DEEPHEALTH

D4.1 Integration of DeepHealth platforms and use cases

Project ref. no.	H2020-ICT-11-2018-2019 GA No. 825111
Project title	Deep-Learning and HPC to Boost Biomedical Applications for Health
Duration of the project	1-01-2019 – 31-12-2021 (36 months)
WP/Task:	WP4/ T4.1, T4.2, T4.3, T4.3, T4.5, T4.6, T4.7
Dissemination level:	CO
Document due Date:	31/03/2020 (M15)
Actual date of delivery	02/04/2020 (M16)
Leader of this deliverable	WINGS
Author (s)	Aimilia Bantouna (WINGS), Ioannis Stournaras (WINGS), Vera Stavroulaki (WINGS), Amalia Ntemou (WINGS), Nelly Giannopoulou (WINGS), Panagiotis Demestichas (WINGS), Evaggelia Tzifa

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 82511

	(WINGS), Katerina Demesticha (WINGS), Ioannis Stenos (WINGS), Ioannis Tzanetis, Panagiotis Vlaheas (WINGS), Paraskevas Bourgos (WINGS), Kostas Tsoumanis (WINGS), Ioanna Drigopoulou (WINGS), Ioannis Dimitriadis (WINGS), Polina Dimiropoulou (WINGS), Pavlos Fragkogiannis (WINGS), Andreina Chietera (THALES), Ynse Hoornenborg (PHILIPS), Luca Pireddu (CRS4), Barbara Cantalupo (UNITO), Diego Benedicto Consejo (EVERIS), Dana Oniga (SIVECO), Jean-Philippe Poli (CEA), Tomas Teijeiro (EPFL), Mariam De la Iglesia Vayá (FISABIO)
Version	V1.0

Document history

Version	Date	Document history/approvals
0.1	14/02/2020	First draft contents
0.2	18/02/2020	Revised Table of Contents
0.3	06/03/2020	1st merged content
0.4	10/03/2020	Complementary inputs
0.5	12/03/2020	2nd round of integration
0.6	17/03/2020	1st round of reviews / revisions
0.7	23/03/2020	Integration of new inputs addressing the comments
0.8	24/03/2020	Executive summary and conclusion
0.9	30/03/2020	Internally reviewed and full draft for final review
1.0	02/04/2020	Ready-to-submit version

DISCLAIMER

This document reflects only the author's views and the European Community is not responsible for any use that may be made of the information it contains.

Copyright

© Copyright 2019 the DEEPHEALTH Consortium

Table of contents

1 Executive summary

This is the 1st deliverable of WP4 “Integration of libraries and use cases in Application Platforms”. Its aim is to encompass all partners platforms and use cases set in the project and to report all integration efforts for each associated platform and its related use cases.

To this end, the document has been organized around the seven (7) platforms considered in the project, i.e.

- Open Innovation platform from Philips,
- MigraineNet platform from WINGS,
- ExpressIF from CEA,
- PIAF platform from Thales,
- Open DeepHealth platform from UNITO,
- Digital Pathology platform from CRS4 and,
- everisLumen from EVERIS.

For each platform, the platform owners have

- described the platform architecture as adjusted for the purposes of DeepHealth,
- explained how each platform will be used to support the use cases that will serve, i.e. reported on the inputs to the platform, the output to be provided to the end-users (use case owners), the models to be trained and/or tested and the technologies to be used for each considered use case;
- analysed the so-far considered integration approach for their platform with the DeepHealth outcomes (DH Front-end, EDDLL and ECVL) as a whole or per use case (when applicable and differentiated); and
- summarized the current integration status and the next steps for compliting the integration activities.

The report concludes in section 9.

2 PF1 - Open Innovation platform (Philips)

2.1 Description of the platform as adjusted for the purposes of DeepHealth

For DeepHealth, the Open Innovation Platform has been transformed into a client server architecture where the model runs on a backend server but the results are visualized via a web browser on the client. See Figure 1 for details.

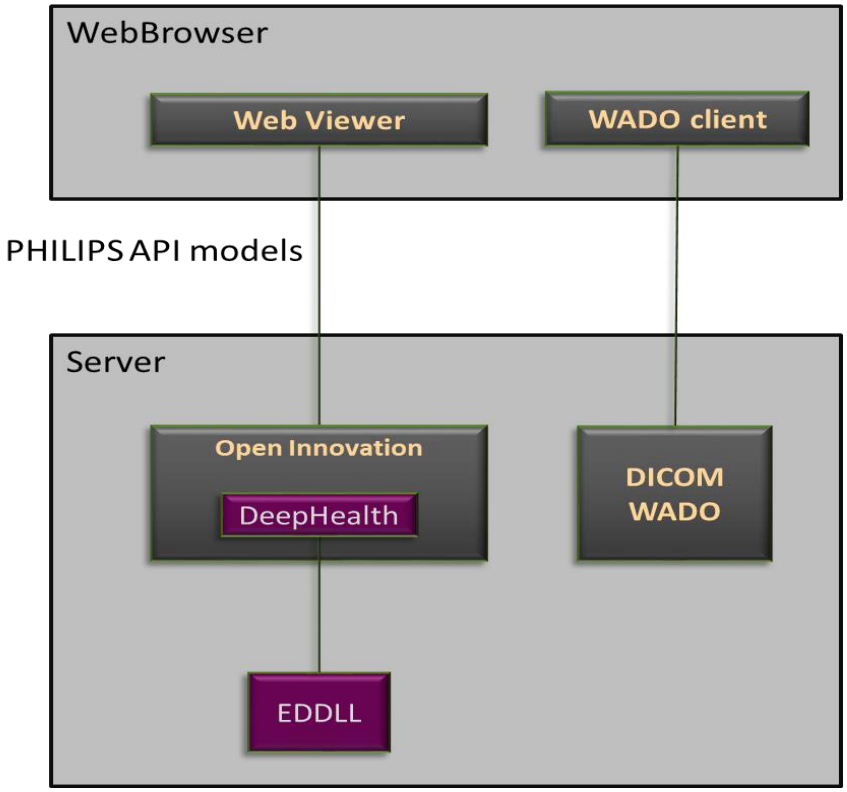


Figure 1. Open Innovation client-server design

The backend is hosted inside a number of Docker containers. Access to the image data is accomplished by providing a DICOM WADO endpoint. WADO is an industry standard, part of the bigger DICOM standard, to retrieve image data via a REST API.

The Open Innovation backend services will provide REST API's for access to EDDL from the client side application. These existing REST API's are extended in the context of DeepHealth, to be able to use EDDL inside the backend. Care has been taken to include the input and output data structures for every UC in DeepHealth. For that we have set up a flexible general data structure, with specifics available per data type.

The client side of the application is hosted on a web server, which in turn is hosted as a Docker container on the backend server. Please note that Philips does not provide the actual server, but we deploy a software only solution to the relevant partners, to safeguard data privacy.

A license agreement has been setup with the relevant partners for usage of this software within this project.

2.2 Description of the platform provisions in the context of DeepHealth use cases

Open Innovation platform provides advanced medical visualization capabilities and the ability to extend these capabilities easily with new research algorithms. Possible algorithms include: segmentation or registration. Its capabilities include support for all imaging modalities, and with data of multiple stakeholders. More information with respect to the use of the platform for each use case follow in the below sub-sections.

2.2.1 UC5: Deep Image Annotation

The Open Innovation Platform will be used for the Deep Image Annotation use case.

Input to the model are Chest X-Ray and Computed Tomography (CT) Images. These images are stored in the DICOM format. They will be read by the Open Innovation Platform and processed to match the input criteria of the AI Model. The output, a sequence of words, will be displayed to the user.

No trained model in the ONNX file format is available at this moment, integration will commence when the use case provider has made a trained model available.

2.2.2 UC7: Major Depression

The Open Innovation Platform will be used for testing the Major Depression use case.

Input to the model are T1, T2 and TTI MR DICOM images. These images are stored in the DICOM format. They will be read by the Open Innovation Platform and processed to match the input criteria of the AI Model. Additionally, some measures values per subject are loaded. The output, a classification of the decease, will be displayed as text to the user.

No trained model in the ONNX file format is available at this moment, integration will commence when the use case provider has made a trained model available.

2.2.3 UC8: Dementia

The Open Innovation Platform will be used for testing the Dementia use case.

The inputs to the model are T2 MRI DICOM images. These images are stored in the DICOM format. They will be read in by the Open Innovation Platform and processed to match the input criteria of the AI Model. Additionally, there is a list of biological markers per subject, in CSV format. The output, a classification of the form of dementia, will be displayed as text to the user.

No trained model in the ONNX file format is available at this moment, integration will commence when the use case provider has made a trained model available.

2.2.4 UC9: Lumbar spine

The Open Innovation Platform will be used for annotation and testing platform the Lumbar spine use case.

Input to the model are various types of MRI DICOM Images. These images are stored in the DICOM format. They will be read by the Open Innovation Platform and processed to match the input criteria of the AI Model. The output, a classification per pixel, will be displayed to the user as a coloured overlay.

No trained model in the ONNX file format is available at this moment, integration will commence when the use case provider has made a trained model available.

Next to this, Open Innovation Platform will provide a means to annotate the images by human doctors for the purpose of determining the ground truth. As the number of images to annotate is rather large, workflow optimization here will benefit the applicable doctors greatly in their classification task.

The images for this use case are stored in an X-NAT server. Before image retrieval, this X-NAT server is to be queried to find the image(s) associated with the selected case. The image retrieval itself is done via the WADO standard.

When the case image is being displayed, the user is presented with a UI to define the classification of a region of the image. To do this, the user will draw a coloured overlay on top of the anatomical image. The chosen classification will determine the colour which will be drawn.

After the user is satisfied with the result, the overlaid coloured image is stored to the database. The image is stored with the classification index as the pixel value. For display purposes, a lookup table to colours is added to the image.

Philips is currently in the process of implementing this and plans to have a demo available for the M18 review.

2.2.5 UC10: Alzheimer's Disease

The Open Innovation Platform will be used for testing the Alzheimer's Disease use case.

The input to the model are MRI DICOM images of the brain. These images are stored in the DICOM format. They will be read by the Open Innovation Platform and processed to match the input criteria of the AI Model. The output is a set of measurements about the size of the white and grey matter. These measurements will be displayed to the user as text.

No trained model in the ONNX file format is available at this moment, integration will commence when the use case provider has made a trained model available.

2.2.6 UC11: Urology

Image Analysis and prediction for Urology will use a dataset comprising of at least 500 cases of renal and adrenal pathology (adrenal adenoma and renal clear cell carcinoma) and 500 cases of normal kidney and adrenal anatomy. The dataset sample will constitute of CT scans and results for the specific cases- DICOM images and biopsy proven results for the pathology involved.

The Open Innovation Platform will be used for testing the Urology use case. The first phase of the integration of Open Innovation platform with UC 11 is to adapt the data to the platform. SCTHB is in the process of gathering the data sets required, of labelling them and pseudonymizing them.

The inputs to the model are CT DICOM images of the abdomen, with and without contrast applied. These images are stored in the DICOM format and will be read by the Open Innovation Platform and processed to match the input criteria of the AI Model. The second phase is to acquire the necessary infrastructure- servers and computers realizing a dedicated closed network. The output is to obtain classification and segmentation of data introduced, i.e., a classification of the tumour type, and will be displayed as text to the user.

This use-case will use also the Deephealth toolkit to train the model for image classification and segmentation in the process of diagnose of renal tumours. No trained model in the ONNX file format is available at this moment, and thus, the integration will start when the use case provider has made a trained model available.

2.2.7 UC12: Skin Cancer

The Open Innovation Platform will be used for testing the Skin Cancer use case. No trained model in the ONNX file format is available at this moment, therefore the integration will commence when the use case provider has made available a trained model.

Although the images to be used are all in the public domain and thus could be handled by the platform owner as well, the platform owner (Philips) required their conversion to the DICOM image format, as this is the native input format of the platform. These images have now been converted and tested to function correctly in the Open Innovation Platform.

In addition to these images, some metadata will be used as input. The output of the model will be the outline of the lesion, which will be shown as a coloured edge on the image to the user. Next to that, an accuracy number will be displayed to the user as text.

2.2.8 UC14: Multiple sclerosis

The Open Innovation Platform will be used for testing the Multiple sclerosis use case. In particular, a trained model for the segmentation of Multiple Sclerosis lesions on brain MRI images will be provided using the ONNX file format. The platform will make possible to load the model and the images and visualize the results of the segmentation task. The input to the model are T1 anatomical MRI DICOM images. These images are stored in the DICOM format. They will be read by the Open Innovation Platform and processed to match the input criteria of the AI Model. The output, a contour of the lesion, will be displayed as a coloured edge on the image to the user. No trained model in the ONNX file format is available at this moment, so integration will commence when the use case provider has made a trained model available.

2.3 Description of the platform interactions with the DeepHealth toolbox

2.3.1 DH Front-end

Not applicable.

2.3.2 EDDLL

Although a fully trained EDDL model is not available, given the examples on GitHub, we did integrate with the library.

Our backend service is written in C#. As it runs inside Docker containers, it is natural to take a Linux OS. This means the code runs on .NET Core. This limits the possibilities to interface with the C++ written EDDL. The full .NET Framework on Windows provides Managed-C++ support for cases like this. This is however not available in .Net Core.

To work around this limitation, we decided to develop a stack of two wrappers, see Figure 2. On top of EDDL, we first created a plain C wrapper, the CEDDL. This wrapper is published on GitHub as open source software. The second wrapper is C# code, which refers to the CEDDL using the P/Invoke mechanism built into .NET Core. This second wrapper will connect the CEDDL methods and data structures to the Philips proprietary C# code.

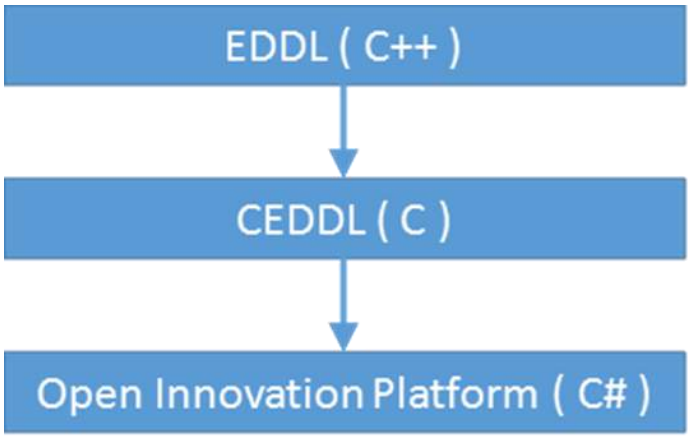


Figure 2. Stack of EDDL wrappers

We have successfully implemented this stack of wrappers for the relevant examples as they are specified in the EDDL repository. A continuous integration pipeline has been set up within Philips, to verify the correct functioning of this integration, while the EDDL is evolving. This ensures we get early feedback when something breaks.

For most use cases, Open Innovation Platform is only involved in the testing phase. To incorporate the trained network of other partners, we rely on the import of models in the ONNX file format. The import and export of this file format is currently under development within the EDDL library. Once this is available, support for it will be added to the Open Innovation Platform.

2.3.3 ECVL

Not applicable.

2.4 Current status and next steps

For the current status, see section 2.3.2, stated above.

Next steps are to develop a ground truth application for FISABIO's UC9 "Lumbar spine". In particular, we plan to demonstrate its ability to manually classify spine images in the context of FISABIO. Alternatively, we can show the integration of the EDDL examples in the continuous integration pipeline. This ground truth application will enable medical staff to annotate the DICOM images with the correct classification. As this is manual labour for several physicians and the number of images is rather large (several thousands), an easy and efficient workflow is essential here. Also, time from these medical experts is precious.

When a trained model in ONNX format becomes available for loading, we can start loading models from the use cases. This will also open up the possibility to work on the pre- and post-processing tasks that most use cases require.

There are several use cases that (partially) use data which is available in the public domain, which will ease the debugging the processing code we need to write. Arrangements therefore have been made, to acquire this data also in the Philips premise. All of these datasets are already being parsed correctly in the Open Innovation Platform, in the sense that they display correctly.

Plans for the Midterm review (M18): During the midterm review in M18, the Open Innovation Platform is planning to demonstrate its ground truth application, which is being developed for FISABIO's UC9 "Lumbar spine". In particular, we plan to demonstrate its ability to manually classify spine images in the context of FISABIO. Alternatively, we can show the integration of the EDDL examples in the continuous integration pipeline.

