Low-rank / Distillation



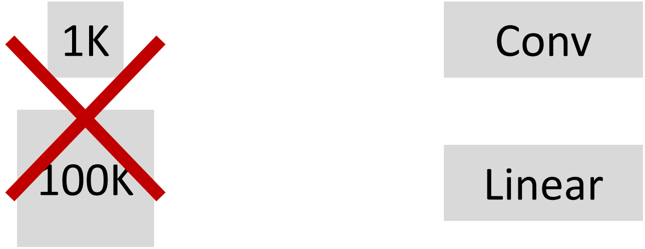
Parameter sharing



All of the above

Limitations of Standard Parameter Sharing

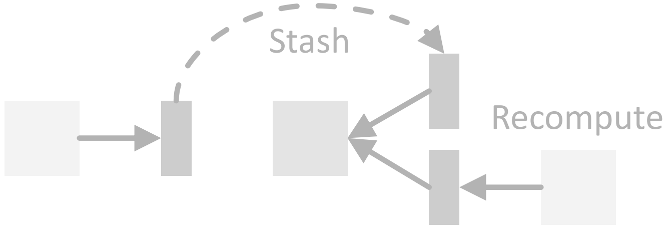
Only share between identical layers



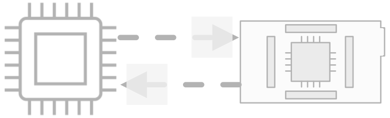
How to Break the Memory Wall

Systems

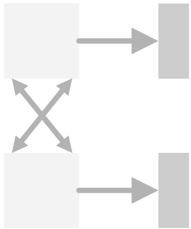
Recomputation



Out-of-core

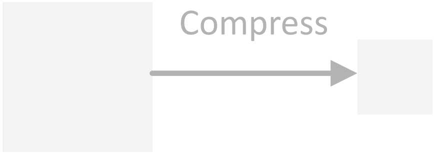


Model parallelism

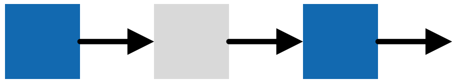


Models

Quantization / Pruning / Low-rank / Distillation



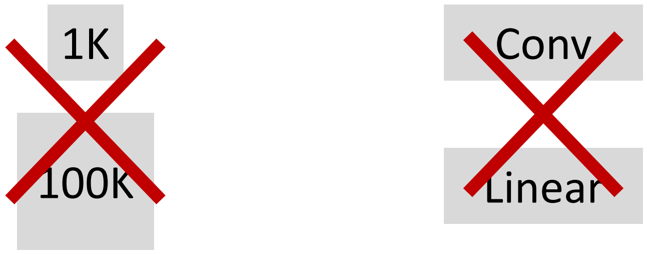
Parameter sharing



All of the above

Limitations of Standard Parameter Sharing

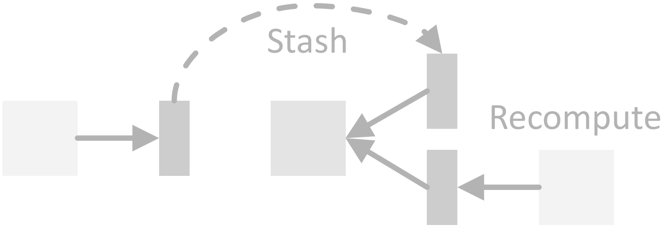
Only share between identical layers



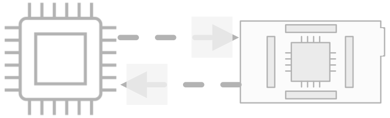
How to Break the Memory Wall

Systems

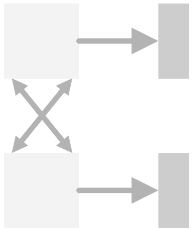
Recomputation



Out-of-core

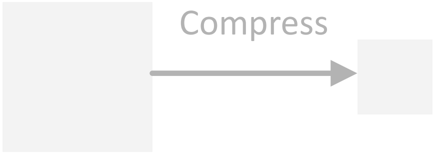


Model parallelism

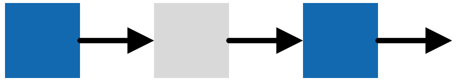


Models

Quantization / Pruning / Low-rank / Distillation



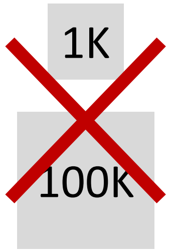
Parameter sharing

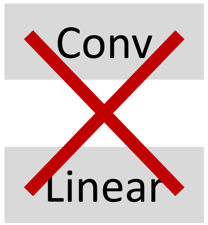


All of the above

Limitations of Standard Parameter Sharing

Only share between identical layers



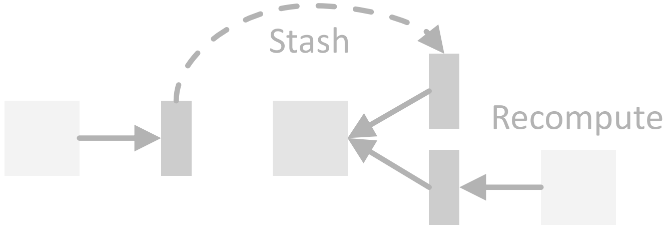


Does not support arbitrary parameter budgets

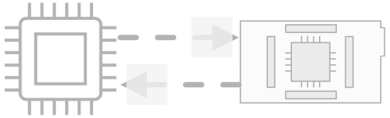
How to Break the Memory Wall

Systems

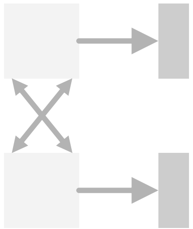
Recomputation



Out-of-core

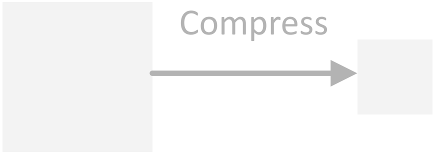


Model parallelism

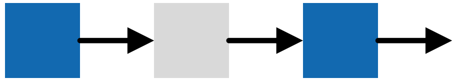


Models

Quantization / Pruning / Low-rank / Distillation



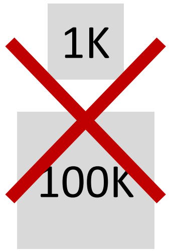
Parameter sharing

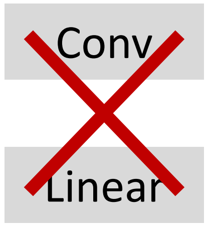


All of the above

Limitations of Standard Parameter Sharing

Only share between identical layers





Does not support arbitrary parameter budgets

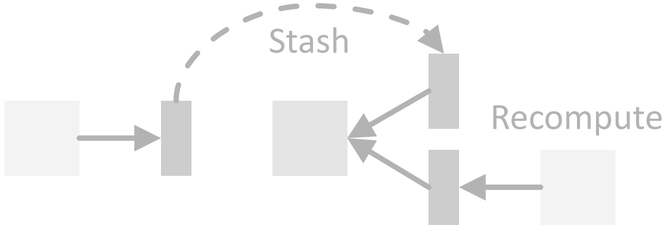
Budget

10K

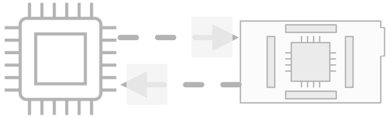
How to Break the Memory Wall

Systems

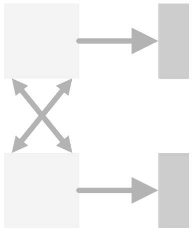
Recomputation



Out-of-core

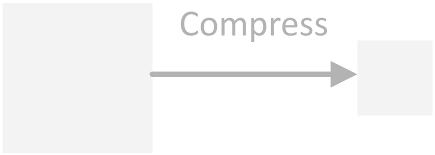


Model parallelism

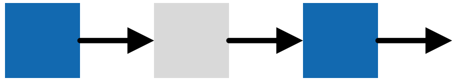


Models

Quantization / Pruning / Low-rank / Distillation



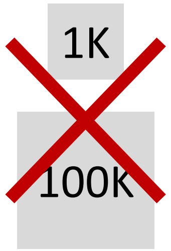
Parameter sharing

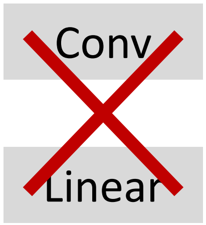


All of the above

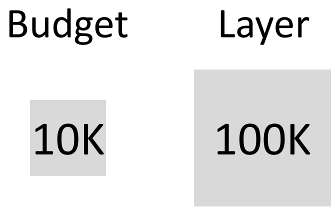
Limitations of Standard Parameter Sharing

Only share between identical layers





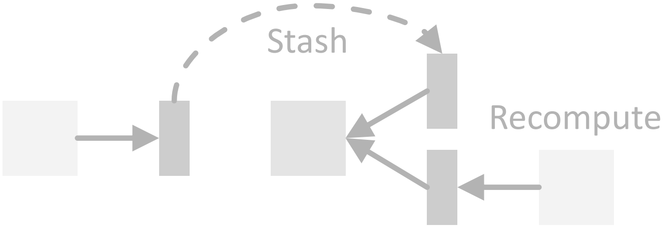
Does not support arbitrary parameter budgets



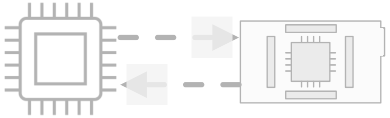
How to Break the Memory Wall

Systems

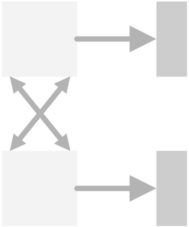
Recomputation



Out-of-core

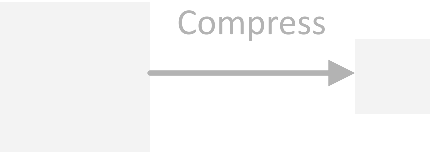


Model parallelism

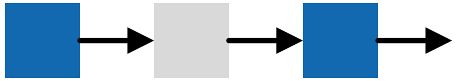


Models

Quantization / Pruning / Low-rank / Distillation



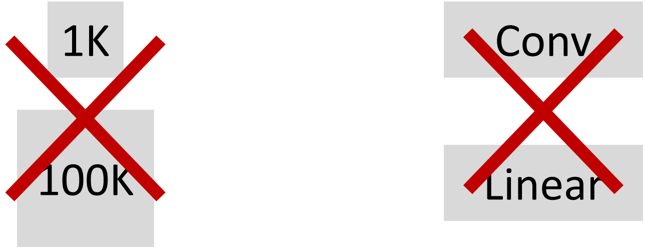
Parameter sharing



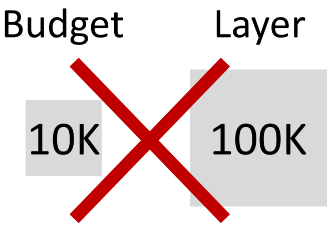
All of the above

Limitations of Standard Parameter Sharing

Only share between identical layers



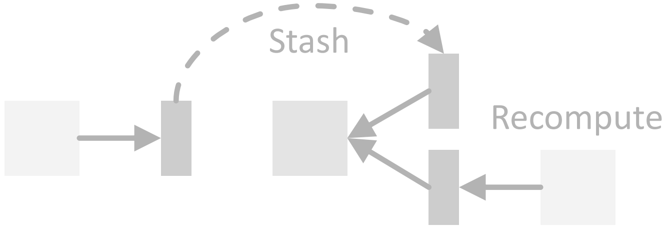
Does not support arbitrary parameter budgets



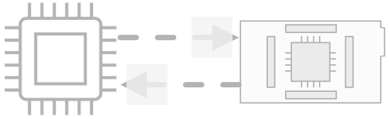
How to Break the Memory Wall

Systems

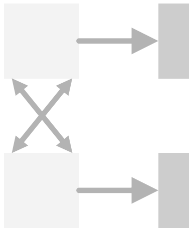
Recomputation



Out-of-core

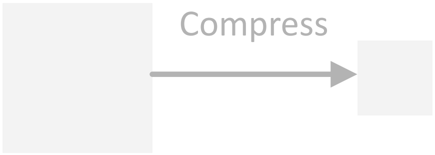


Model parallelism

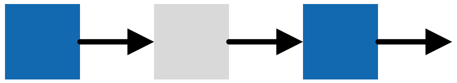


Models

Quantization / Pruning / Low-rank / Distillation



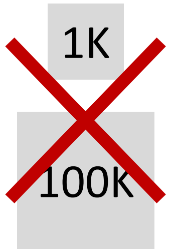
Parameter sharing

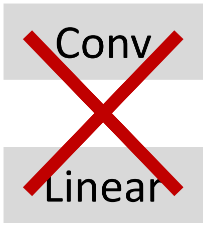


All of the above

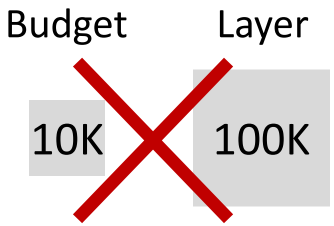
Limitations of Standard Parameter Sharing

Only share between identical layers





Does not support arbitrary parameter budgets



Sharing is manually designed

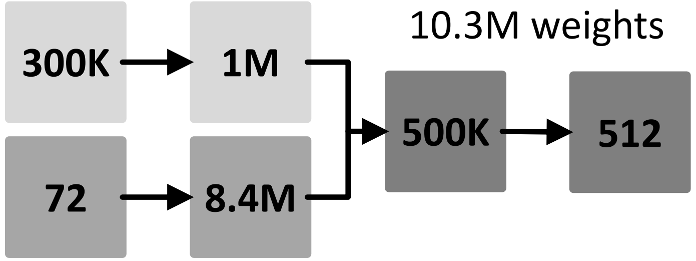
Neural Parameter Allocation Search (NPAS)

Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.

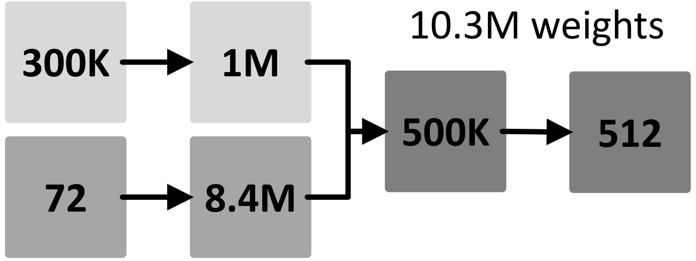
Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



Neural Parameter Allocation Search (NPAS)

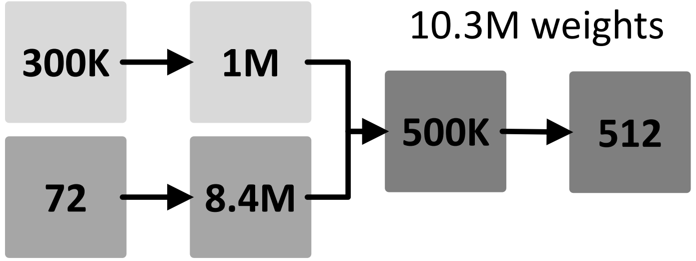
Using *any* parameter budget, train a high-performing neural network using that parameter budget.



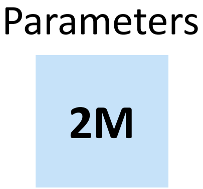
Low-budget NPAS

Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.

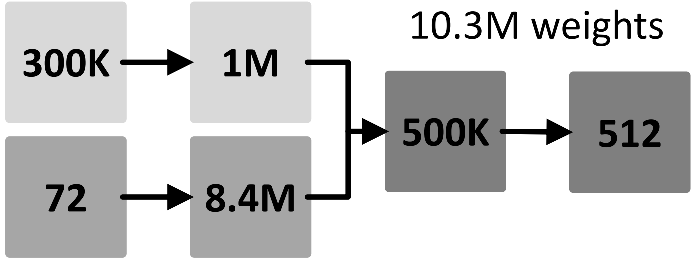


Low-budget NPAS

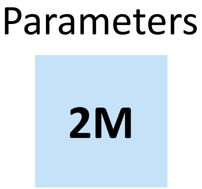


Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.

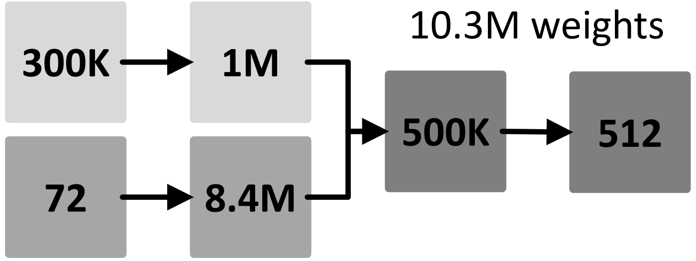


Low-budget NPAS **High-budget NPAS**

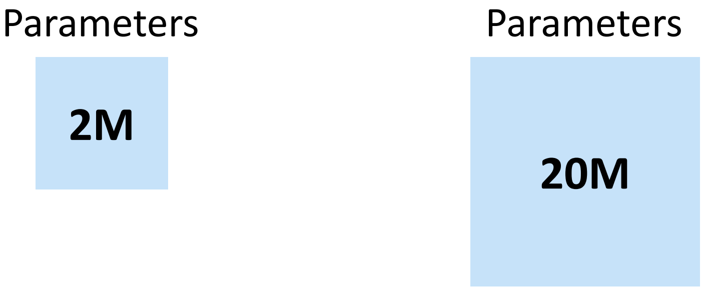


Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.

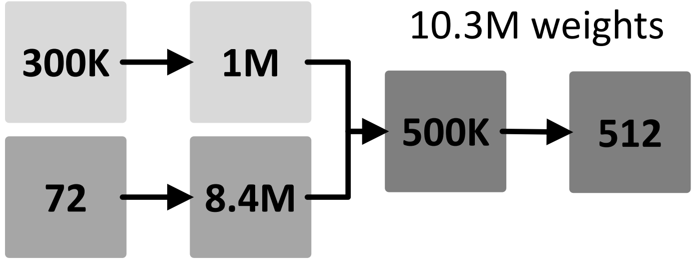


Low-budget NPAS **High-budget NPAS**



Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



10.3M weights

300K

1M

72

8.4M

500K

512

Low-budget NPAS

Parameters

2M

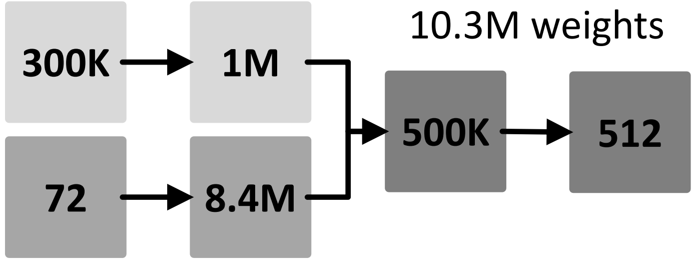
High-budget NPAS

Parameters

20M

Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



1. Parameter mapping

Low-budget NPAS

High-budget NPAS

Parameters

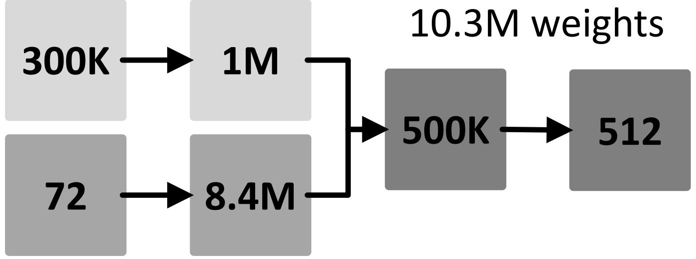
2M

Parameters

20M

Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



1. Parameter mapping

Low-budget NPAS

Parameters

2M

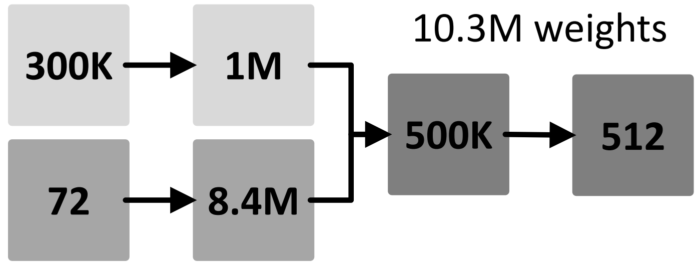
High-budget NPAS

Parameters

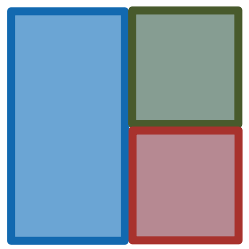
20M

Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



1. Parameter mapping



Low-budget NPAS

High-budget NPAS

Parameters

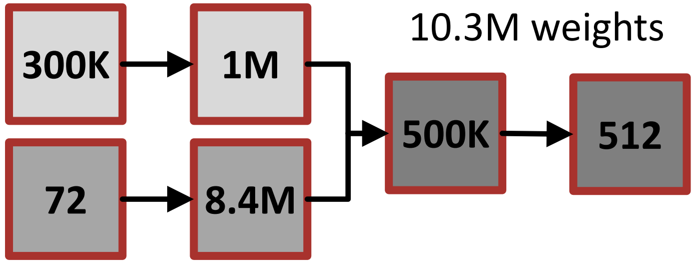
2M

Parameters

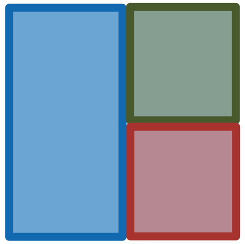
20M

Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



1. Parameter mapping



Low-budget NPAS

Parameters

2M

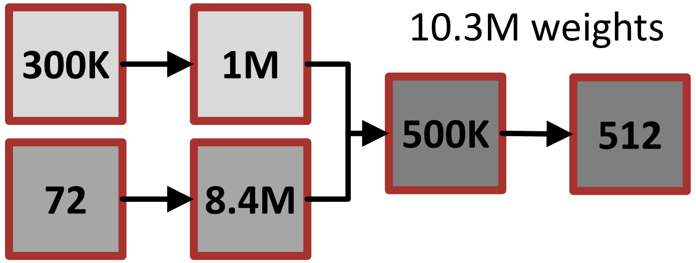
High-budget NPAS

Parameters

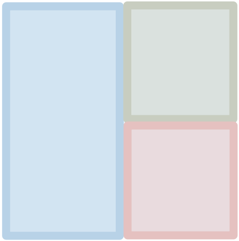
20M

Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



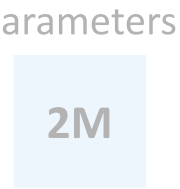
1. Parameter mapping

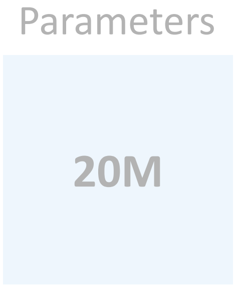


2. Weight generation

Low-budget NPAS

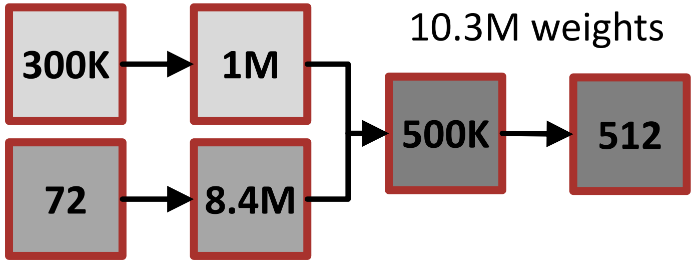
High-budget NPAS



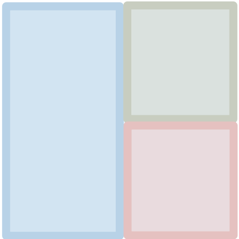


Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



1. Parameter mapping

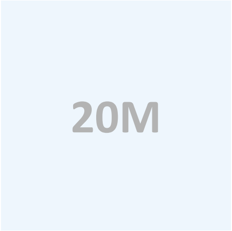


Low-budget NPAS

High-budget NPAS

Parameters

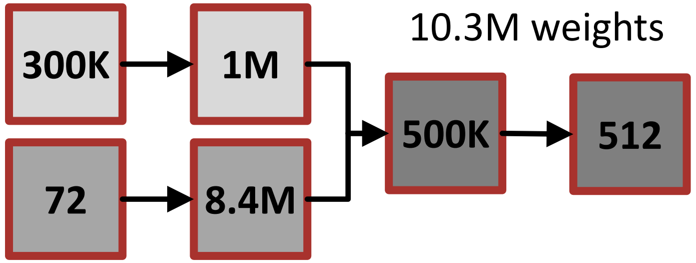
Parameters



2. Weight generation

Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.

