MetaLayer: A Meta-learned BSDF Model for Layered Materials

JIE GUO*, State Key Lab for Novel Software Technology, Nanjing University, China
ZERU LI*, State Key Lab for Novel Software Technology, Nanjing University, China
XUEYAN HE, State Key Lab for Novel Software Technology, Nanjing University, China
BEIBEI WANG, Nankai University and Nanjing University of Science and Technology, China
WENBIN LI, State Key Lab for Novel Software Technology, Nanjing University, China
YANWEN GUO†, State Key Lab for Novel Software Technology, Nanjing University, China
LING-QI YAN, University of California, Santa Barbara, United States of America

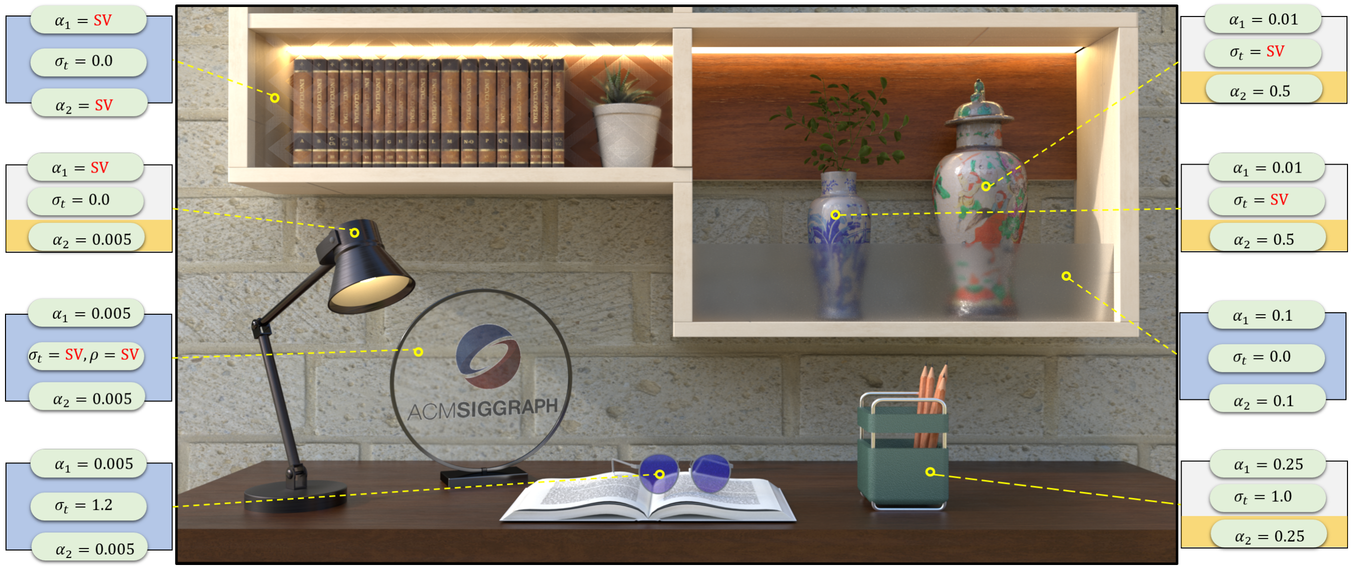


Fig. 1. We propose MetaLayer, a hypernetwork architecture (a special meta-learning strategy) tailored for modeling layered materials covering a wide range of configurations, e.g., reflective/transmissive interfaces, thin/thick media and spatial variations (SV). The proposed model facilitates efficient rendering, convenient material editing, easy integration and fast convergence in Monte Carlo rendering frameworks. Please refer to Table 1 for the meanings of the symbols.

Reproducing the appearance of arbitrary layered materials has long been a critical challenge in computer graphics, with regard to the demanding requirements of both physical accuracy and low computation cost. Recent studies have demonstrated promising results by learning-based representations that implicitly encode the appearance of complex (layered) materials by neural networks. However, existing generally-learned models often struggle between strong representation ability and high runtime performance, and also lack physical parameters for material editing. To address these concerns, we introduce *MetaLayer*, a new methodology leveraging meta-learning for modeling and rendering layered materials. MetaLayer contains two networks: a *BSDFNet* that compactly encodes layered materials into implicit neural representations, and a *MetaNet* that establishes the mapping between the physical parameters of each material and the weights of its corresponding implicit neural representation. A new positional encoding method and a well-designed training strategy are employed to improve the performance and quality of the neural model. As a new learning-based representation, the proposed MetaLayer model provides both fast responses to material editing and high-quality results for a wide range of layered materials, outperforming existing layered BSDF models.

*Joint first authors.
†Corresponding author.

Authors' addresses: Jie Guo, State Key Lab for Novel Software Technology, Nanjing University, China, guojie@nju.edu.cn; Zeru Li, State Key Lab for Novel Software Technology, Nanjing University, China, lzr@smail.nju.edu.cn; Xueyan He, State Key Lab for Novel Software Technology, Nanjing University, China, xueyanhe@smail.nju.edu.cn; Beibei Wang, Nankai University and Nanjing University of Science and Technology, China, beibei.wang@njust.edu.cn; Wenbin Li, State Key Lab for Novel Software Technology, Nanjing University, China, liwenbin@nju.edu.cn; Yanwen Guo, State Key Lab for Novel Software Technology, Nanjing University, China, ywguo@nju.edu.cn; Ling-Qi Yan, lingqi@cs.ucsb.edu, University of California, Santa Barbara, Santa Barbara, United States of America.

0730-0301/2023/12-ART221 $15.00
https://doi.org/10.1145/3618365

CCS Concepts: • **Computing methodologies → Reflectance modeling**; *Neural networks.*

Additional Key Words and Phrases: layered material, meta learning, neural networks, importance sampling

ACM Reference Format:
Jie Guo, Zeru Li, Xueyan He, Beibei Wang, Wenbin Li, Yanwen Guo, and Ling-Qi Yan. 2023. MetaLayer: A Meta-learned BSDF Model for Layered Materials. *ACM Trans. Graph.* 42, 6, Article 221 (December 2023), 15 pages. https://doi.org/10.1145/3618365

1 INTRODUCTION

Real-world materials are mostly layered, containing multiple layers with varying compositions (e.g., coated metals, varnished woods, glazed ceramics, and metallic paints). The complex interactions between the light and layers give rise to visually rich and diversified appearance characteristics that are not exhibited by single-layer materials. Unfortunately, due to the complexity of subsurface light transport, realistically rendering these *layered materials* has long been a daunting and challenging task in computer graphics [Bati et al. 2019; Weidlich and Wilkie 2009, 2011], bottlenecking many rendering engines.

General and accurate layered BSDF (Bidirectional Scattering Distribution Function) models either rely on per-material precomputation [Jakob et al. 2014; Zeltner and Jakob 2018] or resort to stochastic evaluation [Gamboa et al. 2020; Guo et al. 2018; Xia et al. 2020]. Recent precomputation-based models [Jakob et al. 2014; Zeltner and Jakob 2018] succeed in capturing aggregate scattering behaviors for layered materials with various layer components using the adding-doubling strategy [van de Hulst 1980]. However, they require long per-material precomputation time and large storage, making them impractical for spatially-varying cases. Stochastic layered BSDF models, pioneered by Guo et al. [2018], are free from costly precomputation, but suffer from low efficiency and high variance, since Monte Carlo sampling is involved in the estimation of the light scattering inside layers.

With the advent of deep neural networks, learning-based BSDF models [Fan et al. 2022; Hu et al. 2020; Kuznetsov et al. 2021, 2022; Rainer et al. 2019; Tongbuasirilai et al. 2022; Zheng et al. 2022] have emerged as a preferable choice for modeling very complex appearance behaviors compared to conventional models. Deep neural networks are adept at learning robust priors from large-scale reflectance data, allowing us to reach an unprecedented level of realism in visual appearance. However, these learned BSDF models often struggle between strong representation ability (large networks with many parameters) and high runtime performance (small networks with few parameters). Without additional efforts (e.g., designing and training another network [Hu et al. 2020]), they also usually lack physical or perceptual parameters for convenient material editing or effective inverse rendering [Bati et al. 2021]. This is particularly important for layered materials since many graphical applications need the ability to adjust the physical parameters of each layer to reach the desired aggregate appearance. Moreover, these models usually have low generalization ability due to the limited scale of material datasets [Dupuy and Jakob 2018; Filip and Vávra 2014; Matusik et al. 2003].

To tackle the above issues and to make learning-based layered BSDF models more practical, we propose *MetaLayer* which introduces meta-learning [Hospedales et al. 2022; Thrun and Pratt 1998] into layered material modeling and rendering. The key idea behind MetaLayer is learning to learn a neural representation for any layered BSDF. This is a typical hypernetwork design [Ha et al. 2017] which is a special category of meta-learning [Hospedales et al. 2022]. Specifically, MetaLayer involves a *BSDFNet* to compactly encode layered materials into implicit neural representations and a *MetaNet* to establish the mapping between the physical parameters of each material and the weights of its corresponding neural representation. Such a weight generation scheme allows weight sharing across layers of networks [Ha et al. 2017]. To retain high-frequency details and eliminate annoying artifacts, a new positional encoding method named *Rusinkiewicz spherical harmonics encoding* is used in BSDFNet. Once trained jointly via a well-designed training strategy, MetaNet is expected to generate the specific network weights of BSDFNet for any unseen layered material through only one feed-forward propagation, operating in milliseconds. Material editing is enabled by directly changing the input of MetaNet, i.e., the physical parameters of layered materials.

To summarize, the main contributions of this work are:

- a meta-learning framework, i.e., *MetaLayer*, for layered material modeling, providing convenient material editing and better generalization than existing learning-based models,
- a specially-designed training strategy to train *BSDFNet* and *MetaNet* in two phases, enabling stable convergence,
- a new positional encoding method named *Rusinkiewicz spherical harmonics encoding* to retain high-frequency details for directional distributed BSDF data,
- an efficient integration of our MetaLayer within any physically-based renderer, demonstrating high-speed evaluation, little precomputation overhead, and convenient material editing.

2 RELATED WORK

Approximate Layered BSDF Models. Reproducing physically correct appearance for layered materials requires solving a very complex 1D radiative transfer equation [Hanrahan and Krueger 1993; Pharr and Hanrahan 2000]. Generally, interaction phenomena take place not only at the surfaces, but also within any point of the medium. To reduce the computational burden, many practical models resort to certain approximations, in particular, the simplified computation of the light transport within layered materials. For instance, some layered BSDF models lack proper evaluation of multiple scattering inside layers [Gu et al. 2007; Hanrahan and Krueger 1993] considering the high computational overhead of multiple scattering. Others even completely ignore any scattering event within individual layers [Dai et al. 2009; Guo et al. 2017; Weidlich and Wilkie 2007, 2009, 2011]. A prominent layered BSDF model in this category is the statistical framework of Belcour [2018] which approximates the surface reflectance as the summation of multiple lobes derived from directional albedo, incident direction, and roughness. Later, much work has been dedicated to extending this statistical approach to handle anisotropy [Weier and Belcour 2020; Yamaguchi et al. 2019], diffuse interfaces [de Dinechin and Belcour 2022] or arbitrary scattering volumes [Randrianandrasana et al. 2021]. There are also many approximate models targeting specific layered materials such as dusty surfaces [Gu et al. 2007], metallic patinas [Dorsey and Hanrahan 1996], paper [Papas et al. 2014], leaves [Baranoski and Rokne 2001] and human skin [Stam 2001]. Usually, these approximate models make a compromise between physical accuracy and computing time.

