Final Project
Autoencoder For Image Processing

Susanta Kumar Behera (16060)

May 10, 2021

Contents

1 Final Project **2**

1.1 Project Overview . . . 2

1.1.1 Context . . . 2

1.1.2 Dataset . . . 3

1.2 Import Libraries . . . 3

1.3 Data Generator . . . 4

1.4 Preprocessing . . . 5

1.5 Autoencoder . . . 5

1.6 Variational Auto Encoder (VAE) . . . 6

1.6.1 Model Making . . . 8

1.6.2 Training Model . . . 11

1.7 Saving Model . . . 11

1.8 Analysis . . . 11

1.9 Overview . . . 14

1.10 Conclusion . . . 15

1.11 Summary . . . 16

1.12 Source And Acknowledgements . . . 16

1.13 Code . . . 16

Chapter 1

Final Project

1.1 Project Overview

1.1.1 Context

- Now a days autoencoders is polpular in various field. But, there is two applications of autoencoders which is most popular: dimensionality reduction and information retrieval.
- By using these features we can perform various image processing.
- This project has four method of image processing
  - Generate Image
  - Noise Reduction
  - Constructing New Images
  - Colouring Images

1.1.2 Dataset

- The image dataset is from this link `http://vis-www.cs.umass.edu/lfw/`
- This dataset contains 13233 number of facial raw images.
- Although the data is enough for the image processing. Still we can generate more data from the dataset using 'Keras ImageDataGenerator'.

1.2 Import Libraries

This project using these libraries

- import tensorflow as tf
- import matplotlib.pyplot as plt
- $fromsklearn.model_{s}electionimporttrain_{t}est_{s}plit$
- import os
- import pickle
- from tensorflow.keras import Model
- from tensorflow.keras.layers import Input, Conv2D, ReLU, BatchNormalization, Flatten, Dense, Reshape, Conv2DTranspose, Activation, Lambda
- from tensorflow.keras import backend as K
- from tensorflow.keras.optimizers import Adam
- from tensorflow.keras.losses import MeanSquaredError

- import matplotlib.pyplot as plt
- import numpy as np
- from PIL import Image

1.3 Data Generator

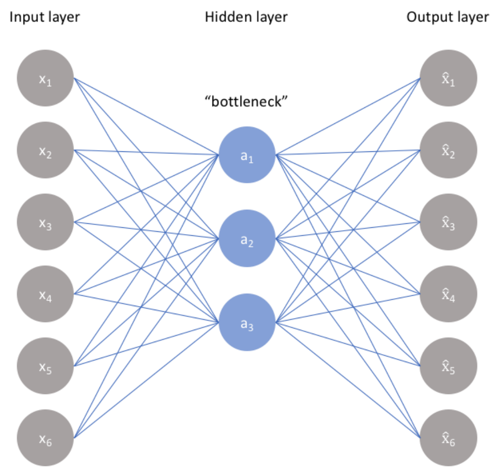
- Deep learning is only relevant when you have a huge amount of data
- In order to make the most of our few training examples, we will "augment" them via a number of random transformations, so that our model would never see twice the exact same picture. This helps prevent overfitting and helps the model generalize better.
- We can achieve this using 'keras.preprocessing.image.ImageDataGenerator' class.
- This class helps us to create images with different features. These features are acieve by changing:
  - Rotation
  - Width and Height Shift
  - Shearing Transformations
  - Zoom
  - Rescale
  - Horizontal Flip
  - Fill newly created pixels

1.4 Preprocessing

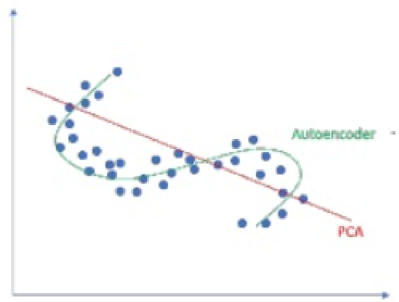
- We can't use the dataset directly inside the model.
- The dataset need to be modified according to the model's requirement.

