Network Compression and Architecture Search in Deep Learning

Haoli Bai

Supervisors:

Prof. Michael Rung-Tsong Lyu

Prof. Irwin Kuo-Chin King

Committee Members:

Prof. Laiwan Chan

Prof. Andrej Bogdanov

Prof. Hsuan-Tien Lin

2021-09-01

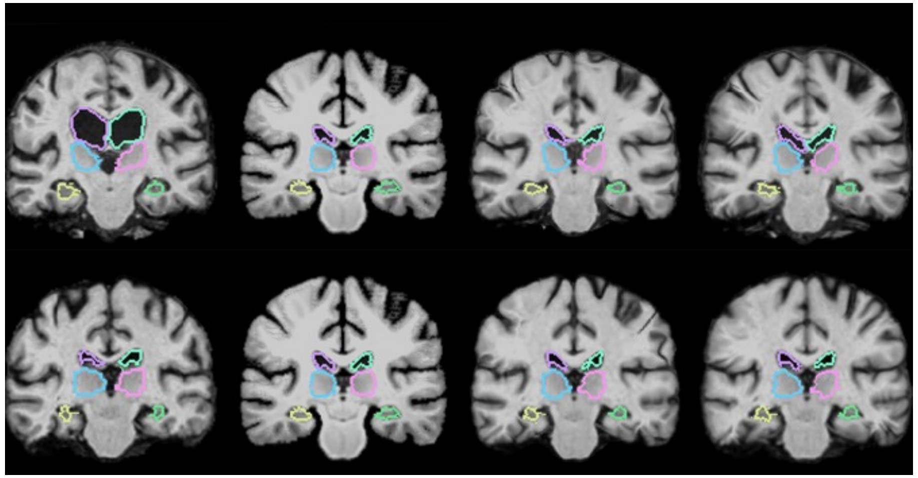
香港中文大學
The Chinese University of Hong Kong

Real-time AI Services

a) Object Detection

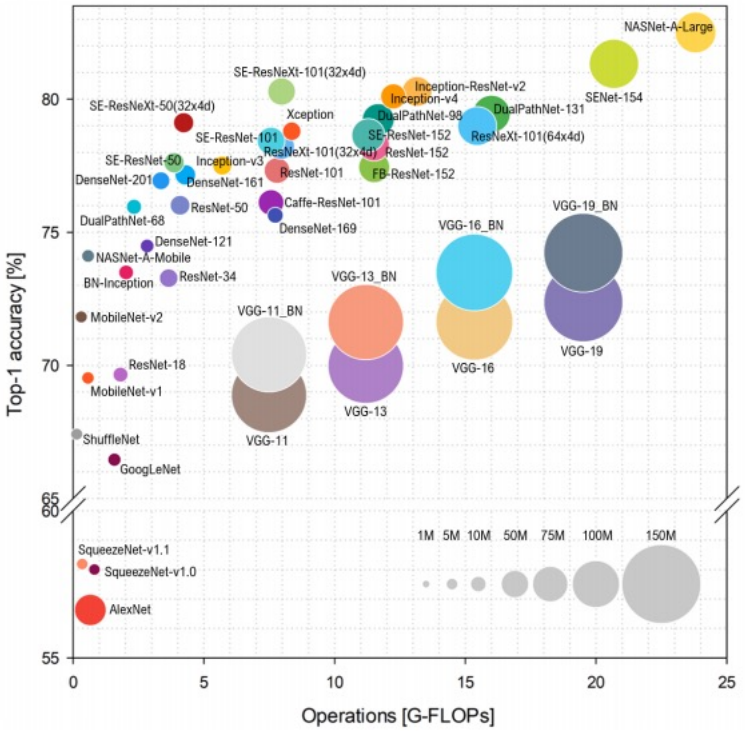
b) Machine Translation

c) Speech Recognition



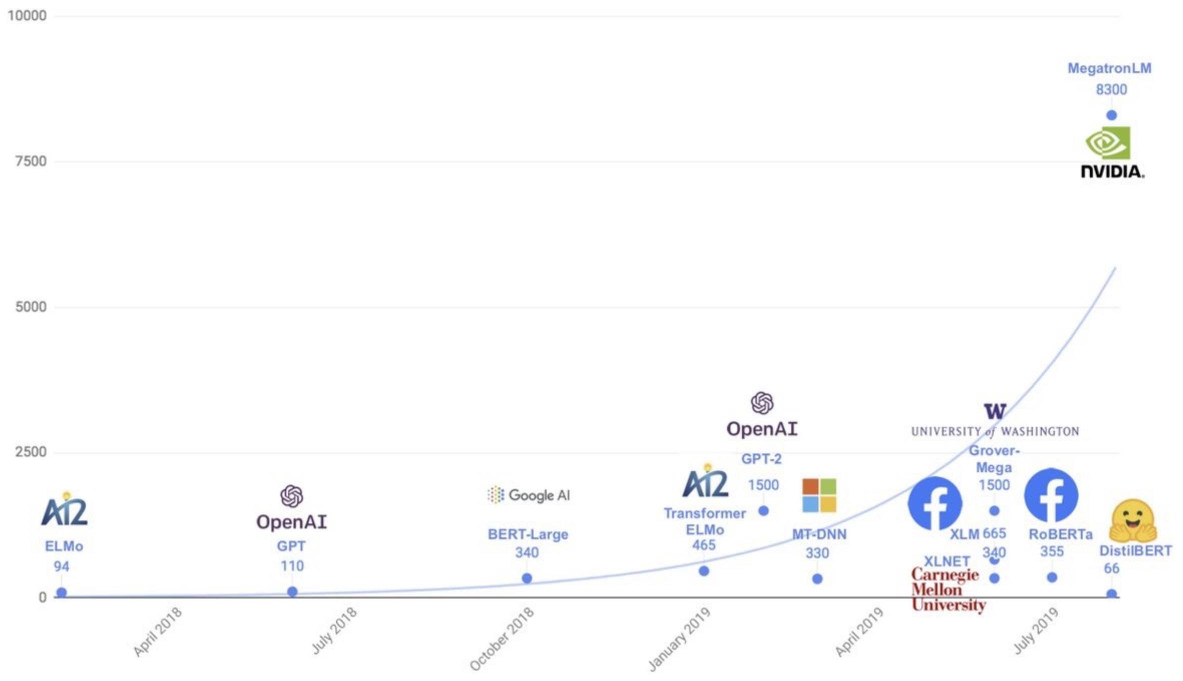
d) Tumor Detection

The Increasing Model Size



a) Computer Vision Models

(Bianco et.al., 2018)



b) Natural Language Processing Models

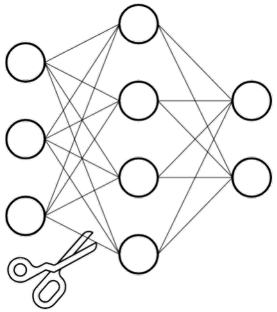
(Sanh et.al., 2020)

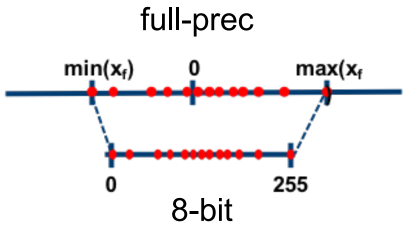
- Efficient deep learning by **network compression** and **neural architecture search**

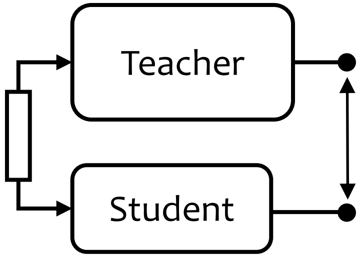
Overview: Network Compression

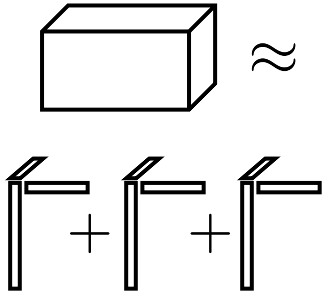
- Common methods

Pruning	Quantization	Knowledge distillation	Tensor factorization	…









Pruning
- Unstructured pruning (Zhu et.al, 2017)
- Structured pruning (He et.al., 2017)

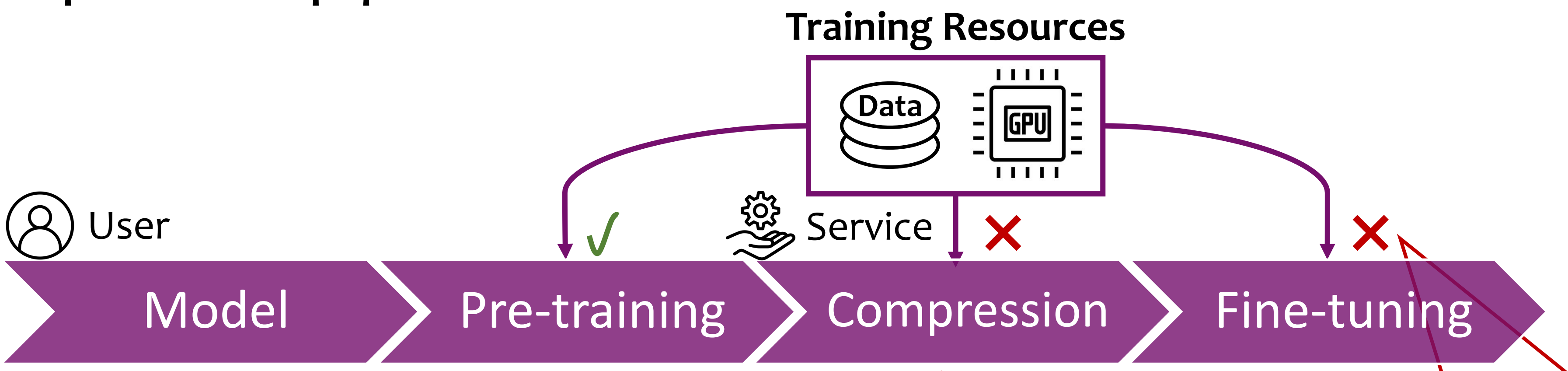
Quantization
- Multi-bit quant (He et.al., 2017)
- Ternarization (Li et.al., 2016)
- Binarization (Courbariaux et.al, 2016)

Knowledge distillation
- Logit (Hinton et.al., 2015)
- Hidden representation (Romero et.al, 2015)

Tensor factorization
- Canonical Polyadic (Lebedev et.al., 2015)
- Tucker (Kim et.al., 2016)

Overview: Network Compression

- Compression pipeline



Challenge 2: sharp performance drop

Caused by extreme compression such as:

- Pruning: higher sparsity
- Quantization: lower bit-width (e.g., binarization)

Challenge 1: access of training resources

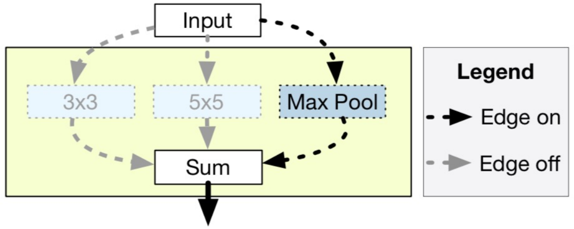
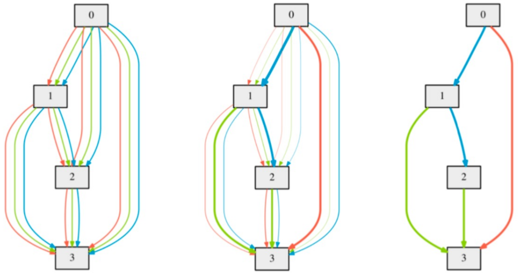
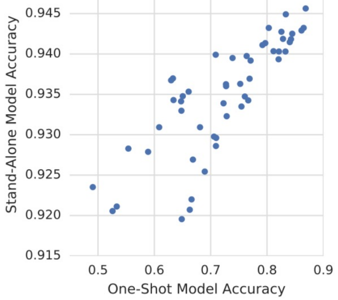
√ : Accessible on the user side

✕ : Restricted for the compression service

- Data: privacy and security
- GPU: quick deployment

Overview: Neural Architecture Search (NAS)

- NAS components

Search Space	Search Strategy	Performance Estimation
		

Search Space

- Basic cell (Zoph et.al, 2017)
- Width and depth (He et.al., 2017)
- Compression strategy (Wang et.al., 2019)

Search Strategy

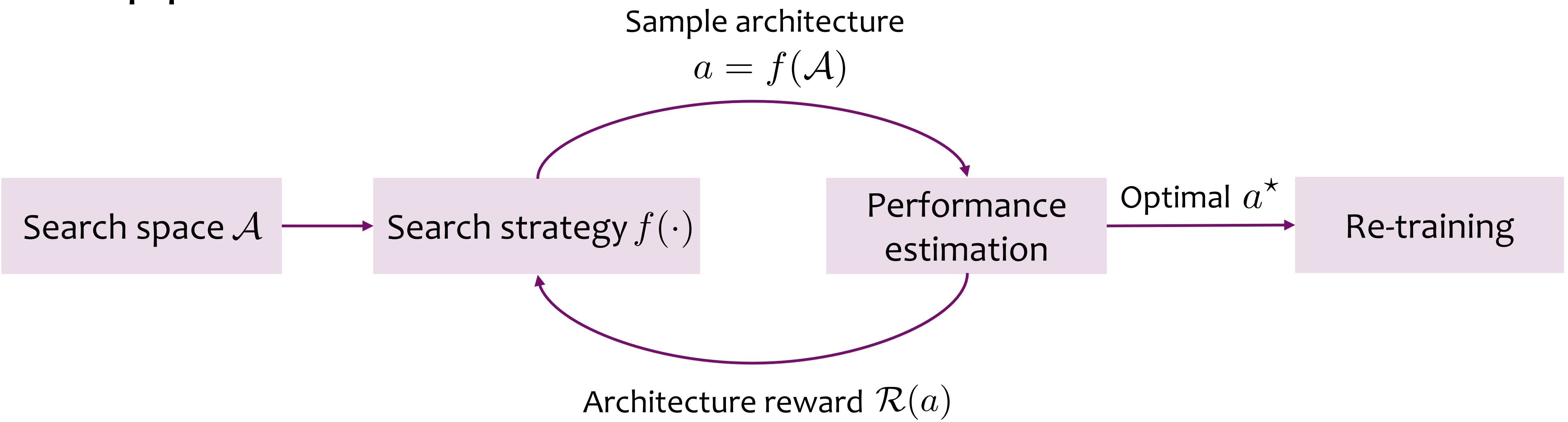
- Differentiable search (Liu et.al., 2019)
- Evolutionary algorithm (Real et.al., 2017)
- Reinforcement learning (Zoph et.al., 2017)

Performance Estimation

- Accuracy (Zoph et.al., 2017)
- Model storage (Zhu et.al, 2017)
- Computational FLOPs (He et.al., 2017)