3 PF2 - MigraineNet platform (WINGS)

3.1 Description of the platform as adjusted for the purposes of DeepHealth

The application and the associated cloud platform have been built using open source cloud infrastructure components. The integral component supporting MigraineNet's functionalities is the MigraineNet Proxy Server which acts as an intermediary between the mobile application and all backend components in the cloud side. Communication with the Proxy Server is performed through RESTFul APIs.

The Knowledge building and Inference components are being extended in the context of DeepHealth. Both components will follow the microservice architecture and will be hosted inside Docker containers with pyEDDL installed in a python environment. In addition, any updates to these backend components will only require the Proxy Server to be updated and not the Android application.

Finally, MigraineNet are being extended to support the connection of wearable devices as part of UC1 (Seizures) and UC7. Thus, the database model will also be updated to support data from wearables that will be utilized to enhance the performance of the models incorporated in the Knowledge and Inference Components.

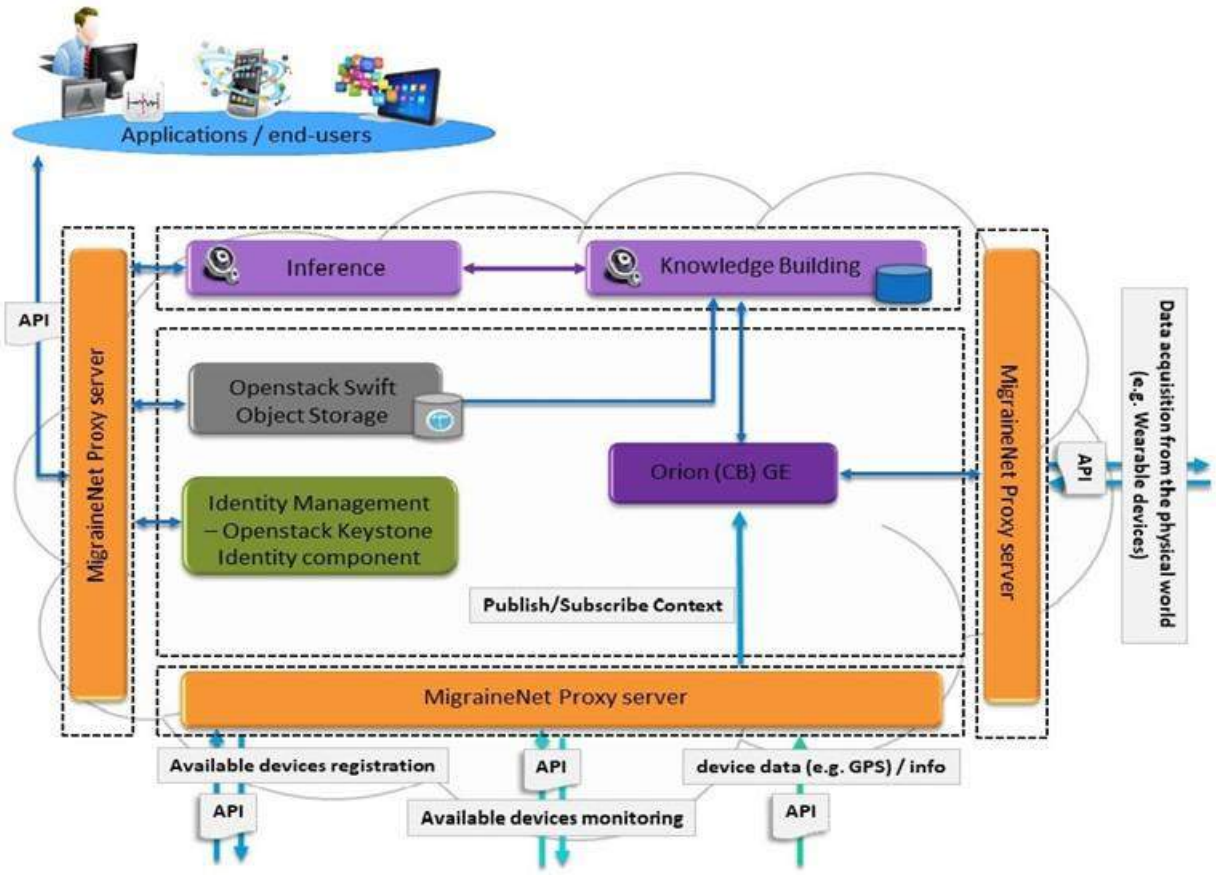


Figure 3. Overall architecture of MigraineNet

Figure 3 depicts an overview of the MigraineNet backend architecture and the interactions among its components.

3.2 UC1: Migraine & Seizures prediction

3.2.1 Description of the platform provisions in the context of the UC

3.2.1.1 Migraine predictions

MigraineNet is a personalized healthcare solution for migraines. It is a cloud-based mobile service that provides patients with deep insights on the causes and on the most suitable ways to obtain some relief during a migraine incident. Furthermore, MigraineNet application is enriched with the functionality of learning the user's daily habits and predicting the probability of getting a migraine within the day. In particular, this functionality is supported by three application specific information models and two components: a) the Knowledge building component and b) the Inference component.

MigraineNet system comprises of specific information models, utilized to provide the information exchanged between the various components of the system. Initially, the user is asked to create his profile and insert some personal information that consist the **User Profile Information Model** which is quite static**.** The information is organised in the sub-sections a) "Profile Information" (e.g. sex, age, weight), "Environmental Information" (e.g. address, noise, humidity, pollution level), c) "Physiological Information" (e.g. sleep habits), d) "Dietary Habits", e) "Medical Information" (e.g. eye problems, sleeping disorders) and f) "Migraine Information" (e.g. migraine history, acute treatment) so as to facilitate the user's answers. Once the user is registered with the system, he/she will be able to provide information with respect to their daily habits that consist the **Daily Form Information Model,** which enables monitoring features that may result in a migraine attack**.** The daily requested information is divided in the sub-sections a) "Physical conditions" (e.g. stress, depression, sleep), b) "Eating habits" (e.g. consumed drinks/foods) c) "Environmental aspects" (e.g. odours, flickering lights) and d) "Medical aspects" (e.g. medication). Finally, the user has the ability to register a new migraine incident and provide information that consist the **Migraine Incident Information Model** which could potentially identify causes and/or treatments. To minimize the information requested to the user when he/she logs a new migraine incident, the information model is organized in the sub-sections a) "Description" (e.g. start/end time, work hours), b) "Symptoms" (e.g. center of pain, pain intensity) and c) "Actions" (e.g. bath, painkillers).

The Knowledge building component receives as input the information provided by the user through the daily form. In addition, information on having a migraine incident within the day or not is retrieved by the Migraine Incident Information Model. The integral part of the component is a mechanism that utilizes the aforementioned information to build knowledge on the user's daily habits and their pattern so as to predict the probability of the user to get a migraine in the coming days. The mechanism is based on Recurrent Neural Network architecture, namely the "Long Short-Term Memory Units" (LSTMs). Recurrent neural networks have demonstrated great success in handling time series data due to their memory capability, i.e. their capability to take into account both the current (t) and the previous (t-n) timestamps when deciding on the next action.

The inference component is designed and developed to complement the knowledge building mechanism described above and acts as the decision-making process when predicting the user's next migraine incident. The mechanism is triggered when the user submits a new "daily form" and is responsible for inferring the possibility of the user to get a migraine within the next 6 days. The outcome is finally announced on the user's "dashboard" in the MigraineNet application and is displayed on the user's calendar along with the probability of getting a migraine.

3.2.1.2 Seizure predictions

A similar approach to the one currently used in migraine predictions, is investigated to be applied to epileptic seizures prediction as well. Recently many studies have shown that non-invasive EEG signals can identify unusual patterns that could forecast the timing of next seizure attacks. However, this medical approach requires EEG electrodes placed on patients' scalp which is an unpleasant method for the subject under monitoring. Many people with epilepsy notice that they are more likely to experience episodes when they are tired, stressed, or have missed medication. In addition, the

uncertainty about when the next seizure will happen as well as the fact that these attacks could have serious consequences (such as injury or death) are of the most distressful issues affecting the patients and their relatives' lives.

For all these reasons, the main aim of utilizing PF2 in the seizures in the context of UC1 is to provide a user-friendly personalized healthcare solution for non-hospitalized epileptic patients who would like to better monitor their seizures and stay safe. This cloud-based system will continuously collect data from mobile app and wearable devices in order to forecast when seizures are more likely to happen. More specifically, the system will use advanced machine learning to a) detect unusual patterns before the episode that may be associated with seizures, and b) immediately notifies caregivers triggering an alarm.

The system will take into account the following different types of information:

- Data from wearable 24h EEG in the case of patients suffering from intractable epilepsy.
- Data from digital questionnaires that patients should provide inputs related to adherence to treatment, stress levels, etc. on a daily basis.
- Data from wearable devices, such as smart watches for recording heart rate and potentially electrodermal activity.

3.2.2 Description of the platform interactions with the DeepHealth toolbox

3.2.2.1 DH Front-end

Not applicable.

3.2.2.2 EDDLL

The migraine prediction module will be addressed using an Encoder-Decoder LSTM like architecture implemented using the Python binding of the EDDLL library. The model is currently implemented in Deeplearning4j. Once recurrent layers become available on the EDDL library, the model will be ported to the EDDLL. Both the Knowledge Building and Inference component will follow the microservice architecture and will be hosted inside Docker containers with all the required dependencies, including pyEDDL, installed and ready to run.

3.2.2.3 ECVL

Not applicable.

3.3 UC7: Major Depression

3.3.1 Description of the platform provisions in the context of the UC

The main aim of utilizing PF2 in the scope of UC7 is to develop a mobile app that will act as personalized disease management tool for people who suffer from symptoms of depression. The benefits of this cloud-based mobile service are **(a)** the **prediction of the progression** of depressive symptoms, **(b)** the **detection of the events** that lead to specific types of progression of depressive symptoms and **(c)** aiding the doctors with personalized **recommendations** on what the user should do or avoid.

This service follows a similar approach to that described above for migraine prediction (UC1). Unlike the aforementioned use case and this neurological disease, depressive episodes have a low frequency and long duration (compared to migraine attacks) and they are characterized by blurry starting and end points. For this reason, PF2 is adapted to predict the progression of symptom severity instead of the onset of the next pathological incidence. To further adapt the platform to the use case, digital questionnaires or wearable devices that collect main vital signs routinely from the human body (e.g. heart rate, sleep and / or movement parameters), are not only applied separately but unitedly, enabling an ecological momentary assessment (EMA) of behaviours and experiences to optimize prediction accuracy.

The “Depression App” will be based on specific information models, utilized to provide the information exchanged between the various components of the system focusing on the collection of the following information from the user:

a. **At outset information / Profile forms**
   At the beginning of the study patients are invited to answer different kind of questions (including both questions usually asked to new patients by the doctor and more). Information about the medical and mental health, treatment history and status, sociodemographic data as well as psychological constructs, such as personality inventory and emotion regulation skills are some of the parameters envisioned to be included in this category.
b. **At intervals / Weekly forms**
   To learn how the depressiveness evolves for each individual patient, depressive symptoms are rated in a weekly manner. Further parameters included in this category are possible predictors of depressiveness that change at a slow rate. This type of questionnaire includes questions related to adherence to treatment according to the treatment plan, major life stressors, or physical health parameters.
c. **EMA / Daily forms**
   This category includes questions about momentary states of the patients. It is currently envisioned that data from wearable devices will trigger the collection of an ecological momentary assessment (EMA). Acute stressors, behaviours, situational descriptors and thoughts might constitute the variables collected.
d. **Continuous information / Data from wearables**
   Physiological parameters, such as heart rate, sleep, movement, acceleration and electrodermal activity (as a potential indicator of stress levels) are investigated as possible parameters to be recorded and to form the basis upon which EMAs will be collected.

The current depression application will be based on the functionality of learning the users’ daily habits and physiological parameters in order to predict the progression of depressive symptoms and detect the events that lead to specific types of progression. In particular, the basic functionality is supported by the application-specific information models as well as the Knowledge building and the Inference component. The system also has the capability to correlate patients according to their characteristics in order to examine if patients with similar profiles re-act in the same way to similar treatments and thus, exploit the data of one user to treat the depression of another.

The application will be able to collect and analyse some or all of the above described information and provide the doctors useful insights and personalized recommendations for their patients. These may include the prediction of the progression of depressive symptoms, how the depression will evolve if the patient keeps following the specific treatment plan and more. These outputs will be provided **only to the doctors** and only as recommendations and at no point to the patient. It is the role of the doctors to evaluate the personalized recommendations based on their medical experience and to guide the patients’ treatment plan including which life stressors to avoid or what activities to engage in so as to decrease depressive symptoms.

3.3.2 Description of the platform interactions with the DeepHealth toolbox

3.3.2.1 DH Front-end

Not applicable.

3.3.2.2 EDDLL

Deep embedded clustering will be utilized to address the challenge of data limitation and to identify patients with similar profiles. Continuous information from wearables will be utilized to train a CNN model that will enable us to automatically acquire a richer low-dimensional representation of the high-dimensional input features and would potentially add more value for diagnosing the current depression level of each patient. Feature extraction from the CNN model will allow us to address 2 main challenges posed in the case of an LSTM approach: a) LSTMs can only remember sequences of 100s but not 1000 or more due to the vanishing gradients problem; and b) LSTM requirement for

lots of resources to train a model when using high-volume inputs such as those produced by wearables.

Accordingly, information from similar groups of patients will be combined with the features extracted from the CNN model to train a LSTM model that will predict the progression of depression. In a nutshell, the CNN model will allow the feature extraction from longer sequences and LSTMs will be used to further analyse the temporal dependencies of the extracted features and data from questionnaires and be investigated for providing doctors personalized recommendations for their patients.

All models will be implemented, trained and tested using the Python binding of EDDLL.

3.3.2.3 ECVL

Not applicable.

3.4 UC13: Epileptic Seizures detection

3.4.1 Description of the platform provisions in the context of the UC

This use case aims at

a) the detection of epileptic seizures on EEG signals and
b) the online seizure prediction with user-dependent models.

EPFL addresses the problem of seizure detection. To that end, EPFL has developed a CNN architecture for epileptic seizure detection from EEG signals, that works directly on the raw signal without requiring any specific pre-processing. The architecture works with segments of 5s duration and it performs a binary classification, producing a “True” output when a seizure pattern is detected on the input.

WINGS has developed a CNN-LSTM architecture for epileptic seizure prediction from EEG signals that works directly on the raw signal without requiring any specific pre-processing or feature extraction step. The architecture works with segments of 5 seconds duration and performs a binary classification, identifying whether a segment belongs to the Inter-ictal or the pre-ictal brain state.

3.4.2 Description of the platform architecture as adjusted for the purposes of the UC

The following adjustments will be made to support the functionality described above:

- Create a new database model that will be responsible for storing the data in a particular format.
- Create a new front-end that will allow the user to submit their own data to the platform. The user will be able to choose between two-options a) utilize the existing model from EPFL to detect seizures in the specified files or b) train a new model utilizing the submitted data to create a personalized seizure prediction tool.
- The knowledge building and Inference components will be modified to incorporate the functionalities developed for this particular use case.

3.4.3 Description of the platform interactions with the DeepHealth toolbox

3.4.3.1 DH Front-end

Not applicable.

3.4.3.2 EDDLL

For the seizure detection, the CNN model provided by EPFL will be trained using the EDDLL at the EPFL facilities and using public and private EEG datasets. Then, the model will be stored using the EDDLL functionality, and transferred to WINGS. At inference time, MigraineNet’s Inference component will load the trained model using the EDDLL functions, and apply the prediction function to new EEG segments. The CNN architecture is currently implemented in Keras/Tensorflow and will

be ported to the EDDLL. Import of trained model using ONNX file format will be utilized. The functionality is not currently supported by EDDLL.

For the seizure prediction, the CNN-LSTM model will be trained using the EDDLL at WINGS facilities using only public EEG datasets provided by physionet[1]. The Knowledge building component will be extended to enable the training of new models on user's request. Thus, the user will be able to train a personalized seizure prediction model on request by submitting the files to the platform. Finally, the Inference component will load either a pretrained model provided by WINGS or the custom personalized model specified by the user and apply the prediction function to new EEG segments. The CNN-LSTM architecture is currently implemented in Pytorch and will be ported to EDDLL using the Python binding once specific modules become available on the EDDL library.

3.4.3.3 ECVL

Not applicable.

3.5 Current status and next steps

Overall, the following activities have already taken place:

- Compilation of EDDLL earlier version for CPU and GPU and testing with basic examples (in the absence of LSTM functionality necessary for the MigraineNet platform)
- Compilation of PyEDDLL earlier version for CPU and GPU and testing with basic examples (in the absence of LSTM functionality necessary for the MigraineNet platform)

While the next steps for the integration of the EDDLL with the MigraineNet platform include:

- Docker image with PyEDDLL for CPU
- Docker image with PyEDDLL for GPU
- Integration of Docker images in MigraineNet as soon as the LSTM functionality is provided.

