Monte Carlo Integration with Normalizing Flows

Thomas Müller [1 2 *] **Brian McWilliams** [1 *] **Fabrice Rousselle** [1 *] **Markus Gross** [1 2] **Jan Novák** [1 *]

Abstract

We use deep neural networks to drive importance sampling in Monte Carlo integration. Our work is based on non-linear independent components estimation (NICE), which we extend with piecewise-polynomial coupling transforms that greatly increase the modeling power of individual coupling layers. To allow learning from Monte Carlo samples, we derive unbiased gradient estimates for the KL and χ^2 divergences that only require unnormalized stochastic estimates of the target distribution. We demonstrate the benefit of our proposed coupling layers on generating natural images from the CelebA dataset, and that of our whole pipeline on light-transport simulation.

1. Introduction

Solving integrals is a fundamental problem of calculus that appears in many disciplines of science and engineering. We are interested in efficiently approximating otherwise difficult to solve integrals with a Monte Carlo estimator $\langle F \rangle_N$

$$F = \int_{\mathcal{D}} f(x)\,\mathrm{d}x \approx \sum_{i=1}^{N} \frac{f(X_i)}{q(X_i)} = \langle F \rangle_N . \qquad (1)$$

Here, $q(x)$ is a probability density function (PDF) that governs the distribution of $X_1, \ldots, X_N$. As long as $q(x)$ is positive for all x with non-zero $f(x)$, the Monte Carlo estimator is unbiased (i.e. $\mathbb{E}[\langle F \rangle_N] = F$) and converges with variance $\mathbb{V}[\langle F \rangle_N] = 1/N\ \mathbb{V}[\langle F \rangle]$. The exact choice of $q(x)$ is important: it greatly influences $\mathbb{V}[\langle F \rangle]$ and thereby the efficiency of $\langle F \rangle_N$. The ultimate goal is to choose $q(x) = p(x) = f(x)/F$, in which case $\mathbb{V}[\langle F \rangle]$ is zero, maximizing efficiency.

We focus on the general setting where little to no prior knowledge about f is given, but f can be observed at a sufficiently high number of points. Many data-driven approaches have been proposed in the past to *learn* $q(x)$ from such observations. These approaches parameterize $q(x;\theta)$ with parameters θ that are optimized to minimize some distance metric $D(p \,\|\, q;\theta)$. Existing parameterizations range from dynamic collections of particles (Del Moral, 1996; Liu & Chen, 1998) over parametric mixture models (Hey & Purgathofer, 2002; Cappé et al., 2004; 2008; Vorba et al., 2014) to non-parametric approaches (Jensen, 1995; Lafortune & Willems, 1995; Müller et al., 2017; Dahm & Keller, 2018).

To this end, generative neural networks have seen relatively little usage in the past because popular models such as variational autoencoders (Kingma & Welling, 2014) and generative adversarial networks (Goodfellow et al., 2014) do not satisfy all key properties that are required in typical Monte Carlo integration problems:

1. Evaluating the density $q(x;\theta)$ of samples (and often also arbitrary data points) must be *exact*[1].
2. Evaluating the density $q(x;\theta)$ *and* the generation of samples X_i must be *fast*.

Neural-network-parameterized "normalizing flows" are of interest, because they permit exact evaluation of the learned PDF by modeling x as a differentiable deterministic bijective mapping of a latent variable z (with known distribution $q(z)$): $x = h^{-1}(z;\theta)$. The learned PDF is then

$$q(x;\theta) = q(z)\left|\det\left(\frac{\partial h(x;\theta)}{\partial x^T}\right)\right|, \qquad (2)$$

where $\frac{\partial h(x;\theta)}{\partial x^T}$ is the Jacobian matrix of $h(x;\theta)$. To ensure an expressive h that leads to a good approximation of p, it is common to compose it from a sequence of simple bijective transformations $h = h_1 \circ \cdots \circ h_L$, each parameterized by a neural network.

Our approach builds on the work of Dinh et al. (2014; 2016) which—unlike more general autoregressive approaches (Chen et al., 2016; Kingma et al., 2016; Papamakarios et al., 2017)—allows *both* fast sampling *and* density evaluation through the use of so-called "coupling layers" to compose transforms.

[*]This work was conducted while the author was employed at the listed institutions [1]Disney Research Studios [2]ETH Zürich. Correspondence to: Thomas Müller <thomas94@gmx.net>.

First workshop on *Invertible Neural Networks and Normalizing Flows* (ICML 2019), Long Beach, CA, USA

[1]Stochastic estimates are not sufficient, even if unbiased, due to $q(x;\theta)$ occurring in the denominator of Equation (1) .

