1 Algorithm of CNN

2 Application in Traffic Sign Recognition

3 How to use it in detection?

4 Descriptor matching

5 Conclusion
Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

Abstract— Multilayer Neural Networks trained with the backpropagation algorithm constitute the best example of a successful Gradient-Based Learning technique. Given an appropriate network architecture, Gradient-Based Learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns such as handwritten characters, with minimal preprocessing. This paper reviews various methods applied to handwritten character recognition and compares them on a standard handwritten digit recognition task. Convolutional Neural Networks, that are specifically designed to deal with the variability of 2D shapes, are

I. Introduction

Over the last several years, machine learning techniques, particularly when applied to neural networks, have played an increasingly important role in the design of pattern recognition systems. In fact, it could be argued that the availability of learning techniques has been a crucial factor in the recent success of pattern recognition applications such as continuous speech recognition and handwriting recognition.
The first module, called the feature extractor, transforms the input patterns so that they can be represented by low-dimensional vectors.

The classifier, on the other hand, is often general purpose and trainable.

The recognition accuracy is largely determined by the ability of the designer to come up with an appropriate set of features.
How to learn the feature extractor itself?

If we input the raw pixels to the multilayer networks and train it, then:

- Typical images are large, and the networks contain several tens of thousands of weights. Such a large number of parameters increases the capacity of the system and therefore requires a larger training set.

- The unstructured nets for image applications have no built-in invariance with respect to translations or local distortions of the inputs.

- The topology of the input is entirely ignored. The input variables can be presented in any (fixed) order without affecting the outcome of the training.
How to learn the feature extractor itself? (cont’d)

Architecture of Convolutional Neural Networks
Rebirth of CNN

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky  
University of Toronto  
kriz@cs.utoronto.ca

Ilya Sutskever  
University of Toronto  
ilya@cs.utoronto.ca

Geoffrey E. Hinton  
University of Toronto  
hinton@cs.utoronto.ca

## Dataset

### ImageNet

- 15 million labeled high-resolution images belonging to roughly 22,000 categories.

### ILSVRC (ImageNet Large-Scale Visual Recognition Challenge)

- A subset of ImageNet with roughly 1000 images in each of 1000 categories.
- There are roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images.
- It is customary to report two error rates: top-1 and top-5, where the top-5 error rate is the fraction of test images for which the correct label is not among the five labels considered most probable by the model.

### Data pre-process

- Rescale the image such that the shorter side was of length 256, and then cropped out the central 256x256 patch from the resulting image.
- Subtract the mean activity over the training set from each pixel.
The Architecture
The Architecture (cont'd)

- ReLU (Rectified Linear Units) Nonlinearity: deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units.

- Training on Multiple GPUs.

- Local Response Normalization: local normalization after ReLU Nonlinearity aids generalization.

- Overlapping Pooling: during training that models with overlapping pooling find it slightly more difficult to overfit.
Reducing Overfitting

Data Augmentation

- generating image translations and horizontal reflections. We do this by extracting random 224x224 patches (and their horizontal reflections) from the 256x256 images and training our network on these extracted patches. This increases the size of our training set by a factor of 2048,

- The second form of data augmentation consists of altering the intensities of the RGB channels in training images. Specifically, we perform PCA on the set of RGB pixel values throughout the ImageNet training set.

Dropout

- We use dropout in the first two fully-connected layer. Without dropout, our network exhibits substantial overfitting.
Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

<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.
Figure 4: **(Left)** Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). **(Right)** Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.
CNN Features off-the-shelf: an Astounding Baseline for Recognition

Ali Sharif Razavian  Hossein Azizpour  Josephine Sullivan  Stefan Carlsson
CVAP, KTH (Royal Institute of Technology)
Stockholm, Sweden
{razavian,azizpour,sullivan,stefanc}@csc.kth.se
CNN result on many datasets

 convolutional neural networks (CNN)

