Noise Removal from Images by Applying Deep Neural Networks

Dijana Ivezic and Caslav Livada(✉)

Faculty of Electrical Engineering, Computing and Information, Osijek, Croatia
dijana.ivezic@etfos.hr, caslav.livada@ferit.hr

Abstract. This article addresses the problem of removing noise from images using neural networks. The type of noise that is removed in this article is salt and pepper noise. This noise represents jumps in intensity in the image, usually caused by data transmission errors. To quantify the success of the noise removal model, PSNR metrics are used. The paper provides a theoretical basis for the task in the form of a description of the neural network and its components, as well as its operation. It also describes deep learning and the differences and advantages of each type. The neural network used in this paper is based on the U-Net architecture. Using the Python programming language and the Keras application user interface, a model was developed to remove added white Gaussian noise and salt and pepper noise of varying intensity from images in the CIFAR-10 and MNIST image databases. The model is applied to the author's image that is not included in the tested databases. The analysis is presented in the PSNR presented for different noise intensities and examples of images with and without noise.

Keywords: Neural networks · Noise removal · Keras

1 Introduction

Image noise is any random change in image brightness that degrades image quality or makes it impossible to read the desired data. It is an unwanted signal added to a pure image signal. The image may be so noisy that details or even the entire image are no longer discernible. Some noise reduction solutions only blur the image and those important details are no longer visible; we lose the important information that led to the capture of the image. Examples of such solutions are the median filter or Gaussian smoothing. Computers can be taught to remove noise in a different way by applying Deep Learning. Deep Learning is a branch of artificial intelligence science that allows us to understand language by computer, recognize images, make decisions based on data, etc. What would be easy for a human to do is very difficult for a computer to do.

Deep Learning gets its name from the fact that the layers of the neural network used go deep, into multiple layers that we can understand by imagining how each layer is responsible for recognizing certain features of images. This work is about using Deep Learning to remove noise from images. When a noisy image is imagined, the parts of the image may not be visible at all under the noise. These parts need to be predicted to eventually get a clean and complete image and are predicted by using the rest of the image, blind or non-blind. In non-blind image filling, the algorithm does not know exactly where the irregularities are, which requires human work. In blind image filling, the algorithm automatically detects the noise to be removed, which makes the system more complex, but more interesting from the point of view of automating noise removal, which is very important for a large number of images. Deep learning learns system parameters from data without human intervention. This work is based on two types of noise: additive white Gaussian noise and salt and pepper noise. Peak signal to noise ratio (PSNR) is used to quantify the success of the denoising model. The higher the PSNR value, the better the quality of the reconstructed image.

2 Related Work

2.1 Real Image Denoising with Feature Attention

According to Anwar and Barnes [1], neural network models often work well on data generated during training, i.e. data obtained by artificially adding noise to a clear image, and are less successful on actual data, which is why the networks are trained. The practicality of such networks is limited, which they exploit in their work by presenting the RIDNet network for blind noise removal. They present a model that ensures top results using a modular network where increasing the number of modules helps improve performance, unlike most previous solutions.

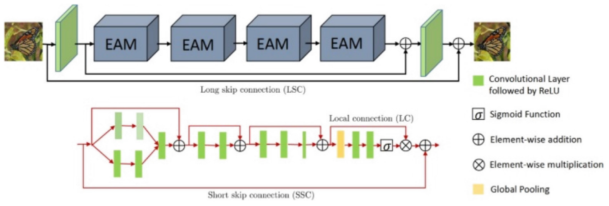

EAM
EAM
EAM
EAM
Long skip connection (LSC)
Local connection (LC)
Short skip connection (SSC)
Convolutional Layer followed by ReLU
Sigmoid Function
Element-wise addition
Element-wise multiplication
Global Pooling

Fig. 1. Proposed architecture [1].

The upper part of Fig. 1 shows the essence of the work of the model. The modules for property extraction and reconstruction are marked in green; at the beginning and at the end. They consist of only one convolutional layer. Following the arrows, we see that after the feature extraction come the EAM ("enhancement attention modules") blocks, which are the main part of a separate feature learning module and whose task is to enhance attention to the features of the whole model. The EAM blocks are shown one by one and there are four of them in the proposed network, it is possible to add more to improve the performance. So the network can be extended to create a very deep network if needed [1].

A comparison of the presented model with other models (the presented model is shown under "Ours", last picture) is shown in Fig. 2. The first image (shown under "Noisy") is an image with noise. The image is taken from the RNI15 database.

The success of the proposed network is clearly visible, especially in the details shown in the figure. In their paper [1], more detailed comparisons are presented as well as the PSNR value versus other solutions.

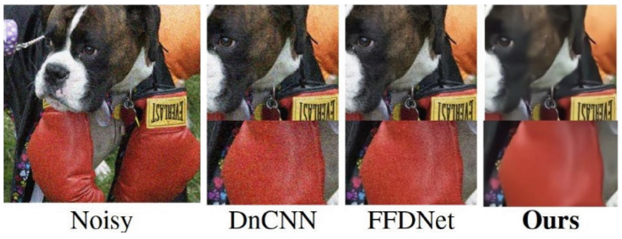


Fig. 2. Comparison with other architectures [1].

