Image denoising using convolutional autoencoders with skip connections

193079014,183079009,183079007

December 2020

1 Introduction

X-Rays, MRI, CT, ultrasound are susceptible to noise and its important to denoise these images before doing any further analysis. Deep learning based image denoising methods have outperformed all conventional denoising method. These networks are less restrictive to specification of noise generative process. Here we used small sample size and used convolutional denoising autoencoder with concatenation of feature maps from encoder to decoder side (similar to UNET). We used heterogeneous medical data for increasing the sample size. We also confirmed that, we can recover signal even when the noise levels are very high, at the point when other methods (conventional methods) will fail.

2 Preliminaries

2.1 Autoencoder

Autoencoder is a type of neural network that tries to learn identity function.It first takes input and maps (encode) it into a hidden representation using an encoding Neural network.This latent representation is mapped back(decoded) into a representation of input. Usually hidden representations are much smaller in dimensions than the input dimension, this is done so that Autoencoder is forced to learn a compressed approximation of the input.In our case, we force the model to learn the reconstruction of the input given its noisy version as the input.Autoencoder used has an architecture with convolutional encoding and decoding layers.Here we used Conv layers, Leaky ReLU, Dropout, BatchNormalization, Concatenation and Transpose convolution.Long skip connections are used to pass features from the encoder path to the decoder path in order to recover spatial information lost during downsampling. Skip connections enable feature reusability and stabilize training and convergence.

2.2 Data

Here we used three different datasets. They include 116 dental X-Ray images,

594 chest X-Ray images and 320 head X-Ray images. We mixed chest X-ray and dental X-ray images into a single dataset containing 710 images . These 710 images were split at 0.8:0.2 as training:validation. Noisy version of the training data was given as the input to my model and the corresponding noiseless ones were taken as ground truth. We used head X-ray to test how well the model worked (data from some other distribution).

2.3 Pre-processing

Images were resized into $256X256$.Images were corrupted with varying levels of noise before training. We used gaussian noise(mean=0, std=0.1),gaussian noise(mean=0, std=0.4), salt and pepper noise (lowerthres=190) one at a time on both training and validation set prior training. Images were rescaled so that each pixels will have a real value between 0 and 1.

2.4 Loss function and evaluation metric

Since we were dealing with real values at the output side, we used MSE between pixels of ground truth and the prediction as our loss function. And, instead of accuracy or PSNR ,we used SSIM(structural similarity index measure) as the evaluation metric. SSIM estimates visual effects of shift in image luminance ,contrast and other errors. These things are called structural changes. If x is the ground truth and y is the prediction them, SSIM is given as

$$SSIM(x,y) = [l(x,y)^{\alpha} c(x,y)^{\beta} s(x,y)^{\gamma}] \tag{1}$$

α,β and γ are greater than 0, we took them as unity. $\mathbf{l}$,c and s are luminance ,contrast and structural component and are calculated as follows.

$$l(x,y) = \frac{2u_x u_y + C_1}{u_x^2 + u_y^2 + C_1} \tag{2}$$

$$c(x,y) = \frac{2\sigma_x \sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{3}$$

$$s(x,y) = \frac{2\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3} \tag{4}$$

where u_x and u_y are the mean of the ground truth and the prediction, σ_x and σ_y are the standard deviation of the ground truth and the prediction and σ_{xy} is the covariance of two datas.

2.5 Training

We used batch size of 32, with 50 epochs for training. Adam optimizer was used with learning rate of 0.002. We used early stopping on minimum validation loss with patience of 10. We used batch Normalization to reduce our training time and dropout to prevent overfitting.

2.6 Model trained with data corrupted with salt and pepper noise with lower threshold=190 (Extreme salt and pepper case)

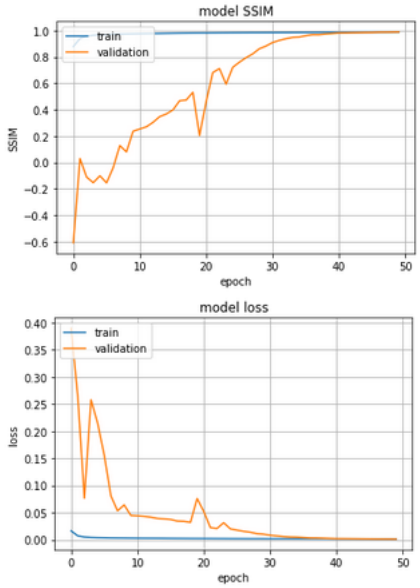

Figure 1.a Training curve - SSIM vs epoch (top row) and loss vs epoch (bottom row)

