ITRA

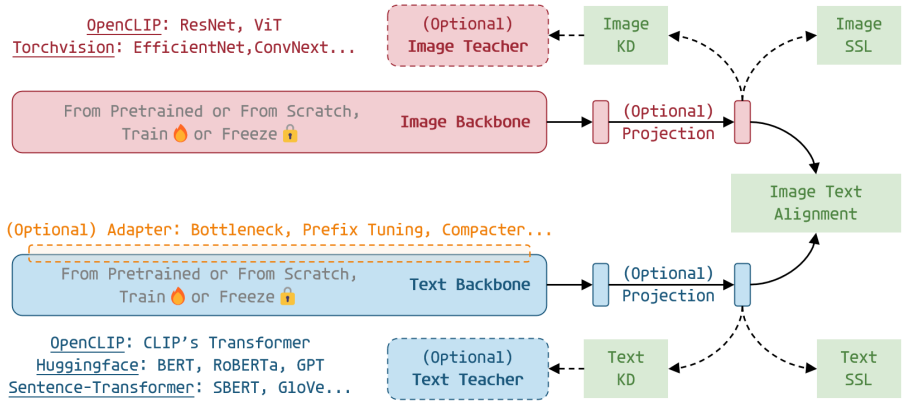
Release 0.1

Delong Chen

Apr 30, 2023

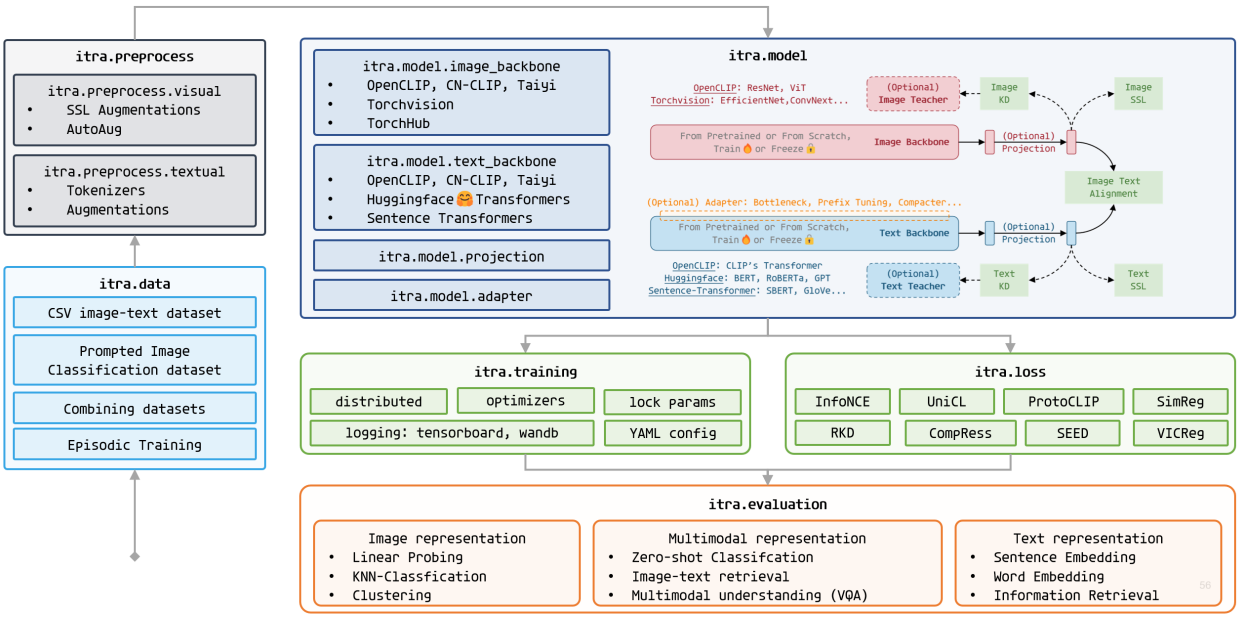
INTRODUCTION

ITRA (abbreviation for Image Text Representation Alignment) is a codebase for flexible and efficient vision language learning. ITRA features a unified interface to easily access state-of-the-art pretrained models, adapters, loss functions from various sources.



ITRA supports training, evaluation and benchmarking on a rich variety of tasks, including zero-shot/k-NN/linear classification, retrieval, word embedding and sentence embedding evaluation. In the meantime, ITRA is also highly modular extensible and configurable, facilitating future development and customization.

Important: ITRA is an ongoing project developing by the Artificial Intelligence of Multi-modality Group (AIM Group, https://multimodality.group) at Hohai University lead by Prof. Fan Liu. A temporary repository of the codebase is located at: https://github.com/ChenDelong1999/ITRA



Note: If you find any bugs or have any recommendations for building ITRA, please raise a issue in the repo, thanks~

CHAPTER

ONE

ABOUT THIS CODEBASE

ITRA is a codebase for flexible and efficient Image Text Representation Alignment…

1.1 Model Builder

- OpenCLIP
- Torchvision (v0.12)
- HuggingFace Transformers
- Sentence Transformers
- Adapter-Transformers
- TorchHub
- ChineseCLIP
- …

1.2 Training Objectives

- CLIP: InfoNCE, ProtoCLIP
- Self-supervised KD: RKD, SEED, CompRess, ProtoCPC, SimReg
- VICReg, BarlowTwins, DINO

1.3 Downstream Evaluation

- Image classification: zero-shot, linear/k-NN, and clustering evaluation (AMI, NMI) (from ProtoCLIP)
- EVEVATER Image Classification Toolkit on 20 datasets
- Image-text retrieval on MS-COCO dataset
- Sentence embeddings (SentEval)
- Passage retrieval on MS-MARCO and Wiki Sections
- Word embeddings: RG65, Simlex999, WordSim353
- Zero-shot VQA (TAP-C) and visual entailment

...

CHAPTER

TWO

RELATED WORK

2.1 OpenCLIP (LAION AI)

[github] | [pypi]

[paper]: Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, Jenia Jitsev. **Reproducible scaling laws for contrastive language-image learning**. *ArXiv preprint*.

Welcome to an open source implementation of OpenAI's CLIP (Contrastive Language-Image Pre-training).

The goal of this repository is to enable training models with contrastive image-text supervision, and to investigate their properties such as robustness to distribution shift. Our starting point is an implementation of CLIP that matches the accuracy of the original CLIP models when trained on the same dataset. Specifically, a ResNet-50 model trained with our codebase on OpenAI's 15 million image subset of YFCC achieves **32.7%** top-1 accuracy on ImageNet. OpenAI's CLIP model reaches **31.3%** when trained on the same subset of YFCC. For ease of experimentation, we also provide code for training on the 3 million images in the Conceptual Captions dataset, where a ResNet-50x4 trained with our codebase reaches 22.2% top-1 ImageNet accuracy.

We further this with a replication study on a dataset of comparable size to OpenAI's, LAION-400M, and with the larger LAION-2B superset. In addition, we study scaling behavior in a paper on reproducible scaling laws for contrastive language-image learning.

