GenDR: A Generalized Differentiable Renderer

Felix Petersen[1] Bastian Goldluecke[1] Christian Borgelt[2] Oliver Deussen[1]
[1]University of Konstanz [2]University of Salzburg

Abstract

In this work, we present and study a generalized family of differentiable renderers. We discuss from scratch which components are necessary for differentiable rendering and formalize the requirements for each component. We instantiate our general differentiable renderer, which generalizes existing differentiable renderers like SoftRas and DIB-R, with an array of different smoothing distributions to cover a large spectrum of reasonable settings. We evaluate an array of differentiable renderer instantiations on the popular ShapeNet 3D reconstruction benchmark and analyze the implications of our results. Surprisingly, the simple uniform distribution yields the best overall results when averaged over 13 classes; in general, however, the optimal choice of distribution heavily depends on the task.

1. Introduction

In the past years, many differentiable renderers have been published. These include the seminal differentiable mesh renderer OpenDR [1], the Neural 3D Mesh Renderer [2], and SoftRas [3] among many others. Using a differentiable renderer enables a multitude of computer vision applications, such as human pose estimation [4], camera intrinsics estimation [5], 3D shape optimization [2], 3D reconstruction [2], [3], [6], and 3D style transfer [2].

A fundamental difference between different classes of differentiable renderers is the choice of the underlying 3D representation. In this work, we focus on differentiable 3D mesh renderers [1]–[3], [6]; however, the aspects that we investigate could also be applied to other differentiable rendering concepts, such as rendering voxels [7], point clouds [8], surfels [9], signed distance functions [10], and other implicit representations [11], [12].

Differentiable mesh renderers can be constructed in different ways: either using an exact and hard renderer with approximate surrogate gradients or using an approximate renderer with natural gradients. Loper *et al*. [1] and Kato *et al*. [2] produce approximate surrogate gradients for their differentiable renderer, while their forward rendering is hard. In contrast, other differentiable renderers approximate the forward rendering in such a way that they produce a natural gradient. This can be achieved by modeling or approximating a renderer under a probabilistic perturbation, which is continuous and makes the renderer differentiable. For that, Rhodin *et al*. [13] model it with a Gaussian distribution, while Liu *et al*. [3] model it with the square root of a logistic distribution, Petersen *et al*. [14] use a logistic distribution, and Chen *et al*. [6] use the exponential distribution. While this variational interpretation of perturbing by a respective distribution is not stressed in some of these papers [3], [6], we believe it is important because it explicitly allows comparing the characteristics of the differentiable renderers. Moreover, the methods that only approximate gradients can also be seen as approximately modelling a perturbation: the gradient computed for the Neural 3D Mesh Renderer [2] is approximately a perturbation by a uniform distribution. Note that, here, the solutions for rendering under perturbations are obtained analytically in closed-form without sampling.

In this work, we introduce a generalized differentiable renderer (GenDR). By choosing an appropriate probability distribution, we can (at least approximately) recover the above differentiable mesh renderers, which shows that a core distinguishing aspect of differentiable renderers is the type of distributions that they model. The choice of probability distribution herein is directly linked to the sigmoid (i.e., S-shaped) function used for the rasterization. For example, a Heaviside sigmoid function corresponding to the Dirac delta distribution yields a conventional non-differentiable renderer, while a logistic sigmoid function of squared distances corresponds to the square root of a logistic distribution. Herein, the sigmoid function is the cumulative distribution function (CDF) of the corresponding distribution. In this work, we select and present an array of distributions and examine their theoretical properties.

Another aspect of approximate differentiable renderers is their aggregation function, i.e., the function that aggregates the occupancy probabilities of all faces for each pixel. Existing differentiable renderers commonly aggregate the probabilities via the probabilistic sum ($\perp^P(a,b) = a + b - ab$ or $1 - \prod_{t \in T}(1 - p_t)$), which corresponds to the probability that at least one face covers the pixel as-

suming that probabilities p_t for each triangle t are stochastically independent (cf. Eq. 4 in [3] or Eq. 6 in [6]). In the field of real-valued logics and adjacent fields, this is well-known as a T-conorm, a relaxed form of the logical 'or'. Two examples of other T-conorms are the maximum T-conorm $\perp^M(a,b) = \max(a,b)$ and the Einstein sum $\perp^E(a,b) = (a+b)/(1+ab)$, which models the relativistic addition of velocities. We generalize our differentiable renderer to use any continuous T-conorm and present a variety of suitable T-conorms.

In total, the set of resulting concrete instances arising from our generalized differentiable renderer and the proposed choices amounts to 1 242 concrete differentiable renderers. We extensively benchmark all of them on a shape optimization task and a camera pose estimation task. Further, we evaluate the best performing and most interesting instances on the popular ShapeNet [15] 13 class single-view 3D reconstruction experiment [2]. Here, we also include those instances that approximate other existing differentiable renderers. We note that we do not introduce a new shading technique in this paper, and rely on existing blended shaders instead.

We summarize our contributions as follows:

- We propose a generalized differentiable mesh renderer.
- We identify existing differentiable renderers (approximately) as instances of our generalized renderer.
- We propose a variety of suitable sigmoid functions and T-conorms and group them by their characteristics.
- We extensively benchmark 1 242 concrete differentiable renderers, analyze which characteristics and families of functions lead to a good performance, and find that the best choice heavily depends on the task, class, or characteristics of the data.

2. Related Work

The related work can be classified into those works that present differentiable renderers and those which apply them, although there is naturally also a significant overlap. For additional details on differentiable rendering approaches, cf. the survey by Kato *et al*. [16].

Analytical Differentiable Renderers. The first large category of differentiable renderers are those which produce approximate gradients in an analytical and sampling-free way. This can either happen by surrogate gradients during backpropagation, as in [2], or by making the forward computation naturally differentiable by perturbing the distances between pixels and triangles analytically in closed-form [6], [17], [18]. Our work falls into this category and is of the second case. Existing works each present their renderer for a specific distribution or sigmoid function. We formally characterize the necessary functions to a differentiable renderer and present an array of options.

