Image Denoising and Inpainting with Deep Neural Networks

Abhinav Anand
2018201037
IIIT, Hyderabad
abhinav.anand@students.iiit.ac.in

Lokesh Singh Mahar
2018201049
IIIT, Hyderabad
lokesh.singh@students.iiit.ac.in

Neeraj Barthwal
2018201069
IIIT, Hyderabad
neeraj.barthwal@students.iiit.ac.in

Pranjal Patidar
2018201094
IIIT, Hyderabad
pranjal.patidar@students.iiit.ac.in

Shubham Rawat
2018201098
IIIT, Hyderabad
shubham.rawat@students.iiit.ac.in

***Abstract*—In this project we deal with image denoising and inpainting which are common image restoration problems. We utilise the deep neural networks which are pre-trained with denoising auto-encoder(DA). This method adapts the auto-encoder(DA) which is originally designed for unsupervised feature learning as presented by Junyuan et al. [1]. Rather than removal of simple patterns like missing pixels, the complex patterns such as a superimposed text is removed automatically in this project. In our implementation there is no need to specify the region from where the superimposed text is to be removed. Finally it is shown that this method is more effective and performance of an unsupervised feature learning can be improved.**

I. Introduction

An image may get corrupted by noise, a superimposed text, artificial editing, noise, camera motion blurring and pixel value errors etc. Our goal is to restore the original image from a corrupted observation. Image restoration is a preprocessing step in many applications as stated by Ishfaq Bashir et al.[2]

Few applications of image restoration are :-

In the area of astronomical applications characterized by poisson noise, Gaussian noise; Image restoration has played a very important role for better study in astronomy.

It is also useful in medical imaging such as computerised tomography(CT) and magnetic resonance imaging (MRI). Since the resolution quality is limited, the acquisition of multiple images is possible.Therefore restoring the image to high quality is pertinent as this can help the surgeon to operate more successfully over the exact part of the body with care.

Over the multispectral bands of satellite imagery, multispectral image restoration can be carried out in order to improve the resolution of the captured satellite images.

Image inpainting can be divided into two types, non-blind and blind inpainting. In non-blind inpainting the affected region has to be provided to the algorithm a priori whereas in blind inpainting method, there is no need to provide the locations of the corrupted pixels or the superimposed text.

We provide solution to some simple datasets like the MNIST digit dataset and the MNIST fashion dataset. The training and testing images were imposed with gaussian and salt pepper noise. Experimental results demonstrate the effectiveness of the proposed method in the tasks of image denoising and blind inpainting.

II. Theory (Model description)

The deep models performs well with real-life problems like denoising and inpainting. In deep neural network as no. of hidden layer increases, the training becomes harder. So we used greedy layer-wise pre-training to initialize the weights for Deep Neural Network. For the layer-wise pre-training Denoising Autoencoders (DA) are used. DA is a two-layer neural network that takes inputs as noisy images. We stacked two such DAs to form deep neural network which is called Stacked Denoising Auto-encoders (SDA).

Denoising Auto-encoder - An autoencoder is a neural network used for dimensionality reduction; that is, for feature selection and extraction. Autoencoders with more hidden layers than inputs run the risk of learning the identity function – where the output simply equals the input – thereby becoming useless. Denoising autoencoders are an extension of the basic autoencoder and represent a stochastic version of it. Denoising autoencoders attempt to address identity-function risk by randomly corrupting input (i.e. introducing noise) that the autoencoder must then reconstruct or denoise.[5]

Stacked Denoising Autoencoder - A stacked denoising autoencoder is simply many denoising autoencoders strung together. A key function of SDAs, and deep learning more generally, is unsupervised pre-training, layer by layer, as input is fed through. Once each layer is pre-trained to conduct feature selection and extraction on the input from the preceding layer, the second stage of supervised fine-tuning can follow (i.e deep neural network).

Let us assume y is the original noise-free image and x is the noisy version of y. To generate x.

$$\mathbf{x} = \eta(\mathbf{y}) \tag{1}$$

where $\eta : \mathbb{R}^n \to \mathbb{R}^n$ is an arbitrary stochastic corrupting process that corrupts the input. Then, the denoising task's learning objective becomes:

$$f = \underset{f}{\operatorname{argmin}} \mathbf{E}_{\mathbf{y}} \| f(\mathbf{x}) - \mathbf{y} \|_2^2 \tag{2}$$

Now considering a Denoising Autoencoder. We have two layers.

$$\mathbf{h}\left(\mathbf{x_i}\right) = \sigma\left(\mathbf{W}\mathbf{x_i} + \mathbf{b}\right) \tag{3}$$

$$\hat{\mathbf{y}}\left(\mathbf{x_i}\right) = \sigma\left(\mathbf{W}'\mathbf{h}\left(\mathbf{x_i}\right) + \mathbf{b}'\right) \tag{4}$$

Where $\sigma(x) = (1 + \exp(-x))^{-1}$ is the sigmoid activation function and $\mathbf{h_i}$ hidden layer activation, $\hat{\mathbf{y}}\left(\mathbf{x_i}\right)$ is an approximation of $\mathbf{y_i}$ and $\Theta = \{\mathbf{W}, \mathbf{b}, \mathbf{W}', \mathbf{b}'\}$ are weights and biases.

