The Tree Species Segmentation Challenge

After the success of the flower classification challenge, an optional student competition we organized during the WS 2022/2023 in context of the Foundations of Deep Learning lecture, we are now organizing a second optional student competition, spanning the two semesters, to give even more students the opportunity to get 'hands-on" with deep learning.

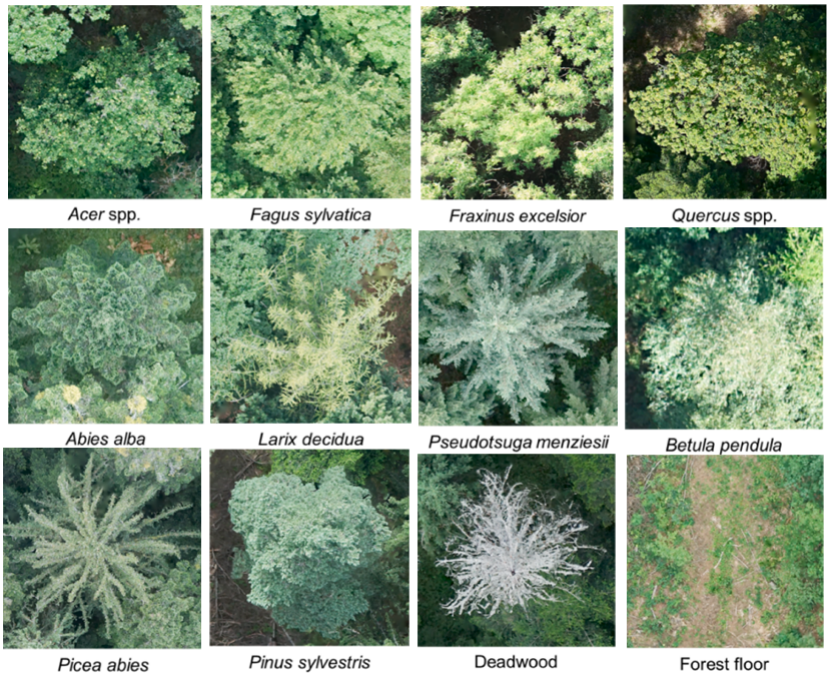
Participation is optional and your submissions will not be graded.

You are allowed to participate alone or in teams of up to 3 students.[1] Note that these teams can, but do not have to be the same as for the DL exercises. The winners will be announced by email, within two weeks after the deadline, and will receive nothing but eternal glory, AutoML.org T-shirts and a spot on the course webpage. To mimick a real-world setting, we give you a lot of freedom in how you tackle this challenge. However, with this freedom comes responsibility. So please play fair and make it easy for us to evaluate your submission. In doubt, please contact us or create a thread on the ILIAS forum.

To get access to the code, use the following link: `https://classroom.github.com/a/9aKtl0Bl`

The Challenge:

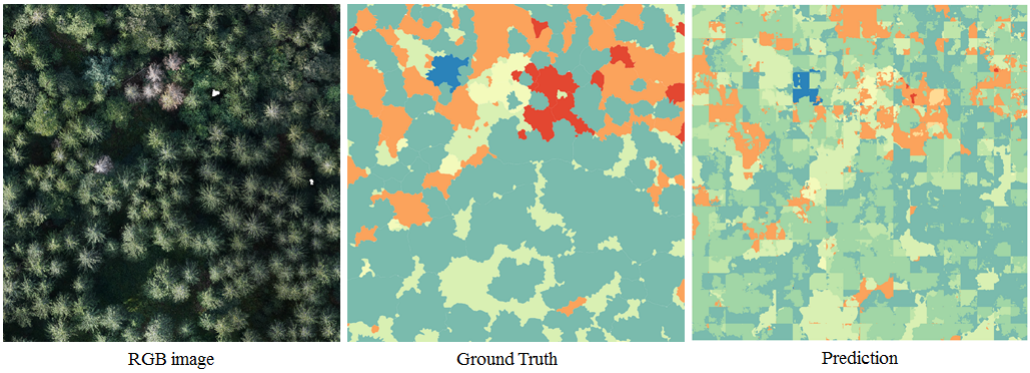
In this challenge, you are to train a model to perform semantic segmentation on a tree dataset. Specifically, given an image of a tree as input, your model must predict a per-pixel semantic labeling of the image.



An example of the 12 classes tree dataset

[1] To participate, you must have been enrolled for the Foundations of Deep Learning course during WS 2022/2023.

Image segmentation example using the baseline model



Example of full image segmentation for trees

To encourage everybody to participate, we require submitted models to have less than 25M parameters. Within these constraints, you are allowed to implement any architecture, and manually or automatically tune the hyperparameters of the model. The following are just a few of the things you can experiment with:

- Learning rate and its optional scheduler
- Different optimizers and their hyperparameters
- Specialized model architectures
- Activation functions
- Regularization
- Data preprocessing
- Cross-validation to ensure you do not overfit to the validation set

Also, you do not have to shy away from more advanced techniques like warm-starting your model with weights from other pre-trained models, or self-supervised learning. You are allowed to use all the scripts and tools you already know from the exercises. However, you are not limited to them.

