**Documentation for Twin Attention Pipeline**

**T**he ‘data’ subfolder contains example data that can be used as inference templates. The directory ‘txt\_file\_by\_emb’ contains input data in raw text pert timepoint format and ‘data\_final’ is the same files preprocessed onto pkl files as used by main inference code.

In the ‘model’ subfolder contains final versions of paper models, ‘late\_embryo\_final.bin’ is the final trained model for inference (matching example data which is drawn from fine tuning experiment 2).

The ‘output’ subfolder is for storing the final output results.

The ‘src’ subfolder contains the code for training process (‘train.py’ is for model training and it uses the model in ‘model.py’)

The ‘src\_preprocessing’ directory contains the code for data preprocessing (‘embryo\_IDs\_as\_ints\_final.pkl’ contains a dictionary of cell names and integer numbers used to represent identities in code. The file ‘main\_convert.py’ is used to transform the files in ‘txt\_file\_by\_emb’ to the files in ‘data \_final’, and it uses the functions in ‘converter\_functions.py’).

Main directory code

The main directory contains a script that demonstrates inference ‘simple\_ inference\_script.py’ script automates pairwise matching. It assumes txt format input are in test and data subfolders, converts both and runs inference which will pick a stage appropriate embryo and match.

The file ‘pair\_inference.py’ can search for a suitable template embryo from data/data\_final for each of the test embryos, then output the inference results in the ‘output’ folder. This is what is called by simple\_inference\_script.py

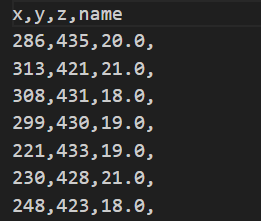
‘inference.py’ contains code for doing inference, not to generate matching but to generate embeddings for cells to be used in KNN classification identification or other tasks. ‘parallel\_inference.py’ is identical but parallelized.

**Usage**

**Inference Pairwise Matching:**

Inference requires comparison data in pkl format, some example data is provide for convenience in data directory.

For quickly using the code to name unknown cells: First, you should input your own unnamed embryo files into ./data/txt\_file\_by\_emb/test (one folder represents one worm. one .txt file represents one embryo at a specific time point, should have ‘x’, ‘y’, ‘z’, ‘name’ columns and ‘name’ columns can be empty; as the figure follows). Second, make sure that ./data/txt\_file\_by\_emb/template has enough templates that have similar cell numbers to your test files. Third, run ‘inference\_script.py’ script (before running, please modify ‘--micron\_scalars’ which refer to the x, y, z micrometer resolution to match your data), the results will be stored in the ‘output’ folder.



**Inference Using Learned Manifold:**

This requires a manifold, a body of named embeddings created with the same trained model, as well as an unknown dataset. A user friendly API for this does not exist, but this task is implemented by code in

The file ‘inference \_parallel.py’ generate a parquet file of named and unnamed embeddings from directories of preprocessed data in hard coded directories within the data directory. It will then create parquet files of the results e.g. ‘metaPC\_embeddings\_new.parquet’ and ‘test\_embeddings\_new.parquet’, which contain the embeddings. This uses the functions in ‘DNC\_pred\_get\_embeddings\_clean\_sept24.py’ and ‘get\_encodings\_model\_clean\_sept24.py’.

KNN classification is performed with standard sklearn functions against this parquet file. A general implementation is not included but a simple demonstration of using the parquet file to do classification is in bin\_gen\_knn.py.

**Training:**

Example training command for training and test data already preprocessed into expected format

nohup python /src/train.py --use\_pretrain 0 --batch\_size 10 --n\_layer 6 --n\_hidden 128 --patience 10 --max\_num\_trial 20 --lr\_decay 0.25 --train\_path [datapath]/train/batched/ --eval\_path [datapath]/eval/batched/ --model\_path ../trainedmodel/real\_retrained\_late\_zhuo\_data\_model > train.txt 2>&1 &

Note both directories are meant to point to data preprocessed and arranged into training batches, see **Preprocessing** section.

Example training command starting with a pretrained model:

nohup python /src/train.py --use\_pretrain 1 --batch\_size 10 --n\_layer 6 --n\_hidden 128 --patience 10 --max\_num\_trial 20 --lr\_decay 0.25 --train\_path [datapath]/train/batched/ --eval\_path [datapath]/eval/batched/ --model\_path ../trainedmodel/real\_retrained\_late\_zhuo\_data\_model > train.txt 2>&1 &

(note there is no command line argument for pretrained location, just to use one, so if using pretrained model, remember to change the variable ‘pretrain\_path’ in src/train.py. which currently points to most recent model)

**Preprocessing:**

**Acetree Format embryos:** sort embryos into train, eval directories. Unzip each zip file, rename the 'nuclei' folders as embryo names, remove the 0-flagged rows, and add a header (resulting txt files sorted into 'by\_emb' and 'by\_stage' organizations)

main\_convert.py then applies zero-centering, micron scaling, random rotation, etc. to a set of nuclei folders, outputs to stage-specific folders in .npy (Transformer training) or .pkl (inference) formats

combine\_db\_wval\_best\_depth.py Applies final agreed-upon sliding window sampling, puts into folders of specified size (needed only for training)

**Compiled single txt file Acetree Results:**The file ‘main\_convert.py’ is used to transform single file per embryo compiled acetree files into data files filtered by editing end times listed in a csv file of editing end tmes The file ‘combine\_db\_wval\_best\_depth.py’ is used to transform these files (saved in ‘npy\_format’ folder) into training batches in ‘batched’ folder as above.