June 10, 2014 15 / 53

Luo Hengliang (Institute of Automation)
For all the experiments we resize the whole image (or cropped sub-window) to 221x221 and input the image to OverFeat. This gives a vector of 4096 dimensions. We have two settings:

- The feature vector is further L2 normalized to unit length for all the experiments. We use the 4096 dimensional feature vector in combination with a Support Vector Machine (SVM) to solve different classification tasks (CNN-SVM).
- We further augment the training set by adding cropped and rotated samples and doing component-wise power transform and report separate results (CNNAug+SVM)
## Pascal VOC 2007 Image Classification Results

<table>
<thead>
<tr>
<th></th>
<th>aero</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>mbike</th>
<th>person</th>
<th>plant</th>
<th>sheep</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHM[8]</td>
<td>76.7</td>
<td>74.7</td>
<td>53.8</td>
<td>72.1</td>
<td>40.4</td>
<td>71.7</td>
<td>83.6</td>
<td>52.5</td>
<td>57.5</td>
<td>62.8</td>
<td>51.1</td>
<td>81.4</td>
<td>71.5</td>
<td>86.5</td>
<td>36.4</td>
<td>55.3</td>
<td>60.6</td>
<td>80.6</td>
<td>57.8</td>
<td>64.7</td>
<td></td>
</tr>
<tr>
<td>AGS[11]</td>
<td>82.2</td>
<td>83.0</td>
<td>58.4</td>
<td>76.1</td>
<td>56.4</td>
<td>77.5</td>
<td>88.8</td>
<td>69.1</td>
<td>62.2</td>
<td>61.8</td>
<td>63.7</td>
<td>51.3</td>
<td>85.4</td>
<td>80.2</td>
<td>91.1</td>
<td>48.1</td>
<td>61.7</td>
<td>67.7</td>
<td>86.3</td>
<td>70.9</td>
<td>71.1</td>
</tr>
<tr>
<td>NUS[39]</td>
<td>82.5</td>
<td>79.6</td>
<td>64.8</td>
<td>73.4</td>
<td>54.2</td>
<td>75.0</td>
<td>77.5</td>
<td>79.2</td>
<td>46.2</td>
<td>62.7</td>
<td>41.4</td>
<td>74.6</td>
<td>85.0</td>
<td>76.8</td>
<td>91.1</td>
<td>53.9</td>
<td>61.0</td>
<td>67.5</td>
<td>83.6</td>
<td>70.6</td>
<td>70.5</td>
</tr>
<tr>
<td>CNN-SVM</td>
<td>88.5</td>
<td>81.0</td>
<td>83.5</td>
<td>82.0</td>
<td>82.0</td>
<td>72.5</td>
<td>85.3</td>
<td>81.6</td>
<td>59.9</td>
<td>58.5</td>
<td>66.5</td>
<td>77.8</td>
<td>81.8</td>
<td>78.8</td>
<td>90.2</td>
<td>54.8</td>
<td>71.1</td>
<td>62.6</td>
<td>87.2</td>
<td>71.8</td>
<td>73.9</td>
</tr>
<tr>
<td>CNNaug-SVM</td>
<td>90.1</td>
<td>84.4</td>
<td>86.5</td>
<td>84.1</td>
<td>48.4</td>
<td>73.4</td>
<td>86.7</td>
<td>85.4</td>
<td>61.3</td>
<td>67.6</td>
<td>69.6</td>
<td>84.0</td>
<td>85.4</td>
<td>80.0</td>
<td>92.0</td>
<td>56.9</td>
<td>76.7</td>
<td>67.3</td>
<td>89.1</td>
<td>74.9</td>
<td>77.2</td>
</tr>
</tbody>
</table>

