

A TRAINING DETAILS

In this section, we provide the training details of our models.

A.1 TRAINING HYPERPARAMETERS

Table 6: Training hyperparameters for the models in Table 1. Step refers to a learning rate schedule where the initial learning rate is divided by 10 after 80% of the epochs have finished, and cos refers to a cosine learning rate schedule with a warmup of 10% of the total epochs.

Model	Epochs	Batch size	Learning rate	Scheduler
Randomly initialized	1000	32	1e-4	step
MAE	1000	128	1e-4	step
Supervised	1000	32	1e-4	step
DeiT	1000	128	1e-4	step
DINO	1000	128	1e-4	cos
CLIP	1000	128	1e-4	cos
DINOv2	1000	128	1e-4	cos

The training hyperparameters for the models in Table 1 can be found in Table 6. Models in Table 2 and Figure 2 were trained with 1000 epochs, batch size 128, learning rate 1e-4, and using the cos scheduler described above. The prompt-tuned model in Table 3 was trained with batch size 32, 1000 epochs, learning rate 1e-3, and the step scheduler; the hyperparameters for the other two can be found in Table 6. The IPC and PONP baselines in Table 4 were trained for 1000 epochs with batch size 128, learning rate 1e-4, and 1000 epochs. Additionally, PONP used the cos learning rate scheduler. The methods in Table 5 were trained for 100 epochs with batch size 64, but all other hyperparameters were the default hyperparameters from Kim et al. (2023).

A.2 INR ARCHITECTURE

Following previous work (Chen & Wang, 2022; Kim et al., 2023; Gu et al., 2023; Lee et al., 2024), our INR architecture is an MLP with 6 layers of hidden dimension 256, positional encoding with dimension 40, and ReLU activations.

B METRICS

In this section, we discuss the metrics used in our paper. Our main task of novel view synthesis from a single view of an object is a task where both image similarity metrics (such as PSNR, SSIM, LPIPS) and image generation metrics (FID) can provide complementary assessments of novel view quality. This is because the generated view may be partially determined by shared structures present in both views, while the other parts are under-determined and need to be generated. Besides PSNR, all other metrics were implemented using the `torchmetrics` library with their default parameters.

PSNR PSNR stands for peak signal-to-noise ratio, and is computed with the formula

$$\text{PSNR}(y, \hat{y}) = -10 \log_{10}(\text{MSE}(y, \hat{y})) \quad (3)$$

where MSE is the mean squared error. PSNR is a measure of the absolute error between a reconstruction \hat{y} and the ground truth y , which makes it less reliable for under- constrained reconstruction tasks such as novel view synthesis from one view of an object, where there may be many possible plausible reconstruction.

SSIM Structural similarity index (SSIM) (Wang et al., 2004) computes the similarity of two images in luminance, contrast, and structure. SSIM is designed to measure the perceived change in structural information rather than the absolute change measured by PSNR. Wang et al. (2004) shows that SSIM better correlates with human ratings than PSNR.

LPIPS LPIPS (Zhang et al., 2018b) measures the similarity between the activations of images computed by a pre-defined neural network. Zhang et al. (2018b) shows that deep similarities given by pre-trained neural networks correlate much better with human judgments than PSNR or SSIM.

FID FID (Heusel et al., 2017) measures the how similar the distribution of generated images is to the distribution of the ground truth images, and is more suited for generative tasks than tasks where there is a defined ground truth. However, it has drawbacks, as discussed in the main text as well as Jayasumana et al. (2024).

C COMPARISON TO PREVIOUS RESULTS

Table 7: Comparison of the results in Table I to previously published results. * indicates that the result was obtained from previous literature by averaging the performance of separate models for the three different classes. Previous results are shown in the first half of the table, while our results are shown in the second half of the table.

Model	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)	FID (\downarrow)
LearnIt (Tancik et al., 2020)	21.33*	-	-	-
Trans-ISR (Chen & Wang, 2022)	22.07*	-	-	-
Trans-ISR (repr. by Kim et al., (2023))	22.04*	-	-	-
PONP (Gu et al., 2023)	22.14*	-	-	-
IPC (Kim et al., 2023)	22.30*	-	-	-
Randomly initialized	20.862	0.8357	0.1511	0.2751
MAE (He et al., 2022)	20.701	0.8312	0.1753	0.2866
Supervised (Dosovitskiy et al., 2020)	21.324	0.8501	0.0966	0.1516
DeiT (Touvron et al., 2021)	21.587	0.8530	0.1125	0.1860
DINO (Caron et al., 2021)	21.737	0.8555	0.1101	0.1810
CLIP (Radford et al., 2021)	21.770	0.8556	0.1126	0.1834
DINOv2 (Oquab et al., 2023)	22.095	0.8609	0.1063	0.1705

In Table 7, we compare our results for single-view novel view synthesis (Tab. I) on the LearnIt ShapeNet dataset (Tancik et al., 2021) to previously published results from LearnIt (Tancik et al., 2021), Trans-ISR (Chen & Wang, 2022), PONP (Gu et al., 2023), and IPC (Kim et al., 2023) which all use the same ISR architecture. We note that these numbers are not directly comparable, as our numbers are obtained on the harder task of learning all three categories simultaneously and without being able to tokenize NVS-specific auxiliary information such as poses. Compared to previous Transformer-based methods (Chen & Wang, 2022; Gu et al., 2023; Kim et al., 2023), our method uses a ViT/B-16 while previous methods use a smaller 6 layer Transformer architecture. We also note that the performance of the Transformer hypernetwork baselines Chen & Wang (2022); Gu et al. (2023); Kim et al. (2023) is significantly degraded in the combined class setting, especially IPC (see Tab. 4).

