GUMBEL-NERF: REPRESENTING UNSEEN OBJECTS AS PART-COMPOSITIONAL NEURAL RADIANCE FIELDS

Yusuke Sekikawa[‡⋆] *Chingwei Hsu*[†⋆] *Satoshi Ikehata*[†] *Rei Kawakami*[†] *Ikuro Sato*[†‡]

[†]Tokyo Institute of Technology, Japan, [⋆]Denso IT Laboratory, Japan

ABSTRACT

We propose Gumbel-NeRF, a mixture-of-expert (MoE) neural radiance fields (NeRF) model with a hindsight expert selection mechanism for synthesizing novel views of unseen objects. Previous studies have shown that the MoE structure provides high-quality representations of a given large-scale scene consisting of many objects. However, we observe that such a MoE NeRF model often produces low-quality representations in the vicinity of experts' boundaries when applied to the task of novel view synthesis of an unseen object from one/few-shot input. We find that this deterioration is primarily caused by the foresight expert selection mechanism, which may leave an unnatural discontinuity in the object shape near the experts' boundaries. Gumbel-NeRF adopts a hindsight expert selection mechanism, which guarantees continuity in the density field even near the experts' boundaries. Experiments using the SRN cars dataset demonstrate the superiority of Gumbel-NeRF over the baselines in terms of various image quality metrics. *The code will be available upon acceptance.*

1. INTRODUCTION

Construction of 3D representations of unseen objects from 2D observations is important for various applications in robotics and autonomous driving, such as semantic mapping [1–3], obstacle avoidance [4–6] and scene understanding [6, 7]. One of the difficulties in this long-standing problem lies in capturing detailed properties of objects, including 3D-shape, texture, material, and reflectance. It becomes even more challenging due to ill-posedness, when only partial observations are available. As an illustrative example, methods of 3D representation construction for novel-view synthesis from one- or few-shot observations are still in high demand for an automotive application of 360-degree surrounding-view system that synthesizes a bird's-eye view near the ego vehicle.

Extensive studies have been conducted to construct 3D scene representations. Conventional geometric reconstruction methods [8, 9] incrementally create explicit representations based on dense observations but struggle to recover unobserved regions. Approaches based on learned discrete representations can model high-level features [10, 11], but they tend to require high resolution and high computational cost. In contrast, works that use continuous representations implicitly reconstruct objects as learnable functions, providing the potential to capture fine details [12–15].

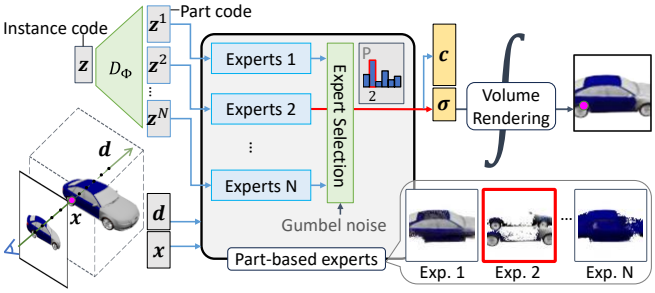


Fig. 1: Overview of **Gumbel-NeRF**. In the forward pass, a set of experts are processed to return densities and radiances. Out of N experts, only one expert with the highest density is selected. This maximum-pooling expert selection guarantees continuity in the final density field, like the original NeRF. Each expert is associated with an expert-specific latent code so that the expert learn to model a part of the object.

Recently, Neural Radiance Field (NeRF) [15] has emerged as a milestone in the realm of continuous implicit representations, primarily designed for single, small-scale scenes. While NeRF is capable of generating remarkably high-quality synthesized images, it typically requires hundreds of images for training and must be optimized separately for each scene.

To overcome the constraints, CodeNeRF [16], extended the original, scene-specific NeRF to model multiple and unseen instances of a semantic category. By conditioning on learnable instance-specific latent codes, CodeNeRF can synthesize novel views of an unseen instance even from highly limited input views. However, as CodeNeRF tries to encapsulate all of the properties into global latent codes using shared MLP, it sometimes struggles to adequately represent the variations of a semantic category when instances have diverse shapes and appearances.

To enhance the expressivity of the NeRF models, Switch-NeRF [17], employed mixture-of-experts (MoE) structure to better represent large-scale scenes. The MoE has a gate module that selects a single expert per input and the selected ex-

⋆ Equal contribution.

pert computes the density and radiance. The set of experts, the gate module, and the other shared parameters are jointly optimized for a given scene and tested on the same scene.

