

Noise2Noise: Learning Image Restoration without Clean Data

Jaakko Lehtinen^{1,2} Jacob Munkberg¹ Jon Hasselgren¹ Samuli Laine¹ Tero Karras¹ Miika Aittala³
Timo Aila¹

Abstract

We apply basic statistical reasoning to signal reconstruction by machine learning — learning to map corrupted observations to clean signals — with a simple and powerful conclusion: under certain common circumstances, it is possible to learn to restore signals without ever observing clean ones, at performance close or equal to training using clean exemplars. We show applications in photographic noise removal, denoising of synthetic Monte Carlo images, and reconstruction of MRI scans from undersampled inputs, all based on only observing corrupted data.

1. Introduction

Signal reconstruction from corrupted or incomplete measurements is an important subfield of statistical data analysis. Recent advances in deep neural networks have sparked significant interest in avoiding the traditional, explicit a priori statistical modeling of signal corruptions, and instead *learning* to map corrupted observations to the unobserved clean versions. This happens by training a regression model, e.g., a convolutional neural network (CNN), with a large number of pairs (\hat{x}_i, y_i) of corrupted inputs \hat{x}_i and clean targets y_i and minimizing the empirical risk

$$\operatorname{argmin}_{\theta} \sum_i L(f_{\theta}(\hat{x}_i), y_i), \quad (1)$$

where f_{θ} is a parametric family of mappings (e.g., CNNs), under the loss function L . We use the notation \hat{x} to underline the fact that the corrupted input $\hat{x} \sim p(\hat{x}|y_i)$ is a random variable distributed according to the clean target. Training data may include, for example, pairs of short and long exposure photographs of the same scene, incomplete and complete k-space samplings of magnetic resonance images, fast-but-noisy and slow-but-converged Monte Carlo renderings of a synthetic scene, etc. Significant advances

¹NVIDIA ²Aalto University ³MIT CSAIL. Correspondence to: Jaakko Lehtinen <jlehtinen@nvidia.com>.

have been reported in several applications, including Gaussian denoising, de-JPEG, text removal (Mao et al., 2016), super-resolution (Ledig et al., 2017), colorization (Zhang et al., 2016), and image inpainting (Iizuka et al., 2017). Yet, obtaining clean training targets is often difficult or tedious. A noise-free photograph requires a long exposure; full MRI sampling is slow enough to preclude dynamic subjects, etc.

In this work, we observe that under suitable, common circumstances, we can *learn to reconstruct signals from only corrupted examples, without ever observing clean signals*, and often do this just as well as if we were using clean examples. As we show below, our conclusion is almost trivial from a statistical perspective, but in practice, it significantly eases learning signal reconstruction by lifting requirements on the availability of clean data.

2. Theoretical background

Assume that we have a set of unreliable measurements (y_1, y_2, \dots) of the room temperature. A common strategy for estimating the true unknown temperature is to find a number z that has the smallest average deviation from the measurements according to some loss function L :

$$\operatorname{argmin}_z \mathbb{E}_y \{L(z, y)\}. \quad (2)$$

For the L_2 loss $L(z, y) = (z - y)^2$, this minimum is found at the arithmetic mean of the observations:

$$z = \mathbb{E}_y \{y\}. \quad (3)$$

The L_1 loss, the sum of absolute deviations $L(z, y) = |z - y|$, in turn, has its optimum at the median of the observations. The general class deviation-minimizing estimators are known as M-estimators (Huber, 1964). From a statistical viewpoint, summary estimation using these common loss functions can be seen as ML estimation by interpreting the loss function as the negative log likelihood.

Training neural network regressors is a generalization of this point estimation procedure. Observe the form of the typical training task for a set of input-target pairs (x_i, y_i) , where the network function $f_{\theta}(x)$ is parameterized by θ :

$$\operatorname{argmin}_{\theta} \mathbb{E}_{(x,y)} \{L(f_{\theta}(x), y)\}. \quad (4)$$

Indeed, if we remove the dependency on input data, and use a trivial f_θ that merely outputs a learned scalar, the task reduces to (2). Conversely, the full training task decomposes to the same minimization problem at every training sample; simple manipulations show that (4) is equivalent to

$$\operatorname{argmin}_\theta \mathbb{E}_x \{ \mathbb{E}_{y|x} \{ L(f_\theta(x), y) \} \}. \quad (5)$$

The network can, in theory, minimize this loss by solving the point estimation problem separately for each input sample. Hence, the properties of the underlying loss are inherited by neural network training.

The usual process of training regressors by Equation 1 over a finite number of input-target pairs (x_i, y_i) hides a subtle point: instead of the 1:1 mapping between inputs and targets (falsely) implied by that process, in reality the mapping is multiple-valued. For example, in a superresolution task (Ledig et al., 2017) over all natural images, a low-resolution image x can be explained by many different high-resolution images y , as knowledge about the exact positions and orientations of the edges and texture is lost in decimation. In other words, $p(y|x)$ is the highly complex distribution of natural images consistent with the low-resolution x . Training a neural network regressor using training pairs of low- and high-resolution images using the L_2 loss, the network learns to output the average of all plausible explanations (e.g., edges shifted by different amounts), which results in spatial blurriness for the network’s predictions. A significant amount of work has been done to combat this well known tendency, for example by using learned discriminator functions as losses (Ledig et al., 2017; Isola et al., 2017).

Our observation is that for certain problems this tendency has an unexpected benefit. A trivial, and, at first sight, useless, property of L_2 minimization is that on expectation, the estimate remains unchanged if we replace the targets with random numbers whose expectations match the targets. This is easy to see: Equation (3) holds, no matter what particular distribution the y s are drawn from. Consequently, the optimal network parameters θ of Equation (5) also remain unchanged, if input-conditioned target distributions $p(y|x)$ are replaced with arbitrary distributions that have the same conditional expected values. *This implies that we can, in principle, corrupt the training targets of a neural network with zero-mean noise without changing what the network learns.* Combining this with the corrupted inputs from Equation 1, we are left with the empirical risk minimization task

$$\operatorname{argmin}_\theta \sum_i L(f_\theta(\hat{x}_i), \hat{y}_i), \quad (6)$$

where both the inputs and the targets are now drawn from a corrupted distribution (not necessarily the same), conditioned on the underlying, unobserved clean target y_i such that $\mathbb{E}\{\hat{y}_i|\hat{x}_i\} = y_i$. Given infinite data, the solution is the

same as that of (1). For finite data, the variance of the estimate is the average variance of the corruptions in the targets, divided by the number of training samples (see supplemental material for proof). Note that in typical denoising tasks the effective number of training samples is large, even if the number of image pairs is limited, because every pixel neighborhood contributes.

