ODELIA

Open Consortium for Decentralized Medical Artificial Intelligence

HORIZON-HLTH-2021-CARE-05-02

Deliverable D1.4

INSTRUCTIONS ON TRAINING AI MODELS WITH SL IN A TANGIBLE DEMONSTRATION CASE

Lead beneficiary	TUD
Author(s)	Oliver Lester Saldanha, Kevin Pfeiffer, Prof. Dr. med. Jakob Nikolas Kather, Jiefu Zhu
Dissemination level	PU
Type	R
Delivery date	11/12/2023

ODELIA is funded by the European Union's Horizon Europe Framework under Grant Agreement 101057091

Funded by the European Union

TABLE OF CONTENTS

DISCLAIMER

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HADEA). Neither the European Union nor the granting authority can be held responsible for them.

SUMMARY

This progress report outlines several significant achievements in the ODELIA D1.4 phase.

INTRODUCTION

The use of artificial intelligence in medicine is progressing, emphasising the importance of accurate machine learning (ML) models. However, challenges arise in sharing extensive patient data, especially across international borders. Federated learning (FL) addresses this by training ML models independently at different sites without data sharing. Yet, FL requires a central coordinator, concentrating control. Swarm learning (SL), a recent development, uses blockchain-based communication, eliminating the need for a leading participant node and allowing dynamic onboarding and dropout of institutions. In this resource, we elucidate SL principles, detailing its architecture and interactions. A step-by-step manual for integrating SL into workflows, illustrated with breast cancer detection on MRI data using Hewlett Packard Enterprise (HPE) Swarm Learning, is provided. Over the next five years, increasing breast cancer screening with MRI will challenge radiologists, but AI offers a solution. Our research combines weakly supervised learning, bypassing detailed annotations, with swarm learning (SL), enabling local AI model training without centralised data sharing. Using a dataset of 922 MRI exams from the USA, we validated models on 427 exams from Germany. Benchmarking various weakly supervised 2D and 3D DL methods, we identified 3D-ResNet-101 as superior. Combining weakly supervised tumour detection with SL involving three computer nodes showcased matching accuracy to centralised training, emphasising the potential of this approach to leverage larger datasets for medical AI without detailed annotations and data sharing constraints.

TANGIBLE DEMONSTRATION CASE: AUTOMATIC BREAST CANCER DETECTION ON MRI DATA

The introduction of new breast cancer screening recommendations, particularly the endorsement of MRI as a screening method, is reshaping diagnostic practices in Europe and the United States [1,2]. Unlike previous guidelines that primarily relied on x-ray mammography [3,4], the evolving landscape, as outlined in the recent EUSOBI guideline [1], emphasises the use of MRI, especially for women with dense breast tissue. This shift necessitates a significant scaling.

PREREQUISITES

These are the software and hardware prerequisites for running the

Hardware Recommendations

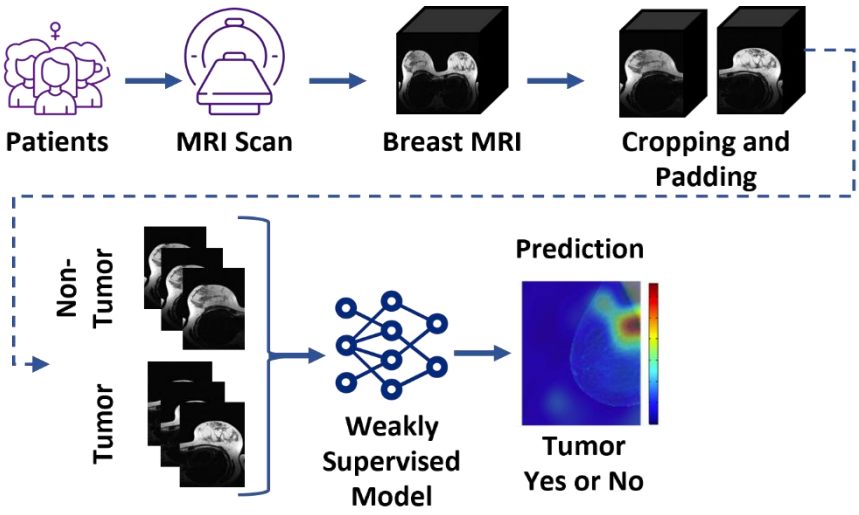
- 64 GB of RAM (32 GB is the absolute minimum)
- 16 CPU cores (8 is the absolute minimum)
- an NVIDIA GPU with 48 GB of RAM (24 is the minimum)
- 8 TB of Storage (4 TB is the absolute minimum)
- We deliberately want to show that we can work with lightweight hardware like this. Here are three quotes for systems like this for less than 10k EUR (Lambda, Dell Precision, and Dell Alienware)
- Typical installation time can take 30 minutes to build up the necessary dependencies and another 1 hour to build up the environment for running demo experiments.