In this work, we tackle the problem of high-quality novel-view syntheses for unseen car instances from one/few-shot input in the aim of enriching automotive applications. A car is composed of *parts*, such as wheels, roof, doors, *etc.*, each of which is often visually similar to those in other cars. Such a commonality would be well handled by a part-based formulation. Intuitively, adopting a similar MoE structure to Switch-NeRE into CodeNeRF with part-specific latents appears to be a good solution for enhancing both expressivity and generalization. However, we observed that the naive combination of the two results in a deterioration in the reconstructed shape around experts' boundaries (see the black car in Fig. 3-CSN). We hypothesize that the primary cause of the deterioration comes from the *foresight* expert selection mechanism; *i.e.*, the gating network selects an expert *before* processing the experts. Due to this gating mechanism, the best expert may not always be chosen in terms of rendering quality, thus it often results in unnatural discontinuity near the experts' boudaries. We remark that discontinuity is an issue specific to novel view syntheses for unseen instances.

To address the above issue, we propose **Gumbel-NeRF**, a conditional MoE NeRF utilizing the *hindsight expert selection mechanism* as depicted in Fig. 1. Gumbel-NeRF replaces the gate network in Switch-NeRF with the simple maximum pooling of the densities of the experts. By this construction, Gumbel-NeRF guarantees continuity in the density field that the entire model returns, just like the original NeRF. In addition, we introduce *expert-specific codes* that represent different parts of a given car so that experts specifically learn to model the corresponding parts, whereas CodeNeRF uses a single code for a whole object. We let the model learn how to decompose into parts, instead of giving supervision about parts. Equipped with the enhanced expressivity and adaptability to test instances, Gumbel-NeRF outperforms the baselines in terms of several image quality metrics on a public benchmark of multi-instance view synthesis of cars.

2. RELATED WORK

2.1. Neural Radiance Fields

NeRF [15] has achieved impressive success in the field of 3D scene representation and novel view synthesis. Using only multi-view supervision, NeRF models the volume density σ and the emitted radiance $\mathbf{c}$ of a scene by a continuous function F_Θ of 3D spatial coordinates $\mathbf{x}$ and 2D viewing direction $\mathbf{d}$ as

$$F_\Theta : (\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma). \tag{1}$$

This function is commonly approximated by MLPs. The constructed radiance fields can then be rendered into pixel values by volume rendering [18] given by

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N_p} T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i, \tag{2}$$

where N_p is the number of sampled points along ray $\mathbf{r}$, δ is the distance between two adjacent sample points, and

$$T_n = \exp\left(- \sum_{m=1}^{n-1} \sigma_m \delta_m \right) \tag{3}$$

can be interpreted as the accumulated transmittance up to sample n. Comparing to discrete, explicit representations that are usually limited by resolution, the implicit nature of NeRF allows synthesizing photorealistic images from arbitrary viewpoints. Despite the success of NeRF, one obvious drawback is that NeRF cannot handle test scenes with limited observations, since it was originally designed to model a single scene.

2.2. Conditional NeRF

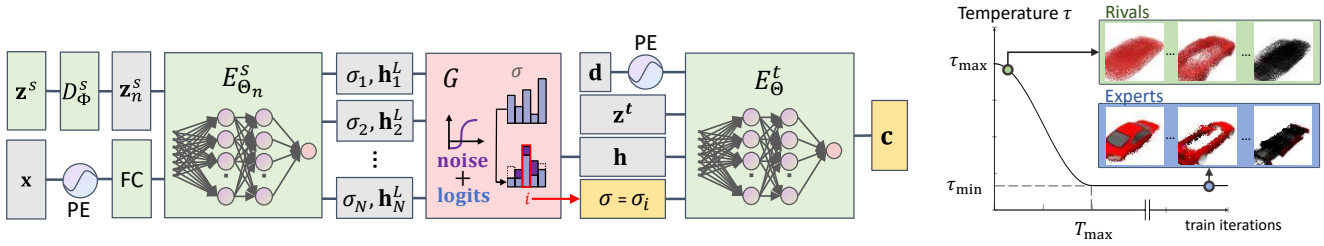
A family of work [16,19–23] extends the capabilities of NeRF to deal with multiple objects by conditioning the field representation with a set of tunable latent variables $\mathbf{z}$ encoded with some prior knowledge. Adjusting $\mathbf{z}$, a field representation can be controlled to represent different objects or scenes. A conditional NeRF can be generally formulated as:

$$F_\Theta(\mathbf{x}, \mathbf{d}, \mathbf{z}) \mapsto (\mathbf{c}, \sigma). \tag{4}$$

