TOPIC: -

IMPLEMENTATION OF IMAGE RECONSTRUCTION TECHNIQUES IN AN APPLICATION

RAMNIRANJAN JHUNJHUNWALA COLLEGE

Department of Information Technology

Ghatkopar (West), Mumbai - 86

2021-2022

Assessment on

Implementation of image reconstruction techniques in an application

In partial fulfillment of M.Sc. (INFORMATION TECHNOLOGY)

By
Mr. Shubhamkumar Yadav

Project Guide
Prof. Punam sindhu

INTODUCTION

Reconstructions that improve image quality can be translated into a reduction of radiation dose because images of the same quality can be reconstructed at lower dose.

Image reconstruction techniques are used to create 2-D and 3-D images from sets of 1-D projections. These reconstruction techniques form the basis for common imaging modalities such as CT, MRI, and PET, and they are useful in medicine, biology, earth science, archaeology, materials science, and nondestructive testing.

Image reconstruction has fundamental impacts on image quality and therefore on radiation dose. For a given radiation dose it is desirable to reconstruct images with the lowest possible noise without sacrificing image accuracy and spatial resolution. Reconstructions that improve image quality can be translated into a reduction of radiation dose because images of the same quality can be reconstructed at lower dose.

The image reconstruction techniques use the measured projection data as input to calculate the density distribution of the desired cross section of the investigated sample as output. Accordingly, the two dimensional image of the desired cross section can be obtained.

Two major categories of reconstruction methods exist, analytical reconstruction and iterative reconstruction (IR) and other one is Back Projection.

What are the common issues in image reconstruction?

Several issues pose a problem with image reconstruction. The first is noise —meaningless data that can interrupt the clarity of an image. In medical imaging, noise can occur as a result of patient movement, interference, shadowing and ghosting. For example, one structure in the body might overshadow another and make it hard to spot. Filtration for noise is one aspect of image reconstruction.

Another issue is scattered or incomplete data. With something such as an X-ray, the image is taken in one film exposure, where X-rays pass through the area of interest and create an image. In other techniques, a patient might be bombarded with radiation or subjected to a magnetic field, generating a substantial amount of data that needs to be assembled to create a picture. The immediate output is not readable or meaningful to a human, and it needs to be passed through an algorithm to generate a picture.

In image reconstruction, there are several approaches that can be taken to filter out the noise without discarding meaningful data and to process the data in a way that will make sense. Iterative reconstruction is a popular technique. The algorithm starts by mapping out low-frequency data, creating a few data points that form the start of the image. Then it overlays a slightly higher frequency, and a higher one, and so on, until a complete image is available.

REQUIREMENTS

Hardware:-

- Operating system- Windows 7,8,10
- Processor- dual core 2.4 GHz (i5 or i7 series Intel processor or equivalent AMD) RAM-4GB

Software:-

- **Python3**

  Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

- **Jupyter Notebook**

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

-

Other Requirements :-

- **Internet connection**
- **Browser**

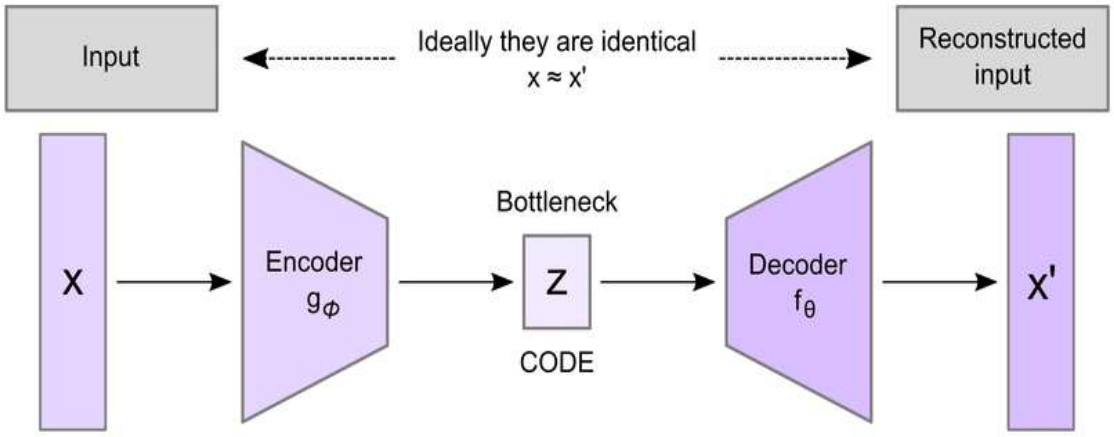
What are Autoencoders?

