DOI: 10.1111/cgf.14754
EUROGRAPHICS 2023 / K. Myszkowski and M. Nießner
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 2

Learning to Learn and Sample BRDFs

Chen Liu Michael Fischer Tobias Ritschel

University College London

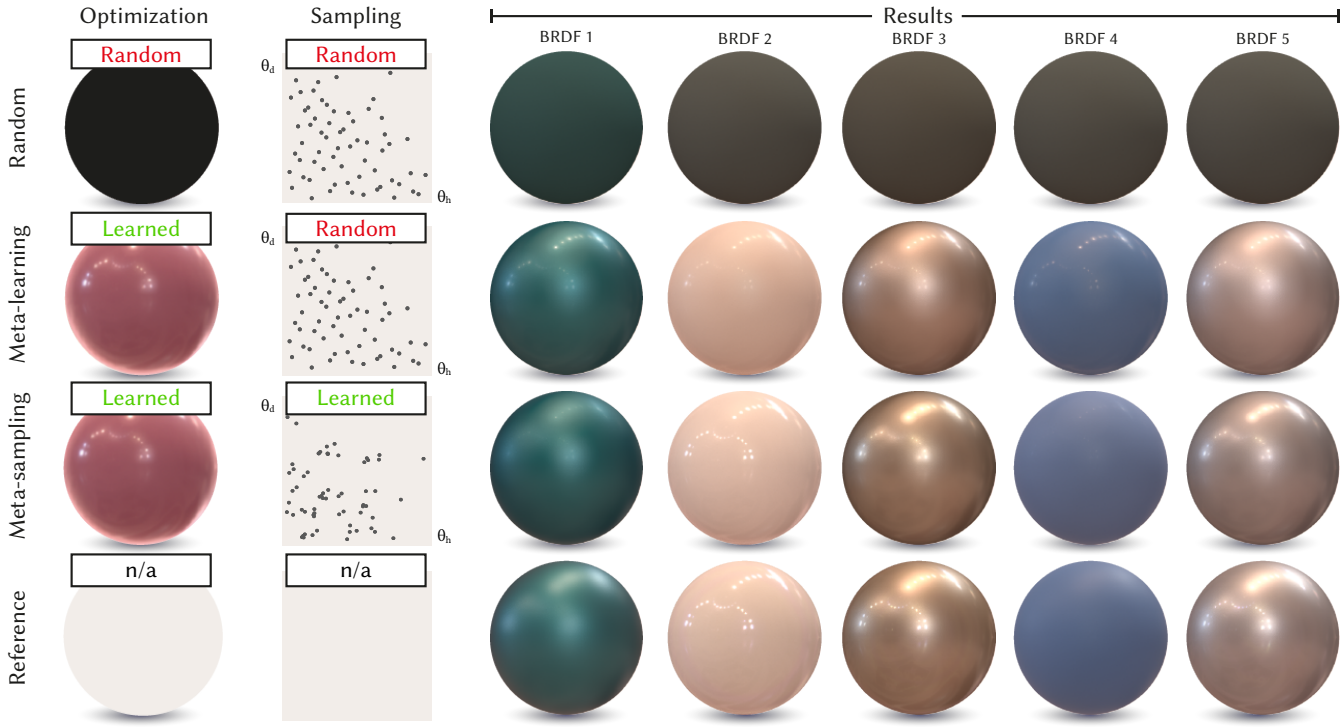

Figure 1: *Equal-time comparison between a neural BRDF model fitted without meta-learning **(top)**, fitted with meta-learning **(second row)** and fitted with our meta-sampling **(third row)**, all at 64 BRDF acquisition samples. Meta-sampling improves the visual quality, as seen right: for the same compute and acquisition time at deployment, the third row is closer to the reference in the fourth row than the second row.*

Abstract

We propose a method to accelerate the joint process of physically acquiring and learning neural Bi-directional Reflectance Distribution Function (BRDF) models. While BRDF learning alone can be accelerated by meta-learning, acquisition remains slow as it relies on a mechanical process. We show that meta-learning can be extended to optimize the physical sampling pattern, too. After our method has been meta-trained for a set of fully-sampled BRDFs, it is able to quickly train on new BRDFs with up to five orders of magnitude fewer physical acquisition samples at similar quality. Our approach also extends to other linear and non-linear BRDF models, which we show in an extensive evaluation.

1. Introduction

Learned representations of BRDFs [NRH*92] offer intuitive editing, compact storage or interpolation of material appearance [RJGW19, HGC*20, RGJW20, SRRW21, FWH*21]. What neural BRDF models so far do not offer is a way to accelerate acquisition. Acquisition is slow, because it is a physical process, where

a device has to change the illumination and capture an optical response, involving mechanical effort. Therefore, the simplest way to decrease capture time is to take fewer acquisition samples. Reducing the number of samples was investigated for linear BRDF models [NJR15, NDM05].

In this work, we reduce the number of BRDF acquisition samples by jointly learning the sample pattern and a non-linear, deep, neural-network based BRDF model. We use meta-learning [FAL17] to optimize the hyper-parameters of an optimization. Furthermore, we extend the Metappearance approach [FR22] to also meta-optimize over the sample pattern ("meta-sampling"), reducing the sample count by five orders of magnitude at similar visual quality. Finally, we show that our idea is applicable to Neural Networks (NNs) as well as to classic models, such as Phong or mixtures of basis BRDFs. Our code is available at `https://github.com/ryushinn/meta-sampling`.

2. Previous Work

BRDFs Classic BRDF models include Phong [Pho75], Cook-Torrance [CT82], Ward [LFTG97] or Disney's shading model [BS12]. These models are compact to store, lend themselves well to manipulation, but face limitations when it comes to reproducing captured materials.

Acquisition Gonioreflectometers measure the reflectance, depending on the light and view direction [Erb80, WSB*98, LFTW06, McA02] but this process remains slow, as it requires the mechanical change of light and sensor position. For spherical objects, this process can be accelerated by imaging all normals at the same time [MWL*99]. We consider acquisition a black box that requires effort (time, energy, heat, etc) linear in the number of acquisition samples. Our aim is to reduce this effort.

BRDF acquisition setups have led to the construction of BRDF databases [NDM05, Mat03], which we will rely on in this work.

Fitting parametric BRDF models requires an optimization [LKG*03, NLGK18], often with complex target functions, a (differentiable) image formation model and many resulting non-linearities. We operate on another layer of abstraction, and ask how to automatically tune this optimization on some training BRDFs, together with the BRDF acquisition's sampling so they jointly perform best on new test BRDF optimizations, i.e., on unseen tasks.

Optimizing acquisition Several approaches have sought to reduce the BRDF capture time, for example using adaptive sampling [FBLS07, DJ18]. Most related to our work is the linear statistical analysis of a set of BRDFs [NJR15]. Authors optimize for a sample pattern, assuming the BRDF they wish to reconstruct can be expressed as a linear combination of basis BRDFs, found using Principal Component Analysis (PCA). [LRR04] and [DJ18] derive BRDF models that also lead more efficient acquisition. Acquisition can be accelerated further when using a more principled objective function [BP20]. Similar ideas were proposed for spatially-varying BRDFs (svBRDFs) [ZCD*16, YXM*16] and for Bi-directional Texture Functions (BTFs) [dBWK18]. Our work differs in that it targets non-linear, deep, representation of BRDFs.

For svBRDFs, light patterns have been optimized together with an auto-encoder for reconstruction [KXH*19, KCW*18]. Similar ideas apply to image-based relighting [XSHR18], related to BRDFs.

Deep BRDF representation Recently, methods have been proposed to represent the BRDF itself by a neural network [RJGW19, HGC*20, RGJW20, SRRW21, FWH*21]. These methods can be more expressive and offer improved editing or interpolation properties. However, fitting them to a new BRDF can be time-consuming for two reasons: first, a lengthy optimization is required, and second, the fitting process makes use of many BRDF samples. For example, [SRRW21] use over 8×10^5 samples to learn a BRDF instance and [HGC*20] even use 100% of the measurements in a BRDF. In our work we combine the idea of optimizing the optimization with also optimizing over the sampling.