Table 1: **Pascal VOC 2007 Image Classification Results** compared to other methods which also use training data outside VOC. The CNN representation is not tuned for the Pascal VOC dataset. However, GHM [8] learns from VOC a joint representation of bag-of-visual-words and contextual information. AGS [11] learns a second layer of representation by clustering the VOC data into subcategories. NUS [39] trains a codebook for the SIFT, HOG and LBP descriptors from the VOC dataset. Oquab et al. [29] fixes all the layers trained on ImageNet then it adds and optimizes two fully connected layers on the VOC dataset and achieves better results (77.7) indicating the potential to boost the performance by further adaptation of the representation to the target task/dataset.
Table of Contents

1 Algorithm of CNN

2 Application in Traffic Sign Recognition

3 How to use it in detection?

4 Descriptor matching

5 Conclusion
GTSRB

- 43 classes, 39,209 training images, 12,630 test images, images size vary from 15x15 to 250x250.

Result

<table>
<thead>
<tr>
<th>CCR (%)</th>
<th>Team</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.46</td>
<td>IDSIA</td>
<td>Committee of CNNs</td>
</tr>
<tr>
<td>99.22</td>
<td>INI-RTCV</td>
<td>Human (best individual)</td>
</tr>
<tr>
<td>98.84</td>
<td>INI-RTCV</td>
<td>Human (average)</td>
</tr>
<tr>
<td>98.31</td>
<td>Sermanet</td>
<td>Multi-scale CNN</td>
</tr>
<tr>
<td>96.14</td>
<td>CAOR</td>
<td>Random forests</td>
</tr>
<tr>
<td>95.68</td>
<td>INI-RTCV</td>
<td>LDA (HOG 2)</td>
</tr>
<tr>
<td>93.18</td>
<td>INI-RTCV</td>
<td>LDA (HOG 1)</td>
</tr>
<tr>
<td>92.34</td>
<td>INI-RTCV</td>
<td>LDA (HOG 3)</td>
</tr>
</tbody>
</table>

Result overview for the final stage of the GTSRB.

---

4 http://benchmark.ini.rub.de/?section=gtsrb&subsection=news
About Torch7

Torch7 is a scientific computing framework with wide support for machine learning algorithms. It is easy to use and provides a very efficient implementation, thanks to an easy and fast scripting language, LuaJIT, and an underlying C implementation\(^a\).

\(^a\)torch.ch

Why Choose Torch7?

- It was recommended by Yann Lecun
- It is very fast
- See more from the website\(^a\) below.

\(^a\)http://www.kdnuggets.com/2014/02/exclusive-yann-lecun-deep-learning-facebook-ai-lab.html

Image pre-processing

Crop the image → Resize to 32x32 → Color to gray → CLAHE

Image preprocessing method from the paper

(g) Origin  (h) Crop  (i) Gray  (j) CLAHE

---

Contrast Limited Adaptive Histogram Equalization

Adaptive histogram equalization

Contrast limited adaptive histogram equalization

CNN Structure

Input:
- Convolutional layers (C1: feature maps, 16x28x28)
- Subsampling layers (S2: maps, 32x10x10)
- Convolutional layers (C3: maps, 32x5x5)
- Subsampling layers (S4: maps, 32x5x5)
- Fully connected layers (F5: layer, 800)
- Fully connected layers (F6: layer, 256)
- Output (43)

CNN Structure

Luo Hengliang (Institute of Automation)
Define Model in Torch7

model: add(nn.SpatialConvolutionMM(1, 16, 5, 5))
model: add(nn.Tanh())
model: add(nn.SpatialLPPooling(16, 2, 2, 2, 2, 2))

model: add(nn.SpatialConvolutionMM(16, 32, 5, 5))
model: add(nn.Tanh())
model: add(nn.SpatialLPPooling(32, 2, 2, 2, 2, 2))

model: add(nn.Reshape(32*5*5))
model: add(nn.Linear(32*5*5, 256))
model: add(nn.Tanh())
model: add(nn.Linear(256, 43))
The Test Dataset Result

Trainning set Iterations
Test Accuracy (%)
The best test accuracy is 97.75%.
Some misclassified examples in test set
Retrain: data augmentation