Plans for midterm review (M18): During the midterm review in M18, the platform is envisioned to demonstrate its integration with the EDDLL using either UC1 and its functionality for migraine prediction or UC13 and its functionality for seizure detection depending the functionalities and the models already provided through EDDLL by M17.

[1] https://physionet.org/content/chbmit/1.0.0/

4 PF3 - ExpressIF (CEA)

4.1 Description of the platform as adjusted for the purposes of DeepHealth

4.1.1 General description

ExpressIF™ is a software developed by CEA LIST since 2010 which aims at modeling expert knowledge with rules written as close as natural language as possible in order to reproduce automatically a human expert reasoning on data [1]. More precisely, ExpressIF™ is what is recently called an XAI (eXplainable Artificial Intelligence). This term, introduced by DARPA in 2016 [2], applies to artificial intelligence software which can explain their decisions, learn from data (numerical data, images, etc.) and accessible from intuitive human-machine interfaces. These are precisely our roadmap while developing ExpressIF™, as shown in Figure 4, which presents an overview of the system.

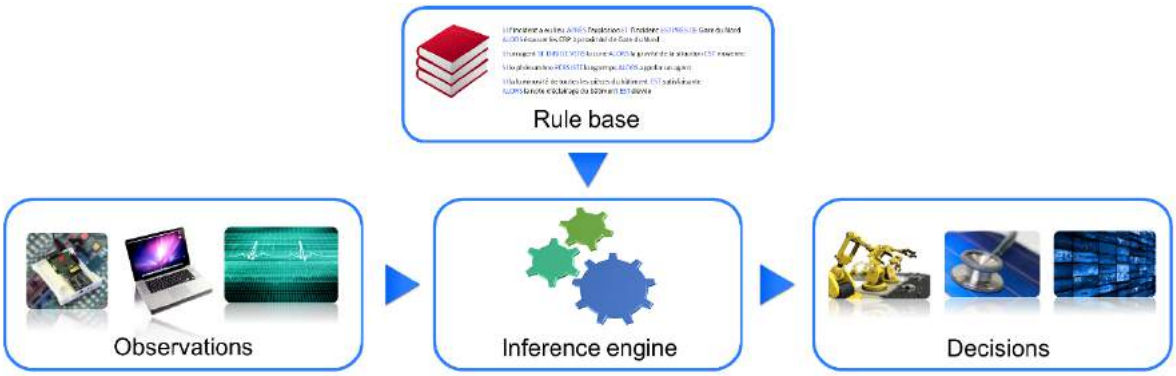


Figure 4 – Overview of ExpressIF™

ExpressIF™ allows to model knowledge using Fuzzy Logic. This logic, introduced in 1965 by L.A. Zadeh [2], is a multi-valued logic which truth values are comprise between [0 ; 1], instead of the classic two values “true” and “false”. Introducing continuous truth-values, fuzzy logic can handle the uncertainty of the knowledge (e.g. a « hot temperature ») and the inaccuracy of the input values. To achieve that, terms close to natural language are formalized by simple-shaped mathematical functions instead of mathematical operators (e.g. comparison operators). Fuzzy logic allows using rules, which are close to the natural language, without mathematical operators, which tend to more comprehensible rule bases, particularly by non-mathematician users. Whereas other fuzzy expert systems only use classical operators such as AND, OR and NOT, ExpressIF™ offers various operators for temporal [4], spatial[6], or even spatio-temporal reasoning [7][8]. These operators can be seen as the vocabulary, which can be used in the rules, representing knowledge in an even more natural syntax.

4.1.2 Web service

For the need of the project, a new version of the web service that allows accessing the different functionalities of ExpressIF™ was developed. It has been implemented as a REST API, with JSON format for interchanging data. The new web service can be host on all platforms (Windows, Linux, etc.) and is based on asp.net core 3.1. It is also extendable, and that is an important feature to cover the different use cases. A web server has been mounted by CEA’s IT teams. The URL will be https://expressif.cea.fr and will be public during summer 2020.

4.1.3 Data provider

Data providers are ExpressIF™ components that allow connecting data sources and adapting their data for the use during the inferences. The concept has been patented before the project and allows specializing the use of the software without modifying the inference engine. In the case of this project,

a specific data provider will be implemented to interface with EDDLL and ECVL and to load the data in a form that is manageable by the inference engine. Concretely, the data adapter receives an image, gets the results of the segmentation and the features that are given by EDDLL and ECVL, converts them in a mathematical format and provide them to the inference engine.

4.1.4 Spatial operators

In DeepHealth, the goal is to reason on medical images to perform an explainable diagnostic. The rules that populate the knowledge base are given by an automatic induction from samples and human expertise. They will particularly use spatial operators that allow expressing relations between regions/objects of an image (e.g., to the left of, inside, touches). Spatial operator can also be visual properties (e.g., shape, textures).

The coherence of the spatial arrangement of entities is very important in medical image understanding. That is why we mainly concentrate on spatial relations when dealing with images. An extensive review of this type of relations is given in [6].

Spatial relations belong to one of the three following categories: topological, metric and structural relations.

We assume that an entity is a region in an image that can be represented as a fuzzy set. It enables to deal with entities whose borders are not well known.

Most of the spatial operators in ExpressIF™ rely on fuzzy morpho-mathematics, especially on two operations called dilation and erosion. Previous work about the acceleration of these operations can be achieved by using SIMD (Single Instruction, Multiple Data) instructions in modern processors that is often considered as an intra-core parallelization [9]. Since .net core 3, SIMD instructions are available in C#, it was decided to port all the ExpressIF™ libraries to .net core 3 leading to some code re-authoring. The dilation and the erosion with SIMD instructions was accordingly re-implemented to get an acceleration of their computation. In average on different images, the gain is about 70% of the computation time. It was one of the quantitative goal for the ExpressIF™ platform.

4.1.5 Relational learning

Relational learning consists in performing classification or annotation based on structural and visual clues. The goal is to extract the relevant properties and relations that define a class or that allows annotating. In DeepHealth, the considered approach aims at proving the viability of the concept [10]. Many improvements may be achieved in changing the relations into rules, in decreasing the combinatory aspect, etc.

4.2 Description of the platform provisions in the context of DeepHealth use cases

4.2.1 UC9: Lumbar spine

During the training phase, the training set must be composed of MRI images that have been segmented, each region annotated and the file labelled with the disease.

During the exploitation phase, extracted regions of segmented MRI have to be annotated in order to recognize potential lumbar spine pathology. Different structures are of interest: vertebrae, intervertebral discs as well as spinal cord and canal. Segmentation of discs are of primary importance as the classification of herniated disc may be based on a deformed shape of the disc [11].

Still, some authors [12][13] try to classify the disease with features that does not depend too much from the shape of the disc. For instance, the features set used for lumbar herniated diagnosis [12] include relative intensity features, shape feature and texture features based on GLCM (grey level co-occurrence matrix) extracted from eight regions of each disc. Gabor features are also extracted in [14] after convolution with Gabor filters of eight orientations.

The review of computerized methods applied to spine analysis in MRI [15] highlights the possibility to use invariant features of the lumbar spine for localization and segmentation of its components in future works. This includes general appearance properties of vertebrae such as symmetries of vertebrae, discs, the spinal canal and cord; general geometric properties such as compactness and connectedness of vertebrae and discs, as well as adjacency relations among vertebrae, discs and the spinal canal. Relevant fuzzy spatial operators may include alignment, symmetry, surrounding, etc. [16].

ExpressIF can reason on image regions to make high-level explainable decision. In the context of lumbar spine diseases, most of them are due to misalignment of the vertebra and/or disks. This kind of information can be processed by ExpressIF and it can be trained on such data to recognize diseases. Most of all, it can provide an explanation of the decision in order to increase the confidence of the decision.

4.2.2 UC12: Skin Cancer

During the training phase, the training set must be composed of dermatologic images that have been segmented, the file labelled with the disease and the file with patient information.

Relevant fuzzy spatial properties and relations will be computed in order to detect clues characterizing skin cancer. They will be chosen according to the literature survey of extracted features [17][18][19][20][21] and computed in a fuzzy way. They will also be chosen according to the different methods used by physicians to diagnose skin cancer, and particularly, melanoma [17][22]: ABCD rule, 7-point checklist, 3-point checklist and Menzies method. These methods accounts for the identification of clues of skin cancer such as asymmetry of the mole, the irregularity of its border, colour irregularity or presence of suspicious colour such as black, grey, blue, dark brown, red and tan, diameter larger than 6 mm, recent change of the mole in size, shape, colour or sensory, or inflammation. In that way, the learned classification rules will provide intrinsic intuitive information about the cause of the decision.

In the feature extraction step, fuzzy logic has been used to characterize colour and asymmetry of the lesion while blotches were extracted from the mole in [23], and fuzzy border consideration is undertaken in [24]. The fuzzy rules will also take into account relevant patient information, such as lesion location, patient declaration about e.g., evolution of the lesion if available. During the exploitation phase, segmented lesions will be analysed and different spatial properties and relations will be computed according to the learned fuzzy rules. ExpressIF is a good way to mix two kind of information: spatial information from the image, but also patient personal information. For the specialists, for skin cancer classification, both types of information are relevant.

4.3 Description of the platform interactions with the DeepHealth toolbox

Figure 5 represents the ExpressIF™ system interactions with the DeepHealth toolbox. We have to distinguish two phases.

The first one consists in inducing a model (i.e. a rule base) from data. The data may differ depending on the use case. To process the data, they are enriched either with segmentation information or features, from EDDLL or ECVL depending on the use case. Relational learning is then performed and a rule base is automatically generated. This rule base can be edited by experts with a dedicated web interface. In the case of lumbar spines (UC9), the same set as the one used by FISABIO will be used to train the model. Eventually, we can train ExpressIF to take into account the defects of automatic segmentation. In that goal, we call directly EDDLL with the model developed by FISABIO to obtain the segmentations. In the skin cancer use case (UC12), we will train ExpressIF on the ground truth, but we will extract features like texture, edges with direct call to ECVL.

The second phase consists in exploiting the system. A specific frontend will be deployed and will allow collecting data: each use case will have its own customized website in order to query the system. These data are then sent to the web service that will launch the inference engine and will then send the results. For UC9, when the image is received, we call directly EDDLL to perform a segmentation

and we send to ExpressIF the image regions and their annotation. The system then applies the rule to make the decision and detect diseases. For UC12, when the image and the extra information on the patient are received, ExpressIF will call directly ECVL to get the region of interest, to crop the image, to get the edges and the texture information. These pieces of information are then sent to ExpressIF for rule application. For both cases, the results are shown as a distribution over the classes and possibly an explanation of the results.

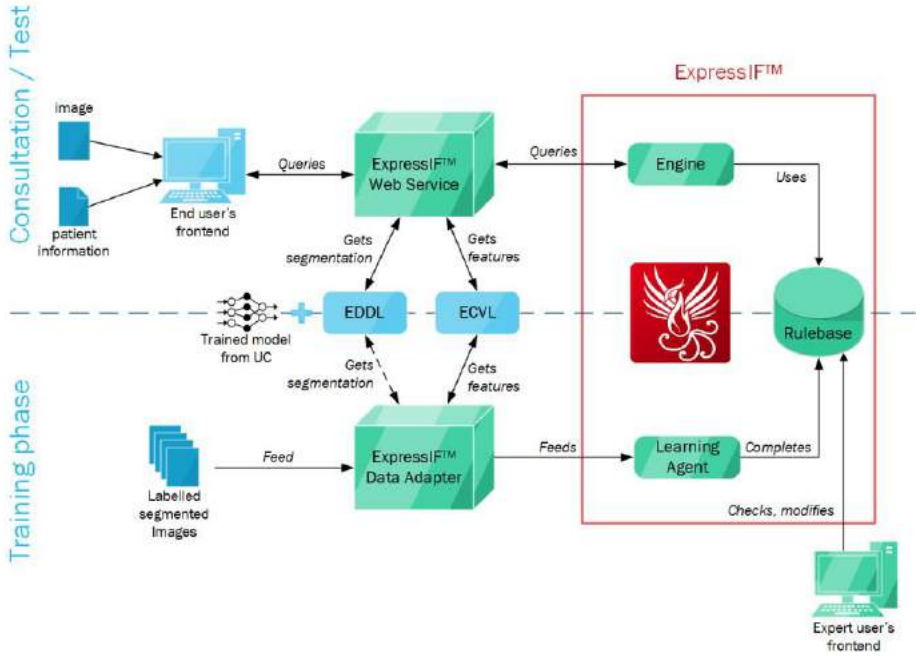


Figure 5 – General interactions overview

4.4 Current status and next steps

Currently, the ExpressIF web service, directional and distance spatial operators as well as a first version of the relational learning algorithm have been implemented. Edges and texture characterization operators have been tested and some improvements will be implemented before M18. A first version of UC12 front-end has also been developed.

Plans for midterm review (M18): During the midterm review in M18, the platform will have its first workflow implemented for UC 12 “Skin Cancer”. ExpressIF™ will be able to take pictures as input and execute a first version of the rule base for this use case. The rule base will be written according to the ABCDE principles from dermatologists and will show the execution of a first version of operators that are involved in such decision process.

5 PF4 - PIAF platform (Thales)

5.1 Description of the platform architecture as adjusted for the purposes DeepHealth

PIAF architecture (see Figure 6) is natively provided with a back-end based on Tensorflow framework. For the time being, the frond-end only supports a single back-end. In order to introduce EDDLL-based image segmentation, in terms of using EDDLL for predicting segmented areas on new images, the PIAF platform needs to be reworked to improve modularity in order to support multiple back-ends.

To support EDDLL and the use case requirements a new back-end is under development for segmentation tasks using EDDLL framework.

The backend takes the form of a Docker container with a REST interface. It accepts requests for processing status and progress, error reporting, running inference on a list of images, and optionally running model training. These interfaces need to be implemented in the new EDDLL backend. For inference, the images need to be cut into patches, batches need to be built and finally the outputs must be fused in order to produce results on large images.

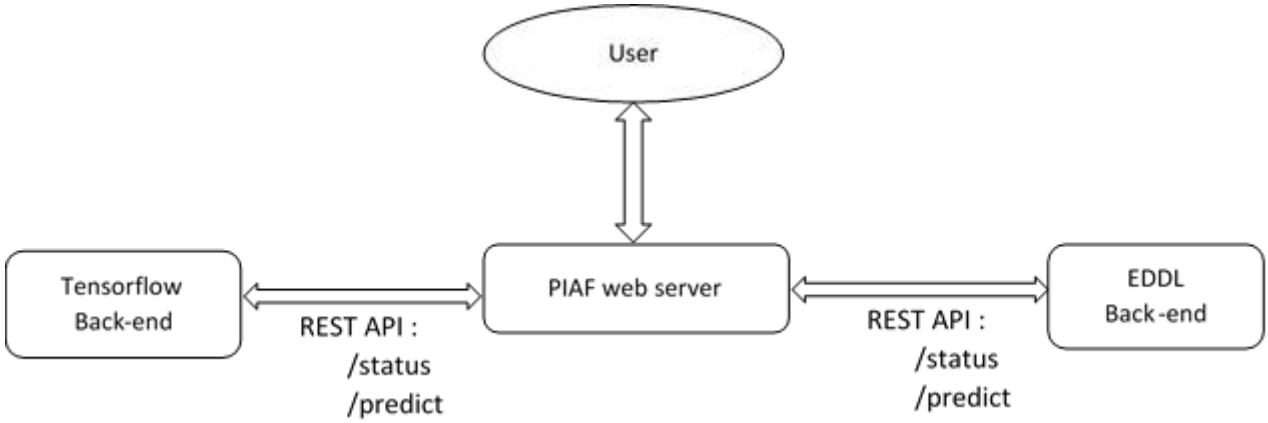


Figure 6. PIAF architecture

5.2 Description of the platform provisions in the context of DeepHealth use cases

The PIAF platform, natively developed for geospatial applications, as presented in Figure 7, is improved in order to support medical image processing. To this purpose, the Thales team is working to include in the platform the ability to read and write files in a format that is appropriate for medical task required by UC7-Major Depression and UC8-Dementia (ex: Nifty).

To fit with the use cases requirements, the team is also working on annotation and visualization tools able to support segmentation tasks that are very common in medical image processing.

The platform provides as a front-end a graphical user interface available via web that includes the executable programs for testing predictive models on new biomedical images loaded by the end-user.

