Convolutional Neural Networks and Autoencoders for Image Deblurring and Denoising

Rohan Agarwal, Aggelos Katsaggelos

2015-02-17

1 Introduction

Applications of image processing such as deblurring, denoising, and super-resolution can be modeled as a convolution on an image. Subsequently blurry and noisy images can undergo the process of *deconvolution* in order to restore them close to the clean form.

Previous research has shown that the deblurring process typically creates artifacts during the deconvolution process. These artifacts in turn are characteristic based on the blurring process, depending on the process used. There have been several publications about addressing specific types of artifacts but the challenge that persists is to create a process robust to different types of noise.

One of latest, most formidable attempts is the Outlier-rejection Convolutional Neural Network (ODCNN) which is a large network tasked with performing deblurring as well as denoising. Another network also taken consideration is the Stacked Sparse Denoising Autoencoder (SSDA). Both of these deep neural networks depart from the approach of using a generative model and instead treat images as data. This however also implies a heavy reliance on empiricism. With inspiration from the ODCNN and SSDA we aim to combine elements of both models to quantifiably improve the similarity between deconvolved images and their clean counterparts.

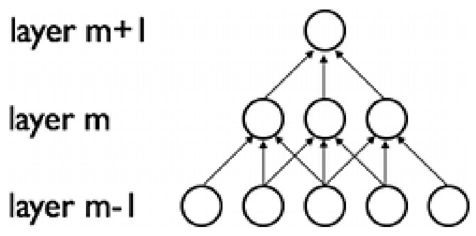
2 Deep Learning Algorithms

Approaches to deblurring and denoising, and even super-resolution using deep learning frequently involve two types of architectures: the convolutional neural network and the autoencoder. A CNN is a neural network with several layers and additional features which is typically connected and trained as a classifier. An autoencoder is a structure with a fully connected hidden layer which transforms a blurry image to what should be a clean image as its output, trained against the original clean image.

2.1 Convolutional Neural Network (CNN)

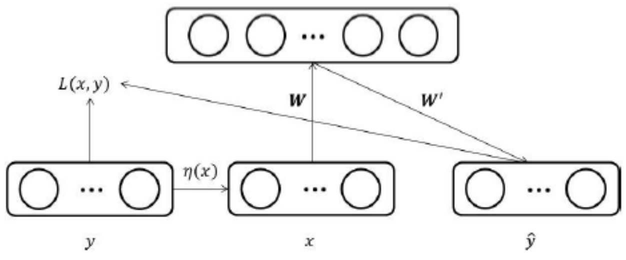
Convolutional neural networks are variants of standard multi-layer perceptrons (MLP), which take advantage of local connectivity to speed up the training process with a large number of neurons. Local connectivity refers to the fact that a hidden layer neuron may only be to a few “nearby” neurons in the previous layer.

In addition to local connectivity, layers can also contain shared weights where the weights entering or exiting a neuron could be the same for multiple neurons preceding it. This allows for training the weights in batches. Additional features in CNN architectures include pooling of individual neurons as well as the rectified linear unit (ReLU). CNN architectures are typically at the end are connected to a fully connected layer which in turn is connected a classifier. A common example of such an architecture is the LeNet network or subsequent ones such as AlexNet and GoogLeNet.



2.2 Autoencoder

An autoencoder is an architecture which contains a single hidden layer h which transforms an input x into an output z which is the same size and dimensions of x. The premise however is that x is formed as the result of convolution with some other input y. Analagous to our situation, y would be an original clean image and x would be a convolved, blurry form of the same image. The goal would be to create a reconstruction z by passing x through the layer h. Subsequently the values of the layer h would be trained based on the difference between z and the original clean image y - either through least square error or cross entropy. This approach allows h to become a hidden representation of the deconvolution operator from x to y. As the quality of the operator increases, the difference between the reconstructed z and the original y decreases.

