Convolutional Neural Network Compression

CAP 6411-Fall 2022
Convolutional Neural Networks

- Deep CNNs excel at object detection and recognition, activity recognition, etc.

- Due to:
  - availability of large amounts of training data
  - dramatic increase in computing resources
Motivation

• Flops increase by 300,000x in 6 years
• Doubling in 3-4 months
• Hardware improving at much slower rate
• Hundreds of GPUs required for training and deployment
• Limit AI access to large corporations (Google, FB, etc.)
• Solution: Efficient Neural Networks
• Compression with pruning
• Analytical methods (Find the best set of filters, discard the rest)
Different CNN Compression Techniques

• Pruning
  • Delete unimportant nodes and weights

• Filter Quantization
  • Quantize the weights to reduce storage requirements

• Filter Approximation
  • Low rank approximations that minimize performance loss
Convolutional layer

32x32x3 image

5x5x3 filter

- Convolve the image with the filter: sliding over the image spatially, computing dot product.
Convolutional Layer

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation map
Convolutional Layer

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

consider a second, green filter

activation maps
Convolutional Layer

- Six 5x5 filters
- Output 6 activation maps
Basis Representation of Filters

- Assume that the data $x(m,n,l)$ is processed by the filters $h_k(m,n,l)$
  \[ y_k(m,n) = x(m,n,l) \ast h_k(m,n,l), \quad 1 \leq k \leq P \]

- The filters $h_k(m,n,l)$ can be expressed in terms of the basis filters $f_i(m,n,l)$:
  \[ h_k(m,n,l) = \sum_{i=1}^{Q} w_{ik} \cdot f_i(m,n,l) \]

- Therefore, the output is given by
  \[ y_k(m,n) = \sum_{i=1}^{Q} w_{ik} \cdot [x(m,n,l) \ast f_i(m,n,l)], \quad 1 \leq k \leq P \]

- Define 1D filters $w_k(q)=[w_{1k}, w_{2k} \cdots w_{Qk}]$ and $z_i(m,n) = x(m,n,l) \ast f_i(m,n,l)$, we get:
  - $y_k(m,n) = w_k(q) \ast z(m,n,q)$
Basis Representation of Filters
• Convolution with $P$ original filters is replaced with two successive convolutions
• First with $Q$ basis filters, followed by $P \times 1$ dimensional filters
• The number of multiplications is reduced by a factor of
\[
\frac{P \cdot L \cdot D^2}{Q \cdot [L \cdot D^2 + P]} \approx \frac{P}{Q}
\]
Principle Component Representation of Filters

• Define \( A = [h_1 \  h_2 \  \cdots \  h_p] \) where \( h_k \) is unrolled filter of size \( LD^2 \times 1 \)
• Further \( M = AA^T \) dimension of \( M \) is \( LD^2 \times LD^2 \)
• Compute the eigen vectors and eigen values of \( M \)
• \( Mf_i = \lambda_i f_i \)
• Define \( F = [f_1 \  f_2 \ \cdots \  f_{LD^2}] \) Matrix with eigen vectors as columns
• \( \lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_{LD}) \) Diagonal matrix with eigenvalues
• Choose \( Q \) dominant eigen vectors so that such that \( \frac{\sum_{i=1}^{Q} \lambda_i}{\sum_{i=1}^{LD^2} \lambda_i} > t \)

• The filters can now be approximated as \( h_k = Fw_k \), where \( w_k = [w_{1k} \ w_{2k} \ \cdots \ w_{Qk}]^T \)
• Given the filters \( h_k \) the weights can be computed as \( w_k = F^T h_k \)
Rank Selection

- Consider Layer 30 of VGG16 Network
  - There are 512 filters, each of size 3 x 3 x 512.
  - Each filter is converted in a 4608 x 1 dimensional vector
  - A is therefore a matrix of size 4608 x 512. \( V = A^T A \) is 512 x 512.
  - Note that \( M = A A^T \) and \( V = A^T A \) share the same non singular eigen-values.
- The eigen-values of this layer show that we need to retain approximately 300 of these filters to represent 90% of the information contained in the original filters.
Fine-Tuning Approach

• Truncation of the basis functions causes performance to drop.
• However, the weights of linear combination can be readjusted to recover loss in performance.
• Therefore, these weights are “fine tuned” while the basis functions are held constant.
• Reposition the filters in the original subspace represented by the basis functions.
• The steps are: i) Compute the PCA of the pre-trained Network, ii) drop weaker eigenvectors, and iii) refine the weights of linear combination.
Spectral Fine Tuning

- Consider the original space domain filters $h_k$
- Recall that $A=[h_1 \ h_2 \ \cdots \ \ h_p]$
- The correlation matrix for the filters is given by $M=AA^T \cong E[h_k h_k']$
- Since $M$ is symmetric and positive definite we can write $M=F \Delta F^T$
- Known as Spectral Decomposition
- $F$ is the matrix of eigenvectors, and $\Delta$ is a diagonal matrix of eigenvalues along the main diagonal
- Now consider the linear model $h_k = Fw_k$
- Then $M=E[h_k h_k'] = E[Fw_k w_k'^T F^T] = FE[w_k w_k'^T F^T]$
- Therefore, $E[w_k w_k'^T] = \Delta$ is a diagonal matrix
- Therefore, adapting $w_k$ is akin to fine tuning the “spectra” of the filters
- The learning and convergence of each of the elements of $w_k$ occurs “independently”, which leads to faster overall convergence.
Evaluation

• Tested on several datasets and network architectures

Datasets
• CIFAR10 (classification, 10 classes, 60k)
• CIFAR100 (classification, 100 classes, 60k)
• SVHN (classification, 10)
• ImageNet (classification, 1000 classes, 1.3 Million)
• MS COCO (object detection, 80 classes)

Network Architectures
• Alexnet (classification, 2012, 0.03 Billion)
• VGG16 (classification, 2014, 0.63 Billion)
• Resnet (classification, 2016, 0.51 Billion)
• Densent (classification, 2017, 18.61 Billion)
• Yolo (object detection, 2018, 65.86 Billion)
Compression depends on data set complexity

The VGG-16 Network was trained on SVHN, CIFAR-10, CIFAR-100 and ImageNet data sets
- The network trained on the SVHN data set was most compressible, followed by CIFAR-10, CIFAR-100, and ImageNet in that order
- The compressibility of the network depends on the complexity of the problem

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FLOPs ↓%</th>
<th>Params ↓%</th>
<th>Acc. ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVHN [27]</td>
<td>90.52</td>
<td>99.26</td>
<td>1.10</td>
</tr>
<tr>
<td>CIFAR10 [22]</td>
<td>82.04</td>
<td>99.07</td>
<td>2.83</td>
</tr>
<tr>
<td>CIFAR100 [22]</td>
<td>65.99</td>
<td>97.72</td>
<td>1.23</td>
</tr>
</tbody>
</table>
Network complexity and Compression

- Using the same CIFAR 100 data set, the more complex networks are more compressible
- Performance after finetuning is comparable to that of the uncompressed network
Compression by Layers in VGG-16
Conclusion

• Simple method for CNN compression which works with several different architectures
• More than 95% reduction in trainable parameters
• Basis Compression algorithm is fast
• Approximation takes seconds
• Fine tuning needs roughly 25 epochs
• Implementation is very general, almost no custom code needed to compress a new network on any dataset.