International Journal of
Science and Engineering Investigations
vol. 11, issue 120, January 2022
Received on January 4, 2022
ISSN: 2251-8843

Application of Deep Learning Methods to Implement Super Resolution

Mohammadreza Faraji[1], Atefeh Hasan-Zadeh[2]
[1]BSc in Computer Engineering, Fouman Faculty of Engineering, College of Engineering, University of Tehran, Iran
[2]Fouman Faculty of Engineering, College of Engineering, University of Tehran, Iran
([1]m.rezafaraji@ut.ac.ir, [2]hasanzadeh.a@ut.ac.ir)

Abstract- Recognizing employees' feelings is also important in various organizations to increase work efficiency. It is also essential for government officials to make rational policies and strategic decisions. Several studies have been carried out to construct an automated emotional recognition system. These systems are usually constructed using brain signals. These studies found that brain signals can be used to classify numerous emotional states. This process appears difficult, especially as the brain signals are not stable because the reactions to the different emotional states that affect the brain signals are varied. Consequently, the performance of emotion recognition systems through brain signals depends on the effectiveness of the algorithms used. Recent attention has been given to the study of electroencephalographic signalling (EEG) because of its availability. Various methods can be used to classify emotions after a series of preprocessing, including the K-nearest neighbour (KNN) algorithm, decision tree, Bayes classifier, and convolutional neural network (CNN). Of course, we can use other deep learning architectures such as Generative adversarial network (GAN) for this classification. Still, in this study, we will use the CNN network architecture to classify emotions. In this paper, we use 3D_CNN networks for emotional classification, which has a good accuracy of 0.980729 in train data. This shows that the usage of this architecture has a small error and performs the task assigned to it with great precision and a small error. We can use different methods to implement super resolution, which today, deep learning methods have been considered due to their unique features and we can see that one of the deep learning methods to implement super resolution is to use a convolutional method that provide us good accuracy. We also find that the smallest changes in network architecture and parameters have a large impact on network accuracy, therefore, we must be very careful in selecting these parameters and also the network architecture according to the network task so that our network has the best efficiency. In this report, we also try to improve this network compared to what has been done so far by changing the architecture and parameters.

Keywords-*Deep Learning, Convolution, Super Resolution, CNN Networks, Optimizer*

I. INTRODUCTION

Before examining the details of this study, we will first review the basic concepts of super resolution and explain the generalities and implementation methods. Super resolution is responsible for the task of taking low resolution (LR) input and upgrading its level to high resolution. In Figure 1 an example of the task for super resolution is shown [1].

Figure 1. An example of the task for super resolution [1]

Among the applications of super resolution, we can mention [3]:

1- Crime detection

2- Producing high resolution films from old films with low quality and resolution

3- Data recovery in-text images

and ...

Figure 2 shows an example of the effect of this method in producing high resolution images.

High-resolution image reconstruction is of great practical value in many areas. This, in turn, necessitates the analysis of the extraordinary reconstruction method for single-character images.

In this regard, the extraordinary reconstruction method for single-frame images, considering the learning of multilevel neural networks, was proposed by Liu et al.

Figure 2. An example of the application of super resolution [4]

This method extends a PMJ model for super-resolution reconstruction while extracting the main feature on the image during the measurement phase. However, the problem with this method is that it cannot eliminate noise in a single frame image and therefore its anti-interference performance is poor [5].

In a study by Peng et al., A method for high-resolution single-frame reconstruction based on non-negative local embedding and non-local regularization is proposed. The method is that in the reconstruction phase, a non-negative local embedding is used to select the number of neighbors. Finally, the non-local similarity of the image is used to construct ordinary non-local terms to change the result of the reconstruction. Of course, the process of this method is complex and easily leads to error generation and reduced image resolution [6].

However, in this method, the errors caused by obtaining images with super-resolution by cameras under the connection of visible light are ignored [7].

The support vector regression (SVR) method for creating single-frame image resolution using super-resolution reconstruction is proposed by Yuan et al. The basis of the work is the use of raster scanning to scan educational images with high and low resolution and leads to the extraction of input vectors and tag pixels from blocks, respectively.

The SVR technique is used to ignore the pixels of the super-resolution image block to complete the super-resolution image reconstruction. However, the disadvantage of this method is that it cannot remove Gaussian noise in a single-frame character image [8].

Dong et al. Developed a CNN, a complete map between low-resolution and high-resolution images, and proposed a model called SRCNN to reconstruct a single-image SR [9].

A three-layer convolution network is used in the SRCNN, and a pre-convolution sampling layer is set by HR to amplify the LR image. The SRCNN architecture is shown in Figure 3.

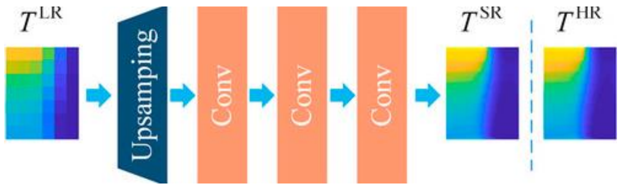

Figure 3. SRCNN architecture [10]

Dong et al. were the first researchers to use the DL method in the SR image. Thus, the use of DL-based SR technology made computer vision very attractive, and based on this, similar massive SR models such as FSRCNN, VDSR, and Lap SRN were designed [11-13].