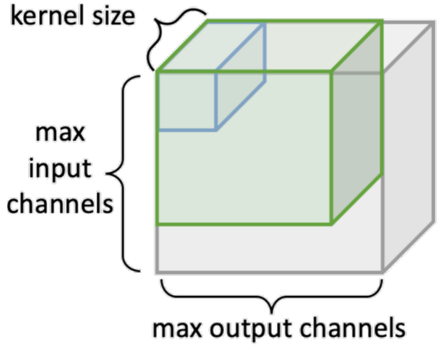
Overview: Neural Architecture Search (NAS)

- NAS pipeline

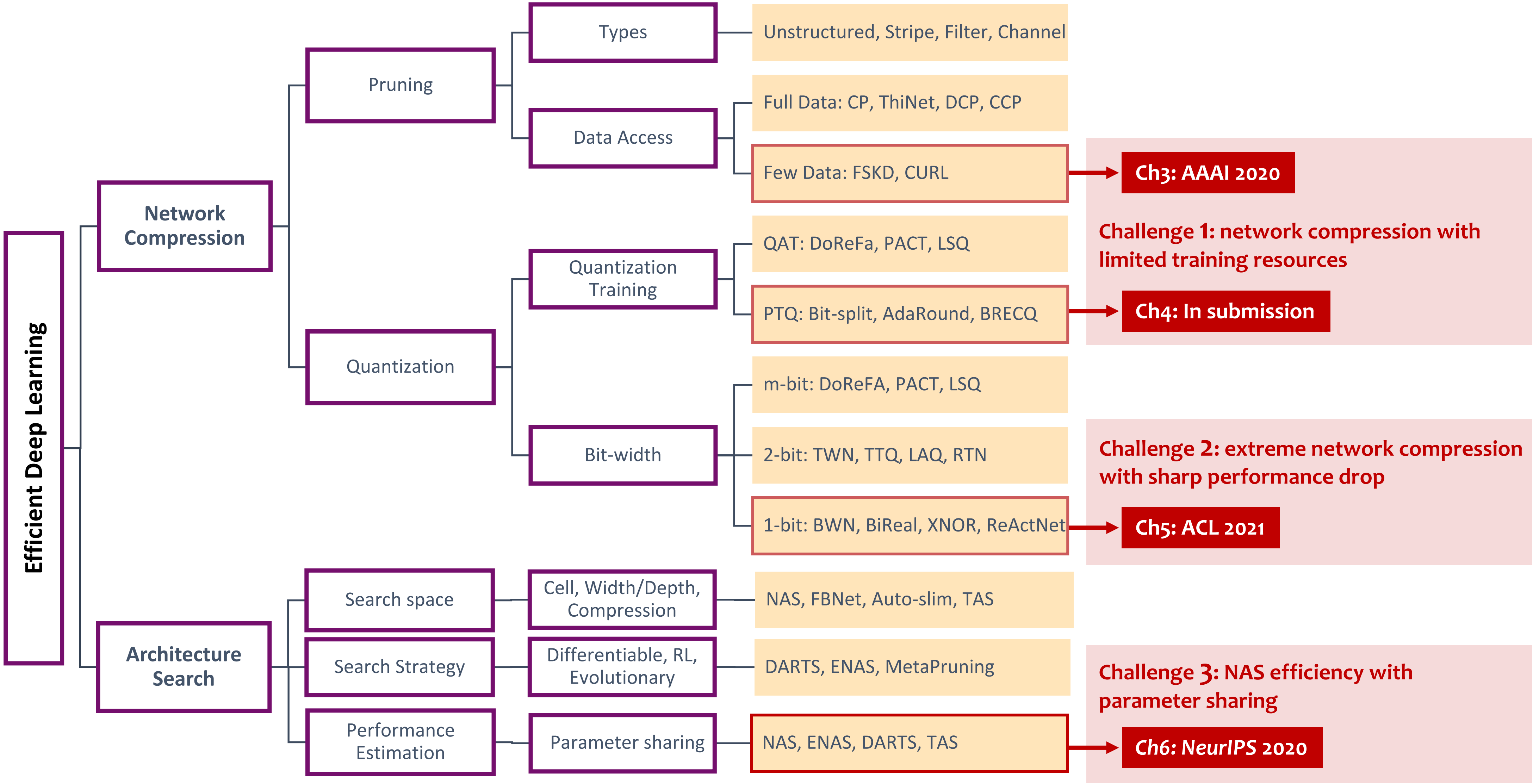


Challenge 3: NAS efficiency with parameter sharing

- Individually evaluating each candidate can take up to 1,000 GPU hours
- Existing solutions: parameter sharing
  - However, the mechanism behind is not well studied



Overall Taxonomy



Outline

Challenge 1: Network Compression with Limited Training Resources

1. **Few-shot Network Pruning via Cross Distillation (AAAI 2020)**
2. Efficient Post-training Quantization for Pre-trained Language Models (In submission)

Challenge 2: Extreme Compression with Sharp Performance Drop

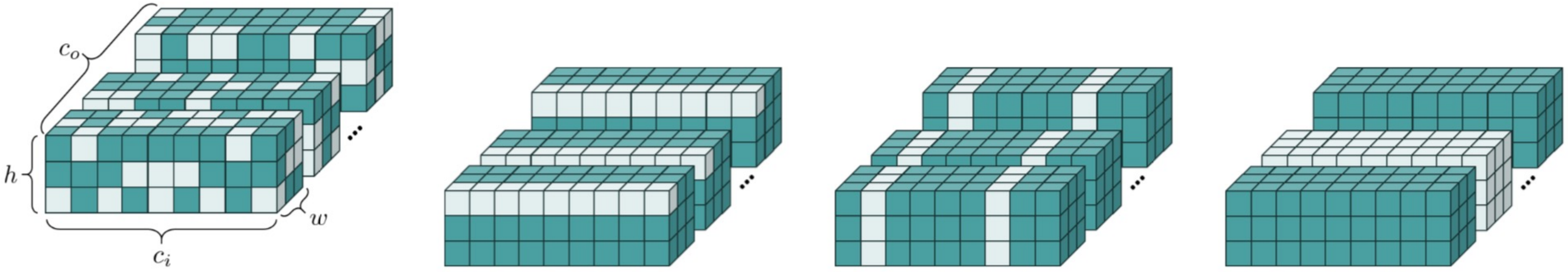
3. BinaryBERT: Pushing the Limit of BERT Quantization (ACL 2021)

Challenge 3: NAS Efficiency with Parameter Sharing

4. Revisit Parameter Sharing for Automatic Neural Channel Number Search (NeurIPS 2020)

Background: Network Pruning

- Given convolutional kernel $\mathbf{w} \in \mathrm{R}^{c_o \times c_i \times k \times k}$, find a mask $\mathbf{m} \in \{0,1\}^{c_o \times c_i \times k \times k}$ such that $\tilde{\mathbf{w}} = \mathbf{w} \odot \mathbf{m}$
- Types of pruning



(a) Unstricuterd Pruning. (b) Stripe Pruning. (c) Channel Pruning. (d) Filter Pruning.

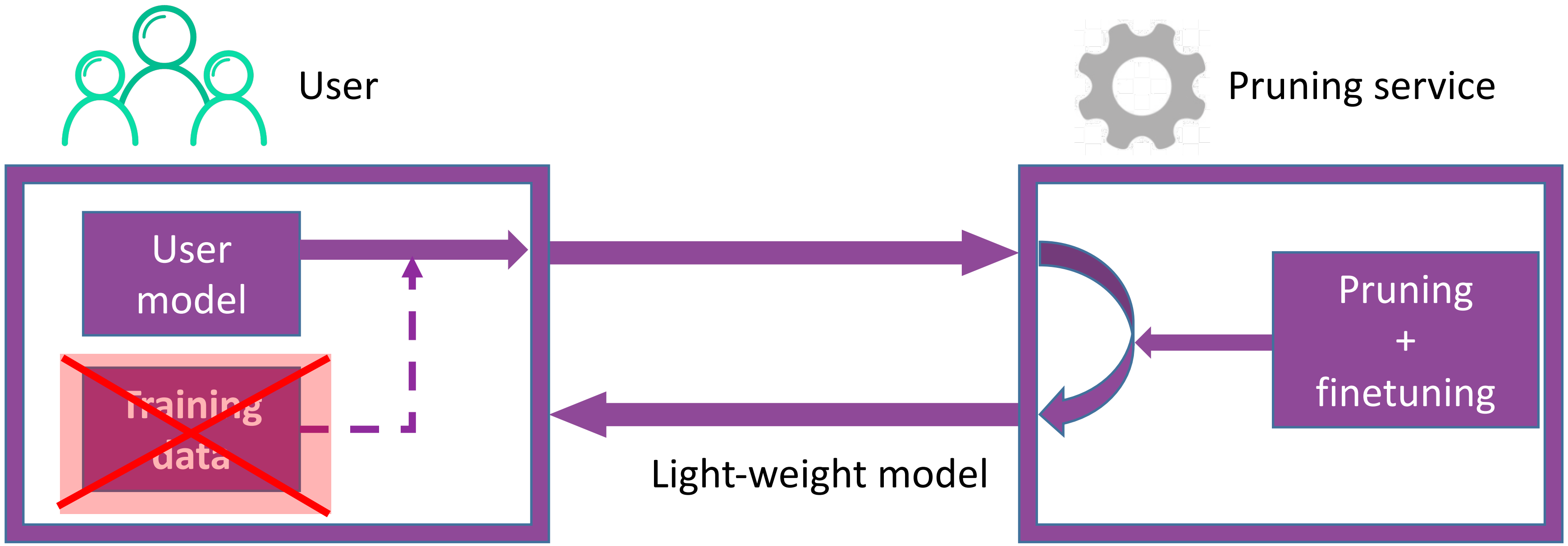
- Pruning criteria (by minimizing the loss change)

$$\ell(\tilde{\mathbf{w}}) \approx \ell(\mathbf{w}) + \mathbf{g}(\mathbf{w})^\top(\tilde{\mathbf{w}} - \mathbf{w}) + \frac{1}{2}(\tilde{\mathbf{w}} - \mathbf{w})^\top \mathbf{H}(\mathbf{w})(\tilde{\mathbf{w}} - \mathbf{w}).$$

1. Magnitude
2. Gradient (sensitivity)
3. Hessian (loss curvature)

Motivation

- **Typical paradigm** for network pruning



- **However,** passing the training data can be risky → Privacy issues!
- **New paradigm:** <u>few-shot network pruning</u> (e.g., 5 images per class)

Prior Methods

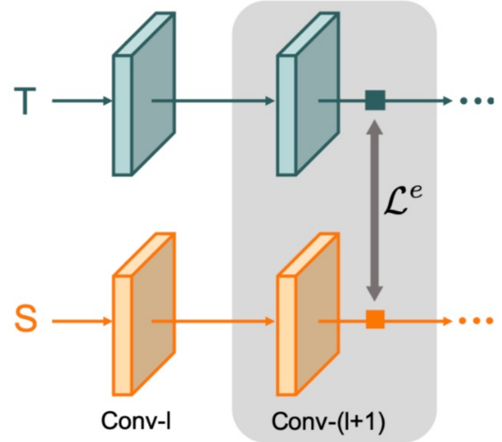
- Pruning resembles knowledge distillation

$\mathcal{F}^T$: Teacher (original unpruned model) $\mathcal{F}^S$: Student (pruned model)

- Minimize the layer-wise Euclidean distance
  - Objective function

Estimation error Pruning regularization

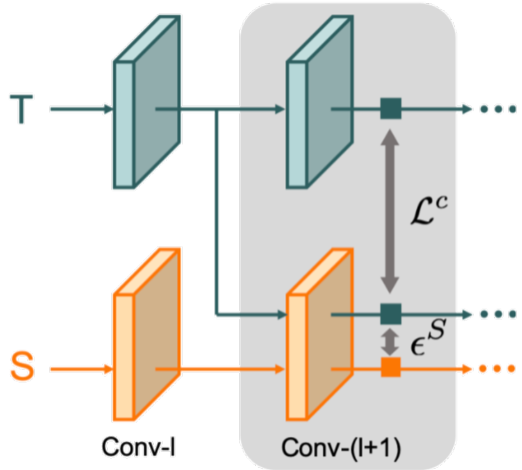
$$\mathbf{w}_*^S = \arg\min_{\mathbf{w}^S} \frac{1}{N} \|\mathbf{w}^T * \mathbf{h}^T - \mathbf{w}^S * \mathbf{h}^S\|_F^2 + \lambda\mathcal{R}(\mathbf{w}^S),$$



  - Layer-wise training: sample-efficient (Zhou et.al., 2020)
  - Poor generalization due to over-fitting to few-shot data
  - Error propagation layer-wisely

Our Approach: Cross Distillation

Correction

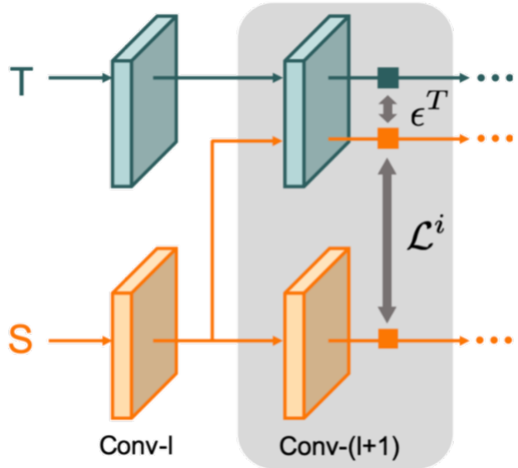


- Motivation
  Student receives clean signal from teacher to reduce error propagation

- Student discrepancy

$$\epsilon^S = \|\mathbf{W}^S * \mathbf{h}^T - \mathbf{W}^S * \mathbf{h}^S\|_F^2$$

Imitation



- Motivation
  Teacher becomes aware of the error accumulated on student

- Teacher discrepancy

$$\epsilon^T = \|\mathbf{W}^T * \mathbf{h}^S - \mathbf{W}^T * \mathbf{h}^T\|_F^2$$

Our Approach: Cross Distillation

- Correction

$$\mathcal{L}^c(\mathbf{w}^S) = \|\mathbf{w}^T * \mathbf{h}^T - \mathbf{w}^S * \mathbf{h}^T\|_F^2$$

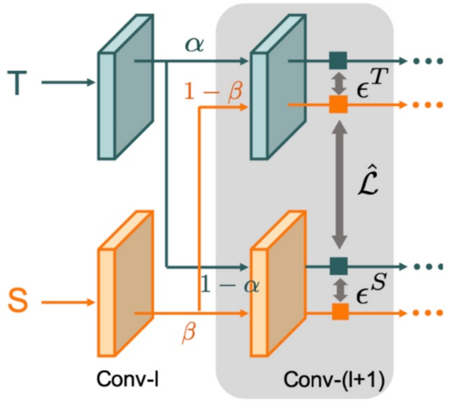
- Imitation

$$\mathcal{L}^i(\mathbf{w}^S) = \|\mathbf{w}^T * \mathbf{h}^S - \mathbf{w}^S * \mathbf{h}^S\|_F^2$$