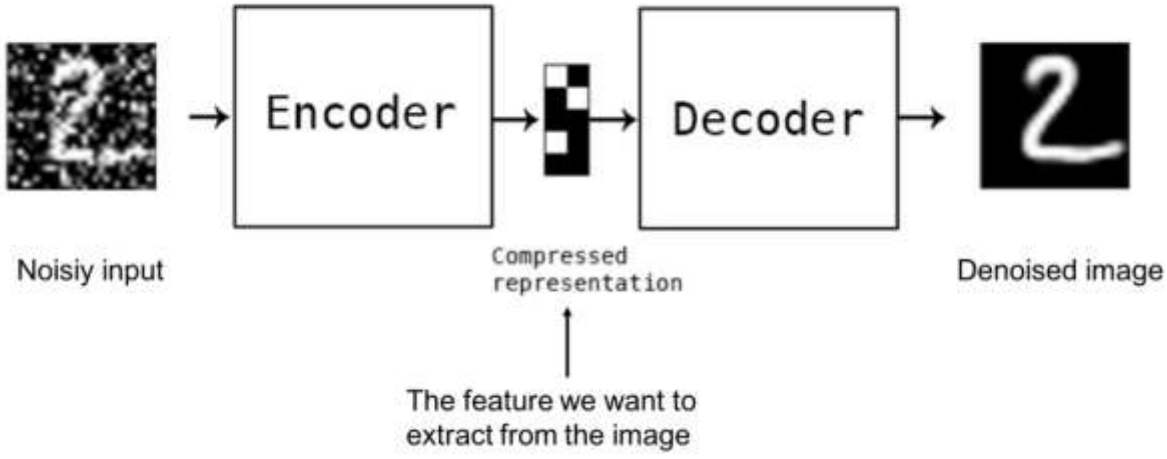
An autoencoder is, by definition, a technique to encode something automatically. By using a neural network, the autoencoder is able to learn how to decompose data (in our case, images) into fairly small bits of data, and then using that representation, reconstruct the original data as closely as it can to the original.

There are two key components in this task:

- Encoder: Learns how to compress the original input into a small encoding
- Decoder: Learns how to restore the original data from that encoding generated by the Encoder

These two are trained together in symbiosis to obtain the most efficient representation of the data that we can reconstruct the original data from, without losing so much of it.





➢ Encoder

The Encoder is tasked with finding the smallest possible representation of data that it can store - extracting the most prominent features of the original data and representing it in a way the decoder can understand.

Think of it as if you are trying to memorize something, like for example memorizing a large number - you try to find a pattern in it that you can memorize and restore the whole sequence from that pattern, as it will be easy to remember shorter pattern than the whole number.

Encoders in their simplest form are simple Artificial Neural Networks (ANNs). Though, there are certain encoders that utilize Convolutional Neural Networks (CNNs), which is a very specific type of ANN.

The encoder takes the input data and generates an encoded version of it - the compressed data. We can then use that compressed data to send it to the user, where it will be decoded and reconstructed.

➢ Decoder

The Decoder works in a similar way to the encoder, but the other way around. It learns to read, instead of generate, these compressed code representations and generate images based on that info. It aims to minimize the loss while reconstructing, obviously.

