On-Device Training Under 256KB Memory

Ji Lin*, Ligeng Zhu*, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, Song Han
MIT, MIT-IBM Watson AI Lab

On-Device Training Enables Customization and Continual Learning

- Security: Data never leaves devices, thus promises security and regularization.
- Customization: Models continually adapt to new data from the sensors.

Challenge: Training is Expensive for Edge

Sparse Layer/Tensor Update

- Conventionally, we update the full model for transfer learning.
- We find some layers are more important than others, so we can sparsely update.

Quantization-Aware Scaling (QAS)

- Re-scale gradients to help convergence with quantized training.

<table>
<thead>
<tr>
<th>Model</th>
<th>ProxylessNAS-Mobile</th>
<th>MbV2-w0.35</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak Mem (KB)</td>
<td>2.0x</td>
<td>2.9×</td>
</tr>
<tr>
<td>TTE kernels are 320 times smaller</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tiny Training Engine (TTE)

- (a) input model
- (b) compile-time autodiff
- (c) graph pruning
- (d) op scheduling
- (e) on-device training

Theoretical saving -> Measured saving:

1. Offload most workloads from runtime to compile-time
2. Compile-time autodiff enables more optimization opportunities.
3. Full support for sparse update schemes

Algorithm-System Co-Design Results

- TensorFlow (cloud) 256KB constraint
- PyTorch (cloud) 41.5 MB
- MNN (edge) 303 MB
- Tiny Training Engine 7.3x
- Quantization-aware scaling 2.0x
- Sparse layer/tensor update 8.8x
- Operator reordering 2.4x

- MobileNetV2-0.35, input size 1x3x128x128.
- Enable Tiny Training Under 256KB Memory on MCUs
- 1000x less memory than PyTorch and Tensorflow.

MCU training demo: shorturl.at/cswPQ
Source code available at github.com/mit-han-lab/tiny-training