2.2 LAVIS - LAnguage VISion (Salesforce Research)

[github] | [doc]

[paper]: Dongxu Li, Junnan Li, Hung Le, Guangsen Wang, Silvio Savarese, Steven C.H. Hoi. **LAVIS: A Library for Language-Vision Intelligence**. *ArXiv preprint*.

LAVIS is a Python deep learning library for LAnguage-and-VISion intelligence research and applications. This library aims to provide engineers and researchers with a one-stop solution to rapidly develop models for their specific multimodal scenarios, and benchmark them across standard and customized datasets. It features a unified interface design to access

- **10+** tasks (retrieval, captioning, visual question answering, multimodal classification etc.);
- **20+** datasets (COCO, Flickr, Nocaps, Conceptual Commons, SBU, etc.);
- **30+** pretrained weights of state-of-the-art foundation language-vision models and their task-specific adaptations, including ALBEF, BLIP, ALPRO, CLIP.

2.3 MMF (Facebook AI Research)

[homepage] | [github] | [doc]

MMF is a modular framework for vision and language multimodal research from Facebook AI Research. MMF contains reference implementations of state-of-the-art vision and language models and has powered multiple research projects at Facebook AI Research. See full list of project inside or built on MMF here.

MMF is powered by PyTorch, allows distributed training and is un-opinionated, scalable and fast. Use MMF to ***bootstrap*** for your next vision and language multimodal research project by following the installation instructions. Take a look at list of MMF features here.

MMF also acts as **starter codebase** for challenges around vision and language datasets (The Hateful Memes, TextVQA, TextCaps and VQA challenges). MMF was formerly known as Pythia. The next video shows an overview of how datasets and models work inside MMF. Checkout MMF's video overview.

2.4 X-modaler (JD AI Research)

[github] | [doc]

[paper]: Yehao Li, Yingwei Pan, Jingwen Chen, Ting Yao, Tao Mei. **X-modaler: A Versatile and High-performance Codebase for Cross-modal Analytics**. *ACMMM Open Source Software Competition*.

X-modaler is a versatile and high-performance codebase for cross-modal analytics. This codebase unifies comprehensive high-quality modules in state-of-the-art vision-language techniques, which are organized in a standardized and user-friendly fashion.

2.5 TorchMultimodal (Facebook AI Research)

[github]

TorchMultimodal is a PyTorch library for training state-of-the-art multimodal multi-task models at scale. It provides:

- A repository of modular and composable building blocks (models, fusion layers, loss functions, datasets and utilities).
- A repository of examples that show how to combine these building blocks with components and common infrastructure from across the PyTorch Ecosystem to replicate state-of-the-art models published in the literature. These examples should serve as baselines for ongoing research in the field, as well as a starting point for future work.

As a first open source example, researchers will be able to train and extend FLAVA using TorchMultimodal.

2.6 Multimodal-Toolkit (Georgian)

[github] | [doc]

[paper] Ken Gu, Akshay Budhkar. **Multimodal-Toolkit: A Package for Learning on Tabular and Text Data with Transformers**. *Third Workshop on Multimodal Artificial Intelligence*.

A toolkit for incorporating multimodal data on top of text data for classification and regression tasks. It uses HuggingFace transformers as the base model for text features. The toolkit adds a combining module that takes the outputs of the transformer in addition to categorical and numerical features to produce rich multimodal features for downstream classification/regression layers. Given a pretrained transformer, the parameters of the combining module and transformer are trained based on the supervised task. For a brief literature review, check out the accompanying blog post on Georgian's Impact Blog.

2.7 Codebase

- https://github.com/megvii-research/mdistiller
- https://github.com/PyRetri/PyRetri

CHAPTER

THREE

CHANGE LOG

3.1 V0.0.1

2023.01.xx

Initial internal release.

CHAPTER

FOUR

INSTALL DEPENDENCIES

- Create a conda environment and install PyTorch:

```
conda create -n ITRA python=3.10.0
conda activate ITRA
```

This repo requires PyTorch (1.12) and torchvision (0.13). Please install them via pytorch official website.

```
conda install pytorch==1.12.0 torchvision==0.13.0 torchaudio==0.12.0 cudatoolkit=10.
↪2 -c pytorch
```

- Clone this repo:

```
# TODO: update repo name
git clone https://github.com/ChenDelong1999/ITRA
cd ITRA
export PYTHONPATH="$PYTHONPATH:$PWD/itra"
```

Note: If import error is occurred later, run `export PYTHONPATH="$PYTHONPATH:$PWD/itra"` again.

- Install additional dependencies:

```
conda install pillow pandas scikit-learn ftfy tqdm matplotlib
conda install -c huggingface transformers
conda install -c conda-forge sentence-transformers
pip install adapter-transformers open_clip_torch pycocotools wandb timm clip-
↪benchmark pyyaml

# TODO: faiss-gpu does not support windows OS, maybe use pip install faiss instead?
pip install faiss-gpu

# ELEVATOR requirements
pip install yacs git+https://github.com/haotian-liu/CLIP_vlp.git vision-evaluation

# TODO: remove nori dependency
pip install nori2
```

CHAPTER

FIVE

PREPARE DATA

5.1 Image-text Pairs Dataset from CSV file

This codebase reads a CSV file (separated by \t) with two columns: a path to an image (`filepath` by default), and a text caption (`title` by default).

Specifying `--train-data 'path/to/your/csvfile.csv'` enables training a model on the dataset, and specifying `--retrieval-data 'path/to/your/csvfile.csv'` and set `--retrieval-frequency` > 0 to perform retrieval evaluation on the dataset.

The script `itra/utils/gather_cc.py` will collect the Conceptual Captions (CC3M) dataset. First, download the Conceptual Captions URLs from here, then run the following script:

```
python3 itra/utils/gather_cc.py path/to/Train_GCC-training.tsv
```

Note: As mentioned in our ProtoCLIP paper, the CC3M dataset was made public by Google in 2018. As noted in our paper, the number of accessible images keeps drooping due to expired image links. This issue is raised by several recent works. In this work, since we can only collect 2,643,718 images (concurrent to our work ProtoCLIP, CyCLIP collected 2,631,703 images), we randomly sample a 2,500,000 subset (75% of full CC3M) from them to train our ProtoCLIP. Considering the dropping accessibility of image links in Conceptual Captions, we call for the use of this dataset size (2.5M) in future benchmarking for better comparability.

Important: The requirement of CC3M validation data of OpenCLIP is removed in this codebase. To perform retrieval evaluation, please use the `--retrieval-data` argument instead. The *webdataset* is no longer supported in this codebase.

5.2 MS COCO Captions dataset

To use MS COCO 2017 Captions dataset, download it to `--datasets-dir` and specifying `--train-data 'mscoco_captions'` or `--retrieval-data 'mscoco_captions'`.

```
<--datasets-dir>
    └─coco2017
        ├── annotations
        ├── train2017
        └── val2017
```

The dataset contains 118k train images and 5k text images, and each image has 4-5 captions. When using the training images, the total samples per epoch is set to 118k, and we chose one caption randomly when calling the `__getitem__` function.