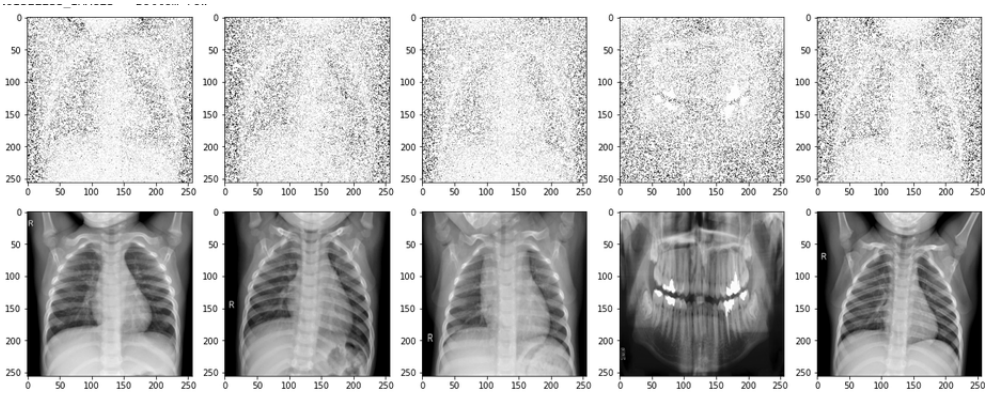

Figure 1.b samples of ground truth (bottom row) and their noisy counterpart (top row)

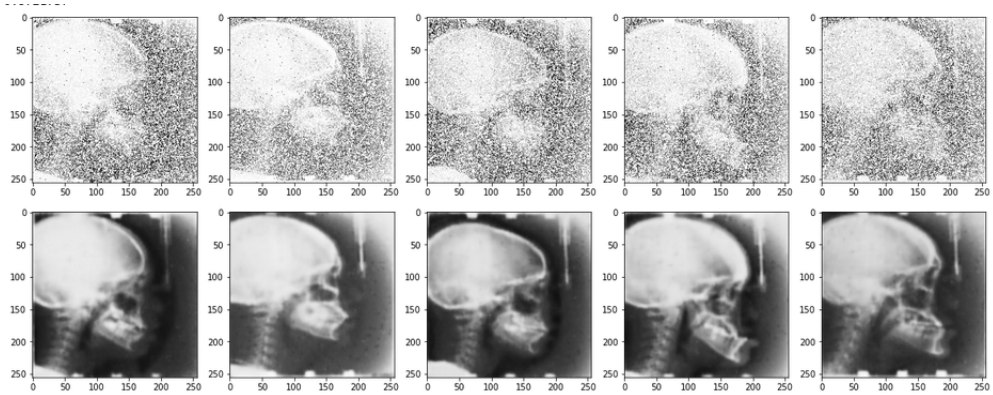

Figure 1.c Test image from some other distribution (corrupted with salt and pepper noise with threshold=150), SSIM=0.9734 (Top row) and filtered output (Bottom row)