Operating System

- Ubuntu 20.04 LTS
  - We have tested the Swarm Learning Environment on [Ubuntu 20.04 LTS, Ubuntu 22.04.2 LTS, Ubuntu 20.04.5 LTS] and they work fine.
  - Any experimental release of Ubuntu greater than LTS 20.04 MAY result in unsuccessful swop node running.
  - It also works on WSL2(Ubuntu 20.04.2 LTS) on Windows systems. WSL1 may have some issues with the docker service.

PREPROCESSING WORKFLOW

The study applied a uniform preprocessing pipeline to all datasets, involving two key steps. Initially, left and right breast volumes were individually cropped to align with the model's single breast volume processing. Each breast volume received a global label indicating malignancy (yes/no) from the Duke or UKA datasets. Following this, the data underwent transformation from DICOM to NIFTI files, with subsequent resampling to achieve a consistent resolution of 256 x 256 x 32 voxels. The processed sequence facilitated the computation of a subtraction image (difference between first post-contrast and pre-contrast T1-weighted images). This pre-processed data was then employed to train a deep learning model for binary classification of the volume into two classes: malignancy, yes or no. This simplification allows for the application of various weakly supervised prediction methods in analysing the problem of tumour detection.

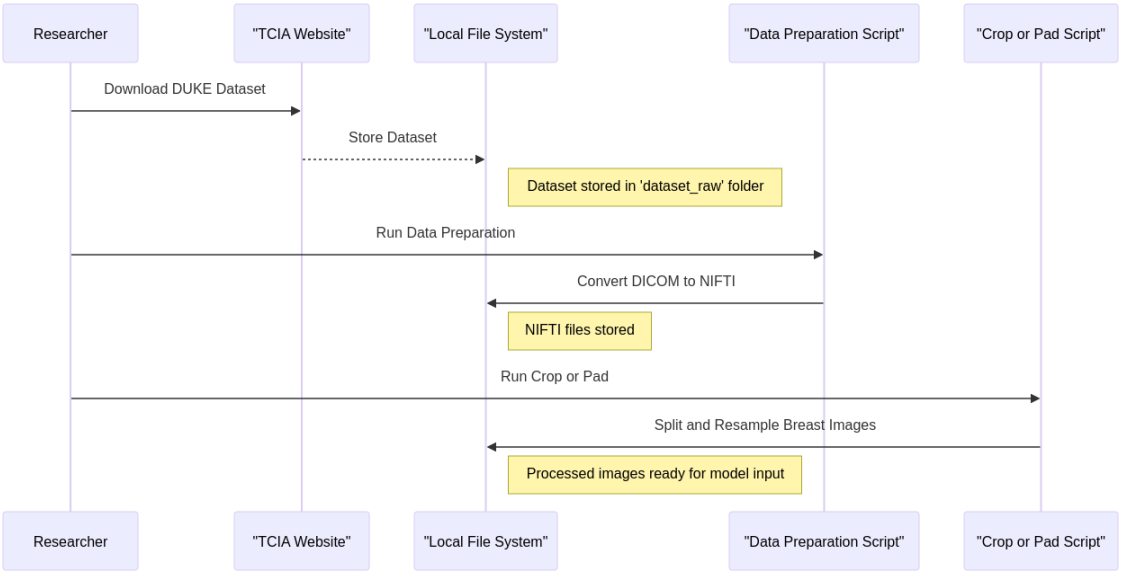


DATA PRE-PROCESSING STEP-BY-STEP GUIDE

The steps taken for data preprocessing can be divided into three parts.

1. Downloading the DUKE Dataset: The researcher downloads the dataset from the TCIA (The Cancer Imaging Archive) website. The dataset is then stored in the 'dataset_raw' folder on the local file system.
2. Running the Data Preparation Script: The researcher executes the data preparation script. This script converts the DICOM files into NIFTI format, which are then stored on the local file system.