The choice of encoding method and conditioning schemes varies across different works. CodeNeRF [16] jointly optimizes a set of instance-specific latent codes along with Θ. During test time, given one or several images of a novel instance, CodeNeRF reconstructs its shape and synthesizes novel views by optimizing the instance-specific latent code. In contrast, PixelNeRF [19] directly obtains latents by extracting local features from input views with a CNN encoder, allowing conditioning on the rendering of the target view. Subsequent works [20] proposed different aggregation method of extracted local features. AutoRF [21] combines the characteristics of both approaches by employing an autoencoder to encode instance-specific latent codes. This enables image synthesis without test-time optimization, while also allowing refinement of image quality through such a process. Nevertheless, stronger supervision requirements including 3D bounding box and panoptic masks are needed.

2.3. Decomposed NeRF

Another family of work [17, 24, 25] decomposes NeRF into multiple sub-models to enhance the network expressivity. The optimization process of a NeRF model involves memorizing the entire scene, and as the scene size scales up, the synthesized image quality tends to deteriorate. Dividing the scene



(a) (b)

Fig. 2: (a) Architecture of Gumbel-NeRF. Trainable parameters, output vectors, and our proposed expert selection layer are shown in green, yellow, and pink boxes, respectively. FC refers to fully connected layer and PE refers to positional encoding. $\mathbf{z}_n^s$ denotes the expert-specific shape code for the n -th expert. Different from Switch-NeRF [17], Gumbel-NeRF processes all experts in parallel to produce candidate densities $\sigma_{1\cdots N}$ and intermediate features $\mathbf{h}_{1\cdots N}^L$. The layer G samples the output from only one expert using the Gumbel-Max trick. (b) Scheduling of the temperature parameter. The temperature parameter used in the Gumbel-Max trick is scheduled to control the level of randomness through the training process. In the early stage, the temperature is set high so that all experts have a nearly equal chance of being selected (rival stage). In this stage, each expert obtains sufficient gradient updates, avoiding collapse (*i.e.*, a vicious cycle where only one expert obtains all the gradient updates and other experts are underoptimized). Toward the end, the temperature is decreased to make experts distinct (expert stage).

into distinct regions and assigning a neural network to each region can effectively address this problem.

To achieve this decomposition, different approaches have been proposed. Block-NeRF [24] design hand-carft decomposition methods based on distance and street blocks, respectively. Another approach, Switch-NeRF [17], combines NeRF with a Sparsely-Gated Mixture of Experts (MoE) [26] framework, where the dispatchment of inputs to the experts (NeRFs) is determined by a jointly-optimized gating network. This approach improves the overall performance by reducing inconsistencies among sub-models and better balancing the complexity of the scene.

Decomposing NeRF also offers the benefit of enabling scene editing capabilities. GIRAFFE [22] uses multiple generative NeRFs to model a scene with multiple objects, which allows for explicit control over individual objects in a scene, offering a disentangled manner of manipulation. PartNeRF [25], decomposes an object category into several semantic parts. Each part is handled by a separate NeRF network, which is trained in its own local coordinates. They employ complicated loss terms to ensure reasonable division of objects. We note that shape editing is not our goal.

3. METHOD

As shown in Fig. 1, our proposed method follows the basic formulation of conditional NeRFs given in Eq. 4, which targets in modeling multiple instances within the same semantic category. Following CodeNeRF [16], our method jointly optimizes the neural radiance field F_Θ, the latent code mapper D_Φ, and a set of instance-specific latent codes $\{\mathbf{z}_m = (\mathbf{z}_m^s, \mathbf{z}_m^t)\}_{m=1}^M$, where M denotes the number of instances in the training set and $(\mathbf{z}_m^s, \mathbf{z}_m^t)$ represent the shape and texture latent codes, respectively. At test time, we optimize the instance-specific latent codes with frozen F_Θ and D_Φ.

