Building A Baseline Convolutional Autoencoder Network for Image Denoising on Fashion MNIST Dataset

Topics In AI (COMP 4740): Final Project Report

University Of Windsor

Submitted To: Prof. Robin Gras

Submitted By:

Vlad Tusinean tusineav@uwindsor.ca, 104823929

Mrinal Walia walia8@uwindsor.ca, 110066886

Diksha diksha@uwindsor.ca, 110062923

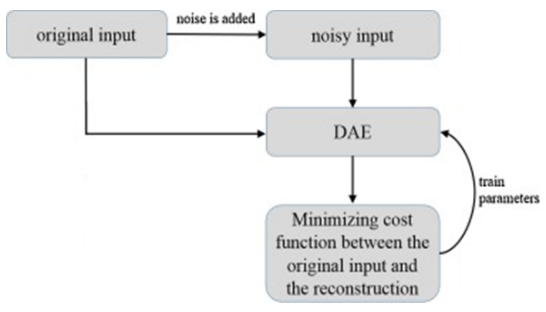
Kaggle Challenge: https://www.kaggle.com/c/insar-fashion-mnist-challenge

GitHub Link: https://github.com/abhiwalia15/Building-A-Baseline-Convolutional-AutoEncoder-Network-For-Image-Denoising-On-Fashion-MNIST-Dataset

Abstract

Image processing is used in real-time in a variety of industries, including security, health care, banking, and face identification. When taking a picture, there's a higher probability that noise will interact with many parts of the environment. To improve the image quality and obtain better classification results, we must clean the image, which is referred to as preprocessing. This study studies a convolutional autoencoder for picture denoising using a proposed compositional subspace method [1].

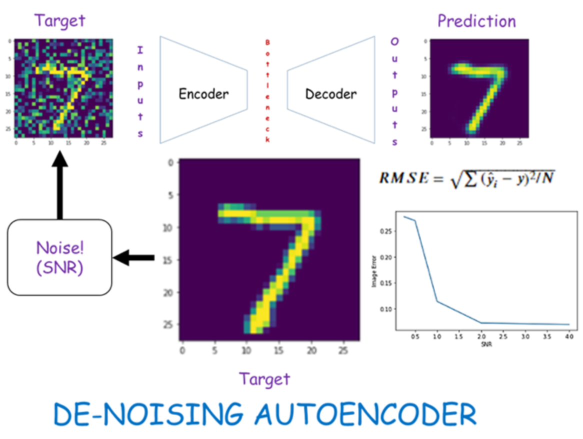
The functional computing stages of a convolutional autoencoder may be understood using this modelling technique, which supplies a structural and rigorous mathematical abstraction. The ideal strategy for modelling a complicated learning function is to use a collection of basic functions to construct a multilayer successive cascaded layers of encoder and decoder network for complex representation. The recommended technique was put to the test using the Fashion-MNIST dataset. The findings of the experiments were discussed and found to be in line with theoretical expectations (autoencoder network is getting a 0.0182373 loss on the training dataset, and we get a loss of 0.01255 for the validation dataset). DAE stands for Denoising AutoEncoder.


original input
noise is added
noisy input
DAE
train parameters
Minimizing cost function between the original input and the reconstruction

[2]

1. Introduction

Convolution Neural Networks (CNN) are immensely powerful tools for analyzing images and drawing out impactful information from them. They are a superior model of artificial neural networks known to give high efficiency in lesser training and testing time. For example, MNIST classifier using artificial neural network with categorical cross entropy and Stochastic Gradient Descent (SGD) is 87% while that achieved by simplest of convolution neural network model is greater than 97%. We can clearly see a remarkable difference in accuracy. Similarly, even learning time is also smaller in convolution neural networks than simple artificial neural networks. CNN uses the pattern recognition feature and hence can successfully be able to classify and detect many images. Any regular CNN implementation used features like padding, strides, volume operations, pooling, and filters. Padding: refers to the process of adding extra data to the edges on matrix so that the Convolution neural network does not lose much information while learning. A valid padding refers to no padding taking place whereas a same padding refers to padding such that the output dimensions still are same as the input. Strides: Stride refers to the number of cells the filter is to be moved in the input to get the next set of results. Pooling: This is used to reduce the size of dimensions and to speed up the learning process.