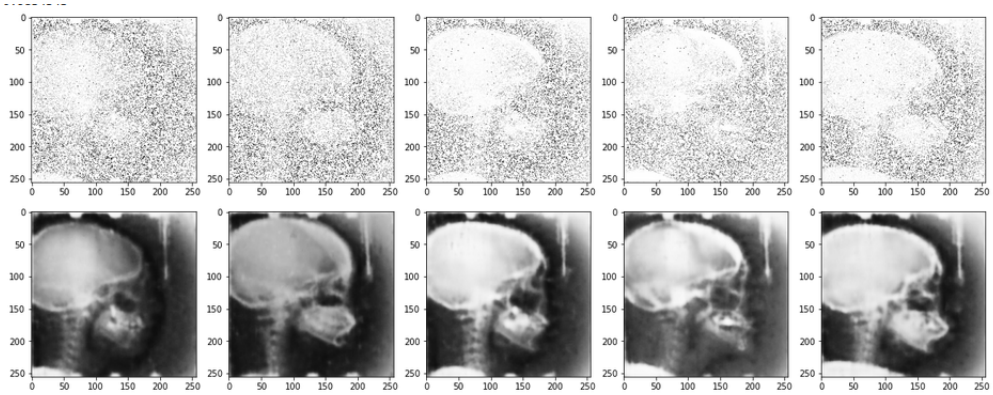

Figure 1.d Test image from some other distribution (corrupted with salt and pepper noise with threshold=200), SSIM=0.90752 (Top row) and filtered output (Bottom row)

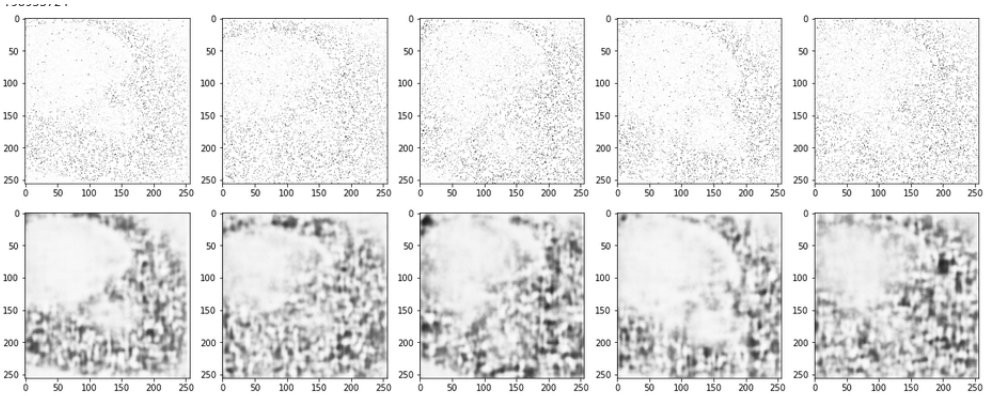

Figure 1.e Test image from some other distribution (corrupted with salt and pepper noise with threshold=235), SSIM=0.4052 (Top row) and filtered output (Bottom row)

It's clearly seen that with varying noise level at input, our network is in position to denoise the input image perfectly. It is also observed that, in cases where human fails to understand what is in the given noisy image (Fig 1.d), our network identifies and denoise it correctly. With extreme noise conditions (ie, Fig 1.e) our network fails in doing its job, we get a small SSIM

2.7 Model trained with data corrupted with gaussian noise with mean=0 and std=0.1

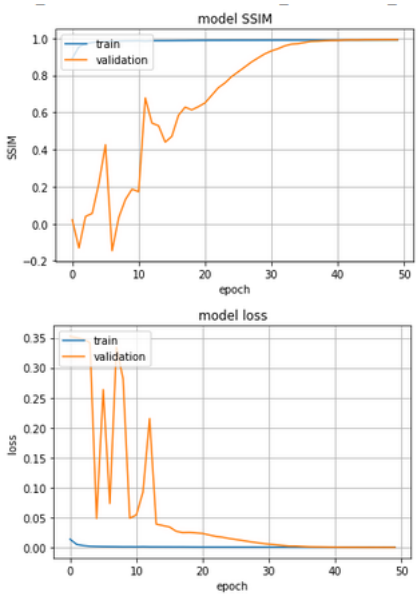

Figure 2.a Training curve - SSIM vs epoch (top row) and loss vs epoch (bottom row)

