task1

November 28, 2022

1 №1

:

```
[1]: !pip install -q tqdm
     !pip install --upgrade --no-cache-dir gdown
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: gdown in /usr/local/lib/python3.7/dist-packages
(4.4.0)
Collecting gdown
  Downloading gdown-4.5.4-py3-none-any.whl (14 kB)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from gdown) (1.15.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages
(from gdown) (4.64.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist-
packages (from gdown) (4.6.3)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.7/dist-
packages (from gdown) (2.23.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-
packages (from gdown) (3.8.0)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests[socks]->gdown) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests[socks]->gdown) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests[socks]->gdown) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests[socks]->gdown) (2022.9.24)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in
/usr/local/lib/python3.7/dist-packages (from requests[socks]->gdown) (1.7.1)
Installing collected packages: gdown
  Attempting uninstall: gdown
    Found existing installation: gdown 4.4.0
    Uninstalling gdown-4.4.0:
      Successfully uninstalled gdown-4.4.0
Successfully installed gdown-4.5.4
```

Google Drive :

```
[2]: from google.colab import drive
     drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

, , (gdrive) :

```
[3]: EVALUATE_ONLY = True
     TEST_ON_LARGE_DATASET = True
     TISSUE_CLASSES = ('ADI', 'BACK', 'DEB', 'LYM', 'MUC', 'MUS', 'NORM', 'STR',␣
      ↪'TUM')
     DATASETS_LINKS = {
         'train': '1XtQzVQ5XbrfxpLHJuL0XBGJ5U7CS-cLi',
         'train_small': '1qd45xXfDwdZjktLFwQb-et-mAaFeCzOR',
         'train_tiny': '1I-2ZOuXLd4QwhZQQltp817Kn3J0Xgbui',
         'test': '1RfPou3pFKpuHDJZ-D9XDFzgvwpUBFlDr',
         'test_small': '1wbRsog0n7uGlHIPGLhyN-PMeT2kdQ2lI',
         'test_tiny': '1viiB0s041CNsAK4itvX8PnYthJ-MDnQc'
     }
```

:

```
[4]: from pathlib import Path
     import numpy as np
     from typing import List
     from tqdm.notebook import tqdm
     from time import sleep
     from PIL import Image
     import IPython.display
     from sklearn.metrics import balanced_accuracy_score
     import gdown
```

```
[5]: # extra imports

     import torch
     from torch import nn
     from torch.utils.tensorboard import SummaryWriter
     from torchvision.transforms.functional import to_pil_image
     import os
     from collections import namedtuple
     import matplotlib.pyplot as plt
```

```
[6]: %load_ext tensorboard
```

1.0.1 Dataset

, , ( ).

```
[7]: class Dataset:

         def __init__(self, name):
             self.name = name
             self.is_loaded = False
             url = f"https://drive.google.com/uc?
      ↪export=download&confirm=pbef&id={DATASETS_LINKS[name]}"
             output = f'{name}.npz'
             gdown.download(url, output, quiet=False)
             print(f'Loading dataset {self.name} from npz.')
             print('before')
             np_obj = np.load(f'{name}.npz')
             print('after')
             self.images = np_obj['data']
             self.labels = np_obj['labels']
             self.n_files = self.images.shape[0]
             self.is_loaded = True
             print(f'Done. Dataset {name} consists of {self.n_files} images.')

         def image(self, i):
             # read i-th image in dataset and return it as numpy array
             if self.is_loaded:
                 return self.images[i, :, :, :]

         def images_seq(self, n=None):
             # sequential access to images inside dataset (is needed for testing)
             for i in range(self.n_files if not n else n):
                 yield self.image(i)

         def random_image_with_label(self):
             # get random image with label from dataset
             i = np.random.randint(self.n_files)
             return self.image(i), self.labels[i]

         def random_batch_with_labels(self, n):
             # create random batch of images with labels (is needed for training)
             indices = np.random.choice(self.n_files, n)
             imgs = []
             for i in indices:
                 img = self.image(i)
                 imgs.append(self.image(i))
```

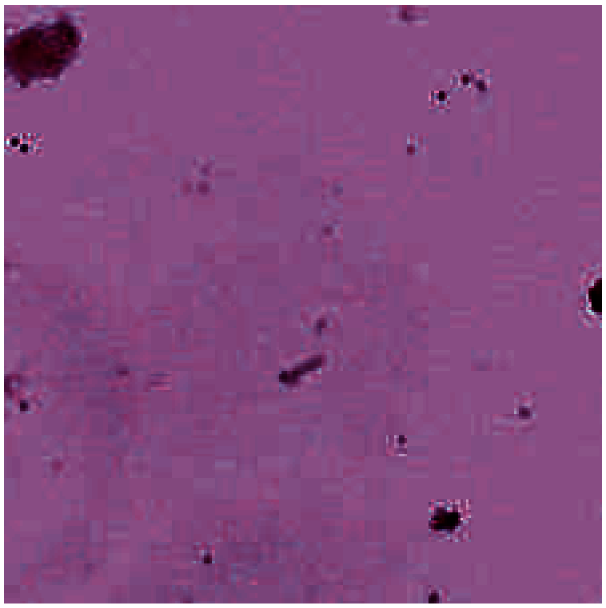
```python
        logits = np.array([self.labels[i] for i in indices])
        return np.stack(imgs), logits

    def image_with_label(self, i: int):
        # return i-th image with label from dataset
        return self.image(i), self.labels[i]
```

1.0.2 Dataset

, . , . ,
.

[8]:
```python
d_train_tiny = Dataset('train_tiny')

img, lbl = d_train_tiny.random_image_with_label()
print()
print(f'Got numpy array of shape {img.shape}, and label with code {lbl}.')
print(f'Label code corresponds to {TISSUE_CLASSES[lbl]} class.')

pil_img = Image.fromarray(img)
IPython.display.display(pil_img)
```

```
Downloading…
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1I-2ZOuXLd4Qwh
ZQQltp817Kn3J0Xgbui
To: /content/train_tiny.npz
100%|      | 105M/105M [00:00<00:00, 275MB/s]

Loading dataset train_tiny from npz.
before
after
Done. Dataset train_tiny consists of 900 images.

