MTIL Arguments Guide

This document provides a comprehensive reference for all command-line arguments available in the MTIL (Multi-Task Incremental Learning) training pipeline.

Table of Contents

Testing

Argument	Type	Default	Description
`--test`	flag	`False`	Enable testing mode

Hyperparameters

Argument	Type	Default	Description
`--model`	str	`ViT-B/16`	CLIP model architecture. Options: `ViT-B/16`, `ViT-B/32`, `ViT-L/14`, `RN50`, `RN101`
`--batch-size`	int	`8`	Training batch size
`--batch-size-eval`	int	`32`	Evaluation batch size
`--lr`	float	`0.001`	Learning rate
`--wd`	float	`0.0`	Weight decay for optimizer

Argument	Type	Default	Description
`--ls`	float	`0.0`	Label smoothing factor
`--warmup_length`	int	`100`	Number of warmup iterations for learning rate scheduler
`--beta2`	float	`0.999`	Beta2 parameter for Adam optimizer

Logging & Training Control

Argument	Type	Default	Description
`--seed`	int	`42`	Random seed for reproducibility
`--epochs`	int	`None`	Number of training epochs (mutually exclusive with `--iterations`)
`--iterations`	int	`None`	Number of training iterations (mutually exclusive with `--epochs`)
`--eval-interval`	int	`None`	Evaluate every N iterations (mutually exclusive with `--eval-every-epoch`)
`--loss-interval`	int	`1000`	Log loss every N iterations
`--eval-every-epoch`	flag	`False`	Evaluate at the end of each epoch
`--eval-only`	flag	`False`	Skip training, only run evaluation
`--save-eval`	flag	`False`	Save evaluation results
`--start-iteration`	int	`None`	Starting iteration (for resuming training)

Experiment Settings

Argument	Type	Default	Description
`--method`	str	`finetune`	Training method. **Choices:** `finetune`, `lwf`, `ZSCL`, `icarl`
`--train-mode`	str	`whole`	Which parts of the model to train. **Choices:** `whole`, `text`, `image`, `image-fc`, `image-fc-fixed`, `fc`
`--data-location`	str	`./data`	Root directory for datasets
`--train-dataset`	str	`None`	Dataset to train on (e.g., `DTD`, `CIFAR100`, `ImageNet`)

Argument	Type	Default	Description
`--eval-datasets`	str	`None`	Comma-separated list of datasets for evaluation
`--text-datasets`	str	`None`	Comma-separated list of datasets for text encoder evaluation
`--template`	str	`None`	Prompt template to use for zero-shot classification

Method Descriptions

- `finetune`: Standard fine-tuning of the model
- `lwf`: Learning without Forgetting - uses distillation from previous model
- `ZSCL`: Zero-Shot Continual Learning - regularizes with reference data/text
- `icarl`: Incremental Classifier and Representation Learning - uses exemplar memory

Train Mode Descriptions

- `whole`: Train the entire model (both encoders)
- `text`: Train only the text encoder
- `image`: Train only the image encoder
- `image-fc`: Train image encoder and classification head
- `image-fc-fixed`: Train image encoder with fixed classification head
- `fc`: Train only the classification head (linear probe)

Single Image Evaluation

Argument	Type	Default	Description
`--eval-single`	str	`None`	Path to a single image for evaluation
`--prompt`	str	`None`	Custom prompt for single image evaluation
`--class-names`	str	`None`	Path to file containing class names (one per line)

Example:

```
python -m src.main \
    --load ckpt/model.pth \
    --eval-single path/to/image.jpg \
    --class-names data/text_classes/imagenet_classes.txt \
    --eval-only
```

Save & Load

Argument	Type	Default	Description

Argument	Type	Default	Description
`--save`	str	`None`	Directory path to save checkpoints
`--load`	str	`None`	Path to checkpoint file to load
`--load_federate`	str	`None`	Comma-separated paths for federated model loading

Model Control (image-fc branch)

Argument	Type	Default	Description
`--fair`	flag	`False`	Enable fair comparison mode
`--we`	flag	`False`	Enable weight ensembling
`--we_wise`	flag	`False`	Enable WiSE weight ensembling
`--we_wise_alpha`	float	`0.98`	Alpha for WiSE weight ensembling
`--moving_avg`	flag	`False`	Use exponential moving average of weights
`--avg_freq`	int	`100`	Frequency of moving average updates
`--mv_avg_decay`	float	`0.999`	Decay rate for moving average
`--mv_avg_model`	str	`n`	Base model for moving average. **Choices:** `n`, `t`, `zeroshot`
`--l2`	float	`0`	L2 regularization strength toward reference model
`--fc-init`	flag	`False`	Reinitialize the classification head
`--fc-setnone`	flag	`False`	Set classification head to None
`--dataset-shift`	flag	`False`	Enable dataset shift mode
`--n_class`	int	`10`	Number of classes for classification head