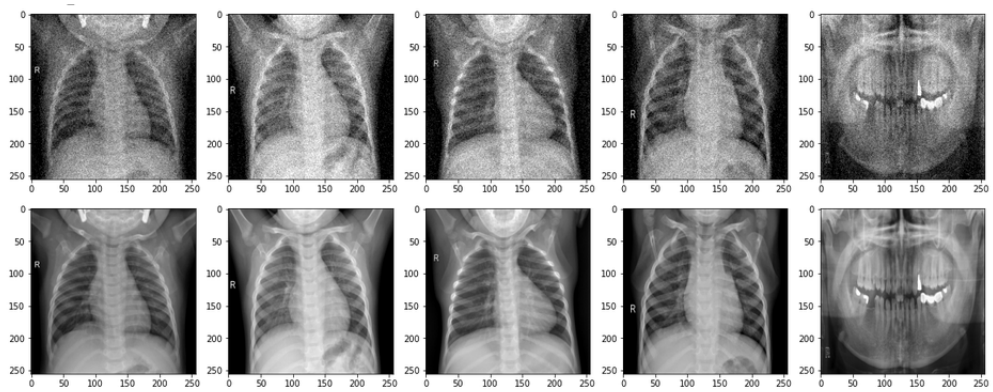

Figure 2.b samples of ground truth (bottom row) and their noisy counterpart (top row)

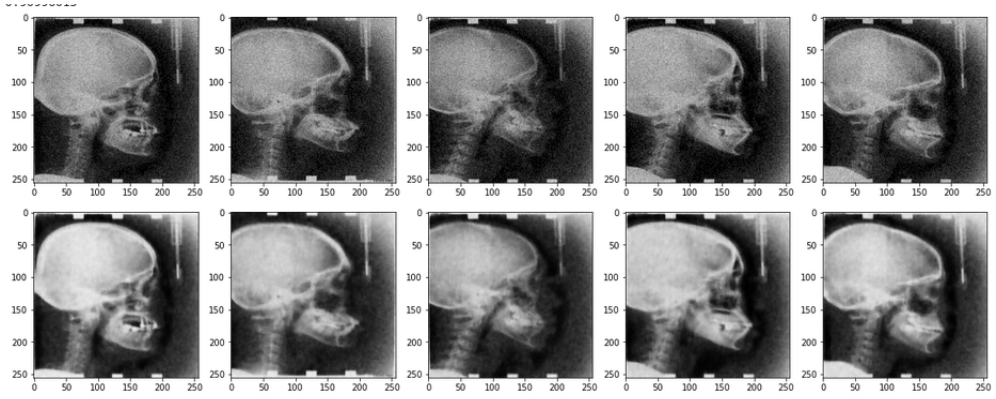

Figure 1.c Test image from some other distribution (corrupted with gaussian noise with mean=0 and std=0.1), SSIM=0.991 (Top row) and filtered output (Bottom row)

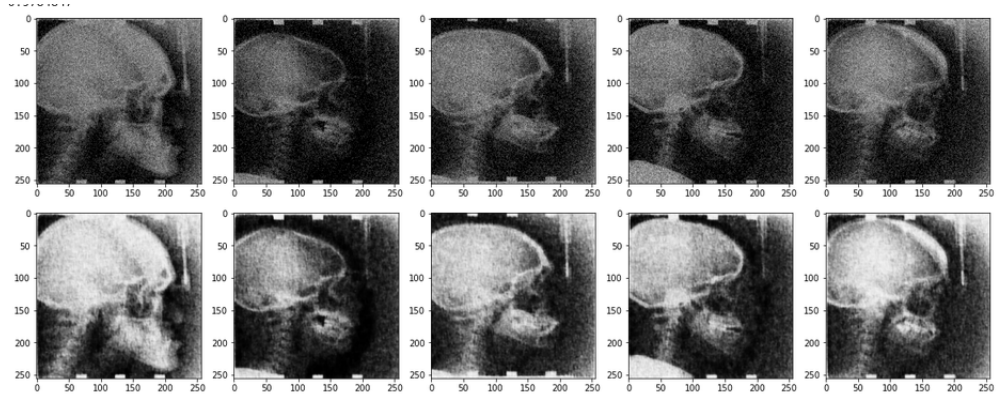

Figure 2.d Test image from some other distribution (corrupted with gaussian noise with mean=0 and std=0.2), SSIM=0.9795 (Top row) and filtered output (Bottom row)