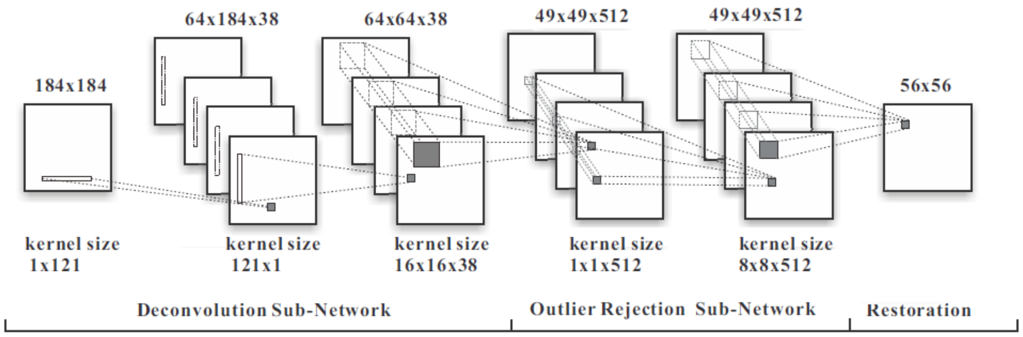


It isn’t necessary that only one hidden layer h is used - it is possible to create a stacked autoencoder which contains multiple hidden layers which are pretrained individually and then cumulatively trained as well using packpropagation.

3 Related Work

In their work *Image Denoising and Inpainting with Deep Neural Networks* by Xie et al., they importantly find that sparse denoising autoencoders are able to capture characteristic types of noise when they write “Our experiments on the image denoising and inpainting tasks demonstrate that SDA is able to learn features that adapt to specific noises from white Gaussian noise to superimposed text.” This is further quantified by training images on specific types of noise (Gaussian vs salt and pepper vs overlayed text) and then testing images on the same as well as different types of noise with a noticeably different PSNR.

Subsequently we see that in *Deep Convolutional Neural Network for Image Deconvolution* by Liu et al., a convolutional neural network is used to replicate the singular value decompositon of the inverse kernel. This yields a network of 4 layers - the first two represent the deblurring process and the latter two represent denoising. The network however has the cost function of an autoencoder with the error being least squares between the original clean image and the reconstruction. Based on this the ODCNN can be seen as a combination of the canonical definitions of convolutional neural networks and stacked autoencoders.



4 Motivation

We observe that the SDA empirically handles specific types of noise very well; it can be trained with different types of noise to becomed “accustomed” to it. However the ODCNN continues to use convolutional layers rather than fully connected ones in the SDA in order to perform denoising. There is also no reported pretraining of individual convolutional layers while there typically is with an autoencoder. Note another unreported aspect of ODCNN: having a cost function between the original image and a reconstructed one requires both images to be the same size. This subsequently implies that padding around images is required when passing kernels.

An idea is therefore to replace the second half of the ODCNN with the SDA architecture in order to improve the ability of the network to capture noise that results from the first deblurring part of the network. Another advantage is that the deblurring process generally tends to produce a characteristic noise pattern of concentric circles. We therefore hypothesize that a series of fully connected, pretrained autoencoder layers would perform better than the convolutional layers previously built.

5 Implementation

Combining the ODCNN and SDA architecture is non-trivial for several reasons: firstly, the source code for SDA is unavailable. Subsequently the ODCNN code available is written from scratch without use of any packages or frameworks. This makes substituting any parts out a challenging affair. A goal is therefore to reproduce both individual networks using a modular structure first, and then combine them in order to measure performance improvements. The tool of choice for this task is the Theano library in Python, chosen due to widespread support as well as comfort with Python.

The task list at hand is therefore to do the following:

1. Implement the SDA structure
2. Create a padding function in Theano for images
3. Create a least squares cost function for a convolutional neural network in Theano
4. Implement ODCNN
5. Substitute the last two layers of ODCNN with the previously created SDA and benchmark

6 Current Progress

1. ~~Implement the SDA structure~~
2. Create a padding function in Theano for images
3. ~~Create a least squares cost function for a convolutional neural network in Theano~~
4. Implement ODCNN
5. Substitute the last two layers of ODCNN with the previously created SDA and benchmark