Got numpy array of shape (224, 224, 3), and label with code 1.
Label code corresponds to BACK class.
```



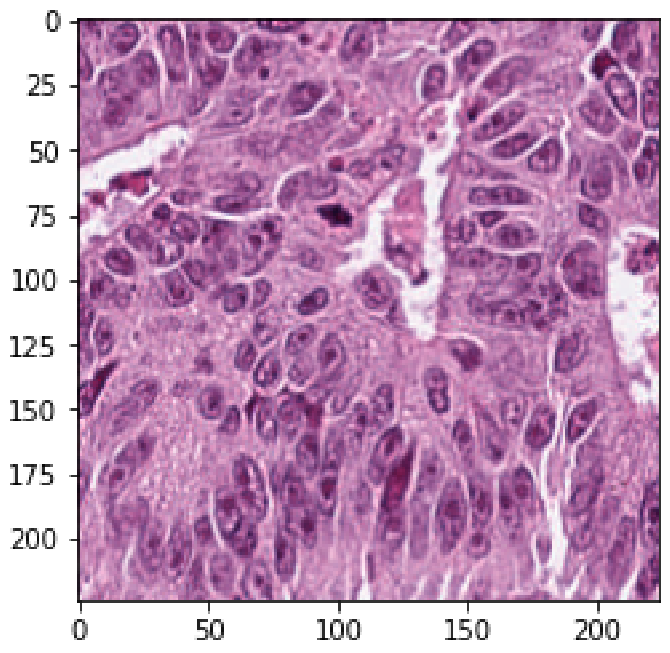
1.0.3 dataloader Pytorch

```
[9]: class TissueDataset(torch.utils.data.Dataset):
         def __init__(self, dataset: Dataset, mode: str, transforms=None):
             # mode : train, validation, full

             images, labels = dataset.images, dataset.labels
             self.transforms = transforms
             # LBL1
             if mode == 'full':
               self.samples = list(zip(images, labels))
             else:
               train_size = int(0.8 * len(images))
               val_size = len(images) - train_size
               train_dataset, val_dataset = torch.utils.data.
       ↪random_split(list(zip(images, labels)), [train_size, val_size])
               if mode == 'train':
                 self.samples = train_dataset
               else:
                 self.samples = val_dataset

         def __getitem__(self, idx: int):
             image, label = self.samples[idx]
             if self.transforms:
                 image = self.transforms(image)
```

```
        image = torch.tensor(image.transpose(2, 0, 1), dtype=torch.float32) /␣
↪255
        return image, label

    def __len__(self):
        return len(self.samples)
```

[10]:
```
# example

data = TissueDataset(d_train_tiny, 'validation')
for img, label in data:
  plt.imshow(img.permute(1, 2, 0))
  print(img.size(), label)
  break
```

```
torch.Size([3, 224, 224]) 8
```



1.0.4 Metrics

, : 1. , 2. .

[11]:
```
class Metrics:
```

```python
    @staticmethod
    def accuracy(gt: List[int], pred: List[int]):
        assert len(gt) == len(pred), 'gt and prediction should be of equal length'
        return sum(int(i[0] == i[1]) for i in zip(gt, pred)) / len(gt)

    @staticmethod
    def accuracy_balanced(gt: List[int], pred: List[int]):
        return balanced_accuracy_score(gt, pred)

    @staticmethod
    def print_all(gt: List[int], pred: List[int], info: str):
        print(f'metrics for {info}:')
        print('\t accuracy {:.4f}:'.format(Metrics.accuracy(gt, pred)))
        print('\t balanced accuracy {:.4f}:'.format(Metrics.accuracy_balanced(gt, pred)))
```

1.0.5 Model

, .

save, load .
.

, , . , ,
, , ,
, .

, , : 1. ; 2. ; 3.
; 4. - ( );
5. (,); 6. ,
(,); 7. /
( ); 8. ;
9. ; 10.
( ) 11. . .
.

.

[12]:
```python
from collections import Counter

class ConvBlock(nn.Module):
    def __init__(self, in_channels: int, out_channels: int):
        super(ConvBlock, self).__init__()

        self.convblock = nn.Sequential(
            nn.Conv2d(in_channels=in_channels, out_channels=out_channels,␣
 ↪kernel_size=3, stride=1, padding=1, bias=True),
            nn.BatchNorm2d(num_features=out_channels),
            nn.ReLU(inplace=True)
        )

    def forward(self, x):
        out = self.convblock(x)
        return out

class ConvNet(nn.Module):
    cfg = [16, "M", 32, "M", 64, "M", 128, "M", 256, "M"]
    def __init__(self, in_channels: int, num_classes: int):
        super().__init__()

        self.features = torch.nn.Sequential()
        last_output = in_channels
        for i in ConvNet.cfg:
            if isinstance(i, int):
                self.features.append(ConvBlock(last_output, i))
                last_output = i
            else:
                self.features.append(torch.nn.MaxPool2d(kernel_size=2))

