Recovering High Frequency Geometric Details in Images and Surfaces

Ning Jin (njin19)

1. Introduction

The goal of the project is to apply deep learning models to recover high frequency *geometric* details in images and surfaces, in particular, folds and wrinkles in clothes and other deformable surfaces. Motivated by recent successes in generic natural image super-resolution [7, 9], we formulate this as an upsampling problem and leverage existing architectures to tackle these tasks.

2. Summary

There are three main components to the project:

- Replicate the networks in [7], and obtain pre-trained models on subset of ImageNet [12] (Section 3).
- Apply transfer learning to recover folds and wrinkles in clothes images in the fashion dataset Chictopia10K [8] (Section 4).
- Extend model to upsample height field surface patches in a synthetic ocean water dataset to add high frequency details (Section 5).

3. Network and Pre-training

3.1. Network choice

Generator Model We follow the reference paper [7] and use their generator architecture as shown in Figure 1 below. For all the experiments in this project, we use 16 residual blocks, where each block has Conv3×3(64)-BN-PReLU-Conv3×3(64)-BN-Element Sum. For the $4\times$ upsampling factor, this generator model has 1.5 million parameters, and for the $8\times$ upsampling factor, it has 1.7 million parameters. Since the model is fully convolutional, the number of parameters is not too large, making it flexible for any input size and also more manageable to train across different dataset. As in the reference paper, we will refer to this model as **SRRes**.

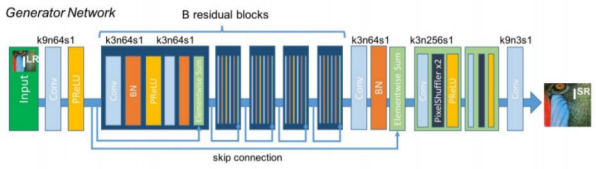


Figure 1: Upsampling model architecture. We use $B = 16$.

GAN-based Model As have been shown in many works, GANs [4] tend to help generative networks to produce less blurry results with higher visual quality. Therefore, as in [7], we also implement a discriminator network and make a GAN version of the model, which we will refer to as **SRGAN**. The disciminator network takes 96×96 image patches and outputs a binary label (real or fake) after a Sigmoid activation. Since it contains a dense layer (1024 units) in the classifier, the network is a lot bigger than the generator, with 23.6 million parameters.

3.2. Cost Function

MSE The content loss can be measured in several different ways, and the most straightforward metric is the pixel-wise (or point-wise) MSE loss. For input I with original size $W \times H$, the term is defined as

$$L_{\text{MSE}}(\hat{I}, I^{\text{gt}}) = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} (\hat{I}_{ij} - I_{ij}^{\text{gt}})^2. \tag{1}$$

For clarity, we henceforth suffix models trained with this cost function with -MSE.

Figure 2: $4\times$ upsampling results. Top: Ground Truth. Bottom: SRRes-MSE model output. These upsampled images already look quite realistic.

Perceptual Alternatively, as [5] argues, perceptual similarity may be better approximated with features from a trained network. A commonly used choice for this purpose is the VGG network [13] trained on the ImageNet [12]. For a given layer with feature map ϕ with dimension $W_\phi \times H_\phi$

on input I, we define the cost as

$$L_{\text{VGG}}(\hat{I}, I^{\text{gt}}) = \frac{1}{W_\phi H_\phi} \sum_{i=1}^{W_\phi} \sum_{j=1}^{H_\phi} ((\phi(\hat{I}))_{ij} - (\phi(I^{\text{gt}}))_{ij})^2. \quad (2)$$

As recommend by [7], we explore two different options for the feature map, one is the activation after the 2^{nd} convolution layer before the 2^{nd} pooling layer, and the other is the activation after the 4^{th} convolution layer before the 5^{th} pooling layer of VGG19. We suffix models trained with these cost functions with -VGG22 and -VGG54 respectively. For models trained with VGG perceptual loss, it is crucial to apply a regularization term to ensure stability and relative smoothness in the generated results, and we use the total variation with a weight of 2×10^{-8} as in [7].

Adversarial For the GAN-based model, we have an adversarial term in the cost function for the generator model, defined as

$$J_{\text{Adv}}(\hat{I}) = -\log(D(\hat{I})), \quad (3)$$

where D refers to the discriminator. Here we use the equivalent formula to allow for better gradient flow [4].

Discriminator When training the discriminator in the GAN-based models, we apply the standard binary cross-entropy loss on the real and the fake images.