5.3 Image Classification Dataset

Add your dataset into `itra/data/classification_datasets.py` and add your dataset name (e.g., 'YourCustomDataset') to `AVALIABLE_CLASSIFICATION_DATASETS`. Then you can use this dataset via `--train-data 'YourCustomDataset'`.

5.4 SentEval Datasets

Codes for SentEval evaluation are modified from SimCSE.

```
cd <--dataset-dir>
wget https://huggingface.co/datasets/princeton-nlp/datasets-for-simcse/resolve/main/
↪senteval.tar
tar xvf senteval.tar
```

Todo:

- MS MARCO
- wiki sections

5.5 EVEVATER Image Classification Datasets

EVEVATER Image Classification Toolkit (Elevater_Toolkit_IC) implemeted standarlized evaluations of vision language models. It covers zero-shot classification, few- / full-shot linear probing, and fully fine tuning on 20 datasets. See paper "*ELEVATER: A Benchmark and Toolkit for Evaluating Language-Augmented Visual Models, NeurIPS 2022 Datasets and Benchmarks Track*" for more details.

We have included Elevater_Toolkit_IC in our codebase (in `itra/evaluation/vision_benchmark`). We have registered new models (clip_zeroshot_eval.py and cls_linear_or_ft_eval.py) following the official instructions. To ensure compatibility, we have made some modifications based on the official Elevater_Toolkit_IC codes at commit `9d39620`, so DO NOT install an Elevater_Toolkit_IC in the environment for this codebase.

To get started first download all dataset following this repo. The downloaded datasets takes about 41Gb storage, and the folder structure should be:

```
.../datasets
└── classification
    ├── caltech_101_20211007
    │   ├── labels.txt
    │   ├── test.txt
    │   ├── test.zip
    │   ├── train.txt
    │   └── train.zip
```

(continues on next page)

(continued from previous page)

```
    ├── cifar100_20200721
    │   ├── labels.txt
    │   ├── test_images.txt
    │   ├── test_images.zip
    │   ├── train_images.txt
    │   └── train_images.zip
    ...
    └── voc2007_20211007
        ├── labels.txt
        ├── test_ic.txt
        ├── test.zip
        ├── train_ic.txt
        ├── train.zip
        └── val_ic.txt

21 directories, 115 files
```

5.6 NORI Datasets on OSS (for Megvii Useres)

- To use Conceptual Captions 3M: `--train-data 's3://chendelonghahab/datasets/ConceptualCaption3M/nori_CC2716261.csv'`

```
# Nori Speed-up Commands
nori speedup 's3://chendelong/datasets/ConceptualCaption3M/CC_3M.nori' --on --replica=2
nori speedup 's3://chendelonghahab/datasets/ConceptualCaption3M/CC2.6M-CC2M.nori/' --on -
↪-replica=2
```

- To use YFCCM-14M: `--train-data 's3://chendelonghahab/datasets/YFCC/YFCC_cleaned_nori.csv'`

```
# zsh
# Nori Speed-up Commands
for ((i=0;i<=100;i++)) {
    echo 'Processing nori part '$i'/100...'
    nori speedup 's3://yzq/mmsl_datasets/YFCC15M/yfcc15m_'$i'.nori' --on --replica=2
}
```

CHAPTER

SIX

LOAD PRETRAINED MULTI-MODAL WEIGHTS

6.1 From OpenCLIP

OpenCLIP (v2.0.2) is an open source implementation of OpenAI's CLIP (Contrastive Language-Image Pre-training). To check all supported model architecture and pre-trained weights, run:

```
import open_clip
open_clip.list_pretrained()
# [('RN50', 'openai'), ('RN50', 'yfcc15m'), ('RN50', 'cc12m'), ('RN50-quickgelu', 'openai'), ('RN50-
↪quickgelu', 'yfcc15m'), ('RN50-quickgelu', 'cc12m'), ('RN101', 'openai'), ('RN101', 'yfcc15m'), (
↪'RN101-quickgelu', 'openai'), ('RN101-quickgelu', 'yfcc15m'), ('RN50x4', 'openai'), ('RN50x16',
↪'openai'), ('RN50x64', 'openai'), ('ViT-B-32', 'openai'), ('ViT-B-32', 'laion400m_e31'), ('ViT-B-
↪32', 'laion400m_e32'), ('ViT-B-32', 'laion2b_e16'), ('ViT-B-32', 'laion2b_s34b_b79k'), ('ViT-B-
↪32-quickgelu', 'openai'), ('ViT-B-32-quickgelu', 'laion400m_e31'), ('ViT-B-32-quickgelu',
↪'laion400m_e32'), ('ViT-B-16', 'openai'), ('ViT-B-16', 'laion400m_e31'), ('ViT-B-16',
↪'laion400m_e32'), ('ViT-B-16-plus-240', 'laion400m_e31'), ('ViT-B-16-plus-240', 'laion400m_
↪e32'), ('ViT-L-14', 'openai'), ('ViT-L-14', 'laion400m_e31'), ('ViT-L-14', 'laion400m_e32'), (
↪'ViT-L-14', 'laion2b_s32b_b82k'), ('ViT-L-14-336', 'openai'), ('ViT-H-14', 'laion2b_s32b_b79k
↪'), ('ViT-g-14', 'laion2b_s12b_b42k'), ('roberta-ViT-B-32', 'laion2b_s12b_b32k'), ('xlm-
↪roberta-base-ViT-B-32', 'laion5b_s13b_b90k'), ('xlm-roberta-large-ViT-H-14', 'frozen_
↪laion5b_s13b_b90k')]
```

To load the official pretrained CLIP (ResNet-50):

```
--image-model 'RN50' --image-model-builder 'openclip' \
--text-model 'RN50' --text-model-builder 'openclip' \
--pretrained-image-model --pretrained-text-model \
```

Optionally, you can load CLIP models pretrained by OpenCLIP instead of OpenAI by specifying `--image-model-tag` and `--text-model-tag`. For example, to load the ViT-H-14 pretrained on LAION-2B:

```
--image-model 'ViT-H-14' --image-model-builder 'openclip' --image-model-tag 'laion2b_
↪s32b_b79k' \
--text-model 'ViT-H-14' --text-model-builder 'openclip'  --text-model-tag 'laion2b_s32b_
↪b79k' \
--pretrained-image-model --pretrained-text-model \
```

6.2 From ChineseCLIP

ChineseCLIP (v1.4) is the Chinese version of CLIP. We use a large-scale Chinese image-text pair dataset (~200M) to train the model, and we hope that it can help users to conveniently achieve image representation generation, cross-modal retrieval and zero-shot image classification for Chinese data. This repo is based on OpenCLIP project.