        spatial_size = 224 / 2**(Counter(ConvNet.cfg)['M'])
        assert spatial_size == int(spatial_size) # has to be int
        spatial_size = int(spatial_size)
        self.classification_head = nn.Sequential(
            nn.Flatten(),
            nn.Linear(last_output * spatial_size * spatial_size , 128),
            nn.BatchNorm1d(128),
            nn.ReLU(),
            nn.Dropout(p=0.5),
            nn.Linear(128, num_classes)
        )

    def forward(self, x):
        out = self.features(x)
        return self.classification_head(out)
```

[13]:
```
device = torch.device('cpu')
if torch.cuda.is_available():
    device = torch.device('cuda', 0)

net = ConvNet(3, 9)
net.to(device)
```

```
[13]: ConvNet(
        (features): Sequential(
          (0): ConvBlock(
            (convblock): Sequential(
              (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
              (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
              (2): ReLU(inplace=True)
            )
          )
          (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
      ceil_mode=False)
          (2): ConvBlock(
            (convblock): Sequential(
              (0): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
              (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
              (2): ReLU(inplace=True)
            )
          )
          (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
      ceil_mode=False)
          (4): ConvBlock(
            (convblock): Sequential(
              (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
              (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
              (2): ReLU(inplace=True)
            )
          )
          (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
      ceil_mode=False)
          (6): ConvBlock(
            (convblock): Sequential(
              (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
              (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
              (2): ReLU(inplace=True)
            )
          )
          (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
      ceil_mode=False)
          (8): ConvBlock(
            (convblock): Sequential(
              (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
              (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
```

```
        (2): ReLU(inplace=True)
      )
    )
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (classification_head): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=12544, out_features=128, bias=True)
    (2): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (3): ReLU()
    (4): Dropout(p=0.5, inplace=False)
    (5): Linear(in_features=128, out_features=9, bias=True)
  )
)
```

[21]:

```
class Model:
    def __init__(self, cfg):
        self.cfg = cfg
        self.device = cfg.device
        self.out_dir = os.path.join(cfg.out_dir, cfg.model_name)
        os.makedirs(self.out_dir, exist_ok=True)
        log_dir = os.path.join(self.out_dir, "runs")
        os.makedirs(log_dir, exist_ok=True)

        self.writer = SummaryWriter(log_dir=log_dir)

        #LBL5
        if cfg.num_pretrained_epoch > 0:
            self.model = ConvNet(3, 9)
            weights_dir = os.path.join(self.out_dir, "weights")
            self.load_pretrained(os.path.join(weights_dir, f"{self.cfg.
 ↪model_name}_epoch_{cfg.num_pretrained_epoch}.pth"))
        else:
            self.model = ConvNet(3, 9)

        self.criterion = nn.CrossEntropyLoss()


    def save(self, path: str):
        torch.save(self.model.state_dict(), path)


    def load(self, name: str):
      # https://drive.google.com/file/d/1nrASFx0MWfRB7f9FLc322uM-I-l8UHL6/view?
 ↪usp=sharing
```

```
        name_to_id_dict = '1nrASFx0MWfRB7f9FLc322uM-I-l8UHL6'
        output = f'{name}.pth'
        gdown.download(f'https://drive.google.com/uc?id={name_to_id_dict}',␣
↪output, quiet=False)
        self.model.load_state_dict(torch.load(output, map_location='cpu'))


    def load_pretrained(self, path: str):
        self.model.load_state_dict(torch.load(path))


    def train(self, train_ds: Dataset):
        self.model.to(self.device)
        params = [p for p in self.model.parameters() if p.requires_grad]
        optimizer = torch.optim.Adam(params, self.cfg.lr)
        lr_scheduler = torch.optim.lr_scheduler.MultiStepLR(optimizer,␣
↪milestones=self.cfg.milestones)

        train_dl, val_dl = self.get_train_dataloaders(
            train_ds,
            self.cfg.batch_size, self.cfg.batch_size_val, self.cfg.num_workers
        )

        weights_dir = os.path.join(self.out_dir, "weights")
        os.makedirs(weights_dir, exist_ok=True)

        for epoch in range(1, self.cfg.epochs + 1):
            print(f"Epoch: {epoch}")

            logs = self.train_epoch(optimizer, train_dl, epoch)
            for key, value in logs.items():
                self.writer.add_scalar(key, value, epoch)

            if lr_scheduler is not None:
                lr_scheduler.step()

            #LBL3
            metrics = self.evaluate(val_dl)
            for key, value in metrics.items():
                self.writer.add_scalar(f"val/{key}", value, epoch)

            if epoch % 4 == 0:
              self.save(os.path.join(weights_dir, f"{self.cfg.
↪model_name}_epoch_{epoch}.pth"))

        #LBL2
        self.save(os.path.join(weights_dir, f"{self.cfg.model_name}.pth"))
    def train_epoch(self, optimizer, train_dl, epoch):
        self.model.train()

        lr_scheduler = None
        if epoch == 1:
            warmup_factor = 1e-3
            warmup_iters = len(train_dl)
            #print(warmup_iters, 'warmup_iters')

            lr_scheduler = torch.optim.lr_scheduler.LinearLR(
                optimizer, start_factor=warmup_factor, total_iters=warmup_iters,
                verbose=True
            )

        for images, targets in tqdm(train_dl):
            optimizer.zero_grad()
            loss = self.criterion(self.model(images.to(self.device)), targets.
↪to(self.device))

            loss.backward()
            optimizer.step()

            if lr_scheduler is not None:
                lr_scheduler.step()

