Video Resolution Upscaling Using Neural Networks

Oleksandr Korotetskyi

ČVUT - FIT

korotole@cvut.cz

December 31, 2021

1 Introduction

The main goal of this project was to create a program that is capable of creation of right-resolution video from the low-resolution one.

Or in other words, since all the videos do consist of frames, the task may be redefined as follows: *reconstruct the high-resolution image using the low-resolution one*. It implies, that later all the recondtructed frames can be grouped together to form a video.

The semestral work was focused primarily on resolution upscalling of a famous *"L'Arrivée d'un train en gare de La Ciotat"* video.

In addition, it is worth mentioning that among the main methods that do not use neural networks, there are such as bicubic and bilinear interpolations, nearest-neighbor etc. But those methods / algorithms often might lead to blurry or a strongly smoothed output image.

For instance, bilinear interpolation just only adds missing pixels from known neighboring pixels.

If it comes to solving this problem using neural networks, then the most standard and used approach is using of *convolutional neural networks (CNNs)*. As the result, the processed (created) image much more resembles the real ones comparely to other methods.

UNet architecture is also widely used for obtaining of super resolution images, however it was not covered by this semestral work.

2 Input data & preprocessing

I have downloaded the very video from the open sources.

Then, I have developed some utilities to split the video into frames and back (see resources section).

3 Methods

Overview of used and/or implemented methods.

3.1 Pre-trained models

Pre-trained models were obtained from Github [1].

Models were trained for 86 epochs of 1000 batches of 8 32x32 augmented patches taken from LR images from DIV2K dataset.

3.1.1 Small RDN + PSNR

The RDN implementation is primarily based on [6].

RDN mainly consists of four parts [6]:

- shallow feature extraction net (SFENet)
- redidual dense blocks (RDBs)
- dense feature fusion (DFF)
- up-sampling net (UPNet)

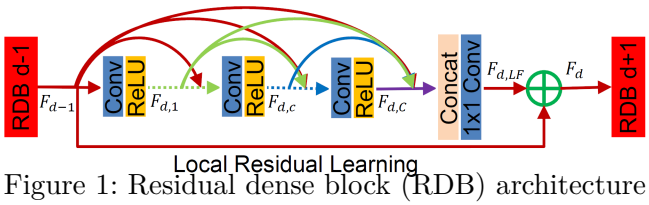

Figure 1: Residual dense block (RDB) architecture [6]

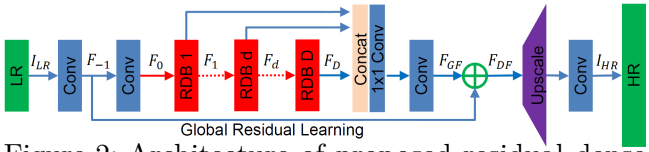

Figure 2: Architecture of proposed residual dense network (RDN) [6]

For evalution Peak signal-to-noise ratio metric was used.

The main parameters of the implemented architecture structure are:

- D - number of Residual Dense Blocks (RDB)
- C - number of convolutional layers stacked inside a RDB
- G - number of feature maps of each convolutional layers inside the RDBs

- G0 - number of feature maps for convolutions outside of RDBs and of each RBD output

Small RDN in our case means D = 10, C = 3, G = 64, G0 = 64. Resulting image is twice the size (in both dimensions) of original LR sample.

3.1.2 Large RDN + PSNR

The model itself is the almost precise copy of hte *small* one. The only thighs that do differ with the previous madel are some parameters of implememted architecture.

For example, in this case D = 20 and C = 6.

3.1.3 RDN, VGG features losses + GANS

This model is also a part of the ISR project [1].

3.1.4 RRDN + ESRGAN

RRDN implementation is based on [5].

While designing this model some flaws of previous method were considered - artefacts in images due to noise and compression.

More advanced loss function is used (previously PSNR) - it is a combination of:

- MSE - original and super-resolution image
- MSE of features from VGG - original and super-resolution image
- GAN output

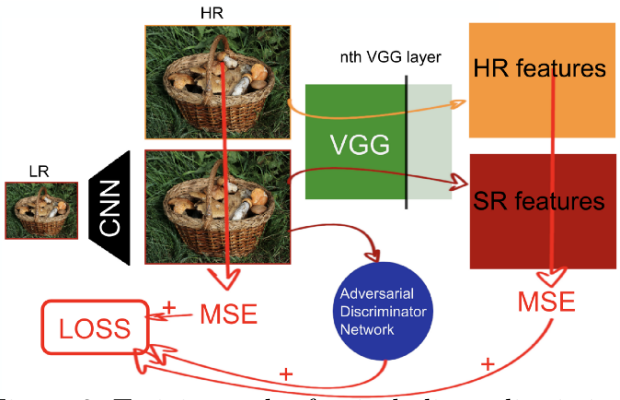

Figure 3: Training cycle after including a discriminator network. The real process also entails the training of the discriminator network, which is not included in this graph [**?**]

Also more complex blocks are used - called Residual in-Residual Dense Block (RRDB), which should be more easy to train than RDB.