- Trade-off between correction and imitation
  - Convex combination of loss terms

$$\tilde{\mathcal{L}} = \mu\mathcal{L}^c + (1-\mu)\mathcal{L}^i, \quad \mu \in [0,1].$$

  - Convex combination of cross connections



$$\begin{bmatrix}\hat{\mathbf{h}}^T \\ \hat{\mathbf{h}}^S\end{bmatrix} = \begin{bmatrix}\alpha & 1-\alpha \\ 1-\beta & \beta\end{bmatrix}\begin{bmatrix}\mathbf{h}^T \\ \mathbf{h}^S\end{bmatrix}, \quad \alpha,\beta \in [0,1]$$

$$\hat{\mathcal{L}}(\mathbf{w}^S) = \| \ (\mathbf{w}^T * \hat{\mathbf{h}}^T) - \ (\mathbf{w}^S * \hat{\mathbf{h}}^S)\|_F^2,$$

Pruning with Regularization $\mathcal{R}(\mathbf{W}^S)$

- Different regularizations on student parameters
  - Structured pruning: $\mathcal{R}(\mathbf{W}^S) = \|\mathbf{W}^S\|_{2,1} = \sum_i \|\mathbf{W}_i^S\|_2$ where $\mathbf{w}_i^S \in R^{c_o \times k \times k}$
  - Unstructured pruning: $\mathcal{R}(\mathbf{W}^S) = \|\mathbf{W}^S\|_1 = \sum_{i,j,h,w} |W_{ijhw}^S|$

- Solve by proximal gradient descent:
  - Structured pruning: $\mathrm{Prox}_{\lambda\|\cdot\|_2}(\mathbf{w}_i^S) = \max(1 - \frac{\lambda}{\|\mathbf{w}_i^S\|_2}, 0) \cdot \mathbf{w}_i^S$
  - Unstructured pruning:

$$\mathrm{Prox}_{\lambda\|\cdot\|_1}(W_{ijhw}^S) = \begin{cases} W_{ijhw}^S - \lambda & W_{ijhw}^S > \lambda \\ 0 & |W_{ijhw}^S| \leq \lambda \\ W_{ijhw}^S + \lambda & W_{ijhw}^S < -\lambda \end{cases}$$

Experimental Results: Structured Pruning

- 50% channel sparsity
- VGG-19 on CIFAR-10
- Few-shot data: {1, 2, 3, 5, 10, 50} data / per class

- CD: convex combin. over loss terms
- SCD: convex combin over feature maps

Methods	1	2	3	5	10	50
L1-norm	$14.36_{\pm 0.00}$	$14.36_{\pm 0.00}$	$14.36_{\pm 0.00}$	$14.36_{\pm 0.00}$	$14.36_{\pm 0.00}$	$14.36_{\pm 0.00}$
BP	$49.24_{\pm 1.76}$	$49.32_{\pm 1.88}$	$51.39_{\pm 1.53}$	$55.73_{\pm 1.19}$	$57.48_{\pm 0.91}$	$64.69_{\pm 0.43}$
FSKD	$47.91_{\pm 1.82}$	$55.44_{\pm 1.71}$	$61.76_{\pm 1.39}$	$65.69_{\pm 1.08}$	$72.20_{\pm 0.74}$	$75.46_{\pm 0.49}$
FitNet	$48.51_{\pm 2.51}$	$71.51_{\pm 2.03}$	$76.22_{\pm 1.95}$	$81.10_{\pm 1.13}$	$85.40_{\pm 1.02}$	$88.46_{\pm 0.76}$
ThiNet	$58.06_{\pm 1.71}$	$72.07_{\pm 1.68}$	$75.37_{\pm 1.59}$	$78.03_{\pm 1.24}$	$81.15_{\pm 0.85}$	$86.12_{\pm 0.45}$
CP	$66.03_{\pm 1.56}$	$75.23_{\pm 1.49}$	$77.98_{\pm 1.47}$	$81.53_{\pm 1.29}$	$83.59_{\pm 0.78}$	$87.27_{\pm 0.27}$
w/o CD	$65.57_{\pm 1.61}$	$75.44_{\pm 1.69}$	$78.40_{\pm 1.53}$	$81.20_{\pm 1.13}$	$84.07_{\pm 0.83}$	$87.67_{\pm 0.29}$
CD	$\mathbf{69.25_{\pm 1.39}}$	$\mathbf{80.65_{\pm 1.47}}$	$\mathbf{82.08_{\pm 1.41}}$	$\mathbf{84.91_{\pm 0.98}}$	$\mathbf{86.61_{\pm 0.71}}$	$87.64_{\pm 0.24}$
SCD	$68.53_{\pm 1.59}$	$76.83_{\pm 1.43}$	$80.16_{\pm 1.32}$	$84.28_{\pm 1.19}$	$86.30_{\pm 0.79}$	$\mathbf{88.65_{\pm 0.33}}$

Experimental Results: Unstructured Pruning

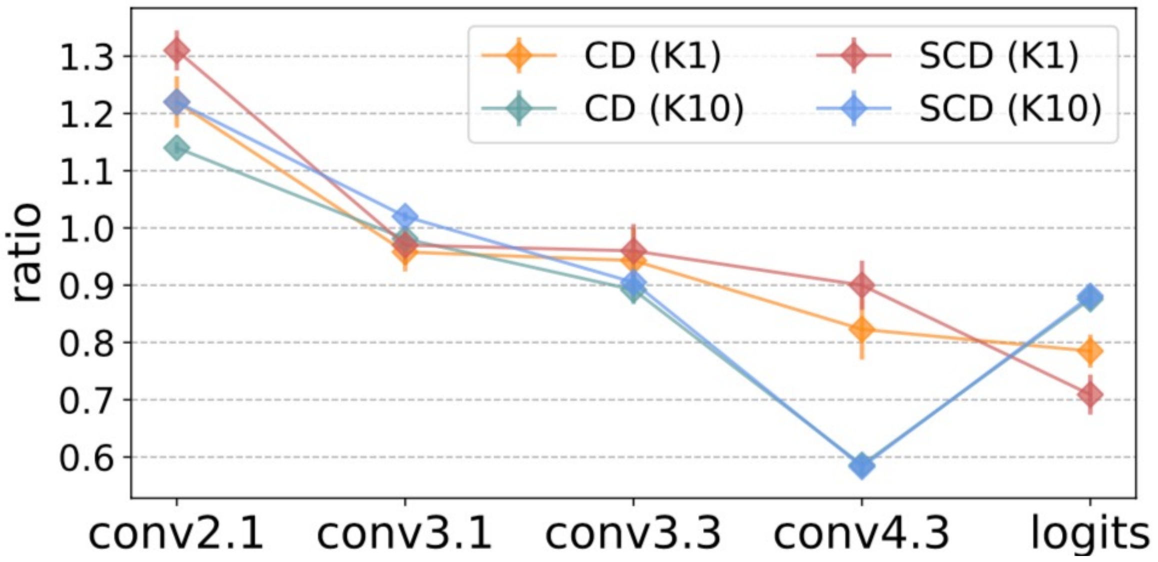
- 50% sparsity
- VGG-19 on ImageNet
- Few-shot data:
  - {50, 100, 500} randomly sampled data in any classes
  - {1, 2, 3} data / per class

Methods	50	100	500	1	2	3
L1-norm	$0.5_{\pm 0.00}$	$0.5_{\pm 0.00}$	$0.5_{\pm 0.00}$	$0.5_{\pm 0.00}$	$0.5_{\pm 0.00}$	$0.5_{\pm 0.00}$
BP	$42.87_{\pm 2.07}$	$48.78_{\pm 1.43}$	$65.47_{\pm 1.15}$	$71.25_{\pm 0.97}$	$74.85_{\pm 0.71}$	$76.04_{\pm 0.48}$
FitNet	$52.66_{\pm 2.93}$	$57.09_{\pm 2.14}$	$76.59_{\pm 1.45}$	$80.14_{\pm 1.23}$	$82.27_{\pm 0.70}$	$83.14_{\pm 0.51}$
w/o CD	$78.73_{\pm 1.78}$	$83.29_{\pm 1.12}$	$85.04_{\pm 0.93}$	$85.36_{\pm 0.61}$	$85.21_{\pm 0.41}$	$85.49_{\pm 0.46}$
CD	$\mathbf{83.81}_{\pm \mathbf{1.49}}$	$86.21_{\pm 1.09}$	$87.19_{\pm 0.96}$	$87.61_{\pm 0.82}$	$87.78_{\pm 0.45}$	$87.86_{\pm 0.39}$
SCD	$83.67_{\pm 1.52}$	$\mathbf{86.72}_{\pm \mathbf{1.23}}$	$\mathbf{87.82}_{\pm \mathbf{1.04}}$	$\mathbf{88.14}_{\pm \mathbf{0.74}}$	$\mathbf{88.23}_{\pm \mathbf{0.61}}$	$\mathbf{88.38}_{\pm \mathbf{0.43}}$

Experimental Results: Discussions

- How cross distillation alleviate the error propagation
- Compare the ratio of estimation error on the test set

$$\text{Ratio} = \frac{\mathcal{L}_{ours}}{\mathcal{L}_{prev}} \qquad (\ \|\mathbf{w}^T * \mathbf{h}^T - \mathbf{w}^S * \mathbf{h}^S\|_F^2 \ )$$



Ratio < 1: generalize better

Summary

- We study the problem of few-shot network pruning, a new pruning paradigm that considers data security issues for users

- We propose cross distillation, a new layer-wise pruning technique with knowledge distillation. The interconnection between teacher and student layers alleviate the error propagation

- Experiments on popular network architectures show that our approach can bring consistent improvement for pruning even when only 1~10 images per class are available

Outline

Challenge 1: Network Compression with Limited Training Resources

1. Few Shot Network Pruning via Cross Distillation (AAAI 2020)
2. **Efficient Post-training Quantization for Pre-trained Language Models (In submission)**

Challenge 2: Extreme Compression with Sharp Performance Drop

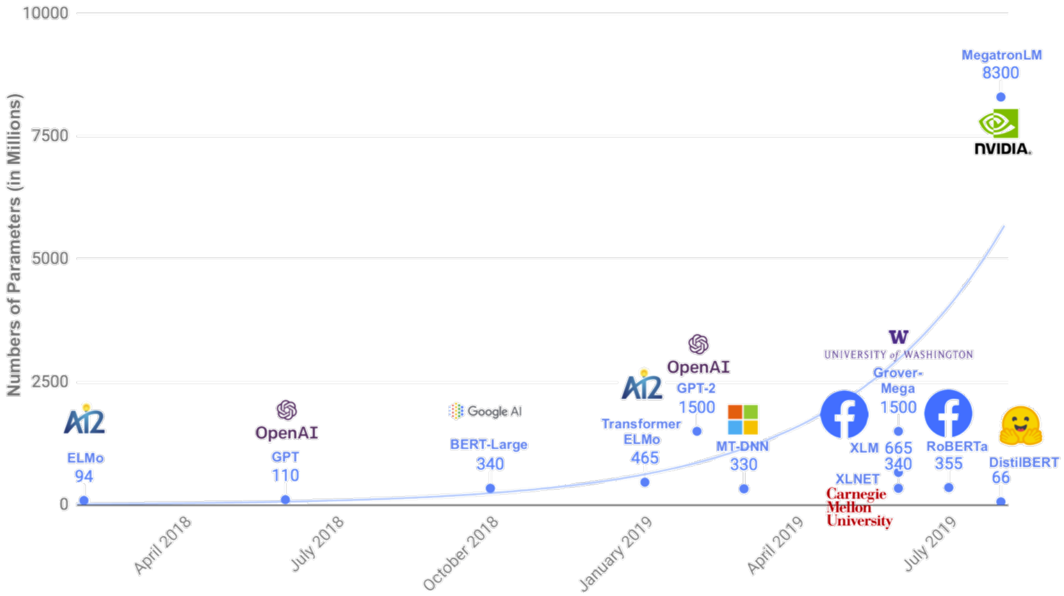
3. BinaryBERT: Pushing the Limit of BERT Quantization (ACL 2021)

Challenge 3: NAS Efficiency with Parameter Sharing

4. Revisit Parameter Sharing for Automatic Neural Channel Number Search (NeurIPS 2020)

Network Quantization in NLP Tasks

- The increasing size of pre-trained models (Sanh et.al., 2020)



- The huge pre-training corpus: slow training
  - BERT (Devlin et.al., 2018) uses BookCorpus (800M words) & English Wikipedia (2500M words)
- Even resource-demanding for network compression
- Efficient quantization pipelines

Background: Quantization

- Given the full-precision parameter $\mathbf{w}$
  - Multi-bit quantization (b-bit):

$$\hat{\mathbf{w}} = \mathcal{Q}_b(\mathbf{w}) = s \cdot \Pi_{\Omega(b)}(\mathbf{w}/s), \quad \Omega(b) == \{-2^{b-1}, ..., 0, ..., 2^{b-1} - 1\}$$

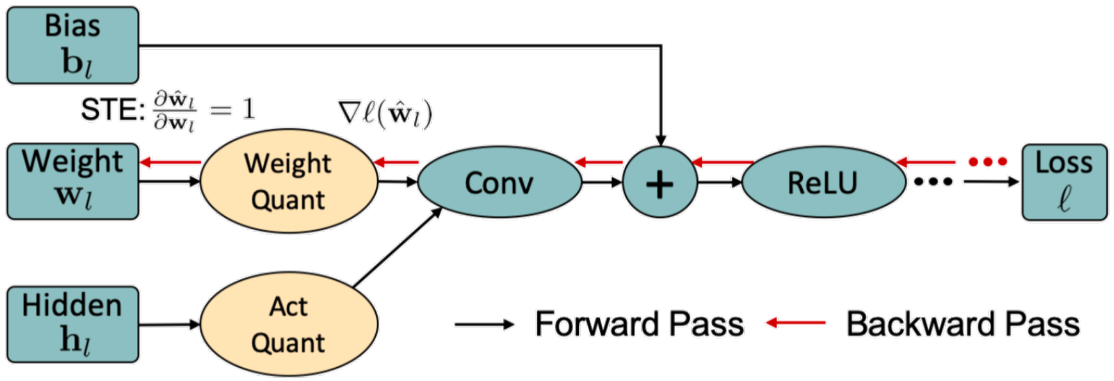
  - Ternarization (2-bit)

$$\hat{w}_i^t = \mathcal{Q}_2(w_i) = \begin{cases} \alpha \cdot \mathrm{sign}(w_i) & |w_i| \geq \Delta \\ 0 & |w_i| < \Delta \end{cases}$$