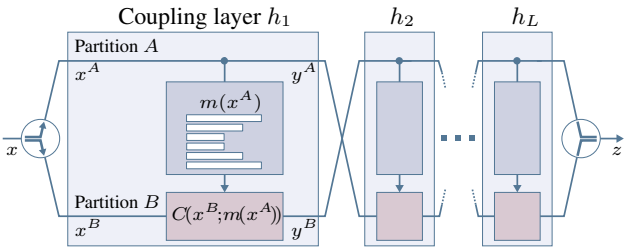

Figure 1. A coupling layer splits the input x into two partitions A and B. One partition is left untouched, whereas dimensions in the other partition are warped using a parametric coupling transform C driven by the output of a neural network m. Multiple coupling layers are composed to achieve expressive transforms.

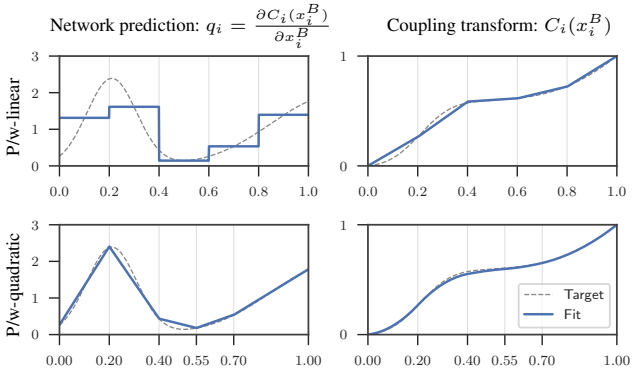

Figure 2. Predicted PDFs (left) and corresponding CDFs (right) with $K = 5$ bins fitted to a target distribution (dashed).

Definition 1 (Coupling layer)**.** *Let $x \in \mathbb{R}^D$ be an input vector, A and B denote disjoint partitions of $[\![1, D]\!]$, and m be a learnable function on $\mathbb{R}^{|A|}$, then the output of a coupling layer $y = (y^A, y^B) = h(x)$ is defined as*

$$y^A = x^A\,, \tag{3}$$
$$y^B = C\big(x^B; m(x^A)\big)\,, \tag{4}$$

where the "coupling transform" $C : \mathbb{R}^{|B|} \times m(\mathbb{R}^{|A|}) \to \mathbb{R}^{|B|}$ is a separable and invertible map.

Coupling layers are assembled by alternating the partitions A and B (see Figure 1), which, together with their separability, ensures an upper-triangular Jacobian and thereby a tractable Jacobian determinant: the product along the diagonal entries.

Dinh et al. (2016) propose to use affine coupling transforms $C(x^B; s, t) = x^B \odot e^s + t$ with scaling s and translation t being the output of a neural network $m(x^A)$. We propose to increase the expressiveness by introducing more sophisticated transforms (as did neural autoregressive flows (Chen et al., 2018)). Concretely, in Section 2 we introduce two *piecewise-polynomial* coupling transforms—piecewise-linear and piecewise-quadratic—that greatly increase the expressive power of individual coupling layers, allowing us to employ fewer of those and thereby reduce the total cost. We illustrate the benefits on a toy 2-D regression problem and test the performance when learning a (high-dimensional) distribution of natural images (CelebA).

In Section 3, we proceed to apply NICE to Monte Carlo integration and propose an optimization strategy for minimizing estimation variance via the KL and χ^2 divergences. We demonstrate the proposed approach in path-traced Monte Carlo light-transport simulation: we use NICE with our polynomial warps to guide the construction of light paths and demonstrate that it outperforms the state of the art at equal sample counts, albeit with larger computational overhead.

2. Piecewise-Polynomial Coupling Layers

We introduce the usage of piecewise polynomials with degrees 1 and 2, i.e. piecewise-linear and piecewise-quadratic warps. In contrast to Dinh et al. (2014; 2016), who assume $x, y \in (-\infty, +\infty)^D$ and Gaussian latent variables, our transforms operate in the unit hypercube (i.e. $x, y \in [0, 1]^D$) with uniformly distributed latent variables. Unbounded domains can still be handled by warping the input of h_1 and the output of h_L e.g. using the sigmoid and logit functions.

Similarly to Dinh and colleagues, we ensure computationally tractable Jacobians via separability: we transform each dimension i independently using a 1-D transform C_i. Operating on unit intervals allows interpreting the warping function C_i as a cumulative distribution function (CDF). To produce each C_i, we instrument the neural network to output the corresponding unnormalized probability density q_i, and construct C_i by integration; see Figure 2 for an illustration.

