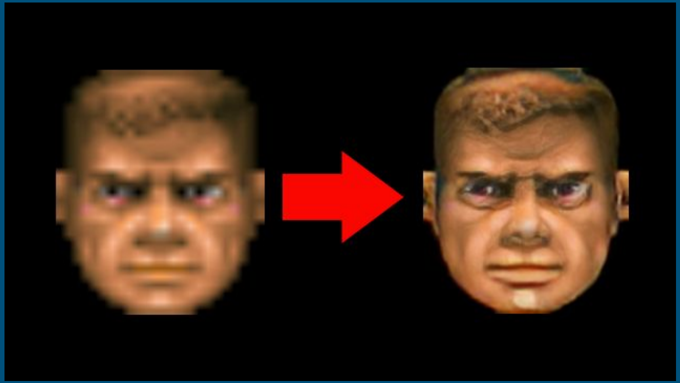
Super-resolution of in-game textures

Bogdan Sydor, Maksym Protsyk

Main idea

We encounter old low-quality images very often in our everyday life and it can be a very unsatisfying experience. Photos taken on old smartphones and old programs look terrible when viewing them on modern screens. This problem also applies to old games which can be almost unplayable without any upgrades made to their graphics and that’s exactly the problem we decided to solve in our project. **Our main goal is to explore and develop methods of textures upscaling and test them out on a real data.**



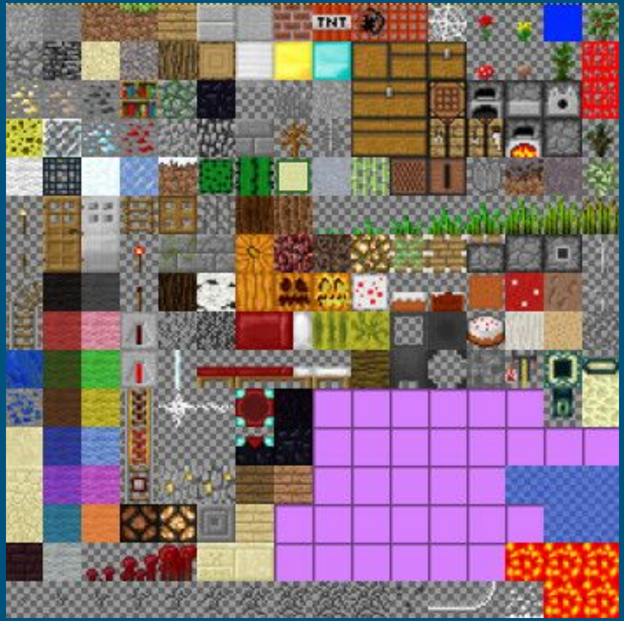
Short results

As a result, we propose:

- Own approach to image upscaling
- Flexible and extendable pipeline for fast upscaling models training with all most used technics
- Tool for images upscaling, using models trained on our pipeline

Data

At the beginning we wanted to work with the kaggle dataset which contained a huge amount of different real-life textures, however, it turned out to be not so suitable for our task and we settled in with a set of minecraft textures, because it is easy to find a significant number of texture packs for this games and this simplifies the data collection process (there are no datasets for textures super-resolution problem).

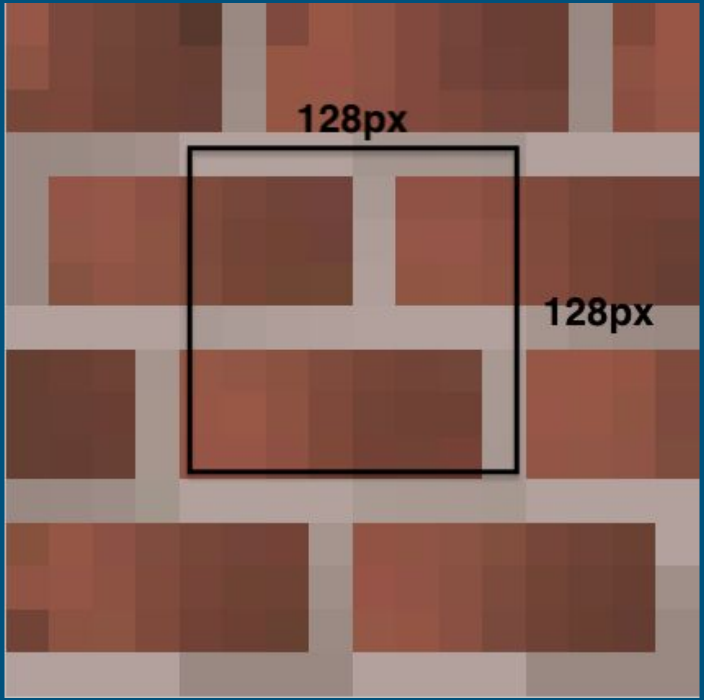


Data pre-processing

Due to the fact that our dataset had images of resolution equal to 512x512 (which needs to much computational power to process) and it was too small, we came up with the following preprocessing algorithm (we apply these actions to the image):

- Horizontal flip (p=0.5)
- Shift + Scale + Rotate
- Random crop to 128x128
- Dividing all pixel values by 255
- Normalization
- Resizing to 64x64 (image that will be upscaled)

The final dataset contained 737 images.



