Simple image deconvolution based on reverse image convolution and backpropagation algorithm

Loc Nguyen
Loc Nguyen's Academic Network, Vietnam
Email: ng_phloc@yahoo.com
Homepage: www.locnguyen.net

Abstract
Deconvolution task is not important in convolutional neural network (CNN) because it is not imperative to recover convoluted image when convolutional layer is important to extract features. However, the deconvolution task is useful in some cases of inspecting and reflecting a convolutional filter as well as trying to improve a generated image when information loss is not serious with regard to trade-off of information loss and specific features such as edge detection and sharpening. This research proposes a duplicated and reverse process of recovering a filtered image. Firstly, source layer and target layer are reversed in accordance with traditional image convolution so as to train the convolutional filter. Secondly, the trained filter is reversed again to derive a deconvolutional operator for recovering the filtered image. The reverse process is associated with backpropagation algorithm which is most popular in learning neural network. Experimental results show that the proposed technique in this research is better to learn the filters that focus on discovering pixel differences. Therefore, the main contribution of this research is to inspect convolutional filters from data.
Keywords: convolutional neural network, convolutional filter, image deconvolution, backpropagation algorithm.

1. Introduction
Convolutional operator or convolutional product which is main operator of convolutional neural network (CNN) does not imply any degradation because it plays the most important role of extracting features of images. However, in some cases, it is necessary to make a so-called deconvolutional operator which is the opposite of convolutional operator in order to recover original images. Especially, deep generative models may need to reinterpret original image from decoded image when encoded image was transformed by convolutional operator. In general, this research focuses on image deconvolution which is not popular in CNN area but there are some typical researches related to image deconvolution. Domain of single image super-resolution (SISR), which is not like the viewpoint of this research, focuses seriously on reconstructing high-resolution image (HR) from low-resolution image (LR). According to Cao et al. (Cao, Yao, & Liang, 2020, p. 394), there are three types of SISR: 1) interpolation-based approach focuses on calculating missing pixels of HR by interpolation equations on pixels of LR, 2) reconstruction-based approach focuses on taking advantages of special aspects of HR such as gradient profile, edge features, and nonlocal means in order to recover missing pixels of HR from LR, and 3) learning-based approach focuses on discovering the relationship between HR and LR by comparing datasets of both HR and LR. Kim and Kwon (Kim & Kwon, 2010, p. 1129), whose work is slightly similar to this research, proposed a repression method to learn a so-called reproducing kernel for minimizing loss function between transformed LR and HR. Please pay attention that their reproducing kernel, which is not traditional filter kernel in CNN, is Gaussian filter function but, essentially, their reproducing kernel is the kernel which is more complex than the matrix kernel of CNN. Actually, the reproducing kernel is used to transform LR into an intermediate form which is in turn compared with HR. According to Cao et al. (Cao, Yao, & Liang, 2020, p. 395), recently researches, which belong to learning-based

approach, are applying CNN into SISR where the transformation from LR to HR is represented by a deep CNN. Concretely, Cao et al. (Cao, Yao, & Liang, 2020, pp. 395-396) proposed a fully networking includes three stages as three layer stacks for image enhancement such as nonlinear enhancement, multiscale feature restoration, and fusion enhancement. Moreover, Cao et al. (Cao, Yao, & Liang, 2020, p. 399) added Kullback-Leibler divergence into loss function for improving CNN training process. In deferent viewpoint, autoencoder (AE) domain which is most similar to this research considers the image deconvolution as decoding process (decoder) of encoded image but please pay attention that AE is not Variational Autoencoders (VAE) generative model. AE assumes that the degradation of decoded image is not concerned because AE tries to restore original image as well as possible. The AE deconvolution is expressed by following propagation rule:

$$y = f\left(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_{ij} + b\right)$$

Where $\boldsymbol{x}$ is source layer whose pixels are denoted x_{ij}, $\boldsymbol{y}$ is target layer (deconvoluted layer) whose pixels are denoted y, $f(x)$ is activation function, $\boldsymbol{w} = (w_{ij})$ is deconvolutional filter, and b is deconvolutional bias. Indeed, AE methodology is almost the same as this research when the filter $\boldsymbol{w}$ is learned by minimizing Euclidean loss function and backpropagation algorithm except that the target layer in AE is not formally accepted to be degraded. Works of Turchenko et al. (Turchenko, Chalmers, & Luczak, 2017, p. 4) and Xu et al. (Xu, Ren, Liu, & Jia, 2014) are typical works in AE domain but Xu et al. (Xu, Ren, Liu, & Jia, 2014, pp. 3-4) went beyond by transforming convolutional space into frequency space via Fourier transformation.