1.5 Autoencoder

- Autoencoder is the made of encoder and decoder. Encoder create a lower dimensional representation(latent space) of the the image (compressed image). In the other hand deocder tries to get back the original image from the latent space.



- Encoder can learn non-linear relationship unlike PCA

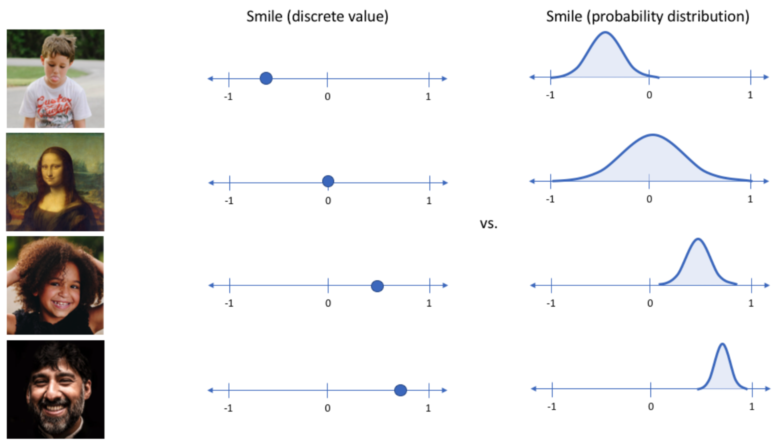


- Training is done by back propageation and reconstruction error
- reconstruction error is defined as

$$E(x, \hat{x}) = RMSE(rootmeansquareerror$$

1.6 Variational Auto Encoder (VAE)

- BY using VAE we can achieve symmetry around origin and minimize the gap between cluster of points.
- A variational autoencoder (VAE) provides a probabilistic manner for describing an observation in latent space.
- Thus, rather than building an encoder which outputs a single value to describe each latent state attribute, we'll formulate our encoder to describe a probability distribution for each latent attribute.



An example of VAE `https://www.jeremyjordan.me/variational-autoencoders/`.

- For achieving this kind of probabilistic distribution, I used multivariant normal distribution.

$$f(x_1, x_2, ..., x_k) = \frac{\exp(-\frac{1}{2}(x-\mu)^T \sum^{-} 1(x-\mu))}{\sqrt{(2\pi)^k} \mid \Sigma \mid}$$

where $\mu = mean$
$\Sigma = CovarianceMatrix$
$\epsilon = StandardnormalDistributio$

- In this model, I am using this formula for convert the latent space point to a distribution

$$Z = \mu + \Sigma\epsilon$$

where $\mu = mean$
$\Sigma = \exp(log(\frac{\sigma^2}{2}))$

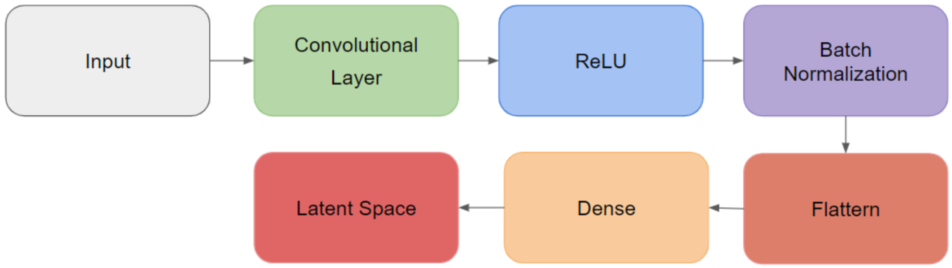
- **Loss Fuction:**

- For improving the loss function, a new term is added with RMSE(root mean square error).
-
$$loss = RMSE + KL$$
- This new term is called KL(Kullback-Leibler Divergence).
- KL is the difference between normal distribution from standard distribution.
-
$$D_{KL}(N(\mu, \sigma) \mid\mid N(0,1) = \frac{1}{2} \sum (1 + log(\sigma^2) - \mu^2 - \sigma^2$$
- Now, I am introducing a new term reconstruction loss weight which gives control over the loss function.
- This is a important parameter and finding the correct value is difficult. If the the value get lower the model loses the image features during training and if we use the higher value, the model acts as a normal AE.
- The new loss function is

$$loss = \alpha * RMSE + KL$$

1.6.1 Model Making

- For creating a autoencoding model, we have to add both encoder and decoder.
- The Encoder model is define as follows:

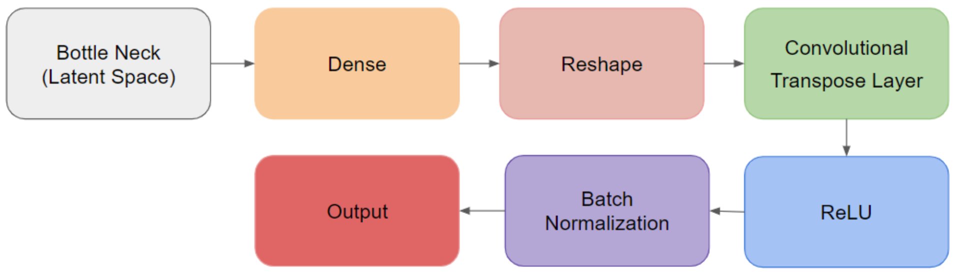


Encoder

- And the summary of the model is:

```
Model: "encoder"
____________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
====================================================================================================
encoder_input (InputLayer)      [(None, 32, 32, 3)]  0
____________________________________________________________________________________________________
encoder_conv_layer_1 (Conv2D)   (None, 32, 32, 32)   896         encoder_input[0][0]
____________________________________________________________________________________________________
encoder_relu_1 (ReLU)           (None, 32, 32, 32)   0           encoder_conv_layer_1[0][0]
____________________________________________________________________________________________________
encoder_bn_1 (BatchNormalizatio (None, 32, 32, 32)   128         encoder_relu_1[0][0]
____________________________________________________________________________________________________
encoder_conv_layer_2 (Conv2D)   (None, 16, 16, 64)   18496       encoder_bn_1[0][0]
____________________________________________________________________________________________________
encoder_relu_2 (ReLU)           (None, 16, 16, 64)   0           encoder_conv_layer_2[0][0]
____________________________________________________________________________________________________
encoder_bn_2 (BatchNormalizatio (None, 16, 16, 64)   256         encoder_relu_2[0][0]
____________________________________________________________________________________________________
encoder_conv_layer_3 (Conv2D)   (None, 8, 8, 64)     36928       encoder_bn_2[0][0]
____________________________________________________________________________________________________
encoder_relu_3 (ReLU)           (None, 8, 8, 64)     0           encoder_conv_layer_3[0][0]
____________________________________________________________________________________________________
encoder_bn_3 (BatchNormalizatio (None, 8, 8, 64)     256         encoder_relu_3[0][0]
____________________________________________________________________________________________________
encoder_conv_layer_4 (Conv2D)   (None, 8, 8, 64)     36928       encoder_bn_3[0][0]
____________________________________________________________________________________________________
encoder_relu_4 (ReLU)           (None, 8, 8, 64)     0           encoder_conv_layer_4[0][0]
____________________________________________________________________________________________________
encoder_bn_4 (BatchNormalizatio (None, 8, 8, 64)     256         encoder_relu_4[0][0]
____________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 4096)         0           encoder_bn_4[0][0]
____________________________________________________________________________________________________
mu (Dense)                      (None, 1024)         4195328     flatten_2[0][0]
____________________________________________________________________________________________________
log_variance (Dense)            (None, 1024)         4195328     flatten_2[0][0]
____________________________________________________________________________________________________
encoder_output (Lambda)         (None, 1024)         0           mu[0][0]
                                                                 log_variance[0][0]
====================================================================================================
Total params: 8,484,800
Trainable params: 8,484,352
Non-trainable params: 448
```

- The Decoder model is define as follows:



Decoder

- And the summary of the model is:

```
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
decoder_input (InputLayer)   [(None, 1024)]            0
_________________________________________________________________
decoder_dense (Dense)        (None, 4096)              4198400
_________________________________________________________________
reshape_2 (Reshape)          (None, 8, 8, 64)          0
_________________________________________________________________
decoder_conv_transpose_layer (None, 8, 8, 64)          36928
_________________________________________________________________
decoder_relu_1 (ReLU)        (None, 8, 8, 64)          0
_________________________________________________________________
decoder_bn_1 (BatchNormaliza (None, 8, 8, 64)          256
_________________________________________________________________
decoder_conv_transpose_layer (None, 16, 16, 64)        36928
_________________________________________________________________
decoder_relu_2 (ReLU)        (None, 16, 16, 64)        0
_________________________________________________________________
decoder_bn_2 (BatchNormaliza (None, 16, 16, 64)        256
_________________________________________________________________
decoder_conv_transpose_layer (None, 32, 32, 64)        36928
_________________________________________________________________
decoder_relu_3 (ReLU)        (None, 32, 32, 64)        0
_________________________________________________________________
decoder_bn_3 (BatchNormaliza (None, 32, 32, 64)        256
_________________________________________________________________
decoder_conv_transpose_layer (None, 32, 32, 3)         1731
_________________________________________________________________
sigmoid_layer (Activation)   (None, 32, 32, 3)         0
=================================================================
Total params: 4,311,683
Trainable params: 4,311,299
Non-trainable params: 384
```

- The summary of the autoencoder is:

```
Model: "autoencoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   [(None, 32, 32, 3)]       0
_________________________________________________________________
encoder (Functional)         (None, 1024)              8484800
_________________________________________________________________
decoder (Functional)         (None, 32, 32, 3)         4311683
=================================================================
Total params: 12,796,483
Trainable params: 12,795,651
Non-trainable params: 832
```

1.6.2 Training Model

- For compiling, I are using Adam optimizer and loss function described above.
- For training, I am using model.fit with validation from 'keras Model' class.
- By training with lot of images, the model able to reconstruct the patter properly.
- To avoid over-fitting, we should use discrete image data.

1.7 Saving Model

- Saving model is really important for further analysis and development.
- In this project, I am saving the model's parameter in .pkl file format and weight in .h5 file format.

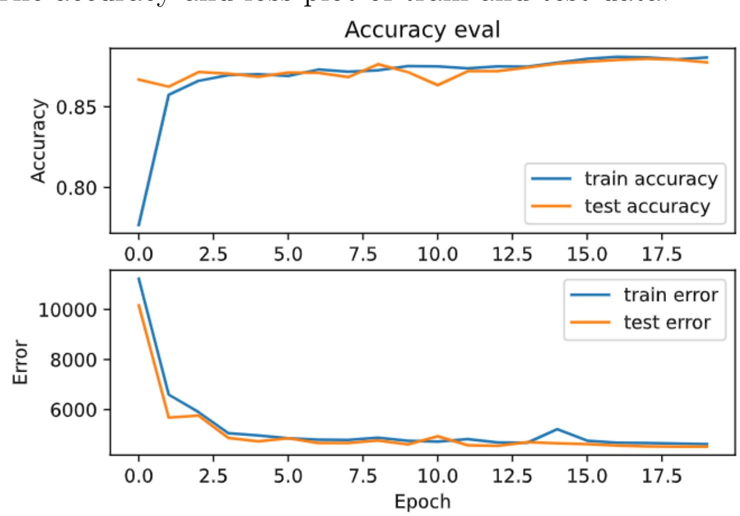
1.8 Analysis

- This project has four section:

  - Image Reconstruction
  - Noise Reduction
  - Generating New Images
  - Colouring Images