Approaches

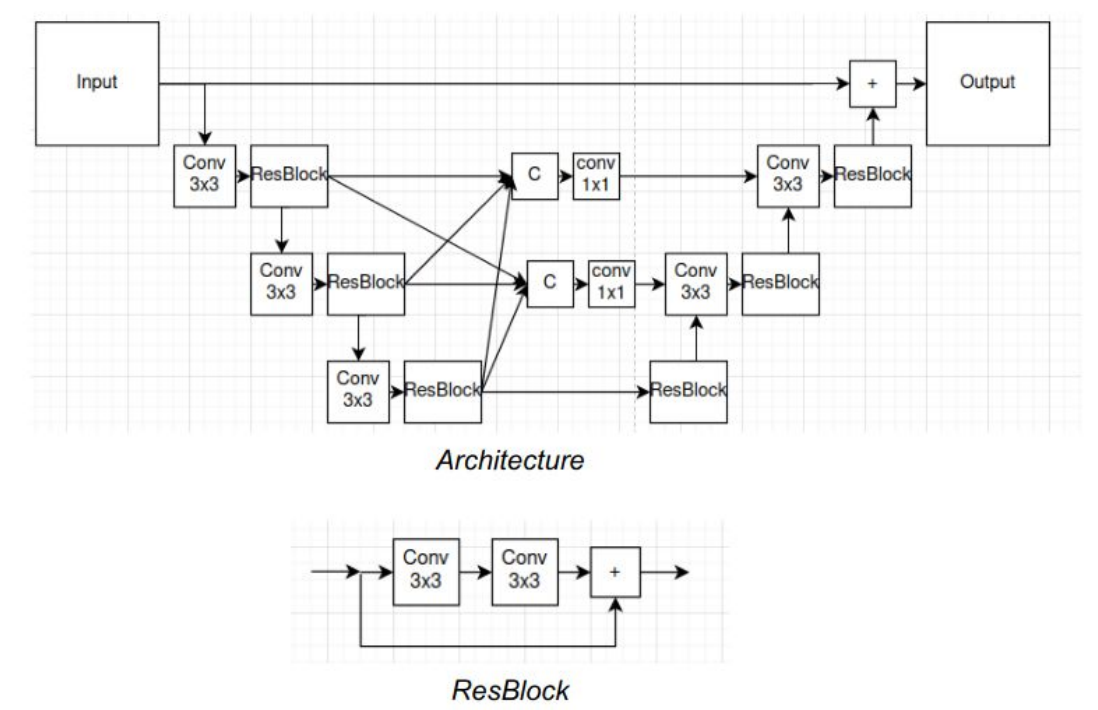
We decided to use neural networks to solve our problem, as they showed results better than classic approaches.

Models: adapted UNet(our), ResUNet(our), SwinIR

Losses: L1, L1+EdgeLoss, MSE

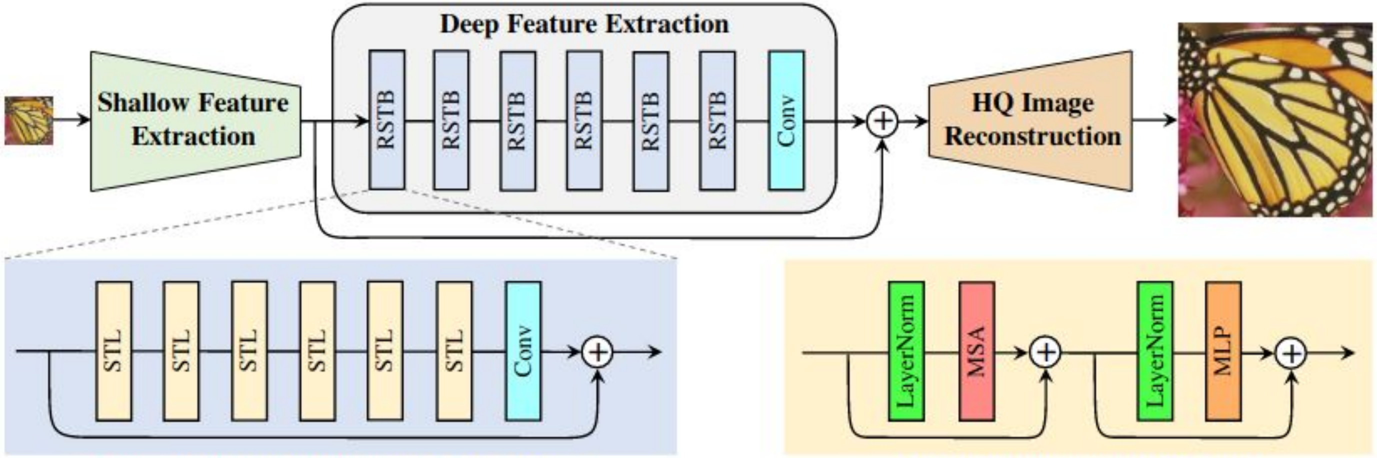
Training approaches: GAN, residuals prediction

Architectures



ResUNet

Architectures



(a) Residual Swin Transformer Block (RSTB)

(b) Swin Transformer Layer (STL)

Figure 2: The architecture of the proposed SwinIR for image restoration.