In general, the aspect of this research, which is different from other research, is that I focus on learning convolutional filter from two convolutional layers so as to contribute such filter to the deconvolution process whereas other researches focused on improving or keeping quality of image. Obviously, image quality improvement is much more important, but this research aims to a viewpoint of mutual relationship between convolutional task and deconvolutional task, which attaches to backpropagation algorithm. Moreover, degradation in image quality is obviously accepted in this research. Therefore, this research has two purposes: 1) training convolutional filter and 2) making the image deconvolution by the trained filter. The trained filter is learned from the process of reversing image convolution and the proposed image deconvolution is a reverse process based on the trained filter too.

2. Methodology

Main layer of convolutional neural network (CNN) is convolutional layer which performs convolutional operator based on a so-called convolutional filter which is a $n_x n$ squared matrix $\boldsymbol{u} = (u_{ij})_{nxn}$. Given a convolutional source layer $\boldsymbol{x}$ represented by its pixel x_{ij} which are operated with the filter $\boldsymbol{u}$ by the convolution operator in order to produce target layer $\boldsymbol{y}$ represented by a resulted pixel y.

$$y = f\left(\sum_{i=1}^{n}\sum_{j=1}^{n} u_{ij}x_{ij} + b\right)$$

Where $f(x)$ is activation function of layer $\boldsymbol{x}$ and its derivative is denoted $f'(x)$. Obviously, in context of CNN, the target layer $\boldsymbol{y}$ will obtain some aspects which depend on specific filters, which does not imply any degradation but in some cases, it is necessary to learn the filter $\boldsymbol{u}$ from source layer $\boldsymbol{x}$ and target layer $\boldsymbol{y}$ and then make the image deconvolution based on the learned filter with note that layer $\boldsymbol{y}$ is smaller than layer $\boldsymbol{x}$ in size. The trick here is to consider the deconvolution process as a reverse process of image convolution. Exactly, layer $\boldsymbol{y}$ becomes

source layer whereas layer $\boldsymbol{x}$ becomes target layer, and the convolutional operator is executed by a $n_x n$ squared matrix $\boldsymbol{w} = (w_{ij})_{nxn}$ as follows:

$$x = f\left(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}y_{ij} + b\right)$$

Please pay attention that layer $\boldsymbol{x}$ is now smaller than layer $\boldsymbol{y}$ in size. Note that the convolutional bias b is ignored without loss of generality in methodology of this research.

$$x = f\left(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}y_{ij}\right) \quad (1)$$

Given resulted pixel x', loss function of the convolutional process is:

$$l(w_{ij}) = \frac{1}{2}\|x' - x\|^2$$

Where the notation ||.|| denotes Euclidean norm. According to backpropagation algorithm, the filter $\boldsymbol{w}$ is learned by applying stochastic gradient descent to the loss function $l(w_{ij})$ as follows:

$$w_{ij} = w_{ij} - \gamma\frac{\partial l(w_{ij})}{\partial w_{ij}} = w_{ij} - \gamma\|x' - x\|f'(x)y_{ij} \quad (2)$$

Rectified linear unit is used as activation function in this research.