Deep material acquisition A popular approach to speed up acquisition of (sv)BRDFs is to learn a mapping from images to BRDFs, either supervised [RRF*16, GRR*17, DDB20, DAD*18, LCY*17], or with some level of self-supervision and differentiable rendering in the mix [PHS20, GLD*19, GSH*20, HDMR21]. These methods produce parameters to classic BRDF models (and inherit their limitations), while we produce a NN that represents the BRDF itself.

Meta-learning From the previous paragraphs, we see that two schools exist on how to represent and acquire BRDFs: either by learning feed-forward networks, typically a CNN that maps images to parameters, or by running optimizations on many carefully calibrated measurements. The first is fast but with limited quality, the second takes longer, but provides better quality. One proposal to bridge this gap is meta-learning. It uses an optimization at test-time to fit to observations, but this optimization has been optimized on a training set of many optimization tasks [FAL17]. This idea has been applied in computer vision [SCT*20, WMM*21, BKW21, TMW*21] and also to visual appearance [MLTFR19, FR22]. These methods assume the training data given and then learn how to optimize on it. In this work we take this further, and also optimize over what the training data needs to be. So instead of learning to optimize with a given set of BRDF samples, we also learn what BRDF samples to take to then fit successfully.

Sampling strategies Learning to sample is not novel in the deep learning community and research on the order of feeding samples can date back to the framework of Curriculum Learning [BTPG16]. [SRJ17, ZZ15, NWS14, KF19] wish to learn an ideal probability distribution, from which Stochastic Gradient Descent (SGD) draws training samples in terms of importance sampling, so the stochastic gradient with limited samples will benefit from the samples' reduced variance. They are expected to obtain the (sub)optimal approximation of the unbiased gradients given a set of samples, but there is no guarantee that the unbiased first-order gradient would guide the best path of optimization.

Active or online sampling research [SS18, DPD22] chooses to generate or label the next sample based on metrics inspired by Curriculum Learning, e.g., gradient norm or loss, to allocate more training time to the samples that will reduce the loss the most. The sampling pattern we find transfers across problem instances and does not need to be re-learned. We operate at a regime of extremely few samples and use orders of magnitudes fewer samples than the aforementioned methods.

3. Meta-sampling

We will first recall classic fitting of a BRDF (Sec. 3.1), then how meta-learning extends this (Sec. 3.2) and finally introduce our contribution: joint meta-learning of the BRDF model's parameters and the samples to take (Sec. 3.3).

Algorithm 1 Learning to learn and to sample BRDFs. The function GRAD computes the gradient of the first argument w.r.t. the second.

```
procedure LEARNTOLEARNANDSAMPLE(𝒯)
    ϕ = UNIFORM()
    ξ = UNIFORM()
    for i ∈ [1, n_o] do
        T = SAMPLETASK(𝒯)
        θ = LEARN(ϕ, ξ, T)
        ϕ = ϕ − Δ_o · GRAD(EVALUATE(θ, T), ϕ)
    end for
    for i ∈ [1, n_s] do
        T = SAMPLETASK(𝒯)
        θ = LEARN(ϕ, ξ, T)
        ξ = ξ − Δ_s · GRAD(EVALUATE(θ, T), ξ)
    end for
    return ϕ, ξ
end procedure

procedure LEARN(ϕ, ξ, T)
    θ = ϕ_init
    for i ∈ [1, n_l] do
        x = SAMPLE(ξ)
        θ = θ − ϕ_step · GRAD(LOSS(x, θ, T), θ)
    end for
    return θ
end procedure

procedure EVALUATE(θ, T)
    return LOSS(SAMPLE(UNIFORM()), θ, T)
end procedure

procedure LOSS(x, θ, T)
    c = cos(x.θ_i)
    return |log(1 + T(x) · c) − log(1 + f_r(x; θ) · c)|
end procedure
```

3.1. Random

We denote classic learning as "Random", as the training samples are drawn uniformly and the model is randomly initialized. It uses the function LEARN in Alg. 1, which is provided with hyper-parameters ϕ (e.g., step size, initialization), parameters of a sampling method ξ and a BRDF we want to learn T[†]. It returns the model parameters θ that best encode this BRDF T.

The function LEARN implements learning via stochastic gradient descent: At each learning iteration i, $i \in \{1,2,...,n_l\}$, the sampler, parameterized by ξ, generates samples $\mathbf{x}$, which are pairs of incoming and outgoing 3D directions in a suitable parametrization (we use Rusinkiewicz angles [Rus98]). The function LOSS then queries the BRDF supervisions and compares it with model's predictions for those samples. We adopt the mean absolute logarithmic error of the cosine weighted reflectance values proposed in [SRRW21] as our loss and descend along the loss gradient to update the parameters θ, usually until convergence (i.e., in classic learning, n_l is large). This chain of update steps starts at a certain model initialization (e.g., Kaiming-He) and uses a certain learning rate, or step-size, both defined by the hyper-parameter vector ϕ.

Classic learning usually does not have the capability of automatically determining good parameters for neither ϕ nor ξ. Usually, hyperparameters like learning rate and model initialization are determined empirically, and samples are drawn randomly from the dataset. We will next look into the first issue, while solving the latter is the contribution of this paper.

3.2. Meta

Meta-learning is depicted in orange in Alg. 1. Gradient-based meta-learning (we use the MAML algorithm [FAL17]) relies on a nested optimization, where the inner optimization loop is tasked with overfitting the model onto a specific BRDF T under the constraint of a very limited number of gradient steps (typically around 10 only, i.e., n_l is small). After completion of this inner loop, the model's final performance is evaluated through the function EVALUATE, which calculates losses at a set of uniform-random test samples. The outer loop then computes the gradient of the performance w.r.t. the meta-parameters ϕ (commonly, learning rate and model initialization), and moves these in a direction that will yield improved performance of the next inner-loop iteration. The outer loop subsequently samples a new task T from a set of tasks $\mathcal{T}$ and the inner loop starts anew, thereby ensuring parameters ϕ that will generalize across all tasks in $\mathcal{T}$.

We follow the approach of Metappearance [FR22] who meta-learn the initialization and step size for a deep BRDF representation. Note that the sampling pattern ξ is still random.

3.3. Meta-Learned sampling

With these steps laid out, we can now define our contribution in blue. We repeat the meta-construction from Sec. 3.2, but now include the parameters of our sampling method ξ in the meta-optimization. To do so, we use ξ to sample a set of n BRDF entries (i.e., obtain many supervision pairs of angle pairs $\mathbf{x}$ and BRDF values $\mathbf{y}$) at the start of the inner loop, all from the same BRDF task. These are the only samples that the learner will have access to within this inner loop completion, and they are fixed throughout the inner loop. Hence, the final model's performance is directly related to the samples produced by ξ, which now allows to compute a meta-gradient w.r.t. ξ. As previously done with ϕ, we use this meta-gradient to adjust ξ such that during the next iteration of the inner loop, the final performance will improve. This is repeated n_s times. Typically n_s is much larger than n, i.e., the meta-sample-learning will see many tasks (training BRDFs) for multiple times (epochs).

Experimentally, we found it necessary to decouple the optimization loops of the optimizer's parameters ϕ and the sampling method

[†] In the meta-learning literature, learning a single BRDF would be called a "task", hence the symbol.

ξ to stabilize training and hence run them consecutively (L. 4-8 and L. 9-13 in Alg. 1, respectively).

Note that this is more than simply augmenting ϕ by some additional dimensions as the samples are not part of the inner-loop optimization (recall, they are sampled once and then fixed), but part of the *supervision* that drives the inner-loop optimization. So we do not only change the way we learn, given the problem, but we also change the problem (i.e. which samples are chosen from the BRDF) such that it can be learned better. We hence aim to discover the subset of BRDF samples that is most informative to our learner.

The sampling method ξ can be parameterized in multiple ways. During classic- and meta-learning, as already mentioned, ξ is a random generator without any learnable parameters, and SAMPLE(ξ) simply returns random uniform numbers. For our learned sampling, we went with the most direct approach and parameterized ξ as an explicit n-dimensional vector of sample coordinates, which we meta-initialized with a low-discrepancy sequence in 3D. ξ then is a set, and not a sequence, as batched SGD averages over gradients from all samples, which is a symmetric operation that makes order irrelevant. Meta-learning ξ then becomes as simple as moving the sample coordinates in small steps, so that after each step, ξ becomes slightly more useful to the inner-loop BRDF learner.

