Letters in Information Technology Education (LITE)
Vol 5, No 2, 2022, pp. 51-54
e-ISSN: 2654-5667

Reconstruction Old Students Image Using The Autoencoder Method

Candra Putra Negara [1],*, Azmi Badhi'uz Zaman [2], Dimas Aji Setiawan [3], Ahmad Azhari [4]
Departement of Informatics, Universitas Ahmad Dahlan, Yogyakarta Indonesia
[1]mthesika@gmail.com, [2]azmibz23@gmail.com, [3]dimas.aji997@gmail.com, [4]ahmad.azhari@tif.uad.ac.id*
*Corresponding author

Article Info

Article history:
Received Jun 21, 2022
Revised July 19, 2022
Accepted Oct 21, 2022

Keyword:
Image Procesessing
Articifial Nervous System
Adam Optimazation
Enchancing Image Resolution

ABSTRACT

Image Processing is image processing with a digital computer to produce new images according to the user's wishes. One implementation is to reconstruct the image. Through the extraction stages are able to get the characteristics of an image. The algorithm used is Adam Optimization, which is an extension of the stochastic gradient reduction that has just seen wider adoption for deep learning applications in computer vision and natural language processing. In this study using the autoencoder technique, which is one variant of artificial neural networks that are generally used to "encode" data. Autoencoder is trained to be able to produce the same output as the input. This image reconstruction aims to process an image whose quality is not very clear to be clear. This if possible can be used to detect someone's face from a distance of photos. In reconstructing this image through the encode and decode process by defining Conv2D and Maxpool, it is processed into training with epoch 100 times while for the prediction process using Keras library. Then the last one gets an accuracy of 0,022. The final result is the output of the reconstructed image and calculation graph.

I. Introduction

In this day and age images or images become important. One of them is for documentation, whether it's formal or informal documentation. Therefore the clarity of the image is needed to ensure that the image is valid. In general, images have a resolution that tends to be low, so it has a density that tends to be less dense, it can make it difficult for humans to recognize a person's face from a long distance photo [1]. So with the reconstruction of the image is able to overcome these problems [2], [3].

In the reconstruction process through several stages, starting from the incoming image and then proceed to the encoding process [4]. The encoding process means drawing one set of passwords to another set of passwords. The encoded image is continued to the bottleneck stage, here the image is narrowed down in order to refine the image [5]. However, in this process requires quite a long time because of the narrowing of the path. Data that is still in the form of a password and that has been completed through the bottleneck and then proceed with the decode process or return to the original image with finer quality [6]. At the final stage of this program also produces a Loss Calculate or calculation of accuracy on the image .

II. The Proposed Method/Algoritihm

A. *Flowchart*

This program consists of two main parts, encoder and decoder. Encoder, which is the process of reducing input to data with fewer dimensions or what is known as code, while decoding functions as the original input code [5]. Our program flowchart can be seen in Figure 1.

B. *Optimization of a replacement for stochastic gradient descent*

Adam is a replacement optimization algorithm for stochastic gradient descent for training models in deep learning that combines the best properties of the AdamGrad and RMSProp algorithm to provide a more optimal algorithm which can handle gradients that spread and have noise. Adam is relatively easy to configure where the default configuration parameters work well for most problems that occur during training .

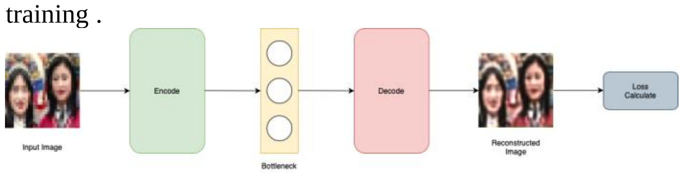


Fig. 1. Program Flowchart

C. *Adam paremeter configuration*

- Alpha: also known as learning rate or step size. Proportional weights have a value of 0.001. Larger values (eg 0.3) result in faster initial learning before the tariff is updated. Smaller values (eg 1.0E-5) slow learning to training
- Beta1: decay rate for first moment estimation (e.g. 0.9)
- Beta2: decay rate for the second moment estimate (e.g. 0.999). This value must be set close to 1.0 if it has a diffused gradient
- Epsilon: Is a very small number to prevent division by zero in implementation (e.g. 10E-8).
- Some of the default parameters used in TensorFlow and Keras recommended by the paper are as follows.
- TensorFlow: learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08.
- Keras: lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0.

