Letters in Information Technology Education (LITE)
Vol 6, No 2, 2023, pp. 1–4
e-ISSN: 2654-5667

Reconstruction of Old Student's Images Using The Autoencoder Method

Ahmad Azhari [a,1,*], Candra Putra Negara [a,2], Azmi Badhi'uz Zaman [a,3], Dimas Aji Setiawan [a,4]
[a] Departement of Informatics, Universitas Ahmad Dahlan, Jalan Ahmad Yani (Ringroad Selatan) Yogyakarta 55166, Indonesia
[1]ahmad.azhari@tif.uad.ac.id; [2]mthesika@gmail.com; [3]azmibz23@gmail.com; [4]dimas.aji997@gmail.com
* Corresponding author

Article Info

Article history:
Received: Dec 10, 2022
Revised: Jan 16, 2023
Accepted: Jan 31, 2023

Keyword:
Image Processing
Artificial Nervous System
Adam Optimization
Enhancing 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 can get the characteristics of an image. The algorithm used is Adam Optimization, an extension of the stochastic gradient reduction that has seen wider adoption for deep learning applications in computer vision and natural language processing. In this study, we use the autoencoder technique, one variant of artificial neural networks generally used to "encode" data. The autoencoder is trained to produce the same output as the input. This image reconstruction aims to process an image whose quality is not very clear, to be precise. This, if possible, can be used to detect someone's face from 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 result is the output of the reconstructed image and calculation graph.

I. INTRODUCTION

In this day and age, images or images become essential. 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, making it difficult for humans to recognize a person's face from a long-distance photo. So, the reconstruction of the image can overcome these problems.

In the reconstruction process, there are several stages, starting from the incoming image to the encoding process [1][2]. The encoding process means drawing one set of passwords to another set of passwords. The encoded image is continued to the bottleneck stage, where the image is narrowed down to refine the image [3]. However, 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 has been completed through the bottleneck proceed with the decoding process or return to the original image with finer quality [4][5]. This program's final stage also produces a Loss Calculation or calculation of accuracy on the image [6][7].

A. Flowchart

This program consists of two main parts: encoder and decoder. Encoder 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. Our program flowchart can be seen in Figure 1.

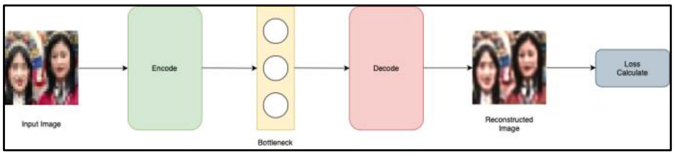


Fig. 1. The program's flowchart

1) 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 algorithms to provide a more optimal algorithm that can handle gradients that spread and have noise [8-10]. Adam is relatively easy to configure, where the default configuration parameters work well for most problems during training.

B. Adam Paremeter Configuration

- Alpha: also known as learning rate or step size. Proportional weights have a value of 0.001. Larger values (e.g., 0.3) result in faster initial learning before updating the tariff. Smaller values (e.g., 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: This is a minimal 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

II. METHODS

In the image reconstruction program, the algorithm used is Adam optimization. The previous part of Adam Automation itself has been explained, and the stages of the image reconstruction process have also been briefly described before [11]. This section will justify optimizing algorithms in the image reconstruction process.

A. Data Acquisition

Image reconstruction reshapes objects from several projected images. The image reconstruction program uses the autoencoder method, and the autoencoder is a Neural Network model with 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 provision of training data is 60% and testing data is 40%.

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 aims to reduce the program's performance because it uses bottlenecks (path narrowing) in later processing, which can result in a long process.

C. Process Stage

Resize is complete; enter the following process. The image is encoded (turns the image into a password) and aims to document the bottleneck process. The bottleneck is the process of narrowing the path. After the bottleneck process, the password (picture) is returned to an image (decoded).

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 vast 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 much data or reduce the dimensions of the data.

E. Final Stage

After going through a few steps above, the picture enters the final stage, Predicting; the completed auto coder continues to predict or predict the image. At this stage, the image will be output smoother than before entering or passing the steps above. In our program, there is still one stage: analysis after prediction. This stage displays the above program process's average and final accuracy.

III. RESULTS AND DISCUSSION

In Image Reconstruction, objects from several projected images are reshaped [12]. The program we designed is image reconstruction using the Autoencoder method, and Autoencoder is a Neural Network model with the same input and output. The Autoencoder studies the input data and attempts to reconstruct the input data.

