Medical Image Analysis KU, WS24
Assignment 2
Denoising

Gregor Gruber
0730864

1 Task Description:

The two task of this assignment are the following:

- Task A: Denoise a CT image slice (see Figure 1) by implementing Tikhonov Denoising and Edge Preserving Denoising (Perona Malik approach)

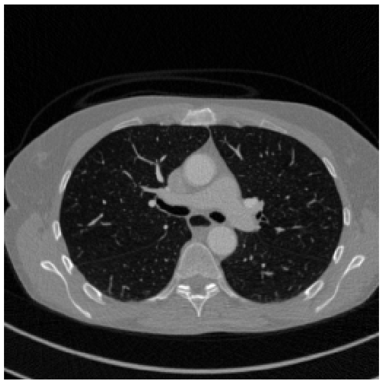


Figure 1: CT thorax slice

- Task B: : Train a deep learning denoising autoencoder on MedMNIST chest X-ray images

2 Task A: To denoise a CT image slice

The first part of Task A was to implement the Tikhonov Denoising on the provided picture of a CT image slice with added noise (normal, sigma 20) as shown in figure 2. To find the optimal solution steepest descent method was chosen, which tried to computing for the best solution by changing λ (between 0.01 and 1). The formula for the steepest decent is the following:

$$u^{t+1} = u^t - \tau(-\Delta + \lambda(u^t - f))$$

The learning rate was provided by $\tau = 0.01$ and a maximum of 5000 iterations was given in the instructions. Further more it was required to implement a check of convergence while computing the 5000 iterations. Here the relative difference $\frac{||u^{t+1}-u^t||}{||u^t||}$ should not exceede $\epsilon = 1e^{-4}$.

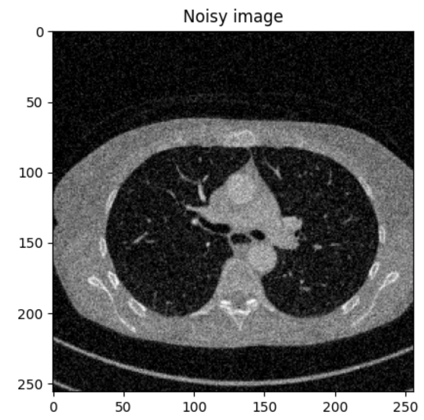
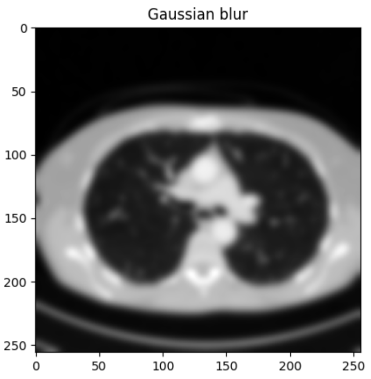
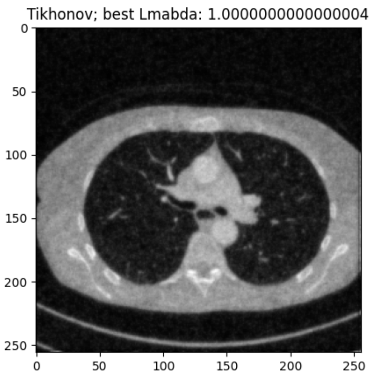


Figure 2: CT thorax slice with noise

The measure for finding the best output of the denoised image was comparing it to the input image (Figure 1) by the Root Mean Squared Error (RMSE). In 3b you can see the output image of the Tikhonov Denoising at $\lambda = 1$ which has a RMSE of **10.871927287965907**. This is already better as when applying a gaussian filter as shown in figure 3a (RMSE = 17.580176159385402).



(a) CT thorax slice after Gaussian blur



(b) CT thorax slice after Tikhonov Denoising

Figure 3: Comparison Gaussian and Tikhonov Denoising

The second part was to implement an Edge-Preserving Denoising, inspired by Perona-Malik. For this we needed to introduce a diffusion tensor D using the g-function. Further more derive the modified Euler-Lagrange equation for Perona-Malik and then again implement the deepest decent like for the Tikhonov Denoising.