Data augmentation
- scaling: [0.8, 1.2]
- translation: random
- rotation: [−15°, 15°]
Recognize the detection results

<table>
<thead>
<tr>
<th>Class</th>
<th>Total</th>
<th>Without Aug.</th>
<th>With Aug.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Danger</td>
<td>62</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>Mandatory</td>
<td>56</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>Prohibitory</td>
<td>168</td>
<td>48</td>
<td>1</td>
</tr>
</tbody>
</table>

Some “hard” detection results
Torch7 doesn’t support Windows now, but we need to create a gui demo application in Windows.

So I write the forward cnn in C++ at https://github.com/beenfrog/cnn-forward
Table of Contents

1 Algorithm of CNN

2 Application in Traffic Sign Recognition

3 How to use it in detection?

4 Descriptor matching

5 Conclusion
Rich feature hierarchies for accurate object detection and semantic segmentation

Tech report

Ross Girshick¹  Jeff Donahue¹,²  Trevor Darrell¹,²  Jitendra Malik¹
¹UC Berkeley and ²ICSI
{rgb,jdonahue,trevor,malik}@eecs.berkeley.edu

---

Object detection system overview

**R-CNN: Regions with CNN features**

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

**overview**
- takes an input image
- extracts around 2000 bottom-up region proposals
- computes features for each proposal using a large convolutional neural network (CNN)
- classifies each region using class-specific linear SVMs
Region proposals: Selective Search

Selective Search

We extract a 4096-dimensional feature vector from each region proposal using our own implementation of the CNN of [Hinton2012].

In order to compute features for a region proposal, we must first convert the image data in that region into a fixed 224x224 pixel size.
Training

CNN pre-training

1. Pre-train CNN for image classification
   
   ![Diagram of CNN pre-training]
   
   - Large auxiliary dataset (ImageNet)

CNN fine-tuning

2. Fine-tune CNN on target dataset and task
   
   ![Diagram of CNN fine-tuning]
   
   - Small target dataset (PASCAL VOC)
   - (optional)

Object category classifiers

3. Train linear predictor for detection
   
   ![Diagram of object category classifiers]
   
   - Region proposals
   - Small target dataset (PASCAL VOC)
   - 2000 warped windows / image
   - CNN features
   - Per class SVM
   - Training labels
## R-CNN: Result

<table>
<thead>
<tr>
<th>VOC 2010 test</th>
<th>aero</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>mbike</th>
<th>person</th>
<th>plant</th>
<th>sheep</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM HOG [19]</td>
<td>45.6</td>
<td>49.0</td>
<td>11.0</td>
<td>11.6</td>
<td>27.2</td>
<td>50.5</td>
<td>43.1</td>
<td>23.6</td>
<td>17.0</td>
<td>23.2</td>
<td>10.7</td>
<td>20.5</td>
<td>42.5</td>
<td>44.5</td>
<td>41.3</td>
<td>8.7</td>
<td>29.0</td>
<td>18.7</td>
<td>40.0</td>
<td>34.5</td>
<td>29.6</td>
</tr>
<tr>
<td>SegDPM [18]</td>
<td>56.4</td>
<td>48.0</td>
<td>24.3</td>
<td>21.8</td>
<td><strong>31.3</strong></td>
<td><strong>51.3</strong></td>
<td>47.3</td>
<td>48.2</td>
<td>16.1</td>
<td>29.4</td>
<td>19.0</td>
<td>37.5</td>
<td>44.1</td>
<td>51.5</td>
<td>44.4</td>
<td>12.6</td>
<td>32.1</td>
<td>28.8</td>
<td><strong>48.9</strong></td>
<td>39.1</td>
<td>36.6</td>
</tr>
<tr>
<td>UVA [36]</td>
<td>56.2</td>
<td>42.4</td>
<td>15.3</td>
<td>12.6</td>
<td>21.8</td>
<td>49.3</td>
<td>36.8</td>
<td>46.1</td>
<td>12.9</td>
<td>32.1</td>
<td>30.0</td>
<td>36.5</td>
<td>43.5</td>
<td>52.9</td>
<td>32.9</td>
<td>15.3</td>
<td>41.1</td>
<td><strong>31.8</strong></td>
<td>47.0</td>
<td>44.8</td>
<td>35.1</td>
</tr>
<tr>
<td><strong>ours (R-CNN fc7)</strong></td>
<td><strong>65.4</strong></td>
<td><strong>56.5</strong></td>
<td><strong>45.1</strong></td>
<td><strong>28.5</strong></td>
<td><strong>24.0</strong></td>
<td><strong>50.1</strong></td>
<td><strong>49.1</strong></td>
<td><strong>58.3</strong></td>
<td><strong>20.6</strong></td>
<td><strong>38.5</strong></td>
<td><strong>31.1</strong></td>
<td><strong>57.5</strong></td>
<td><strong>50.7</strong></td>
<td><strong>60.3</strong></td>
<td><strong>44.7</strong></td>
<td><strong>21.6</strong></td>
<td><strong>48.5</strong></td>
<td><strong>24.9</strong></td>
<td><strong>48.0</strong></td>
<td><strong>46.5</strong></td>
<td><strong>43.5</strong></td>
</tr>
</tbody>
</table>

