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
Real-time AI Services

a) Object Detection

b) Machine Translation

c) Speech Recognition

d) Tumor Detection
The Increasing Model Size

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

a) Computer Vision Models  
(Bianco et.al., 2018)

b) Natural Language Processing Models  
(Sanh et.al., 2020)
Overview: Network Compression

- **Common methods**

  **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 1: access of training resources**

- ✓: Accessible on the user side
  - Data: privacy and security
  - GPU: quick deployment
- ✗: Restricted for the compression service

**Challenge 2: sharp performance drop**

Caused by extreme compression such as:
- Pruning: higher sparsity
- Quantization: lower bit-width (e.g., binarization)
Overview: Neural Architecture Search (NAS)

- **NAS components**

**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**
  
  ![Diagram of NAS pipeline]

  - Sample architecture: \( a = f(A) \)
  - Search space \( A \)
  - Search strategy \( f(\cdot) \)
  - Performance estimation
  - Optimal \( a^* \)
  - Re-training

  Architecture reward \( R(a) \)

  **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

Network Compression

- Pruning
  - Types: Unstructured, Stripe, Filter, Channel
  - Data Access: Full Data: CP, ThiNet, DCP, CCP
  - Few Data: FSKD, CURL
  - Quantization Training: QAT: DoReFa, PACT, LSQ
  - PTQ: Bit-split, AdaRound, BRECQ
  - Bit-width: m-bit: DoReFa, PACT, LSQ
  - 2-bit: TWN, TTQ, LAQ, RTN
  - 1-bit: BWN, BiReal, XNOR, ReActNet

Quantization

- Bit-width

Architecture Search

- Search space
  - Cell, Width/Depth, Compression: NAS, FBNet, Auto-slim, TAS
  - Differentiable, RL, Evolutionary: DARTS, ENAS, MetaPruning
  - Parameter sharing: NAS, ENAS, DARTS, TAS

Search Strategy

- Differentiable, RL, Evolutionary

Performance Estimation

- Parameter sharing

Efficient Deep Learning

Challenge 1: network compression with limited training resources

Challenge 2: extreme network compression with sharp performance drop

Challenge 3: NAS efficiency with parameter sharing

Ch3: AAAI 2020

Ch4: In submission

Ch5: ACL 2021

Ch6: NeurIPS 2020
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 $\mathbf{w} \in \mathbb{R}^{c_o \times c_i \times k \times k}$, 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

- Pruning criteria (by minimizing the loss change)

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

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

- **Typical paradigm** for network pruning

![Diagram showing the typical paradigm for network pruning]

- However, passing the training data can be risky
- **New paradigm:** few-shot network pruning (e.g., 5 images per class)
Prior Methods

- Pruning resembles knowledge distillation

\[ \mathcal{F}^T : \text{Teacher (original unpruned model)} \quad \mathcal{F}^S : \text{Student (pruned model)} \]

- Minimize the layer-wise Euclidean distance

  - Objective function

  \[
  w^*_S = \arg \min_{w^S} \frac{1}{N} \| w^T \ast h^T - w^S \ast h^S \|_F^2 + \lambda \mathcal{R}(w^S),
  \]

  - 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**
  \[ \epsilon^S = \| W^S \ast h^T - W^S \ast h^S \|_F^2 \]

**Imitation**

- **Motivation**
  Teacher becomes aware of the error accumulated on student

- **Teacher discrepancy**
  \[ \epsilon^T = \| W^T \ast h^S - W^T \ast h^T \|_F^2 \]
Our Approach: Cross Distillation

- **Correction**
  \[ \mathcal{L}^c(w^S) = \| w^T \ast h^T - w^S \ast h^T \|_F^2, \]

- **Imitation**
  \[ \mathcal{L}^i(w^S) = \| w^T \ast h^S - w^S \ast h^S \|_F^2, \]

- 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

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

\[ \tilde{\mathcal{L}}(w^S) = \| (w^T \ast \hat{h}^T) - (w^S \ast \hat{h}^S) \|_F^2, \]
Pruning with Regularization $\mathcal{R}(W^S)$

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

- Solve by proximal gradient descent:

  - Structured pruning: $\text{Prox}_{\lambda \|\cdot\|_2}(w^S_i) = \max(1 - \frac{\lambda}{\|w^S_i\|_2}, 0) \cdot w^S_i$
  
  - Unstructured pruning:

$$\text{Prox}_{\lambda \|\cdot\|_1}(W^S_{ijhw}) = \begin{cases} 
W^S_{ijhw} - \lambda & W^S_{ijhw} > \lambda \\
0 & |W^S_{ijhw}| \leq \lambda \\
W^S_{ijhw} + \lambda & W^S_{ijhw} < -\lambda
\end{cases}$$
Experimental Results: Structured Pruning

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

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

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

<table>
<thead>
<tr>
<th>Methods</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1-norm</td>
<td>0.5±0.00</td>
<td>0.5±0.00</td>
<td>0.5±0.00</td>
<td>0.5±0.00</td>
<td>0.5±0.00</td>
<td>0.5±0.00</td>
</tr>
<tr>
<td>BP</td>
<td>42.87±2.07</td>
<td>48.78±1.43</td>
<td>65.47±1.15</td>
<td>71.25±0.97</td>
<td>74.85±0.71</td>
<td>76.04±0.48</td>
</tr>
<tr>
<td>FitNet</td>
<td>52.66±2.93</td>
<td>57.09±2.14</td>
<td>76.59±1.45</td>
<td>80.14±1.23</td>
<td>82.27±0.70</td>
<td>83.14±0.51</td>
</tr>
<tr>
<td>w/o CD</td>
<td>78.73±1.78</td>
<td>83.29±1.12</td>
<td>85.04±0.93</td>
<td>85.36±0.61</td>
<td>85.21±0.41</td>
<td>85.49±0.46</td>
</tr>
<tr>
<td>CD</td>
<td>83.81±1.49</td>
<td>86.21±1.09</td>
<td>87.19±0.96</td>
<td>87.61±0.82</td>
<td>87.78±0.45</td>
<td>87.86±0.39</td>
</tr>
<tr>
<td>SCD</td>
<td>83.67±1.52</td>
<td>86.72±1.23</td>
<td>87.82±1.04</td>
<td>88.14±0.74</td>
<td>88.23±0.61</td>
<td>88.38±0.43</td>
</tr>
</tbody>
</table>
Experimental Results: Discussions

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

$$\text{Ratio} = \frac{L_{ours}}{L_{prev}} \left( \| w^T \cdot h^T - w^S \cdot h^S \|_F^2 \right)$$

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 $w$
  - Multi-bit quantization ($b$-bit):
    \[
    \hat{w} = Q_b(w) = s \cdot \Pi_{\Omega(b)}(w/s), \quad \Omega(b) = \{-2^{b-1}, ..., 0, ..., 2^{b-1} - 1\}
    \]
  - Ternarization (2-bit)
    \[
    \hat{w}_i^t = Q_2(w_i) = \begin{cases} 
    \alpha \cdot \text{sign}(w_i) & |w_i| \geq \Delta \\
    0 & |w_i| < \Delta
    \end{cases}
    \]
  - Binarization (1-bit)
    \[
    \hat{w}_i^b = Q_1(w_i) = \alpha \cdot \text{sign}(w_i).
    \]

- Quantization workflow
Background: Quantization

- Training
  - Quantization-aware training (QAT): **cross entropy over full data**
    \[
    \min_{w,s} E_{x \sim D} [\ell(x; \hat{w}, s)], \quad \text{s.t. } \hat{w} = Q_b(w).
    \]
  - Post-training quantization (PTQ): **reconstruction error over few data**
    \[
    \min_{w,s} \| \hat{w}^\top \hat{a} - w^\top a \|^2, \quad \text{s.t. } \hat{w} = Q_b(w). \quad \text{(Similar to layer-wise pruning)}
    \]

- Comparison

![Graphs showing performance metrics](image)
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

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

where \( \mathbf{f}_l \) and \( \hat{\mathbf{f}}_l \) 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: \[ \tilde{f}_{ln} = \lambda f_{ln} + (1 - \lambda) \hat{f}_{ln}, \quad \lambda \in [0, 1], \] (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)