[3] Image

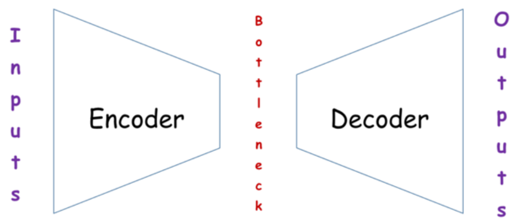
Basically, autoencoding is a data compressing technique. An autoencoder has two parts, the encoder, and the decoder [4]. The autoencoder neural network learns to recreate a compressed representation of the input data. The encoder encodes the data, and the decoder decodes the data. For example, let the input data be x. Then, we can define the encoding function as f(x). A hidden layer h learns the encoding and we get, h=f(x). Finally, a decoding function g reconstructs the input as r=g(h).

Typically, autoencoders are formed of 3 main parts:

1) The encoder

2) The bottleneck layers

3) The decoder



[5]

Information is fed from the paw side into the encoding network which reduces the dimensionality of the computer file because it moves to the correct. At the bottleneck layer, the information has been reduced to the littlest representation allowed by the model. Beyond the bottleneck layer, the information is re-expanded to the next order dimension another time.

Throughout the training process, the inputs are fed through the model and an output representation is obtained. That output is compared with a target (often the first image) representation, and the difference is employed to tune the model’s weight. The interesting aspect of an autoencoder is that the accuracy of the model’s output is directly associated with the dimensions of the bottleneck layer.

2. Literature Review

A. CNN (Convolutional Neural Network) For Image Denoising

Image denoising through the application of a CNN is not new. Many works currently exist that implement some form of convolution neural network for a specific task[6]. (Zhang et al., 2016) proposed one of the most popular examples of this, cited by many papers in this field [7]. This paper introduced a convolution neural network for image denoising that was not specific to a noise value. In fact, this single network could handle variable noise using a residual learning strategy. This paper showed it was possible to develop a performant network for image denoising despite the variability in noise. Image denoising via CNN was taken further by (Zhang et. al., 2018) [8]. where a network was developed that could handle a wider range of noise levels and tackle spatially variant noise. This network was also faster than BM3D without sacrificing denoising performance. (Tian et al., 2020) introduces a dual denoising CNN consisting of four modules: a feature extraction block, an enhancement block, a compression block, and a reconstruction block [9]. This dual network had the ability to extract diverse features to boost the generalizability of the denoiser to complex tasks, fuse global with local features to recover minute details, and reduce complexity through a small filter.

B. Current Development of Auto-Encoders for Image Denoising

One of the famous uses for autoencoders is for image denoising, and many works have been completed in this problem domain. Komal et. al present a general convolutional autoencoder network for use in image denoising, proving that their proposed model for PSNR achieves higher performance than conventional models. Image denoising is one of the applications the of autoencoders. (Keerthi Nayani et al., 1970) proves an autoencoder for the use of image enhancement, which aids greatly in image denoising. They focused on the fashion MNIST dataset, showing incredible results with both denoising and resolution enhancement. Image denoising with autoencoders also has applications in the medical field.

Image denoising has always been fundamental to medical imaging, and the development of autoencoders has greatly aided this field. (Gondara, 2017) proves that an autoencoder can achieve desirable results for image denoising on medical images [10].

3. Auto-Encoder Network Architecture

Auto-Encoders are an artificial neural network introduced in the 1980s by Hinton and the PDP group. It was initially introduced to address the problem of backpropagation without a teacher and using the input data as the teacher. Autoencoders networks are very efficient networks that can learn encodings of unlabeled data or unsupervised learning by training the networks to ignore the signal/ noise in the data representations. These networks are very efficient in compressing and encoding the data, then learning how to reconstruct the data back to the original input representations (as close as possible) from the reduced encoded representations. For this reason, autoencoder networks are primarily used to learn low-dimensional representations (encodings) by training the network to capture the most essential part of the input image for higher-dimensional data. At the very base level, the autoencoder network forms three sub-models: an encoder architecture, latent view representation and a decoder architecture. [12]

1. **Encoder Architecture:** An encoder architecture compresses the input data into an encoded representation of a series of layers with decreasing number of nodes (typically smaller than the input data). And then, it ultimately reduces it to a latent view representation.
2. **Latent View Representation:** This module is also called a bottleneck, and it is the lowest level space in which the inputs are reduced to have compressed knowledge representations. This is the most critical part of the network as the information is preserved in this module.
3. **Decoder Architecture:** The autoencoder's decoding architecture handles decompressing the knowledge representations and is the mirror image of the encoding network with an increasing number of nodes in every layer. It reconstructs the data from its encoded form to similar output inputs, then compares it to the ground truth values to calculate the loss.