2.1. Piecewise-Linear Coupling Transform

We begin by investigating piecewise-linear coupling transforms. Recall that we partition the D-dimensional input vector in two disjoint groups, A and B, such that $x = (x^A, x^B)$. We divide the unit dimensions in partition B into K bins of equal width $w = K^{-1}$. To define all $|B|$ transforms at once, we instrument the network $m(x^A)$ to predict a matrix $\widehat{Q} \in \mathbb{R}^{|B| \times K}$. Each i-th row of $\widehat{Q}$ defines the unnormalized probability *mass* function of the warp in dimension i of x^B; we normalize the rows using the softmax function σ and denote the normalized matrix Q; $Q_i = \sigma(\widehat{Q}_i)$. The PDF in i-th dimension is then defined as $q_i(x_i^B) = Q_{ib}/w$, where b is the bin that contains the scalar value x_i^B.

In order to obtain a piecewise-linear coupling transform C_i, which is used to warp dimension i, we integrate the PDF:

$$C_i(x_i^B; Q) = \int_0^{x_i^B} q_i(t)\,\mathrm{d}t = \alpha Q_{ib} + \sum_{k=1}^{b-1} Q_{ik}\,, \tag{5}$$

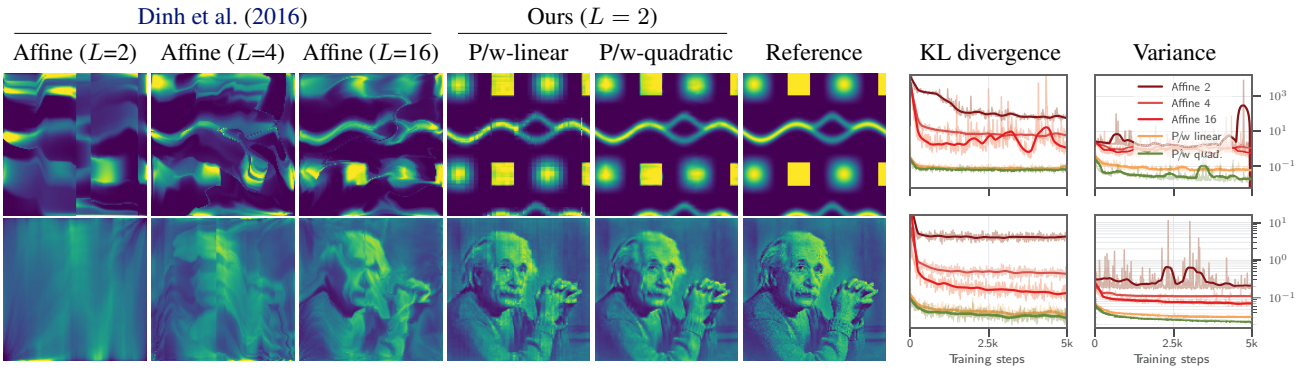

Figure 3. Our 32-bin piecewise-linear (4-th column) and 32-bin piecewise-quadratic (5-th column) coupling layers achieve superior performance compared to affine coupling layers (Dinh et al., 2016) on some 2-D regression problems, despite using roughly 7 times fewer model parameters than the result with 16 affine layers. The false-colored distributions were obtained by optimizing KL divergence with uniformly drawn samples over the 2D image domain. The plots on the right show the training error (KL divergence) and the variance of estimating the integral of the image using samples drawn from the learned densities; low values indicate effective importance sampling.

where $\alpha = Kx_i^B - \lfloor Kx_i^B \rfloor$ represents the relative position of x_i^B in bin b.

Since $C(x^B; Q)$ is separable by definition, its Jacobian matrix is diagonal and the determinant is equal to the product of the diagonal terms. These are equal to $\frac{\partial C_i(x;Q)}{\partial x} = q_i(x_i^B)$.

To reduce the number of bins K required for a good fit, we would like the transform to have an adaptive resolution, i.e. we want the network to also predict bin *widths*. Unfortunately, these *cannot* be easily optimized with gradient descent in the piecewise-*linear* case; see Appendix A.

2.2. Piecewise-Quadratic Coupling Transform

In order to facilitate optimization of the bin widths, we increase the order of the polynomials and utilize piecewise-quadratic warps. We parameterize these analogously to the piecewise-linear transforms instrumenting the neural network $m(x^A)$ to output softmax-normalized vertex heights in matrix $V \in \mathbb{R}^{|B|\times(K+1)}$, and additional softmax-normalized horizontal differences between neighboring vertices (bin widths) in matrix $W \in \mathbb{R}^{|B|\times K}$. A precise description of the transform can be found in Müller et al. (2018).