2.2 A Physics-Based Noise Formation Model for Extreme Low-Light Raw Denoising

Learned algorithms for removing image noise have difficulty using actual images from cameras or other sources because they are trained with training images that do not follow real noise well. The problem that Vei et al. [2] are addressing is modeling noise that is very similar to noise in real images, mainly noise caused by electronics in digital devices and noise that occurs in low light. It is clear that collecting real data is a tedious and expensive task, and that the quality of the modeled data greatly affects the usability of systems trained on real images using this data. By studying the physical processes involved in the generation of real images, they have developed a better model for the added noise in images based on CMOS photosensors that describes the real noise well. The obtained data show that the existing models for noise removal from images trained on the obtained more realistic data are more successful. The researchers [2] also introduce the adjustment of the parameters of their noise generation to describe as accurately as possible the noise generated on the selected vendor's camera.

Figure 3 shows four different stages in which different types of noise are inevitably added to the photons from nature in the digital camera. In the blue part, the recording noise is added to the photons entering the camera lens because of the nature of the light particles. In the orange part, which refers to electrons, the signal from the previous stage is multiplied by a quantity that describes the error in the signal coming out of the sensor, and the noise of the so-called dark current, which is generated in light sensors when there is no photon, is added. The green part shows the addition of the noise caused by the random movement of electrons in the conductor, resulting in voltage changes. The last part is the addition of the noise caused by quantization in an analogue-to-digital converter.

2.3 Noise2Void

Deep neural networks are traditionally trained with two sets of images. One set of images contains pure images, and these images are taken as the absolute truth and used

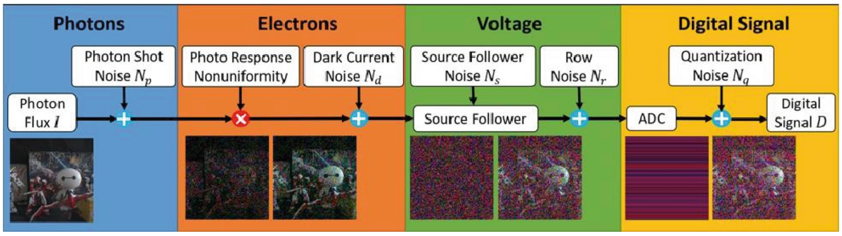


Fig. 3. Visualization of noise and display of the final image in individual stages [2].

as reference in training. The second set of images contains images with noise, which we then want to remove. Sometimes it is difficult, expensive, or even impossible to obtain clean images with which to train a network, and such networks have their limitations. Due to these limitations, Lehtinen et al. [3] have proposed a model of a network that trains with two noisy image sets. These two sets do not contain the same images, but the noise in the images of the first set is different from the noise in the images of the second set. Such an approach is called “Noise2Noise”. This is great if a similar can be used image multiple times and when retrieving these images, different noises due to random physical processes are achieved, which should be the only difference between the images of the two sets. Krull et al. [4] have even fewer requirements, so they only need one set for training, a set of images with noise. They realise that it is sometimes impossible to obtain two images that differ only in the noise, or even that it is generally impossible to obtain two images. They see an application in medicine or laboratory imaging where it is important to remove noise from the one available image of which there is no pure version or a version with different noise. They point out that their version is sometimes less successful than previous solutions, but that the difference is very small. This difference is understandable because their network simply works with less available information. The model has been called Noise2Void [4].

2.4 Multi-level Wavelet-CNN for Image Restoration

When developing neural network models, scientists often have to decide between how long to run and train the network and how much detail they want to get in the processed image. To increase the detail of an image, researchers often make their networks deeper, have multiple layers, or increase the size of the filters applied to the image during training. Both of these methods result in an increase in program duration. In their work, Liu et al. [5] try to eliminate this problem, at least partially, by presenting their new model that preserves textures in processed images very well and has a high PSNR value. The execution time in the proposed architecture is clearly not the fastest so far, but the goal is to achieve a balance between preserved features and execution speed. The main idea is to use the discrete wavelet transform, which is similar to the Fourier transform, but instead of representing the signal as a sum of infinite sine waves, they use signals of short duration at specific times of specific frequencies. Such a transform is reversible. Its model can be used not only to remove noise, but also to increase image resolution and eliminate errors in JPEG compression.

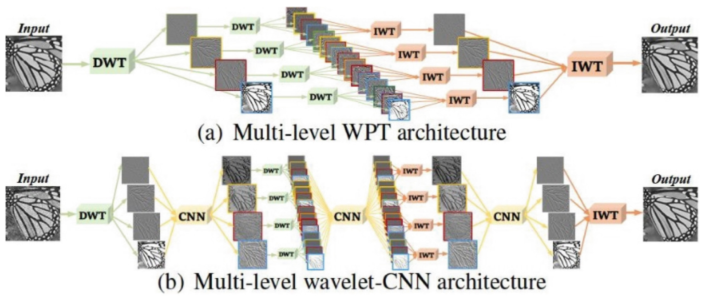

Input
DWT
IWT
Output
(a) Multi-level WPT architecture
Input
DWT
CNN
CNN
CNN
IWT
Output
(b) Multi-level wavelet-CNN architecture

Fig. 4. Display of discrete wavelet transformation (a) and proposed architecture (b) [5].

Four images are obtained representing the previous image by applying the discrete wave transformation and forming a plane. On these new images, the transformation can be applied again to obtain four more smaller images (Fig. 4 (a)) [5]. The resulting images are important features of the original image on which the network is trained. Figure 4 (a) also shows the reversibility of the discrete wave transform. The proposed architecture is based on the U-net architecture with the use of the discrete wave transform instead of the originally proposed methods for image transformation. After each transform application, a convolutional neural network is generated (Fig. 4 (b)).