Precomputation-based Layered BSDF models. The difficulty in designing general layered BSDF models resides in correctly accounting for multiple scattering within the layered structures. For efficient evaluation, one flourishing way is to precompute the angular distribution of the materials and store it using some compact representations. The first generic layered BSDF model based on precomputation is proposed by Jakob et al. [2014] which relies on an expensive angular/Fourier mode matrix representation. This model can be seen as a significant extension to Stam's discretization approach [Stam 2001] and was later further extended by Zeltner and Jakob [2018] to handle anisotropic interfaces. Ergun et al. [2016] extended Jakob et al.'s model to predict the appearance of car paint from the paint composition. Despite their high accuracy and efficiency at runtime, these models are generally impractical for spatially-varying structures due to long per-material precomputation time and significant storage overhead. They also easily suffer from ringing artifacts, especially at grazing angles. In contrast, our meta-learning-based solution is free from costly per-material precomputation, while still guaranteeing excellent visual effects for any layered material.

Stochastic Layered BSDF models. To achieve accurate evaluation of layered BSDFs with arbitrary compositions, another promising strategy is to use the Monte Carlo simulation which accounts for all light transport paths naturally [Novák et al. 2018]. However, directly applying the Monte Carlo simulation to layered material rendering is prohibitively expensive, since a large number of scattering events may happen at the layer interfaces or inside the internal medium. Guo et al. [2018] proposed a position-free Monte Carlo method that leverages a path-space simplification tailored to the layered material context. The proposed model is accurate, unbiased, and general, but is still plagued with high variance on account of the stochastic nature, leading to a very long convergence time. Several subsequent studies tried to reduce variance by developing more efficient Monte Carlo methods [Gamboa et al. 2020; Xia et al. 2020]. Gamboa et al. [2020] presented an efficient path construction method to sample and evaluate high-throughput, low-variance paths through an arbitrary number of interfaces and media layers. Xia et al. [2020] introduced pair-product sampling and multi-product sampling to better take advantage of the layered structure and reduce variance compared to Guo et al.'s approach. Despite these efforts for variance reduction, a straightforward implementation of these stochastic models is rather inefficient and impractical for the production purposes.

Learning-based Appearance Modeling. Recent years have witnessed great progress in reproducing visual appearance using deep learning [Dong 2019]. A series of neural methods have been developed from different perspectives for appearance modeling. Some existing neural methods aim at recovering physically-plausible material maps from one or a small number of input images using end-to-end trained neural networks [Aittala et al. 2016; Deschaintre et al. 2018, 2019; Guo et al. 2021; Li et al. 2017, 2018]. Others try to exploit a low-dimensional latent space from the material data, so as to intelligently reduce the dimensionality of the data for measured BRDF [Fan et al. 2022; Hu et al. 2020; Zheng et al. 2022], spatially-varying BRDF [Gao et al. 2019; Guo et al. 2020], bidirectional texture function (BTF) [Kuznetsov et al. 2021, 2022; Rainer et al. 2020, 2019], etc. After compressing raw high-dimensional material data using material-specific neural networks, each material is represented as a low-dimensional latent code. However, these latent codes usually lack physical parameters for intuitive material editing or effective inverse rendering [Bati et al. 2021]. Recently, Xu et al. [2022] encoded BRDF, visibility, and lighting using a set of neural basis functions.

Unlike these methods, we encode each layered material within the weights of a trained neural network called BSDFNet. The weights are predicted by a meta-network called MetaNet, given some physically meaningful material parameters such as surface roughness and extinction coefficient of the medium. The weight prediction is a typical hypernetwork design, one of the meta-learning strategies in neural networks [Hospedales et al. 2022; Hu et al. 2019]. This facilitates the compression, rendering, and editing of arbitrary layered materials. Hypernetworks have also been used in modeling appearance maps [Maximov et al. 2019] and 3D faces [Bi et al. 2021].

A similar meta-learning idea is also mentioned in the work of Sztrajman et al. [2021] in which each BRDF is overfitted to the weights of a small neural network and is further compressed into a low dimensional embedding. However, the small scale of the neural network limits its representation ability, and the choice of overfitted weights makes their method only support BRDF interpolation. Different from that work, our neural representation of layered materials has stronger expressiveness and robustness thanks to a weight sharing strategy introduced in BSDFNet. It also significantly expands the material editing ability due to the special design of MetaNet and the training strategy. More recently, a Metappearance model is proposed by Fischer et al. [2022] which also leverages meta-learning for visual appearance reproduction. However, their meta-learning paradigm is significantly different from ours. They chose a model-agnostic meta-learning framework to improve the inference accuracy of trained models, while our goal is to use a meta-network to produce weights of another network, thus supporting convenient material editing and fast evaluation.

3 THE METALAYER MODEL

In this section, we describe our MetaLayer model and its implementation details.

3.1 Motivation and Overview

Recent learning-based BSDF models typically need to train the networks for every new material, and the material is encoded in the

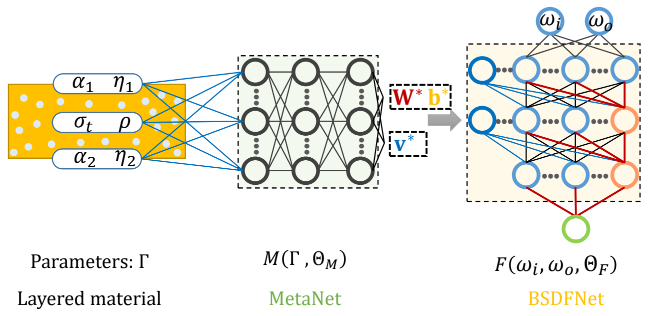


Fig. 2. High-level overview of our MetaLayer model. A set of physical parameters of a layered material are mapped to the network parameters of BSDFNet, through another neural network: MetaNet. Once converged after training, MetaLayer can model the appearance of a wide range of layered materials and BSDFNet can be easily integrated into any renderer as a parametric BSDF model.

network parameters [Sztrajman et al. 2021] or a fixed-length latent vector [Fan et al. 2022; Hu et al. 2020; Zheng et al. 2022] after training for an enormous number of iterations. Generally, each of these models is restricted to representing only materials within its learned latent space, potentially limiting its ability to express previously unseen materials. More critically, these methods are hard for material editing, since no connection is established between the materials' neural representations and their physical parameters. Consequently, only material interpolation is readily supported by most existing neural methods. However, the ability of explicit parameter tuning is important for layered materials since many graphical applications, including both forward rendering [Belcour 2018] and inverse rendering [Bati et al. 2021], need the ability to adjust the physical parameters of each layer to determine the desired aggregate appearance.

We address the above limitations with the proposed MetaLayer model, a meta-learning-based method for modeling arbitrary layered materials. The basic idea is illustrated in Fig. 2. Specifically, we represent the BSDF of any layered material as a parameter function $F(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \Theta_F)$ with the trainable parameters Θ_F. Currently, F is realized as a fully-connected multi-layer perceptron (MLP) with positional encoding. This network, named BSDFNet, is optimized to map from a pair of input directions $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$ to the RGB reflectance value at that pair of directions. Theoretically, such a coordinate-based neural representation is continuous and naturally has an infinite resolution.

Meta-learning is normally used to train a meta network to produce the weights of another network. In our context, we construct a meta neural network, named MetaNet, to generate the weights of BSDFNet (taken as the base network). MetaNet, also realized as an MLP M with its trainable weights Θ_M, takes a set of physical parameters Γ of a layered material and produces the partial weights (Θ_F^*) of the corresponding BSDFNet. These partial weights (Θ_F^*), together with other weights of BSDFNet which are frozen after training and are shared across all layered materials, are expected to faithfully reproduce the appearance of that layered material. This *weight sharing strategy* is adopted to reduce the scale of the neural BSDF model, while still preserving strong expressiveness and high generalization ability.

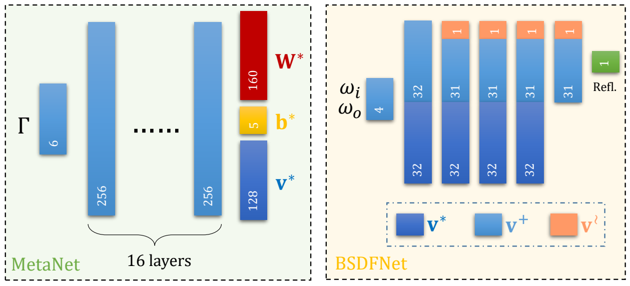


Fig. 3. Detailed architectures of MetaNet (left) and BSDFNet (right). Both are realized by MLPs with the number of neurons in each layer displayed on the colored bar.

Overall, our meta-learned layered BSDF model is formally given by the following equations:

$$M(\Gamma; \Theta_M) = \Theta_F^* \tag{1}$$

$$F(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \Theta_F) = f_s(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \Gamma) \tag{2}$$

where Θ_F^* denotes a subset of Θ_F which is generated by the MetaNet M and $f_s(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \Gamma)$ returns the actual BSDF value of the corresponding layered material with physical parameters Γ evaluated at $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$. Both BSDFNet and MetaNet are trained together on $N = N_\Gamma \times N_{\omega_i} \times N_{\omega_o}$ samples. In particular, the weights Θ_F are optimized such that F matches f_s as closely as possible for all coordinates $\boldsymbol{\omega}_i, \boldsymbol{\omega}_o$, i.e.,

$$\hat{\Theta}_F = \arg\min_{\Theta_F} \sum_{l=1}^{N_\Gamma} \sum_{m=1}^{N_{\omega_i}} \sum_{n=1}^{N_{\omega_o}} \mathcal{L}\left(F(\boldsymbol{\omega}_i^{(m)}, \boldsymbol{\omega}_o^{(n)}; \Theta_F), f_s(\boldsymbol{\omega}_i^{(m)}, \boldsymbol{\omega}_o^{(n)}, \Gamma^{(l)})\right) \tag{3}$$

where $\mathcal{L}$ is a task-specific loss measuring the error between f_s and F. Once trained, we are able to generate the specified network parameters of BSDFNet through one feed-forward propagation in the MetaNet, given a set of physical parameters of a layered material. Then, this specific BSDFNet can be integrated into any renderer, working as a standard parametric BRDF model.

The specific design of this BSDF model leads to the following benefits:

- It is efficient and possesses low variance since Monte Carlo sampling is not required in evaluating this model.
- It supports convenient material editing with physically meaningful parameters.
- It does not rely on any costly precomputation and hence is friendly to spatially-varying cases.
- It is a general model which supports a wide range of layered materials.

3.2 BSDFNet

In our MetaLayer model, every layered material is represented as a deep neural network F which takes as input a pair of normalized directions $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$. By encoding data into this coordinate-based network, the appearance of a layered material will be essentially represented using the weights (or parameters) of the network. One key design choice of this network is that some weights (Θ_F^*) are inferred from a set of physical parameters Γ by another deep neural network, while others are shared (by the weight sharing strategy) across all materials.

