Compressing Neural Networks Structures

Spring 2018
The University of Iowa

Zhe Li
Content

- Motivation
- Where to focus on to compress
- Different compressing methods
- SEP-Net (My Research Work)
- Experimental Results
Motivation

Example: Image classification

Accuracy on Image Classification

Accuracy (%)

Shallow Learning
Deep learning

Accuracy: Test on ImageNet Benchmark Dataset (about 1.2 million images, 1000 classes)
Motivation

Example: Image classification

Model size too large to be deployed to mobile phone or embedded system

\[ \text{model size} = \# \text{ of parameters} \times 4 \text{ Bytes} \]
Motivation

- Can we reduce model size without sacrificing performance loss for mobile phone or embedded application?
  - Reducing memory usage in neural network.

- If so, how?
  - Locating where memory cost are from.
  - Applying different techniques (group-wise convolution, quantization, pruning, etc.) to reduce memory usage.
Content

- Motivation
- Where to focus on to compress
- Different compressing methods
- SEP-Net (My Research Work)
- Experimental Results
Locating parameters - History View

- AlexNet (2012 ImageNet Competition Winner)
  - Breakthrough work.
  - Proposed and implemented in University of Toronto
  - Based on open source library CudaConvNet
  - Existing open source libraries such as Caffe, Tensorflow and others follows some design principle of CudaConvNet.
Locating parameters - History View

- **AlexNet**

![Diagram of AlexNet architecture](https://via.placeholder.com/150)

- **Highlights**
  
  Input: 3*224*224 size image
  
  Output: 1000 size vector, using one index to indicate specific class
  
  Structure: 5 convolution layers, 3 fully connected layers
  
  Special feature: Two GPU cards are used

---

Locating parameters - History View

- **AlexNet**

- **Compute # of parameters (first convolution layer)**
  - 96 filters of size 11*11*3 (3 colors of 11*11 window)
  - \((11\times11\times3 + 1)\times96 = 34944\)
Locating parameters - History View

- **AlexNet**

<table>
<thead>
<tr>
<th>Layer Type</th>
<th># Parameters</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv</td>
<td>3.7M</td>
<td>6%</td>
</tr>
<tr>
<td>FC</td>
<td>58.6M</td>
<td>94%</td>
</tr>
</tbody>
</table>

Questions: Which layer should we focus on for reducing # of parameters?
Locating parameters - History View

- VGG-Net (2013 ImageNet Competition Winner)

- 13 layers convolution layers
- 3 layers fully connected layers
- Researchers were focusing on improving on accuracy
  - CIFAR-10: validation accuracy of 93.56%
  - CIFAR-100: validation accuracy of 70.48%.

Locating parameters - History View

- VGG-Net (2013 ImageNet Competition Winner)
Locating parameters - History View

VGG-Net (2013 ImageNet Competition Winner)

<table>
<thead>
<tr>
<th>Layer</th>
<th>Type</th>
<th># Params</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv1</td>
<td>1.7K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv2</td>
<td>36.8K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv3</td>
<td>73.7K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv4</td>
<td>147.4K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv5</td>
<td>294.9K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv13</td>
<td>2.4M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC14</td>
<td>102.77M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC15</td>
<td>16.7M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC16</td>
<td>4.1M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>138M</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Layer type # Params Percent
Conv 14.4M 10.4
FC 123.6M 89.6

Questions: Which layer should we focus on for reducing # of parameters?
Locating parameters - History View

- GoogleNet (2014 ImageNet Competition Winner)

- Design base module: **inception module**
- **Multiple path** compared with previous network structure

Locating parameters - History View

- GoogleNet (2014 ImageNet Competition Winner)
- 9 inception modules
- 2 separable losses function
- Remove fully connected layers
- Model size reduce to 50MB
Locating parameters - History View

- ResNet (2015 ImageNet Competition Winner)

More layers are not necessary to give better performance!