The output is evaluated by comparing the reconstructed image by the original one, using a Mean Square Error (MSE) - the more similar it is to the original, the smaller the error.

At this point, we propagate backwards and update all the parameters from the decoder to the encoder. Therefore, based on the differences between the input and output images, both the decoder and encoder get evaluated at their jobs and update their parameters to become better.

Building an Autoencoder

- **Keras**

Keras is a Python framework that makes building neural networks simpler. It allows us to stack layers of different types to create a deep neural network - which we will do to build an autoencoder.

Step-1: To import and load the required libraries.

```
import numpy as np

from keras.preprocessing.image import img_to_array
from keras.layers import Dense, Conv2D, MaxPooling2D, UpSampling2D
from keras.models import Sequential
import cv2
import matplotlib.pyplot as plt
```

- **Numpy**

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

- **Matplotlib**

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

- **OpenCV**

It is a type of machine learning library that is Open Source and is meant for Computer vision processing functionalities and hence it is known as OpenCV. It was built to provide a basic understanding and common infrastructure for all Computer Vision software to accelerate the growth of commercial usage. It has a BSD License and hence this makes OpenCV very easy for various businesses to modify and use the code.

It contains over 2500 different algorithms which range from normal to the latest machine learning algorithms and they all can be used in a daily project for personal use. All the algorithms can be utilized to recognize and detect faces, classify human actions, identify objects in various video feeds. It can also be used to track the camera movements, extract 3D models of any object, track moving objects, make 3D cloud points from a stereo camera. It can combine individual images to produce a higher resolution image of a whole scene, removing red eyes, understanding the background scenery, and use an augmented library. OpenCV has over 47 thousand people in their community and over 18 million downloads. As more people are using machine learning to solve their problems, the use of OpenCV is also increasing and people are creating multiple projects like gesture sensing bots or using a phone only by gestures, and many more.

Step 2 :- uploading image :-

```
img=cv2.imread('Mona_Lisa.jpg', 1)
rgb_img=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rgb_img=cv2.resize(rgb_img, (256,256))
img_data.append(img_to_array(rgb_img))
img_final=np.reshape(img_data, (len(img_data),256, 256, 3))
img_final=img_final.astype('float32')/255
```

Step 3 :- coding

```
import numpy as np

from keras.preprocessing.image import img_to_array
from keras.layers import Dense, Conv2D, MaxPooling2D, UpSampling2D
from keras.models import Sequential
import cv2
import matplotlib.pyplot as plt

np.random.seed(42)

img_size=256


img_data=[]

img=cv2.imread('Mona_Lisa.jpg', 1)
rgb_img=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rgb_img=cv2.resize(rgb_img, (256,256))
img_data.append(img_to_array(rgb_img))
img_final=np.reshape(img_data, (len(img_data),256, 256, 3))
img_final=img_final.astype('float32')/255
```

```
model=Sequential()

model.add(Conv2D(64, (3,3), activation='relu', padding='same', input_shape=(256,256,3)))
model.add(MaxPooling2D((2,2), padding='same'))
model.add(Conv2D(32, (3,3),activation='relu',padding='same'))
model.add(MaxPooling2D((2,2), padding='same'))
model.add(Conv2D(16, (3,3),activation='relu',padding='same'))
model.add(MaxPooling2D((2,2), padding='same'))

model.add(Conv2D(16, (3,3), activation='relu', padding='same'))
model.add(UpSampling2D((2,2)))

model.add(Conv2D(32, (3,3), activation='relu', padding='same'))
model.add(UpSampling2D((2,2)))

model.add(Conv2D(64, (3,3), activation='relu', padding='same'))
model.add(UpSampling2D((2,2)))

model.add(Conv2D(3, (3,3), activation='relu', padding='same'))

model.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])

model.summary()

model.fit(img_final, img_final, epochs=2000, shuffle=True)


pred=model.predict(img_final)


plt.imshow(pred[0].reshape(256,256,3))
```

