NETWORK COMPRESSION

Hung-yi Lee 李宏毅
Smaller Model

Less parameters

Deploying ML models in resource-constrained environments

Lower latency, Privacy, etc.
Outline

• Network Pruning
• Knowledge Distillation
• Parameter Quantization
• Architecture Design
• Dynamic Computation

We will not talk about hard-ware solution today.
Network Pruning
Network can be pruned

• Networks are typically over-parameterized (there is significant redundant weights or neurons)
• Prune them!

(NIPS, 1989)
Network Pruning

- Importance of a weight:
  absolute values, life long ...

- Importance of a neuron:
  the number of times it wasn’t zero on a given data set ...

- After pruning, the accuracy will drop (hopefully not too much)

- Fine-tuning on training data for recover

- Don’t prune too much at once, or the network won’t recover.
Network Pruning - Practical Issue

- Weight pruning

The network architecture becomes irregular.

Prune some weights

Hard to implement, hard to speedup ......
Network Pruning - Practical Issue

- Weight pruning

Network Pruning - Practical Issue

- Neuron pruning

The network architecture is regular.

Prune some neurons

Easy to implement, easy to speedup ......
Why Pruning?

• How about simply train a smaller network?
• It is widely known that smaller network is more difficult to learn successfully.
  • Larger network is easier to optimize?
    https://www.youtube.com/watch?v=_VuWvQU MQVk
• Lottery Ticket Hypothesis
  https://arxiv.org/abs/1803.03635
Why Pruning?
Lottery Ticket Hypothesis

Train

Large Network

Sub-network

Sub-network

Train

Large Network

Sub-network

Sub-network

Win!

Win!
Why Pruning?
Lottery Ticket Hypothesis

Random init

Trained

Pruned

Random Init weights

Trained weight

Another random Init weights

Random init

Again

Random init

Original

Random init
Why Pruning?
Lottery Ticket Hypothesis

- Different pruning strategy

- “sign-ificance” of initial weights: Keeping the sign is critical
  0.9, 3.1, -9.1, 8.5 ...... → +α, +α, - α, +α ......

- Pruning weights from a network with random weights

  Weight Agnostic Neural Networks  https://arxiv.org/abs/1906.04358
Why Pruning?

- Rethinking the Value of Network Pruning

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Unpruned</th>
<th>Pruned Model</th>
<th>Fine-tuned</th>
<th>Scratch-E</th>
<th>Scratch-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>VGG-16</td>
<td>93.63 (±0.16)</td>
<td>VGG-16-A</td>
<td>93.41 (±0.12)</td>
<td>93.62 (±0.11)</td>
<td>93.78 (±0.15)</td>
</tr>
<tr>
<td></td>
<td>ResNet-56</td>
<td>93.14 (±0.12)</td>
<td>ResNet-56-A</td>
<td>92.97 (±0.17)</td>
<td>92.96 (±0.26)</td>
<td>93.09 (±0.14)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ResNet-56-B</td>
<td>92.67 (±0.14)</td>
<td>92.54 (±0.19)</td>
<td>93.05 (±0.18)</td>
</tr>
<tr>
<td></td>
<td>ResNet-110</td>
<td>93.14 (±0.24)</td>
<td>ResNet-110-A</td>
<td>93.14 (±0.16)</td>
<td>93.25 (±0.29)</td>
<td>93.22 (±0.22)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ResNet-110-B</td>
<td>92.69 (±0.09)</td>
<td>92.89 (±0.43)</td>
<td>93.60 (±0.25)</td>
</tr>
<tr>
<td>ImageNet</td>
<td>ResNet-34</td>
<td>73.31</td>
<td>ResNet-34-A</td>
<td>72.56</td>
<td>72.77</td>
<td>73.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ResNet-34-B</td>
<td>72.29</td>
<td>72.55</td>
<td>72.91</td>
</tr>
</tbody>
</table>

- **New** random initialization, not **original** random initialization in “Lottery Ticket Hypothesis”
- Limitation of “Lottery Ticket Hypothesis” (small lr, unstructured)

Knowledge Distillation
Knowledge Distillation

Providing the information that “1” is similar to “7”

Learning target

“1”: 0.7, “7”: 0.2, “9”: 0.1

Teacher Net (Large)

Student Net (Small)

Cross-entropy minimization

Knowledge Distillation

Do Deep Nets Really Need to be Deep?
Knowledge Distillation

Knowledge Distillation
Do Deep Nets Really Need to be Deep?