2.5 Deep Image Prior

Ulyanov et al. [6] believe that a deep neural network can recognize image features without training and that the success of the network depends on adapting the network architecture to the data. In the trained networks, the network parameters are set to random values and changed by computing information from a large number of images. The trained networks perform well on images similar to those on which they were trained. The proposed model uses a single image to read new parameters from randomly set parameters and obtain a new, improved image in terms of noise reduction or resolution increase. The authors use the structure of the network as a source of information and show very good results without requiring a large number of images to train.

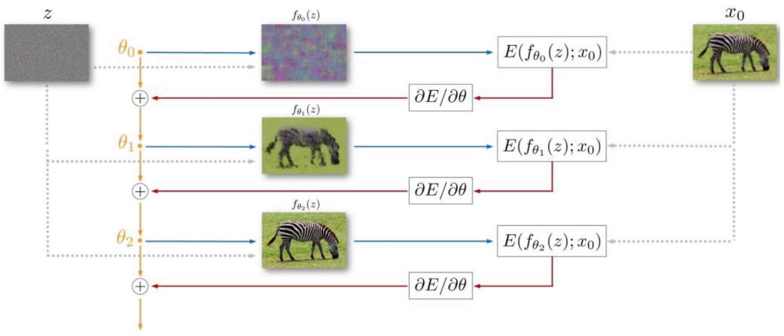
Each iteration of the neural network searches for the smallest difference between the image obtained by the network and the initial image of the network with random parameters. By reducing these differences, the obtained parameters are used in painting a new image until a satisfactory result is obtained.

x_0 denotes the initial image, $f_\theta(\mathrm{z})$ denotes the obtained images with different mesh parameters, Z denotes the fixed tensor on which new images are painted on. E denotes the differences between the images, and θ denotes the network parameters [6, 7]. This is illustrated in Fig. 5.

3 Theoretical Background and Technologies Used

3.1 Deep Learning

Convolutional neural networks Convolutional neural networks show very good results in detecting objects on images, removing noise or similar problems in image processing.



z
$f_{\theta_0}(z)$
x_0
θ_0
$E(f_{\theta_0}(z); x_0)$
$\partial E/\partial\theta$
$f_{\theta_1}(z)$
θ_1
$E(f_{\theta_1}(z); x_0)$
$\partial E/\partial\theta$
$f_{\theta_2}(z)$
θ_2
$E(f_{\theta_2}(z); x_0)$
$\partial E/\partial\theta$

Fig. 5. Basic idea of the model [6].

They are based on MLP type networks with some modifications. They allow very deep networks in a way that eliminates some of the problems of deep MLP networks, such as overfitting to the data and a large number of adjustable parameters, speeding up their training. They take their name from the linear mathematical operation of convolution, which replaces matrix operations during training. Such an operation is denoted by $*$. For continuous functions (signals), it is calculated using Equation 1. The function f is often referred to as the input, the function g as the kernel of the operation, and the output of the operation as the feature map. In convolution, one of the functions is rotated and shifted by the parameter τ. The operation is commutative and is shown in Fig. 6, where the input is the input and the kernel is the kernel.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{1}$$

Since computers do not work with continuous but with discrete values, the discrete convolution is more interesting in this case and is represented by Eq. 2.

$$(f * g)(t) = \sum_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2}$$

When processing an image as an input, there is no one-dimensional function, but a convolution on a two-dimensional image with a two-dimensional core. Image is deoted as J, and the nucleus as K; the convolution is shown by Eq. 3 [8].

$$(I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n) \tag{3}$$

Intuitively, neurons can be thought of as pixels of an image arranged in rows and columns. Each neuron receives the intensity of a particular pixel as input. As in MLP networks, these input neurons are connected to the next hidden layer, but not together; instead, a particular neuron from the next layer is connected to a smaller number of neurons from the input layer (Fig. 7). The local receptive held represents the fixed

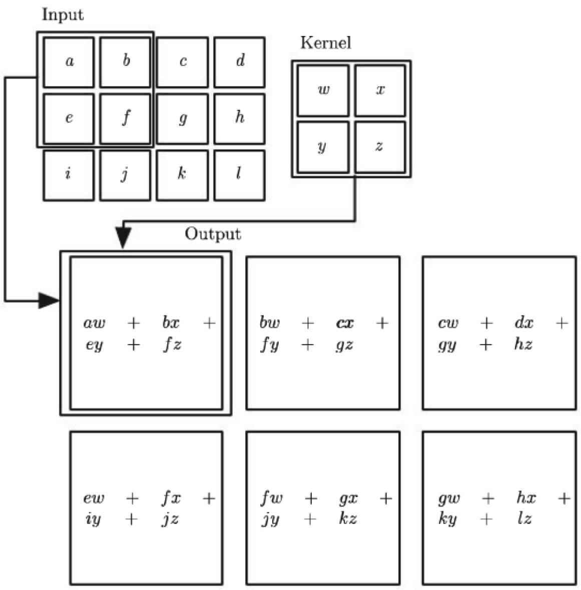


Fig. 6. Convolution [8].

number of neurons to which the neuron from the first hidden layer is connected, and is moved around the image for a given step so that each neuron from the first hidden layer studies a part of the image bounding this field.

