Deep Spatial Pyramid Ensemble for Cultural Event Recognition

Xiu-Shen Wei, Bin-Bin Gao and Jianxin Wu*
National Key Laboratory for Novel Software Technology, Nanjing University

Dec. 12, 2015    Santiago, Chile
Outline

- Background
- Deep Spatial Pyramid (DSP) and its ensemble
- Implementation details
- Experimental results
Background

Cultural Event Recognition
Deep Spatial Pyramid (DSP)

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network’s input is 150,528-dimensional, and the number of neurons in the network’s remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size $5 \times 5 \times 48$.

The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 192$. The fully-connected layers have 4096 neurons each.

4 Reducing Overfitting

Our neural network architecture has 60 million parameters. Although the 1000 classes of ILSVRC make each training example impose 10 bits of constraint on the mapping from image to label, this turns out to be insufficient to learn so many parameters without considerable overfitting. Below, we describe the two primary ways in which we combat overfitting.

4.1 Data Augmentation

The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformations (e.g., [25, 4, 5]). We employ two distinct forms of data augmentation, both of which allow transformed images to be produced from the original images with very little computation, so the transformed images do not need to be stored on disk.

In our implementation, the transformed images are generated in Python code on the CPU while the GPU is training on the previous batch of images. So these data augmentation schemes are, in effect, computationally free.

The first form of data augmentation consists of generating image translations and horizontal reflections. We do this by extracting random $224 \times 224$ patches (and their horizontal reflections) from the $256 \times 256$ images and training our network on these extracted patches. This increases the size of our training set by a factor of 2048, though the resulting training examples are, of course, highly interdependent. Without this scheme, our network suffers from substantial overfitting, which would have forced us to use much smaller networks. At test time, the network makes a prediction by extracting five $224 \times 224$ patches (the four corner patches and the center patch) as well as their horizontal reflections (hence ten patches in all), and averaging the predictions made by the network’s softmax layer on the ten patches.

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. To each training image, we add multiples of the found principal components.

This is the reason why the input images in Figure 2 are $224 \times 224 \times 3$-dimensional.
Deep Spatial Pyramid (DSP)

**CNN:**

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network’s input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size $5 \times 5 \times 48$.

The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 192$. The fully-connected layers have 4096 neurons each.

**4 Reducing Overfitting**

Our neural network architecture has 60 million parameters. Although the 1000 classes of ILSVRC make each training example impose 10 bits of constraint on the mapping from image to label, this turns out to be insufficient to learn so many parameters without considerable overfitting. Below, we describe the two primary ways in which we combat overfitting.

**4.1 Data Augmentation**

The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformations (e.g., [25, 4, 5]). We employ two distinct forms of data augmentation, both of which allow transformed images to be produced from the original images with very little computation, so the transformed images do not need to be stored on disk. In our implementation, the transformed images are generated in Python code on the CPU while the GPU is training on the previous batch of images. So these data augmentation schemes are, in effect, computationally free.

The first form of data augmentation consists of generating image translations and horizontal reflections. We do this by extracting random $224 \times 224$ patches (and their horizontal reflections) from the $256 \times 256$ images and training our network on these extracted patches. This increases the size of our training set by a factor of 2048, though the resulting training examples are, of course, highly interdependent. Without this scheme, our network suffers from substantial overfitting, which would have forced us to use much smaller networks. At test time, the network makes a prediction by extracting five $224 \times 224$ patches (the four corner patches and the center patch) as well as their horizontal reflections (hence ten patches in all), and averaging the predictions made by the network’s softmax layer on the ten patches.

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. To each training image, we add multiples of the found principal components.
Deep Spatial Pyramid (DSP)

CNN:

DSP:

Input image (Any resolution)  Pre-trained CNN  Activations  Spatial pyramid  The Fisher Vector encoding

Figure courtesy of [A. Krizhevsky et al., NIPS’ 12]
Deep Spatial Pyramid (DSP)

**CNN:**

![Diagram of CNN architecture](image)

**DSP:**

![Diagram of DSP architecture](image)

**Figure courtesy of [A. Krizhevsky et al., NIPS’ 12]**
Deep Spatial Pyramid (DSP)

CNN:

DSP:

Input image (Any resolution) → Pre-trained CNN → Activations → Spatial pyramid → Dictionary (GMM) → Fisher Vector → Power normalization → l2-normalization

Figure courtesy of [A. Krizhevsky et al., NIPS’ 12]
l2 matrix normalization in DSP:

\[ x_t \leftarrow x_t / \|X\|_2 \]

matrix spectral norm

*\(d\)-dimentional deep descriptors
DSP (con’t)

$l_2$ matrix normalization in DSP:

$d$-dimensional deep descriptors

\[ x_t \leftarrow x_t / \|X\|_2 \]

matrix spectral norm

Results of the different normalization methods:

<table>
<thead>
<tr>
<th></th>
<th>Caltech101</th>
<th>Stanford40</th>
<th>Scene15</th>
<th>Indoor67</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>90.63</td>
<td>74.84</td>
<td>90.75</td>
<td>71.20</td>
</tr>
<tr>
<td>$l_2$ vector</td>
<td>92.02</td>
<td>73.41</td>
<td>90.92</td>
<td>74.03</td>
</tr>
<tr>
<td>$l_2$ matrix</td>
<td><strong>92.56</strong></td>
<td><strong>78.43</strong></td>
<td><strong>90.99</strong></td>
<td><strong>74.55</strong></td>
</tr>
<tr>
<td>PCA+$l_2$ matrix</td>
<td>91.95</td>
<td>75.69</td>
<td>90.22</td>
<td>71.79</td>
</tr>
</tbody>
</table>
Encoding deep descriptors by FV:

\[
\begin{align*}
\mathbf{f}_{\mu_k}(X) &= \frac{1}{\sqrt{\omega_k}} \sum_{t=1}^{T} \gamma_t(k) \left( \frac{x_t - \mu_k}{\sigma_k} \right), \\
\mathbf{f}_{\sigma_k}(X) &= \frac{1}{\sqrt{2\omega_k}} \sum_{t=1}^{T} \gamma_t(k) \left[ \frac{(x_t - \mu_k)^2}{\sigma_k^2} - 1 \right].
\end{align*}
\]
DSP (con’t)

Encoding deep descriptors by FV:

\[
f_{\mu_k}(X) = \frac{1}{\sqrt{\omega_k}} \sum_{t=1}^{T} \gamma_t(k) \left( \frac{x_t - \mu_k}{\sigma_k} \right),
\]

\[
f_{\sigma_k}(X) = \frac{1}{\sqrt{2\omega_k}} \sum_{t=1}^{T} \gamma_t(k) \left[ \frac{(x_t - \mu_k)^2}{\sigma_k^2} - 1 \right].
\]

Multi-scale DSP:

\[
f_m = \frac{1}{S} \sum_{s=1}^{S} f_s \quad S = \{1.4, 1.2, 1.0, 0.8\}
\]
DSP (con’t)

Classification performance with different $K$:

(a) *Caltech-101* and *Scene15*  

(b) *Stanford40* and *Indoor67*
DSP (con’t)

Plot of $w$ values in DSP:

(a) Caltech101_64  (b) Caltech101_256  (c) Caltech256_64  (d) Caltech256_256

(i) Stanford40_64  (j) Stanford40_256  (k) VOC_64  (l) VOC_256
DSP (con’t)

Classification accuracy/MAP comparisons:

Table 3. Recognition accuracy (or mAP) comparisons on seven datasets. The highest accuracy (mAP) of each column is marked in bold. [17]’s results were achieved using VGG Net-D and VGG Net-E, evaluation was measured by mean class recall on Caltech-101, Caltech-256 instead of accuracy.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
<th>Caltech-101</th>
<th>Caltech-256</th>
<th>VOC 2007</th>
<th>Scene15</th>
<th>SUN397</th>
<th>MIT Indoor67</th>
<th>Stanford40</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoA</td>
<td></td>
<td><strong>93.42±0.50</strong></td>
<td>-</td>
<td>82.44</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[9]</td>
<td></td>
<td>93.42±0.50</td>
<td>-</td>
<td>82.44</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[7]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>51.98</td>
<td>68.88</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[27]</td>
<td></td>
<td>-</td>
<td>-</td>
<td>82.13</td>
<td>-</td>
<td>-</td>
<td>77.56</td>
<td>-</td>
</tr>
<tr>
<td>[30]</td>
<td></td>
<td>84.79±0.66</td>
<td>65.06±0.25</td>
<td>-</td>
<td>91.59±0.48</td>
<td>53.86±0.21</td>
<td>70.80</td>
<td>55.28±0.64</td>
</tr>
<tr>
<td>[1]</td>
<td></td>
<td>88.35±0.56</td>
<td>77.61±0.12</td>
<td>82.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[17]</td>
<td></td>
<td>92.7±0.5 (*)</td>
<td><strong>86.2±0.3(*)</strong></td>
<td><strong>89.7</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Baseline</td>
<td>Fc8</td>
<td>90.55±0.31</td>
<td>82.02±0.12</td>
<td>84.61</td>
<td>89.88±0.76</td>
<td>53.90±0.45</td>
<td>69.78</td>
<td>71.53±0.34</td>
</tr>
<tr>
<td>Pool5+FV</td>
<td></td>
<td>90.03±0.75</td>
<td>79.48±0.53</td>
<td>88.12</td>
<td>89.00±0.42</td>
<td>51.39±0.51</td>
<td>71.57</td>
<td>73.96±0.52</td>
</tr>
<tr>
<td>DSP</td>
<td></td>
<td>94.66±0.26</td>
<td>84.22±0.11</td>
<td>88.60</td>
<td>91.13±0.77</td>
<td>57.27±0.34</td>
<td>76.34</td>
<td>79.75±0.34</td>
</tr>
<tr>
<td>Our</td>
<td>Ms-DSP</td>
<td><strong>95.11±0.26</strong></td>
<td><strong>85.47±0.14</strong></td>
<td><strong>89.31</strong></td>
<td><strong>91.78±0.22</strong></td>
<td><strong>59.78±0.47</strong></td>
<td><strong>78.28</strong></td>
<td><strong>80.81±0.29</strong></td>
</tr>
</tbody>
</table>
DSP Ensemble

