Neural Differential Equations for Single Image Super-Resolution

Teven Le Scao
Hugging Face *
teven@huggingface.co

Abstract

Although Neural Differential Equations have shown promise on toy problems such as MNIST, they have yet to be successfully applied to more challenging tasks. Inspired by variational methods for image restoration relying on partial differential equations, we choose to benchmark several forms of Neural DEs and backpropagation methods on single image super-resolution. The adjoint method previously proposed for gradient estimation has no theoretical stability guarantees; we find a practical case where this makes it unusable, and show that discrete sensitivity analysis has better stability. In our experiments[1]*, differential models match the performance of a state-of-the art super-resolution model.*

1. Introduction

A growing body of work is exploring the connection between deep learning and dynamical systems. Historically, the objective has been to help solve differential equations with deep learning [10]. Inversely, recent work has borrowed from dynamical systems literature to train neural networks [16]. Neural Ordinary Differential Equations [4] are such an approach that approximates functions with a differential equation parameterized by a neural network. They take the input as the boundary condition of an initial value problem and integrate the equation to produce the output.

This paper attempts to benchmark different forms of Neural ODEs, which have mostly been tested on early image classification datasets such as MNIST or CIFAR for now. Since good performance there does not necessarily transfer to harder problems, we use supervised single image super-resolution (SR). We choose SR as partial differential equations (PDEs) have been used on that task [5][6]. We can then train a Neural ODE parameterized by a convolutional neural network to learn the discretized form of an adequate PDE, considering the action of the CNN like a finite difference approximation of the equation.

Our contributions are threefold. First, we test Neural ODEs variants and corresponding optimization techniques [8][9]. We find that dimension augmentation and time-dependence have positive impacts, and that discrete reverse sensitivity analysis is significantly more stable than the adjoint method advocated in [4]. Then, we compare Neural ODEs with the residual networks usually employed for super-resolution. We find they're competitive with state-of-the art systems while using a fraction of the parameter size, although this does not translate to a speed increase. Finally, we investigate the benefits of their computational adaptivity, and observe a correlation between the amount of computation they use on an image and the performance gain of residual networks on that image as they grow deeper.

Figure 1. Our model learns a partial differential equation parameterized by a neural network that smoothly transforms the low-resolution input (left) into a high-resolution reconstruction (right).

2. Related work

Supervised single-image super-resolution is a task where low- and high-resolution image pairs are provided as training data to learn a mapping from the low-resolution space to the high-resolution space. Residual networks [12] are typically used. Those are trained with a combination of pixel-wise loss, image classifier features [7] (those should be the same for the generated image and the original) and adversarial networks [13]. We replace residual generators with Neural ODEs, and stay in the non-adversarial case.

There has been extensive work on the relationship between **residual networks and dynamical systems**. Most of it leverages the connection to build principled residual architectures [14][3][20][25]. In contrast, Neural ODEs [4] introduce a concurrent architecture that continuously de-

*Work performed while at the Czech Technical University in Prague
[1]Code available at github.com/TevenLeScao/BasicSR

forms inputs according to a differential equation for a fixed amount of integration time to produce the output. We study its behaviour and performance on a real-world task.

Image restoration tasks, such as denoising [22], deblurring [18], inpainting [21] have historically been solvable with **total variation methods** [19] which naturally lead to PDEs. Super-resolution has also used differential methods [6][5], so Neural ODEs, which learn differential equations to fit the data in a supervised way, are a promising class of models for the task. Previous work that learns equation parameters [6] typically uses constraints on the filters in order to leverage the exact equivalence between convolution filters and spatial differentiation [2]. In contrast, we use a neural network approach where we overspecify the model and use larger datasets and limited a priori knowledge.

3. Approach

3.1. Architecture

We use as reference the residual-in-residual-dense-blocks (RRDB) network [23], a state-of-the art architecture that has proved effective for both adversarial and non-adversarial training. We replace the main residual module with a differential module as shown in fig. 2. Inputs go through a first convolution to expand the RGB channels to F filters. Then, a residual network or Neural ODE iterates over the image in filter space to perform feature extraction. Finally, after upsampling to the final resolution and applying a convolution in filter space, the output convolution returns to RGB channel space to yield the image. We use LeakyRELU activations and no batch normalization.