        #LBL4
        print(f"train loss: {loss}")
        return {"loss": loss.detach()}


    def evaluate(self, val_dl):
        self.model.eval()

        prediction, target = [], []
        for images, targets in tqdm(val_dl):
            outputs = self.model(images.to(self.device)).to('cpu')
            target += list(targets.numpy())
            _, predicted = torch.max(outputs, 1)
            prediction += list(predicted.detach().numpy())

        acc = Metrics.accuracy(target, prediction)
        balanced_acc = Metrics.accuracy_balanced(target, prediction)

        #LBL4
        print(f"val accuracy: {acc}")
        return {"accuracy": acc, "balanced_acc": balanced_acc}


    def get_train_dataloaders(self, ds_numpy, batch_size, batch_size_val,␣
↪num_workers):
        train_ds = TissueDataset(ds_numpy, "train")
        val_ds = TissueDataset(ds_numpy, "validation")

        train_dl = torch.utils.data.DataLoader(
            train_ds,
            batch_size=batch_size,
            num_workers=num_workers,
            shuffle=True
        )
        val_dl = torch.utils.data.DataLoader(
            val_ds,
            batch_size=batch_size_val,
            num_workers=num_workers,
            shuffle=False,
        )
        return train_dl, val_dl


    def test_on_dataset(self, dataset: Dataset, limit=None):
        self.model.eval()
        self.model.to('cpu')
        predictions = []
        n = dataset.n_files if not limit else int(dataset.n_files * limit)
        for img in tqdm(dataset.images_seq(n), total=n):
            predictions.append(self.test_on_image(img))
        return predictions


    def test_on_image(self, img: np.ndarray):
        img = torch.tensor(img.transpose(2, 0, 1), dtype=torch.float32) / 255
        _, prediction = torch.max(self.model(img.unsqueeze(0)), 1)
        return prediction.numpy()
```

1.0.6

.

. . 'train_small' 'test_small'.

[17]:
```
d_train = Dataset('train')
d_test = Dataset('test')
```

```
Downloading…
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1XtQzVQ5Xbrfxp
LHJuL0XBGJ5U7CS-cLi
To: /content/train.npz
100%|      | 2.10G/2.10G [00:54<00:00, 38.2MB/s]
```

```
Loading dataset train from npz.
before
after
Done. Dataset train consists of 18000 images.
```

```
Downloading…
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1RfPou3pFKpuHD
JZ-D9XDFzgvwpUBFlDr
To: /content/test.npz
100%|      | 525M/525M [00:12<00:00, 42.0MB/s]
```

```
Loading dataset test from npz.
before
after
Done. Dataset test consists of 4500 images.
```

[18]:
```
cfg_dict = {
    "out_dir": Path('drive/MyDrive/nn_msu/'),
    "batch_size": 64,
    "batch_size_val": 64,
    "num_workers": 2,
    "model_name": 'ConvNet',
    "num_pretrained_epoch": 0,
    "device": device,
    "epochs": 10,
    "lr": 0.001,
    "weight_decay": 0.0001,
    "milestones": [6, 8, 9],
}

cfg = namedtuple("Config", cfg_dict.keys())(**cfg_dict)
```

[19]:
```
EVALUATE_ONLY = False
```

1.0.7 Train

[22]:
```
model = Model(cfg)
if not EVALUATE_ONLY:
    model.train(d_train)
    model.save('ConvNetModel')
else:
    #todo: your link goes here
```