III. METHOD

The application development model for the preparation of In the image reconstruction program the algorithm used is adam optimization. The previous part of Adam Automation itself has been explained, the stages of the image reconstruction process have also been briefly explained before. This section will explain the use of optimization algorithms in the image reconstruction process [7], [8].

A. *Data Acquisition*

Image reconstruction basically reshapes objects from several projected images. The image reconstruction program uses the autoencoder method, autoencoder is a Neural Network model that has the same input and output. The autoencoder studies the input data and attempts to reconstruct the input data. Data was obtained from Accurate Image Super-Resolution Using Very Deep Convolutional Networks (Seoul National University) research http://cv.snu.ac.kr/research/VDSR. The amount of data used in the dataset is 132 JPG images with the provisions of training data is 60% and testing data is 40% [9], [10].

B. *First Stage*

The image to be reconstructed is obtained from the dataset (the image is inserted in the dataset). After reading the image data, the resize becomes smaller. This resizing itself aims to reduce the performance of the program, because in later processing it uses bottlenecks (path narrowing) which can result in a process that takes a long time [11].

C. *Process Stage*

Resize is complete, enter the next process. The image is encoded (turns the image into a password), it aims to enter the bottleneck process. Bottleneck the process of narrowing the path. After the bottleneck process is complete, the password (picture) is then returned to an image (decoded) [12].

D. *Process of Adam Optimization*

After the decoding process, the data is back to drawing. After that enter into the adam optimization algorithm used for image Autoencoder. Autoencoder is used to reduce the dimensions of features (Dimensionality Reduction). If we have data that has a very high dimension (data with a very large number of features) it can be that each feature is spread over each dimension of the data so that each of the existing data looks very different. To overcome these problems we need very much data or reduce the dimensions of the data [13].

E. *Last Stage*

After going through a few steps above. The picture enters the final stage, Predicting, the completed autocoder is continued to predict or predict the image. At this stage the image will be output smoother than the image before entering or passing the steps above. In our program, after prediction there is still one stage, namely analysis. This stage displays the average accuracy and final accuracy of the above program process.

IV. RESULT AND DISCUSSION

In Image Reconstruction basically reshapes objects from several projected images. The program that we designed is image reconstruction using the Autoencoder method, Autoencoder is a Neural Network model that has the same input and output. The Autoencoder studies the input data and attempts to reconstruct the input data.

A. *Process Encode and Decode*

Making encoder and decoder models, as in the source code below we have 3 encoder layers and 4 layer decoders. In Input_img we have to adjust to the previous image shape.

We define Conv2D, and MaxPool on the encoder with ReLu activation on each of its neurons, and define Conv2D and UpSampling2D on the decoder with ReLu activation on each neuron. Then we do a compiling with Adam optimizer like in Figure 2.

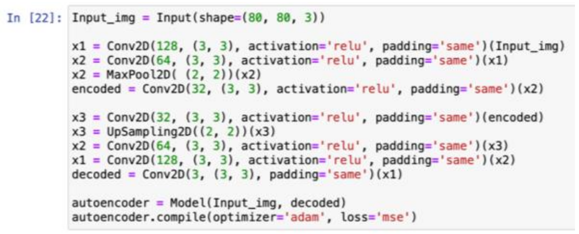
B. *Training and Predicting*

The training process with epoch 100 iterations, and batches of 128. In this process takes a long time and a large CPU usage. Predicting performs test data predictions using the Model from a hard library. For the training process can be seen in Figure 3 while for predicting can be seen in Figure 4 while the results of image reconstruction can be seen in Figure 5.

