Supplementary: Pre-train, Self-train, Distill: A simple recipe for Supersizing 3D Reconstruction

Kalyan Vasudev Alwala[1] Abhinav Gupta[12] Shubham Tulsiani[2]
[1]Meta AI Research [2]Carnegie Mellon University
https://shubhtuls.github.io/ss3d/

1. Additional Training Details

Input. In all phases of training, we consider zero-padded single-instance only color images as input to our network. Particularly, the input to our network is a square image of size 224 where the larger dimension of the instance bounding box is resized to 224. The images correspond to segmented objects, and the background pixel values are also set to zero.

1.1. Pre-Training from Synthetic 3D Data

We consider multi-view supervision for training training the model. For each input image in each iteration, we consider rays from 5 other views (among a total of 20 available views) and their associated cameras for supervision. Unlike input images which have a size of 224, the images used for supervision are resized to 128x128. This is to ensure that we can cover more region with a single ray for faster training of the network. All the camera poses we use for supervision are with respect to a standardised canonical object frame with X axis being left to right, and Z axis being upright. From each label instance, we sample 340 rays for supervision. Thus, the effective number of rays we use for supervising each input image is 5*340. Additionally, on each camera ray we query 100 points uniformly up to 2 meters during volume rendering. Thus for each input instance we query 5*340*100 number of points from the input-conditioned decoder.

We initialize our ResNet-34 based encoder with pre-trained weights from ImageNet-1k classification task. These pre-trained weights are borrowed from TorchVision. And our decoder's weights are initialized uniformly using PyTorch's default initialization scheme. We train our network for about 450 epochs in multi-gpu setting using Distributed Data Parallel (DDP). We train on 32 Nvidia-V100 32GB GPU's with a batch size of 8 on each GPU. Thus, our effective batch size is 8*32. This approximately took about 2.5 days to train using 32 GPU's. We train our network with Adam optimizer [1] with a starting learning rate of 5^{-5} that is updated conditionally using a reduce on plateau learning rate scheduler with a reduction factor and threshold of 0.5 and 5^{-3} respectively.

1.2. Self-Training from Image Collections

As mentioned in the main manuscript, we don't have access to cameras during this phase and we train our models to infer shape in canonical object frame using multi-hypothesis cameras. At the beginning of training, for each instance, we initialize a multi-hypothesis camera consisting of 10 cameras randomly. For the first 10 epochs, we freeze the model weights and update each multi-hypothesis camera using the multi-hypothesis camera loss suggested in the original manuscript. This ensures that even before we begin training our network, our multi-hypothesis cameras are at an acceptable pose. We then learn both our model and multi-hypothesis cameras for additional 40 epochs. Another important point to note is that we have a different update rate for multi-hypothesis cameras and model parameters. In each mini-batch step, we update multi-hypothesis camera parameters 10 times for every model update. This is achieved through an additional multi-hypothesis camera optimization loop in the training step.

We consider Adam optimizer with a constant learning rate of 10^-6 and 0.1 for training for model and multi-hypothesis cameras respectively. We again train each category-specific model on 16 Nvidia-V100 32 GPU's using DDP training strategy with a batch size of 4 on each GPU. This phase took about 1-3 days for each category-level expert based on the dataset size.

Additionally, similar to pre-training, for each camera in multi-hypothesis camera we consider 340 rays with 100 samples on each ray.

1.3. Multi-Category Distillation

Unlike prior phases, that relies on ray-wise supervision, as mentioned in the main manuscript, we perform a point-wise supervision to train our final unified model. We randomly sample 25000 3D points in the standardized canonical object frame for every image in each step step.

We train this network using Adam optimizer with a learning rage of 10^{-6} for 75 epochs on 32 Nvidia-V100 GPU's using DDP. This training phase is relatively faster in comparison to prior phases and took about 2 days to distill all category-level experts and synthetic data-only model into a single model.

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015. 1