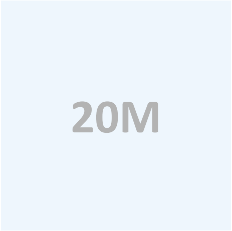


Low-budget NPAS

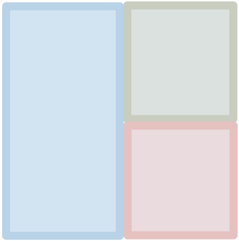
Parameters

High-budget NPAS

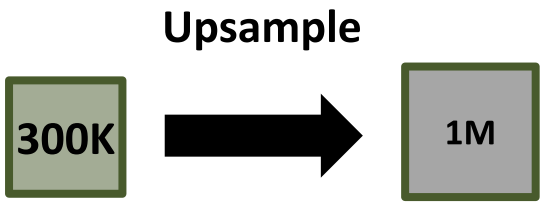
Parameters



1. Parameter mapping

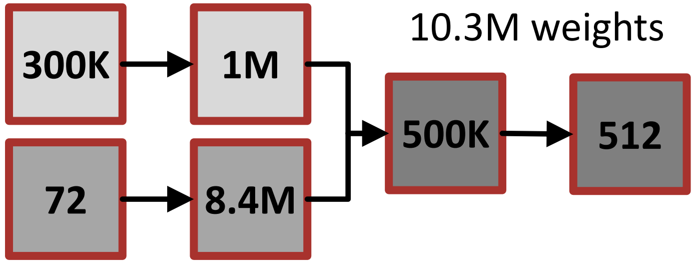


2. Weight generation

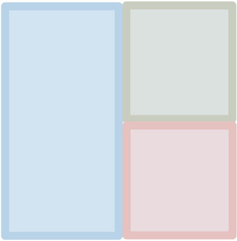


Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.

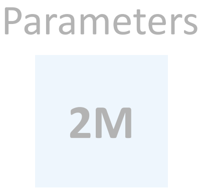


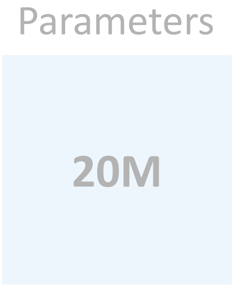
1. Parameter mapping



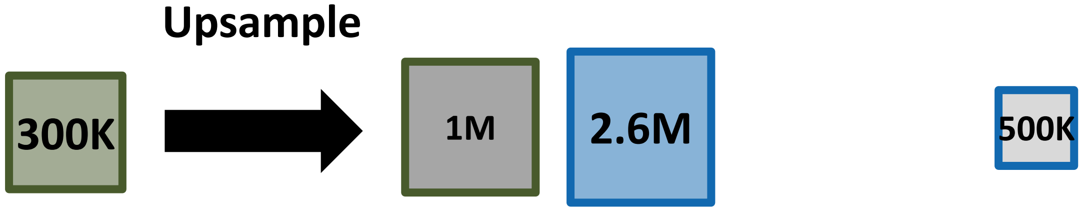
Low-budget NPAS

High-budget NPAS



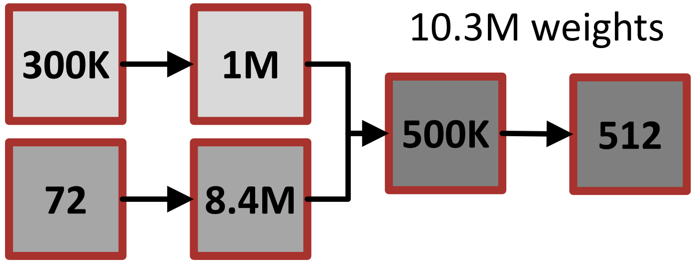


2. Weight generation

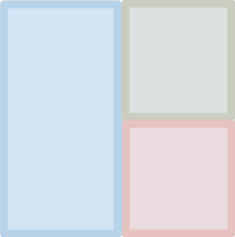


Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



1. Parameter mapping

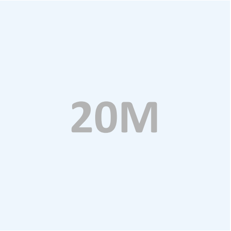


Low-budget NPAS

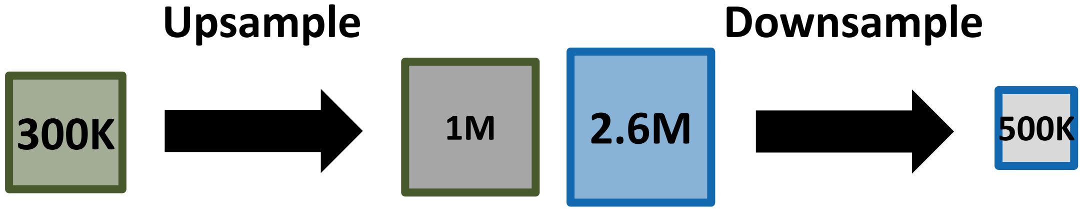
High-budget NPAS

Parameters

Parameters

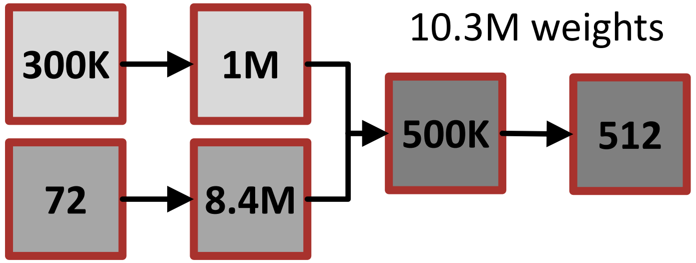


2. Weight generation

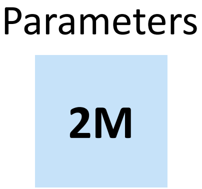


Neural Parameter Allocation Search (NPAS)

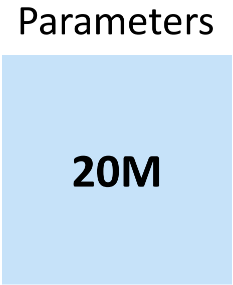
Using *any* parameter budget, train a high-performing neural network using that parameter budget.



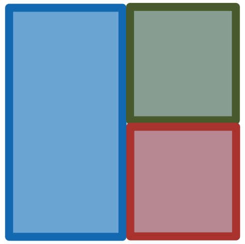
Low-budget NPAS



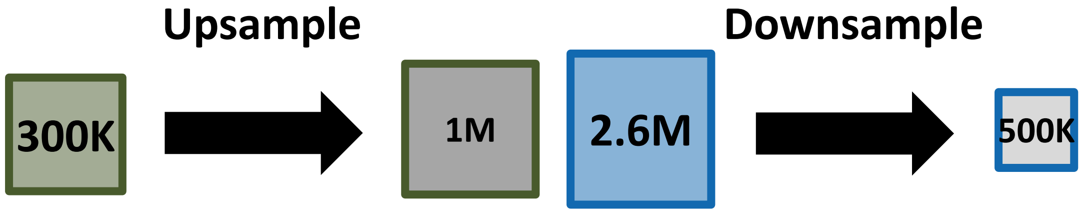
High-budget NPAS



1. Parameter mapping



2. Weight generation



Advantages of Parameter Sharing

Advantages of Parameter Sharing

Parameter sharing reduces memory during training and inference

(In the low-budget case)

Advantages of Parameter Sharing

Parameter sharing reduces memory during training and inference

(In the low-budget case)

The same model …

Advantages of Parameter Sharing

Parameter sharing reduces memory during training and inference

(In the low-budget case)

The same model …

Fits in smaller devices

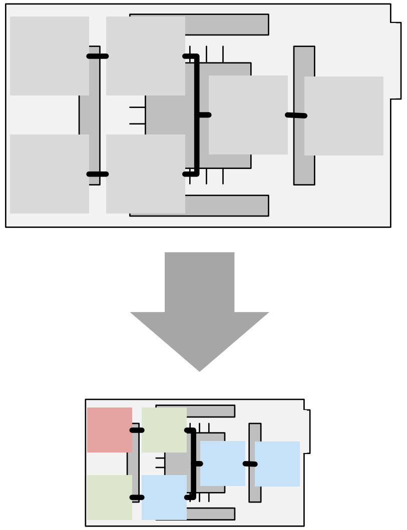
Advantages of Parameter Sharing

Parameter sharing reduces memory during training and inference

(In the low-budget case)

The same model …

Fits in smaller devices



Advantages of Parameter Sharing

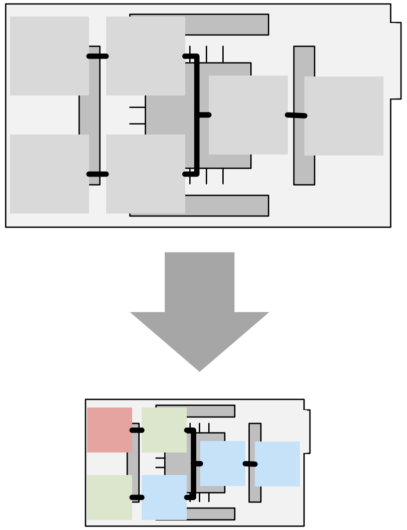
Parameter sharing reduces memory during training and inference

(In the low-budget case)

The same model …

Fits in smaller devices

Uses less communication



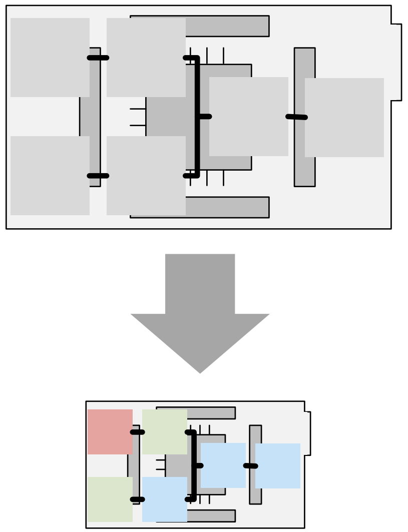
Advantages of Parameter Sharing

Parameter sharing reduces memory during training and inference

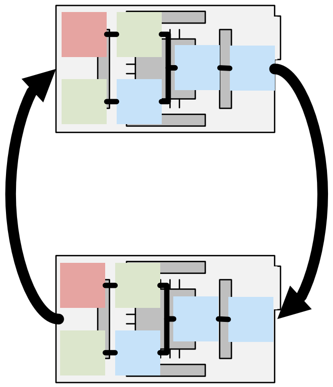
(In the low-budget case)

The same model …

Fits in smaller devices



Uses less communication



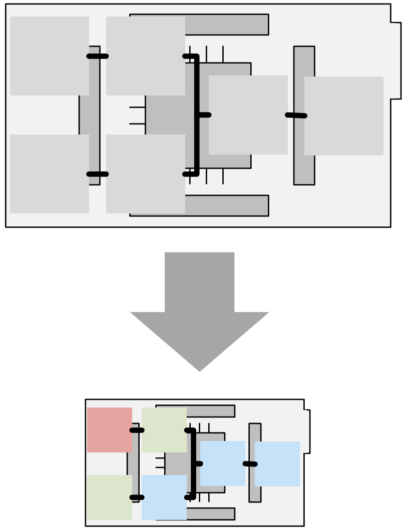
Advantages of Parameter Sharing

Parameter sharing reduces memory during training and inference

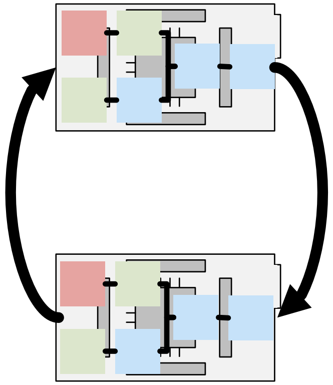
(In the low-budget case)

The same model …

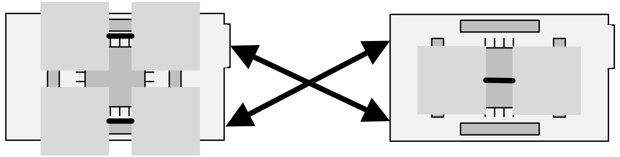
Fits in smaller devices



Uses less communication



Needs less model-parallelism



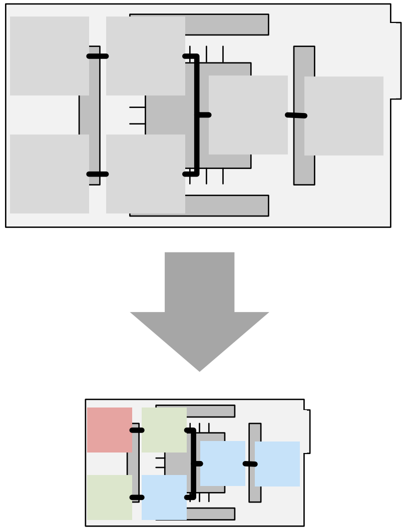
Advantages of Parameter Sharing

Parameter sharing reduces memory during training and inference

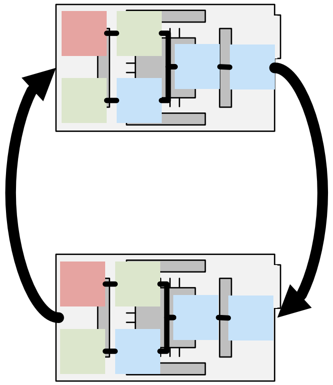
(In the low-budget case)

The same model …

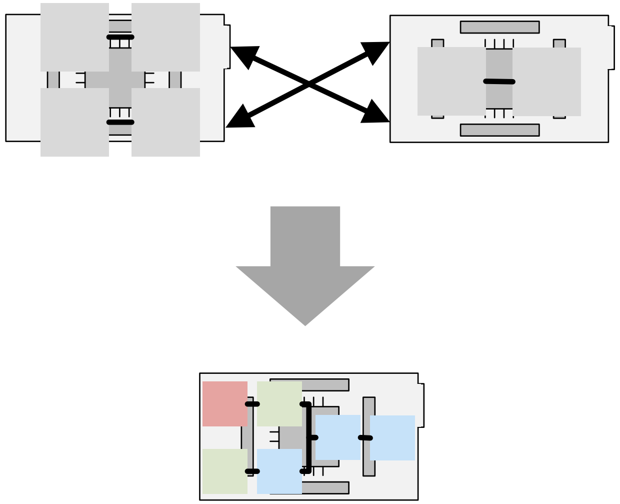
Fits in smaller devices



Uses less communication

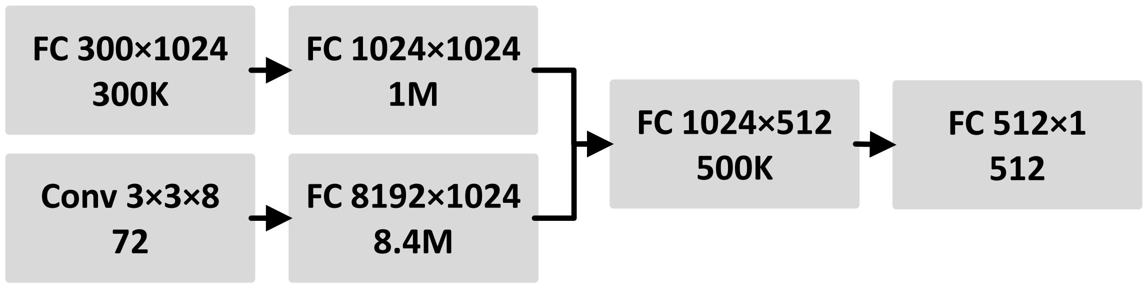


Needs less model-parallelism

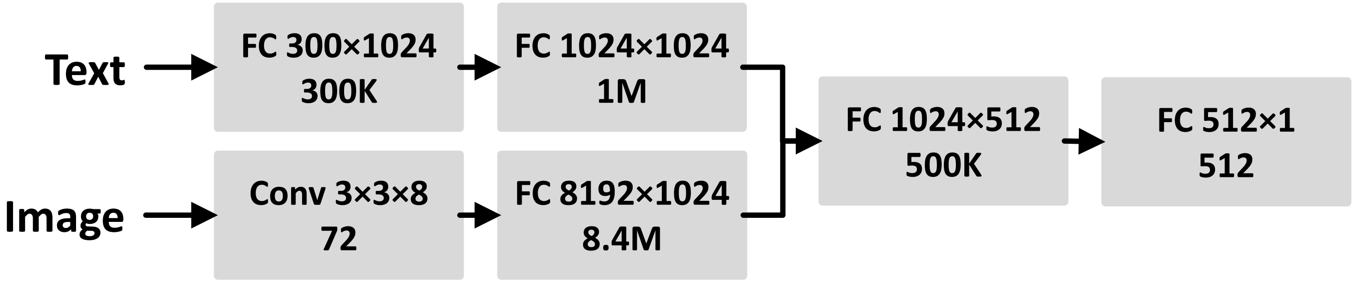


Shapeshifter Networks (SSNs)

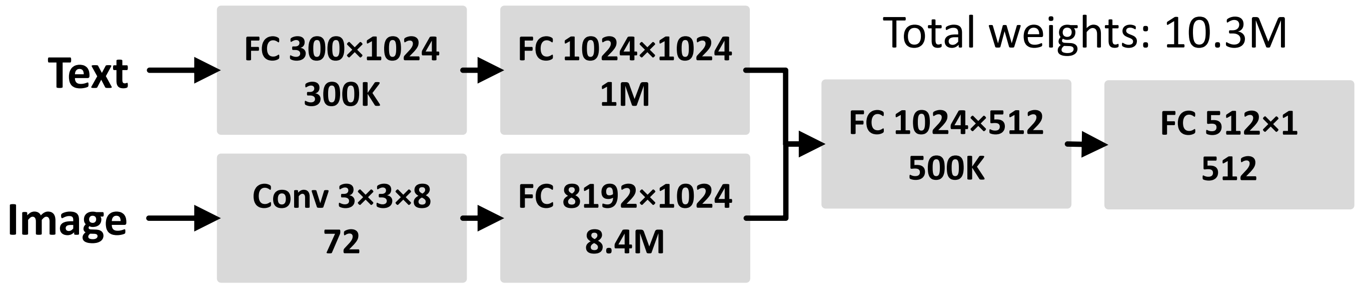
Shapeshifter Networks (SSNs)



Shapeshifter Networks (SSNs)



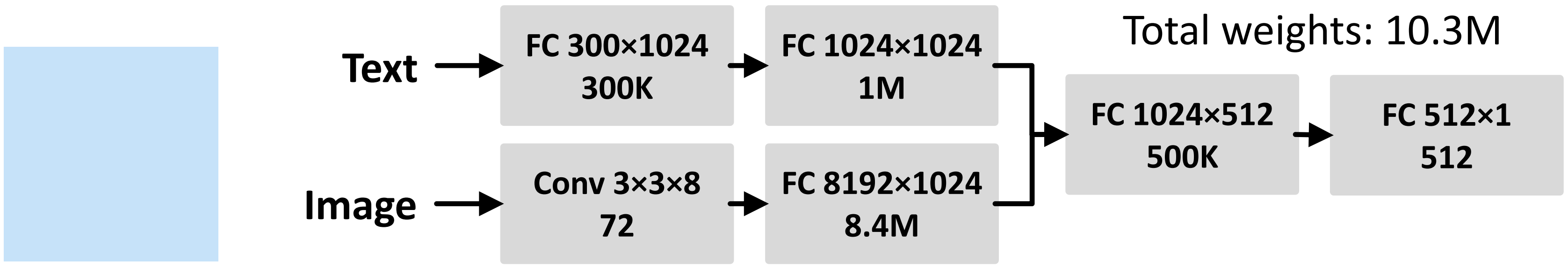
Shapeshifter Networks (SSNs)



Total weights: 10.3M

Shapeshifter Networks (SSNs)

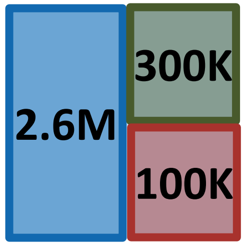
Parameter budget: 3M



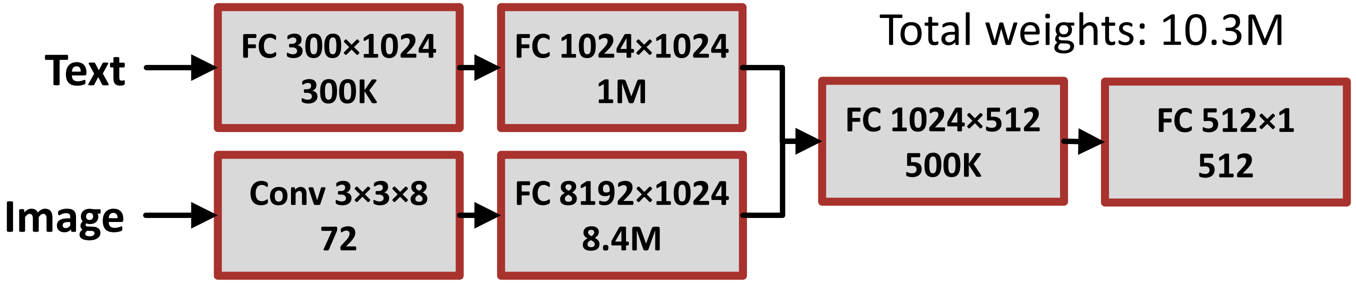
Shapeshifter Networks (SSNs)