Fig. 2a illustrates the architecture of our proposed approach, which consists of two key elements: (i) a set of N experts, each of which couples to corresponding expert-specific latent codes, and (ii) an expert selection mechanism that selects a single expert for a given input. To effectively train experts, we introduce (iii) a rival-to-expert training strategy that controls the level of randomness through the training so that different experts grow to model different parts of objects (see Fig. 2b). These elements are described in detail below.

3.1. Part-Specific Experts

Similarly to Switch-NeRF [17] and PartNeRF [25], the neural implicit representation F_Θ consists of a set of N submodels $\{E_{\Theta_n}\}_{n=1}^N$, which we refer to as experts. In addition to the original 5D inputs (location $\mathbf{x}$ and viewing direction $\mathbf{d}$), each expert E_{Θ_n} is exclusively conditioned on part-specific latent codes $\mathbf{z}_{m,n} = (\mathbf{z}_{m,n}^s, \mathbf{z}_{m,n}^t)$, associated to the m-th instance and to n-th expert. Here, N is a predefined constant.

The experts' design is adopted from that of the vanilla NeRF [15], such that E_{Θ_n} is composed of two parts: a shape MLP $E_{\Theta_n}^s$ and a texture head $E_{\Theta_n}^t$. The density σ depends only on $\mathbf{x}$, while the radiance $\mathbf{c}$ depends on both $\mathbf{x}$ and $\mathbf{d}$. Following Switch-NeRF, the texture code and the texture head for predicting the final outputs are designed to be shared across experts; namely, $\mathbf{z}_m^t = \mathbf{z}_{m,n}^t, E_\Theta^t = E_{\Theta_n}^t, n = 1, \cdots, N$.

To obtain the expert-specific latent codes, we utilize a latent code mapper D_Φ that includes separate mappings for shape D_Φ^s and texture D_Φ^t. The shape mapping consists of N

linear functions that map the instance-specific shape codes $\mathbf{z}^s_m$ to N expert-specific shape codes $\{\mathbf{z}^s_{m,n}\}_{n=1}^N$ associated to corresponding experts. As for the texture codes, since a unified RGB head is used, we let the texture mapping D^t_Φ be simply an identity function. Formally, the latent code mappers are given as:

$$D^s_\Phi : \mathbf{z}^s_m \mapsto \{\mathbf{z}^s_{m,n}\}_{n=1}^N, \tag{5}$$

$$D^t_\Phi : \mathbf{z}^t_m \mapsto \mathbf{z}^t_m, \qquad m = 1,\dots,M. \tag{6}$$

The expert parameters and the corresponding code are optimized in a co-adaptive manner so that the expert-code pair well represents a specific part of the target object. We will empirically show that experts can learn to decompose objects into similar parts without explicit supervision of parts. We omit the subscript m for simplicity in the rest of the paper.

3.2. Density-Based Expert Selection

Gumbel-NeRF adopts the hindsight-based density-based expert selection rule, as opposed to the foresight gating network adopted in Switch-NeRF [17]. The use of a gating network to select an expert before processing experts might seem reasonable for our task setting; however, we have observed that the foresight gating mechanism often degrades the performance of the model, as is also evident in other literature [27]. The MoE design in Switch-NeRF prevents the gating network from sharing expert information, resulting in a solely location-based expert selection mechanism. The design is well-suited for a single, large-scale scene. This is because the complexity of a single scene is fixed so that Switch-NeRF can efficiently decompose the scene into regions, each of which can then be treated by vanilla NeRF reconstructions. However, in our task setting, we argue that the capacity of experts may be better leveraged when they specialize in handling regions with similar shape and appearance properties, not locations, across many instances. Moreover, the MoE design in the Switch-NeRF does not guarantee continuity in the predicted density field, as opposed to the original NeRF. Especially for the task of novel view syntheses of unseen instances, we observed an unnatural discontinuity in the predicted density field, resulting in severe deterioration of synthesizing quality (see the black car in Fig. 3-CSN).