**Table 1:** Detection average precision (%) on VOC 2010 test. Our method competes in the comp4 track due to our use of outside data from ImageNet. Our system is most directly comparable to UVA (row 3) since both methods use the same selective search region proposal mechanism, but differ in features. We compare to methods before rescoring with inter-detector context and/or image classification.

<table>
<thead>
<tr>
<th>VOC 2007 test</th>
<th>aero</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>mbike</th>
<th>person</th>
<th>plant</th>
<th>sheep</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN pool5</td>
<td>49.3</td>
<td>58.0</td>
<td>29.7</td>
<td>22.2</td>
<td>20.6</td>
<td>47.7</td>
<td>56.8</td>
<td>43.6</td>
<td>16.0</td>
<td>39.7</td>
<td>37.7</td>
<td>39.6</td>
<td>49.6</td>
<td>55.6</td>
<td>37.5</td>
<td>20.6</td>
<td>40.5</td>
<td>37.4</td>
<td>47.8</td>
<td>51.3</td>
<td>40.1</td>
</tr>
<tr>
<td>R-CNN fc6</td>
<td>56.1</td>
<td>58.8</td>
<td>34.4</td>
<td>29.6</td>
<td>22.6</td>
<td>50.4</td>
<td>58.0</td>
<td>52.5</td>
<td>18.3</td>
<td>40.1</td>
<td>41.3</td>
<td>46.8</td>
<td>49.5</td>
<td>53.5</td>
<td>39.7</td>
<td>23.0</td>
<td>46.4</td>
<td>36.4</td>
<td>50.8</td>
<td>59.0</td>
<td>43.4</td>
</tr>
<tr>
<td>R-CNN fc7</td>
<td>53.1</td>
<td>58.9</td>
<td>35.4</td>
<td>29.6</td>
<td>22.3</td>
<td>50.0</td>
<td>57.7</td>
<td>52.4</td>
<td>19.1</td>
<td>43.5</td>
<td>40.8</td>
<td>43.6</td>
<td>47.6</td>
<td>54.0</td>
<td>39.1</td>
<td>23.0</td>
<td>42.3</td>
<td>33.6</td>
<td>51.4</td>
<td>55.2</td>
<td>42.6</td>
</tr>
<tr>
<td>R-CNN FT pool5</td>
<td>55.6</td>
<td>57.5</td>
<td>31.5</td>
<td>23.1</td>
<td>23.2</td>
<td>46.3</td>
<td>59.0</td>
<td>49.2</td>
<td>16.5</td>
<td>43.1</td>
<td>37.8</td>
<td>39.7</td>
<td>51.5</td>
<td>55.4</td>
<td>40.4</td>
<td>23.9</td>
<td>46.3</td>
<td>37.9</td>
<td>49.7</td>
<td>54.1</td>
<td>42.1</td>
</tr>
<tr>
<td>R-CNN FT fc6</td>
<td>61.8</td>
<td>62.0</td>
<td>38.8</td>
<td>35.7</td>
<td>29.4</td>
<td>52.5</td>
<td><strong>61.9</strong></td>
<td>53.9</td>
<td>22.6</td>
<td>49.7</td>
<td>40.5</td>
<td><strong>48.8</strong></td>
<td>49.9</td>
<td><strong>57.3</strong></td>
<td>44.5</td>
<td><strong>28.5</strong></td>
<td>50.4</td>
<td><strong>40.2</strong></td>
<td>54.3</td>
<td>61.2</td>
<td>47.2</td>
</tr>
<tr>
<td>R-CNN FT fc7</td>
