

Meeting 7 - Teksta daļas darbi

Jautājums

Q: Kāpēc aprakstī dziļo mašīnmācību un modeļu arhitektūras, ja darbs ir par datu ielādes metodēm.

A: Datu ielādes metodes tiek izmantotas ar ResNet modeļiem, kuriem tika aprakstīta pamat-teorija. Izprast modeļus svarīgi, lai novērstu to, ka datu ielādi aizkavē tieši nepareizi sagatavota modeļu apmācība. Jāpanāk, ka pārējās pētījuma daļas strādā ļoti efektīvi, lai varētu pētīt tieši datu ielādi.

Bakalaura atskaite

Sagaatavot atskaiti, iekļaut tajā šo struktūru (pamaini nosaukumus pēc saviem ieskatiem) + nepieciešams uzrakstīt 1.

1. Ievads

1.1. Dziļā māšīnmācīšanās

```
1 | * Pamata arhitektūras - Linārie slāņi, Aktivizācijas funkcijas, Softmax (priekš k
```

```
2 | * Kļūdas funkcijas - MSE, CCE, BCE
```

```
3 | * Atpakaļizplatīšanās algoritms (grafiski un ar vienādojumiem)
```

```
4 | * Apmācām Parametru Optizācijas algoritmi SGD
```

```
5 | * Metrikas - novērtēt cik labs rezultāts Accuracy, F1
```

1.2. Attēlu klasifikācija

- ConvNets
- ResNet / DenseNet (Novēršam vanishing gradient, deeper nets, SOTA)

1.3. PyTorch vide

- * Modeļu implementāciju <> viendājumi
- * Datu ielādes process (DataSet)

2. Metodoloģija

TODO

1. Ņemt līkās apakšnodaļas ievādā. Aprakstīt lewad nodaļas (15lpp). Aprakstīt Metodoloģiju - Datu ielādes metodes, Datu kopas, Apmācības protokolu (15lpp)