To overcome the above-mentioned issues, we devised to place the expert selection *after* the expert blocks. With this construction, the expert selection relies on as much information provided by the experts as possible. In our proposed method, the given inputs pass through all the shape MLPs, and only one of the outputs contributes to the final neural field. This process is formulated as:

$$E^s_{\Theta_n} : (\mathbf{x}, \mathbf{z}^s_n) \mapsto (\mathbf{h}^L_n, \sigma_n),\ n = 1,\dots,N \tag{7}$$

$$G : \{(\sigma_n, \mathbf{h}^L_n)\}_{n=1}^N \mapsto (\sigma, \mathbf{h}), \tag{8}$$

where $\mathbf{h}$ are intermediate features. The expert selector G basically adopts the maximum pooling of the densities returned by the experts; that is, an expert having the highest density is only used and the other experts are ignored. With this construction, the model guarantees continuity in the predicted density field. To address router collapse, in which a particular expert is always chosen, we employ the Gumbel-Max trick [28]. Namely, Gumbel noises are added to the output densities to control the chance rate for the n-th expert to be selected. We first treat $\{\sigma_n\}_{n=1}^N$ as the unnormalized "probabilities" and calculate the normalized log probabilities using a LogSoftmax function:

$$\mathbf{logits}_n = \log\left(\frac{\exp(\frac{\log\sigma_n}{\tau})}{\sum_{i=1}^N \exp(\frac{\log\sigma_i}{\tau})}\right),\ n = 1,\dots,N, \tag{9}$$

where τ is the temperature parameter to control the level of randomness, which will be further discussed in Sec. 3.3. Then, the output is selected by:

$$\sigma = \mathbb{1}_{\max}(\mathbf{logits} + \mathbf{g}) \cdot (\sigma_1, \sigma_2, \dots, \sigma_N), \tag{10}$$

$$\mathbf{h}^L = \mathbb{1}_{\max}(\mathbf{logits} + \mathbf{g}) \cdot (\mathbf{h}^L_1, \mathbf{h}^L_2, \dots, \mathbf{h}^L_N), \tag{11}$$

where

$$\begin{aligned}\mathbb{1}_{\max}(\mathbf{y}) &= \delta^\top_{n,\underset{i}{\operatorname{argmax}}(\mathbf{y}_i)} \\ &= \begin{cases} 1 & \text{if } n = i \\ 0 & \text{otherwise} \end{cases},\ n = 1,\dots,N\end{aligned} \tag{12}$$

is a one-hot vector having 1 in the index corresponding to the maximum value of a vector $\mathbf{y}$, and $\mathbf{g}$ is a vector of N i.i.d. Gumbel noise samples drawn from the standard Gumbel distribution using the inverse transform sampling technique:

$$g_j \sim -\log(-\log(\text{Uniform}(0,1))),\ j = 1,\dots,N. \tag{13}$$

3.3. Rival-to-Expert Training Strategy

The temperature parameter $\tau > 0$ in Eq. 9 plays a crucial role in determining the level of randomness of G. When τ increases, the Softmax approaches the uniform distribution, resulting in nearly random sampling. In contrast, a smaller τ encourages a more consistent distribution, where logits maintain their original ranking and are less affected by the added Gumbel noise.

During training, we schedule τ using a cosine annealing followed by a constant final value as

$$\tau(t) = \begin{cases} \tau_{\min} + \frac{\tau_{\max}-\tau_{\min}}{2}\left(1 + \cos\frac{\pi t}{T_{\max}}\right) & \text{if } t \le T_{\max} \\ \tau_{\min} & \text{otherwise,} \end{cases} \tag{14}$$

where $\tau_{\max}, \tau_{\min}, T_{\max}$ are the initial temperature, final temperature, and the duration of cosine annealing in terms of the percentage of the total training iterations, respectively (See Fig. 2b). By this scheduling, during the earlier stage, a higher level of randomness is introduced in selecting the experts. We

refer to this stage as the *rival stage*, where all experts have a similar chance of being selected and acquiring gradient updates. In this stage, the experts act as rivals, collectively modeling the overall scene in a coarse manner. As the temperature approaches $\tau_{\min}$, the training progresses to the *expert stage*. During this stage, the experts become "real experts," as the selection process reaches a more stable state. This stability enables them to focus exclusively on the regions of interest and continuously refine the quality of reconstruction throughout the rest of the training process. This rival-to-expert training scheme effectively prevents the router collapse problem and training instabilities without the use of additional loss terms. The entire model is trained end-to-end by minimizing the photometric loss:

$$\min_{\Theta,\Phi,\{\mathbf{z}_m\}_{m=1}^{M}} \sum_{\mathbf{r}\in\mathscr{R}} \|\hat{C}(\mathbf{r}) - C(\mathbf{r})\|^2, \tag{15}$$

where $\hat{C}$, C and $\mathscr{R}$ are the rendered pixel value, the ground truth pixel value and the set of rays in a training batch, respectively.