<table>
<thead>
<tr>
<th>#Bits (W-E-A)</th>
<th>Quant Method</th>
<th>BERT-base</th>
<th>BERT-large</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (min)</td>
<td>Mem (G)</td>
<td># Data (K)</td>
</tr>
<tr>
<td>full-prec</td>
<td>220</td>
<td>8.6</td>
<td>393</td>
</tr>
<tr>
<td>4-4-8</td>
<td>QAT</td>
<td>1,320</td>
<td>11.9</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>28</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>MREM-S</td>
<td>36</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td>MREM-P</td>
<td>9</td>
<td>3.7</td>
</tr>
<tr>
<td>MNLI</td>
<td>QAT</td>
<td>882</td>
<td>11.9</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>24</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>MREM-S</td>
<td>24</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td>MREM-P</td>
<td>6 3.7×4</td>
<td>4</td>
</tr>
<tr>
<td>2-2-8</td>
<td>QAT</td>
<td>875</td>
<td>11.9</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>24</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>MREM-S</td>
<td>24</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td>MREM-P</td>
<td>6 3.7×4</td>
<td>4</td>
</tr>
</tbody>
</table>
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  |
|--------------|--------------|-----------|-----|--------|-----|------|-------|------|-------|------|-----|-----|-----|
| -            | 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        | X   | 76.6   | -   | -    | -     | 84.6 | -     | -    | -   | 77.3|
| Q-BERT      | 2/4-8-8      | 53        | X   | 83.5   | -   | -    | -     | 92.6 | -     | -    | -   | 86.0|
| Quant-Noise  | PQ           | 38        | X   | 83.6   | -   | -    | -     | -    | -     | -    | -   | 77.0|
| TernaryBERT  | 2-2-8        | 28        | X   | 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   | -   | -    | -     | -    | -     | -    | -   | 77.0|
| GOBO         | 2-2-32       | 28        | ✓   | 71.0   | -   | -    | -     | -    | -     | -    | -   | 77.0|
| MREM-S       | 4-4-8        | 50        | ✓   | 83.5±0.1| 90.2±0.1| 91.2±0.1| 91.4±0.4| 55.1±0.8| 89.1±0.1| 84.8±0.0| 71.8±0.0| 82.4±0.1|
| MREM-P       | 4-4-8        | 50        | ✓   | 83.4±0.1| 90.2±0.1| 91.0±0.2| 91.5±0.4| 54.7±0.9| 89.1±0.1| 86.3±0.0| 71.1±0.0| 82.2±0.1|
| MREM-P       | 2-2-8        | 28        | ✓   | 82.3±0.2| 89.4±0.1| 90.3±0.2| 91.3±0.4| 52.9±1.2| 88.3±0.2| 85.8±0.0| 72.9±0.0| 81.6±0.2|
Experiments: Effect of Teacher Forcing

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

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

![Graphs showing accuracy and vocabulary coverage for different tasks](image)
Background: Underlying Challenges

- Visualization of loss landscape

- Perturbation as follows:
  \[
  \tilde{w}_x = w_x + x \cdot 1_x, \quad \tilde{w}_y = w_y + y \cdot 1_y, 
  \]
  where \( \bar{w}_x \) is the average value of \( w_x \), and
  \[
  x \in \{ \pm 0.2\bar{w}_x, \pm 0.4\bar{w}_x, \ldots, \pm 1.0\bar{w}_x \} 
  \]
Background: Underlying Challenges

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

- Measuring the steepness of loss curvature

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

- $\mathbf{e} = \mathbf{w} - \hat{\mathbf{w}}$ 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

$$[\mathbf{w}_1^b]_i = \begin{cases} a \cdot w^t_i & \text{if } \hat{w}_i^t \neq 0 \\ b + w^t_i & \text{if } \hat{w}_i^t = 0, w^t_i > 0 \\ b & \text{otherwise} \end{cases}$$

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

- Next: solve $a$ and $b$
Methodology: Ternary Weight Split

- TWS allows closed-form solution as

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

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

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

\[
\begin{align*}
\max_s & \quad \mathbf{u}^\top \mathbf{s} \\
\text{s.t.} & \quad \mathbf{c}^\top \mathbf{s} \leq C - C_0, \quad \mathbf{s} \in \{0, 1\}^Z, \\
& \quad \mathbf{s} \in \{0, 1\}^Z
\end{align*}
\]

where \( C \) 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

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

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

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

Moving towards a better minima

Size reduction

\[
\frac{418}{17} = 24.5
\]
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