2.3. Analysis

In Figure 3 we trained a RealNVP-based model using affine transforms and our piecewise-polynomial transforms on toy 2-D distributions using uniformly sampled training data[2]. We optimize $D_{\mathrm{KL}}(p \,\|\, q; \theta)$ using the stochastic gradient described in Section 3.1. Every per-layer neural network has the same architecture (except for the input and output layers); see Müller et al. (2018) for details. The cost of evaluating the flow is proportional to the number of coupling layers. As in prior works, the neural networks remain the computational bottleneck. The larger parameter sets required for our piecewise-polynomial transforms increase the number of trainable parameters by 20% compared to affine transforms.

When using $L = 2$ coupling layers, the piecewise-polynomial transforms consistently perform better thanks to their significantly larger modeling power, and outperform even large numbers (e.g. $L = 16$) of affine coupling layers.

In Figure 4, we tested the piecewise-quadratic coupling layers also on a high-dimensional density-estimation problem: learning the distribution of faces in the CelebFaces Attributes dataset (Liu et al., 2015); we use the same data as Dinh et al. (2016). Our architecture is based on the authors' publicly available implementation and differs only in the used coupling layer and the depth of the network—we use 4 recursive subdivisions while the authors use 5, resulting in 28 versus 35 coupling layers. We chose $K = 4$ bins. Since our coupling layers operate on the $[0, 1]^D$ domain, we do not use batch normalization on the transformed data.

The visual quality of our results is comparable to that obtained by (Dinh et al., 2016), although we perform marginally better in terms of bits per dimension: we yield 2.85 and 2.89 on training and validation data, respectively, whereas Dinh et al. (2016) reported 2.97 and 3.02. We tried decreasing the number of coupling layers while increasing the number of bins within each of them, but the results became overall worse, leading to the hypothesis that—unlike the 2-D case—learning the high-dimensional distribution of natural images benefits more from having many coupling layers rather than having fewer but more expressive ones.

[2]We also ran the same experiment with equally weighted i.i.d. samples drawn *proportional* to the reference function—i.e. in a density-estimation setting—producing near-identical results.

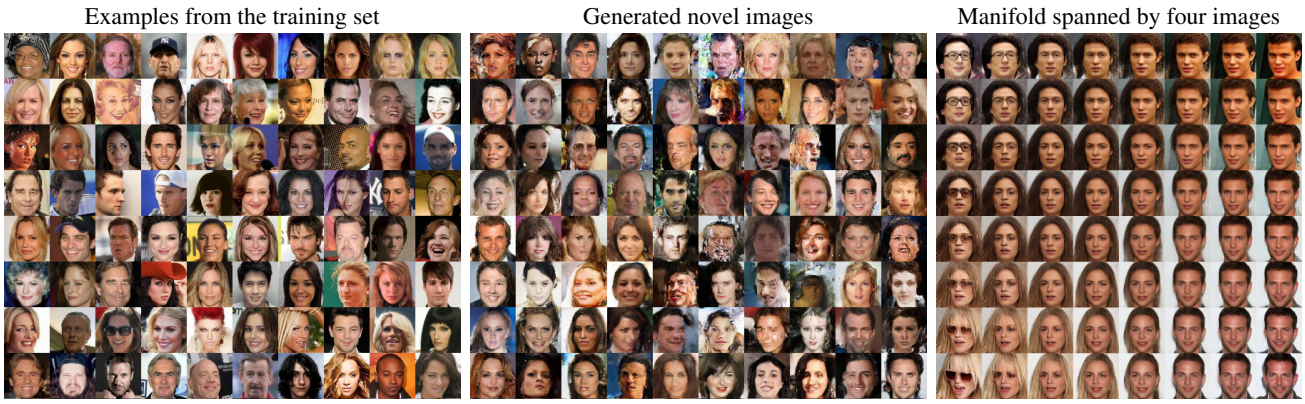

Figure 4. Generative modeling of facial photographs using the architecture of Dinh et al. (2016) with our piecewise-quadratic coupling transform. We show training examples (left), faces generated by our trained model (middle), and a manifold of faces spanned by linear interpolation of 4 training examples in latent space (right; training examples are in the corners).