3.3. Pre-training on ImageNet

We download and randomly sample the training set (uniformly across the 1000 classes) of the ImageNet dataset [12] to obtain 350K and 50K images for our training set and dev set. As in [7], we train our SRRes models with a batch size of 16 on 96×96 random crops for a total of 45 epochs (10^6 iterations) using the Adam optimizer [6] at a learning rate of 10^{-4}. Figure 2 shows the visual results of the trained SRRes-MSE. Table 1 summarizes the quantitative results on the trained SRRes models evaluated on the benchmarks Set5 [1], Set14 [18], ans BSD100 [10], using the evaluation procedures in [7]. We are able to mostly match the performance to our reference paper in terms of both PSNR and SSIM [16]. We notice slightly worse performance on our VGG22 version compared to the reference paper, and we conjecture that this is due to differences in the pretrained VGG across different deep learning frameworks (we use PyTorch [11] whereas the reference paper uses Theano [15] and Lasagne [3]).

	SRRes-MSE		SRRes-VGG22	
Set5	Reference	Ours	Reference	Ours
PSNR	32.05	32.08	30.51	30.04
SSIM	0.9019	0.9016	0.8803	0.8693
Set14	Reference	Ours	Reference	Ours
PSNR	28.49	28.66	27.19	26.83
SSIM	0.8184	0.8203	0.7807	0.7643
BSD100	Reference	Ours	Reference	Ours
PSNR	27.58	27.58	26.32	26.13
SSIM	0.7620	0.7617	0.7191	0.7048

Table 1: Quantitative evaluation results on our trained SRRes models. We are able to match the reference performance.

A note on quantitative evaluation The evaluation procedure in [7] is as following: remove border strip, convert to YUV4MPEG video stream, and then using daala tools [2] [1] to compute PNSR and SSIM on the Y-channel. We do not find this procedure particularly convincing nor necessary; nonetheless, we replicated this pipeline to match their results. In addition, we have an alternative procedure to compute these metrics purely in the RGB image space, and Table 2 shows the comparison of quantitative metrics between our and their procedures. In the following sections, we will only list the results from this our pipeline.

Set5	Ref Eval		Our Eval	
SRRes-MSE	Ref	Ours	Ref	Ours
PSNR	32.05	32.08	30.05	30.08
SSIM	0.9016	0.9019	0.8658	0.8662

Table 2: Comparison of evaluation procedures on quantitative results. Our procedure generally leads to slightly lower numbers.

Note that we do not pre-train the GAN-based version of our models. It is less clear that the discriminator would learn useful features that could be transfered to our tasks of interest - recovering high frequency gemetric details in images and surfaces, and we decide to train the discriminator of the GAN-based model from scratch for the fashion dataset, which we think might be a easier task than pre-training on the ImageNet and then fine-tuning, considering the notorious instability of GANs. For the surface dataset, we do not employ a GAN.

4. Recover Folds and Wrinkles in Images of Clothes

Figure 3: Sample images from the Chictopia10K dataset. Top: original images. Bottom: images with the background removed.

[1] https://github.com/xiph/daala (We used commit number d10a875).

Since there is no direct dataset on folds and wrinkles in clothes images, we resort to a fashion dataset that contains a rich set of images with different types of clothing.

4.1. Dataset

The fashion dataset we use here is the Chictopia10K [8]. It contains fashion images with a focus on clothes and accessories. Furthermore, it is annotated with detailed semantic segmentation labels. For instance, see Figure 3 for some sample images with and without the background removed. One can see that the semantic segmentation provided by the dataset is roughly correct for the most part, although sometimes missing small regions in the foreground.

We split the dataset 80/20/20 into 14125 training images, 1766 dev images, and 1766 test images. During training and evaluation, we mask out the background to be black (pixel value is 0) based on the semantic label. When obtaining random crops on the training image, we sample a few times and find the window with maximum fraction of foreground pixels in order to direct the network's attention to the objects of interest. While the absolute minimum fraction is 25%, in reality it is often much higher ($> 90\%$).

4.2. Experiments

We start with our pre-trained SRRes-MSE model, and compare the following fine-tuning options: SRRes-MSE, SRRes-VGG22, and SRGAN-VGG54. For each model, we use the Adam optimizer [6] starting with learning rate of 10^{-4}, and decreasing to 10^{-5} when the cost plateaus. Training takes between 10^5 to 5×10^5 iterations. The baseline is bicubic upsampling.