Monte-Carlo Differentiable Renderers. An alternative to analytical differentiable renderers are those which are based on Monte-Carlo sampling techniques. The first example for this is the "redner" path tracer by Li *et al*. [19], who use edge sampling to approximate the gradients of their renderer. Loubet *et al*. [20] build on these ideas and reparameterize the involved discontinuous integrands yielding improved gradient estimates. Zhang *et al*. [21] extend these ideas by differentiating the full path integrals which makes the method more efficient and effective. Lidec *et al*. [22] approach Monte-Carlo differentiable rendering by estimating the gradients of a differentiable renderer via the perturbed optimizers method [23].

Applications. Popular applications for differentiable renderers are pose [1]–[3], [5], [6], [22], [24], shape [2], [18], [21], [24], material [25], [26], texture [3], [6], [20], and lighting [21] estimation. Here, the parameters of an initial scene are optimized to match the scene in a reference image or a set of reference images. Another interesting application is single-view 3D shape prediction without 3D supervision. Here, a neural network predicts a 3D representation from a single image, and the rendering of the image is compared to the original input image. This learning process is primarily guided by supervision of the object silhouette. It is possible to omit this supervision via adversarial style transfer [27]. Other applications are generating new 3D shapes that match a data set [28], [29] as well as adversarial examples in the real world [30].

In our experiments, we use optimization for pose and shape to benchmark *all* proposed differentiable renderer combinations. As the single-view 3D mesh reconstruction is a complex experiment requiring training a neural network, we benchmark our method on this task only for a selected subset of differentiable renderers.

T-norms and T-conorms. T-norms and T-conorms (triangular norms and conorms) are binary functions that generalize the logical conjunction ('and') and disjunction ('or'), respectively, to real-valued logics or probability spaces [31], [32]. A generalization of 'or' is necessary in a differentiable renderer to aggregate the occlusion caused by faces. The existing analytical differentiable renderers all use the probabilistic T-conorm.

3. Generalized Differentiable Renderer

In this section, we present our generalized differentiable mesh renderer. With a differentiable renderer, we refer to a renderer that is continuous everywhere and differentiable almost everywhere (a.e.). Note that, in this context, continuity is a stricter criterion than differentiable a.e. because formally (i) conventional renderers are already differentiable a.e. (which does not mean that they can provide any meaningful gradients), and (ii) almost all existing "differentiable" renderers are not actually differentiable everywhere.

Let us start by introducing how a classic hard rendering algorithm operates. The first step is to bring all objects into image space, which is typically a sequence of affine transformations followed by the camera projection. This step is already differentiable. The second step is the rasterization: For each pixel, we need to compute the set of faces (typically triangles) which cover it. If the pixel is covered by at least one face, the face that is closest to the camera is displayed.

3.1. Differentiable Occlusion Test

To make the test whether a pixel p is occluded by a face t differentiable, we start by computing the signed Euclidean distance $d(p,t)$ between pixel and face boundary. By convention, pixels inside the triangle have a positive distance, pixels outside the triangle a negative distance. For pixels exactly on the boundary, the distance to the face is 0.

For a hard occlusion test, we would just check whether $d(p,t)$ is non-negative. In a differentiable renderer, we instead introduce a perturbation in the form of a probability distribution with density f together with a temperature or scale parameter $\tau > 0$. We then evaluate the probability that the perturbed distance $d(p,t) - \tau\epsilon$ is non-negative, where ϵ is distributed according to f. Thus, we compute the probability that t occludes p as

$$\begin{aligned}\mathbb{P}_{\epsilon\sim f}(d(p,t) - \tau\epsilon \geq 0) &= \mathbb{P}_{\epsilon\sim f}(\epsilon \leq d(p,t)/\tau) \\ &= \int_{-\infty}^{d(p,t)/\tau} f(x)\,dx = F\left(\frac{d(p,t)}{\tau}\right),\end{aligned} \tag{1}$$

where F is the CDF of the distribution f and thus yields a closed-form solution for the desired probability (provided that F has a closed-form solution or can be appropriately approximated). In a differentiable renderer, we require F being continuous. Typically, F has the S-shape of a sigmoid function, see Table 1. Therefore, we refer to CDFs as sigmoid functions in this paper.

Most existing differentiable renderers use sigmoid functions or transformations thereof, see Section 4, to softly evaluate whether a pixel lies inside a triangle. This accords to the probabilistic interpretation in Equation (1) where the probability distribution is defined via the sigmoid function used in each case. Here, the logistic sigmoid function is a popular choice of such a sigmoid function. Note that, recently, it has frequently been referred to as "the" sigmoid in the literature, which is not to be confused with the original and more general terminology.

Example 1 (Logistic Sigmoid)**.** $F_L(x) = 1/(1+\exp(-x))$ *is the logistic sigmoid function, which corresponds to the logistic distribution.*

3.2. Aggregation

The second step to be made differentiable is the aggregation of multiple faces. While this is conventionally done via a logical 'or', the differentiable real-valued counterpart is a T-conorm. T-conorms are formally defined as follows.

Definition 2 (T-conorm)**.** *A T-conorm is a binary operation* $\perp : [0,1] \times [0,1] \to [0,1]$*, which satisfies*

- *associativity:* $\perp(a, \perp(b,c)) = \perp(\perp(a,b), c)$*,*
- *commutativity:* $\perp(a,b) = \perp(b,a)$*,*
- *monotonicity:* $(a \leq c) \wedge (b \leq d) \Rightarrow \perp(a,b) \leq \perp(c,d)$*,*
- 0 *is a neutral element* $\perp(a,0) = a$*.*

Remark 3 (T-conorms and T-norms)**.** *While T-conorms* $\perp$ *are the real-valued equivalents of the logical 'or', so-called T-norms* $\top$ *are the real-valued equivalents of the logical 'and'. Certain T-conorms and T-norms are dual in the sense that one can derive one from the other using a complement (typically* $1-x$*) and De Morgan's laws (*$\top(a,b) = 1 - \perp(1-a, 1-b)$*).*

Let us proceed by stating the T-conorm which is used in all applicable previous approximate differentiable renderers with natural gradients.

Example 4 (Probabilistic Sum)**.** *The* probabilistic sum *is a T-conorm that corresponds to the probability that at least one out of two independent events occurs. It is defined as*

$$\perp^P(a,b) = a + b - ab. \tag{2}$$

An alternative to this is the Einstein sum, which is based on the relativistic addition of velocities.