Our framework

- Cultural Images (VGG16)
- ImageNet (VGG16)
- Cultural Images (VGG19)
- ImageNet (VGG19)
- Place Images (Place-CNN)
DSP Ensemble

Our framework

- **Cultural Images VGG16**
- **ImageNet VGG16**
- **Cultural Images VGG19**
- **ImageNet VGG19**
- **Place Images Place-CNN**

[Diagram of DSP Ensemble process]

**Early fusion**

**Concatenation**

Table 1. Recognition mAP comparisons of the Development phase. Note that, “FT” stands for the fine-tuned deep networks; “SS” is for
viewed in color.

<table>
<thead>
<tr>
<th>Model</th>
<th>mAP</th>
<th>FT VGG Net-D</th>
<th>FT VGG Net-E</th>
<th>SS VGG Net-D</th>
<th>SS VGG Net-E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.841</td>
<td>0.784</td>
<td>0.784</td>
<td>0.773</td>
<td>0.773</td>
</tr>
<tr>
<td></td>
<td>0.802</td>
<td>0.784</td>
<td>0.784</td>
<td>0.773</td>
<td>0.773</td>
</tr>
<tr>
<td></td>
<td>0.784</td>
<td>0.784</td>
<td>0.784</td>
<td>0.773</td>
<td>0.773</td>
</tr>
<tr>
<td></td>
<td>0.773</td>
<td>0.784</td>
<td>0.784</td>
<td>0.773</td>
<td>0.773</td>
</tr>
</tbody>
</table>


C. Sun and R. Nevatia. ACTIVE: activity concept transitions in video event classification. In


Different feature maps from different networks:

(a) The original image
(b) Fine-tuned VGG Net-D
(c) VGG Net-D
(d) Fine-tuned VGG Net-E
(e) VGG Net-E
(f) Place-CNN
Implementation details

Distributions of the number of training images in Dev. and Final Evaluation:

(a) The original distribution of the training set in Development.

(c) The original distribution of the training set in Final Evaluation.
Implementation details

Distributions of the number of training images in Dev. and Final Evaluation:

- **The original distribution of the training set in Development**
  - The number of images
  - # of classes

- **The original distribution of the training set in Final Evaluation**
  - The number of images
  - # of classes

Late fusion
Implementation details

Distributions of the number of training images in Dev. and Final Evaluation:

(a) The original distribution of the training set in Development
(b) The distribution of the training set in Development after crops
(c) The original distribution of the training set in Final Evaluation
(d) The distribution of the training set in Final Evaluation after crops

Late fusion
Experimental results

Recognition mAP comparisons of the Development phase. Note that, “FT” stands for the fine-tuned deep networks; “SS” is for single scale, and “MS” is for multiple scales.

<table>
<thead>
<tr>
<th></th>
<th>VGG Net-D</th>
<th>VGG Net-E</th>
<th>FT VGG Net-D</th>
<th>FT VGG Net-E</th>
<th>Place-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>0.761</td>
<td>0.762</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>MS</td>
<td>0.770</td>
<td>0.773</td>
<td>0.779</td>
<td>0.769</td>
<td>0.640</td>
</tr>
<tr>
<td>Late fusion</td>
<td>0.782</td>
<td>0.784</td>
<td>0.802</td>
<td>0.791</td>
<td>0.649</td>
</tr>
<tr>
<td>Ensemble</td>
<td></td>
<td></td>
<td></td>
<td>0.841</td>
<td></td>
</tr>
</tbody>
</table>
Experimental results

Recognition mAP comparisons of the Development phase. Note that, “FT” stands for the fine-tuned deep networks; “SS” is for single scale, and “MS” is for multiple scales.

<table>
<thead>
<tr>
<th></th>
<th>VGG Net-D</th>
<th>VGG Net-E</th>
<th>FT VGG Net-D</th>
<th>FT VGG Net-E</th>
<th>Place-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>0.761</td>
<td>0.762</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>MS</td>
<td>0.770</td>
<td>0.773</td>
<td>0.779</td>
<td>0.769</td>
<td>0.640</td>
</tr>
<tr>
<td>Late fusion</td>
<td>0.782</td>
<td>0.784</td>
<td>0.802</td>
<td>0.791</td>
<td>0.649</td>
</tr>
<tr>
<td>Ensemble</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.841</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VIPL-ICT-CAS</td>
<td>0.854</td>
</tr>
<tr>
<td>2</td>
<td>FV (Ours)</td>
<td>0.851</td>
</tr>
<tr>
<td>3</td>
<td>MMLAB</td>
<td>0.847</td>
</tr>
<tr>
<td>4</td>
<td>NU&amp;C</td>
<td>0.824</td>
</tr>
<tr>
<td>5</td>
<td>CVL_ETHZ</td>
<td>0.798</td>
</tr>
</tbody>
</table>
Thank you!