In many image restoration tasks, the expectation of the corrupted input data *is* the clean target that we seek to restore. Low-light photography is an example: a long, noise-free exposure is the average of short, independent, noisy exposures. The above findings indicate, that in these cases, and bar potential numerical issues, we can dispose of the clean targets entirely, as long as we are able to observe each source image twice – a task that is often significantly less costly than acquiring the clean target.

Similar observations can be made about other loss functions. The L_1 loss recovers the median of the targets, meaning that neural networks can be trained to repair images with significant (up to 50%) outlier content, again only requiring access to pairs of such corrupted images.

In the next sections, we present a wide variety of examples demonstrating that these theoretical capabilities are also efficiently realizable in practice, allowing one to “blindly” learn signal reconstruction models on par with state-of-the-art methods that make use of clean examples — using precisely the same training methodology, and often without appreciable drawbacks in training time or performance.

3. Practical experiments

We now experimentally study the practical properties of noisy-target training and identify various cases where clean targets are not needed. We start with standard, simple noise distributions in Sections 3.1 and 3.2, and continue to the much harder, analytically intractable Monte Carlo noise in image synthesis in Section 3.3. In Section 3.4, we observe that image reconstruction from sub-Nyquist spectral samplings in magnetic resonance imaging (MRI) can be learned from corrupted observations only.

3.1. Additive Gaussian noise

We will first study the effect of corrupted targets using additive Gaussian noise. This is a simple distribution, from which we can draw samples and thus generate an unlimited amount of synthetic training data by corrupting clean images. As the noise has zero mean, we use the L_2 loss for training to recover the mean.

Here our baseline is a recent state-of-the-art method “RED30” (Mao et al., 2016), a 30-layer hierarchical residual network with 128 feature maps, which has been demon-

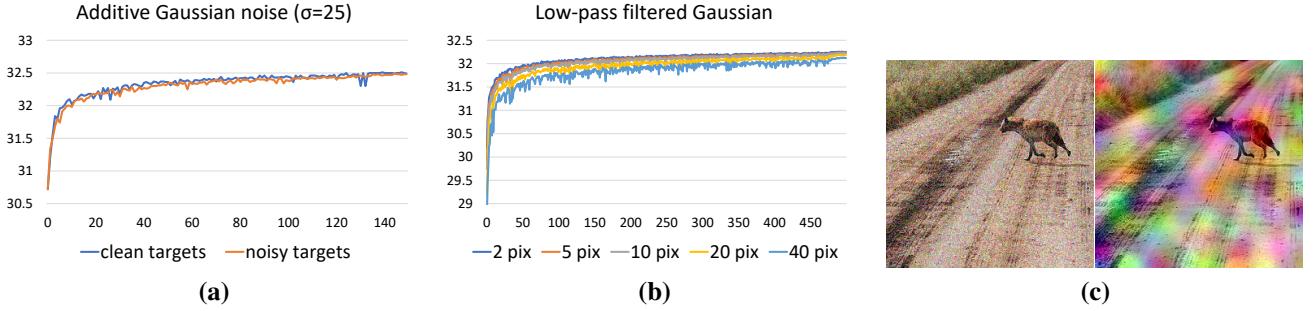


Figure 1. Denoising performance (decibels in KODAK dataset) as a function of training epoch for additive Gaussian noise. (a) Clean and noisy targets lead to very similar convergence speed and eventual denoising quality. (b) If we furthermore blur the noisy targets using a Gaussian filter of different widths (stddev in pixels), we observe that lower frequency noise is more persistent, but in all cases the network converges towards a similar denoising quality. (c) A noisy target ($\sigma = 25$) and the same target with the noise blurred with a 10 pixel filter.

Table 1. PSNR results from three test datasets KODAK, BSD300, and SET14 for Gaussian, Poisson, and Bernoulli noise. The comparison methods are BM3D, Inverse Anscombe transform (ANSC), and deep image prior (DIP).

	Gaussian ($\sigma=25$)			Poisson ($\lambda=30$)			Bernoulli ($p=0.5$)		
	clean	noisy	BM3D	clean	noisy	ANSC	clean	noisy	DIP
Kodak	32.50	32.48	31.82	31.52	31.50	29.15	33.01	33.17	30.78
BSD300	31.07	31.06	30.34	30.18	30.16	27.56	31.04	31.16	28.97
Set14	31.31	31.28	30.50	30.07	30.06	28.36	31.51	31.72	30.67
Average	31.63	31.61	30.89	30.59	30.57	28.36	31.85	32.02	30.14

strated to be very effective in a wide range of image restoration tasks, including Gaussian noise. We train the network using 256×256 -pixel crops drawn from the 50k images in the IMAGENET validation set. We furthermore randomize the noise standard deviation $\sigma \in [0, 50]$ separately for each training example, i.e., the network has to estimate the magnitude of noise while removing it (“blind” denoising).

We use three well-known datasets: BSD300 (Martin et al., 2001), SET14 (Zeyde et al., 2010), and KODAK¹. As summarized in Table 1, the behavior is qualitatively similar in all three sets, and thus we discuss the averages. When trained using the standard way with clean targets (Equation 1), the RED30 network achieves an average quality of 31.63 ± 0.02 dB with $\sigma = 25$. The confidence interval was computed by training 5 networks from different random initializations. The widely used benchmark denoiser BM3D (Dabov et al., 2007) gives ~ 0.7 dB worse results. When we modify the training to use noisy targets (Equation 6) instead, the denoising performance remains equally good. Furthermore, the training converges just as quickly, as shown in Figure 1a. This leads us to conclude that clean targets are completely unnecessary in this application. This perhaps surprising observation holds also with different networks and network capacities. Figure 2a shows an example result.

For all further tests we switch from RED30 to a shallower U-Net (Ronneberger et al., 2015) that is roughly $10 \times$ faster to train and gives similar results (-0.2 dB in Gaussian noise). The U-Net architecture and training parameters are described in the supplemental material.