The detailed architecture of BSDFNet is depicted in the right panel of Fig. 3. As seen, this network consists of five hidden layers with 64 neurons per hidden layer, except the last one which only has 32 neurons. ReLU is adopted as the activation function. Let $\mathbf{v}_i$ be the output of the i-th layer. Note that $\mathbf{v}_i$ in each layer (except the first and last layers) can be subdivided into three parts which are visualized as different colors in Fig. 3:

- $\mathbf{v}^*$: values predicted directly by MetaNet during the inference stage,
- $\mathbf{v}^+$: values computed by the frozen (shared) weights ($\mathbf{W}^+$ and $\mathbf{b}^+$) of BSDFNet,
- $\mathbf{v}^\wr$: values computed by predicted weights ($\mathbf{W}^*$ and $\mathbf{b}^*$) from MetaNet.

According to the above rules, layer i of the network is computed as

$$\mathbf{v}_{i+1}^+ = a(\mathbf{W}_i^+(\mathbf{v}_i^* \oplus \mathbf{v}_i^+ \oplus \mathbf{v}_i^\wr) + \mathbf{b}_i^+) \tag{4}$$

$$\mathbf{v}_{i+1}^\wr = a(\mathbf{W}_i^*(\mathbf{v}_i^+ \oplus \mathbf{v}_i^\wr) + \mathbf{b}_i^*) \tag{5}$$

where $a(\cdot)$ is the element-wise activation function and $\oplus$ denotes element-wise concatenation. Currently, we opt to predict the different channels of reflectance/transmittance independently. Therefore, the output of the last layer is only a scalar. We observe that this can reduce the risk of color drifting.

In our current design of BSDFNet, not only its partial weights ($\mathbf{W}^*$ and $\mathbf{b}^*$) are predicted by MetaNet, but also some neurons' values, i.e., $\mathbf{v}^*$. These neurons serve as important connections between BSDFNet and MetaNet, enabling valid back-propagation of gradients from BSDFNet to MetaNet. Without these neurons, gradients may vanish since the activation functions could enforce some outputs to be zero. In this case, the weights $\mathbf{W}^*$ and $\mathbf{b}^*$ have little impact on the back-propagation of gradients. More detailed derivations and discussions are provided in the supplemental material.

The splitting of $\mathbf{v}$ and the weight sharing strategy adopted in each hidden layer are specially designed to significantly reduce the storage cost and avoid overfitting. With the weight sharing strategy, only 293 parameters (32×4 for $\mathbf{v}^*$, 32×5 for $\mathbf{W}^*$ and 5 for $\mathbf{b}^*$) should be stored for each layered material. A large portion of network parameters are frozen and shared across different materials. The reduced dimensionality of the predicted parameter space also stabilizes the training process and speeds up convergence.

Comparison with Prior Representations. This network has over 13K parameters which are sufficient to reproduce the complex appearance of a wide range of layered materials. In comparison, NBRDF proposed by Sztrajman et al. [2021] only has two small hidden layers for a total of 675 parameters which are further compressed into 32 values. DeepBRDF [Hu et al. 2020] generally adopts a 10-dimensional latent code to represent each measured BRDF. From this, our BSDFNet has a much stronger representative ability than these previous models. This is visually validated in Fig. 4. As seen, NBRDF fails to preserve high-frequency glossy highlights, due to its limited representation ability. Even if we increase the scale of NBRDF (i.e., NBRDF+ in Fig. 4) to match the number of trainable parameters as our BSDFNet, it still produces sub-optimal results as highlighted in the error maps, due to the lack of proper positional encoding as explained in the next subsection. Note that the difficulty of training for meta-learning is closely related to the number of weights to be predicted. NBRDF enables easy convergence by reducing its scale, but at the cost of low representative ability. We guarantee both strong representative ability and easy convergence via weight splitting and sharing. Fan et al. [2022] recently adopted a neural network (NLBRDF) with over 1G trainable parameters to encode layered BRDFs. Without specially-designed acceleration with GPUs, such a large network will incur significant overhead to existing renderers. In comparison, our meta-learning design guarantees the strong expressiveness of BSDFNet with far fewer parameters, and also allows convenient material editing which is not easily supported by Fan et al.'s work.

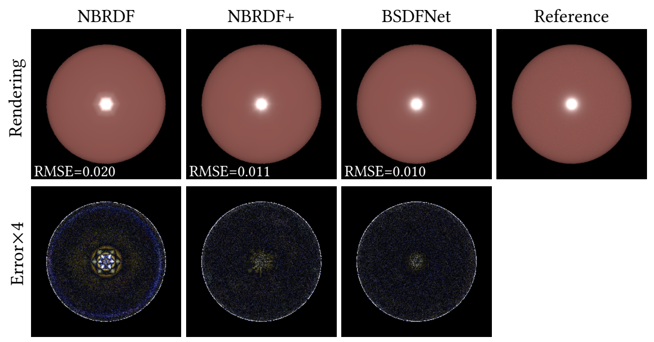


Fig. 4. Comparing the representation ability of NBRDF [Sztrajman et al. 2021], an extended version of NBRDF (NBRDF+), and our BSDFNet. The second row shows the error maps (×4 for better visualization) with respect to the reference images which are generated by Guo et al.'s bidirectional method [Guo et al. 2018]. RMSEs (root mean square errors) are also provided for each method in comparison. More comparisons are provided in the supplemental material.

3.3 MetaNet

State-of-the-art neural BSDF models (e.g., DeepBRDF, NBRDF, and NLBRDF) often lack the ability to quickly generalize in a sample efficient manner to new unseen materials. They also lack editability. For instance, handling a new material for NLBRDF requires extracting the latent code from this material, which comes with a certain cost of generating training examples and optimizing the latent vector with some additional back-propagate steps. In contrast, our MetaNet is designed and trained to predict the partial weights of our coordinate-based network (BSDFNet). This makes our approach more expressive than prior works, allowing it to represent a wider range of appearance. The editability is realized by the mapping between explicit material parameters and BSDFNet's partial weights. Specifically, we represent the material variability by an explicit material parameter vector Γ. In our current setting, Γ includes the surface properties of two interfaces (the surface roughness: α_1

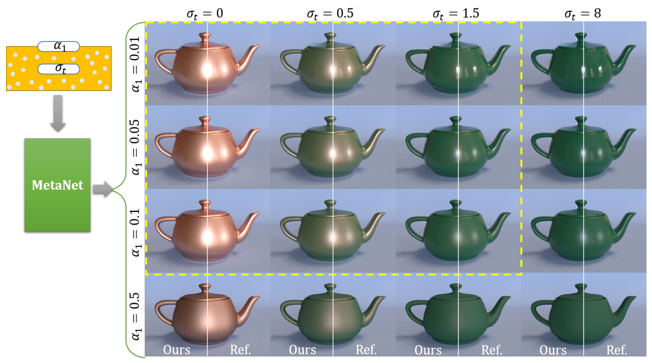


Fig. 5. Illustration of layered material editing via MetaNet. Materials in the dashed yellow box are inside the convex hull of the training set, while others are outside the convex hull, demonstrating the ability of material extrapolation of MetaNet. The reference images are generated by Guo et al.'s bidirectional method [Guo et al. 2018].

and α_2, the relative index of refraction: η_1 and η_2) and the scattering properties of the internal medium[1] (the extinction coefficient: σ_t and the single-scattering albedo: ρ). For a conductor interface, we replace its relative index of refraction with the corresponding Schlick reflectance R_0. The goal of MetaNet is to take a specific Γ as input and generate the partial weights of BSDFNet Θ_F^* to make sure that $f_s(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \Gamma)$ and $F(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \Theta_F)$ behave similarly.

The architecture of MetaNet is shown in the left panel of Fig. 3. It has 16 hidden layers, each of which has 256 neurons. As aforementioned, we currently predict independently for each RGB channel. This results in 6 neurons for the input layer.

We train MetaNet and BSDFNet using a specially-designed training strategy. Once converged, MetaNet directly maps a set of physical parameters Γ to a vector of network parameters of BSDFNet including $\mathbf{W}^*$, $\mathbf{b}^*$ and $\mathbf{v}^*$, through only a forward pass. This is quite different from some gradient-based meta-learning methods [Andrychowicz et al. 2016; Finn et al. 2017; Fischer and Ritschel 2022] which require solving an optimization problem to fine-tune the weights at test time and is convenient for material editing. We illustrate the ability of convenient material editing using MetaNet in Fig. 5. Here, we adjust σ_t and α_1. New layered materials can be generated immediately and their appearance varies smoothly. Note that the strong generalization ability of meta-learning even allows appearance extrapolation (e.g., the materials outside the dashed yellow box in Fig. 5) which is not supported by previous models (e.g., DeepBRDF and NLBRDF).

3.4 Rusinkiewicz Spherical Harmonics Encoding

Although coordinate-based MLPs (like BSDFNet and MetaNet in our framework) provide an efficient way to encode complex scattering behaviors of layered materials, they have difficulty in learning high-frequency functions, such as the near specular reflections for very smooth surfaces. Recent studies have shown that this spectral bias [Basri et al. 2020; Rahaman et al. 2019] can be overcome by using positional encodings [Mildenhall et al. 2020; Müller et al. 2021; Tancik et al. 2020] which significantly improve the representation ability of MLPs.

[1]We adopt an isotropic phase function in our current implementation.

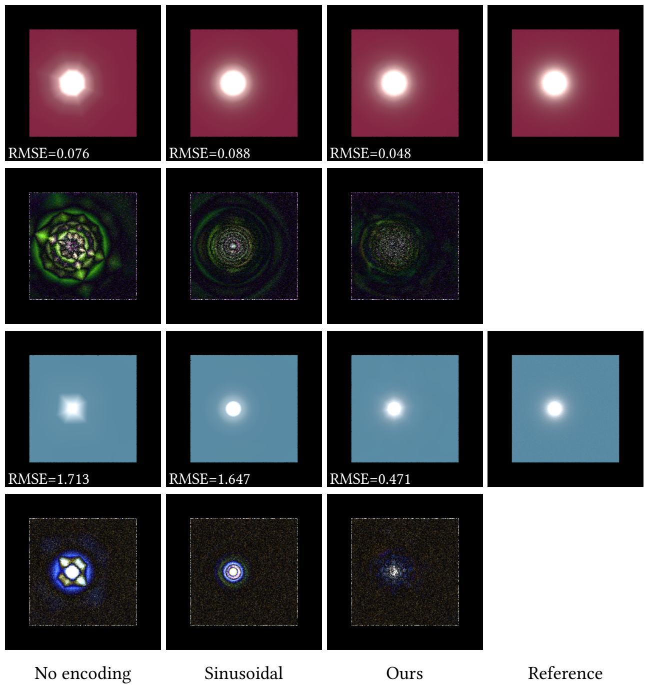


Fig. 6. Comparison of different positional encoding methods. From left to right, we show two predicted BSDFs without positional encoding, with sinusoidal encoding used in [Mildenhall et al. 2020], with our Rusinkiewicz spherical harmonics encoding and the reference. The BSDFs are parameterized by the Rusinkiewicz coordinates. Error maps (×4) are shown for better comparison. More results are provided in the supplemental material.