Quantitative results We compute the PSNR and SSIM on the test set. Since our model takes the masked image as input to predict the masked output, we compute the metrics using a mask as well (otherwise, we would be artificially inflating performance). Table 3 shows our quantitative results across different models. It is interesting to see the significantly lower evaluation results on this dataset even after fine-tuning, illustrating the challenges for learning on fashion images compared to generic natural images. We hypothesize that the textures in clothing and accessories pose difficulty for the model. As expected, SRRes-MSE does the best in quantitative measurements, whereas SRRes-VGG22 and SRGAN-VGG54 do worse since they do not directly optimize for the pixel-wise losses.

	Baseline	SRRes-MSE	SRRes-VGG22	SRGAN-VGG54
PSNR	20.75	24.4	23.6	22.58
SSIM	0.7198	0.7833	0.7523	0.7087

Table 3: Quantitative evaluation on the test set of the fashion dataset. Note that metrics are computed on the masked images.

Figure 4: Visual comparison on a test example. SRGAN-VGG54 gives the best visual result in terms of recovering fine wrinkles.

Qualitative results See Figure 4 for a comparison of all the models. Although SRRes-MSE gives the best quantitative performance, it does not successfully learn to recover missing high freqyuency geometric details. While SRRes-VGG22 should theoretically perform better than SRRes-MSE, in reality we do not notice a significant difference in visual quality. On the other hand, SRGAN-VGG54 produces quite promising results in recovering folds and wrinkles in clothing that the downsampled images completely lost, although it certainly is still lacking compared with the ground truth in terms of the level of details. Figure 5 shows another example from the test set where the SRGAN-VGG54 model has learned to recover fine details. One can clearly see that the model has learned to recover fine wrinkle patterns on the sleeve of the leather jacket and the pants.

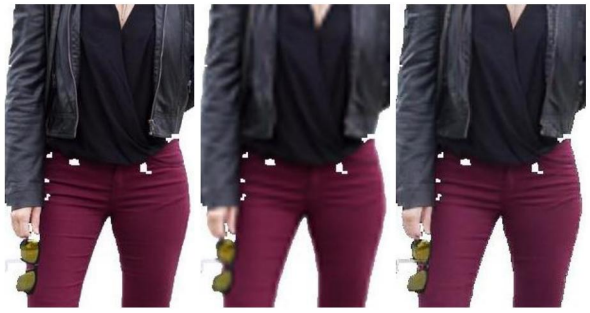


Figure 5: SRGAN-VGG54 has learned to recover a lot of fine details missing from the downsampled image, for example, on the sleeve and the pants.

4.3. Remarks

We have several remarks for the experiment on this dataset. First, unfortunately, even though we are mostly interested in the high frequency *geometric* details such as those shown in Figures 4 and 5, the images in the dataset also contain a large quantity of rich high frequency texture patterns, which the model is forced to learn to upsample as well. This distracts the model from focusing only on the geometric features and thus we believe lowers the visual quality on our task of interest. In addition, another source

of distraction comes from the human face and skin shown in the pictures. For future work, if possible, we would like to perform a more fine-grained background removal that also mask out the body parts not covered in clothing.

Second, we note overall color differences in the output from SRGAN-VGG54 compared with output from the other models. It is curious to see the slightly less saturated colors from this model, perhaps because VGG54 features are rather high-level and the cost function do not enforce exact pixel-wise color matching.

5. Resolve High Frequency Details in Surfaces

Compared with images, 3D geometries are cleaner to work with in a sense - there is no complex image formation process, no lighting variation, no texture patterns, etc. On the other hand, they also pose more challenges due to the higher dimension and the arbitraty topology. To make it more tractable, we experiment with 2.5D surfaces in this project, i.e., height field surfaces.

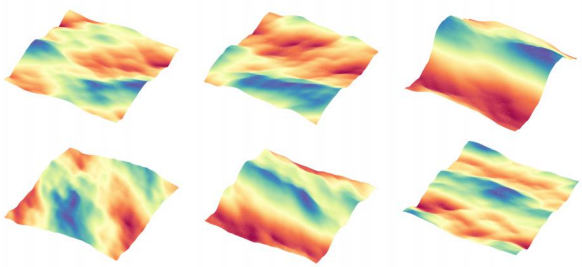


Figure 6: Sample surfaces from the synthetic surface dataset. The surfaces contain rich patterns across a large frequency range.

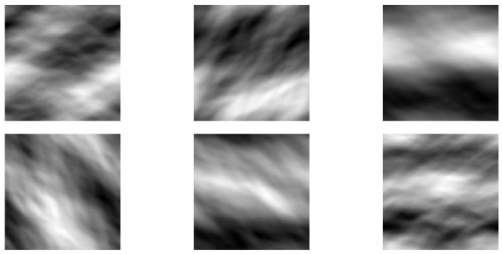


Figure 7: Sample depth images generated from corresponding surfaces shown in Figure 6.

