Installation and Documentation Manual: Omics-PLIP Alignment

August 13, 2024

Contents

1 Introduction

This manual provides a comprehensive guide to recreate the analysis involving the alignment of a pathology vision foundation model with an existing omics knowledge base. The workflow includes the extraction of tissue tiles from whole slide images (WSI), preprocessing these tiles, splitting data for training, and finally extracting features for risk classification. The overall objective is to align the feature vectors of the PLIP model with omics features to enhance cancer research.

1.1 Overview of Steps

This manual covers the following steps:

1. **Extraction of Tissue Tiles**: Extracting tiles from whole slide images.
2. **Preprocessing of Tiles**: Color normalization and attachment of omics feature vectors to the images.
3. **Data Splitting**: Creating train, test, and validation sets.
4. **Feature Extraction**: Extracting features from the PLIP model for each tile.
5. **Risk Classification**: Using extracted features for classification and risk prediction.
6. **Dataset Creation for Benchmark Models**: Processing the primary and secondary cohorts to create datasets for benchmark models (ResNet50, InceptionV3, VGG16).
7. **Benchmark Against Existing Models**: Training and evaluating benchmark models (ResNet50, InceptionV3, VGG16) on the created datasets and comparing their performance.

1.2 Who Should Use This Manual?

This manual is designed for researchers and developers interested in the intersection of AI and biomedical imaging. It provides detailed steps to replicate the entire workflow from raw data to model alignment and feature extraction, useful for both academic and practical applications.

2 Installation

2.1 Python Package Setup

Note: It is recommended to use a Conda or Python virtual environment to avoid conflicts with existing packages.

2.1.1 Create a Separate Environment

```
# Using Conda
conda create --name omics-plip-env python=3.11
conda activate omics-plip-env

# Using Python's venv
python -m venv omics-plip-env
source omics-plip-env/bin/activate

# On Windows use:
omics-plip-env\Scripts\activate
```

2.1.2 Install Required Packages

```
pip install -r requirements.txt
```

3 Scripts and Workflow

3.1 Extract Tissue Tiles from WSI

This script extracts tissue tiles from whole slide images using the SlideProcessor class.

```
python3 extract_tiles_from_wsi.py -d wsi_data/ -o wsi_tiles/ -w 70
```

Arguments:

- **Input**: Whole slide images in the directory specified by `-d`.
- **Output**: Extracted tiles saved in the directory specified by `-o`.
- **Options**: `-w` sets the number of worker threads (default: 70).

3.2 Preprocess Tiles and Attach Omics Features

This script preprocesses the tiles and attaches omics feature vectors to the images.

```
python3 pre_process_tiles.py --csv_file ./data/standardized.p4.scores.csv --root_dir \
 ./wsi_tiles/ --save_dir ./pro_tiles
```

Arguments:

- **Input**: CSV file with omics data and extracted tiles from the previous step.
- **Output**: Preprocessed tiles with attached omics features, saved in the directory specified by `--save_dir`.

3.3 Create Train, Test, and Validation Sets

This script splits the preprocessed tiles into train, test, and validation sets.

```
python3 make_train_data_for_omics_plip.py --source_dir ./pro_tiles/ --dataset_dir \
 ./train_data/
```

Arguments:

- **Input**: Preprocessed tiles from the previous step.
- **Output**: Train, test, and validation sets, saved in the directory specified by `--dataset_dir`.

3.4 Train Omics-PLIP Model

This script trains the Genomic PLIP model using the prepared datasets.

```
python3 train_omics_plip_model.py --data_dir ./train_data/ --model_save_path \
 ./save_model/genomics_plip.pth --pretrained_model_path ./plip --num_workers 70
```

Arguments:

- **Input**: Train, validate, and test datasets.
- **Output**: Trained model saved in the path specified by `--model_save_path`.
- **Options**: Number of worker threads specified by `--num_workers`.

3.5 Extract Omics-Aligned Features

This script extracts features from the trained Omics-PLIP model for risk classification.

```
python3 extract_omics_aligned_tiles_features.py --data_dir ./pro_tiles/ --save_dir \
./omics_align_features/ --model_path ./save_model/genomics_plip.pth
```

Arguments :

- **Input**: Preprocessed tiles and trained Omics-PLIP model.
- **Output**: Extracted features saved in the directory specified by `--save_dir`.