Even the growth of DL technology was used as a technique for DL-based SR reconstruction in fluid mechanics, in which Fukami et al. first used DL-based SR technology in fluid mechanics [14].

A hybrid multi-scale perfusion model (DSC/MS) has been proposed for the reconstruction of a two-dimensional isotropic turbulence flow field in operation. Deng et al. proposed two hostile network-based models of SRGAN and ESRGAN to increase the spatial resolution of the complex awakening flow behind the cylinders [15].

A new model of MTPC has also been proposed by Yu et al. and used for forced isotropic turbulence [16].

In general, a lot of research has been done in this field, some of which we mentioned in this part of the study.

II. Analysis of an Example of an Implemented Method

In the following, we will examine each of the implementation methods mentioned in Figure 4.

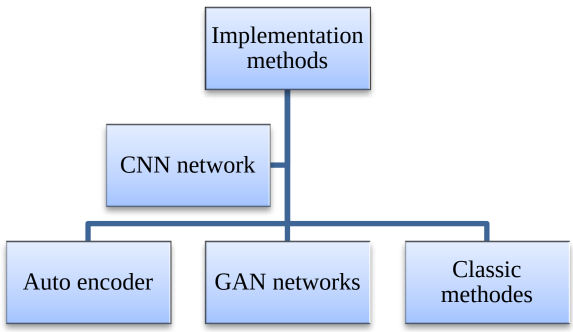

Figure 4. Implementation methods

Of course, we avoid resorting to classical methods because they are obsolete today and have been replaced by deep learning methods.

A. CNN networks

Figure 5 shows the architecture of using this network that we use the convolution layer and introduce the low resolution image as input to the network and receive the high resolution image as output.

B. Autoencoder

The autoencoder architecture for the super resolution task can be seen in Figure 6, which consists of two layers of decoder and encoder, the convolution operation is performed in one which and the decoder operation in the other layer.

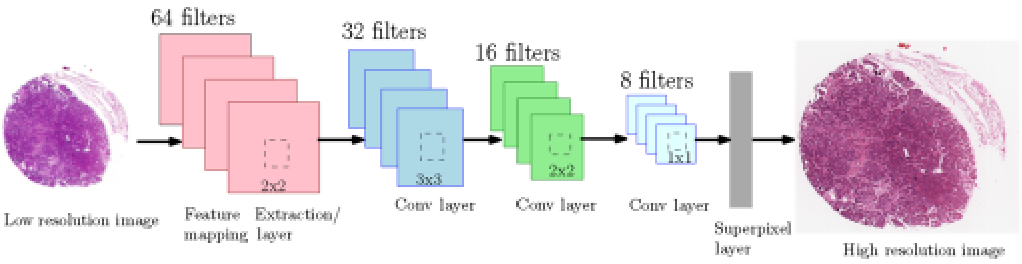

Figure 5. CNN network architecture for super resolution [17]

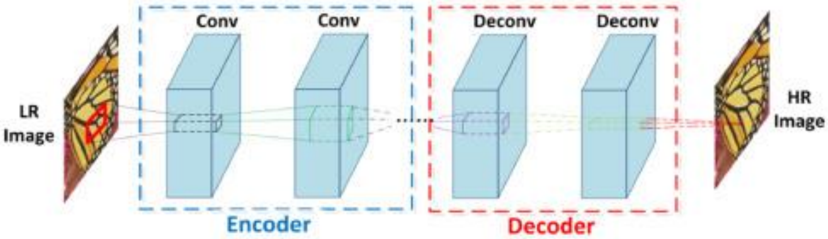

Figure 6. An example of an autoencoder architecture for the super resolution task [18]

C. GAN networks

In the figure below, we see the architecture of the GAN network for super resolution. As can be seen in Figure 7, this network consists of two production networks and two differentiation networks that, among two production networks, one is responsible for producing low-resolution images from high-resolution images, and the other performs the opposite, and the two differentiation networks are responsible for the detection of fake images.

There are various datasets in the field of super resolution, some of which are shown in Table 1 with some relevant features.