All neurons in the first hidden layer have the same threshold and equal weights for connections with the previous layer. This allows each hidden layer to recognize a particular feature that may be located anywhere in the image. The mapping between the input neurons and the next hidden layer represents a feature map, and the common thresholds and weights are called the kernel or filter. A single convolutional layer is defined with multiple feature maps [7].

Each layer of the convolutional network consists of different operations at the entrance of the respective layer. After the described convolution, the information passes through the activation function of each neuron, and then the output of this activation function is processed by aggregating values. This aggregation allows only the essential information for daytime processing to be stored in the rest of the network, further reducing the parameters required to define an individual network. For convolutional meshes, it is often important to identify whether an image contains the desired features, rather than exactly where they are located in a pixel. The aggregation layer processes the output of the individual feature maps and describes whether the desired feature is located in a particular part of the image. An example of an aggregation would be max-pooling, an association where a layer detects the maximum outputs in a particular part of the image. They allow translation independence, in that translating certain features does not significantly change the outputs of the aggregation layers, and they also allow independence from different feature transformations by processing outputs from different

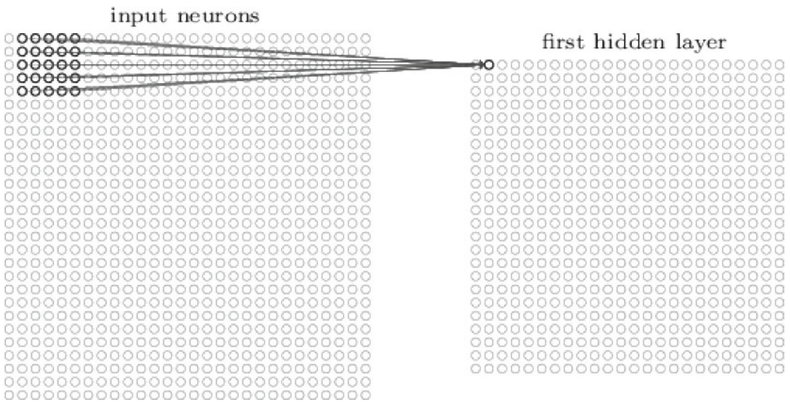


Fig. 7. Intuitive representation of convolution [7].

convolutional layers that are parameterized differently, in which case they would need to associate fewer layers than convolutional layers (Fig. 8) [7, 8].

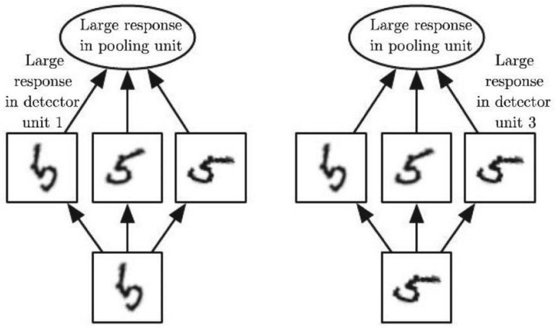

Large response in pooling unit
Large response in detector unit 1
Large response in pooling unit
Large response in detector unit 3

Fig. 8. The aggregation layer allows independence from different feature transformations [8].

The convolution layer is not rotation-independent, so different parameters are required for each number rotation. The last row of Fig. 8 shows two examples of the number five rotated differently, and the row above shows the convolution layer that detects a particular rotation of the number. The left rotation of the number five was detected by the first convolutional layer, while the right rotation was detected by the third convolutional layer. The positive thing about the association layers is that a separate layer is not required for each rotation, but a good response to each of the rotations indicates that the number five is in the given mapping. This simplifies the network; the next number of neurons in the network decreases.

Autoencoder Autocoders are convolutional networks that aim to map input and output almost identically and train like any other feedforward network. They consist of a part that encodes data and a part that decodes data. An autocoder that learns to literally map input to output has learned to simulate an identity function that is not useful. Their purpose is to learn the useful features of images, which is accomplished by constraining their architecture so that they are unable to learn the simple function specified. The constraints force the network to ignore some of the input's information, which is useful in denoising. By gradient descent, autoencoders minimize the error function like general neural networks, but with different parameters, resulting in a denoising autoencoder (DAE) that minimizes the difference between output (processed image with noise) and presented image without noise. DAE thus has the task of removing noise instead of blindly copying the input [4, 8].

U-Net architecture Ronnenberger et al. [9] present a fast network that makes good use of the input images and provides good query of the required features and their localization.

The mesh consists of two parts: a narrowing part and a widening part, which is an innovation compared to other solutions. The narrowing part follows the usual convolutional network with convolution and feature aggregation based on max-pooling aggregation with a step of two. After each feature aggregation, the number of feature maps is doubled so that the context can be passed to other layers. The activation function is the ReLU (rectified linear unit) mentioned above.

The expanding part also follows the usual convolutional network, but instead of merging, it uses inverse convolution, which instead of compressing features, expands the compressed information from the previous layers and halves the number of feature maps. Each inverse convolution adds information from the corresponding constriction layer to the current layer, introducing information about the position of features and creating an architecture reminiscent of the letter U (see Fig. 9). The blue arrows show the convolution with 3 × 3 kernel, the gray arrows add previous matching layers, the red arrows compress the features, the green arrows convert the inverse convolution, and the light blue arrows convert the convolution with 1 × 1 kernel.