The ChineseCLIP models are also available on Huggingface, but here we import the model via cn_clip package for convenience since its codes are similar to OpenCLIP

To list available models (please see Model Card provided by ChineseCLIP for more details):

```
from cn_clip.clip import available_models
available_models()
# ['ViT-B-16', 'ViT-L-14', 'ViT-L-14-336', 'ViT-H-14', 'RN50']
```

To load a ChineseCLIP with ResNet-50:

```
--image-model 'RN50' --image-model-builder 'chineseclip' \
--text-model 'RN50' --text-model-builder 'chineseclip' \
--pretrained-image-model --pretrained-text-model \
```

6.3 From Taiyi-CLIP

Taiyi-CLIP -employs chinese-roberta-wwm for the language encoder, and apply the ViT-B-32 in CLIP for the vision encoder. They freeze the vision encoder and tune the language encoder to speed up and stabilize the pre-training process. Moreover, they apply Noah-Wukong dataset (100M) and Zero dataset (23M) as the pre-training datasets. See their documentations for details.

There are two CLIP models available via `Taiyi-CLIP`: `Taiyi-CLIP-Roberta-102M-Chinese` (doc) and `Taiyi-CLIP-Roberta-large-326M-Chinese` (doc). These two models are trained by Locked Image Tuning (LiT) on the ViT-B-32 and ViT-L-14 of OpenAI's CLIP. Therefore, to load these model:

```
# Taiyi-CLIP-Roberta-102M-Chinese
--image-model 'ViT-B-32' --image-model-builder 'openclip' \
--text-model 'IDEA-CCNL/Taiyi-CLIP-Roberta-102M-Chinese' --text-model-builder
↪'huggingface' \
--pretrained-image-model --pretrained-text-model \

# Taiyi-CLIP-Roberta-large-326M-Chinese
--image-model 'ViT-L-14' --image-model-builder 'openclip' \
--text-model 'IDEA-CCNL/Taiyi-CLIP-Roberta-large-326M-Chinese' --text-model-builder
↪'huggingface' \
--pretrained-image-model --pretrained-text-model \
```

CHAPTER

SEVEN

LOAD PRETRAINED UNI-MODAL WEIGHTS

7.1 Image Backbone

7.1.1 From Torchvision

To check all supported model architecture and pretrained weigths, run the following command or see this page (v0.12).

```
import torchvision
torchvision.models.__dict__.keys()
```

```
--image-model-builder 'torchvision' --image-model 'resnet50' \
--image-model-builder 'torchvision' --image-model 'resnet50' --pretrained-image-model \
--image-model-builder 'torchvision' --image-model 'alexnet' \
--image-model-builder 'torchvision' --image-model 'convnext_tiny' \
--image-model-builder 'torchvision' --image-model 'wide_resnet50_2' \
--image-model-builder 'torchvision' --image-model 'vgg11' \
--image-model-builder 'torchvision' --image-model 'squeezenet1_0' \
--image-model-builder 'torchvision' --image-model 'inception_v3' \
--image-model-builder 'torchvision' --image-model 'mobilenet_v3_small' \
--image-model-builder 'torchvision' --image-model 'mnasnet0_5' \
--image-model-builder 'torchvision' --image-model 'shufflenet_v2_x0_5' \
--image-model-builder 'torchvision' --image-model 'efficientnet_b0' \
--image-model-builder 'torchvision' --image-model 'regnet_y_400mf' \
--image-model-builder 'torchvision' --image-model 'vit_b_16' \
```

7.1.2 From Torch Hub

```
import torch
for github in ['swav', 'dino', 'vicreg', 'barlowtwins', 'swag', 'deit']:
    print(f'{github}:\t', torch.hub.list(f'facebookresearch/{github}'))
```

```
--image-model-builder 'torchhub' --image-model 'resnet50' --image-model-tag
↪'facebookresearch/swav:main' \
--image-model-builder 'torchhub' --image-model 'dino_vits16' --image-model-tag
↪'facebookresearch/dino:main' \
--image-model-builder 'torchhub' --image-model 'resnet50' --image-model-tag
↪'facebookresearch/vicreg:main' \
--image-model-builder 'torchhub' --image-model 'resnet50' --image-model-tag
```

(continues on next page)

(continued from previous page)

```
↪'facebookresearch/barlowtwins:main' \
--image-model-builder 'torchhub' --image-model 'regnety_16gf' --image-model-tag
↪'facebookresearch/swag:main' \
...
```

https://github.com/facebookresearch/VICRegL import torch model = torch.hub.load('facebookresearch/vicregl:main', 'resnet50_alpha0p9') model = torch.hub.load('facebookresearch/vicregl:main', 'resnet50_alpha0p75') model = torch.hub.load('facebookresearch/vicregl:main', 'convnext_small_alpha0p9') model = torch.hub.load('facebookresearch/vicregl:main', 'convnext_small_alpha0p75') model = torch.hub.load('facebookresearch/vicregl:main', 'convnext_base_alpha0p9') model = torch.hub.load('facebookresearch/vicregl:main', 'convnext_base_alpha0p75') model = torch.hub.load('facebookresearch/vicregl:main', 'convnext_xlarge_alpha0p75')

For more details, see:

- https://github.com/facebookresearch/swav
- https://github.com/facebookresearch/dino
- https://github.com/facebookresearch/vicreg
- https://github.com/facebookresearch/barlowtwins
- https://github.com/facebookresearch/SWAG
- https://github.com/facebookresearch/deit/blob/main/README_deit.md

---

7.2 Text Backbone

7.2.1 From HuggingFaceTransformers

For more details, see HuggingFace Transformers. Currently, only 'from pretrained' mode is supported (i.e., you cannot train a huggingface transformer from scratch now). Standard models like BERT/RoBERTa are supported, but whether other models are also supported is not sure…

7.2.2 From Sentence Transformers

The Sentence Transformers liberary provides powerfull sentence embeddings. Please see pretrained models for more detials. Loading sentence transformers via huggingface and specify `--text-pooler='mean'` is recommended, though it is also supported to load the model via sentence transformer:

```
# recommended:
--text-model-builder 'huggingface'  --text-model 'sentence-transformers/all-mpnet-base-v2
↪' --text-pooler='mean'
# not recommended:
--text-model-builder 'sbert'  --text-model 'all-mpnet-base-v2'
```

However, it seems that word embedding models (GloVe and Komninos) in sentence-transformers cannot be loaded via huggingface.

CHAPTER

EIGHT

CUSTOM TRAINING DATA

8.1 Episodic Training

```
--dataset-size 14000000 --episode-size 4000000 --train-data 'cache/yfcc_nori.csv' --nori-
↪dataset\
--epochs 28 --save-frequency 28 --batch-size 64 --workers 8 \
```

8.2 Combining Multiple Datasets

...

(weighting strategy...)

CHAPTER

NINE

LOSS FUNCTIONS

CHAPTER

TEN

USE ADAPTERS

The Adapter-Transformers liberary enables Delta-tuning on popular huggingface transformers. See Model Overview for available adaptations, and see the Docs and AdapterHub for more details.