```python
model.load('ConvNetModel')
```

```
Epoch: 1
Adjusting learning rate of group 0 to 1.0000e-06.
```

```
  0%|          | 0/225 [00:00<?, ?it/s]
```

```
Adjusting learning rate of group 0 to 5.4400e-06.
Adjusting learning rate of group 0 to 9.8800e-06.
Adjusting learning rate of group 0 to 1.4320e-05.
Adjusting learning rate of group 0 to 1.8760e-05.
Adjusting learning rate of group 0 to 2.3200e-05.
Adjusting learning rate of group 0 to 2.7640e-05.
Adjusting learning rate of group 0 to 3.2080e-05.
Adjusting learning rate of group 0 to 3.6520e-05.
Adjusting learning rate of group 0 to 4.0960e-05.
Adjusting learning rate of group 0 to 4.5400e-05.
Adjusting learning rate of group 0 to 4.9840e-05.
Adjusting learning rate of group 0 to 5.4280e-05.
Adjusting learning rate of group 0 to 5.8720e-05.
Adjusting learning rate of group 0 to 6.3160e-05.
Adjusting learning rate of group 0 to 6.7600e-05.
Adjusting learning rate of group 0 to 7.2040e-05.
Adjusting learning rate of group 0 to 7.6480e-05.
Adjusting learning rate of group 0 to 8.0920e-05.
Adjusting learning rate of group 0 to 8.5360e-05.
Adjusting learning rate of group 0 to 8.9800e-05.
Adjusting learning rate of group 0 to 9.4240e-05.
Adjusting learning rate of group 0 to 9.8680e-05.
Adjusting learning rate of group 0 to 1.0312e-04.
Adjusting learning rate of group 0 to 1.0756e-04.
Adjusting learning rate of group 0 to 1.1200e-04.
Adjusting learning rate of group 0 to 1.1644e-04.
Adjusting learning rate of group 0 to 1.2088e-04.
Adjusting learning rate of group 0 to 1.2532e-04.
Adjusting learning rate of group 0 to 1.2976e-04.
Adjusting learning rate of group 0 to 1.3420e-04.
Adjusting learning rate of group 0 to 1.3864e-04.
Adjusting learning rate of group 0 to 1.4308e-04.
Adjusting learning rate of group 0 to 1.4752e-04.
Adjusting learning rate of group 0 to 1.5196e-04.
Adjusting learning rate of group 0 to 1.5640e-04.
Adjusting learning rate of group 0 to 1.6084e-04.
Adjusting learning rate of group 0 to 1.6528e-04.
Adjusting learning rate of group 0 to 1.6972e-04.
Adjusting learning rate of group 0 to 1.7416e-04.
Adjusting learning rate of group 0 to 1.7860e-04.
Adjusting learning rate of group 0 to 1.8304e-04.
Adjusting learning rate of group 0 to 1.8748e-04.
Adjusting learning rate of group 0 to 1.9192e-04.
Adjusting learning rate of group 0 to 1.9636e-04.
Adjusting learning rate of group 0 to 2.0080e-04.
Adjusting learning rate of group 0 to 2.0524e-04.
Adjusting learning rate of group 0 to 2.0968e-04.
Adjusting learning rate of group 0 to 2.1412e-04.
Adjusting learning rate of group 0 to 2.1856e-04.
Adjusting learning rate of group 0 to 2.2300e-04.
Adjusting learning rate of group 0 to 2.2744e-04.
Adjusting learning rate of group 0 to 2.3188e-04.
Adjusting learning rate of group 0 to 2.3632e-04.
Adjusting learning rate of group 0 to 2.4076e-04.
Adjusting learning rate of group 0 to 2.4520e-04.
Adjusting learning rate of group 0 to 2.4964e-04.
Adjusting learning rate of group 0 to 2.5408e-04.
Adjusting learning rate of group 0 to 2.5852e-04.
Adjusting learning rate of group 0 to 2.6296e-04.
Adjusting learning rate of group 0 to 2.6740e-04.
Adjusting learning rate of group 0 to 2.7184e-04.
Adjusting learning rate of group 0 to 2.7628e-04.
Adjusting learning rate of group 0 to 2.8072e-04.
Adjusting learning rate of group 0 to 2.8516e-04.
Adjusting learning rate of group 0 to 2.8960e-04.
Adjusting learning rate of group 0 to 2.9404e-04.
Adjusting learning rate of group 0 to 2.9848e-04.
Adjusting learning rate of group 0 to 3.0292e-04.
Adjusting learning rate of group 0 to 3.0736e-04.
Adjusting learning rate of group 0 to 3.1180e-04.
Adjusting learning rate of group 0 to 3.1624e-04.
Adjusting learning rate of group 0 to 3.2068e-04.
Adjusting learning rate of group 0 to 3.2512e-04.
Adjusting learning rate of group 0 to 3.2956e-04.
Adjusting learning rate of group 0 to 3.3400e-04.
Adjusting learning rate of group 0 to 3.3844e-04.
Adjusting learning rate of group 0 to 3.4288e-04.
Adjusting learning rate of group 0 to 3.4732e-04.
Adjusting learning rate of group 0 to 3.5176e-04.
Adjusting learning rate of group 0 to 3.5620e-04.
Adjusting learning rate of group 0 to 3.6064e-04.
Adjusting learning rate of group 0 to 3.6508e-04.
Adjusting learning rate of group 0 to 3.6952e-04.
Adjusting learning rate of group 0 to 3.7396e-04.
Adjusting learning rate of group 0 to 3.7840e-04.
Adjusting learning rate of group 0 to 3.8284e-04.
Adjusting learning rate of group 0 to 3.8728e-04.
Adjusting learning rate of group 0 to 3.9172e-04.
Adjusting learning rate of group 0 to 3.9616e-04.
Adjusting learning rate of group 0 to 4.0060e-04.
Adjusting learning rate of group 0 to 4.0504e-04.
Adjusting learning rate of group 0 to 4.0948e-04.
Adjusting learning rate of group 0 to 4.1392e-04.
Adjusting learning rate of group 0 to 4.1836e-04.
Adjusting learning rate of group 0 to 4.2280e-04.
Adjusting learning rate of group 0 to 4.2724e-04.
Adjusting learning rate of group 0 to 4.3168e-04.
Adjusting learning rate of group 0 to 4.3612e-04.
Adjusting learning rate of group 0 to 4.4056e-04.
Adjusting learning rate of group 0 to 4.4500e-04.
Adjusting learning rate of group 0 to 4.4944e-04.
Adjusting learning rate of group 0 to 4.5388e-04.
Adjusting learning rate of group 0 to 4.5832e-04.
Adjusting learning rate of group 0 to 4.6276e-04.
Adjusting learning rate of group 0 to 4.6720e-04.
Adjusting learning rate of group 0 to 4.7164e-04.
Adjusting learning rate of group 0 to 4.7608e-04.
Adjusting learning rate of group 0 to 4.8052e-04.
Adjusting learning rate of group 0 to 4.8496e-04.
Adjusting learning rate of group 0 to 4.8940e-04.
Adjusting learning rate of group 0 to 4.9384e-04.