To allow our BSDFNet to learn high-frequency spherical signals on $\mathcal{S}^2$ better, we adopt a new positional encoding method named Rusinkiewicz Spherical Harmonics Encoding. This method firstly converts the input coordinates of BSDFNet (i.e., $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$) into the Rusinkiewicz coordinates [Rusinkiewicz 1998]: $\boldsymbol{\omega}_h$ and $\boldsymbol{\omega}_d$. The Rusinkiewicz coordinate system adopts an efficient sampling strategy for tabulation by allowing dense sampling near specular highlight regions, thus providing a much better suited set of variables to encode specular lobes. Then, the following mapping is applied to both $\boldsymbol{\omega}_h$ and $\boldsymbol{\omega}_d$:

$$\gamma(\boldsymbol{\omega}) = \left[\boldsymbol{\omega}, Y_0^0(\boldsymbol{\omega}), Y_1^{-1}(\boldsymbol{\omega}), Y_1^0(\boldsymbol{\omega}), Y_1^1(\boldsymbol{\omega}), \dots, Y_l^m(\boldsymbol{\omega}), \dots\right]^\top \quad (6)$$

to map them to a higher-dimensional space. Here, $Y_l^m(\boldsymbol{\omega})$ is a series of spherical harmonics (SH) basis functions of varying order l and degree m. Note that our Rusinkiewicz spherical harmonics encoding is deterministic and allows our network to bias towards data that has more frequency content along $\boldsymbol{\omega}_d$ and $\boldsymbol{\omega}_h$. We currently choose up to ninth order SH basis functions for positional encoding, thus resulting in a total of 168 input dimensions to BSDFNet.

To show the effectiveness of our Rusinkiewicz spherical harmonics encoding method, we compare predicted BSDFs using different positional encoding methods in Fig. 6. As expected, the fitted BSDFs without any positional encoding method fail to recover high-frequency signals of glossy highlights due to the low dimensionality of the input. Incorporating positional encodings, such as 2D sinusoidal encodings widely used in previous work [Mildenhall et al. 2020; Müller et al. 2021], may alleviate this problem. However, sinusoidal encodings will cause distracting ghosting artifacts along the glossy lobes. In comparison, our proposed encoding method is free from these artifacts and achieves high-quality results that are much closer to the reference.

3.5 Two-phase Training Strategy

To speed up convergence, our networks are trained in a two-phase manner, as summarized in Algorithm 1. In the first phase, BSDFNet and MetaNet are trained jointly using the dataset described below. This stage is designed to help the training process to be initially more stable, since BSDFNet and MetaNet have different objectives. In the second phase, the training steps of BSDFNet and MetaNet are interleaved in a meta-training manner, as specified in lines 9-24 in Algorithm 1. During this phase, each time we sample a layered material from the dataset, its sample set $\mathbf{X}$ (with N_{ω_i} varying incident direction ω_i and N_{ω_o} varying outgoing direction ω_o) is split into two subsets: $\mathbf{X}_1$ and $\mathbf{X}_2$. $\mathbf{X}_1$ is first used to train BSDFNet for K_1 steps, while freezing the weights in MetaNet. These training steps aim to stabilize the shared weights in BSDFNet. Then, $\mathbf{X}_2$ is used to train MetaNet for K_2 steps, updating Θ_M only. This training phase lets our framework generalize to new materials more easily.

3.6 Training Details

We train our networks with TensorFlow [Abadi et al. 2015] on a server with eight NVIDIA 2080Ti GPUs. We use the Adam optimizer with default parameters and an initial learning rate of 0.0005 to train all networks. The learning rate decays by 0.99 for every 8 epochs. For layered materials with reflective/conductive base layers, we train a single model with 225 epochs (100 epochs for the first training phase and 125 epochs for the second training phase), taking roughly 50 hours. For layered materials with transmissive/dielectric base layers, we train two models for the BRDF part and the BTDF part separately. Each model is trained with 200 epochs (100 epochs for the first training phase and 100 epochs for the second training phase), taking roughly 48 hours.

Loss Function. The loss function for training our networks is inspired by the reverse Huber loss which has shown great success in the dense depth prediction problem [Laina et al. 2016; Wang et al. 2020]. Specifically, our loss function is defined as

$$\mathcal{L}(f, f_s) = \begin{cases} |\mathcal{T}(f) - \mathcal{T}(f_s)| & |\mathcal{T}(f) - \mathcal{T}(f_s)| \le t \\ \frac{\|\mathcal{T}(f) - \mathcal{T}(f_s)\|_2^2 + t^2}{2t} & |\mathcal{T}(f) - \mathcal{T}(f_s)| > t \end{cases} \tag{7}$$

where t is a positive threshold and $\mathcal{T}$ is the μ-law transformation [Kalantari and Ramamoorthi 2017] used to compress the high dynamic range of the reflectance value:

$$\mathcal{T}(f) = \text{sign}(f)\frac{\log(1 + \text{abs}(f)\mu)}{\log(1 + \mu)} \tag{8}$$

Algorithm 1 Two-phase training

Input: A training dataset $\mathcal{D}$ with $N = N_\Gamma \times N_{\omega_i} \times N_{\omega_o}$ samples
Output: Trained MetaNet $M(\Gamma, \Theta_M)$ and BSDFNet $F(\omega_i, \omega_o, \Theta_F)$

```
1:  Randomly initialize $\Theta_M$ and $\Theta_F$
2:  for iterations in the first phase do
3:      Sample a material from $\mathcal{D}$ with a sample set $\mathbf{X}$
4:      for samples in $\mathbf{X}$ do
5:          Feed-forward propagation of $M(\Theta_M)$ and $F(\Theta_F)$
6:          Evaluate the loss $\mathcal{L}$ and update both $\Theta_M$ and $\Theta_F$
7:      end for
8:  end for
9:  for iterations in the second phase do
10:     Sample a material from $\mathcal{D}$ with a sample set $\mathbf{X}$
11:     Split $\mathbf{X}$ into $\mathbf{X}_1$ and $\mathbf{X}_2$
12:     for $K_1$ steps do
13:         for samples in $\mathbf{X}_1$ do
14:             Feed-forward propagation of $F(\Theta_F)$
15:             Evaluate the loss $\mathcal{L}$ and update $\Theta_F$
16:         end for
17:     end for
18:     for $K_2$ steps do
19:         for samples in $\mathbf{X}_2$ do
20:             Feed-forward propagation of $M(\Theta_M)$ and $F(\Theta_F)$
21:             Evaluate the loss $\mathcal{L}$ and update $\Theta_M$
22:         end for
23:     end for
24: end for
```

Table 1. Sampling functions used to generate layered materials for the LayeredBRDF and LayeredBTDF datasets. $\mathcal{U}$ denotes the uniform distribution and $\mathcal{V}$ means sampling the set with the discrete uniform distribution.

Parameter	Symbol	Sampling
Roughness for top interface	α_1	$10^{\mathcal{U}(-3,-0.5)}$
Roughness for bottom interface	α_2	$10^{\mathcal{U}(-3,0)}$
IOR for dielectric interface	η_1, η_2	$\mathcal{U}(1.05, 2)$
Fresnel for conductive interface	R_0	$\mathcal{U}(0, 1)$
Single-scattering albedo for medium	ρ	$1 - \mathcal{U}(0, 1)^2$
Extinction coefficient for medium	σ_t	$\mathcal{V}(\{0, 1, 2, 5\})$

with μ controlling the amount of compression. We currently set $t = 0.1$ and $\mu = 32$. Empirically, this loss function provides a good balance between L1 error and L2 error. In our context, L2 aims to reconstruct the potential large values at highlight regions while L1 is responsible for preserving the long tail of the layered BSDF. Moreover, this loss function is continuous and first order differentiable at the threshold t where the switch from L1 to L2 occurs.

Dataset Preparation. To train the proposed networks, we collect two datasets: a *LayeredBRDF* dataset for layered materials with reflective/conductive base layers and a *LayeredBTDF* dataset for layered materials with transmissive/dielectric base layers. Both datasets are generated according to Guo et al.'s method, using a sampling rate of 128 spp. The LayeredBRDF dataset contains $N_\Gamma = 12,000$ BRDFs and the LayeredBTDF dataset contains $N_\Gamma = 10,000$ BSDFs. The distributions of their parameters are listed in Table 1. Currently, we use GGX as the normal distribution function at interfaces, resulting in two roughness parameters α_1 and α_2 for either conductive or dielectric interfaces. For conductive base layers, we adopt the Schlick Fresnel approximation with R_0. The inside medium is described by the extinction coefficient σ_t and single-scattering albedo ρ. For each BRDF in the LayeredBRDF dataset, we sample 25^4 pairs of spherical angles (θ,ϕ) of the $\boldsymbol{\omega}_h$ and $\boldsymbol{\omega}_d$ vectors in the Rusinkiewicz parameterization [Rusinkiewicz 1998]. We perform stratified sampling along the elevation angle θ and azimuth angle ψ on the upper hemisphere, where $\theta \in [0, \frac{\pi}{2})$ and $\phi \in [0, 2\pi)$. For BTDFs in the LayeredBTDF dataset, the elevation angle θ covers the whole sphere, i.e., $\theta \in [0, \pi)$, resulting in 4×25^4 pairs of spherical angles.

4 RENDERING WITH METALAYER

Renderer Integration. Integrating our Metalayer model into existing renderers is quite easy and straightforward, which can be seen as another benefit of this model. Once BSDFNet and MetaNet converge, BSDFNet can be used as similarly as other parametric BSDF models since BSDFNet in our current design is lightweight and portable. Consequently, the learned BSDFNet can run without the support of GPU resources, thus neglecting any workload from data transfer between CPU and GPU. Except for positional encodings which are precomputed and stored in a look-up table, BSDFNet only contains matrix multiplication/addition operations and ReLU activation functions that are easy for data-level parallelism. In practice, we parallelize these operations via the Intel Advanced Vector Extensions 512 (AVX-512) instruction set [Intel 2023].

For MetaNet, it should run only once for a given material, although it contains much more parameters than BSDFNet. This design significantly improves the performance of our model during runtime, while still preserving its strong representation ability by transferring much computation to the precomputation step. To support spatially-varying layered BSDFs, only the parameters predicted by MetaNet are stored in textures. Other trainable parameters in BSDFNet are fixed and shared across all layered materials.

Importance Sampling. The usage of a new BSDF model in a ray/path tracing-based renderer also requires the ability to properly sample the BSDF and to estimate the probability of this sampling. Considering that the appearance of layered materials typically has multiple dominant lobes, we adopt the following multi-lobe probability distribution, inspired by [Belcour 2018]:

$$p(\boldsymbol{\omega}_o|\boldsymbol{\omega}_i) = \frac{E_{\text{all}}}{\sum_{i=0}^{N} E_i p_i} \sum_{i=0}^{N} E_i \rho_i(\alpha_i) \tag{9}$$

where E_i is the energy for the i-th lobe, E_{all} is the total energy, p_i is the probability density function (PDF) of sampling the visible normals [Heitz and d'Eon 2014], and ρ_i is the PDF of the i-th microfacet model. Currently, we select two lobes (R and TRT) for layered materials with reflective base layers and select three lobes (R, TRT, and TT) for layered materials with transmissive base layers. Please refer to [Belcour 2018] for more details on the estimation of the parameters. During rendering, we randomly selected one lobe based on E_i and importance sample the visible normals [Heitz and d'Eon 2014].

Table 2. Comparison of different learning-based BSDF models. Our proposed MetaLayer model supports BTDFs with two transmissive interfaces, textured material parameters and perceptual material editing. It can also run with pure CPU resources. Here, – means partial support.