Example 5 (Einstein Sum)**.** *The* Einstein sum *is a T-conorm that corresponds to the velocity addition under special relativity:*

$$\perp^P(a,b) = \frac{a+b}{1+ab}. \tag{3}$$

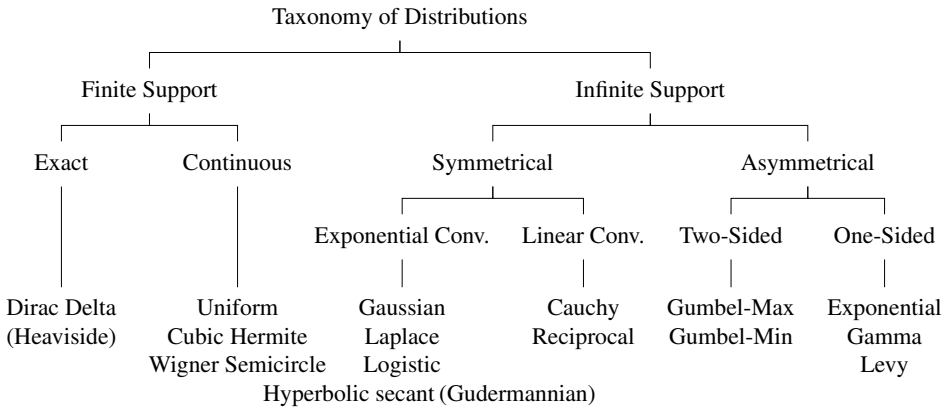
Combining the above concepts, we can compute the occupancy or coverage of a pixel p given a set of faces T as

$$\mathcal{A}_O(p,T) = \bigperp_{t\in T} F(\,d(p,t)/\tau\,). \tag{4}$$

3.3. Shading

The coloring of faces is handled via the Phong model or any other shading model, which is already differentiable. In the literature, Chen *et al*. [6] compare different choices. Finally, to aggregate the coloring of each pixel depending on the distance of the face to the camera (depth), there are two popular choices in the literature: no depth perturbations and taking the closest triangle (like [1], [2], [6]) and Gumbel depth perturbations (like [3], [18]). Only the latter choice is truly continuous, and the closed-form solution for Gumbel depth perturbations is the well known $\mathrm{softmin}$. As there are (i) no closed-form solutions for adequate alternatives to Gumbel perturbations in the literature, and (ii)

Figure 1. Taxonomy of probability distributions corresponding to sigmoid functions. The subdivisions are chosen wrt. properties that have a categorically different influence on the behavior of the corresponding renderer. The order of splits when going down in the tree (which could be chosen differently, e.g., symmetric/asymmetric could be the first split) reflects the importance of the properties.



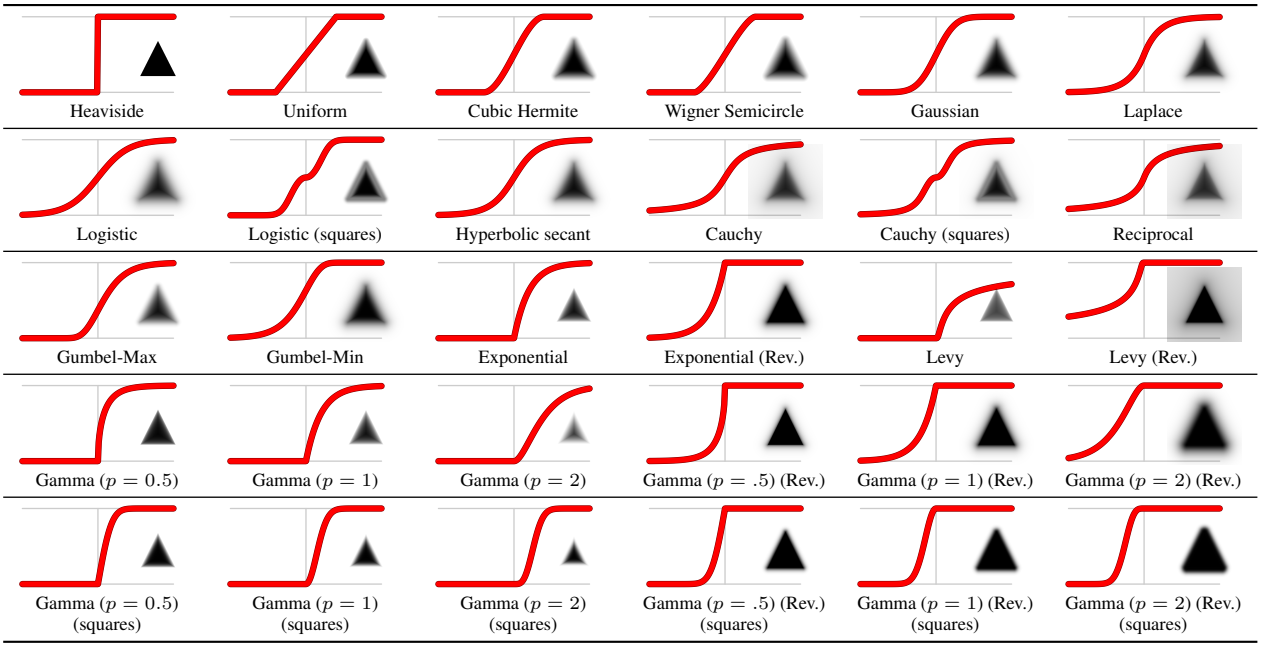


Table 1. Visualization of a selection of sigmoid functions, which are the CDFs of probability distributions. For each distribution, we display a single rendered triangle to demonstrate their different effects.

these two options have been extensively studied in the literature [1]–[3], [6], [18], [22], in this work we do not modify this component and focus on the differentiable silhouette computation and aggregation. While we implement both options in GenDR, in our evaluation, we perform all experiments agnostic to the choice of shading aggregation as the experiments rely solely on the silhouette.