Enforcing valid samples. Not all vectors $\mathbf{x}$ in the unit cube $[0,1]^3$ are valid Rusinkiewicz samples, as some configurations result in views below the horizon (the white regions in Fig. 2). As a consequence, optimizing ξ might result in individual samples moving into these invalid regions. Simply redefining the BRDF to have a specific constant value in these regions (e.g., -1.0) leads to areas where the gradient w.r.t. the sample position is zero. This will ultimately result in wasted samples that cannot adjust any longer, as a sample loses all gradient information as soon as it reaches such a constant region and cannot recover. Instead of redefining the loss, we opt to extend it by a barrier function LOSS_B that forces invalid samples back into the defined regions. One way to achieve this is to penalize their distance to the origin in the ϕ_d-plane (Fig. 2), as in

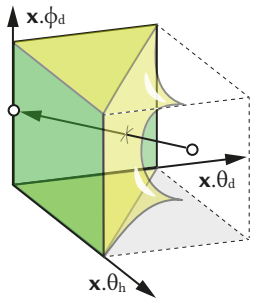

Figure 2: *Projection.*

$$\text{LOSS}_\text{B}(\mathbf{x},\theta,T) = \begin{cases} \text{LOSS}(\mathbf{x},\theta,T) & \text{if } \mathbf{x} \text{ is valid, and} \\ \mathbf{x}.\theta_\text{h}^2 + \mathbf{x}.\theta_\text{d}^2 & \text{if else.} \end{cases} \quad (1)$$

This function, applied to all individual samples, is used as a drop-in replacement for LOSS in Alg. 1. Also, all the random uniform samples mentioned above and henceforth are in fact uniform in the valid region, rather than in the cube, which can be achieved by rejecting invalid samples.

3.4. Models

To underline the generality of our proposed algorithm, we show that meta-sampling can increase performance on four increasingly complex BRDF models (**PHONG**, **COOK-TORRANCE**, **LINEAR** and **NEURAL**). We will now shortly describe these models.

Phong One of the simplest, yet widely used BRDF models is the physical version [LW94] of the **PHONG** [Pho75] model:

$$f_\text{r}(\mathbf{x}) = k_\text{d}\frac{1}{\pi} + k_\text{s}\frac{q+2}{2\pi}\max(\langle \mathbf{x}.\omega_\text{o}, \mathbf{x}.\mathbf{r}\rangle, 0.0)^q, \quad (2)$$

where $\langle\cdot\rangle$ denotes the dot product between outgoing and reflected direction $\mathbf{x}.\omega_\text{o}$ and $\mathbf{x}.\mathbf{r}$, respectively. We re-parametrize k_d and k_s by

$$\begin{aligned} k_\text{d} &= k_\text{sum} k_\text{ratio} \\ k_\text{s} &= k_\text{sum}(1-k_\text{ratio}), \text{ where} \\ k_\text{sum} &= \sigma(\lambda_\text{sum}) \quad \text{and} \quad k_\text{ratio} = \sigma(\lambda_\text{ratio}). \end{aligned}$$

σ is the Sigmoid function and hence ensures $k_\text{d} + k_\text{s} \leq 1$. Furthermore, we linearize q by means of an exponential mapping. The learnable parameters hence are $\lambda_\text{sum}, \lambda_\text{ratio} \in \mathbb{R}^3$ and the scalar glossiness q. We meta-learn their initial values and a learning rate per parameter.

Cook-Torrance The **PHONG** model is easy to implement and cheap to evaluate, but often does not produce realistic appearance, which is why we include the more sophisticated **COOK-TORRANCE** [CT82] model in our experiments. **COOK-TORRANCE** explicitly defines the characteristics of a surface's normal distribution N (we use Beckmann), the geometric attenuation G in the surface, and the Fresnel effect F, for which we use Schlick's approximation [Sch94]. As also implemented in [NDM05], we compute the reflectance as

$$f_\text{r}(\mathbf{x}) = k_\text{d}\frac{1}{\pi} + k_\text{s}\frac{D(\alpha,\mathbf{x})G(\mathbf{x})F(F_0,\mathbf{x})}{\pi\cos(\mathbf{x}.\theta_\text{i})\cos(\mathbf{x}.\theta_\text{o})}. \quad (3)$$

The learnable parameters are k_d, k_s, roughness α, and the Fresnel value F_0. We use a Sigmoid to constrain them within (0, 1).

Linear The next higher level of complexity is a **LINEAR** model, as proposed by [NDM05] and refined by [NJR15]. Here, every BRDF is a linear combination of m basis BRDFs:

$$f_\text{r}(\mathbf{x}) = (\mathsf{A}\cdot(\mathbf{w}|1)^\mathsf{T})[\mathbf{x}], \quad (4)$$

where A is an (affine) matrix of m basis BRDFs in Rusinkiewicz parametrization and their mean, $\mathbf{w} \in \mathbb{R}^m$ is a weight vector and $[\cdot]$ is a 3D table lookup. We follow the PCA method from [NJR15] and [NDM05] to construct A from MERL, which is the same for all BRDFs. m typically is a small number, like 5 we employ here. The weight vector $\mathbf{w}$ changes for each BRDF and is fitted to n BRDF observations $\mathbf{y} \in \mathbb{R}^n$ at n direction pairs $\mathbf{x}$ in closed form:

$$\mathbf{w} = (\hat{\mathsf{A}}^\mathsf{T}\hat{\mathsf{A}} + \eta\mathsf{I})^{-1}\hat{\mathsf{A}}^\mathsf{T}\mathbf{y}, \quad (5)$$

where $\hat{\mathsf{A}} \in \mathbb{R}^{n\times m} = \mathsf{A}[\mathbf{x}]$ is the "reduced" basis, a lookup into the full basis at $\mathbf{x}$. η is a regularization weight, set to $\eta = 40$, as proposed in [NJR15]. The closed-form Eq. 5 replaces the SGD loop in function LEARN, so there are no meta-learnable SGD parameters for the **LINEAR** model. However, optimizing the samples still impacts the quality, even if the solution is closed-form. η could be meta-tuned, but we did not explore this. Our approach is general enough to support non-linearity, but also flexible enough to efficiently support a closed-form special case.

Neural networks An even more sophisticated way of encoding BRDF data is non-linear **NEURAL** networks. In our case, we want

the network the encode the mapping from light- and view-direction to reflectance data. As in [SRRW21], we use a simple two-layer Multi-Layer Perceptron (MLP) with 21 hidden units per layer, ReLU activations, and an exponential activation for the final layer. We use this architecture as it is simple and efficient (only 675 trainable parameters), yet highly expressive. A low number of parameters is desirable, as computing higher order gradients in the meta-learning inner loop is memory-intensive. The learnable parameters are the model's weights, fitted to each new BRDF. The meta-learnable parameters are the initialization and step sizes.

3.5. Implementation

We implement our method in PyTorch [PGM*19] and use the learn2learn framework [AMD*20] as our meta-learning library. The outer-loop optimization for the meta-learned optimization happens via Adam with learning rate 1×10^{-4} (note that the Adam step was replaced by vanilla SGD in Alg. 1, Lines 7 and 12, for ease of exposition). For the optimization of the inner loop, we use MetaSGD [LZCL17], i.e., optimize a per-parameter learning rate together with the respective method's parameters. MetaSGD is initialized with 1×10^{-3}, and we run 20 inner-loop steps of MAML optimization.

To train ξ, we also use Adam in the outer-loop with learning rate 5×10^{-4} and a cosine annealing scheduler [LH17]. As mentioned in Sec. 3.3, we initialize the sampler with a low-discrepancy sequence and the guess-reuse techniques proposed in [NJR15]. More specifically, we start from training $n_0 = 1$ samples with multiple guesses for the best loss in train set. Then, when we train more $n_i = 2n_{i-1}$ samples, we reuse those n_{i-1} learned samples and initialize the other half with a 3D Sobol sequence.

In practice, the two for-loops implementing the meta outer loop in Alg. 1 could operate batched to improve parallelism and to smooth gradients. We experimented with meta-batching these loops, but observed no benefit, and hence set the meta-batchsize to one. So this layer of complexity is omitted from the pseudo-code for clarity.

Moreover, we have manually adjusted the learning rate for the **Random** method of the **NEURAL** model. In [SRRW21], the authors use 5×10^{-4}, whereas we use 1×10^{-3}. Without this change, **Random** in Fig. 1 would be entirely black.

