Brain MRI Super-Resolution Network

Lingxiao Gao 44708627

Introduction

The model implemented is able to reconstructs a high-resolution version of an image from a low-resolution (factor of 4) version given.

The working principles were to train a customized CNN network with tuple of (low-res image, high-res image) and optimize mean squared error between prediction and high-res image.

Peak signal-to-noise ratio (PSNR) is commonly used to quantify reconstruction quality for images and is defined via MSE. Lower MSE indicates higher PSNR and reconstruction quality.

The basic structure was inspired by ESPCN (Efficient Sub-Pixel CNN) proposed by Shi, 2016 and Keras implementation by Long, 2020. The model was also embedded with Residual Dense Blocks, inspired by Chakraborty, 2021. Residual blocks prevented early saturation or degradation with accuracy from increasing number of layers.

ADNI MRI Dataset were used in this work (see citation).

Importance of this work

MRI in modern medicine require continuous monitoring human organs in real-time. High-resolution MRI scan could take up to 90 mintues for each scan, and hardly capture the dynamic change in human body. Real-time MRI was possible only with low image quality fast scans, as MRI scanning is done in time-based k-space. Super-Resolution algorithms will be very helpful to provide a higher-resoltion image for better disease diagnosing.

Dependencies and Versions

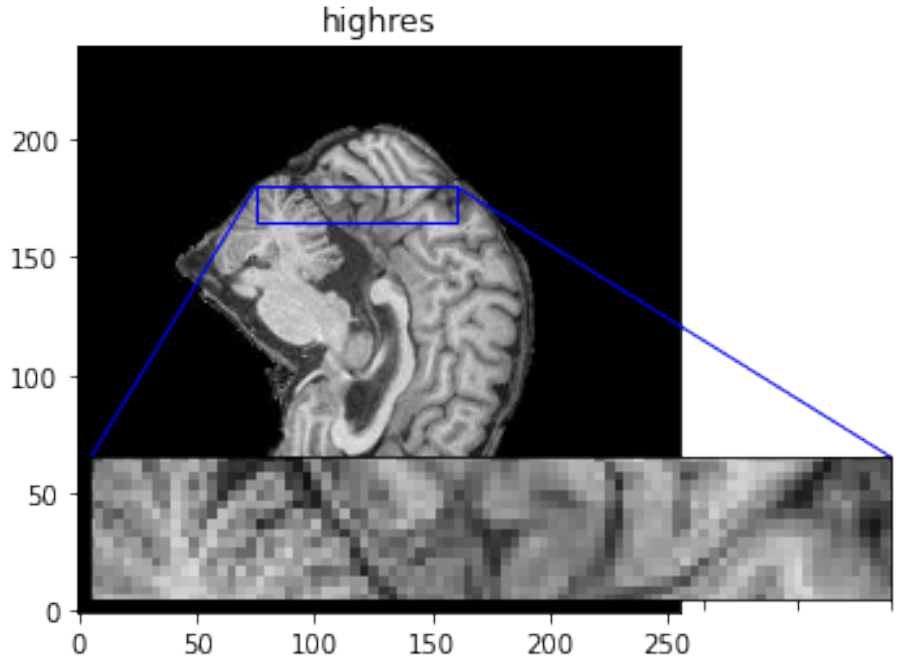
- Ubuntu 16.04 or higher (Ubuntu 22.04.1 LTS was in used)
- NVIDIA® GPU drivers version 450.80.02 or higher (515.76 was in used)

Package	Version
Graphic Card	GTX 3080
System	Ubuntu 22.04.1 LTS
NVIDIA® GPU drivers	515.76
CUDA Toolkit	11.2
cuDNN SDK	8.1.0
Python	3.9.13

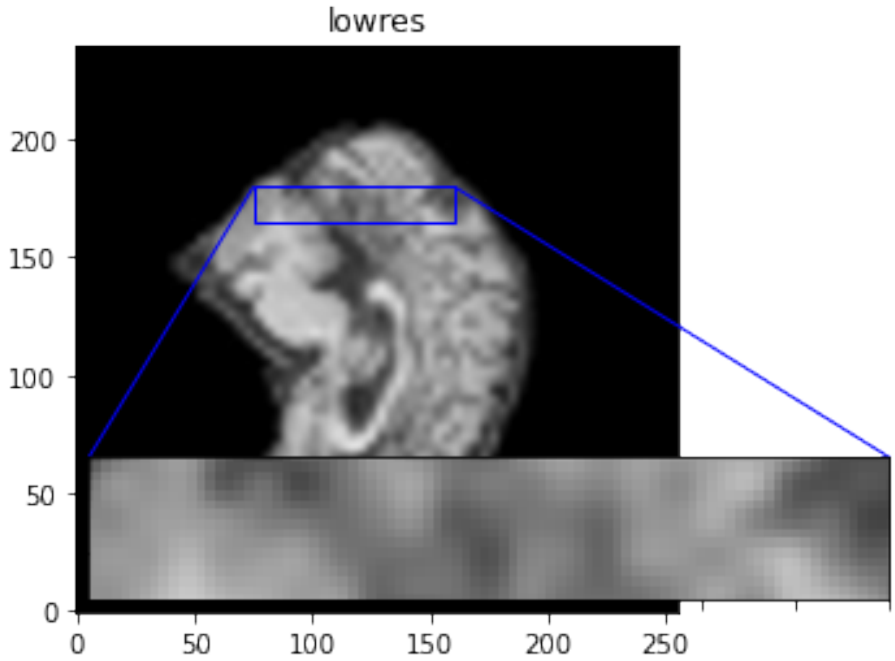
Package	Version
Numpy	1.21.5
Keras	1.1.2
Pillow	2.9.1
matplotlib	3.5.2
Tensorflow	2.9.1