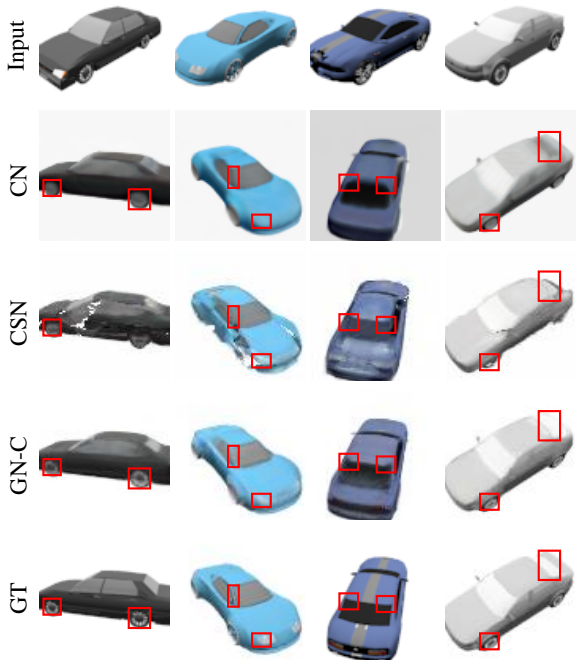


Fig. 3: Qualitative results of novel view synthesis of unseen objects using one-shot test-time optimization. Compared to CodeNeRF (CN) and Coded Switch-NeRF (CSN), our Gumbel-NeRF (GN-C) generally produces higher quality, especially for those parts marked by red boxes.

4. EVALUATION

4.1. Datasets

We conduct experiments on a re-rendered version of the "car" category of ShapeNet [29] dataset provided by SRN [14]. ShapeNet contains 3,514 instances of cars (2,458 for training, 704 for testing, and 352 for validation). Each training instance accompanies 50 images rendered from random viewpoints, while each testing instance accompanies 251 images rendered from the same set of predefined camera poses on an Archimedean spiral. All images contain a single foreground car against a white background.

4.2. Baselines

We compare our method, Gumbel-NeRF, with CodeNeRF [16], which is also capable of handling multi-instance datasets with only 2D supervision. To show the effectiveness of our gate-free, density-based expert selection design, we also compare it with "*Coded Switch-NeRF*", a naive extension of Switch-NeRF [17] where their gating network and experts are conditioned on latent codes. The components of Coded Switch-NeRF can be written as:

$$G : \left(\mathbf{x}, \mathbf{z}^s; \{\mathbf{z}_n^s, F_{\Theta_n}\}_{n=1}^{N}\right) \mapsto \left(\mathbf{z}_{n^*}^s, F_{\Theta_{n^*}}^s\right) \tag{16}$$

$$F_{\Theta_{n^*}}^s : (\mathbf{x}, \mathbf{z}_{n^*}^s) \mapsto \mathbf{h}\,. \tag{17}$$

4.3. Evaluation Metrics

We report the results with standard metrics for evaluating image quality: PSNR, SSIM [30] and the VGG, AlexNet and SqueezeNet versions of LPIPS [31].

4.4. Implementation Details

We implement Gumbel-NeRF and Coded Switch-NeRF with $N = 4$ experts based on the released code of Switch-NeRF [17]. We modify CodeNeRF's MLPs as the Gumbel-NeRF's expert architecture when comparing with CodeNeRF, but use half of its channels per layer to keep the number of parameters the same. Compared with Coded Switch-NeRF, the architecture of experts follows that of Switch-NeRF. We use 256-dimensional instance-specific codes and 128/256-dimensional part-specific codes for CodeNeRF-style and Switch-NeRF-style experts, respectively.

We follow the training and test-time optimization setups of CodeNeRF [16], with carefully selected hyper-parameters to provide a fair comparison. Specifically, we train the models on 128 NVIDIA P100 GPUs with a total batch size of 327,680 points for 25k iterations using centered-cropped images and another 20k iterations using uncropped images. For test-time optimization, we iteratively refine only the instance-specific latent codes for 200 iterations.