Learning target
“1”: 0.7, “7”: 0.2, “9”: 0.1

Cross-entropy minimization

Ensemble
Average many models

N Networks

Student Net (Small)
Knowledge Distillation

- Temperature for softmax

\[
y_i' = \frac{\exp(y_i)}{\sum_j \exp(y_j)} \quad \quad \quad T = 100
\]

\[
y_i' = \frac{\exp(y_i/T)}{\sum_j \exp(y_j/T)}
\]

\[
y_1 = 100 \quad \quad y_1' = 1
\]
\[
y_2 = 10 \quad \quad y_2' \approx 0
\]
\[
y_3 = 1 \quad \quad y_3' \approx 0
\]
\[
y_1/T = 1 \quad \quad y_1' = 0.56
\]
\[
y_2/T = 0.1 \quad \quad y_2' = 0.23
\]
\[
y_3/T = 0.01 \quad \quad y_3' = 0.21
\]
Parameter Quantization
Parameter Quantization

• 1. Using less bits to represent a value
• 2. Weight clustering

weights in a network

<table>
<thead>
<tr>
<th></th>
<th>0.5</th>
<th>1.3</th>
<th>4.3</th>
<th>-0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>-0.2</td>
<td>-1.2</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>3.0</td>
<td>-0.4</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>-0.5</td>
<td>-0.1</td>
<td>-3.4</td>
<td>-5.0</td>
<td></td>
</tr>
</tbody>
</table>
Parameter Quantization

1. Using less bits to represent a value
2. Weight clustering
3. Represent frequent clusters by less bits, represent rare clusters by more bits
   - e.g. Huffman encoding
Binary Weights

Your weights are always +1 or -1

- Binary Connect

network with binary weights

network with real value weights

- Negative gradient (compute on binary weights)
- Update direction (compute on real weights)
Binary Weights

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST</th>
<th>CIFAR-10</th>
<th>SVHN</th>
</tr>
</thead>
<tbody>
<tr>
<td>No regularizer</td>
<td>1.30 ± 0.04%</td>
<td>10.64%</td>
<td>2.44%</td>
</tr>
<tr>
<td>BinaryConnect (det.)</td>
<td>1.29 ± 0.08%</td>
<td>9.90%</td>
<td>2.30%</td>
</tr>
<tr>
<td>BinaryConnect (stoch.)</td>
<td>1.18 ± 0.04%</td>
<td><strong>8.27%</strong></td>
<td>2.15%</td>
</tr>
<tr>
<td>50% Dropout</td>
<td>1.01 ± 0.04%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

https://arxiv.org/abs/1511.00363
Architecture Design

Depthwise Separable Convolution
Review: Standard CNN

Input feature map

2 channels

$3 \times 3 \times 2 \times 4 = 72$ parameters
Depthwise Separable Convolution

1. Depthwise Convolution

- Filter number = Input channel number
- Each filter only considers one channel.
- The filters are $k \times k$ matrices
- There is no interaction between channels.
Depthwise Separable Convolution

1. Depthwise Convolution

\[ 3 \times 3 \times 2 = 18 \]

2. Pointwise Convolution

\[ 1 \times 1 \text{ filter} \]

\[ 2 \times 4 = 8 \]
$I$: number of input channels

$O$: number of output channels

$k \times k$: kernel size

\[
\frac{k \times k \times I + I \times O}{k \times k \times I \times O} = \frac{1}{O} + \frac{1}{k \times k} \quad (k \times k \times I + I \times O)
\]
Low rank approximation

\[ W \approx U V \]

Less parameters
To learn more ......

- SqueezeNet
- MobileNet
- ShuffleNet
- Xception
  - https://arxiv.org/abs/1610.02357
- GhostNet
Dynamic Computation
Dynamic Computation

• The network adjusts the computation it need.

Different devices

high/low battery

• Why don’t we prepare a set of models?
Dynamic Depth

\[ L = e_1 + e_2 + \cdots + e_L \]

Does it work well?

Multi-Scale Dense Network (MSDNet)
https://arxiv.org/abs/1703.09844
Dynamic Width

$L = e_1 + e_2 + e_3$

Simmable Neural Networks
https://arxiv.org/abs/1812.08928
Computation based on Sample Difficulty

- SkipNet: Learning Dynamic Routing in Convolutional Networks
- Runtime Neural Pruning
- BlockDrop: Dynamic Inference Paths in Residual Networks
Concluding Remarks

- Network Pruning
- Knowledge Distillation
- Parameter Quantization
- Architecture Design
- Dynamic Computation