Individual Report of Final Project

Yixi Liang

Department of Data Science, The George Washington University

DATS_6203_10: Machine Learning II

Amir Jafari

Dec 12, 2022

Table of Content

Introduction

In this final project, we use the dataset from Unsplash.com (https://unsplash.com/data). It provides the world's largest open library dataset for free. The Unsplash Dataset is created by 250,000+ contributing photographers and billions of searches across thousands of applications, uses, and contexts. And we use this dataset to Improve the quality of low-resolution images. In this final project, our group try several things, such as traditional method to do Interpolations, Autoencoder and General Adversarial Network.

Description of individual work

In this section I try to use autoencoder to do the image super resolution. Firstly, I use autoencoder sample code from Keras official document to try vanilla autoencoder, and then I try the different structure of autoencoder, and finally I make our own design structure. The autoencoder designed by (Shaikh, 2022) is good. But there are still some problems remain. The image looks good but not as good as the high resolution one, so I try to change the decoder part from CNN to CNN2dtranspose.

Firstly, I use the traditional method to improve the image, which are, INTER_NEAREST, INTER_LINEAR, INTER_AREA, INTER_CUBIC, INTER_LANCZOS4.

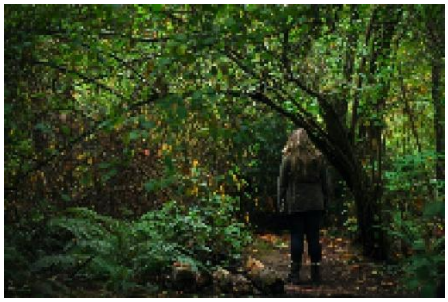
Figure 1

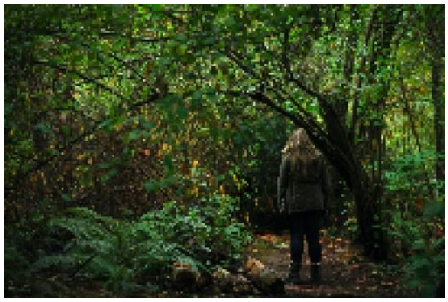
Script of traditional method.

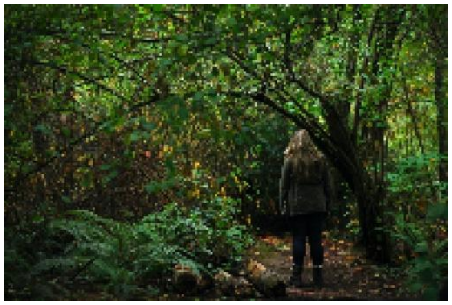
```python
for i in range(1, 4):
    img_l = cv2.imread(lowres_folder + f'/{i}_6.jpg')
    img_h = cv2.imread(hires_folder + f'/{i}.jpg')
    size = (1200, 800)

    new_img = cv2.resize(img_l, size, interpolation=cv2.INTER_NEAREST)
    new_img_path = base_directory + f'/test_cv/{i}_cv_INTER_NEAREST.jpg'
    cv2.imwrite(new_img_path, new_img)

    new_img = cv2.resize(img_l, size, interpolation=cv2.INTER_LINEAR)
    new_img_path = base_directory + f'/test_cv/{i}_cv_INTER_LINEAR.jpg'
    cv2.imwrite(new_img_path, new_img)

    new_img = cv2.resize(img_l, size, interpolation=cv2.INTER_AREA)
    new_img_path = base_directory + f'/test_cv/{i}_cv_INTER_AREA.jpg'
    cv2.imwrite(new_img_path, new_img)

    new_img = cv2.resize(img_l, size, interpolation=cv2.INTER_CUBIC)
    new_img_path = base_directory + f'/test_cv/{i}_cv_INTER_CUBIC.jpg'
    cv2.imwrite(new_img_path, new_img)

    new_img = cv2.resize(img_l, size, interpolation=cv2.INTER_LANCZOS4)
    new_img_path = base_directory + f'/test_cv/{i}_cv_INTER_LANCZOS4.jpg'
    cv2.imwrite(new_img_path, new_img)

    # sacle = 4
    # new_img = bicubic(img_l, sacle)
    # new_img_path = base_directory + f'/test_cv/{i}_cv_bicubic.jpg'
    # cv2.imwrite(new_img_path, new_img)

print('Finish')
```

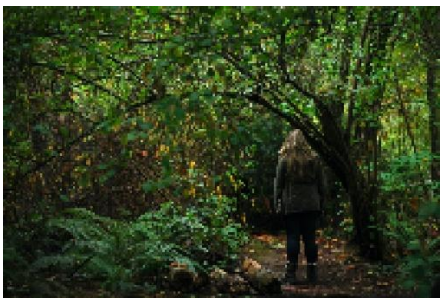
Figure 2

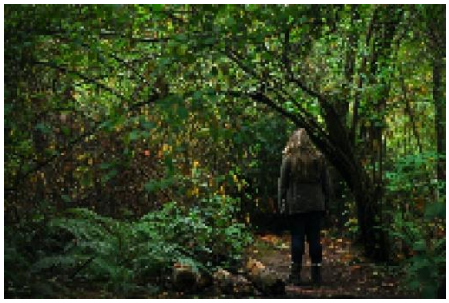
Traditional method in cv of INTER_NEAREST, INTER_LINEAR, INTER_AREA, INTER_CUBIC, INTER_LANCZOS4.