Convergence speed Clearly, every training example asks the network to perform an impossible task: there is no way it could succeed in transforming one instance of the noise to another. Consequently, the training loss does actually not decrease during training in any significant way, and the loss gradients continue to be quite large. Why do the larger, noisier gradients not affect convergence speed? While the activation gradients are indeed noisy, the weight gradients are in fact relatively clean because Gaussian noise is independent and identically distributed (i.i.d.) in all pixels, and the weight gradients get averaged over 2^{16} pixels in our fully convolutional network.

Figure 1b makes the situation artificially harder by introducing inter-pixel correlation to the corruption. We achieve this by training a series of networks so that the Gaussian noise in the targets is blurred using progressively larger Gaussian filters, followed by normalization to restore variance. The corruption remains zero-mean. An example is shown in Figure 1c. As the correlation increases, the effective averaging of weight gradients decreases, and the weight updates become noisier. This makes the convergence slower, but even with extreme blur the eventual denoising quality is very similar (within 0.1 dB).

3.2. Other synthetic noises

We will now experiment with other types of synthetic noise. The training setup is the same as described above.

Poisson noise is the dominant source of noise in photographs. It is zero mean like Gaussian, but harder to remove because it is signal-dependent. We again use the L_2 loss and vary the noise magnitude $\lambda \in [0, 50]$ during training.

¹<http://r0k.us/graphics/kodak/>

When clean target images are used, we get an average of 30.59 ± 0.02 dB, while noisy targets give an equally good 30.57 ± 0.02 dB, again at similar convergence speed. A comparison method based on Ascombe transform (Mäkitalo & Foi, 2011) — which first transforms the input so that Poisson noise turns into Gaussian noise, removes it using BM3D, and finally does an inverse transform — gives over 2 dB lower quality than our method.

The other forms of noise in cameras, e.g. dark current and quantization, are small compared to Poisson noise, can be made zero-mean (Hasinoff et al., 2016), and hence pose no problems for training with noisy targets. We conclude that noise-free training data is unnecessary in this application.

That said, saturation (gamut clipping) does break our assumptions, as parts of the noise distribution are discarded and the expectation of the remaining part is no longer correct. Saturation is unwanted for other reasons as well, so this is not a significant practical limitation.

Multiplicative Bernoulli noise (aka binomial noise) constructs a random mask m that is 1 for valid pixels and 0 for zeroed/missing pixels. To avoid backpropagating gradients from missing pixels, we exclude them from the loss using that mask:

$$\operatorname{argmin}_{\theta} \sum_i (m \odot (f_{\theta}(\hat{x}_i) - \hat{y}_i))^2, \quad (7)$$

as described by Ulyanov et al. (2017) in the context of their deep image prior (DIP).

The probability of corrupted pixels is denoted with p ; in our training we vary $p \in [0.0, 0.95]$ and during testing $p = 0.5$. Training with clean targets gives an average of 31.85 ± 0.03 dB, noisy targets (separate m for input and target) give a slightly higher 32.02 ± 0.03 dB, possibly because noisy targets effectively implement a form of dropout (Srivastava et al., 2014) at the network output. DIP was almost 2 dB worse — DIP is not a learning-based solution, and as such very different from our approach, but it shares the property that neither clean examples nor an explicit model of the corruption is needed. We used the “Image reconstruction” setup as described in the DIP supplemental material.²

Text removal Figure 3 demonstrates blind text removal. The corruption consists of a large, varying number of random strings in random places, also on top of each other, and furthermore so that the font size and color are randomized as well. The font and string orientation remain fixed.

The network is trained using independently corrupted input and target pairs. The probability of corrupted pixels p is approximately $[0, 0.5]$ during training, and $p \approx 0.25$ during testing. In this test the mean (L_2 loss) is not the correct answer because the overlaid text has colors unrelated to the

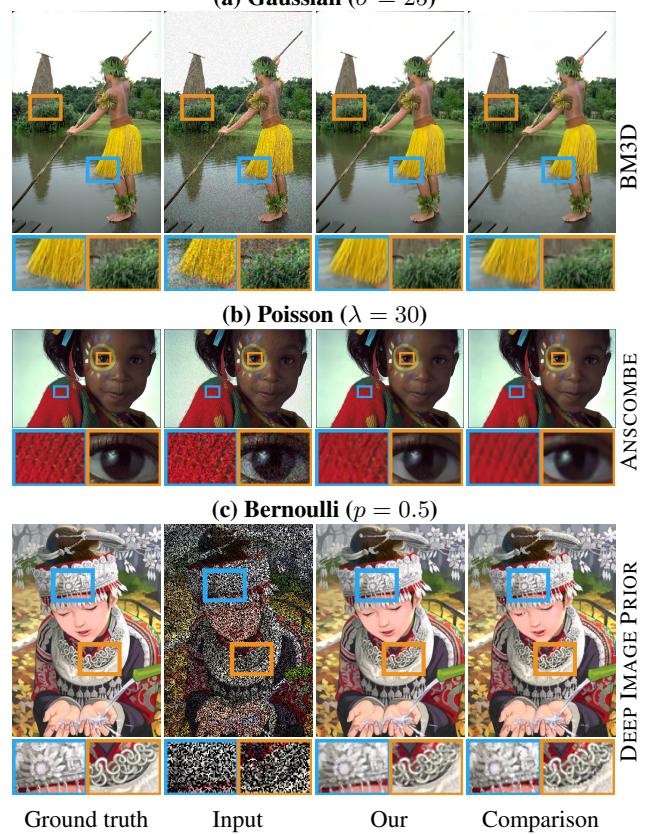


Figure 2. Example results for Gaussian, Poisson, and Bernoulli noise. Our result was computed by using noisy targets — the corresponding result with clean targets is omitted because it is virtually identical in all three cases, as discussed in the text. A different comparison method is used for each noise type.

actual image, and the resulting image would incorrectly tend towards a linear combination of the right answer and the average text color (medium gray). However, with any reasonable amount of overlaid text, a pixel retains the original color more often than not, and therefore the median is the correct statistic. Hence, we use $L_1 = |f_{\theta}(\hat{x}) - \hat{y}|$ as the loss function. Figure 3 shows an example result.