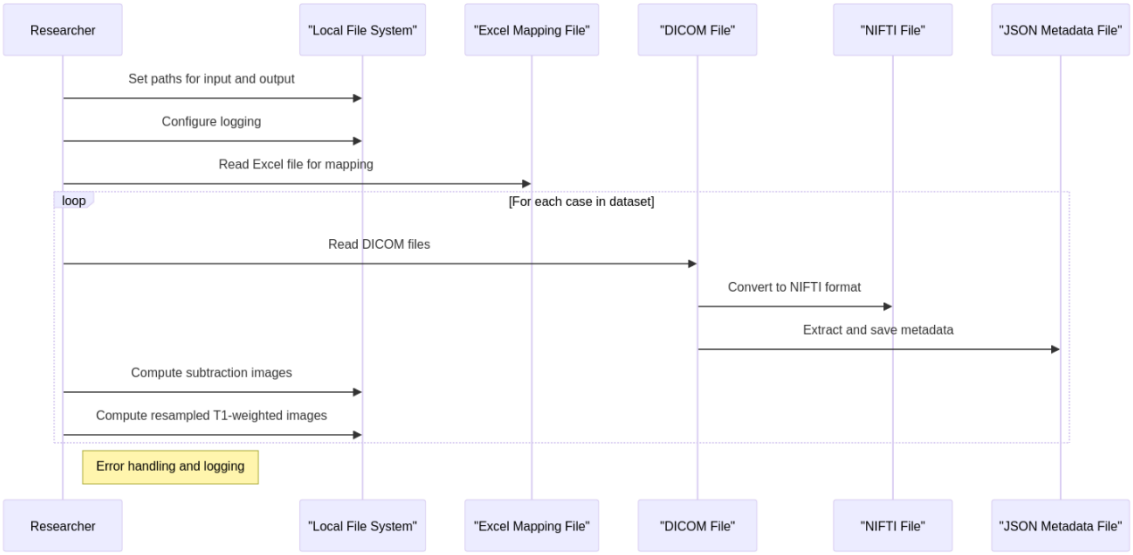
3. Running the Crop or Pad Script: Finally, the researcher runs the crop or pad script. This script processes the breast images by splitting them into left and right sides and resampling them to a uniform shape. The processed images are then ready to be used as input for the different models.



DETAILED PRE-PROCESSING WORKFLOW DESCRIPTION

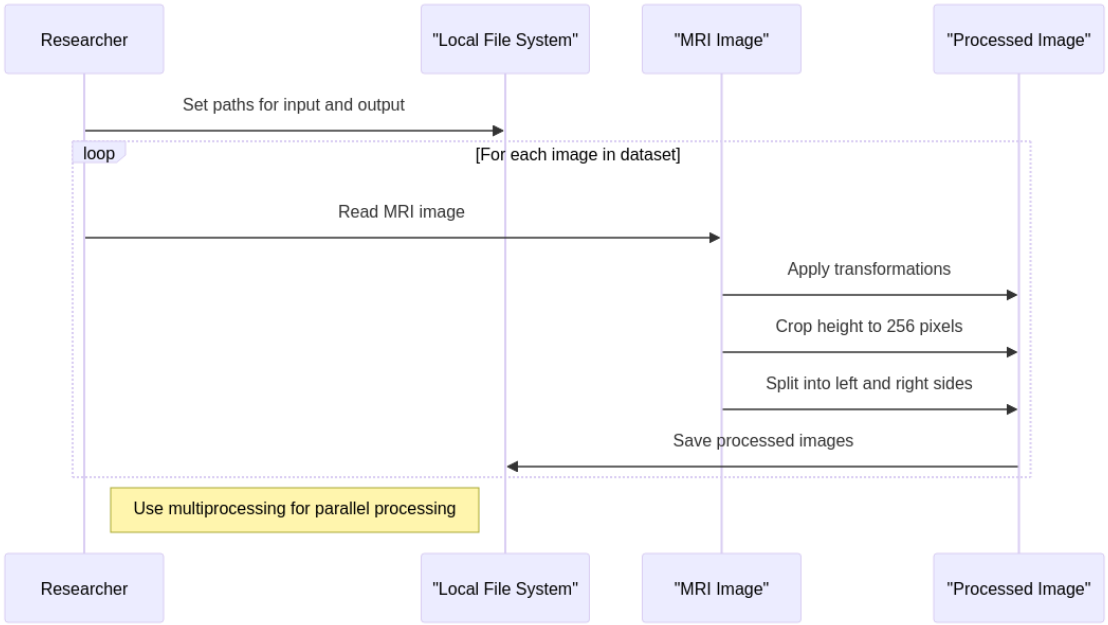
Data Preparation

1. Metadata Extraction:
    - A predefined list of DICOM metadata keys (metadatakeys) is used to extract relevant information from the DICOM files. This metadata is later saved as a JSON file.
2. Mapping File Paths to Sequence Names:
    - The script reads an Excel file (Breast-Cancer-MRI-filepath_filename-mapping.xlsx) to map file paths to sequence names and SeriesInstanceUIDs. This mapping is crucial for identifying and organising the data.
3. Reading and Processing DICOM Files:
    - For each case in the dataset, the script reads DICOM files, extracts metadata, and converts the images to the NIFTI format. It also saves the metadata for each sequence as a JSON file.
4. Computing Additional Images:
    - The script computes subtraction images (sub.nii.gz) by subtracting pre-contrast images from post-contrast images.
    - It also resamples T1-weighted images to match the dimensions of the dynamic sequence images.