We have made the following adapters available in this codebase:

- Projection Head Adapters
  - Linear projection head
  - DINO MLP Head (optionally with a prototype layer in the last)

CHAPTER

ELEVEN

FREEZE MODEL PARAMETERS DURING TRAINING

```
  # lock image tower, i.e., Locked Image Tuning (LiT) https://arxiv.org/abs/2111.07991
--lock-image-model \

# lock all weight in image tower, while only train the text tower
--lock-image-partial 'weight' \

# only unlock all weight in image tower, while other params are locked
--lock-image-partial '!weight' --lock-image-model \

# Only train the first layer (transformer block) of the image backbone
--lock-image-partial '!resblocks.0'  --lock-image-model \

# Only unfreeze all bias and norm params, i.e., Bias and Normalization Optimization␣
↪(BiNor) https://arxiv.org/abs/2203.07190
--lock-image-partial '!bias,!ln,!bn' --lock-text-partial '!bias,!ln' --lock-image-model ␣
↪--lock-text-model \
```

CHAPTER

TWELVE

EVALUATE PRETRAINED MODELS

12.1 Zero-shot Image-text Retrieval

CLIP is a strong model for zero-shot image text retrieval. Since the official paper only reports the performance of the largest CLIP ViT-L-14-336 (standard 32 epoch plus an additional pretraining epoch with 336x336 resolution), here we present our evaluation of other architectures of CLIP. See paper-with-code leader board for performance comparison with other zero-shot retrieval methods.

For ViT-L-14-336, there is a small gap between our implemented evaluation and the officially reported results. We suspect it is caused by image pre-processing: the above re-implementations use the default `Resize` transform as implemented in the official CLIP repo, while COCO images are mostly not square, it creates a small train-test domain gap due to distortion. If we alternatively use a `ResizeMaxSize` as implemented here, the results then surpass the official reported performance.

Changing `Resize` into `ResizeMaxSize` brings +2.06 improvement for ViT-L-14-336. However, we find that the benifit of this modification is not consistent across different backbones. As shown in the following table, generally, `ResizeMaxSize` is more beneficial for large models, and especially the models that have been trained to process HD images (e.g., it is quite beneficial for ViT-L-14-336 but not that much for ViT-L-14).

Therefore, to keep it simple, we will use the default `Resize` transform in the following experiments.

```
# 1x2080ti machine
python itra/training/main.py \
    --linear-frequency 0  --zeroshot-frequency 0 --retrieval-frequency 0  --nlp-eval-
↪frequency 1 --datasets-dir '/data/Datasets' \
    --retrieval-data 'mscoco_captions' \
    --image-model 'RN50' --image-model-builder 'openclip'  \
    --text-model 'RN50' --text-model-builder 'openclip'  \
    --pretrained-image-model --pretrained-text-model \
    --logs 'logs/MSCOCO-zeroshot'  --name 'RN50x4-openclip-zeroshot-retrieval


# [('RN50', 'openai'), ('RN50', 'yfcc15m'), ('RN50', 'cc12m'), ('RN50-quickgelu', 'openai
↪'), ('RN50-quickgelu', 'yfcc15m'), ('RN50-quickgelu', 'cc12m'), ('RN101', 'openai'), (
↪'RN101', 'yfcc15m'), ('RN101-quickgelu', 'openai'), ('RN101-quickgelu', 'yfcc15m'), (
↪'RN50x4', 'openai'), ('RN50x16', 'openai'), ('RN50x64', 'openai'), ('ViT-B-32', 'openai
↪'), ('ViT-B-32', 'laion400m_e31'), ('ViT-B-32', 'laion400m_e32'), ('ViT-B-32',
↪'laion2b_e16'), ('ViT-B-32', 'laion2b_s34b_b79k'), ('ViT-B-32-quickgelu', 'openai'), (
↪'ViT-B-32-quickgelu', 'laion400m_e31'), ('ViT-B-32-quickgelu', 'laion400m_e32'), ('ViT-
↪B-16', 'openai'), ('ViT-B-16', 'laion400m_e31'), ('ViT-B-16', 'laion400m_e32'), ('ViT-
↪B-16-plus-240', 'laion400m_e31'), ('ViT-B-16-plus-240', 'laion400m_e32'), ('ViT-L-14',
↪'openai'), ('ViT-L-14', 'laion400m_e31'), ('ViT-L-14', 'laion400m_e32'), ('ViT-L-14',
```

(continues on next page)

(continued from previous page)

```
↪'laion2b_s32b_b82k'), ('ViT-L-14-336', 'openai'), ('ViT-H-14', 'laion2b_s32b_b79k'), (
↪'ViT-g-14', 'laion2b_s12b_b42k'), ('roberta-ViT-B-32', 'laion2b_s12b_b32k'), ('xlm-
↪roberta-base-ViT-B-32', 'laion5b_s13b_b90k'), ('xlm-roberta-large-ViT-H-14', 'frozen_
↪laion5b_s13b_b90k')]
```

Coming soon…

12.2 Zero-shot Image Classification

Coming soon…

12.3 Linear Probing and KNN CClassification

Coming soon…

12.4 Clustering Evaluation

Coming soon…

12.5 Sentence Embedding Evaluation

STS-Benchmark, SICK…

MS MARCO Passage Retrval…

Word embeddings..

Coming soon…

12.6 ELEVATOR Image Classification Benchmark

You can perform EVEVATOR evaluations of the model trained by this codebase, by making necessary modifications and run the following commands:

```
conda activate vlkd
cd /data/codes/ProtoRKD
export PYTHONPATH="$PWD/src/training/evaluations:$PWD/src"

# zero-shot:       model_cfg='clip_zeroshot_eval'      mode='zeroshot'\
# few-shot:        model_cfg='cls_linear_or_ft_eval'   mode='linear_probe' num_shots=5 \
# linear prob:     model_cfg='cls_linear_or_ft_eval'   mode='linear_probe' num_shots=-1 \
# fine-tune:       model_cfg='cls_linear_or_ft_eval'   mode='finetune'     num_shots=-1 \

for dataset (caltech101 cifar10 cifar100 country211 dtd eurosat-clip fer2013 fgvc-
↪aircraft-2013b flower102 food101 gtsrb hateful-memes kitti-distance mnist oxford-iiit-
```

(continues on next page)