Method	BTDF	Texture	Edit	CPU
NBRDF [Sztrajman et al. 2021]	×	×	×	✓
NLBRDF [Fan et al. 2022]	×	✓	–	×
MetaLayer (Ours)	✓	✓	✓	✓

Summary of Abilities. Our MetaLayer model has many desired properties when integrated into existing renderers. Table 2 compares the abilities of the proposed MetaLayer model against several existing models. Currently, only our model fully supports layered BTDFs with two transmissive interfaces. It seems that some other models may also be extended to allow the evaluation of transmittance, but the extension is not that straightforward. Supporting textured or spatially-varying material parameters is also important for BSDF models, which is enabled by our model and NLBRDF [Fan et al. 2022]. Perhaps the main advantage of our model, as compared with existing learning-based models, is its convenience in material editing. NLBRDF [Fan et al. 2022] also allows material editing, but needs significant efforts to fine-tune the latent codes of unseen materials. Another benefit of our model is that it can run with pure CPU resources, thanks to the lightweight design of BSDFNet. Therefore, porting this model to different platforms is relatively easy, and running the model does not involve time-consuming CPU-GPU data transfer.

5 RESULTS AND DISCUSSION

We have implemented our MetaLayer model in the Mitsuba renderer [Jakob 2010] as a new BSDF plugin. In this section, we demonstrate the effectiveness and efficiency of our method on layered materials with various combinations of parameters. All test materials in this section are new and never appear in the training dataset. All results are generated on a workstation with an Intel Core i9-9900X processor and 64GB RAM. As aforementioned, our lightweight BSDF model does not require any GPU resource at runtime. Reference images are rendered using Guo et al.'s bidirectional method [Guo et al. 2018] at a high sampling rate (2048 spp by default). Please refer to the accompanying video for animated versions of several scenes.

5.1 Comparison on Reflective Layers

We first compare our method with previous work on modeling layered materials with reflective base layers. Fig. 7 compares our neural method to Belcour's efficient model based on statistical operators [Belcour 2018]. As a typical approximate model, it has many

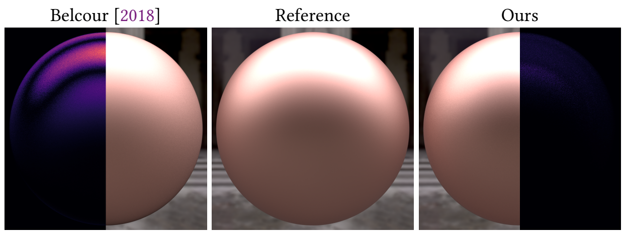


Fig. 7. Comparison to the approximate model of Belcour [2018] on a layered material with rough interfaces ($\alpha_1 = 0.1$, $\alpha_2 = 0.3$).

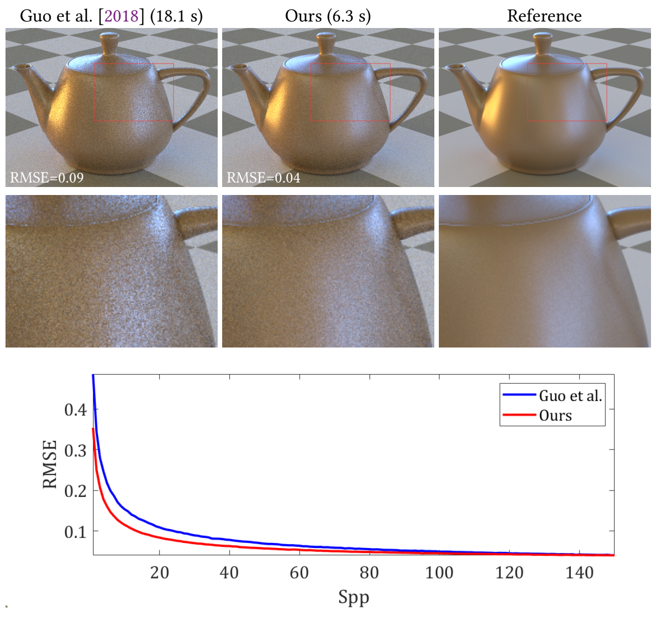


Fig. 8. Comparison to the method of Guo et al. [2018] with an equal sampling rate (64 spp, in the first row). Closeups in the second row highlight the high variance caused by stochastic evaluation in Guo et al.'s BSDF model. Note that our BSDFNet, as a parametric BSDF model, runs much faster than Guo et al.'s model at the same sampling rate for this scene. The last row shows the evolution of RMSE of both methods with respect to the sampling rate (Spp).

limitations such as inaccuracy for very rough interfaces and ignorance of multiple scattering. As expected, our method produces the appearance of layered materials much closer to the reference than the Belcour's model, especially for rough interfaces ($\alpha_1 = 0.1$, $\alpha_2 = 0.3$) shown in Fig. 7. For a fair comparison, the layered material in Fig. 7 does not contain any scattering medium ($\sigma_t = 0$).

In Fig. 8, we compare our method to the model of Guo et al. [2018], the most accurate layered BSDF model to date. The medium's thickness is set as $\sigma_t = 1.0$, which incurs obvio us multiple scattering events that are forbidden by Belcour's model. However, our model can correctly reproduce the appearance of this layered material, closely matching the reference. Note that our method consumes much less time than Guo et al.'s model on this material when

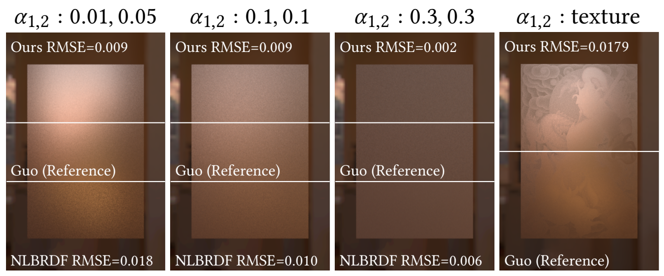


Fig. 9. Comparison to NLBRDF [Fan et al. 2022] and the method of Guo et al. [2018] with an equal sampling rate (256 spp) on a conductive plane with varying roughness. Note that NLBRDF does not support spatially-varying roughness.

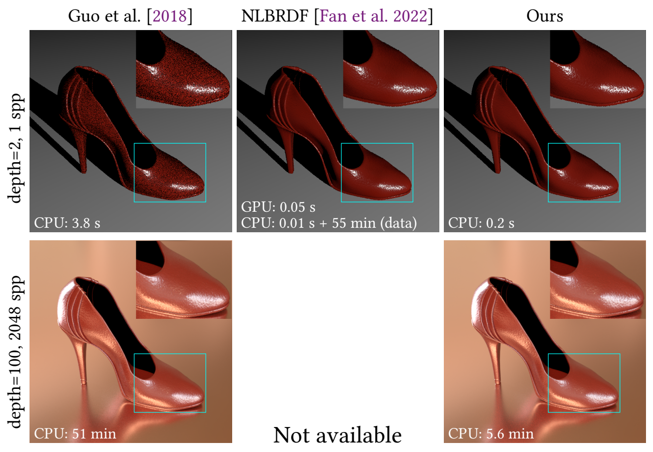


Fig. 10. Comparison to NLBRDF [Fan et al. 2022] and the method of Guo et al. [2018] on the Shoe scene. Top row: tracing with the maximum path depth 2. Bottom row: tracing with the maximum path depth 100. The CPU-GPU collaborative rendering mode in NLBRDF currently makes it difficult to handle path tracing with long paths, e.g., depth=100.

rendering at the same sampling rate (64 spp). It also achieves low variance as highlighted in the closeups, implying that rendering with our neural BSDF model converges much faster. The convergence curves in the last row clearly show that our method converges faster than the model of Guo et al.

In Figs. 9 and 10, we make comparisons to Fan et al.'s NLBRDF model [Fan et al. 2022] which also leverages neural networks to represent layered materials. Compared with our neural method, NLBRDF generally has two limitations in practice. First, NLBRDF needs to prepare BRDF data (roughly 55 minutes) for materials with any unseen parameters and optimize their latent codes using the data before rendering. This restricts it to only support material interpolation within BRDFs with known data and latent codes. In particular, the surface properties (e.g., surface roughness) are not allowed to be freely edited. Therefore, spatially-varying surface properties

Table 3. Quantitative comparison in terms of mean and standard deviation (STD) against NBRDF and an extended version of NBRDF (NBRDF+). The values (RMSE) are counted over 100 layered BRDFs with random parameters, lit by a single point light. Best scores are highlighted in **bold**.

	NBRDF	NBRDF+	Ours
Mean	0.891	0.873	**0.483**
STD	1.960	2.488	**1.043**

Table 4. Quantitative evaluation in terms of mean and standard deviation (STD) of different positional encoding methods. The values (RMSE) are counted over 100 layered BRDFs with random parameters, lit by a single point light. Best scores are highlighted in **bold**.

	Sinusoidal (ω_i, ω_o)	Sinusoidal (ω_h, ω_d)	SH (ω_i, ω_o)	SH (Ours) (ω_h, ω_d)
Mean	1.249	1.265	1.230	**0.998**
STD	2.847	2.241	2.833	**1.900**

are not supported by NLBRDF [2]. With our MetaNet, the weights of each material can be determined in less than 1 second. This allows us to freely edit any material parameters and easily support spatially-varying cases, as shown in Fig. 9. Second, the CPU-GPU collaborative rendering strategy in Fan et al.'s method is difficult to handle path tracing with long paths, such as the case in the second row of Fig. 10. Again, our method is free from this limitation and can efficiently run in a wide range of complex scenes.

Table 3 further demonstrates the advantage of our model by quantitatively comparing against NBRDF [Sztrajman et al. 2021] and its variant on 100 randomly selected layered BRDFs from the test set. More visual comparisons are provided in the supplemental material.

To further verify the efficiency and high fidelity of our model, we make a series of pair-wise comparisons with Guo et al.'s method on the `Dragon` scene with varying extinction coefficient σ_t. Fig. 11 shows the results when σ_t increases from 0.0 to 3.0. As seen, our method produces visually similar appearance to Guo et al.'s method on different layered materials. The absence of Monte Carlo sampling in our neural model helps us to achieve lower-variance results than Guo et al.'s on different configurations, at the same sampling rate (64 spp). Quantitative evaluations in terms of root mean square error (RMSE), as compared with high-quality reference images, further show the benefit of our model in reducing the variance. It should be noted that the runtime cost of Guo et al.'s model increases prominently as σ_t increases, since a larger σ_t incurs more scattering events in the medium. This is clearly shown in Fig. 12 where we plot the time spent by layered materials with varying extinction coefficient σ_t. The comparisons tell that our neural model has a constant runtime cost since the architecture of BSDFNet is fixed. When $\sigma_t = 5.0$, our method achieves nearly 6× runtime acceleration as compared with Guo et al.'s method on this `Dragon` scene at the same sampling rate.

5.2 Comparison on Transmissive Layers

In Fig. 13, we compare our MetaLayer model to Guo et al.'s model on layered materials with transmissive base layers. With proper training on our LayeredBTDF dataset, our model successfully reproduces the translucent effects caused by transmissive layers. The continuous representation of BSDFNet allows us to generate images with less noise, as compared with Guo et al.'s model which involves Monte Carlo sampling in evaluation. Similar to the reflective cases in Fig. 11, Guo et al.'s model suffers a significant degradation in runtime performance and image quality (e.g., RMSE) as the extinction coefficient σ_t increases. This is further evidenced in the last row of Fig. 13 where we see our model converges faster than its competitor on this transmissive case ($\sigma_t = 2.0$).

[2]NLBRDF only allows the scattering properties of the internal medium to be spatially-varying.