Random-valued impulse noise replaces some pixels with noise and retains the colors of others. Instead of the standard salt and pepper noise (randomly replacing pixels with black or white), we study a harder distribution where each pixel is replaced with a random color drawn from the uniform distribution $[0, 1]^3$ with probability p and retains its color with probability $1 - p$. The pixels’ color distributions are a Dirac at the original color plus a uniform distribution, with relative weights given by the replacement probability p . In this case, neither the mean nor the median yield the correct result; the desired output is the *mode* of the distribution (the Dirac spike). The distribution remains unimodal. For approximate mode seeking, we use an annealed version

²https://dmitryulyanov.github.io/deep_image_prior



Figure 3. In case of text removal the mean (L_2 loss) is not the correct answer but median (L_1) is.

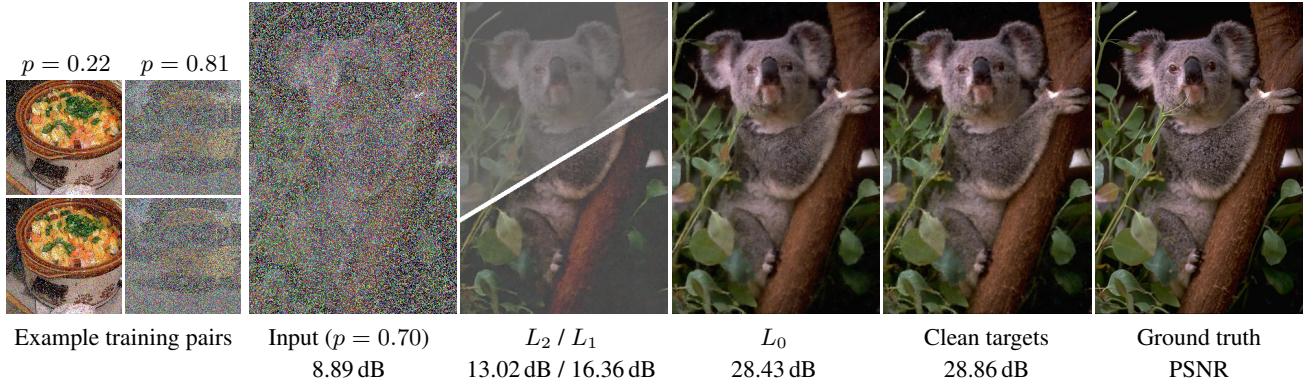


Figure 4. In case of random-valued impulse noise the mode (L_0) produces unbiased results, unlike mean (L_2) or median (L_1).

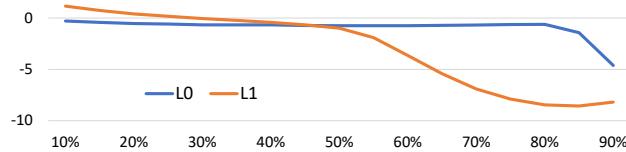


Figure 5. PSNR of noisy-target training relative to clean targets with a varying percentage of target pixels corrupted by RGB impulse noise. In this test a separate network was trained for each corruption level, and the graph was averaged over the KODAK dataset.

of the “ L_0 loss” function defined as $(|f_\theta(\hat{x}) - \hat{y}| + \epsilon)^\gamma$, where $\epsilon = 10^{-8}$, where γ is annealed linearly from 2 to 0 during training. This annealing did not cause any numerical issues in our tests. The relationship of the L_0 loss and mode seeking is analyzed in the supplement.

We again train the network using noisy inputs and noisy targets, where the probability of corrupted pixels is randomized separately for each pair from $[0, 0.95]$. Figure 4 shows the inference results when 70% input pixels are randomized. Training with L_2 loss biases the results heavily towards gray, because the result tends towards a linear combination the correct answer and and mean of the uniform random corrup-

tion. As predicted by theory, the L_1 loss gives good results as long as fewer than 50% of the pixels are randomized, but beyond that threshold it quickly starts to bias dark and bright areas towards gray (Figure 5). L_0 , on the other hand, shows little bias even with extreme corruptions (e.g. 90% pixels), because of all the possible pixel values, the correct answer (e.g. 10%) is still the most common.

3.3 Monte Carlo rendering

Physically accurate renderings of virtual environments are most often generated through a process known as Monte Carlo path tracing. This amounts to drawing random sequences of scattering events (“light paths”) in the scene that connect light sources and virtual sensors, and integrating the radiance carried by them over all possible paths (Veach & Guibas, 1995). The Monte Carlo integrator is constructed such that the intensity of each pixel is the expectation of the random path sampling process, i.e., the sampling noise is zero-mean. However, despite decades of research into importance sampling techniques, little else can be said about the distribution. It varies from pixel to pixel, heavily depends on the scene configuration and rendering parameters, and can be arbitrarily multimodal. Some lighting effects,

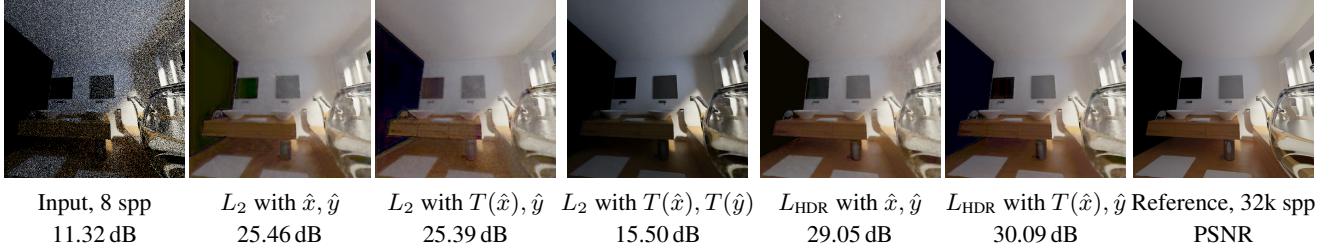


Figure 6. Comparison of various loss functions for training a Monte Carlo denoiser with noisy target images rendered at 8 samples per pixel (spp). In this high-dynamic range setting, our custom relative loss L_{HDR} is clearly superior to L_2 . Applying a non-linear tone map to the inputs is beneficial, while applying it to the target images skews the distribution of noise and leads to wrong, visibly too dark results.

such as focused caustics, also result in extremely long-tailed distributions with rare, bright outliers.

