Clement Faber
cmfaber73@gmail.com

PROBA-V Super Resolution Challenge

kelvins.esa.int/proba-v-super-resolution

Incubit assignment report

The Proba-V Super Resolution challenge aims at leveraging the availability of multiple low-resolution satellite images of a given location, to try and approximate a higher resolution image ground truth-image that is obtained less frequently. The low-resolution images may be taken a long time apart, which implies varying lighting condition, weather conditions, obstruction by clouds… Such variation, except in the case of obstructions, could actually be beneficial to resolve a higher level of detail, which could make way for more sophisticated approaches than in the case of single image super resolution.

For this challenge, a ML-based approach seems like it could outperform the bicubic interpolation baseline, as the quantity of data available would train a model to make context specific guess in regard to the details to resolve.

I. Data aggregation

Each scene has between 9 and a maximum of 30 LR (Low resolution) images, and one image of its HR (High resolution) counterpart. The LR images possess varying levels of obstruction, making some worth more than another in term of coverage and useful data. Some kind of data aggregation strategy has to be adopted, especially for neural networks, as variable input size is not straight forward, and to weed out images that are not suitable.

I tried two main approaches for this task:

- Pick the N clearest images in regard to cloud coverage for each scene
- Perform some arithmetic aggregation of the LR images (mean, median), also taking occlusion into account.

The one that proved the most effective in my case was the following; *fuse all the LR images of a given scene into 2 image aggregates*. For a given pixel coordinate in a scene.

- Takes the mean value in all non-occluded images, and if all images are occluded then takes the mean of value in all images
- Takes the median of all images, occluded ones included.

The loose reasoning here is that the median will be resilient to obscured values, whereas the mean might be swayed more easily hence this different strategy to counteract this.

II. Architecture

Studying the state of the art in regard to multi image super resolution reveals that there aren't that many common methods that seems to perform well overall in our case, as they are mostly meant for video. As such, this data aggregation preprocessing allows for the

use Single Image super resolution methods, which has a much more extensive state of the art.

The strategy is to simply use the aggregates as different layers of a same input image, i.e. concatenated into a same input tensor.

On the use of GANs: Considering the task at hand, the aim is to help resolve correctly details that should be present in the image. Producing aesthetically pleasing images is not a priority. The use of GANs in super resolution tasks helps produce sharper and more satisfying images, but at the expense of ground truth accuracy, as the network will often hallucinate details, textures and patterns which makes for more aesthetically pleasing results. This is fine for images that are only meant for 'multimedia' use, but might be harmful in our case, hence why I am choosing not to pursue such methods.

My goal was to try and apply simple CNN architectures, and to tinker with parameters (kernel size, filter depth, number of layers…) as to see how far such a method can go. This choice was made with time constrains in mind, as a more sophisticated and thought out architecture could definitely make room for improvements.

Additionally, some single images SR methods directly upscale their input through bilinear interpolation to the expected resolution, before feeding it to the network (i.e. SRCNN [1]). Other methods simply use decovolution layers to upscale the image (i.e. FSRCNN [2]). Trying both type of approaches, this second method is actually more efficient in regards to training time, memory use, as well as yielding better results.

All the architectures I tried have been trained using either the 2 layer agregate, or a selection of the 5 clearest images (5 layers) as their inputs.

III. Training

In regards to what has been discussed in part II, training has been carried out with quite a few variations in regards to architecture

- Upscaled 384x384 input, with 2 or 5 layers, with variations on simple CNN architectures, while trying out SRCNN [1]
- 128x128 Input, with 2 or 5 layers, trying out :
    - [2] FSRCNN, an hourglass based network that actually reduces input size
    - [3] SRResNet, a deep residual network (hence might be more prone to overfitting on such a dataset)
    - Variations of the architectures above

As in the above papers, a default Adam optimizer was used, with a learning rate of 10^{-3} and parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$

In regards to code, the embiggen module [4], made by a competitor in this challenge, was used as it provided quite a few implementation of scoring functions for instance.