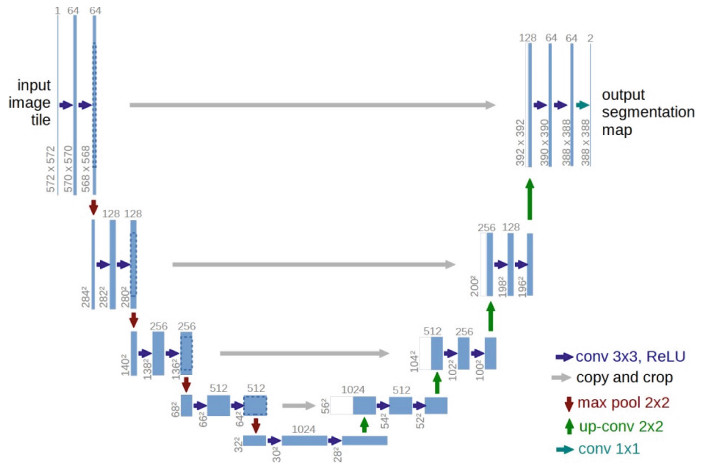


Fig. 9. U-Net architecture [10].

3.2 Keras

Keras is a Python application interface that allows users to easily implement deep learning algorithms. The authors of Keras believe that the principles of Deep Learning are simple and that their implementation should be simple as well. Learning how to use this application user interface is very easy, especially for some simpler projects, the interface supports even the most complex ideas, which of course affects the usability. Basic predefined modules can be used or new modules can be created as needed, Keras is very customizable. It enables engineers and scientists to turn their ideas into tangible results faster. It supports saving already trained networks for use on other computers or platforms, as well as training on processors, graphics cards, and other hardware [11].

4 Image Noise Removal

4.1 Chosen Model

The architecture of the model is based on the U-Net architecture explained in Sect. 3.1, with Fig. 9 showing the architecture of the original work. There are 5 levels of the descending part of the net with 4 associations of features starting from 64 feature maps. When doubled at each level, the left part of the letter U ends up with 1024 feature maps, and such a network has about 31 million variable parameters in total, which challenges the limitation of hardware and execution time. If the U-Net architecture is adjusted so that the initial feature map contains 32 instead of 64, the number of network parameters changes drastically to about 7 million, and such a network can be trained much faster with fewer computing resources. Such an adapted architecture was used to implement autoencoders and remove noise with blind padding.

If the number of feature maps is reduced further, e.g., to 16, it is clear that the number of customizable mesh parameters would be reduced again to about 2 million parameters. However, such a network slows down the error function values much more slowly; it requires more epochs to train and achieves worse results for the same values of the additive white Gaussian noise and salt and pepper noise parameters. Several different initial feature maps were tested. Empirical data suggest that as the size of the network is reduced, the error function values start at a higher number, and as the network remembers fewer features, the quality of the reconstructed images is lower.

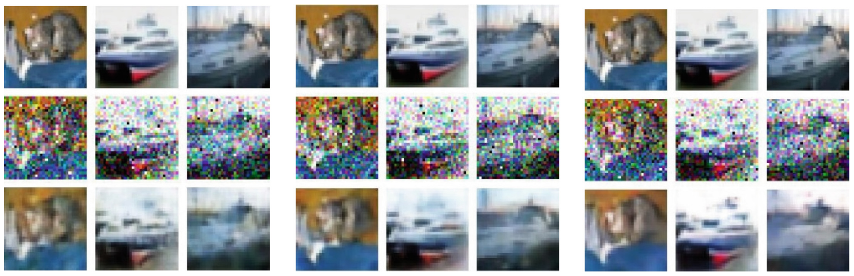


Fig. 10. Noise removal with different starting number of feature maps: 16, 32, 64.

Noise removal is shown in Fig. 10 with a varying number of feature maps, with noise-free images in the first row, images with noise in the next row, and noise-removed images in the last row. In the first part of Fig. 10, it was found that the noise removal is much worse than in the other two parts of the image where the noise is removed uniformly. All 3 networks were trained with the same amount of noise and with the same number of levels. Figure 10 confirms the choice of 32 feature maps as the network parameters, since the noise removal is sufficiently good and less computational resources are required.

The CIFAR-10 and MNIST image databases together contain 110,000 training images. Computing the error after each image propagated through the network would take a long time, so the error is computed after a single series of multiple images. Increasing the number of images in a series speeds up the runtime, but the computer hardware requirements are also higher; the training requires more memory. The number of 150 images in a series during the network training was chosen, and such an approximation to the actual errors does not have a great impact on the training result.

The network can be trained with only one of the two image databases listed, and the results obtained are of course interesting. If fewer images pass through the net during training, you train faster. With the same amount of noise and the same number of training epochs, two models were created, one for each of the noise types.

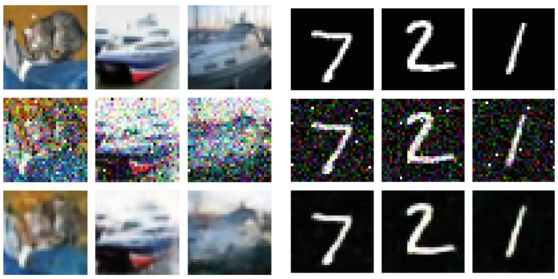


Fig. 11. Noise removal by a model trained on the CIFAR-10 image base.