The platform will also offer the possibility to select the more appropriate topology according to the medical task to execute.

The web-based interface implies several benefits for the expert-users of our UCs, in particular:

- They do not have to manage individual software installs and updates whenever there is a change to the PIAF software.
- They would no longer need to verify that the software will work on various combinations of hardware and operating systems. Recently OVGU, in charge of UC7 and UC8, obtained their

own HPC capacities, and the PIAF web browser interface allows them to run the inference on any combination of hardware and operating system of their cluster.

- Expert-user, no longer have to worry about compatibility between a graphical user interface (GUI) and the server, which means that they can update to new versions of the software without wondering whether it will be compatible with the server.

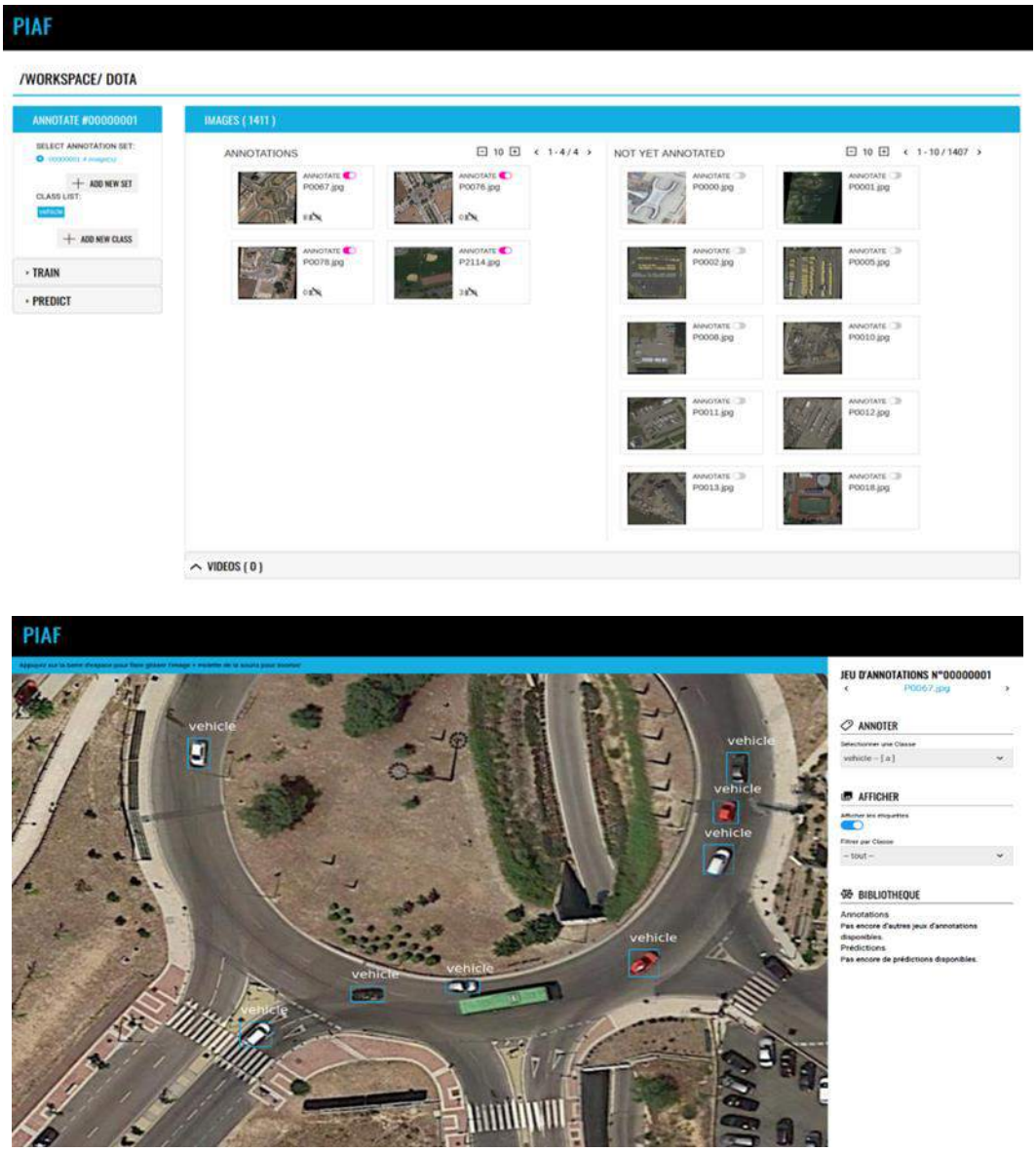


Figure 7. PIAF interface for geospatial applications

5.2.1 UC7: Major Depression

For UC7 we are exploring different possibilities in order to elaborate a predictive model such us:

- Exploit the anatomical scans highlighted in T1-weighted images.
- Functional scans (T2*-weighted / BOLD-fMRI scans) to track the oxygenation levels in each voxel of the brain
- Beta-maps: used to fit a model for each voxel in the brain when a patient is involved or not in an emotional task.

All these strategies highlighted potential short comings that we are still being evaluated in order to select the best way to fulfil the end-user requirements.

5.2.2 UC8: Dementia

Considering UC8, the PIAF platform takes as input Magnet Resonance Imaging (MRI) T1 Nifty images of the brain and allows to visualize the hippocampal region segmentation, estimating the area covered. The health control is performed repeating the inference on this brain area loading MRI brain images took at different patient ages. In fact, the size of the hippocampal region area decreases in case of cognitive impairments.

In the future, other developments will be forecasted to keep track of the changes of the hippocampal region features (e.g., the evolution of size of the area).

5.3 Description of the platform interactions with the DeepHealth toolbox

5.3.1 DH Front-end

Not applicable.

5.3.2 EDDLL

PIAF currently uses Tensorflow to run inference on new images. In order to interface PIAF with EDDLL, the team built a new back-end for the PIAF platform. This backend takes the form of a Docker container that runs a REST service which provides functionality to run prediction, through EDDLL, on images that are selected in the front-end Human–Machine Interface (HMI). Thus, the EDDLL framework will be packaged in the Docker container and images will be fed to it through the REST API. The EDDLL framework will import ONNX models and will run inference on images. Thus, THALES needs to convert the new or pre-existent models to ONNX. The results will be returned in the REST query as binary masks.

The team has created Docker container with the last EDDLL library version and has already tested several EDDLL functions in order to

- Implement a U-Net network and convert it to ONNX and EDDLL format (Figure 8),
- Test the obtained ONNX model on other frameworks, to verify the correctness of each layer of the topology
- Test the EDDLL Load function on the U-Net ONNX model

In parallel, we have developed a REST server has been developed in C++ language that will be used for creating the EDDLL back-end.

5.3.3 ECVL

Not applicable.

5.4 Current status and next steps

Considering the PIAF infrastructure to host EDDLL and biomedical use cases, we have already integrated the EDDL components into a new back-end of the PIAF architecture. A preliminary activity that includes the development of the utilities to import DICOM and NIFTI formats on PIAF is still on-going. The intent here is to conclude this task before the midterm review.

Other on-going activities concern the model development for use case 7 and 8. In particular, a test model for use case 8 is already implemented in Keras and convert to EDDLL as presented in Figure 8. Currently, we are finalizing our test on EDDLL “loading” functionality and the comparison with the Keras. In the next months we will improve the UC8 toy model in order to obtain better results. We will also get involved in the implementation of an ad hoc model for UC7 since we were involved in the activities related to the PIAF architecture refinements to host EDDL and UC8 predictive model developments.

Plans for the Midterm review (M18): During the midterm review in M18, the PIAF is forecasted to demonstrate its integration with the EDDLL using a toy model similar to the one that will be used for

UC8. In particular, the platform will demonstrate its functionality to load a pre-existent model, load new medical images and perform the inference phase.

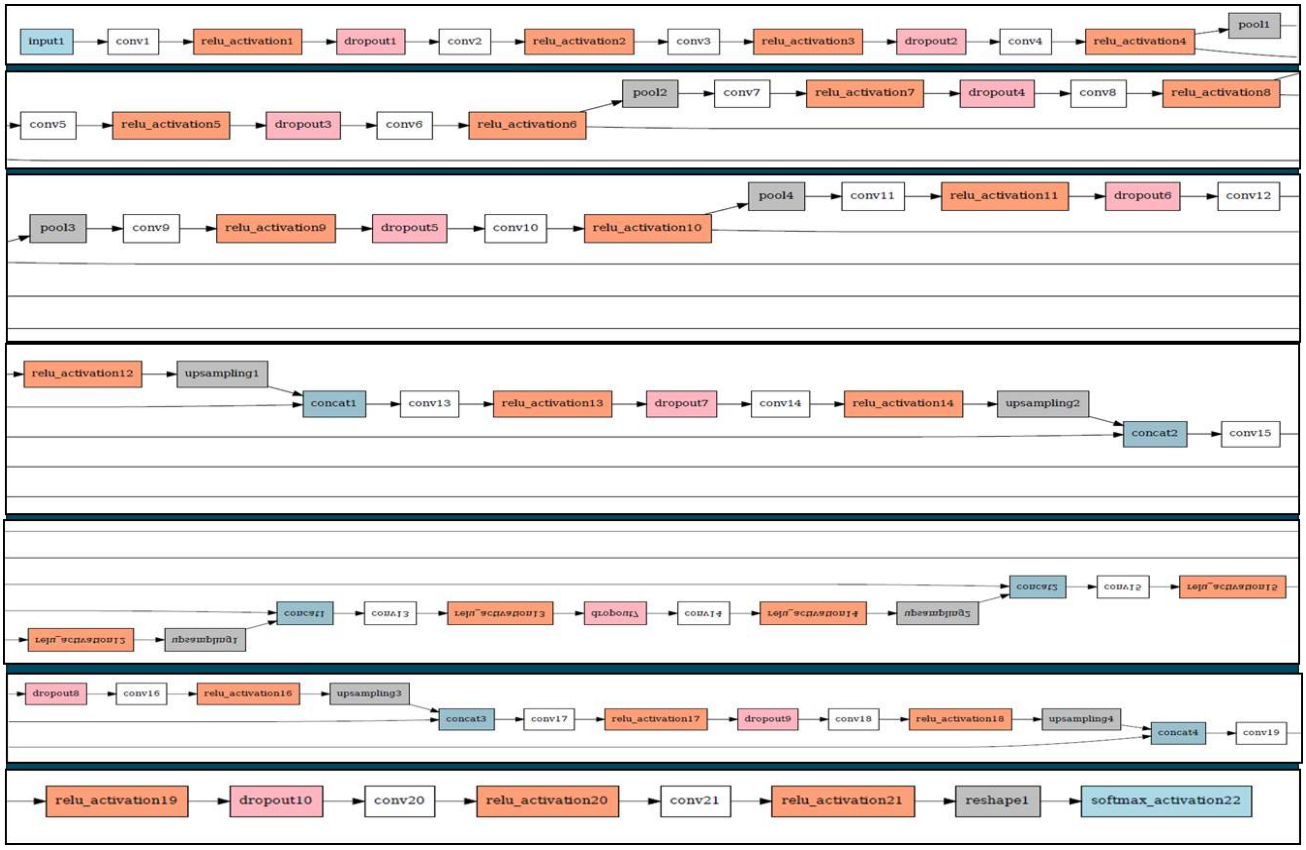


Figure 8. UNet model for brain segmentation using EDDLL functions

6 PF5 – Open DeepHealth platform (UNITO)

6.1 Description of platform as adjusted for the purposes of DeepHealth

OpenDeepHealth (ODH) platform has been developed as part of the University of Turin HPC4AI infrastructure, that is a federated Openstack cloud with multi-tenant private Kubernetes instances.

The Deep Learning cluster of HPC4AI is composed by:

- 10 Compute nodes (currently being extended to 20): Intel 40 cores HT, 4xT4 GPU per node, 10TB flash per node
- 4 all-flash storage nodes: Intel 40 cores HT, 48TB flash per node (~200TB)
- 2x25 Gb/s bonded ethernet

The overall software architecture is based on the open-source OpenStack cloud technology with **multi-tenant** configuration. Multi-tenancy allows combining resource sharing through virtualization with resources and data isolation for each tenant. In HPC4AI, each OpenStack project can be configured as an **HPC Secure Tenant** (HST), that is a VPN with controlled access comprising an elastic configuration of private hosts, storage and network. Access to the HST through a VPN gateway guarantees the access to the overall environment in the tenant, either a Kubernetes cluster or a scheduling manager system. Mechanisms to provide seamless secure access to the overall system inside the HST are being currently analysed and tested.

The OpenDeepHealth platform is defined as an HST where a multi-Tenancy **Kubernetes** Container Cluster is deployed, taking advantages of container and orchestration technology. This cluster is defined for specifically integrating ECVL and EDDLL in a single, easy to use platform supporting expert users for model training and testing.

Each use-case is provided with its own Kubernetes **namespace**, only accessible through an authentication mechanism provided by a DeepHealth access point, in turn connected with the VPN gateway. The namespace has its own computing resources and persistent volumes for data storage. **Docker** containers integrating both **ECVL** and **EDDLL** libraries, in both C++ and Python releases, are also provided so that use-case applications can easily access to DeepHealth libraries.

As a first step, configuration files (e.g. helm files) for application deployment in Kubernetes environment are provided. Further mechanisms, mainly based on configuration templates, are currently studied to provide a complete Deployment-as-a-Service technology in ODH. Integration with DeepHealth front-end and back-end is also being evaluated to verify if it can provide users with a richer experience in accessing ODH platform.

An orchestration layer on top of Kubernetes cluster is also available provided by the StreamFlow workflow manager[2]. StreamFlow allows to easily manage workflow modelling and execution in different environments enhancing Kubernetes ability to handle with different computational steps. Also, if different components in a use-case application can be detected and modelled as workflows (e.g. pipeline of different computational steps) StreamFlow makes possible to describe an execution plan spawning across multiple sites and, specifically in this case, this will allow accessing to Open Computing Cluster for Advanced data Manipulation (OCCAM) SuperComputer, a traditional HPC cluster facility that is also part of the HPC4AI infrastructure.

It is worth noticing that ODH platform is an application independent machine learning (ML) platform, which means that it provides Artificial Intelligence (AI) experts with resources, both hardware and software, to train and test their own ML application without further adaptation or configuration.

[2] StreamFlow: cross-breeding cloud with HPC Published in ArXiv 2020 https://arxiv.org/abs/2002.01558

The actual output of the platform is an easy-to-use efficient environment for training and testing the models developed by the ML experts based on medical needs. So, from this perspective, platform provision in the context of the use case are:

- Data storage and management, in terms of private volumes and data movement support tools.
- Computing facilities, in terms of both hardware, that is CPU and GPU nodes, and deployment tools
- AI environment, in terms of ML libraries enabling efficient model development.

Specifically concerning platform interaction with the DeepHealth toolbox, the platform provides Docker containing EDDLL/ECVL libraries in different flavour, that is running on CPU and GPU and in both C++ and python bindings. Integration with DH front-end is still under evaluation.

Specific details, such as the kind of input/output data, and the exact EDDLL/ECVL library bundle and functionality used for implementing each use case are provided in the following sections.