Crop and Pad the Data

1. Function Definitions:
   - crop_breast_height: This function crops the height of the breast images to 256 pixels while trying to cover the breast area based on intensity localisation.
   - preprocess: This function performs the main preprocessing steps for each image in the dataset.
2. Preprocessing Steps in preprocess Function:
   - Setting Target Spacing and Shape: The script sets the target spacing and shape for the images.
   - Reference Image Resampling: It resamples the 'pre.nii.gz' image to the target spacing.
   - Transformation Pipeline: A series of transformations is applied, including resampling to the reference image, cropping, or padding to the target shape, and converting to a canonical orientation.
   - Height Cropping: The script crops the height of the image to 256 pixels.
   - Splitting Image into Left and Right Sides: The image is split into left and right sides, each cropped accordingly.



SWARM LEARNING IMPLEMENTATION GUIDE

Our swarm learning implementation cantered on co-training machine learning models for MRI data prediction across multiple physically distinct computers. This decentralised approach allowed each participating site to maintain proprietary data without direct sharing, fostering collaborative model training. The swarm learning network, featuring three nodes, engaged in model weight exchange at synchronisation events, occurring at defined intervals. Model weights were averaged at each event, enabling collaborative learning, with a weighted approach inspired by successful cancer research applications. Blockchain, specifically Ethereum, managed metadata for model synchronisation, ensuring transparency. Utilising Hewlett Packard Enterprise's framework, we integrated four key components: Swarm Learning process, Swarm Network process, identity management, and HPE license management, for an efficient and cohesive implementation of swarm learning principles. We provide a detailed description of our SL process, along with a small sample dataset, and instructions on how to reproduce our experiments using our code in https://github.com/KatherLab/swarm-learning-hpe/tree/exp_duke_comparison.

The operational workflow of Swarm Learning encompasses three primary phases: Initialisation and Onboarding, Installation and Configuration, and Integration and Training.

Initialisation and Onboarding:

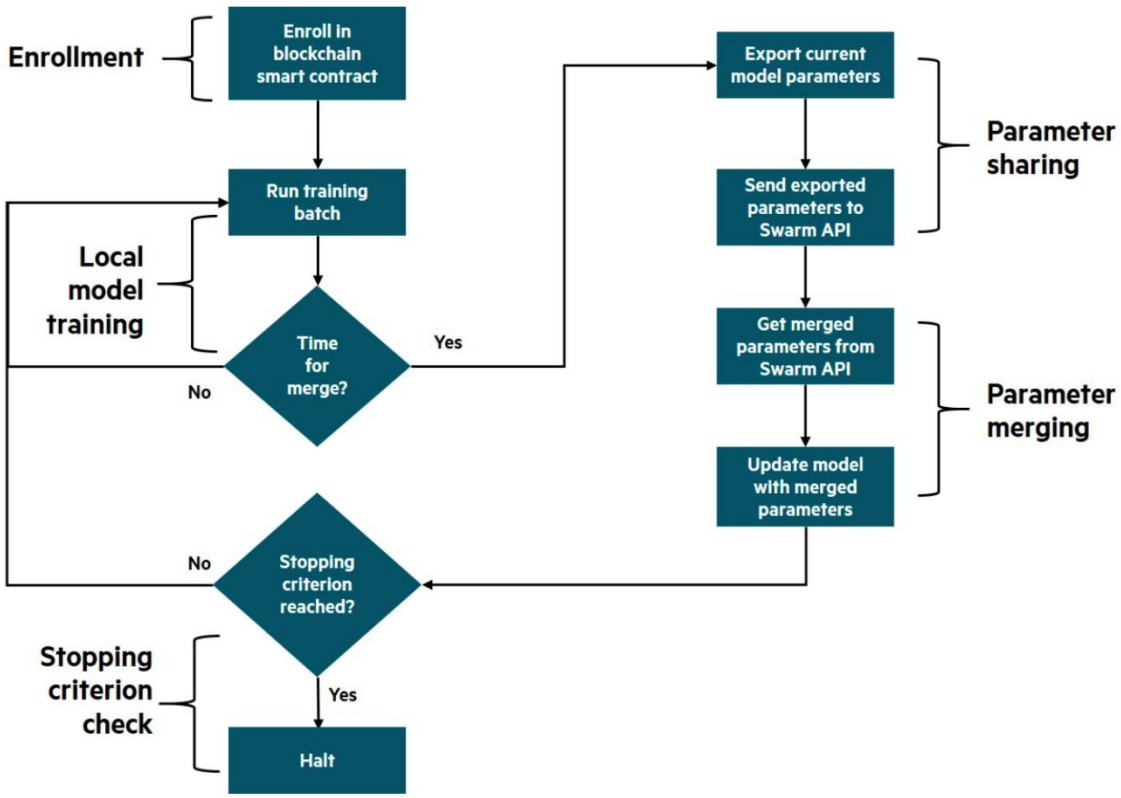
Onboarding is an offline process where entities interested in Swarm-based ML collaborate to establish operational and legal requirements. This phase involves formulating agreements for sharing parameters, ensuring node visibility across organisations, and defining the common model to be trained. Configurable parameters, a synchronisation frequency, and, if applicable, a reward system are agreed upon during this phase.

