Implementation of Convolutional Autoencoding in Brain Tumor MRI Scan Images

Nithin Sai Jalukuru
School of Engineering and Applied Sciences
University at Buffalo, Buffalo, NY, USA
njalukur@buffalo.edu

Abstract : In this present work, For the tumor MRI brain images, Convolutional Autoencoders is implemented with fine-tuned hyperparameters. An auto encoder is a particular kind of neural network that is primarily made to encode input into a compressed and meaningful representation and then decode it so that the reconstructed input is comparable to the original.

Keywords : Image processing, Augmentation, Autoencoder , Convolutional Autoencoding

1.INTRODUCTION

Autoencoders have been first introduced as a neural network that is trained to reconstruct its input. Their main purpose is learning in an unsupervised manner an "informative" representation of the data that can be used for various implications .The problem is defined to learn the functions $A : R^n \to R^p$ (Encoder) and $B : R^p \to R^n$ (Decoder) that satisfy,

$$argmin_{A,B} E[\Delta(x, B \circ A(x))]$$

where E is the expectation over the distribution of x, and Δ is the reconstruction loss function, which measures the distance between the output of the decoder and the input.

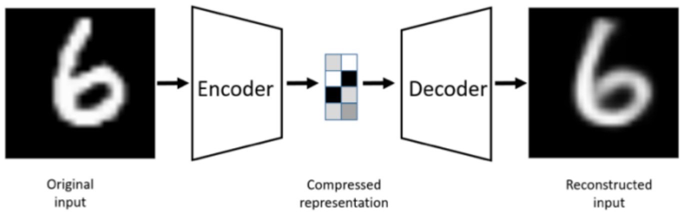


Fig1 : An Auto Encoder Example.

It is significant to note that, from the perspective of applications, one does not want autoencoders to merely learn to copy of the input x. To put it another way, autoencoders are typically constrained in some ways so that they can roughly learn the copy of the inputs.

Autoencoders have historically been utilized for feature extraction or dimensionality reduction . However, as deep convolutional networks and other popular deep learning models have elevated the role of the autoencoder in generative modeling, multiple extended autoencoder models have been proposed by various researchers.

The extended autoencoder models can be roughly classified into three categories:

(1) Extended models based on instantiation of encoder function and decoder function.

(2) Extended models based on regularization technique.

Let's Learn about Convolutional Auto Encoder.

1.1 Convolutional Autoencoder (CAE)

By implementing encoder and decoder functions using convolutional neural networks (CNN), the AE is expanded into the CAE. Convolutional layers and pooling layers are the fundamental components of CNN; a convolutional layer is composed of numerous convolutional nodes with inputs that are two-dimensional feature maps, and the learning parameters are the components of filter matrices. The structure of a convolutional node can be illustrated by figure 2

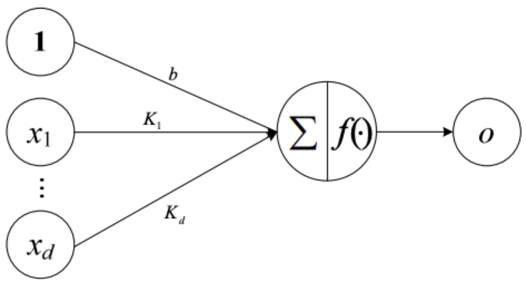


Fig 2 : Structure of Convolutional Node

In figure 2, the 1 is a matrix whose all elements are 1, the b is bias which is a scalar, the x_i $(i = 1,2,3 \dots, d)$ are d input 2- dimensional feature maps which are d matrixes, K_i $(i = 1,2,3 \dots, d)$ are d filter matrixes which are also d matrixes, but they are usually small matrixes .the Σ is sum operator, the f is an activation function, the o is the output 2-D feature map which is a matrix. The node in figure 2 can be modeled by the following equation

$$o = f\left(\sum_{i=1}^{d} K_i * x_i + b * 1\right)$$

Where * is the convolutional operator.