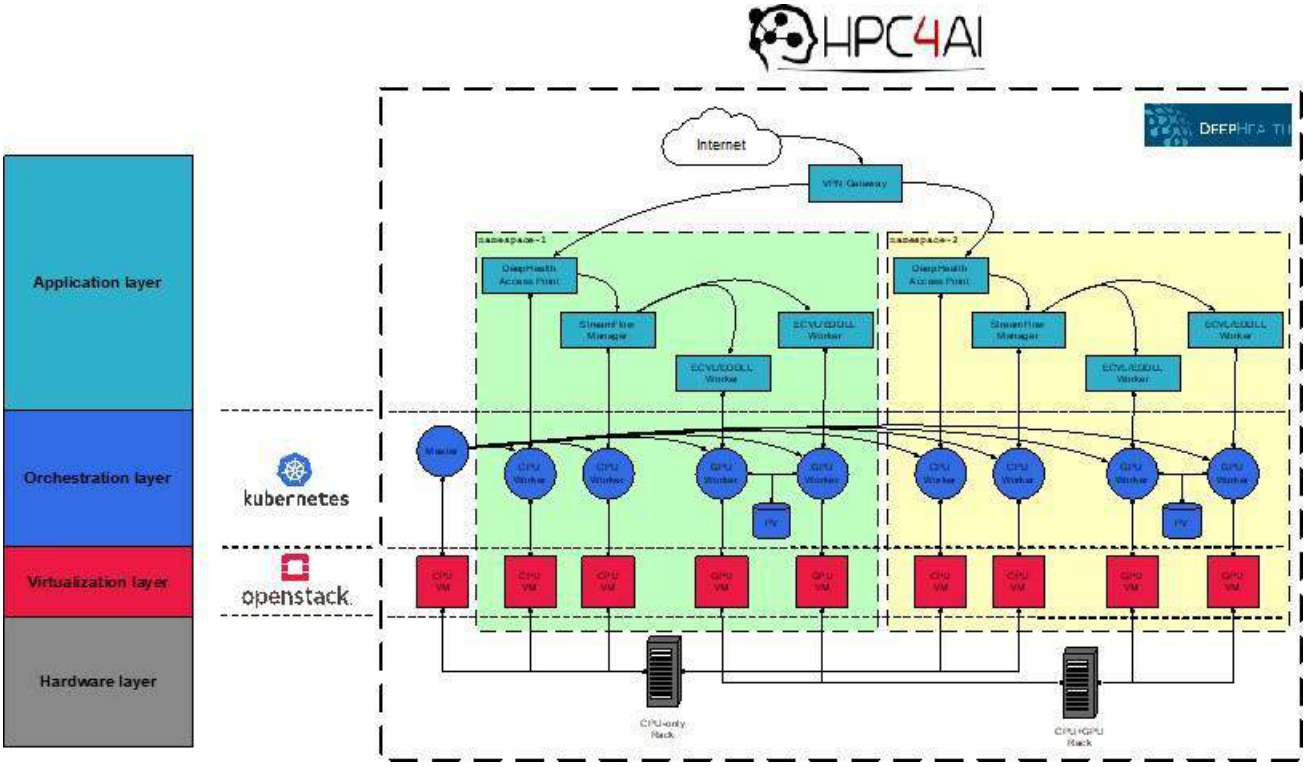


Figure 9. ODH platform

6.2 UC2: UnitoPath

6.2.1 Description of the platform provisions in the context of the UC

UC2 targets classification task for gastrointestinal pathology, specifically classification colon biopsies according to the requirements detailed in D1.1[3]. In Figure 10 the overall integration plan is reported schematically. Whole hematoxylin and eosin (H&E) slides, scanned on a high-throughput Hamamatsu NanoZoomer S210 scanner are being selected by medical personnel at UNITO.

The images are completely anonymized and accessible through UNITO internal OMERO repository, that allow sharing only within UNITO network with proper credentials. The dataset is constituted by high resolution microscopy images: RGB, optical microscopic image in '.ndpi' file format,

[3] D1.1 Use cases and applications specifications and requirements

multiresolution pyramid, up to 80000x80000 pixel resolution at the highest level. Each image is accompanied by the ancillary classification information, namely:

- classification label
- Region of Interest (ROI) (part of the image used by the pathologist to classify) in NDPA file format.

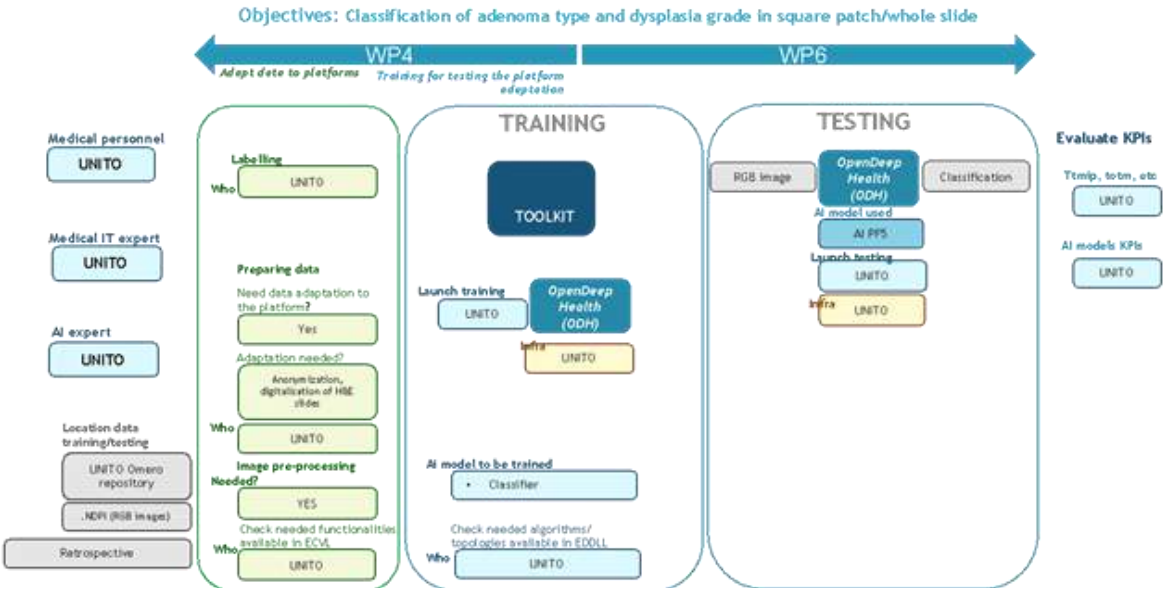


Figure 10. UC2 integration plan on ODH

The data need to be pre-processed to get ready for training. In particular, whole H&E slides undergo a number of processing steps including:

- Color conversion
- Contrast enhancement
- Image normalization
- Segmentation, e.g. Otsu method
- Connected components
- Rescale images
- Extraction of fixed resolution image patches

This image processing steps are performed by UNITO ML expert. The output of this process is a dataset of image patches with classification labels, ready for the training/testing of Deep Learning models. An example of image patches extracted from whole slide image is provide in Figure 11.

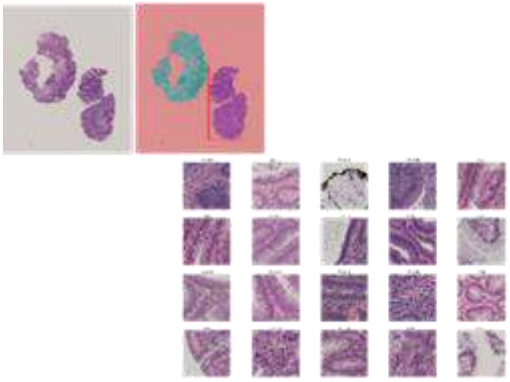


Figure 11. Creation of image patches from whole slides.

6.2.2 Description of the platform interactions with the DeepHealth toolbox

6.2.2.1 DH Front-end

Not applicable.

6.2.2.2 EDDLL

The classification task will be solved with a ResNet like architecture implemented by EDDLL functionalities. Classical Adam and Stochastic Gradient Descent learning algorithms will be used to train the models. Since the dataset is significantly unbalanced proper definition of the loss function, e.g. focal loss, must be used. The EDDLL will be exploited by means of python biding provided on ODH platform.

6.2.2.3 ECVL

The UC2 will leverage on integration with ECVL and OpenSlide library to process the huge resolution whole-slide images.

6.3 UC4: Brain

6.3.1 Description of the platform provisions in the context of the UC

UC3 aims at generating synthetic Computer Tomography Perfusion (CTP) maps of the brain from a reduced data set, according to the requirements detailed in D1.1. The dataset consists of CT scans and target perfusion maps (ground truth). In Figure 12 the overall integration plan is reported schematically.

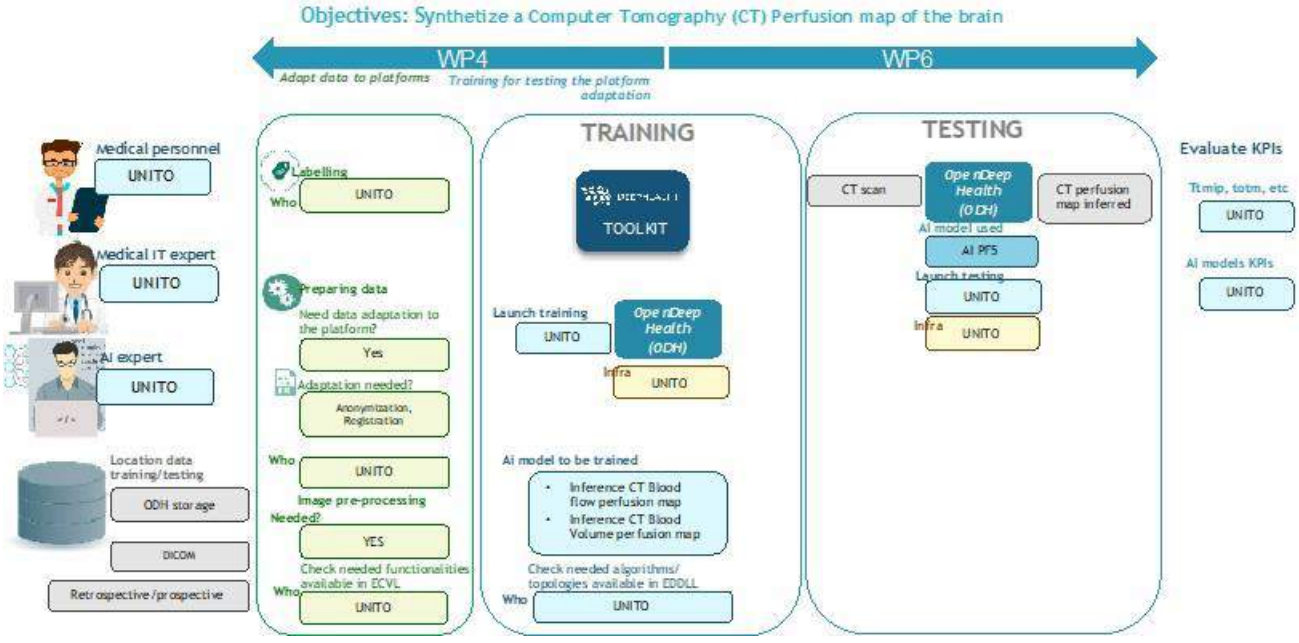


Figure 12. UC3 integration plan on ODH.

CTP measures, with a sample time of roughly 1 Hz, the passage of a contrast medium bolus into the brain, on a pixel-by-pixel basis. These are collected and selected by medical personnel at UNITO. CT scans are completely anonymized and stored on the Open DeepHealth storage system, which allows sharing only within UNITO network with proper credentials. The dataset consists of temporal sequences of monochrome2 512x512 pixel brain CT scans stored in 16 bits per pixel. Each of these temporal sequences is accompanied by four ground truths, which are the target perfusion maps:

- CBV: map of the total blood volume
- CBF: map of the blood flow

- MTT: map of the blood mean transit time
- TTP: map of the blood's time to reach the maximum flow

The input CT scans need to be pre-processed, requiring:

- Image normalization
- Segmentation using e.g. Otsu method
- Calibration and alignment through the time sequence

The image processing steps are performed by UNITO ML experts. The output of this process is a dataset of input CT time sequences with perfusion maps associated, ready for training/testing purposes. An example of CT time sequences and target perfusion maps is shown in Figure 13.

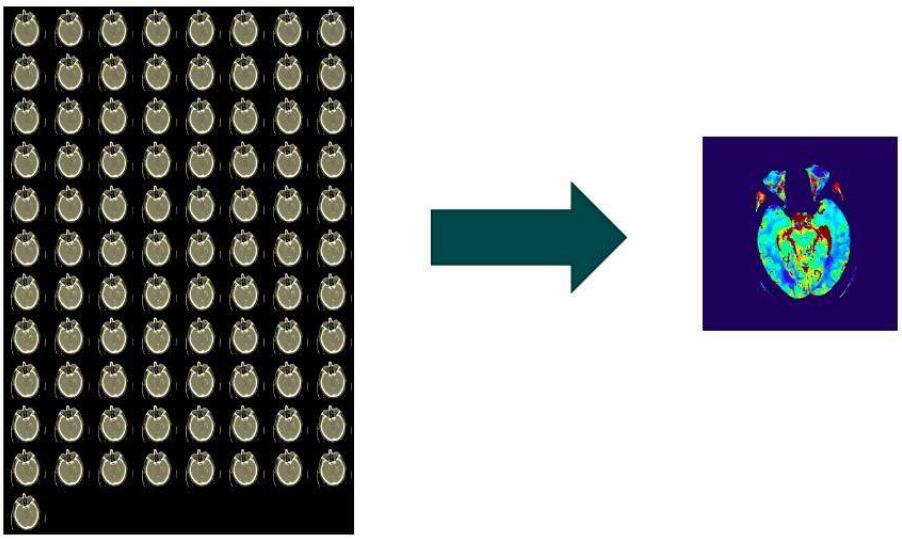


Figure 13. Example of perfusion generation (right) from temporal CT scans (left).

6.3.2 Description of the platform interactions with the DeepHealth toolbox

6.3.2.1 DH Front-end

Not applicable.

6.3.2.2 EDDLL

The perfusion map generation will be exploited using a U-Net like model, implemented using python binding of EDDLL. Classical SGD learning algorithm and learning rate decay policy will be used at training time. Since the target is an image, MSE based loss will be used.

6.3.2.3 ECVL

Input CT scans will be in DICOM format; hence, ECVL tool will be used to import and convert CT scans into a usable format for UC3.

6.4 UC4: Chest

6.4.1 Description of the platform provisions in the context of the UC

The main purpose of UC4 will be to recognize lung nodules using chest CT scans, providing radiologists an efficient tool for daily activity, according to the requirements detailed in D1.1. In Figure 14 the overall integration plan is schematically reported.

Lung nodules are quite common incidental findings in CT (computed tomography) scans and can be defined as small focal lesions (ranging from 5 to 30 mm) that can be solitary or multiple. CT scans are completely anonymized and stored on the Open DeepHealth storage system, which allows

sharing only within UNITO network with proper credentials. The dataset consists of monochrome2 512x512 pixel brain CT scans. These CT scans are selected and associated to a ground-truth segmentation of lung nodules by medical personnel at Citta' della Salute University Hospital (CDSS).

The input CT scans need to be pre-processed, requiring:

- Image normalization
- Segmentation using e.g. Otsu method
- Histogram equalization methods
- Conversion to Hounsfield Units
- Thresholding and morphological operators

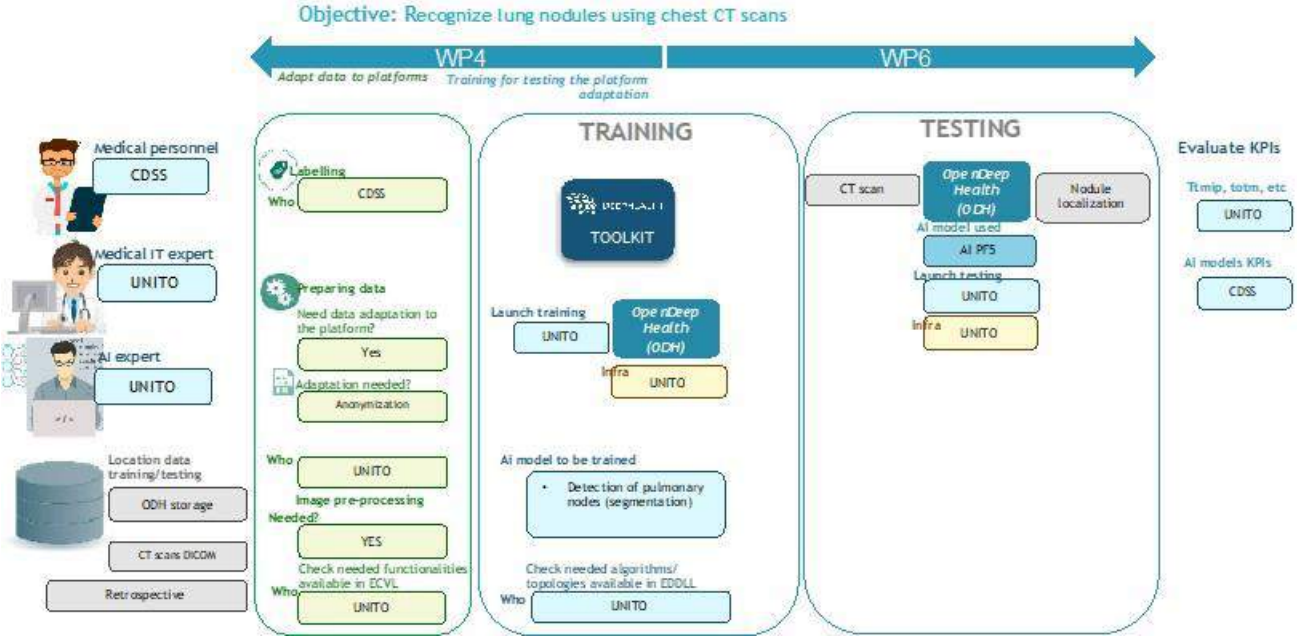


Figure 14. UC4 integration plan on ODH.

