CellOmics Coding Document

Yuxiang Zhang

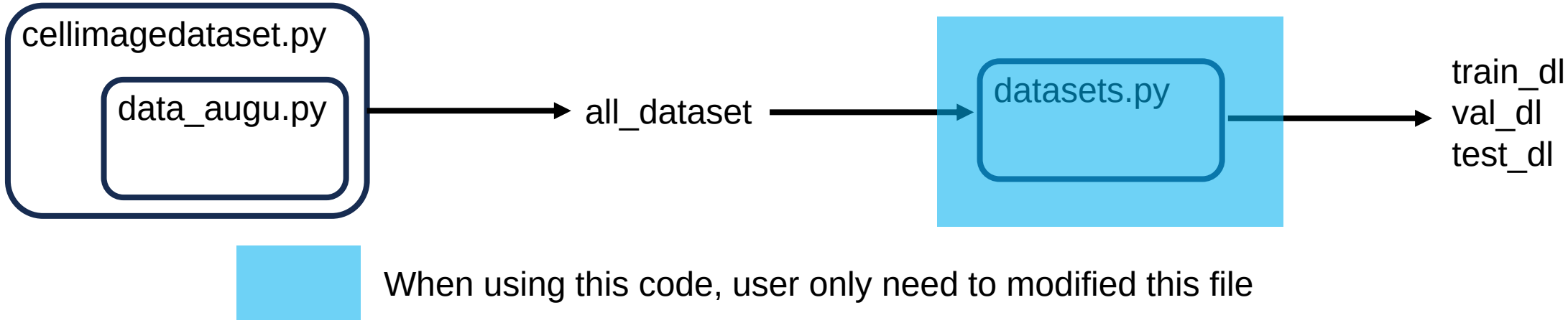
Details

- dataset
- model
- CellOmicsV2train.py
- CellOmicsV4train.py
- CellOmicsV4train.py

dataset

- cellimagedataset.py
  - Return the Dataset class (Pytorch) for different dataset
- data_augu.py
  - Different image data augmentation functions
- datasets.py
  - Return the Dataloader class (Pytorch) for train, validate and test

Pipeline:



cellimagedataset.py

- class
  - EMTCellImageDataset → get_cell_image_dataset
  - iPSImageDataset → get_ips_dataset
- function

The root of label file

Transforms
Simple / Mask
Output format

```
def __init__(self, data_root, csv_file, transform=None, transform_output='mask', downstream_task='None'):
    self.data_root = data_root
    self.data_df = pd.read_csv(csv_file)
    self.transform = transform
    self.transform_output = transform_output
    self.downstream_task = downstream_task
```

- downstream_task
  - 2typesclassification | 3typesclassification → Cell type classification
  - 2typesPCAregression | 3typesPCAregression → PCA coordinates regression
  - 2typesLevelregression | 3typesLevelregression → Gene expression level regression

data_augu.py

- class
  - SimpleTrasform → Simple augmentation
  - MaskContrastiveViewGenerator → **Mask based augmentation**
  - RegionAugmentation → **Region augmentation**

Numbers of view
Normalize or not

Version is same as number in GM slides

```
def __init__(self, num_patch=4, version='v1', normalize=False) -> None:
    self.num_patch = num_patch
    self.normalize = normalize
    self.version = version# Local or Global for Region Augmentation
```

For mask based augmentation, we should calculate the mask for each image using **mask_calcu-batch_cal.py**
PS: DELL sever is recommended

```
def main():
    num_threads = 20
    with Pool(num_threads) as pool:
        pool.map(process_image, image_filenames)
```

Multithreads calculation of mask image

datasets.py (API for dataset)

- load_dataset → dataset
- get_cell_image_dataloader → dataloader

```python
def load_dataset(data_name='emt-cell', num_patch=4,
                 transform_type='offical', normalize=False, downstream_task='None', sample_id=False):
    """
    Loads a dataset for training and testing.
    Parameters:
        data_name (str): name of the dataset
            - 'emt-cell': EMT cell image dataset
            - 'ips-cell': iPS cell image dataset
        num_patch (int): number of patches to be extracted from each image (4 is recommended)
        transform_type (str):
            - 'simple': simple transform (only resize and ToTensor)
            - 'mask-v1': using cellpose based mask (only for EMT cell image dataset)
            - 'mask-v2': using cellpose based mask (global + local, only for EMT cell image dataset)
            - 'region-v1': using Region Augmentation (only local, Region Augmentation V1)
            - 'region-v2': using Region Augmentation (global + local, Region Augmentation V2)
        downstream_task (str):
            - 'None': only for feature extraction
            - '3typesclassification': for 3 cell types classification task
            - '2typesclassification': for 2 cell types classification task
            - '3typesPCAregression': for 3 cell coordinates regression task
            - '2typesPCAregression': for 2 cell coordinates regression task
            - '3typesLevelregression': for 3 gene expression level regression task
            - '2typesLevelregression': for 2 gene expression level regression task
    Returns:
        dataset (torch.data.dataset)
    """
```

- Load different dataset to Dataset class (Pytorch)

```python
def get_cell_image_dataloader(dataset,
                              batch_size=32,
                              k_fold=5,
                              test_shuffle=False,
                              dataset_return=False
                             ):
```