Fig. 4: Visualization of the decomposition provided by Coded Switch-NeRF (CSN) and Gumbel-NeRF (GN). Images in each column are rendered from only the 3D points handled by the corresponding expert.

Table 1: Quantitative evaluation on ShapeNet-SRN cars test set. Note that we clip the rendered values to 0-1.

Method	# of params	PSNR↑	SSIM↑	LPIPS↓		
				VGG	Alex	Squeeze
CN	0.7M	19.66	0.882	0.150	0.161	0.119
GN-C	0.8M	**21.51**	**0.892**	**0.119**	**0.138**	**0.104**
CSN	4.1M	19.50	0.864	0.145	0.160	0.110
GN-S	3.9M	**21.43**	**0.890**	**0.114**	**0.126**	**0.088**

During training, the learning rate starts with an initial value of 1.3×10^{-3} and exponentially decays with a decay factor of 0.1 for cropped images, then remains constant for uncropped images. During test-time optimization, the same exponential decay scheduler is used, but with an initial learning rate of 2.0×10^{-2}. The models are trained with AdamW optimizer. The parameters of Gumbel-NeRF cosine annealing are set as $\tau_{\max} = 10, \tau_{\min} = 0.5$, and $T_{\max} = 20\%$ for training and as constant $\tau_{\min}$ for testing time optimization.

5. RESULTS

Reconstruction of Unseen Objects. We perform an evaluation on the ShapeNet-SRN car test set, following the SRN [14] and CodeNeRF [16], which calculates the average values of each metric using images except those used as test-time optimization input. We conducted a one-shot optimization using only one input image from the same viewpoints across instances. We show the quantitative and qualitative results of CodeNeRF (**CN**) [16], Coded Switch-NeRF (**CSN**), and Gumbel-NeRF (**GS-C** and **GS-S**, where "-C" and "-S" refers to CodeNeRF-style and Switch-NeRF-style expert architecture, respectively) in Table 1 and Fig. 3. As can be seen, with a similar number of parameters, Gumbel-NeRF outperforms Code-NeRF in all metrics. Our proposed method also outperforms Coded Switch-NeRF, showing the effectiveness of our gate-free density-based expert selection design despite the trade-off in processing time. We can visually inspect that our model has no artifacts in the experts' boundary, while Coded Switch-NeRF has an unnatural discontinuity in shape.

Part Decomposition. We visualize the decomposition of objects in Fig. 4 using images rendered from the 3D points handled by each expert. As can be seen, the Gumbel-NeRF is capable of learning a more consistent decomposition across objects compared to Coded Switch-NeRF. For example, expert 4 of our Gumbel-NeRF generates 4 wheels for the left and right instances, while expert 4 of Coded Switch-NeRF generates almost nothing for the left instance and generates some lower components for the right instance. The inconsistent decomposition by Coded Switch-NeRF is due to the expert selection mechanism, in which the expert selection is done before "seeing" the expert's quality. Furthermore, Gumbel-NeRF demonstrates a better ability to prevent the router collapse problem, as it evenly utilizes all experts in the training phase.

6. CONCLUSION

We proposed Gumbel-NeRF, a conditional neural radiance field capable of constructing the 3D representations of unseen car instances from one/few 2D observations. Our proposed method leverages the hindsight expert selection mechanism, which guarantees the continuous transition in the predicted density field, successfully increasing the expressibility of latent code-based NeRF. We also propose a novel rival-to-expert training strategy in order to balance the utilization of experts. Through experiments on the ShapeNet-SRN cars dataset, we demonstrate that our method outperforms CodeNeRF and Coded Switch-NeRF in terms of several image quality metrics, proving its superior adaptability in capturing the details of unseen instances.

Acknowledgement. This work is supported by JSPS KAKENHI JP22H03642 and DENSO IT LAB Recognition and Learning Algorithm Collaborative Research Chair.

7. REFERENCES

[1] Mehdi Hosseinzadeh, Kejie Li, Yasir Latif, and Ian Reid, "Real-time monocular object-model aware sparse slam," in *ICRA*. IEEE, 2019, pp. 7123–7129.

[2] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart, and Juan Nieto, "Volumetric instance-aware semantic mapping and 3d object discovery," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3037–3044, 2019.

[3] Edgar Sucar, Kentaro Wada, and Andrew Davison, "Nodeslam: Neural object descriptors for multi-view shape reconstruction," in *3DV*. IEEE, 2020, pp. 949–958.