(continued from previous page)

```
↪pets patchcamelyon rendered-sst2 resisc45-clip stanfordcar voc2007classification)
{
    #---> REPLACE THIS LINE WITH ONE OF FOUR OPTIONS ABOVE <---#
    log_dir=# <YOUR EXPERIMENT DIR> \
    ckpt_epoch=# <WHICH EPOCH> \
    dataset_root=# <YOUR DATASET DIR> \
    dataset=$dataset \
    disable_hyperparameter_tuning=True \
        bash run_evevater_eval.sh
}
```

for example,

```
conda activate vlkd
cd /data/codes/ProtoRKD
export PYTHONPATH="$PWD/src/training/evaluations:$PWD/src"

for dataset (caltech101 cifar10 cifar100 country211 dtd eurosat-clip fer2013 fgvc-
↪aircraft-2013b flower102 food101 gtsrb hateful-memes kitti-distance mnist oxford-iiit-
↪pets patchcamelyon rendered-sst2 resisc45-clip stanfordcar voc2007classification)
{
    model_cfg='cls_linear_or_ft_eval'   mode='finetune'     num_shots=-1 \
    log_dir='/data/codes/ProtoRKD/logs/codebase_test/U[mobilenet_v3_large-h2]-L[CLIP-
↪from-RN50]-bs1024-YFCC-56ep-lr1e-5' \
    ckpt_epoch=56 \
    dataset=$dataset \
    disable_hyperparameter_tuning=True \
    dataset_root='/data/codes/ProtoRKD/src/training/evaluations/vision_benchmark/outputs/
↪datasets'\
        bash run_evevater_eval.sh
}
```

Then you can generate submission file for EvalAI. For more details, please see official instructions.

```
python src/training/evaluations/vision_benchmark/commands/prepare_submit.py \
  --combine_path 'logs/codebase_test/L[mobilenet_v3_small-h2]-L[CLIP-from-RN50]-bs1024-
↪YFCC-8ep/clip_zeroshot_eval/log/predictions/zeroshot_eval_wiki_False_wnh_False_wnd_
↪False_gpt3_Falseagg_WIKI_AND_GPT3_gpt3count_0'
```

We provide a simple script to summarize the results:

```
python src/utils/summarize_ELEVATER_results.py
Input your log dir (end with "../ELEVATER_evaluation/<eval_mode>"):
>>> logs/U[mobilenet_v3_large-h2]-L[CLIP-from-RN50]-bs1024-YFCC-56ep-lr1e-5/ELEVATER_
↪evaluation/zeroshot
                         Dsataset  zeroshot-accuracy%
0                     caltech-101             70.4490
1                       cifar-10             72.8000
2                      cifar-100             37.1700
3                     country211              7.0570
4                             dtd             31.5430
5                     eurosat_clip             25.3000
6                        fer-2013             21.8170
```

(continues on next page)

(continued from previous page)

```
7   fgvc-aircraft-2013b-variants102              5.1620
8                 oxford-flower-102             45.4590
9                         food-101              40.3290
10                           gtsrb               8.8600
11                   hateful-memes              52.4110
12                  kitti-distance              14.3460
13                           mnist              11.0400
14                oxford-iiit-pets              65.2600
15                  patch-camelyon              50.7600
16                   rendered-sst2              47.8860
17                   resisc45_clip              23.2740
18                   stanford-cars               5.0990
19          voc-2007-classification              77.5720
20                         Average              35.6797
saved to logs/U[mobilenet_v3_large-h2]-L[CLIP-from-RN50]-bs1024-YFCC-56ep-lr1e-5/
↪ELEVATER_evaluation/zeroshot/summary.csv
```

CHAPTER

THIRTEEN

CLIP PRETRAINING

First, assume that you have already created an environment with required dependencies, prepared data for pre-training and downstream evaluations.

Then you can activate the environment and modify the `PYTHONPATH` variable, such that modules can be imported successfully.

```
conda activate ITRA
export PYTHONPATH="$PYTHONPATH:$PWD/src"
```

13.1 Standard Contrastive Language Image Pretraining From Scratch

Training a CLIP from scratch is the most straight forward usage of `ITRA`. By specifying `--loss 'InfoNCE'`, the model will contrast image and text samples within a batch.

```
# Example command for a 8x2080ti machine
torchrun --nproc_per_node 8 -m training.main \
    --dataset-size 14000000 --episode-size 14000000 --train-data 'cache/yfcc_nori.csv' --
↪nori-dataset\
    --epochs 8 --save-frequency 8 --batch-size 64 --workers 8 \
    --lr 5e-4 --warmup 2000 --wd 0.5 --max-grad-norm 5 \
    --image-model 'RN50' --image-model-builder 'openclip' --text-model 'RN50' --text-
↪model-builder 'openclip'\
    --loss 'InfoNCE' \
    --report-to tensorboard --logs 'logs/example-usage/clip-pretraining/YFCC14M-8_epoch-
↪RN50'
```

13.2 Train a Tiny CLIP

- AlexNet, MobileNet?
- Small SBERT?
- GloVe Embeddings?

CHAPTER

FOURTEEN

FINE-TUNING CLIP FOR MS-COCO RETRIEVAL

In this section, we present an example usage and some empirical guides of fine-tuning CLIP for image-text retrieval. We aim to improve the retrieval performance based on the strong zero-shot retrieval ability (see our evaluation report) of CLIP by fine-tuning CLIP on MS COCO Captions training set (118k images) with the InfoNCE loss. Contents and key findings of this section are listed as follows:

- Fine-tuning CLIP on MS COCO training set improves the retrieval mean recall by +15% compared to raw zero-shot retrieval.
- Proper hyper-parameters can bring at least +1% improvement.
- Scale up batch size by partially freeze CLIP weights brings +1% improvement.
- Compared to the zero-shot retrieval mean recall=58.39% of RN50 CLIP, at last we achieve 76.02% mean recall (17.63% improvement) by fine-tuning it with a 8x2080ti machine.

14.1 Getting Started: Naive Fine-tuning Baseline

First, assume that you have already created an environment with required dependencies, prepared csv datasets for pre-training and downstream evaluations. Then you can activate the environment and modify the `PYTHONPATH` variable, such that modules can be imported successfully.

```
conda activate ITRA
cd path/to/ITRA/
export PYTHONPATH="$PYTHONPATH:$PWD/itra"
```

Then we can start to fine-tune a CLIP on MS-COCO captions 2017 training set (118k images). The results should be compared with the paper-with-code leaderboard. Our baseline setting are listed as follows, we use a single-node machine with 8 NVIDIA GeForce 2080ti GPUs for training, one training epoch takes about 3.5 minutes.

- backbone: ResNet50
- batch_size: 32x8=256
- dataset_size: 118287
- epochs: 10
- lr: 1e-05
- opt: adamw
- use_bn_sync: False
- warmup: 100
- weight_decay: 0.5

```
torchrun --nproc_per_node 8 -m training.main \
    --train-data 'mscoco_captions' --retrieval-data 'mscoco_captions' \
    --retrieval-frequency 1 --datasets-dir '/data/Datasets' \
    --epochs 10 --save-frequency 0 --batch-size 32 --workers 2 \
    --lr 1e-5 --warmup 100 --weight_decay 0.5 --max-grad-norm 5 \
    --image-model 'RN50' --image-model-builder 'openclip' --text-model 'RN50' --text-
↪model-builder 'openclip'\
    --pretrained-image-model --pretrained-text-model \
    --loss 'InfoNCE' \
    --report-to tensorboard --logs 'logs/MSCOCO-RN50'  --name '10ep-bs256-lr1e-5-wd0.5'
```

Under this configuration, fine-tuning significantly improves the retrieval performance (58.39→73.98, +15.59).