Locating parameters - History View

- ResNet (2015 ImageNet Competition Winner)
  - Designed so-called residual block

![Residual Block Diagram]

A residual block

\[ F(x) + x \]
Locating parameters - History View

- ResNet (2015 ImageNet Competition Winner)
  - 152 convolutional layers and no fully connected layer

Skip layer

96.6% test accuracy, 200MB Model size
Summary from History

- **AlexNet → VGG-Net → GoogleNet → ResNet**
  - Test accuracy are increasing to plateau (96.6%)
  - Reducing model size by removing fully connected layer
  - Still need to reduce model size but without losing performance

Questions: Which layer should we focus on for reducing # of parameters?
Content

- Motivation
- Where to focus on to compress
- **Compressing for Convolutional layer**
- SEP-Net (My Research Work)
- Experimental Results
Targeting on convolution layer

- Group-wise convolution

Why can we do this?

Take an example:
- Input feature map: 16 x 48 x 48
- Output feature map: 32 x 48 x 48
- Filter size: 3 x 3
- Padding and stride: 1, 1

Original convolution:
- 16 X 11 X 11 X 32

Group wise convolution: (4 group)
- 16/4 x 11 x 11 x 32

Reducing # of parameters in convolution k times (k group convolution)
Targeting on convolution layer

- Depth-wise separable convolution

Why can we do this?

https://www.slideshare.net/DongWonShin4/depthwise-separable-convolution
Targeting on convolution layer

- Point wise convolution

(c) $1 \times 1$ Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

$1 \times 1$ filters serve as data transformation
Weight Pruning

- Magnitude-based Method: Iterative Pruning + Retraining

Weight Pruning Implementation

- Before pruning: $w_{ij}$ in $W[i, j]$
- After pruning: $w_{ij}$ in list $L_i = \{ (j, w_{ij}) : j \geq 0, w_{ij} > \varepsilon \}$
- To reduce the memory, $j$ can be replaced by a three-bits integer to store the next available position.
- Example: Instead of $L = \{ (1, 3.4), (4, 0.9), (15, 1.7) \}$, we use $M = \{ (1, 3.4), (3, 0.9), (8, 0), (3, 1.7) \}$
Weight Pruning

1. Choose a neural network architecture
2. Train this network to converge
3. Pruning the weights based on threshold
4. Train the network to converge
# Weight Pruning

## Experiment Results

<table>
<thead>
<tr>
<th>Network</th>
<th>Top-1 Error</th>
<th>Top-5 Error</th>
<th># of Params</th>
<th>Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet-300-100 Ref</td>
<td>1.64%</td>
<td>-</td>
<td>267K</td>
<td>12X</td>
</tr>
<tr>
<td>LeNet-300-100 Pruned</td>
<td>1.59%</td>
<td>-</td>
<td>22K</td>
<td></td>
</tr>
<tr>
<td>LeNet-5 Ref</td>
<td>0.80%</td>
<td>-</td>
<td>431K</td>
<td>12X</td>
</tr>
<tr>
<td>LeNet-5 Pruned</td>
<td>0.77%</td>
<td>-</td>
<td>36K</td>
<td></td>
</tr>
<tr>
<td>AlexNet Ref</td>
<td>42.78%</td>
<td>19.73%</td>
<td>61M</td>
<td>9X</td>
</tr>
<tr>
<td>AlexNet Pruned</td>
<td>42.77%</td>
<td>19.67%</td>
<td>6.7M</td>
<td></td>
</tr>
<tr>
<td>VGG-16 Ref</td>
<td>31.50%</td>
<td>11.32%</td>
<td>138M</td>
<td>13X</td>
</tr>
<tr>
<td>VGG-16 Pruned</td>
<td>31.34%</td>
<td>10.88%</td>
<td>10.3M</td>
<td></td>
</tr>
</tbody>
</table>

Content

- Motivation
- Where to focus on to compress
- Compressing for Convolutional layer
- SEP-Net (My Research Work)
- Experimental Results
Targeting on convolution layer

- Quantization

General case

Quantization we used for simple purpose
Pattern Binarization (Quantization)

- $k \times k$ ($k > 1$) filters serve as spatial pattern extraction.
- $1 \times 1$ filters serve as data transformation.
- Reduced memory usage of model dramatically.