<td><strong>60.3</strong></td>
<td><strong>62.5</strong></td>
<td><strong>41.4</strong></td>
<td><strong>37.9</strong></td>
<td><strong>29.0</strong></td>
<td><strong>52.6</strong></td>
<td><strong>61.6</strong></td>
<td><strong>56.3</strong></td>
<td><strong>24.9</strong></td>
<td><strong>52.3</strong></td>
<td><strong>41.9</strong></td>
<td><strong>48.1</strong></td>
<td><strong>54.3</strong></td>
<td><strong>57.0</strong></td>
<td><strong>45.0</strong></td>
<td><strong>26.9</strong></td>
<td><strong>51.8</strong></td>
<td><strong>38.1</strong></td>
<td><strong>56.6</strong></td>
<td><strong>62.2</strong></td>
<td><strong>48.0</strong></td>
</tr>
<tr>
<td>DPM HOG [19]</td>
<td>33.2</td>
<td>60.3</td>
<td>10.2</td>
<td>16.1</td>
<td>27.3</td>
<td>54.3</td>
<td>58.2</td>
<td>23.0</td>
<td>20.0</td>
<td>24.1</td>
<td>26.7</td>
<td>12.7</td>
<td><strong>58.1</strong></td>
<td>48.2</td>
<td>43.2</td>
<td>12.0</td>
<td>21.1</td>
<td>36.1</td>
<td>46.0</td>
<td>43.5</td>
<td>33.7</td>
</tr>
<tr>
<td>DPM ST [29]</td>
<td>23.8</td>
<td>58.2</td>
<td>10.5</td>
<td>8.5</td>
<td>27.1</td>
<td>50.4</td>
<td>52.0</td>
<td>7.3</td>
<td>19.2</td>
<td>22.8</td>
<td>18.1</td>
<td>8.0</td>
<td>55.9</td>
<td>44.8</td>
<td>32.4</td>
<td>13.3</td>
<td>15.9</td>
<td>22.8</td>
<td>46.2</td>
<td>44.9</td>
<td>29.1</td>
</tr>
<tr>
<td>DPM HSC [32]</td>
<td>32.2</td>
<td>58.3</td>
<td>11.5</td>
<td>16.3</td>
<td><strong>30.6</strong></td>
<td>49.9</td>
<td>54.8</td>
<td>23.5</td>
<td>21.5</td>
<td>27.7</td>
<td>34.0</td>
<td>13.7</td>
<td><strong>58.1</strong></td>
<td>51.6</td>
<td>39.9</td>
<td>12.4</td>
<td>23.5</td>
<td>34.4</td>
<td>47.4</td>
<td>45.2</td>
<td>34.3</td>
</tr>
</tbody>
</table>

**Table 2:** Detection average precision (%) on VOC 2007 test. Rows 1-3 show results for our CNN pre-trained on ILSVRC 2012. Rows 4-6 show results for our CNN pre-trained on ILSVRC 2012 and then fine-tuned (“FT”) on VOC 2007 trainval. Rows 7-9 present DPM methods as a strong baseline comparison. The first uses only HOG, while the next two use feature learning to augment or replace it.
Question from me: How to use the CNN effectively in object detection? The traditional sliding window method may be too slow. There are some works focused on generating region proposals first, such as http://arxiv.org/abs/1311.2524, any other new approaches? Thanks!