5.3 Spatially-varying Cases

The lightweight design of our BSDFNet allows us to support spatially-varying BSDFs, without any GPU assistance. When the parameters of a layered material are encoded in textures, we run MetaNet for each texel and the resulting weights of BSDFNet are stored in a look-up table (LUT). This LUT is then retrieved at rendering time, with each slot representing a layered material. We currently use the nearest neighbor search to retrieve the LUT, without introducing additional storage and computational overhead. However, our method supports linearly interpolating the LUT. Fig. 15 shows our results of the `Globe` scene with spatially-varying BRDFs. Here, we show an equal-time comparison with Guo et al's bidirectional method. As seen, our method is able to produce high-quality result with much lower variance as compared with Guo et al's method. Even at the same sampling rate (64 spp), our method still outperforms its competitor due to the intrinsic smoothness and robustness brought by our trained neural representation.

In Fig. 16, we edit the `Kettle` scene with spatially-varying layered materials (reflective base layers). Processing these textures takes less than one minute with MetaNet. The processing time increases linearly with the resolution of the texture and is much lower than the rendering time. Editing of transmissive layered materials is demonstrated in Fig. 17 where we edit either the surface roughness or the albedo of the medium, with spatially-varying parameters.

Concerning memory consumption, our method currently requires a 293D vector for each material. For instance, the material in the left image of Fig. 16 requires roughly 1.7GB memories. In comparison, Fan's method only requires a 32D latent code per material. However, this 32D code should be re-optimized for each unseen material and relies on a very large network (1G parameters) for decoding.

5.4 Validation of Network Design

Our networks have several key design choices. To validate their effectiveness, we report the performance of our complete model as

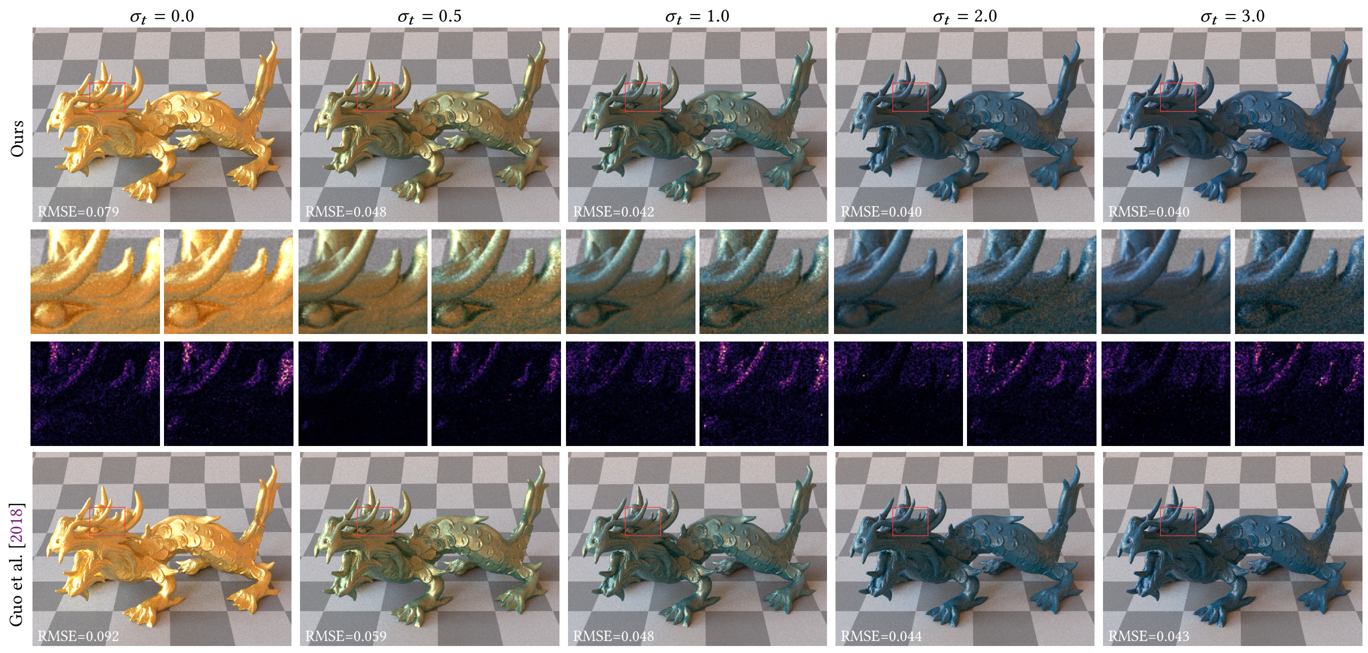


Fig. 11. Impact of the extinction coefficient σ_t of the medium. We compare our method to Guo et al. [2018] on the Dragon scene with increasing σ_t. RMSE values are provided for each image. Closeups in the second row and their corresponding error maps in the third row (left: Ours, right: Guo et al.) highlight the benefit of method in variance reduction.

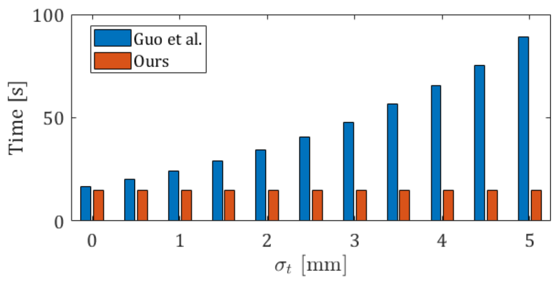


Fig. 12. Variations of the runtime cost (at 64 spp) of the Dragon scene with respect to the extinction coefficient σ_t.

compared with two ablated models. In Fig. 18, we plot the evolution of the training loss for the three models in comparison. The first ablated model (the green curve in Fig. 18) predicts the weights of two neurons $\mathbf{v}^l$ per-layer instead of one neuron in our current design (see Fig. 3). This makes BSDFNet more flexible but leads to poor convergence, due to the large variations of more weights. Due to the relatively large scale of our BSDFNet, predicting all weights, in the same way as in NBRDF [Sztrajman et al. 2021], will make the training hard to converge. The second ablated model (the blue curve in Fig. 18) removes $\mathbf{v}^*$ completely. As explained previously, this has the risk of incurring the vanishing gradient problem. Obviously, our complete model converges faster than its ablated variants.

As an important building block in our framework, the Rusinkiewicz spherical harmonics encoding method has already shown its advantage in preserving high-frequency details for some typical glossy lobes in Fig. 6. In Table 4, we further validate its superiority by quantitatively comparing against some other encoding methods on 100 randomly selected layered BRDFs from the test set. More visual comparisons are provided in the supplemental material.

5.5 Validation of the Loss Function

We also validate the effectiveness of the loss function, as compared with the conventional L1 loss and L2 loss. Our loss function combines the benefits of both L1 loss and L2 loss. This allows the trained model to preserve both low-frequency information (diffuse reflection) and high-frequency information (glossy highlights). Table 5 reports the prediction accuracy in terms of RMSE for models trained on different loss functions. We evaluate the accuracy using the rendered images under either point lighting or environmental lighting. The model trained with our loss function achieves the lowest error in either case. A typical example is shown in Fig. 19. The error occurs mainly at the highlight regions, while our loss function is beneficial for reducing this error, as clearly shown by the scan line comparisons.

5.6 Validation of the Two-phase Training Strategy

We finally validate the effectiveness of the proposed two-phase training strategy in Fig. 20. As compared with the traditional one-phase training strategy that always trains BSDFNet and MetaNet

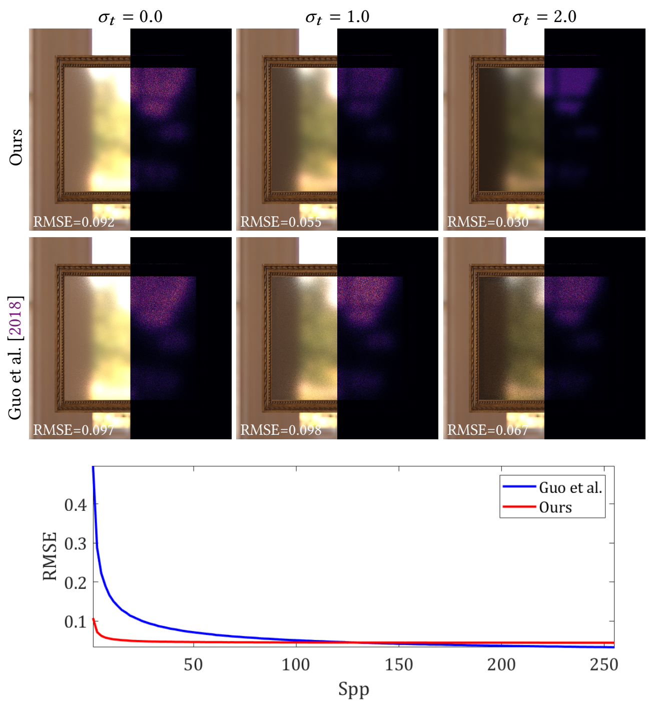


Fig. 13. Rendering the Frosted Glass scene ($\alpha_1 = \alpha_2 = 0.02$) using our MetaLayer model (the top row) and Guo et al.'s model (the bottom row), respectively. RMSE values are provided for each image. The last row shows the evolution of RMSE of both models on the transmissive case ($\sigma_t = 2.0$) with respect to the sampling rate (Spp).

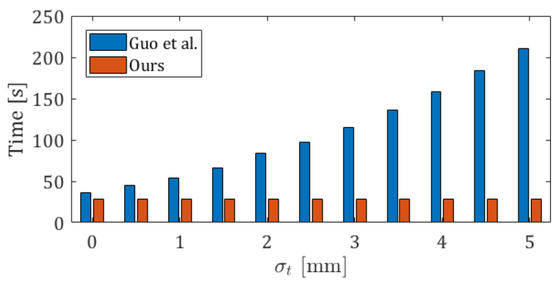


Fig. 14. Variations of the runtime cost (at 64 spp) of the Frosted Glass scene with respect to the extinction coefficient σ_t.

Table 5. Quantitative evaluation (in terms of RMSE) of different loss functions used to train our networks. The values are averaged over 100 layered BRDFs with random parameters, lit by a single point light and an environment light respectively. Best scores are highlighted in **bold**.

