Digital Image Processing, Spring 2018

Final Report
DUE DATE: June 27, 2018

Team 23: R06945003 林鈺盛 B03902129 陳鵬宇

Paper title

Title: Fast Image Processing with Fully-Convolutional Networks
Conference: ICCV 2017
Authors: Qifeng Chen, Jai Xu Intel Labs and Vladlen Koltun

Motivation

Image processing can be used in everywhere. Like robotic navigation systems, traffic, healthcare, camera edge technology, etc. Faster image processing can save us tons of time. Learning how to accelerate the procedure of image processing is undoubtedly worthy.

Problem definition

Use convolution neural network (CNN) to implement image processing methods mentioned in class.

- Pencil Drawing
- Multi-scale Toning
- Photographic Style
- Nonlocal Dehazing

Algorithm

Preliminaries

Given:

- $\mathbf{I}$: image in RGB space and
- f: operator, where $\mathbf{I}$ and $f(\mathbf{I})$ have the same resolution.

Our goal is:

- to approximate f with another operator $\hat{f}$, that is $f(\mathbf{I}) \approx \hat{f}(\mathbf{I})$ for all images and
- to find a broadly applicable image processing operator.

There are three desirable criteria we want to meet with:

1. Accuracy
2. Speed
3. Compactness

Context Aggregation Networks (CAN)

The primary architecture the paper used is **multi-scale context aggregation network (CAN)**
Now let's describe the parameterization in detail.
The data lay out over layers: $\{\mathbf{L}^0, \ldots, \mathbf{L}^d\}$.

- Dimension of $\mathbf{L}^0$ (input) and $\mathbf{L}^d$ (output): $m \times n \times 3$.
- Dimension of $\mathbf{L}^s$ $(1 \leq s \leq d-1)$: $m \times n \times w$, where w is the width (features) of each layer.

$\mathbf{L}^s$ can be computed from the previous layer $\mathbf{L}^{s-1}$ as follows:

$$\mathbf{L}_i^s = \Phi\left(\Psi^s\left(b_i^s + \sum_j \mathbf{L}_j^{s-1} *_{r_s} \mathbf{K}_{i,j}^s\right)\right)$$

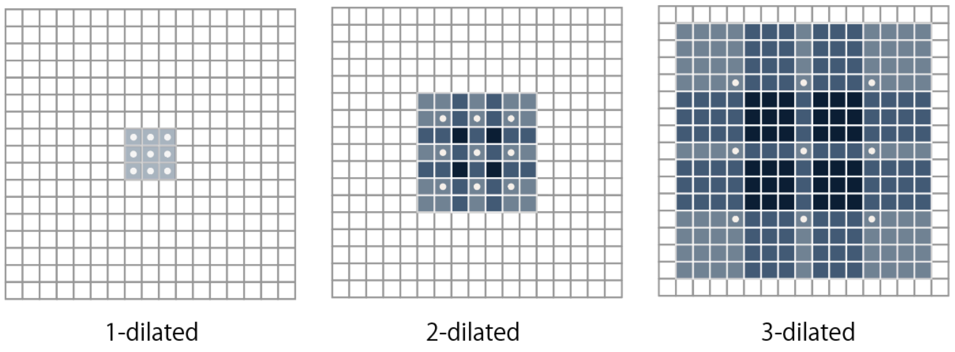
and $\Psi^s(x)$ is computed as follow:

$$\Psi^s(x) = \lambda_s x + \mu_s BN(x),$$

where $\lambda_s, \mu_s \in \mathbb{R}$ are learned scalar weights and BN is the batch normalization operator. We'll keep training these two scalars for each iteration.
Specically, for image coordinates x:

$$(\mathbf{L}_j^{s-1} *_{r_s} \mathbf{K}_{i,j}^s)(x) = \sum_{a + r_s b = x} \mathbf{L}_j^{s-1}(a)\mathbf{K}_{i,j}^s(b)$$

Performing dilated convolution needs to specify the kernel size, it looks like:



Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1,\ldots,m}\}$; Parameters to be learned: λ, β
Output: $\{y_i = BN_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{normalize}$$

$$y_i \gamma \hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i) \qquad \text{scale and shift}$$

Activation function (Leaky ReLU)

- Original: $\Phi(x) = \max(0.01x, x)$
- In this paper: $\Phi(x) = \max(0.2x, x)$

Training

The network is trained on a set of input-output pairs that contain images before and after the application of original operator: $\mathcal{D} = \{\mathbf{I}_i, f(\mathbf{I}_i)\}$.

- the parameters of the network are the kernel weights $\mathcal{K} = \{\mathbf{K}^s_{i,j}\}_{s,i,j}$ and
- the scalar biases $\mathcal{B} = \{b^s_i\}_{s,i}$.