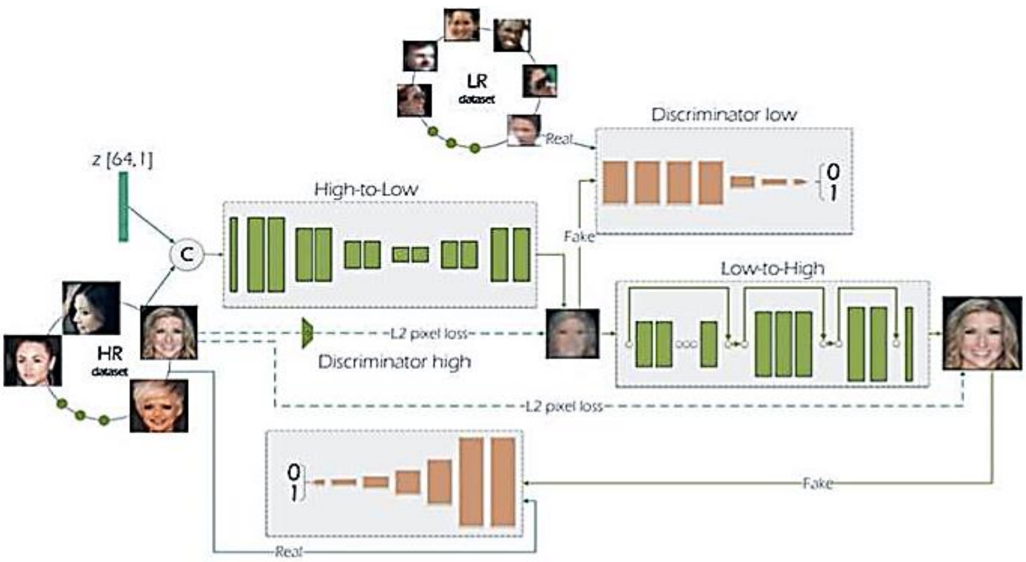

Figure 7. An example of a GAN network architecture for super resolution [19]

TABLE I. AN EXAMPLE OF THE DATASETS USED FOR SUPER RESOLUTION [20]

Name	Number of images/pairs	Image format	Type	Resolution	Details of images
BSD100[21]	100	PNG	Unpaired	(480, 320)	100 images of animals, people, buildings, scenic views etc.
BSDS300[21]	300	JPG	Unpaired	(430, 370)	300 images of animals, people, buildings, scenic views, plants, etc.
BSDS500[22]	500	JPG	Unpaired	(430, 370)	Extended version of BSD 300 with additional 200 images
CelebA[23]	202599	PNG	Unpaired	(2048, 1024)	Over 40 attribute defined categories of celebrities
DIV2K[24]	1000	PNG	Paired	(2048, 1024)	Objects, People, Animals, scenery, nature
Manga109[25]	109	PNG	Unpaired	(800, 1150)	109 manga volumes drawn by professional manga artists in Japan
Urban100[26]	100	PNG	Unpaired	(1000, 800)	Urban buildings, architecture

Usually, some datasets in this field use coupled images of a low resolution image in which they are present and, according to the training we provide to the network, they create high resolution images from low resolution images [27].

D. CNN model

The basis of the CNN model presented in this paper is the use of an efficient subpixel link neural network. In this way, first, a series of convolutional layers are used to produce low-resolution feature vectors and then in the final layer it is transformed into a high-resolution image. The equations governing the input behaviour and hidden layers are given below [28]

$$f^1(I^{LR}; W_1, b_1) = \phi(W_1 * I^{LR} + b_1)$$

$$f^L(I^{LR}; W_{1:L}, b_{1:L}) = \phi(W_L * f^{L-1} I^{LR} + b_L)$$

$$PS(T)_{x,y,c} = T_{\lfloor x/r \rfloor, \lfloor y/r \rfloor}, C.r.mod(y,r) + C.r.mod(x,r) + c$$

where L represents the number of network layers, I^{LR} is the input image of LR, and wL, bL and I are the learnable weights of the network. Figure 8 shows a diagram of this architecture.

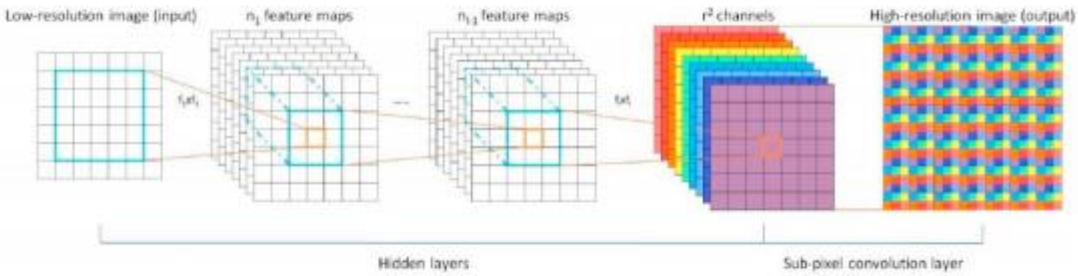

Figure 8. Architecture of CNN network

An example of the architecture used is depicted in Figure 9. In a sample code used to process high and low resolution images, we process the image data. The basis of the work is that first the considered images are given from the RGB color space to the YUV color space. The image is then cropped for the input data (low resolution images). After recovering the y channel, its size is changed by the area method. Of course, since humans are more sensitive to changes in radiance, only the radiance channel in the YUV color space is considered. For the target data (high resolution images), only the image is cut and the y channel is retrieved [30].

In Figure 10 we can see how the network works on a series of data sets.

To evaluate the image quality, the PSNR criterion has been used, which works by comparing the signal-to-noise ratio, the signal power level with the noise power level, and is usually expressed in decibels. The higher signal-to-noise ratio represents the better characteristic for a system; because more useful information is received in the form of a signal, than unwanted information or noise [32].