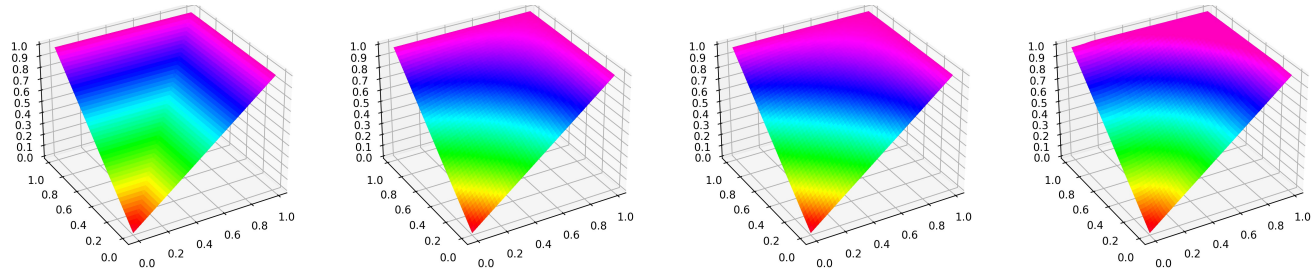
4. Instantiations of the GenDR

Let us proceed by discussing instantiations of the generalized differentiable renderer (GenDR).

Distributions. Figure 1 provides a taxonomy of the distributions and sigmoid functions that are visualized in Table 1. We classify the distributions into those with finite support as well as others with infinite support, where the support is the set of points for which the PDF is greater than zero. Note that the CDFs are constant outside the support region. Among the distributions with *finite support*, there is the *exact* Dirac delta distribution corresponding to the Heaviside function, which yields a discrete, non-differentiable renderer. There are also *continuous* distributions allowing meaningful gradients, but (due to finite support) only in a limited proximity to each face. Here, we have, among others, the uniform distribution, which corresponds to a piecewise linear step function. The derivative of the uniform distribution is equivalent or very similar (due to minor implementation aspects) to the surrogate gradient of the Neural 3D Mesh Renderer [2]. The distributions with *infinite support* can be categorized into symmetrical and asymmetrical. Among the symmetrical distributions, the Gaussian, the Laplace, the logistic, and the hyperbolic secant have an *exponential convergence* behavior or exponen-

T-conorm		equal to / where	continuous	contin. diff.	strict	idempotent	nilpotent	Archimedean	$\uparrow$ / $\downarrow$ wrt. p
(Logical 'or')	$\vee$		(✗)	(✗)	—	(✓)	—	—	—
Maximum	$\perp^M$		✓	✗	✗	✓	✗	✗	—
Probabilistic	$\perp^P$	$= \perp_1^H = \perp_1^A$	✓	✓	✓	✗	✗	✓	—
Einstein	$\perp^E$	$= \perp_0^H$	✓	✓	✓	✗	✗	✓	—
Hamacher	$\perp_p^H$	$p \in (0, \infty)$	✓	✓	✓	✗	✗	✓	$\downarrow$
Frank	$\perp_p^F$	$p \in (0, \infty)$	✓	✓	✓	✗	✗	✓	$\downarrow$
Yager	$\perp_p^Y$	$p \in (0, \infty)$	✓	✗	✗	✗	✓	✓	$\uparrow$
Aczél-Alsina	$\perp_p^A$	$p \in (0, \infty)$	✓	✓	✓	✗	✗	✓	$\uparrow$
Dombi	$\perp_p^D$	$p \in (0, \infty)$	✓	✓	✓	✗	✗	✓	$\uparrow$
Schweizer-Sklar	$\perp_p^{SS}$	$p \in (-\infty, 0)$	✓	✓	✓	✗	✗	✓	—

Table 2. Overview over a selection of suitable T-conorms, which we also benchmark.



(a) (b) (c) (d)

Figure 2. Plot of four selected T-conorms. From left to right: Maximum, Probabilistic, Einstein, and Yager (w/ $p = 2$). While (b) and (c) are smooth, the Yager T-conorm (d) is non-smooth, it plateaus and the value is constant outside the unit circle.

tial decay of probability density. On the other hand, there is also the Cauchy distribution which has a *linear convergence*. This yields a significantly different behavior. We include the algebraic function $x \mapsto x/(2 + 2|x|) + 1/2$ and call it reciprocal sigmoid. This also has a *linear convergence*. Finally, we consider *asymmetrical* distributions with infinite support. The Gumbel-Max and Gumbel-Min are extreme value distributions [33] and *two-sided*, which means that their support covers both positive and negative arguments. The exponential, Gamma, and Levy distributions are one-sided distributions. Here, it is important to not only consider the original distributions but also their mirrored or reversed variants, as well as shifted variations as can be seen in the last three rows of Table 1.

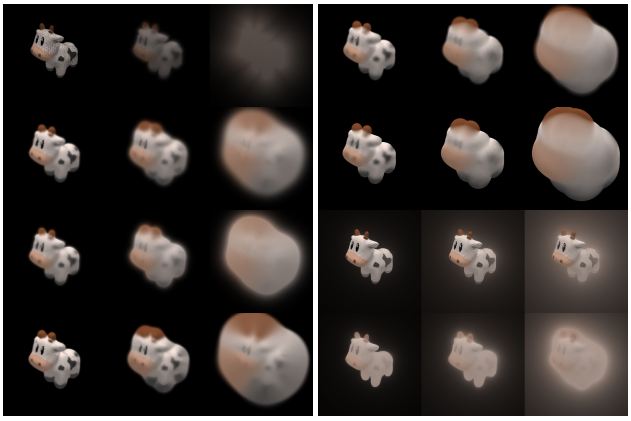


Figure 3. Visual comparison of different instances of GenDR. In each image, moving from left to right increases the temperature or scale τ of the distribution. Left: we use a logistic distribution to perturb the triangles and use different T-norms for aggregation (top to bottom: $\perp^M, \perp^P, \perp_2^Y, \perp_{0.5}^A$). Right: for the two first rows, we use a uniform distribution and use $\perp_2^Y$ and $\perp_{0.5}^A$. For the last two rows, we use a Cauchy distribution and use $\perp^P$ and $\perp_2^Y$.