All of these effects make the removal of Monte Carlo noise much more difficult than removing, e.g., Gaussian noise. On the other hand, the problem is somewhat alleviated by the possibility of generating auxiliary information that has been empirically found to correlate with the clean result during data generation. In our experiments, the denoiser input consists of not only the per-pixel luminance values, but also the average albedo (i.e., texture color) and normal vector of the surfaces visible at each pixel.

High dynamic range (HDR) Even with adequate sampling, the pixel luminances may differ from each other by several orders of magnitude, and are thus typically represented using floating-point values. In order to construct an image suitable for the generally 8-bit display devices, this high dynamic range needs to be compressed to a fixed range using a tone mapping operator. A large variety of such operators have been proposed (Cerdá-Company et al., 2016), and in this work we consider a variant of Reinhard’s global operator (Reinhard et al., 2002): $T(v) = (v/(1+v))^{1/2.2}$, where v is a scalar luminance value, possibly pre-scaled with an image-wide exposure constant. This operator maps any $v \geq 0$ into range $0 \leq T(v) < 1$.

The combination of virtually unbounded range of luminances and the nonlinearity of operator T poses a problem. If we attempt to train a denoiser that outputs luminance values v , a standard MSE loss $L_2 = (f_\theta(\hat{x}) - \hat{y})^2$ will be dominated by the long-tail effects (outliers) in the targets, and training does not converge. On the other hand, if the denoiser were to output tonemapped values $T(v)$, the nonlinearity of T would make the expected value of noisy target images $\mathbb{E}\{T(v)\}$ different from the clean training target $T(\mathbb{E}\{v\})$, leading to incorrect predictions.

A metric often used for measuring the quality of HDR images is the relative MSE (Rousselle et al., 2011), where the squared difference is divided by the square of approximate luminance of the pixel, i.e., $(f_\theta(\hat{x}) - \hat{y})^2 / (\hat{y} + \epsilon)^2$. However, this metric suffers from the same nonlinearity

problem as comparing of tonemapped outputs. Therefore, we propose to use the network output, which tends towards the correct value in the limit, in the denominator: $L_{\text{HDR}} = (f_\theta(\hat{x}) - \hat{y})^2 / (f_\theta(\hat{x}) + 0.01)^2$. It can be shown that L_{HDR} converges to the correct expected value as long as we consider the gradient of the denominator to be zero.

Finally, we have observed experimentally that it is beneficial to tone map the input image $T(\hat{x})$ supplied to the network instead of using the HDR input image \hat{x} . The network continues to output non-tonemapped (linear-scale) luminance values, retaining the correctness of the expected value. Figure 6 evaluates the different loss functions.

Denoising Monte Carlo rendered images We trained a denoiser for Monte Carlo path traced images rendered using 64 samples per pixel (spp). Our training set consisted of 860 architectural images, and the validation was done using 34 images from a different set of scenes. Three versions of the training images were rendered: two with 64 spp using different random seeds (noisy input, noisy target), and one with 131k spp (clean target). The validation images were rendered in both 64 spp (input) and 131k spp (reference) versions. All images were 960×540 pixels in size, and as mentioned earlier, we also saved the albedo and normal buffers for all of the input images. Even with such a small dataset, rendering the 131k spp clean images was a strenuous effort — for example, Figure 7d took 40 minutes to render on a high-end graphics server with 8 × NVIDIA Tesla P100 GPUs and a 40-core Intel Xeon CPU.

The average PSNR of the 64 spp validation inputs with respect to the corresponding reference images was 22.31 dB (see Figure 7a for an example). The network trained for 2000 epochs using clean target images reached an average PSNR of 31.83 dB on the validation set, whereas the similarly trained network using noisy target images gave 0.5 dB lower quality. Example results corresponding to these runs are shown in Figure 7b,c – the training took 12 hours with a single NVIDIA Tesla P100 GPU.

When we trained the networks further, both continued to improve slowly, and after 4000 epochs the noisy targets finally



Figure 7. Denoising a Monte Carlo rendered image. (a) Image rendered with 64 samples per pixel. (b) Denoised 64 spp input, where the denoiser network was trained using 64 spp target images. (c) Same as previous, but with denoiser trained using the reference images as clean target images. (d) Reference image rendered with 131 072 samples per pixel. PSNR values refer to the images shown here, see text for averages over the entire validation set.

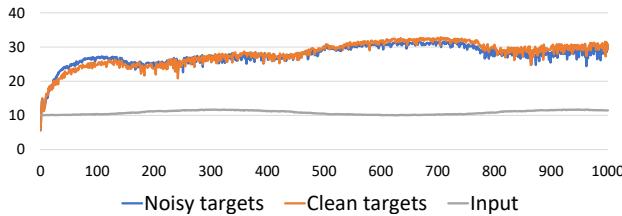


Figure 8. Online training PSNR during a 1000-frame flythrough of the scene in Figure 6. Noisy target images are almost as good for learning as clean targets, but are over 2000× faster to render (190 milliseconds vs 7 minutes per frame in this scene). Both denoisers offer a substantial improvement over the noisy input.

matched the 31.83 dB, i.e., noisy targets took approximately twice as long to converge. However, the gap between the two methods had not narrowed appreciably, leading us to believe that some quality difference will remain even in the limit. This is indeed what we would expect, since the training dataset contained a very limited number of training pairs (and thus noise realizations) due to the cost of generating the clean target images for the comparison method, and we wanted to test both methods using matching data. That said, given that noisy targets are 2000 times faster to produce, one could trivially produce a larger quantity of them as training data, train an even higher quality network, and still realize vast gains in data collection plus training.

Online training Since it can be very difficult to collect a sufficiently large corpus of Monte Carlo rendered images for training a generally applicable denoiser, another possibility is to train a denoiser to be specific to a single 3D scene, e.g., a game level or a movie shot (Chaitanya et al., 2017). In this context it can be desirable to train the denoiser dynamically, while walking through the scene. In order to maintain interactive frame rates, we can afford only few samples per pixel, and thus both input and target images will be inherently noisy.

Figure 8 shows the convergence plots for an experiment where we trained a denoiser from scratch for the duration of 1000 frames in a scene flythrough. On an NVIDIA Titan V GPU, path tracing a single 512×512 pixel image with 8 spp took 190 ms, and we rendered two images to act as input and target. A single network training iteration with a random 256×256 pixel crop took 11.25 ms and we performed eight of them per frame. Finally, we denoised both rendered images, each taking 15 ms, and averaged the result to produce the final denoised image shown to the user. The total frame time was thus 500 ms for rendering, training and inference.

