Lightweight Super-Resolution - Deep Machine Learning

Jacob Eriksson, Sebastian Löf

Chalmers University of Technology
Electrical Engineering Department

CHALMERS UNIVERSITY OF TECHNOLOGY

Introduction – What Is Super-Resolution?

Problem: given a low-resolution (LR) image, we want to generate its high-resolution (HR) counterpart using AI.

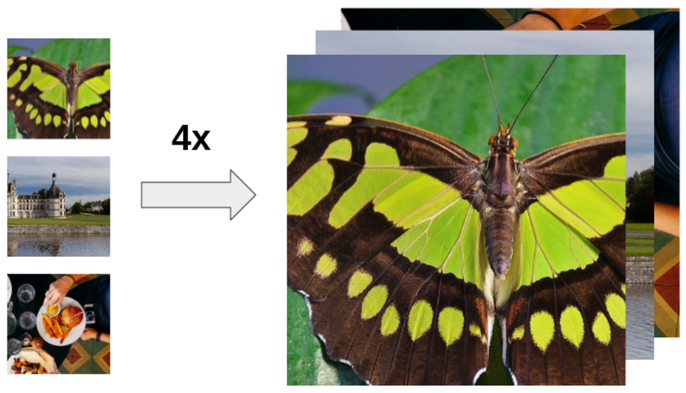

Figure 1: Perfect 4x upscaling of sample images from the DIV2k dataset [1].

- Traditionally handled by deterministic algorithms (e.g. bicubic interpolation), but they result in too blurry images.
- With DNNs we can learn the details present in HR images.
- We can train on any image dataset – the images are the targest. No labels are needed, only raw images!

A downside of DNNs is the increased computation time over traditional algorithms. Can we alleviate this concern?

Proposed Solution – MESRGAN

To generate super-resolution (SR) images of high fidelity, we follow the architectures of SRGAN [3] and ESRGAN [10].

This entails a GAN [4] architecture of:
- G_{θ_G}: an SR image generator in the form of a deep CNN.
- D_{θ_D}: a discriminator, also in the form of a CNN, that tells SR images apart from HR images.

Training is split into 2 stages, each with its own loss function.

1. First we optimize G_{θ_G} with respect to a pixel-wise content loss $\mathcal{L}_\mathrm{C}$ of reconstructed images until convergence:

$$\hat{\theta}_G = \arg\min_{\theta_G} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_\mathrm{C}(G_{\theta_G}(\mathbf{I}_\mathrm{LR}^{(i)}), \mathbf{I}_\mathrm{HR}^{(i)})$$

2. Then, we optimize G_{θ_G} with respect to a combination of:
   - the content loss $\mathcal{L}_\mathrm{C}$.
   - an adversarial $\mathcal{L}_\mathrm{A}$ loss from D_{θ_D}.
   - a feature-based loss $\mathcal{L}_\mathrm{F}$ using a pre-trained VGG19 classifier.

Intuition:
- $\mathcal{L}_\mathrm{C}$ encourages pixel-wise accuracy.
- $\mathcal{L}_\mathrm{A}$ encourages plausible images with fine details.
- $\mathcal{L}_\mathrm{F}$ encourages preservation of classification features.

In our proposed model, mobile ESRGAN (**MESRGAN**), we
- inherit the design of ESRGAN,
- ...but replace convolutional layers with the lightweight *bottleneck residual block* from MobileNetV2 [7].

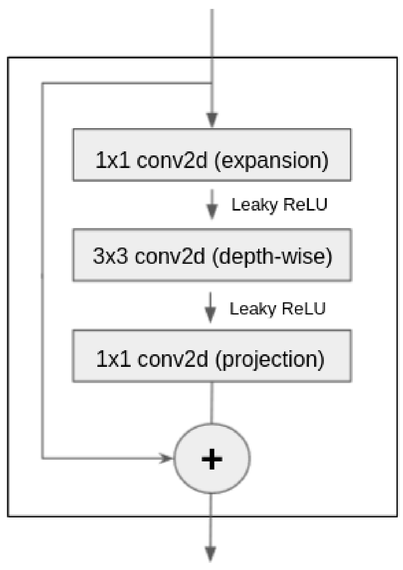

Figure 2: Bottleneck residual block from MobileNetV2, but adjusted to ESRGAN by using Leaky ReLU.