Answer by ylecun: ConvNets are not too slow for detection. Look at our paper on OverFeat [Sermanet et al. ICLR 2014], on pedestrian detection [Sermanet et al. CVPR 2013], and on face detection [Osadchy et al. JMLR 2007] and [Vaillant et al. 1994]. The key insight is that you can apply a ConvNet.....convolutionally over a large image, without having to recompute the entire network at every location (because much of the computation would be redundant). We have known this since the early 90’s.

Answer by osdf: A recent paper that takes the idea of avoiding recomputations to CNNs with max-pooling operations: Fast image scanning with deep max-pooling convolutional neural networks.
Fast Image Scanning

Speed up the forward computation of CNN$^{14}$

Both schemes follow the same intuition: that CNN filter banks can be approximated using a low rank basis of filters that are separable in the spatial domain.

---

Table of Contents

1 Algorithm of CNN

2 Application in Traffic Sign Recognition

3 How to use it in detection?

4 Descriptor matching

5 Conclusion
Descriptor Matching with Convolutional Neural Networks: a Comparison to SIFT

**Philipp Fischer**
Department of Computer Science
University of Freiburg
fischer@cs.uni-freiburg.de

**Alexey Dosovitskiy**
Department of Computer Science
University of Freiburg
dosovits@cs.uni-freiburg.de

**Thomas Brox**
Department of Computer Science
University of Freiburg
brox@cs.uni-freiburg.de

---

### Supervised Training

- We used a pre-trained model form [Hinton2012].

### Unsupervised Training

- We used random images from Flickr because we expect those to be better representatives of the distribution of natural images.
- Next $N = 16000$ “seed” patches of size 64x64 pixels were extracted randomly from different images at various locations and scales. Each of these “seed” patches was declared to represent a surrogate class of its own.
- These classes were augmented by applying $K = 150$ random transformations to each of the “seed” patches. Each transformation was a composition of random elementary transformations. These included translation, scale variation, rotation, color variation, contrast variation, and also blur, which is often relevant for matching problems.
- As a result we obtained a surrogate labeled dataset with $N$ classes containing $K$ samples each. We used these data to train a convolutional neural network.
The New Test Data Set

Figure 1: Some base images used for generating the dataset.

Figure 2: Most extreme versions of the transformations applied to the base images. From left to right: blur, lighting change, nonlinear deformation, perspective change, rotation, zoom.
Figure 4: Mean average precision on the larger dataset for various transformations. Except for the blur transformation, both neural nets perform consistently better than SIFT. The unsupervised net is also better on blur.
Table 1: Feature computation times for a patch of 91 by 91 pixels on a single CPU. On a GPU, the convolutional networks both need around 5.5ms per image.

<table>
<thead>
<tr>
<th>Method</th>
<th>SIFT</th>
<th>ImageNet CNN</th>
<th>Unsup. CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>2.95ms ± 0.04</td>
<td>11.1ms ± 0.28</td>
<td>37.6ms ± 0.6</td>
</tr>
</tbody>
</table>
Table of Contents

