DLCV HW4 Report

tags: DLCV

Course	Student ID	Name	Date
2022 Fall NTU DLCV	R10943109	Shiuan-Yun Ding	2022-12-13

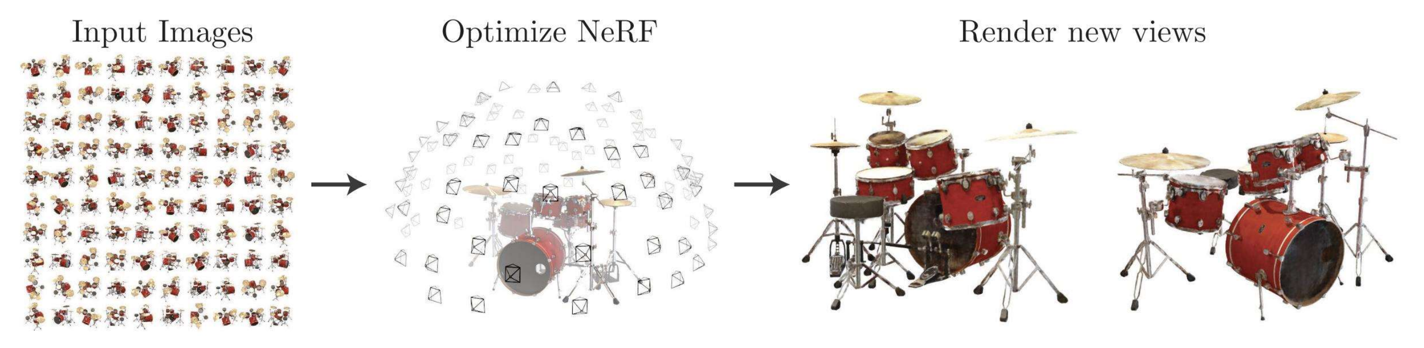
HackMD Link: https://hackmd.io/@mirkat1206/B17X7Evwj

Problem 1: 3D Novel View Synthesis (50%)

Metrix	Baseline (10%)	My Results
PSNR	35	35.19
SSIM	0.97	0.9742

(5%) Please explain

a. the NeRF idea in your own words



The general NeRF idea

- NeRF tries to use a deep fully-connected neural network without any convolution layer to synthesize views of an object.
- NeRF takes a set of images of an object from different angle as inputs of training. For inference, it takes a 5D coordinate (x, y, z, θ, φ) as inputs, and outputs a single volume density and view-dependent RGB color, which can be rendered as an image of view of the object from that specific position and angle.
- I think this problem is very useful for movie/cartoon/game industry, because they need to render 3D models a lot.

b. which part of NeRF do you think is the most important

Two improvements in NeRF make it outstands the other methods:

- Positional Encoding
  - The author of NeRF states that the network that directly takesa 5D coordinate (x, y, z, θ, φ) as inputs cannot generate high quality results. In particular, this method fails at images with high-frequency variation in color and geometry.
  - NeRF copes with this problem by using the idea of positional encoding. It first map the original 5D coordinate (x, y, z, θ, φ) into a higher dimensional space, then sends the encoded coordinates into MLP.
- Hierarchical volume sampling
  - The author points out that the previous works are not efficient enough because they spend much time on sampling redundent points.
  - NeRF proposed a hierarchical volume sampling method that first optimize a "coarse" network, and based on the "coarse" networks, NeRF can produce a "fine" network that uses a more informed sampling points.
  - Compared with the previous works that use uniformed sampling points, NeRF with biased, nonuniformed sampling points can render a better image with high efficiency.

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

- Pros

- NeRF can render an images with highest quality. The details in both geometry and apperance remain clear with NeRF, while other works produce blurred details.
- Require only 5 MB for the entire network weights.

- Cons
  - The orignal NeRF requires at least 12 hours for training, while LLFF only takes 10 minutes to train.

Comparison with previous works

Reference

- NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis (https://arxiv.org/pdf/2003.08934.pdf)
- Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction (https://arxiv.org/pdf/2111.11215.pdf)
- NeRF神经辐射场ECCV2020 (https://blog.csdn.net/qq_40943760/article/details/125189835)

(10%) Describe the implementation details of Direct Voxel Grid Optimization (DVGO)

Inspired by NeRF, DVGO proposes a super fast convergence approach that can train a model with comparable quality with NeRF but reduce the training length from over 12 hours to less than 15 minutes.

Compared with NeRF using 5D coordinate (x, y, z, θ, φ), DVGO uses voxel-grid representation (x, V), where x is the queried 3D point, V is the voxel grid.

