Bone Suppression as a Transformation in Self-Supervised Learning for Tuberculosis (TB) Diagnosis from Chest X-Rays.

Abstract

Tuberculosis (TB) remains one of the world's deadliest diseases. Early detection is crucial but there are limitations and complications associated with current diagnostic methods. Therefore, there is a need for affordable, accessible alternatives. Despite the availability of unlabeled medical imaging data, the lack of publicly available labeled imaging data provides limitations. This paper posits that leveraging the unlabeled data through self-supervised learning (SSL) can be a potential solution. A prevalent question in applying SSL to medical imaging is the efficacy of standard data augmentations in capturing meaningful representations inherent to such data. Addressing this, our research contrasts the performance of domain-specific data augmentations against generic augmentations in SSL for TB classification from Chest X-Ray (CXR) images. Our primary focus is on bone suppression, a domain-specific technique that removes the skeletal structures from a CXR whilst leaving the rest of the image untouched. We examine the effects of the bone suppression transformation in 4 different SSL methods. Subsequent to the self-supervised pre-training, our models were evaluated in a linear and full model fine-tuning setting using a smaller labeled dataset. Our overall results, derived from various performance metrics, indicated an increase in performance when utilizing the bone suppression technique as a data augmentation. This finding not only underscores the potential of domain-specific augmentations in medical imaging but also paves the way for further exploration into novel augmentations tailored to specific diseases and domains.

View the full paper

Structure

Directories

- `CXR-bone-suppression` contains the weights for the bone suppression model.
- `SSL` contains the adapted code for the different SSL algorithms. View original code.

Scripts

- `SSL/byol.py` contains the adapted code for the BYOL framework from PySSL. This file should not be run directly.
- `SSL/dino.py` contains the adapted code for the DINO framework from PySSL This file should not be run directly.
- `SSL/simclr.py` contains the adapted code for the SimCLR framework from PySSL This file should not be run directly.
- `SSL/swav.py` contains the adapted code for the SwAV framwork from PySSL This file should not be run directly.
- `bone_supp_model.py` contains the code which creates the bone suppression model. This file should not be run directly.
- `bs_byol.py` contains the code which pretrains, fine-tunes and evaluates the BYOL models.

- `bs_dino.py` contains the code which pretrains, fine-tunes and evaluates the DINO models.
- `bs_simclr.py` contains the code which pretrains, fine-tunes and evaluates the SimCLR models.
- `bs_swav.py` contains the code which pretrains, fine-tunes and evaluates the SwAV models.
- `create_bone_supp.py` contains the code which creates the bone suppressed images and saves them to a folder for a given dataset.
- `datasets.py` contains the code which helps load, preprocess and access the datasets. This file should not be run directly.
- `fine_tune_model.py` contains the code which creates the structure of the fine-tuned models. This file should not be run directly.
- `sweep.py` contains the code which performed the hyperparameter sweeps for the fine-tuned models.

Environments

A `requirements.txt` file has been provided in order to recreate the conda environment which was used. Ensure your environment is set up before running any of these scripts.

Datasets

The 2 publicly available datasets which were used can be found here:

- TBX11
- IEEE

Run Files

Running SSL Models:

To perform operations with the Self-Supervised Learning (SSL) models, use the following command format:
`python3 <bs_ssl_script>.py <training_mode> <augmentation_scheme>`

<ssl_script>: The script corresponding to the SSL technique, choose from:

- `bs_byol`
- `bs_dino`
- `bs_swav`
- `bs_simclr`

<training_mode>: The desired mode of operation, options are:

- `pretrain`
- `train`
- `evaluate`

<augmentation_scheme>: The data augmentation strategy to apply, options are:

- `default`
- `bone_supp`
- `bone_default`
- `combo`

Note, <training_mode> must first be `pretrain` which will create a directory that contains the weights and loss for the specific SSL method and data augmentation strategy chosen. These weights are then used during fine-tuning, and so, the <training_mode> `train` can only be run after `pretrain`. Similarly, <training_mode> `evaluate` must only be run after `train`, as it utilizes the fine-tuned weights from this operation.

Running these files will create directories which contain the model weights, loss values, evaluation metrics and curves where applicable.

Running Hyperparameter sweeps for the pretrained models:

To run any hyperparameter sweeps, use the following command format: `python3 sweep.py <ssl_method> <augmentation_scheme> <fine_tuning_mode>`

<ssl_method>: The type of SSL model that is being fine-tuned, options are:

- `byol`
- `dino`
- `simclr`
- `swav`

<augmentation_scheme>: The data augmentation strategy to apply, options are:

- `default`
- `bone_supp`
- `bone_default`
- `combo`

<fine_tuning_mode>: The type of fine-tuning operation to perform, options are:

- `linear`, which performs linear evaluation
- `full`, which performs full model fine-tuning

Execution on Wits Cluster

The slurm scripts have 3 sbatch commands which should not be altered and include:

- job name
- time
- cluster partition

To execute scripts on the Wits cluster, use the provided slurm scripts with the following syntax: `sbatch --output=output.txt ./<script_name>.sh`.

For example, to create bone suppressed images, run: `sbatch --output=output.txt ./create_datasets.sh`. This will run the `create_bone_supp.py` script. An `output.txt` file will be created and will contain all the logs. You can name this `.txt` file anything you like.

We provide a set of Slurm scripts to facilitate various operations on the SSL models:

- `run_byol_bs.sh`
- `run_dino_bs.sh`
- `run_simclr_bs.sh`

- `run_swav_bs.sh`

The final line in these scripts need to be edited depending on the specific arguments chosen. Then, the slurm file can be run.

In addition, we provide a `sweep.sh` script which handles the parameter sweeps for the SSL models. The final line in this file needs to be edited depending on the specific desired arguments. Thereafter, the slurm file can be run.

Running Locally (not recommended)

In order to run the files locally, use the `python3 <python_script_name>.py <_args>` command where <_args> represents any potential arguments that have already been discussed.

Acknowledgements

We have made use of the following repositories:

- PySSL
- CXR-Bone-Suppression
- LARS Optimizer