FNSNF
Swiss National Science Foundation
CRSK-3_196515

Dafne

Deep Anatomical Federated Network

Francesco Santini
University Hospital Basel, Basel, Switzerland
University of Basel, Allschwil, Switzerland

Main developers: Jakob Wasserthal, Abramo Agosti; Medical supervision: Anna Pichiecchio

University of Basel
Department of Biomedical Engineering
Universitätsspital Basel
fm Fondazione Mondino
Istituto Neurologico Nazionale a Carattere Scientifico | IRCCS
BAMM Basel Muscle MRI

Declaration of Financial Interests or Relationships

Speaker Name: Francesco Santini

I have no financial interests or relationships to disclose with regard to the subject matter of this presentation.

UNI BASEL

Why are you not doing DL segmentation **right now?**

Here comes Dafne *

- It has the best kind of learning
  - **Deep learning**
  - **Federated learning**
    - It collects improvements from all over the world
    - And it preserves data privacy!
  - **Continuous incremental learning**
    - It learns from your own expertise, even from few examples!
- It has an easy **user interface**
  - Everything is under your control
- It's **free**, **multiplatform**, and **open source**!

* Dafne is the Greek name for bay leaf
– and also a girl name

Outline of this session

- Overview of the principles behind Dafne
- Live demo
- Extending Dafne together
  - Information for developers