A batch size of 10 was used, and the score on a given batch was periodically evaluated to keep track of the training. 400 epochs were used on most runs, which corresponds to about 4 hours of training time on a single Nvidia Tesla V100-SXM2 GPU. Training was monitored using a wrapper for Tensorboard, making it compatible with PyTorch. Side by side image reconstruction was also produced each epoch to monitor qualitative progress.

As the dataset only provides ground truth high resolution images, scoring was done on the whole train set only, which is not ideal in regards to monitoring overfitting.

Here is a schematic of one of the first simple CNN architecture, with layer and kernel sizes annoted :

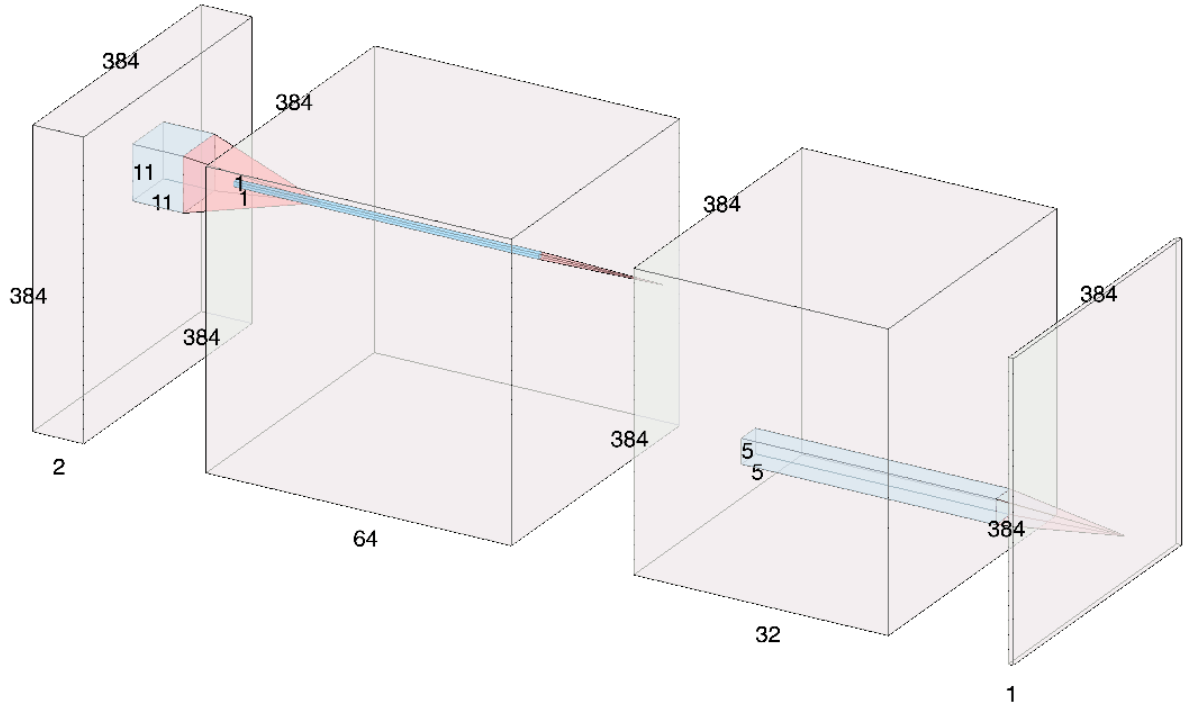

Fig. 1, A simple CNN model using an upscaled input

This obtained a score of 0.993522193563, a marginal improvement overbaseline, but nothing stellar in regards to the leaderboard results.

Visualizing many models, it appeared that batch norm or instance norm slows down training, which is probably due to the 2 NIR and RED bands present within the image data (and general variation in luminosity between scenes).

In orange, a model using normalization scheme, while in blue one without.