Installation and Configuration:

Following onboarding, consortium members download and install the Swarm platform on their nodes. This phase implements the finalised configuration of the Swarm Learning network. The platform is initiated, connecting nodes in the Swarm network, which overlays a blockchain network onto the underlying IP network. The boot-up sequence is organised, with sentinel nodes starting up first, followed by the remaining nodes.

Integration and Training:

Swarm Learning provides user-friendly APIs for seamless integration with frameworks like PyTorch or Keras (TensorFlow). The model training process involves the following stages:



- **Enrolment:**
Nodes enrol in the Swarm smart contract, recording essential attributes, including the URI, enabling other nodes to download their trained parameters.

- **Local Model Training:**
Nodes iteratively train their local models over multiple rounds (epochs), sharing parameter values with other nodes after each epoch.

- **Parameter Sharing:**
Nodes share parameters when a specified minimum threshold is reached. An elected leader oversees merging parameters from local training across all nodes after each epoch.

- **Parameter Merging:**
The leader combines parameter files using merge algorithms (mean, weighted mean, median), unifying parameter values from all nodes into a new file.

- **Stopping Criterion Check:**
Nodes evaluate the model using updated parameter values and communicate validation outcomes. If the stopping criterion is met, the Swarm Learning process concludes; otherwise, stages are repeated until criteria are fulfilled.

- **Monetisation and Rewards:**
If allowed, rewards reflecting each participant's contributions are computed and dispensed. The system state is assessed against the stopping criterion, repeating stages until criteria are met or

concluded. This comprehensive process underscores the decentralised and collaborative nature of Swarm Learning in machine learning model training.

INTEGRATION WITH EXISTING WORKFLOW

SL has the flexibility to convert any Python3-based ML program using Keras (with TensorFlow 2 backend) or PyTorch into a Swarm Learning ML program with minimal adjustments to the model training code. Simply incorporate the SwarmCallback API object into the existing code and refer to the provided examples in the Swarm Learning package for guidance. The ML program can utilise any parameterised supervised learning model, whether it's fully trainable or partially trainable, such as in the case of transfer learning. The Swarm Learning framework accommodates a variety of model architectures to suit individual needs.

To convert an ML program into a Swarm Learning ML program:

Import SwarmCallback class from the Swarm library.

from swarmlearning.tf import SwarmCallback

SwarmCallback represents a custom callback class equipped with a range of functions strategically employed at different stages of the training process. Throughout the training phase, this callback, along with its set of functions, offers insights into internal states and model statistics to the Swarm Learning framework. Specifically, SwarmCallback executes operations unique to Swarm Learning, such as sharing parameters with all network peers at the conclusion of a synchronisation interval.
On TensorFlow-based Keras platforms, SwarmCallback aligns with the Keras `tf.keras.callbacks.Callback` class. Keras automatically triggers the methods of this class at relevant stages during training.
In the case of PyTorch, which lacks a built-in Callback class, users are required to invoke the methods of this class explicitly on PyTorch-based platforms. The subsequent section details these methods, and their application is exemplified in the MNIST sample program.

Instantiate an object of the SwarmCallback class:

```
from swarmlearning.tf import SwarmCallback #for TensorFlow
from swarmlearning.pyt import SwarmCallback #for PyTorch

# Create Swarm callback
swarmCallback = SwarmCallback(syncFrequency=128,
minPeers=min_peers,
useAdaptiveSync=True,
adsValData=(x_test, y_test),
adsValBatchSize=8,
swarmCallback.logger.setLevel(logging.DEBUG)
```

Parameters:

- syncFrequency: Specifies the number of batches of local training to be performed between two swarm sync rounds. If adaptive sync enabled, this is the frequency to be used at the start. This is a mandatory parameter.
- minPeers: Specifies the minimum number of SL peers required during each synchronisation round for Swarm Learning to proceed further. This is a mandatory parameter.