Parameter budget: 3M

$P = 3$ **parameter groups**

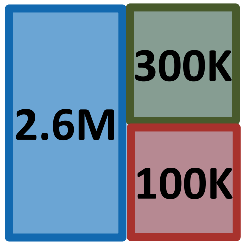


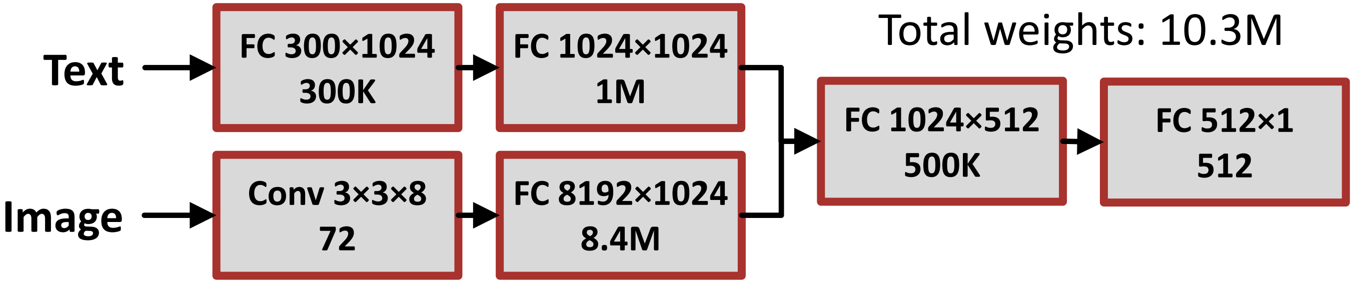
Total weights: 10.3M



Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups



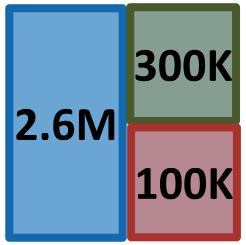


Total weights: 10.3M

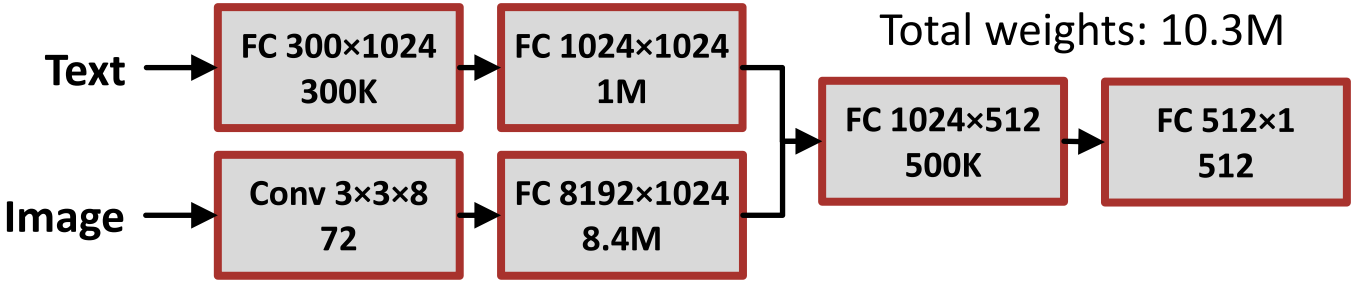
Weight Generation

Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups



Total weights: 10.3M

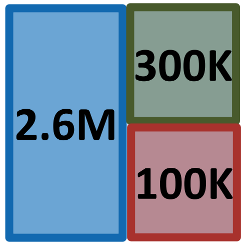


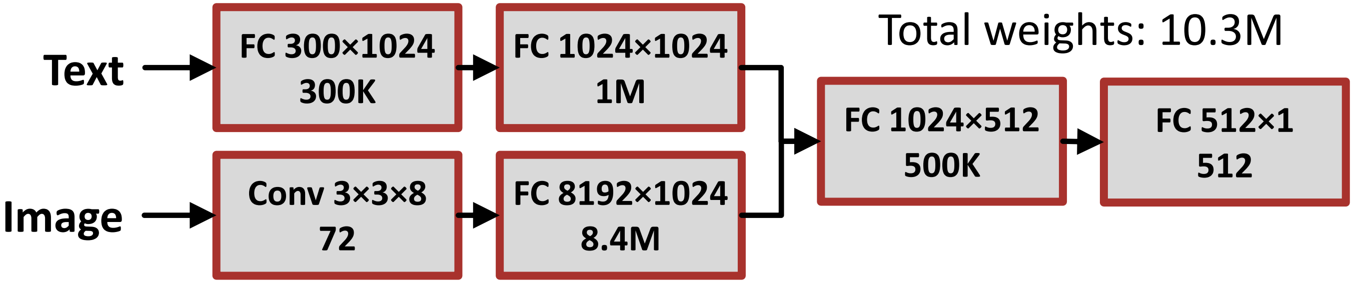
Weight Generation

Upsample:

Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups



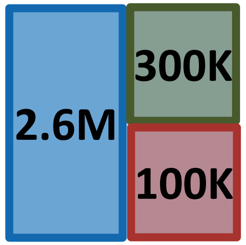


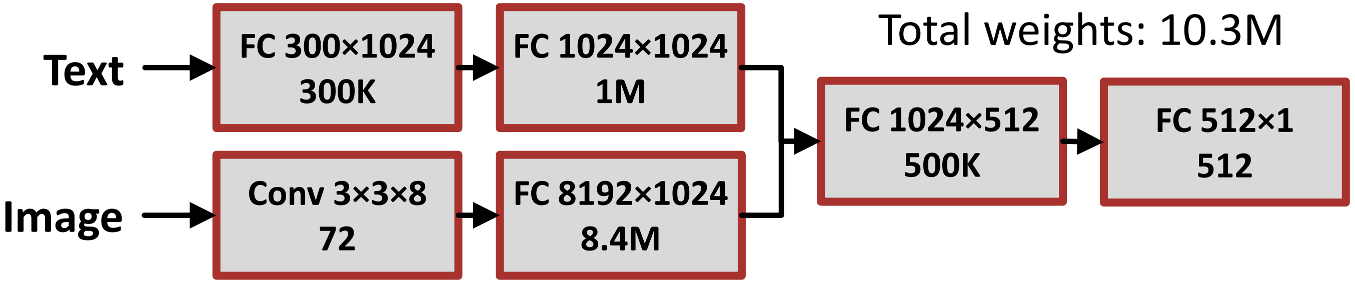
Weight Generation

Upsample: Interpolate

Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ **parameter groups**

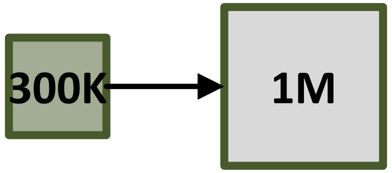




Total weights: 10.3M

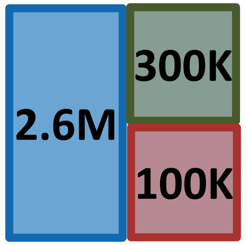
Weight Generation

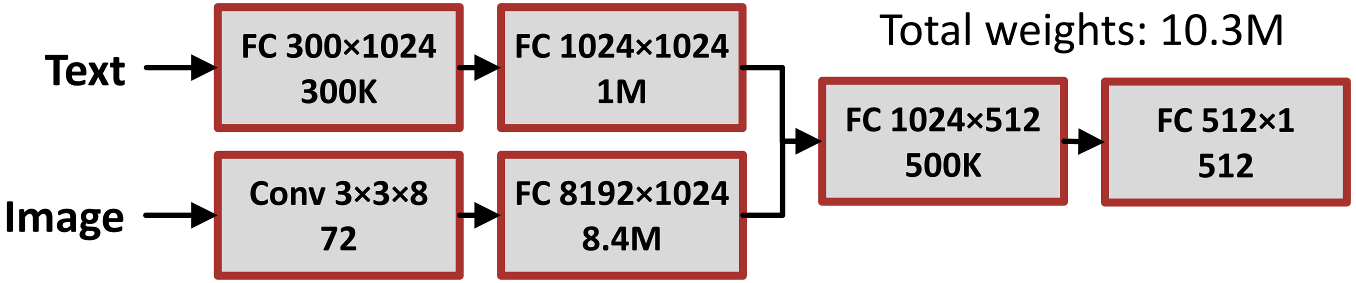
Upsample: Interpolate



Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups

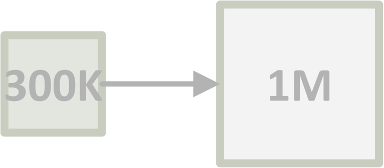




Total weights: 10.3M

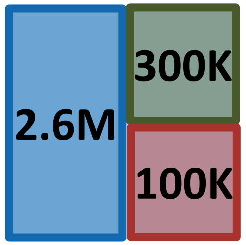
Weight Generation

Upsample: Interpolate Mask

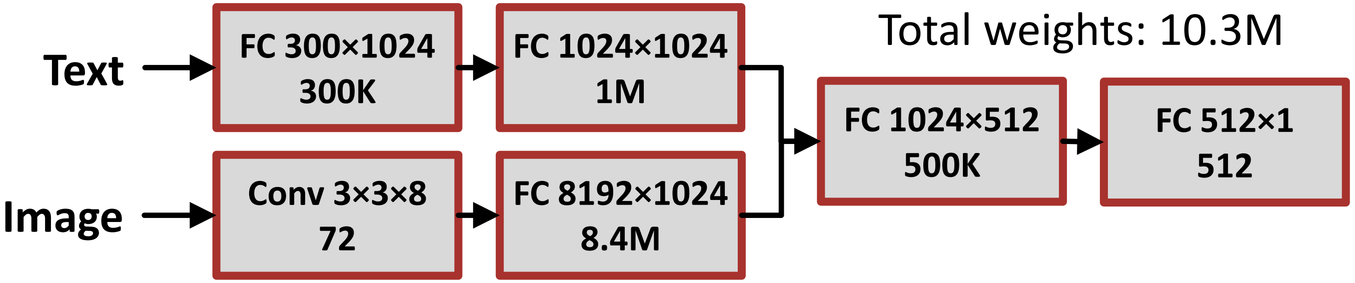


Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups

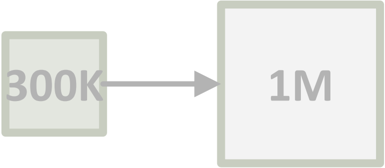


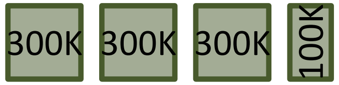
Total weights: 10.3M



Weight Generation

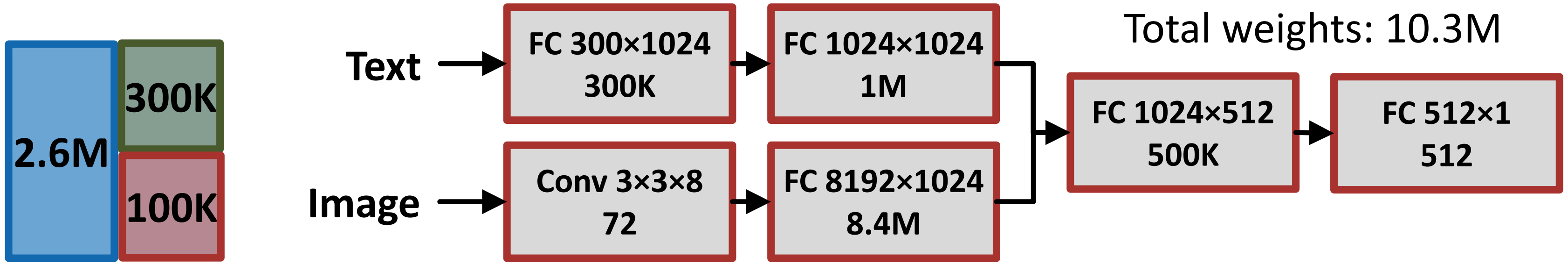
Upsample: Interpolate Mask





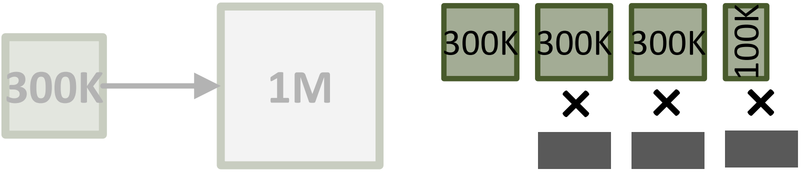
Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ **parameter groups**



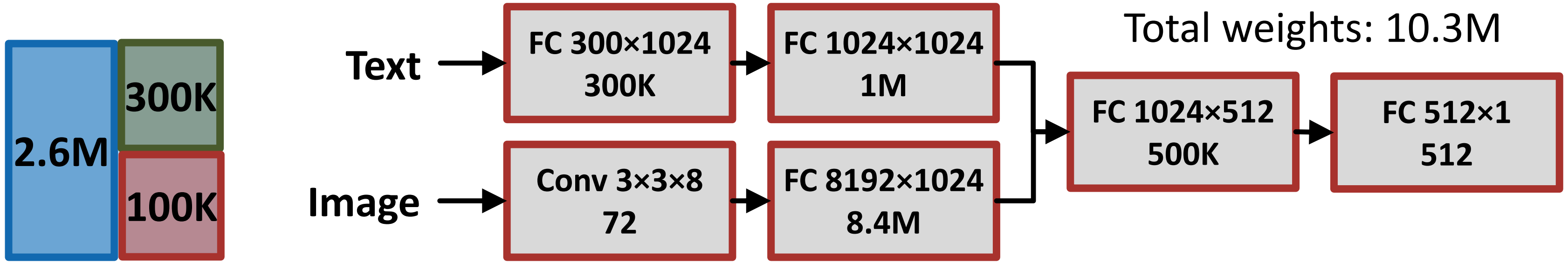
Weight Generation

Upsample: Interpolate Mask



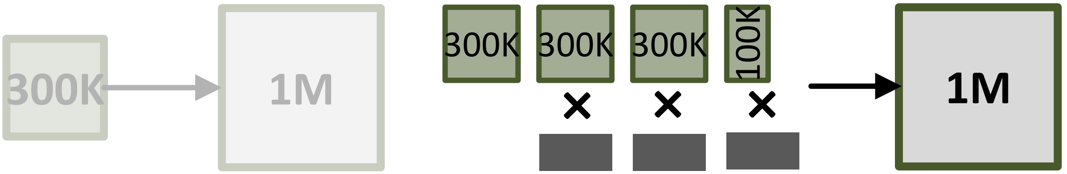
Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups



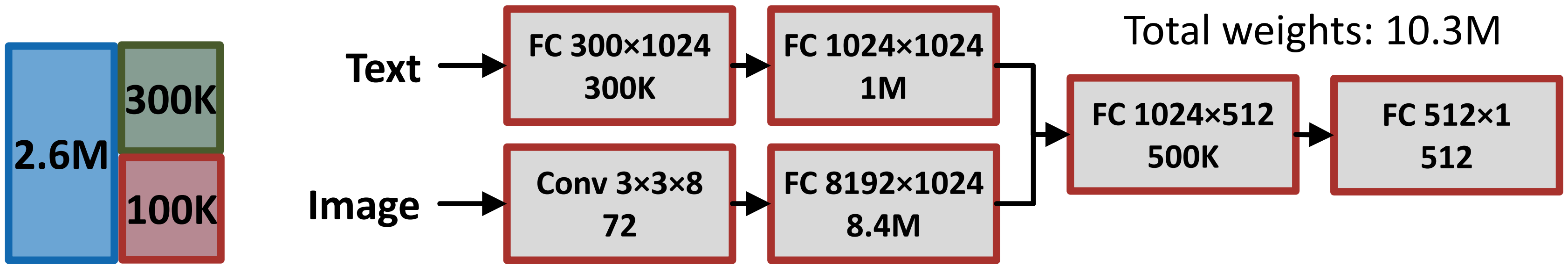
Weight Generation

Upsample: Interpolate Mask



Shapeshifter Networks (SSNs)

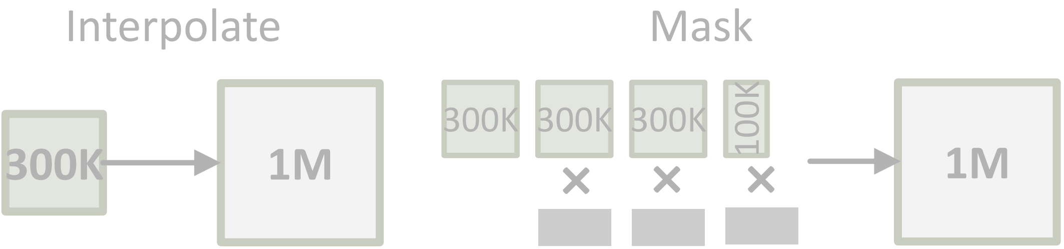
Parameter budget: 3M
$P = 3$ parameter groups



Total weights: 10.3M

Weight Generation

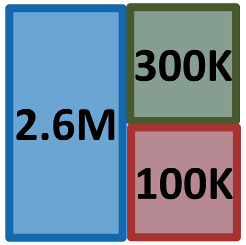
Upsample:

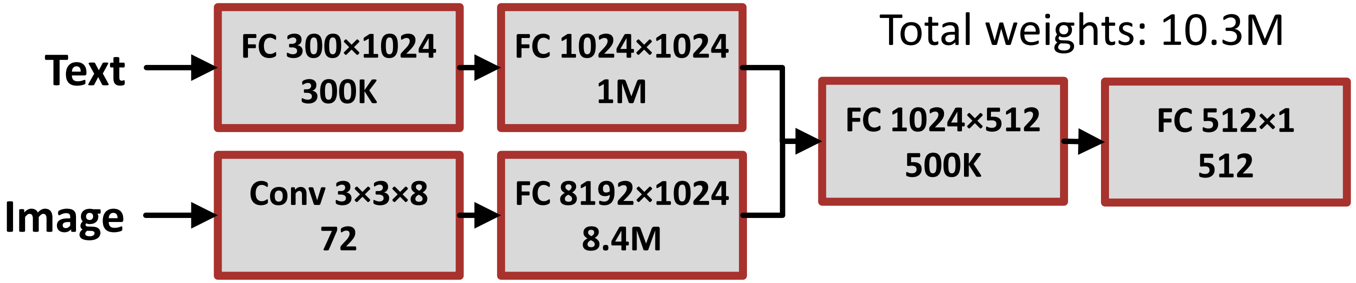


Downsample:

Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups




Total weights: 10.3M

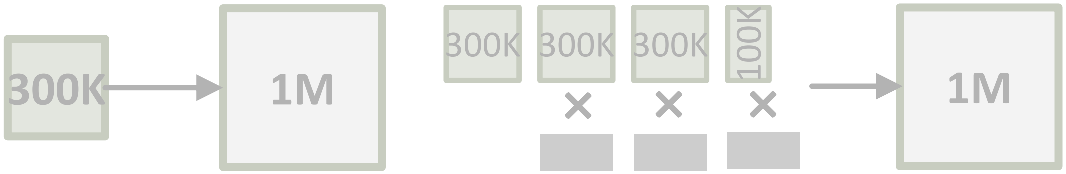
Text → FC 300×1024 300K → FC 1024×1024 1M

Image → Conv 3×3×8 72 → FC 8192×1024 8.4M

→ FC 1024×512 500K → FC 512×1 512

Weight Generation

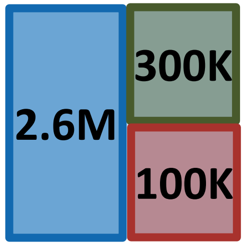
Upsample: Interpolate Mask

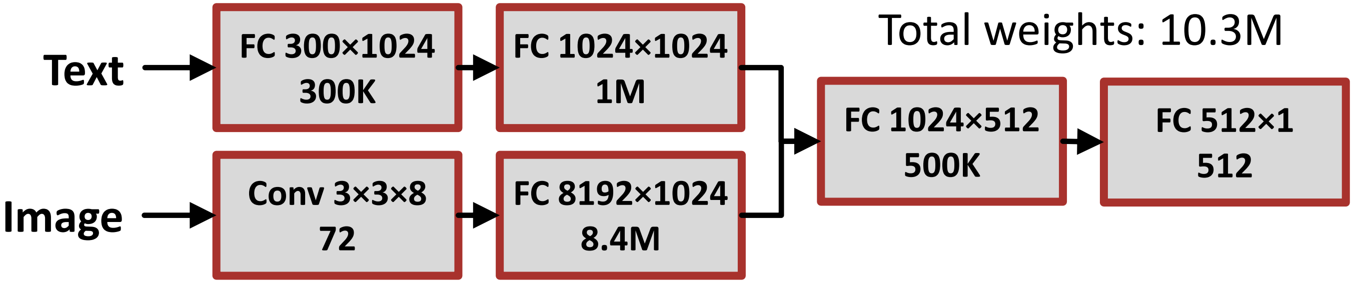


Downsample:

Shapeshifter Networks (SSNs)

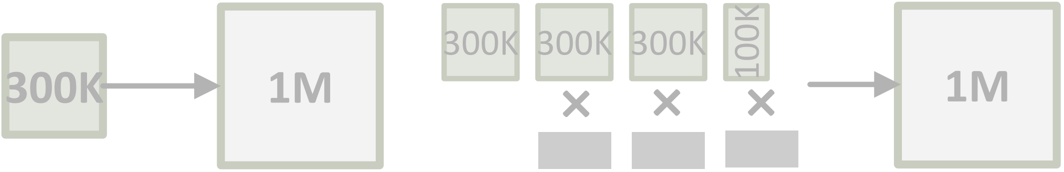
Parameter budget: 3M
$P = 3$ **parameter groups**





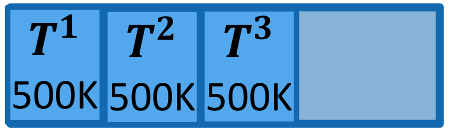
Weight Generation

Upsample: Interpolate Mask



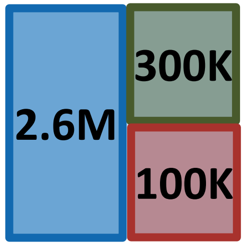
Downsample:

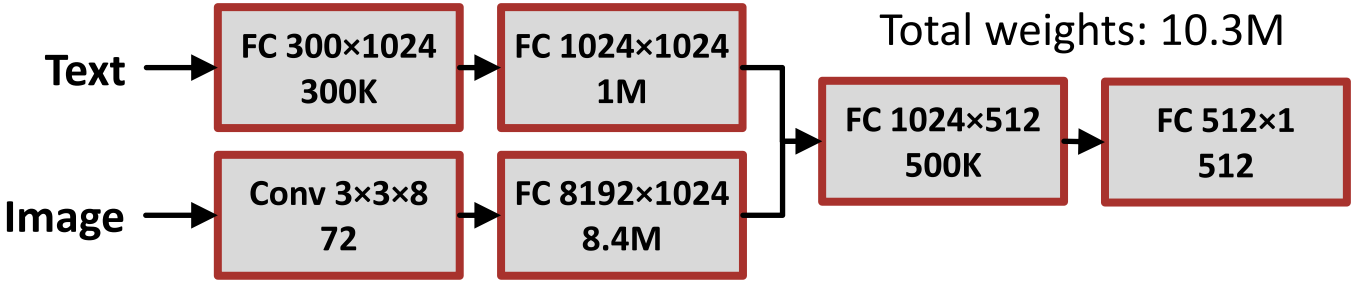
$K = 3$ templates



Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ **parameter groups**

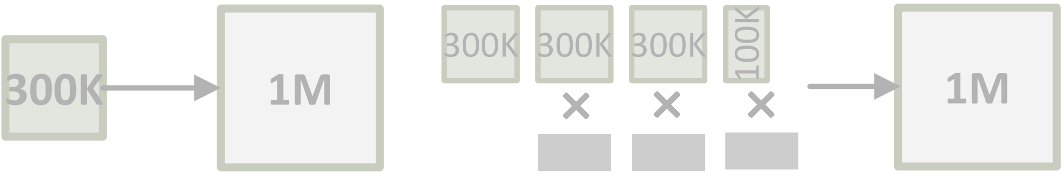




Total weights: 10.3M

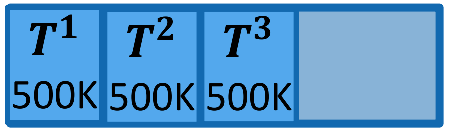
Weight Generation

Upsample: Interpolate Mask



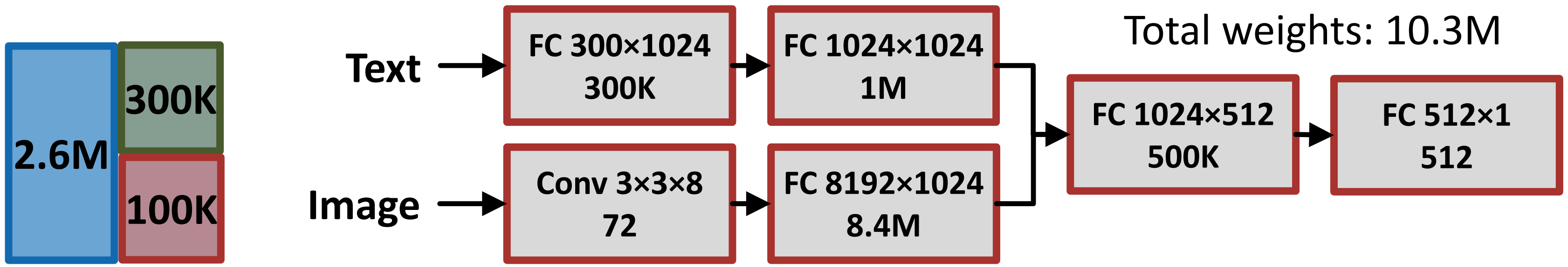
Downsample: WAvg
[Savarese & Maire, 2019]

$K = 3$ templates



Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups



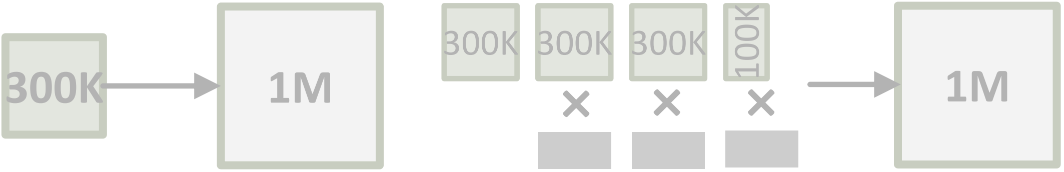
Total weights: 10.3M

Weight Generation

Upsample:

Interpolate

Mask

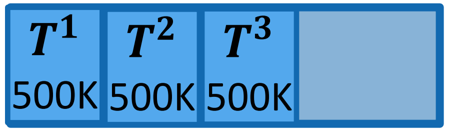


Downsample:

WAvg
[Savarese & Maire, 2019]

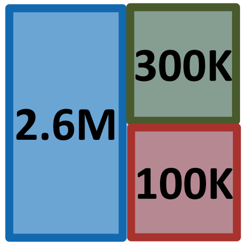
Coefficients α_i

$K = 3$ templates

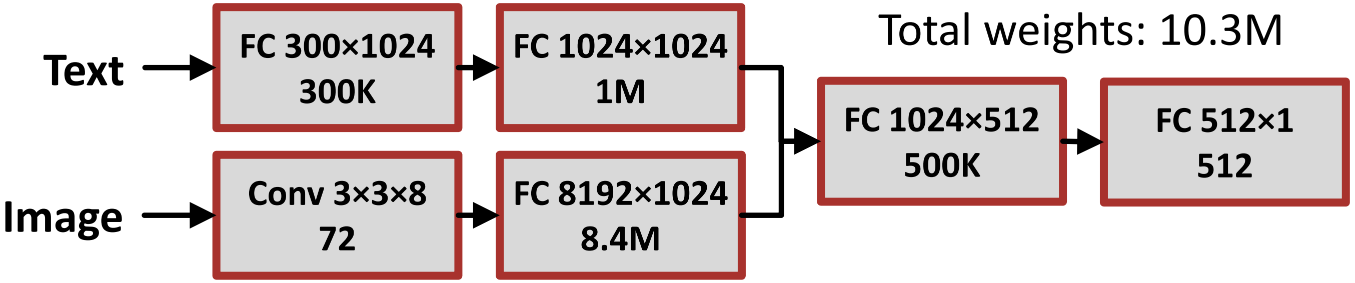


Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups

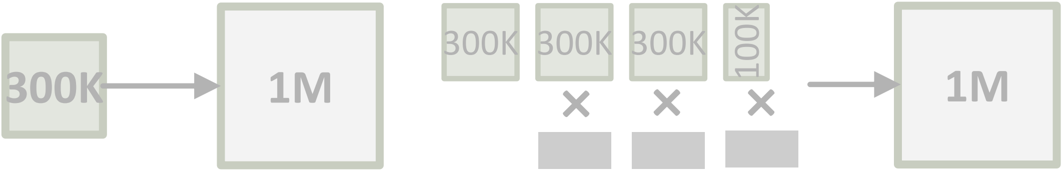


Total weights: 10.3M



Weight Generation

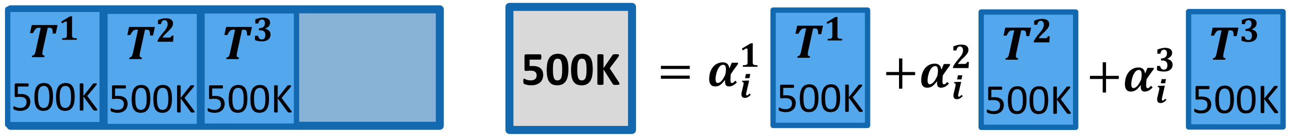
Upsample: Interpolate Mask



Downsample: WAvg
[Savarese & Maire, 2019]

$K = 3$ templates

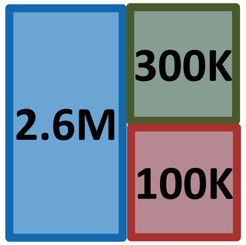
Coefficients α_i

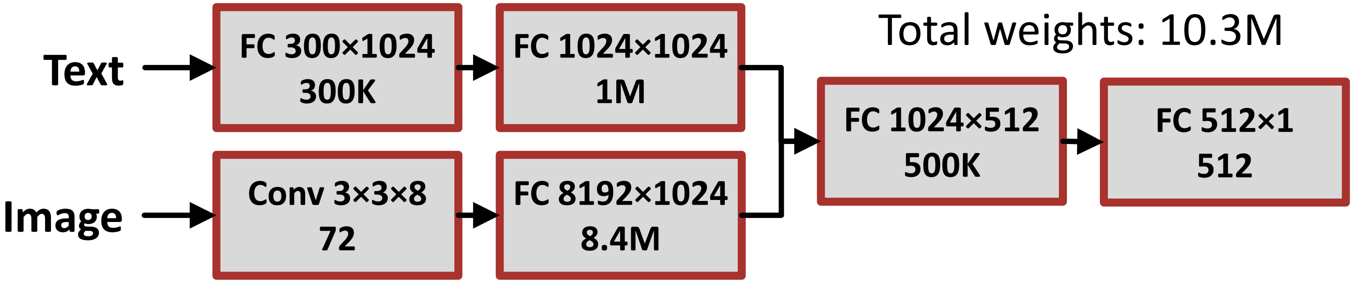


Shapeshifter Networks (SSNs)

Parameter budget: 3M

$P = 3$ parameter groups

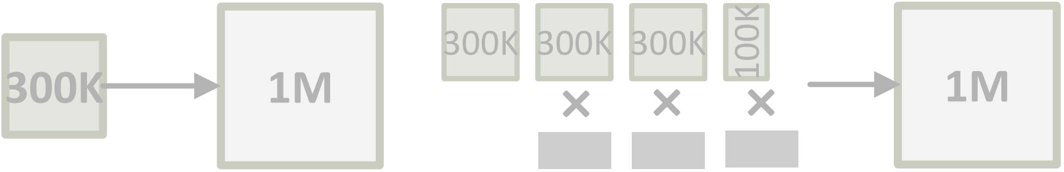




Total weights: 10.3M

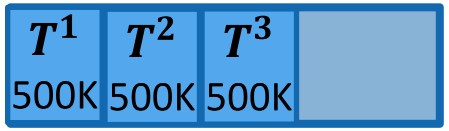
Weight Generation

Upsample: Interpolate Mask

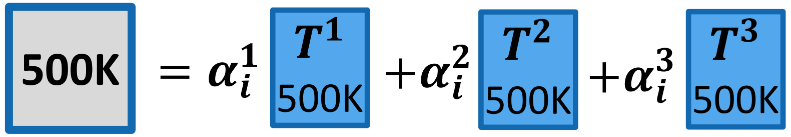


Downsample: WAvg [Savarese & Maire, 2019] Embedding

$K = 3$ templates

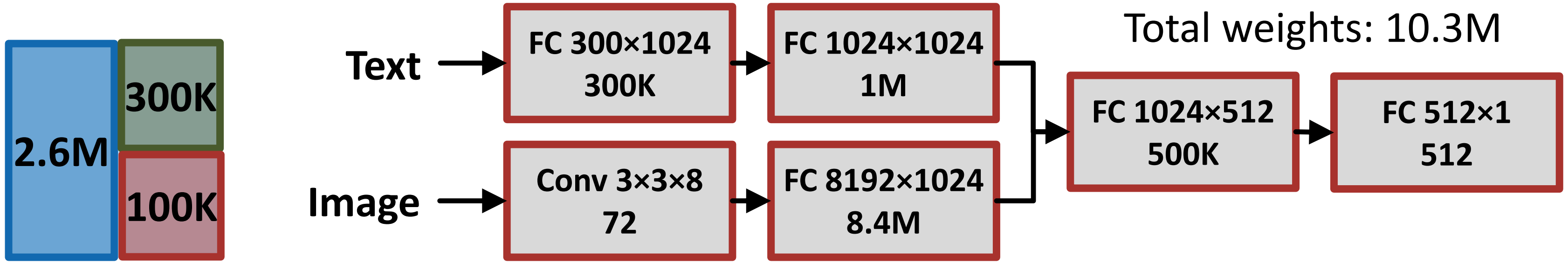


Coefficients α_i



Shapeshifter Networks (SSNs)

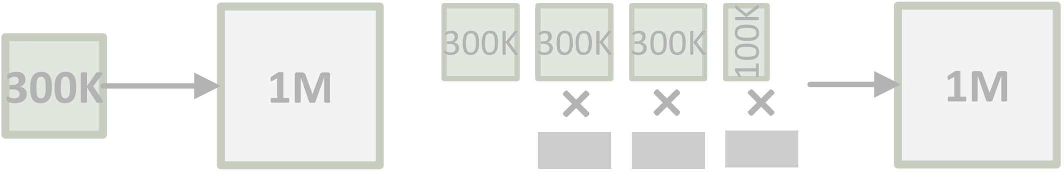
Parameter budget: 3M
$\boldsymbol{P = 3}$ **parameter groups**



Total weights: 10.3M

Weight Generation

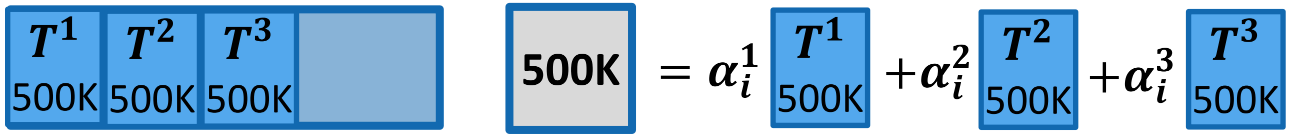
Upsample: Interpolate Mask



Downsample: WAvg [Savarese & Maire, 2019] Embedding

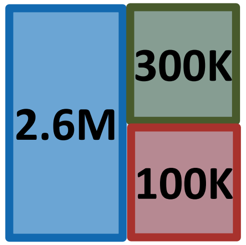
Coefficients α_i Vector ϕ_i

$K = 3$ templates

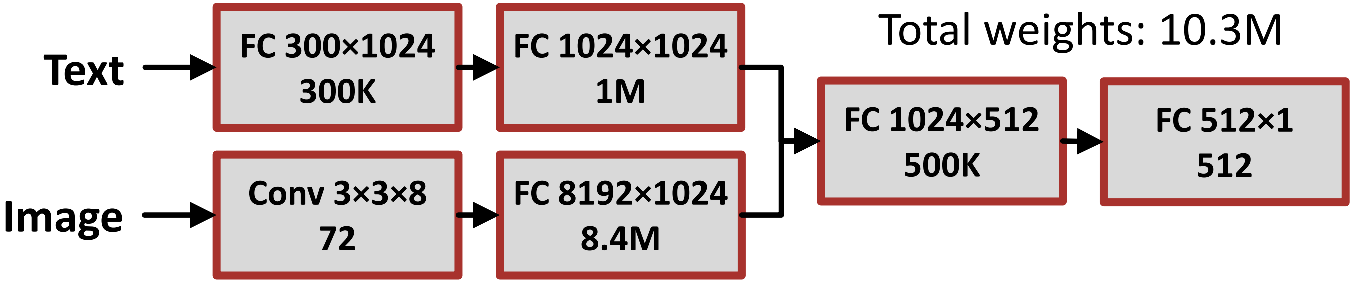


Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ **parameter groups**

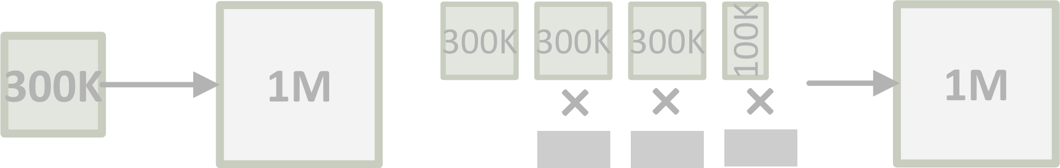


Total weights: 10.3M



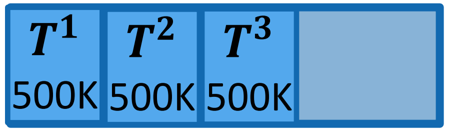
Weight Generation

Upsample: Interpolate Mask



Downsample: WAvg [Savarese & Maire, 2019] Embedding

$K = 3$ templates



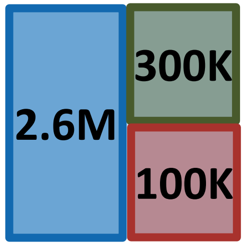
Coefficients α_i

Vector $\phi_i \longrightarrow \alpha_i = W\phi_i + b$

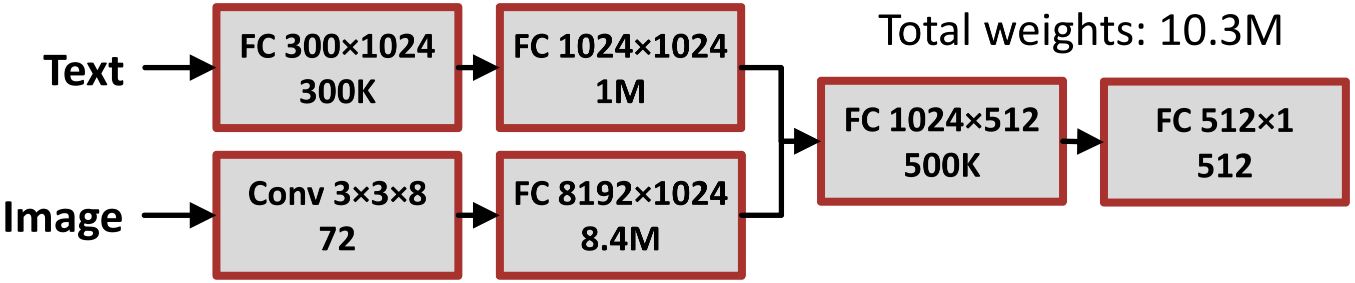
$$\boxed{500K} = \alpha_i^1 \boxed{T^1\ 500K} + \alpha_i^2 \boxed{T^2\ 500K} + \alpha_i^3 \boxed{T^3\ 500K}$$

Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ **parameter groups**


2.6M
300K
100K

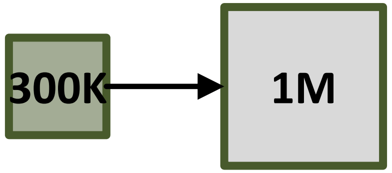
Total weights: 10.3M


Text
FC 300×1024 300K
FC 1024×1024 1M
Image
Conv 3×3×8 72
FC 8192×1024 8.4M
FC 1024×512 500K
FC 512×1 512

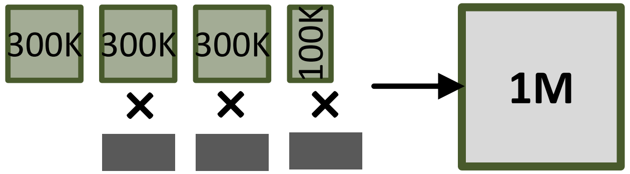
Weight Generation

Upsample:

Interpolate


300K
1M

Mask

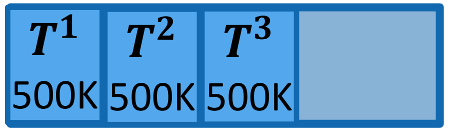

300K 300K 300K 100K
× × ×
1M

Downsample:

WAvg
[Savarese & Maire, 2019]

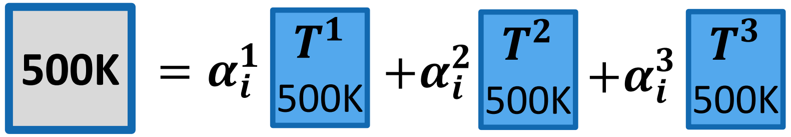
Embedding

$K = 3$ templates


T^1 500K T^2 500K T^3 500K

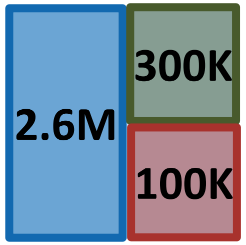
Coefficients α_i

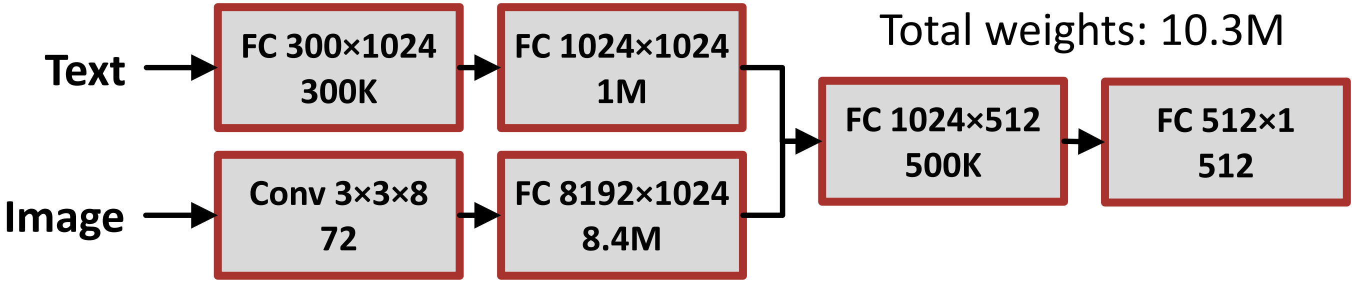
Vector $\phi_i \longrightarrow \alpha_i = W\phi_i + b$


$$500K = \alpha_i^1 \, T^1_{500K} + \alpha_i^2 \, T^2_{500K} + \alpha_i^3 \, T^3_{500K}$$

Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups



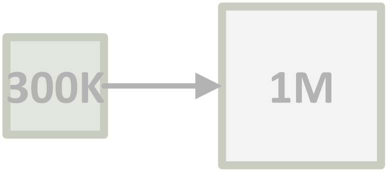


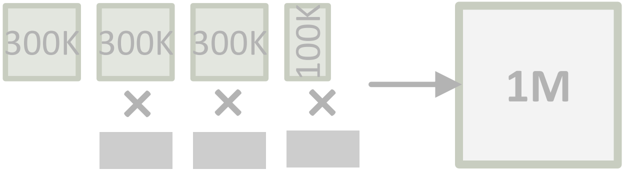
Total weights: 10.3M

Weight Generation

Parameter Mapping

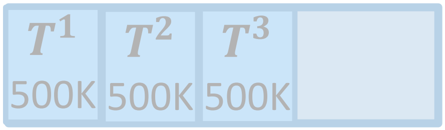
Upsample: Interpolate Mask





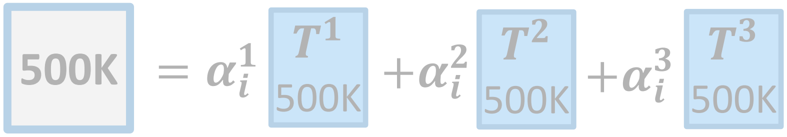
Downsample: WAvg [Savarese & Maire, 2019] Embedding

$K = 3$ templates



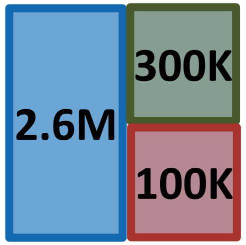
Coefficients α_i

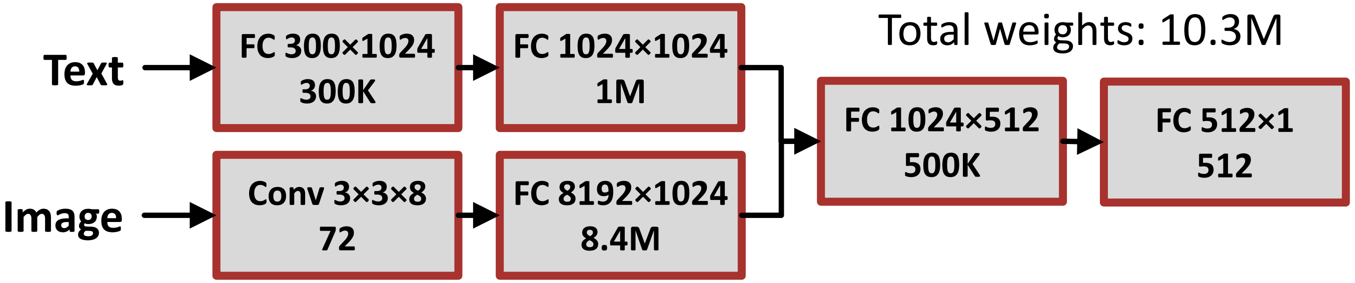
Vector ϕ_i ⟶ $\alpha_i = W\phi_i + b$



Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups

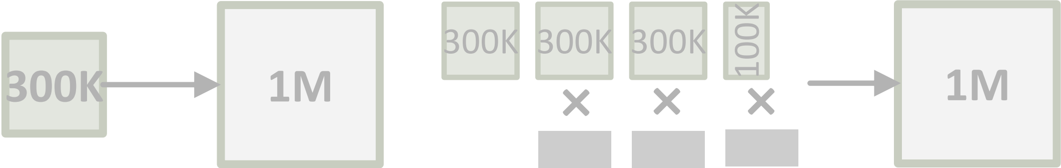




Total weights: 10.3M

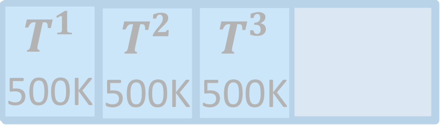
Weight Generation

Upsample: Interpolate Mask



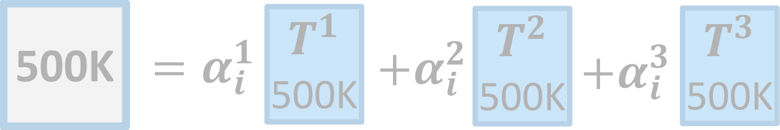
Downsample: WAvg [Savarese & Maire, 2019] Embedding