OutPut:-

At this point, we can summarize the results:

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_25 (Conv2D)           (None, 256, 256, 64)      1792
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 128, 128, 64)      0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 128, 128, 32)      18464
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 64, 64, 16)        4624
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 32, 32, 16)        0
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 32, 32, 16)        2320
_________________________________________________________________
```

Note the None here refers to the instance index, as we give the data to the model

```
up_sampling2d_9 (UpSampling2 (None, 64, 64, 16)        0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 64, 64, 32)        4640
_________________________________________________________________
up_sampling2d_10 (UpSampling (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 128, 128, 64)      18496
_________________________________________________________________
up_sampling2d_11 (UpSampling (None, 256, 256, 64)      0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 256, 256, 3)       1731
=================================================================
Total params: 52,067
Trainable params: 52,067
Non-trainable params: 0
_________________________________________________________________
```

The `epochs` variable defines how many times we want the training data to be passed through the model.

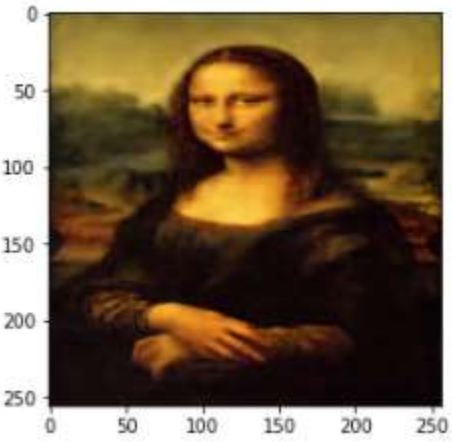
```
_________________________________________________________________
Epoch 1/2000
1/1 [==============================] - 0s 2ms/step - loss: 0.0967 - accuracy: 0.9501
Epoch 2/2000
1/1 [==============================] - 0s 2ms/step - loss: 0.0877 - accuracy: 0.9787
Epoch 3/2000
1/1 [==============================] - 0s 1ms/step - loss: 0.0735 - accuracy: 0.9821
Epoch 4/2000
1/1 [==============================] - 0s 1ms/step - loss: 0.0536 - accuracy: 0.9852
Epoch 5/2000
1/1 [==============================] - 0s 2ms/step - loss: 0.0433 - accuracy: 0.9852
Epoch 6/2000
1/1 [==============================] - 0s 998us/step - loss: 0.0584 - accuracy: 0.9852
Epoch 7/2000
1/1 [==============================] - 0s 998us/step - loss: 0.0478 - accuracy: 0.9852
Epoch 8/2000
1/1 [==============================] - 0s 2ms/step - loss: 0.0417 - accuracy: 0.9852
Epoch 9/2000
1/1 [==============================] - 0s 1ms/step - loss: 0.0437 - accuracy: 0.9852
```

In this model we want data to be train 2000 times.

```
Epoch 1995/2000
1/1 [==============================] - 0s 1ms/step - loss: 0.0018 - accuracy: 0.9843
Epoch 1996/2000
1/1 [==============================] - 0s 2ms/step - loss: 0.0018 - accuracy: 0.9861
Epoch 1997/2000
1/1 [==============================] - 0s 2ms/step - loss: 0.0018 - accuracy: 0.9849
Epoch 1998/2000
1/1 [==============================] - 0s 2ms/step - loss: 0.0018 - accuracy: 0.9857
Epoch 1999/2000
1/1 [==============================] - 0s 1ms/step - loss: 0.0018 - accuracy: 0.9857
Epoch 2000/2000
1/1 [==============================] - 0s 2ms/step - loss: 0.0018 - accuracy: 0.9852
```

Image output:-

```
Out[8]: <matplotlib.image.AxesImage at 0x1b036dd47c0>
```



REFERENCES

https://www.in.mathsworks.com/

https://www.sciencedirect.com/

https://www. towardsdatascience.com/

https://www. stakeabuse.com/