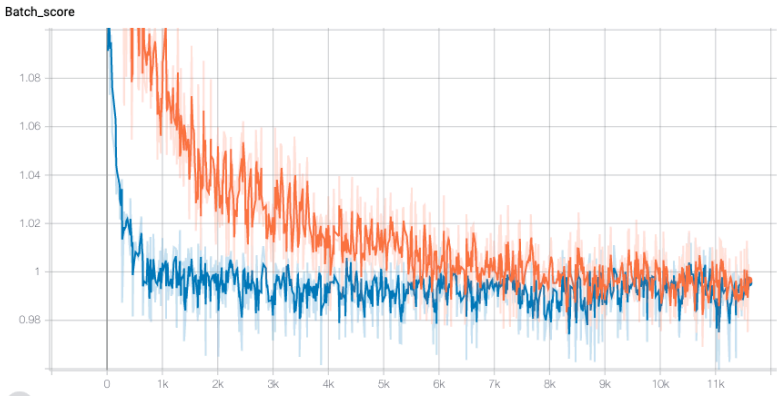

Fig. 2, Tensorboard graph of batch scores over time, smoothed curves

Models such as SRResNet prouved to be really unstable while training, and as probably really prone to over fitting with the high number of parameters.

In this figure on the right, we can visualize the unstable nature of SRResNet's training (about 4 hours), where the network would diverge back into a worse model, several times.

