liver-ct-segmentation Documentation

Release 1.0.0

Lukas Heumos

Mar 29, 2021

CONTENTS:

CHAPTER

ONE

LIVER-CT-SEGMENTATION

Liver-tumor segmentation of computed tomography scans using a U-Net model.

- Free software: MIT
- Documentation: https://liver-ct-segmentation.readthedocs.io.

A reproducible, Pytorch-based model for liver-tumor segmentation of computed tomography (CT) scans using a 3D U-Net architecture. This project uses the Liver Tumor Segmentation Benchmark (LiTS) dataset to train a simplified U-Net model for semantic segmentation of liver and tumor tissue (background, liver, tumor) from abdominal CT scans.

·docs/images/u_net_lits.png

A reproducibility analysis was conducted using three different experimental setups, a standard setup with disregard to reproducible calculations (Random), a setup where random seeds are defined (Seed), and the mlf-core deterministic setup (Deterministic). The LiTS dataset was randomly sampled to define a small test set (10%, 13 tomograms) and models were trained for 1000 epochs with the remainder of the tomograms, using the abovementioned experimental setups (10 training runs per setup). Reproducibility of prediction was tested by evaluating the performance of the models on the test set, using Intersection over Union (IoU) as a metric (Jaccard index).

·docs/images/iou_boxplots.png

1.1 Architecture

A reduced 3D U-Net architecture. The U-Net is a convolutional "encoder-decoder" model for semantic segmentation of 2D and 3D images. In this simplified model, convolutional layers with a stride of 2 are used for down-sampling, while the up-sampling operation was performed with the nearest neighbor algorithm. Here, convolutions use filter sizes of 3x3x3, dropout is applied to every convolutional layer, and the softmax function is used on the last layer to produce class pseudo-probabilities. Blue boxes correspond to 3D multi-channel feature maps, with the number of channels denoted on top, and the size of the spatial dimensions marked in the lower left.

·docs/images/u_net_architecture.png

1.2 Credits

This package was created with mlf-core using Cookiecutter.

CHAPTER

TWO

USAGE

2.1 Setup

mlf-core based mlflow projects require either Conda or Docker to be installed. The usage of Docker is highly preferred, since it ensures that system-intelligence can fetch all required and accessible hardware. This cannot be guaranteed for MacOS let alone Windows environments.

2.1.1 Conda

There is no further setup required besides having Conda installed and CUDA configured for GPU support. mlflow will create a new environment for every run.

2.1.2 Docker

If you use Docker you should not need to build the Docker container manually, since it should be available on Github Packages or another registry. However, if you want to build it manually for e.g. development purposes, ensure that the names matches the defined name in the ``MLproject``file. This is sufficient to train on the CPU. If you want to train using the GPU you need to have the NVIDIA Container Toolkit installed.

2.2 Training

2.2.1 Training on the CPU

Set your desired environment in the MLproject file. Start training using `mlflow run .`. No further parameters are required.

2.2.2 Training using GPUs

Conda environments will automatically use the GPU if available. Docker requires the accessible GPUs to be passed as runtime parameters. To train using all gpus run `mlflow run . -A t-A gpus=all -P gpus=<<num_of_gpus>> -P acc=ddp`. To train only on CPU it is sufficient to call `mlflow run . -A t`. To train on a single GPU, you can call `mlflow run . -A t -A gpus=all -P gpus=1` and for multiple GPUs (for example 2) `mlflow run . -A t -A gpus=all -P gpus=2 -P accelerator=ddp`. You can replace `all` with specific GPU ids (e.g. 0) if desired.

2.2.3 Parameters

- gpus Number of gpus to train with [2: int]
- accelerator Accelerator connecting to the Lightning Trainer ['ddp': string]
- max_epochs: Number of epochs to train [1000: int]
- general-seed: Python, Random, Numpy seed [0: int]
- pytorch-seed: Pytorch specific seed [0: int]
- training-batch-size: Batch size for training batches [1: int]
- test-batch-size: Batch size for test batches [1: int]
- lr: Learning rate of the optimizer [0.0001: float]
- log-interval: Number of batches to train for before logging [3000: int]
- class-weights: Class weights for loss function (separated by commas) ['0.2, 1.0, 2.5': string]
- test-percent: Can be used to separate train and test sets (unused) [0.15: float]
- test-epochs: Number of epochs between validations [10: int]
- dataset-path: Path to dataset ['/data/': string]
- dataset-size: Can be used to reduce dataset size (unused) [131: int]
- n-channels: Number of input channels for U-Net [1: int]
- n-class: Number of classes for U-Net [3: int]
- num_workers: Number of workers for data loading [24: int]
- dropout-rate: Dropout rate for U-Net [0.25: float]

CHAPTER

THREE

MODEL

3.1 Overview

The trained model is used for liver-tumor segmentation of computed tomography (CT) scans.

3.2 Training and test data

The training data origins from the Liver Tumor Segmentation Benchmark LiTS. 131 tomograms are part of the LiTS training dataset, from those 10% were randomly selected as a small test set (13 tomograms). The datase is available via codalab, or as a torrent.

3.3 Model architecture

The model is based on Pytorch and Pytorch Lightning.

A reduced 3D U-Net architecture. The U-Net is a convolutional "encoder-decoder" model for semantic segmentation of 2D and 3D images. In this simplified model, convolutional layers with a stride of 2 are used for down-sampling, while the up-sampling operation was performed with the nearest neighbor algorithm. Here, convolutions use filter sizes of 3x3x3, dropout is applied to every convolutional layer, and the softmax function is used on the last layer to produce class pseudo-probabilities. Blue boxes correspond to 3D multi-channel feature maps, with the number of channels denoted on top, and the size of the spatial dimensions marked in the lower left.

3.4 Evaluation

A reproducibility analysis was conducted using three different experimental setups, a standard setup with disregard to reproducible calculations (Random), a setup where random seeds are defined (Seed), and the mlf-core deterministic setup (Deterministic). The LiTS dataset was randomly sampled to define a small test set (10%, 13 tomograms) and models were trained for 1000 epochs with the remainder of the tomograms, using the abovementioned experimental setups (10 training runs per setup). Reproducibility of prediction was tested by evaluating the performance of the models on the test set, using Intersection over Union (IoU) as a metric (Jaccard index).

The full training history is viewable by running the mlflow user interface inside the root directory of this project: `mlflow ui`.

3.5 Hyperparameter selection

Hyperparameters were chosen on widely known strong defaults.

1. `Adam optimizer` was chosen for strong, general performance.
2. The learning rate was set to `0.0001`
3. The class-weights were set to `'0.2, 1.0, 2.5'` for the background, liver, and tumor respectively.
4. The dropout rate was set to `0.25`

CHAPTER

FOUR

CREDITS

4.1 Development Lead

- Luis Kuhn <luis.kuhn-cuellar@qbic.uni-tuebingen.de>
- Lukas Heumos <lukas.heumos@posteo.net>

4.2 Contributors

None yet. Why not be the first?

CHAPTER

FIVE

CHANGELOG

This project adheres to Semantic Versioning.

5.1 1.0.0 (2021-03-28)

Added

- First implementation of U-net for liver cancer segmentation
- Several runs conducted for the mlf-core paper

Fixed

Dependencies

Deprecated

CHAPTER

SIX

CONTRIBUTOR COVENANT CODE OF CONDUCT

6.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

6.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

6.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

6.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

6.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by opening an issue. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

6.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search