In conventional models, the model core consists of hundreds of convolution layers arranged in an intricate architecture. In RRDB networks, the basic unit is a dense block of five convolutions with a residual connection bypassing it. Those are then arranged into groups of three units with residuals. The final model contains a pre-specified number of those groups: 5 in our low-data experiment, 20 in the high-data one, for a total of 75 to 300 layers of varying sizes. In contrast, the Neural ODE uses only one set of stacked convolutions, which we call the ODE function: 2 in the low-data regime and 7 in the high-data one. We initialize a time variable t at $t_0 = 0$ and an ODE state u_0 at the input value. At every step k, the ODE solver takes in the current state $u_k \approx u(t_k)$ and computes the time increment Δt and state increment Δu according to the ODE function. The exact calculation depends on the solver; here, the Dormand-Prince method makes several function calls between t and $+\Delta t$. The integration then proceeds forward with $t_{k+1} = t + \Delta t$ and $u_{k+1} = u_k + \Delta u$, until we reach a pre-specified $t = T_{final}$, at which point the ODE block outputs the final value $u(T)$ for the rest of the network. We concatenate a time channel that contains the current integration time t to the filter channels to model time dependence [4]. The differential system uses considerably less parameters: our high-data differential image generator, for example, is 25 times lighter than its conventional equivalent.

Finally, we also experiment with dimension augmentation [8], where extra dimensions are concatenated to the latent space in order to allow possible trajectories that would have to cross in the lower-dimensional space and ANODEs [9], which implement a checkpointing scheme to ensure the stability of the gradient signal.

3.2. Backpropagation methods

A Neural ODE attempts to map inputs $\mathbf{x}(0)$ to outputs $\mathbf{x}(T)$ that approximate target $\mathbf{y}$ by the continuous action of dynamical system $\dot{\mathbf{x}} = f(\mathbf{x}, t, \theta)$, with f a neural network of parameters θ. Training it is finding the minimum of

$$loss(\theta) = d(\mathbf{y}, \mathbf{x}(0) + \int_0^T f(\mathbf{x}(t), t, \theta) dt) \qquad (1)$$

with regard to θ, where d is the distance function we're using for training and T is an arbitrary integration time. In order to use gradient descent, we must derive through the integral. Fortunately, if we consider this as an optimal control problem, we can use use sensitivity analysis to compute the derivative of the integral term in (1) with regard to θ.

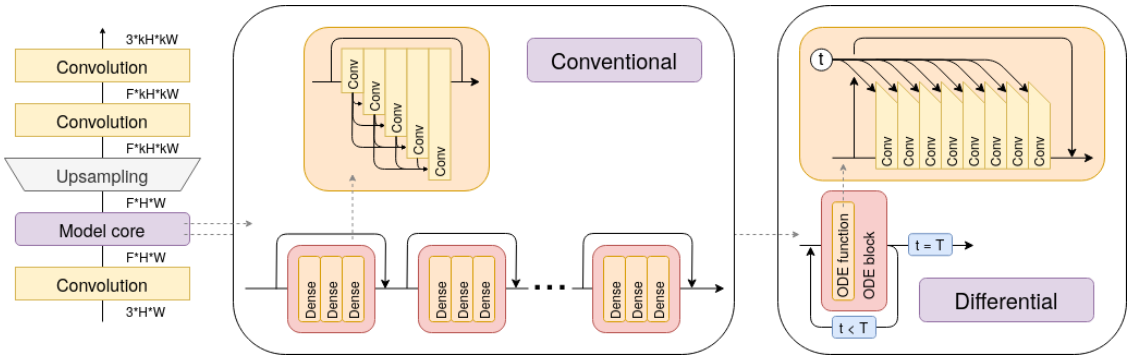

Figure 2. The super-resolution network. A first convolution extends the number of channels to F filters. Then, a model core performs feature extraction in filter space. This feature map is upsampled to the desired scale, then a set of convolutions produces the final.

When the number of parameters of the model is large compared to the dimension of the space of the differential equation, as here, adjoint sensitivity is the most efficient technique [24]. It first computes the trajectory of the system normally, then solves another differential equation system, the adjoint, that computes the derivatives backwards in time along that trajectory. It is the method used in [4] which allows for constant memory cost as a function of depth when used without checkpointing, ie saving intermediary values of the forward pass. However, in this case, due to error accumulation, the trajectory of the adjoint equation may completely diverge from the trajectory of the forward pass if there's no additional stability guarantees (for example, a reversible integrator, which may have worse accuracy).