1. Few Shot Network Pruning via Cross Distillation (AAAI 2020)
2. Efficient Post-training Quantization for Pre-trained Language Models (In submission)
3. BinaryBERT: Pushing the Limit of BERT Quantization (ACL 2021)
4. Revisit Parameter Sharing for Automatic Neural Channel Number Search (NeurIPS 2020)

Challenge 1: Network Compression with Limited Training Resources
Challenge 2: Extreme Compression with Sharp Performance Drop
Challenge 3: NAS Efficiency with Parameter Sharing
Background: Reinforcement Learning based NAS

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

\[
\max_\theta J(\theta) = \mathbb{E}_{a \sim \pi_\theta} R(a)(a, w^*(a)),
\]
\[
s.t. \quad w^*(a) = \arg\min_{\mathcal{w}(a)} \mathcal{L}(a, \mathcal{w}(a)) \text{ and } \mathcal{B}(a) \leq B,
\]

- Computationally intractible to compute \( w^*(a) \) for evaluation
- Associating \( a \) with different \( w(a) \) 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

![Diagram showing the workflow of neural architecture search with parameter sharing.](image)
Previous Parameter Sharing Schemes

- Summarization of previous parameter sharing schemes

- 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 $\mathcal{P} \mathcal{Q}$

$$W_{i,o}^{i,o} = (Q_i^i)^\top \times_2 \mathcal{W} \times_1 P_o,$$

Ordinal selection

Independent selection
Methodology: Affine Parameter Sharing

- Quantitative measurement with affine parameter sharing

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

$\text{Cov}(\mathbf{X}, \mathbf{Y}) = \mathbb{E}[(\mathbf{X} - \mathbb{E}(\mathbf{X})) \otimes (\mathbf{Y} - \mathbb{E}(\mathbf{Y}))^\top] \in \mathbb{R}^{m \times n \times \tilde{m} \times \tilde{n}}$, where $\otimes$ is the Kronecker product.

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

- Ordinal selection: maximum
- Independent selection: minimum
Methodology: Parameter Sharing Effect

The two sides of parameter sharing

- Parameter sharing benefits efficient searching

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

A positive cosine value indicates a descent direction

- Parameter sharing couples architecture optimization

\[
W^t = (Q^t)^\top W^0 P^t - \eta \sum_{i_t = i_t, o_t = o_t} (Q^t)^\top Q^t \left( \nabla_W \mathcal{L}(W^{i,\tilde{o}}) \right) (P^{i,\tilde{o}})^\top P^t - \eta \sum_{i_t \neq i_t, or \ o_t \neq o_t} (Q^t)^\top Q^t \left( \nabla_W \mathcal{L}(W^{\tilde{i},\tilde{o}}) \right) (P^{\tilde{i},\tilde{o}})^\top P^t
\]

Normal updates on the current candidate

Coupled updates from other candidates
Methodology: Affine Parameter Sharing

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

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

$$\sum_{i \neq i_t, o_t} (Q^t) \top Q^i (\nabla_w \mathcal{L}(W^i)) (P^i) \top P^t$$

![Graphs showing the relationship between parameter sharing and logΦ](image-url)
Methodology: Transitionary Affine Parameter Sharing

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

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

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

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

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

$$\mathbf{p}_x^o \leftarrow \Pi_{\mathcal{U}} (\mathbf{p}_x^o - \tau \nabla_{\mathbf{p}_x^o} \Phi), \quad \mathbf{q}_y^i \leftarrow \Pi_{\mathcal{U}} (\mathbf{q}_y^i - \tau \nabla_{\mathbf{q}_y^i} \Phi)$$
Experiments: Effect of Parameter Sharing

- Efficient training

- Architecture discrimination
Experiments: Main Results

- **ImageNet Results**

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

- **ACCs under varying FLOPs**

![Graph showing ACCs under varying FLOPs](image)
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


Preprints:


* : equal contributions
Thank you!