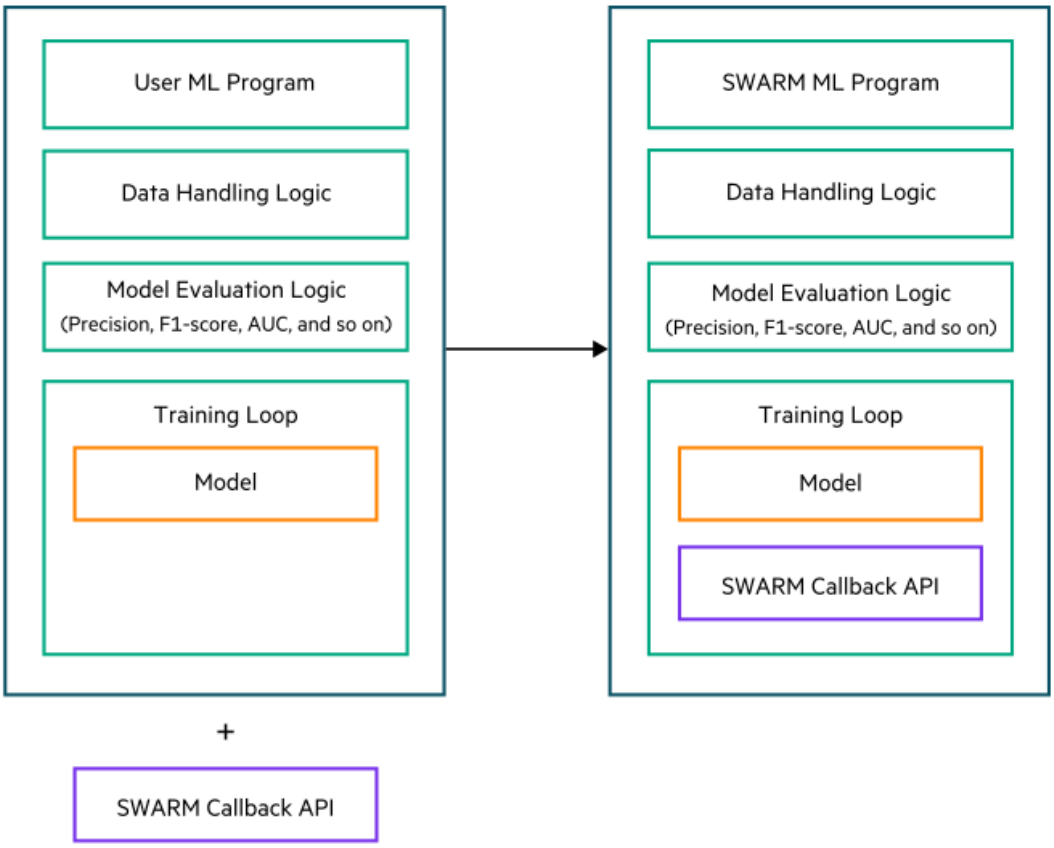
- useAdaptiveSync: Modulate the next syncFrequency value post each synchronisation round based on performance on validation data. The default value is false. This is an optional parameter.
- adsValData: Specifies the dataset for generating metrics for adaptive sync logic. It can be either an (x_val, y_val) tuple or a generator. This is an optional parameter for Swarm training.
- adsValBatchSize: Specifies the batch size for adsValData if batch processing is required. This is used when useAdaptiveSync is turned ON. This is an optional parameter for Swarm training.
- checkinModelOnTrainEnd: Specifies the merging behaviour of an SL node upon reaching the stopping criterion, indicating the completion of its local training, while awaiting the conclusion of training by all other peers. Throughout this waiting period, the SL node refrains from training the model with local data. The parameter determines the characteristics of the weights that this SL node contributes to the merging process and is optional. The permissible values for this parameter are:
    - inactive: The node abstains from contributing its weights to the merging process but actively participates as a non-contributing peer. It's important to note that specifying "inactive" for all nodes is an invalid configuration.
    - snapshot: The node consistently contributes the weights it possessed when reaching the stopping criterion and does not accept merged weights. DEFAULT
    - active: The node mimics an active training state (with no local model training) and contributes the current merged weights obtained from the Swarm merge. This setting allows the final model to potentially favour nodes that conclude their training at the end.
- trainingContract: Training contract associated with this learning. It is a user-defined string. This is an optional parameter. Default value is defaultbb.cqdb.sml.hpe.
- nodeWeightage: A number between 1–100 to indicate the relative importance of this node compared with others during the parameter merge process. This is an optional parameter. By default, all nodes are equal and have the same weight-age of one.
- mlPlatform: Specifies ML platform. Allowed values are either TF, KERAS or PYTORCH. This is an optional parameter. If TF platform is used, the default value is KERAS. If PYTORCH platform is used, the default value is PYTORCH.
- mergeMethod: Specifies what kind of merge method to be used in Swarm merge process. When SL node becomes leader, it reads this parameter and performs merge of intermediate trainable parameters (weights and biases). This is an optional parameter. Allowed values are as follows:
    - mean: Merges the model's trainable parameters using the weighted mean method. This is the default merge method.
    - coordmedian: Merges the model's trainable parameters using the weighted coordinate wise median method.
    - geomedian: Merges the model's trainable parameters using the weighted geometric median method.
- logger: Provides information about Python logger. This is an optional parameter. SwarmCallback class invokes info, debug, and error methods of this logger for logging. If no logger is passed, then SwarmCallback class creates its own logger from basic python logger. If required, user can get hold of this logger instance to change the log level as follows:

```
import logging
from swarmlearning.tf import SwarmCallback
swCallback = SwarmCallback(syncFrequency=128, minPeers=3)
swCallback.logger.setLevel(logging.DEBUG)
```