C. *Analysis*

The analysis produces a comparison graph of loss and epoch in training and testing. Besides this, the calculation is also done. Calculations performed:

Mean Loss	: 0.019340856559574605
Mean Validation Loss	: 0.017299360148608684
Mean Square Error	: 0.0022930305
Mean Absolute Error	: 0. .033474296



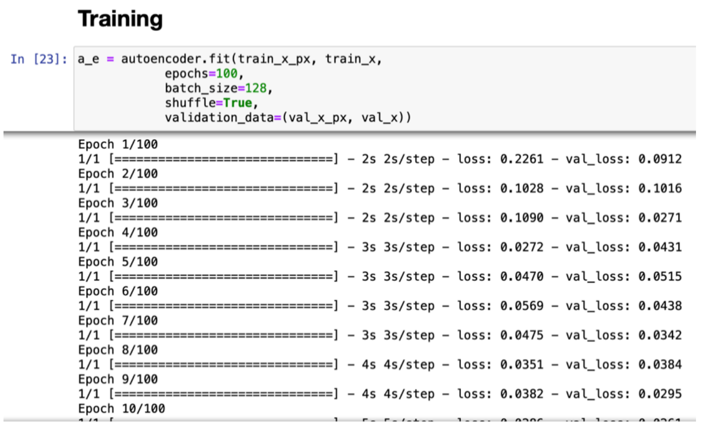
```
In [22]: Input_img = Input(shape=(80, 80, 3))

         x1 = Conv2D(128, (3, 3), activation='relu', padding='same')(Input_img)
         x2 = Conv2D(64, (3, 3), activation='relu', padding='same')(x1)
         x2 = MaxPool2D( (2, 2))(x2)
         encoded = Conv2D(32, (3, 3), activation='relu', padding='same')(x2)

         x3 = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded)
         x3 = UpSampling2D((2, 2))(x3)
         x2 = Conv2D(64, (3, 3), activation='relu', padding='same')(x3)
         x1 = Conv2D(128, (3, 3), activation='relu', padding='same')(x2)
         decoded = Conv2D(3, (3, 3), padding='same')(x1)

         autoencoder = Model(Input_img, decoded)
         autoencoder.compile(optimizer='adam', loss='mse')
```

Fig. 2.

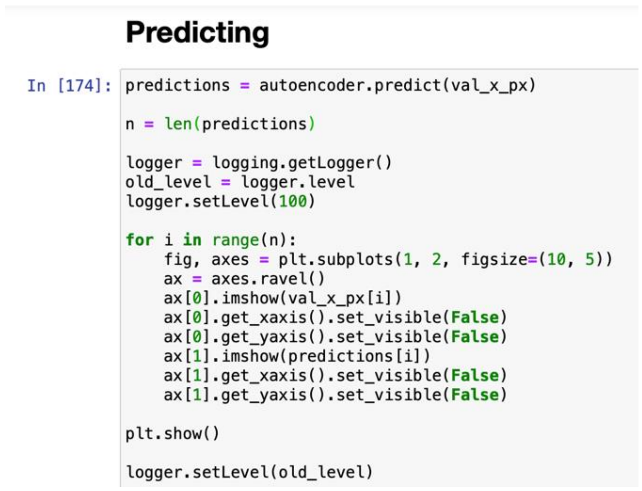


Training

```
In [23]: a_e = autoencoder.fit(train_x_px, train_x,
                       epochs=100,
                       batch_size=128,
                       shuffle=True,
                       validation_data=(val_x_px, val_x))
```

```
Epoch 1/100
1/1 [==============================] - 2s 2s/step - loss: 0.2261 - val_loss: 0.0912
Epoch 2/100
1/1 [==============================] - 2s 2s/step - loss: 0.1028 - val_loss: 0.1016
Epoch 3/100
1/1 [==============================] - 2s 2s/step - loss: 0.1090 - val_loss: 0.0271
Epoch 4/100
1/1 [==============================] - 3s 3s/step - loss: 0.0272 - val_loss: 0.0431
Epoch 5/100
1/1 [==============================] - 3s 3s/step - loss: 0.0470 - val_loss: 0.0515
Epoch 6/100
1/1 [==============================] - 3s 3s/step - loss: 0.0569 - val_loss: 0.0438
Epoch 7/100
1/1 [==============================] - 3s 3s/step - loss: 0.0475 - val_loss: 0.0342
Epoch 8/100
1/1 [==============================] - 4s 4s/step - loss: 0.0351 - val_loss: 0.0384
Epoch 9/100
1/1 [==============================] - 4s 4s/step - loss: 0.0382 - val_loss: 0.0295
Epoch 10/100
```

Fig. 3.



