ImageNet Classification with Deep Convolutional Neural Networks

By Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton

Presented by Jay Gokhale
Problem: Large-Scale object class definition
Motivation

• Number of image classes close to the scale of what a human can differentiate
• A large amount of data means algorithms need to be scalable
Key Contributions

• Efficient method to learn parameters of a convolutional net

• Amazing benchmark results
Background

• Neural Network
Background: Fully Connected NN

- Forward Propagation
- $n_2 = W_{21} p_1 + b_1$
- $a_2 = f(n_2)$
Background: Convolutional NN

- Same as fully connected but with constraints
Background: history

• Convolutional Neural Nets are old
  – Invented in 1980 by Kunihiko Fukushima

• Technology and dataset sizes limited learning
Overview of Implementation

Convolutional Layers

Fully Connected Layers
Why Deep?

• Learn feature hierarchies

Optimizations to speed up training

- Use Rectified Linear Units: \( f(x) = \max(0, x) \)
Optimizations to speed up training

• Train with two GPUs
• Overlap pooling
• Local response normalization
Combating overfitting

• Many parameters -> can easily overfit
  – Data augmentation
  – Dropout
Data Augmentation

• Create fake “new data”
  – Adding Noise
  – Sampling random 224x224 Patches
  – Horizontal Reflection
Dropout
Learning

• Backpropagation/ Stochastic Gradient Descent
  – Glorified chain rule

• Used a momentum method to update weights

\[ v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w} \mid_{w_i} \right\rangle_{\mathcal{D}_i} \]

\[ w_{i+1} := w_i + v_{i+1} \]

| \( v_{i+1} \) | Momentum at iteration \( i+1 \) | 0.0005 = weight decay |
| \( w_{i+1} \) | weight at iteration \( i+1 \) | 0.9 = friction coefficient |
| \( \epsilon \) | learning rate |
| \( L \) | cost function |
Strengths

• Great benchmark results
• Takes raw data as input
  – Learns own features
• (Empirically,) accuracies don’t seem to have plateaued yet.
  – 2014 paper achieves 10% error rate
Weaknesses

• Relatively slow to train
• A lot of hyper parameters
  – Number of layers, number of neurons/ layer, etc.
Experimental Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse coding [2]</td>
<td>47.1%</td>
<td>28.2%</td>
</tr>
<tr>
<td>SIFT + FVs [24]</td>
<td>45.7%</td>
<td>25.7%</td>
</tr>
<tr>
<td>CNN</td>
<td>37.5%</td>
<td>17.0%</td>
</tr>
</tbody>
</table>

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1 (val)</th>
<th>Top-5 (val)</th>
<th>Top-5 (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT + FVs [7]</td>
<td>—</td>
<td>—</td>
<td>26.2%</td>
</tr>
<tr>
<td>1 CNN</td>
<td>40.7%</td>
<td>18.2%</td>
<td>—</td>
</tr>
<tr>
<td>5 CNNs</td>
<td>38.1%</td>
<td>16.4%</td>
<td>16.4%</td>
</tr>
<tr>
<td>1 CNN*</td>
<td>39.0%</td>
<td>16.6%</td>
<td>—</td>
</tr>
<tr>
<td>7 CNNs*</td>
<td>36.7%</td>
<td>15.4%</td>
<td>15.3%</td>
</tr>
</tbody>
</table>

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were “pre-trained” to classify the entire ImageNet 2011 Fall release. See Section 6 for details.
The End

• Questions?