ZSCL (Zero-Shot Continual Learning)

Argument	Type	Default	Description
`--ref-dataset`	str	`None`	Reference dataset for regularization (e.g., `ImageNet`)
`--ref-sentences`	str	`None`	Reference text embeddings (e.g., `conceptual_captions`)
`--ref-model`	str	`None`	Path to reference model checkpoint
`--ref-wise`	flag	`False`	Use WiSE-FT for reference model
`--ref_wise_alpha`	float	`0.8`	Alpha for reference WiSE-FT
`--T`	float	`2.0`	Temperature for distillation loss
`--num`	float	`64`	Number of reference samples per batch

Example (ZSCL training):

```
python -m src.main \
    --method ZSCL \
    --train-dataset DTD \
    --ref-dataset ImageNet \
    --ref-sentences conceptual_captions \
    --T 2.0 \
    --num 64
```

iCaRL

Argument	Type	Default	Description
`--dataset_order`	str	`None`	Comma-separated list of datasets defining task order
`--memory_size`	int	`10000`	Total number of exemplars to store in memory

Example:

```
python -m src.main \
    --method icarl \
    --dataset_order "CIFAR10,CIFAR100,DTD" \
    --memory_size 2000
```

Loss Functions

Argument	Type	Default	Description
`--weight_adjust`	flag	`False`	Enable weight adjustment for loss
`--feature_mse`	flag	`False`	Add MSE loss on features
`--image_loss`	flag	`False`	Enable image encoder loss
`--text_loss`	flag	`False`	Enable text encoder loss
`--ablation_loss_2`	flag	`False`	Enable second ablation loss variant

WiSE-FT (Weight-Space Ensembling)

Argument	Type	Default	Description
`--wise_merge`	flag	`False`	Use WiSE merging during training
`--wise_ft`	flag	`False`	Use WiSE-FT during evaluation
`--wise_ft_model`	str	`n`	Model to ensemble with. **Choices:** `n`, `zeroshot`

Argument	Type	Default	Description
`--wise_ft_alpha`	float	`0.8`	Interpolation factor (0=finetuned, 1=reference)
`--wise-ft`	flag	`False`	Alternative flag for WiSE-FT
`--alpha`	float	`0.5`	Alpha for `--wise-ft` flag

WiSE-FT Formula:

```
final_weights = alpha * finetuned_weights + (1 - alpha) * zeroshot_weights
```

Experiment Organization

Argument	Type	Default	Description
`--exp_name`	str	`None`	Name of the experiment for organization
`--results-db`	str	`results.jsonl`	Path to JSONL file for storing results
`--cache-dir`	str	`None`	Directory for caching features and encoders

Model Freezing

Argument	Type	Default	Description
`--freeze-encoder`	flag	`False`	Freeze the image encoder (for linear probing)
`--freeze-fc`	int	`0`	Number of FC layers to freeze

Learning Without Forgetting (LwF)

Argument	Type	Default	Description
`--lwf`	flag	`False`	Enable LwF distillation loss
`--basic_model_load`	str	`None`	Comma-separated paths to load base classifiers
`--fc_load`	str	`None`	Comma-separated paths to load FC layers
`--keep_old_heads`	int	`0`	Number of old classification heads to retain

Baseline & TRIO

Argument	Type	Default	Description
`--baseline`	flag	`False`	Run baseline experiment
`--trio`	flag	`False`	Enable TRIO method

Argument	Type	Default	Description
`--control-dataset`	str	`None`	Control dataset for TRIO
`--control-dataset-add`	str	`None`	Additional control dataset
`noise`	positional	-	Use random noise for regularization
`--rff`	flag	`False`	Enable random Fourier features

Fisher Weighting

Argument	Type	Default	Description
`--fisher`	str	`None`	Comma-separated paths to Fisher information matrices
`--fisher_floor`	float	`1e-8`	Minimum value for Fisher weights (numerical stability)

Thesis-Specific Arguments

Argument	Type	Default	Description
`--freeze`	flag	`False`	Freeze model parameters
`--mixup`	int	`None`	Enable mixup augmentation with specified alpha
`--orthogonal-gradients`	int	`None`	Number of orthogonal gradient projections
`--orthogonal-gradients-path`	str	`None`	Path(s) to orthogonal gradient basis (multiple allowed)
`--untrained`	flag	`False`	Use untrained (random) model weights
`--custom-finetune`	flag	`False`	Enable custom fine-tuning procedure
`--max-evaluation-size`	int	`None`	Limit evaluation dataset size

LoRA (Low-Rank Adaptation)

LoRA enables parameter-efficient fine-tuning by freezing the base CLIP model and training small low-rank adapter layers. This significantly reduces memory usage and training time while maintaining good performance.