To come to the modified Euler-Lagrange equation we need to put the g-function

$$g(||\nabla f||_2) = \frac{1}{1 + (\frac{||\nabla f||_2}{K})^2}$$

into the energy functional as follows:

$$E[u] = \int_{\Omega} \left[\frac{1}{2} \nabla u^T D \nabla u + \frac{\lambda}{2}(u - f)^2 \right] d\vec{x},$$

now we can compute all derivatives of the Euler-Lagrange equation

$$\frac{\delta E}{\delta u} = \frac{\partial E}{\partial u} - \frac{\partial}{\partial x}\frac{\partial E}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial E}{\partial u_y} = 0$$

where $\frac{\partial E}{\partial u}$ is:

$$\frac{\partial}{\partial u}\left(\frac{\lambda}{2}(u - f)^2\right) = \lambda(u - f)$$

for the next term we get:

$$\frac{\partial}{\partial u_x}\left(\frac{1}{2} g(\|\nabla f\|_2) u_x^2\right) = g(\|\nabla f\|_2) u_x$$

and similar for the next term:

$$\frac{\partial}{\partial u_y}\left(\frac{1}{2} g(\|\nabla f\|_2) u_y^2\right) = g(\|\nabla f\|_2) u_y$$

now we can combine these terms again into:

$$-\frac{\partial}{\partial x}\left(g(\|\nabla f\|_2) u_x\right) - \frac{\partial}{\partial y}\left(g(\|\nabla f\|_2) u_y\right)$$

which becomes this term:

$$-\nabla \cdot (g(\|\nabla f\|_2)\nabla u)$$

and if we combine all terms we get the modified Euler-Lagrange equation:

$$\lambda(u - f) - \nabla \cdot (g(\|\nabla f\|_2)\nabla u) = 0$$

In the implementation the process was to loop through different K and different λ. In figure 4 you can see the best output with $\lambda = 0.36$ and K=10 which resulted in an RMSE=**9.407150965280463**

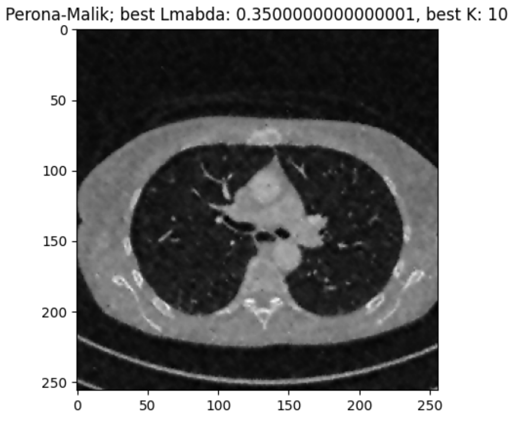


Figure 4: CT thorax slice with Perona-Malik Denoising

Optional Task: Another measure the find out the similarity between two images is the Structural Similarity Index (SSIM). The model is based on perception and includes luminance masking and contrast masking terms. Compared to RMSE, which is a absolute error which focus solely on pixel-by-pixel differences, it considers a strong dependence between pixels. The formula to calculate the SSIM is [1]:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

The main benefit of the SSIM is that it relates more closely to what the human eye/mind perceives in a picture and represents this in a value between 0 and 1 (1 being better). But this benefit also comes with shortcomings, the index is computationally more intense then the RMSE and can only be used for images. Whereas the RMSE is easy to compute and can be used for various usecases.

3 Task B: Train a pytorch deep learning denoising autoencoder

As a layout for the denoising autoencoder the following was chosen:

Encoder:

It uses a Sequential layout with 2 convolutional layers (nn.Conv2d) followed by pooling layers (nn.MaxPool2d) and a ReLU activation function (nn.ReLU).

- Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)): The first layer has a single input, the gray scale image; it has 16 channels produced by the convolution in the first layer and 32 in the second layer; a 3x3 kernel size (as given in the instructions); a stride of 1, meaning the filter moves by one pixel; and adds padding of 1 pixel of zeros around the input
- ReLU(): the ReLU activation function is used to also cover unliniarities (as given in the instructions)
- MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False): The pooling layers downsample the feature map from 28x28 to 14x14 and in the second step to 7x7

Decoder:

It uses a Sequential layout with 2 transpose convolutional layers (nn.ConvTranspose2d) followed by ReLU activation function (nn.ReLU) and a final convolutional layer with a sigmoid activation function.

- ConvTranspose2d(32, 16, kernel_size=(2, 2), stride=(2, 2)): The first layer takes the input from the last layer of the Encoder and starts with the deconstruction of the compressed representation by upsampling by the factor of 2.
- ReLU(): again a ReLU activation function is used.
- Conv2d(1, 1, kernel_size=(1, 1), stride=(1, 1)): the final layer is again a convolutional layer taking the input 1 and an output of 1 to refine the reconstruction.
- Sigmoid(): The final step is a sigmoid activation function to ensure that the output is between 0 and 1.