3. Application to Monte Carlo Integration

In this section, we apply the NICE framework to Monte Carlo integration. Our goal is to reduce estimation variance by learning sampling PDFs $q(x;\theta)$ that are as close as possible to $p(x) = f(x)/F$ from unbiased noisy observations of the integrand $\langle f(x)\rangle$ (i.e. $\mathbb{E}[\langle f(x)\rangle] = f(x)$). We follow the standard approach of quantifying the distance using one of the commonly used divergence metrics and optimizing θ with a stochastic-gradient-descent based technique; we use Adam (Kingma & Ba, 2014). While all divergence metrics reach their minimum if both distributions are equal, they differ in shape and therefore produce different q in practice.

3.1. Minimizing Kullback-Leibler Divergence

It is well known that the gradient of the KL divergence is the expected negative log-likelihood gradient

$$\nabla_\theta D_{\mathrm{KL}}(p \,\|\, q;\theta) = \mathop{\mathbb{E}}_{X\sim p}\left[-\nabla_\theta \log q(X;\theta)\right]. \tag{6}$$

In our setting, however, we are unable to sample from p (we are trying to learn to do so, after all), so we write the expectation in terms of $X \sim q$

$$\nabla_\theta D_{\mathrm{KL}}(p \,\|\, q;\theta) = \mathop{\mathbb{E}}_{X\sim q}\left[-\frac{p(X)}{q(X;\theta)}\nabla_\theta \log q(X;\theta)\right]. \tag{7}$$

In most integration problems, $p(x)$ is only accessible in an unnormalized form through $f(x)$: $p(x) = f(x)/F$. Since F is unknown—this is what we are trying to estimate in the first place—the gradient can be estimated only up to the global scale factor F. This is not an issue since common optimizers (e.g. Adam) compensate for this factor. Furthermore, substituting $f(x)$ with $\langle f(x)\rangle$ as required above does not affect the expectation due to linearity. Equation (7) therefore shows that minimizing the KL divergence via gradient descent is equivalent to minimizing the negative log likelihood weighted by MC estimates of F.

3.2. Minimizing Variance via χ^2 Divergence

The arguably most attractive quantity to minimize in the context of (unbiased) Monte Carlo integration is the variance of the estimator. Inspired by previous works that strive to directly minimize variance (Herholz et al., 2016; 2018; Pantaleoni & Heitz, 2017; Vévoda et al., 2018), we demonstrate how this can be achieved for the MC estimator $\langle f(x)\rangle / q(X;\theta)$, with $X \sim q$, via gradient descent. We begin with the variance gradient and simplify:

$$\begin{aligned}\nabla_\theta \mathop{\mathbb{V}}_{X\sim q}\left[\frac{\langle f(X)\rangle}{q(X;\theta)}\right] &= \nabla_\theta \int_\Omega \frac{\langle f(x)\rangle^2}{q(x;\theta)}\,\mathrm{d}x \\ &= \int_\Omega \langle f(x)\rangle^2 \nabla_\theta \frac{1}{q(x;\theta)}\,\mathrm{d}x \\ &= \mathop{\mathbb{E}}_{X\sim q}\left[-\left(\frac{\langle f(X)\rangle}{q(X;\theta)}\right)^2 \nabla_\theta \log q(X;\theta)\right].\end{aligned} \tag{8}$$

Relation to the Pearson χ^2 divergence Upon close inspection it turns out that if $f(x)$ is known exactly, then the variance objective gradient (Equation 8) is proportional to the Pearson χ^2 divergence gradient $\nabla_\theta D_{\chi^2}(p \,\|\, q;\theta)$:

$$\begin{aligned}\nabla_\theta D_{\chi^2}(p \,\|\, q;\theta) &= \nabla_\theta \int_\Omega \frac{(p(x) - q(x;\theta))^2}{q(x;\theta)}\,\mathrm{d}x \\ &= \nabla_\theta \int_\Omega \frac{p(x)^2}{q(x;\theta)}\,\mathrm{d}x \\ &= \frac{1}{F^2}\nabla_\theta \mathop{\mathbb{V}}_{X\sim q}\left[\frac{f(X)}{q(X;\theta)}\right].\end{aligned} \tag{9}$$

As such, minimizing the variance of a Monte Carlo estimator often amounts to minimizing the Pearson χ^2 divergence between the ground-truth and the learned distributions.

Connection to the KL divergences Notably, the stochastic variance gradient and that of the KL divergence differ only in the weight applied to the log likelihood. In $\nabla_\theta D_{\mathrm{KL}}$ the log likelihood is weighted by the Monte Carlo weight, whereas when optimizing variance, the log likelihood is weighted by the *squared* Monte Carlo weight. The variance thus penalizes large discrepancies stronger, specifically, low values of q in regions of large density p. As such, it tends to produce more conservative q than D_{KL}.

