Initial Notes:

Define the correct paths in the various scripts for saving the dataset (`dataPATH`) and the results (`resultsPATH`):

- **dataPATH**: Directory where the MNIST dataset will be stored.
- **resultsPATH**: Directory where experiment results will be saved. This must include the following subfolders:
  - `clustering`
  - `collapse`
  - `coloring`
  - `plots`
  - `training`

Script Descriptions:

==========

- **coloring.py**: Contains functions for computing fibrations and opfibrations in linear layers.
- **model.py**: Defines the MLP model (a 3-hidden-layer neural network). The `MLP` class (a `nn.Module`) includes methods for computing fibrations, opfibrations, and coverings using the network's current parameters.

  ==========

- **training.py**: Script for training the MLP model on the MNIST classification task.
  - In the `training` subfolder of `resultsPATH`, the model weights are saved throughout training as `weight_batch_bb.pth` (where `bb = 0, ..., 599`).
  - Training performance (accuracy) is saved in `accuracy.pth` (a 1D tensor with 600 elements).
- **generate_activity.py**: Generates node activity using:
  1. Samples from the evaluation subset of the dataset (10,000 samples).
  2. Randomly generated samples (200 samples).

  This is computed after the 599 training steps. The results are saved in:
  - `activity_batch_599.pth`
  - `activity_random_input_batch_599.pth`

  ==========

- **fibration_coloring.py**: Computes fibrations for the MLP model at different training steps (bb = 0, ..., 599).
  A **`threshold`** parameter determines fibrations:
  - `threshold = 0` → Each node is in its own fiber (no symmetries).
  - `threshold = 2` → All nodes are in a single large fiber (fully symmetric).

Results are stored in `coloring/fibration_batch_bb.pth` for different thresholds (0, 0.01, ..., 1.00).

- **synchronization_clusters.py:** Computes synchronization clusters per layer based on network activity from random inputs (see `generate_activity.py`).
  Clustering requires an `epsilon` parameter (100 values between `0` and `30`). Note: `epsilon` depends on the network's activity distribution. (*Work in progress: Normalization of `epsilon`.*)
  Results are saved in `clustering/clusters_batch_599.pth` for the network at step `bb = 599` across different `epsilon` values.

  ==========

- **collapse_during_training.py**, **collapse_post_training.py**, **plt_during_training.py**, **plt_post_training.py:** These scripts compare the performance of the original model against collapsed models (using fibration symmetries) at different training stages and thresholds. They also evaluate the reduction in network size.

  ==========

- **matching.py:** Compares synchronization clusters and fibrations, establishing a relationship between `epsilon` (synchronization) and `threshold` (fibration).