	L1 loss	L2 loss	Our loss
Point light	0.096	0.106	**0.092**
Environment light	0.018	0.018	**0.017**

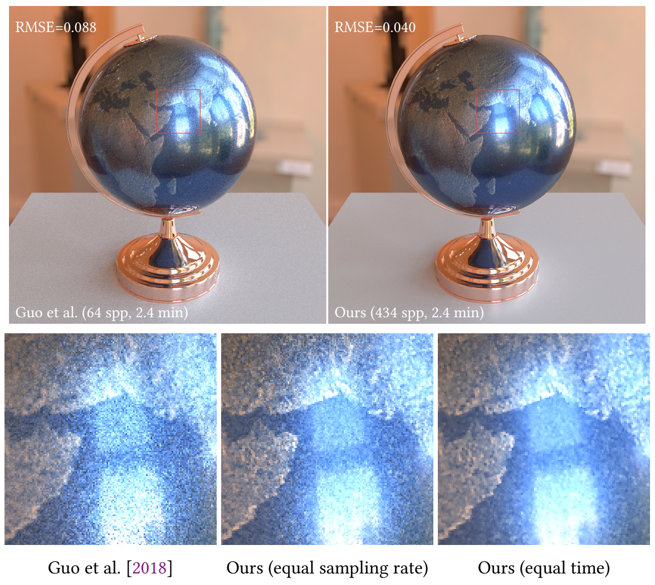
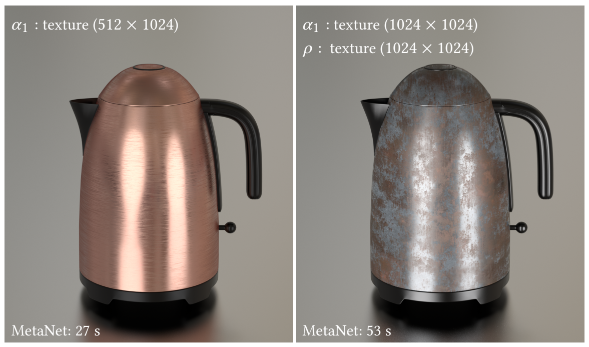
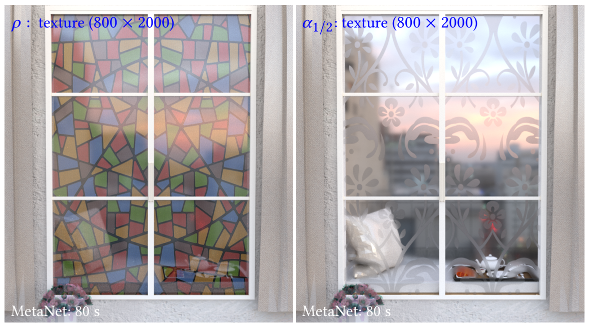


Fig. 15. Visual comparison to the method of Guo et al. [2018] on the Globe scene with spatially-varying layered materials.



α_1 : texture (512 × 1024)
MetaNet: 27 s
α_1 : texture (1024 × 1024)
ρ : texture (1024 × 1024)
MetaNet: 53 s

Fig. 16. Editing reflective layered materials with spatially-varying (textured) parameters. The resolution of each texture and the processing time of MetaNet are shown for each case.



ρ : texture (800 × 2000)
MetaNet: 80 s
$\alpha_{1/2}$: texture (800 × 2000)
MetaNet: 80 s

Fig. 17. Editing transmissive layered materials with spatially-varying (textured) parameters. The resolution of each texture and the processing time of MetaNet are shown for each case.

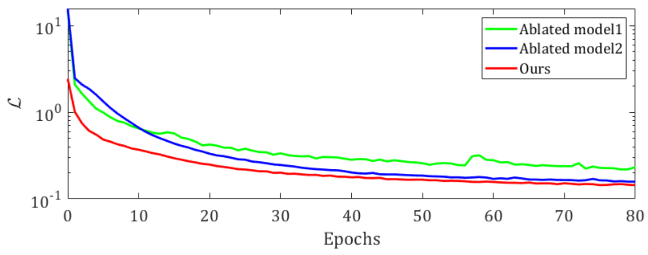


Fig. 18. Training loss versus epochs for different variants of our model.

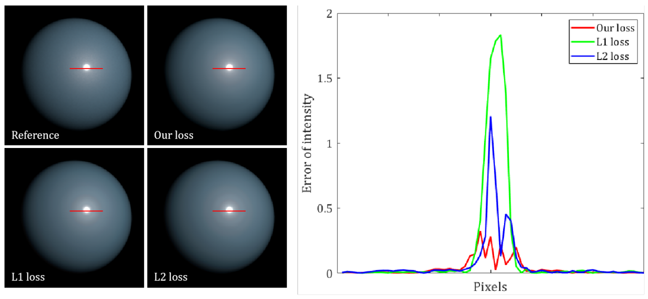


Fig. 19. Validation of different loss functions. Here, we compare the error of intensity (as compared with reference) along the red scan line.

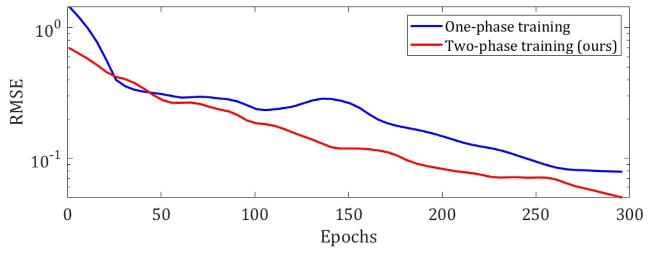


Fig. 20. Evolution of RMSE (evaluated on 100 randomly selected layered BRDFs from the test set) with respect to the training epoch, for two different training strategies.

jointly, our specially-designed training strategy makes the training more stable and allows the trained model to recover layered BSDFs with lower errors.

6 DISCUSSION, LIMITATION AND FUTURE WORK

Smooth Surfaces. Our MetaLayer model may incur large errors for very smooth surfaces with near-specular reflection, possibles due to the μ-law transformation and the limited accuracy of floating-point operations in the networks. As shown in Fig. 21, when the surface roughness approaches 0.001, large errors appear at the highlight region, resulting in inconsistent shading as compared with the reference.

Multiple Layers and Anisotropy. Our model currently handles a single layer with two interfaces. The extension to multiple layers is straightforward. We only need to generate the training examples from multi-layered materials and expand the parameter set Γ

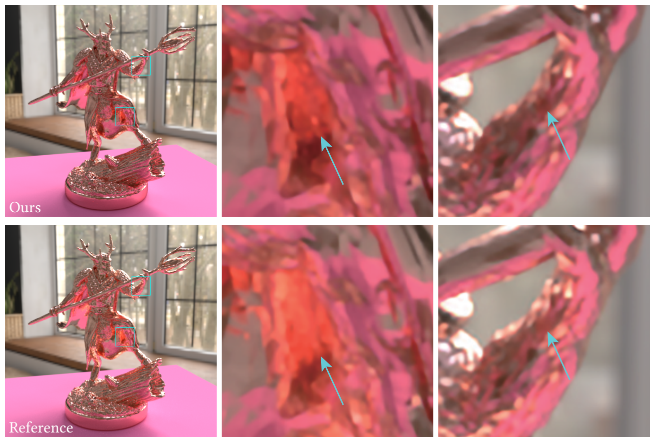


Fig. 21. A failure case for our model. Our model under-estimates the highlight (see the closeups) for very small roughness, e.g., $\alpha_1 = 0.001$ and $\alpha_2 = 0.01$ in this case. The cyan arrows in the closeups highlight the differences.

(as well as the input nodes of MetaNet) by including other layers' properties. Similarly, our model can also be extended to handle anisotropic interfaces [Weier and Belcour 2020] by adjusting the parameter set and training set. However, as the number of parameters increases, training our networks becomes difficult. We leave this as the future work.

Optimal Importance Sampling. Importance sampling is the key to ray/path tracing-based renderers. The proposed multi-lobe sampling algorithm is simple, efficient, and satisfies many layered materials. However, this is not the optimal importance sampling algorithm for layered materials since layering multiple interfaces would result in more than two lobes. Consequently, investigating an optimal and robust importance sampling strategy for arbitrary layered materials (including anisotropy and multiple layers) deserves further study.

Measured Materials. Adopting data-driven BRDF models using real material measurements has always been the pursuit of many physically-based renderers. However, extending our MetaLayer model to support measured materials is not that straightforward, since MetaNet requires explicit parameters of the material as input. A possible solution is to compress measured materials into a low-dimensional latent space (like DeepBRDF [Hu et al. 2020]), and then feed the latent code to MetaNet. Nevertheless, one major goal of the proposed layered BSDF model is convenient artistic design for layered materials which is not compatible with measured materials.

Energy Conservation. As a neural representation based on neural networks, our model does not guarantee energy conservation since the trained networks may produce unpredictable error. However, in practice, we never observed any artifact that is caused by the violation of energy conservation. Investigating neural BSDF models that always conserve energy would be an interesting future work.

7 CONCLUSION

Layered materials are ubiquitous in the natural world. This paper has proposed a new method, named as MetaLayer, to reproduce the complex appearance from layered materials, with high visual quality, low runtime cost, and strong editing ability. The key idea is incorporating meta-learning into layered appearance modeling, which addresses several tough issues of previous neural appearance models, making learning-based appearance models more practical. Besides, several novel designs, including a new positional encoding method, a well-designed training strategy and a weight sharing strategy, further improve the effectiveness and efficiency of the proposed model. To our best knowledge, this is the first layered material model that incorporates meta-learning. We believe the new methodology behind our MetaLayer model can be applied to other complex appearance modeling/rendering scenarios.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable suggestions. This work was supported by the National Natural Science Foundation of China (No. 61972194 and No. 62032011) and the Natural Science Foundation of Jiangsu Province (No. BK20211147).

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

Miika Aittala, Timo Aila, and Jaakko Lehtinen. 2016. Reflectance Modeling by Neural Texture Synthesis. *ACM Trans. Graph.* 35, 4, Article 65 (jul 2016), 13 pages.

Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. 2016. Learning to Learn by Gradient Descent by Gradient Descent. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (Barcelona, Spain) *(NIPS'16)*. 3988–3996.

Gladimir V. G. Baranoski and Jon G. Rokne. 2001. Efficiently simulating scattering of light by leaves. *The Visual Computer* 17 (2001), 491–505.

Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. 2020. Frequency Bias in Neural Networks for Input of Non-Uniform Density. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 64, 10 pages.

Mégane Bati, Pascal Barla, and Romain Pacanowski. 2021. An Inverse Method for the Exploration of Layered Material Appearance. *ACM Trans. Graph.* 40, 4, Article 176 (jul 2021), 15 pages.

Mégane Bati, Romain Pacanowski, and Pascal Barla. 2019. Comparative Study of Layered Material Models. In *Workshop on Material Appearance Modeling*, Reinhard Klein and Holly Rushmeier (Eds.). The Eurographics Association, 17–21.

Laurent Belcour. 2018. Efficient Rendering of Layered Materials Using an Atomic Decomposition with Statistical Operators. *ACM Trans. Graph.* 37, 4, Article 73 (jul 2018), 15 pages.

Sai Bi, Stephen Lombardi, Shunsuke Saito, Tomas Simon, Shih-En Wei, Kevyn Mcphail, Ravi Ramamoorthi, Yaser Sheikh, and Jason Saragih. 2021. Deep Relightable Appearance Models for Animatable Faces. *ACM Trans. Graph.* 40, 4, Article 89 (jul 2021), 15 pages.

Qiang Dai, Jiaping Wang, Yiming Liu, John Snyder, Enhua Wu, and Baining Guo. 2009. The Dual-microfacet Model for Capturing Thin Transparent Slabs. *Computer Graphics Forum* 28, 7 (2009), 1917–1925.

Heloise de Dinechin and Laurent Belcour. 2022. Rendering Layered Materials with Diffuse Interfaces. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 1, Article 13 (may 2022), 12 pages.

Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. 2018. Single-Image SVBRDF Capture with a Rendering-Aware Deep Network. *ACM Trans. Graph.* 37, 4, Article 128 (July 2018), 15 pages.

Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. 2019. Flexible SVBRDF Capture with a Multi-Image Deep Network. *Computer Graphics Forum* 38, 4 (2019), 1–13.

Yue Dong. 2019. Deep appearance modeling: A survey. *Visual Informatics* 3, 2 (2019), 59–68.

Julie Dorsey and Pat Hanrahan. 1996. Modeling and Rendering of Metallic Patinas. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 387–396.

Jonathan Dupuy and Wenzel Jakob. 2018. An Adaptive Parameterization for Efficient Material Acquisition and Rendering. *ACM Trans. Graph.* 37, 6, Article 274 (dec 2018), 14 pages.