Predicting

```
In [174]: predictions = autoencoder.predict(val_x_px)

          n = len(predictions)

          logger = logging.getLogger()
          old_level = logger.level
          logger.setLevel(100)

          for i in range(n):
              fig, axes = plt.subplots(1, 2, figsize=(10, 5))
              ax = axes.ravel()
              ax[0].imshow(val_x_px[i])
              ax[0].get_xaxis().set_visible(False)
              ax[0].get_yaxis().set_visible(False)
              ax[1].imshow(predictions[i])
              ax[1].get_xaxis().set_visible(False)
              ax[1].get_yaxis().set_visible(False)

          plt.show()

          logger.setLevel(old_level)
```

Fig. 4.

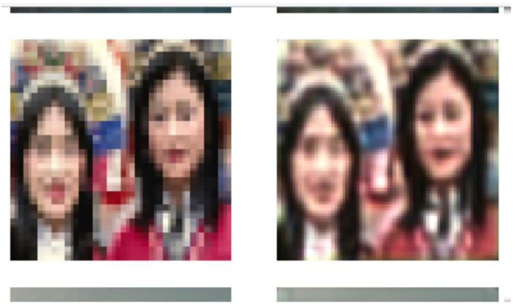
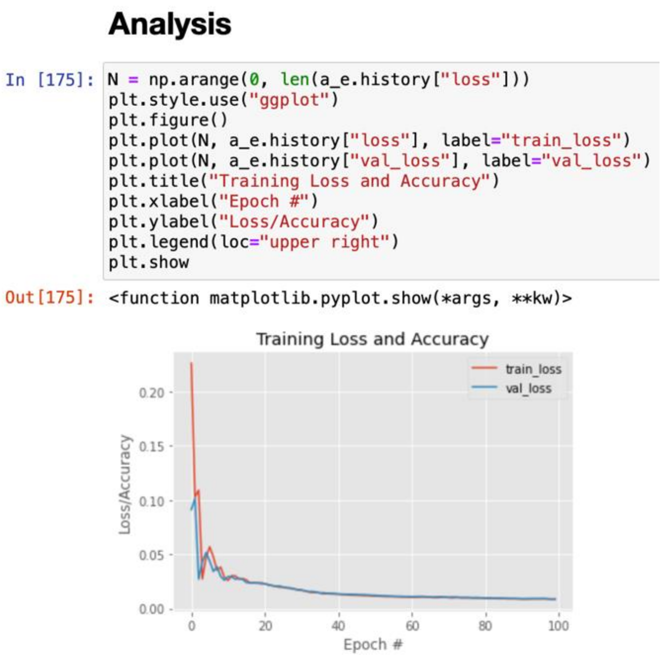


Fig. 5.

For more details, the graph can be seen in Figure 6 while the calculation results can be seen in Figure 7. According to Figure 6 it is shown that epochs 0 to 20 have accuracy which tends to be inaccurate. this underlies to do epochs more than 20 times for better results. By doing epoch 100 times, it can produce a very good mean square error value as shown in Figure 7. Good image quality can increase users' interest in using media, this has been proven in research on the use of augmented reality media [1], [14].

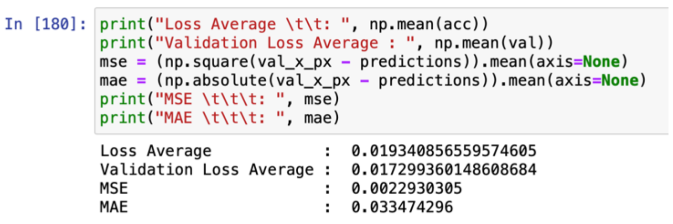


Analysis

```
In [175]: N = np.arange(0, len(a_e.history["loss"]))
          plt.style.use("ggplot")
          plt.figure()
          plt.plot(N, a_e.history["loss"], label="train_loss")
          plt.plot(N, a_e.history["val_loss"], label="val_loss")
          plt.title("Training Loss and Accuracy")
          plt.xlabel("Epoch #")
          plt.ylabel("Loss/Accuracy")
          plt.legend(loc="upper right")
          plt.show

Out[175]: <function matplotlib.pyplot.show(*args, **kw)>
```

Fig. 6.



```
In [180]: print("Loss Average \t\t: ", np.mean(acc))
          print("Validation Loss Average : ", np.mean(val))
          mse = (np.square(val_x_px - predictions)).mean(axis=None)
          mae = (np.absolute(val_x_px - predictions)).mean(axis=None)
          print("MSE \t\t\t: ", mse)
          print("MAE \t\t\t: ", mae)

          Loss Average            :  0.019340856559574605
          Validation Loss Average :  0.017299360148608684
          MSE                     :  0.0022930305
          MAE                     :  0.033474296
```

Fig. 7.

