Reading Report

2019.6.30

1 BN, IN, GN and LN

A family of feature normalization methods, including BN, LN, IN and GN, perform the following computation:

$$x_i = \frac{1}{\sigma_i}(x_i - \mu_i)$$

x means the feature computed by a layer, and i is the index.
In the case of 2D images, $i = (i_N, i_C, i_H, i_W)$ is a 4D vector indexing the features in (N, C, H, W) order, where N is the batch axis, C is the channel axis, and H and W are the spatial height and width axes.
μ and σ are the mean and standard deviation

$$\mu_i = \frac{1}{m} \sum_{k \in S_i} x_k$$

$$\sigma_i = \sqrt{\frac{1}{m} \sum_{K \in S_i} (x_k - \mu_i)^2 + \sigma}$$

$S_i$is the set of pixels in which the mean and standard are computed. The m is the size of the set.
Many types of feature normalization methods mainly differ in how the set S_i is defined

1.1 Batch Normalization

$$S_i = \{k | k_C = i_C\}$$

where i_C donates the sub-index of i along the C axis. This means that the pixels sharing the same channel index are normalized together BN normalizes the features by the mean and variance computed within a (mini-)batch.

1.2 Instance Normalization

$$S_i = \{k|k_N = i_N, k_C = i_C\}$$

meaning that IN computes μ and σ along the (H, W) axes for each sample.

1.3 Group Normalization

$$S_i = \{k|k_N = i_N, \lfloor \frac{k_C}{c/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor\}$$

Here G is the number of groups, which is a pre-defined hyper-parameter. C/G is the number of channels per group.

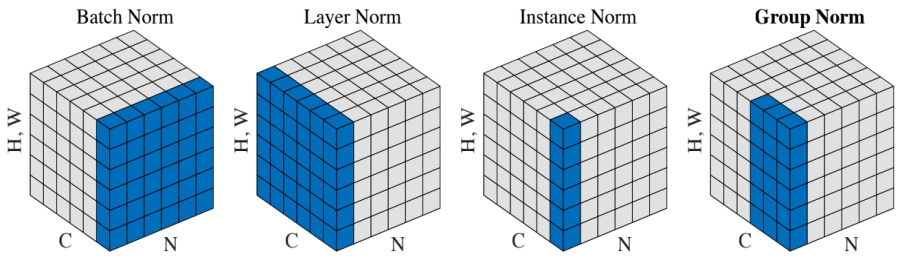
1.4 Layer Normalization

$$S_i = \{k|k_N = i_N\}$$

meaning that LN computes μ and σ along the (C, H, W) axes for each sample.

1.5 Comparison

methods.png methods.bb



2 Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising

1.An end-to-end trainable CNN network was proposed, and the residual learning strategy was adopted to implicitly remove clean images from the hidden layer of the network. That is, input is noisy image and output is residual image with clean image removed.

The motivation is that using residual learning to describe identity maps or approximate identity maps is better than learning clean pictures directly
2. find that residual learning and batch normalization can greatly benefit the CNN learning as they can not only speed up the training but also boost the denoising performance. For Gaussian denoising with a certain noise level, DnCNN outperforms state-of-the-art methods in terms of both quantitative metrics and visual quality
3.can be used for many noises, including Gaussian, Possion, Bernouli noises and other synthetic noises, like Multiplicative Bernoulli noise, Text removal, Random-valued impulse noise .

2.1 the architecture of the proposed DnCNN network

1. loss function

```
loss =K.sum(K.square(y_pred - y_true))/2
```

2. architecture

```
def DnCNN(depth,filters=64,image_channels=1, use_bnorm=True):
    layer_count = 0
    inpt = Input(shape=(None,None,image_channels),name =
        'input'+str(layer_count))
    # 1st layer, Conv+relu
    layer_count += 1
    x = Conv2D(filters=filters, kernel_size=(3,3),
        strides=(1,1),kernel_initializer='Orthogonal', padding='same',name =
        'conv'+str(layer_count))(inpt)
    layer_count += 1
    x = Activation('relu',name = 'relu'+str(layer_count))(x)
    # depth-2 layers, Conv+BN+relu
    for i in range(depth-2):
        layer_count += 1
        x = Conv2D(filters=filters, kernel_size=(3,3),
            strides=(1,1),kernel_initializer='Orthogonal',
            padding='same',use_bias = False,name = 'conv'+str(layer_count))(x)
        if use_bnorm:
            layer_count += 1
    #x = BatchNormalization(axis=3, momentum=0.1,epsilon=0.0001, name =
        'bn'+str(layer_count))(x)
```

```
    x = BatchNormalization(axis=3, momentum=0.0,epsilon=0.0001, name =
        'bn'+str(layer_count))(x)
    layer_count += 1
    x = Activation('relu',name = 'relu'+str(layer_count))(x)
# last layer, Conv
layer_count += 1
x = Conv2D(filters=image_channels, kernel_size=(3,3), strides=(1,1),
    kernel_initializer='Orthogonal',padding='same',use_bias = False,name =
    'conv'+str(layer_count))(x)
layer_count += 1
x = Subtract(name = 'subtract' + str(layer_count))([inpt, x])
# input - noise
model = Model(inputs=inpt, outputs=x)

return model
```