3.6 Create Train Test Validation Split for Risk Classification

This script processes the metadata and extracted features to create datasets for risk classification. The data is split into training, testing, and validation sets, and the resulting data is saved as a compressed NumPy file for later use.

```
python3 make_train_data_for_risk_classification.py --data_dir ../../aug31/kali_extract/ \
 --metadata_file ./data/data1.hnsc.p3.csv --save_dir ./save_tr/
```

Arguments:

- **Input**:
  - **–data_dir**: Directory containing the extracted tissue features.
  - **–metadata_file**: CSV file containing metadata for the samples. This file should include a column named ‘Class‘ which is mapped to binary classes for risk classification.
- **Output**: The processed datasets, including the training, testing, and validation sets, are saved in a compressed NumPy file in the directory specified by `--save_dir`.

Notes:

- The script uses patient IDs from the metadata file to filter and organize the extracted features.
- Data is organized into bags, padded to ensure uniform length, and then split into train, test, and validation sets.
- The saved file can be loaded later using `numpy.load()` for model training and evaluation.

3.7 Train the Risk Classifier

This script trains a multiple instance learning (MIL) classifier on the preprocessed data. The model is designed to process bags of instances (e.g., tiles) and output a risk prediction.

```
python3 train_risk_classifier.py --data_file \
./save_tr/training_risk_classifier_data.npz --save_dir ./model_save/ --epochs 50
```

Arguments:

- **Input**:
  - **–data_file**: Path to the saved compressed NumPy file containing the training and validation datasets.
- **Output**:
  - **–save_dir**: Directory where the trained model will be saved. The best model during training will be saved as `risk_classifier_model.keras`, and the final model after all epochs will be saved as `risk_classifier_model.h5`.
- **Options**:
  - **–epochs**: Number of training epochs (default: 50).

Notes:

- The model uses a TimeDistributed layer to process each instance in the bag individually, followed by a GlobalAveragePooling layer to aggregate the information across the instances.
- Early stopping and learning rate scheduling are implemented to optimize the training process.
- Class weights are computed to address any imbalance in the dataset.

3.8 Evaluate the Risk Classifier

This script evaluates the trained multiple instance learning (MIL) classifier on the training, validation, and test datasets. It computes various performance metrics and saves the results for further analysis.

```
python3 evaluate_risk_classifier.py --data_file ./save_tr/training_risk_classifier_data.npz \
--model_path ./model_save/risk_classifier_model.h5 --save_dir ./evaluation_results/
```

Arguments:

- **Input**:
  - **–data_file**: Path to the saved compressed NumPy file containing the training, validation, and test datasets.
  - **–model_path**: Path to the saved trained model file (e.g., `risk_classifier_model.h5`).
- **Output**:
  - **–save_dir**: Directory where the evaluation results will be saved. This includes the predictions for each dataset and a JSON file containing the evaluation metrics.

Metrics:

- **Loss**: Binary cross-entropy loss on the dataset.
- **Accuracy**: The proportion of correct predictions.
- **AUC**: Area under the receiver operating characteristic (ROC) curve.
- **Precision**: The ratio of true positive predictions to the total predicted positives.
- **Recall**: The ratio of true positive predictions to the total actual positives.
- **F1 Score**: The harmonic mean of precision and recall.

Notes:

- The script evaluates the model on training, validation, and test datasets separately.
- Evaluation results are displayed in a tabular format and saved in a JSON file named `evaluation_metrics.json`.
- The script also saves the raw predictions for each dataset, allowing for further analysis such as generating ROC curves.

3.9 Evaluate the Risk Classifier on External Test Cohort

This script evaluates the trained multiple instance learning (MIL) classifier on an external test cohort (referred to as ‘test2‘). The script handles preprocessing of the external cohort, including loading metadata, creating bags of tiles, padding, and evaluating the model.

```
python3 evaluate_on_test_cohort2.py --metadata_file ./data/data2.hnsc.p3.csv --data_dir \
../../aug31/kali_extract/ --model_path ./save_risk_model/risk_classifier_model.h5 --save_dir
./evaluation_results/
```

Arguments:

- **Input**:
  - **–metadata_file**: Path to the CSV file containing metadata for the external test cohort (‘test2‘).
  - **–data_dir**: Directory containing the extracted tissue features for the external test cohort.
  - **–model_path**: Path to the saved trained model file (e.g., `risk_classifier_model.h5`).
- **Output**:
  - **–save_dir**: Directory where the evaluation results will be saved. This includes the predictions for ‘test2‘ and a JSON file containing the evaluation metrics.

Metrics:

- **Loss**: Binary cross-entropy loss on the ‘test2‘ dataset.
- **Accuracy**: The proportion of correct predictions.
- **AUC**: Area under the receiver operating characteristic (ROC) curve.
- **Precision**: The ratio of true positive predictions to the total predicted positives.
- **Recall**: The ratio of true positive predictions to the total actual positives.