$$f(x) = \max(x, 0)$$
$$f'(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

The execution of backpropagation algorithm is slid over the source layer $\boldsymbol{y}$ by $n_x n$ blocks with note that γ ($0 < \gamma \leq 1$) is learning rate. After filter $\boldsymbol{w}$ is learned, the first purpose of this research is reached and then, the second purpose, which is easier, is to perform the image deconvolution based on the trained filter $\boldsymbol{w}$. Because $\boldsymbol{w}$ is a convolutional filter, obviously it cannot be applied into deconvolutional task. However, recall that the deconvolution process is considered as a reverse process of image convolution, the deconvolutional task will be performed by a reverse operator with the filter $\boldsymbol{w}$, in which layer $\boldsymbol{x}$ and layer $\boldsymbol{y}$ are reversed again in order to get back its original roles as source layer and target layer, respectively as follows:

$$y_{ij} = \frac{x - \sum_{k\neq i}\sum_{l\neq j} w_{kl}y_{kl}}{w_{ij}} \quad (3)$$

If y_{kl} does not exist, it is set to be x as a smoothing trick. Therefore, although the convolutional filter $\boldsymbol{w}$ cannot be applied into deconvolutional task, this research implies a duplicated transformation in order to take advantages of filter $\boldsymbol{w}$ for image deconvolution.

3. Experimental results and discussions

The experiment is performed on a laptop with CPU AMD64 4 processors, 4GB RAM, Windows 10, and Java 15. The dataset is a set of ten original 180x250 images and three 3_x3 convolution filters such as blur filter 1/9{{1, 1, 1}, {1, 1, 1}, {1, 1, 1}}, sharpening filter {{0, –1, 0}, {–1, 5, –1}, {{0, –1, 0}}, and edge-detection filter {{–1, –1, –1}, {–1, 8, –1}, {–1, –1, –1}} are tested. After these convolutional filters are executed, images cannot be recovered well except blur filter because filtered images are seriously modified and three times smaller. Therefore, filtered images are zoomed three times, which will be compared with deconvoluted images produced by the technique of reverse image deconvolution in this research. Exactly, let MAE0 be mean absolute error of a filtered image and an original image and let MAE be mean absolute error of a deconvoluted image and an original image.

$$\mathrm{MAE0} = \frac{1}{N}\sum_i \frac{1}{n_i}\sum_j |\mathrm{imageFiltered}[j] - \mathrm{image}[i][j]|$$
$$\mathrm{MAE} = \frac{1}{N}\sum_i \frac{1}{n_i}\sum_j |\mathrm{imageDecov}[j] - \mathrm{image}[i][j]| \quad (4)$$

Where notation |.| denotes absolute value, N is the number of images N=10, and n_i is the number of pixels of the i^{th} image. Obviously, image[i][j] denotes the j^{th} pixel of the i^{th} image with note that image, imageFiltered, and imageDecov are original image, filtered image, and deconvoluted image, respectively. For each filter, a so-called loss ratio r between MAE and MAE0 is compared. The smaller the loss ratio r is, the better the deconvolutional task is.

$$r = \frac{|\mathrm{MAE} - \mathrm{MAE0}|}{\mathrm{MAE0}} \quad (5)$$

The test is done with 19 learning rates γ = 1, 0.9,…, 0.1, 0.09, 0.001 because stochastic gradient descent (SGD) algorithm is affected by learning rate. Table 1 shows MAE, MAE0, and loss ratios of the three filters with regard to ten learning rates from 1 down to 0.1.

		MAE	MAE0	Loss
γ=1	Blur	0.2881	0.0727	296.1781%
	Sharpening	0.2200	0.1758	25.1110%
	Edge	0.5563	0.5482	1.4685%
γ=0.9	Blur	0.1988	0.0727	173.3775%
	Sharpening	0.2232	0.1758	26.9450%
	Edge	0.5538	0.5482	1.0093%
γ=0.8	Blur	0.3629	0.0727	398.9925%
	Sharpening	0.2579	0.1758	46.6521%
	Edge	0.5541	0.5482	1.0713%
γ=0.7	Blur	0.1246	0.0727	71.2817%
	Sharpening	0.2201	0.1758	25.1523%
	Edge	0.5558	0.5482	1.3837%
γ=0.6	Blur	0.0950	0.0727	30.5891%
	Sharpening	0.2329	0.1758	32.4306%
	Edge	0.5555	0.5482	1.3313%
γ=0.5	Blur	0.1300	0.0727	78.7666%
	Sharpening	0.2020	0.1758	14.8955%
	Edge	0.5570	0.5482	1.6023%
γ=0.4	Blur	0.1024	0.0727	40.7506%
	Sharpening	0.1976	0.1758	12.3684%
	Edge	0.5588	0.5482	1.9245%
γ=0.3	Blur	0.0707	0.0727	2.8118%
	Sharpening	0.1782	0.1758	1.3286%
	Edge	0.5592	0.5482	1.9991%
γ=0.2	Blur	0.0707	0.0727	2.8118%
	Sharpening	0.1927	0.1758	9.5685%
	Edge	0.5595	0.5482	2.0560%
γ=0.1	Blur	0.0707	0.0727	2.8118%
	Sharpening	0.1757	0.1758	0.0734%
	Edge	0.5595	0.5482	2.0584%