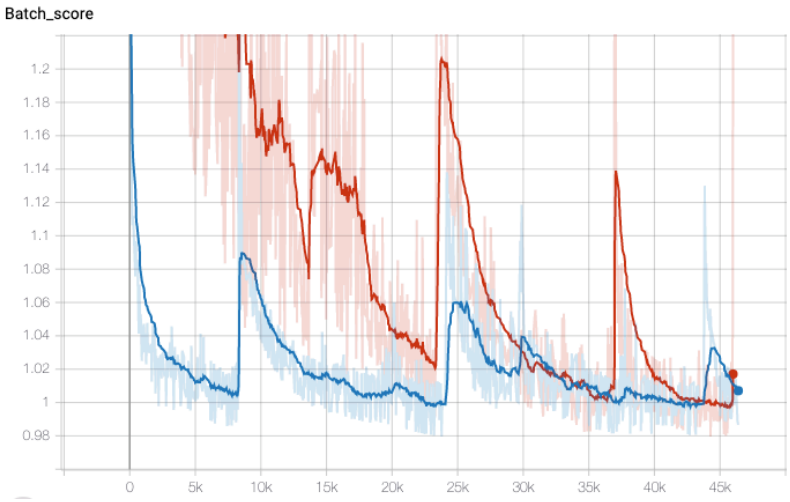

Fig. 3, Tensorboard graph of batch scores over time, smoothed curves

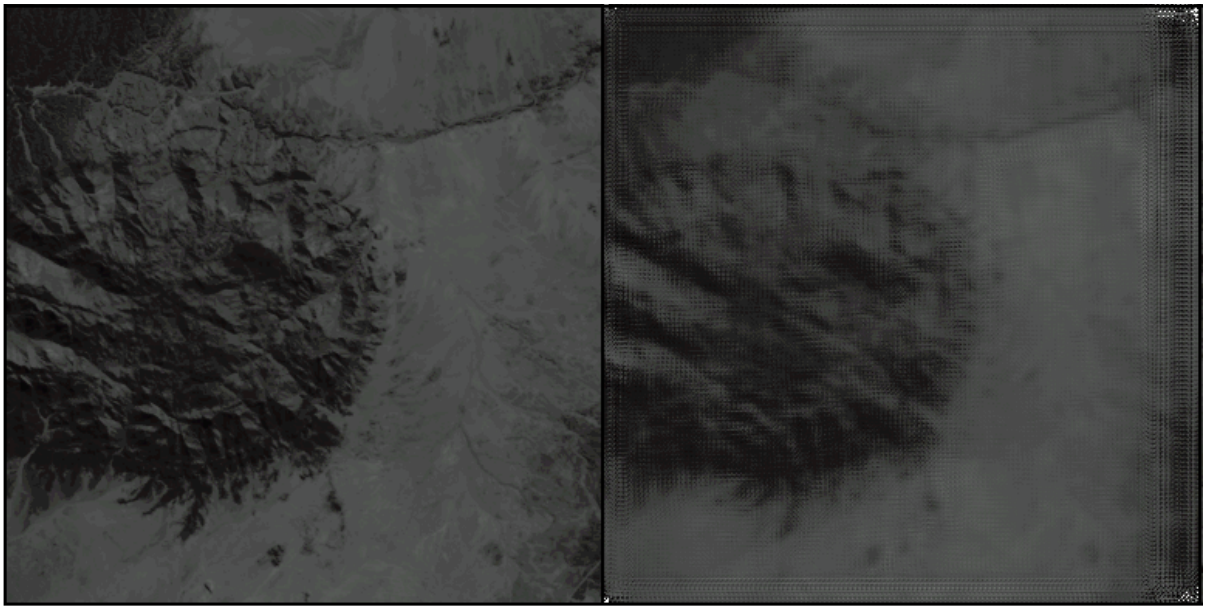

Fig. 4, Left, HR ground truth. Right, Super-Resolved SRResNet image

Even when scores approached 1.0 or below for SRResNet, the quality of the Super-Resolved image was actually plagued with artifacts (right and top borders on Fig.4 as an exemple), even when reducing the depth of the network.

IV. Current best network

The architecture that ended up performing the best was an implementation of FSRCNN [2] (upscaling is done through the use of deconvolution layers), using the 2-layer aggregates as input (the 5 clearest images as input would reach slightly higher scores).

Compared to the default implementation, I use (d=64, s=16) instead of (d=48, s=12) for layer depth. Keeping the number of mapping operation to m=8.

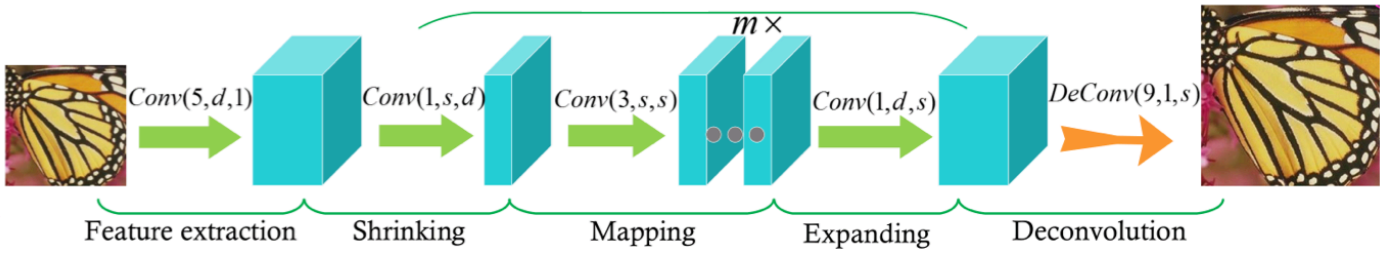

Fig. 5, diagram illustrating the different steps of FSRCNN

After optimizing this architecture, the score reached over the whole training set is **0.990704683785194,** which is again an improvement over baseline. Here are some of examples of Super-Resolved images by this network:

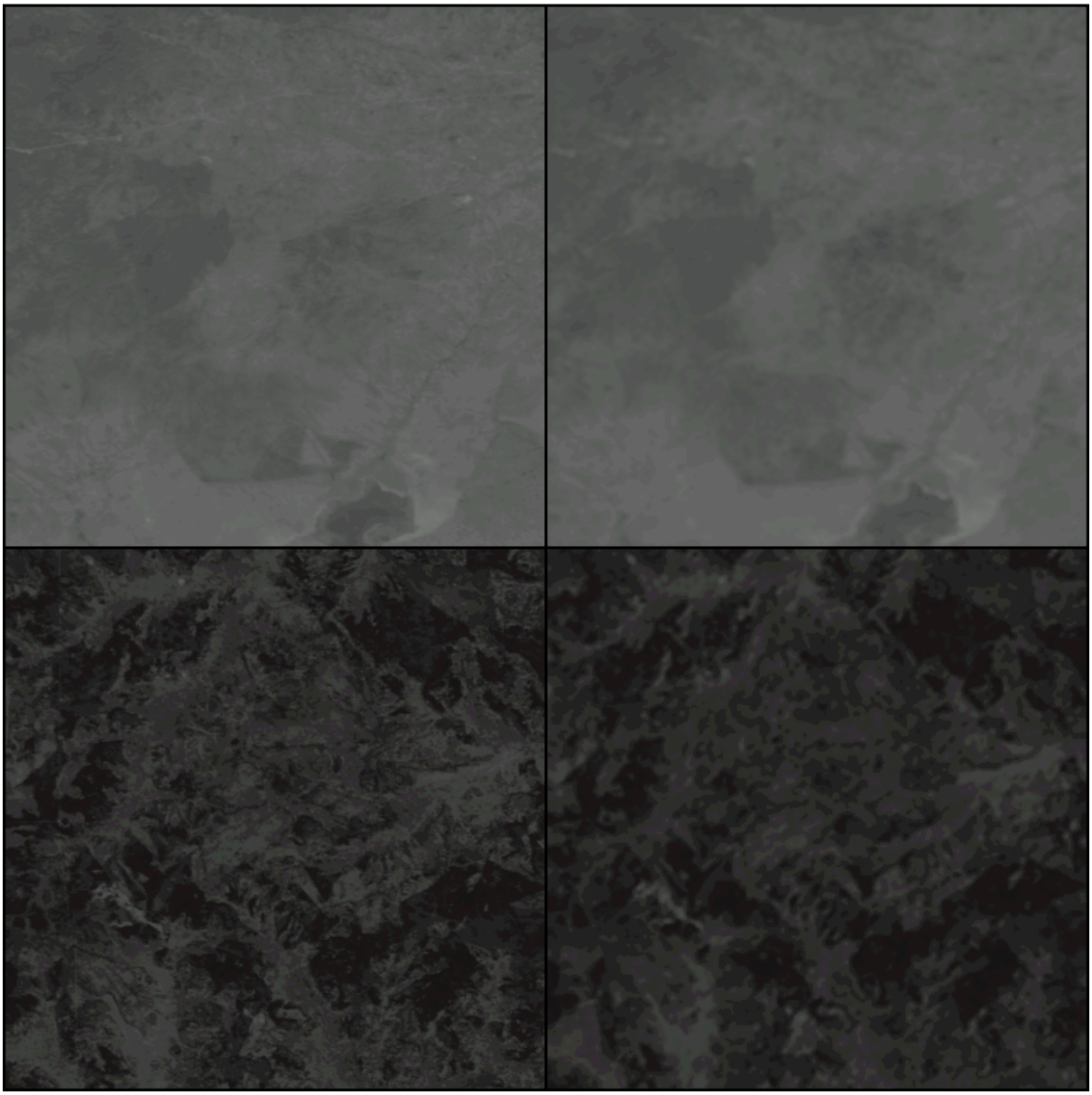

Fig. 6, FSRCNN-like architecture. Left, HR ground truth. Right, Super-Resolved images

Conclusion

With a simplistic approach in mind from the beginning (given current time constrains), my method would be ranked around the middle of the challenge leaderboard. But there is indeed a lot of room for improvement, as my score (0.9907) is actually fairly far from the best result submission (0.9473).

Obvious improvements to be made lie within the strategy to use and/or aggregate as much data as possible from the LR images. My simplistic way of feeding inputs to the network is probably a big bottleneck as to why my architectures aren't able to perform much better. Moreover, fusing processed LR images later on in the pipeline could actually be an even better strategy, as to not lose as many details through aggregation.

Links

kelvins.esa.int/proba-v-super-resolution

[1] Image Super-Resolution Using Deep Convolutional Networks, 2015

[2] Accelerating the Super-Resolution Convolutional Neural Network, 2016

[3] Enhanced Deep Residual Networks for Single Image Super-Resolution, 2017

[4] Embiggen module