<table>
<thead>
<tr>
<th>-0.0219</th>
<th>0.0408</th>
<th>-0.0547</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0855</td>
<td>0.0478</td>
<td>-0.0510</td>
</tr>
<tr>
<td>-0.0105</td>
<td>0.0924</td>
<td>-0.0126</td>
</tr>
</tbody>
</table>

A trained 3 x 3 filter from GoogleNet (left) and its binarized filter (right)
Pattern Binarization (Quantization)

How much can we reduce memory usage by the above?

Before: 9 parameters $\rightarrow$ 9 x 4 Bytes
After: 9 parameters $\rightarrow$ 9 bits
How to use pattern binarization?

- Easily adopted to any successful networks structures such as GoogleNet, ResNet as following procedure:

1. Train a neural network
2. Binarize 3x3 convolution filters
3. Fine tune Dnn model
Implementation

- Keep binarization filter same during fine-tuning.

**How?**

Forward to compute loss

- Set learning rate as 0

Backward to update gradient

\[ \mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \ell(\text{prediction}, \text{truth}) \]
Pattern Residual Block

- Features maps from 1 x 1 and k x k convolutions are added together.
- Adding 1 x 1 convolutions offsets the change incurred by the binarization.
SEP-Net Module

- 1 x 1 convolution layers: dimension reduction.
- 2 PRB blocks with different output channels.
- 1 x 1 convolution layer: dimension recovery.
- Skip connection.
The Proposed SEP-Net Structure

- Proposed SEP-Net for mobile/embedded devices.
- The designed SEP-Net has 1.3M parameters (5.2MB)
Content

- Motivation
- Where to focus on to compress
- Compressing for Convolutional layer
- SEP-Net (My Research Work)
- Experimental Results
Experimental Results

- 1000 classes image recognition

<table>
<thead>
<tr>
<th>model</th>
<th>Original Model</th>
<th>After Binarization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Parameter</td>
<td>Top1-Top5</td>
</tr>
<tr>
<td>GoolgeNet</td>
<td>6.99M</td>
<td>0.6865 0.8891</td>
</tr>
<tr>
<td>C-Inception</td>
<td>5.10M</td>
<td>0.6480 0.8630</td>
</tr>
</tbody>
</table>
Experimental Results

The designed SEP-Net

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameter Number</th>
<th>Size (bytes)</th>
<th>Top-1 Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNet</td>
<td>1.3M</td>
<td>5.2MB</td>
<td>0.637</td>
</tr>
<tr>
<td></td>
<td>2.6M</td>
<td>10.4MB</td>
<td>0.684</td>
</tr>
<tr>
<td>SEP-Net-R</td>
<td>1.3M (small)</td>
<td>5.2MB</td>
<td>0.658</td>
</tr>
<tr>
<td></td>
<td>1.7M (large)</td>
<td>6.7MB</td>
<td>0.667</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>1.2M</td>
<td>4.8MB</td>
<td>0.604</td>
</tr>
<tr>
<td>MobileNet</td>
<td>1.3M</td>
<td>5.2MB</td>
<td>0.637</td>
</tr>
<tr>
<td>SEP-Net-R (small)</td>
<td>1.3M</td>
<td>5.2MB</td>
<td>0.658</td>
</tr>
<tr>
<td>SEP-Net-B (small)</td>
<td>1.1M</td>
<td>4.2MB</td>
<td>0.637</td>
</tr>
<tr>
<td>SEP-Net-BQ (small)</td>
<td>1.1M</td>
<td>1.3MB</td>
<td>0.635</td>
</tr>
</tbody>
</table>

SEP-Net-R: SEP-Net with raw valued weights
SEP-Net-B: SEP-Net with pattern binarization
SEP-Net-BQ: SEP-Net with pattern binarization and other weights quantized using linear quantization with 8 bits
Summary

- From AlexNet to VGG-Net, GoogleNet, ResNet, test accuracy is steadily increasing to plateau
- Reducing model size by removing FC layers
- Further Reducing model size by targeting on convolution layer through group-wise convolution, depth separable convolution, quantization, pruning etc.