SoftRas [3] squares the absolute part of the distance before applying the logistic sigmoid function and thus models the square roots of logistic perturbations. Instead of modifying the argument of F, we instead interpret it as applying a transformed counterpart CDF F_{sq}, which is more in line with the probabilistic interpretation in Equation (1). More precisely, we compute the occlusion probability as

$$F_{\text{sq}}(d(p,t)/\tau) := F(|d(p,t)| \cdot d(p,t)/\tau)\,. \tag{5}$$

That means that for each choice of F, we obtain a counterpart F_{sq}. A selection of these for different CDFs F is visualized in Table 1 denoted by "(squares)". For a mathematical definition of each sigmoid function, see SM B.

Aggregations. Table 2 provides an overview over selected T-conorms and displays their properties. The logical 'or' is not a T-conorm but the discrete and discontinuous equivalent, which is why we include it here. While there are also discontinuous T-conorms such as the drastic T-cornom, these are naturally not suitable for a differentiable renderer, which is why we exlude them. All except for the Max and Yager T-conorms are continuously differentiable.

Renderer	Distribution	T-conorm
OpenDR [1]	Uniform (backward)	—
N3MR [2]	Uniform (backward)	—
Rhodin *et al.* [13]	Gaussian	$\perp^P$
SoftRas [2]	Square-root of Logistic	$\perp^P$
Log. Relax [14]	Logistic	$\perp^P$
DIB-R [6]	Exponential	$\perp^P$

Table 3. Differentiable renderers that are (approximately) special cases of GenDR. OpenDR and N3MR do not use a specific T-conorm as their forward computation is hard.

The top four rows in Table 2 contain individual T-conorms, and the remainder are families of T-conorms. Here, we selected only suitable ranges for the parameter p. Note that there are some cases in which the T-conorms coincide, e.g, $\perp^P = \perp_1^H = \perp_1^A$. A discussion of the remaining properties and a mathematical definition of each T-conorm can be found in SM C. Figure 2 displays some of the T-conorms and illustrates different properties. In Figure 3, we display example renderings with different settings and provide a visual comparison on how the aggregation function affects rendering.

Existing Special Cases of GenDR. In Table 3, we list which existing differentiable renderers are conceptually instances of GenDR. These renderers do each have some other differences, but one key difference lies in the type of distribution employed. Differences regarding shading are also discussed at the end of Section 3.

5. Experiments[1]

5.1. Shape Optimization

Our first experiment is a shape optimization task. Here, we use the mesh of an *airplane*, and render it from 24 azimuths using a hard renderer. The task is to optimize a mesh (initialized as a sphere) to fit the silhouette of the airplane within 100 optimization steps. Limiting the task to 100 optimization steps is critical for two reasons: (i) The task can be considered to be solved perfectly with any differentiable renderer that produces the correct gradient sign within a large number of steps, but we are interested in the quality of the gradients for the optimization task and how efficient each renderer is. (ii) The total evaluation is computationally expensive because we evaluate a total of 1 242 renderers and perform a grid search over the distribution parameters for each one to provide a fair and reliable comparison.

Setup. For optimization, we use the Adam optimizer [34] with parameters $\beta_1 = 0.5, \beta_2 = 0.95$. For each setting, we perform a grid search over three learning rates ($\lambda \in \{10^{-1.25}, 10^{-1.5}, 10^{-1.75}\}$) and temperatures $\tau \in \{10^{-0.1 \cdot n} \,|\, n \in \mathbb{N}, 0 \leq n \leq 80\}$. Here, $\lambda = 10^{-1.5} \approx 0.03$

[1]The source code will be available at github.com/Felix-Petersen/gendr.

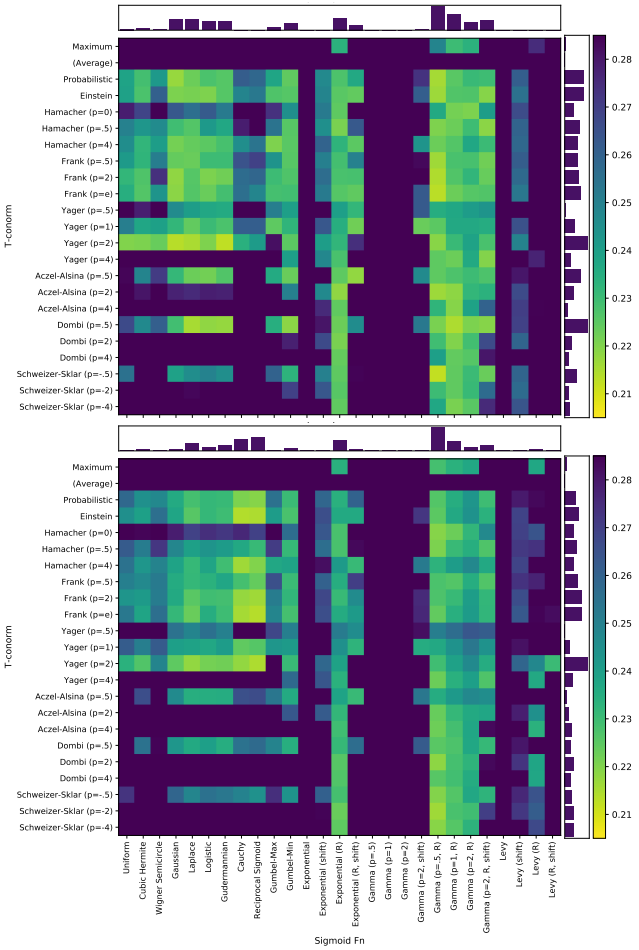


Figure 4. Results for the 24-view airplane shape optimization task. The optimization is done within a tight budget of 100 steps and the metric is the loss, i.e., lower (=yellow) is better. Top: original set of distributions F. Bottom: the respective counter-parts F_{sq} in the same location. The marginal histograms display participation in the top 10% combinations.

performs best in almost all cases. As for the scale hyperparameter, it is important to use a fine-grained as well as large grid because this behaves differently for each distribution. Here, we intentionally chose the grid larger than the range of reasonable values to ensure that the best choice is used for each setting; the extreme values for the scale were never optimal. We perform this evaluation from five different elevation angles $\{-60°, -30°, 0°, 30°, 60°\}$ as independent runs, and average the final results for each renderer instance. Additional results for the experiment applied to the model of a chair can be found in SM D.