3.3. Application to Light-Transport Simulation

In Figure 5, we apply NICE with our piecewise-polynomial coupling layers to Monte Carlo simulation of light transport, which estimates the amount of light arriving at a sensor by constructing millions of random walks that mimic photon trajectories. The variance of the estimation depends on how well the distribution of random walks—referred to as light paths—matches the light transport. We learn PDFs for constructing the light paths *online*, i.e. during rendering: we start with randomly initialized networks and gradually optimize their weights using the aforementioned loss gradients computed for a stream of light-path samples. The learned PDFs are used to construct new samples of higher quality, leading to estimates with less variance and thereby images with less noise.

We compare rendered images at equal sample count (an equal number of light paths traced to compute each image) against a baseline path tracer (PT-Unidir) and prior path-guiding approaches based on SD-trees (PPG) by Müller et al. (2017) and Gaussian mixture models (GMM) by Vorba et al. (2014). Our algorithm achieves the lowest error when optimized with the KL divergence; quantified using the "mean absolute percentage error" (MAPE). For additional results, visualizations, and implementation details please see an extended version of this report (Müller et al., 2018).

4. Conclusion

We extended NICE using piecewise-polynomial transforms that increase the modeling power of individual coupling layers. We also discussed two schemes for optimizing trainable parameters within the context of Monte Carlo integration, in particular we pointed out that D_{χ^2} is closely related to integration variance. Finally, we demonstrated that instances of normalizing flows, such as the ones discussed, can yield state-of-the-art results in realistic image synthesis where images are obtained by means of sampling transport paths. We believe that other particle-transport problems can also benefit from these approaches.

Acknowledgments

We thank Thijs Vogels for bringing RealNVP to our attention and Sebastian Herholz, Yining Karl Li, and Jacob Munkberg for valuable feedback. We are grateful to Vorba et al. (2014) and Dinh et al. (2016) for releasing the source code of their work. We also thank the following people for providing scenes and models that appear in our figures: Benedikt Bitterli (2016), Johannes Hanika (NECKLACE), Samuli Laine and Olesya Jakob (COPPER HAIRBALL), Jay-Artist (COUNTRY KITCHEN), thecali (SPACESHIP), and Tiziano Portenier (BATHROOM, BOOKSHELF).

References

Bitterli, B. Rendering resources, 2016. https://benedikt-bitterli.me/resources/.

Cappé, O., Guillin, A., Marin, J.-M., and Robert, C. P. Population monte carlo. *Journal of Computational and Graphical Statistics*, 13(4):907–929, 2004. doi: 10.1198/106186004X12803.

Cappé, O., Douc, R., Guillin, A., Marin, J.-M., and Robert, C. P. Adaptive importance sampling in general mixture classes. *Statistics and Computing*, 18(4):447–459, December 2008. doi: 10.1007/s11222-008-9059-x.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. *arXiv:1806.07366*, June 2018.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. *arXiv:1611.02731*, 2016.

Dahm, K. and Keller, A. Learning light transport the reinforced way. In Owen, A. B. and Glynn, P. W. (eds.), *Monte Carlo and Quasi-Monte Carlo Methods*, pp. 181–195. Springer International Publishing, 2018.

Del Moral, P. Non linear filtering: Interacting particle solution. *Markov Processes and Related Fields*, 2:555–580, March 1996.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv:1410.8516*, October 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv:1605.08803*, March 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