3 Noise2Noise: Learning Image Restoration without Clean Data

to find a number z that has the smallest average deviation from the measurements according to some loss function L:

$$\underset{z}{argmin}\, E_y \{L(z, y)\}$$

maybe L_0, L_1, or L_2 loss.
The general class of deviation-minimizing estimators are known as M-estimators. From a statistical viewpoint, summary estimation using these common loss functions can be seen as ML estimation by interpreting the loss function as the negative log likelihood.
Training neural network regressors is a generalization of this point estimation procedure

$$\underset{\theta}{argmin}\, E_{(x,y)} \{L(f_\theta(x), y)\}$$

which also can be represented by

$$\underset{\theta}{argmin}\, E_x\{E_{y|x} \{L(f_\theta(x), y)\}\}$$

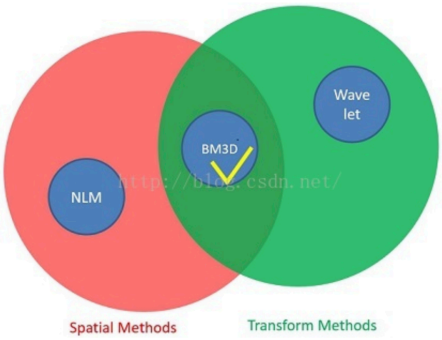
The observation is that for certain problems this tendency has an unexpected benefit. A trivial, and, at first sight, useless, property of L2 minimization is that on expectation, the estimate remains unchanged if we replace the targets with random numbers whose expectations match the targets.
This implies that corrupt the training targets of a neural network with zero-mean noise without changing what the network learns. Combining this with the corrupted inputs, we are left with the empirical risk minimization task.

$$\underset{\theta}{argmin} \sum_{i} L(f_\theta(\widehat{x_i}), \widehat{y_i})$$

where both the inputs and the targets are now drawn from a corrupted distribution.Interestingly, none of the above relies on a likelihood model of the corruption, nor a density model (prior) for the underlying clean image manifold.

4 Basic Denoise methods

4.1 BM3D



1. Basic estimate
   1.1 Block-wise estimate
   (a) Grouping Find blocks that are similar to the currently processed one and then stack them together in a 3D array (group).
   (b) Collaborative hard-thresholding Apply a 3D transform to the formed group, attenuate the noise by hard-thresholding of the transform coefficients, invert 3D transform to produce estimates of all grouped blocks, and return the estimates of the blocks to their origial positions
   1.2 Aggregation

2. Final estimate

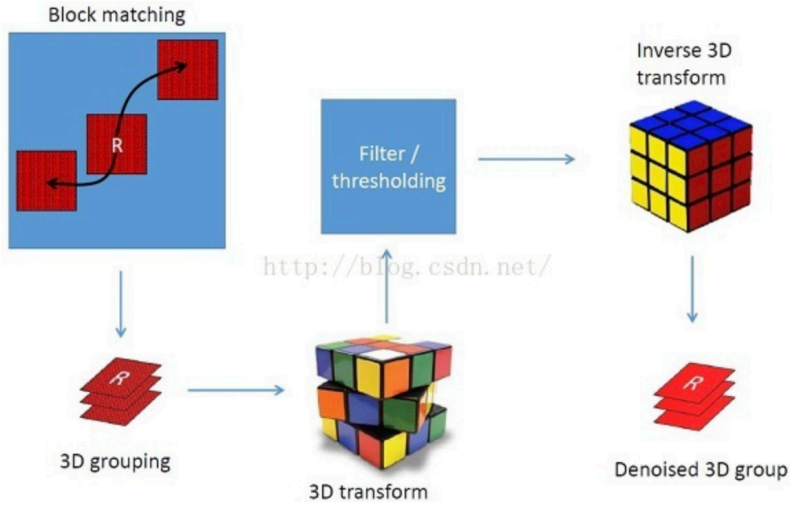
   2.1 Block-wise estimate

   (a) Grouping Use BM within the basic estimate to nd the locations of the blocks similar to the currently processed one. Using these locations,form two groups (3D arrays), one from the noisy image and one from the basic estimate.

   (b) Collaborative Wiener filter Apply a 3D transform on both groups. Perform Wiener filtering on the noisy one using the energy spectrum of the basic estimate as the true energy spectrum of the basic estimate as the true energy spectrum.

