ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
Ilya Sutskever
Geoffrey Hinton

University of Toronto
Canada

Paper with same name to appear in NIPS 2012
Main idea
Architecture
Technical details
Neural networks

• A neuron

\[ x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3) \]

\( x \) is called the total input to the neuron, and \( f(x) \) is its output.

• A neural network

A neural network computes a differentiable function of its input. For example, ours computes: \( p(\text{label} \mid \text{an input image}) \)
Convolutional neural networks

• Here's a one-dimensional convolutional neural network

• Each hidden neuron applies **the same localized, linear filter** to the input
Convolution in 2D

Input “image”

Filter bank

Output map
Local pooling

Diagram showing the concept of local pooling in a neural network.
Overview of our model

- **Deep**: 7 hidden “weight” layers
- **Learned**: all feature extractors initialized at white Gaussian noise and learned from the data
- Entirely supervised
- More data = good

**Convolutional layer**: convolves its input with a bank of 3D filters, then applies point-wise non-linearity

**Fully-connected layer**: applies linear filters to its input, then applies point-wise non-linearity
Overview of our model

- Trained with stochastic gradient descent on two NVIDIA GPUs for about a week
- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- **Final feature layer**: 4096-dimensional

**Convolutional layer**: convolves its input with a bank of 3D filters, then applies point-wise non-linearity

**Fully-connected layer**: applies linear filters to its input, then applies point-wise non-linearity
96 learned low-level filters
Main idea

Architecture

Technical details
Training

Using stochastic gradient descent and the backpropagation algorithm (just repeated application of the chain rule)

One output unit per class

\[ x_i = \text{total input to output unit } i \]

\[ f(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{1000} \exp(x_j)} \]

We maximize the log-probability of the correct label, \( \log f(x_t) \)
Our model

- Max-pooling layers follow first, second, and fifth convolutional layers
- The number of neurons in each layer is given by 253440, 186624, 64896, 64896, 43264, 4096, 4096, 1000
Main idea
Architecture

→ Technical details
Input representation

- Centered (0-mean) RGB values.
Neurons

\[ f(x) = \tanh(x) \]

\[ x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3) \]

\( x \) is called the total input to the neuron, and \( f(x) \) is its output.

Very bad (slow to train)

Very good (quick to train)
Data augmentation

- Our neural net has 60M real-valued parameters and 650,000 neurons
- It overfits a lot. Therefore we train on 224x224 patches extracted randomly from 256x256 images, and also their horizontal reflections.
Testing

• Average predictions made at five 224x224 patches and their horizontal reflections (four corner patches and center patch)

• Logistic regression has the nice property that it outputs a probability distribution over the class labels

• Therefore no score normalization or calibration is necessary to combine the predictions of different models (or the same model on different patches), as would be necessary with an SVM.
Dropout

- Independently set each hidden unit activity to zero with 0.5 probability
- We do this in the two globally-connected hidden layers at the net's output
Implementation

• The only thing that needs to be stored on disk is the raw image data.

• We stored it in JPEG format. It can be loaded and decoded entirely in parallel with training.

• Therefore only 27GB of disk storage is needed to train this system.

• Uses about 2GB of RAM on each GPU, and around 5GB of system memory during training.
Implementation

- Written in Python/C++/CUDA
- Sort of like an instruction pipeline, with the following 4 instructions happening in parallel:
  - Train on batch $n$ (on GPUs)
  - Copy batch $n+1$ to GPU memory
  - Transform batch $n+2$ (on CPU)
  - Load batch $n+3$ from disk (on CPU)
### Validation classification

<table>
<thead>
<tr>
<th>mite</th>
<th>container ship</th>
<th>motor scooter</th>
<th>leopard</th>
</tr>
</thead>
<tbody>
<tr>
<td>black widow</td>
<td>container ship</td>
<td>motor scooter</td>
<td>leopard</td>
</tr>
<tr>
<td>cockroach</td>
<td>lifeboat</td>
<td>go-kart</td>
<td>jaguar</td>
</tr>
<tr>
<td>tick</td>
<td>amphibian</td>
<td>moped</td>
<td>cheetah</td>
</tr>
<tr>
<td>starfish</td>
<td>fireboat</td>
<td>bumper car</td>
<td>snow leopard</td>
</tr>
<tr>
<td></td>
<td>drilling platform</td>
<td>golfcart</td>
<td>Egyptian cat</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>grille</th>
<th>mushroom</th>
<th>cherry</th>
<th>Madagascar cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>convertible</td>
<td>agaric</td>
<td>dalmatian</td>
<td>squirrel monkey</td>
</tr>
<tr>
<td>grille</td>
<td>mushroom</td>
<td>grape</td>
<td>spider monkey</td>
</tr>
<tr>
<td>pickup</td>
<td>jelly fungus</td>
<td>elderberry</td>
<td>titi</td>
</tr>
<tr>
<td>beach wagon</td>
<td>gill fungus</td>
<td>fordshire bullterrier</td>
<td>indri</td>
</tr>
<tr>
<td>fire engine</td>
<td>dead-man’s-fingers</td>
<td>currant</td>
<td>howler monkey</td>
</tr>
</tbody>
</table>
## Validation classification