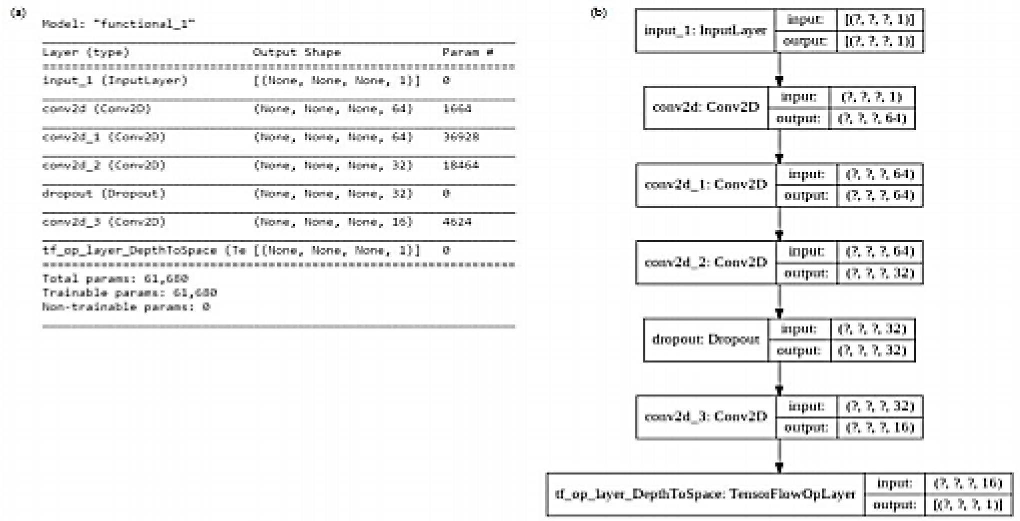

Figure 9. An example of the architecture used [29]

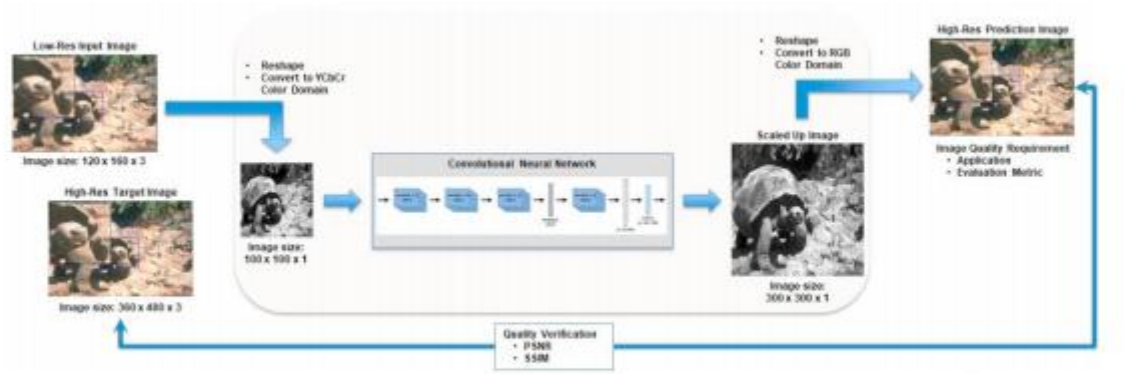

Figure 10. How to implement a network on an image [31]

For example, when a sound signal has a signal-to-noise ratio of 100 decibels, it means that the sound signal level is 100 times higher than the noise signal level in that system. This is why the signal-to-noise ratio of 100 is a better characteristic than the SNR of 70, and the computational formulas for this criterion are as follows [33].

$$PSNR = 10.log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20.log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$
$$= 20.log_{10}(MAX_I) - 10.log_{10}(MSE)$$

SSIM is also a structural comparison criterion for two images based on the structure of natural images. The structure of natural images is such that pixels are highly dependent on adjacent pixels, and this dependence contains important information about the structure of objects in the image. By calculating SSIM, the structural similarity in the neighbourhood of each pixel is calculated separately and its relationships are as follows: [34]

$$SSIM(I,\hat{I}) = \left(\frac{2\mu_I\mu_{\hat{I}}+C_1}{\mu_I^2+\mu_{\hat{I}}^2+C_1}\right)^{\alpha} \cdot \left(\frac{2\sigma_I\sigma_{\hat{I}}+C_2}{\sigma_I^2+\sigma_{\hat{I}}^2+C_2}\right)^{\beta} \cdot \left(\frac{\sigma_{I\hat{I}}+C_3}{\sigma_I\sigma_{\hat{I}}+C_3}\right)^{\gamma}$$

where μ_I and σ_I are the mean and standard deviation of the original image, respectively, and $\mu_{\hat{I}}$ and $\sigma_{\hat{I}}$ are the mean and standard deviation of the compressed image. $\sigma_{I\hat{I}}$ shows the correlation between the two images $I, \hat{I}$, and parameters α, β, γ are the weight of each function.

Finally, the architecture of Figure 11 has been used.