The model performance was tracked via the MSE loss over the epochs as shown in table 1.

Epoch	Training loss	Validation loss
1/30	0.07343716160409419	0.048124148050255555
5/30	0.004550849524757397	0.003715791562983347
10/30	0.0011650287672871245	0.0012178779884927376
15/30	0.000869487616759103	0.0009229189885091581
20/30	0.0007993132366111702	0.0008209354155501744
25/30	0.000765695559893847	0.000792091389850203
30/30	0.000745399376997081	0.0008123943577289773

Table 1: MSE train and validation loss over epochs

In figure 5 you can see the train and validation loss over the the different epochs. The Loss for the test set was MSE Test loss: **0.0007752167298873477**
And finally the average test SSIM: **0.9365307664651307**

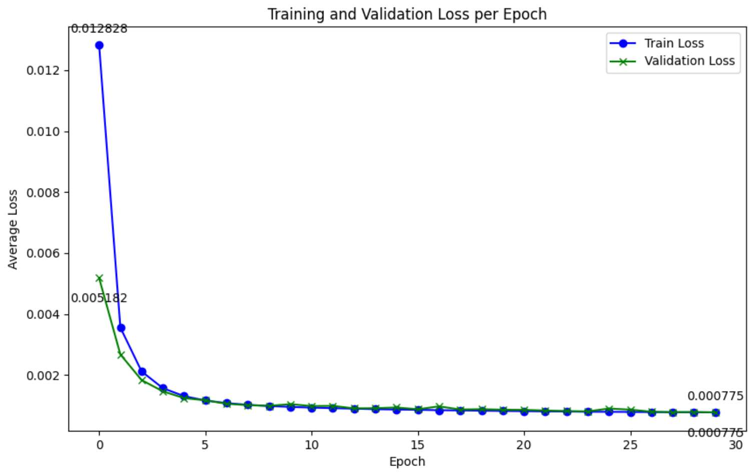


Figure 5: Train and validation loss over epochs

In figure 6 you can see the differences of the original image, the image with the applied noise and the reconstructed image out of the neural network after the first epoch.

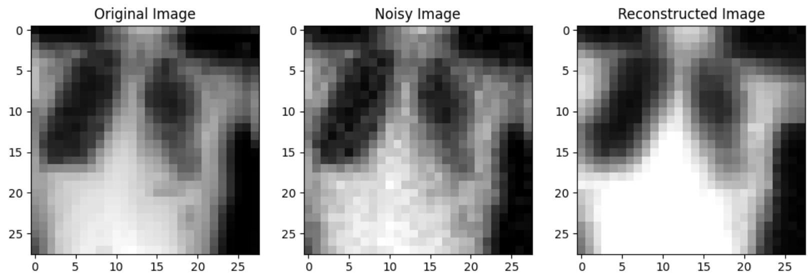


Figure 6: Comparison Original image, noisy image and reconstructed image of epoch 0

In figure 7 you can see the differences of the original image, the image with the applied noise and the reconstructed image out of the neural network after the the last training epoch.

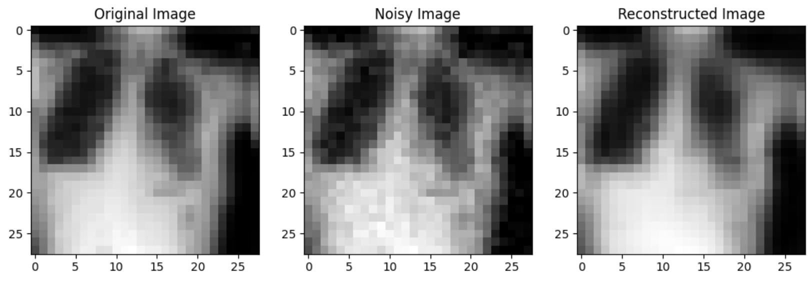


Figure 7: Comparison Original image, noisy image and reconstructed image of epoch 29

As you can see in the two figures above and in the evolution of the training and validation loss, with every epoch the neural network gets better and better in reconstructing the noisy images. But what you can also see in the tables 1 is that the test and validation loss for the last view epochs does not decrease drastically. Meaning that you always have to balance the training efforts (here in computing time) with the excepted error.

References

[1] Wikipedia. *Structural similarity index measure*. Dec. 2024. URL: `https://en.wikipedia.org/wiki/Structural_similarity_index_measure`.