PMLDL’21. Project Progress D1.3

Student Information

Student: Anna Boronina

Group: BS18-DS-01

Telegram: @whoorma

The idea: using GAN for pointcloud generation

Progress

1. Before building a GAN, I decided to build an autoencoder based on PointNet architecture. I found out that for pointclouds there is a special loss function called Chamfer distance. It calculates distance between one point from pointcloud1 and the closest point from pointcloud 2. It was the first step towards “wow something is working”.
2. I use existing architecture for PointNet (didn’t see a reason to implement my own from scratch at the beginning, because I need to get just *something*)
3. I decided to skip training with labels for now, so using only one class out of nine.
4. I have the following transformations:
    1. `PointSampler`- chooses N points from a pointcloud (I use N=1024)
    2. `Normalize` - normalizes pointcloud’s points
    3. `RandomNoise` - adds random gaussian noise to the pointcloud; The authors of PointNet mention that it improves performance
    4. `RandomRotation_z` - random rotation around Z axis. Needed to make the model view-invariant
    5. `ToSorted` - sorts the points; first - by X axis numbers, then by Y axis and then by Z; it’s not mentioned in the original paper, but when I’ve been facing problems during a whole week, I decided to add this transformation; the last best results had this transformation included, but I don’t know how it would be without this transformation - will know soon though
    6. `ToTensor` - typical transformation to transform a list of lists of lists to `torch.tensor`.

5. I switched from SGD to Adam, and then to AdamW. Nothing was working with different numbers for optimizers and schedulers, the loss was decreasing very slow. Like this:

```
optimizer = optim.AdamW(model.parameters(), lr=0.001, betas=
(0.9, 0.999))
loss_func = chamfer.chamfer_distance
```

```
1:      loss: 114.74627102122588
500:    loss: 95.31617267587606
4000:   loss: 88.1720572846602
4500:   loss: 87.78182278087965
5000:   loss: 90.27712236516616
5500:   loss: 82.83565093657549
6000:   loss: 82.32193874695722
```

6. Then I did this (mainly, changed `betas` for `AdamW` and didn't add any noise to the input):

```
optimizer = optim.AdamW(model.parameters(), lr=0.0009, betas=
(0.8, 0.8))
loss_func = chamfer.chamfer_distance # ChamferDistance()
```

```
1:      loss1: 115.91247917624081
1000:   loss2: 73.62187641110442
2000:   loss2: 26.4102506280857
3000:   loss2: 14.324622097653501
4000:   loss1: 13.436617977478925
5000:   loss1: 12.253603479441475
6000:   loss1: 10.979960245244643
```

7. After it finally worked, I decided to clean up a bit and splitted the project into several .ipynb files: two of them are active - the first one contains dataloaders creation (I create them once and then save them, they might contain different classes and transformations), another one contains PointNet Autoencoder. Additionally, I decided that it's time to record my results. Something like this:

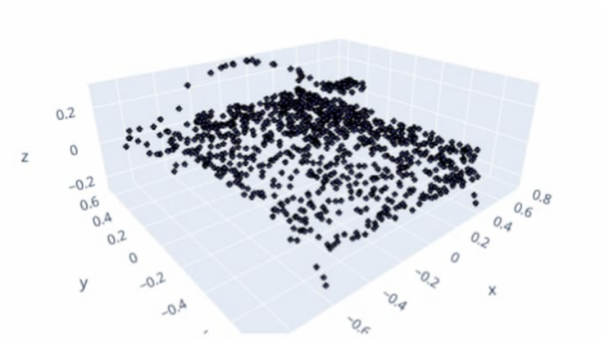
Dataloader Name	Batch Size	Tags	Hyperparameters
dataloader_beds	32	PointSampler(1024) Normalize RandomNoise ToTensor RandomRotation_z ToSorted	
dataloader_beds_no_rot	32	PointSampler(1024) Normalize RandomNoise ToTensor ToSorted	

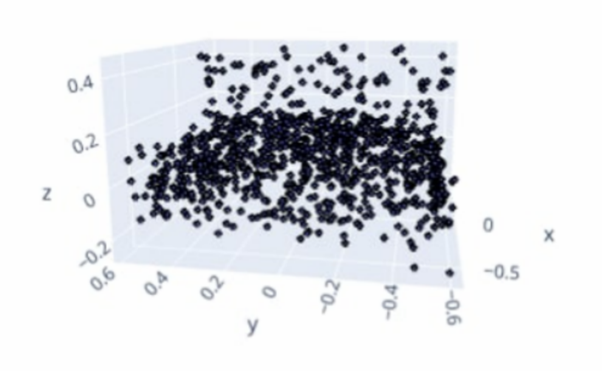
dataloader_beds_no_noise_rot	32	PointSampler(1024) Normalize ToTensor ToSorted	`pc_autoencoder = PointNetAE(num_points=1024)`
			`pc_autoencoder.to(device)`
			`optimizer = optim.AdamW(pc_autoencoder.parameters(), lr=0.0009, betas=(0.8, 0.8))`
			`scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=5000, gamma=0.8)`
			`loss_func = chamfer.chamfer_distance # ChamferDistance()`
			`train_with_chamfer_dist(pc_autoencoder, beds_loader, loss_func, optimizer,`
			`train_func=train_pcautoencoder, epochs=20000, print_every_e=60, scheduler=scheduler)`

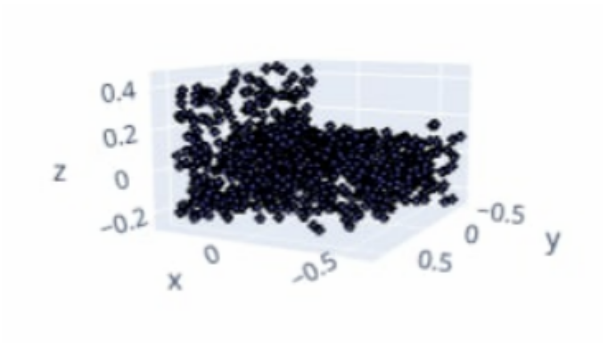
8. Originally, a pointcloud is an unstructured set of points, and PointNet Architecture is invariant to $n!$ permutations, where n is the number of points in a pointcloud. Nevertheless, as I mentioned before, I created `ToSorted` transformation to structure pointclouds. When I tried to train an autoencoder with the same hyperparameters but without `ToSorted` transform, the model didn't learn anything. Also, maybe now, when a pointcloud is similar to images (sorted by X axis, then by Y, then by Z) I can also try to apply 3d convolution with kernel other than 1.

Results

First image: the original bed
Second and third images: output of the autoencoder (it looks like a bed!)







Links

Autoencoder training

Data preprocessing and dataloaders creation

Folder with everything project-related

What's next

1. Train autoencoder on several classes, not just one
2. Try different combinations of transformations
3. Create architecture for GAN