Guide of Use for the Feature Cube framework

This is a guide for using the whole-slide image (WSI) classifier, based on the feature cube framework, located in the /vast/update_wsi_classifier/3class/, accessed by the Ubuntu-running servers 219.252.39.225 and 219.252.39.226.

The framework has 2 main modes: training and production. The training mode consists of training and testing the WSI classifier, while the production mode consists of applying the framework to a list of whole slides and saving the results in a database.

Requirements

```
numpy==1.21.4
tqdm==4.62.3
scikit-image==0.19.1
torch==1.10.1+cu113
torchvision==0.11.2+cu113
pymysql==1.0.2
pandas==1.3.5
pillow==7.0.0
```

Training

To train the WSI classifier simply run `python train_wsi_classifier.py` in the command line.

- The configuration file is located at `src/training/train_config.py`.
- The list of slides should be stored as a .csv file as follows:

file	label	mode	path
name of the file	N, D, or M	train, val, or test	path of the slide
2019S003993001	D	test	slides/stomach/original/D/2019S003993001.mrxs

- Samples codes demonstrating how to store the paths as shown are located at `save_stomach_slide_distribution.py` and `save_colon_slide_distribution.py`.
- The patch-level classifier is located at `lossdiff/merged_{x}_lossdif_balanced.pkl`, where `{x}` is either colon or stomach. These models are based on `torchvision.models.densenet201()`.

The file `train_wsi_classifier.py` runs 4 key functions:

- **extract_patches** (located in src/extract_patches) saves the patches extracted from `slide_distribution_path` to `tile_dir`.
    - **slide_distribution_path:** (str) path to the slide distribution file (.csv).
    - **tile_dir:** (str) directory in which the patches are saved.
    - **tile_size:** (int) size of the patches. The patches are meant to be squared (e.g., 256x256), therefore only one number is required.
    - **overlap**: (int) number of pixels to be overlapped.
    - **resolution_factor:** (int) zoom level in which the patches are going to be extracted.
    - **tile_ext:** (str) extension for the patches (e.g., jpg or png).
    - **limit_bounds:** (bool) True to render only the non-empty slide region.
    - **workers:** (int) number of workers for the multi-processed tiler.

- **generate_feature_cubes** (located in src/generate_feature_cubes) converts the slides (being patched) to feature cubes. The feature cubes are saved into three folders: `{feature_cube_path}/train`, `{feature_cube_path}/val`, and `{feature_cube_path}/test`.
    - **slide_distribution_path:** (str) path to the slide distribution file (.csv).
    - **patch_classifier_path:** (str) path to the patch classifier. By default the classifier is located at `lossdiff/merged_{x}_lossdif_balanced.pkl`, where `{x}` is either colon or stomach. These models are based on `torchvision.models.densenet201()`
    - **tile_dir:** (str) path to the patches.
    - **feature_cube_path**: (str) path where the feature cubes are saved.
    - **lsize:** (int) length of the feature cube.
    - **csize**: (int) channels (depth) of the feature cube.
    - **batch_size**: (int) number of instances in a training batch.
    - **tile_size:** (int) dimension of the patch.
    - **resolution_factor:** (int) zoom level in which the patches are going to be extracted.
    - **overlap**: (int) number of pixels to be overlapped.
    - **already_patched:** (bool) True if you wish to extract patches again. False by default.

- **train_classifier** (located in src/training/train_slide_classifier) ****trains a model using the feature cube approach. **This function returns** `torchvision.models.densenet201()`.
    - **diagnosis:** (list) List of classes. E.g., D, M, and N.
    - **model_path:** (str) path where the model is saved.

    - **data_path:** (str) path where the feature cubes are located.
    - **batch_size:** (int) number of instances in a training batch.
    - **num_epochs**: (int) number of epochs.
    - **lr:** (float) learning rate.
    - **dropout**: (bool) True if you wish to apply dropout.

- **test_classifier** (located in src.training/train_slide_classifier) tests the model once it is trained.
    - **model:** (`torchvision.models.densenet201()`) the model that was returned from **train_classifier**.
    - **diagnosis**: (list) list of classes. By default: [D, M, N].
    - **data_patch**: (str) path where the feature cubes are located.
    - **batch_size:** (int) number of instances in a test batch.

Production

To use the feature cube framework and save the results in a database simply run `python main.py` in the command line.

- The configuration file is located at `project_config.py`

- By default, the slides should be stored under a directory named `/test/`, this can be changed in `project_config.py`

```
EXAMPLE:
test/
|-- 2019S003993001
|-- 2019S003993002
|-- 2019S003993003
...
```

- The configuration of the database is located in `project_config.py`:

```
mariadb = {
    "host": "219.252.39.14",
    "user": "root",
    "password": "seegeneai2020",
    "db": "digital_pathology",
}
```

- The patch-level classifier is located at `lossdiff/merged_{x}_lossdif_balanced.pkl`, where `{x}` is either colon or stomach. Keep in mind that this model is based on

`torchvision.models.densenet201()`

- The slide-level classifier is located at `models/{x}/slide_classifier/{version}/slide_classifier.pt`, where `{x}` is either color or stomach, and `{version}` is the version of the model. The models are based on `torchvision.models.densenet201()`.

The file `main.py` consists of two main steps: step 2 and step 3. Step 1, binary classifier, has been discontinued.

- In step 2 the feature cubes are generated for a single whole slide using the patch classifier. The source code is located in `src/production/step2_patch_level_3class_classifier`.

- In step 3 the slide-level prediction is done by the slide classifier and saved in the database. The source code is located in `src/production/step3_slide_level_3class_classifier`.

- Step 2 and step 3 are run in a loop until all the slides in the dataset are assessed.

The following is a simplified pseudo-code of `main.py`

```
sql: "SELECT slide_name, slide_path FROM slides_queue WHERE slide_level_label
= %s AND anatomy =%s AND num_classes = %s order by date_time_added"
execute sql
slide_list: fetch sql

for slide_path in slide_list:
                # Extract patches from the slide
    save_patches(slide_path)

    # Generate feature cube
    step2(slide_path)

    # Get slide prediction
    step3(slide_path)

    # Remote directory with the patches
    shutil.rmtree(tile_dir)
```