Deep Learning for Super-Resolution and Compression

Presented by Yehuda Dar

Seminar in Computer Vision (236815)
Spring 2017

Presentation Outline

Overview exemplary utilisations of deep neural networks to:

• Image super-resolution

• Image compression
  • Artifact reduction for decompressed images.

The Super-Resolution Problem

The single-image super-resolution task:

Given a low-resolution image, construct its high-resolution version.

The naïve solution:

simple linear shift-invariant interpolation (e.g., bicubic).

The Super-Resolution Problem

Overview exemplary utilisations of deep neural networks to:

• Image super-resolution

• Image compression
  • Artifact reduction for decompressed images.

Part I: Image Super-Resolution

Based on the papers:

• Dong et al., "Learning a Deep Convolutional Network for Image Super-Resolution", ECCV 2014.
The Super-Resolution Problem: The Inverse-Problem Perspective

Contemporary super-resolution methods consider the task as an ill-posed inverse problem:

• The perspective justification:
  Many high-resolution images may lead to the given low-resolution image.

• Solution is determined using a regularization/prior applied at a patch-level or the complete candidate solution.

• The prior is often learned based on correspondence between low and high resolution patches.

SR-CNN: Paper Contributions
(Dong et al., 2016)

• Propose a fully convolutional neural network for image super resolution
  • Learns an end-to-end mapping between low and high resolution images.
  • An alternative to the explicit regularized inverse-problem approach.

• Super-resolution using sparse representations is conceptually interpreted as an instance of the proposed design.

• Compete with state-of-the-art super-resolution performance in terms of quality and speed.

SR-CNN: The Proposed Design

For a fixed upscaling factor:

\[
Y = \begin{bmatrix} f_1 \times f_1 & f_2 \times f_2 & f_3 \times f_3 \end{bmatrix}
\]

\[
F(0')
\]

Patch extraction and representation Non-linear mapping Reconstruction

The Super-Resolution Problem: The Inverse-Problem Perspective

A high-resolution image \( x_0 \in \mathbb{R}^x \) is deteriorated via

\[
y = SHx_0 + n
\]

where

- \( H \) is a linear blurring filter
- \( S \) is a linear downsampling operator
- \( \mathbf{n} \) is a column-vector of white Gaussian noise.

The general image-level optimization formulation for the super-resolution task as an inverse problem:

\[
\hat{x} = \arg\min_x ||SHx - y||^2 + \lambda s(x)
\]

where

- \( x \in \mathbb{R}^x \) is the restored-signal candidate.
- \( s() \) is a regularizer, which returns a lower value for a more likely candidate solution, weighted by the parameter \( \lambda \geq 0 \).
SR-CNN: The Proposed Design

The Convolutional Neural Network for Super Resolution:

Patch extraction and representation:
\[ F_1(Y) = \max(0, W_1 \ast Y + B_1) \]
where \( W_1 \) corresponds to \( n_1 \) filters of support \( c \times f_1 \times f_1 \)

Non-linear mapping:
\[ F_2(Y'') = \max(0, W_2 \ast F_1(Y') + B_2) \]
where \( W_2 \) corresponds to \( n_2 \) filters of support \( n_1 \times f_2 \times f_2 \)

Reconstruction:
\[ F(Y) = W_3 + F_2(Y'') + B_3 \]
where \( W_3 \) corresponds to \( c \) filters of support \( n_1 \times f_3 \times f_3 \)

The result is a high-resolution image (of \( c \) color channels).

Sparsity-based Super-Resolution as an Instance of The Proposed SR-CNN

Yang et al. (2010) proposed the following fundamental principles for super-resolution using sparse representations:

- Use pairs of low and high resolution patches for training.
- Jointly learn high and low resolution dictionaries, such that using the same sparse representation vector we can reconstruct a patch in its low and high resolutions.

The interpretation of the sparsity-based method as an instance of the SR-CNN conceptual design:

- Patch extraction: Project each patch on \( n_2 \) components of a low-resolution dictionary. Each dictionary component can be considered as a linear filter.
- Non-linear mapping: Sparse coding of each \( n_2 \)-length vector, resulting in a vector of size \( n_2 = n_2 \). Since sparse coding is an iterative procedure, it is not feed forward – in contrast to the proposed approach.
- Reconstruction: Project each \( n_2 \)-coefficient vector on a high-resolution dictionary. These high-resolution patches are averaged.

The proposed CNN-based framework is better than the sparsity-based approach since:

- It is more general and flexible.
- Fully feed-forward, permitting an end-to-end optimization.
SR-CNN: Network Training

The end-to-end mapping, $F_r$, is determined by learning the network parameters: $\theta = \{W_1, W_2, B_1, B_2, B_3\}$.

The training gets:
- A set of high-resolution images: $\{X_i\}_{i=1}^N$.
- The corresponding low-resolution images: $\{Y_i\}_{i=1}^N$.

The learning loss function:
$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \| Y_i - X_i \|_2$$

Loss minimization via Stochastic Gradient Descent with standard backpropagation.