As can be seen, five CONV layers and RELU activity functions have been used. The paading same is also used, a summary of this architecture can be observed in Figure 12.

```python
def get_model(upscale_factor=3, channels=1):
    conv_args = {
        "activation": "relu",
        "kernel_initializer": "Orthogonal",
        "padding": "same",
    }
    inputs = keras.Input(shape=(None, None, channels))
    x = layers.Conv2D(64, 5, **conv_args)(inputs)
    x = layers.Conv2D(64, 3, **conv_args)(x)
    x = layers.Conv2D(32, 3, **conv_args)(x)
    x = layers.Conv2D(channels * (upscale_factor ** 2), 3, **conv_args)(x)
    outputs = tf.nn.depth_to_space(x, upscale_factor)

    return keras.Model(inputs, outputs)
```

Figure 11. A summary of the architecture

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None, None, 1)]   0
_________________________________________________________________
conv2d (Conv2D)              (None, None, None, 64)    1664
_________________________________________________________________
conv2d_1 (Conv2D)            (None, None, None, 64)    36928
_________________________________________________________________
conv2d_2 (Conv2D)            (None, None, None, 32)    18464
_________________________________________________________________
conv2d_3 (Conv2D)            (None, None, None, 9)     2601
_________________________________________________________________
tf.nn.depth_to_space (TFOpLa (None, None, None, 1)     0
=================================================================
Total params: 59,657
Trainable params: 59,657
Non-trainable params: 0
```

Figure 12. An example of the results for an ADAM optimizer

As can be seen in Figure 10, the set of parameters produced by this network is 59657, which can be inferred that this network is a small network and the probability of over-fit is low due to the production parameters of this network. This model is trained using an ADAM optimizer in 100 epochs with a learning rate of 0.001, which reaches the error rate of 0.0025, which is an example of the results shown in Figure 13.

III. UPGRADING THE REVIEWS

In the next step, we try to achieve more accurate accuracy with changes in network architecture and model parameters. In the continuation of this article, we will examine these changes, which are as follows:

1- Changing epoch for network training

2- Changing the optimizers and examining their effect in reducing network error

3- Changing the number of convolutional layers used in network architecture

4- Changing activity functions in network architecture in order to investigate their impact on network accuracy

The review of these materials is done in the following and we will specifically examine each of them and their effect on reducing network error.

A. Epoch

In this section, we try to change the number of epochs to see the variations of the network accuracy, which can be seen in Table 2 and Figures 13 and 14.

As we can see, with increasing the number of epochs, we obtain less error, and according to Table 2 and Figure 13, with 150 epochs of network training, the error we obtained has improved compared to the main code. Of course, this does not mean that the network error always decreases with increasing the number of epochs, sometimes our network may become overfit due to the increase in the number of epochs.

TABLE II. IMPACT OF EPOCHS ON ERROR AND PSNR OF NETWORK

Epochs	loss	psnr
50	0.0026	26.91
75	0.0025	27.02
100	0.0025	27.14
150	0.0022	27.35
200	0.0024	27.19

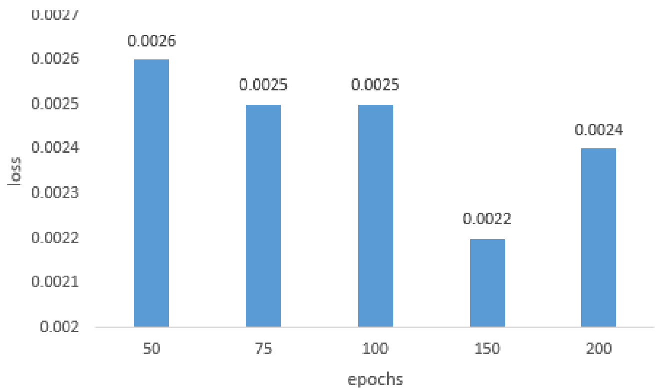

Figure 13. Impact of epochs on error of network

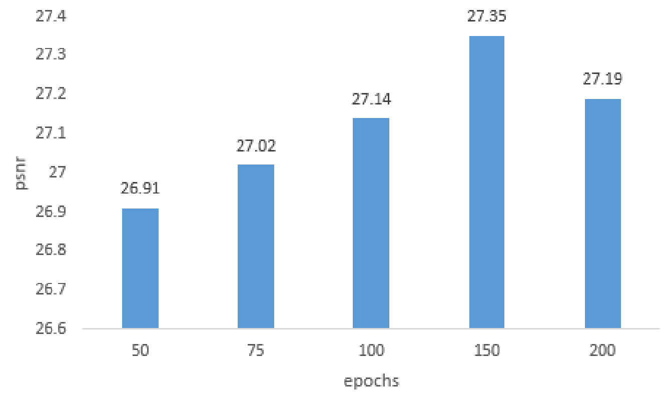

Figure 14. Impact of epochs in psnr of network

B. *Optimizer*