Another possibility is discrete reverse sensitivity: simply propagating the gradients through the internal operations of the solver using the chain rule. This is equivalent to backpropagation, or, in automatic differentiation terms, reverse accumulation. It requires access to the solver operations, rather than treating it like a black box like the adjoint method does, and does not seem to scale as well on performance benchmarks [17]. However, it does not have the instability risk of the adjoint method. As most of the work comparing those methods has been performed on problems stemming from physical systems, which might not be representative of the equations that appear in machine learning applications, we will test and compare both methods.

4. Experiments

We conducted experiments using five usual super-resolution datasets: BSD [15] and DIV2K [1] for training, and Urban 100, General 100 and Set 14 for testing. Preprocessing measures and hyperparameters are presented in appendix A, and visual results in appendix C.

4.1. Architecture search on the BSD dataset

As this smaller dataset allows an extensive grid architecture search, we identify the best-performing differential system on the BSD dataset before moving on to DIV2K. We train baseline neural ODEs (NODEs), augmented ODEs and ANODEs, all with an ODE function of two convolution layers. Wherever possible, we test time-dependent (e.g. concatenating a channel with the current integration time to the image tensor to model a time-dependent PDE) and autonomous (e.g. no time dependence) ODE blocks. Finally, both discrete and adjoint training are presented. We compare with the state-of-the-art RRDBNet generator from [23]. Table 4 presents test PSNRs for each system.

In this low-data regime, differential systems are competitive against RRDBNet, with Augmented time-dependent ODEs beating it by an encouraging 0.10 PSNR, although the visual quality difference is minimal. We attribute the slightly lower PSNR of the conventional model mostly to overfitting, as higher validation scores for the state-of-the-art CNN do not translate to better PSNR on the testing dataset, Urban 100. In contrast, those are consistent for the augmented time-dependent ODE. Finally, adjoint optimization does not seem to have a consistent effect. Baseline NODEs may lead to less stable adjoint gradients than Augmented ODEs, explaining the difference in performance.

Table 1. PSNR results training on BSD and testing on the concatenation of all datasets. Time-dependency, adjoint optimization, and dimension augmentation have a generally positive effect. Augmented time-dependent ODE are the best-performing models, with the adjoint-optimized version beating RRDB by 0.43 PSNR. Results are presented split by dataset in appendix B.

Architecture		**Discrete**	**Adjoint**
RRDBNet [23]		25.33	-
ANODE [9]		24.40	-
NODE	Autonomous	24.15	25.57
	Time-dependent	25.67	25.61
Augmented [8]	Aut.	25.24	25.67
	Time-dep.	25.19	**25.76**

4.2. Full training on the DIV2K dataset

We keep using the augmented time-dependent architecture. As in the low-data regime, the difference in PSNR is not noticeable without zooming in. Images with a significant difference in score one way or another tend to be hard images with black-and-white striped patterns, where the worse-performing model generates stripes of the wrong color or not aligned, which causes a heavy PSNR penalty without influencing perceived image quality noticeably.

Notably, **no adjoint-optimized model trained successfully**. As mentioned in section 3.2, for an arbitrary solving method and differential equation, there is no guarantee that the adjoint problem we need to solve in the backward pass is tractable. Previous work has trained image classifiers on MNIST or CIFAR without stability issues. However, in our experiments on this task with larger models, there is invariably a training batch for which it becomes intractable. Stability guarantees seem necessary to impose bounds on the number of steps the adjoint pass requires: we've observed the backward pass require up to 10^5 function evaluations (i.e. model calls) before stopping for lack of time as this had made it around 2000 times longer than usual.

Table 2. PSNR results training on the DIV2K dataset. Training with the adjoint method is intractable.

Architecture	**Discrete**	**Adjoint**
RRDBNet	**26.88**	-
Augmented time-dependent ODE	26.69	*divergent*

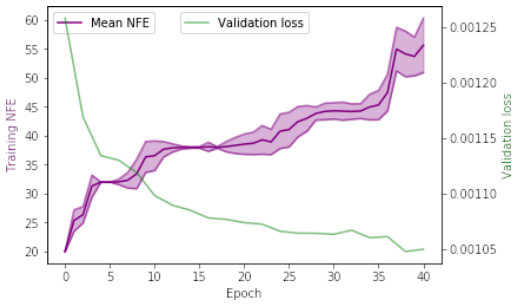

Figure 3. Evolution of the mean number of function evaluations per validation pass, with a band of a standard deviation.