These parameters are optimized to fit the action of the operator f across all images in the training set. We train with an image-space regression loss:

$$\ell(\mathcal{K}, \mathcal{B}) = \sum_i \frac{1}{N_i} ||\hat{f}(\mathbf{I}_i; \mathcal{K}, \mathcal{B}) - f(\mathbf{I}_i)||^2,$$

where N_i is the number of pixels in image $\mathbf{I}_i$.

Parameters

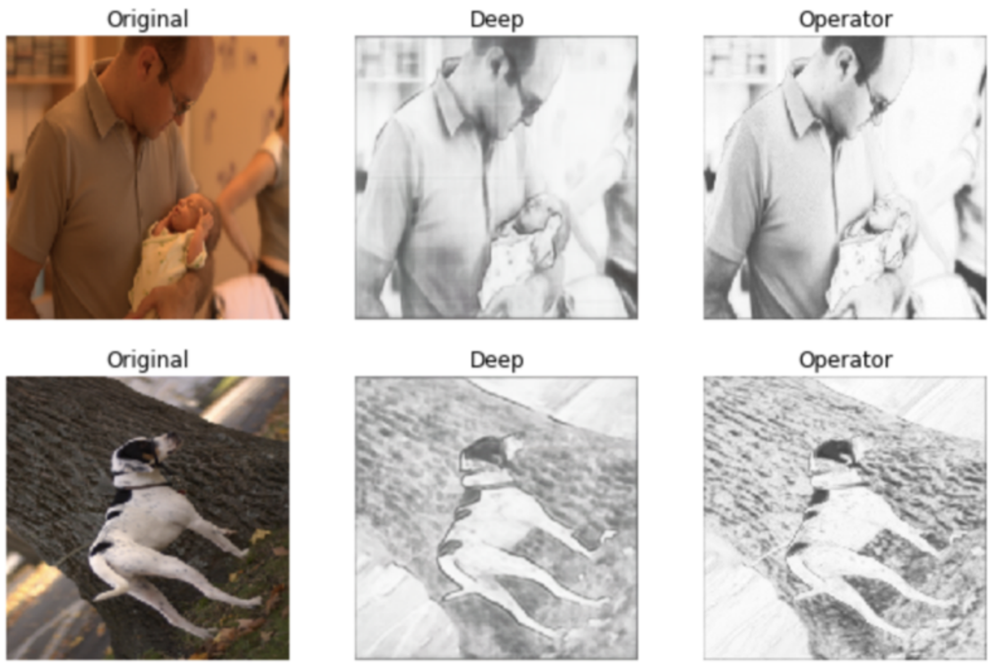
- epochs: 200
- batch size: 16
- optimizer: Adam(lr = 0.0003)
- loss function: MSE

Data spec

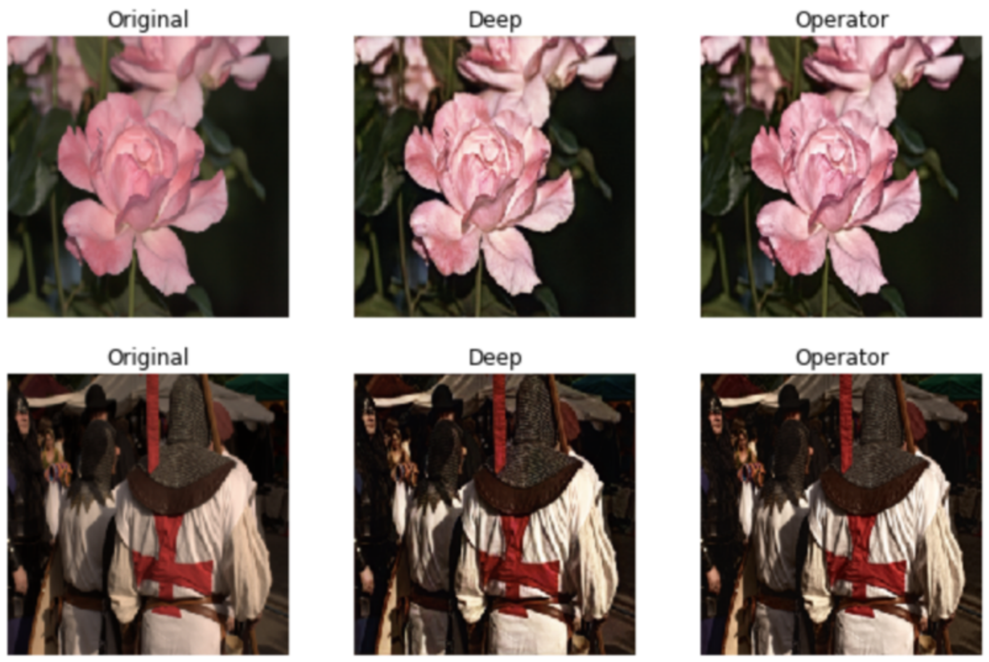
- MIT-Adobe 5k
- 2500/2500 training/test set
- 250 for validation
- Resize to 256×256 for training