1 Algorithm of CNN

2 Application in Traffic Sign Recognition

3 How to use it in detection?

4 Descriptor matching

5 Conclusion
### Deep ConvNets: astounding baseline for vision

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Performance</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>[Sermanet et al 2014]: OverFeat (fine-tuned features for each task)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(tasks are ordered by increasing difficulty)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• image classification</td>
<td>ImageNet LSVRC 2013</td>
<td>competitive</td>
</tr>
<tr>
<td></td>
<td>Dogs vs Cats Kaggle challenge 2014</td>
<td>state of the art</td>
</tr>
<tr>
<td>• object localization</td>
<td>ImageNet LSVRC 2013</td>
<td>state of the art</td>
</tr>
<tr>
<td>• object detection</td>
<td>ImageNet LSVRC 2013</td>
<td>state of the art</td>
</tr>
</tbody>
</table>

| **[Razavian et al, 2014]: public OverFeat library (no retraining) + SVM**  |                                       |             |
| (simplest approach possible on purpose, no attempt at more complex classifiers) |                                       |             |
| (tasks are ordered by “distance” from classification task on which OverFeat was trained) |                                       |             |
| • image classification | Pascal VOC 2007                       | competitive | 73.9% mAP |
|                   | MIT-67                               | competitive | 58.4% mAP |
| • scene recognition | Caltech-UCSD Birds 200-2011          | competitive | 53.3% mAP |
|                   | Oxford 102 Flowers                   | competitive | 74.70% mAP |
| • fine grained recognition | UIUC 64 object attributes          | state of the art | 89.0% mAUC |
|                   | H3D Human Attributes                | state of the art | 70.78% mAP |
| • attribute detection | Oxford 5k buildings                | ?              | 0.52       |
|                   | Paris 6k buildings                  | ?              | 0.676      |
| • image retrieval  | Sculp6k                              | ?              | 0.269      |
| (search by image similarity) | Holidays                            | competitive | 0.646      |
|                   | UKBench                              | relatively poor | 3.05       |

---


Luo Hengliang (Institute of Automation)  
Convolutional Neural Networks (CNN)  
June 10, 2014  
49 / 53
### Deep ConvNets: astounding baseline for vision (cont’d)

<table>
<thead>
<tr>
<th>Study</th>
<th>Dataset</th>
<th>Performance</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Zeiler et al 2013]</td>
<td>ImageNet LSVRC 2013, Caltech-101 (15, 30 samples per class), Caltech-256 (15, 60 samples per class), Pascal VOC 2012</td>
<td>state of the art, competitive</td>
<td>11.2% error 83.8%, 86.5%</td>
</tr>
<tr>
<td>[Donahue et al, 2014]: DeCAF+SVM</td>
<td>Caltech-101 (30 classes), Amazon -&gt; Webcam, DSLR -&gt; Webcam, Caltech-UCSD Birds 200-2011, SUN-397</td>
<td>state of the art, competitive</td>
<td>86.91%</td>
</tr>
<tr>
<td>[Girshick et al, 2013]</td>
<td>Pascal VOC 2007, Pascal VOC 2010 (comp4), Pascal VOC 2011 (comp6)</td>
<td>state of the art</td>
<td>48.0% mAP</td>
</tr>
<tr>
<td>[Oquab et al, 2013]</td>
<td>Pascal VOC 2007, Pascal VOC 2012, Pascal VOC 2012 (action classification)</td>
<td>state of the art</td>
<td>77.7% mAP</td>
</tr>
</tbody>
</table>

Luo Hengliang (Institute of Automation)  Convolutional Neural Networks (CNN)  June 10, 2014
### Deep ConvNets: astounding baseline for vision (cont’d)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Performance</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>[Khan et al 2014]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>● shadow detection</td>
<td>UCF</td>
<td>state of the art</td>
</tr>
<tr>
<td></td>
<td>CMU</td>
<td>state of the art</td>
</tr>
<tr>
<td></td>
<td>UIUC</td>
<td>state of the art</td>
</tr>
<tr>
<td><strong>[Sander Dieleman, 2014]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>● image attributes</td>
<td>Kaggle Galaxy Zoo challenge</td>
<td>state of the art</td>
</tr>
</tbody>
</table>
Future works

- Use the feature learned from CNN in other vision tasks.
- Unsupervised Learning.
- ...
Thank you!