One of the applications of the autoencoder network lies in playing with images. However, most images used in image processing and computer vision tasks have noise caused by various intrinsic or extrinsic conditions. These are very disturbing and hard to deal with, and hence image denoising may be a fundamental challenge here. This is where autoencoders come into play. Using the encoding and decoding architecture, we can discover structure within the image to develop a compressed version of the image and then train the neural network to try to copy its inputs to its outputs. Autoencoders are helpful for image denoising due to their conventional architecture, and they can be used to get rid of the noise in the images. Our project proved this by implementing a baseline autoencoder network in Python for image denoising on the Fashion MNIST dataset. Let's see this in action now.

4. Implementation

This section will show how we implement the autoencoder network to denoise the Fashion MNIST dataset using Python.

1. Import Required Libraries

We start by importing all the required packages. We use TensorFlow and Keras framework to design our architecture. We also import pandas, NumPy and sklearn to perform the preprocessing on the dataset. Then we use matplotlib to plot basic visualizations, and one crucial library used is imgaug, which is used to add noise to the dataset.

2. Load FASHION MNIST Dataset

We are using the csv files of the dataset. Each picture in the dataset is 28 pixels in height and width, forming a total of 784 pixels. Each pixel value has a value associated with it, which shows the lightness or darkness of that pixel. The test and train datasets have 785 columns for each pixel value, and the first column in the dataset stands for the class labels (see below). We have 60,000 images in the training dataset and 10,000 sample images in the test dataset. [13]

Below are the following labels assigned to each sample in the train and test dataset.

- 0 | T-shirt/top
- 1 | Trouser
- 2 | Pullover
- 3 | Dress
- 4 | Coat
- 5 | Sandal
- 6 | Shirt
- 7 | Sneaker
- 8 | Bag
- 9 | Ankle boot

# Original dataset was downloaded from https://github.com/zalandoresearch/fashion-mnist

# Dataset was converted to CSV with this script: https://pjreddie.com/projects/mnist-in-csv/

3. Visualizations

We use matplotlib to plot the images from the test and train dataset. It will help us see and visualize the results.

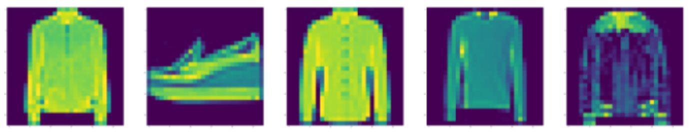
4. Data Preprocessing

We use pandas, NumPy and sklearn to preprocess the data before training. We first normalize and reshape each image in the training set. We are reshaping our input images in the form of a 28*28 form matrix as this is the shape accepted by our convolutional layers.

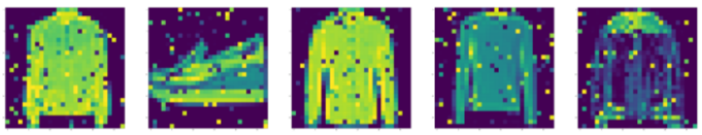
In our case, we are intentionally adding noise to the training images set. We use the imaug library to augment the images with diverse types of noises. We introduce salt and pepper noise to our training set because this type of noise augments sharp and sudden disturbances in the image signals, also called impulse noise. It looks like sparsely occurring white and black pixels. [14]

The results before and after adding the noise using the salt and pepper method can be seen below:

Before:



After:



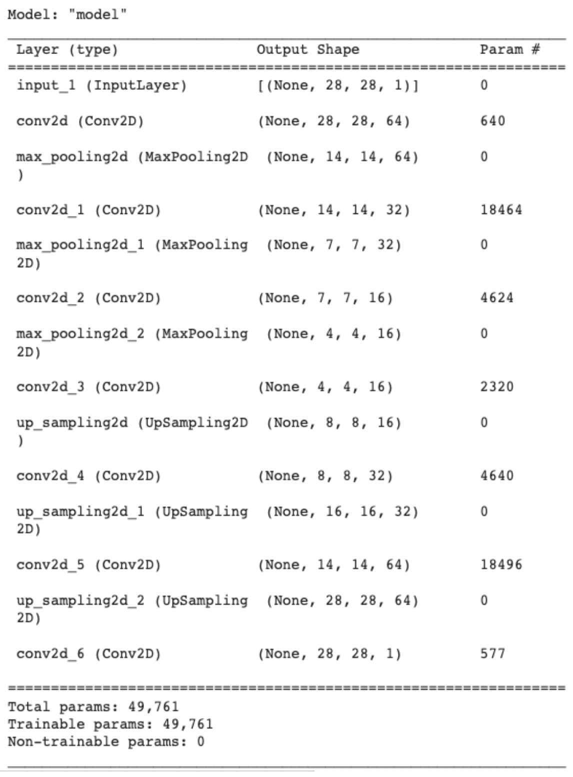
5. Building Architecture

Our autoencoder architecture includes:

Encoder: 3 convolutional and 3 Max Pooling layers (to down sample the image dimensions) stacked on top of each other. We have used Relu as the activation function with padding as the same.

Decoder: The decoding architecture has the same convolutional layers of the exact dimensions in a reverse manner, but instead of max-pooling layers, we have 3 upsampling layers this time. The activation function (relu) and the padding used are the same. The upsampling layers will upsample the dimensions of the input vector to a much higher dimension because the mac pooling action is non-inverse; hence the upsample layer is used to project the reconstruction from a low-resolution feature space.

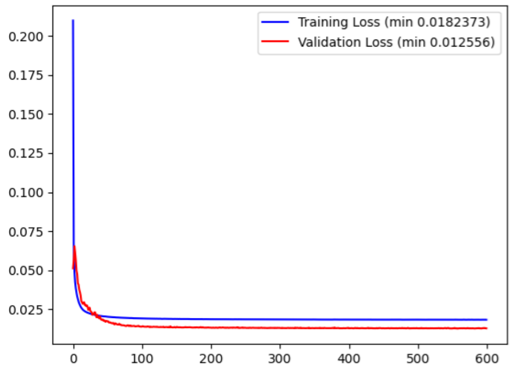
Below, you can see the summary of the model:



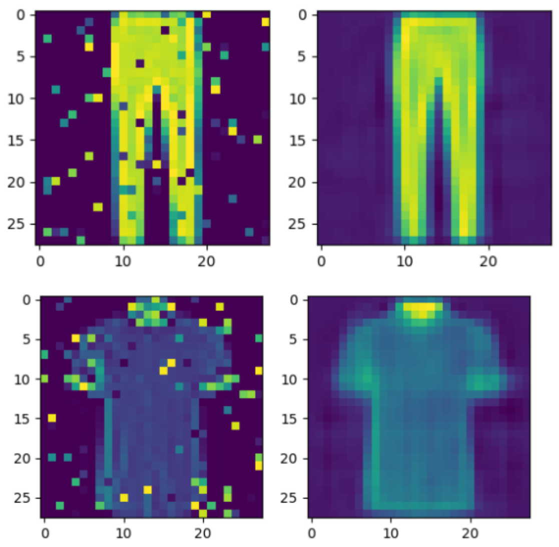
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 28, 28, 1)]       0

 conv2d (Conv2D)             (None, 28, 28, 64)        640

 max_pooling2d (MaxPooling2D  (None, 14, 14, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 32)        18464

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 32)         0
 2D)

 conv2d_2 (Conv2D)           (None, 7, 7, 16)          4624

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 16)         0
 2D)

 conv2d_3 (Conv2D)           (None, 4, 4, 16)          2320

 up_sampling2d (UpSampling2D  (None, 8, 8, 16)         0
 )

 conv2d_4 (Conv2D)           (None, 8, 8, 32)          4640

 up_sampling2d_1 (UpSampling  (None, 16, 16, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 14, 14, 64)        18496

 up_sampling2d_2 (UpSampling  (None, 28, 28, 64)       0
 2D)