Results. In Figure 4, we display the results of our evaluation. We can observe that the regular distributions F typically perform better than the counterpart F_{sq}, except for the case of Cauchy and reciprocal sigmoid, which are those with a linear convergence rate. We explain this by the fact that by squaring the distance before applying the

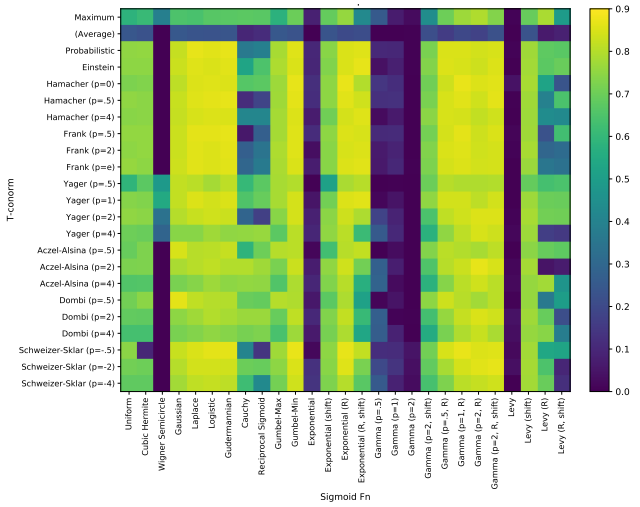


Figure 5. Results for the tea pot camera pose optimization task. The optimization is done with a temperature τ that is scheduled to decay. The metric is fraction of camera poses recovered, while the initialization angle errors are uniformly sampled from $[15°, 75°]$. The figure shows the original set of distributions F, the plot for the respective F_{sq} can be found in SM D.

sigmoid function, the function has a quadratic convergence rate instead. As the linearly converging functions also perform poorly in comparison to the exponentially converging functions (Gaussian, Laplace, Logistic, Gudermannian), we conclude that linear convergence is inferior to quadratic and exponential convergence. Columns $1-3$ contain the distributions with finite support, and these do not perform very well on this task. The block of exponentially decaying distributions (columns $4-7$) performs well. The block of linearly decaying distributions (columns $8-9$) performs badly, as discussed above. The block of Levy distributions (last 4 columns) performs even worse because it has an even slower convergence. Here, it also becomes slightly better in the squared setting, but it still exhibits worse performance than for linear convergence.

Comparison of Distributions. Gumbel, exponential, and gamma distributions do not all perform equally well, but Gumbel-Min, the reversed exponential, and the reversed gamma are all competitive. Confer Table 1 where it becomes clear that this is because Gumbel-Max, exponential and gamma have all of their mass inside the triangle, i.e., they yield smaller faces. This is problematic because in this case, it can cause gaps between neighboring triangles, which hinders optimization. As the reverse counterparts yield larger faces and do not suffer from this problem, they perform better. Note that, in this respect, the asymmetrical distributions have an advantage over the symmetrical distributions because symmetrical distributions always have an accumulated density of 0.5 at the edge, and thus the size of the face stays the same. We can see that, among the asymmetrical distributions, Gamma performs best.

Comparison of T-conorms. We find that $\perp^M$ and "average" (which is not a T-conorm but was used as a baseline in [3]) perform poorly. Also, $\perp_4^Y$, $\perp_2^A$, $\perp_4^A$, $\perp_2^D$, $\perp_4^D$, $\perp_{-2}^{SS}$, and $\perp_{-4}^{SS}$ perform poorly overall. This can be explained as they are rather extreme members of their respective T-norm families; in all of them, the pth power is involved, which can become a problematic component, e.g., x^4 is vanishingly small for $x = 0.5$. Interestingly, the gamma and the exponential distributions still perform well with these, likely since they are not symmetric and have an accumulated probability of 1 on the edge. Notably, the Yager T-conorm ($p = 2$) performs very well, although having a plateau and thus no meaningful gradient outside the unit disc, see Figure 2.

Finally, we compute histograms of how many times each respective distribution and T-conorm is involved in the best 10% of overall results. This is independent for the top and bottom plots. We can observe that Gamma ($p = 0.5$, Reversed) performs the best overall (because it is more robust to the choice of T-conorm). Among the T-conorms, we find that $\perp_2^Y$ and $\perp_{0.5}^D$ perform best. The probabilistic and Einstein sums perform equally, and share the next place.

5.2. Camera Pose Optimization

In our second experiment, the goal is to find the camera pose for a model of a *teapot* from a reference image. The angle is randomly modified by an angle uniformly drawn from $[15°, 75°]$, and the distance and camera view angle are also randomized. We sample 600 pairs of a reference image and an initialization and use this set of settings for each method. For optimization, we use Adam with a learning rate of either 0.1 or 0.3 (via grid search) and optimize for 1000 steps. During the optimization, we transition an initial scale of $\sigma = 10^{-1}$ logarithmically to a final value of $\sigma = 10^{-7}$. This allows us to avoid a grid search for the optimal scale, and makes sense since an initially large σ is beneficial for pose optimization, because a smoother model has a higher probability of finding the correct orientation of the object. This contrasts with the setting of shape estimation, where this would be fatal because the vertices would collapse to the center.

Results. In Figure 5, we display the results of this experiment. A corresponding image of the counterpart distributions F_{sq} as well as results for the experiment applied to the model of a chair can be found in SM D. The metric is the fraction of settings which achieved matching the ground truth pose up to $3°$. We find that in this experiment, the results are similar to those in the shape optimization experiment. Note that there are larger yellow areas because the color map ranges from 0% to 90%, while in the shape optimization plot the color map ranges in a rather narrow loss range.