5.1. Dataset

We sample from the Phillips spectrum [14] to generate ocean water patches with interesting surface shapes. This synthetic dataset contains 20K deformable surfaces represented as height fields, each with size 128×128. Samples in the dataset contain a rich set of features across different frequencies, and therefore serve as good testing cases for a surface upsampling method. Figure 6 shows some sample surface patches.

We split the dataset 80/20/20 into 16000 training images, 2000 dev images, and 2000 test images. For each surface, we convert the height field into a grayscale image in the range [0, 255]. In order to use the pre-trained SRRes-MSE model for $4\times$ upsampling, we also convert the grayscale image to a standard RGB image with 3 channels. Since the sample size is uniform and is quite manageable at 128×128, we do not apply a random crop and directly train on the whole depth image. See Figure 7 for the generated depth images from samples in Figure 6.

5.2. Experiments

We start with $4\times$ upsampling, where the bicubic baseline already does a decent job. We then fine-tune the pre-trained SRRes-MSE model on the training set of this surface dataset, and are able to achieve a significant improvement over the baseline both quantitatively and qualitatively.

Encouraged by the success from $4\times$ up-sampling on these surface patches, we proceed to experiment with $8\times$ up-sampling as well. Since we do not have a pre-trained with $8\times$ factor on the ImageNet, we train the model from scratch for 100 epochs using the Adam optimizer [6] at a learning rate of 10^{-4} with batch size 16. Here we also modify the network to input and output grayscale images (1 channel instead of 3), which is indeed more appropriate for this task.

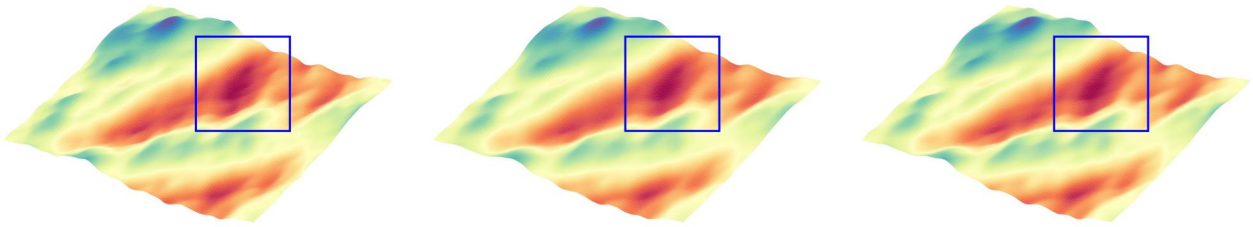
Due to the different nature of surface geometry upsampling compared with image super-resolution, we use only the most straightforward MSE cost for these tasks. It is unclear if perceptual loss is appropriate under this setting, and even it is, VGG19 feature map almost certainly does not seem appropriate for this application, and thus we do not pursue those VGG-based models.

Quantitative Evaluation Since our goal is to upsample height field surfaces, it is reasonable to evaluate directly on the height field MSE. Since we have scaled each depth image to be in the range of $[0, 255]$, here we re-scale the height field using the mean scaling of the dataset. However, since our training uses depth images, we will also include evaluation in the image space with PSNR and SSIM.

Table 4 shows the quantitative evaluation results. Our models do a lot better than their baseline counterparts for both $4\times$ and $8\times$ upsampling across all metrics, and the improvement is especially large for the more challenging $8\times$ upsampling task.

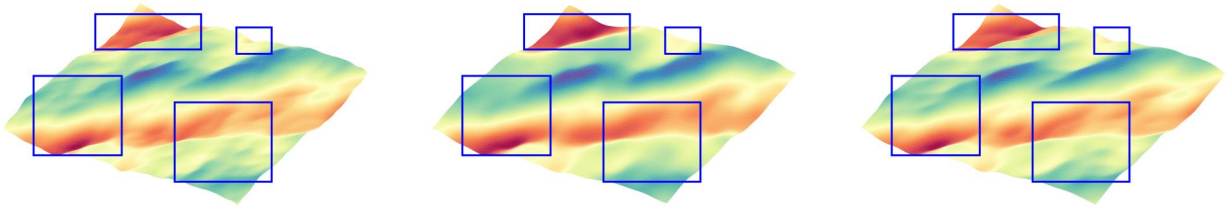
	4x		**8x**	
	Bicubic	Model	Bicubic	Model
PSNR	42.0	46.2	32.8	40.2
SSIM	0.983	0.993	0.917	0.975
Height Field MSE	6.22e-4	2.34e-4	5.30e-3	8.66e-4

Table 4: Quantitative evaluation on the synthetic surface test set.