Experiments and achieved metrics

Initial dataset -

wandb

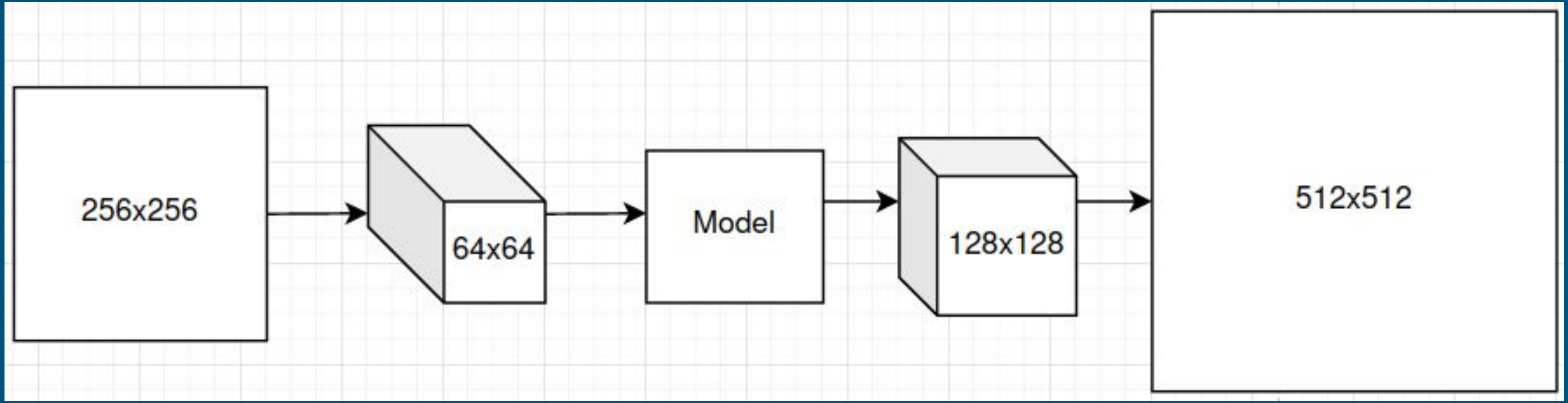
Model	Output	PSNR	SSIM
modified UNet	full image	21.0	0.65
modified UNet	residuals	29.0	0.83
ResUNet	full image	26.0	0.75
ResUNet	residuals	31.45	0.94

Updated dataset -

wandb

Model	Output	Batch Size	PSNR	SSIM
ResUNet	residuals	32	32.3	0.93
ResUNet	residuals	4	34.4	0.93
ResUNet	residuals	1	38.8	0.92
SwinIR	full image	1	36.1	0.92
SwinIR	residuals	1	40.0	0.92

Final upscaling pipeline

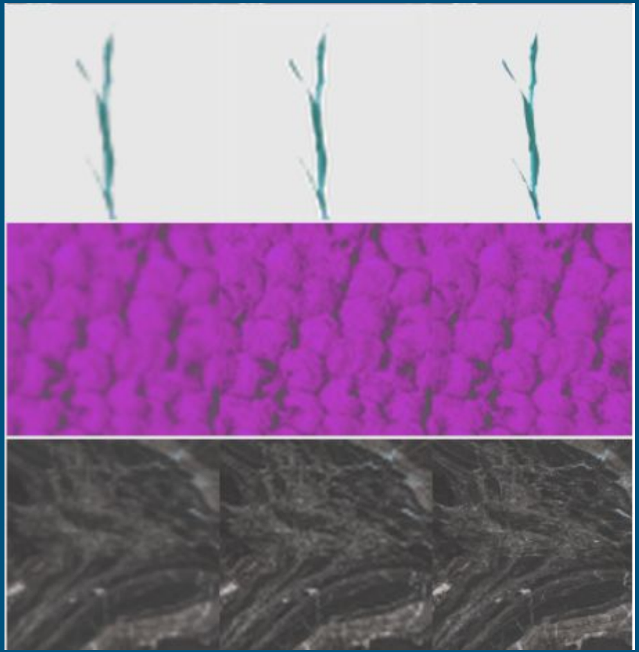


PSNR = 39

SSIM = 0.935

Visual results

Input Output Target



Practical conclusions

Tran image2image models with batch size 1

GANS is not the best choice when the target is specific

Give to NN work as small as possible as we did with residuals prediction

Conclusions

- Own approach to image upscaling
- Flexible and extendable pipeline for fast upscaling models training with all most used technics
- Tool for images upscaling, using models trained on our pipeline