  - Binarization (1-bit)

$$\hat{w}_i^b = \mathcal{Q}_1(w_i) = \alpha \cdot \mathrm{sign}(w_i).$$

- Quantization workflow



Background: Quantization

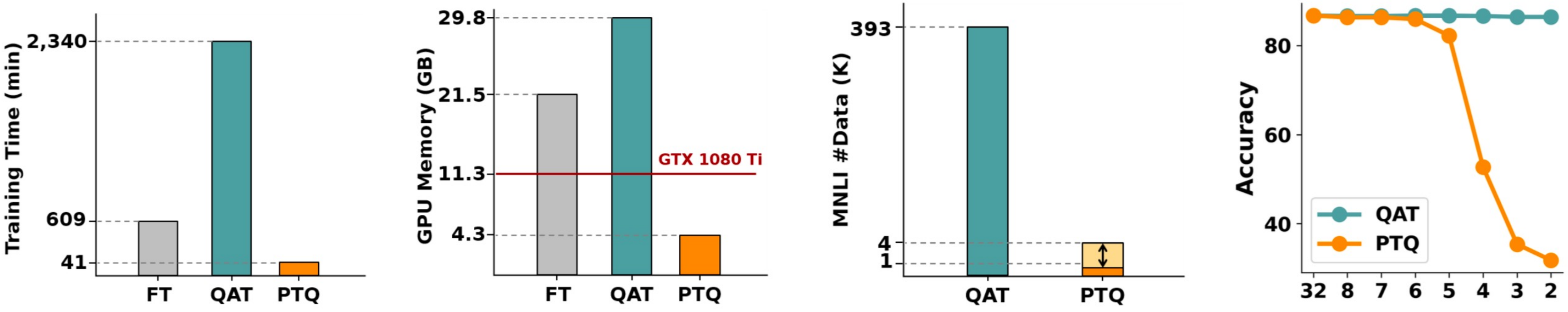
- Training
  - Quantization-aware training (QAT): cross entropy over full data

$$\min_{\mathbf{w},\mathbf{s}} \mathrm{E}_{\mathbf{x}\sim\mathcal{D}} \left[\ell(\mathbf{x}; \hat{\mathbf{w}}, \mathbf{s})\right], \quad \text{s.t. } \hat{\mathbf{w}} = \mathcal{Q}_b(\mathbf{w}).$$

  - Post-training quantization (PTQ): reconstruction error over few data

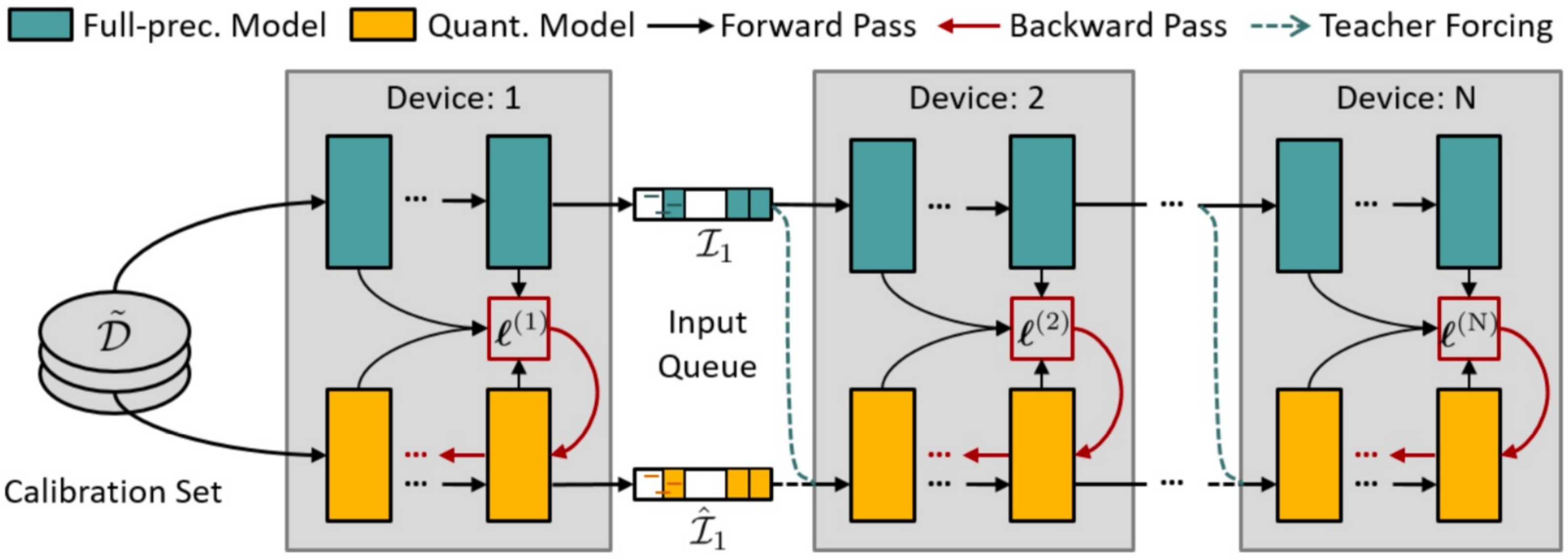
$$\min_{\mathbf{w},\mathbf{s}} \|\hat{\mathbf{w}}^\top \hat{\mathbf{a}} - \mathbf{w}^\top \mathbf{a}\|^2, \quad \text{s.t. } \hat{\mathbf{w}} = \mathcal{Q}_b(\mathbf{w}).$$ (Similar to layer-wise pruning)

- Comparison



(a) Training Time. (b) Memory. (c) Data Accessibility. (d) Weight Quantization.

Methodology: Model Splitting

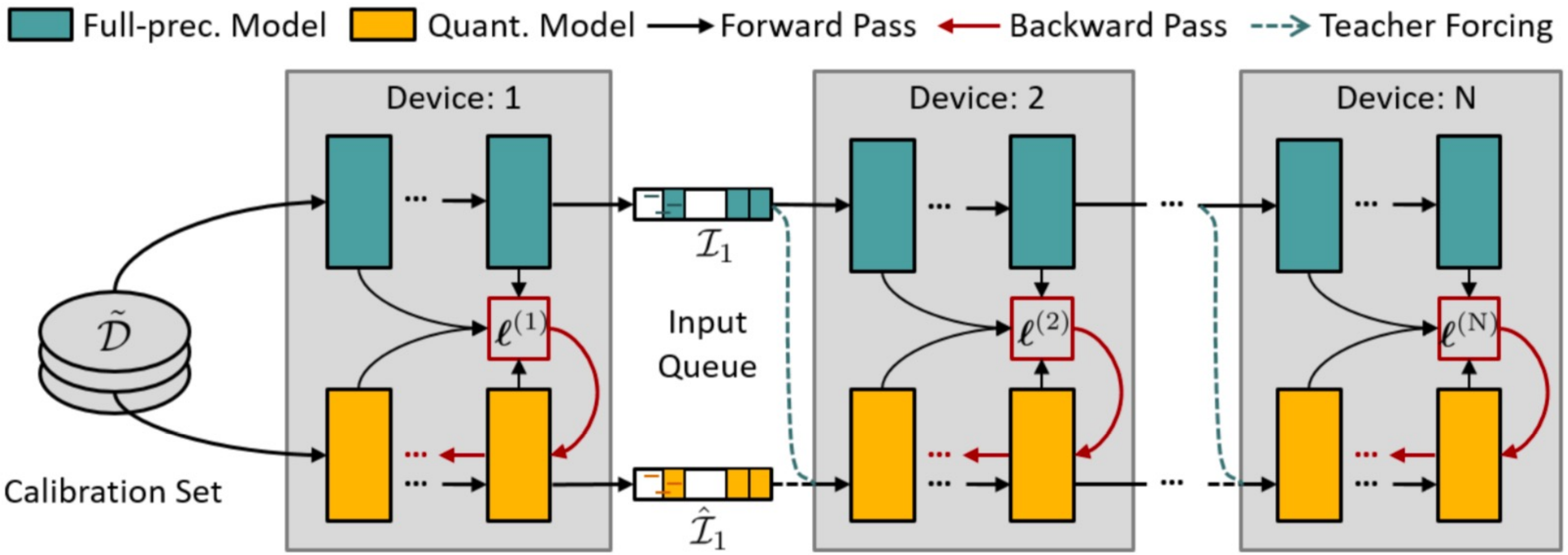


- Goal: improve post-training quantization while keeping its advantages
- Approach: split the language model into multiple modules
- Improvement: layer-wise -> module-wise

$$\min_{\mathbf{w},\mathbf{s}} \|\hat{\mathbf{w}}^\top \hat{\mathbf{a}} - \mathbf{w}^\top \mathbf{a}\|^2, \quad \Rightarrow \quad \min_{\mathbf{w}_n,\mathbf{s}_n} \ell^{(n)} \triangleq \sum_{l\in[l_n,l_{n+1})} \|\hat{\boldsymbol{f}}_l - \boldsymbol{f}_l\|^2,$$

where $\boldsymbol{f}_l$ and $\hat{\boldsymbol{f}}_l$ are the full-precision and quantized output of each module

Methodology: Parallel Training



- Training procedure:
  - Sequential training: one by one
  - Parallel training: an input queue help achieve theoretical speedup
- Teacher forcing $\tilde{\boldsymbol{f}}_{l_n} = \lambda \boldsymbol{f}_{l_n} + (1-\lambda)\hat{\boldsymbol{f}}_{l_n}, \quad \lambda \in [0,1],$ (resembles cross distillation)
- Adapt to normal training: $\lambda_t = \max(1 - \frac{t}{T_0}, \ 0)$

Experiments: Main Results

- Text classification (MNLI)
- Only 4K training instances (original dataset: 393K instances)
- Our approach: MREM-S (sequential) and MREM-P (parallel)

	#Bits (W-E-A)	Quant Method	BERT-base Time (min)	BERT-base Mem (G)	BERT-base # Data (K)	BERT-base Acc m(%)	BERT-base Acc mm(%)	BERT-large Time (min)	BERT-large Mem (G)	BERT-large # Data (K)	BERT-large Acc m(%)	BERT-large Acc mm(%)
MNLI	*full-prec*	N/A	220	8.6	393	84.5	84.9	609	21.5	393	86.7	85.9
	4-4-8	QAT	1,320	11.9	393	84.6	84.9	3,180	29.8	393	86.9	86.7
		REM	28	2.5	4	$73.3_{\pm0.3}$	$74.9_{\pm0.2}$	84	5.5	4	$70.0_{\pm0.4}$	$71.8_{\pm0.3}$
		MREM-S	36	4.6	4	$83.5_{\pm0.1}$	$83.9_{\pm0.1}$	84	10.8	4	$86.1_{\pm0.1}$	$85.9_{\pm0.1}$
		MREM-P	9	3.7	4	$83.4_{\pm0.1}$	$83.7_{\pm0.1}$	21	8.6	4	$85.5_{\pm0.1}$	$85.4_{\pm0.2}$
	2-2-8	QAT	882	11.9	393	84.4	84.6	2,340	29.8	393	86.5	86.1
		REM	24	2.5	4	$71.6_{\pm0.4}$	$73.4_{\pm0.4}$	64	5.5	4	$66.9_{\pm0.4}$	$68.6_{\pm0.7}$
		MREM-S	24	4.6	4	$82.7_{\pm0.2}$	$82.7_{\pm0.2}$	64	10.8	4	$85.4_{\pm0.2}$	$85.3_{\pm0.2}$
		MREM-P	6	$3.7_{\times4}$	4	$82.3_{\pm0.2}$	$82.6_{\pm0.2}$	16	$8.6_{\times4}$	4	$84.6_{\pm0.2}$	$84.6_{\pm0.1}$
	2-2-4	QAT	875	11.9	393	83.5	84.2	2,280	29.8	393	85.8	85.9
		REM	24	2.5	4	$58.3_{\pm0.5}$	$60.6_{\pm0.6}$	64	5.5	4	$48.8_{\pm0.6}$	$51.4_{\pm0.8}$
		MREM-S	24	4.6	4	$81.1_{\pm0.2}$	$81.5_{\pm0.2}$	64	10.8	4	$83.6_{\pm0.2}$	$83.7_{\pm0.2}$
		MREM-P	6	$3.7_{\times4}$	4	$80.8_{\pm0.2}$	$81.2_{\pm0.2}$	16	$8.6_{\times4}$	4	$83.0_{\pm0.3}$	$83.2_{\pm0.2}$