In this section, we will examine the effect of different optimizers on network accuracy. For this purpose, we used a series of optimizers in the colab environment, the details of which can be seen in Table 3 and Figures 15 and 16 (in each case, the network is trained in 100 epochs).

TABLE III. IMPACT OF ALL TYPES OF OPTIMIZERS ON NETWORK ERROR

optimizer	loss	psnr
Adam	0.0025	27.14
SGD	0.0616	12.33
RMS prop	0.0023	28.16
Ftrl	0.0569	12.33
Ada delta	0.0876	10.75

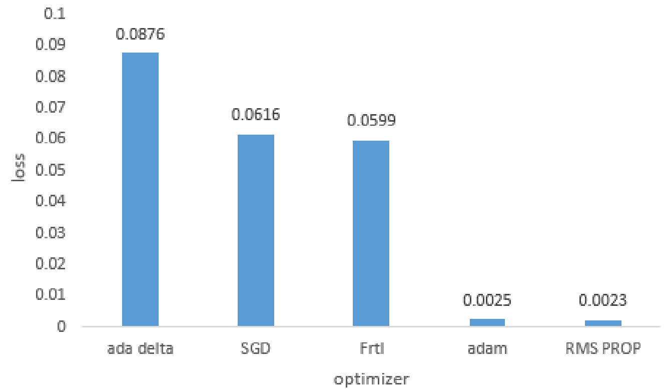

Figure 15. Impact of all types of optimizers on network error

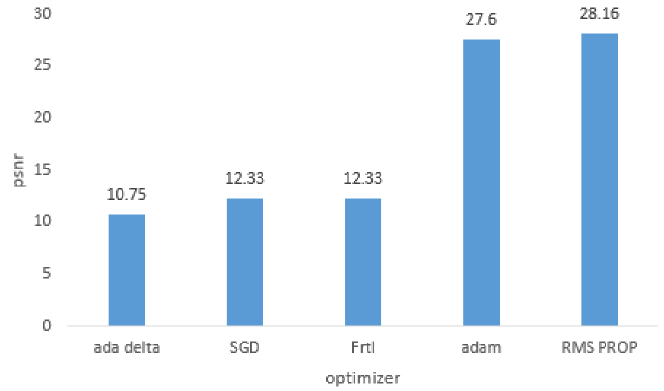

Figure 16. Impact of all types of optimizers on psnr

Based on the observations of this section, only Adam and RMS prop optimizers have acceptable performance and improve network performance. It was also found that using the RMS prop optimizer resulted in less error than the Adam optimizer, which improved network performance.

In general, it can be said that the effect of optimizers in reducing errors cannot be hidden and choosing an appropriate optimizer according to the network task is very important and necessary because it has a significant impact on network performance.

C. *Change of the number of CONV*

In this section, we intend to make a series of changes in the network architecture and first change the number of convolutions, the results of which can be seen in Table 4 and Figures 17 and 18 (During the training, we trained the network in 100 epochs and also used the Adam optimizer).

According to the observations made in Table 4 and Figures 17 and 18, it is concluded that changing the convolution size does not have much effect on the network error.

In the previous study, the main code used 64 convolutions in each layer, but we were able to improve the error rate to some extent by using 128 convolutions in each layer, which can be seen in Table 4 and Figures 17 and 18. As we all know, reducing errors by as much as one-millionth of a percentage point affects the performance of our network and we should not neglect it.

TABLE IV. IMPACT OF NUMBER OF CONVULSIONS ON NETWORK ERROR

Number of convulsions	loss	psnr
32	0.0025	27.14
64	0.0025	27.09
128	0.0024	27.17
256	0.0025	27.13

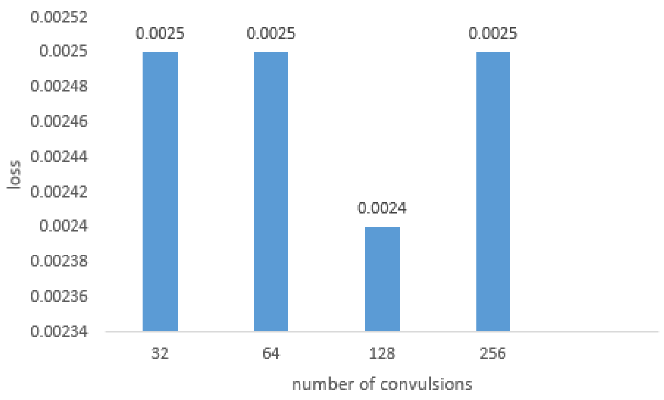

Figure 17. Impact of number of convulsions on network error

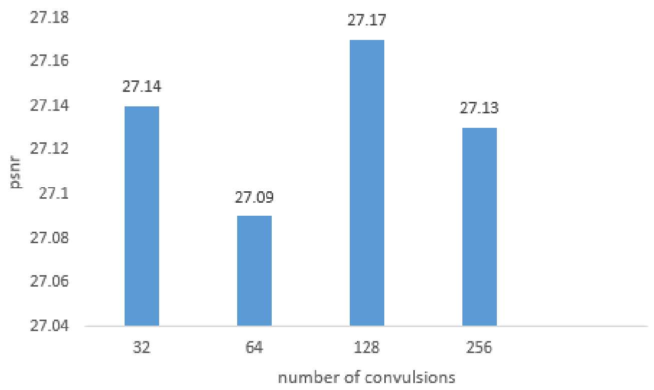

Figure 18. Impact of number of convulsions on psnr

D. *Change of the activity function*

In this section, we use different activity functions to understand their effect on improving network performance (we trained the network in 100 epochs)

According to the observations in Table 5 and Figures 19 and 20, the activity function also has an important effect on network accuracy. In the previous examination, the relu activity function was used in the original code, but we were able to reduce the network error rate from 0.0025 to 0.0024 using the elu activity function, which improves network performance.

TABLE V. IMPACT OF ACTIVITY FUNCTION ON NETWORK ERROR

Activation function	loss	psnr
relu	0.0025	27.14
elu	0.0024	27.19
selu	0.0028	25.86
sigmoid	0.0025	27.09
tanh	0.0027	24.56

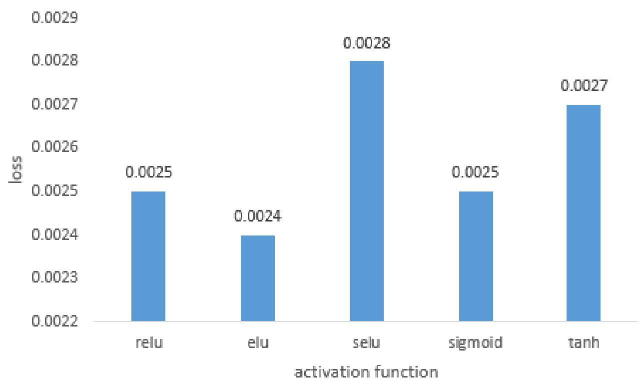

Figure 19. Impact of activity function on network error

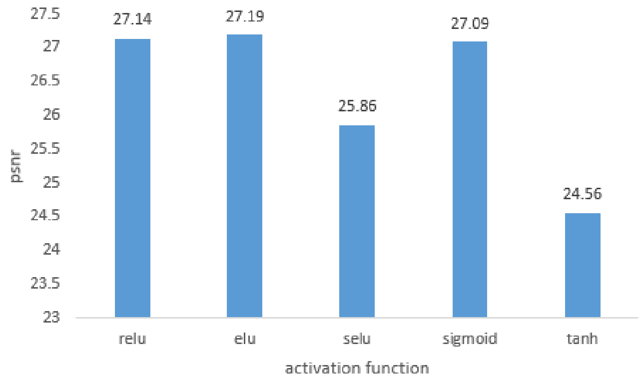

Figure 20. Impact of activity function on psnr

IV. CONCLUSION

In this paper, we examined the issue of super resolution, which is of great importance today and is used in various fields. The results showed that;

1. We can use different methods to implement super resolution, which today, deep learning methods have been considered due to their unique features and we saw that one of the deep learning methods to implement super resolution is to use a convolutional method that provide us good accuracy.