As can be seen in Figure 8, a denoiser trained with clean output images does not perform appreciably better than the denoiser trained on the noisy images. Rendering a single clean target image takes approximately 7 minutes in this scene, so it is obvious that the quality-vs-time tradeoff favors training with noisy target images instead.

3.4. Magnetic resonance imaging (MRI)

Magnetic Resonance Imaging (MRI) produces volumetric images of biological tissues by applying carefully controlled magnetic and RF fields that vary over space and time, and measuring the radio frequency emissions of molecules precessing in those fields by coils. Mathematically, this process amounts to sampling the Fourier transform (the “*k*-space”) of the signal. Modern MRI techniques have long relied on compressed sensing (CS) to cheat the Nyquist-Shannon limit: they undersample *k*-space, and perform non-linear reconstruction that removes aliasing by exploiting the sparsity of the image in a suitable transform domain (Lustig et al., 2008). Effective compressed sensing requires sampling patterns that yield measurements with *incoherent* errors that do not tend to magnify, as they would due to phase alignments with standard regular sampling grids.

We observe that if we turn the *k*-space sampling into a ran-

dom process with a known probability density $p(k)$ over the frequencies k , our main idea applies. In particular, we model the k -space sampling operation as a Bernoulli process where each individual frequency has a probability $p(k) = e^{-\lambda|k|}$ of being selected for acquisition.³ The frequencies that are retained are weighted by the inverse of the selection probability, and non-chosen frequencies are set to zero. Clearly, the expectation of this “Russian roulette” process is the correct spectrum. The parameter λ controls the overall fraction of k -space retained; in the following experiments, we choose it so that 10% of the samples are retained relative to a full Nyquist-Shannon sampling. The undersampled spectra are transformed to the primal image domain by the standard inverse Fourier transform. An example of an undersampled input/target picture, the corresponding fully sampled reference, and their spectra, are shown in Figure 9(a, d).

Now we simply set up a regression problem of the form (6) and train a convolutional neural network using pairs of two independent undersampled images \hat{x} and \hat{y} of the same volume. As the spectra of the input and target are correct on expectation, and the Fourier transform is linear, we use the L_2 loss. Additionally, we improve the result slightly by enforcing the exact preservation of frequencies that are present in the input image \hat{x} by Fourier transforming the result $f_\theta(\hat{x})$, replacing the frequencies with those from the input, and transforming back to the primal domain before computing the loss: the final loss reads $(\mathcal{F}^{-1}(R_{\hat{x}}(\mathcal{F}(f_\theta(\hat{x})))) - \hat{y})^2$, where R denotes the replacement of non-zero frequencies from the input. This process is trained end-to-end.

We perform proof-of-concept experiments on 2D slices extracted from the IXI brain scan MRI dataset.⁴ To simulate spectral sampling, we draw random samples from the inverse FFT of the (already reconstructed) images in the dataset. Hence, in deviation from actual MRI samples, our data is real-valued and has the periodicity of the discrete FFT built-in. The training set contained 4936 images in 256×256 resolution from 50 subjects, and for validation we chose 500 random images from 10 different subjects. The baseline PSNR of the sparsely-sampled input images was 20.03 dB when reconstructed directly using IFFT. The network trained for 300 epochs with noisy targets reached an average PSNR of 31.10 dB on the validation data, and the network trained with clean targets reached 31.14 dB. Here the training with clean targets is similar to prior art (Wang et al., 2016; Lee et al., 2017). Training took 13 hours on an NVIDIA Tesla P100 GPU. Figure 9(b, c) shows an example of reconstruction results between convolutional networks trained with noisy and clean targets, respectively.

³Our simplified example deviates from practical MRI in the sense that we do not sample the spectra along 1D trajectories. However, we believe that designing pulse sequences that lead to similar pseudo-random sampling characteristics is straightforward.

⁴<http://brain-development.org/ixi-dataset> → T1 images.

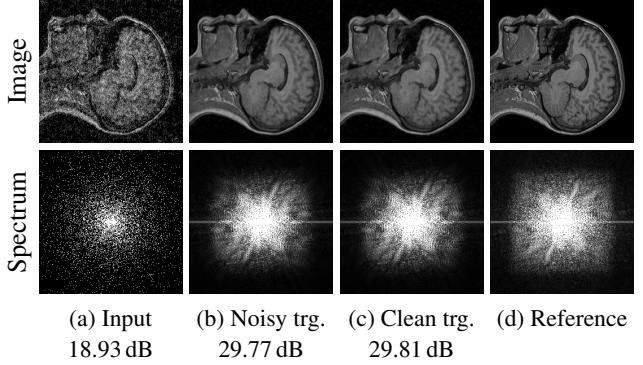


Figure 9. MRI reconstruction example. (a) Input image with only 10% of spectrum samples retained and scaled by $1/p$. (b) Reconstruction by a network trained with noisy target images similar to the input image. (c) Same as previous, but training done with clean target images similar to the reference image. (d) Original, uncorrupted image. PSNR values refer to the images shown here, see text for averages over the entire validation set.

In line with super-resolution approaches, we observe a smoothing effect both when training with clean and corrupted targets. Many recent MRI techniques attempt to increase apparent resolution by, for example, generative adversarial networks (GAN) (Quan et al., 2017). However, in terms of PSNR, our results quite closely match their reported results.

4. Discussion

We have shown that simple statistical arguments lead to surprising new capabilities in learned signal recovery; it is possible to recover signals under complex corruptions without observing clean signals, at performance levels equal or close to using clean target data. There are several real-world situations where obtaining clean training data is difficult: low-light photography (e.g., astronomical imaging), physically-based image synthesis, and magnetic resonance imaging. Our proof-of-concept demonstrations point the way to significant potential benefits in these applications by removing the need for potentially strenuous collection of clean data. Of course, there is no free lunch – we cannot learn to pick up features that are not there in the input data – but this applies equally to training with clean targets.

AmbientGAN (Ashish Bora, 2018) is a recent idea that allows principled training of generative adversarial networks (Goodfellow et al., 2014) using corrupted observations. In stark contrast to our approach, they need a perfect, explicit computational model of the corruption; our requirement of the knowledge of the appropriate statistical summary (mean, median) is significantly less limiting. We find combining ideas along both paths intriguing.