The model trained on the CIFAR-10 image database removes the standard noise from images from both databases, but the images with noise removed from the MNIST database are no longer completely black and white (see Fig. 11). Noise removal based on this model is satisfactory. When training with images from the MNIST database only, the situation is completely different (see Fig. 12). The denoised images from the MNIST database are completely black and white, like the original image, but the images from the CIFAR-10 database are also colorless, unlike the original images. A network trained in this way does not give satisfactory results because the network has not been exposed to color images. The model for this paper is trained on both image databases because it produces the best results for images from both databases.

It remains to choose the initial amount of the two types of noise.

The parameters that determine the strength of the noise are the standard deviation (Gaussian distributions) σ for additive white Gaussian noise and the probability (pixels that have a minimum or maximum value) p for salt and pepper noise.

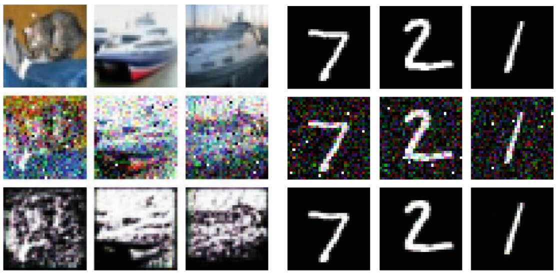


Fig. 12. Noise removal by a model trained on the MNIST image base.

Figure 13 shows the different noise intensities in the images from CIFAR-10 and MNIST. Images from the CIFAR-10 database are much less resistant to noise than images from another database due to their greater complexity, having fewer edges to detect, and being black and white. It was found that the image of the plane is difficult to detect already at the third increase of the noise parameters, while the image of the number 0 is detectable at all noise intensities. Parameters $\sigma - 30, p - 0.04$ were chosen for network training.

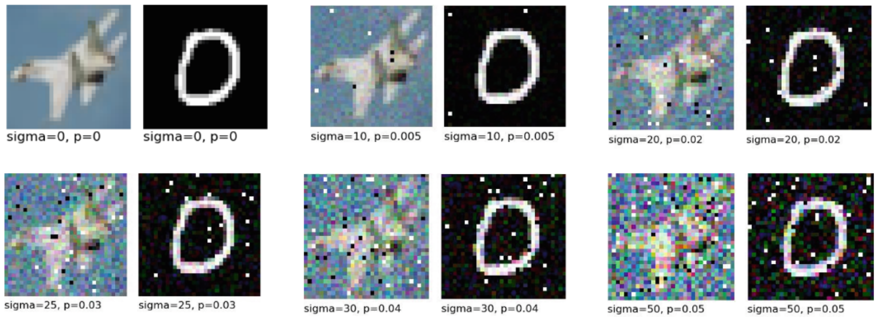


Fig. 13. Various intensities of σ and p.

4.2 Experimental Results

The network was trained for 20 epochs with a lot size of 150. Noise reduction is first shown for images with $\sigma - 30$ and $p - 0.04$ as selected for training the network.

The noise removal from the CIFAR-10 image database (first three thumbnails) is shown on the left in Fig. 14, and the noise removal from the MNIST database is shown on the right. The thumbnails show, in this order: the original image, the image with noise, and the image with noise removed. It can be seen that the original image and the image with noise removed are not completely identical. This discrepancy is reflected in a PSNR of about 16 dB for noisy images compared to the original images. The PSNR for the denoised image from the CIFAR-10 database is 29.12 dB and from the MNIST database is 32.22 dB.

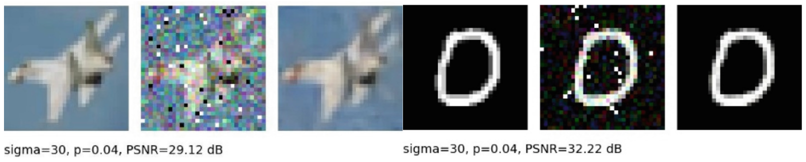

sigma=30, p=0.04, PSNR=29.12 dB
sigma=30, p=0.04, PSNR=32.22 dB

Fig. 14. Noise removal with $\sigma - 30$ and $p - 0.04$.

The improvement after denoising is clearly visible and is shown numerically. The images from the MNIST database are more resistant to noise due to their simplicity and are easier to denoise, which is evident from the reported PSNR values.

Removing only one of the two types of noise listed was also successful. Figure 15 shows the removal of only additive white Gaussian noise, and Fig. 16 shows the removal of only salt and pepper noise.

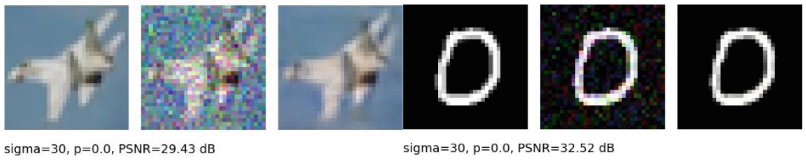

sigma=30, p=0.0, PSNR=29.43 dB
sigma=30, p=0.0, PSNR=32.52 dB

Fig. 15. Noise removal with $\sigma - 30$ and $p - 0.0$.

The PSNR between the original and the image with Gaussian noise is about 18 dB for images from the CIFAR-10 database and about 22 dB for images from the MNIST database. The PSNR for the image with Gaussian noise removed is better than the PSNR for the image with both noises removed.