[4] Zhenggang Tang, Balakumar Sundaralingam, Jonathan Tremblay, Bowen Wen, Ye Yuan, Stephen Tyree, Charles Loop, Alexander Schwing, and Stan Birchfield, "Rgb-only reconstruction of tabletop scenes for collision-free manipulator control," in *ICRA*. IEEE, 2023, pp. 1778–1785.

[5] Kyel Ok, Katherine Liu, and Nicholas Roy, "Hierarchical object map estimation for efficient and robust navigation," in *ICRA*. IEEE, 2021, pp. 1132–1139.

[6] Yunzhi Lin, Jonathan Tremblay, Stephen Tyree, Patricio A Vela, and Stan Birchfield, "Multi-view fusion for multi-level robotic scene understanding," in *IROS*. IEEE, 2021.

[7] J Krishna Murthy, GV Sai Krishna, Falak Chhaya, and K Madhava Krishna, "Reconstructing vehicles from a single image: Shape priors for road scene understanding," in *ICRA*. IEEE, 2017, pp. 724–731.

[8] Johannes Lutz Schönberger and Jan-Michael Frahm, "Structure-from-motion revisited," in *CVPR*, 2016.

[9] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison, "Dtam: Dense tracking and mapping in real-time," in *ICCV*. IEEE, 2011, pp. 2320–2327.

[10] Xinchen Yan, Jasmined Hsu, Mohammad Khansari, Yunfei Bai, Arkanath Pathak, Abhinav Gupta, James Davidson, and Honglak Lee, "Learning 6-dof grasping interaction via deep geometry-aware 3d representations," in *ICRA*. IEEE, 2018.

[11] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh, "Neural volumes: Learning dynamic renderable volumes from images," *arXiv preprint arXiv:1906.07751*, 2019.

[12] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *CVPR*, 2019.

[13] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," in *ICCV*, 2019, pp. 2304–2314.

[14] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein, "Scene representation networks: Continuous 3d-structure-aware neural scene representations," *NeurIPS*, vol. 32, 2019.

[15] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Comm. of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[16] Wonbong Jang and Lourdes Agapito, "Codenerf: Disentangled neural radiance fields for object categories," in *ICCV*, 2021.

[17] MI Zhenxing and Dan Xu, "Switch-nerf: Learning scene decomposition with mixture of experts for large-scale neural radiance fields," in *ICLR*, 2022.

[18] James T Kajiya and Brian P Von Herzen, "Ray tracing volume densities," *SIGGRAPH*, vol. 18, no. 3, pp. 165–174, 1984.

[19] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa, "pixelnerf: Neural radiance fields from one or few images," in *CVPR*, 2021, pp. 4578–4587.

[20] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny, "Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction," in *ICCV*, 2021.

[21] Norman Müller, Andrea Simonelli, Lorenzo Porzi, Samuel Rota Bulo, Matthias Nießner, and Peter Kontschieder, "Autorf: Learning 3d object radiance fields from single view observations," in *CVPR*, 2022, pp. 3971–3980.

[22] Michael Niemeyer and Andreas Geiger, "Giraffe: Representing scenes as compositional generative neural feature fields," in *CVPR*, 2021, pp. 11453–11464.

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al., "Learning transferable visual models from natural language supervision," in *ICML*. PMLR, 2021, pp. 8748–8763.

[24] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar, "Block-nerf: Scalable large scene neural view synthesis," in *CVPR*, 2022, pp. 8248–8258.

[25] Konstantinos Tertikas, Pascalidou Despoina, Boxiao Pan, Jeong Joon Park, Mikaela Angelina Uy, Ioannis Emiris, Yannis Avrithis, and Leonidas Guibas, "Partnerf: Generating part-aware editable 3d shapes without 3d supervision," *arXiv preprint arXiv:2303.09554*, 2023.

[26] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[27] Amelie Royer, Ilia Karmanov, Andrii Skliar, Babak Ehteshami Bejnordi, and Tijmen Blankevoort, "Revisiting single-gated mixtures of experts," *arXiv preprint arXiv:2304.05497*, 2023.

[28] Emil Julius Gumbel, *Statistical theory of extreme values and some practical applications: a series of lectures*, 1948.

[29] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al., "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[30] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE TIP*, vol. 13, no. 4, pp. 600–612, 2004.

[31] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018, pp. 586–595.