For reconstruction loss :

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left\| \mathbf{y_i} - \hat{\mathbf{y}}\left(\mathbf{x_i}\right) \right\| \tag{5}$$

Now dealing with large images is tough, instead of directly processing the entire image we will draw overlapping patches from that images. The input to DAs will be the patches, not the entire image. y_i will be the x_ith noisy patch and x_i will be the i_{th} original patch of y_i.

To combine the virtues of sparse coding and neural networks and avoid over-fitting, we train a DA to minimize the reconstruction loss regularized by a sparsity-inducing term:

$$L_1(\mathbf{X}, \mathbf{Y}; \theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left\| \mathbf{y}_i - \hat{\mathbf{y}}\left(\mathbf{x}_i\right) \right\|_2^2 + \beta \mathbf{KL}(\hat{\rho} \| \rho) + \frac{\lambda}{2} \left(\|\mathbf{W}\|_F^2 + \|\mathbf{W}'\|_F^2 \right) \tag{6}$$

L2 regularizer is used to avoid overfitting and KL divergence is used for achieving sparsity. Where

$$\mathrm{KL}(\hat{\rho} \| \rho) = \sum_{j=1}^{|\hat{\rho}|} \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{(1 - \rho)}{1 - \hat{\rho}_j}$$

,

$$\hat{\rho} = \frac{1}{N} \sum_{i}^{N} \mathbf{h}\left(\mathbf{x_i}\right)$$

Then training the second DA with hidden layer activation of first DA as the input for second DA. This way stacking of DAs is done. From this both DAs we will get $W1, W1', W2, W2', b1, b1', b2, b2'$ which will be used in deep neural network (DNN) as pre-trained parameters. If we stacked k DAs then we will have 2k weights and bias which will be used in DNN with $2k - 1$ hidden layers.

Now this entire network will be trained using the standard back-propagation algorithm

$$L_2(\mathbf{X}, \mathbf{Y}; \theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left\| \mathbf{y_i} - \mathbf{y}\left(\mathbf{x_i}\right) \right\|_2^2 + \frac{\lambda}{2} \sum_{j=1}^{2K} \left(\|\mathbf{W_j}\|_F^2 \right) \tag{7}$$

Adam Optimizer - We have used Adam Optimizer for Minimising our Loss. Adam is a popular algorithm in the field of deep learning because it achieves good results fast. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. Here is the algorithm stated:

ALGORITHM: Adam, our proposed algorithm for stochastic optimization. g_t^2 indicates the element wise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α : step size
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0 : Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1^{st} moment vector)
$v_0 \leftarrow 0$ (Initialize 2^{nd} moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**

- $t \leftarrow t + 1$
- $g_t \leftarrow \nabla_\theta f_t\left(\theta_{t-1}\right)$ (Get β_1) $\cdot\, g_t$ (Update biased first moment estimate)
- $m_t \leftarrow \beta_1 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased first moment essimate)
- $\hat{m}_t \leftarrow m_t / \left(1 - \beta_1^t\right)$ (Compute bias-corrected first moment estimate)
- $\hat{v}_t \leftarrow v_t / \left(1 - \beta_2^f\right)$ (Compute bias-corrected second raw moment estimate)
- $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$(Update parameters)

end while
return θ_t

IMPLEMENTATION

Our focus is mainly on denoising and inpainting of grey-scale images. We use a set of images collected from the MNIST digit data and MNIST fashion data as our training set and standard testing set. We create noisy images using from clean training and testing images by applying Gaussian Noise to them. Image patches are then extracted from both clean and noisy images to train SSDAs. We employ Peak Signal to Noise Ratio (PSNR) to quantify denoising results: $10log10(2552/\sigma 2e)$, where $sigma2e$ is the mean squared error. PSNR is one of the standard indicators used for evaluating image denoising results.

A. Denoising White Gaussian Noise

Add Noise: We first corrupt images of both testing and training set with additive white Gaussian noise of various standard deviations.

Train data: For the proposed method, one SSDA model is trained for each noise level. We evaluate different hyper-parameter combinations and report the best result. We set K to 2 for all cases because adding more layers may slightly improve the performance but require much more training time. For MNIST data, we did not use patches in our images as images are of low dimensions. This is true for both MNIST digit and MNIST fashion data set. For high dimension images, we try different patch sizes and find that higher noise level generally requires larger patch size.The dimension of hidden layers is generally set to be a constant factor times the dimension of the input.

Test data: For testing the data we generate a loss function and minimize the loss using ADAM Optimizer. For MNIST data set there is no use to depatchify the patches as we have used single image in training. But for large images we had depatchify the image patches and produce outputs. For large dimension images the optimizer fall into a memory error.