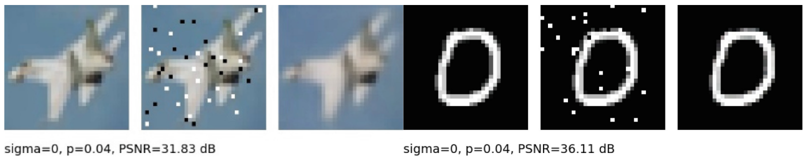

sigma=0, p=0.04, PSNR=31.83 dB
sigma=0, p=0.04, PSNR=36.11 dB

Fig. 16. Noise removal with $\sigma - 0$ and $p - 0.04$.

When only salt and pepper noise is used, the PSNR increases from about 20 dB and 17 dB to 31.83 dB and 36.11 dB for CIFAR-10 and MNIST basis images, respectively, which is an even larger increase compared to the previous examples. The network removes both noises well, both together and individually, regardless of the fact that the images were not exposed with the application of individual noise during training.

The question arises as to what results the network achieves with different noise parameters than the examples in Fig. 13. The noise from the thumbnails left and right (smaller and larger) was selected from the noise selected for training (thumbnails from Fig. 13). These are $\sigma - 25$, $p - 0.03$ and $\sigma - 50$, $p - 0.05$.

If the noise is reduced compared to the noise during training, the PSNR increases (see Fig. 17), otherwise the PSNR decreases (see Fig. 18). The difference can be seen in the appearance of the images themselves. Deep networks do not literally learn where the noise is in the image (that would be difficult since noise is always random), but they learn what part of the image we consider the original part of the image, what noise was added, and how to remove it. The network has seen only certain noise intensities during training, while it has not seen others, and it gives good results even when the parameters are changed. When much more noise is added than it was trained to see, the network only partially removes the noise, but is still relatively successful.

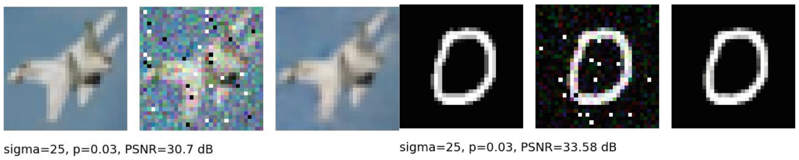

sigma=25, p=0.03, PSNR=30.7 dB
sigma=25, p=0.03, PSNR=33.58 dB

Fig. 17. Noise removal with $\sigma - 25$ and $p - 0.04$.

It is noticeable that the network does a worse job of removing noise based on the CIFAR-10 images, while it does an almost complete job of removing noise based on the MNIST images due to the simplicity of these images as mentioned earlier. In any case, the PSNR improves over the image with noise.

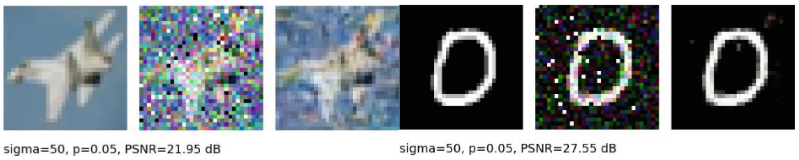

sigma=50, p=0.05, PSNR=21.95 dB
sigma=50, p=0.05, PSNR=27.55 dB

Fig. 18. Noise removal with $\sigma - 50$ and $p - 0.05$.

Table 1 shows the results obtained for both image databases together. The green color indicates the experiment with the images whose σ and p are equal to the one used in the training. The results of the other experiments can be compared with this first experiment. An experiment with reduced noise is highlighted in blue, and an experiment with increased noise is highlighted in red. The previous conclusions are clearly visible at one point. Experiments without a certain type of noise are shown colorless. The first PSNR value indicates the difference between the original image and the image with noise reduction, the second the difference between the original image and the image with noise. Comparing these two PSNRs, a quantitative improvement in image quality can be seen in each of the experiments.

4.3 Application of the Model

The beauty of Deep Learning is the application of the model to similar problems, but with which the network has not yet encountered. Noise removal from a 3088×3088

image taken with a mobile device was demonstrated. Since the network expects a tensor (32,32,3) as input, rather than (3088,3088,3), the image is adjusted in two ways. The first method reduces the image to a 32 × 32 image, which loses a lot of image detail, but still produces a new image that the network has not yet seen. The second method is to split a higher resolution image into 32 × 32 parts and finally combine them into a whole image. The noise used is the noise selected during training.

Table 1. Results obtained for certain σ and p.

Image Database	σ	p	$PSNR_U$	$PSNR_U$
CIFAR-10	30	0.04	29.12	16.06
	30	0.00	29.43	18.81
	0	0.04	31.83	19.67
	25	0.03	30.70	17.54
	50	0.05	21.95	13.25
MNIST	30	0.04	32.22	15.93
	30	0.00	32.52	21.62
	0	0.04	36.11	17.31
	25	0.03	33.58	16.42
	50	0.05	27.55	13.49

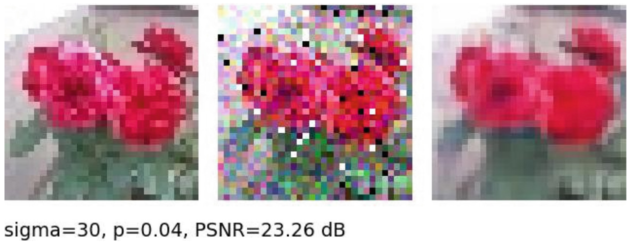


Fig. 19. Noise removal from the new image with reduced resolution.