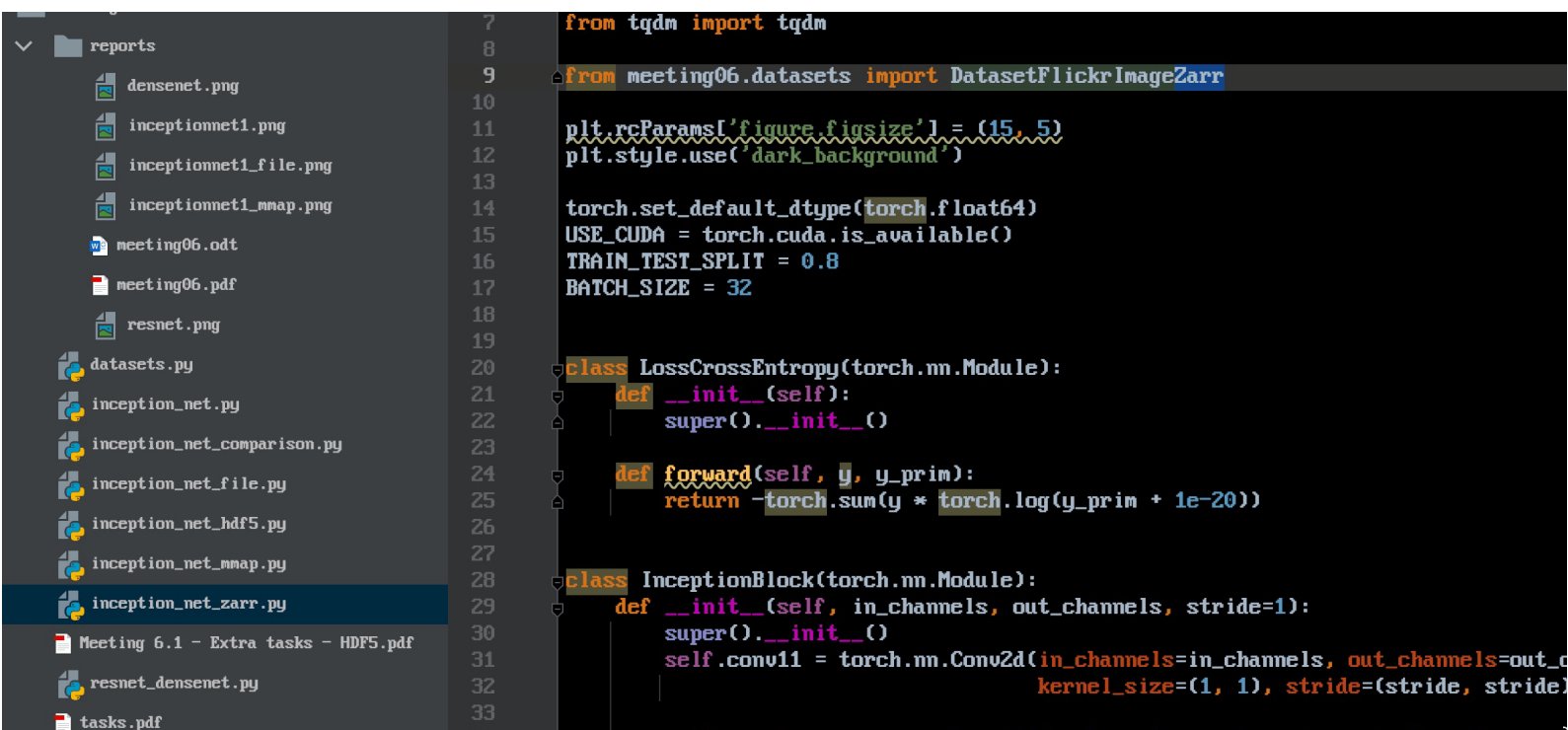
Table of Contents

1. Ievads.....	3
1.1. Dziļā māšīnmācīšanās.....	3
Pamata arhitektūras.....	3
Linārie slāņi.....	3
Aktivizācijas funkcijas.....	3
Kļūdas funkcijas.....	3
MAE.....	3
MSE.....	3
CCE.....	4
BCE.....	4
Atpakaļizplatīšanās algoritms.....	4
Apmācāmo parametru optimizācijas algoritmi.....	4
Metrikas.....	4
1.2. Attēlu klasifikācija.....	5
1.3. PyTorch vide.....	5
2. Metodoloģija.....	5
2.1. Datu ielādes metodes.....	5
2.1.1. Failu sistēma.....	5
2.1.2. Linux mmap.....	5
2.1.3. nVidia mmap.....	5
2.1.4. HDF5.....	5
2.1.5. PostgreSQL.....	5
2.2. Datu kopas.....	5
2.2.1. CIFAR10.....	5
2.2.2. Tiny ImageNet.....	5
2.3. Apmācības protokols.....	5
3. Rezultāti.....	6
4. Tālākie pētījumi.....	6
5. Secinājumi.....	6

2. Nvidia-mmap metode (cupy mmap, izmantojot cp.asarray) <https://github.com/cupy/cupy/issues/3431>

3. Vēl viena metode, kuru testēt torchvision read_image (vajadzētu būt ātrāk par PIL) https://pytorch.org/vision/stable/generated/torchvision.io.read_image.html#torchvision.io.read_image

4. Laba ideja atrast vēl citas metods ZARR



```
7 from tqdm import tqdm
8
9 from meeting06.datasets import DatasetFlickrImageZarr
10
11 plt.rcParams['figure.figsize'] = (15, 5)
12 plt.style.use('dark_background')
13
14 torch.set_default_dtype(torch.float64)
15 USE_CUDA = torch.cuda.is_available()
16 TRAIN_TEST_SPLIT = 0.8
17 BATCH_SIZE = 32
18
19
20 class LossCrossEntropy(torch.nn.Module):
21     def __init__(self):
22         super().__init__()
23
24     def forward(self, y, y_prim):
25         return -torch.sum(y * torch.log(y_prim + 1e-20))
26
27
28 class InceptionBlock(torch.nn.Module):
29     def __init__(self, in_channels, out_channels, stride=1):
30         super().__init__()
31         self.conv11 = torch.nn.Conv2d(in_channels=in_channels, out_channels=out_channels, kernel_size=(1, 1), stride=(stride, stride))
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
```

