hw4_r11944004

Student: 李勝維

ID: r11944004

Department: GINM11

Problem 1: 3D Novel View Synthesis

1. Please explain:

a. The NeRF idea in your own words
NeRF is able to construct 3d novel view by representing the schene by implicit representation.
Instead of common vertex / mesh representation, NeRF represents the schene by a function:
$F_\theta = (x, d) \rightarrow (c, \sigma)$, where x is a 3d vector representing a space coordinate, d is a 2d vector representing a viewing angle, c is a 3d vector representing a color by its r, g, b component and sigma is a scalar representing the density of the point.
In other words, NeRF can render a scene with view dependent color and light reflection by varying the function input.

b. Which part of NeRF do you think is the most important
Since the MLP approximating the aforementioned function, F_θ, is the core component of NeRF. I think the MLP network design is the most important part of the work. The MLP first processes the input coordinate x, to produce the density sigma and a intermediate vector. After this, another MLP processes the intermediate vector and the input viewing angle, d, to produce the view dependent color. In my opion, this approach greatly improves the effectness and efficiency of the training process, which leads to better convergence and performance.

c. Compare NeRF's pros/cons w.r.t. other novel view synthesis work
Pros:

- Training NeRF does not need to access the ground truth 3D model.

- NeRF can produce highly-detailed images and not be limited to simple shapes with low geometric complexity.

Cons:

- Training NeRF usually takes a long time
- The trained NeRF is only applicable to a single scene

2. Describe the implementation details of Direct Voxel Grid Optimization(DVGO) for the given dataset. You need to explain DVGO's method in your own ways.

To speed up NeRF training, DVGO deploys these techniques:

1. A density voxel grid to produce sharp details in a lower grid resolution
2. Coarse 3D space searching + fine detail reconstruction including view-dependent effects

Density Voxel Grid Representation

DVGO uses a voxel grid to represent 3D scenes and store information about different modalities, such as density and color. This allows for efficient rendering and high levels of detail in the final image. To improve the quality of the rendered image, DVGO also proposes the use of a density voxel grid, which only stores density values for volume rendering. This can help reduce the negative effects of lower voxel resolution on the final image. Overall, the use of voxel grids in DVGO allows for effective and efficient volume rendering.

After considering three different settings:

1. softplus → alpha → interpolation
2. softplus → interpolation→ alpha
3. interpolation→ softplus → alpha

DVGO shows that the third setting produces the most clear results, since it applies all the non-linear activation after the trilinear interpolation.

Coarse 3D Space Searching

DVGO first identifies the coarse 3D areas of interest in a scene by finding a bounding box that tightly encloses the camera frustums of training views. This allows DVGO to focus on the fine details of the scene during the reconstruction process.

Because most scenes contain a significant amount of free space, this approach allows DVGO to efficiently reconstruct the scene while still capturing important details.

Fine Scene Representation

In the final stage of the reconstruction process, DVGO uses a higher-resolution density voxel grid to capture fine details. To represent color emission in a way that produces high-quality results while also maintaining a fast training speed, DVGO employs a hybrid representation that combines both explicit and implicit techniques. This allows DVGO to avoid the drawbacks of each individual approach, resulting in more accurate and efficient scene reconstruction.

Hyperparameters:

Please read the next section. I use setting A as my final setting.

3. Given novel view camera pose from transforms_val.json, your model should render novel view images. Please evaluate your generated images and ground truth images with the following three metrics (mentioned in the NeRF paper). Try to use at least two different hyperparameter settings and discuss/analyze the results.

Explain the Meaning of These Metrics:

- PSNR: Peak signal-to-noise ratio, is a measure of the fidelity of an image reconstruction algorithm. It is calculated by taking the ratio of the maximum possible signal power to the power of the difference between the original and reconstructed images. A higher PSNR value indicates that the reconstructed image is of higher quality.
- SSIM: Structural similarity index, is a metric that measures the similarity between two images. It takes into account both the luminance and the structure of the images, and is typically used to evaluate the quality of image compression algorithms. A higher SSIM value indicates that the two images are more similar.
- LPIPS: Learned Perceptual Image Patch Similarity, is a metric that measures the similarity between two images from the perspective of a NN. It is calculated by

training the NN on a large dataset of human judgments of image similarity, and then using the trained network to evaluate the similarity of two images. A lower LPIPS value indicates that the two images are more similar.

Setting	PSNR	SSIM	LPIPS with VGG
A	35.64211573600769	0.9772212050642978	0.036156923398375514
B	35.28139705657959	0.9754230033450471	0.03761120233684778

(Unlisted settings follow the default)

Setting A:

- Coarse network is trained for 15000 iterations; Fine network is trained for 60000 iterations
- Fine network is a 4-layer MLP with width = 256

Setting B:

- Default values from the DVGO repo

Problem 2: Self-Supervised Pre-training for Image Classification

1. Describe the implementation details of your SSL method for pre-training the ResNet50 backbone. (Include but not limited to the name of the SSL method you used, data augmentation for SSL, learning rate schedule, optimizer, and batch size setting for this pre-training phase)

I adapted the code from https://github.com/lucidrains/byol-pytorch as the implementation.

Bootstrap Your Own Latent, BYOL, uses two NNs to learn: the online and target network.