B. Denoising White Gaussian Noise

For the image inpainting task, we test our model on the text removal problem.

Add Noise: Both the training and testing set are composed of images with superimposed text of various fonts and sizes from 18-pix to 36-pix.

Train data: For the proposed method, one SSDA model is trained for each noise level. We evaluate different hyper-parameter combinations and report the best result. We set K to 2 for all cases because adding more layers may slightly improve the performance but require much more training time. For MNIST data, we did not use patches in our images as images are of low dimensions. This is true for both MNIST digit and MNIST fashion data set. For high dimension images, we try different patch sizes and find that higher noise level generally requires larger patch size.The dimension of hidden layers is generally set to be a constant factor times the dimension of the input.

Test data: For testing the data we generate a loss function and minimize the loss using ADAM Optimizer. For MNIST data set there is no use to depatchify the patches as we have used single image in training. But for large images we had depatchify the image patches and produce outputs. For large dimension images the optimizer fall into a memory error.

III. Problems faced

For larger images training time will be more so what we did was down-sampled the original images. And directly processing entire image is intractable, so we instead draw overlapping patches of down-sampled images.

IV. Result analysis

On fashion MNIST data with gaussian noise the loss we obtained is 0.0037 and PSNR value is 56.1488 (fig 2).

To check the original result images visit
https://github.com/abnvanand/MTP2019-ImageDenoisingInpaintinggithub

In each section the first row is the original image and second row shows the noisy (or inpainted) version of the original image and the last row shows the result obtained after denoising (or inpainting).

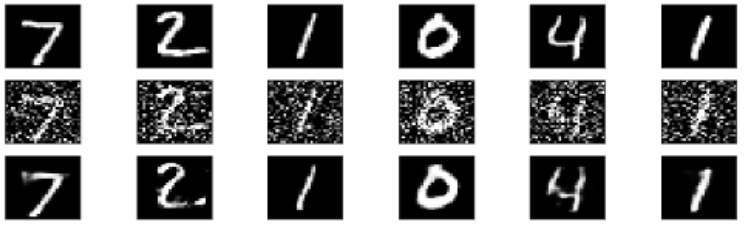

Fig. 1. Denoising(gaussian noise) task on MNIST digit data set.

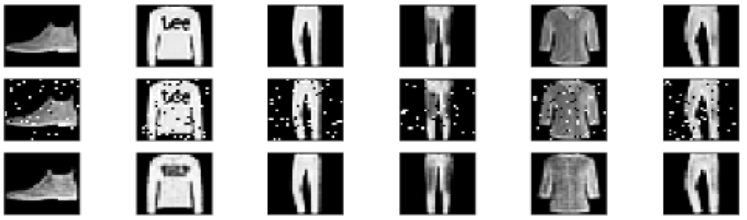

Fig. 2. Denoising(gaussian noise) task on MNIST fashion data set.

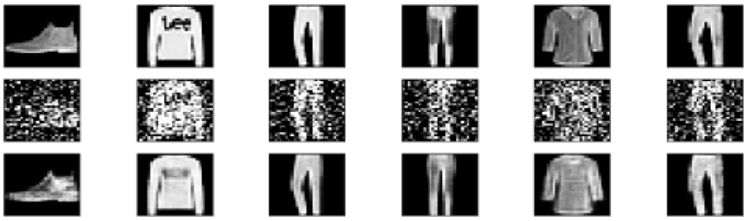

Fig. 3. Denoising(salt-pepper noise) task on MNIST fashion data set.

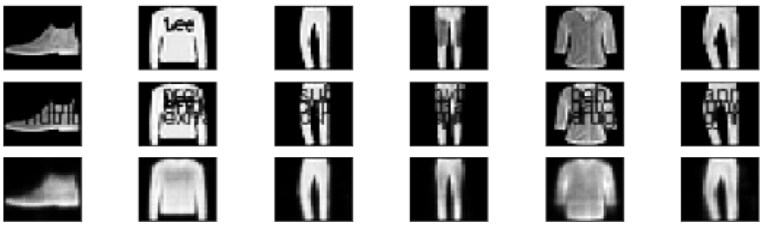

Fig. 4. Inpainting task on MNIST fashion data set.

References

[1] Junyuan Xie, Linli Xu, Enhong Chen "Image Denoising and Inpainting with Deep Neural Networks" . http://staff.ustc.edu.cn/ linlixu/papers/nips12.pdf

[2] MNIST fashion dataset:-http://yann.lecun.com/exdb/mnist/

[3] MNIST digit dataset:-http://yann.lecun.com/exdb/mnist/

[4] Ishfaq Bashir, Adil Majeed, Owais Khursheed Image restoration and the various restoration techniques used in the field of digital image processing.

[5] Diederik P. Kingma, Jimmy Lei Ba "ADAM: A method for Stochastic Optimization"

[6] Keras Implementation :-https://keras.io