Adjusting learning rate of group 0 to 4.9828e-04.
Adjusting learning rate of group 0 to 5.0272e-04.
Adjusting learning rate of group 0 to 5.0716e-04.
Adjusting learning rate of group 0 to 5.1160e-04.
Adjusting learning rate of group 0 to 5.1604e-04.
Adjusting learning rate of group 0 to 5.2048e-04.
Adjusting learning rate of group 0 to 5.2492e-04.
Adjusting learning rate of group 0 to 5.2936e-04.
Adjusting learning rate of group 0 to 5.3380e-04.
Adjusting learning rate of group 0 to 5.3824e-04.
Adjusting learning rate of group 0 to 5.4268e-04.
Adjusting learning rate of group 0 to 5.4712e-04.
Adjusting learning rate of group 0 to 5.5156e-04.
Adjusting learning rate of group 0 to 5.5600e-04.
Adjusting learning rate of group 0 to 5.6044e-04.
Adjusting learning rate of group 0 to 5.6488e-04.
Adjusting learning rate of group 0 to 5.6932e-04.
Adjusting learning rate of group 0 to 5.7376e-04.
Adjusting learning rate of group 0 to 5.7820e-04.
Adjusting learning rate of group 0 to 5.8264e-04.
Adjusting learning rate of group 0 to 5.8708e-04.
Adjusting learning rate of group 0 to 5.9152e-04.
Adjusting learning rate of group 0 to 5.9596e-04.
Adjusting learning rate of group 0 to 6.0040e-04.
Adjusting learning rate of group 0 to 6.0484e-04.
Adjusting learning rate of group 0 to 6.0928e-04.
Adjusting learning rate of group 0 to 6.1372e-04.
Adjusting learning rate of group 0 to 6.1816e-04.
Adjusting learning rate of group 0 to 6.2260e-04.
Adjusting learning rate of group 0 to 6.2704e-04.
Adjusting learning rate of group 0 to 6.3148e-04.
Adjusting learning rate of group 0 to 6.3592e-04.
Adjusting learning rate of group 0 to 6.4036e-04.
Adjusting learning rate of group 0 to 6.4480e-04.
Adjusting learning rate of group 0 to 6.4924e-04.
Adjusting learning rate of group 0 to 6.5368e-04.
Adjusting learning rate of group 0 to 6.5812e-04.
Adjusting learning rate of group 0 to 6.6256e-04.
Adjusting learning rate of group 0 to 6.6700e-04.
Adjusting learning rate of group 0 to 6.7144e-04.
Adjusting learning rate of group 0 to 6.7588e-04.
Adjusting learning rate of group 0 to 6.8032e-04.
Adjusting learning rate of group 0 to 6.8476e-04.
Adjusting learning rate of group 0 to 6.8920e-04.
Adjusting learning rate of group 0 to 6.9364e-04.
Adjusting learning rate of group 0 to 6.9808e-04.
Adjusting learning rate of group 0 to 7.0252e-04.
Adjusting learning rate of group 0 to 7.0696e-04.
Adjusting learning rate of group 0 to 7.1140e-04.
Adjusting learning rate of group 0 to 7.1584e-04.
Adjusting learning rate of group 0 to 7.2028e-04.
Adjusting learning rate of group 0 to 7.2472e-04.
Adjusting learning rate of group 0 to 7.2916e-04.
Adjusting learning rate of group 0 to 7.3360e-04.
Adjusting learning rate of group 0 to 7.3804e-04.
Adjusting learning rate of group 0 to 7.4248e-04.
Adjusting learning rate of group 0 to 7.4692e-04.
Adjusting learning rate of group 0 to 7.5136e-04.
Adjusting learning rate of group 0 to 7.5580e-04.
Adjusting learning rate of group 0 to 7.6024e-04.
Adjusting learning rate of group 0 to 7.6468e-04.
Adjusting learning rate of group 0 to 7.6912e-04.
Adjusting learning rate of group 0 to 7.7356e-04.
Adjusting learning rate of group 0 to 7.7800e-04.
Adjusting learning rate of group 0 to 7.8244e-04.
Adjusting learning rate of group 0 to 7.8688e-04.
Adjusting learning rate of group 0 to 7.9132e-04.
Adjusting learning rate of group 0 to 7.9576e-04.
Adjusting learning rate of group 0 to 8.0020e-04.
Adjusting learning rate of group 0 to 8.0464e-04.
Adjusting learning rate of group 0 to 8.0908e-04.
Adjusting learning rate of group 0 to 8.1352e-04.
Adjusting learning rate of group 0 to 8.1796e-04.
Adjusting learning rate of group 0 to 8.2240e-04.
Adjusting learning rate of group 0 to 8.2684e-04.
Adjusting learning rate of group 0 to 8.3128e-04.
Adjusting learning rate of group 0 to 8.3572e-04.
Adjusting learning rate of group 0 to 8.4016e-04.
Adjusting learning rate of group 0 to 8.4460e-04.
Adjusting learning rate of group 0 to 8.4904e-04.
Adjusting learning rate of group 0 to 8.5348e-04.
Adjusting learning rate of group 0 to 8.5792e-04.
Adjusting learning rate of group 0 to 8.6236e-04.
Adjusting learning rate of group 0 to 8.6680e-04.
Adjusting learning rate of group 0 to 8.7124e-04.
Adjusting learning rate of group 0 to 8.7568e-04.
Adjusting learning rate of group 0 to 8.8012e-04.
Adjusting learning rate of group 0 to 8.8456e-04.
Adjusting learning rate of group 0 to 8.8900e-04.
Adjusting learning rate of group 0 to 8.9344e-04.
Adjusting learning rate of group 0 to 8.9788e-04.
Adjusting learning rate of group 0 to 9.0232e-04.
Adjusting learning rate of group 0 to 9.0676e-04.
Adjusting learning rate of group 0 to 9.1120e-04.
Adjusting learning rate of group 0 to 9.1564e-04.
Adjusting learning rate of group 0 to 9.2008e-04.
Adjusting learning rate of group 0 to 9.2452e-04.
Adjusting learning rate of group 0 to 9.2896e-04.
Adjusting learning rate of group 0 to 9.3340e-04.
Adjusting learning rate of group 0 to 9.3784e-04.
Adjusting learning rate of group 0 to 9.4228e-04.
Adjusting learning rate of group 0 to 9.4672e-04.
Adjusting learning rate of group 0 to 9.5116e-04.
Adjusting learning rate of group 0 to 9.5560e-04.
Adjusting learning rate of group 0 to 9.6004e-04.
Adjusting learning rate of group 0 to 9.6448e-04.
Adjusting learning rate of group 0 to 9.6892e-04.
Adjusting learning rate of group 0 to 9.7336e-04.
Adjusting learning rate of group 0 to 9.7780e-04.
Adjusting learning rate of group 0 to 9.8224e-04.
Adjusting learning rate of group 0 to 9.8668e-04.
Adjusting learning rate of group 0 to 9.9112e-04.
Adjusting learning rate of group 0 to 9.9556e-04.
Adjusting learning rate of group 0 to 1.0000e-03.
train loss: 0.5640280842781067