Serkan Ergun, Sermet Önel, and Aydin Ozturk. 2016. A General Micro-flake Model for Predicting the Appearance of Car Paint. In *Eurographics Symposium on Rendering - Experimental Ideas & Implementations*, Elmar Eisemann and Eugene Fiume (Eds.). The Eurographics Association.

Jiahui Fan, Beibei Wang, Miloš Hašan, Jian Yang, and Ling-Qi Yan. 2022. Neural Layered BRDFs. (2022).

J. Filip and R. Vávra. 2014. Template-Based Sampling of Anisotropic BRDFs. *Computer Graphics Forum* 33, 7 (2014), 91–99.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17)*. JMLR.org, 1126–1135.

Michael Fischer and Tobias Ritschel. 2022. Metappearance: Meta-Learning for Visual Appearance Reproduction. *ACM Trans Graph (Proc. SIGGRAPH Asia)* 41, 4 (2022).

Luis E. Gamboa, Adrien Gruson, and Derek Nowrouzezahrai. 2020. An Efficient Transport Estimator for Complex Layered Materials. *Computer Graphics Forum* 39, 2 (2020), 363–371.

Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. 2019. Deep Inverse Rendering for High-Resolution SVBRDF Estimation from an Arbitrary Number of Images. *ACM Trans. Graph.* 38, 4, Article 134 (July 2019), 15 pages.

Jinwei Gu, Ravi Ramamoorthi, Peter Belhumeur, and Shree Nayar. 2007. Dirty Glass: Rendering Contamination on Transparent Surfaces. In *Rendering Techniques*, Jan Kautz and Sumanta Pattanaik (Eds.). The Eurographics Association.

Jie Guo, Shuichang Lai, Chengzhi Tao, Yuelong Cai, Lei Wang, Yanwen Guo, and Ling-Qi Yan. 2021. Highlight-Aware Two-Stream Network for Single-Image SVBRDF Acquisition. *ACM Trans. Graph.* 40, 4, Article 123 (jul 2021), 14 pages.

Jie Guo, Jinghui Qian, Yanwen Guo, and Jingui Pan. 2017. Rendering Thin Transparent Layers with Extended Normal Distribution Functions. *IEEE Transactions on Visualization and Computer Graphics* 23, 9 (2017), 2108–2119.

Yu Guo, Miloš Hašan, and Shuang Zhao. 2018. Position-Free Monte Carlo Simulation for Arbitrary Layered BSDFs. *ACM Trans. Graph.* 37, 6, Article 279 (dec 2018), 14 pages.

Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. 2020. MaterialGAN: Reflectance Capture Using a Generative SVBRDF Model. *ACM Trans. Graph.* 39, 6, Article 254 (Nov. 2020), 13 pages.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. HyperNetworks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkpACe1lx

Pat Hanrahan and Wolfgang Krueger. 1993. Reflection from Layered Surfaces Due to Subsurface Scattering. Association for Computing Machinery, New York, NY, USA, 165–174.

E. Heitz and E. d'Eon. 2014. Importance Sampling Microfacet-Based BSDFs using the Distribution of Visible Normals. *Computer Graphics Forum* 33, 4 (2014), 103–112.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2022. Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 9 (2022), 5149–5169.

Bingyang Hu, Jie Guo, Yanjun Chen, Mengtian Li, and Yanwen Guo. 2020. DeepBRDF: A Deep Representation for Manipulating Measured BRDF. *Computer Graphics Forum* 39, 2 (2020), 157–166.

Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. 2019. Meta-SR: A Magnification-Arbitrary Network for Super-Resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Intel. 2023. Intel Advanced Vector Extensions 512. https://www.intel.com/content/www/us/en/architecture-and-technology/avx-512-overview.html.

Wenzel Jakob. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

Wenzel Jakob, Eugene d'Eon, Otto Jakob, and Steve Marschner. 2014. A Comprehensive Framework for Rendering Layered Materials. *ACM Trans. Graph.* 33, 4, Article 118 (jul 2014), 14 pages.

Nima Khademi Kalantari and Ravi Ramamoorthi. 2017. Deep High Dynamic Range Imaging of Dynamic Scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4 (2017).

Alexandr Kuznetsov, Krishna Mullia, Zexiang Xu, Miloš Hašan, and Ravi Ramamoorthi. 2021. NeuMIP: Multi-Resolution Neural Materials. *Transactions on Graphics (Proceedings of SIGGRAPH)* 40, 4, Article 175 (July 2021), 13 pages.

Alexandr Kuznetsov, Xuezheng Wang, Krishna Mullia, Fujun Luan, Zexiang Xu, Miloš Hašan, and Ravi Ramamoorthi. 2022. Rendering Neural Materials on Curved Surfaces. *SIGGRAPH '22 Conference Proceedings* (2022), 9 pages. https://doi.org/10.1145/3528233.3530721

Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. 2016. Deeper Depth Prediction with Fully Convolutional Residual Networks. In *2016 Fourth International Conference on 3D Vision (3DV)*. 239–248.

Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. 2017. Modeling Surface Appearance from a Single Photograph Using Self-Augmented Convolutional Neural Networks. *ACM Trans. Graph.* 36, 4, Article 45 (July 2017), 11 pages.

Zhengqin Li, Kalyan Sunkavalli, and Manmohan Chandraker. 2018. Materials for Masses: SVBRDF Acquisition with a Single Mobile Phone Image. In *Computer Vision – ECCV 2018*. Springer International Publishing, Cham, 74–90.

Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. 2003. A Data-Driven Reflectance Model. *ACM Trans. Graph.* 22, 3 (jul 2003), 759–769.

M. Maximov, T. Ritschel, L. Leal-Taixe, and M. Fritz. 2019. Deep Appearance Maps. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Los Alamitos, CA, USA, 8728–8737.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*. Springer, 405–421.

Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. 2021. Real-Time Neural Radiance Caching for Path Tracing. *ACM Trans. Graph.* 40, 4, Article 36 (jul 2021), 16 pages.

Jan Novák, Iliyan Georgiev, Johannes Hanika, Jaroslav Křivánek, and Wojciech Jarosz. 2018. Monte Carlo Methods for Physically Based Volume Rendering. In *ACM SIGGRAPH 2018 Courses* (Vancouver, British Columbia, Canada) *(SIGGRAPH '18)*. Association for Computing Machinery, New York, NY, USA, Article 14, 1 pages.

Marios Papas, Krystle de Mesa, and Henrik Wann Jensen. 2014. A Physically-Based BSDF for Modeling the Appearance of Paper. *Computer Graphics Forum* 33, 4 (2014), 133–142.

Matt Pharr and Pat Hanrahan. 2000. Monte Carlo Evaluation of Non-Linear Scattering Equations for Subsurface Reflection. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 75–84.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. 2019. On the Spectral Bias of Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 5301–5310.

Gilles Rainer, Abhijeet Ghosh, Wenzel Jakob, and Tim Weyrich. 2020. Unified Neural Encoding of BTFs. *Computer Graphics Forum* 39, 2 (2020), 167–178.

Gilles Rainer, Wenzel Jakob, Abhijeet Ghosh, and Tim Weyrich. 2019. Neural BTF Compression and Interpolation. *Computer Graphics Forum (Proceedings of Eurographics)* 38, 2 (March 2019).

Joël Randrianandrasana, Patrick Callet, and Laurent Lucas. 2021. Transfer Matrix Based Layered Materials Rendering. *ACM Trans. Graph.* 40, 4, Article 177 (jul 2021), 16 pages.

Szymon M. Rusinkiewicz. 1998. A New Change of Variables for Efficient BRDF Representation. In *Rendering Techniques '98*, George Drettakis and Nelson Max (Eds.). Springer Vienna, Vienna, 11–22.

Jos Stam. 2001. An Illumination Model for a Skin Layer Bounded by Rough Surfaces. In *Proceedings of the 12th Eurographics Conference on Rendering* (London, UK) *(EGWR'01)*. Eurographics Association, Goslar, DEU, 39–52.

Alejandro Sztrajman, Gilles Rainer, Tobias Ritschel, and Tim Weyrich. 2021. Neural BRDF Representation and Importance Sampling. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 332–346.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *CoRR* abs/2006.10739 (2020). arXiv:2006.10739 https://arxiv.org/abs/2006.10739

Sebastian Thrun and Lorien Pratt. 1998. *Learning to Learn: Introduction and Overview.* Kluwer Academic Publishers, USA, 3–17.

Tanaboon Tongbuasirilai, Jonas Unger, Christine Guillemot, and Ehsan Miandji. 2022. A Sparse Non-Parametric BRDF Model. *ACM Trans. Graph.* (apr 2022). Just Accepted.

H.C. van de Hulst. 1980. *Multiple Light Scattering.* Academic Press.

Fu-En Wang, Yu-Hsuan Yeh, Min Sun, Wei-Chen Chiu, and Yi-Hsuan Tsai. 2020. BiFuse: Monocular 360 Depth Estimation via Bi-Projection Fusion. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 459–468.

Andrea Weidlich and Alexander Wilkie. 2007. Arbitrarily Layered Micro-Facet Surfaces. In *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia (GRAPHITE '07)*. 171–178.

Andrea Weidlich and Alexander Wilkie. 2009. Exploring the Potential of Layered BRDF Models. In *ACM SIGGRAPH ASIA 2009 Courses (SIGGRAPH ASIA '09)*. Association for Computing Machinery, New York, NY, USA, Article 7, 58 pages.

Andrea Weidlich and Alexander Wilkie. 2011. Thinking in Layers: Modeling with Layered Materials. In *SIGGRAPH Asia 2011 Courses* (Hong Kong, China) *(SA '11)*. Association for Computing Machinery, New York, NY, USA, Article 20, 43 pages.

Philippe Weier and Laurent Belcour. 2020. Rendering Layered Materials with Anisotropic Interfaces. *Journal of Computer Graphics Techniques (JCGT)* 9, 2 (20 June 2020), 37–57.

Mengqi (Mandy) Xia, Bruce Walter, Christophe Hery, and Steve Marschner. 2020. Gaussian Product Sampling for Rendering Layered Materials. *Computer Graphics Forum* 39, 1 (2020), 420–435.

Zilin Xu, Zheng Zeng, Lifan Wu, Lu Wang, and Ling-Qi Yan. 2022. Lightweight Neural Basis Functions for All-Frequency Shading. In *SIGGRAPH Asia 2022 Conference Papers (SA '22)*. Association for Computing Machinery, Article 14, 9 pages.

Tomoya Yamaguchi, Tatsuya Yatagawa, Yusuke Tokuyoshi, and Shigeo Morishima. 2019. Real-Time Rendering of Layered Materials with Anisotropic Normal Distributions. In *SIGGRAPH Asia 2019 Technical Briefs* (Brisbane, QLD, Australia) *(SA '19)*. Association for Computing Machinery, New York, NY, USA, 87–90.

Tizian Zeltner and Wenzel Jakob. 2018. The Layer Laboratory: A Calculus for Additive and Subtractive Composition of Anisotropic Surface Reflectance. *ACM Trans. Graph.* 37, 4, Article 74 (jul 2018), 14 pages.

Chuankun Zheng, Ruzhang Zheng, Rui Wang, Shuang Zhao, and Hujun Bao. 2022. A Compact Representation of Measured BRDFs Using Neural Processes. *ACM Trans. Graph.* 41, 2, Article 14 (nov 2022), 15 pages.