- **F1 Score**: The harmonic mean of precision and recall.

Notes:

- The script processes the external cohort by loading the metadata, creating bags of tiles, padding the data, and finally evaluating the model on the 'test2' dataset.
- Evaluation results are saved in a JSON file named `evaluation_metrics_test2.json`.
- The script also saves the raw predictions for 'test2', allowing for further analysis.

3.10 Create Dataset for Benchmark Models

This script processes the primary and secondary cohorts to create a dataset for training and evaluating benchmark models like ResNet50, InceptionV3, and VGG16. The script takes metadata CSV files for both cohorts and the location of raw tissue tiles to create train, test, validate, and test2 datasets. The generated dataset is saved in the specified directory.

```
python3 make_dataset_for_benchmark_models.py --primary_metadata ./data/data1.hnsc.p3.csv \
--secondary_metadata ./data/data2.hnsc.p3.csv --source_dir ../../Dataset/tiles_1024/ \
--dataset_dir ./benchmark_data/ --num_tiles_per_patient 595
```

Arguments:

- **Input**:
  - **–primary_metadata**: Path to the metadata CSV file for the primary cohort.
  - **–secondary_metadata**: Path to the metadata CSV file for the secondary cohort.
  - **–source_dir**: Directory containing the raw tissue tiles, organized by patient ID.
- **Output**:
  - **–dataset_dir**: Directory where the train, test, validate, and test2 datasets will be saved.
- **Options**:
  - **–num_tiles_per_patient**: Number of tiles to select per patient (default: 595).

Notes:

- The script processes both the primary and secondary cohorts to create the final dataset used for training and evaluating benchmark models.
- The resulting dataset will have four folders: `train`, `test`, `validate`, and `test2`.

3.11 Benchmark Against Existing Models

This section describes the training and evaluation of benchmark models (ResNet50, InceptionV3, and VGG16) on the dataset created in the previous step. The scripts are designed to train the models on the generated datasets and save evaluation metrics.

3.11.1 Train and Evaluate ResNet50

This script trains and evaluates a ResNet50 model on the benchmark dataset. The model is trained using the training and validation datasets and evaluated on the test and test2 datasets.

```
python3 benchmark_train_resnet50.py --dataset_dir ./benchmark_data/ --save_dir \
./benchmark_results/ --epochs 10
```

Arguments:

- **Input**:
  - **–dataset_dir**: Directory containing the train, validate, test, and test2 datasets.
- **Output**:
  - **–save_dir**: Directory where the trained model and evaluation results will be saved.
- **Options**:
  - **–epochs**: Number of training epochs (default: 10).

Notes:

- The script saves the training logs and evaluation metrics for each dataset in the specified save directory.

3.11.2 Train and Evaluate InceptionV3

This script trains and evaluates an InceptionV3 model on the benchmark dataset. Similar to the ResNet50 script, the model is trained on the train and validate datasets and evaluated on the test and test2 datasets.

```
python3 benchmark_train_inception.py --dataset_dir ./benchmark_data/ --save_dir \
./benchmark_results/ --epochs 10
```

Arguments:

- **Input**:
  - **–dataset_dir**: Directory containing the train, validate, test, and test2 datasets.
- **Output**:
  - **–save_dir**: Directory where the trained model and evaluation results will be saved.
- **Options**:
  - **–epochs**: Number of training epochs (default: 10).

3.11.3 Train and Evaluate VGG16

This script trains and evaluates a VGG16 model on the benchmark dataset. Like the other models, VGG16 is trained on the train and validate datasets and evaluated on the test and test2 datasets.

```
python3 benchmark_train_vgg16.py --dataset_dir ./benchmark_data/ --save_dir \
./benchmark_results/ --epochs 10
```

Arguments:

- **Input**:
  - **–dataset_dir**: Directory containing the train, validate, test, and test2 datasets.
- **Output**:
  - **–save_dir**: Directory where the trained model and evaluation results will be saved.
- **Options**:
  - **–epochs**: Number of training epochs (default: 10).

Notes:

- Each script saves the training logs and evaluation metrics for the respective model in the specified save directory.
- The models are evaluated on both the test and external test (test2) datasets, with metrics saved for further analysis.

4 Limitations

1. The process might be time-consuming depending on the size of the whole slide images and the number of tiles.
2. Some scripts require high computational resources and might not run efficiently on systems with low processing power.

5 Computational Resources

- Source Code Repository: https://github.com/VatsalPatel18/Omics-PLIP