The resolution of the acquired image was reduced, noise was added, and this noise was eliminated (see Fig. 19). From about 16 dB for PSNR, the improvement is 23.26 dB. The network did a good job of removing the noise on the previously unseen image of a rose.

The image was split into smaller parts, noise was applied to each part, and then removed. The network does a good job on this example as well. Details can be seen in Fig. 20, where the drawbacks of this approach can be seen, but the results are certainly better than expected. Small images can be seen that make up the higher resolution image in the removed noise image, but these results are considered good since the network is not trained to remove noise from such images.

A PSNR of about 16 dB results in 31.47 dB for the entire image shown in Fig. 21. Without details, it is difficult to see the difference.

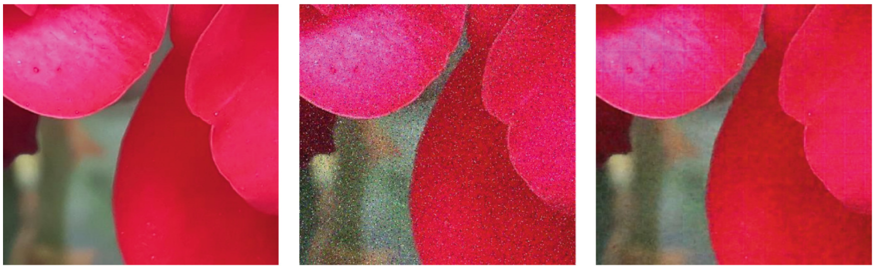


Fig. 20. High resolution image details.

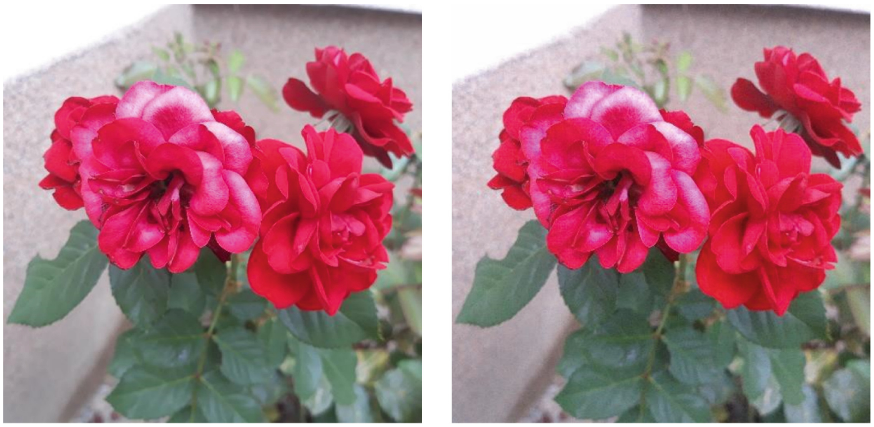


Fig. 21. Original image (left) and denoised image (right).

5 Conclusion

Additive white Gaussian noise and salt and pepper noise were successfully removed from images in the CIFAR-10 and MNIST image databases. It has been shown that regardless of the amount of added noise, the network eliminates it at least partially. Image quality is multiplied in all examples. Deep neural networks, despite their complexity, apply well to this problem. Understanding and applying deep neural networks in detail is not an easy task, but it has been shown here to be worthwhile. The deeper we get into the "why" and "how" questions, the more decades of human research and technological development have brought us to the present day. Using the Keras API to implement a neural network model is far simpler and allows anyone interested to model and train at least a simple network and perhaps encourage that person to delve deeper into the world of neural networks. Even complex mathematics does not give us answers to determine certain parameters of the network, but one can arrive at an optimal solution with various trials, and with experience these trials become more and more meaningful. We have determined some parameters empirically and thus gained experience that we will use in further work with neural networks.

Of course, these types of noise can be eliminated by changing the architecture with even better results, and the changes require the knowledge gained by working in this area. Some of the different architectures that show very good results are presented in this paper.

References

1. Anwar, S., Barnes, N.: Real image denoising with feature attention. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3155–3164 (2019)
2. Wei, K., Fu, Y., Yang, J., Huang, H.: A physics-based noise formation model for extreme low-light raw denoising. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2758–2767 (2020)
3. Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., Aila, T: Noise2Noise: Learning Image Restoration Without Clean Data (2018). In: arXiv preprint arXiv:1803.04189
4. Krull, A., Buchholz, T.O., Jug, F.: Noise2void-learning denoising from single noisy images, In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2129–2137 (2019)
5. Liu, P., Zhang, H., Zhang, K., Lin, L., Zuo, W.: Noise2Noise: Multi-level wavelet-CNN for image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 773–782 (2018)
6. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Deep image prior. Int. J. Comput. Vision **128**(7), 1867–1888 (2020). https://doi.org/10.1007/s11263-020-01303-4
7. Nielsen, M.A.: Neural Networks and Deep Learning, vol. 25. Determination Press, San Francisco (2015)
8. Kriesel, D.: A Brief Introduction to Neural Networks. http://www.dkriesel.com/en/science/neural_networks. Last Accessed 5 Dec 2021
9. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 234–241 (2015)
10. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
11. Keras: https://keras.io/about/. Last accessed 12 Feb 2022