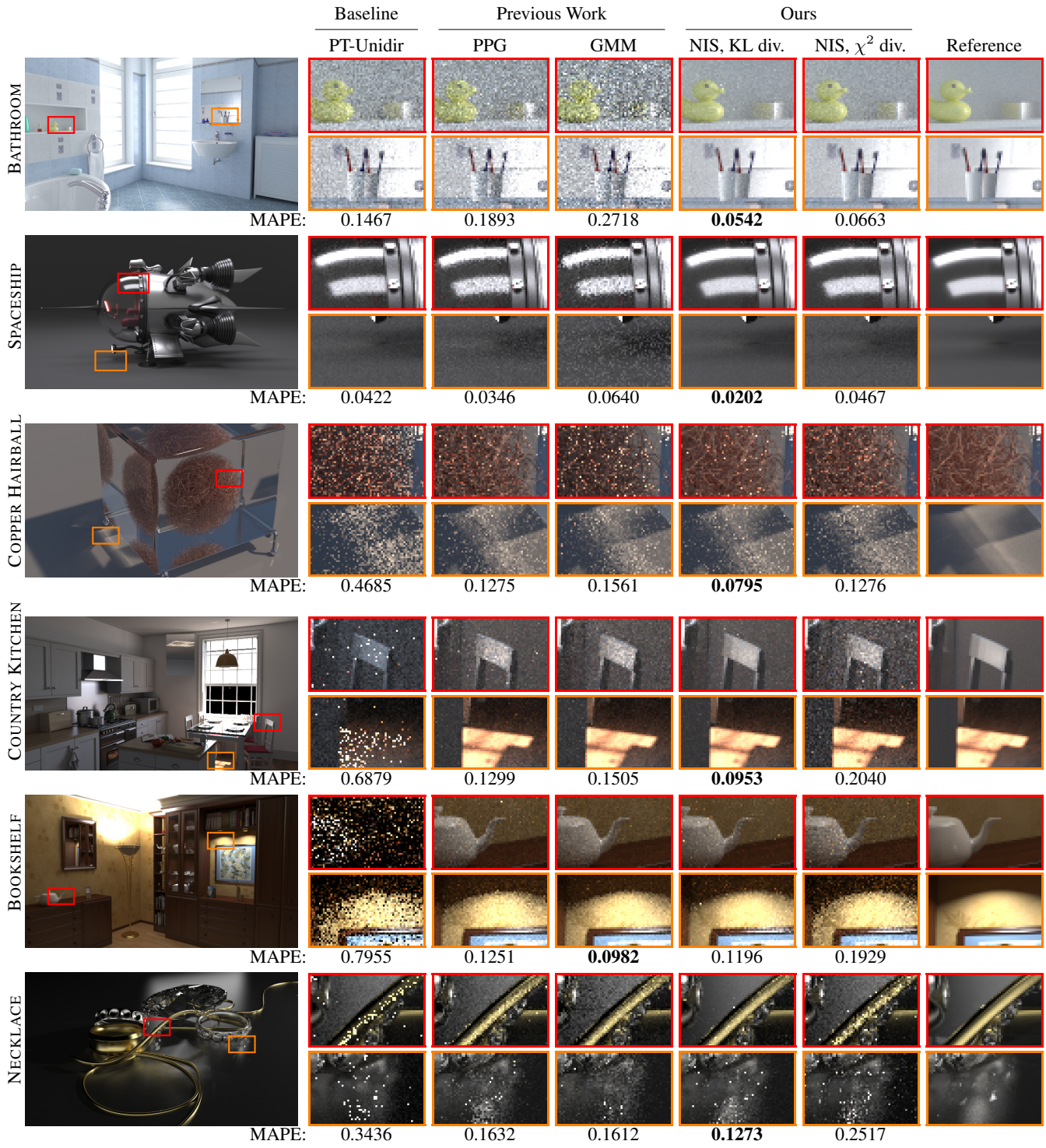

Figure 5. Neural path guiding. We compare a baseline uni-directional path tracer (PT-Unidir), the algorithms of Müller et al. (2017) (PPG) and Vorba et al. (2014) (GMM), and our framework with $L = 4$ piecewise-quadratic coupling layers sampling the rendering equation, when optimizing either the KL or the χ^2 divergence. Overall, sampling light transport with the KL divergence yields the most robust results. Additional results and error visualizations can be found in an extended version of this report (Müller et al., 2018).

Herholz, S., Elek, O., Vorba, J., Lensch, H., and Křivánek, J. Product Importance Sampling for Light Transport Path Guiding. *Computer Graphics Forum*, 2016. doi: 10.1111/cgf.12950.

Herholz, S., Elek, O., Schindel, J., Křivánek, J., and Lensch, H. P. A. A Unified Manifold Framework for Efficient BRDF Sampling based on Parametric Mixture Models. In Jakob, W. and Hachisuka, T. (eds.), *Eurographics Symposium on Rendering - Experimental Ideas & Implementations*. The Eurographics Association, 2018.

Hey, H. and Purgathofer, W. Importance sampling with hemispherical particle footprints. In *Proceedings of the 18th Spring Conference on Computer Graphics*, SCCG '02, pp. 107–114. ACM, 2002. doi: 10.1145/584458.584476.

Jensen, H. W. Importance driven path tracing using the photon map. In *Rendering Techniques*, pp. 326–335, Vienna, 1995. Springer Vienna. doi: 10.1007/978-3-7091-9430-0_31.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980*, June 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, April 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751, 2016.

Lafortune, E. P. and Willems, Y. D. A 5d tree to reduce the variance of monte carlo ray tracing. In *Rendering Techniques '95 (Proc. of the 6th Eurographics Workshop on Rendering)*, pp. 11–20, 1995. doi: 10.1007/978-3-7091-9430-0_2.