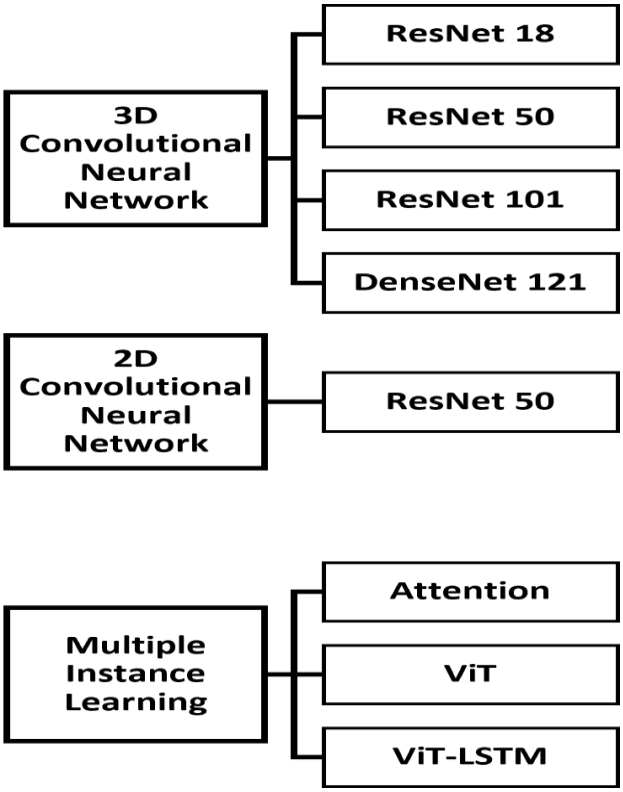
Use the SwarmCallback object for training the model.

- For Keras platforms:
  - Pass the object to the list of callbacks in Keras training code. The class methods are invoked automatically.
    - *model.fit(..., callbacks = [swarm_callback])*
    - *SwarmCallback can be included along with other callbacks also:*
    - *es_callback = EarlyStopping(...)*

    - model.fit(..., callbacks = [es_callback, swarm_callback])
- For PyTorch platforms, you must invoke the class methods:
    - *Call on_train_begin() before starting the model training:*
    - *swarmCallback.on_train_begin()*
    - *Call on_batch_end() after the end of each batch training:*
    - *swarmCallback.on_batch_end()*
    - *Call on_epoch_end() after the end of each epoch training:*
    - *swarmCallback.on_epoch_end(epoch)*
    - *Call on_train_end() after the end of the model training:*
    - *swarmCallback.on_train_end()*



TECHINIQUE SELCETION FOR WEEKLY SUPERVISED LEARNING



We can use various deep learning (DL) models selected through a thorough literature survey, encompassing radiology and technical publications. The chosen models include 2D-CNNs [5–8], 3D-CNNs [9,10], multiple-instance learning (MIL) approaches, and vision transformer (ViT) models [11,12], all applied to pre-processed difference images. Here's a detailed breakdown:

2D-CNNs:

Architecture: Based on the 2D-ResNet50 architecture.
Process: 3D MRI data is segmented into 32 slices, each with 256 x 256 pixels.
Approach: Utilises weakly supervised learning to label slices based on volume.
Prediction: Aggregates individual slice scores to generate volume-level predictions.

3D-CNNs:

Architectures Used: 3D-ResNet18, 3D-ResNet50, 3D-ResNet101, and 3D-DenseNet121.
Advantage: Adapts 2D-CNN designs for 3D data, capturing volumetric and spatial information.
MIL-based Methods:
ResNet18 Model: Pre-trained on ImageNet for feature extraction.

MIL Models:

Att-MIL (Attention-based MIL): Employs a multilayer perceptron with an attention mechanism.
ViT-MIL (Vision Transformer-based MIL): Utilises a transformer network with multi-headed self-attention.
ViT-LSTM-MIL: Combines Vision Transformer, LSTM, and MIL for improved classification on 2D slices.
These methods represent a diverse set of approaches for breast MRI classification, each tailored to address specific challenges and nuances in the data.

FINAL IMPLEMENTATION

In the intricate process of leveraging Swarm Learning for tumor prediction, the initial crucial phase involves meticulous preparation of the dataset. This entails thorough cleaning, formatting, and organising the data to meet the stringent requirements of the impending model. Subsequently, the Swarm Learning platform is meticulously set up, a task that encompasses the onboarding of multiple entities, the definition of operational and legal requirements, and the establishment of parameters for the decentralised system. With these foundational elements in place, the next pivotal step is the judicious selection of an appropriate machine learning method tailored to the unique characteristics of the problem at hand. Once this critical decision is made, the model training phase commences, enrolling nodes in the Swarm smart contract, and facilitating local training while synchronising parameters at designated intervals. With the trained model in hand, the focus shifts to deployment on a separate test dataset, allowing for a comprehensive assessment of its efficacy in predicting tumour presence. This intricate setup is a culmination of meticulous planning and execution, ensuring that each element is meticulously configured for optimal performance. Only after these meticulous preparations can the model be set in motion, ultimately yielding results that can be visualised and analysed to gauge the success of the Swarm Learning-based tumour prediction model.