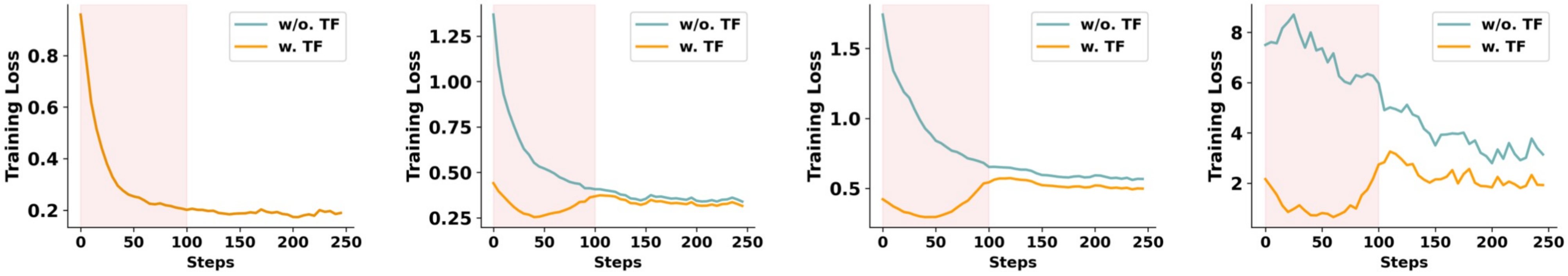
Experiments: Compare with Existing SOTA

- Compare with existing SOTA (both QAT and PTQ baselines)
- On GLUE benchmark

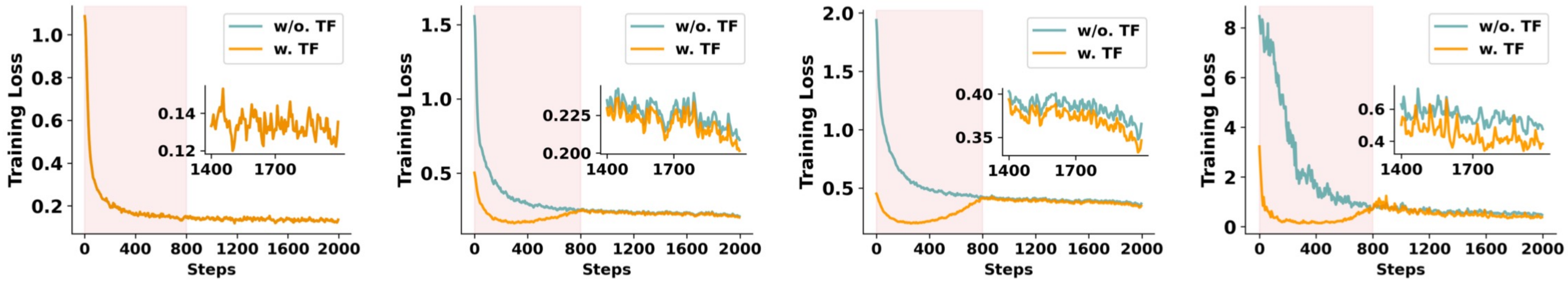
Quant Method	#Bits (W-E-A)	Size (MB)	PTQ	MNLI-m	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
-	*full-prec.*	418	-	84.9	91.4	92.1	93.2	59.7	90.1	86.3	72.2	83.9
Q-BERT	2-8-8	43	✗	76.6	-	-	84.6	-	-	-	-	-
Q-BERT	2/4-8-8	53	✗	83.5	-	-	92.6	-	-	-	-	-
Quant-Noise	PQ	38	✗	83.6	-	-	-	-	-	-	-	-
TernaryBERT	2-2-8	28	✗	83.3	90.1	91.1	92.8	55.7	87.9	87.5	72.9	82.7
GOBO	3-4-32	43	✓	83.7	-	-	-	-	88.3	-	-	-
GOBO	2-2-32	28	✓	71.0	-	-	-	-	82.7	-	-	-
MREM-S	4-4-8	50	✓	$83.5_{\pm 0.1}$	$90.2_{\pm 0.1}$	$91.2_{\pm 0.1}$	$91.4_{\pm 0.4}$	$55.1_{\pm 0.8}$	$89.1_{\pm 0.1}$	$84.8_{\pm 0.0}$	$71.8_{\pm 0.0}$	$82.4_{\pm 0.1}$
	2-2-8	28	✓	$82.7_{\pm 0.2}$	$89.6_{\pm 0.1}$	$90.3_{\pm 0.2}$	$91.2_{\pm 0.4}$	$52.3_{\pm 1.0}$	$88.7_{\pm 0.1}$	$86.0_{\pm 0.0}$	$71.1_{\pm 0.0}$	$81.5_{\pm 0.2}$
MREM-P	4-4-8	50	✓	$83.4_{\pm 0.1}$	$90.2_{\pm 0.1}$	$91.0_{\pm 0.2}$	$91.5_{\pm 0.4}$	$54.7_{\pm 0.9}$	$89.1_{\pm 0.1}$	$86.3_{\pm 0.0}$	$71.1_{\pm 0.0}$	$82.2_{\pm 0.1}$
	2-2-8	28	✓	$82.3_{\pm 0.2}$	$89.4_{\pm 0.1}$	$90.3_{\pm 0.2}$	$91.3_{\pm 0.4}$	$52.9_{\pm 1.2}$	$88.3_{\pm 0.2}$	$85.8_{\pm 0.0}$	$72.9_{\pm 0.0}$	$81.6_{\pm 0.2}$

Experiments: Effect of Teacher Forcing

- Loss curves with 250 training steps (up) and 2,000 training steps (down)

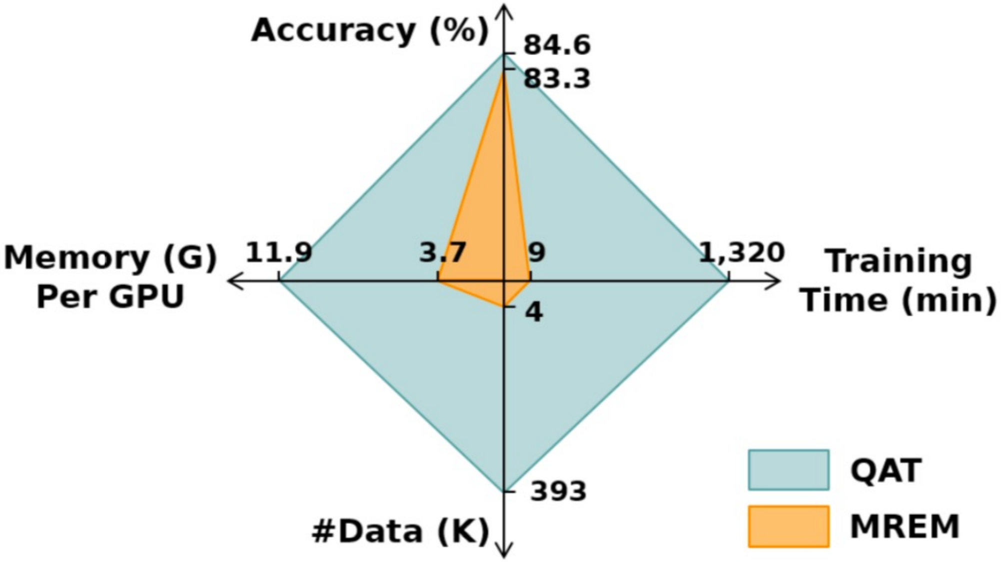


(a) 250 Steps, Module-1 (b) 250 Steps, Module-2 (c) 250 Steps, Module-3 (d) 250 Steps, Module-4



(e) 2,000 Steps, Module-1 (f) 2,000 Steps, Module-2 (g) 2,000 Steps, Module-3 (h) 2,000 Steps, Module-4

Summary



- We investigate post-training quantization (PTQ) for pre-trained language models
- The proposed PTQ method enjoys quick training (36x ~ 144x faster), light memory consumption (3x savings) with only 4K instances (<1%) and reasonable performance (1.3% drop compared with QAT)
- The designed parallel strategy further achieves theoretical training speed-up (e.g., 4x on 4 GPUs)

Outline

Challenge 1: Network Compression with Limited Training Resources

1. Few-Shot Network Pruning via Cross Distillation (AAAI 2020)

2. Efficient Post-training Quantization for Pre-trained Language Models (In submission)

Challenge 2: Extreme Compression with Sharp Performance Drop

3. **BinaryBERT: Pushing the Limit of BERT Quantization (ACL 2021)**

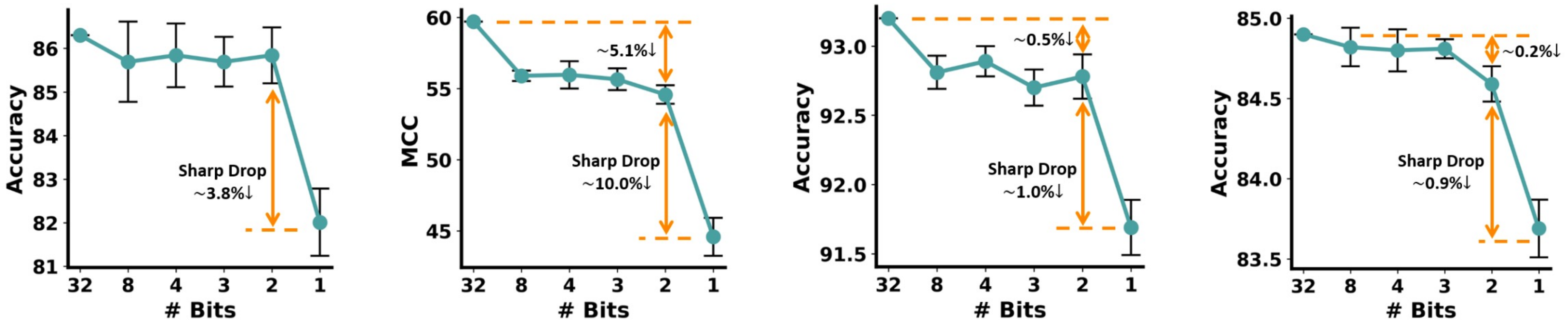
Challenge 3: NAS Efficiency with Parameter Sharing

4. Revisit Parameter Sharing for Automatic Neural Channel Number Search (NeurIPS 2020)

Introduction

- Advantages of binarization (1-bit):
  - The most size reduction
  - Conversion of floating-point multiplication to cheap integer addition
  - Fast and energy-saving on edge devices

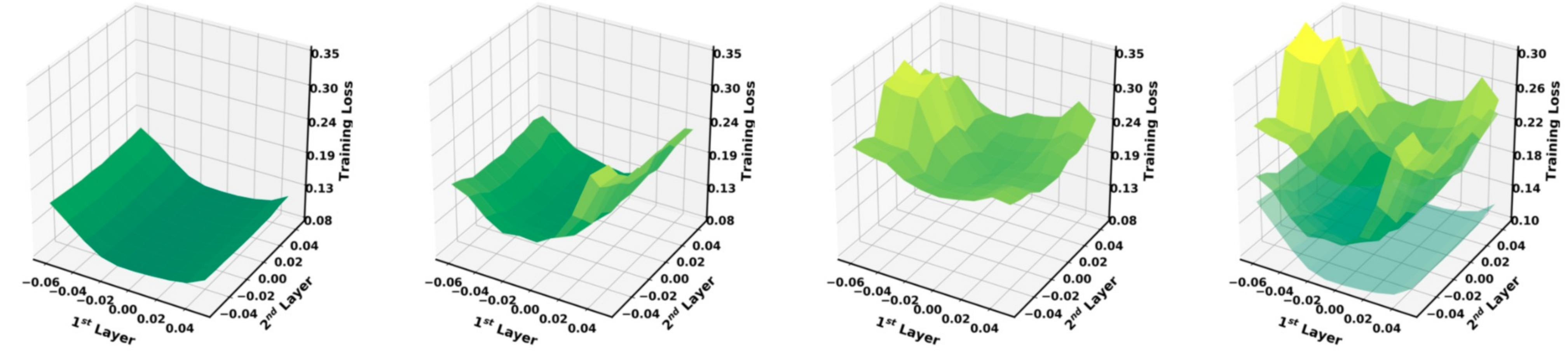
- However, it is **HARD** to train a binary BERT directly



(a) MRPC. (b) CoLA. (c) SST-2. (d) MNLI-m.

Background: Underlying Challenges

- Visualization of loss landscape



(a) Full-precision Model. (b) Ternary Model. (c) Binary Model. (d) All Together.

- Perturbation as follows:

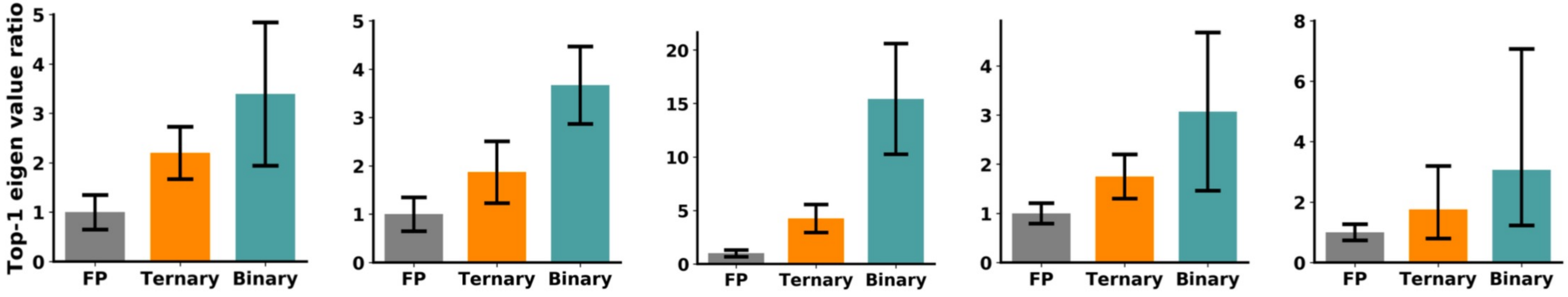
$$\tilde{\mathbf{w}}_x = \mathbf{w}_x + x \cdot \mathbf{1}_x, \quad \tilde{\mathbf{w}}_y = \mathbf{w}_y + y \cdot \mathbf{1}_y,$$

where $\bar{w}_x$ is the average value of $\mathbf{w}_x$, and

$$x \in \{\pm 0.2\bar{w}_x, \pm 0.4\bar{w}_x, ..., \pm 1.0\bar{w}_x\}$$

Background: Underlying Challenges

- The top-1 eigenvalue of Hessian matrix $\mathbf{H}$ at different parts



(a) MHA-QK. (b) MHA-V. (c) MHA-O. (d) FFN-Mid. (e) FFN-Out.

- Measuring the steepness of loss curvature

$$\ell(\hat{\mathbf{w}}) - \ell(\mathbf{w}) \approx \boldsymbol{\epsilon}^\top \mathbf{H} \boldsymbol{\epsilon} \leq \lambda_{\max} \|\boldsymbol{\epsilon}\|^2,$$

- $\boldsymbol{\epsilon} = \mathbf{w} - \hat{\mathbf{w}}$ is the quantization noise
- Top-1 eigenvalue reflects the quantization sensitivity

Methodology: Ternary Weight Split

- First train a ternary BERT as the bridge model

- For each ternary weight $\mathbf{w}^t$ and its quantized counterpart $\hat{\mathbf{w}}^t$, we apply ternary weight splitting (TWS) as

$$\mathbf{w}^t = \mathbf{w}_1^b + \mathbf{w}_2^b, \quad \hat{\mathbf{w}}^t = \hat{\mathbf{w}}_1^b + \hat{\mathbf{w}}_2^b \ .$$

- TWS ensures equivalency, inheriting knowledge from ternary model
- We assign the following form of solution

$$[\mathbf{w}_1^b]_i = \begin{cases} a \cdot w_i^t & \text{if } \hat{w}_i^t \neq 0 \\ b + w_i^t & \text{if } \hat{w}_i^t = 0, w_i^t > 0 \\ b & \text{otherwise} \end{cases} \quad [\mathbf{w}_2^b]_i = \begin{cases} (1-a) w_i^t & \text{if } \hat{w}_i^t \neq 0 \\ -b & \text{if } \hat{w}_i^t = 0, w_i^t > 0 \\ -b + w_i^t & \text{otherwise} \end{cases}$$

- Next: solve a and b

Methodology: Ternary Weight Split

- TWS allows closed-form solution as

$$a = \frac{\sum_{i\in\mathcal{I}}|w_i^t| + \sum_{j\in\mathcal{J}}|w_j^t| - \sum_{k\in\mathcal{K}}|w_k^t|}{2\sum_{i\in\mathcal{I}}|w_i^t|},$$

$$b = \frac{\frac{n}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}|w_i^t| - \sum_{i=1}^{n}|w_i^t|}{2(|\mathcal{J}| + |\mathcal{K}|)},$$

- where $\mathcal{I} = \{i \mid \hat{w}_i^t \neq 0\}$, $\mathcal{J} = \{j \mid \hat{w}_j^t = 0 \text{ and } w_j^t > 0\}$, $\mathcal{K} = \{k \mid \hat{w}_k^t = 0 \text{ and } w_k^t < 0\}$.
- TWS can be finished immediately
- Detailed derivations can be found in the thesis