Secondly, I use autoencoder sample code from Keras official document to try vanilla autoencoder, and then I try the different structure of autoencoder, and finally I make my own structure.

Figure 3

Architecture of Autoencoder.

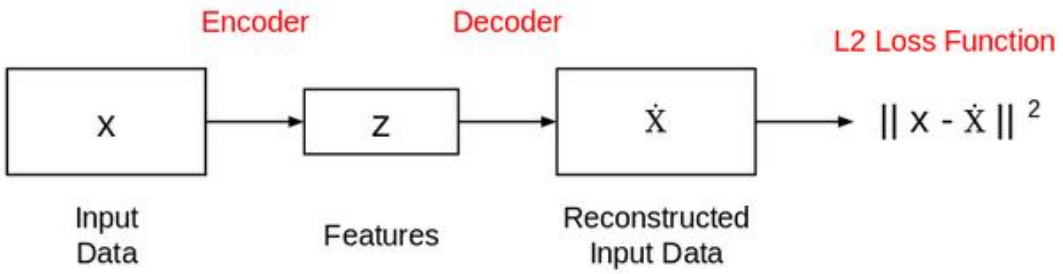


Figure 4

Script of autoencoder structure from (Shaikh, 2022)

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_1 (InputLayer)         [(None, 800, 1200,  0           []
                             3)]

conv2d (Conv2D)              (None, 800, 1200, 6  1792        ['input_1[0][0]']
                             4)

conv2d_1 (Conv2D)            (None, 800, 1200, 6  36928       ['conv2d[0][0]']
                             4)

max_pooling2d (MaxPooling2D)   (None, 400, 600, 64  0          ['conv2d_1[0][0]']
                               )

dropout (Dropout)            (None, 400, 600, 64  0           ['max_pooling2d[0][0]']
                             )

conv2d_2 (Conv2D)            (None, 400, 600, 12  73856       ['dropout[0][0]']
                             8)

conv2d_3 (Conv2D)            (None, 400, 600, 12  147584      ['conv2d_2[0][0]']
                             8)

max_pooling2d_1 (MaxPooling2D)  (None, 200, 300, 12  0         ['conv2d_3[0][0]']
                             8)

conv2d_4 (Conv2D)            (None, 200, 300, 25  295168      ['max_pooling2d_1[0][0]']
                             6)

up_sampling2d (UpSampling2D)   (None, 400, 600, 25  0          ['conv2d_4[0][0]']
                             6)

conv2d_5 (Conv2D)            (None, 400, 600, 12  295040      ['up_sampling2d[0][0]']
                             8)

conv2d_6 (Conv2D)            (None, 400, 600, 12  147584      ['conv2d_5[0][0]']
                             8)

add (Add)                    (None, 400, 600, 12  0           ['conv2d_3[0][0]',
                             8)                               'conv2d_6[0][0]']

up_sampling2d_1 (UpSampling2D)  (None, 800, 1200, 1  0         ['add[0][0]']
                             28)

conv2d_7 (Conv2D)            (None, 800, 1200, 6  73792       ['up_sampling2d_1[0][0]']
                             4)

conv2d_8 (Conv2D)            (None, 800, 1200, 6  36928       ['conv2d_7[0][0]']
                             4)

add_1 (Add)                  (None, 800, 1200, 6  0           ['conv2d_8[0][0]',
                             4)                               'conv2d_1[0][0]']

conv2d_9 (Conv2D)            (None, 800, 1200, 3  1731        ['add_1[0][0]']
                             )