Table 1. Loss ratios of filters regarding learning rates from 1 down to 0.1.

By summarizing table 1, average loss ratios are listed in table 2.

	MAE	MAE0	Loss
Blur	0.1514	0.0727	109.8372%
Sharpening	0.2100	0.1758	19.4525%
Edge	0.5570	0.5482	1.5904%

Table 2. Average loss ratios of filters

From table 3, it is easy to recognize that sharpening filter and edge detection filter obtain good results with small loss ratios (19.4525% and 1.5904%) where edge detection is the best one (1.5904%), which implies that the proposed reverse method is suitable to the convolutional filters that focus on discovering pixel differences inside image. However, this improvement is insignificant because sharpening filters only keep most important features, which increase information loss. For instance, given whereas the perfect edge detection is {{–1, –1, –1}, {–1, 8, –1}, {–1, –1, –1}}, the best filter estimations of edge detection whose average loss ratio is 1.5904% with learning rate $\gamma = 0.9$ for color channels such as red, green, and blue are:

Red	–2.9976	–3.6497	–2.4164
	–2.6011	22.1964	–2.292
	–2.974	–2.6711	–3.2228
Green	–2.7992	–3.236	–2.8844
	–3.4417	24.4267	–3.4824
	–2.9609	–2.4793	–2.6858
Blue	–4.8567	–5.7483	–4.8785
	–4.5762	33.0547	–4.888
	–4.3127	–5.1539	–4.0629

It is easy to recognize that the estimated filters relatively keep the proportions between weights, for instance, the ratio –8 is relatively approximated. However, the magnitude of estimated filters is three times approximately larger than the magnitude of the perfect edge detection is {{–1, –1, –1}, {–1, 8, –1}, {–1, –1, –1}}.

4. Conclusions

In general, the image deconvolution in this research is simple with only two convolutional layers associated with backpropagation algorithm and stochastic gradient descent algorithm in reverse direction. Therefore, the restoration result is not as good as single image super-resolution (SISR) and deep neural network in autoencoder (AE), but the main contribution of this research is to inspect convolutional filters from together with mutual relationship between convolution and deconvolution. In the future trend, I will try to extend and improve the deconvolution process with deep neural network having more than two layers like AE did.

References

Cao, F., Yao, K., & Liang, J. (2020, September 23). Deconvolutional neural network for image super-resolution. *Neural Networks, 132*, 394-404. doi:10.1016/j.neunet.2020.09.017

Kim, K., & Kwon, Y. (2010, January 22). Single-Image Super-resolution Using Sparse Regression and Natural Image Prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*(6), 1127-1133. doi:10.1109/TPAMI.2010.25

Turchenko, V., Chalmers, E., & Luczak, A. (2017, January 18). A Deep Convolutional Auto-Encoder with Pooling - Unpooling Layers in Caffe. *arXiv preprint*. doi:10.48550/arXiv.1701.04949

Xu, L., Ren, J. S., Liu, C., & Jia, J. (2014). Deep Convolutional Neural Network for Image Deconvolution. *Deep Convolutional Neural Network for Image (NIPS 2014). 27.* NeurIPS. Retrieved from

https://proceedings.neurips.cc/paper_files/paper/2014/hash/1c1d4df596d01da60385f0bb17a4a9e0-Abstract.html