In practice, the training data is:
- $\{X_i\}_{i=1}^N$ are sub-images of $33 \times 33$ pixels.
- $\{Y_i\}_{i=1}^N$ are obtained by Gaussian blurring + subsampling + bicubic upscaling.

Application/Testing suits to images of arbitrary sizes.

SR-CNN: Experiments

Comparing training with different database sizes:

The effect of "big data" for training is moderate:
- Training on 91 images provided testing PSNR of 32.39 dB.
- Training on ImageNet provided testing PSNR of 32.52 dB.

Network settings:
- $f_1 = 9$
- $f_2 = 1$
- $f_3 = 5$
- $k = 64$
- $k_1 = 44$

The Effect of Network "width" (Number of Filters):

In general, more filters lead to:
- Increase in performance (output PSNR).
- Longer run-time.

<table>
<thead>
<tr>
<th>Filters in Layer 1</th>
<th>Filters in Layer 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_1 = 128$</td>
<td>$n_1 = 64$</td>
</tr>
<tr>
<td>$n_2 = 64$</td>
<td>$n_2 = 32$</td>
</tr>
<tr>
<td>$n_3 = 32$</td>
<td>$n_2 = 16$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PSNR</th>
<th>Time (sec)</th>
<th>PSNR</th>
<th>Time (sec)</th>
<th>PSNR</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.60</td>
<td>0.60</td>
<td>32.52</td>
<td>0.18</td>
<td>32.26</td>
<td>0.05</td>
</tr>
</tbody>
</table>
SR-CNN: Experiments

The Effect of the Filter Size:

Increasing the second-layer filter size is beneficial!
- Neighborhood information in the non-linear mapping stage is useful.
- Note that larger filter sizes increase the run-time.
- Experiments for larger $f_2$ and $f_3$ (where $f_2 = 1$) showed a marginal PSNR gain.

SR-CNN: Experiments

Comparison to state-of-the-art super-resolution methods:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FTN</td>
<td>3</td>
<td>26.90</td>
<td>26.67</td>
<td>26.72</td>
<td>27.54</td>
<td>30.16</td>
<td>30.24</td>
</tr>
<tr>
<td>4</td>
<td>25.94</td>
<td>26.67</td>
<td>26.72</td>
<td>27.54</td>
<td>23.18</td>
<td>23.18</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>26.67</td>
<td>25.33</td>
<td>25.33</td>
<td>25.33</td>
<td>23.48</td>
<td>23.48</td>
<td></td>
</tr>
<tr>
<td>SISR</td>
<td>2</td>
<td>0.8524</td>
<td>0.8086</td>
<td>0.8035</td>
<td>0.8035</td>
<td>0.8035</td>
<td>0.8035</td>
</tr>
<tr>
<td>4</td>
<td>0.7640</td>
<td>0.7640</td>
<td>0.7640</td>
<td>0.7640</td>
<td>0.7640</td>
<td>0.7640</td>
<td></td>
</tr>
</tbody>
</table>

Tested on 200 images (the "BSD200" dataset).

SR-CNN: Experiments

The Effect of the Network Depth (Number of Layers):

- The 4-layer networks converge slower than the 3-layer network.
- Sufficient training time will bring the deeper networks to the 3-layer performance.
- Authors' claim: Deeper networks do not necessarily provide better super-resolution performance.
**SR-CNN: Experiments**

Upscaling by 3

**SR-CNN: Conclusions**

- A fully convolutional neural network for image super resolution was presented.
  - Relying on learning an end-to-end mapping between low and high resolution images.
- Super-resolution using sparse representations was exhibited as a particular realization of the proposed CNN-based framework.
- Compete with the leading super-resolution methods in terms of PSNR and SSIM.

**SR-CNN: Drawbacks and Improvements**

**SR-CNN weaknesses**
- Considers relatively local region information.
- Slow convergence of training.
- A trained network suits for a single upscaling factor.

**Improved design by Kim et al. (CVPR 2016)**:
- Large receptive fields via a very deep convolutional net.
  - 20 layers of 3x3 filters provides receptive field of 41x41 size (SR-CNN with 9-1-5 setting has a receptive field of 13x13).
- Residual-learning CNN
  - The residual between high-res to low-res is provided by the network.
  - Reconstruction by adding the residual to the low-res input.
  - Propose a single CNN for multi-scale super-resolution.

**Improved Network Design by Kim et al.**

Figure taken from Kim et al. (2016)
Part II: Compression-Artifact Reduction

Based on the paper:
Dong et al., "Compression Artifacts Reduction by a Deep Convolutional Network", ICCV 2015.

Image Compression Artifacts

- Artifacts are usually applied by postprocessing the decompressed image.
- Compression artifacts may degrade low-level vision tasks such as edge detection, super resolution, and more.
- The artifacts severity increases as the compression bit-rate reduces.
- The artifact type stems from the compression architecture:
  - JPEG → blockiness
  - JPEG2000 → ringing + blur
  - HEVC (stills) → blur + some blockiness

Paper Contributions

(Dong et al., 2015)

- Propose a deep convolutional network for artifact reduction in compressed images.
  - Learns an end-to-end mapping between precompressed and decompressed images.
- Apply transfer learning for the following easy-to-hard settings:
  - Transfer shallow to deeper model.
  - Transfer high to low compression quality.
  - Transfer standard compression to real use case.
- Compete with the state-of-the-art compression-artifact reduction techniques for JPEG compression.
Artifact Reduction CNN: Network Depth

Due to the intricate nature of compression artifacts (e.g., blockiness):

• 3-layer network (as for super-resolution) was found insufficient.