Results

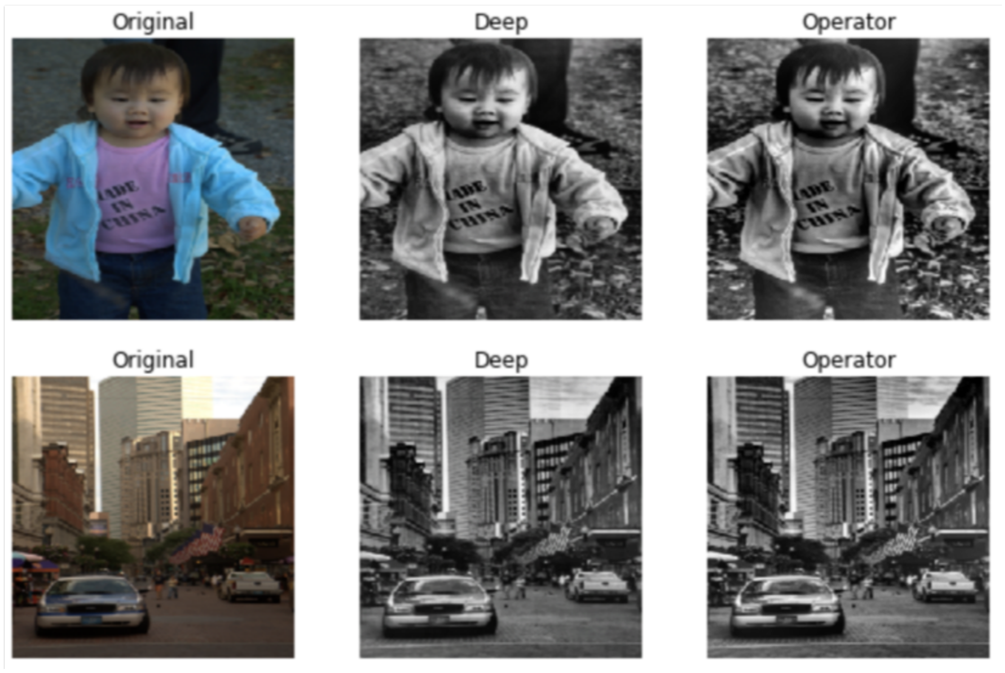
Pencil Drawing



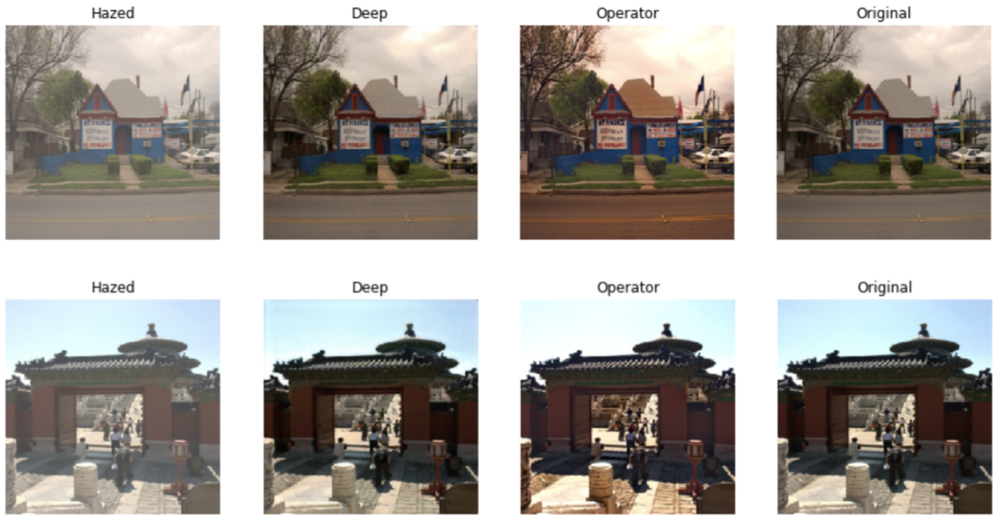
Multi-scale Toning



Photographic Style



Nonlocal Dehazing



Discussion

- With sufficient computing resources, high-resolution images may be used for training
- Run time on predicting 2500 test images:
  - Operator: 2919 sec
  - Deep model without GPU: 1814 sec
  - Deep model with GPU: 70 sec
- PSNR of Dehazing task:

  - Deep model：29.32
  - Operator：17.64

Reference

[1] Fast Image Processing with Fully-Convolutional Networks
[2] Multi-scale context aggregation by dilated convolutions