$K = 3$ templates



Coefficients α_i

Vector $\phi_i \longrightarrow \alpha_i = W\phi_i + b$

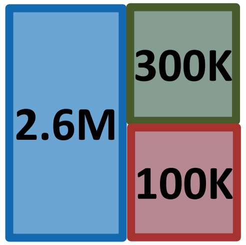


Parameter Mapping

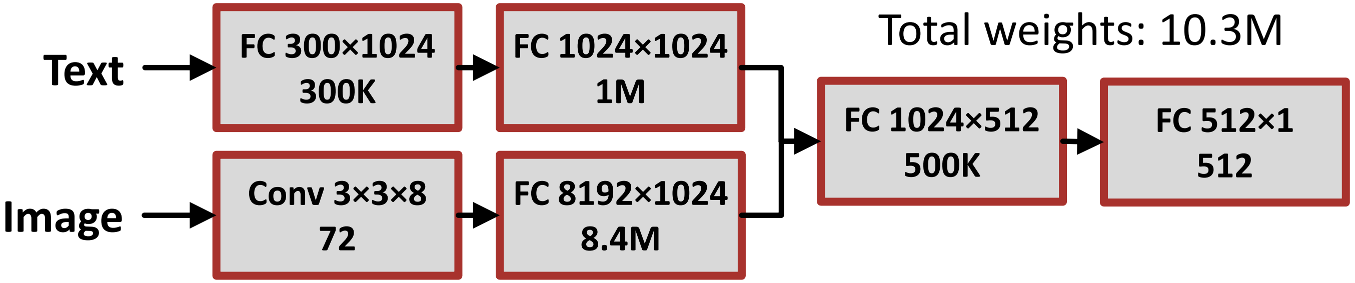
Learn layer representations

Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ **parameter groups**



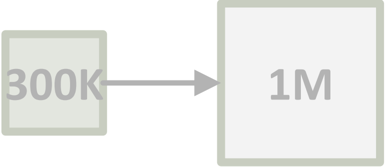
Total weights: 10.3M



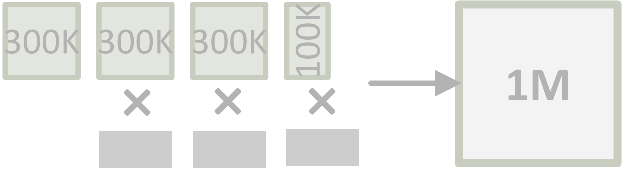
Weight Generation

Upsample:

Interpolate



Mask

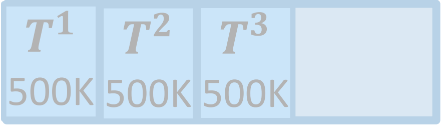


Downsample:

WAvg
[Savarese & Maire, 2019]

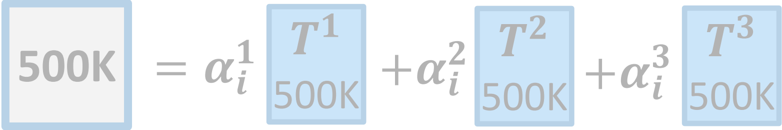
Embedding

$K = 3$ templates



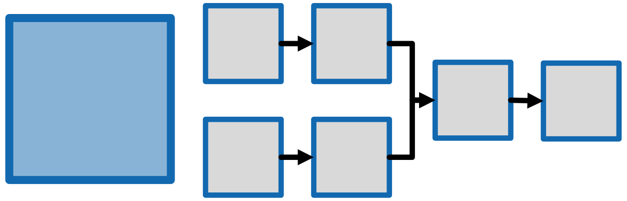
Coefficients α_i

Vector $\phi_i \longrightarrow \alpha_i = W\phi_i + b$



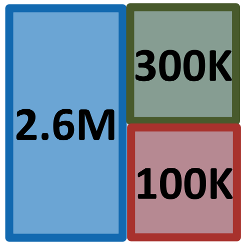
Parameter Mapping

Learn layer representations

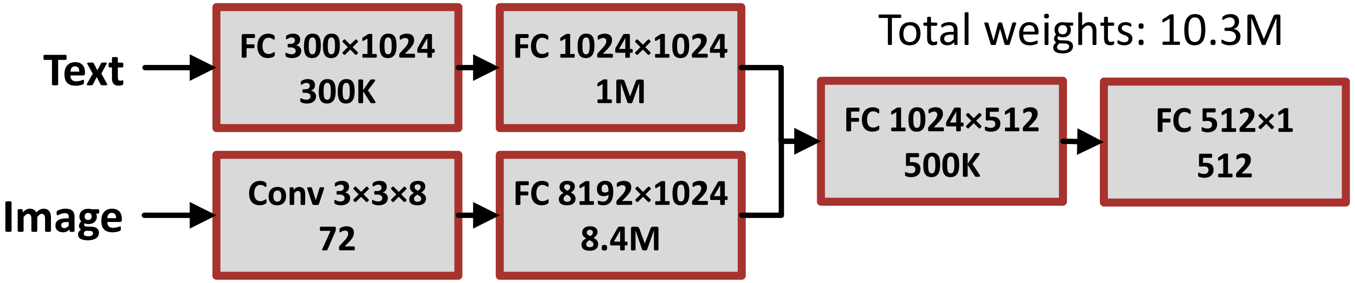


Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups

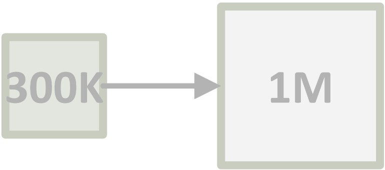


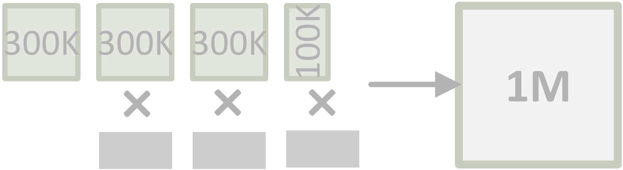
Total weights: 10.3M



Weight Generation

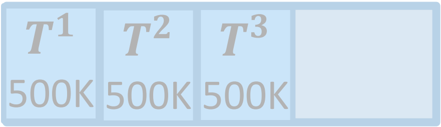
Upsample: Interpolate Mask





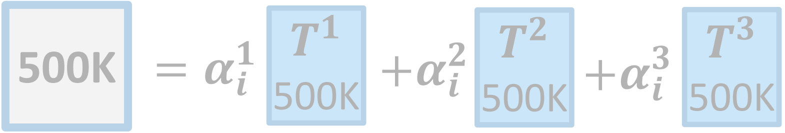
Downsample: WAvg [Savarese & Maire, 2019] Embedding

$K = 3$ templates



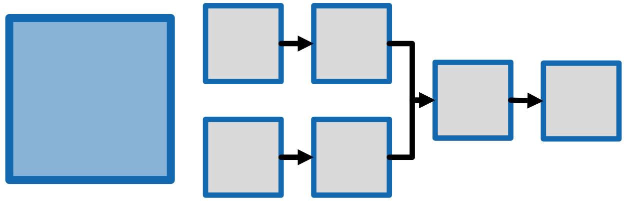
Coefficients α_i

Vector $\phi_i \longrightarrow \alpha_i = W\phi_i + b$



Parameter Mapping

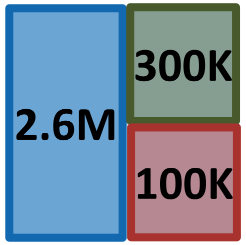
Learn layer representations

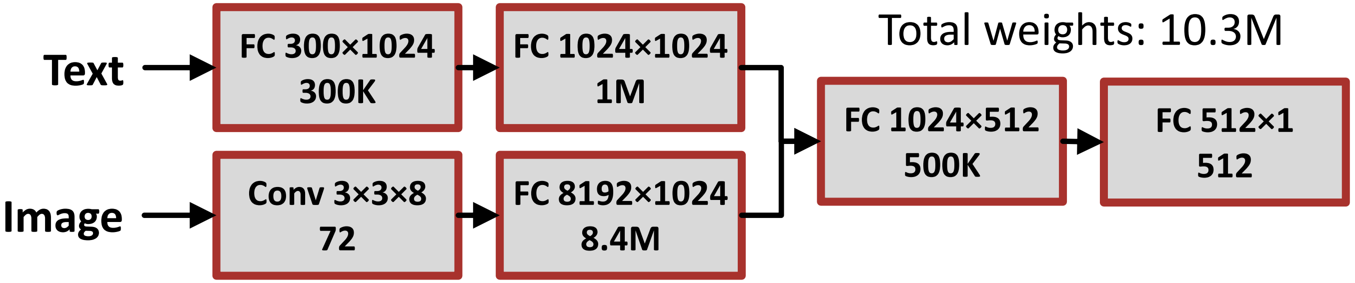


Short pretraining

Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups

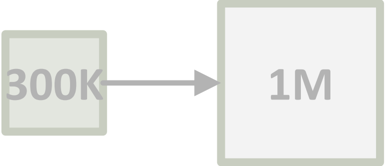




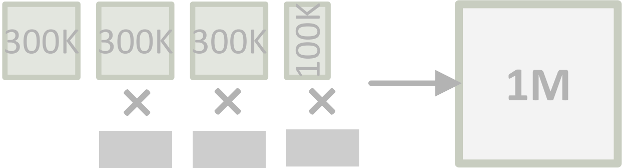
Total weights: 10.3M

Weight Generation

Upsample: Interpolate



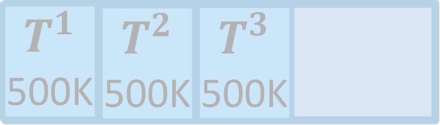
Mask



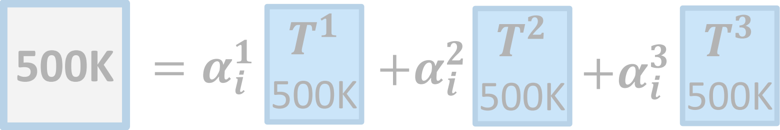
Downsample: WAvg [Savarese & Maire, 2019]

Embedding

$K = 3$ templates



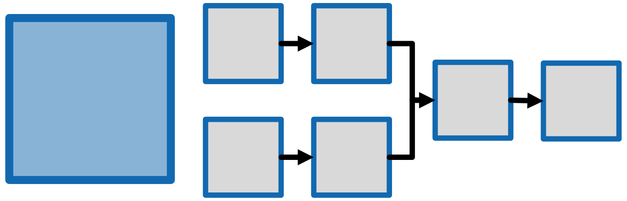
Coefficients α_i

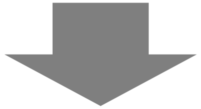


Vector $\phi_i \longrightarrow \alpha_i = W\phi_i + b$

Parameter Mapping

Learn layer representations





Short pretraining

Cluster layer representations

Performance: Question Answering

Performance: Question Answering

	Model	#Params	SQuAD v1.1	SQuAD v2.0
Base	BERT	108M	90.4 / 83.2	80.4 / 77.6
Large	BERT	334M	92.2 / 85.5	85.0 / 82.2

[Devlin et al., 2018; Lan et al., 2020]

Performance: Question Answering

	Model	#Params	SQuAD v1.1	SQuAD v2.0
Base	BERT	108M	90.4 / 83.2	80.4 / 77.6
	ALBERT	12M	89.2 / 82.1	79.8 / 76.9
Large	BERT	334M	92.2 / 85.5	85.0 / 82.2
	ALBERT	18M	90.5 / 83.7	82.1 / 79.3

[Devlin et al., 2018; Lan et al., 2020]

Performance: Question Answering

Model		#Params	SQuAD v1.1	SQuAD v2.0
Base	BERT	108M	90.4 / 83.2	80.4 / 77.6
	ALBERT	12M	89.2 / 82.1	79.8 / 76.9
	SSN-BERT	12M	90.1 / 83.0	80.7 / 78.0
Large	BERT	334M	92.2 / 85.5	85.0 / 82.2
	ALBERT	18M	90.5 / 83.7	82.1 / 79.3
	SSN-BERT	18M	91.1 / 84.4	83.0 / 80.1

[Devlin et al., 2018; Lan et al., 2020]

Performance: Question Answering

Model		#Params	SQuAD v1.1	SQuAD v2.0
Base	BERT	108M	90.4 / 83.2	80.4 / 77.6
	ALBERT	12M	89.2 / 82.1	79.8 / 76.9
	SSN-BERT	12M	90.1 / 83.0	80.7 / 78.0
Large	BERT	334M	92.2 / 85.5	85.0 / 82.2
	ALBERT	18M	90.5 / 83.7	82.1 / 79.3
	SSN-BERT	18M	91.1 / 84.4	83.0 / 80.1

1.4× faster training speed compared to BERT-Large on 128 V100 GPUs

[Devlin et al., 2018; Lan et al., 2020]

Performance: Question Answering

Model		#Params	SQuAD v1.1	SQuAD v2.0
Base	BERT	108M	90.4 / 83.2	80.4 / 77.6
	ALBERT	12M	89.2 / 82.1	79.8 / 76.9
	SSN-BERT	12M	90.1 / 83.0	80.7 / 78.0
Large	BERT	334M	92.2 / 85.5	85.0 / 82.2
	ALBERT	18M	90.5 / 83.7	82.1 / 79.3
	SSN-BERT	18M	91.1 / 84.4	83.0 / 80.1

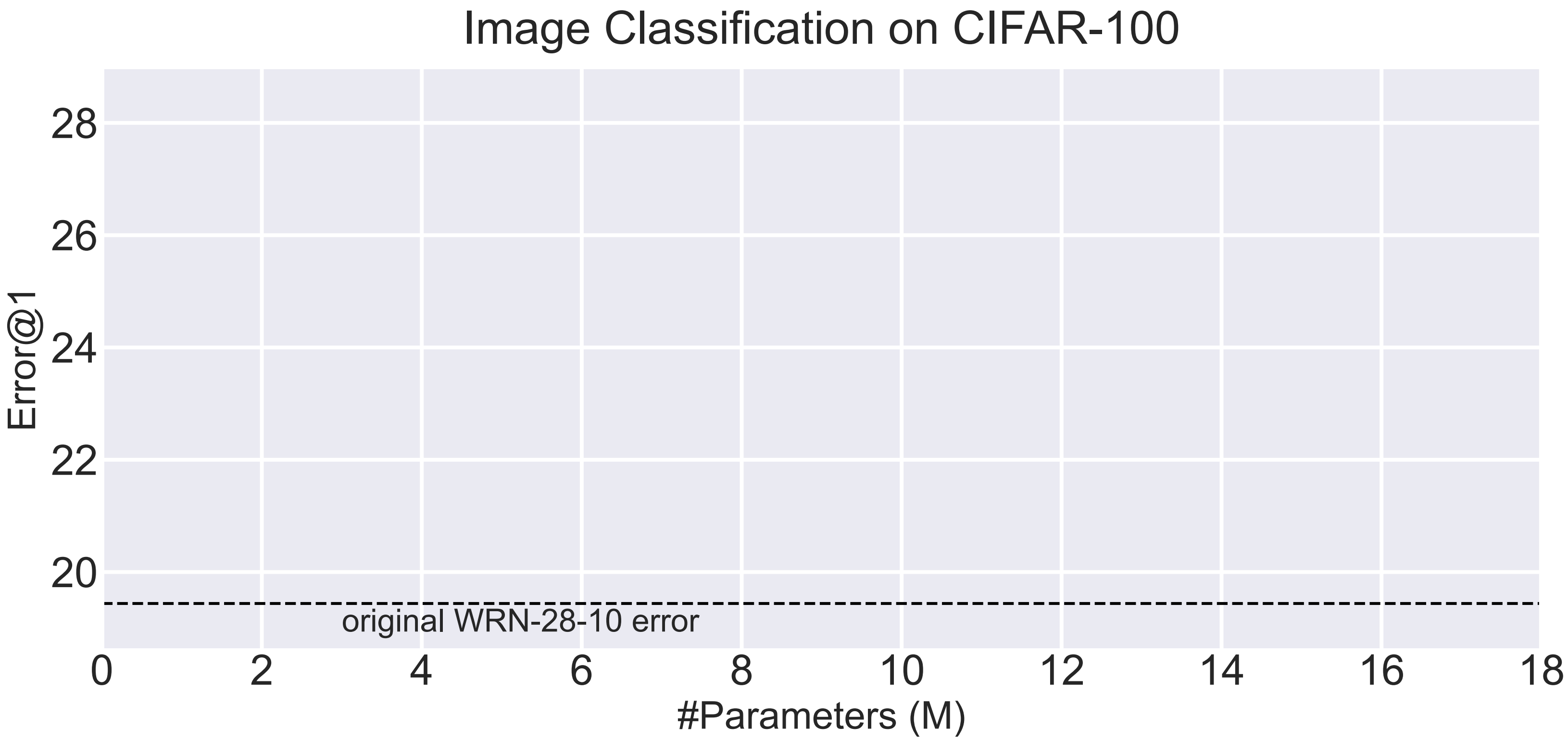
1.4× fasless memory usage compared to BERT-Large on 128 V100 GPUs

1 3   3 3 3 less memory usage

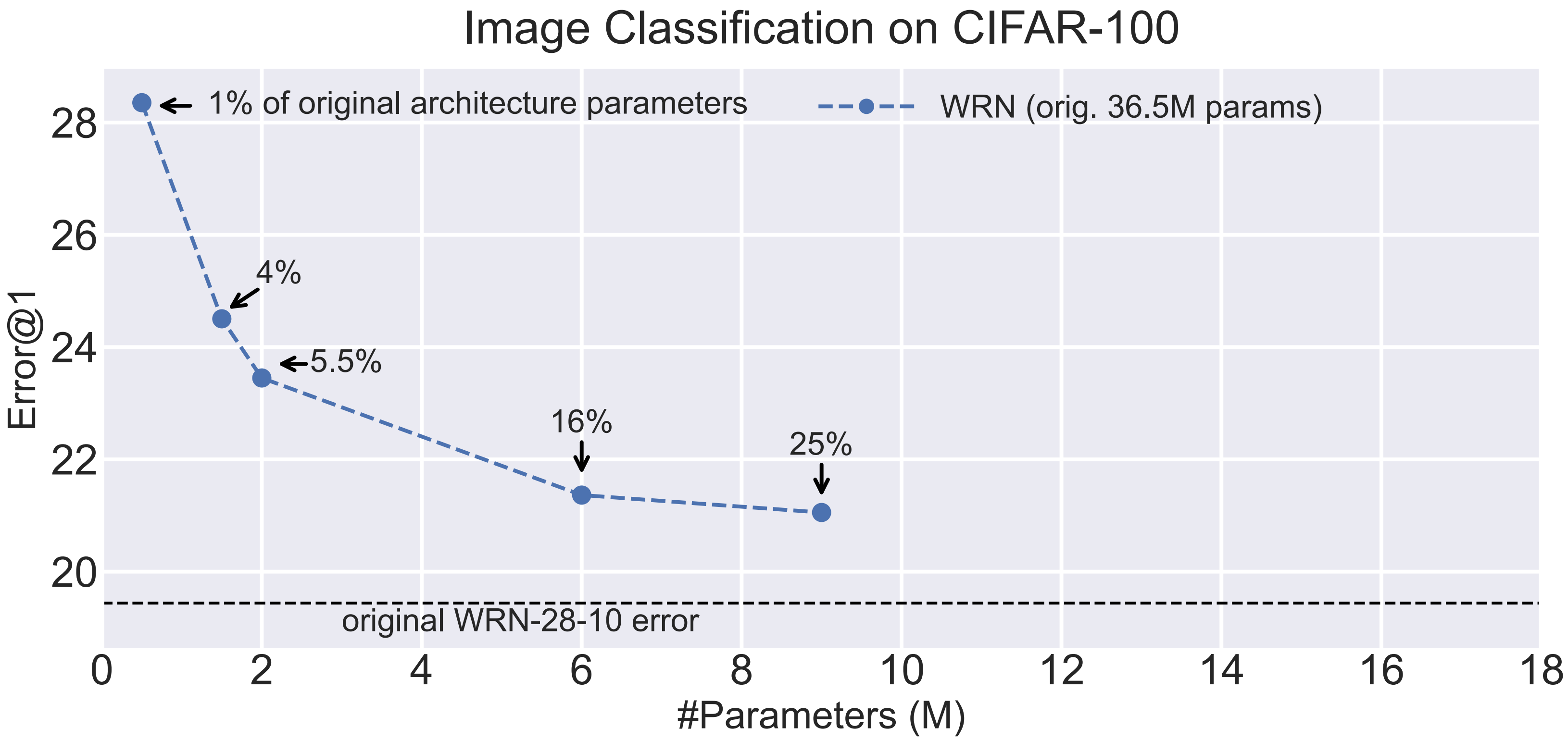
[Devlin et al., 2018; Lan et al., 2020]

