VISSL Baseline Documentation

Purpose

This documentation provides essential guidelines for setting up and running the VISSL baseline for the jigsaw task within our project. Our aim is to establish a robust and reliable baseline, leveraging the capabilities of VISSL to handle this specific task efficiently.

What is VISSL?

VISSL is a library built on top of PyTorch, designed for state-of-the-art Self-Supervised Learning (SSL). It is developed by Facebook AI and offers a versatile platform for SSL research and applications. VISSL provides a range of SSL algorithms and benchmarks, making it a valuable tool for developing models that learn from unlabeled data.

How to Use the Project

Prerequisites

- Ensure you have Docker installed on your system.
- Obtain the dataset required for the jigsaw task and ensure it is accessible.

Step 1: Modify the Dataset Path

- Locate the eval_resnet_gpu.yaml configuration file in the project directory.
- Open the file and modify the dataset source path to point to your dataset's location. Ensure the path is correctly set to enable the model to access and process the dataset.

Step 2: Build the Docker Container

- Open a terminal window and navigate to the project's root directory.
- Run the following command to build the Docker container:

docker build -t vissl_jigsaw_baseline .

This command builds a Docker container named vissl_jigsaw_baseline based on the Dockerfile provided in the project directory. (The needed requirements in the requirements.txt will be installed automatically.)

Step 3: Run the Docker Container

After successfully building the container, run it using the following command:

docker run -it --gpus all vissl_jigsaw_baseline

This command runs the container and allocates all available GPUs to it, which is crucial for training the model efficiently.

Training settings:

The provided YAML (eval_resnet_gpu.yaml) configuration file for VISSL contains several important settings that are key to understanding how the training process is structured and optimized. Here's an overview of some of the crucial elements:

1. Verbose Logging and Frequency: The `VERBOSE: True` setting enables detailed logging of the training process, and `LOG_FREQUENCY: 200` determines how often (every 200 iterations) the logs are produced.

2. **Testing Settings**: `TEST_ONLY: False` indicates that the model will be trained and not just tested. `TEST_EVERY_NUM_EPOCH: 1` means the model will be tested after every epoch. `TEST_MODEL: True` enables testing of the model during training.

3. Data Processing: The `DATA` section specifies important parameters like `NUM_DATALOADER_WORKERS: 5`, which sets the number of workers for data loading. The training and testing datasets are configured to be sourced from disk folders with specified transformations like `RandomResizedCrop`, `Normalize`, etc., to preprocess the images.

4. Model Settings: The `MODEL` section describes the model's architecture and settings. It includes `FEATURE_EVAL_SETTINGS` for evaluation mode and feature extraction settings, and `TRUNK` settings with `RESNETS: DEPTH: 50` specifying the use of a ResNet-50 architecture.

5. Optimization Parameters: Under `OPTIMIZER`, settings like `weight_decay: 0.0005`, `momentum: 0.9`, and `num_epochs: 28` are specified. The learning rate schedule is defined with `values: [0.01, 0.001, 0.0001, 0.00001]` and `milestones: [8, 16, 24]`, indicating a multistep decay at specified epochs.

6. Distributed Training: The `DISTRIBUTED` section includes settings for distributed training, like `NUM_NODES: 1` and `NUM_PROC_PER_NODE: 8`, indicating single-node training with 8 processes per node.

7. Checkpointing: In the `CHECKPOINT` section, `AUTO_RESUME: True` and `CHECKPOINT_FREQUENCY: 1` indicate that training will automatically resume from the latest checkpoint and checkpoints will be created after every epoch.

Error during the training:

During the training phase, our team encountered a persistent error originating from within the VISSL repository. Despite extensive efforts, we were unable to resolve this issue. Consequently, the training of the baseline model did not reach a successful conclusion. It appears that the instability we experienced may be attributed to the jigsaw component of the repository. This inference is supported by the observation that over the years, various tutorials and support files related to this component have been systematically removed, potentially indicating underlying issues with this specific segment of the VISSL framework.