The image processing steps are performed by UNITO ML experts. The output of this process is a dataset of input CT scans with lung nodules segmentation associated, ready for training/testing purposes. An example of CT time sequences and target perfusion maps is shown in Figure 15.

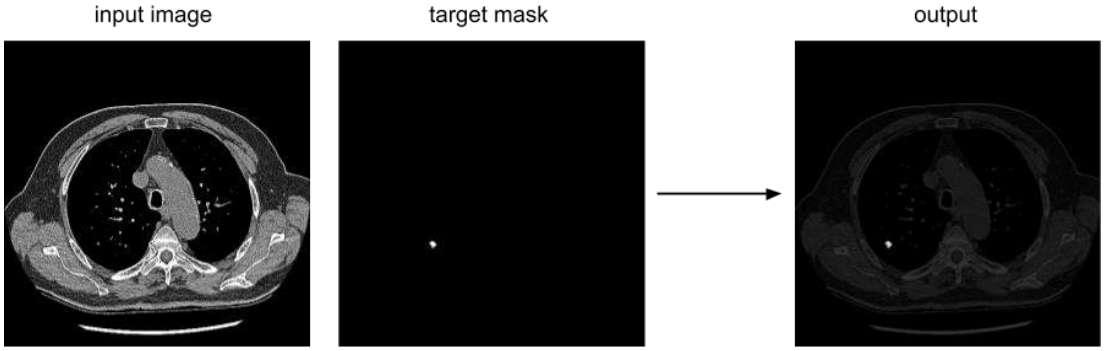


Figure 15. Example of input image and segmentation mask.

6.4.2 Description of the platform interactions with the DeepHealth toolbox

6.4.2.1 DH Front-end

Not applicable.

6.4.2.2 EDDLL

The lung nodules segmentation problem will be addressed using a U-Net like architecture implemented using the Python binding of EDDLL. Classical optimizers like Adam or SGD learning optimizers will be used, jointly to learning rate decay policies and regularization techniques like weight decay. Since the target is a segmentation mask, combination of loss functions like cross-entropy and segmentation overlap metrics like dice score and intersection over union will be used.

6.4.2.3 ECVL

Input CT scans will be in DICOM format; hence, ECVL tool will be used to import and convert CT scans into a usable format for UC4.

6.5 UC5: Deep Image Annotation

6.5.1 Description of the platform provisions in the context of the UC

UC5 “Deep Image Annotation” aims at implementing a combination of supervised neural models able to generate natural (and meaningful) text given one or more medical images as input. This is clearly a multimodal task, that needs annotated and related image and text datasets, where each image has a unique text associated with it.

Since no data are going to be shared by partners in the Consortium due to privacy concerns, UC5 is going to use a public dataset, namely the Indiana University Chest X-Rays, freely available and described at the URL https://openi.nlm.nih.gov/faq. Basically, it is a collection of chest radiographies, distributed in DICOM and PNG formats, with associated de-identified semi-structured reports, distributed as XML text files. Such XML files contain both metadata about the examination and the report written by a radiologist. UC5 uses only MeSH terms associated to the examination (as labels/categories for the deep convolutional neural network) and the three texts in natural language that compose the standard American medical report: “indication”, “findings”, and “impression”. Each section is used to train a single recurrent neural network: this means that at the end of the training stage there are three different models, one per section of the report.

During testing and in production mode, the final model can be seen as a pipeline of two stages. First, the images are labelled using the original MeSH terms. Then, its internal representation and the labels are fed to the recurrent model, that starts generating text.

The dataset will be pre-processed and organized outside of the OpenDeepHealth architecture, that will need to host only the result of such pre-processing in its data repository. Since the recurrent neural models use word embeddings, OpenDeepHealth will host also one or more embeddings, such as the Bio-NLP and the BioWordVec ones for the English language.

Figure 16 summarizes how the ODH platform will support UC5 in terms of partners’ collaboration and roles as well as inputs, outputs and models to be considered in this case.

6.5.2 Description of the platform interactions with the DeepHealth toolbox

6.5.2.1 DH Front-end

Not applicable.

6.5.2.2 EDDLL

UC5 is based on recurrent neural networks that will be implemented in EDDLL, specifically GRU and LSTM units. Once those neural networks are available in EDDLL, UC5 will interact with EDDLL using

the Python binding. EDDLL is expected to fully support the huge matrixes containing the word embeddings.

6.5.2.3 ECVL

UC5 will experiment the deep convolutional neural networks implemented in ECVL in order to choose the most suited one for the specific dataset. UC5 will interact with ECVL using the Python binding. It is worth pointing out that UC5 is being developed using 8-bit PNG input images. Only in a later stage, UC5 might be able to load and process different file formats, in particular DICOM images with a higher bit depth. In all cases, UC5 uses single-channel (monochromatic) images.

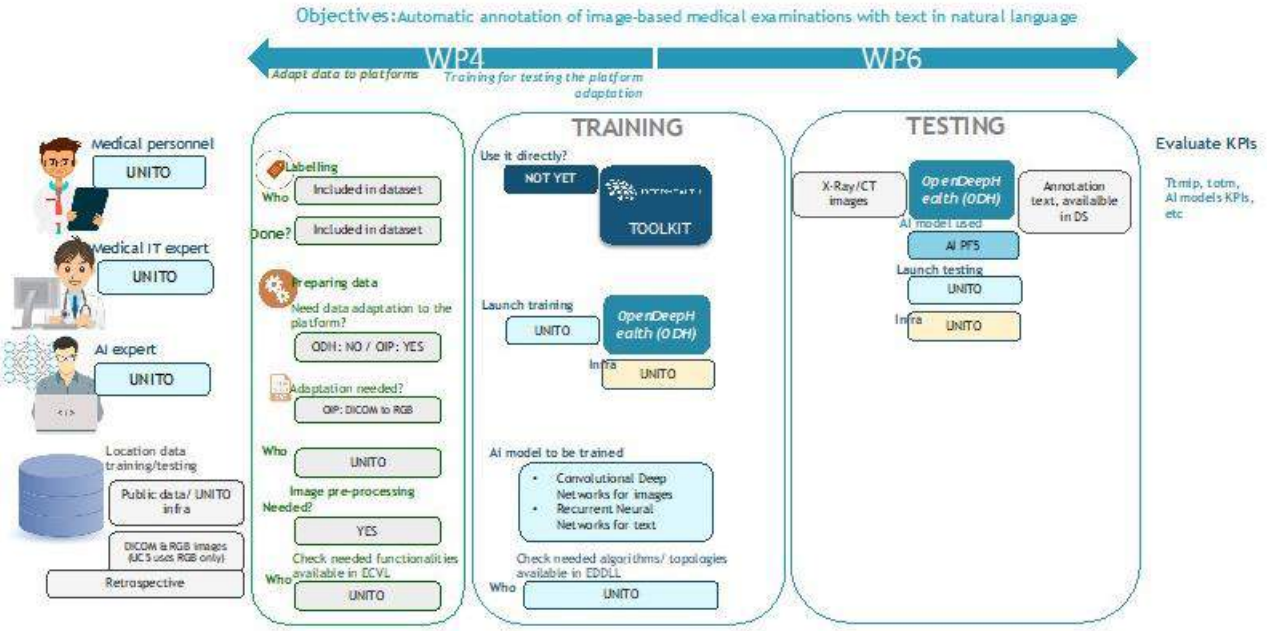


Figure 16. UC5 integration plan on ODH

6.6 UC9: Lumbar spine

6.6.1 Description of the platform provisions in the context of the UC

The aim addressed in this use case is to obtain the structures of lumbar spine segmentation to study clinical-radiological correlations in lumbar physiopathology's.

To achieve this, the dataset to be used consists of 23688 studies belonging to the 2015-2016 period of years. All this data has been extracted from BIMCV and it is distributed between 17 sanity departments. In total, this dataset contains more than 124,000 RM images and has been structured with MIDS Standard.

In order to use this dataset in the ODH platform, the input MR scans need to be pre-processed requiring:

- Linear coregister;
- Filtering;
- Magnetic field inhomogeneity correction; and
- Normalization.

The envisioned outcome of the platform to support doctors' decisions is the segmentation of lumbar structures.

Figure 17 and Figure 18 depict the considered workflow of the use case and the overall integration plan of the use case with the ODH platform accordingly.

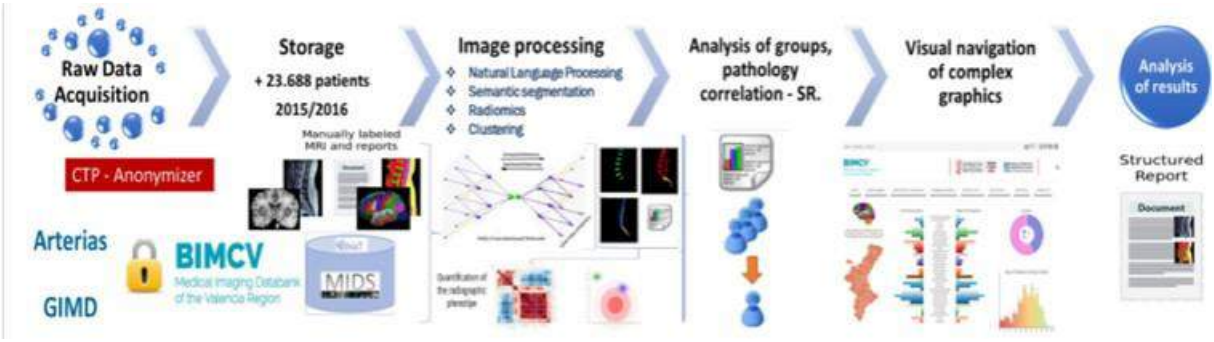


Figure 17. UC9 workflow

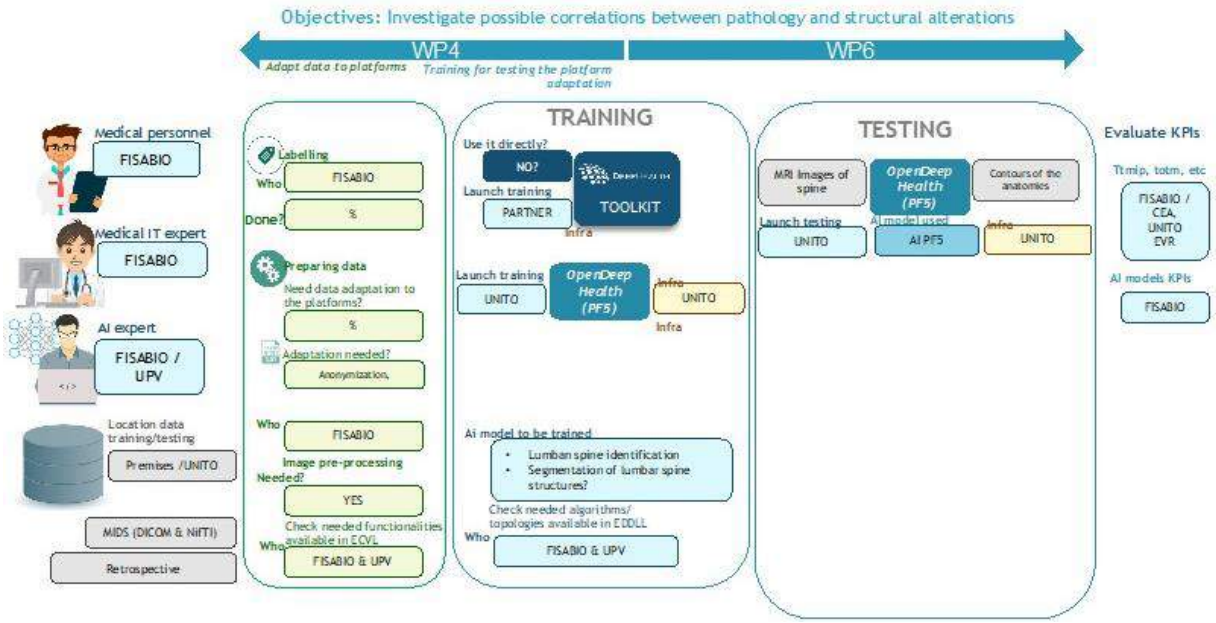


Figure 18. UC9 integration plan on ODH

6.6.2 Description of the platform interactions with the DeepHealth toolbox

6.6.2.1 DH Front-end

Not applicable.

6.6.2.2 EDDLL

Training and testing/inference phases will be implemented using EDDLL library exposed in Docker containers integrated in the ODH platform. It is not defined yet which functionality will be used for this use case.

6.6.2.3 ECVL

Training and testing/inference phases will be implemented using ECVL library exposed in Docker containers integrated in the ODH platform. It is not defined yet which functionality will be used for this use case.

6.7 UC10: Alzheimer's Disease

6.7.1 Description of the platform provisions in the context of the UC

The Alzheimer Disease consist in neurodegenerative condition that has no cure, it is hard to diagnose in the early stages of the disease. There is a prevalence of 5-8% in people over 60 years and nowadays, more than 50 million people suffer this illness. It is important to detect and diagnose it

while still in the early stages of Alzheimer's disease to minimize its long-term impact through palliative and/or cognitive conductual treatments.

To this end, this use case aims to obtain a brain magnetic resonance imaging classifier in Alzheimer Disease using deep learning techniques. The dataset to be used consists of 2461 cases extracted from BIMCV (Medical Imaging Databank of the Valencia Region)[4]. This data will be combined with Alzheimer Disease Neuroimaging Initiative (ADNI)[5] and Open Access Series of Imaging Studies (OASIS)[6], obeying the Medical Imaging Data Structure standard (MIDS) as an input format. All the MRI are completely anonymized and will be stored on the OpenDeepHealth storage system, that is a private persistent volume defined in the platform tenant configured for this use case.

In order to allow the dataset use from the ODH platform, the input MR scans need to be pre-processed requiring (see Figure 19):

- Parenchyma extraction (FLS-BET);
- Linear transformation from native to reference image (FLS-FLIRT); and
- Freesurfer processing to build the Ground Truth.

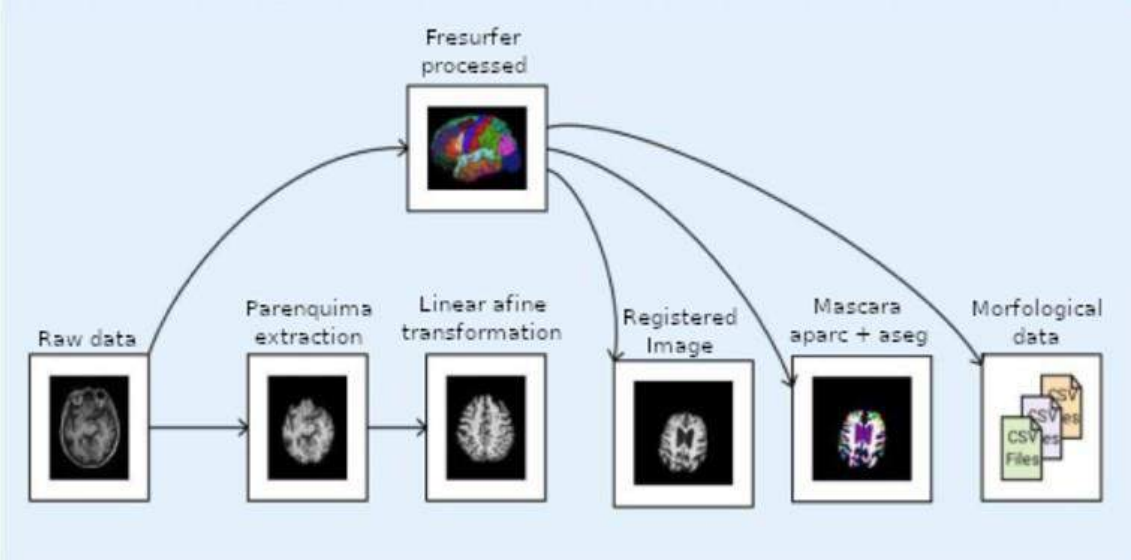


Figure 19. UC10 pre-processing sequence

The envisioned outcome when exploiting the considered data with the ODH platform is the possibility to classify images with respect to belonging or not to Alzheimer Disease. This UC is a representative problem of classification as can be seen in Figure 20.