 conv2d_6 (Conv2D)           (None, 28, 28, 1)         577

=================================================================
Total params: 49,761
Trainable params: 49,761
Non-trainable params: 0
_________________________________________________________________
```

6. Evaluation

Our autoencoder network is getting a 0.0182373 loss on the training dataset, and we get a loss of 0.01255 for the validation dataset. We use adam optimizer and mse loss function for our autoencoder network. The loss plot is given below:

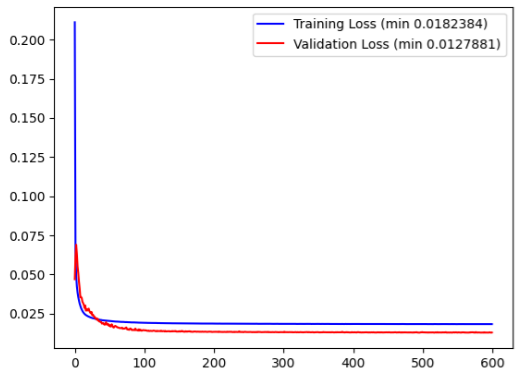


The predictions can be seen below. On the left side, we have a random sample from the dataset with augmented salt and pepper noise, while, on the right side, you can see the prediction made by the autoencoder network.

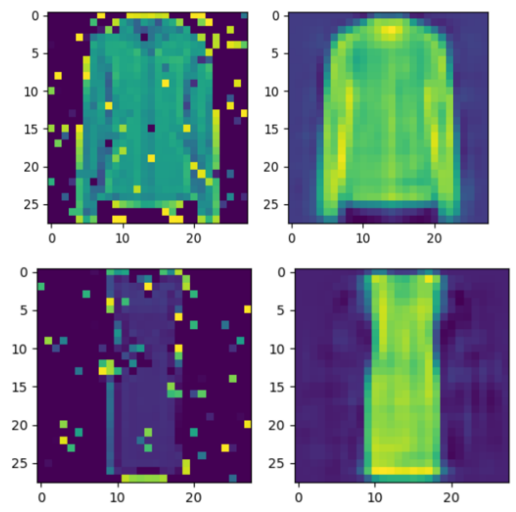


7. Performance Comparison with A Simple CNN Architecture

We have designed a CNN architecture with 4 Conv2D and 2 Maxpooling layers followed by dropout and batch normalization layers. The activation function used is Tanh. We have a flattened layer and two fully connected dense layers at the end of the architecture. We are training the model for 600 epochs, and we are saving the model after training. Using the adam optimizer and mse loss function, we can get a loss of 0.0187 for the training dataset, and for the validation dataset, we get a loss of 0.0132. The loss plot is given below:



The predictions can be seen below. On the left side, we have a random sample from the dataset with augmented salt and pepper noise, while, on the right side, you can see the prediction made by the CNN network.



5. Conclusion

Thus, this report supplies the detailed analysis of building a Convolutional-AutoEncoder-Network-For-Image-Denoising-On-Fashion-MNIST-Dataset considering the different images taken into consideration while predicting the output of the image. Thus, depending upon the input images a comprehensive comparison of a convolutional neural network for image denoising to an autoencoder is done.

In this project, we are intentionally adding noise to the training images set. We introduce salt and pepper noise to our training set because this type of noise augments sharp and sudden disturbances in the image signals, also called impulse noise. Convolutional neural systems (CNNs) have achieved shocking accomplishments over an assortment of areas, including clinical research, and an expanding interest has risen in radiology. Regarding the preprocessing of images, this project explored a particular noise type. The model would train with a unique noise *per batch*, allowing it to generalize better to the data.

6. Future Works

Our work explored a comprehensive comparison of a convolutional neural network for image denoising to an autoencoder for the same task. Of course, more work can be done to test all aspects of the problem, such as applying a grid search of all possible hyperparameters and creating a testing schema that does not involve manually comparing the images. Additionally, more work can be done to visualize the results of these models such as confusion matrices.[15]

Regarding the preprocessing of images, our work explored a specific noise type and noise value. We applied salt and pepper noise at a rate of 0.1 to our training data. Different noise types and noise values should be explored. One possible method is to change our code to include a generator. This generator would allow the model to dynamically load a batch into memory for training/validation purposes. This would allow a custom preprocessing function to apply to these images. In this preprocessing function, the noise type and noise severity could be randomized. That way, the model would train with a unique noise *per batch*, allowing it to generalize better to the data.

