SRSML24: STM Machine Learning Module

Steven R. Schofield

April 20, 2025

Overview

This module provides tools for machine learning analysis of scanning tunnelling microscopy (STM) data, including autoencoder models, clustering tools, and STM-specific preprocessing.

Getting the Code

Clone the repository from GitHub:

```
git clone https://github.com/srschofield/SRSML24.git
```

Installation

It is recommended to create a clean Python environment using `conda`. The following steps assume you are working on a macOS system with Apple Silicon:

```
# create and activate environment
conda create --name srsml24 python=3.8 -y
conda activate srsml24

# install packages
pip install -r requirements-macos.txt
```

Known Working Configuration

This module has been tested and is known to work with the following configuration on macOS 15.0.1 (Apple Silicon, M3 Pro chip):

Package	Version
python	3.8
tensorflow-macos	2.13.0
tensorflow-metal	1.0.1
numpy	1.24.3
pandas	2.0.3
matplotlib	3.7.5
scikit-learn	1.3.2
scipy	1.10.1
opencv-python	4.11.0.86
Pillow	10.4.0
joblib	1.4.2
jupyter	1.1.1
ipykernel	6.29.5
keras-core	0.1.5
spiepy	0.2.1
access2thematrix	0.4.4

Table 1: Verified package versions for macOS (Apple Silicon) environment

These packages can be installed using the `requirements-macos.txt` file. The Python version is critical: other versions may cause compatibility issues with TensorFlow or other packages on Apple Silicon.

Python Files

- `data_prep.py` – Functions for data preparation, including slicing STM images into windows and saving them in efficient formats.
- `model.py` – Defines convolutional autoencoder and UNET-style models.
- `utils.py` – Utility functions for loading/saving models, feature arrays, and results.

License

Parameter Summary

Parameter	Description
General	
`job_name`	Label for the run, it will be the folder name for output.
`verbose`	If `True`, enables more detailed print output.
Matrix data file processing	
`flatten_method`	Method used to flatten STM images before analysis. Options are 'none', 'iterate_mask', 'poly_xy'.
`pixel_density`	All images will be converted to this pixel density (px/nm).
`pixel_ratio`	Images that have ratio of fast/slow scan direction less than this will be discared. Setting to 1 means only complete (square) images are kept.
`data_scaling`	Multiplicative factor for z-height data. Setting to 1.e9 means that the range 0–1 (used for training) corresponds to 1 nm.
Window generation	
`window_size`	Side length of square image windows (in pixels).
`window_pitch`	Spacing between adjacent windows during tiling.
Data saving	
(Should remain defaults but options can be useful for examining data manually.)	
`save_windows`	If `True`, saves image windows as `.npy` files (True).
`together`	If `True`, saves windows per image in a single file (True).
`save_jpg`	If `True`, saves full STM images as JPGs (False).
`collate`	If `True`, flattens directory structure into one folder. (False).
`save_window_jpgs`	If `True`, saves image windows as JPGs. (False)
Autoencoder	
`model_name`	Label used to save and load the trained autoencoder model.
`batch_size`	Number of windows per training batch.
`buffer_size`	Size of shuffle buffer.
`learning_rate`	Learning rate for the optimizer.
`epochs`	Number of training epochs.
Clustering	
`cluster_model_name`	Name used when saving the clustering model.
`cluster_batch_size`	Number of latent vectors per clustering batch.
`cluster_buffer_size`	Size of buffer for clustering shuffle.
`num_clusters`	Number of clusters to form using KMeans.
`n_init`	Number of initializations for KMeans.
`max_iter`	Max iterations for KMeans convergence.
`reassignment_ratio`	Fraction of centroids reassigned each step.
Image prediction	
`predict_window_pitch`	Window spacing during prediction step.
`mtrx_train_data_limit`	Max number of training MTRX files to use.
`mtrx_test_data_limit`	Max number of validation MTRX files to use.
`train_data_limit`	Limit on number of training windows.
`test_data_limit`	Limit on number of validation windows.