Performance: Image Classification

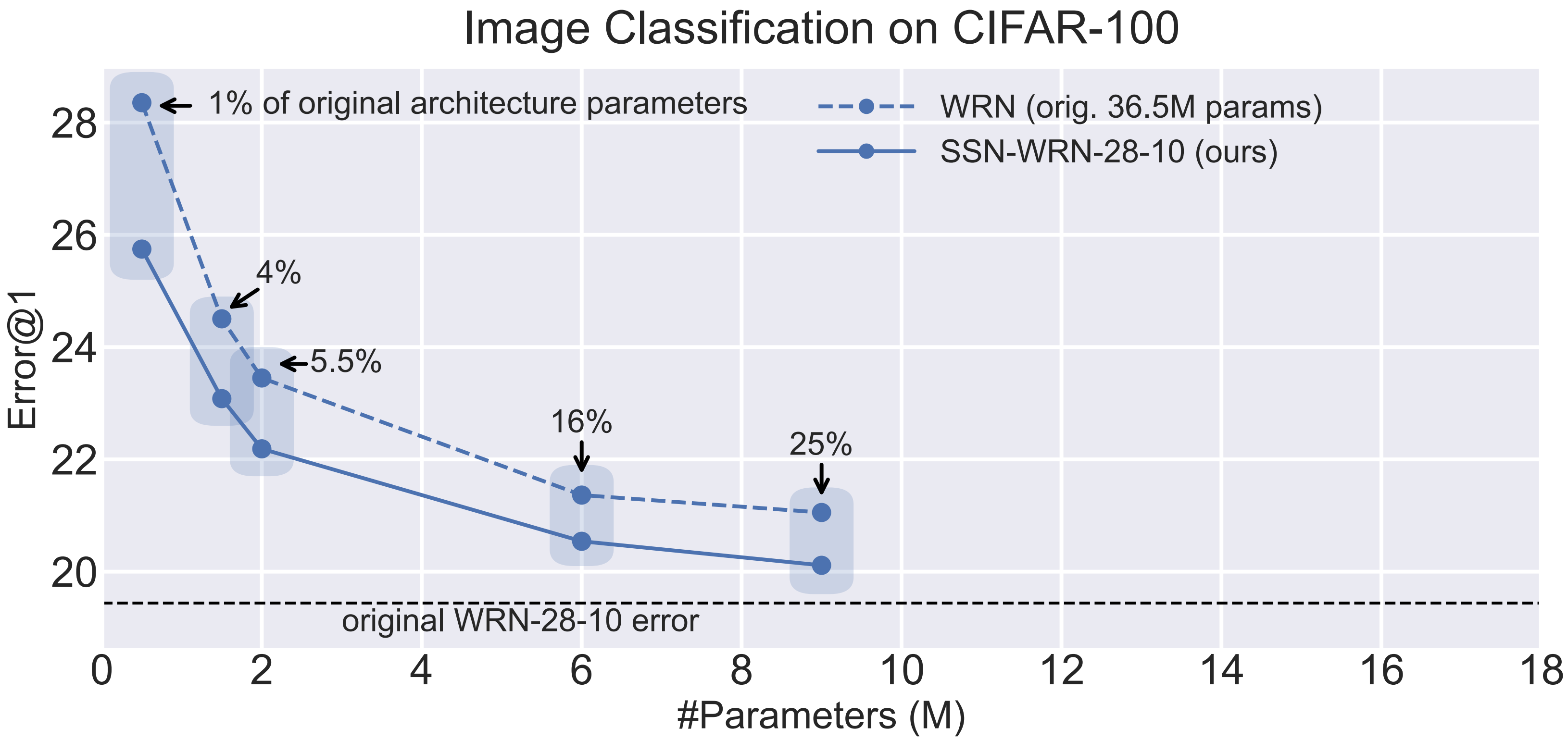
Performance: Image Classification



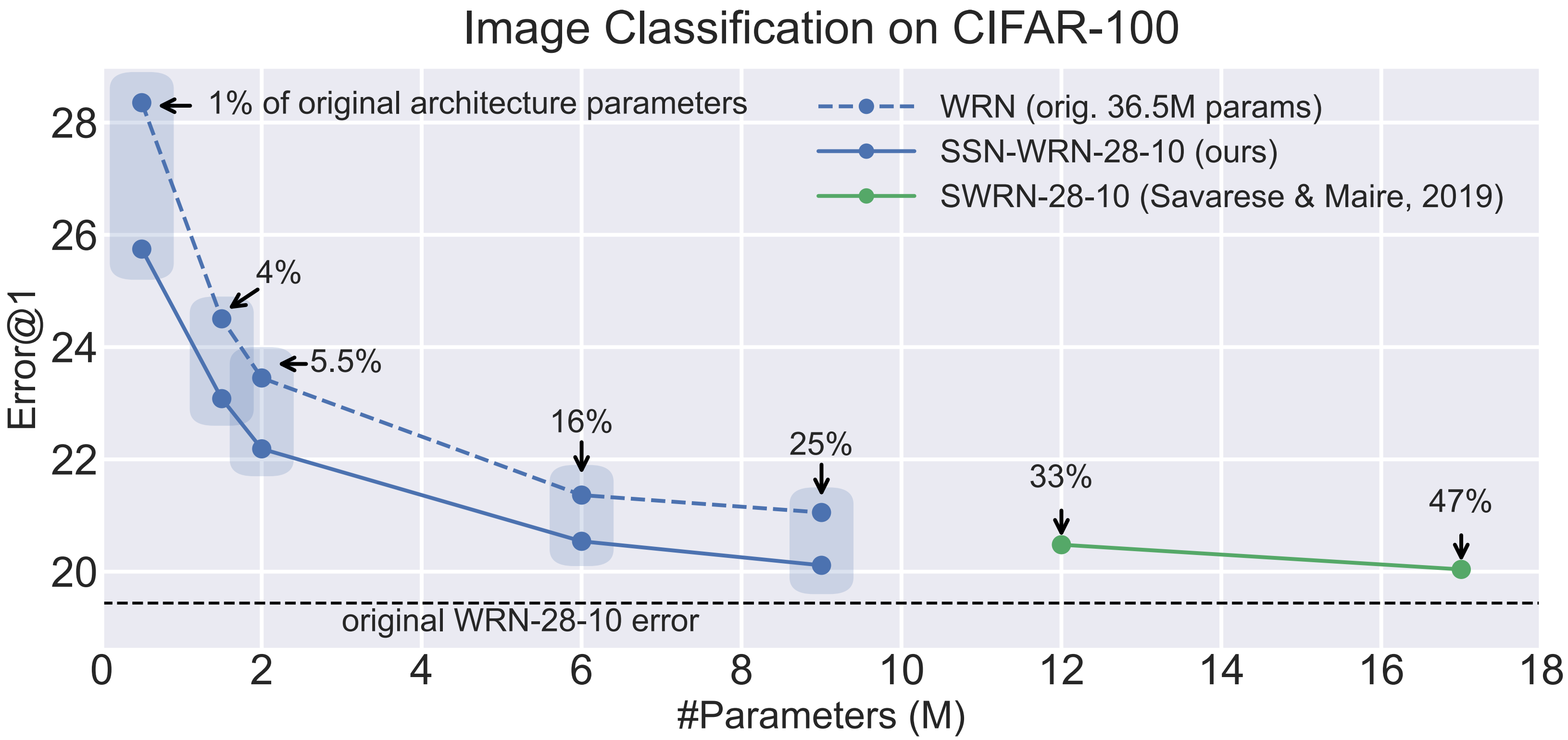
Performance: Image Classification



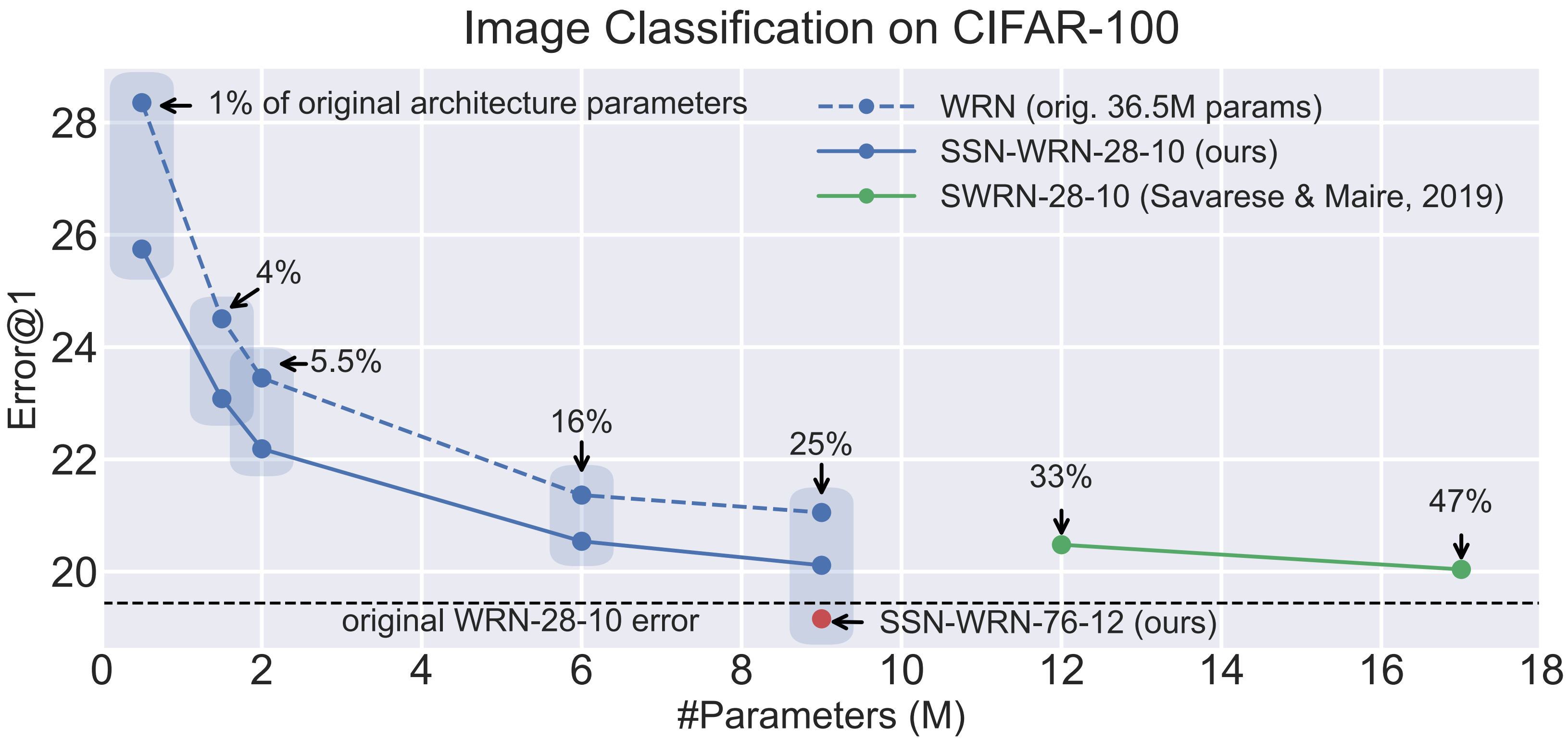
Performance: Image Classification



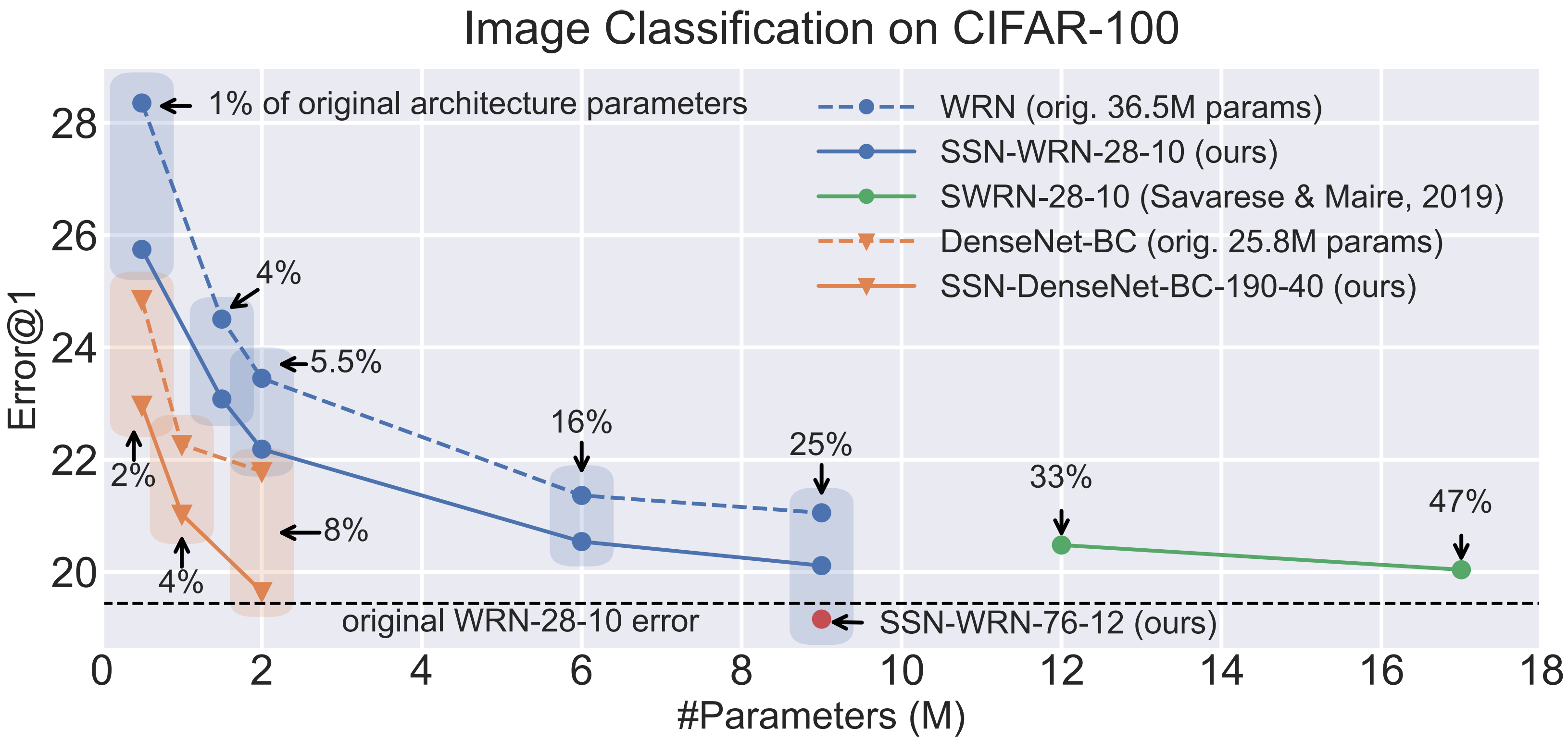
Performance: Image Classification



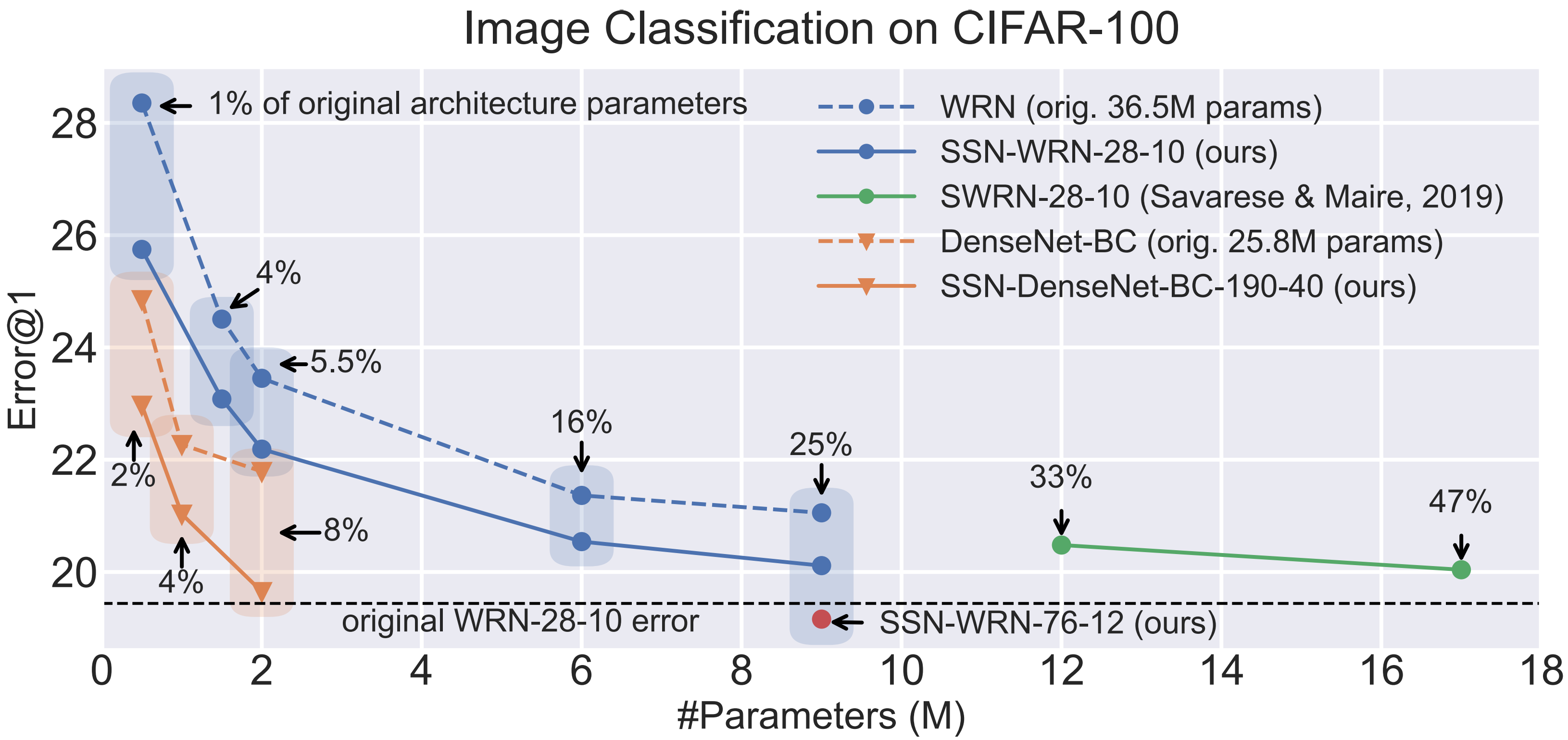
Performance: Image Classification



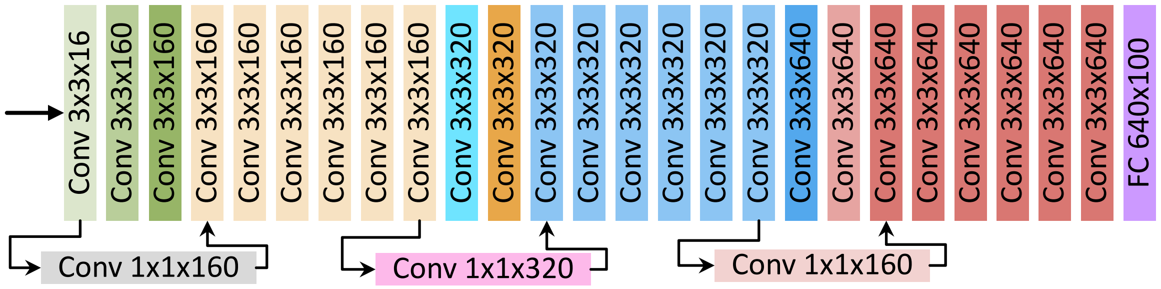
Performance: Image Classification



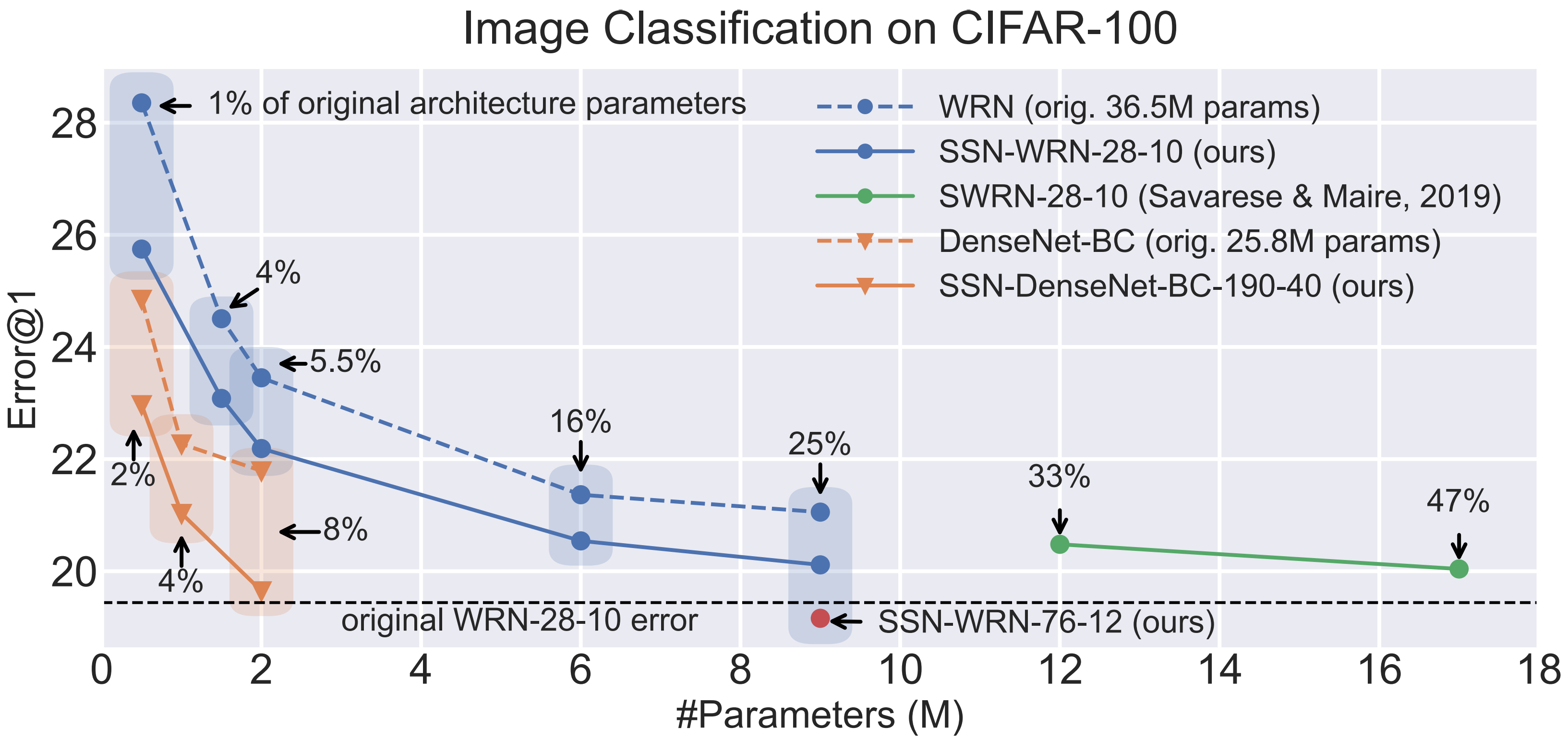
Performance: Image Classification



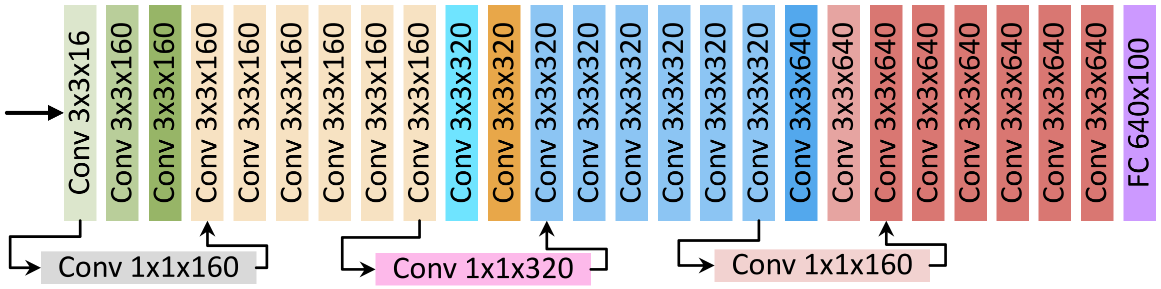
Manual (Savarese & Maire, 2019)