Methodology: Ternary Weight Split

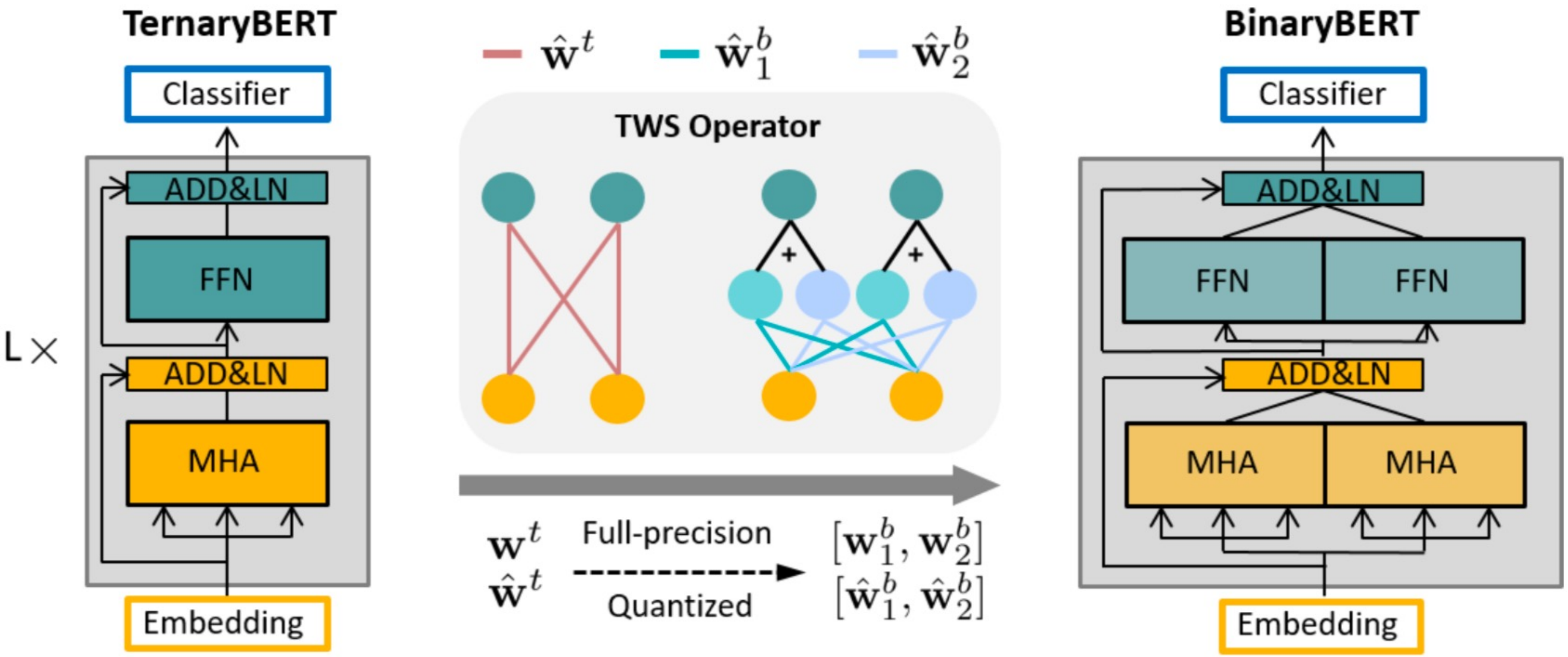


Figure 4: The overall workflow of training BinaryBERT. We first train a half-sized ternary BERT model, and then apply ternary weight splitting operator (Equations (6) and (7)) to obtain the latent full-precision and quantized weights as the initialization of the full-sized BinaryBERT. We then fine-tune BinaryBERT for further refinement.

Methodology: Adaptive Splitting

- Adaptive splitting: fit BinaryBERT to various edge devices
- Train a ternary and binary mixed BERT, and split the ternary (sensitive) ones
- Equivalent to mixed-precision, but enjoy hard-ware efficiency
- Formulation: a combinatorial optimization problem

$$
\begin{aligned}
\max_{\mathbf{s}} \quad & \mathbf{u}^\top \mathbf{s} \\
\text{s.t.} \quad & \mathbf{c}^\top \mathbf{s} \leq \mathcal{C} - \mathcal{C}_0, \;\; \mathbf{s} \in \{0,1\}^Z, \\
& \mathbf{s} \in \{0,1\}^Z
\end{aligned}
$$

where $\mathcal{C}$ is the resource constraint, and $\mathbf{u} \in \mathbb{R}_+^Z$ is the utility vector

- The utility $\mathbf{u}$ can be measured by performance gain from ternarization
- A knapsack problem, solved by dynamic programing

Experiments: Main Results

- GLUE benchmark (test set results)
- TWS (ours): ternary weight splitting
- BWN: train binary model from scratch

#	Quant	#Bits (W-E-A)	Size (MB)	FLOPs (G)	DA	MNLI -m/mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
1	-	*full-prec.*	417.6	22.5	-	84.5/84.1	89.5	91.3	93.0	54.9	84.4	87.9	69.9	82.2
2	BWN	1-1-8	13.4	3.1	✗	83.3/83.4	88.9	**90.1**	92.3	38.1	81.2	**86.1**	63.1	78.5
3	TWS	1-1-8	16.5	3.1	✗	**84.1/83.6**	**89.0**	90.0	**93.1**	**50.5**	**83.4**	86.0	**65.8**	**80.6**
4	BWN	1-1-4	13.4	1.5	✗	83.5/82.5	**89.0**	**89.4**	92.3	26.7	78.9	84.2	59.9	76.3
5	TWS	1-1-4	16.5	1.5	✗	**83.6/82.9**	**89.0**	89.3	**93.1**	**37.4**	**82.5**	**85.9**	**62.7**	**78.5**
6	BWN	1-1-8	13.4	3.1	✓	83.3/83.4	88.9	**90.3**	91.3	48.4	**83.2**	**86.3**	66.1	80.1
7	TWS	1-1-8	16.5	3.1	✓	**84.1/83.5**	**89.0**	89.8	**91.9**	**51.6**	82.3	85.9	**67.3**	**80.6**
8	BWN	1-1-4	13.4	1.5	✓	83.5/82.5	**89.0**	**89.9**	92.0	45.0	81.9	85.2	64.1	79.2
9	TWS	1-1-4	16.5	1.5	✓	**83.6/82.9**	**89.0**	89.7	**93.1**	**47.9**	**82.9**	**86.6**	**65.8**	**80.2**

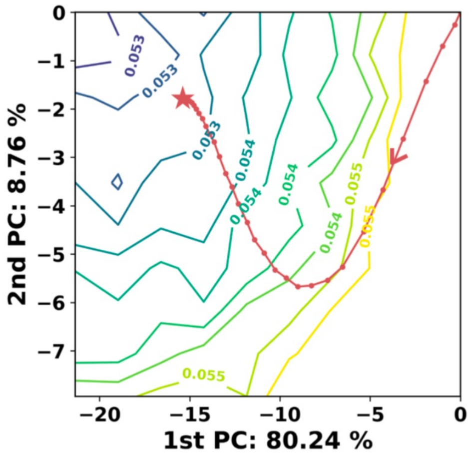
Experiments: More Results

- Compare with SOTA

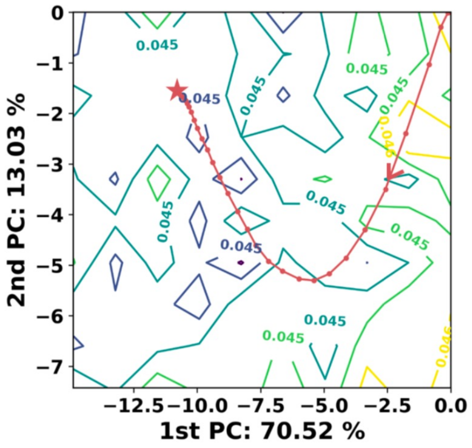
Table 4: Comparison with other state-of-the-art methods on development set of MNLI-m and SQuAD v1.1.

Method	#Bits (W-E-A)	Size (MB)	Ratio (↓)	MNLI -m	SQuAD v1.1
BERT-base	*full-prec.*	418	1.0	84.6	80.8/88.5
DistilBERT	*full-prec.*	250	1.7	81.6	79.1/86.9
LayerDrop-6L	*full-prec.*	328	1.3	82.9	-
LayerDrop-3L	*full-prec.*	224	1.9	78.6	-
TinyBERT-6L	*full-prec.*	55	7.6	82.8	79.7/87.5
ALBERT-E128	*full-prec.*	45	9.3	81.6	82.3/89.3
ALBERT-E768	*full-prec.*	120	3.5	82.0	81.5/88.6
Quant-Noise	PQ	11.0	38	83.6	-
Q-BERT	2/4-8-8	53	7.9	83.5	79.9/87.5
Q-BERT	2/3-8-8	46	9.1	81.8	79.3/87.0
Q-BERT	2-8-8	28	15.0	76.6	69.7/79.6
GOBO	3-4-32	43	9.7	83.7	-
GOBO	2-2-32	28	15.0	71.0	-
TernaryBERT	2-2-8	28	15.0	83.5	79.9/87.4
BinaryBERT	**1-1-8**	**17**	**24.6**	**84.2**	**80.8/88.3**
BinaryBERT	**1-1-4**	**17**	**24.6**	**83.9**	**79.3/87.2**

- Optimization trajectory after splitting
  - Follow (Li et.al, 2017)



(c) 8-bit Activation.



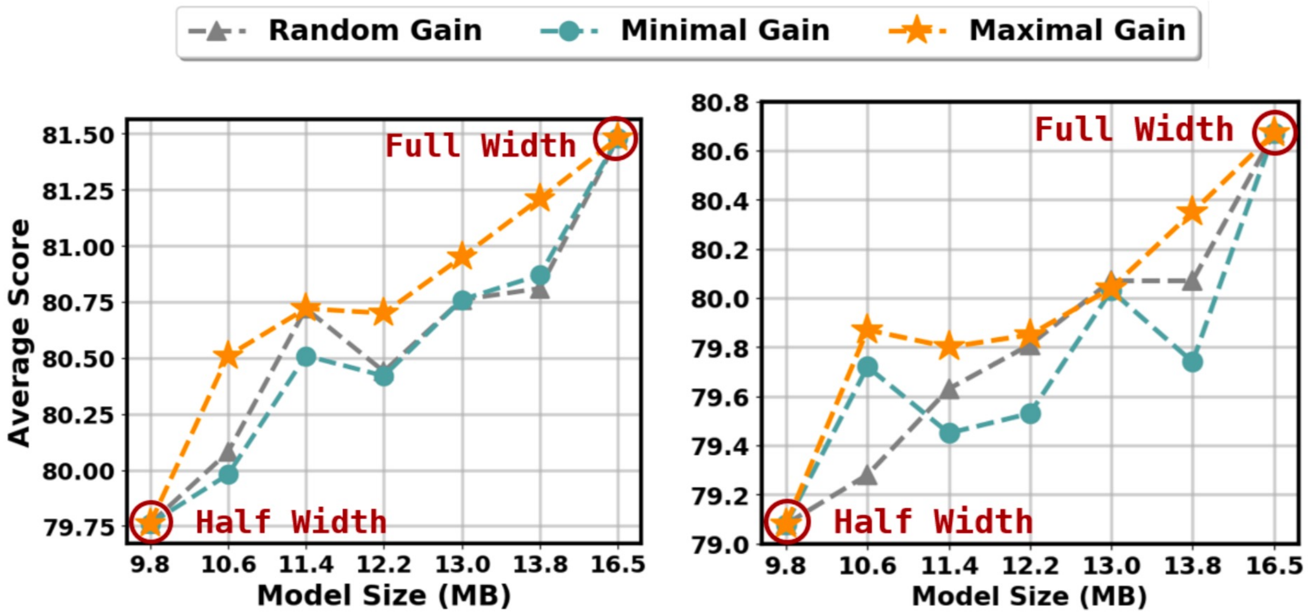
(d) 4-bit Activation.

Moving towards a better minima

Size reduction
418/17 = 24.5

Experiments: Adaptive Splitting Results

- **Maximal Gain**
  split the most sensitive
- Random Gain
  split in the random way
- Minimal Gain
  split the most insensitive



(a) 8-bit Activation. (b) 4-bit Activation.

Summary

- We find that directly training a BinaryBERT suffers from large performance drop due to the steep loss landscape issues
- We thus propose ternary weight splitting, by first training a ternaryBERT as the initialization of the full-sized BinaryBERT
- The proposed approach also supports adaptive splitting, which can flexibly adjust the model size depending on hardware constraints
- We achieve new state-of-the-art BERT quantization results, being 24x smaller in size with only 0.4% accuracy drop compared with the full precision model

Outline

Challenge 1: Network Compression with Limited Training Resources

1 Few-Shot Network Pruning via Cross Distillation (AAAI 2020)

2 Efficient Post-training Quantization for Pre-trained Language Models (In submission)

Challenge 2: Extreme Compression with Sharp Performance Drop

3 BinaryBERT: Pushing the Limit of BERT Quantization (ACL 2021)

Challenge 3: NAS Efficiency with Parameter Sharing

4 **Revisit Parameter Sharing for Automatic Neural Channel Number Search (NeurIPS 2020)**

Background: Reinforcement Learning based NAS

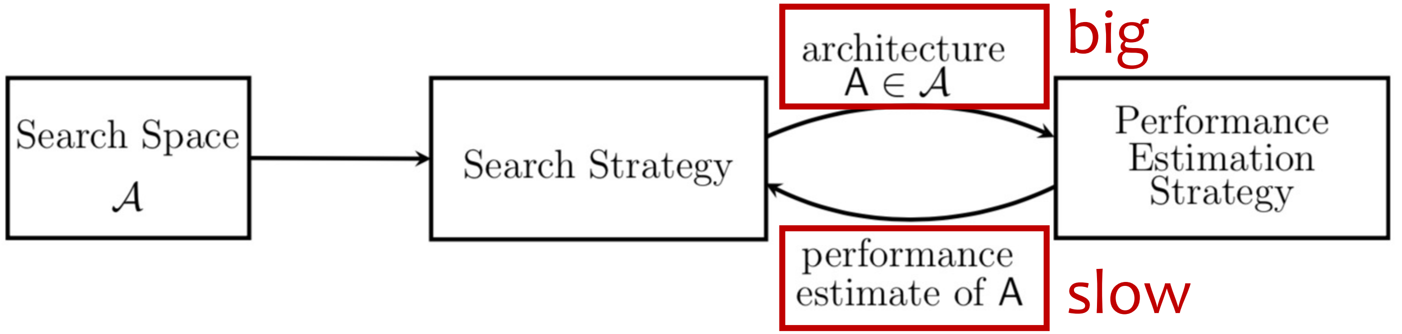
- Bi-level optimization problem
  - Inside: minimize the loss function w.r.t. candidate parameter $\mathbf{w}(a)$
  - Outside: level: maximize the reward function by policy gradient

$$\max_{\theta} J(\theta) = \mathrm{E}_{a\sim\pi_\theta}\mathcal{R}(a)\Big(a, \mathbf{w}^*(a)\Big),$$
$$\text{s.t. } \mathbf{w}^*(a) = \arg\min_{\mathbf{w}(a)} \mathcal{L}\Big(a, \mathbf{w}(a)\Big) \text{ and } \mathcal{B}\Big(a\Big) \leq B,$$