For implementation, same as NeRF, DVGO first searches the coarse geometry of a scene, then reconstructs the fine detail including view-dependent effects. The reasons that DVGO is much faster than NeRF are as the following:

- Use post-activation interpolation on voxel density to produce sharp surfaces in lower grid resolution.
- Prone the original voxel density optimization to suboptimal geometry solutions.

By using the above method, DVGO achieves ~ 45x speedup.

(15%) Evaluate the generated images and ground truchth images

Three metrics

#	# of fine-tune iter	# of voxel	step size	PSNR	SSIM	LPIPS
0 (default)	20000	160 ** 3	0.5	35.1671	0.9744	0.0411
1	20000	160 ** 3	0.75	35.0645	0.9738	0.0424
2	20000	160 ** 3	0.25	35.2174	0.9745	0.0413
3	40000	160 ** 3	0.25	35.2054	0.9746	0.0407
4	20000	240 ** 3	0.5	**35.3512**	**0.9759**	0.0382
5	20000	320 ** 3	0.5	35.3325	**0.9759**	**0.0376**

Discussion

- From the experiments (0, 1, 2), we can see that step size does not make much difference.
- From the experiments (2, 3), we can see that # of fine-tune iter does not make much difference.
- From the experiments (0, 4, 5), we can see that # of voxel makes much difference.
- Even though the experiments (4, 5) have the highest performance, the training cost and storage cost are the highest as well. Therefore, I choose the experiment 0 as the final version of this homework.

Explain the meaning of three metrics

- PSNR (Peak Signal to Noise Ratio)
  - The ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity.
  - The bigger, the better.
- SSIM (Structural Similarity Index)
  - It measures the "structural information" similarity between two images.
  - The value is from 0 (worst) to 1 (best).
- LPIPS (Learned Perceptual Image Patch Similarity)
  - Evaluate the distance between image patches.
  - Higher means more different, while lower means more similar.
- reference:
  - 有真实参照的图像质量的客观评估指标:SSIM、PSNR和LPIPS (https://zhuanlan.zhihu.com/p/309892873)
  - https://github.com/richzhang/PerceptualSimilarity
  - Wikipedia

Problem 2: Self-Supervised Pre-training for Image Classification (50%)

	Simple Baseline (5%)	Strong Baseline (5%)	My Result
Accuracy	0.36	0.40	0.4557

(10%) Describe the implementation deatils

Pre-training (Backbone)

- Method : SSL Method BYOLhttps://github.com/lucidrains/byol-pytorch
- Backbone Model : Resnet50
- Data Augmentation

```
transform=transforms.Compose([
    transforms.Resize(128),
    transforms.CenterCrop(128),
    transforms.ToTensor(),
    transforms.Normalize(mean, std),
])
```

Optimizer	Learning Rate	Batch Size	Epochs
Adam	3e-4	64	47

Fine-tuning (Classifier)

- Classifier Model

```
self.classifier = nn.Sequential(
    nn.Linear(1000, 512),
    nn.BatchNorm1d(512),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(512, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(256, 65)
)
```

- Data Augmentation

```
transform=transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.CenterCrop((128)),
    transforms.ColorJitter(brightness=0.5, contrast=0.5, saturation=0.5, hue=0.5),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation((-30, 30), expand=False),
    transforms.ToTensor(),
    transforms.Normalize(mean, std),
])
```

Optimizer	Learning Rate	Batch Size	Epochs
Adam	default	64	200

- Reference
  - BYOL: Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning (Paper Explained) (https://www.youtube.com/watch?v=YPfUiOMYOEE)

(20%) Complete the table

Setting	Pre-training (Mini-ImageNet)	Fine-tuning (Office-Home dataset)	Validataion Accuracy (Office-Home dataset)
A	-	train backbone + classifier	Train: 3889/3951 (98.43%) Test: 148/406 (36.45%)
B	w/ label (by TA)	train backbone + classifier	Train: 3925/3951 (99.34%) Test: 188/406 (46.31%)
C	w/o label (by me)	train backbone + classifier	Train: 3892/3951 (98.51%) Test: 185/406 (45.57%)
D	w/ label (by TA)	fix backbone train classifier	Train: 683/3951 (17.31%) Test: 57/406 (14.04%)
E	w/o label (by me)	fix backbone train classifier	Train: 674/3951 (17.06%) Test: 55/406 (13.55%)

Discussion

- Before doing data augmentation, the fine-tuning method would easily fall into mode collapse.
- After doing data augmentation, we know that, from the difference between accuracies of train set and test set, the models A/B/C fall into over-fitting problem.
- The resulting differences between B/C and between D/E are not very obvious. Based on other students' results, C should be better than B, and E should be better than D. The reason that my result does not show this trend may be because
    i. I did not train my backbone well enough
    ii. Both B/C fall into over-fitting
    iii. both D/E need more epochs for training
- However, my results have passed the baseline, I decide not to futher optimize my model.