Performance: Image Classification

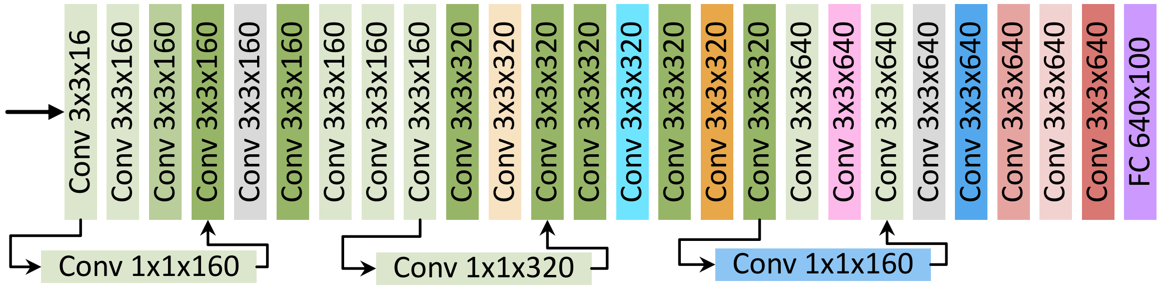


Manual (Savarese & Maire, 2019)



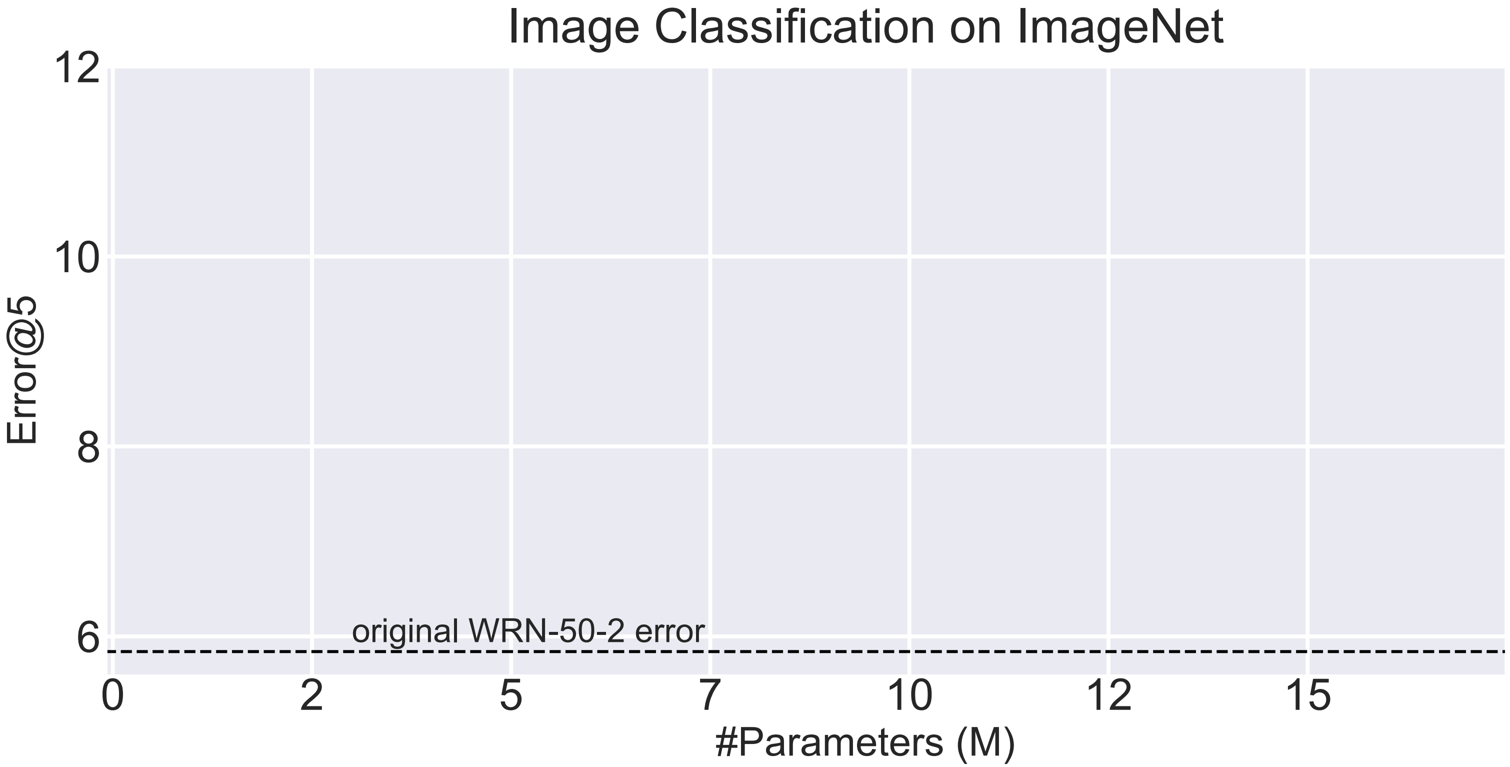
Conv 3x3x16
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
FC 640x100
Conv 1x1x160
Conv 1x1x320
Conv 1x1x160

Learned (ours)

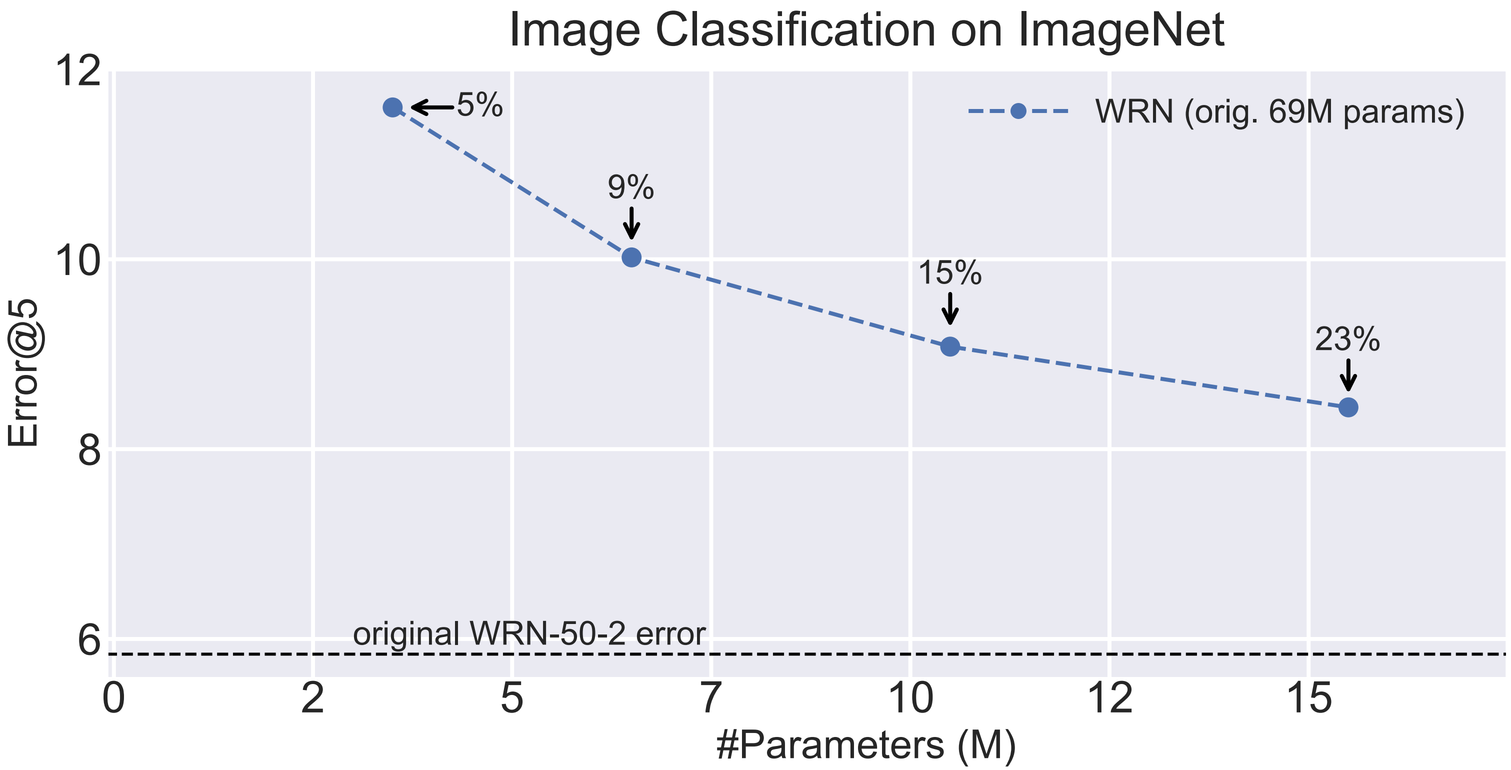


Conv 3x3x16
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x160
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x320
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
Conv 3x3x640
FC 640x100
Conv 1x1x160
Conv 1x1x320
Conv 1x1x160

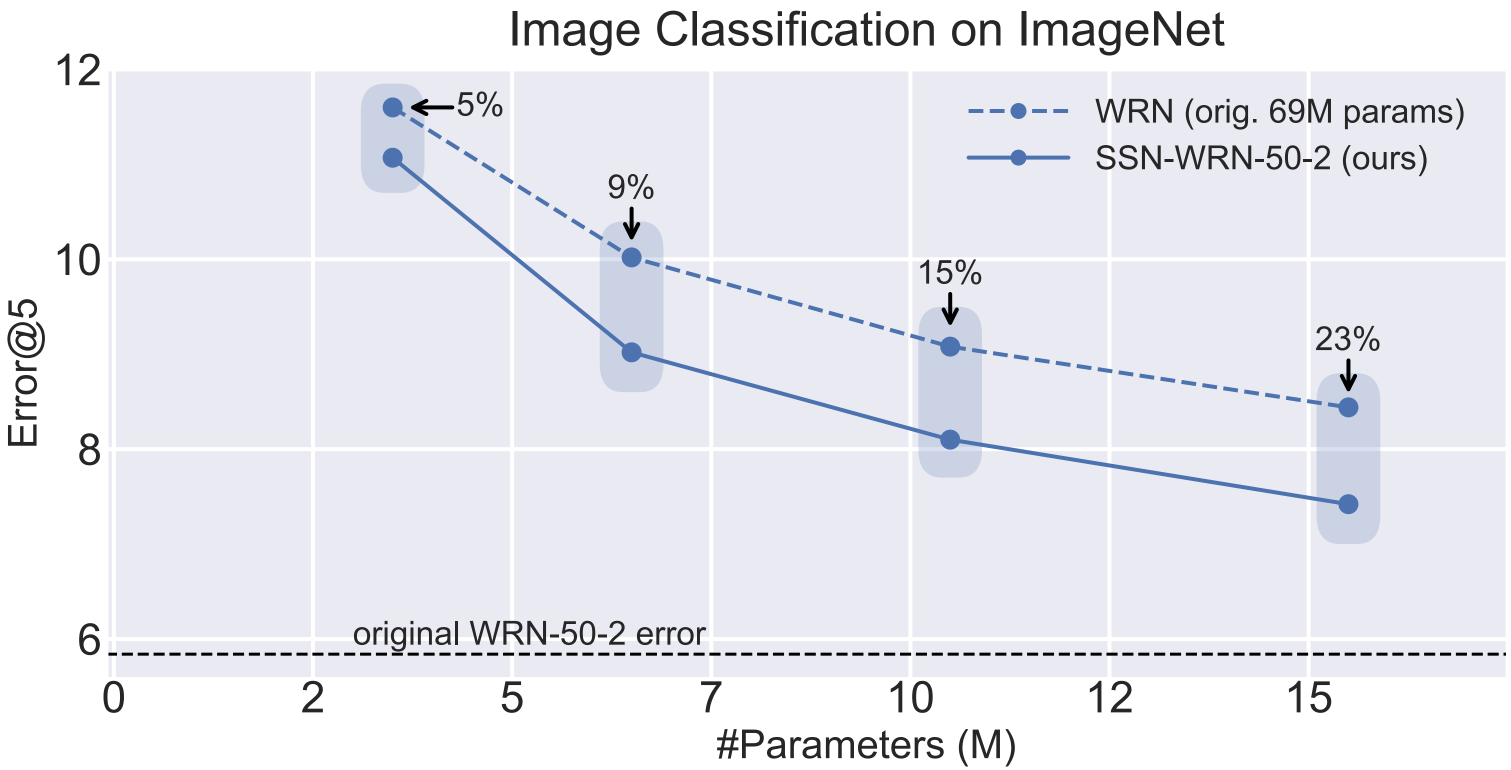
Performance: Image Classification



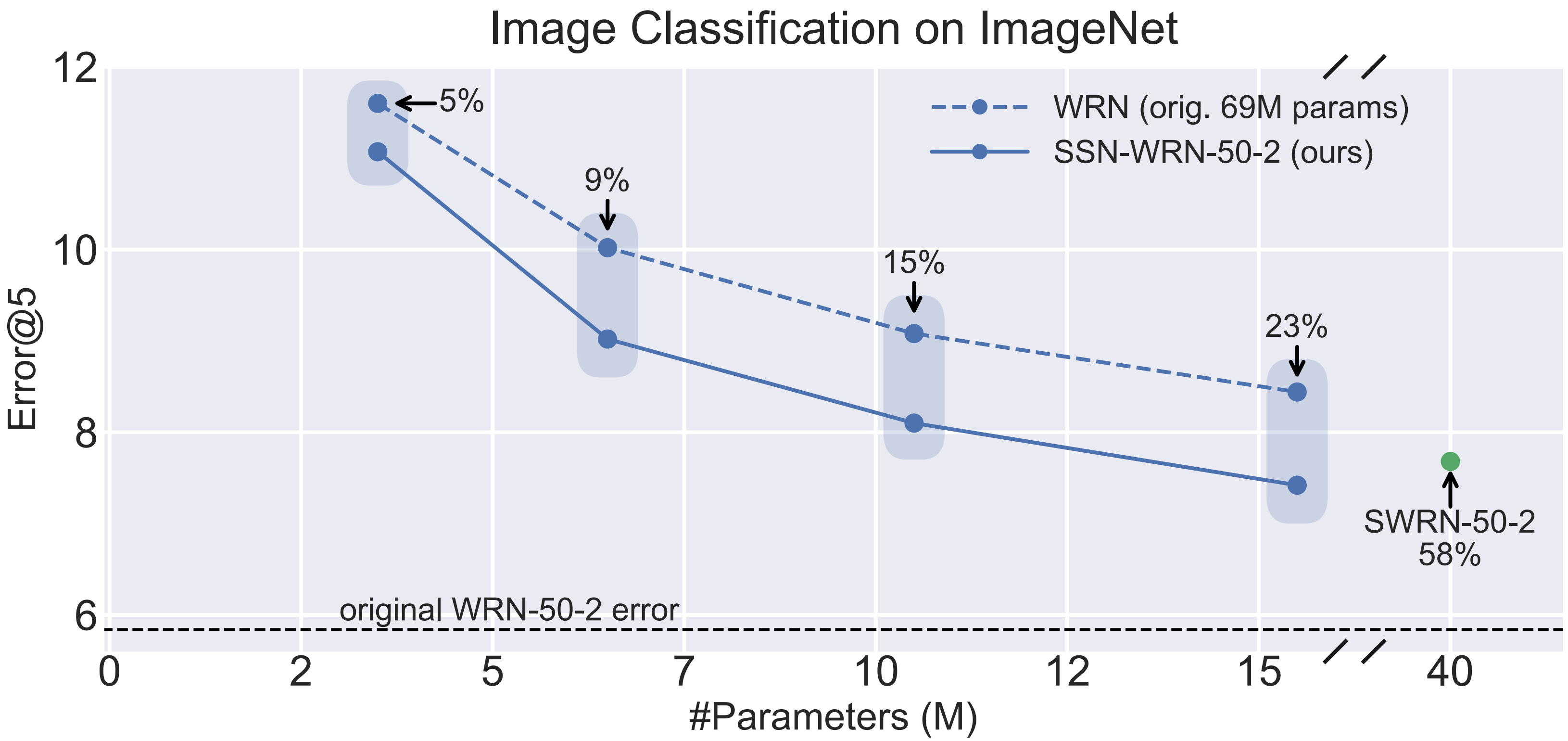
Performance: Image Classification



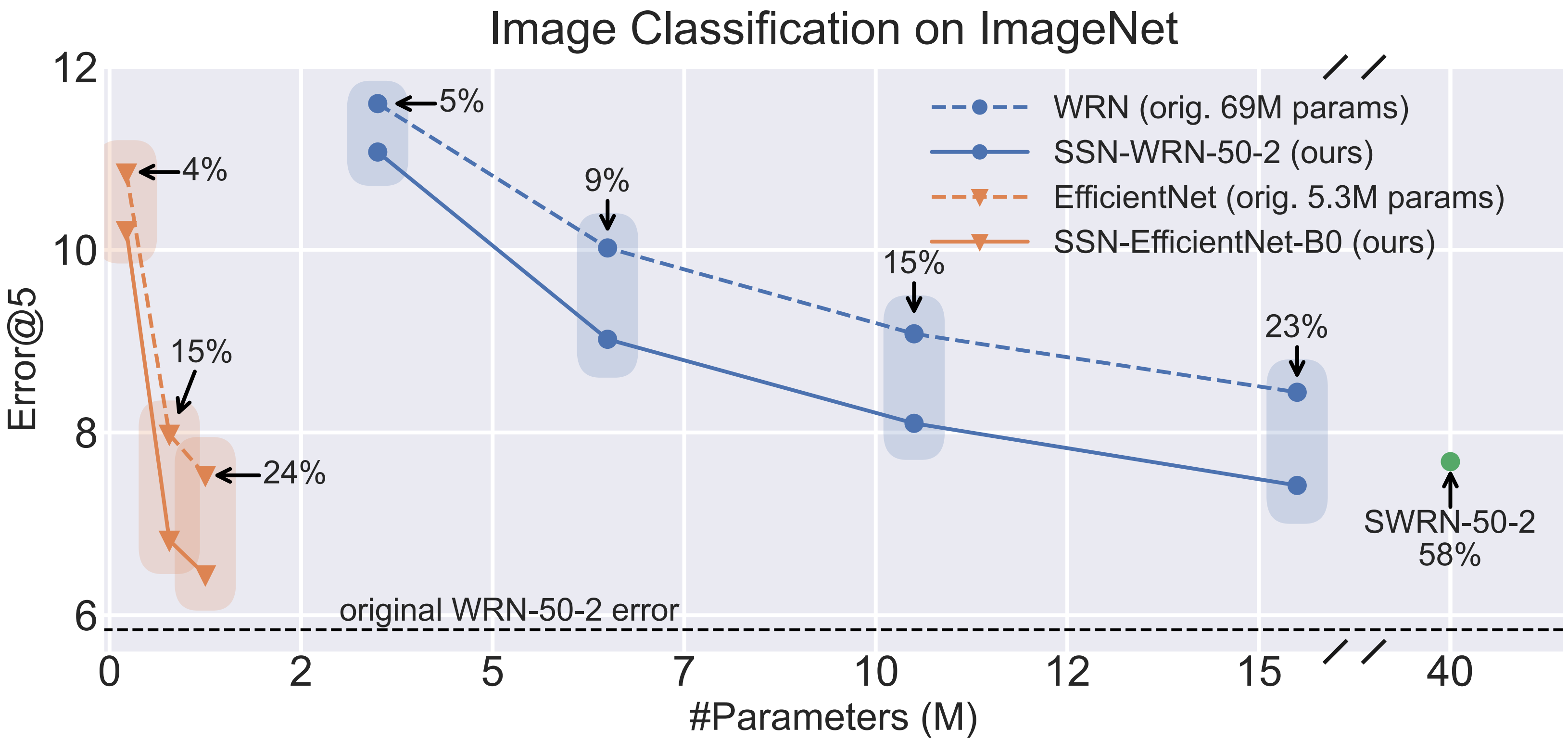
Performance: Image Classification



Performance: Image Classification



Performance: Image Classification



Pruning

[Lin et al., 2020]

Pruning

Pruning

CIFAR-10

Method	Error@1		% original inference flop	% original params
	ResNet-56			
Base	6.74		100.0	100.0

[Lin et al., 2020]

Pruning

Pruning

CIFAR-10

Method	Error@1		% original inference flop	% original params
	ResNet-56	HB(4×)-SSN		
Base	6.74	5.21	100.0	100.0

[Lin et al., 2020]

Pruning

Pruning

CIFAR-10

Method	Error@1		% original inference flop	% original params
	ResNet-56	HB(4×)-SSN		
Base	6.74	5.21	100.0	100.0
HRank	6.48	5.86	70.7	83.2
HRank	10.75	9.94	20.3	28.4

[Lin et al., 2020]

Pruning

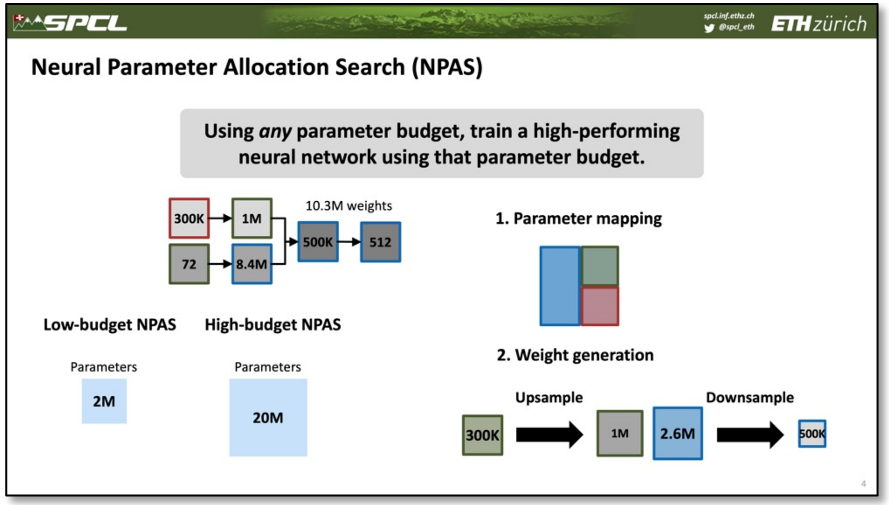
Pruning
CIFAR-10

Method	Error@1		% original inference flop	% original params
	ResNet-56	HB(4×)-SSN		
Base	6.74	5.21	100.0	100.0
HRank	6.48	5.86	70.7	83.2
HRank	10.75	9.94	20.3	28.4
LB-SSN	9.26	—	100.6	12.2

[Lin et al., 2020]

Conclusions

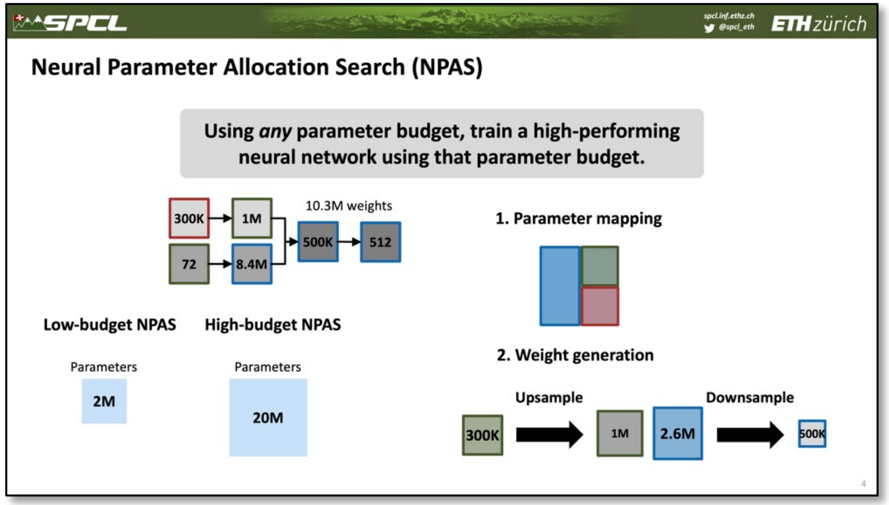
Conclusions



Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.