Note: Here Florence and PTP-BLIP are respectively the two-stream and single-stream SoTA retrieval methods at paper-with-code leaderboard by 2022.12.

14.2 Tuning Hyper-parameters

1. Learning Rate. We vary the learning rate from 5e-6 to 1e-4, and find that **1e-5 and 2e-5 are good for a batch size of 256**. This results confirms the observations in this paper, where the authors showed that good ImageNet fine-tuning of CILP ViT-B-16 needs a quite small learning rate (2e-5 and 3e-5 for a batch size of 2048).

2. Weight Decay. The author of SLIP paper observed that a larger weight decay (0.5) is beneficial for CLIP. Our experiments also showed that **CLIP can also handle a very large value of weight decay** (i.e., 2.50). Here the training data have 118k samples, and we believe that such property can further benefit CLIP fine-tuning when the data is limited. Our results, as shown in the following table, show that **CLIP is pretty robust to weight decay changes**: when vary the value from 0.01 to 2.50, the performance changes in a range of only +- 0.43.

3. Training Length. Similar to the experiments in FLIP, our experiments showed that **scaling training epochs cannot lead to further performance improvement**. Only 5 or 10 epochs are not sufficient, but 15-20 epochs seems already reached the saturation.

4. Batch Size. It is well known that batch size has a crucial impact for contrastive learning methods. We confirm this point by varying batch size from 32 to 800 (the maximum allowed batch size for ResNet-50 CLIP on a 8x2080ti machine) while changing learning rate according to liner scaling rule. It shows that **scaling down batch size leads to significant performance drop**:

5. Improved Naive Baseline with Better Hyper-parameters. Combining all the above hyper-parameter sweep observations together, we increase the mean recall of naive fine-tuning baseline from 73.98 to 75.04.

```
torchrun --nproc_per_node 8 -m training.main \
    --train-data 'mscoco_captions' --retrieval-data 'mscoco_captions' \
    --retrieval-frequency 1 --datasets-dir '/data/Datasets' \
    --epochs 15 --save-frequency 0 --batch-size 100 --workers 2 \
    --lr 3125e-8 --warmup 100 --weight_decay 1.0 --max-grad-norm 5 \
    --image-model 'RN50' --image-model-builder 'openclip' --text-model 'RN50' --text-
↪model-builder 'openclip'\
    --pretrained-image-model --pretrained-text-model \
    --loss 'InfoNCE' \
    --report-to tensorboard --logs 'logs/MSCOCO-RN50'  --name '15ep-bs800-lr3125e-8-wd1.0
↪'
```

Results:

14.3 Scaling up Batch Size by Partially Freeze Weights

- lock image and partially fine-tune text
- lock text and partially fine-tune image
- Scale up batchsize

```
torchrun --nproc_per_node 8 -m training.main \
    --train-data 'mscoco_captions' --retrieval-data 'mscoco_captions' \
    --retrieval-frequency 1 --datasets-dir '/data/Datasets' \
    --epochs 15 --save-frequency 15 --batch-size 224 --workers 4 \
    --lr 7e-5 --warmup 100 --weight_decay 1.0 --max-grad-norm 5 \
    --image-model 'RN50' --image-model-builder 'openclip' --text-model 'RN50' --text-
↪model-builder 'openclip'\
    --pretrained-image-model --pretrained-text-model --lock-image-model \
    --lock-text-partial 'positional_embedding,token_embedding' \
    --lock-image-partial '!attnpool,!layer4' \
    --loss 'InfoNCE' \
    --report-to tensorboard --logs 'logs/MSCOCO-RN50-partial'  --name 'save-lock-image(!
↪attnpool,!layer4)-lock-text(positional_embedding,token_embedding)-bs1792-lr7e-5'
```

14.4 More Tricks for Fine-tuning

14.4.1 Layer-wise Learning Rate Decay (LLDR)

```
--layer_decay_image 0.9 --layer_decay_text 1 \
```

```
for layer_decay_text in 1.0 0.95 0.9 0.85 0.8 0.75 0.7 0.65 0.6;
do
torchrun --nproc_per_node 8 -m training.main \
    --train-data 'mscoco_captions' --retrieval-data 'mscoco_captions' \
    --retrieval-frequency 1 --datasets-dir '/data/Datasets' \
    --epochs 15 --save-frequency 0 --batch-size 224 --workers 2 \
    --lr 7e-5 --warmup 100 --weight_decay 1.0 --max-grad-norm 5 \
    --image-model 'RN50' --image-model-builder 'openclip' --text-model 'RN50' --text-
↪model-builder 'openclip'\
    --pretrained-image-model --pretrained-text-model --lock-image-model \
    --lock-text-partial 'positional_embedding,token_embedding' \
    --lock-image-partial '!attnpool,!layer4' \
    --loss 'InfoNCE' \
    --report-to tensorboard --logs 'logs/MSCOCO-RN50-LLDR'  --name 'layer_decay_text='
↪$layer_decay_text \
    --layer_decay_text $layer_decay_text;
done
```

14.4.2 Exponential Moving Average (EMA)

```
--model_ema --model_ema_decay 0.998 \
```

```
for model_ema_decay in 0.99999 0.9999 0.9995 0.999 0.995 0.99 0.95 0.9 0.8;
do
torchrun --nproc_per_node 8 -m training.main \
    --train-data 'mscoco_captions' --retrieval-data 'mscoco_captions' \
    --retrieval-frequency 1 --datasets-dir '/data/Datasets' \
    --epochs 15 --save-frequency 0 --batch-size 224 --workers 2 \
    --lr 7e-5 --warmup 100 --weight_decay 1.0 --max-grad-norm 5 \
    --image-model 'RN50' --image-model-builder 'openclip' --text-model 'RN50' --text-
↪model-builder 'openclip'\
    --pretrained-image-model --pretrained-text-model --lock-image-model \
    --lock-text-partial 'positional_embedding,token_embedding' \
    --lock-image-partial '!attnpool,!layer4' \
    --loss 'InfoNCE' \
    --model_ema --model_ema_decay $model_ema_decay \
    --report-to tensorboard --logs 'logs/MSCOCO-RN50-EMA'  --name 'model_ema_decay='
↪$model_ema_decay;
done
```

14.4.3 Wise-FT. Evaluate the model with weight space ensemble

Wise-FT

```
--eval-with-wise-ft 0.5 \
```

```
for alpha in 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 ;
do
python itra/training/main.py \
    --zeroshot-frequency 1 --retrieval-frequency 1 --retrieval-data 'mscoco_captions' --
↪datasets-dir '/data/Datasets' \
    --image-model 'RN50' --image-model-builder 'openclip'  \
    --text-model 'RN50' --text-model-builder 'openclip'  \
    --pretrained-image-model --pretrained-text-model \
    --resume 'logs/MSCOCO-RN50-partial/save-lock-image(!attnpool,!layer4)-lock-
↪text(positional_embedding,token_embedding)-bs1792-lr7e-5/checkpoints/epoch_15.pt' \
    --eval-with-wise-ft $alpha \
    --logs 'logs/MSCOCO-RN50-WiseFT'  --name 'zs+retrieval-WiseFT='$alpha;
done
```