- Load dataset to dataloader class (Pytorch)
- Cross validation setting
- batch size

model

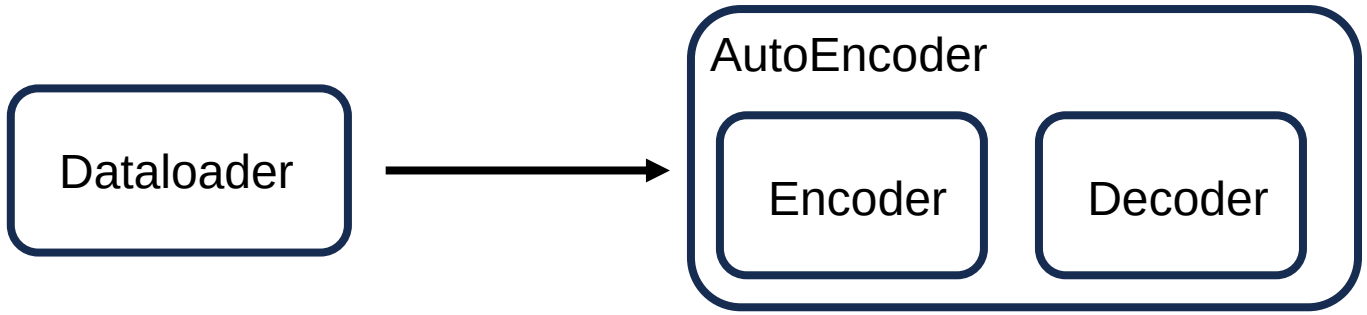
- eval.py
  - Dimensional reduction visualization of extracted features
- loss.py
  - Different loss function for model training and monitoring
- models.py
  - Different block of model (e.g. encoder, decoder, …)
- optim.py
  - LARS strategy for optimization in training process
  - Mostly copy-paste from Layer-wise Adaptive Rate Scaling

models.py

- class
  - encoder
  - decoder
  - AutoEncoder → Combine encoder and decoder for training/test
  - MLPregressionModel → Very bad performance, not recommended

- encoder
  - resnet18-cellimage
  - vit-in21k
  - vit-in21k-noproj
    - Delete the projection head

- decoder
  - Using deconvolution based decoder
  - Linear is **not** recommended



loss.py

- class
  - Similarity_Loss → Cosine similarity for feature extraction
  - mutual_loss → Entropy based mutual loss for global and local feature
  - TotalCodingRate → Only for monitoring, avoiding collapsing

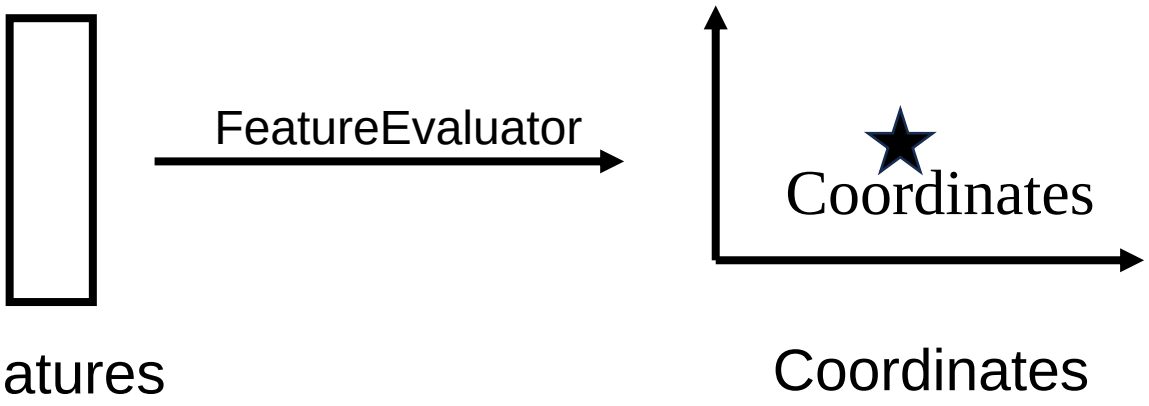
```
# Maximize similarity
return -z_sim, z_sim_out
```

```
scaled_mutual_info = torch.sigmoid(mutual_info)
return -scaled_mutual_info
```

“-”: Minimize the loss function, maximize similarity and mutual info

eval.py

- class
  - FeatureEvaluator → Dimensional reduction visualization



Features

Coordinates

Method for dimensional reduction
- t-SNE
- PCA
- UMAP

Quick Check scripts

```
# on_click for QuickCheck
def on_click(self, event, latent_space, sample_ids):
    if event.xdata is not None and event.ydata is not None:
        # Get the clicked point's index in the latent_space array
        distances = np.sqrt((latent_space[:, 0] - event.xdata) ** 2 +
                            (latent_space[:, 1] - event.ydata) ** 2)
        nearest_point_idx = np.argmin(distances)
        found_time = 0
        # Display the corresponding image
        if nearest_point_idx < len(latent_space):
            # Use nearest_point_idx to get the corresponding image index
            image_index = sample_ids[nearest_point_idx]
            print(type(image_index))
            self.display_image(image_index[0])
        else:
            found_time += 1
            print("{found_time}: Failed, please zoom in and try again!".format(found_time=found_time))
```

Click data point -> show original image

Usage:
When running CellOmicsV3test.py, add **–qc True**

Training, validation and test

- CellOmicsV2train.py
- CellOmicsV3train.py
- CellOmicsV3test.py
- CellOmicsV4train.py
- util.py

run_CellOmics.py
(A easy-use script)

```
parser.add_argument("--running_mode", type=str, default="train",
                    choices=["train", "test"],
                    help="choosing train or eval mode")
parser.add_argument("--model_version", type=str, default="V3",
                    choices=["V2", "V3", "V4"])
parser.add_argument("--gpus", type=str, default="0",
                    help='Used GPUs, grammer is same as linux command line')
parser.add_argument("--logname", type=str, default="training.log",
                    help="log file name")
```

- Details were written in scripts