4. Evaluation

Our evaluation uses one methodology (Sec. 4.1) to produce qualitative (Sec. 4.2) and quantitative (Sec. 4.3) results, which are complemented by some final ablation experiments (Sec. 4.4).

4.1. Methodology

Our evaluation is on i) a *dataset*, involving ii) several *metrics* to measure success, iii) *methods* to learn a model and iv) *models* describing BRDFs. We will now detail all of these four aspects.

Dataset We use the popular MERL [Mat03] dataset for our experiments. MERL consists of 100 measured BRDFs, where each BRDF is composed of $90\times90\times180 = 1{,}458{,}000$ angular configurations $(\theta_h,\theta_d,\phi_d)$ in Rusinkiewicz parametrization and one RGB reflectance per triplet. The measured BRDFs range from diffuse to highly specular, and we organize our data in the classic random $80\%-20\%$ train-test split. Moreover, we also test our approach on the additional eight BRDFs provided by [NJR15] and the BRDF data from the RGL material database [DJ18].

Metrics We employ four different metrics. The first is the mean absolute logarithmic error of the cosine weighted BRDF values, which we use as our optimization loss (LOSS in Alg. 1) and hence refer to as the metric "Loss". The other three are image-based DSSIM, L2, and PSNR, for which we render the BRDF on a sphere under environment illumination. For Loss, DSSIM and L2, less is better, whereas for PSNR, higher values are better. All values are reported on unseen test BRDFs, i.e., neither method has had access to any of the evaluation data during training.

Methods We compare three training paradigms: **Random** denotes conventional NN training (as in [SRRW21]), following Sec. 3.1. **Meta** follows [FR22] (for details, cf. Sec. 3.2), and **Ours** is described in Sec. 3.3. As we use a closed form solution for the **LINEAR** model, there is no counterpart of **Meta**. We hence employ the condition number optimization proposed in [NJR15] as the baseline to compare against, which is referred to as **NJR15**. All methods are compared under *equal time*, i.e., using the same number of gradient steps, unless said otherwise. As for **Random** and **Meta**, we report average results on five independent experiments with different random seeds.

Models We study applications to all four different models, **PHONG**, **COOK-TORRANCE**, **LINEAR** (using five basis functions) and **NEURAL** as explained in Sec. 3.4. Note that **LINEAR** is not identical to the specific combination of method and model in [NJR15], which we study separately.

4.2. Qualitative results

The key qualitative results are shown in the right part of Fig. 1, where different methods are applied to learn **NEURAL** for different BRDFs for an equal sample count of $n = 64$ at equal time. At this point, a common learner, **Random**, has not made much progress from the init, which, under the uniform initialization proposed in the original publication [SRRW21], results in the black BRDF depicted in the first row. The second row shows **Meta**, that has learned a more informative init (the pink sphere on the left), and manages to converge to different materials with a low number of samples (compared to [SRRW21], [HGC*20] or [FR22]). However, the appearance is a bit "stereotype", i.e., the differences between BRDFs mainly happen via color changes, but the specularity does not match, and the nuances in glossiness are not picked up correctly either. Looking at the fourth column, **Ours**, we see that the very different appearance characteristics (e.g., highlights, their shape, color, gloss, etc.) from the reference are faithfully reproduced and that **Ours** matches the reference in the last row most closely.

Fig. 3 shows a similar comparison for which we contrast all methods with all models for a diffuse BRDF at a very low sample count of $n = 8$ samples only (upper part). The effects we saw previously at $n = 64$ samples become even more pronounced: for models **PHONG**, **COOK-TORRANCE** and **LINEAR**, we see that **Random** completely fails to faithfully recover the BRDF, and even advanced learning methods like **Meta** struggle with this highly constrained setting

(top and bottom row). The lower part in Fig. 3 shows the same configuration with a specular BRDF, for which we utilize $n = 32$ samples, as specular samples are harder to optimize. Again, **Ours** is closest to the reference. Overall, across all models, **Ours** performs best, which shows that our meta-sampling truely gathers valuable information that helps to accelerate the fitting process over drawing random samples.

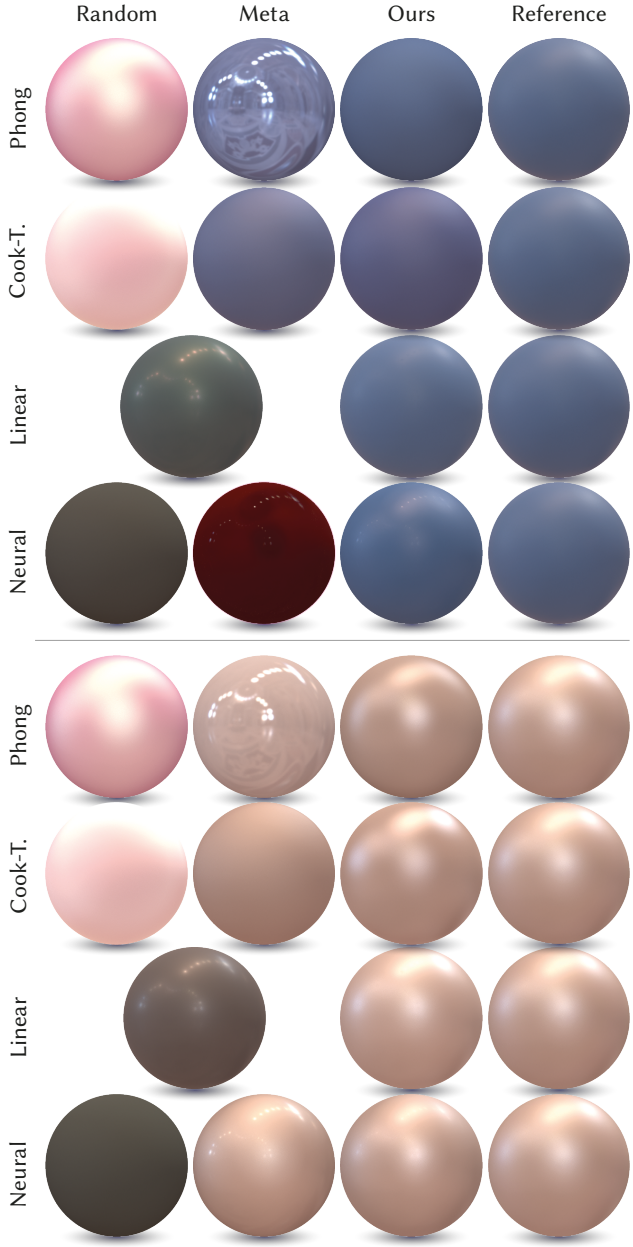

Figure 3: *Result for all methods **(horizontal)** and models **(vertical)** at equal sample count for the test-set BRDFs* `blue-rubber` *(top) and* `silver-paint` *(bottom) at $n = 8$ and $n = 32$, respectively.*

4.3. Quantitative results

Same-sample count Our main quantitative result is seen in Tbl. 1, where we study a same-sample-count ($n = 8$) setting for all models (four major column blocks) and all methods (rows) under all metrics (four-blocks of columns). We see, that for all models according to all metrics, **Ours** is able to improve upon **Meta**, which again is an improvement over **Random**. The improvement is ranging between a factor of two and four. Most importantly, our method is able to improve the quality of a neural BRDF model by co-optimizing the sample pattern, the main contribution of this work. As Tbl. 1 only displays the average outcome across all test BRDFs, we further display each method's performance on all individual BRDFs in the test set at $n = 8$ in Fig. 5. To this end, we sort the resulting BRDFs according to the loss values achieved by **Ours**, i.e., each horizontal point in Fig. 5 is a BRDF, and contrast them against the other methods. We see that we do not only achieve better mean performance, as reported in Tbl. 1, but outperform the other methods for every BRDF.