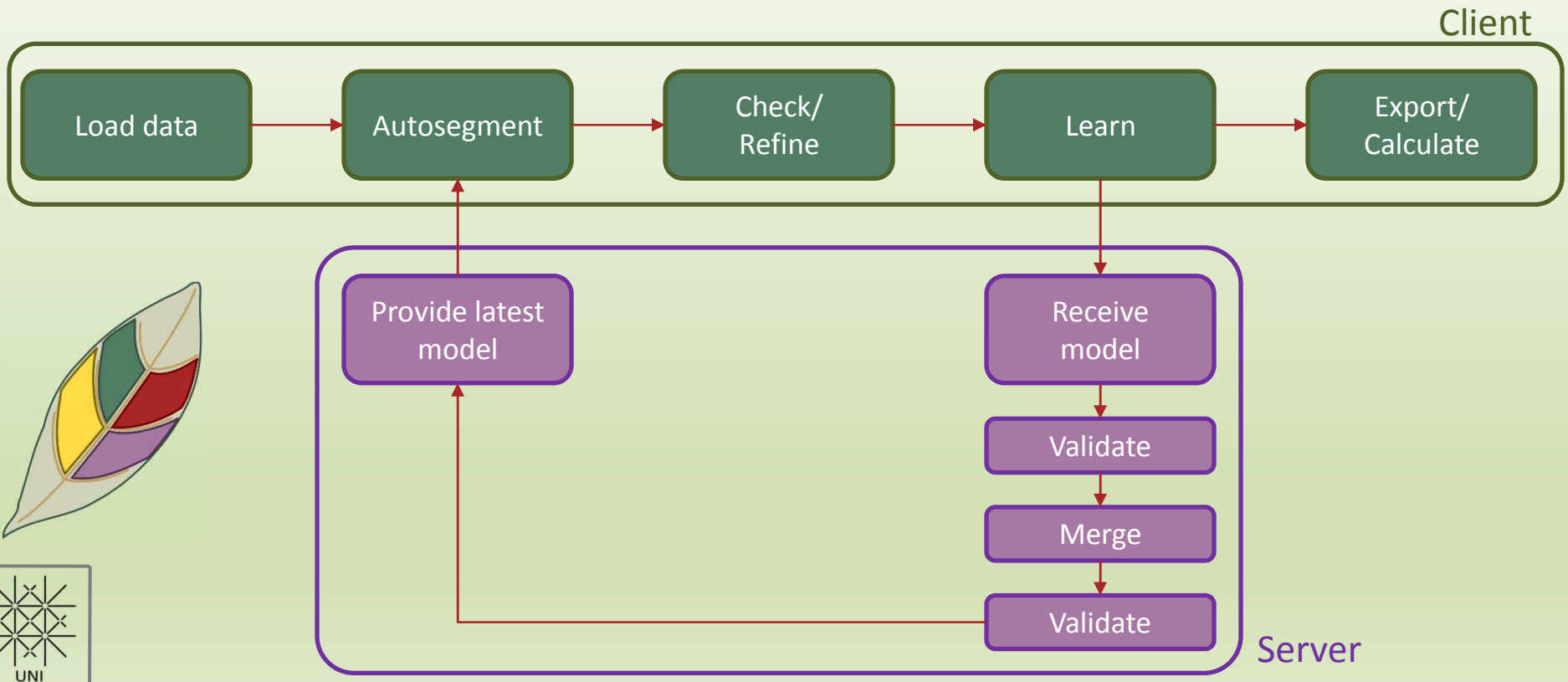
Part I

DAFNE'S PRINCIPLES

The main pillars of Dafne

1. Data privacy
   - Your data are never shared with us
2. Continuous learning
   - Incremental learning on mini-batches
3. Accountability
   - Dafne is always supervised

Dafne Workflow



Current core features

- 2D segmentation network
  - Performance reasons, can be easily changed
- Muscle segmentation of the **thigh** and **leg**
  - Extensible, see part 3
- Concept of “Federated Learning”
  - Model training is done by the client
    - Data privacy
    - Distributed computing
    - Adaptation to different data
- Concept of “Continuous incremental learning”
  - Imperfect pretrained models
    - Necessary for usability and to avoid catastrophic forgetting

Getting Dafne

- https://dafne.network/
- Free and multiplatform
  - 100% Python
  - Precompiled binaries for Windows and MacOS
  - (Linux planned – use source)
- Need API key to communicate with the server
  - Use ISMRM21 for this session

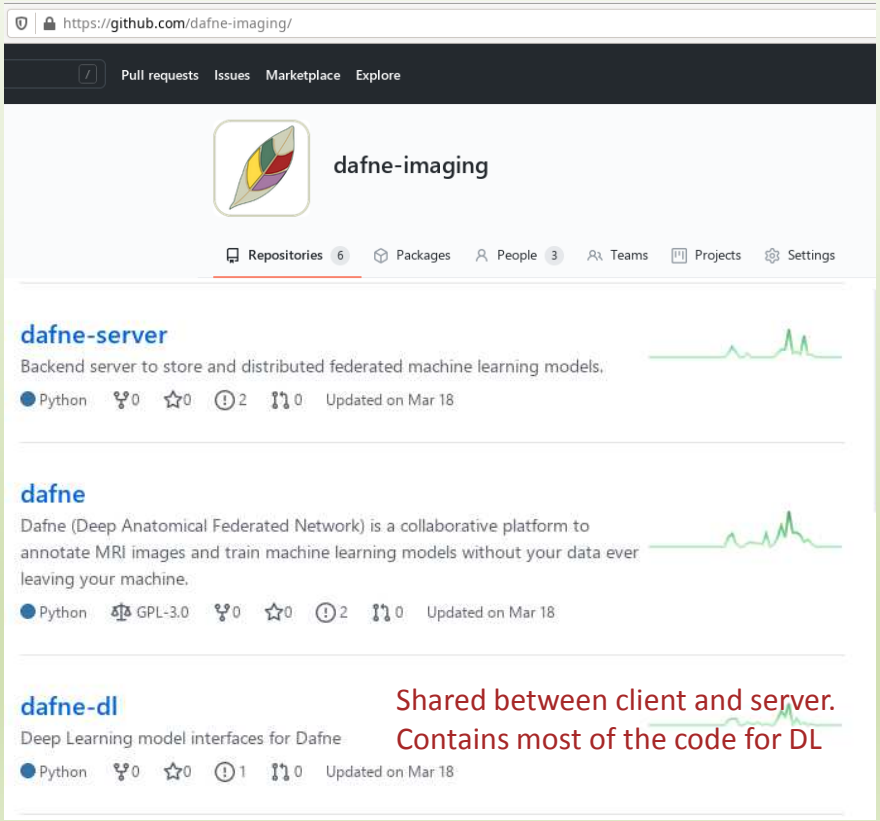
Part II

LIVE DEMO

Part III

EXTENDING DAFNE

Dafne-imaging github



Shared between client and server.
Contains most of the code for DL

Deep learning models

- Self-contained serialized Python objects (methods + data)
  - Uses `dill` for serialization.
- General interface (from dafne-dl/interfaces.py):
  - `init_model()`
  - `calc_delta(model)` – returns delta = model - self
  - `apply_delta(delta: model)` – returns self + delta
  - `factor_multiply(factor: float)` – returns factor*self
  - `incremental_learn(training_data: dict, training_outputs: any)`
  - `apply(data: dict)`
  - `reset_timestamp()` – resets the internal timestamp