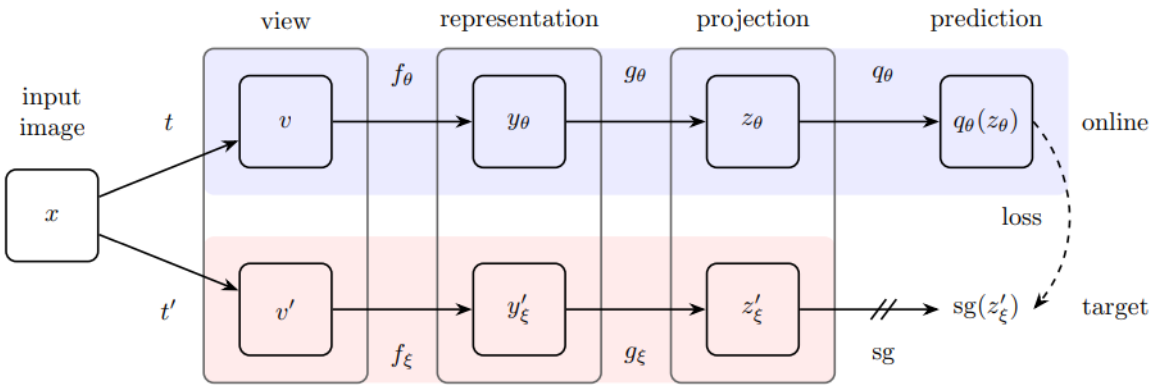


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\text{sg}(z'_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded, and y_θ is used as the image representation.

The online network both task an augmented image as input, and output its representation. The objective of the online network is to predict the output of the target network, i.e., produces the same representation the two different augmented version of the same image. The online network is updated by SGD while the target network is updated by an exponential moving average on the online network.

BYOL implements the idea of contrastive learning by using batch normalization: instead of comparing positive samples and negative samples, batch normalization allows BYOL to compare the output of an image and the “average image” to perform contrastive learning. This idea is described in this article: Understanding Self-Supervised and Contrastive Learning with "Bootstrap Your Own Latent" (BYOL) - generally intelligent

Hyperparameters:

SSL method: BYOL

Data augmentation:

```
RandomApply: ColorJitter(0.8, 0.8, 0.8, 0.2) with p=0.3
RandomGrayscale(p=0.2),
RandomHorizontalFlip(),
RandomApply: GaussianBlur((3, 3), (1.0, 2.0)) with p=0.2
RandomResizedCrop((128, 128)),
Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

Backbone architecture:

- Resnet50 (torchvision.models.resnet50)

Learning algorithm(backbone):

- Trained for 1000 epochs
- Optimizer: Adam with learning rate of 1e-4
- Scheduler: Linear warmup with 1000 steps + Cosine annealing
- FP16 auto mixed precision training
- Batch size of 512

Classifier architecture:

- Four-layer MLP with hidden_size of 256
- Dropout of 0.1
- LeakyReLU with slope of 0.2

Learning algorithm(classifier):

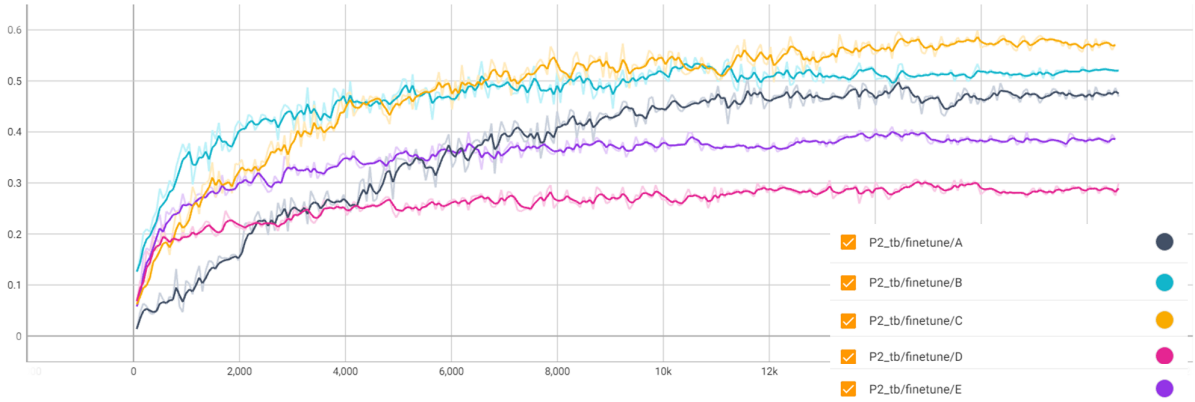
- Applies the same data augmentation as SSL
- Trained for 300 epochs
- Optimizer: Adam with learning rate of 5e-4
- Scheduler: Linear warmup with 100 steps + Cosine annealing
- FP16 auto mixed precision training
- Batch size of 64

Please conduct the Image classification on Office-Home dataset as the downstream task. Also, please complete the following Table, which contains different image classification setting, and discuss/analyze the results.

Setting	Pre-training	Fine-tuning	Validation Acc %
A	X	Full	50.7%
B	w/ label	Full	55.1%
C	w/o label	Full	59.9%
D	w/ label	Freeze backbone	30.8%

E	w/o label	Freeze backbone	38.4%

Validation accuracy to training step fig.



Comparing settings D and E

Since the backbones are frozen, we can compare the performance of the given backbones by evaluating setting E and setting D. It is surprising that the unsupervised backbone in setting E outperforms the one in setting D.

In order to understand the reasons behind this, I trained another unsupervised backbone with only 200 epochs.

Backbone pre-training	Validation Acc %
Supervised (D)	30.8%
Unsupervised+1000 epochs (E)	38.4%
Unsupervised+200 epochs (new)	25.8%

My hypothesis is that the longer training of the unsupervised backbone in setting E is the cause of its superior performance compared to the shorter-trained backbone in setting D.

Comparing settings A, B, and C

Without the benefit of pretraining, setting A has the slowest convergence and the poorest performance.

When comparing setting B and setting C, it is observed that setting B, which uses a supervised backbone, converges faster. However, the unsupervised backbone in setting C performs better than the supervised one, which is unexpected.