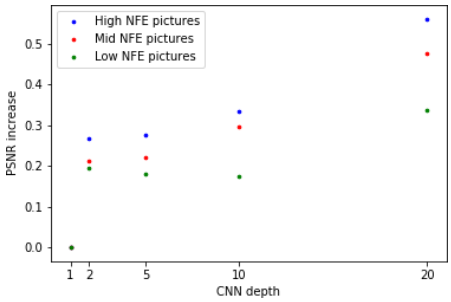

Figure 4. Performance increase with more blocks for the RRDB architecture on testing images that required low, medium, and high numbers of function evaluations. The increase in performance is higher on the pictures that required more function evaluations.

4.3. Adaptive computation

One of the strategies of differential equation solvers that Neural ODEs aim to benefit from is adaptive computation. Indeed, modern solvers are able to vary the number of function evaluations (NFE) depending on the difficulty of the problem and the desired error tolerance. Previous work has found that NFE increases consistently as the model is training [4] and interpreted it as proof that the modeled ODE gets harder as it learns to fit the data, although dimension augmentation reduces that effect [8]. Fig. 3 plots this increase over the training of our DIV2K differential model. At the end of training, NFE is up to between 50 and 80 evaluations (corresponding to 8 to 13 steps). The variance is especially interesting as we're using batches of 16 random patches, which we expected to have a homogeneizing effect.

In order to investigate whether the difficulty of the ODE correlates with the difficulty of the task, we look at the NFE for each model call rather than over whole epochs. In order to measure the difficulty of an image, we train four traditional residual models of varying (1, 2, 5, 10) depths in addition to our best performing 20-blocks RRDB network. One of the main premises of Neural ODEs is that additional solver method calls are equivalent to the model emulating a deeper conventional network. If that adaptive computation plays a role in ODE performance, there should be a link between which images prompt the solver to require more function evaluations and which images see the biggest performance increase with more CNN layers.

At test time, since a single image corresponds to around 40 patches of heterogeneous difficulty (as foreground tends to be easy and background to be hard), function evaluations are more clustered. We find that most (75 out of 100) images demand 8 solver steps, whereas 12 images require 7 steps and 13 images require 9. We dub those the medium, low and high NFE groups respectively and plot for each the increase in performance for RRDB as CNN depth increases in fig. 4. As PSNR does not always increase with depth for the low NFE images while the high NFE pictures see it consistently increase, the differential equation solver does seem to require more calls in cases where it helps performance, although the spread is smaller than at training time.

Table 3. Test PSNR performance, model size, and epoch time for various conventional networks and the ODE model.

Model	Parameters	Epoch time	PSNR
1 RRDB blocks	$0.87 * 10^6$	2mn 46s	23.80
2 RRDB blocks	$1.6 * 10^6$	3mn 20s	24.02
5 RRDB blocks	$3.7 * 10^6$	5mn 2s	24.03
10 RRDB blocks	$7.3 * 10^6$	8mn 4s	24.09
20 RRDB blocks	$15 * 10^6$	13mn 57s	**24.27**
7-layer ODE	$0.57 * 10^6$	10-17mn	**24.10**

As shown in table 3, the differential model achieves similar performance to traditional residual networks with only a fraction of the parameters; however, its speed is comparable, especially at the end of training when it requires more function evaluations. In a sense, Neural ODEs can be seen as a hypernetwork [11], with a limited set of parameters defining an extensive set of computations.

5. Conclusion

We have presented a novel model for single image super-resolution that applies a neural differential equation instead of discrete pre-defined residual convolutional layers. We have also shown that the lack of stability guarantees of the continuous adjoint method materializes into practical intractability, and found that differential models use adaptive computation, allocating more function evaluations to problems that can benefit. This allowed us to match the performance a state-of-the art super-resolution with only a fraction of the parameters, although with similar time cost.

The author would like to thank Tomáš Mikolov, Josef Šivic, and Germán Kruszewski for their guidance and advice. This work was supported by the European Regional Development Fund under the project IMPACT no. CZ.02.1.01/0.0/0.0/15 003/0000468.