==================================================================================================
Total params: 1,110,403
Trainable params: 1,110,403
Non-trainable params: 0
```

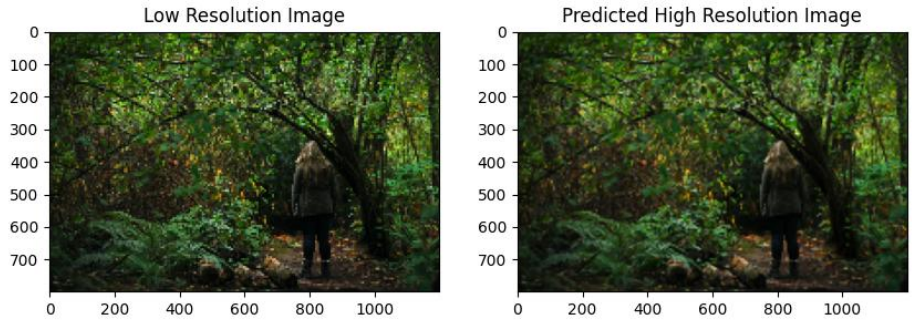
Figure 5

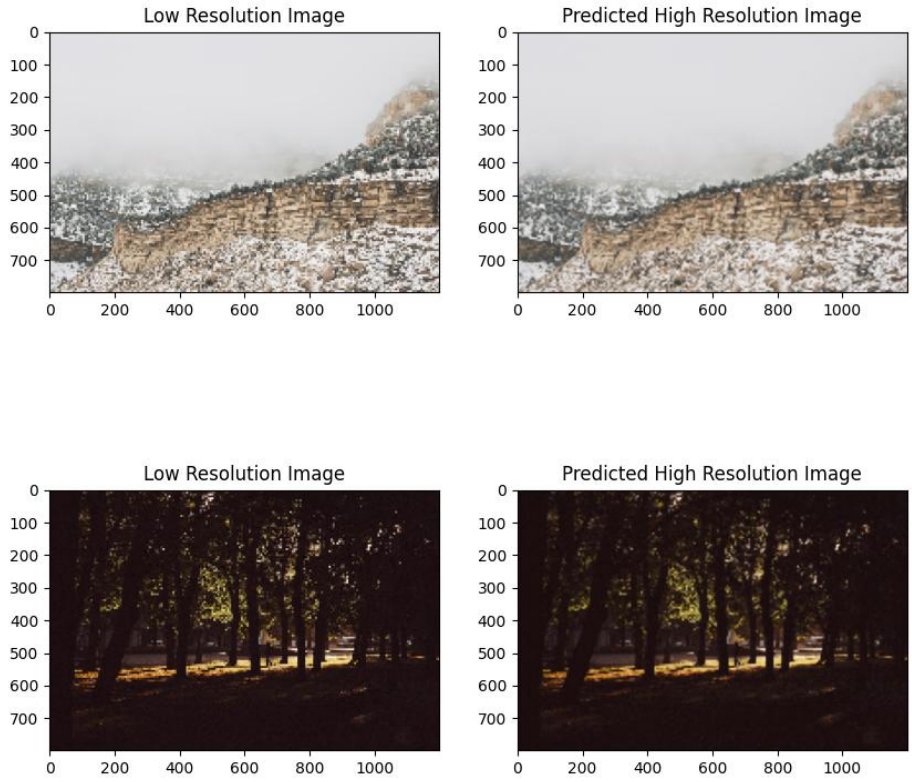
Script of autoencoder structure we change.

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
encoder_input (InputLayer)   [(None, 800, 1200,  0           []
                             3)]

conv2d (Conv2D)              (None, 800, 1200, 6  1792        ['encoder_input[0][0]']
                             4)

conv2d_1 (Conv2D)            (None, 800, 1200, 6  36928       ['conv2d[0][0]']
                             4)

conv2d_2 (Conv2D)            (None, 800, 1200, 6  36928       ['conv2d_1[0][0]']
                             4)

max_pooling2d (MaxPooling2D)   (None, 400, 600, 64  0          ['conv2d_2[0][0]']
                             )

dropout (Dropout)            (None, 400, 600, 64  0           ['max_pooling2d[0][0]']
                             )

conv2d_3 (Conv2D)            (None, 400, 600, 12  73856       ['dropout[0][0]']
                             8)

conv2d_4 (Conv2D)            (None, 400, 600, 12  147584      ['conv2d_3[0][0]']
                             8)

max_pooling2d_1 (MaxPooling2D)  (None, 200, 300, 12  0         ['conv2d_4[0][0]']
                             8)

conv2d_5 (Conv2D)            (None, 200, 300, 25  295168      ['max_pooling2d_1[0][0]']
                             6)

conv2d_transpose (Conv2DTransp  (None, 400, 600, 25  590080    ['conv2d_5[0][0]']
ose)                         6)

conv2d_transpose_1 (Conv2DTran  (None, 400, 600, 12  295040    ['conv2d_transpose[0][0]']
spose)                       8)

add (Add)                    (None, 400, 600, 12  0           ['conv2d_4[0][0]',
                             8)                               'conv2d_transpose_1[0][0]']

conv2d_transpose_2 (Conv2DTran  (None, 800, 1200, 1  147584    ['add[0][0]']
spose)                       28)

conv2d_transpose_3 (Conv2DTran  (None, 800, 1200, 6  73792     ['conv2d_transpose_2[0][0]']
spose)                       4)

conv2d_transpose_4 (Conv2DTran  (None, 800, 1200, 6  36928     ['conv2d_transpose_3[0][0]']
spose)                       4)

add_1 (Add)                  (None, 800, 1200, 6  0           ['conv2d_transpose_4[0][0]',
                             4)                               'conv2d_2[0][0]']

conv2d_transpose_5 (Conv2DTran  (None, 800, 1200, 3  1731      ['add_1[0][0]']
spose)                       )

==================================================================================================
Total params: 1,737,411
Trainable params: 1,737,411
Non-trainable params: 0
```