  0%|          | 0/57 [00:00<?, ?it/s]


val accuracy: 0.7813888888888889
Epoch: 2
```

```
  0%|          | 0/225 [00:00<?, ?it/s]

train loss: 0.3132307529449463

  0%|          | 0/57 [00:00<?, ?it/s]

val accuracy: 0.5336111111111111
Epoch: 3

  0%|          | 0/225 [00:00<?, ?it/s]

train loss: 0.274355947971344

  0%|          | 0/57 [00:00<?, ?it/s]

val accuracy: 0.4538888888888889
Epoch: 4

  0%|          | 0/225 [00:00<?, ?it/s]

train loss: 0.35548290610313416

  0%|          | 0/57 [00:00<?, ?it/s]

val accuracy: 0.8763888888888889
Epoch: 5

  0%|          | 0/225 [00:00<?, ?it/s]

train loss: 0.2006574273109436

  0%|          | 0/57 [00:00<?, ?it/s]

val accuracy: 0.8108333333333333
Epoch: 6

  0%|          | 0/225 [00:00<?, ?it/s]

train loss: 0.15478897094726562

  0%|          | 0/57 [00:00<?, ?it/s]

val accuracy: 0.8591666666666666
Epoch: 7

  0%|          | 0/225 [00:00<?, ?it/s]
```

```
train loss: 0.06800190359354019

  0%|          | 0/57 [00:00<?, ?it/s]


val accuracy: 0.9869444444444444
Epoch: 8

  0%|          | 0/225 [00:00<?, ?it/s]


train loss: 0.02574928291141987

  0%|          | 0/57 [00:00<?, ?it/s]


val accuracy: 0.9911111111111112
Epoch: 9

  0%|          | 0/225 [00:00<?, ?it/s]


train loss: 0.028608273714780807

  0%|          | 0/57 [00:00<?, ?it/s]


val accuracy: 0.99
Epoch: 10

  0%|          | 0/225 [00:00<?, ?it/s]


train loss: 0.07655631750822067

  0%|          | 0/57 [00:00<?, ?it/s]


val accuracy: 0.9905555555555555
```

1.0.8 Test

[23]:
```
EVALUATE_ONLY = True

model = Model(cfg)
if not EVALUATE_ONLY:
    model.train(d_train)
    model.save('ConvNetModel')
else:
    #todo: your link goes here
    model.load('ConvNetModel')
```

```
Downloading…
From: https://drive.google.com/uc?id=1nrASFx0MWfRB7f9FLc322uM-I-l8UHL6
```

```
To: /content/ConvNetModel.pth
100%|     | 8.02M/8.02M [00:00<00:00, 177MB/s]
```

:

```
[24]: # evaluating model on 10% of test dataset
pred_1 = model.test_on_dataset(d_test, limit=0.1)
Metrics.print_all(d_test.labels[:len(pred_1)], pred_1, '10% of test')
```

```
  0%|          | 0/450 [00:00<?, ?it/s]


metrics for 10% of test:
         accuracy 0.9956:
         balanced accuracy 0.9956:
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1987:
UserWarning: y_pred contains classes not in y_true
  warnings.warn("y_pred contains classes not in y_true")
```

:

```
[25]: # evaluating model on full test dataset (may take time)
if TEST_ON_LARGE_DATASET:
    pred_2 = model.test_on_dataset(d_test)
    Metrics.print_all(d_test.labels, pred_2, 'test')
```

```
  0%|          | 0/4500 [00:00<?, ?it/s]


metrics for test:
         accuracy 0.9729:
         balanced accuracy 0.9729:
```

```
[26]: #LBL6
%tensorboard --logdir drive/MyDrive/nn_msu/ConvNet/runs
```

```
<IPython.core.display.Javascript object>
```

.

, . . ,
. .

pdf (->)
pdf .

1.0.9

. , test_tiny,
(2%) test. ,
.

, .

[28]:
```python
final_model = Model(cfg)
final_model.load('best')
d_test_tiny = Dataset('test_tiny')
pred = model.test_on_dataset(d_test_tiny)
Metrics.print_all(d_test_tiny.labels, pred, 'test-tiny')
```

```
Downloading…
From: https://drive.google.com/uc?id=1nrASFx0MWfRB7f9FLc322uM-I-l8UHL6
To: /content/best.pth
100%|      | 8.02M/8.02M [00:00<00:00, 62.2MB/s]
Downloading…
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1viiB0s041CNsA
K4itvX8PnYthJ-MDnQc
To: /content/test_tiny.npz
100%|      | 10.6M/10.6M [00:00<00:00, 134MB/s]

Loading dataset test_tiny from npz.
before
after
Done. Dataset test_tiny consists of 90 images.