• Adding at least one convolutional layer was found beneficial.

The authors’ interpretation:

Observation:
The complicated artifacts lead to noisy patterns in features extracted by the first layer.

The solution:
Follow the first layer by one or more “feature enhancement” layers to clean the noisy features.

Artifact Reduction CNN: Network Structure

Four convolutional layers:

\[
F_1(Y') = \max(0, W_1 \cdot Y + B_1) \\
F_2(Y') = \max(0, W_2 \cdot F_1(Y') + B_2) \\
F_3(Y') = \max(0, W_3 \cdot F_2(Y') + B_2) \\
F(Y) = W_3 \cdot F_3(Y') + B_3
\]

Artifact Reduction CNN: Network Training

The end-to-end mapping, \( F \), is determined by learning the network parameters: \( \theta = \{ W_1, W_2, W_3, B_1, B_2, B_3 \} \).

The training gets:

• A set of ground-truth images: \( (X_i)^{n}_{i=1} \)

• The corresponding decompressed images: \( (Y_i)^{n}_{i=1} \)

The learning loss function:

\[
L(\theta) = \frac{1}{N} \sum_{i=1}^{N} || F(Y_i; \theta) - X_i ||^2
\]

Loss minimization via Stochastic Gradient Descent with standard back propagation.

In experiments, the training data is:

• \( (X_i)^{n}_{i=1} \) are sub-images of 32 × 32 pixels.

• \( (Y_i)^{n}_{i=1} \) are obtained by JPEG compression.

Application/Testing suits to images of arbitrary sizes.

Artifact Reduction CNN: Training via Easy-to-Hard Transfer

• Simple initialization of the parameters (e.g., Gaussian-distributed random values) leads to inadequate learning results.

• The authors suggested to initialize the first layers with values obtained by training an easier artifact-reduction task network:

The easy network (to transfer from):

Shallow to deeper model:

High to low compression quality:

Standard compression to real use case:

In experiments, the training data is:

• \( (X_i)^{n}_{i=1} \) are sub-images of 32 × 32 pixels.

• \( (Y_i)^{n}_{i=1} \) are obtained by JPEG compression.

Application/Testing suits to images of arbitrary sizes.
Artifact Reduction CNN: Training via Easy-to-Hard Transfer

The authors justified transferring filters from **high to low compression qualities** by showing the similarities in the learned filters:

(a) High compression quality (quality 20 in Matlab encoder)

(b) Low compression quality (quality 10 in Matlab encoder)

The above correspondingly-dotted filters are considered to be similar.

Artifact Reduction CNN: Experimental Results

<table>
<thead>
<tr>
<th>Eval. Mat</th>
<th>Quality</th>
<th>JPEG</th>
<th>SA-DCT</th>
<th>AR-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>31.70</td>
<td>28.65</td>
<td>31.29</td>
<td>31.29</td>
</tr>
<tr>
<td>20</td>
<td>30.67</td>
<td>30.61</td>
<td>30.61</td>
<td>30.61</td>
</tr>
<tr>
<td>30</td>
<td>29.41</td>
<td>28.06</td>
<td>32.08</td>
<td>32.08</td>
</tr>
<tr>
<td>40</td>
<td>22.35</td>
<td>22.35</td>
<td>32.99</td>
<td>32.99</td>
</tr>
</tbody>
</table>

| SSIM      |         |      |        |        |
| 10        | 0.8033  | 0.8033| 0.8217 | 0.8217 |
| 20        | 0.8863  | 0.8863| 0.8871 | 0.8871 |
| 30        | 0.9006  | 0.9006| 0.9106 | 0.9106 |
| 40        | 0.9173  | 0.9173| 0.9306 | 0.9306 |

**Artificial Reduction CNN: Conclusion**

- A **four-layer convolutional network** was proposed for reducing artifacts in decompressed images.
- A **transfer-learning** approach was suggested for improving training of **complicated** artifact-reduction tasks **using** initializations obtained from training an **easier** AR-CNN.
- Experiments showed that AR-CNN **compete** with artifact reduction techniques for JPEG compression.
  - Also showed benefits upon using the SR-CNN structure.
Artifact Reduction CNN: Conclusion

**Drawbacks:**

- The AR-CNN achieves moderate improvements over the SA-DCT method (from 2007). However, there are newer (non-DNN) methods that it does not necessarily surpass (e.g., Zhang et al., IEEE TIP, 2013).

- The proposed approach is not expected to work that well on compression methods that operate on large blocks, e.g.,
  - JPEG2000 that works on tiles of at least 128x128 pixels.
  - HEVC-stills that considers large blocks (e.g., 64x64 pixels) and adaptively partitions them in a recursive manner.

While AR-CNN is not suitable for JPEG2000/HEVC, these where treated by various non-DNN artifact-reduction methods.