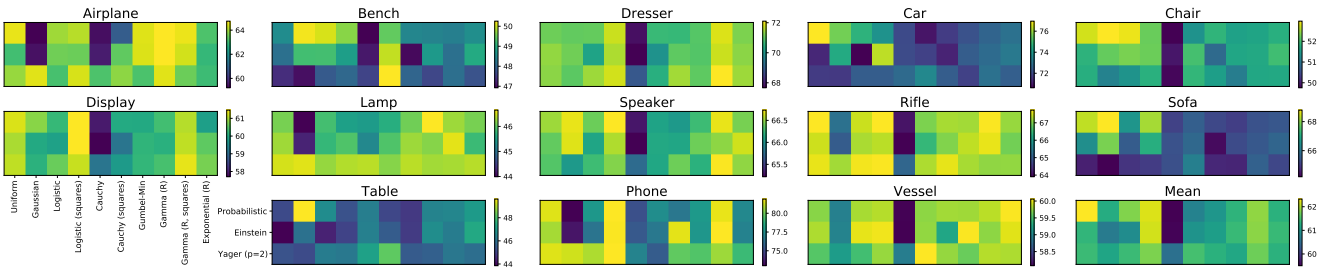


Figure 6. Single-view reconstruction results for each of the 30 selected renderers as a 3D IoU (in %) heatmap for each class. While the uniform distribution (w/ $\perp^P$) performs best on average and the square root of logistic (w/ $\perp^P$, $\perp^E$) performs second-best on average, the optimal setting depends on the characteristics of the respective classes. For the 'Airplane' class, the Gamma distribution performed best and this is also the distribution that performed best in the airplane shape optimization experiment in Section 5.1. For classes of furniture with legs, such as 'Bench', 'Chair', and 'Table', we find that the Gaussian distribution consistently performs best. The pairs of similar classes 'Display'+'Phone', 'Dresser'+'Speaker', and 'Vessel'+'Rifle' also show a similar performance patterns. For example, dresser and speakers tend to be cuboid, while rifles and vessels tend to be rather long and slim. Considering the Guassian distribution, it is interesting to see that for some classes $\perp^P$ and $\perp^E$ perform better, while for other classes $\perp_2^Y$ performs much better.

Method	Airplane	Bench	Dresser	Car	Chair	Display	Lamp	Speaker	Rifle	Sofa	Table	Phone	Vessel	*Mean*
Kato *et al.* [2] N3MR (Uniform Backward)	0.6172	0.4998	0.7143	0.7095	0.4990	0.5831	0.4126	0.6536	0.6322	0.6735	0.4829	0.7777	0.5645	0.6015
Liu *et al.* [3] SoftRas (Square-root of Logistic)	0.6419	0.5080	0.7116	0.7697	0.5270	0.6156	0.4628	0.6654	0.6811	0.6878	0.4487	0.7895	0.5953	0.6234
Chen *et al.* [6] DIB-R (Exponential)	0.570	0.498	0.763	0.788	0.527	0.588	0.403	0.726	0.561	0.677	0.508	0.743	0.609	0.612
Probabilistic + Uniform (≈[1], [2])	0.6456	0.4855	0.7113	**0.7696**	0.5276	0.6126	0.4611	0.6651	**0.6773**	0.6835	0.4514	0.8148	0.5971	**0.6232**
Probabilistic + Logistic (=[14])	0.6396	0.5005	0.7105	0.7471	0.5288	0.6022	0.4586	0.6639	0.6742	0.6660	0.4666	0.7771	0.5980	0.6179
Probabilistic + Logistic (squares) (=[3])	0.6416	0.4966	**0.7175**	0.7386	0.5224	**0.6147**	0.4550	**0.6673**	**0.6771**	0.6818	0.4529	**0.8186**	0.5984	0.6217
Probabilistic + Exponential (R) (=[6])	0.6321	0.4857	0.7123	0.7298	0.5178	0.5983	0.4611	0.6642	0.6713	0.6546	0.4700	0.7717	**0.6005**	0.6130
Probabilistic + Gaussian (≈[13])	0.5922	**0.5020**	0.7104	0.7561	**0.5297**	0.6080	0.4399	**0.6668**	0.6533	**0.6879**	**0.4961**	0.7301	0.5894	0.6125
Probabilistic + Gamma (R)	**0.6473**	0.4842	0.7093	0.7220	0.5159	0.6033	**0.4665**	0.6626	0.6719	0.6505	0.4642	0.7778	0.5978	0.6133
Einstein + Gamma (R, squares)	0.6438	0.4816	**0.7174**	0.7284	0.5170	0.6111	0.4654	0.6647	0.6760	0.6546	0.4626	**0.8189**	0.5973	0.6184
Yager (p=2) + Cauchy (squares)	0.6380	**0.5026**	0.7047	0.7359	0.5188	0.5976	0.4617	0.6612	0.6726	0.6619	0.4819	0.7560	**0.6006**	0.6149

Table 4. Selected single-view reconstruction results measured in 3D IoU.

5.3. Single-View 3D Reconstruction

Setup. Finally, we reproduce the popular ShapeNet single-view 3D reconstruction experiment from [2], [3], [6], [14]. We select three T-conorms ($\perp^P$, $\perp^E$, $\perp_2^Y$) and 10 distributions (Uniform, Gaussian, Logistic, Logistic (squares), Cauchy, Cauchy (squares), Gumbel-Min, Gamma (R, $p = 0.5$), Gamma (R, $p = 0.5$, squares), and Exponential (R)). These have been selected because they have been used in previous works, are notable (Cauchy, Gumbel-Min, Einstein), or have performed especially well in the aircraft shape optimization experiment (Gamma, Yager). For each setting, we perform a grid search of τ at resolution $10^{0.5}$. Further experimental details can be found in SM A.

Results. In Figure 6, we display and discuss the class-wise results for all 30 selected renderers. In Table 4, we show the (self-) reported results for existing differentiable renderers in the top block. In the bottom block, we display our results for the methods that are equivalent (=) or very similar (≈) to the six existing differentiable renderers. The differences for equivalent methods can be explained with small variations in the setting and minor implementation differences. Additionally, we include three noteworthy alternative renderers, such as the one that also performed best on the prior airplane shape optimization task. We conclude that the optimal choice of renderer heavily depends on the characteristics of the 3D models and the task. Surprisingly, we find that the simple uniform method achieves consistently good results and the best average score.

6. Discussion and Conclusion

In this work, we generalized differentiable mesh renderers and explored a large space of instantiations of our generalized renderer GenDR. We found that there are significant differences between different distributions for the occlusion test but also between different T-conorms for the aggregation. In our experiments, we observed that the choice of renderer has a large impact on the kind of models that can be rendered most effectively. We find that the uniform distribution outperforms the other tested distributions on average, which is surprising considering it simplicity. Remarkably, the uniform distribution had already been used implicitly for the early surrogate gradient renderers but was later discarded for the approximate differentiable renderers.