5. Datasets un modeļus un citus hyper params padot ar argparse

```
parser = argparse.ArgumentParser(add_help=False)
```

```

3 # Dir creation params
4 parser.add_argument('-id', default=0, type=int)
5 parser.add_argument('-sequence_name', default='sequence', type=str)
6 parser.add_argument('-run_name', default='run', type=str)
7
8 parser.add_argument('-learning_rate', default=1e-3, type=float)
9
10 parser.add_argument('-model', default='model_emo_VAE_v3', type=str)
11 parser.add_argument('-datasource', default='datasource_emo_v2', type=str)
12
13 args, args_other = parser.parse_known_args()
14
15 Model = getattr(__import__('modules_core.' + args.model, fromlist=['Model']), 'Model')
16 get_data_datasets = getattr(__import__('modules_core.' + args.datasource, fromlist=
    ['get_data_datasets']),
17                             'get_data_datasets')
18
19 model_inst = Model()
20 y_prim = model_inst.forward(x)

```

6. Pārbaudīt vai tiešām self.x nav np.array / list un nestāv uz RAM

```

168 # WORKAROUND: save time
169 if idx >= MAX_LEN - 1:
170     break
171
172 self.y[:] = np.array(y)
173
174 self.root = zarr.open(str(dataset_file_path), mode='r')
175 self.x = self.root['samples']
176 self.y = F.one_hot(torch.LongTensor(self.root['labels']))
177 self.data_length = len(self.y)
178
179 def __len__(self):

```

self.x ja tas ir numpy, tad viss ir RAM
ja tas ir ZARR datu tips, tad
visticamāk nav

7. Var testēt ietekmi uz datu ielādes ātrumu datu tipiem

```

self.dataset_shape = (self.data_length, 3, self.image_height, self.image_width)

self.x = open_memmap(
    str(dataset_file_path), mode='w+', dtype='float64', shape=self.dataset_shape
)

```

float16

tieši prims math operācijām
cast float32

8. Data loader test dažādus num_workers un pin_memory!

```

__initialized = False

def __init__(self, dataset: Dataset[T_col], batch_size: Optional[int] = 1,
             shuffle: bool = False, sampler: Optional[Sampler] = None,
             batch_sampler: Optional[Sampler[Sequence]] = None,
             num_workers: int = 0, collate_fn: Optional[_collate_fn_t] = None,
             pin_memory: bool = False, drop_last: bool = False,
             timeout: float = 0, worker_init_fn: Optional[_worker_init_fn_t] = None,
             multiprocessing_context=None, generator=None,
             persistent_workers: bool = False):
    torch._C._log_api_usage_once("python.data_loader")

    if num_workers < 0:

```

0 = ielādē tajā pašā procesā, kas ir training
1 => 1 parallel
8 => 8 parāllel

9. Sadalīt makaronus

```

return self.data_length

def __getitem__(self, idx):
    x_mmap = self.x[idx]
    x = torch.from_numpy(np.copy(self.x[idx]))
    x_t = torch.FloatTensor(x)
    return x, self.y[idx]

class DatasetFlickrImageHDF5(torch.utils.data.Dataset):
    def __init__(self, root: str = 'data', force: bool = False, chs

```

10. Pieraksts tensor casting

```

# TODO: what to use???
y.append(0)
torch.LongTensor(...)
self.y[:] = torch.tensor(y, torch.long)
else:
    self.data_length = self.data.shape[0]

```

11. Vēlāk var arī testēt - GPU 16bit float => 32bit float

<https://github.com/NVIDIA/apex>

12. ⚠️ Atrast citu datu kopu Flickr datu kopas vietā, kur high res attēli un dotas klases, ja nevar paspēt tad vnk prognozēt cik gari vidēji ir teikumi katram paraugam ar MSE