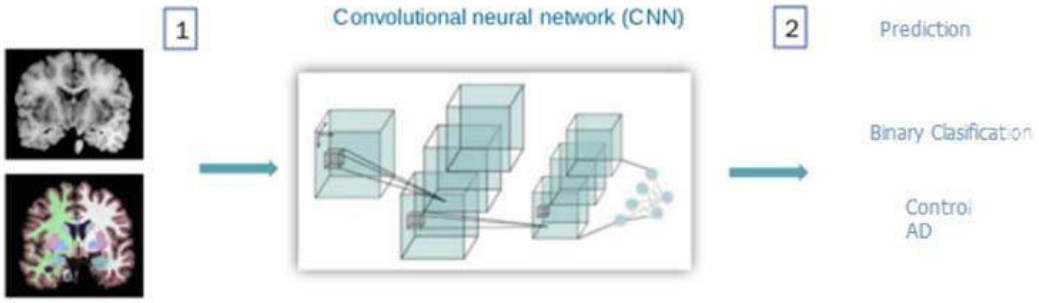


Figure 20. UC10 workflow

[4] http:/bimcv.cipf.es
[5] http://adni.loni.usc.edu
[6] https://www.oasis-brains.org/

Figure 21 depicts the overall integration plan of the use case with the ODH platform.

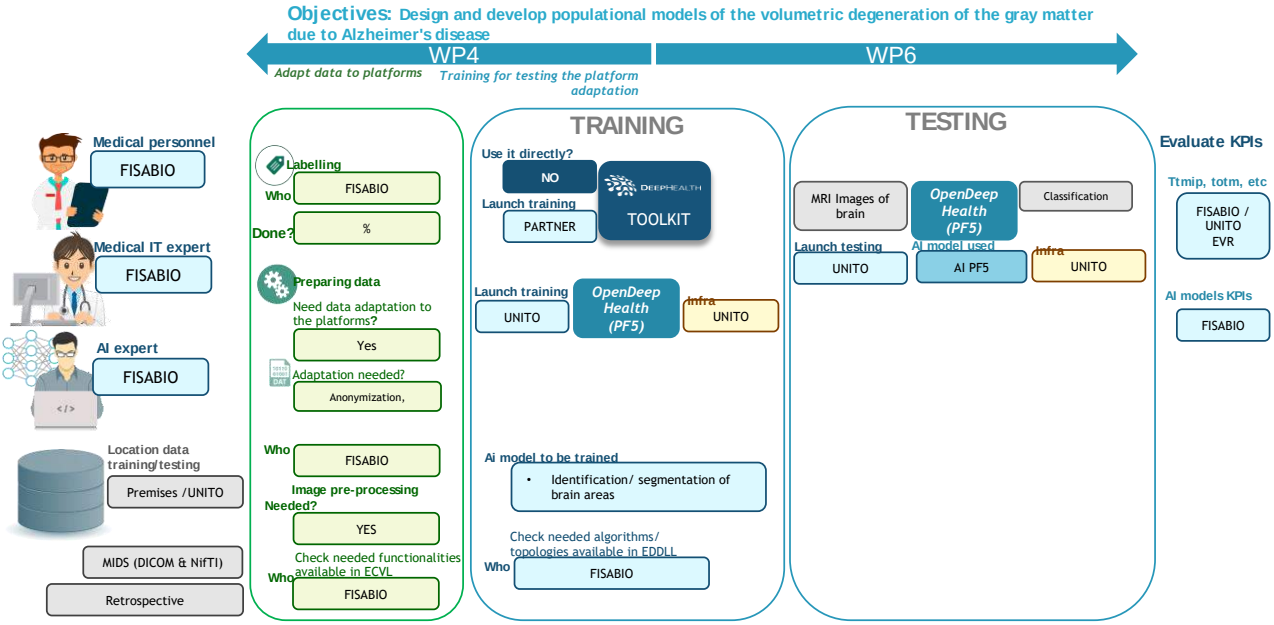


Figure 21. UC10 Integration Plan on ODH

6.7.2 Description of the platform interactions with the DeepHealth toolbox

6.7.2.1 DH Front-end

Not applicable.

6.7.2.2 EDDLL

Training and testing/inference phases will be implemented using EDDLL library exposed in Docker containers integrated in the ODH platform. It is not defined yet which functionality will be used for this use case.

6.7.2.3 ECVL

Training and testing/inference phases will be implemented using ECVL library exposed in Docker containers integrated in the ODH platform. It is not defined yet which functionality will be used for this use case.

6.8 Current status and next steps

Following the activities in the first phase of the project, the current status of the ODH platform is the following:

- An OpenDeepHealth Kubernetes Container Cluster has been deployed and configured. It currently contains a unique namespace but additional namespace will be defined as private environment for each use case. This namespace is currently used as a testing environment to provide demonstration of integration with DeepHealth toolbox mainly using GPU nodes.
- Latest version, so far, of Docker images for running software based on EDDLL and ECVL libraries and their Python wrappers (PyEDDLL and PYECVL) have been integrated in the OpenDeepHealth cluster.
- Integration of DeepHealth front-end and back-end has just started and it is still under investigation.

Main steps foreseen in the next phase are mainly focused on completing integration with DeepHealth toolbox and enhancing the overall ML environment in the ODH platform. In more details next actions are:

- Updating Docker images integrated in ODH platform as new versions of the EDDLL/ECVL libraries are released.
- Completing integration of DeepHealth front-end and back-end.
- Integrating and testing HPC support for DeepHealth libraries when released.
- Providing private ODH integrated environment for each use case in different Kubernetes namespaces, fulfilling specific software and hardware requirements if any.
- Developing and enhancing automatic deployment tools, based on StreamFlow orchestration tool.

Plans for the Midterm review (M18): During the midterm review in M18, ODH functionality will be demonstrated, running an ML example application composed of two steps: training and inference. In the first step, data will be loaded and images will be prepared applying ECVL functionalities (e.g. classification). The training step will apply EDDLL functions to produce a trained model. The resulting model will be provided as input to the second step that will perform the inference phase using both ECVL and EDDLL functionalities. The specific dataset and network used in the demonstrator will be decided choosing from the tests that are currently available. It will demonstrate how DeepHealth toolbox is integrated in ODH and how it can be used by ML applications.

7 PF6 - Digital Pathology Platform (CRS4)

7.1 Description of the platform architecture as adjusted for the purposes of the UC

The software architecture of CRS4's Digital Pathology platform as used in the context of DeepHealth is illustrated in Figure 22. The platform's main components are the image repository and the annotation database. The image repository is based on the OpenMicroscopy OMERO server [26], extended by the purpose-built ome_seadragon software component. As mentioned in the previous section, digital pathology images have particularly high resolution and pyramidal structure. The OMERO server and ome_seadragon are designed to natively handle this type of image.

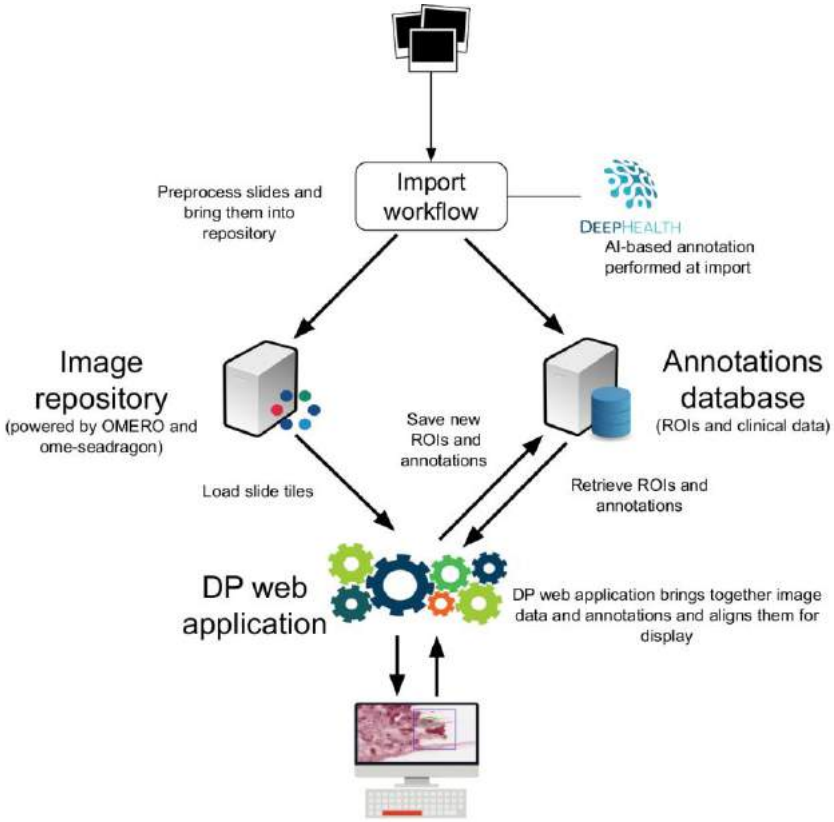


Figure 22. Software architecture of the CRS4 Digital Pathology platform.

The annotation database is a PostgreSQL relational Data Management Management System (DBMS). The web application integrates data from these two services into a single graphical user interface. The DP platform supports all the major virtual slide formats thanks to the integration of the OpenSlide library [27]; this same library has been integrated with ECVL and thus provides support for the same data formats to the DeepHealth products. The resulting virtual microscope allows the on-line remote visualization and exploration of these very large virtual slides with 2D annotations.

7.1.1 Cloud adaptation

To better adapt to the Use Case scenario, the DP platform has been adapted for deployment in a cloud-native environment. All the components of the system have been containerized using Docker container images and the platform deployment is currently managed via docker-compose. Porting to a Kubernetes deployment is planned to support scalable distributed, cloud-native deployment scenarios, to achieve native compatibility with common managed Kubernetes platforms, and to allow simpler integration of the distributed inference techniques being developed within the DeepHealth cloud adaptation tasks (T2.4 and T3.3).

7.1.2 Inference functionality

DeepHealth-based deep learning models will be used to computationally generate annotations for slides imported into the UC6 prostate-cancer use case. In particular, models are being created to localize and grade tumour tissue. Model inference functionality is being integrated into the DP platform through its slide import workflow. More specifically, once generated by the digital microscopes, new slides transferred and imported into the DP installation through an import workflow that applies any required pre-processing – depending on the specific configuration – and integrates the new data into the image and annotation repositories. This workflow is being extended to support putting the new data through AI-based analysis using the DeepHealth libraries.

Each annotating deep learning model is packaged as an executable Docker container image that provides inference functionality. The import workflow will be configured to invoke these inferencing containers, pass the data to them (via mounting the appropriate data volumes to the container), retrieve the resulting annotations and import them into the annotations database. The DP import workflow can of course be configured to run any number of these annotating container images. Moreover, annotation operations are batched, where a single invocation of the inferencing process will operate on many new microscope slides.

This approach ensures good decoupling between the core platform functionality and the new AI-based functionality. Moreover, by appropriately tagging container images and recording the tag with the computationally generated annotation we have a solid way to track exactly which version of the model was used to generate it and ensure its reproducibility.

While we expect this approach to be sufficiently scalable for the use case requirements, the distributed inference work being produced by T3.3 can easily be integrated should it be required.

7.1.3 Model training

Model training is not part of the DP platform. Instead, models are trained externally and the platform's slide import workflow is configured to use them. More details about the model training pipeline are provided in later sections. Implementing Active Learning in the context of UC6 will be addressed in the next version of this deliverable (D4.2).

7.2 Description of the platform provisions in the context of UC6 - Promort

In a typical clinical or health research pathology workflow, tissue samples are collected, sliced and scanned by a high-resolution digital optical microscope. The **resulting** digital pathology images – also known as *slides* – are generated by special-purpose microscope scanners and are very high in resolution (currently in the order of 100,000 by 250,000 pixels). To facilitate visualization, different image zoom levels are typically computed from the high-resolution data and compiled into a single pyramidal image file. These files are relatively large, on the order of 1 GB per image. Thus, they entail special handling requirements, different from other images in the health domain (e.g., X-ray images). Pathologists examine and annotate these images, marking and measuring regions of interest (ROIs), including tumour areas, and assigning global annotations or scores that describe the overall situation they observe.

The CRS4 Digital Pathology (DP) platform integrates a virtual microscope with slide annotation, tailored for digital pathology activities. It provides a basis to create specialized, vertical applications to support this digital pathology process through a graphical, web-based interface that allows the examining pathologist to interactively zoom and pan around up to full magnification thanks to an image tiling mechanism. The platform also provides a palette of tools to draw and measure accurate ROIs following irregular tissue contours.

7.2.1 Slide annotation

CRS4's DP platform includes functionality to manually annotate digital pathology slides through its web interface. For each slide, the manual annotation process involves two stages:

- mark Regions of Interest (ROI) on the image and acquire measurements;
- give a clinical classification of the previously acquired ROIs.

Acquired annotation data can be exported (e.g., as CSV files) and reports can be automatically generated and sent if required, facilitating integration of the platform in more complex research workflows. The platform has flexible support for annotation workflows (e.g., single or multiple reviews on the same slide, collaborative review, etc.) and roles are defined to manage the users that can perform one or both annotation steps or only access acquired data. This feature allows it to support a variety of study designs.

Within the context of DeepHealth, work is ongoing to refactor the definition of the supported annotations so that it can be extended and reconfigured more easily. First and foremost, this functionality is required to support UC6. For instance, detailed prostate tumour ROI annotation labels for multiple Gleason grades are being added, as are attributes to identify tissue formations (e.g., cribriform patterns, glands). These labels are being used to more finely re-annotate existing images to support the deep learning model-building process. In addition, this functionality will allow the platform to more easily be reconfigured for use with other types of tumour.

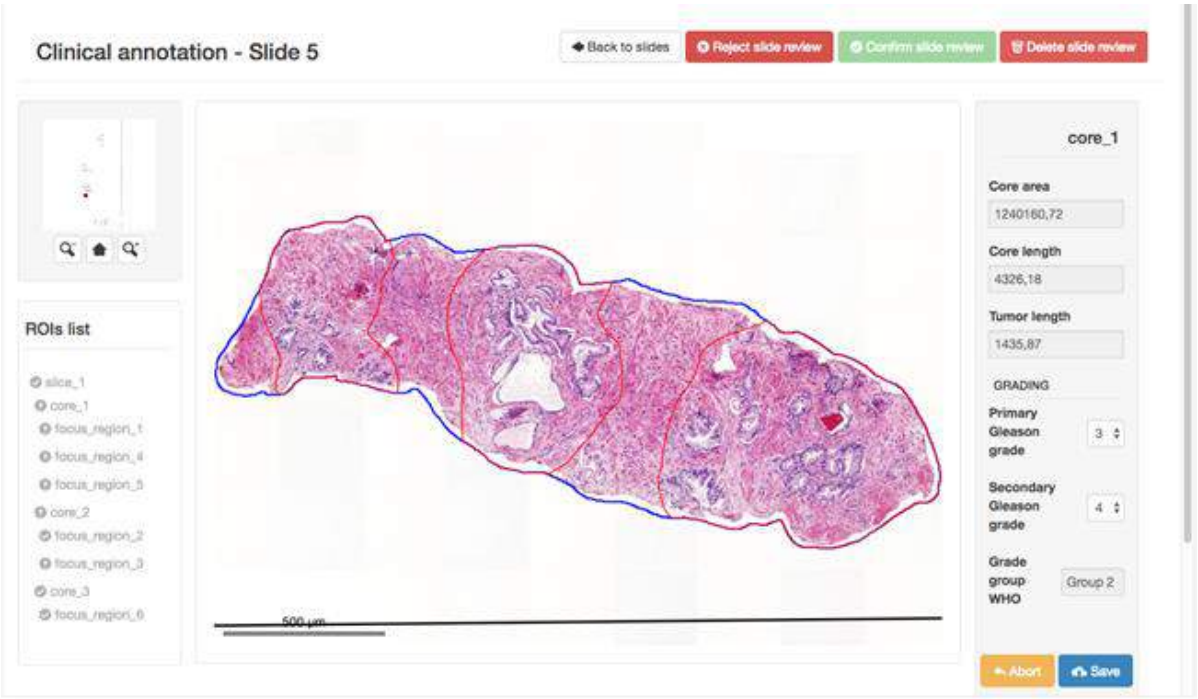


Figure 23. Slide annotation screen.

Currently, the pathologist manually identifies tissue regions as ROIs and attaches labels or information to them. Measurements can also be acquired. DeepHealth integration will support this process with computationally generated annotations.

7.2.2 Computational slide annotation

The overall process to be supported by the DP platform for the UC6 is a continuous *active learning* process [25] as illustrated in Figure 24. In brief, a first model is trained from the currently available data. This first model will be used to put the platform into use, adding computer-generated annotations to the slides that are presented to pathologists in their clinical workflow. At the slide annotation and review stage, the pathologists have the opportunity to provide feedback on the computer-generated annotations. This feedback is then integrated into the learning process, along with any new data, to generate a new and improved model which is in turn tested and deployed – and then the process is repeated.