Same-quality In Fig. 4 we visualize the influence of sample count on performance, for all models on all methods, according to two representative metrics, Loss in BRDF space and DSSIM in image space. In each plot, the horizontal axis is the sample count in log scale between 1 and 512. The vertical axis, also in log scale, is scaled identically per metric, so as to enable comparisons between the individual models. We show the upper quality bound for each model for an unlimited number of samples and time as a red dot in the lower right. This is the target for all methods economizing on sample count to chase. Always, less is better. The red lines show **Random** and confirm our qualitative findings from Sec. 4.2, as it does not manage to do much at such low sample counts, even with many steps. The yellow lines show **Meta**, which has seen many inner-loop trainings and hence can learn to cope with fewer samples, but has no way to change *what* is sampled. The blue line, denoting **Ours**, consistently performs better across the entire sample range, documenting that the improvement claimed in the previous table and figures generalizes to all sample counts. This view is made explicit in Tbl. 4, which shows how many more samples are required to achieve the quality of **Ours** for all models and all metrics if the sample count is fixed to $n = 8$. It by now is clear that a common learner will take much longer, but we also see a good improvement of more than one order of magnitude in sample effort reduction for **Meta**. The improvement for the simple **PHONG** model seems to be larger than for more complex models.

For the **LINEAR** model, we also computed the performance of **NJR15**, who equally assume a linear BRDF model, on our test data (green line). Essentially, this compares using their method of selecting samples to our method of selecting samples for a comparable model (PCA). In image space, we see that for larger sample counts ($n \geq 16$), their way of selecting samples is slightly better than **Ours**, whereas for sample counts below 16, our method is superior. Interestingly, this finding does not transfer to BRDF space, where we consistently outperform **NJR15** for all n. We hence assume that their optimization method (minimizing the condition number) is more closely related to human perception than our training loss, and hence achieves a lower DSSIM. Also, **NJR15** is not yet as good as the full **NEURAL**, to which our method extends. This implies that

Table 1: *Mean test-set performance of all methods on all models at $n = 8$ samples according to different metrics.*

Model →	PHONG				COOK-TORRANCE				LINEAR				NEURAL			
Method ↓	↓ Loss	↓ DSSIM	↓ L2	↑ PSNR	↓ Loss	↓ DSSIM	↓ L2	↑ PSNR	↓ Loss	↓ DSSIM	↓ L2	↑ PSNR	↓ Loss	↓ DSSIM	↓ L2	↑ PSNR
`Random`	0.085	0.103	0.093	12.54	0.098	0.123	0.124	10.75	0.028	0.029	0.007	23.49	0.065	0.105	0.067	13.38
`Meta`	0.084	0.068	0.026	18.82	0.044	0.039	0.007	24.21					0.059	0.125	0.052	15.23
`NJR15`	—	—	—	—	—	—	—	—	0.016	0.012	0.002	28.45	—	—	—	—
`Ours`	**0.044**	**0.034**	**0.006**	**24.51**	**0.039**	**0.029**	**0.003**	**27.09**	**0.011**	**0.009**	**0.001**	**33.70**	**0.035**	**0.033**	**0.005**	**27.50**

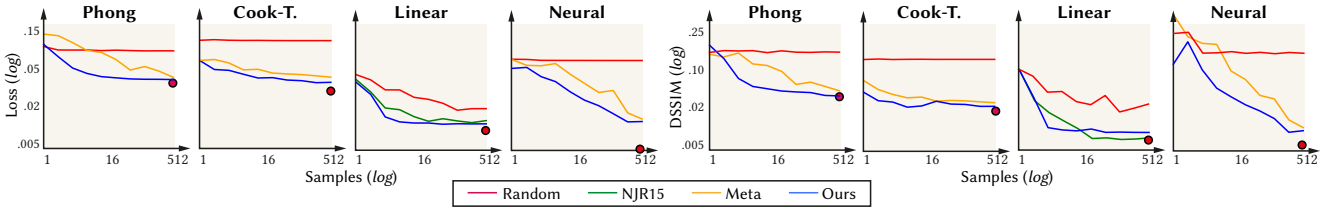

Figure 4: *Performance (**vertical**, log scale) of different learning methods (**colors**) for different models according to different metrics (Log. MAE in BRDF space and image-based DSSIM in every pair, lower is better) depending on the sample count (**horizontal**, log scale). The red dot indicates the theoretical optimum, when giving the model five orders of magnitude more samples, i.e., compute and acquisition time.*

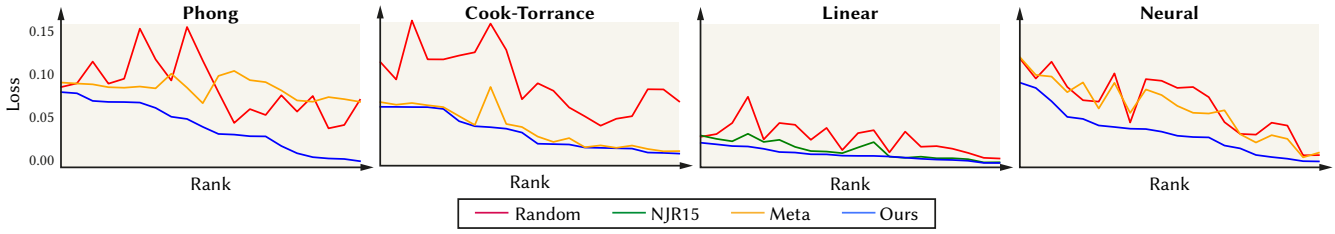

Figure 5: *Test set loss (**vertical**) per BRDF (**horizontal**), sorted based on the results of `Ours` in decreasing order.*

the benefit of our method depends on the target model: When the aim is to get most of the (many) BRDF samples regardless of model complexity and usefulness (PCA is not compact, not fast to evaluate, etc), `NJR15` can be superior. If a compact and efficient model, e.g., NEURAL, is to be used with only few samples, `Ours` is to be preferred. We have repeated similar experiments with 240 instead of five basis functions, leading to similar results, but with the additional disadvantages in storage, train and test compute requirements.

Additional BRDF data We also compared to `NJR15` on their dataset of 8 additional BRDFs in Tbl. 2 and Fig. 6. At similar sample counts, our method performs better according to all metrics. We see that our method, albeit trained on MERL only, can find sample patterns that generalize reliably to BRDFs from a dataset unseen during training, even if these were acquired with a different acquisition setup.

Table 2: *Comparison of `Ours` and [NJR15] on their dataset at $n = 2$.*

	↓ Loss	↓ DSSIM	↓ L2	↑ PSNR
`NJR15`	0.023	0.128	0.003	26.28
`Ours`	**0.018**	**0.009**	**0.002**	**30.02**

We also compare to the isotropic part of RGL material database [DJ18] and show the results in Tbl. 3. Although the overall loss slightly increases due to data shift, our meta-learned samples over the MERL corpus generalize to the new dataset and lead to better reconstructions.

Table 3: *Evaluation of `Ours` on the RGL database [DJ18] at $n = 8$.*

		↓ Loss	↓ DSSIM	↓ L2	↑ PSNR
PHONG	`Meta`	0.095	0.070	0.025	17.980
	`Ours`	**0.044**	**0.040**	**0.009**	**22.686**
COOK-T.	`Meta`	0.068	0.059	0.026	17.824
	`Ours`	**0.067**	**0.051**	**0.009**	**21.563**
LINEAR	`NJR15`	0.050	0.068	0.025	21.897
	`Ours`	**0.030**	**0.041**	**0.016**	**23.849**
NEURAL	`Meta`	0.074	0.168	0.110	12.228
	`Ours`	**0.044**	**0.062**	**0.011**	**21.889**

Sample count Here we explore the performance of our sampler ξ with only a limited budget of samples. We visualize this for all models in Fig. 8, where each column depicts the outcome of a

Table 4: *Relative number of samples required by other methods to achieve our quality at $n = 8$ samples. Here, less is better for all metrics.*

Model →	PHONG				COOK-TORRANCE				LINEAR				NEURAL			
Method ↓	Loss	DSSIM	L2	PSNR	Loss	DSSIM	L2	PSNR	Loss	DSSIM	L2	PSNR	Loss	DSSIM	L2	PSNR
Random	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$	$10^5\times$
Meta	48×	37.5×	24×	8×	8×	64×	64×	64×					4×	6×	8×	6.5×
NJR15	—	—	—	—	—	—	—	—	8×	3×	3×	4×	—	—	—	—
Ours	**1**×	**1**×	**1**×	**1**×	**1**×	**1**×	**1**×	**1**×	**1**×	**1**×	**1**×	**1**×	**1**×	**1**×	**1**×	**1**×