Liu, J. S. and Chen, R. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pp. 3730–3738, Washington, DC, USA, 2015. IEEE Computer Society. doi: 10.1109/ICCV.2015.425.

Müller, T., Gross, M., and Novák, J. Practical path guiding for efficient light-transport simulation. *Computer Graphics Forum*, 36(4):91–100, June 2017. doi: 10.1111/cgf.13227.

Müller, T., McWilliams, B., Rousselle, F., Gross, M., and Novák, J. Neural importance sampling. *The Recursive Journal of Scientology*, abs/42, 2018.

Pantaleoni, J. and Heitz, E. Notes on optimal approximations for importance sampling. *arXiv:1707.08358*, July 2017.

Papamakarios, G., Murray, I., and Pavlakou, T. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Vévoda, P., Kondapaneni, I., and Křivánek, J. Bayesian online regression for adaptive direct illumination sampling. *ACM Trans. Graph.*, 37(4), 2018.

Vorba, J., Karlík, O., Šik, M., Ritschel, T., and Křivánek, J. On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph.*, 33(4), August 2014.

A. Adaptive Bin Sizes in Piecewise-Linear Coupling Transforms

Without loss of generality, we investigate the simplified scenario of a one-dimensional input $A = \emptyset$ and $B = \{1\}$, a single coupling layer $L = 1$ and the KL-divergence loss function. Further, let the coupling layer admit a piecewise-linear coupling transform—i.e. it predicts a piecewise-constant PDF—with $K = 2$ bins. Let the width W of the 2 bins be controlled by traininable parameter $\theta \in \mathbb{R}$ such that $W_1 = \theta$ and $W_2 = 1 - \theta$ and $S = Q_1\theta + Q_2(1 - \theta)$, then

$$q(x;\theta) = \begin{cases} Q_1/S & \text{if } x < \theta \\ Q_2/S & \text{otherwise.} \end{cases} \tag{10}$$

Using Eq. 7, the gradient of the KL divergence w.r.t. θ is

$$\nabla_\theta D_{\mathrm{KL}}(p \,\|\, q;\theta) = \nabla_\theta \int_0^1 \begin{cases} p(x)\log(Q_1/S) & \text{if } x < \theta \\ p(x)\log(Q_2/S) & \text{otherwise} \end{cases} \mathrm{d}x\,, \tag{11}$$

where—in contrast to our piecewise-quadratic transform—the gradient can *not* be moved into the integral due to the discontinuity of q at θ. This prevents us from expressing the stochastic gradient of Monte Carlo samples with respect to θ in closed form and therefore optimizing with it.

We also investigate ignoring this limitation and performing the simplification of Equation (7) regardlessly, resulting in

$$\nabla_\theta D_{\mathrm{KL}}(p \,\|\, q;\theta) \approx \mathbb{E}\left[\begin{cases} p(X)\left(1 - \frac{Q_2}{Q_1}\right) & \text{if } X < \theta \\ p(X)\left(\frac{Q_1}{Q_2} - 1\right) & \text{otherwise} \end{cases} \right], \tag{12}$$

which has the same sign *regardless of the value of* θ, resulting in divergent behavior.

A similarly undesirable behavior emerges when normalizing q in a slightly different way by interpreting Q as probability masses rather than unnormalized densities:

$$q(x;\theta) = \begin{cases} Q_1/\theta & \text{if } x < \theta \\ Q_2/(1-\theta) & \text{otherwise.} \end{cases} \tag{13}$$

The KL divergence gradient is then

$$\begin{aligned} \nabla_\theta D_{\mathrm{KL}}(p \,\|\, q;\theta) &\approx \int_0^1 \begin{cases} p(x)/\theta & \text{if } x < \theta \\ p(x)/(\theta-1) & \text{otherwise,} \end{cases} \mathrm{d}x \\ &= \frac{1}{\theta}\int_0^\theta p(x)\,\mathrm{d}x - \frac{1}{1-\theta}\int_\theta^1 p(x)\,\mathrm{d}x\,. \end{aligned} \tag{14}$$

To illustrate the flawed nature of this gradient, consider the simple scenario of $p(x) = 1$, in which the RHS *always* equals to zero, suggesting *any* θ being a local minimum. However, θ clearly influences $D_{\mathrm{KL}}(p \,\|\, q;\theta)$ in this example, and therefore can not be optimal everywhere. Empirical investigations with other shapes of p, e.g. the examples from Figure 3, also suffer from a broken optimization and do not converge to a meaningful result.

While we only discuss a simplified setting here, the simplification in Equation (7) is also invalid in the *general* case of piecewise-linear coupling functions, likewise leading to a broken optimization.