Model Compression: Weights Pruning for RNNs

Luyu Wang
University of Waterloo & RBC Research Institute
July 17, 2017
Outlines

• Introduction to Model Compression
• RNN Model Compression via Weights Pruning
• Experimental Results
• Conclusion
The Need for Model Compression

- **Success of Deep Neural Network Models**
- **Vast datasets, GPU for training**
- Production environment
- Deep neural nets need lots of computational to make inferences
The Need for Model Compression

- Success of Deep Neural Network Models
- Vast datasets, GPU for training
- Production environment
- Deep neural nets need lots of computational to make inferences
The Need for Model Compression

- Success of Deep Neural Network Models
- Vast datasets, GPU for training
- Production environment
- Deep neural nets need lots of computational to make inferences
The Need for Model Compression

- Success of Deep Neural Network Models
- Vast datasets, GPU for training
- Production environment
- Deep neural nets need lots of computational to make inferences

Question: less cost for deployed models?
Methods for Model Compression

- Weights quantization (or even binarization)
- Model distillation
- Weights pruning
Weights Pruning

- Learn the connectivity via normal network training
- Prune the low-weight connections
- Retrain the sparse network

## Weights Pruning

<table>
<thead>
<tr>
<th>Network</th>
<th>Top-1 Error</th>
<th>Top-5 Error</th>
<th>Parameters</th>
<th>Compression Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet-300-100 Ref</td>
<td>1.64%</td>
<td>-</td>
<td>267K</td>
<td></td>
</tr>
<tr>
<td>LeNet-300-100 Pruned</td>
<td>1.59%</td>
<td>-</td>
<td>22K</td>
<td>12×</td>
</tr>
<tr>
<td>LeNet-5 Ref</td>
<td>0.80%</td>
<td>-</td>
<td>431K</td>
<td></td>
</tr>
<tr>
<td>LeNet-5 Pruned</td>
<td>0.77%</td>
<td>-</td>
<td>36K</td>
<td>12×</td>
</tr>
<tr>
<td>AlexNet Ref</td>
<td>42.78%</td>
<td>19.73%</td>
<td>61M</td>
<td></td>
</tr>
<tr>
<td>AlexNet Pruned</td>
<td>42.77%</td>
<td>19.67%</td>
<td>6.7M</td>
<td>9×</td>
</tr>
<tr>
<td>VGG16 Ref</td>
<td>31.50%</td>
<td>11.32%</td>
<td>138M</td>
<td></td>
</tr>
<tr>
<td>VGG16 Pruned</td>
<td>31.34%</td>
<td>10.88%</td>
<td>10.3M</td>
<td>13×</td>
</tr>
</tbody>
</table>
Weights Pruning

AlexNet

VGG
Neural Machine Translation Models via Pruning

Pruning RNNs

- Vanilla RNN
- LSTM
- GRU

- MNIST Dataset 28x28
- Hidden states = 128
- RMSprop with fixed learning rate 0.001
- PyTorch on AWS GPU server
Pruning RNNs

Pre-train
Pruning RNNs

Pruning
Pruning RNNs

Re-train
Pruning RNNs

- Retraining closes the gap
- 95% pruned - accuracy loss smaller than 1%
- LSTM and GRU more resilient to pruning - additional redundancy in gates
Pruning RNNs

- Retraining closes the gap
- 95% pruned - accuracy loss smaller than 1%
- LSTM and GRU more resilient to pruning - additional redundancy in gates
Pruning RNNs

- Retraining closes the gap
- 95% pruned - accuracy loss smaller than 1%
- LSTM and GRU more resilient to pruning - additional redundancy in gates
- More layers - more can be pruned for LSTM & GRU (RNN?)
1 Layer RNN - 95% Pruned

- Input to hidden
- Hidden to hidden
- Hidden to output

Remaining Parameters
Pruned Parameters

Hidden to hidden more redundancy

\[
\begin{align*}
    a^{(t)} &= b + Wh^{(t-1)} + Ux^{(t)} \\
    h^{(t)} &= \psi(a^{(t)}) \\
    o^{(t)} &= c + Vh^{(t)} \\
    \hat{y}^{(t)} &= \phi(o^{(t)})
\end{align*}
\]
1 Layer LSTM - 95% Pruned

Input Gate

Input to hidden - input gate
Input to hidden - forget gate
Input to hidden - cell gate
Input to hidden - output gate
hidden to hidden - input gate
hidden to hidden - forget gate
hidden to hidden - cell gate
hidden to hidden - output gate
hidden to output

Hidden to hidden more redundancy

Remaining Parameters | Pruned Parameters

\[ i^{(t)} = \sigma (b^i + W^i [h^{(t-1)}, x^{(t)}]) \]
\[ g^{(t)} = \tanh (b^g + W^g [h^{(t-1)}, x^{(t)}]) \]
\[ f^{(t)} = \sigma (b^f + W^f [h^{(t-1)}, x^{(t)}]) \]
\[ c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot g^{(t)} \]
\[ o^{(t)} = \sigma (b^o + W^o [h^{(t-1)}, x^{(t)}]) \]
\[ h^{(t)} = o^{(t)} \odot \tanh (c^{(t)}) \]
1 Layer LSTM - 95% Pruned