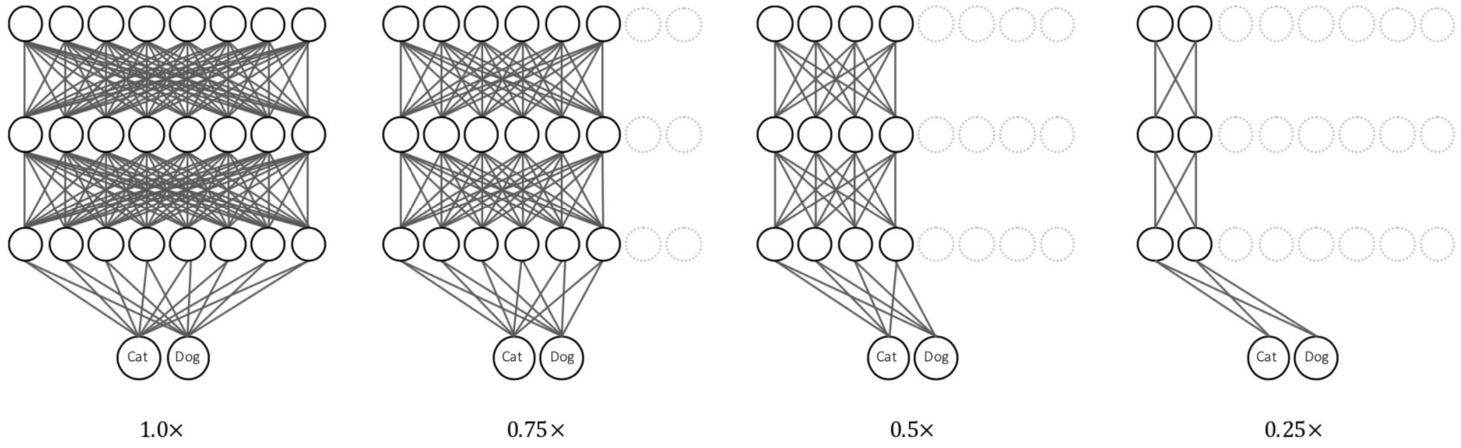
- Computationally intractible to compute $\mathbf{w}^*(\alpha)$ for evaluation
- Associating a with different $\mathbf{w}(a)$ make the supernet too large

Background: Parameter Sharing

- Recall the workflow of neural architecture search (Elsken et.al., 2019)



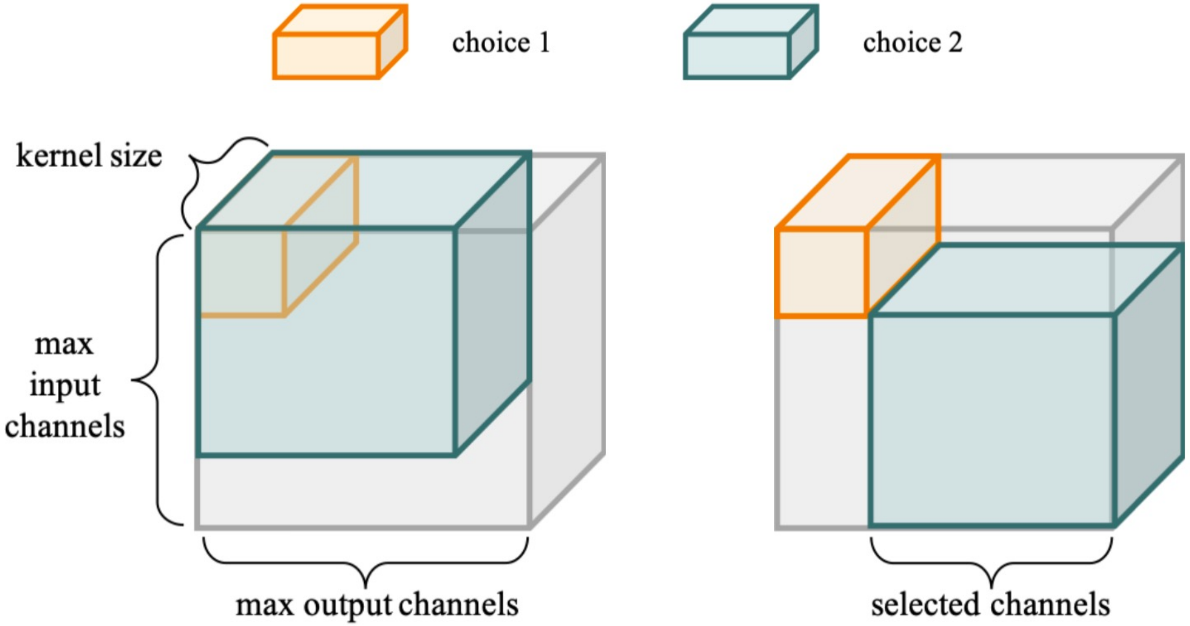
- Parameter sharing is widely used to improve the searching efficiency



Slimmable Net (Yu et.al., 2018)

Previous Parameter Sharing Schemes

- Summarization of previous parameter sharing schemes



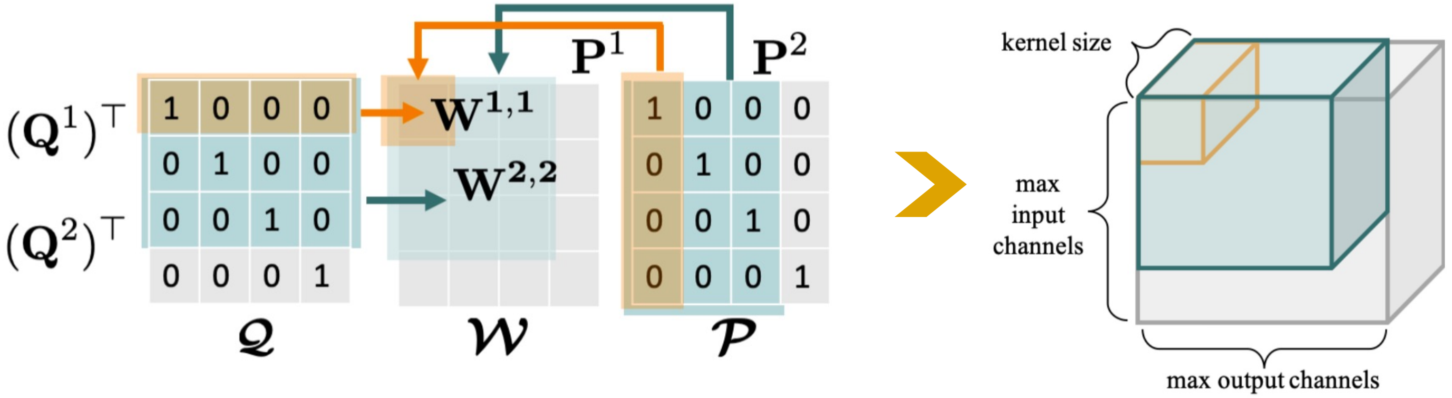
Ordinal selection | Independent selection

- We aim at a better understanding of parameter sharing in NAS

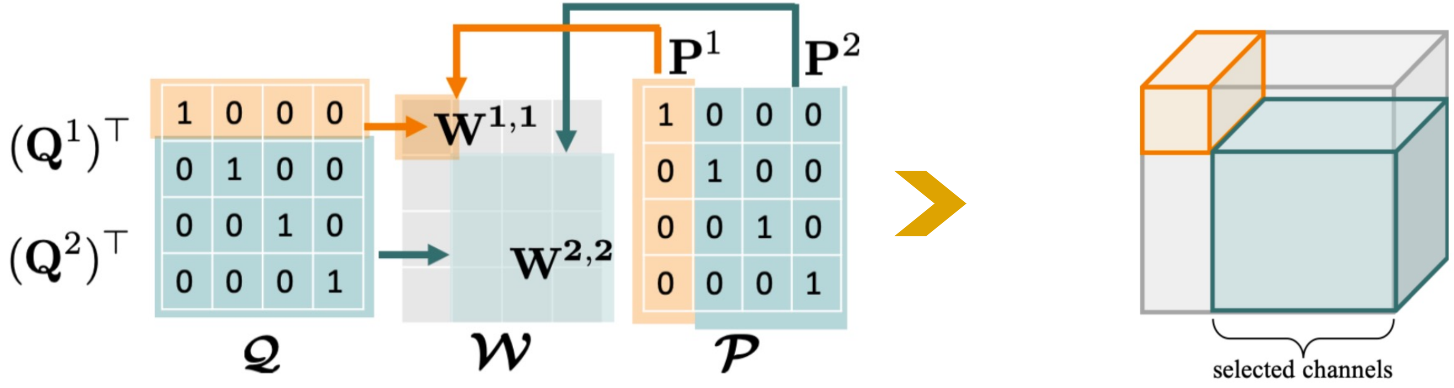
Methodology: Affine Parameter Sharing

- Parameter sharing can be achieved by **affine transformation**
- Meta weight $\mathcal{W}$, transformation matrices $\mathcal{P}$ $\mathcal{Q}$

$$\mathbf{W}^{i,o} = (\mathbf{Q}^{i})^{\top} \times_2 \mathcal{W} \times_1 \mathbf{P}^{o},$$



Ordinal selection



Independent selection

Methodology: Affine Parameter Sharing

- Quantitative measurement with affine parameter sharing

Definition

Definition 3.1. *Assuming each element of meta weight $\mathcal{W}$ follows the standard normal distribution, the **level of affine parameter sharing** is defined as the Frobenius norm of cross-covariance matrix*[2] *between candidate parameters $\mathbf{W}^{i,o}$ and $\mathbf{W}^{\tilde{i},\tilde{o}}$, i.e. $\phi(i,o;\tilde{i},\tilde{o}) = \| \operatorname{Cov}(\mathbf{W}^{i,o}, \mathbf{W}^{\tilde{i},\tilde{o}}) \|_F^2$.*

[2] The cross-covariance matrix between $\mathbf{X} \in \mathbb{R}^{m\times n}$ and $\mathbf{Y} \in \mathbb{R}^{\tilde{m}\times\tilde{n}}$ is defined as $\operatorname{Cov}(\mathbf{X},\mathbf{Y}) = \mathbb{E}[(\mathbf{X}-\mathbb{E}(\mathbf{X})) \otimes (\mathbf{Y}-\mathbb{E}(\mathbf{Y}))^\top] \in \mathbb{R}^{m\times n\times\tilde{m}\times\tilde{n}}$, where $\otimes$ is the Kronecker product.

Theorem

Theorem 3.1. *For $\forall i \le \tilde{i}$ and $\forall o \le \tilde{o}$, the overall level Φ of APS is maximized if $\mathcal{R}(\mathbf{Q}^i) \subseteq \mathcal{R}(\mathbf{Q}^{\tilde{i}})$ and $\mathcal{R}(\mathbf{P}^o) \subseteq \mathcal{R}(\mathbf{P}^{\tilde{o}})$. Φ is minimized if $\mathcal{R}(\mathbf{Q}^i) \subseteq \mathcal{R}^\perp(\mathbf{Q}^{\tilde{i}})$ and $\mathcal{R}(\mathbf{P}^o) \subseteq \mathcal{R}^\perp(\mathbf{P}^{\tilde{o}})$.*

- Ordinal selection: maximum
- Independent selection: minimum

Methodology: Parameter Sharing Effect

The two sides of parameter sharing

- Parameter sharing benefits efficient searching

$$\cos(\mathbf{g}, \tilde{\mathbf{g}}) = \frac{\mathbf{g}^\top \tilde{\mathbf{g}}}{\|\mathbf{g}\|_2 \cdot \|\tilde{\mathbf{g}}\|_2}, \quad \text{where} \ \mathbf{g} = \nabla_{\mathcal{W}} \mathcal{L}(\mathbf{W}^{i,o}), \ \tilde{\mathbf{g}} = \nabla_{\mathcal{W}} \mathcal{L}(\mathbf{W}^{\tilde{i},\tilde{o}})$$

A positive cosine value indicates a descent direction

- Parameter sharing couples architecture optimization

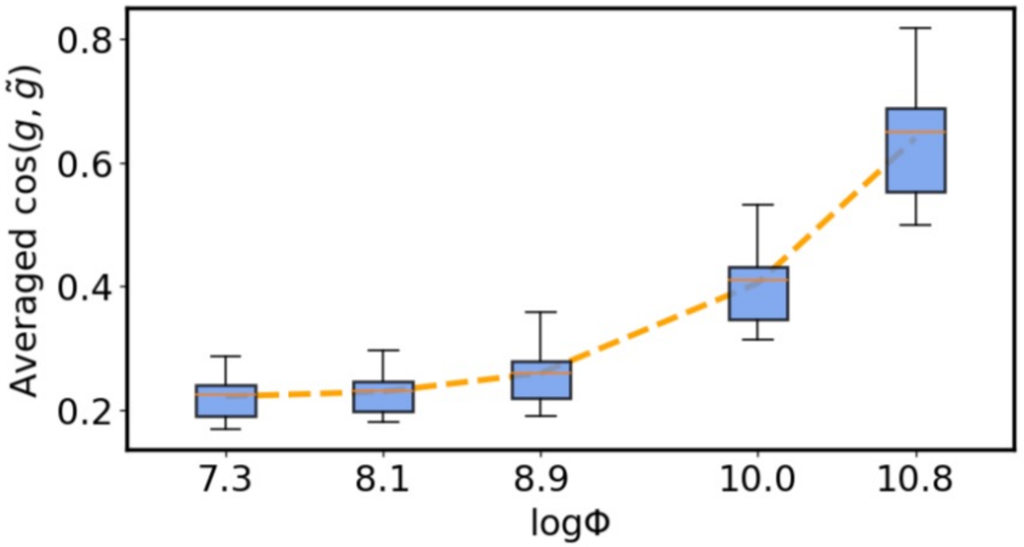
$$\mathbf{W}^t = (\mathbf{Q}^t)^\top \mathcal{W}^0 \mathbf{P}^t - \eta \underbrace{\sum_{\substack{i_{\tilde{t}} = i_t \\ o_{\tilde{t}} = o_t}} (\mathbf{Q}^t)^\top \mathbf{Q}^{\tilde{t}} (\nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}^{\tilde{t}})) (\mathbf{P}^{\tilde{t}})^\top \mathbf{P}^t}_{\text{Normal updates on the current candidate}} - \eta \underbrace{\sum_{\substack{i_{\tilde{t}} \neq i_t, \text{or} \\ o_{\tilde{t}} \neq o_t}} (\mathbf{Q}^t)^\top \mathbf{Q}^{\tilde{t}} (\nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}^{\tilde{t}})) (\mathbf{P}^{\tilde{t}})^\top \mathbf{P}^t}_{\text{Coupled updates from other candidates}}$$