CONCLUSION

This document describes how to train AI models with SL based on a tangible demonstration case of breast cancer tumour detection using MRI data. In conclusion, this report serves as a comprehensive guide for training AI models using Swarm Learning (SL) based on a tangible demonstration case. The step-by-step instructions provided cover the entire process, from initialisation and onboarding to installation, configuration, and integration. The practical application of SL in a real-world scenario is highlighted, emphasising the advantages of decentralised training without the need for centralised data sharing. The report underscores the flexibility of SL in accommodating multiple nodes, allowing for dynamic onboarding and dropout of participating institutions. Additional detailed information and comprehensive results about the presented topic can be found in our research paper titled 'Swarm Learning with Weak Supervision Enables Automatic Breast Cancer Detection in Magnetic Resonance Imaging.' Currently under revision for the Nature Communication journal, this paper delves deeper into the intricacies of our work and offers a more extensive exploration of the use case for training AI models with magnetic resonance imaging (MRI) data. It is worth noting that all collaborators associated with the ODELIA project actively contributed to the scientific endeavours outlined in this paper. Their collective involvement underscores the collaborative nature of this research, shedding light on the collaborative efforts that have shaped the insights and findings presented in the publication. Overall, this report equips practitioners with the knowledge and practical insights needed to implement SL for training AI models, fostering advancements in machine learning while preserving data integrity and privacy.

REFERENCES

[1] Boberg KM, Chapman RW, Hirschfield GM, Lohse AW, Manns MP, Schrumpf E, et al. Overlap syndromes: the International Autoimmune Hepatitis Group (IAIHG) position statement on a controversial issue. J Hepatol 2011;54:374–85.

[2] Zen Y, Harada K, Sasaki M, Tsuneyama K, Matsui K, Haratake J, et al. Are bile duct lesions of primary biliary cirrhosis distinguishable from those of autoimmune hepatitis and chronic viral hepatitis? Interobserver histological agreement on trimmed bile ducts. J Gastroenterol 2005;40:164–70.

[3] Nakanuma Y, Zen Y, Harada K, Sasaki M, Nonomura A, Uehara T, et al. Application of a new histological staging and grading system for primary biliary cirrhosis to liver biopsy specimens: Interobserver agreement. Pathol Int 2010;60:167–74.

[4] Nam D, Chapiro J, Paradis V, Seraphin TP, Kather JN. Artificial intelligence in liver diseases: Improving diagnostics, prognostics and response prediction. JHEP Rep 2022;4:100443.

[5] Schelb P, Kohl S, Radtke JP, Wiesenfarth M, Kickingereder P, Bickelhaupt S, et al. Classification of Cancer at Prostate MRI: Deep Learning versus Clinical PI-RADS Assessment. Radiology. 2019;293: 607–617.

[6] Ranjbarzadeh R, Bagherian Kasgari A, Jafarzadeh Ghoushchi S, Anari S, Naseri M, Bendechache M. Brain tumor segmentation based on deep learning and an attention mechanism using MRI multi-modalities brain images. Sci Rep. 2021;11: 10930.

[7] Zhu W, Sun L, Huang J, Han L, Zhang D. Dual Attention Multi-Instance Deep Learning for Alzheimer's Disease Diagnosis With Structural MRI. IEEE Trans Med Imaging. 2021;40: 2354–2366.

[8] Guadagnolo D, Mastromoro G, Di Palma F, Pizzuti A, Marchionni E. Prenatal Exome Sequencing: Background, Current Practice and Future Perspectives-A Systematic Review. Diagnostics (Basel). 2021;11. doi:10.3390/diagnostics11020224

[9] Zeineldin RA, Karar ME, Coburger J, Wirtz CR, Burgert O. DeepSeg: deep neural network framework for automatic brain tumor segmentation using magnetic resonance FLAIR images. Int J Comput Assist Radiol Surg. 2020;15: 909–920.

[10] Urakubo H, Bullmann T, Kubota Y, Oba S, Ishii S. UNI-EM: An Environment for Deep Neural Network-Based Automated Segmentation of Neuronal Electron Microscopic Images. Sci Rep. 2019;9: 19413.

[11] Ranjbarzadeh R, Bagherian Kasgari A, Jafarzadeh Ghoushchi S, Anari S, Naseri M, Bendechache M. Brain tumor segmentation based on deep learning and an attention mechanism using MRI multi-modalities brain images. Sci Rep. 2021;11: 10930.

[12] Zhu W, Sun L, Huang J, Han L, Zhang D. Dual Attention Multi-Instance Deep Learning for Alzheimer's Disease Diagnosis With Structural MRI. IEEE Trans Med Imaging. 2021;40: 2354–2366.