D LIMITATIONS

One limitation of our method is we do not tokenize task-specific information such as pose and camera parameters for novel view synthesis. Previous results suggest that this may further improve performance. Another limitation is that we have only used the simple volume renderer and simple NeRF (Mildenhall et al., 2021) of Tancik et al. (2020), but better results could be obtained by using a more sophisticated volume renderer and ISR. Another limitation is that we only investigate fine-tuning and freezing the foundation model backbone, but other approaches may perform better. We also were not able to investigate using larger datasets such as Objaverse (Deitke et al., 2023).

E PARAMETER-EFFICIENT FINE-TUNING

In this section, we make a preliminary investigation of parameter-efficient fine-tuning (PEFT) methods as an alternative to full fine-tuning and freezing. The intuition behind using PEFT is to avoid potential

Table 8: Comparison of the four different training strategies, including LoRA (Hu et al., 2022), using pre-trained DINO (Caron et al., 2021) on the NVS task. We find that LoRA models outperform prompt-tuned (frozen encoder) models in all metrics with only 2M more parameters, while performing second-best overall with only 2.4% of the parameters of a fully fine-tuned model.

Method	Trainable Parameters	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)	FID (\downarrow)
Randomly initialized	87.3M (100%)	20.862	0.8357	0.1511	0.2751
Frozen	100K (0.11%)	21.035	0.8335	0.1767	0.3212
LoRA (Hu et al., 2022)	2.1M (2.41%)	21.246	0.8397	0.1678	0.3052
Fine-tuned	87.3M (100%)	21.737	0.8555	0.1101	0.1810

Table 9: Comparison of hypernetwork generalizability to classes unseen during training using random initialization, fine-tuning from DINOv2 (Oquab et al., 2023), prompt tuning with frozen DINOv2, and LoRA (Hu et al., 2022). Each method was trained with only two of the classes in the ShapeNet NVS dataset and evaluated on the third, unseen class. The best metrics are highlighted in **bold**. In the last section, the average over all settings is reported for each of the methods.

Method	Training \rightarrow Test	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)	FID (\downarrow)
Randomly initialized	cars, chairs \rightarrow lamps	17.377	0.7959	0.1898	0.1179
Frozen	cars, chairs \rightarrow lamps	18.346	0.7972	0.2941	0.3033
Fine-tuned	cars, chairs \rightarrow lamps	17.474	0.7977	0.1903	0.0956
LoRA (Hu et al., 2022)	cars, chairs \rightarrow lamps	18.184	0.8038	0.2437	0.2248
Randomly initialized	cars, lamps \rightarrow chairs	13.163	0.6212	0.3751	1.0199
Frozen	cars, lamps \rightarrow chairs	13.536	0.6112	0.4279	1.0017
Fine-tuned	cars, lamps \rightarrow chairs	13.322	0.6238	0.3845	0.9680
LoRA (Hu et al., 2022)	cars, lamps \rightarrow chairs	13.077	0.6206	0.3848	0.9775
Randomly initialized	chairs, lamps \rightarrow cars	15.431	0.7521	0.2987	0.3276
Frozen	chairs, lamps \rightarrow cars	15.503	0.7465	0.3607	0.3623
Fine-tuned	chairs, lamps \rightarrow cars	15.382	0.7692	0.2310	0.1548
LoRA (Hu et al., 2022)	chairs, lamps \rightarrow cars	16.432	0.7704	0.3094	0.3362
Randomly initialized	Average	15.324	0.7231	0.2879	0.4885
Frozen	Average	15.795	0.7183	0.3609	0.5558
Fine-tuned	Average	15.393	0.7302	0.2686	0.4601
LoRA (Hu et al., 2022)	Average	15.898	0.7316	0.3126	0.5128

catastrophic forgetting, as hypothesized in Section 3.5. To do this, we perform parameter-efficient fine-tuning using low-rank adaptation (LoRA) (Hu et al., 2022).

As shown in Figure 8, LoRA outperforms freezing the pre-trained encoder in all metrics while not using many more parameters (2.1M vs 0.1M parameters, respectively). LoRA also performs second-best overall, while only having 2.4% of the parameters of the best model, the model trained with full fine-tuning, and outperforming the model trained from a random initialization.

In the generalization setting (Table 9), we find that on average, LoRA performs the best in PSNR and SSIM, while full fine-tuning performs the best in LPIPS and FID. The overall performance of LoRA seems to suggest that LoRA may be able to mitigate potential catastrophic forgetting. We also find that, as in the previous section, LoRA models outperform the frozen encoder models in all metrics. We also find that models which update all the parameters perform clearly better in LPIPS and FID, and that this is a general trend. Further analysis is needed to determine the cause for this.