The CNN and autoencoder were applied to the Fashion MNIST dataset, which is already normalized to some degree. It would be interesting to explore the performance of the model on denormalized images, or perhaps colored images. After all, image denoising does not only take place on a depth of one dimension.

7. Task Distribution

Group Member Names	Tasks
Vlad Tusinean	CNN Implementation, Model Outputs, Report
Mrinal Walia	Autoencoder Implementation, Report
DIksha	Implementation Report

8. References

[1]. M. Y. W. Teow, "Convolutional Autoencoder for Image Denoising: A Compositional Subspace nd Technology (IICAIET), 2021, pp. 1-6, doi: 10.1109/IICAIET51634.2021.9573657.

[2] How to Reduce Image Noise Using an Autoencoder / Science, M. D. (2021). *DE-NOISING AUTOENCODER* [Photograph]. MLearning.Ai. https://medium.com/mlearning-ai/how-to-reduce-image-noise-using-an-autoencoder-e01146e3ce00

[3] How to Reduce Image Noise Using an Autoencoder / Science, M. D. (2021). *DE-NOISING AUTOENCODER* [Photograph]. MLearning.Ai. https://medium.com/mlearning-ai/how-to-reduce-image-noise-using-an-autoencoder-e01146e3ce00

[4] Science, M. D. (2021b, March 19). *How To Reduce Image Noise Using an Autoencoder* [Research paper]. MLearning.Ai. https://medium.com/mlearning-ai/how-to-reduce-image-noise-using-an-autoencoder-e01146e3ce00

[5] How to Reduce Image Noise Using an Autoencoder / Science, M. D. (2021). *DE-NOISING AUTOENCODER* [Photograph]. MLearning.Ai. https://medium.com/mlearning-ai/how-to-reduce-image-noise-using-an-autoencoder-e01146e3ce00

[6] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," *arXiv.org*, 13-Aug-2016. [Online]. Available: https://arxiv.org/abs/1608.03981. [Accessed: 23-Apr-2022].

[2] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a fast and flexible solution for CNN-based image denoising," *IEEE Xplore*, 25-May-2018. [Online]. Available: https://ieeexplore.ieee.org/document/8365806. [Accessed: 23-Apr-2022].

[7] C. Tian, Y. Xu, W. Zuo, B. Du, C.-W. Lin, and D. Zhang, "Designing and training of a dual CNN for image denoising," *arXiv.org*, 08-Jul-2020. [Online]. Available: https://arxiv.org/abs/2007.03951. [Accessed: 23-Apr-2022].

[8] A. S. Keerthi Nayani, C. Sekhar, M. Srinivasa Rao, and K. Venkata Rao, "Enhancing image resolution and denoising using autoencoder," *SpringerLink*, 01-Jan-1970. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-15-8335-3_50. [Accessed: 23-Apr-2022].

[9]. L. Gondara, "Medical image denoising using convolutional denoising autoencoders," *IEEE Xplore*, 07-Feb-2017. [Online]. Available: https://ieeexplore.ieee.org/document/7836672. [Accessed: 23-Apr-2022].

[10] Autoencoders, Unsupervised Learning, and Deep Architectures,

http://proceedings.mlr.press/v27/baldi12a.html.

[11]. [机器学习] UFLDL笔记 - Autoencoders and Sparsity_WangBo_NLPR的博客...11 https://blog.csdn.net/walilk/article/details/78168358.

[12]. *Sparse Autoencoders using L1 Regularization with PyTorch*, https://debuggercafe.com/sparse-autoencoders-using-l1-regularization-with-pytorch/.

[13] *Image Synthesis with Sparse Autoencoders using L1 ...*, 619a8ddb07be5f31b7a6d051/Image-Synthesis-with-Sparse-Autoencoders-using-L1-Regularization.pdf.

[14]. *How To Reduce Image Noise Using an Autoencoder - Mr. Data ...*, https://mrdatascience.com/how-to-reduce-image-noise-using-an-autoencoder/.

[15]. *How To Reduce Image Noise Using An Autoencoder | by Mr ...*, https://medium.com/mlearning-ai/how-to-reduce-image-noise-using-an-autoencoder-e01146e3ce00.