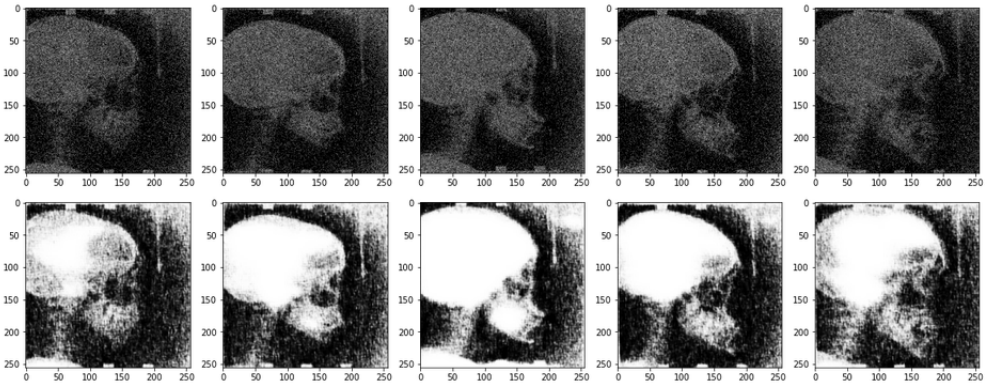

Figure 2.e Test image from some other distribution (corrupted with gaussian noise with mean=0 and std=0.0.4), SSIM=0.9109 (Top row) and filtered output (Bottom row)

Its observed that denoising performance (given by SSIM) is high for std=0.1 and decreases as we keep on increasing std of gaussian noise in test data. For std=0.4, the denoised image is not perfect and many fine details are lost, overall the performance is better than the conventional filter.

2.8 Model trained with data corrupted with gaussian noise with mean=0 and std=0.4

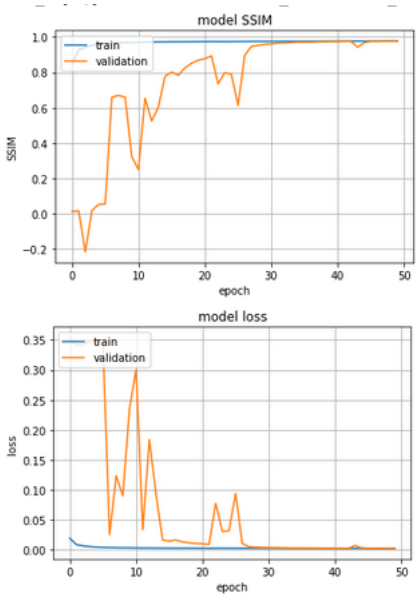

Figure 3.a Training curve - SSIM vs epoch (top row) and loss vs epoch (bottom row)

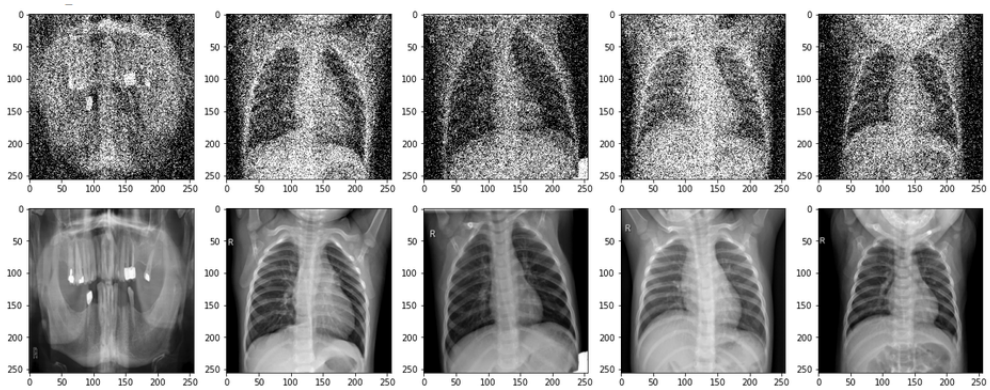

Figure 3.b samples of ground truth (bottom row) and their noisy counterpart (top row)