Acknowledgments. This work was supported by the DFG in the Cluster of Excellence EXC 2117 (Project-ID 390829875) and the SFB Transregio 161 (Project-ID 251654672), and the Land Salzburg within the WISS 2025 project IDA-Lab (20102-F1901166-KZP and 20204-WISS/225/197-2019).

References

[1] M. M. Loper and M. J. Black, "OpenDR: An approximate differentiable renderer," in *Proc. European Conference on Computer Vision (ECCV)*, 2014.

[2] H. Kato, Y. Ushiku, and T. Harada, "Neural 3D mesh renderer," in *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[3] S. Liu, T. Li, W. Chen, and H. Li, "Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning," in *Proc. International Conference on Computer Vision (ICCV)*, 2019.

[4] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, "Keep it smpl: Automatic estimation of 3d human pose and shape from a single image," in *Proc. European Conference on Computer Vision (ECCV)*, 2016.

[5] A. Palazzi, L. Bergamini, S. Calderara, and R. Cucchiara, "End-to-end 6-DoF object pose estimation through differentiable rasterization," in *Proc. European Conference on Computer Vision Workshops (ECCVW)*, 2019.

[6] W. Chen, J. Gao, H. Ling, *et al.*, "Learning to predict 3D objects with an interpolation-based differentiable renderer," in *Proc. Neural Information Processing Systems (NeurIPS)*, 2019.

[7] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective transformer nets: Learning single-view 3D object reconstruction without 3D supervision," in *Proc. Neural Information Processing Systems (NeurIPS)*, 2016.

[8] E. Insafutdinov and A. Dosovitskiy, "Unsupervised learning of shape and pose with differentiable point clouds," in *Proc. Neural Information Processing Systems (NeurIPS)*, 2018.

[9] W. Yifan, F. Serena, S. Wu, C. Öztireli, and O. Sorkine-Hornung, "Differentiable surface splatting for point-based geometry processing," *ACM Transactions on Graphics*, vol. 38, no. 6, 2019.

[10] Y. Jiang, D. Ji, Z. Han, and M. Zwicker, "SDFDiff: Differentiable rendering of signed distance fields for 3D shape optimization," in *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] S. Liu, S. Saito, W. Chen, and H. Li, "Learning to infer implicit surfaces without 3d supervision," 2019.

[12] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations," in *Proc. Neural Information Processing Systems (NeurIPS)*, 2019.

[13] H. Rhodin, N. Robertini, C. Richardt, H.-P. Seidel, and C. Theobalt, "A versatile scene model with differentiable visibility applied to generative pose estimation," in *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[14] F. Petersen, C. Borgelt, H. Kuehne, and O. Deussen, "Learning with Algorithmic Supervision via Continuous Relaxations," in *Proc. Neural Information Processing Systems (NeurIPS)*, 2021.

[15] A. X. Chang, T. Funkhouser, L. Guibas, *et al.*, "ShapeNet: An information-rich 3D model repository," *Computing Research Repository (CoRR) in arXiv*, 2015.

[16] H. Kato, D. Beker, M. Morariu, *et al.*, "Differentiable rendering: A survey," *arXiv preprint arXiv:2006.12057*, 2020.

[17] H.-T. D. Liu, M. Tao, C.-L. Li, D. Nowrouzezahrai, and A. Jacobson, "Adversarial geometry and lighting using a differentiable renderer," *Computing Research Repository (CoRR) in arXiv*, 2018.

[18] F. Petersen, A. H. Bermano, O. Deussen, and D. Cohen-Or, "Pix2Vex: Image-to-Geometry Reconstruction using a Smooth Differentiable Renderer," *arXiv preprint arXiv:1903.11149*, 2019.

[19] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, "Differentiable Monte Carlo ray tracing through edge sampling," *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, vol. 37, no. 6, pp. 1–11, 2018.

[20] G. Loubet, N. Holzschuch, and J. Wenzel, "Reparameterizing discontinuous integrands for differentiable rendering," *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 2019.

[21] C. Zhang, B. Miller, K. Yan, I. Gkioulekas, and S. Zhao, "Path-space differentiable rendering," in *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2020.

[22] Q. L. Lidec, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable rendering with perturbed optimizers," in *Proc. Neural Information Processing Systems (NeurIPS)*, 2021.

[23] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach, "Learning with Differentiable Perturbed Optimizers," in *Proc. Neural Information Processing Systems (NeurIPS)*, 2020.

[24] N. Ravi, J. Reizenstein, D. Novotny, *et al.*, "Accelerating 3d deep learning with pytorch3d," *arXiv preprint arXiv:2007.08501*, 2020.

[25] G. Liu, D. Ceylan, E. Yumer, J. Yang, and J.-M. Lien, "Material editing using a physically based rendering network," in *Proc. International Conference on Computer Vision (ICCV)*, 2017.

[26] L. Shi, B. Li, M. Hašan, *et al.*, "MATch: Differentiable material graphs for procedural material capture," in *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2020.

[27] F. Petersen, B. Goldluecke, O. Deussen, and H. Kuehne, "Style Agnostic 3D Reconstruction via Adversarial Style Transfer," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[28] P. Henzler, N. Mitra, and T. Ritschel, "Escaping plato's cave using adversarial training: 3D shape from unstructured 2D image collections," in *Proc. International Conference on Computer Vision (ICCV)*, 2019.

[29] C. H. L. Paul Henderson Vagia Tsiminaki, "Leveraging 2d data to learn textured 3d mesh generation," in *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[30] H.-T. D. Liu, M. Tao, C.-L. Li, D. Nowrouzezahrai, and A. Jacobson, "Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer," in *International Conference on Learning Representations (ICLR)*, 2019.

[31] E. P. Klement, R. Mesiar, and E. Pap, *Triangular norms*. Springer Science & Business Media, 2013, vol. 8.

[32] E. van Krieken, E. Acar, and F. van Harmelen, "Analyzing differentiable fuzzy logic operators," *Artificial Intelligence*, vol. 302, p. 103 602, 2022.

[33] S. Coles, *An introduction to statistical modeling of extreme values*, ser. Springer Series in Statistics. Springer-Verlag, 2001.

[34] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.