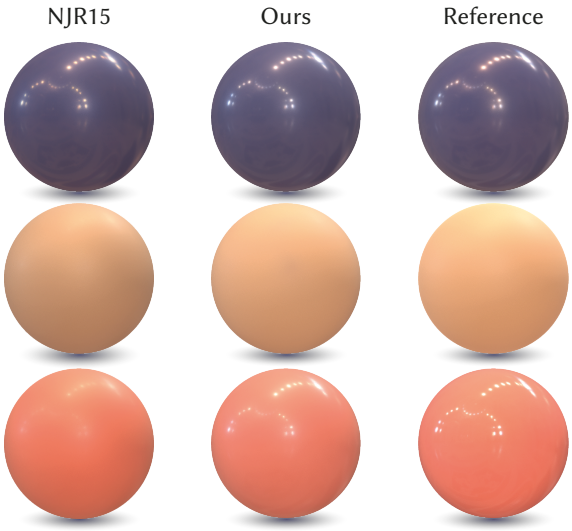

Figure 6: *Results on data from [NJR15] at $n = 2$ samples.*

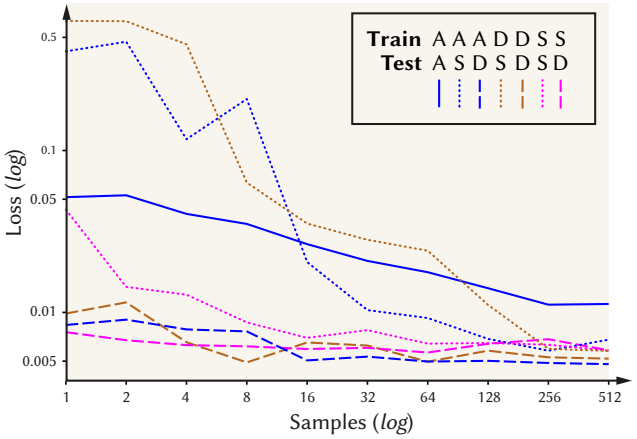

Figure 7: *Error (vertical, less is better) at different numbers of samples (horizontal) for different variants of train and test subsets (colors, line styles). While the training subset corresponds to colors, the test subset maps to line styles (see legend). For a discussion, please see "Subsets" in Sec. 4.4.*

full training run with the number of samples annotated below. In general, all methods increase their final quality with more samples. We specifically picked a specular BRDF, as these are harder to optimize for (for diffuse, all samples could be in a similar spot). At roughly $n = 8$ samples, we can see first highlights developing. Moreover, once a certain quality is reached, e.g., , $n \geq 64$, the increase in performance with more samples starts diminishing. This is confirmed by Fig. 4, especially for **LINEAR** and **PHONG**, and further validates our approach and theory, as in expectation, i.e., $n \to \infty$, the optimization of samples does not matter.

Convergence We further investigate whether meta-sampling will converge to a solution no worse than pure random sampling. Due to memory constraints, we cannot directly train **Ours** with the same number of samples as **Random**, like 10,000, as that would require back-propagation to 10,000 nested samplings and network executions. Fortunately, a method much simpler is already better than **Random**: if we first run **Meta** or **Ours** for 20 steps with 32 learned samples, followed by 10,000 steps with 512 random samples, **Ours** already performs better than 20+10,000 steps with 512 purely random samples (0.0117 Loss for **Random**, 0.0067 for **Meta** and 0.0065 for **Ours**). This confirms that **Ours** does not hamper convergence, but instead, improves it.

Moreover, when repeating the setting of Tbl. 1 with randomized Quasi-Monte-Carlo initializations (five independent meta-sampling trainings), the standard-deviation for **Ours** is less than 5% of the total error reported in Tbl. 1. We thus conclude that our method consistently converges to similarly good samples. Moreover, when iterating the sampler- and model-training once more after convergence, the Loss of **Ours** improves by 0.9/2.86% over the values reported in Tbl. 1 for **PHONG** and **NEURAL**, respectively.

Actual patterns The result of meta-sampling for all three methods can be seen in Fig. 9 for 8 samples, and in Fig. 1 for 64 samples and the neural model. We found it difficult to assign interpretation to those patterns, other than that they seem to group samples and seem not to have the tendency to fill space evenly.

We verify our learned pattern's robustness by randomly changing points and evaluating the resulting variance. We report the increase in loss with respect to Tbl. 1 in Tbl. 5 for r random out of 8 learned samples, averaged over five independent runs. For all configurations, the loss increases, especially notable for the **LINEAR** model, indicating the specific sample positions are relevant.

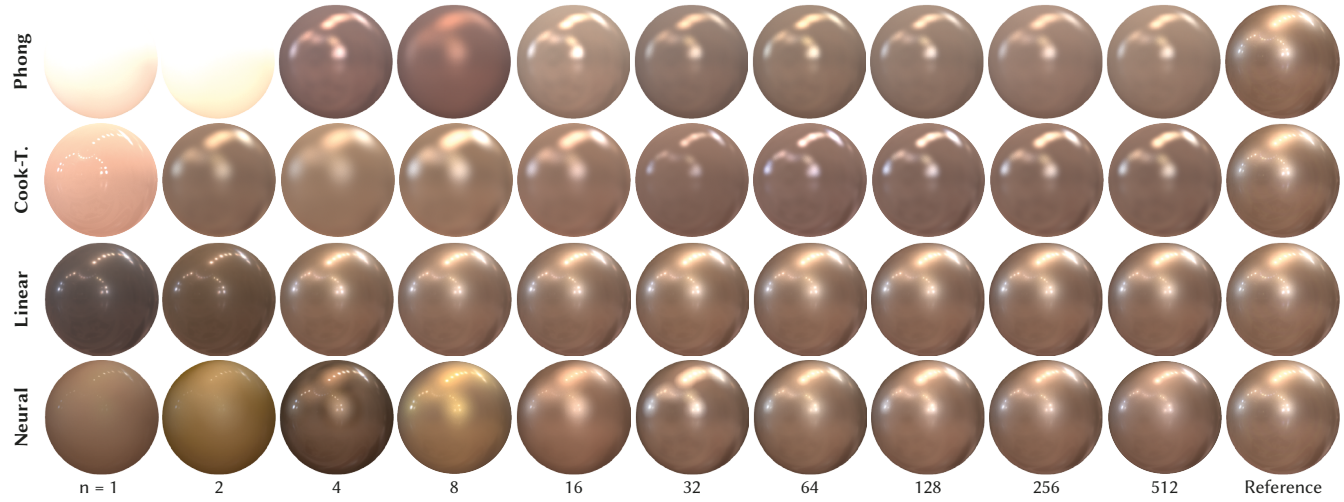

Figure 8: *We show the results of using our proposed meta-sampling for the training of each model (vertical) on the BRDF* `two-layer-gold` *for an increasing number of samples (horizontal). Note that each result is an individual training, not a progression of one training along a row.*

Table 5: *Loss-increase for r random out of eight learned samples.*

	PHONG	COOK-T.	LINEAR	NEURAL
$r = 2$	8.3%	6.9%	13.9%	7.3%
$r = 4$	12.6%	10.5%	21.4%	14.2%

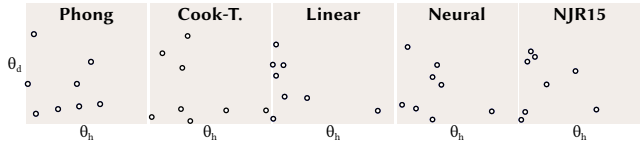

Figure 9: *Projections of our meta-sampling patterns at $n = 8$ for all models. For reference, we also show the pattern from [NJR15].*

4.4. Ablations

Here we confirm the relation of our approach to several variants.

Mean-BRDF importance sampling A straightforward approach to reduce the variance of the stochastic gradient is to importance sample for the average BRDF. To test this, we log-averaged all BRDFs in MERL and then created an inverse Cumulative Distribution Function (CDF) to use for importance sampling, again employing low-discrepancy samples. Such a model performs on average more than 4 times worse across all sample counts, in particular for many samples, where it can be over 10 times worse.