Acknowledgments

Bill Dally, David Luebke, Aaron Lefohn for discussions and supporting the research; NVIDIA Research staff for suggestions and discussion; Runa Lober and Gunter Sprenger for synthetic off-line training data; Jacopo Pantaleoni for the interactive renderer used in on-line training; Samuli Vuorinen for initial photography test data; Koos Zevenhoven for discussions on MRI basics.

References

- Ashish Bora, Eric Price, Alexandros G. Dimakis. AmbientGAN: Generative models from lossy measurements. *ICLR*, 2018.
- Cerdá-Company, Xim, Párraga, C. Alejandro, and Otazu, Xavier. Which tone-mapping operator is the best? A comparative study of perceptual quality. *CoRR*, abs/1601.04450, 2016.
- Chaitanya, Chakravarty R. Alla, Kaplanyan, Anton S., Schied, Christoph, Salvi, Marco, Lefohn, Aaron, Nowrouzezahrai, Derek, and Aila, Timo. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graph.*, 36(4):98:1–98:12, 2017.
- Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Trans. Image Process.*, 16(8):2080–2095, 2007.
- Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative Adversarial Networks. In *NIPS*, 2014.
- Hasinoff, Sam, Sharlet, Dillon, Geiss, Ryan, Adams, Andrew, Barron, Jonathan T., Kainz, Florian, Chen, Jiawen, and Levoy, Marc. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Trans. Graph.*, 35(6):192:1–192:12, 2016.
- He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.
- Huber, Peter J. Robust estimation of a location parameter. *Ann. Math. Statist.*, 35(1):73–101, 1964.
- Iizuka, Satoshi, Simo-Serra, Edgar, and Ishikawa, Hiroshi. Globally and locally consistent image completion. *ACM Trans. Graph.*, 36(4):107:1–107:14, 2017.
- Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. Image-to-image translation with conditional adversarial networks. In *Proc. CVPR 2017*, 2017.
- Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. In *ICLR*, 2015.
- Ledig, Christian, Theis, Lucas, Huszar, Ferenc, Caballero, Jose, Aitken, Andrew P., Tejani, Alykhan, Totz, Johannes, Wang, Zehan, and Shi, Wenzhe. Photo-realistic single image super-resolution using a generative adversarial network. In *Proc. CVPR*, pp. 105–114, 2017.
- Lee, D., Yoo, J., and Ye, J. C. Deep residual learning for compressed sensing MRI. In *Proc. IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pp. 15–18, 2017.
- Lustig, Michael, Donoho, David L., Santos, Juan M., and Pauly, John M. Compressed sensing MRI. In *IEEE Signal Processing Magazine*, volume 25, pp. 72–82, 2008.
- Maas, Andrew L, Hannun, Awni Y, and Ng, Andrew. Rectifier nonlinearities improve neural network acoustic models. In *Proc. International Conference on Machine Learning (ICML)*, volume 30, 2013.
- Mao, Xiao-Jiao, Shen, Chunhua, and Yang, Yu-Bin. Image restoration using convolutional auto-encoders with symmetric skip connections. In *Proc. NIPS*, 2016.
- Martin, D., Fowlkes, C., Tal, D., and Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*, volume 2, pp. 416–423, 2001.
- Mäkitalo, Markku and Foi, Alessandro. Optimal inversion of the Anscombe transformation in low-count Poisson image denoising. *IEEE Trans. Image Process.*, 20(1):99–109, 2011.
- Quan, Tran Minh, Nguyen-Duc, Thanh, and Jeong, Won-Ki. Compressed sensing MRI reconstruction with cyclic loss in generative adversarial networks. *CoRR*, abs/1709.00753, 2017.
- Reinhard, Erik, Stark, Michael, Shirley, Peter, and Ferwerda, James. Photographic tone reproduction for digital images. *ACM Trans. Graph.*, 21(3):267–276, 2002.
- Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. U-net: Convolutional networks for biomedical image segmentation. *MICCAI*, 9351:234–241, 2015.
- Rousselle, Fabrice, Knaus, Claude, and Zwicker, Matthias. Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph.*, 30(6):159:1–159:12, 2011.
- Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout:

A simple way to prevent neural networks from overfitting.
Journal of Machine Learning Research, 15:1929–1958,
2014.

Ulyanov, Dmitry, Vedaldi, Andrea, and Lempitsky, Victor S.
Deep image prior. *CoRR*, abs/1711.10925, 2017.

Veach, Eric and Guibas, Leonidas J. Optimally combining
sampling techniques for Monte Carlo rendering. In *Proc. ACM SIGGRAPH 95*, pp. 419–428, 1995.

Wang, S., Su, Z., Ying, L., Peng, X., Zhu, S., Liang, F.,
Feng, D., and Liang, D. Accelerating magnetic resonance
imaging via deep learning. In *Proc. IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pp.
514–517, 2016.

Zeyde, R., Elad, M., and Protter, M. On single image scale-
up using sparse-representations. In *Proc. Curves and
Surfaces: 7th International Conference*, pp. 711–730,
2010.

Zhang, Richard, Isola, Phillip, and Efros, Alexei A. Colorful
image colorization. In *Proc. ECCV*, pp. 649–666, 2016.

A. Appendix

A.1. Network architecture

Table 2 shows the structure of the U-network (Ronneberger et al., 2015) used in all of our tests, with the exception of the first test in Section 3.1 that used the “RED30” network (Mao et al., 2016). For all basic noise and text removal experiments with RGB images, the number of input and output channels were $n = m = 3$. For Monte Carlo denoising we had $n = 9, m = 3$, i.e., input contained RGB pixel color, RGB albedo, and a 3D normal vector per pixel. The MRI reconstruction was done with monochrome images ($n = m = 1$). Input images were represented in range $[-0.5, 0.5]$.

A.2. Training parameters

The network weights were initialized following He et al. (2015). No batch normalization, dropout or other regularization techniques were used. Training was done using ADAM (Kingma & Ba, 2015) with parameter values $\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-8}$.

Learning rate was kept at a constant value during training except for a brief rampdown period at where it was smoothly brought to zero. Learning rate of 0.001 was used for all experiments except Monte Carlo denoising, where 0.0003 was found to provide better stability. Minibatch size of 4 was used in all experiments.