2. We also found that the smallest changes in network architecture and parameters have a large impact on network accuracy, therefore, we must be very careful in selecting these parameters and also the network architecture according to the network task so that our network has the best efficiency. In this report, we also tried to improve this network compared to what has been done so far by changing the architecture and parameters.

3. In this study, we tried to increase network accuracy and reduce its error by changing the parameters and network architecture. In our network, the error was 0.0025, which we were able to reduce with the changes we made in the network, which is very important and effective in the network output in the deep learning network.

4. In this study, we were able to reduce the network error rate by applying a series of changes such as using the RMS PROP optimizer, network training in 150 epochs, using the elu activity function and changing the number of convolutional layers of the designed network. This leads to better image recognition, which in turn has a wide range of applications in various fields.

References

[1] T. Peleg and M. Elad, "A statistical prediction model based on sparse representations for single image super-resolution", IEEE transactions on image processing, 23(6), pp. 2569-2582, 2014.

[2] C. Peyrard, M. Baccouche, F. Mamalet and C. Garcia, "ICDAR2015 competition on text image super-resolution", In 2015 13th International Conference on Document Analysis and Recognition (ICDAR), pp. 1201-1205.

[3] M. Rajnoha, A. Mezina, R. Burget, "Multi-frame labeled faces database: Towards face super-resolution from realistic video sequences", Applied Sciences, 10(20), 7213, 2020.

[4] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network", In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4681-4690, 2017.

[5] N. Liu, C. H. Li, "Single image super-resolution reconstruction via deep convolutional neural network", China Sciencepaper, 10, pp. 201-206, 2015.

[6] Y. P. Peng, B. J. Ning, X. B. Gao, "Single-frame image super-resolution reconstruction algorithm based on nonnegative neighbor embedding and non-local means regularization", Computer Science, 42, pp. 104-107, 2015.

[7] C. W. Chow, R. J. Shiu, Y. C. Liu, Y. Liu, C. H. Yeh, "Non-flickering 100 m RGB visible light communication transmission based on a CMOS image sensor", Opt Express, 26, pp. 70-79, 2018.

[8] Q. P. Yuan, H. J. Lin, Z. H. Chen, "Single image super-resolution reconstruction using support vector regression", Opt Precis Eng, 24, pp. 2302-2309, 2016.

[9] C. Dong, C. C. Loy, K. He, X. Tang, "Learning a deep convolutional network for image super-resolution", European Conference on Computer Vision, pp. 184-199, 2014.

[10] C. Kong, J. T. Chang, Y. F. Li, R. Y. Chen, "Deep learning methods for super-resolution reconstruction of temperature fields in a supersonic combustor", AIP Advances, 10(11), pp. 115021, 2020.

[11] C. Dong, C. C. Loy, X. Tang, "Accelerating the super-resolution convolutional neural network", European Conference on Computer Vision, pp. 391-407, 2016.

[12] J. Kim, J. K. Lee, K. M. Lee, "Accurate image super-resolution using very deep convolutional networks", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2016.

[13] W. S. Lai, J. B. Huang, N. Ahuja, M. H. Yang, "Deep Laplacian pyramid networks for fast and accurate super-resolution", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[14] K. Fukami, K. Taira, "Super-resolution reconstruction of turbulent flows with machine learning", J. Fluid Mech, 870(106), https://doi.org/10.1017/jfm.2019.238, 2019.

[15] Z. Deng, C. He, Y. Liu, K. C. Kim, "Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework", Phys. Fluids, 31, 125111, https://doi.org/10.1063/1.5127031, 2019.

[16] B. Liu, J. Tang, H. Huang, X. Y. Lu, "Deep learning methods for super-resolution reconstruction of turbulent flows", Phys. Fluids, 32, 025105, https://doi.org/10.1063/1.5140772.