Architecture of proposed Residual in-Residual Dense Network (RRDN):

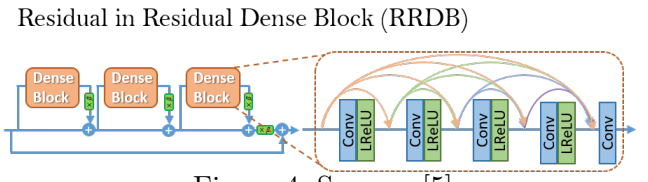

Figure 4: Source: [5]

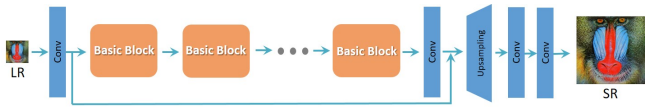

Figure 5: Source: [5]

ESRGAN uses discriminator called Relativistic GAN, which reports probability of picture A being more realistic than B.

The main parameters of the architecture structure are:

- T - number of Residual in Residual Dense Blocks (RRDB)
- D - number of Residual Dense Blocks (RDB) insider each RRDB
- C - number of convolutional layers stacked inside a RDB
- G - number of feature maps of each convolutional layers inside the RDBs
- G0 - number of feature maps for convolutions outside of RDBs and of each RBD output

Pretrained architecture is shaped as: T= 10, D = 3, C = 4, G = 32, G0 = 32. Resulting image is 4 times the size (in both dimensions) of original LR sample.

3.2 Custom trained model

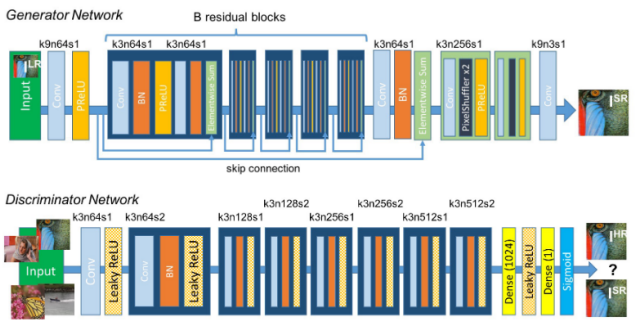

Figure 6: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer

After conducting experimets with pretrained models, I started developing the custom one. The inspiration and implementation were obtained from [2] which is based on SRGANS.

The idea of my implemetation is to use inverted residual blocks instead of usual ones for the fast operations at model architechture.

3.2.1 Training

In the first step I have obtained about 30 parts of video ranging through movies filmed at 1900 by simple crawler I have written.

Later, I was about obtaining the exact images for training, but after testing on the particularly selected one, I realized that my model is errorneous. Something at the very model architecture went wrong.

4 Results

The obtained results seem to demonstrate that 1st and 2nd networks ("Small RDN + PSNR" and "Large RDN + PSNR") show almost the same output in average. That is, a larger number of RDBs and convolutions layers does not give much increase in the result.

The 3rd network ("RDN, VGG features losses + GANS"), as experiments have shown, works rather poorly with high scale factor or in the case of a large number of details in the image, because it considers them as noise and tries excessively to smooth them out. And finally, the 4th network ("RRDN + ESRGAN") shows itself as the best. It gives the best results of image quality.

Moreover, I have conducted some experimets with the customly developed model, however I was not able to produce the adequate results due to some implementation-related errors.

Resources:

- Video before resoluition upscalling: `https://youtu.be/r2gPs5x0zYo`
- Video after resolution upscalling: `https://youtu.be/1GokJ6XmqHI`
- GitHub with all the source codes & custom model: `https://github.com/csraea/VidResUp`
- GitLab Repository: `https://gitlab.fit.cvut.cz/korotole/mvi-2021`
- Google Collab with experiments: `https://colab.research.google.com/drive/1O0f0BEZcQ6-BdOOm0jMtYDRHswVRd3pi?usp=sharing`

5 Conclusion

All the objectives were met - I have tried different approaches for resolution upscaling of a short video clip and I have implemeted the source code that really works.

Best results were obtained using pre-trained ISR RRDN + ESRGANS. Most probably better results would have been obtained if I trained the ISR implementation on my own samples + using more than 1k samples.

To obtain best results it would make sense to use something like Waifu 2x [3], which was developed for the purpose of upscaling animated pictures. Another options are EDVR [4] or 3D convolutional network, which are better solutions for working with video - they are not focused on single frame at one time.

In conclusion, unfortunately, as it was mentioned previously, my custom developed model is not capable of video resolution upscalling at the moment.

Future developments might be aimed at finishing my custom model, trying further upscalling methods and, perhaps, combining them.

References

[1] Francesco Cardinale. Isr. `https://github.com/idealo/image-super-resolution`, 2018.

[2] Hasnain Raza et al. Fast-srgan. `https://github.com/HasnainRaz/Fast-SRGAN`, 2019.

[3] nagadomi. waifu2x. `https://github.com/HasnainRaz/Fast-SRGAN`, 2015.

[4] Xintao Wang, Kelvin C.K. Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. June 2019.

[5] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *The European Conference on Computer Vision Workshops (ECCVW)*, September 2018.

[6] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, 2018.