   2.2Aggregation

approximately process of BM3D.png approximately process of BM3D.bb



wiener filtering

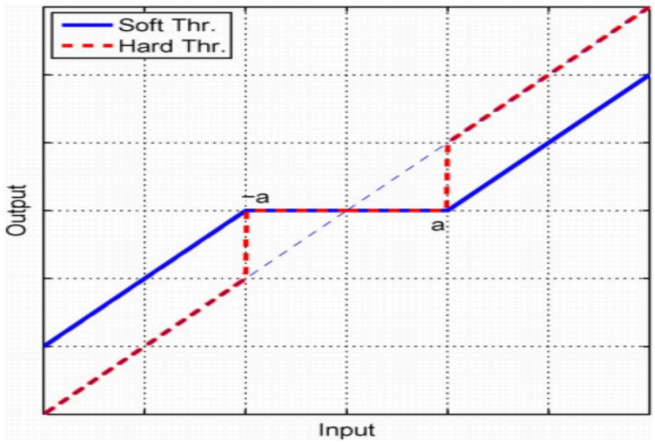
the Wiener filter is a filter used to produce an estimate of a desired or target random process by linear time-invariant (LTI) filtering of an observed noisy process, assuming known stationary signal and noise spectra, and additive noise. The Wiener filter minimizes the mean square error between the estimated random process and the desired process.

Hard Thresholding

4.2 DCT

A discrete cosine transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from lossy compression of audio (e.g. MP3)

and soft threhold.png and soft threhold.bb



and images (e.g. JPEG) (where small high-frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations.

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0, \ldots, N-1$$

5 Data Sets and Data Process

5.1 Data Sets

Currently, the establishment of denoising data sets can be mainly divided into the following three ways:

(1).Obtain high-quality images from existing image databases, and then do image processing (such as linear change and brightness adjustment) and add artificial synthetic noise to generate noise images according to the noise model

t is simple and time-saving, and high-quality images can be obtained directly from the Internet. However, due to the artificial synthesis of noise, the network trained on this data set has certain differences from the real noise images, so the de-noising effect on the real noise images is limited

(2).For the same scene, shoot low ISO image as ground truth and high ISO image as noise image, and adjust the exposure time and other camera parameters to make the brightness of the two images consistent Process

If a single low-iso image is used as the ground truth, residual noise will inevitably occur, and there may be brightness difference and misalignment with the noisy image

(3).Shoot multiple images of the same scene continuously, then do image processing (such as image registration, abnormal image elimination, etc.), and then synthesize ground

truth with weighted average.
The amount of work is relatively large, and the image needs to be strictly aligned, but the generally obtained ground truth is of high quality.

Gaussian Denoising

BSD68

Single Image Super-Resolution

Set5 Set14 BSD100 Urban100

JPEG Image Deblocking

Classic5 LIVE1

5.2 Evaluation function

PSNR

Peak signal-to-noise ratio. The higher PSNR, the better picture.

$$PSNR = 10 * log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

MAX_I is the maximum number of picture Matrix.

SSIM

structural similarity index

$$\mu_X = \frac{1}{R*C}\sum_{i=1}^{R}\sum_{j=1}^{C} X(i,j)$$

$${\sigma_X}^2 = \frac{1}{R*C-1}\sum_{i=1}^{R}\sum_{j=1}^{C} \left(X(i,j)-u_X\right)^2$$

$${\sigma_{XY}}^2 = \frac{1}{R*C-1}\sum_{i=1}^{R}\sum_{j=1}^{C} \left(X(i,j)-u_X\right)\left(Y(i,j)-u_Y\right)$$

$$L(X,Y) = \frac{2\mu_X\mu_Y + C1}{{\mu_X}^2 + {\mu_Y}^2 + C1}$$

$$C(X,Y) = \frac{2\sigma_X\sigma_Y + C2}{{\sigma_X}^2 + {\sigma_Y}^2 + C2}$$

$$S(X,Y) = \frac{\sigma_{XY}^2 + C3}{\sigma_X\sigma_Y + C3}$$

$$SSIM(X,Y) = L(X,Y) * C(X,Y) * S(X,Y)$$

$C1 = (k_1L)^2$,$C2 = (k_2L)^2$
as usual, $k_1 = 0.01$, $k_2 = 0.03$,$L = 2^{bitsperpixel} - 1$

MSE

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

MAE

$$MAE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}|I(i,j) - K(i,j)|$$

6 My codes and results

6.1 My Codes

WGAN

https://github.com/SherryCal/weekly-reading-report.io/blob/master/codes/WGAN.py

pix-to-pix GAN denoise

https://github.com/SherryCal/weekly-reading-report.io/blob/master/codes/pixel-to-pixel

6.2 Results

I plan to read some about data sets analysis and result evaluation.(use GOPRO Large)