The nodes of a pooling layer, also known as a *sampling layer,* are obtained by sampling the corresponding convolutional layer nodes. Therefore, if there are m nodes in the convolutional layer, m nodes in the pooling layer must also exist.

The CAE can be trained using the gradient descent algorithm or the stochastic gradient descent algorithm because a *CNN is a feedforward neural network*.

2.EXPERIMENTATION

The **Brain MRI Images for Brain Tumor Detection** is taken from Kaggle. This data consists of 155 of samples without tumor and 98 samples with tumors. Following images show the patients with tumor and without tumor.

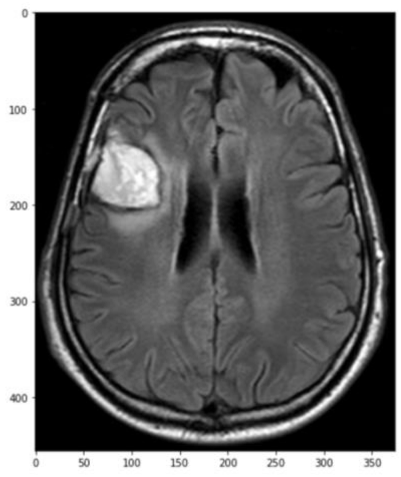


Fig 3 : Tumor Sample

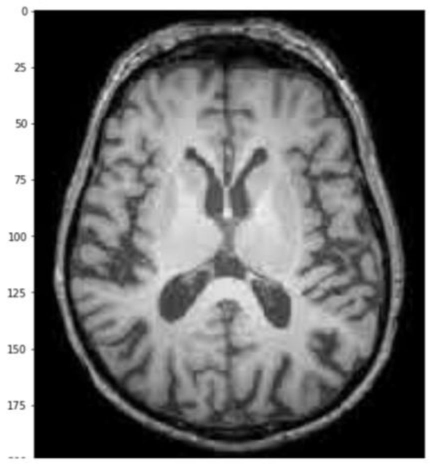


Fig 4 : Non-Tumor Sample

As the samples of tumor and non-tumor are less , the data has been augmented. If the sample is not a tumor, then the generated samples are 2 (i.e. for each non tumor image , it will generate 2 samples for each image) . Likewise, If the sample is a tumor one , for each image 2 samples are generated. Therefore, total number of samples after augmentation is 759

Following images show the augmented tumor and non-tumor samples.

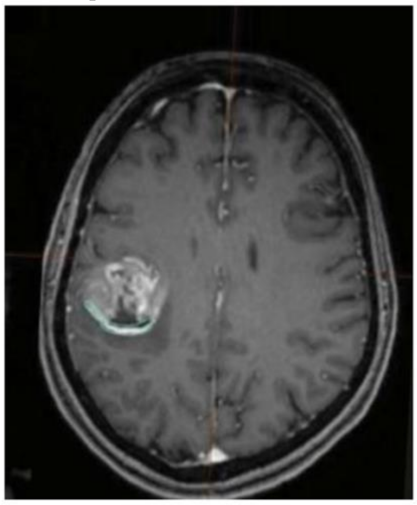


Fig 5 : Augmented tumor sample

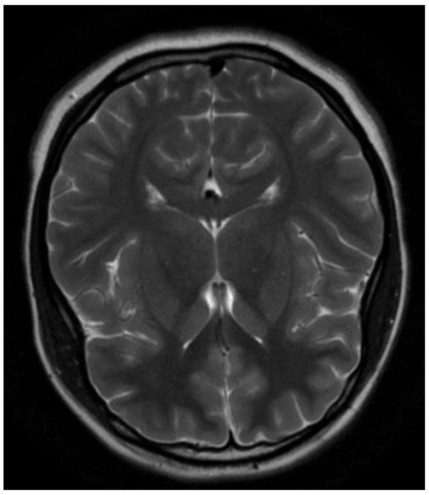


Fig 6 : Augmented Non-Tumor Sample

2.1 Data Cleaning

It can be seen from the above images, there is a lot of waste space around the image. It's better to crop the image for every sample (tumor, non-tumor). Following steps have been followed for cropping the image. First step is to load the image , convert into greyscale and slightly blur image. The image is thresholded in the next step, and any small noise tweaks are then removed by applying a series of erosions and dilations. Finding contours in a thresholded image comes next, after which the biggest one is selected.