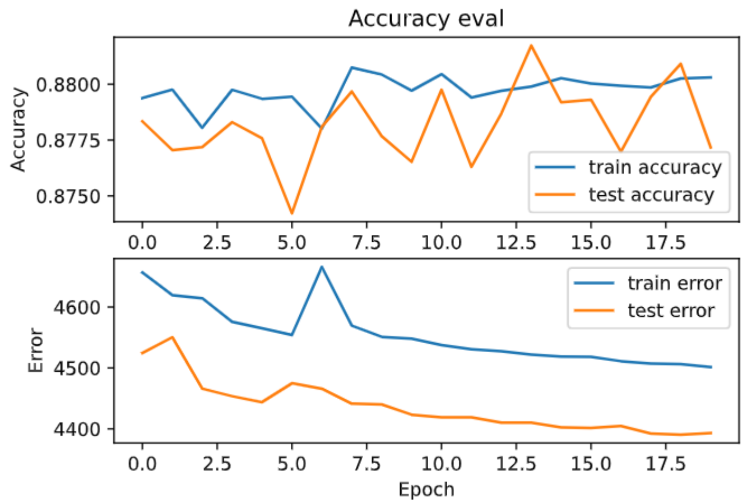
- In image reconstruction, all the images encode and decode in the autoencoder model and gives the weights of the autoencoder. By using these weights we can reconstruct any image.

- The accuracy and loss plot of train and test data:

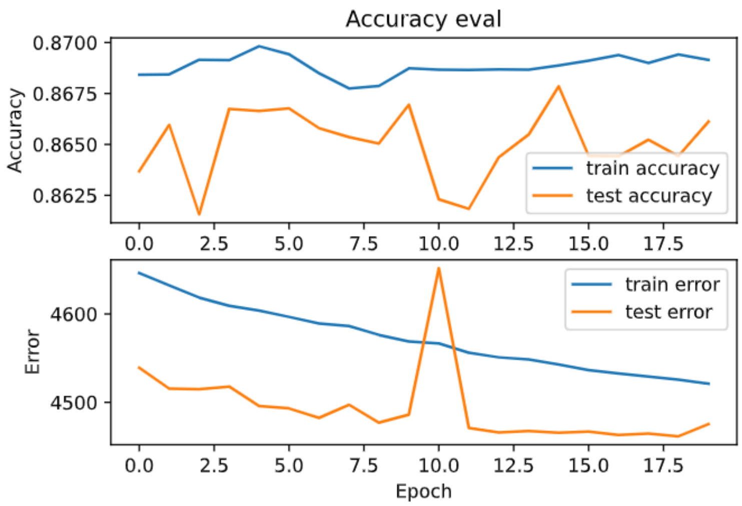


- In noise reduction, the model is trained as noise-image input and original image output. After the training, we can reduce noise of any image using this model weights.

- The accuracy and loss plot of train and test data:



- In this model the new image is constructed by adding two latent space. Because, the latent space contain all the important information and by adding these we can enhance the image data and make a new one.
- In coloring images, the model is trained as gray scale image input and original RGB image output. After the training, we can construct the color of any image using this model weights.
- The accuracy and loss plot of train and test data:

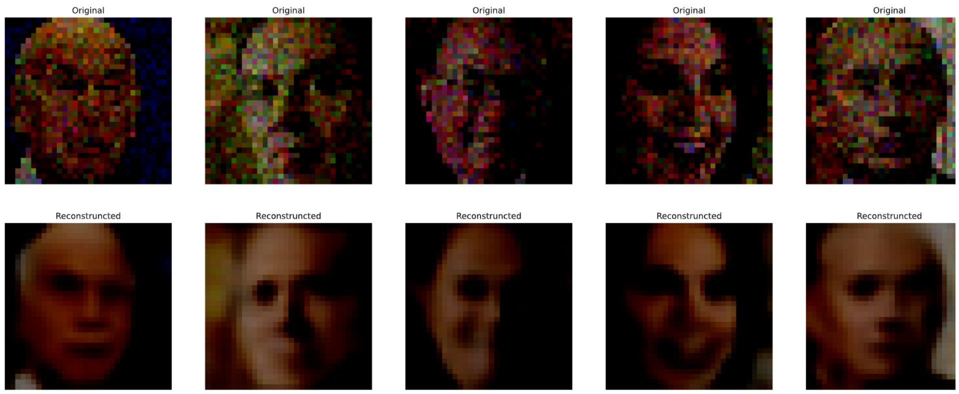


1.9 Overview

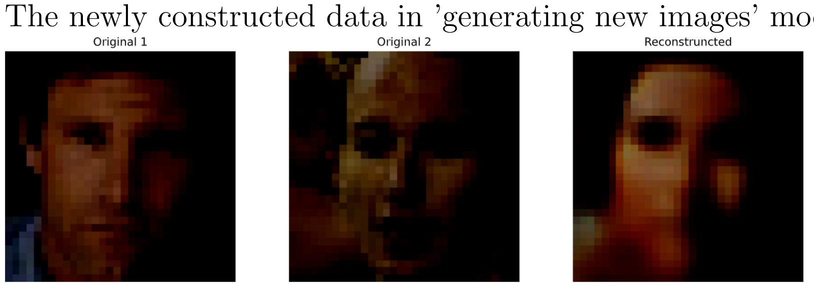
- All the model able to construct some extent of image. Despite of lower amount of data and lower training time, all the model performs really well.

- The original and reconstructed data in 'image reconstruction' model

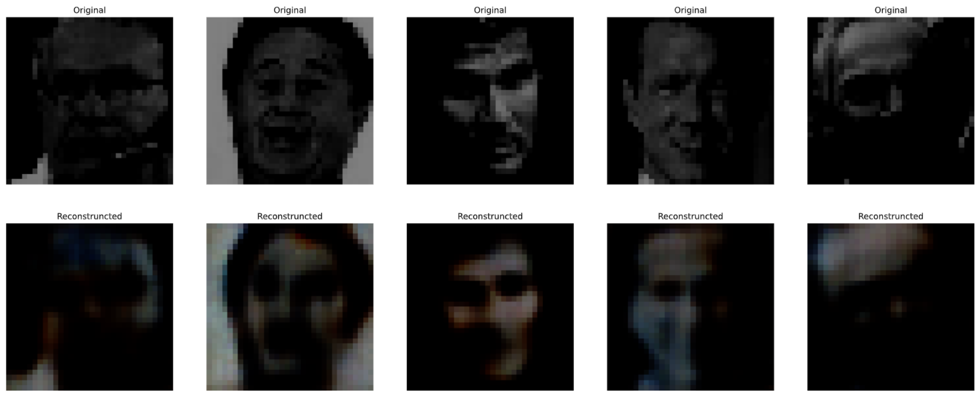
- The noise and reconstructed data in 'noise reduction' model



- The newly constructed data in 'generating new images' model



- The gray scale and reconstructed coloured images in 'colouring image' model



1.10 Conclusion

- 'Image reconstruction' shows good accuracy and lower amount of loss.
- But 'noise reduction model' and 'image colouring model' don't shows good accuracy and lower amount of loss.
- Still all the reconstructed images are not clear and lose a lot of details.
- To over come this problem we need to use higher number of data, enough dimension for the latent space for storing all the necessary details and a good value of reconstruction loss weight.

1.11 Summary

- This project was a great journey as I could implement the various tools and methods for image processing and analysis. During this project, I learnt and implemented various method of VAE, analysis and image processing.

- This project is a small demostation of the power of autoencoder in image processing. By accuring proper amount of data and tools we can able create wonder in image processing.

1.12 Source And Acknowledgements

- This project mainly based on the article `https://stackabuse.com/autoencoders-for-image-reconstruction-in-python-and-keras/` by Ali Abdelaal.

- All the image dataset is from the `http://vis-www.cs.umass.edu/lfw/lfw.tgz`

1.13 Code

- This project is made with lot of python code. The main code is the 'reecho model'.

- 'Reecho model' is available in .ipynb and .html format.

- There is also a test code as 'test model' in .py format for testing the whole project.

- The github link for the whole project is : `https://github.com/psbsoftware22/reecho.git`