Figure 6

Result of the autoencoder

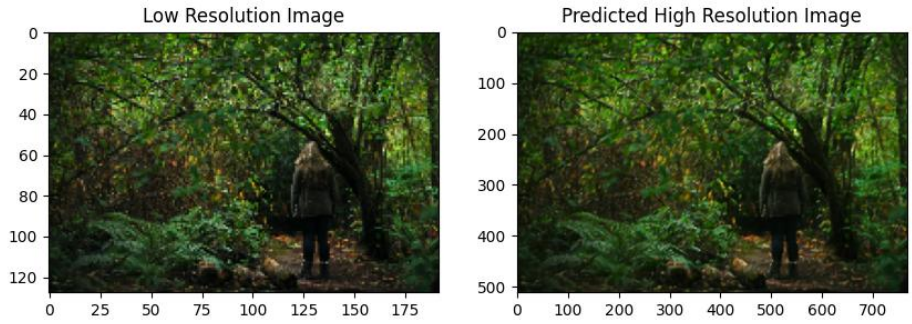


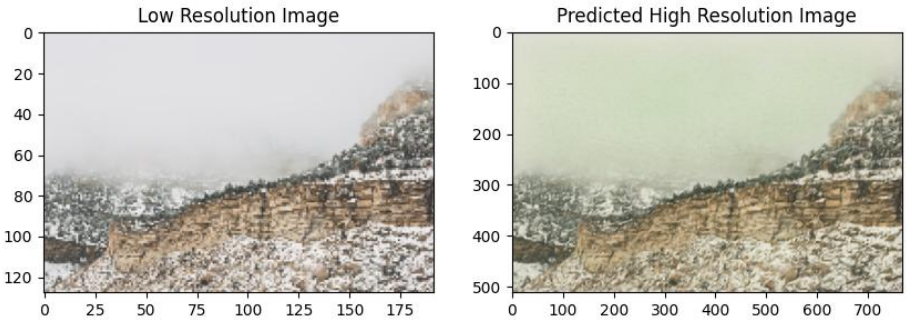


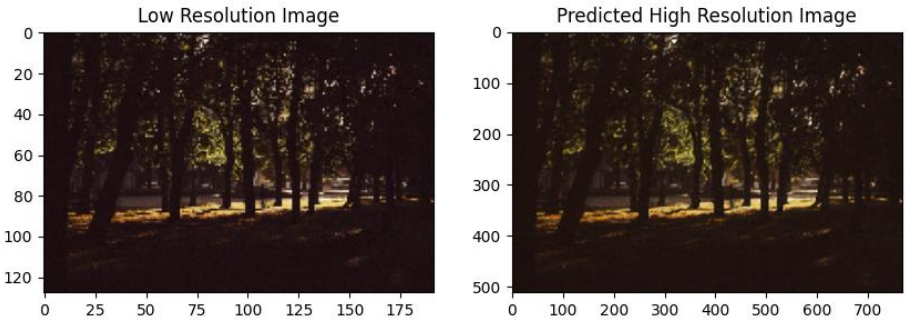
Finally, I also try GAN. but I did not go any deeper into it. I only run the original code.

Figure 7

Result of the GAN







Summary

Autoencoder's performance is significant, but there are still many problems here. When we use the lowest resolution as our input, the predicted image is just like blur the boundary of the lowest resolution image, it is not actually improving the quality of the image. After that we also find Variational Autoencoder. But we find that VAE is kind of generative model, and the output of VAE is stochastic, which is not suitable for our project. The percentage of the code is 50%.

References

Ilopezfr. (n.d.). Image-superres/image_super_resolution_using_autoencoders.ipynb at master · ILOPEZFR/image-superres. GitHub. Retrieved December 12, 2022, from https://github.com/ilopezfr/image-superres/blob/master/Image_Super_Resolution_using_Autoencoders.ipynb

Joshi, P. (2020, May 8). Autoencoders: Enhancing image resolution. Analytics Vidhya. Retrieved December 12, 2022, from https://www.analyticsvidhya.com/blog/2020/02/what-is-autoencoder-enhance-image-resolution/

Shaikh, Q. (2022, October 14). *Image super resolution (from unsplash)*. Kaggle. Retrieved December 11, 2022, from https://www.kaggle.com/datasets/quadeer15sh/image-super-resolution-from-unsplash