Experiments

We trained an instance of SRGAN [3], ESRGAN [10], and **MESRGAN** (ours) on a fusion of the DIV2K [1], Flickr2K [8], and OutdoorSceneTrain [9] image datasets.

The resulting HR images were then:
1. randomly cropped every training iteration into patches of fixed size.
2. normalized with respect to the dataset.
3. downscaled by a factor of 4 into corresponding LR images.

To compare models, we measured:
- forward pass runtime
- peak-signal-to-noise ratio (PSNR)
- structural similarity index measure (SSIM)
- perceptual index (PI) [2]

Results

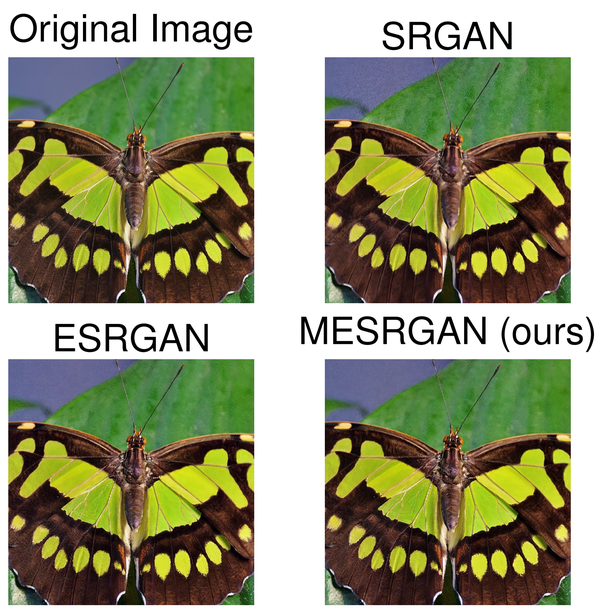

Figure 3: SR images generated by our implementations of SRGAN, ESRGAN, and MESRGAN (ours).

DIV2K-90	SRGAN	ESRGAN	MESRGAN
PSNR	**25.40**	24.52	24.39
SSIM	**0.96**	0.95	0.95
PI	3.57	**3.11**	3.13
RMSE	**13.61**	15.82	15.86

Runtime Benchmark (ms) **Batch size**			
1	26	33	155
2	50	62	299
4	134	126	596
8	279	266	1204
16	561	534	2402

Table 1: SRGAN vs. ESRGAN vs. MESRGAN in terms of evaluation scores (top) and runtime performance of a forward pass averaged over 100 runs (bottom).

From MESRGAN, we expected:
- faster runtime, since the introduced *bottleneck residual block* requires half the amount of multiplication operations as the replaced convolutional layers.
- image quality somewhere in-between the SRGAN and ESRGAN models.

However, as shown in Figure 3 and Table 1, only the latter was observed.

After some research, we found that:
- depth-wise convolution is significantly slower in PyTorch than standard convolution due to limited CUDA support [6].
- cuDNN 8.1.0 seems to have mitigated this issue [5], but PyTorch does not support this version yet. Using TensorFlow appears to be an option [6].

References

[1] E. Agustsson and R. Timofte.
Ntire 2017 challenge on single image super-resolution: Dataset and study.

[2] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor.
The 2018 pirm challenge on perceptual image super-resolution, 2019.

[3] Christian et al.
Photo-realistic single image super-resolution using a generative adversarial network, 2017.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio.
Generative adversarial networks, 2014.

[5] Nvidia Corporation.
cudnn release 8.1.0, 2021.

[6] PyTorch contributors.
Fp32 depthwise convolution is slow in gpu, 2021.

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen.
Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[8] Timofte et al.
Ntire 2017 challenge on single image super-resolution: Methods and results.

[9] X. Wang, K. Yu, C. Dong, and C. C. Loy.
Recovering realistic texture in image super-resolution by deep spatial feature transform, 2018.

[10] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, and X. Tang.
Esrgan: Enhanced super-resolution generative adversarial networks, 2018.

jaceri@student.chalmers.se, slof@student.chalmers.se