References

[1] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[2] Jian-Feng Cai, Bin Dong, Stanley Osher, and Zuowei Shen. Image restoration: Total variation, wavelet frames, and beyond. *Journal of the American Mathematical Society*, 25(4):1033–1089, 2012.

[3] Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. Multi-level residual networks from dynamical systems view. In *International Conference on Learning Representations*, 2018.

[4] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6571–6583. Curran Associates, Inc., 2018.

[5] Yuanxu Chen, Yupin Luo, and Dongcheng Hu. A general approach to blind image super-resolution using a PDE framework. In Shipeng Li, Fernando Pereira, Heung-Yeung Shum, and Andrew G. Tescher, editors, *Visual Communications and Image Processing 2005*, volume 5960, pages 1819 – 1830. International Society for Optics and Photonics, SPIE, 2005.

[6] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, June 2017.

[7] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *CoRR*, abs/1602.02644, 2016.

[8] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *arXiv preprint arXiv:1904.01681*, 2019.

[9] Amir Gholaminejad, Kurt Keutzer, and George Biros. ANODE: Unconditionally accurate memory-efficient gradients for neural odes. In *IJCAI*, 2019.

[10] R. González-García, R. Rico-Martínez, and I.G. Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers and Chemical Engineering*, 22:S965 – S968, 1998. European Symposium on Computer Aided Process Engineering-8.

[11] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. *CoRR*, abs/1609.09106, 2016.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[13] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[14] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *CoRR*, abs/1710.10121, 2017.

[15] 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 *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[16] Michael Muehlebach and Michael I. Jordan. A dynamical systems perspective on Nesterov acceleration. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 4656–4662, 2019.

[17] Christopher Rackauckas, Yingbo Ma, Vaibhav Dixit, Xingjian Guo, Mike Innes, Jarrett Revels, Joakim Nyberg, and Vijay Ivaturi. A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions. *CoRR*, abs/1812.01892, 2018.

[18] Leonid Rudin, Pierre-Luis Lions, and Stanley Osher. *Multiplicative Denoising and Deblurring: Theory and Algorithms*, pages 103–119. Springer New York, New York, NY, 2003.

[19] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation-based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259 – 268, 1992.

[20] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *CoRR*, abs/1804.04272, 2018.

[21] Carola-Bibiane Schönlieb. *Partial Differential Equation Methods for Image Inpainting*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2015.

[22] Seongjai Kim. PDE-based image restoration: a hybrid model and color image denoising. *IEEE Transactions on Image Processing*, 15(5):1163–1170, May 2006.

[23] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. ESRGAN: enhanced super-resolution gen-

erative adversarial networks. *CoRR*, abs/1809.00219, 2018.

[24] Hong. Zhang and Adrian. Sandu. FATODE: A library for forward, adjoint, and tangent linear integration of ODEs. *SIAM Journal on Scientific Computing*, 36(5):C504–C523, 2014.

[25] Mai Zhu and Chong Fu. Convolutional neural networks combined with Runge-Kutta methods. *CoRR*, abs/1802.08831, 2018.

A. Datasets, evaluation and training procedures

The BSD dataset was originally gathered for image segmentation tasks and contains 300 photos of variable, small size, around 400x400 pixels large. Those are separated into 200 training images and 100 testing ones. After using bicubic downsampling to generate factor 4x low-resolution images and dividing the images into smaller patches for training following the pre-processing pipeline in [23], we obtain a training set of 1200 patches. By contrast, the DIV2K dataset is much richer; it contains 900 images of higher starting resolution, separated in 800 training and 100 validation images. We apply the same process as previously and obtain a training set of 33152 patches, 28 times bigger. The testing datasets are not pre-processed.

In order to present fair comparisons with the state of the art, we re-use many of the hyperparameters of [23] for training, such as learning rates (2.10^{-4}), number of filters (64), weight decay (none), and the image pre-processing procedure parameters. We differ by implementing validation-based learning rate decay: if a number (3 in our case) of validation results do not improve over the previous best result, we reduce the learning rate, and stop training when it reaches a minimum value. We use the default *torchdiffeq* solving parameters.

B. Low-data results split by dataset

C. Image results and comparisons

Table 4. PSNR results training on BSD split by dataset. Performance is consistent across datasets.