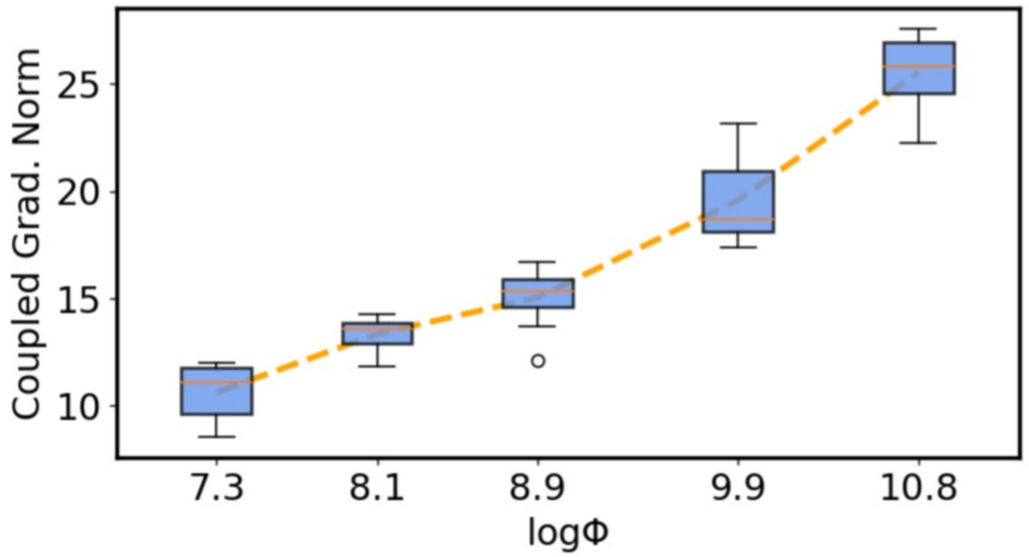
Methodology: Affine Parameter Sharing

- How does the parameter sharing level relate to the following aspects

$$\cos(\mathbf{g}, \tilde{\mathbf{g}}) = \frac{\mathbf{g}^\top \tilde{\mathbf{g}}}{\|\mathbf{g}\|_2 \cdot \|\tilde{\mathbf{g}}\|_2} \uparrow$$

$$\sum_{\substack{i_{\tilde{t}} \neq i_t, \text{or} \\ o_{\tilde{t}} \neq o_t}} (\mathbf{Q}^t)^\top \mathbf{Q}^{\tilde{t}} (\nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}^{\tilde{t}})) (\mathbf{P}^{\tilde{t}})^\top \mathbf{P}^t \uparrow$$





Methodology: Transitionary Affine Parameter Sharing

- A large cosine value benefits efficient training
- A large coupled gradient norm may bring less discriminative architectures
- Initialize Φ with maximum and gradually anneal it by:

$$\min_{\boldsymbol{\mathcal{P}},\boldsymbol{\mathcal{Q}}} \Phi \triangleq \sum_{i\leq\tilde{i}}\sum_{o\leq\tilde{o}} \left\| \mathrm{Cov}\left(\mathbf{W}^{i,o}, \mathbf{W}^{\tilde{i},\tilde{o}}\right) \right\|_F^2,$$

$$\text{s.t. } \left\|\mathbf{p}_x^o\right\|_2^2 = 1, \text{ for } x \in \{1, ..., c_o\} \text{ and } o \in \mathcal{A},$$

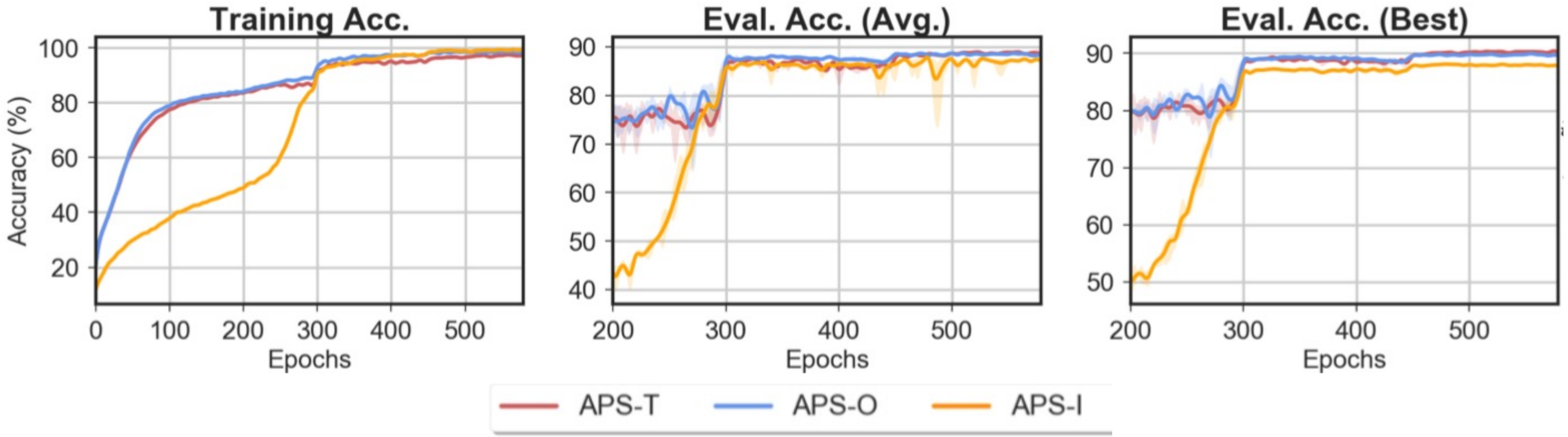
$$\left\|\mathbf{q}_y^i\right\|_2^2 = 1, \text{ for } y \in \{1, ..., c_i\} \text{ and } i \in \mathcal{A},$$

where in each update, we project them back to unit length:

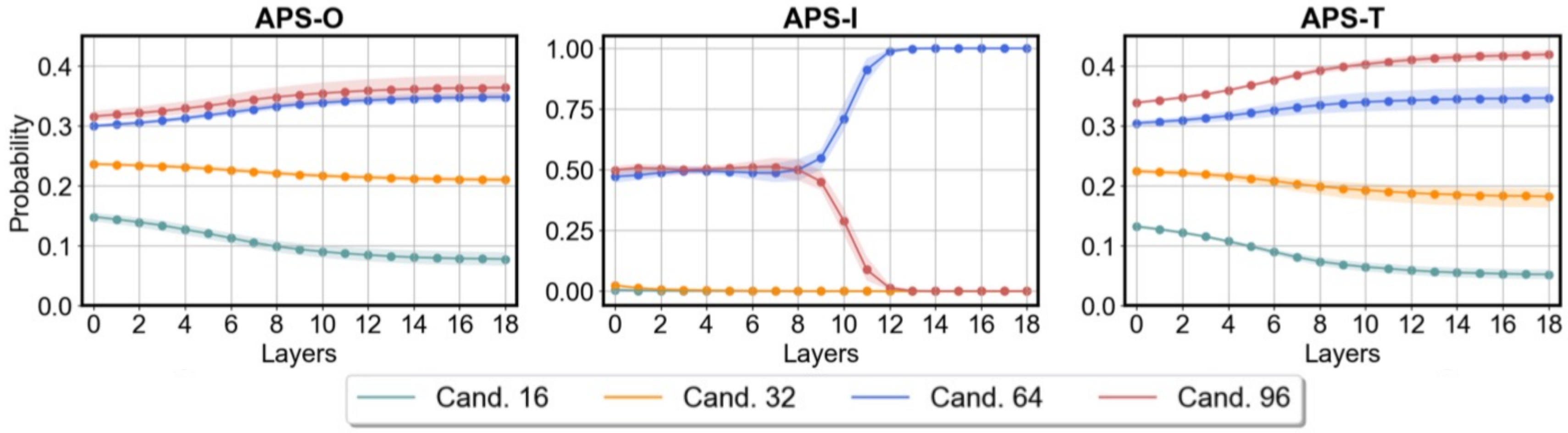
$$\mathbf{p}_x^o \leftarrow \Pi_{\mathcal{U}}(\mathbf{p}_x^o - \tau\nabla_{\mathbf{p}_x^o}\Phi), \ \mathbf{q}_y^i \leftarrow \Pi_{\mathcal{U}}(\mathbf{q}_y^i - \tau\nabla_{\mathbf{q}_y^i}\Phi)$$

Experiments: Effect of Parameter Sharing

- Efficient training



- Architecture discrimination

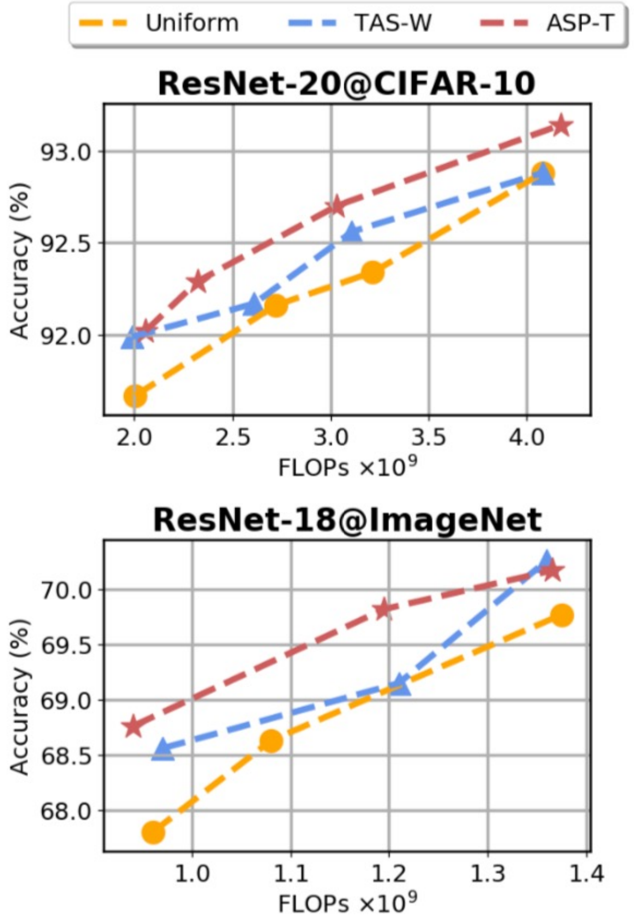


Experiments: Main Results

- ImageNet Results

Methods	Types	Top-1 Acc	Top-5 Acc	FLOPs	Ratio↓
Resnet-18 [5]	-	69.76%	89.08%	1.82G	0.0%
LCCL [1]	*HC*	66.33%	86.94%	1.19G	34.6%
SFP [6]	*HC*	67.10%	87.78%	1.06G	41.8%
FPGM [7]	*HC*	68.41%	88.48%	1.06G	41.8%
TAS [2]	*Auto*	69.15%	89.19%	1.21G	33.3%
APS-T	*Auto*	69.34%	88.89%	1.05G	41.8%
APS-T	*Auto*	70.17%	89.59%	1.36G	24.9%
APS-T	*Auto*	71.67%	90.36%	1.83G	-0.9%
MobileNet-V2 [19]	-	71.80%	91.00%	314M	0.0%
$\times$0.65 scaling	*HC*	67.20%	-	140M	55.4%
MetaPrune [15]	*Auto*	68.20%	-	140M	53.3%
MetaPrune [15]	*Auto*	72.70%	-	300M	4.4%
AutoSlim [25]	*Auto*	72.49%	90.50%	305M	2.9%
AutoSlim* [25]	*Auto*	74.20%	-	305M	2.9%
APS-T	*Auto*	68.96%	88.48%	156M	50.3%
APS-T	*Auto*	72.83%	90.75%	314 M	0.0%

- ACCs under varying FLOPs



Summary

- We propose affine parameter sharing as a general framework to unify previous hand-crafted parameter sharing heuristics
- We define a metric to qualitatively measure the parameter sharing level, and find it improves searching efficiency but at the cost of less architecture discrimination
- We thus design a transitionary parameter sharing strategy that balances searching efficiency and architecture discrimination, which can stably pick out the best architecture choices
- Extensive empirical results show that our searching algorithm outperforms a number of strong NAS baselines across different model sizes and architectures

Outline

Challenge 1: Network Compression with Limited Training Resources

1. Few Shot Network Pruning via Cross Distillation (AAAI 2020)
2. Efficient Post-training Quantization for Pre-trained Language Models (In submission)

Challenge 2: Extreme Compression with Sharp Performance Drop

3. BinaryBERT: Pushing the Limit of BERT Quantization (ACL 2021)

Challenge 3: NAS Efficiency with Parameter Sharing

4. Revisit Parameter Sharing for Automatic Neural Channel Number Search (NeurIPS 2020)

Future Work

- **Network compression**
  - Data unavailable: domain adaptation
  - Trillion-scale models
- **Neural architecture search**
  - Few-shot NAS: fast-training before evaluation
  - Refining the search space

Acknowledgement

- **My supervisors:**
  Prof. Michael R. Lyu and Prof. Irwin King
- **My committee members:**
  Prof. Laiwan Chan, Prof. Andrej Bogdanov and Prof. Hsuan-Tien Lin
- **Research collaborators**
- **Our research group members**
- **My parents and girlfriend**

Publication

* : equal contributions

- **Haoli Bai**, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael R. Lyu, Irwin King. BinaryBERT: Pushing the Limit of BERT Quantization. **ACL, 2021.**
- Xianghong Fang*, **Haoli Bai***, Jian Li, Zenglin Xu, Michael R. Lyu, Irwin King. Discrete Auto-regressive Variational Attention Models for Text Modeling. **IJCNN, 2021.**
- Jiaxing Wang*, **Haoli Bai**, Jiaxiang Wu, Xupeng Shi, Junzhou Huang, Irwin King, Michael R. Lyu, Jian Cheng. Revisiting Parameter Sharing for Automatic Neural Channel Number Search. **NeurIPS, 2020.**
- **Haoli Bai**, Jiaxiang Wu, Irwin King, Michael R. Lyu. Few Shot Network Compression via Cross Distillation. **AAAI, 2020.**
- **Haoli Bai**, Zhuangbin Chen, Irwin King, Michael R. Lyu, Zenglin Xu. Neural Relational Topic Models for Scientific Article Analysis. **CIKM, 2018.**

Preprints:

- **Haoli Bai**, Jiaxiang Wu, Mingyang Yi, Irwin King, Michael R. Lyu. Cross Distillation: A Unified Approach for Few-shot Network Compression. Under submission, 2021.
- **Haoli Bai**, Lu Hou, Lifeng Shang, Xin Jiang, Irwin King, Michael R. Lyu. Towards Efficient Post-training Quantization of Pre-trained Language Models. Under submission, 2021.
- Chung Yiu Yau, **Haoli Bai**, Michael R. Lyu, Irwin King. DAP-BERT: Differentiable Architecture Pruning of BERT. Under submission, 2021.

Thank you!

香港中文大學
The Chinese University of Hong Kong