Example Input and Output

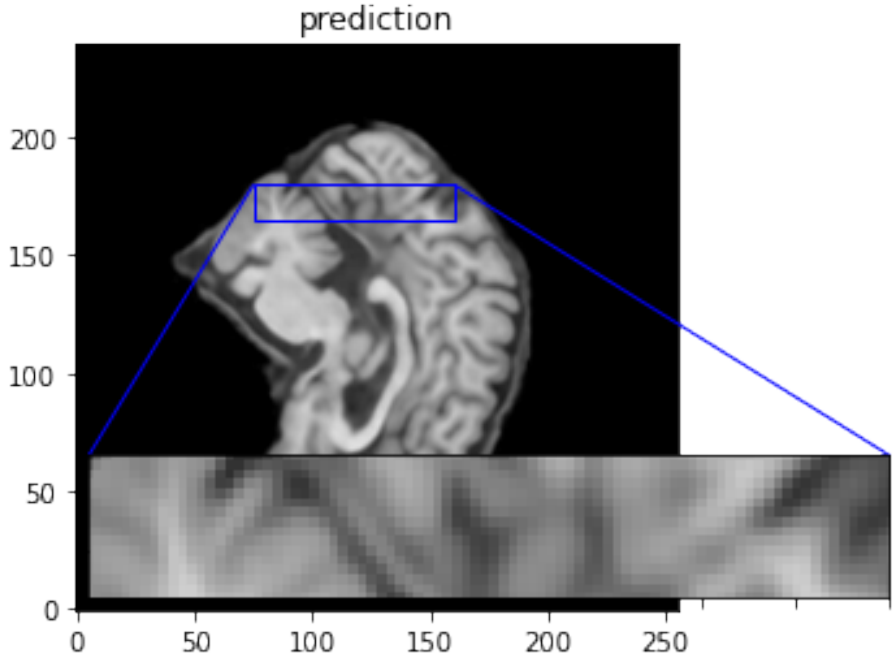
The original image before down-sampling by factor of 4:



The model input is:



The model output is:

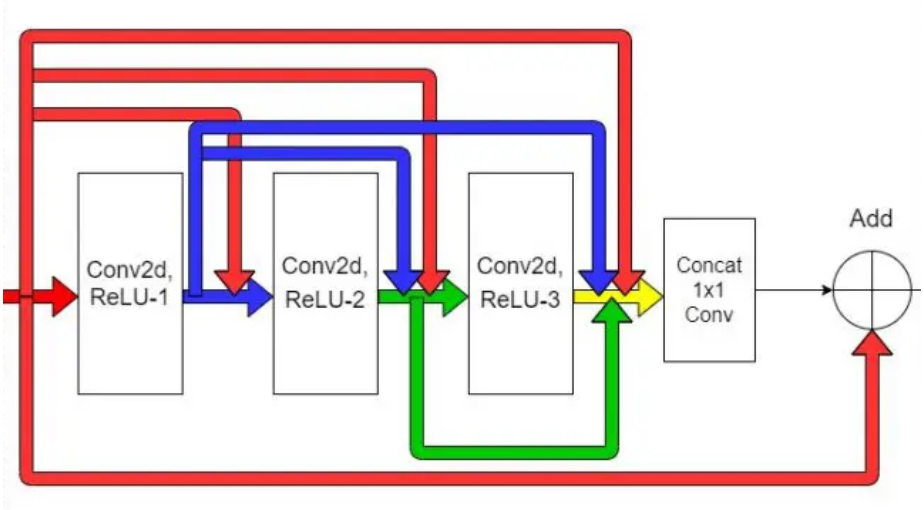


PSNR of low resolution image and high resolution image is 25.5161
PSNR of predict and high resolution is 27.4348
PSNR improvement between low resolution and prediction 1.9188

Model Summary

The model can be seen as:
Input => Conv => Conv => Residual_Block(4 Conv) => Conv => Residual_Block(4 Conv) => Sub-pixel_Conv.

As mentioned above Residual blocks prevented early saturation or degradation with accuracy from increasing number of layers.