The final step is to use the four extreme points to crop the brain out of the original image. Let's examine the results obtained.

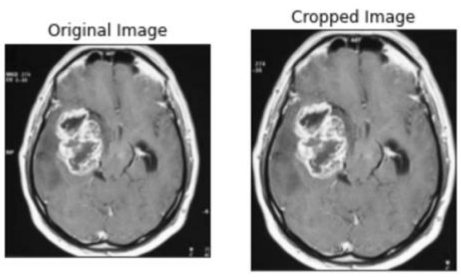


Fig 7 : Original Image and Cropped image (Tumor sample)

Following image shows the cropped images (randomly)

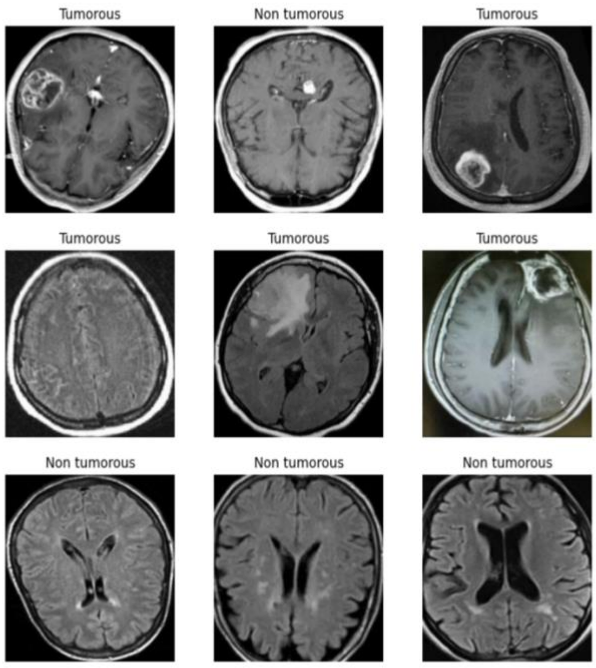


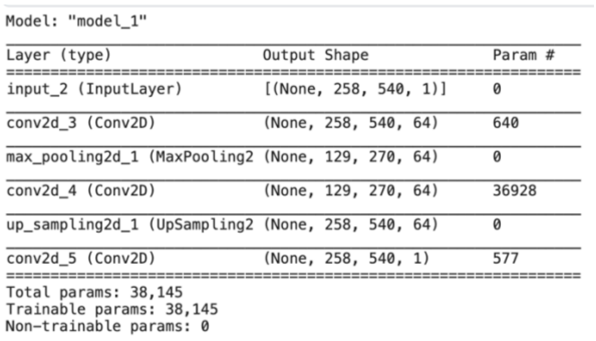
Fig 8 : Cropped Images

Samples split into x_train (465 samples) and x_test (294 samples) .Furthermore 93 validation samples are taken from training set (93 validation, x train 372). And the data is ready for training the model.

3.CONVOLUTION AUTO ENCODER MODELLING

Adamax optimizer is used with learning rate as *0.0001* for the auto encoder classifier. In encoding function ,a convolutional layer with “reLu” activation function is used with “same” padding and then pooling layer is applied. While decoding same layers has been added as of encoding and the output layer has the “sigmoid” activation function.

Finally, the model is trained with training and validation data with 100 epochs and with batch size 20.A model has been selected as shown in fig 9



```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 258, 540, 1)]     0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 258, 540, 64)      640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 129, 270, 64)      0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 129, 270, 64)      36928
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 258, 540, 64)      0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 258, 540, 1)       577
=================================================================
Total params: 38,145
Trainable params: 38,145
Non-trainable params: 0
```

Fig 9 : Model with layers and optimal parameters

At epoch = 1, the training loss is 0.1153 and validation loss (testing set in our case , for training the model) is 0.1107.