GT Bicubic Model

Figure 8: $4\times$ upsampling result on a random test example. Both bicubic and model output look quite good, but model output shows finer details, for example, in the blue box region.

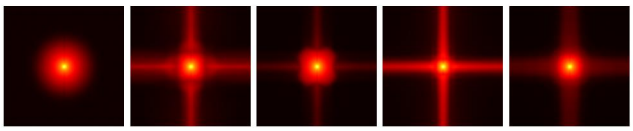


GT Bicubic Model

Figure 9: $8\times$ upsampling result on a random test example. Model output clearly has more details than the bicubic baseline, although also still visibly lacking compared to the ground truth.

Qualitative results We show the result on 4x upsampling in Figure 8. Out model has small improvement over the bicubic result on high frequency details. We show the result on 8x upsampling in Figure 9. Here the improvement is much more obvious. Nevertheless, we do note that the model output is still missing some high frequency details present in the ground truth samples.

Frequency Analysis We further perform a frequency space analysis to compare model outputs with baseline outputs and the ground truth samples on the test set. Figure 10 shows the power spectra in log-scale. Note that we normalize the power spectrum for each example (sum to 1) before averaging over them. Figure 10 verifies that our model has managed to learn to add higher frequency details back to the surfaces, and that the output spectrum more closely approximates the true spectrum than the bicubic baseline.



GT Bicubic $4\times$ Model $4\times$ Bicubic $8\times$ Model $8\times$

Figure 10: Log-scale normalized power spectrum averaged over the test set. The plots demonstrate that the model has learned to recover higher frequency details to some extent, though not perfectly, and that it has significant improvement over the baseline.

6. Conclusion and Future Work

In conclusion, we have explored recovering and adding fine high frequency *geometric* details in images and surfaces. We are able to apply deep learning models to this problem and achieve promising results on different datasets such as fashion images and synthetic height field surfaces. The choice of a fully convolutional deep residual network has helped us maintain flexibility and a moderate model size, thereby facilitating training and generalization. Transfer learning has played a key role in the upsampling of fashion images, where features learned from generic natural images are also useful for this task. For future work, apart from those mentioned in Section 4.3, we would like to extend the model to arbitrary true 3D surfaces.

7. Acknowledgements

We would like to thank the course instructors for their efforts in making this class. We would also like to thank them for their help on this project. Finally, we would like to thank the instructors and Amazon for the AWS credits given to us for training the models.

8. Code

GitHub private repository link: GitHub repository [2]. Invitation has been sent to cs230-stanford. The code to generate the surface patches is part of the PhysBAM library [3], and not included in the repository.

[2] https://github.com/njin19/super_resolution

[3] http://physbam.stanford.edu/

References

[1] M. Bevilacqua, A. Roumy, C. Guillemot, and M. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *British Machine Vision Conference, BMVC 2012, Surrey, UK, September 3-7, 2012*, pages 1–10, 2012.

[2] T. J. Daede, N. E. Egge, J. Valin, G. Martres, and T. B. Terriberry. Daala: A perceptually-driven next generation video codec. *CoRR*, abs/1603.03129, 2016.

[3] S. Dieleman, J. Schlter, C. Raffel, E. Olson, S. K. Snderby, D. Nouri, et al. Lasagne: First release., Aug. 2015.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[5] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 694–711, Cham, 2016. Springer International Publishing.

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[7] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[8] X. Liang, C. Xu, X. Shen, J. Yang, S. Liu, J. Tang, L. Lin, and S. Yan. Human parsing with contextualized convolutional neural network. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1386–1394, Dec 2015.

[9] X. Mao, C. Shen, and Y. Yang. Image restoration using convolutional auto-encoders with symmetric skip connections. *CoRR*, abs/1606.08921, 2016.

[10] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423 vol.2, 2001.

[11] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[14] J. Tessendorf, C. C, and J. Tessendorf. Simulating ocean water. 1999.

[15] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[16] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[17] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg. Parsing clothing in fashion photographs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3570–3577, June 2012.

[18] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In J.-D. Boissonnat, P. Chenin, A. Cohen, C. Gout, T. Lyche, M.-L. Mazure, and L. Schumaker, editors, *Curves and Surfaces*, pages 711–730, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.