A. Process Encode and Decode

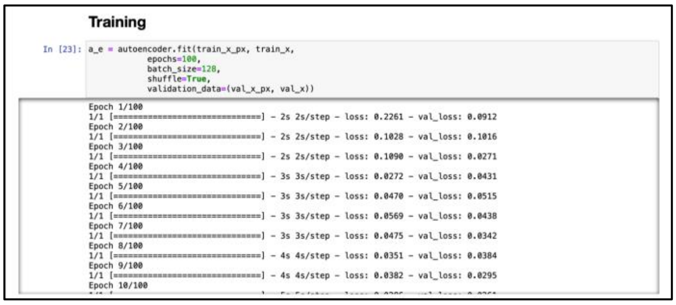
As in the source code below, we are making encoder and decoder models; we have three encoder layers and four-layer decoders. In Input_img, we have to adjust to the previous image shape. We define Conv2D and MaxPool on the encoder with ReLu activation on each of its neurons and define Conv2D and UpSampling2D on the decoder with ReLu activation on each neuron. Then we do a compiling with Adam optimizer like in Figure 2.

```
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. The Adam optimizer's code program

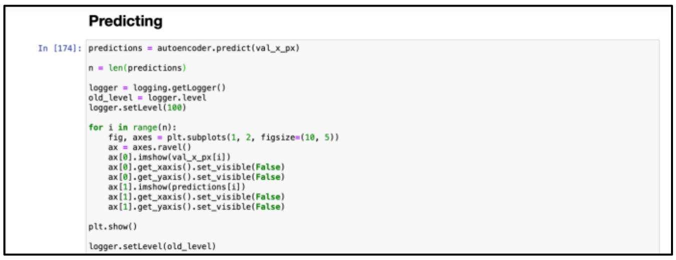
B. Training and Predicting

The training process with epoch 100 iterations and batches of 128. This process takes a long time and an extensive CPU usage. Predicting performs test data predictions using the Model from a complex library. For the training, the process can be seen in Figure 3, the prediction in Figure 4, and the image reconstruction results in Figure 5.



```
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. Training process



```
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. Predicting process

Fig. 5. Image reconstruction results

C. Analysis

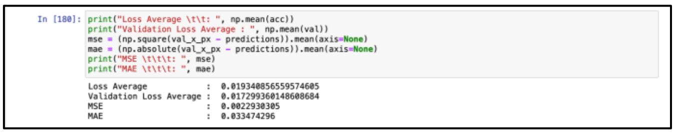
The analysis produces a comparison graph of loss and epoch in training and testing. Besides this, the calculation is also done. The performed calculation is done as shown in Table I.

TABLE I. PERFORMED CALCULATION

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

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 an excellent mean square error value, as shown in Figure 7

Fig. 6. The graph results



```
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. The calculation results.

IV. DISCUSSION

In this study, it can be concluded that the reconstruction of images by image processing using the Autoencoder method can analyze an image and produce a new image with finer quality. By entering the more excellent image resolution, it has more explicit images. The Autoencoder process is expected to produce output by 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 is load data, data distribution, and fitting training; after the training stage is completed, it is processed to the predicting stage, the predicting stage, namely error checking training, Encoder, and decoder models, and subsequently produces a prediction. After the prediction stage is completed, it is processed to the analysis stage; the analysis stage is plotting a graph and calculating the mean square error and Mean Absolute Error. After all the stages are finished, the output can be seen as a reconstructed image and a calculation graph from the Mean square Error and Mean Absultance Error.

References

[1] Reconstruction Bottlenecks in Object-Centric Generative Models, arXiv:2007.06245v1 [cs.LG] 13 Jul 2020.

[2] Accelerated Fbp For Computed Tomography Image Reconstruction, arXiv:2007.06289v1 [eess.IV] 13 Jul 2020.

[3] Super-Resolution Remote Imaging Using Time Encoded Remote Apertures, arXiv:2007.08667v1 [eess.IV] 16 Jul 2020

[4] Sparse Linear Networks With A Fixed Butterfly Structure: Theory And Practice, arXiv:2007.08864v1 [cs.LG] 17 Jul 2020

[5] D2D: Learning to find good image matching and manipulation correspondences, arXiv:2007.08480v1 [cs.CV] 16 Jul 2020

[6] CHANNEL-WISE AUTOREGRESSIVE ENTROPY MODELS FOR LEARNED IMAGE COMPRESSION, arXiv:2007.08739v1 [eess.IV] 17 Jul 2020

[7] Learning with Privileged Information for Efficient Image Super-Resolution, arXiv:2007.07524v1 [cs.CV] 15 Jul 2020

[8] Image Super-Resolution Based on CNN Using Multilabel Gene Expression Programming, Appl. Sci. 2020, 10, 854

[9] Super-Resolution Reconstruction of Remote Sensing Images Using Multifractal Analysis, Sensors 2009, 9

[10] Millimeter-Wave InSAR Image Reconstruction Approach by Total Variation Regularized Matrix Completion, Remote Sens. 2018, 10, 1053

[11] Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach, Sakshi Indolia et al. / Procedia Computer Science 132 (2018) 679–688

[12] High-Resolution Image Inpainting Based on Multi-Scale Neural Network, Electronics 2019, 8, 1370.