At epoch = 100, training loss is 0.0008754 and testing loss is 0.000885. There is not much difference between training and validation loss. So, no overfitting of the data is happening, and data is trained well.

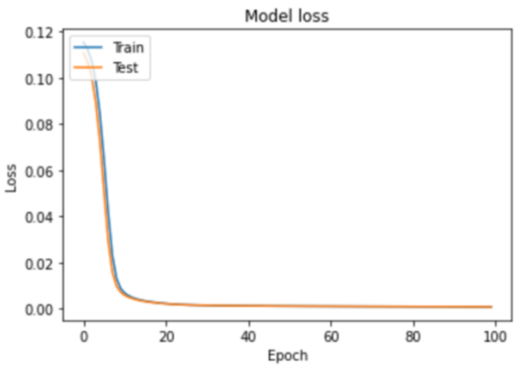


Fig 10 : Model loss while training

Both Training and testing (validation) loss is converging after a point.

4 TEST RESULTS

After the model is trained, now let’s test this convolutional encoder on the testing set and see the results.

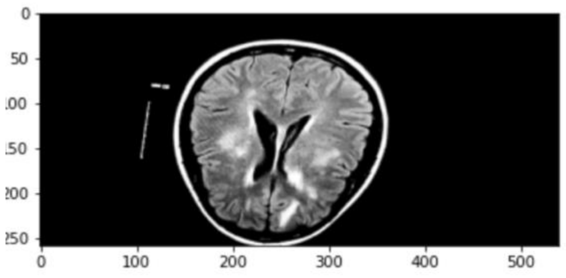


Fig 11: Original Test Sample

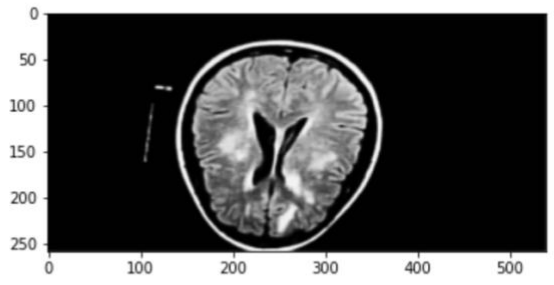


Fig 12 : Predicted Sample

Sample 2,

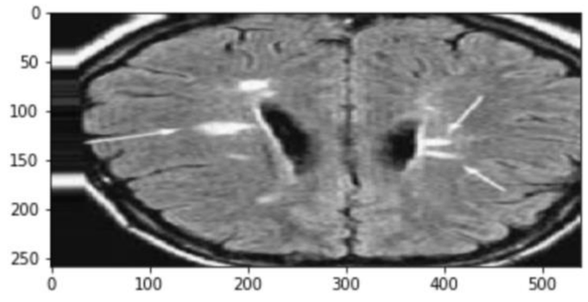


Fig 13 : Test sample 2

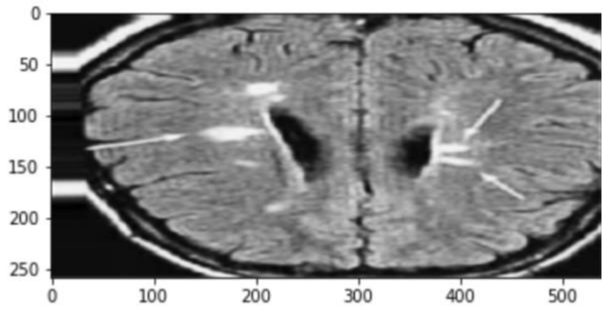


Fig 14 : Predicted sample 2

From the above two test samples , the CAE model has accurately recreated the image sample.

5 CONCLUSIONS

With Learning rate 0.0001, the model has a very low training loss, and this can be further improved by with different set of optimizers and with different epochs. There is a scope for improving this model further.

5 REFERENCES

1. CSE 474/574 Lecture slides, Prof.Chen
2. Sufang ZHANG at. El “*Autoencoder and its various variants*”.
3. Dor Bank, “*Auto Encoders*”
4. Brain MRI With Convolutional Autoencoder, Kaggle