

InfoCLIP++ Implementation: Usage Guide

1. Environment Setup

1.1 Prerequisites

Python 3.8–3.10 (compatible with PyTorch 1.13.1 and dependencies)

CUDA 11.3–11.7 (required for GPU acceleration; CPU-only training is unsupported for full pre-training)

Minimum 16GB GPU memory (for batch size 32; reduce batch size in config.json for smaller GPUs)

Internet connection (to download pre-trained backbones: DINOv2, BERT)

1.2 Installation

Clone the repository (if applicable)

```
git clone <repository-url>
```

```
cd infoclip-plus-plus
```

Install required dependencies

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

2. Dataset Preparation

2.1 Required Datasets

ImageNet-1K: For zero-shot classification task

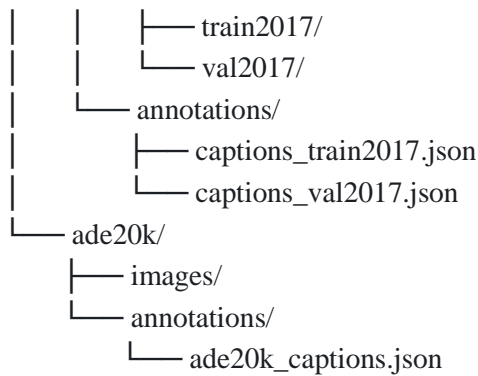
COCO 2017: For cross-modal retrieval task

ADE20K: For fine-grained alignment task

2.2 Dataset Structure

Organize your datasets as follows:

```
datasets/
├── imagenet/
│   ├── train/
│   │   ├── n01440764/
│   │   └── ... (other classes)
│   └── val/
│       ├── n01440764/
│       └── ... (other classes)
├── coco/
└── images/
```



2.3 Configuration File Update

Modify `config.json` to point to your dataset locations:

```
"imagenet_root": "/path/to/your/datasets/imagenet",
"coco_root": "/path/to/your/datasets/coco/images",
"coco_train_ann": "/path/to/your/datasets/coco/annotations/captions_train2017.json",
"coco_val_ann": "/path/to/your/datasets/coco/annotations/captions_val2017.json",
"ade20k_root": "/path/to/your/datasets/ade20k/images",
"ade20k_ann": "/path/to/your/datasets/ade20k/annotations/ade20k_captions.json"
```

3. Training the Model

3.1 Basic Training Command

```
python trainer.py --config config.json
```

3.2 Training with Custom Parameters

You can override configuration parameters via command line:

```
python trainer.py --config config.json \
    --batch_size 16 \
    --num_epochs 50 \
    --lr 5e-4 \
    --save_dir ./custom_results
```

3.3 Training Monitoring

Checkpoints are saved in the `save_dir` directory

Training logs are written to `training.log` in the save directory

Loss curves and metrics are saved as PNG files

4. Evaluating the Model

4.1 Evaluating a Trained Model

To evaluate a trained model on all three tasks:

```
python trainer.py --config config.json \  
    --eval-only \  
    --resume ./results/checkpoint_best.pth
```

4.2 Task-Specific Evaluation

To evaluate on specific tasks only:

Evaluate only on ImageNet zero-shot classification

```
python trainer.py --config config.json \  
    --eval-only \  
    --resume ./results/checkpoint_best.pth \  
    --tasks imagenet
```

Evaluate on COCO and ADE20K

```
python trainer.py --config config.json \  
    --eval-only \  
    --resume ./results/checkpoint_best.pth \  
    --tasks coco ade20k
```

5. Understanding the Output

5.1 Training Output

Periodic loss values for each component of the unified loss

Learning rate updates

Validation metrics after each epoch

Checkpoint saves when validation performance improves

5.2 Evaluation Output

ImageNet-1K: Top-1 and Top-5 accuracy for zero-shot classification

COCO: Recall@1, Recall@5, Recall@10 for image-to-text and text-to-image retrieval

ADE20K: Fine-grained alignment metrics including HGR correlation scores and alignment error

5.3 Saved Results

Trained model checkpoints

Evaluation metrics in JSON format

Visualization of alignment results (for ADE20K)

Training curves for loss and key metrics

6. Advanced Options

6.1 Hardware Configuration

Force CPU training (not recommended for full training):

```
python trainer.py --config config.json --device cpu
```

Specify specific GPU:

```
python trainer.py --config config.json --device cuda:1
```

6.2 Hyperparameter Tuning

Adjust key hyperparameters in `config.json` or via command line:

Window size for pixel-level OT (`window_size`)

Rank parameter for low-rank approximation (`r`)

Temperature parameters for attention mechanisms (`tau`)

Loss weights (`lambda2`, `lambda3`, `lambda4`, `lambda5`)

6.3 Ablation Studies

To disable specific components for ablation studies:

Disable pixel-level OT

```
python trainer.py --config config.json --no-pixel-ot
```

Disable adaptive routing

```
python trainer.py --config config.json --no-adaptive-routing
```