Argument	Type	Default	Description
`--lora`	flag	`False`	Enable LoRA training. Freezes base model and trains only LoRA adapter layers.
`--lora-r`	int	`8`	LoRA rank (dimension of low-rank matrices). Higher = more capacity but more parameters.
`--lora-alpha`	int	`16`	LoRA scaling factor. Effective scaling is `alpha/r`.

Argument	Type	Default	Description
`--lora-dropout`	float	`0.1`	Dropout probability applied to LoRA layers.
`--lora-target-modules`	str	`None`	Comma-separated module names to apply LoRA. Default: attention and MLP layers.
`--lora-bias`	str	`none`	Which biases to train. **Choices:** `none`, `all`, `lora_only`

CLIP target modules

- Attention only: attn.in_proj_weight,attn.out_proj.weight
- MLP only: mlp.c_fc.weight,mlp.c_proj.weight

LoRA Concepts

- **Rank (`r`)**: Controls the capacity of LoRA adapters. Typical values: 4, 8, 16, 32. Lower rank = fewer parameters but potentially less expressive.
- **Alpha**: Scaling factor for LoRA updates. The effective learning rate scaling is `alpha/r`.
- **Target Modules**: By default, LoRA is applied to attention projections and MLP layers in both visual and text encoders.

Requirements

LoRA requires the `peft` library:

```
pip install peft
```

Common Usage Examples

Basic Fine-tuning

```
python -m src.main \
    --method finetune \
    --train-mode whole \
    --train-dataset DTD \
    --eval-datasets DTD,ImageNet \
    --iterations 1000 \
    --lr 1e-5 \
    --batch-size 32 \
    --save ckpt/dtd_finetune
```

ZSCL with Reference Data

```
python -m src.main \
    --method ZSCL \
    --train-mode whole \
    --train-dataset DTD \
    --ref-dataset ImageNet \
    --ref-sentences conceptual_captions \
    --iterations 1000 \
    --lr 1e-5 \
    --T 2.0 \
    --save ckpt/dtd_zscl
```

Evaluation Only

```
python -m src.main \
    --load ckpt/model.pth \
    --eval-datasets DTD,CIFAR100,ImageNet \
    --eval-only
```

WiSE-FT Evaluation

```
python -m src.main \
    --load ckpt/model.pth \
    --eval-datasets ImageNet \
    --eval-only \
    --wise_ft \
    --wise_ft_alpha 0.5
```

Linear Probe (FC only)

```
python -m src.main \
    --method finetune \
    --train-mode fc \
    --freeze-encoder \
    --train-dataset CIFAR100 \
    --n_class 100 \
    --iterations 5000 \
    --save ckpt/cifar100_probe
```

LoRA Fine-tuning

```
# Basic LoRA training (default settings)
python -m src.main \
    --method finetune \
    --train-mode whole \
    --lora \
    --train-dataset DTD \
    --iterations 1000 \
    --lr 1e-4 \
    --save ckpt/dtd_lora

# LoRA with custom configuration
python -m src.main \
    --method finetune \
    --train-mode whole \
    --lora \
    --lora-r 16 \
    --lora-alpha 32 \
    --lora-dropout 0.05 \
    --train-dataset DTD \
    --iterations 1000 \
    --lr 1e-4 \
    --save ckpt/dtd_lora_r16

# LoRA with ZSCL (continual learning)
python -m src.main \
    --method ZSCL \
    --train-mode whole \
    --lora \
    --lora-r 8 \
    --train-dataset DTD \
    --ref-dataset ImageNet \
    --iterations 1000 \
    --save ckpt/dtd_lora_zscl
```

Notes

- `--epochs` and `--iterations` are mutually exclusive; use one or the other
- `--eval-interval` and `--eval-every-epoch` are mutually exclusive
- Device is automatically set to CUDA if available, otherwise CPU
- For ZSCL, both `--ref-dataset` and `--ref-sentences` are typically used together for best results