<table>
<thead>
<tr>
<th>Lens Cap</th>
<th>Abacus</th>
<th>Slug</th>
<th>Hen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reflex camera</td>
<td>Polaroid camera</td>
<td>Pencil sharpener</td>
<td>Switch</td>
</tr>
<tr>
<td>Typewriter keyboard</td>
<td>Space bar</td>
<td>Computer keyboard</td>
<td>Accordian</td>
</tr>
<tr>
<td>Tiger</td>
<td>Chambered nautilus</td>
<td>Tape player</td>
<td>Planetarium</td>
</tr>
<tr>
<td>Tiger</td>
<td>Cat</td>
<td>Tabby</td>
<td>Boxer</td>
</tr>
<tr>
<td>Lampshade</td>
<td>Throne</td>
<td>Goblet</td>
<td>Table lamp</td>
</tr>
<tr>
<td>Cellular telephone</td>
<td>Slot</td>
<td>Reflex camera</td>
<td>Dial telephone</td>
</tr>
<tr>
<td>Dome</td>
<td>Mosque</td>
<td>Radio telescope</td>
<td>Steel arch bridge</td>
</tr>
</tbody>
</table>
Validation classification

koala  tiger  European fire salamander  loggerhead
wombat  Norwegian elkhound  wild boar  wallaby
        koala  tiger  cat  jaguar  lynx  leopard
        spotted salamander  common newt  long-horned beetle  box turtle
        African crocodile  Gila monster  loggerhead  mud turtle  leatherback turtle

seat belt  television  sliding door  wallaby
seat belt  ice lolly  hotdog  burrito  Band Aid
        television  microwave  monitor  screen  car mirror
        sliding door  shoji  window shade  window screen  four-poster
        hare  wallaby  wood rabbit  Lakeland terrier  kit fox
### Validation localizations

<table>
<thead>
<tr>
<th>bookshop</th>
<th>coyote</th>
<th>cradle</th>
<th>wood rabbit</th>
</tr>
</thead>
<tbody>
<tr>
<td>balance beam</td>
<td>grey fox</td>
<td>cradle</td>
<td>hare</td>
</tr>
<tr>
<td>cinema</td>
<td>kit fox</td>
<td>bassinet</td>
<td>wood rabbit</td>
</tr>
<tr>
<td>marimba</td>
<td>red fox</td>
<td>diaper</td>
<td>grey fox</td>
</tr>
<tr>
<td>parallel bars</td>
<td>coyote</td>
<td>crib</td>
<td>coyote</td>
</tr>
<tr>
<td>computer keyboard</td>
<td>dhole</td>
<td>bath towel</td>
<td>wallaby</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bottlecap</td>
<td>harvester</td>
<td>garter snake</td>
<td>Walker hound</td>
</tr>
<tr>
<td>magnetic compass</td>
<td>bottleneck</td>
<td>diamondback</td>
<td>beagle</td>
</tr>
<tr>
<td>puck</td>
<td>thresher</td>
<td>leatherback</td>
<td>Walker hound</td>
</tr>
<tr>
<td>stopwatch</td>
<td>plow</td>
<td>turtle</td>
<td>English foxhound</td>
</tr>
<tr>
<td>disk brake</td>
<td>tractor</td>
<td>sandbar</td>
<td>muzzle</td>
</tr>
<tr>
<td></td>
<td>tow truck</td>
<td>echidna</td>
<td>Italian greyhound</td>
</tr>
</tbody>
</table>
Validation localizations
Retrieval experiments

First column contains query images from ILSVRC-2010 test set, remaining columns contain retrieved images from training set.
Retrieval experiments