Here are a few important topics concerning the competition:

- **Evaluation:**
  - The final performance will be measured in terms of the mean per-class Intersection over Union (mIoU) score of your model on an **unseen** test set. Assume we will fly our drones out and photograph some trees.
  - To get a taste of how we shall evaluate your model, try running `evaluate_model.py`. The script loads your saved model from `models` folder and evaluates its mIoU score on validation data loaded from the `dataset/test` folder.
  - To evaluate your model, we will populate this folder `dataset/test` with the unseen test data.
- **Hardware:**
  - Since semantic segmentation is a more challenging task than image classification it requires larger models, latent representations, data, and training on CPUs will likely take too long. We therefore

strongly recommend you to use GPUs.
  - You may use any kind of hardware that is available to you. For example, Google Colab repeatedly offers a VM with a GPU for at most 12 hours at a time for free.[2]
  - You are also free to use the pool computers.[3] You can use ssh to log in remotely to these computers as follows: `ssh yourpoolaccount@login.informatik.uni-freiburg.de`
    See `SetupGuideTFPool.pdf` for more info on how to use the pool computers.
    Also, see Pool-FAQ for further information.
- **Implementation Constraints:**
  - Your model should be written using PyTorch.
  - Do not modify the code inside `src/eval`. Your model will be evaluated using this code.
  - Other than that, you are free to extend/modify the baseline code given to you or write your own code from scratch.
- **Github Repository Constraints:**
  - **Keep your repository under 150 MB.**
  - Github does not allow files larger than 100MB to be tracked in repositories. **So make sure your trained model is under this limit.**
  - You are allowed to push only one trained model.
  - Your code has to be in the `master` branch before the deadline.
  - There is no limit on the number of pushes to the `master` branch.

As a starting point, we provide you with the following:

- `dataset` folder. This folder is initially empty, but will contain a subset of the original tree species segmentation dataset and train/validation/test split.[4] We only consider 12 of the 14 classes provided by the full dataset, so that is what you are given. The data (roughly 1.3GB) will be downloaded automatically when first running the starter's code, or you can manually download the tarball from Google drive: `https://drive.google.com/u/0/uc?id=1nkyipbJ19JR1XsgQumm-X_dV42SDPZbR`
  Do not push this data to your remote github repository!
- `eval/evaluate.py` and `eval/dataset.py` containing the code we shall use to evaluate your model and load the data. **Do not edit these files.**
- `data_augmentations.py` containing two sample data augmentation pipelines.
- `training.py` containing the code to train the model.
- `main.py` containing the code to load the data, train, evaluate, and, optionally, save the model.
  You can run it from the root directory using `python -m src.main` (will download the data automatically).
- `unet.py` containing `SampleModel`, as an example of what a model for semantic segmentation looks like. More specifically, it implements a small U-Net architecture with 7.8M parameters.
- A sample saved model in `src/models`. This file contains the weights of the `SampleModel` given above trained for 50 epochs using the default pipeline that is provided to you.
- `evaluate_model.py` which the organizers will use to evaluate your model.
  You can run it from the root directory using `python -m src.evaluate_model` (will download the data automatically).
- `requirements.txt` which contains the list of libraries required to run the given pipeline.
- `notebooks/dl2022_competition2_example.ipynb` which is a Google Colab notebook example for training the baseline model on GPUs.

Your submission must include:

- **A fully functional training pipeline**. This must include the code for your model, data augmenta-

[2]The baseline model was trained using Google Colab and required 3 - 4 GPU hours.
[3]Important: Your TF pool home directory disk quota is insufficient to install PyTorch. Please contact us as soon as possible to request additional storage space.
[4]You are free to consider different splits, use cross-validation, or even use all the data to train your final model(s).

tions you use and the code for training the model.[5]

- `requirements.txt` updated with any additional libraries you use.
- **The file(s) with the saved weights of your trained model(s)**, similar to `models/sample_model`, in the `models` folder.[6]
- `submission.md` must be populated with the answers to the following questions:
  - The number of parameters in your model(s). You can find this out using `torchsummary.summary`.
  - A *brief* description of your approach to the problem.
  - Command to run to train your model(s) from scratch with your hyperparameter settings and data augmentations. For example, from the root directory:
    `python -m src.main --epochs 50 --batch_size 8`
    You're free to edit `main.py` however you please to make this work (E.g., to accept hyperparameters as options, or hard-code them). You can also add new files, if you wish. **We must be able to train your model by running a single command.**
  - Command to run to evaluate your saved model(s) with your data augmentation pipeline.
    For example:
    `python -m src.evaluate_model --model SampleModel --saved-model-file sample_model`
    Again, you're free to edit `evaluate_model.py` however you want, **but ultimately, evaluation MUST be done by invoking `evaluate_model(...)` function in `eval/evaluate.py`. We must also be able to evaluate your model by running a single command.**
- *sample_submission.zip* contains a sample of what a submission must look like.

This project is due on 02.05.2023 23:59 CEST.

[5] If you use model warm-starting or other such methods, that should also be included in this pipeline. We must be able to run your code and train your model from scratch, as you did.

[6] Github has a limit of 100MB for tracked files. Your `model` must hence be smaller than 100MB.