  0%|          | 0/90 [00:00<?, ?it/s]


metrics for test-tiny:
         accuracy 0.9333:
         balanced accuracy 0.9333:
```

Google Drive.

[29]:
```python
drive.flush_and_unmount()
```

---

2 ” ”

, .

2.0.1

- timeit :

[ ]:
```python
import timeit

def factorial(n):
    res = 1
    for i in range(1, n + 1):
        res *= i
    return res


def f():
    return factorial(n=1000)

n_runs = 128
print(f'Function f is caluclated {n_runs} times in {timeit.timeit(f,␣
 ↪number=n_runs)}s.')
```

2.0.2 Scikit-learn

” ” scikit-learn (https://scikit-learn.org/stable/). MNIST SVM:

[ ]:
```
# Standard scientific Python imports
import matplotlib.pyplot as plt

# Import datasets, classifiers and performance metrics
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split

# The digits dataset
digits = datasets.load_digits()

# The data that we are interested in is made of 8x8 images of digits, let's
# have a look at the first 4 images, stored in the `images` attribute of the
# dataset.  If we were working from image files, we could load them using
# matplotlib.pyplot.imread.  Note that each image must have the same size. For␣
 ↪these
# images, we know which digit they represent: it is given in the 'target' of
# the dataset.
_, axes = plt.subplots(2, 4)
images_and_labels = list(zip(digits.images, digits.target))
for ax, (image, label) in zip(axes[0, :], images_and_labels[:4]):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('Training: %i' % label)

# To apply a classifier on this data, we need to flatten the image, to
# turn the data in a (samples, feature) matrix:
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))
```

```
# Create a classifier: a support vector classifier
classifier = svm.SVC(gamma=0.001)

# Split data into train and test subsets
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False)

# We learn the digits on the first half of the digits
classifier.fit(X_train, y_train)

# Now predict the value of the digit on the second half:
predicted = classifier.predict(X_test)

images_and_predictions = list(zip(digits.images[n_samples // 2:], predicted))
for ax, (image, prediction) in zip(axes[1, :], images_and_predictions[:4]):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('Prediction: %i' % prediction)

print("Classification report for classifier %s:\n%s\n"
      % (classifier, metrics.classification_report(y_test, predicted)))
disp = metrics.plot_confusion_matrix(classifier, X_test, y_test)
disp.figure_.suptitle("Confusion Matrix")
print("Confusion matrix:\n%s" % disp.confusion_matrix)

plt.show()
```

2.0.3 Scikit-image

, numpy,
, , scikit-image (https://scikit-image.org/). Canny edge detector.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage as ndi

from skimage import feature



# Generate noisy image of a square
im = np.zeros((128, 128))
im[32:-32, 32:-32] = 1

im = ndi.rotate(im, 15, mode='constant')
im = ndi.gaussian_filter(im, 4)
im += 0.2 * np.random.random(im.shape)
```

```
# Compute the Canny filter for two values of sigma
edges1 = feature.canny(im)
edges2 = feature.canny(im, sigma=3)

# display results
fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(8, 3),
                                    sharex=True, sharey=True)

ax1.imshow(im, cmap=plt.cm.gray)
ax1.axis('off')
ax1.set_title('noisy image', fontsize=20)

ax2.imshow(edges1, cmap=plt.cm.gray)
ax2.axis('off')
ax2.set_title(r'Canny filter, $\sigma=1$', fontsize=20)

ax3.imshow(edges2, cmap=plt.cm.gray)
ax3.axis('off')
ax3.set_title(r'Canny filter, $\sigma=3$', fontsize=20)

fig.tight_layout()

plt.show()
```

2.0.4 Tensorflow 2

Tensorflow 2.
, MNIST.

```
[ ]: # Install TensorFlow

import tensorflow as tf

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

```python
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)

model.evaluate(x_test,  y_test, verbose=2)
```

, Google Colab
GPU TPU. " " -> " ".

Tensorflow 2
https://www.tensorflow.org/tutorials?hl=ru.

, Tensorflow 2. ,
, TensorFlow 2. , (
), : https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly.

2.0.5 Numba

, ,
for python , JIT- Numba
(https://numba.pydata.org/). Numba Google Colab : 1.
https://colab.research.google.com/github/cbernet/maldives/blob/master/numba/numba_cuda.ipynb
2. https://colab.research.google.com/github/evaneschneider/parallel-programming/blob/master/COMPASS_gpu_intro.ipynb

, Numba , , ,
. Numba .

2.0.6 zip Google Drive

zip . ,
zip . ,
Google Drive.

2 , tmp PROJECT_DIR, tmp tmp.zip.

```python
[ ]: PROJECT_DIR = "/dev/prak_nn_1/"
     arr1 = np.random.rand(100, 100, 3) * 255
     arr2 = np.random.rand(100, 100, 3) * 255

     img1 = Image.fromarray(arr1.astype('uint8'))
     img2 = Image.fromarray(arr2.astype('uint8'))

     p = "/content/drive/MyDrive/" + PROJECT_DIR
```

```
if not (Path(p) / 'tmp').exists():
    (Path(p) / 'tmp').mkdir()

img1.save(str(Path(p) / 'tmp' / 'img1.png'))
img2.save(str(Path(p) / 'tmp' / 'img2.png'))

%cd $p
!zip -r "tmp.zip" "tmp"
```

tmp.zip tmp2 PROJECT_DIR. tmp2 tmp, 2 .

[ ]:
```
p = "/content/drive/MyDrive/" + PROJECT_DIR
%cd $p
!unzip -uq "tmp.zip" -d "tmp2"
```