Data dictionaries

- This applies to how data is passed by Dafne currently. The model specification itself is agnostic.
- `apply()`
  - Input:
  - `{'image': 2D image, 'resolution': sequence with pixel sizes, 'split_laterality': bool}`
  - Output:
  - `{'label': 2D mask, …}`
- `incremental_learn()`
  - training_data
  - `{'image_list': sequence of 2D images, 'resolution': sequence with pixel sizes}`
  - training_outputs
  - `[{'label': 2D mask, …}, …]` (corresponding to the image_list sequence).

Generic implementation

- From dafne-dl/DynamicDLModel.py
- Define *unbound* functions and pass them as arguments to the constructor
- There source is transparently retrieved and serialized
- Functions to save/retrieve model weights to/from dill-serializable

```
class DynamicDLModel(DeepLearningClass):

    """
    Class to represent a deep learning model that can be serialized/deserialized
    """
    def __init__(self, model_id,  # a unique ID to avoid mixing different models
                 init_model_function,  # inits the model. Accepts no parameters and returns the model
                 apply_model_function,  # function that applies the model. Has the object, and image, and a sequence containing resolutions as parameters
                 weights_to_model_function = default_keras_weights_to_model_function,  # put model weights inside the model.
                 model_to_weights_function = default_keras_model_to_weights_function,  # get the weights from the model in a pickable format
                 calc_delta_function = default_keras_delta_function,  # calculate the weight delta
                 apply_delta_function = default_keras_add_weights_function,  # apply a weight delta
                 weight_copy_function = default_keras_weight_copy_function,  # create a deep copy of weights
                 factor_multiply_function = default_keras_multiply_function,
                 incremental_learn_function = None,  # function to perform an incremental learning step
                 weights = None,  # initial weights
                 timestamp_id = None,
                 is_delta = False):
```

Example

- Check the sources of
  - dafne-dl/DynamicDLModel.py
  - dafne/generate_thigh_split_model.py
    - This script “packs” the original pretrained thigh model and stores it as a dill file.

Model providers

- Models are retrieved/uploaded by the ModelProviders
- General interface in dafne-dl/interfaces.py
  - `load_model(str)`
  - `upload_model(str, model, client_dice_score)`
  - `_upload_bytes(data)` – to upload generic data
  - `available_models()` – not part of the basic interface but implemented
- Remote and Local model providers implemented
  - Remote: talks to the server using HTTPS
  - Local: looks into a local directory

Server integration of models

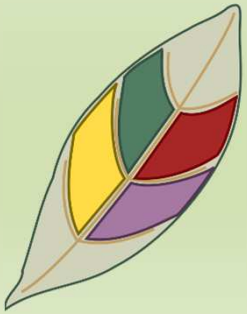
- The server dynamically identifies the available models.
- Models and test data are inserted in a directory structure →
- The merged model is currently an average of the existing model and of the uploaded model
- The test data is used for validation.
- The server is currently hosted on a Google cloud VM

server_db/

- api_keys.txt
- log.txt
- models/
  - *[model_name]/*
    - timestamp_merged_model.model
    - timestamp_merged_model.sha256
    - …
    - uploads/
      - timestamp_username.model
  - …/
- test_data/
  - *[model_name]/*
    - data_package.npz
    - …
  - …/

Conclusion

- Simple, extensible framework for model integration
- Currently based on Tensorflow but agnostic
- Necessity of pretrained models for usability and stability



Thank you for your attention

Check the website:
https://dafne.network/
Contact me directly:
francesco.santini@unibas.ch

Dafne is a collaboration between:

- University Hospital Basel (Switzerland)
- University of Basel (Switzerland)
- Fondazione Mondino (Pavia, Italy)

Supported by the SNSF

Swiss National Science Foundation
CRSK-3_196515