Architecture			**Set 14**	**General 100**	**Urban 100**	**Concatenation**
RRDBNet [23]			25.62	27.84	22.79	25.33
ANODE [9]			24.77	26.78	21.97	24.40
NODE	Autonomous	Discrete	24.55	26.45	21.79	24.15
		Adjoint	25.82	28.11	23.00	25.57
	Time-dependent	Discrete	25.89	28.21	23.10	25.67
		Adjoint	25.85	28.15	23.03	25.61
Augmented	Autonomous	Discrete	25.47	27.76	22.69	25.24
		Adjoint	25.88	28.20	23.11	25.67
	Time-dependent	Discrete	25.47	27.60	22.75	25.19
		Adjoint	**25.96**	**28.30**	**23.20**	**25.76**

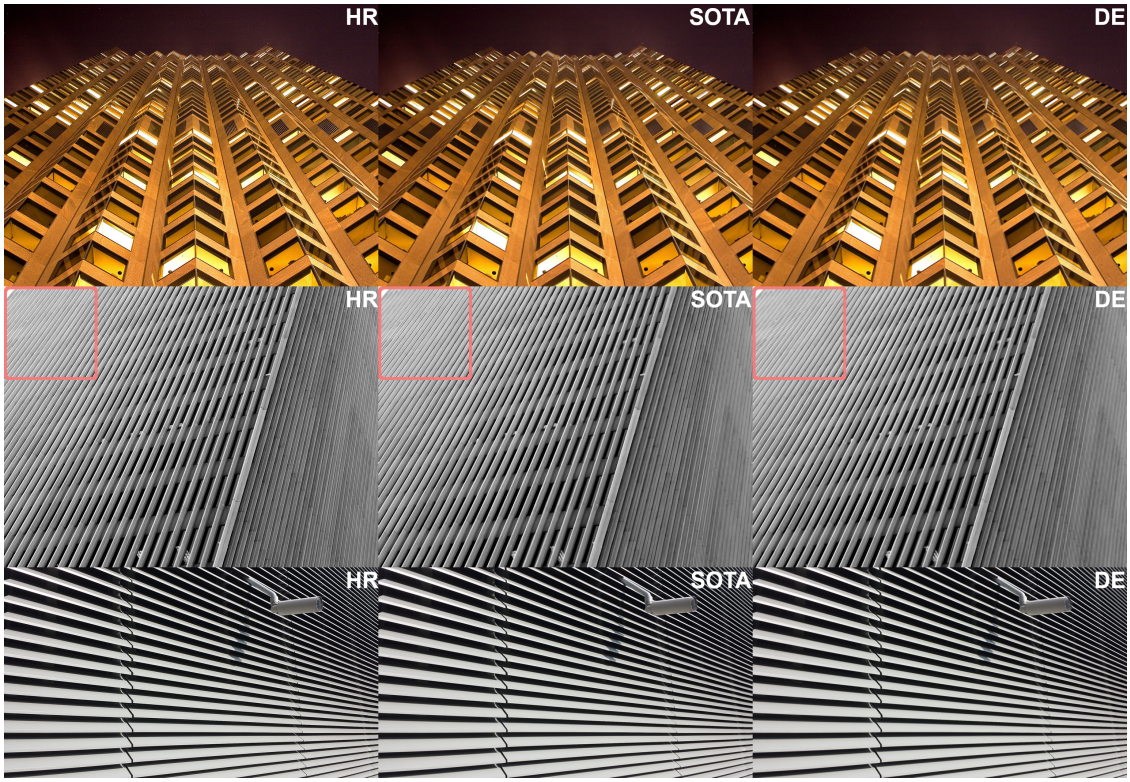

Figure 5. A comparison between images produced by the best-performing differential (center) and state-of-the-art (right) models and their high-resolution target (left). The first row shows an image where both models have typical PSNRs (24.28 for the ODE, 24.47 for the state-of-the-art). The second row shows the image where the ODE outperforms the state-of-the-art the most (1.47 PSNR difference). The third row shows the image where the state-of-the-art outperforms the ODE the most (1.08 difference) High score discrepancy images on both sides are hard black-and-white striped images where minute anti-aliasing or alignment differences are heavily penalized; this high variation in PSNR does not correlate well with human perception.