Conclusions



Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.

300K → 1M
72 → 8.4M
10.3M weights
500K → 512

Low-budget NPAS

Parameters

2M

High-budget NPAS

Parameters

20M

1. Parameter mapping

2. Weight generation

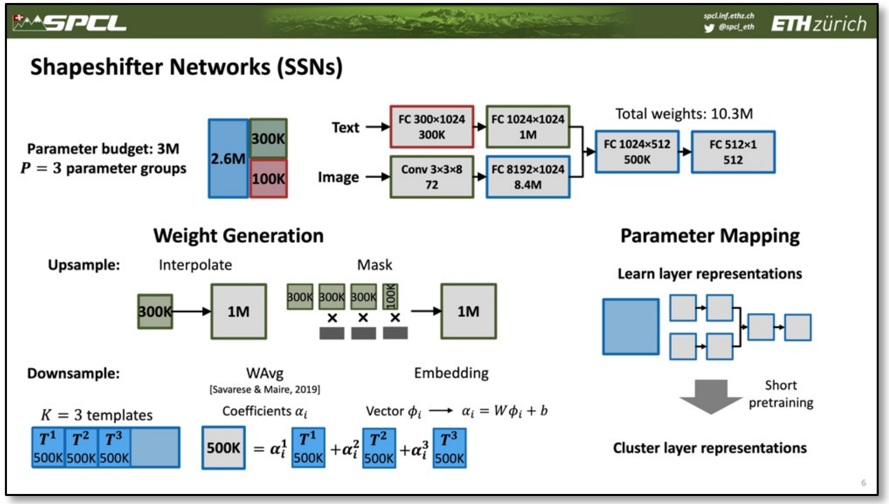
Upsample

300K → 1M

Downsample

2.6M → 500K

4



Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups

2.6M
300K
100K

Text → FC 300×1024 300K → FC 1024×1024 1M

Image → Conv 3×3×8 72 → FC 8192×1024 8.4M

Total weights: 10.3M

FC 1024×512 500K → FC 512×1 512

Weight Generation

Upsample:

Interpolate

300K → 1M

Mask

300K 300K 300K 100K × × × → 1M

Downsample:

$K = 3$ templates

T^1 500K T^2 500K T^3 500K

WAvg
[Savarese & Maire, 2019]

Coefficients α_i

Embedding

Vector $\phi_i \longrightarrow \alpha_i = W\phi_i + b$

$$500\text{K} = \alpha_i^1 \, T^1_{500\text{K}} + \alpha_i^2 \, T^2_{500\text{K}} + \alpha_i^3 \, T^3_{500\text{K}}$$

Parameter Mapping

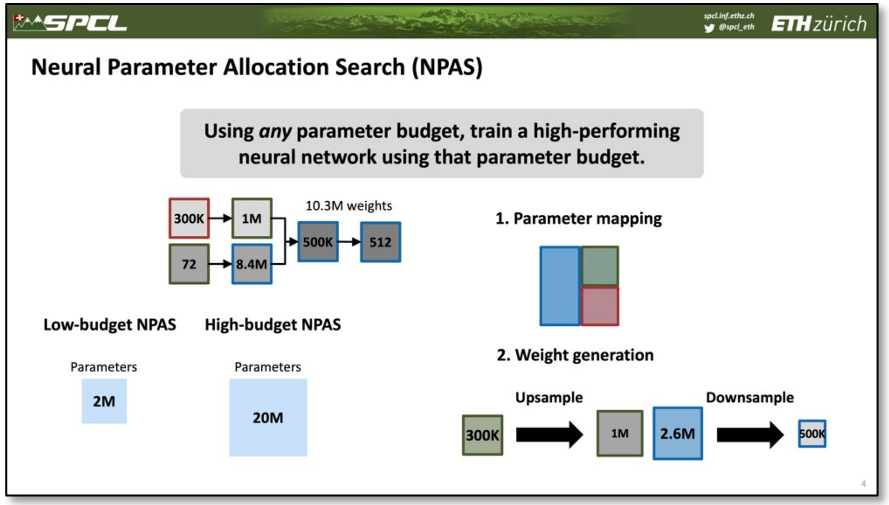
Learn layer representations

Short pretraining

Cluster layer representations

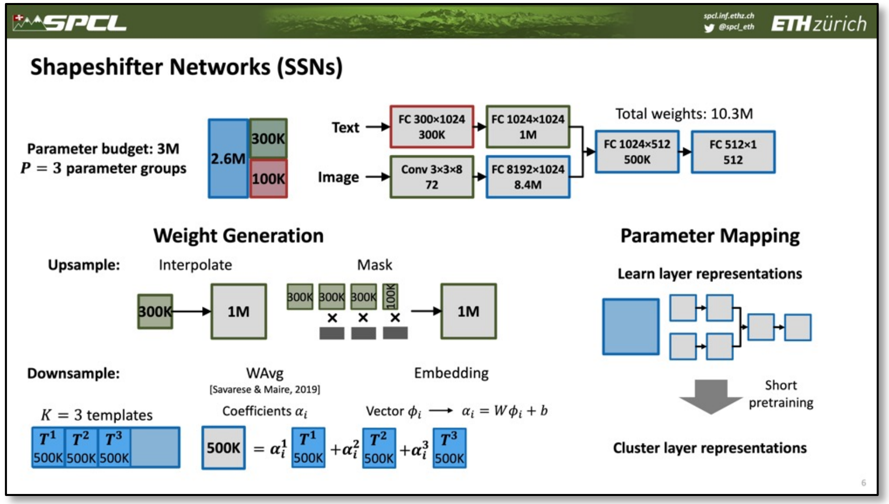
6

Conclusions



Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups

Weight Generation

Upsample: Interpolate, Mask

Downsample: WAvg [Savarese & Maire, 2019], Embedding

$K = 3$ templates

Coefficients α_i

Vector $\phi_i \longrightarrow \alpha_i = W\phi_i + b$

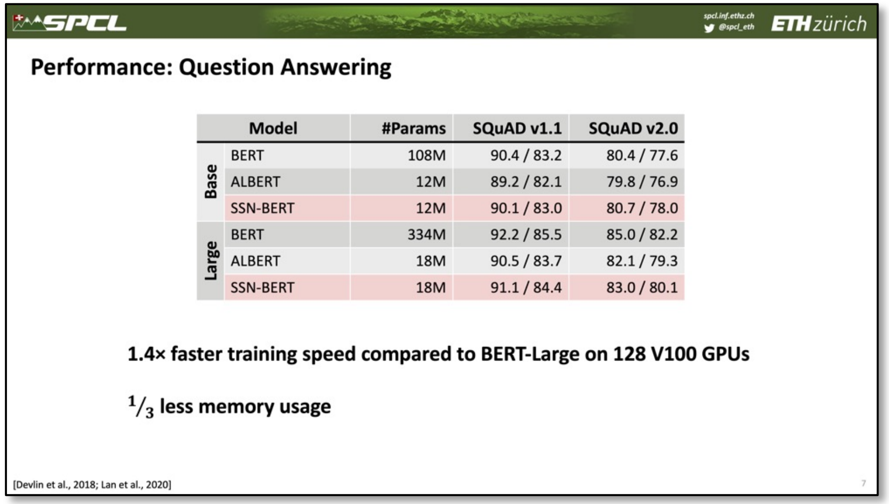
$500K = \alpha_i^1 T^1 + \alpha_i^2 T^2 + \alpha_i^3 T^3$

Parameter Mapping

Learn layer representations

Short pretraining

Cluster layer representations



Performance: Question Answering

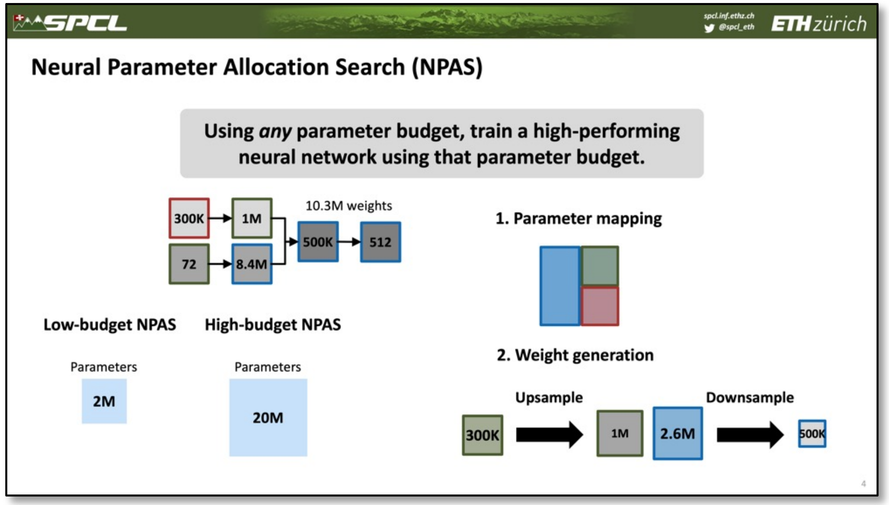
	Model	#Params	SQuAD v1.1	SQuAD v2.0
Base	BERT	108M	90.4 / 83.2	80.4 / 77.6
	ALBERT	12M	89.2 / 82.1	79.8 / 76.9
	SSN-BERT	12M	90.1 / 83.0	80.7 / 78.0
Large	BERT	334M	92.2 / 85.5	85.0 / 82.2
	ALBERT	18M	90.5 / 83.7	82.1 / 79.3
	SSN-BERT	18M	91.1 / 84.4	83.0 / 80.1

1.4× faster training speed compared to BERT-Large on 128 V100 GPUs

$^1/_3$ less memory usage

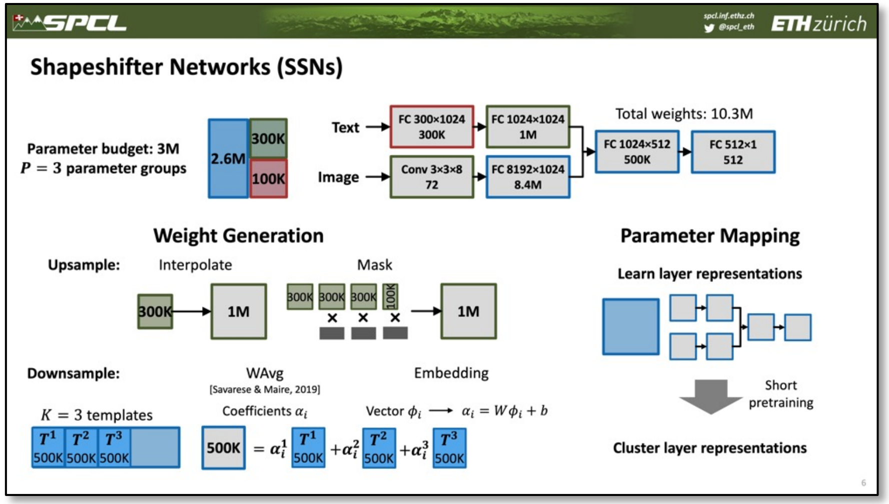
[Devlin et al., 2018; Lan et al., 2020]

Conclusions

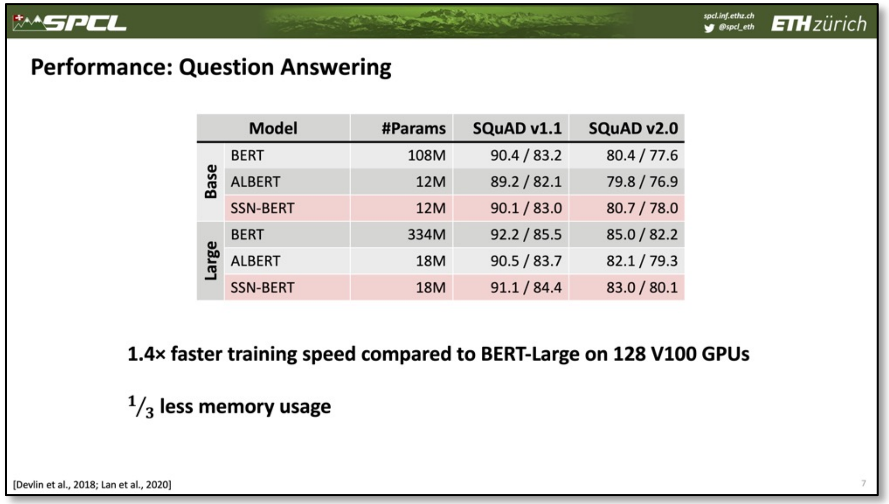


Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



Shapeshifter Networks (SSNs)



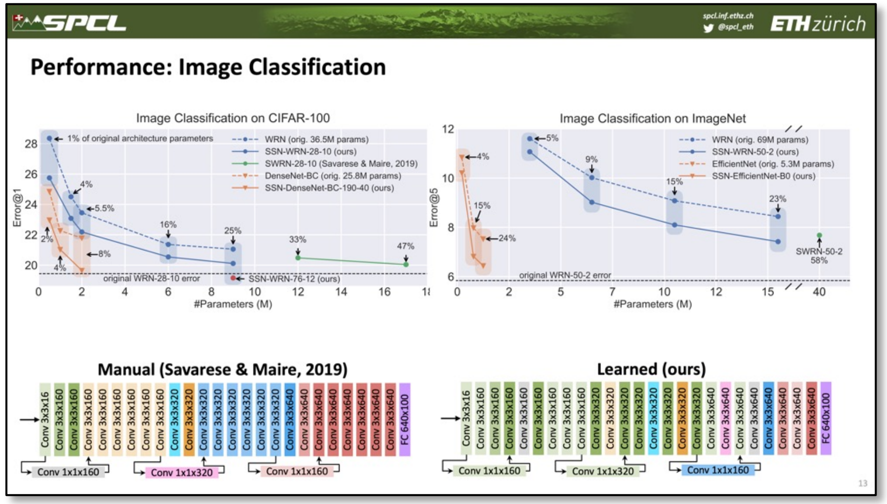
Performance: Question Answering

	Model	#Params	SQuAD v1.1	SQuAD v2.0
Base	BERT	108M	90.4 / 83.2	80.4 / 77.6
	ALBERT	12M	89.2 / 82.1	79.8 / 76.9
	SSN-BERT	12M	90.1 / 83.0	80.7 / 78.0
Large	BERT	334M	92.2 / 85.5	85.0 / 82.2
	ALBERT	18M	90.5 / 83.7	82.1 / 79.3
	SSN-BERT	18M	91.1 / 84.4	83.0 / 80.1

1.4× faster training speed compared to BERT-Large on 128 V100 GPUs

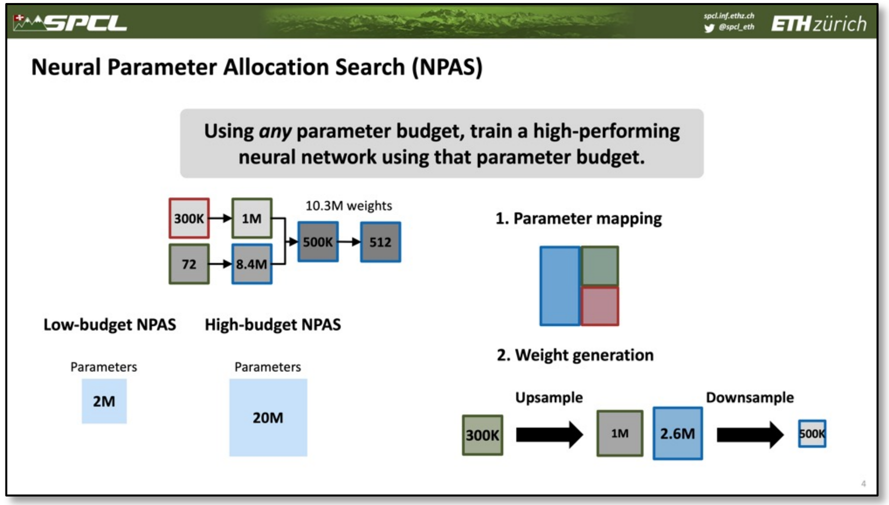
$^1/_3$ less memory usage

[Devlin et al., 2018; Lan et al., 2020]



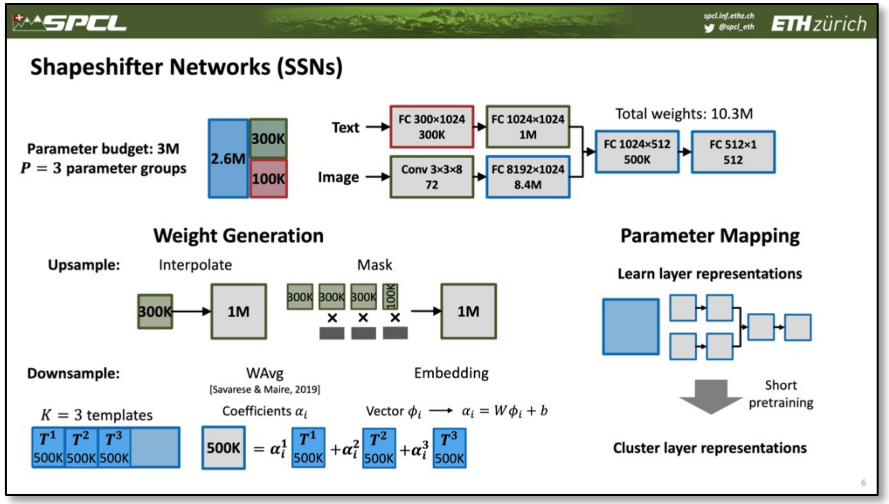
Performance: Image Classification

Conclusions



Neural Parameter Allocation Search (NPAS)

Using *any* parameter budget, train a high-performing neural network using that parameter budget.



Shapeshifter Networks (SSNs)

Parameter budget: 3M
$P = 3$ parameter groups

Weight Generation

Upsample: Interpolate / Mask

Downsample: WAvg [Savarese & Maire, 2019] Coefficients α_i / Embedding Vector $\phi_i \longrightarrow \alpha_i = W\phi_i + b$

$K = 3$ templates

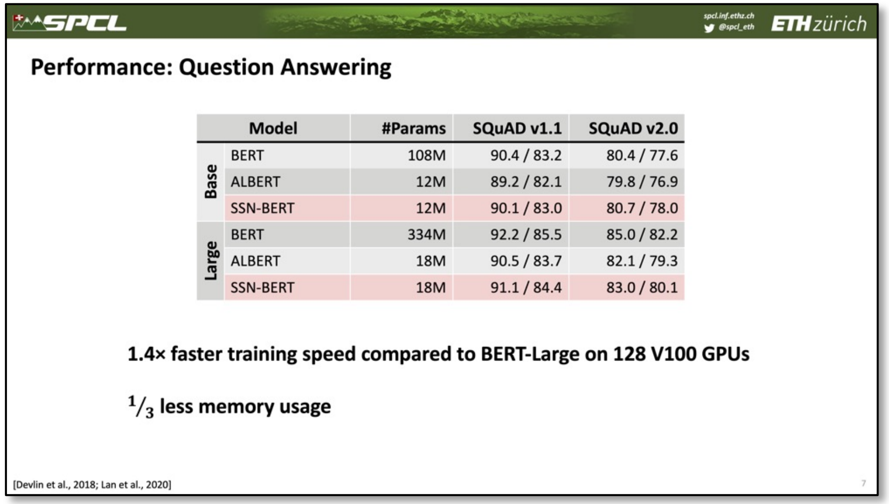
$500K = \alpha_i^1 T^1 + \alpha_i^2 T^2 + \alpha_i^3 T^3$

Parameter Mapping

Learn layer representations

Short pretraining

Cluster layer representations



Performance: Question Answering

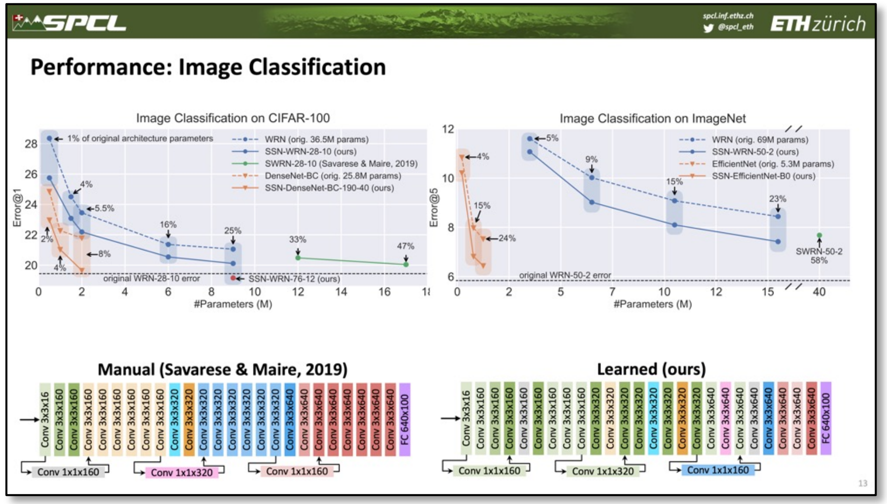
	Model	#Params	SQuAD v1.1	SQuAD v2.0
Base	BERT	108M	90.4 / 83.2	80.4 / 77.6
	ALBERT	12M	89.2 / 82.1	79.8 / 76.9
	SSN-BERT	12M	90.1 / 83.0	80.7 / 78.0
Large	BERT	334M	92.2 / 85.5	85.0 / 82.2
	ALBERT	18M	90.5 / 83.7	82.1 / 79.3
	SSN-BERT	18M	91.1 / 84.4	83.0 / 80.1

1.4× faster training speed compared to BERT-Large on 128 V100 GPUs

$^1/_3$ less memory usage

[Devlin et al., 2018; Lan et al., 2020]

7



Performance: Image Classification

https://github.com/BryanPlummer/SSN