Introduction

Convolutional Neural Networks have proven to be able to achieve considerable performance in many tasks in computer vision. However, CNNs still have a lot of shortcomings and challenges:

- Over-parameterization
- Computationally expensive at training and inference time
- Unsuitable for real-time applications

In this paper, we propose a new technique that improves the trade-off between accuracy and complexity by:

- Improve existing progressive pruning technique by combining hard and soft pruning
- Handle the pruning of momentum tensor during back-propagation
- Introduce a criterion that’s more suited for progressive pruning

Progressive Gradient Pruning - Forward Pass and Backward Pass

A forward pass calculation is used to calculate the pruning ratio at each epoch:  
\[ p_t = \exp \left( \frac{\log(1 - t)}{T_{prune}} \right) \]  
(1)

where \( T_{prune} \) is a hyper-parameter that defines the ratio of output channels to be pruned, and \( t \in \{1, 2, ..., T\} \) is the epoch. Since we progressively prune layer by layer and epoch by epoch, we calculate the number of weak channels or the number of remaining channels at each layer, \( n_{t \in F} \). Given ratio \( p_t \) at epoch \( t \), the number of weak output channels for any layer is defined as:

\[ n_{t \in F} = n_{t-1} \times (1 - p_t) \]  
(2)

where \( n_{t-1} \) can be the original number of output channels of any layer. Using the number of weak channels \( n_{t \in F} \), we can prune that amount of channels at each layer and each epochs.

Progressive Gradient Pruning - Forward Pass and Backward Pass

Contribution

This work contributes to the field of CNN compression by proposing a novel progressive pruning technique that combines soft and hard pruning. The technique is designed to improve the balance between accuracy and computational efficiency, particularly in real-time applications. The proposed method demonstrates significant performance improvements over existing techniques while maintaining a lower computational cost.

Table 1: Performance of pruning methods for training Faster R-CNN with VGG16 backbone on the Pascal VOC detection dataset with \( t_{prune} = 20\% \).

<table>
<thead>
<tr>
<th>Method</th>
<th>No. Params</th>
<th>mAP</th>
<th>Training Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline VGG16</td>
<td>137M</td>
<td>70.0</td>
<td>46.3</td>
</tr>
<tr>
<td>S4P3P</td>
<td>97M</td>
<td>69.6</td>
<td>74.3</td>
</tr>
<tr>
<td>Soft Pruned</td>
<td>74M</td>
<td>68.1</td>
<td>90.4</td>
</tr>
<tr>
<td>Hard Pruned</td>
<td>54M</td>
<td>64.0</td>
<td>115.2</td>
</tr>
</tbody>
</table>

Pruning on domain adaptation

Table 2: Performance of our methods for training DAN with VGG16 backbone on Office-31 with \( t_{prune} = 20\% \).

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Source-only VGG</th>
<th>VGG</th>
<th>TCF</th>
<th>RPGP_GN</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ( \rightarrow ) W</td>
<td>68.5</td>
<td>74.0</td>
<td>76.1</td>
<td>80.0</td>
</tr>
<tr>
<td>A ( \rightarrow ) D</td>
<td>61.6</td>
<td>72.3</td>
<td>76.4</td>
<td>79.7</td>
</tr>
<tr>
<td>W ( \rightarrow ) A</td>
<td>74.3</td>
<td>75.7</td>
<td>77.8</td>
<td>93.4</td>
</tr>
<tr>
<td>D ( \rightarrow ) W</td>
<td>94.5</td>
<td>94.0</td>
<td>96.1</td>
<td>98.5</td>
</tr>
<tr>
<td>Average</td>
<td>68.4</td>
<td>74.5</td>
<td>76.2</td>
<td>80.0</td>
</tr>
</tbody>
</table>

Conclusion

The proposed progressive gradient pruning technique offers several advantages:

- Improved performance on existing progressive pruning techniques
- Allows pruning of supervised or unsupervised tasks
- A pruning criterion suitable for pruning during training
- A faster pruning and training time

References


Eric Granger
Genetec Inc.