A.3. Finite corrupted data in L_2 minimization

Let us compute the expected error in L_2 norm minimization task when corrupted targets $\{\hat{y}_i\}_{i=1}^N$ are used in place of the clean targets $\{y_i\}_{i=1}^N$, with N a finite number. Let y_i be arbitrary random variables, such that $\mathbb{E}\{\hat{y}_i\} = y_i$. As usual, the point of least deviation is found at the respective mean. The expected squared difference between these means across realizations of the noise is then:

$$\begin{aligned} & \mathbb{E}_{\hat{y}} \left[\frac{1}{N} \sum_i y_i - \frac{1}{N} \sum_i \hat{y}_i \right]^2 \\ &= \frac{1}{N^2} \left[\mathbb{E}_{\hat{y}} \left(\sum_i y_i \right)^2 - 2\mathbb{E}_{\hat{y}} \left[\left(\sum_i y_i \right) \left(\sum_i \hat{y}_i \right) \right] + \mathbb{E}_{\hat{y}} \left(\sum_i \hat{y}_i \right)^2 \right] \\ &= \frac{1}{N^2} \text{Var} \left(\sum_i \hat{y}_i \right) \\ &= \frac{1}{N} \left[\frac{1}{N} \sum_i \sum_j \text{Cov}(\hat{y}_i, \hat{y}_j) \right] \end{aligned} \quad (8)$$

In the intermediate steps, we have used $\mathbb{E}_{\hat{y}}(\sum_i \hat{y}_i) = \sum_i y_i$ and basic properties of (co)variance. If the corruptions are

NAME	N_{out}	FUNCTION
INPUT	n	
ENC_CONV0	48	Convolution 3×3
ENC_CONV1	48	Convolution 3×3
POOL1	48	Maxpool 2×2
ENC_CONV2	48	Convolution 3×3
POOL2	48	Maxpool 2×2
ENC_CONV3	48	Convolution 3×3
POOL3	48	Maxpool 2×2
ENC_CONV4	48	Convolution 3×3
POOL4	48	Maxpool 2×2
ENC_CONV5	48	Convolution 3×3
POOL5	48	Maxpool 2×2
ENC_CONV6	48	Convolution 3×3
UPSAMPLE5	48	Upsample 2×2
CONCAT5	96	Concatenate output of POOL4
DEC_CONV5A	96	Convolution 3×3
DEC_CONV5B	96	Convolution 3×3
UPSAMPLE4	96	Upsample 2×2
CONCAT4	144	Concatenate output of POOL3
DEC_CONV4A	96	Convolution 3×3
DEC_CONV4B	96	Convolution 3×3
UPSAMPLE3	96	Upsample 2×2
CONCAT3	144	Concatenate output of POOL2
DEC_CONV3A	96	Convolution 3×3
DEC_CONV3B	96	Convolution 3×3
UPSAMPLE2	96	Upsample 2×2
CONCAT2	144	Concatenate output of POOL1
DEC_CONV2A	96	Convolution 3×3
DEC_CONV2B	96	Convolution 3×3
UPSAMPLE1	96	Upsample 2×2
CONCAT1	$96+n$	Concatenate INPUT
DEC_CONV1A	64	Convolution 3×3
DEC_CONV1B	32	Convolution 3×3
DEV_CONV1C	m	Convolution 3×3 , linear act.

Table 2. Network architecture used in our experiments. N_{out} denotes the number of output feature maps for each layer. Number of network input channels n and output channels m depend on the experiment. All convolutions use padding mode “same”, and except for the last layer are followed by leaky ReLU activation function (Maas et al., 2013) with $\alpha = 0.1$. Other layers have linear activation. Upsampling is nearest-neighbor.

mutually uncorrelated, the last row simplifies to

$$\frac{1}{N} \left[\frac{1}{N} \sum_i \text{Var}(y_i) \right] \quad (9)$$

In either case, the variance of the estimate is the average (co)variance of the corruptions, divided by the number of samples N . Therefore, the error approaches zero as the number of samples grows. The estimate is unbiased in the

sense that it is correct on expectation, even with a finite amount of data.

The above derivation assumes scalar target variables. When \hat{y}_i are images, N is to be taken as the total number of scalars in the images, i.e., #images \times #pixels/image \times #color channels.

A.4. Mode seeking and the “ L_0 ” norm

Interestingly, while the “ L_0 norm” could intuitively be expected to converge to an exact mode, i.e. a local maximum of the probability density function of the data, theoretical analysis reveals that it recovers a slightly different point. While an actual mode is a zero-crossing of the *derivative* of the PDF, the L_0 norm minimization recovers a zero-crossing of its *Hilbert transform* instead. We have verified this behavior in a variety of numerical experiments, and, in practice, we find that the estimate is typically close to the true mode. This can be explained by the fact that the Hilbert transform approximates differentiation (with a sign flip): the latter is a multiplication by $i\omega$ in the Fourier domain, whereas the Hilbert transform is a multiplication by $-i \operatorname{sgn}(\omega)$.

For a continuous data density $q(x)$, the norm minimization task for L_p amounts to finding a point x^* that has a minimal expected p -norm distance (suitably normalized, and omitting the p th root) from points $y \sim q(y)$:

$$\begin{aligned} x^* &= \underset{x}{\operatorname{argmin}} \mathbb{E}_{y \sim q} \left\{ \frac{1}{p} |x - y|^p \right\} \\ &= \underset{x}{\operatorname{argmin}} \int \frac{1}{p} |x - y|^p q(y) dy \end{aligned} \quad (10)$$

Following the typical procedure, the minimizer is found at a root of the derivative of the expression under argmin :

$$\begin{aligned} 0 &= \frac{\partial}{\partial x} \int \frac{1}{p} |x - y|^p q(y) dy \\ &= \int \operatorname{sgn}(x - y) |x - y|^{p-1} q(y) dy \end{aligned} \quad (11)$$

This equality holds also when we take $\lim_{p \rightarrow 0}$. The usual results for L_2 and L_1 norms can readily be derived from this form. For the L_0 case, we take $p = 0$ and obtain

$$\begin{aligned} 0 &= \int \operatorname{sgn}(x - y) |x - y|^{-1} q(y) dy \\ &= \int \frac{1}{x - y} q(y) dy. \end{aligned} \quad (12)$$

The right hand side is the formula for the Hilbert transform of $q(x)$, up to a constant multiplier.