Subsets We now explore how our meta-sampling behaves on selected subsets of the dataset that share specific semantics of MERL. To this end, we meta-train the model NEURAL on the full MERL training set as before, but now only use a specific subset for the training of the sampler ξ. We use two subsets: one with diffuse ($\mathcal{D}$) and one with specular ($\mathcal{S}$) materials. Moreover, let $\mathcal{A}$ denote the set of all BRDFs. In the following, we will write $X \times Y$ to denote training of the sampler on subset X and evaluation of its performance on subset Y. In this notation, $\mathcal{A} \times \mathcal{A}$ is what we already considered in the previous sections: train the sampler on the training set, evaluate its performance on the test set.

We show the results for different train- and test permutations in Fig. 7. $\mathcal{A} \times \mathcal{S}$ performs worse and $\mathcal{A} \times \mathcal{D}$ performs better than $\mathcal{A} \times \mathcal{A}$, indicating specular is harder than diffuse when trained on both. Importantly, $\mathcal{S} \times \mathcal{S}$ and $\mathcal{D} \times \mathcal{D}$, so methods that were tested on what they were trained on, perform better than $\mathcal{D} \times \mathcal{S}$ and $\mathcal{S} \times \mathcal{D}$, methods that test on something they were not trained for. In general, that is not an impressive feature for a learned method, but at the same time it proves that the meta-optimized sampling pattern adapts to the characteristics of the dataset, and does not just create some generic useful sample pattern, akin to some perturbed low-discrepancy sequence.

5. Discussion

Sample model Normalizing Flows [RM15] look like a well-suited alternative parametrization for our sample model ξ: they generate distributions, produce probability density for samples in the inverse direction and could provide an infinite stream of samples instead of a finite set. In practice, we have found these properties not relevant, or not applicable to our case, as the generative nature and additional complexity adds further variance to a process that has already two meta-levels and stacks of optimizations. For tasks other than BRDF, this might become relevant in future work.

Bias Moreover, getting the probability of a sample is important in tasks where we want to retain unbiased estimates, such as in Monte Carlo rendering. Note that while we sample unevenly, we do not attempt to divide by the probability density to produce unbiased estimates of gradients, as it is not clear whether a biased gradient estimate can ultimately not be better than an unbiased one [DPD22]. What matters more is that the outer meta-optimizer sees the effect of those gradients and can factor it into the optimization by changing the initialization or step sizes.

Real-world acquisition cost We assume a naïve cost model of a gonioreflectometer that takes samples in isolation and for which the order of samples does not matter. An actual device will deviate from this model for several reasons. The first is, that multiple samples (slices) can be taken at once by capturing entire images. Our model assumes samples to be taken in isolation. The second main simplification is, that the cost of taking a sample while already moving along a trajectory is much lower than changing the direction, which requires acceleration. Or model assumes that every change of sample direction has the same cost. For a discussion of BRDF cost properties, please see [GGG*16].

Progressivity The way we meta-optimize implies that the test-time optimization is good once converged after a fixed number of steps. A progressive or interruptible version could be optimized so that it delivers optimal result throughout the entire optimization, by adding all intermediate inner loss values to the meta loss.

Fixed sample count We currently use a fixed-size vector of sample directions, while in practice a method with varying sample counts would be more flexible. In particular, a method with no limit on the sample count, which eventually converges the same way as random sampling would do. This could be achieved by meta-learning a generative model of samples, e.g., using Normalizing Flow.

Adaptivity The sampling pattern is not on-line [VKŠ*14] or adaptive [DPD22], but the same for each BRDF. A pattern that adapts to some other condition, or maybe to the outcome of previous samples, would be a relevant avenue of future work.

Time and space overhead Meta-learning with MAML [FAL17] requires the computation of the Hessian-vector product, making it very memory intensive. We found that this does not apply so much to our scenario, as the most memory intensive operation is the training of the NBRDF network (2.76 GB VRAM), a property of [SRRW21] and [FR22]. The overhead of meta-sampling is negligible in comparison: for **`Ours`** at $n = 10/20$ samples, we require an additional $40.9/51.2$ MB of VRAM, respectively, as we are merely optimizing scalar variables. There is no overhead at inference time.

6. Conclusion

We have described a method to reduce the number of samples required to fit a non-linear BRDF model, such as a NN. To this end, we jointly optimize over the model parameters and the sampling parameters. Our approach reduce the number of samples required by substantial factors while achieving the same quality.

In future work, we would like to apply the meta-sampling to other domains where samples can be freely taken, such as light field compression [SRF*21], radiance caching [MRNK21] or path guiding [VKŠ*14].

Acknowledgments
We would like to appreciate the anonymous reviewers for their constructive comments and suggestions. We also thank the valuable feedback from Vlastimil Havran, Stavros Diolatzis, and Gilles Rainer.

References

[AMD*20] ARNOLD S. M. R., MAHAJAN P., DATTA D., BUNNER I., ZARKIAS K. S.: learn2learn: A library for Meta-Learning research.

[BKW21] BERGMAN A. W., KELLNHOFER P., WETZSTEIN G.: Fast training of neural lumigraph representations using meta learning. In *NeurIPS* (2021).

[BP20] BIERON J., PEERS P.: An adaptive BRDF fitting metric. In *Comp. Graph. Forum* (2020), vol. 39, pp. 59–74.

[BS12] BURLEY B., STUDIOS W. D. A.: Physically-based shading at Disney. In *ACM SIGGRAPH* (2012), vol. 2012, pp. 1–7.

[BTPG16] BOUCHARD G., TROUILLON T., PEREZ J., GAIDON A.: Online learning to sample, 2016. arXiv:1506.09016.

[CT82] COOK R. L., TORRANCE K. E.: A reflectance model for computer graphics. *ACM Tran Graph 1*, 1 (1982), 7–24.

[DAD*18] DESCHAINTRE V., AITTALA M., DURAND F., DRETTAKIS G., BOUSSEAU A.: Single-image SVBRDF capture with a rendering-aware deep network. *ACM Trans. Graph. (Proc. SIGGRAPH) 37*, 4 (2018).

[dBWK18] DEN BROK D., WEINMANN M., KLEIN R.: Rapid material capture through sparse and multiplexed measurements. *Computers & Graphics 73* (2018), 26–36.

[DDB20] DESCHAINTRE V., DRETTAKIS G., BOUSSEAU A.: Guided fine-tuning for large-scale material transfer. In *Comp Graph Forum* (2020), vol. 39.

[DJ18] DUPUY J., JAKOB W.: An adaptive parameterization for efficient material acquisition and rendering. *ACM Transactions on graphics (TOG) 37*, 6 (2018), 1–14.

[DPD22] DIOLATZIS S., PHILIP J., DRETTAKIS G.: Active exploration for neural global illumination of variable scenes. *ACM Trans Graph 41*, 5 (2022).

[Erb80] ERB W.: Computer-controlled gonioreflectometer for the measurement of spectral reflection characteristics. *Applied optics 19*, 22 (1980), 3789–3794.

[FAL17] FINN C., ABBEEL P., LEVINE S.: Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML* (2017).

[FBLS07] FUCHS M., BLANZ V., LENSCH H. P., SEIDEL H.-P.: Adaptive sampling of reflectance fields. *ACM Trans Graph 26*, 2 (2007).

[FR22] FISCHER M., RITSCHEL T.: Metappearance: Meta-learning for visual appearance reproduction. *ACM Trans Graph (Proc SIGGRAPH Asia)* (2022).

[FWH*21] FAN J., WANG B., HAŠAN M., YANG J., YAN L.-Q.: Neural BRDFs: Representation and operations. *arXiv:2111.03797* (2021).

[GGG*16] GUARNERA D., GUARNERA G. C., GHOSH A., DENK C., GLENCROSS M.: Brdf representation and acquisition.

[GLD*19] GAO D., LI X., DONG Y., PEERS P., XU K., TONG X.: Deep inverse rendering for high-resolution SVBRDF estimation from an arbitrary number of images. *ACM Trans. Graph. 38*, 4 (2019).

[GRR*17] GEORGOULIS S., REMATAS K., RITSCHEL T., GAVVES E., FRITZ M., VAN GOOL L., TUYTELAARS T.: Reflectance and natural illumination from single-material specular objects using deep learning. *IEEE PAMI 40*, 8 (2017).

[GSH*20] GUO Y., SMITH C., HAŠAN M., SUNKAVALLI K., ZHAO S.: MaterialGAN: reflectance capture using a generative svBRDF model. *arXiv:2010.00114* (2020).