14.4.4 rsicd retrieval

```
# 1x2080ti machine RSICD
torchrun --nproc_per_node 8 -m training.main \
    --train-data '/data/Datasets/RSICD/csv/rsicd_train.csv' --images-dir '/data/Datasets/
↪RSICD/RSICD_images/RSICD_images' \
    --csv-separator '\t' --csv-img-key 'filename' --csv-caption-key 'title' \
    --retrieval-data '/data/Datasets/RSICD/csv/rsicd_test.csv' --retrieval-images-dir '/
↪data/Datasets/RSICD/RSICD_images/RSICD_images' \
    --retrieval-csv-separator '\t' --retrieval-csv-img-key 'filename' --retrieval-csv-
↪caption-key 'title' \
    --retrieval-frequency 1  --datasets-dir '/data/Datasets' \
    --epochs 30 --save-frequency 0 --batch-size 16 --workers 2 \
    --lr 1e-6 --warmup 100 --weight_decay 0.5 --max-grad-norm 5 \
    --image-model 'ViT-L-14-336' --image-model-builder 'openclip' \
    --text-model 'ViT-L-14-336' --text-model-builder 'openclip' \
    --pretrained-image-model --pretrained-text-model \
    --lock-image-model --lock-text-model \
    --lock-image-partial '!ln_post,!resblocks.23,!resblocks.22,!resblocks.21,!resblocks.
↪20,!resblocks.19,!resblocks.18' \
    --lock-text-partial '!text_projection,!ln_final,!resblocks.11,!resblocks.10,!
↪resblocks.9' \
    --loss 'InfoNCE' --layer_decay_image 0.9 --layer_decay_text 0.9 \
    --report-to tensorboard --logs 'logs/RSICD-ViT-L-14'  --name '30ep-b128-lr1e-5-
↪unlock-image-text-last0.75-lldr0.9'
```

python itra/training/main.py –config-yaml ‘logs/params.yml’ –name ‘custom-name’

python itra/training/main.py –episode-size 10000 –train-data ‘mscoco_captions’ –retrieval-data ‘mscoco_captions’ –retrieval-frequency 1 –datasets-dir ‘/data/Datasets’ –epochs 15 –save-frequency 0 –batch-size 100 –workers 2 –lr 1e-4 –warmup 100 –weight_decay 1.0 –max-grad-norm 5 –image-model ‘RN50’ –image-model-builder ‘openclip’ –text-model ‘RN50’ –text-model-builder ‘openclip’–pretrained-image-model –pretrained-text-model –lock-image-model –lock-text-model –loss ‘InfoNCE’ –prompt –n-prompt 4 –report-to tensorboard –logs ‘logs/test’ –name ‘coco-finetune-nprompt-4’

CHAPTER

FIFTEEN

IMAGE CLASSIFICATION (UNICL)

UniCL: Unified Contrastive Learning in Image-Text-Label Space

15.1 Train an Image Classification Model From scratch

Compare to MMClassification

- resnet18_cifar.py
- cifar10_bs16.py
- cifar10_bs128.py

```
# Single GPU classification
python itra/training/main.py \
    --train-data 'CIFAR10' \
    --linear-frequency 20  --zeroshot-frequency 20 --datasets-dir '/data/Datasets' \
    --epochs 200 --save-frequency 0 --batch-size 128 --workers 4 \
    --opt 'sgd' --lr 0.1 --warmup 100 --weight_decay 0.0001 \
    --image-model 'resnet18' --image-model-builder 'torchvision' --image-resolution 32  -
↪-image-head-n-layers 1 \
    --pretrained-text-model \
    --text-model 'RN50' --text-model-builder 'openclip' --lock-text-model --text-head-n-
↪layers 1  \
    --loss 'CrossEntropy' --joint-projection-dim 10 \
    --report-to tensorboard --logs 'logs/UniCL-Classification'  --name
↪'resnet18(scratch)-CIFAR10-200ep-CrossEntropy+linear_eval'

# Single GPU classification
python itra/training/main.py \
    --train-data 'CIFAR10' \
    --linear-frequency 5 --zeroshot-frequency 5 --datasets-dir '/data/Datasets' \
    --epochs 200 --save-frequency 0 --batch-size 128 --workers 4 \
    --opt 'sgd' --lr 0.1 --warmup 100 --weight_decay 0.0001 \
    --image-model 'resnet18' --image-model-builder 'torchvision' --image-resolution 32  -
↪-image-head-n-layers 1 \
    --pretrained-text-model \
    --text-model 'RN50' --text-model-builder 'openclip' --lock-text-model --text-head-n-
↪layers 1  \
    --loss 'InfoNCE' --joint-projection-dim 1024 \
    --report-to tensorboard --logs 'logs/UniCL-Classification'  --name
↪'resnet18(scratch)-CIFAR10-200ep-InfoNCE+linear_eval'
```

15.2 Fine-tuning CLIP for ImageNet Classification

Re-implement this paper.

python itra/training/main.py –train-data 'mscoco_captions' –retrieval-data 'mscoco_captions' –dataset-size 1000 –retrieval-frequency 1 –datasets-dir '/data/Datasets' –epochs 1 –save-frequency 1 –batch-size 32 –workers 2 –lr 1e-5 –warmup 100 –weight_decay 0.5 –max-grad-norm 5 –image-model 'RN50' –image-model-builder 'openclip' –text-model 'RN50' –text-model-builder 'openclip'–pretrained-image-model –pretrained-text-model –loss 'InfoNCE' –report-to tensorboard –logs 'logs/test' –name 'RN'

CHAPTER

SIXTEEN

LANGUAGE-TO-VISION KNOWLEDGE DISTILLATION

Coming soon…

CHAPTER

SEVENTEEN

VISION-TO-LANGUAGE KNOWLEDGE DISTILLATION

Coming soon…

Todo: New features incoming

- Refract main.py
- Write help messages for arguments
- Use YAML
- **Project**
  - install as package
  - Pypi package publishing
- **Evaluation reports**
  - zero-shot classification
  - linear/knn classification
  - clustering evaluation
  - SentEval
  - word embedding
  - MS Marco retrieval
  - Chinese CLIPs' Evaluation Reports (ImageNet-CN zero-shot, MC-COCO-CN retrieval)
- **Implementations**
  - UniCL-based image classification
  - Validate loss functions
  - Validate Adapters
  - SimCSE and PromptBERT re-implementation
  - Vision-to-language Knowledge Distillation
  - Language-to-vision Knowledge Distillation
  - Teacher selection based on Information Bottleneck Theory