Cell Gate

- Input to hidden - input gate
- Input to hidden - forget gate
- Input to hidden - cell gate
- Input to hidden - output gate
- Hidden to hidden - input gate
- Hidden to hidden - forget gate
- Hidden to hidden - cell gate
- Hidden to hidden - output gate
- Hidden to output

Remaining Parameters | Pruned Parameters

Hidden to hidden more redundancy

\[
\begin{align*}
  i^{(t)} &= \sigma \left( b^i + W^i \left[ h^{(t-1)}, x^{(t)} \right] \right) \\
  g^{(t)} &= \tanh \left( b^g + W^g \left[ h^{(t-1)}, x^{(t)} \right] \right) \\
  f^{(t)} &= \sigma \left( b^f + W^f \left[ h^{(t-1)}, x^{(t)} \right] \right) \\
  c^{(t)} &= f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot g^{(t)} \\
  o^{(t)} &= \sigma \left( b^o + W^o \left[ h^{(t-1)}, x^{(t)} \right] \right) \\
  h^{(t)} &= o^{(t)} \odot \tanh(c^{(t)})
\end{align*}
\]
1 Layer LSTM - 95% Pruned

Forget Gate

Input to hidden - input gate
Input to hidden - forget gate
Input to hidden - cell gate
Input to hidden - output gate
Hidden to hidden - input gate
Hidden to hidden - forget gate
Hidden to hidden - cell gate
Hidden to hidden - output gate
Hidden to output

0% 15% 30% 45% 60% 75% 90%

Remaining Parameters
Pruned Parameters

Hidden to hidden more redundancy

\[
\begin{align*}
    i^{(t)} &= \sigma \left( b^i + W^i \left[ h^{(t-1)}, x^{(t)} \right] \right) \\
    g^{(t)} &= \tanh \left( b^g + W^g \left[ h^{(t-1)}, x^{(t)} \right] \right) \\
    f^{(t)} &= \sigma \left( b^f + W^f \left[ h^{(t-1)}, x^{(t)} \right] \right) \\
    c^{(t)} &= f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot g^{(t)} \\
    o^{(t)} &= \sigma \left( b^o + W^o \left[ h^{(t-1)}, x^{(t)} \right] \right) \\
    h^{(t)} &= o^{(t)} \odot \tanh(c^{(t)})
\end{align*}
\]
1 Layer LSTM - 95% Pruned
Output Gate

Remaining Parameters  Pruned Parameters

Input to hidden - input gate
Input to hidden - forget gate
Input to hidden - cell gate
Input to hidden - output gate
hidden to hidden - input gate
hidden to hidden - forget gate
hidden to hidden - cell gate
hidden to hidden - output gate
hidden to output

Hidden to hidden more redundancy

\[
i^{(t)} = \sigma(b^i + W^i[h^{(t-1)}, x^{(t)}])
\]
\[
g^{(t)} = \tanh(b^g + W^g[h^{(t-1)}, x^{(t)}])
\]
\[
f^{(t)} = \sigma(b^f + W^f[h^{(t-1)}, x^{(t)}])
\]
\[
c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot g^{(t)}
\]
\[
o^{(t)} = \sigma(b^o + W^o[h^{(t-1)}, x^{(t)}])
\]
\[
h^{(t)} = o^{(t)} \odot \tanh(c^{(t)})
\]
1 Layer GRU - 95% Pruned

Input Gate

Hidden to hidden more redundancy
Input gate less redundancy

\[
\begin{align*}
    z^{(t)} &= \sigma \left( b^z + W^z \left[ h^{(t-1)}, x^{(t)} \right] \right) \\
    r^{(t)} &= \sigma \left( b^r + W^r \left[ h^{(t-1)}, x^{(t)} \right] \right) \\
    g^{(t)} &= \tanh \left( b^g + W^g \left[ r^{(t)} \odot h^{(t-1)}, x^{(t)} \right] \right) \\
    h^{(t)} &= (1 - z^{(t)}) \odot h^{(t-1)} + z^{(t)} \odot g^{(t)}
\end{align*}
\]
1 Layer GRU - 95% Pruned Reset Gate

Hidden to hidden more redundancy
Input gate less redundancy
1 Layer GRU - 95% Pruned
Target State

- Input to hidden - reset gate
- Input to hidden - input gate
- Input to hidden - target state
- Hidden to hidden - reset gate
- Hidden to hidden - input gate
- Hidden to hidden - target state
- Hidden to output

Remaining Parameters
Pruned Parameters

Hidden to hidden more redundancy
Input gate less redundancy

\[
\begin{align*}
z(t) &= \sigma \left( b^z + W^z \left[ h^{(t-1)} , x(t) \right] \right) \\
r(t) &= \sigma \left( b^r + W^r \left[ h^{(t-1)} , x(t) \right] \right) \\
g(t) &= \tanh \left( b^g + W^g \left[ r(t) \odot h^{(t-1)} , x(t) \right] \right) \\
h(t) &= (1 - z(t)) \odot h^{(t-1)} + z(t) \odot g(t)
\end{align*}
\]
Summary

• On MNIST, RNN, LSTM, GRU can be pruned by 95% without significant accuracy loss

• LSTM and GRU have more redundancy than RNN

• Hidden to hidden layers have more redundancy