V. Conclusions

In this study it can be concluded that the reconstruction of images by image processing using the Autoencoder method is able to analyze an image and produce a new image with finer quality. By entering the greater image resolution it certainly produces more clear images. In the Autoencoder process it is expected to produce output in accordance with the input. Reconstruction of this image by inputting the image is then processed into the encode and decoded later in the training. At this stage of the training that is load data, data distribution and fitting training, after the training stage is completed then it is processed to the predicting stage, the predicting stage, namely error checking training, Encorder and decorder models and subsequently produce a prediction, After the prediction stage is completed then it is processed to the analysis stage, the analysis stage is plotting graph, calculation of mean square error, and Mean Abosulute Error. After all the stages are finished, the output can be seen in the form of a reconstructed image and get a calculation graph from Mean square Error and Mean Absultance Error.

References

[1] W. N. Hidayat, A. T. Oktaviani, T. A. Sutikno, R. K. Sari, H. Elmunsyah, and T. A. Sandy, "Camlearn as Photography Mobile Learning Application and its Effects on Academic Achievement," in *2019 International*

Conference on Electrical, Electronics and Information Engineering (ICEEIE), Oct. 2019, pp. 168–171. doi: 10.1109/ICEEIE47180.-2019.8981467.

[2] A. Dolmatova, M. Chukalina, and D. Nikolaev, "Accelerated FBP for computed tomography image reconstruction," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 3030–3034.

[3] M. G. Hu, J. F. Wang, and Y. Ge, "Super-resolution reconstruction of remote sensing images using multifractal analysis," *Sensors*, vol. 9, no. 11, pp. 8669–8683, 2009.

[4] Y. Zhang, W. Miao, Z. Lin, H. Gao, and S. Shi, "Millimeter-wave InSAR image reconstruction approach by total variation regularized matrix completion," *Remote Sens.*, vol. 10, no. 7, p. 1053, 2018.

[5] J. H. Nam and A. Velten, "Super-resolution remote imaging using time encoded remote apertures," *Appl. Sci.*, vol. 10, no. 18, p. 6458, 2020.

[6] M. Engelcke, O. P. Jones, and I. Posner, "Reconstruction bottlenecks in Object-Centric generative models," in *arXiv preprint arXiv:2007.06245*, 2020.

[7] O. Wiles, S. Ehrhardt, and A. Zisserman, "Co-attention for conditioned image matching," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 15920–15929.

[8] D. Minnen and S. Singh, "Channel-wise autoregressive entropy models for learned image compression," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 3339–3343.

[9] S. Indolia, A. K. Goswami, S. P. Mishra, and P. Asopa, "Conceptual understanding of convolutional neural network-a deep learning approach," *Procedia Comput. Sci.*, vol. 132, pp. 679–688, 2018.

[10] T. Sun, W. Fang, W. Chen, Y. Yao, F. Bi, and B. Wu, "High-resolution image inpainting based on multi-scale neural network," *Electronics*, vol. 8, no. 11, p. 1370, 2019.

[11] N. Ailon, O. Leibovitch, and V. Nair, "Sparse linear networks with a fixed butterfly structure: theory and practice," in *Uncertainty in Artificial Intelligence*, 2021, pp. 1174–1184.

[12] J. Tang, C. Huang, J. Liu, and H. Zhu, "Image super-resolution based on CNN using multilabel gene expression programming," *Appl. Sci.*, vol. 10, no. 3, p. 854, 2020.

[13] W. Lee, J. Lee, D. Kim, and B. Ham, "Learning with privileged information for efficient image super-resolution," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, 2020, pp. 465–482.

[14] W. N. Hidayat, T. A. Sutikno, H. Elmunsyah, A. Prasasti, L. F. Tumelisya, and W. M. Utomo, "User Experience Design of Augmented Reality-based Mobile Learning Media for English Subjects through User-Centered Design Approach," in *2021 7th International Conference on Education and Technology (ICET)*, Sep. 2021, pp. 171–176. doi: 10.1109/ICET53279.2021.9575121.