[HDMR21] HENZLER P., DESCHAINTRE V., MITRA N. J., RITSCHEL T.: Generative modelling of BRDF textures from flash images. *ACM Trans Graph (Proc. SIGGRAPH Asia) 40*, 6 (2021).

[HGC*20] HU B., GUO J., CHEN Y., LI M., GUO Y.: DeepBRDF: A deep representation for manipulating measured BRDF. *Comp Graph Forum 39*, 2 (2020).

[KCW*18] KANG K., CHEN Z., WANG J., ZHOU K., WU H.: Efficient reflectance capture using an autoencoder. *ACM Trans. Graph. 37*, 4 (2018).

[KF19] KATHAROPOULOS A., FLEURET F.: Not all samples are created equal: Deep learning with importance sampling. *arXiv:1803.00942* (2019).

[KXH*19] KANG K., XIE C., HE C., YI M., GU M., CHEN Z., ZHOU K., WU H.: Learning efficient illumination multiplexing for joint capture of reflectance and shape. *ACM Trans. Graph. 38*, 6 (2019), 165–1.

[LCY*17] LIU G., CEYLAN D., YUMER E., YANG J., LIEN J.-M.: Material editing using a physically based rendering network. In *CVPR* (2017).

[LFTG97] LAFORTUNE E. P., FOO S.-C., TORRANCE K. E., GREENBERG D. P.: Non-linear approximation of reflectance functions. In *SIGGRAPH* (1997), pp. 117–126.

[LFTW06] LI H., FOO S.-C., TORRANCE K. E., WESTIN S. H.: Automated three-axis gonioreflectometer for computer graphics applications. *Optical Engineering 45*, 4 (2006).

[LH17] LOSHCHILOV I., HUTTER F.: SGDR: Stochastic Gradient Descent with Warm Restarts, 2017.

[LKG*03] LENSCH H. P., KAUTZ J., GOESELE M., HEIDRICH W., SEIDEL H.-P.: Image-based reconstruction of spatial appearance and geometric detail. *ACM Trans. Graph. 22*, 2 (2003).

[LRR04] LAWRENCE J., RUSINKIEWICZ S., RAMAMOORTHI R.: Efficient brdf importance sampling using a factored representation. *ACM Trans. Graph. (Proc. SIGGRAPH) 23*, 3 (2004), 496–505.

[LW94] LAFORTUNE E. P., WILLEMS Y. D.: Using the modified Phong reflectance model for physically based rendering.

[LZCL17] LI Z., ZHOU F., CHEN F., LI H.: Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv:1707.09835* (2017).

[Mat03] MATUSIK W.: *A data-driven reflectance model*. PhD thesis, Massachusetts Institute of Technology, 2003.

[McA02] MCALLISTER D. K.: *A generalized surface appearance representation for computer graphics*. PhD thesis, 2002.

[MLTFR19] MAXIMOV M., LEAL-TAIXÉ L., FRITZ M., RITSCHEL T.: Deep appearance maps. In *ICCV* (2019).

[MRNK21] MÜLLER T., ROUSSELLE F., NOVÁK J., KELLER A.: Real-time neural radiance caching for path tracing. *ACM Trans. Graph. 40*, 4 (Aug. 2021), 36:1–36:16.

[MWL*99] MARSCHNER S. R., WESTIN S. H., LAFORTUNE E. P., TORRANCE K. E., GREENBERG D. P.: Image-based BRDF measurement including human skin. In *EGWR* (1999), pp. 131–144.

[NDM05] NGAN A., DURAND F., MATUSIK W.: Experimental analysis of BRDF models. *Rendering Techniques* (2005).

[NJR15] NIELSEN J. B., JENSEN H. W., RAMAMOORTHI R.: On optimal, minimal BRDF sampling for reflectance acquisition. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 34*, 6 (2015).

[NLGK18] NAM G., LEE J. H., GUTIERREZ D., KIM M. H.: Practical SVBRDF acquisition of 3D objects with unstructured flash photography. *ACM Trans. Graph. 37*, 6 (2018).

[NRH*92] NICODEMUS F. E., RICHMOND J. C., HSIA J. J., GINSBERG I., LIMPERIS T.: Geometrical considerations and nomenclature for reflectance. *NBS monograph 160* (1992), 4.

[NWS14] NEEDELL D., WARD R., SREBRO N.: Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm. In *NeurIPS* (2014), vol. 27.

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., DESMAISON A., KOPF A., YANG E., DEVITO Z., RAISON M., TEJANI A., CHILAMKURTHY S., STEINER B., FANG L., BAI J., CHINTALA S.: Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019.

[Pho75] PHONG B. T.: Illumination for computer generated pictures. *Communications of the ACM 18*, 6 (1975), 311–317.

[PHS20] PARK J. J., HOLYNSKI A., SEITZ S. M.: Seeing the world in a bag of chips. In *CVPR* (2020).

[RGJW20] RAINER G., GHOSH A., JAKOB W., WEYRICH T.: Unified neural encoding of btfs. *Comp Graph Forum 39*, 2 (2020).

[RJGW19] RAINER G., JAKOB W., GHOSH A., WEYRICH T.: Neural BTF compression and interpolation. *Comp Graph Forum 38*, 2 (2019).

[RM15] REZENDE D., MOHAMED S.: Variational inference with normalizing flows. In *ICML* (2015).

[RRF*16] REMATAS K., RITSCHEL T., FRITZ M., GAVVES E., TUYTELAARS T.: Deep reflectance maps. In *CVPR* (2016), pp. 4508–4516.

[Rus98] RUSINKIEWICZ S. M.: A new change of variables for efficient BRDF representation. In *EGWR* (1998).

[Sch94] SCHLICK C.: An inexpensive brdf model for physically-based rendering. In *Computer graphics forum* (1994), vol. 13, Wiley Online Library, pp. 233–246.

[SCT*20] SITZMANN V., CHAN E. R., TUCKER R., SNAVELY N., WETZSTEIN G.: MetaSDF: Meta-learning signed distance functions. *arXiv:2006.09662* (2020).

[SRF*21] SITZMANN V., REZCHIKOV S., FREEMAN W. T., TENENBAUM J. B., DURAND F.: Light field networks: Neural scene representations with single-evaluation rendering. In *NeurIPS* (2021).

[SRJ17] STICH S. U., RAJ A., JAGGI M.: Safe adaptive importance sampling. In *NIPS* (2017), vol. 30.

[SRRW21] SZTRAJMAN A., RAINER G., RITSCHEL T., WEYRICH T.: Neural BRDF representation and importance sampling. *Comp Graph Forum 40*, 6 (2021).

[SS18] SENER O., SAVARESE S.: Active learning for convolutional neural networks: a core-set aproach, 2018. arXiv:1708.00489.

[TMW*21] TANCIK M., MILDENHALL B., WANG T., SCHMIDT D., SRINIVASAN P. P., BARRON J. T., NG R.: Learned initializations for optimizing coordinate-based neural representations. In *CVPR* (2021).

[VKŠ*14] VORBA J., KARLÍK O., ŠIK M., RITSCHEL T., KŘIVÁNEK J.: On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph. 33*, 4 (2014).

[WMM*21] WANG S., MIHAJLOVIC M., MA Q., GEIGER A., TANG S.: MetaAvatar: Learning animatable clothed human models from few depth images. *arXiv:2106.11944* (2021).

[WSB*98] WHITE D. R., SAUNDERS P., BONSEY S. J., VAN DE VEN J., EDGAR H.: Reflectometer for measuring the bidirectional reflectance of rough surfaces. *Applied optics 37*, 16 (1998), 3450–3454.

[XSHR18] XU Z., SUNKAVALLI K., HADAP S., RAMAMOORTHI R.: Deep image-based relighting from optimal sparse samples. *ACM Trans Graph 37*, 4 (2018), 1–13.

[YXM*16] YU J., XU Z., MANNINO M., JENSEN H. W., RAMAMOORTHI R.: Sparse sampling for image-based SVBRDF acquisition. *Image 1* (2016), 1.

[ZCD*16] ZHOU Z., CHEN G., DONG Y., WIPF D., YU Y., SNYDER J., TONG X.: Sparse-as-possible SVBRDF acquisition. *ACM Trans Graph 35*, 6 (2016), 1–12.

[ZZ15] ZHAO P., ZHANG T.: Stochastic optimization with importance sampling for regularized loss minimization. In *ICML* (2015), pp. 1–9.