Detail of model

As shown below

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, None, None,  0           []
                                 1)]

 conv2d (Conv2D)                (None, None, None,   1664        ['input_1[0][0]']
                                64)

 conv2d_1 (Conv2D)              (None, None, None,   36928       ['conv2d[0][0]']
                                64)

 tf.identity (TFOpLambda)       (None, None, None,   0           ['conv2d_1[0][0]']
                                64)

 conv2d_2 (Conv2D)              (None, None, None,   36928       ['tf.identity[0][0]']
                                64)

 tf.concat (TFOpLambda)         (None, None, None,   0           ['conv2d_1[0][0]',
                                128)                             'conv2d_2[0][0]']

 conv2d_3 (Conv2D)              (None, None, None,   73792       ['tf.concat[0][0]']
                                64)

 tf.concat_1 (TFOpLambda)       (None, None, None,   0           ['conv2d_1[0][0]',
                                192)                             'conv2d_2[0][0]',
                                                                 'conv2d_3[0][0]']

 conv2d_4 (Conv2D)              (None, None, None,   110656      ['tf.concat_1[0][0]']
                                64)

 tf.concat_2 (TFOpLambda)       (None, None, None,   0           ['conv2d_1[0][0]',
                                256)                             'conv2d_2[0][0]',
                                                                 'conv2d_3[0][0]',
                                                                 'conv2d_4[0][0]']

 conv2d_5 (Conv2D)              (None, None, None,   16448       ['tf.concat_2[0][0]']
                                64)

 add (Add)                      (None, None, None,   0           ['conv2d_5[0][0]',
                                64)                              'conv2d_1[0][0]']

 conv2d_6 (Conv2D)              (None, None, None,   18464       ['add[0][0]']
                                32)

 tf.identity_1 (TFOpLambda)     (None, None, None,   0           ['conv2d_6[0][0]']
                                32)

 conv2d_7 (Conv2D)              (None, None, None,   9248        ['tf.identity_1[0][0]']
                                32)

 tf.concat_3 (TFOpLambda)       (None, None, None,   0           ['conv2d_6[0][0]',
                                64)                              'conv2d_7[0][0]']

 conv2d_8 (Conv2D)              (None, None, None,   18464       ['tf.concat_3[0][0]']
                                32)

 tf.concat_4 (TFOpLambda)       (None, None, None,   0           ['conv2d_6[0][0]',
                                96)                              'conv2d_7[0][0]',
                                                                 'conv2d_8[0][0]']

 conv2d_9 (Conv2D)              (None, None, None,   27680       ['tf.concat_4[0][0]']
                                32)

 tf.concat_5 (TFOpLambda)       (None, None, None,   0           ['conv2d_6[0][0]',
                                128)                             'conv2d_7[0][0]',
                                                                 'conv2d_8[0][0]',
                                                                 'conv2d_9[0][0]']

 conv2d_10 (Conv2D)             (None, None, None,   4128        ['tf.concat_5[0][0]']
                                32)

 add_1 (Add)                    (None, None, None,   0           ['conv2d_10[0][0]',
                                32)                              'conv2d_6[0][0]']

 conv2d_11 (Conv2D)             (None, None, None,   4624        ['add_1[0][0]']
                                16)

 tf.nn.depth_to_space (TFOpLamb  (None, None, None,   0          ['conv2d_11[0][0]']
 da)                            1)

==================================================================================================
Total params: 359,024
Trainable params: 359,024
Non-trainable params: 0
```

Pre-processing

The dataset has 30520 samples of Alzheimer's disease (AD) and Cognitive Normal (CN).

Training set	Validation set	Testing set
17216	4304	9000

The dataset was resized to 300x300 and normalized from scale of (0,255) to (0,1). Both train_ds and validation_ds were assigned in form of **tuple** (low-res_ds, high-res_ds).

Utility functions

These are some important utility functions, details are commented thoughout the code.

dataset_preprocessing

Scale normalization, color space convertion and create tuple of dataset for trainning.

setup_dataset

Create image dataset from directory with 8:2 ratio.

scaling

Normalize an image from scale of (0,255) to (0,1).

process_input/target

Convert rgb color spcae to yuv color space to extract luminance information.

get_lowres_image

Utlize BICUBIC to downsample an image by a specified factor.

upscale_image

Reconstruct a low-resolution image to a high-resolution through a model.

plot_results

Plot results with additional zoom-in at a facotor of 4

ESPCNCallback

The ESPCNCallback object will compute and display the PSNR metric during training and testing

Reference

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., ... & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1874-1883).

Long, X. (2020). Image Super-Resolution using an Efficient Sub-Pixel CNN. https://keras.io/examples/vision/super_resolution_sub_pixel/#define-callbacks-to-monitor-training

ADNI dataset for Alzheimer's disease. (2022). ADNI MRI Dataset [Data set]. https://adni.loni.usc.edu/