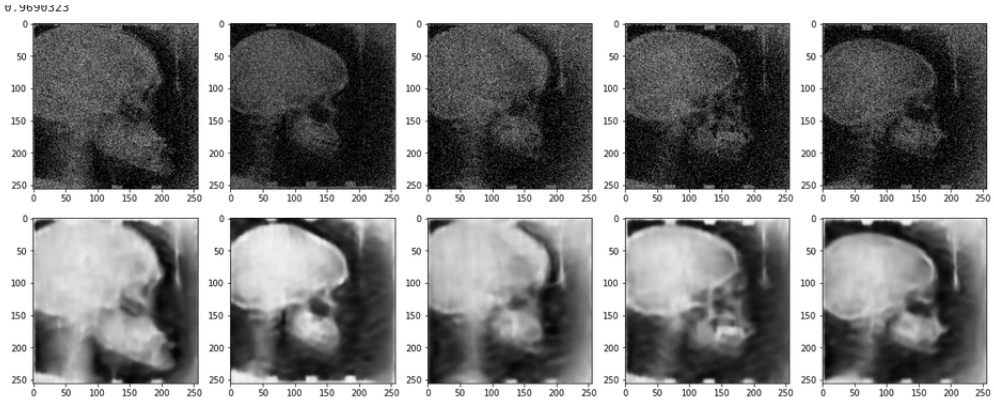

Figure 3.c Test image from some other distribution (corrupted with gaussian noise with mean=0 and std=0.2), SSIM=0.9708 (Top row) and filtered output (Bottom row)

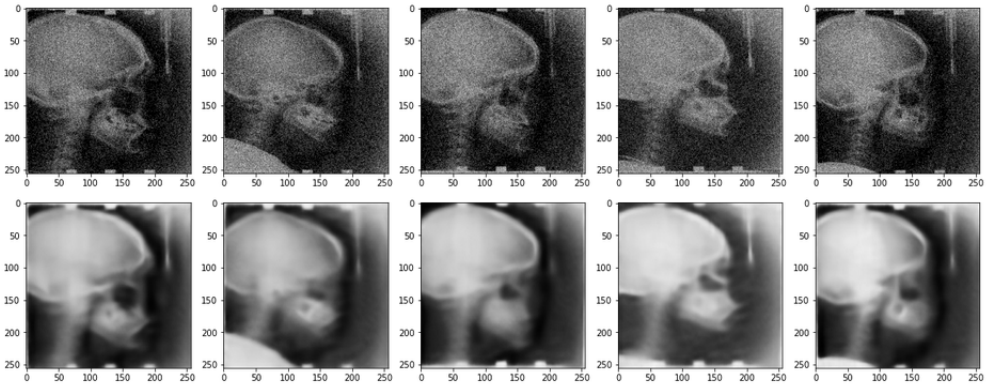

Figure 3.d Test image from some other distribution (corrupted with gaussian noise with mean=0 and std=0.4), SSIM=0.9604 (Top row) and filtered output (Bottom row)

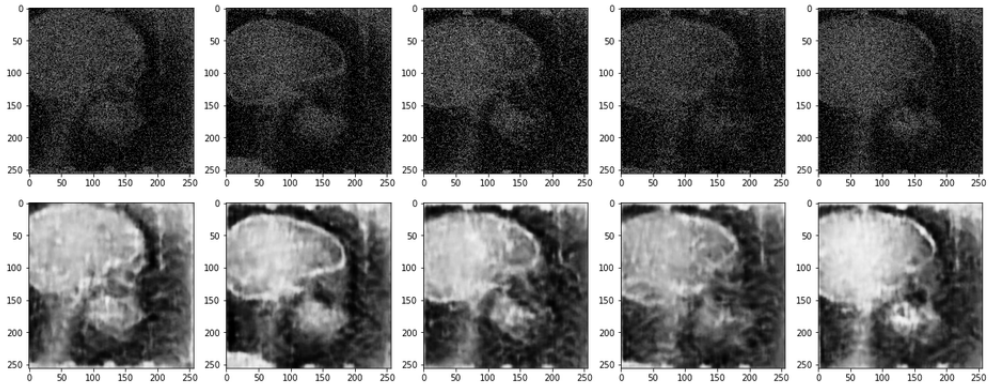

Figure 3.e Test image from some other distribution (corrupted with gaussian noise with mean=0 and std=0.6), SSIM=0.9563 (Top row) and filtered output (Bottom row)

As we increase the std of noise added to the test image to a very large value ,say above 0.6 the denoising performance degrades, even though the model is capable of understanding the objects in the image, the details are lost and denoising is somewhat partial. For std like 0.4, 0.2 the performance is still better. Its observed that while training we don't want training input to be too noisy as in case 3, std =0.1 will do a good job

3 Reference

Lovedeep Gondara. Medical image denoising using convolutional autoencoder