[17] L. Mukherjee, H. D. Bui, A. Keikhosravi, A. Loeffler, K. W. Eliceiri, "Super-resolution recurrent convolutional neural networks for learning with multi-resolution whole slide images", Journal of biomedical optics, 24(12), 126003, 2019.

[18] H. Liu, J. Han, S. Hou, L. Shao, Y. Ruan, "Single image super-resolution using a deep encoder–decoder symmetrical network with iterative back projection", Neurocomputing, 282, pp. 52-59, 2018.

[19] A. Bulat, J. Yang, G. Tzimiropoulos, "To learn image super-resolution, use a gan to learn how to do image degradation first", In Proceedings of the European conference on computer vision (ECCV), pp. 185-200, 2018.

[20] S. M. A. Bashir, Y. Wang, M. Khan and Y. Niu, "A comprehensive review of deep learning-based single image super-resolution", Peer J Computer Science, 7, e621, 2021.

[21] D. Martin, C. Fowlkes, D. Tal, J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics", In Proceedings Eighth IEEE International Conference on Computer Vision, ICCV 2001, IEEE, 2, pp. 416-423, 2001.

[22] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, "Contour detection and hierarchical image segmentation", IEEE transactions on pattern analysis and machine intelligence, 33(5), pp. 898-916, 2010.

[23] Z. Liu, P. Luo, X. Wang, X. Tang, "Deep learning face attributes in the wild", In Proceedings of the IEEE international conference on computer vision, pp. 3730-3738, 2010.

[24] E. Agustsson, R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study", In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 126-135, 2017.

[25] A. Fujimoto, T. Ogawa, K. Yamamoto, Y. Matsui, T. Yamasaki, K. Aizawa, "Manga109 dataset and creation of metadata", In Proceedings of the 1st international workshop on comics analysis, processing and understanding, pp. 1-5, 2016.

[26] J. B. Huang, A. Singh, N. Ahuja, "Single image super-resolution from transformed self-exemplars", In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5197-5206, 2015.

[27] D. Engin, A. Genç, H. Kemal Ekenel, "Cycle-dehaze: Enhanced cyclegan for single image dehazing", In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 825-833, 2018.

[28] W. Shi, J. Caballero, F. Huszár, T. Johannes, A. P. Aitken, R. Bishop, D. Rueckert, Z. Wang, "Real-Time Single Image and Video Super-Resolution Using and Efficient Sub-Pixel Convolutional Neural Network", Cornell University Computer Science, Computer Vision [cs.CV] Archive; arXiv:1609.05158v2, 2016.

[29] K. K. Lee, A. Ashok, "Surpassing Rayleigh limit: Fisher information analysis of partially coherent source(s)", Optics and Photonics for Information Processing XIII, 11136, 2019.

[30] B. B. Lee, P. R. Martin, A. Valberg, "Sensitivity of macaque retinal ganglion cells to chromatic and luminance flicker", The Journal of Physiology, 414, pp. 223-243, 1989.

[31] W. Symolon, C. Dagli, "Single-Image Super Resolution Using Convolutional Neural Network", Procedia Computer Science, 185, pp. 213-222, 2021.

[32] Q. Huynh-Thu, M. Ghanbari, "Scope of validity of PSNR in image/video quality assessment", Electronics Letters, 44(13), pp. 800-801, 2008.

[33] M. Mendoza, "CubeSat Design Specification (CDS)", Rev. 13, The CubeSat Program, Cal Poly SLO. https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b623370,13b6c063a655a/1458157095454/cds_rev13_final2.pdf; accessed on 14 June 2019.

[34] D. Bradley, L. Brandon, "Orbital Edge Computing: Machine Inference in Space", IEEE Computer Architecture Letters, 18(1), pp. 59-62, 2019.

How to Cite this Article:

Faraji, M. & Hasan-Zadeh, A. (2022). Application of Deep Learning Methods to Implement Super Resolution. International Journal of Science and Engineering Investigations (IJSEI), 11(120), 1-10. http://www.ijsei.com/papers/ijsei-1112022-01.pdf