Work has been started to extend the CRS4 Digital Pathology platform to manage the process of importing new slides and subjecting them to a computational annotation process (see step 2 in Figure 24). The platform's data model has been extended to support the notion of computationally generated annotation – as opposed to human-generated. The conventional annotation model supported by the platform is a closed multi-point polygon, and at the moment the computationally generated annotations have the same restriction. This annotation model entails that probability maps generated by deep learning models will be processed to generate shapes that identify regions of interest. That is, based on a minimum required probability threshold: pixels will be selected and contoured; neighbouring regions will be merged; contours will be smoothed. The resulting contour will form a closed polygon which will be registered as a computationally generated annotation in the platform.

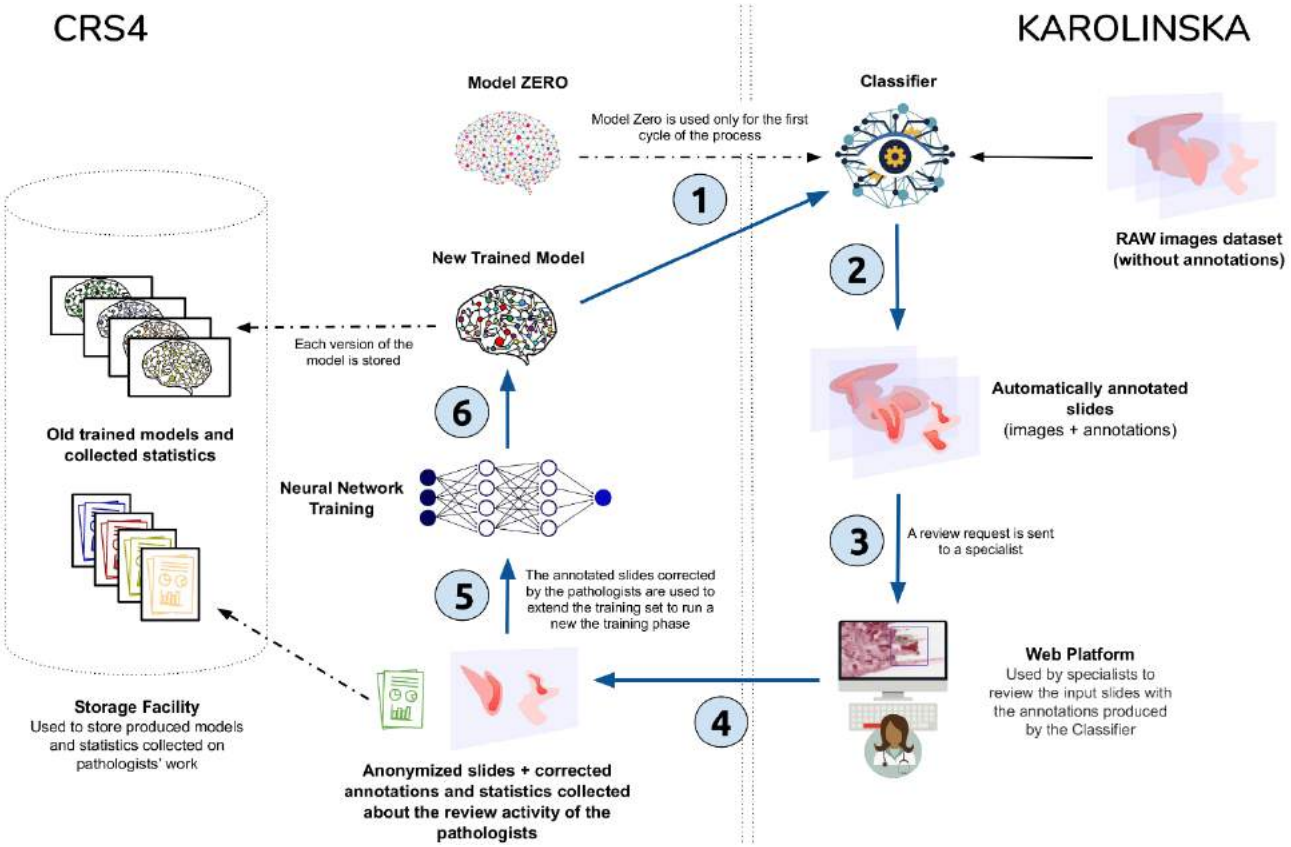


Figure 24. Continuous model improvement process being developed in UC6.

Work is planned to extend the platform's annotation model to support image masks directly – rather than being restricted to closed polygons. This feature will permit more flexibility in the user interface to computationally generate annotations. It will allow the user to superimpose probability heat maps directly onto the image, or to dynamically select a desired contouring probability threshold. CRS4 and KI plan to try various user interface approaches and collect feedback from pathologists in search for a solution that minimizes the "time to annotate slide" KPI.

Simultaneously, the platform design is being updated to integrate the deep learning models that, through inference, will be used to generate the annotations (model training is discussed later in this report). Automatic annotation will be performed as an additional step during the slide import process – an offline process that happens prior to the slides being collected in worklists for pathologists. In this first integration that is being implemented, the platform will use the models through abstract annotation-generating functions working on the whole slide, which will allow it to support any kind of

ensemble or complex model design. The annotation evaluation and feedback mechanisms are still being studied and will be reported in the next version of this deliverable.

7.3 Description of the platform interactions with the DeepHealth toolbox

7.3.1 DH Front-end

The model training process is performed through an external pipeline, which generates a deep neural network model to be then loaded into the DP platform where it can be used for inference. The DeepHealth front-end is thus not applicable to PF6/UC6

7.3.2 EDDLL

7.3.2.1 Identification of tumour areas

Early implementations of the models were prototyped using the Keras API and the Tensorflow library. This approach allowed to begin work on the model early in the DeepHealth project timeline, while the EDDLL and ECVL were still in preliminary development, and identify key features that were required from the DeepHealth libraries for a successful outcome of PF6 and the related use case UC6.

Based on initial experiments, a patch classification approach has been adopted (as opposed to, for instance, semantic segmentation via a U-Net). This approach decomposes the histopathology slides in small patches (e.g., 256x256 pixels) and classifies each tissue patch as a whole (as normal/tumoral or grades the degree of tumour severity.

Following on exploratory work testing different network architectures, an approach based on a pretrained VGG-16 [28] was adopted. The VGG-16 is a network model used on various deep learning classification problems. For our problem, a VGG-16 network pretrained on the standard ImageNet database was selected and its last layers were retrained on the prostate tumour dataset.

The standard VGG-16 network is now well supported by the EDDLL, along with the functionality required for the use of transfer learning. Therefore, work to port the training pipeline and model from Keras and Tensorflow to EDDLL has been started.

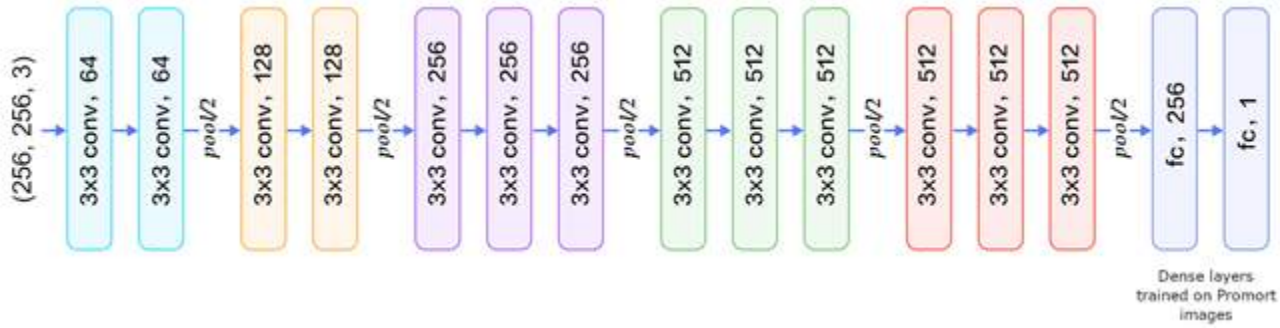


Figure 25. VGG-16 architecture. The convolutional layers have been trained on ImageNet database. With respect to the original VGG-16 architecture, the last three fully connected layers have been replaced by two dense layers with different sizes and trained on the UC6 prostate tumour images (source of the original image: https://bit.ly/2T9O2a2)

7.3.2.2 Gleason grading

This work will be described in the next WP4 deliverable (D4.2).

7.3.3 ECVL

The ECVL will be used for the image pre-processing stage in both model training and inference. More specifically, given the patch classification approach adopted for UC6, training or applying the model requires the extraction of 256x256-pixel patches from the images.

For model training, the workflow begins by processing selected images in the repository to extract a large set of patches at maximum (40x optical) magnification. We begin by computing tissue masks to avoid background areas of the slides. The tissue mask is obtained by means of a Linear Discriminant

Analysis (LDA) classifier trained on tissue and background regions of randomly selected slides. The patch extraction is parameterized to generate patch sets with required characteristics, in particular class balance and minimum tissue coverage ratio (a patch is considered valid if the tissue coverage is higher than a set threshold). After one or more patch sets have been created, three subsets of patches are selected to create the training, validation and test set. To minimize correlations among these sets, which could result in overestimating classification performance, case identifiers are used to ensure that data from any given case is used to generate patches for only one of the three patch sets.

Currently, this prototype pre-processing and patch generation pipeline is based on the OpenSlide library and custom code. In the near future, we plan on porting it to the ECVL. In particular, the ECVL has recently integrated OpenSlide, which entails that the UC6 pipeline will be able to load and process the digital microscope slides directly with the ECVL and thus easily leverage the features it provides to improve the model building pipeline through relevant features – e.g., data augmentation.

On the other hand, image handling requirements during model inference are simpler. They consist in accessing the slide data at the zoom level appropriate for the model and systematically iterating over the image patches. This operation has also been prototyped with the OpenSlide library and custom code, but we expect it to be trivial to port to the ECVL with the recent integration of OpenSlide.

7.4 Current status and next steps

In summary, the DP platform has been extended to support the notion of computationally generated annotations. Work is ongoing to refactor the definition of the supported image annotations so that it can be extended for the requirements of UC6 and, in general, reconfigured more easily. Deep learning models for prostate tumour identification have been prototyped, along with their image pre-processing pipelines. These components are still under active development. The platform's software components have been containerized and currently run in docker-compose. Work has also been started to extend the platform's image import workflow to integrate model inferencing functionality.

In the following period, ongoing activities will be continued. Deep learning models and pipelines will be ported to the DeepHealth toolkit and integration with the DP platform will be completed. The platform will also be extended to implement the model feedback mechanism which requires active learning. Further, a Gleason grading model will be created and the cloud transformation of the platform will be completed by porting it to run on Kubernetes.

Plans for midterm review (M18): During the midterm review in M18, the platform PF6 is envisioned to demonstrate its integration with the DeepHealth outcomes using UC6 Promort. In particular, the platform will demonstrate:

- Manual slide annotation by pathologist.
- Running VGG-16 training pipeline using ECVL and EDDLL.
- Screenshots of tumour localization output generated by a DeepHealth-based deep learning model.

8 PF7 - everisLumen (EVERIS)

8.1 Description of the platform as adjusted for the purposes of DeepHealth

Lumen is a Jupyter Notebook-based platform deployed in Amazon Web Services (AWS), in a Kubernetes Cluster. The integration tasks of DeepHealth toolkit (EDDL & ECVL libraries, using the pyEDDL and pyECVL) are the following:

- Create a Docker for CPU with Jupyter (For exploration)
  - Docker image with EDDL & ECVL libraries
  - Docker container running in Lumen with pyEDDL & pyECVL installed in a python environment
- Create a Docker for CPU (For Training & Testing/Inference)
  - Docker image with EDDL & ECVL libraries
  - Docker container running in Lumen with pyEDDL & pyECVL installed in a python environment
- Create a Docker for GPU with Jupyter (For exploration)
  - Docker image with EDDL & ECVL libraries
  - Docker container running in Lumen with pyEDDL & pyECVL installed in a python environment
- Create a Docker for GPU (For Training & Testing/Inference)
  - Docker image with EDDL & ECVL libraries
  - Docker container running in Lumen with pyEDDL & pyECVL installed in a python environment

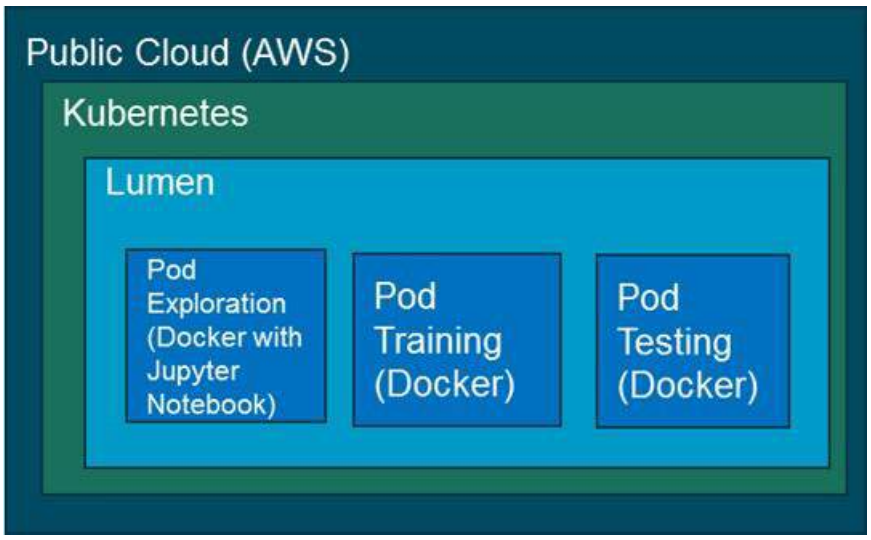


Figure 26. everisLumen architecture

Privacy concerns may arise if neural networks are trained in a server outside the premises of the owners of the dataset. This training outside the premises can be chosen to be able to use high performance computers or to be able to train a single network from databases located in different locations.

One of the options to be able to guarantee privacy and distribute computing is split learning. The philosophy behind split learning is that the first part of the network is trained on premises on the client, but the vast majority of the computation is performed in a server. Some features of the training images have to be sent from the client to the server, as well as some gradients have to be sent from the server to the client(s) during training.

The objective of this exploratory research is to create a minimum viable product that is able to:

- Divide a pretrained network into 2 different models with minimal intervention;
- Allow the possibility of several clients working together to train the same network;
- Explore the possibility of distributing the training in the server;
- Explore the possibility of adding an encoder to the client and a decoder to the server to lower the amount of data transmission between client and server; and
- Lower the amount of information that can be recovered of the original images from the features since the beginning of the training, and define a metric.

More details about split learning and the followed approach can be found in [30] and [31]. More specifically, the authors in [30] explain the initial concept and how to divide a network while authors in [31] address the necessity for increased privacy security in the considered system.

8.2 UC9: Lumbar spine

8.2.1 Description of the platform provisions in the context of the UC

The aim of UC9 is to obtain the structures of lumbar spine segmentation to study clinical-radiological correlations in lumbar physiopathology's following the workflow presented in Figure 17.

As already described in section 6.6, the dataset consists of 23688 studies belonging to the 2015-2016 period of years. All this data has been extracted from BIMCV and it is distributed between 17 sanity departments. In total, this dataset contains more than 124,000 RM images. This dataset has been structure with MIDS Standard.

To allow the use of the data from the everisLumen platform, the MR scans need to be pre-processed using:

- Linear coregister;
- Filtering;
- Magnetic field inhomogeneity correction; and
- Normalization.

The anonymized datasets, having being pre-processed, will be uploaded to Lumen storage during the training and test phases according to FISABIO directives. If this anonymization could not be completed or there are other legal issues, another option would be to do a training phase with sensitive information in FISABIO without using Lumen (using images with sensitive information), to extract features for the following phase of the training. Once these features are available, we could use them to continue the training in Lumen, uploading the features to Lumen storage, as represented in Figure 27.

The first phase of the training will use Sensitive Information in the format of Medical images (and therefore will be performed in the premises of the data provider) while the second phase of the training will be based on the features extracted from the Medical Images (i.e., using anonymized information).

8.2.2 Description of the platform interactions with the DeepHealth toolbox

8.2.2.1 DH Front-end

Not applicable.

8.2.2.2 EDDLL

EDDLL will be used in training and testing/inference. To this end, Lumen will be provisioned with Docker images for CPU and GPU, with EDDLL libraries installed in them. Moreover, in theses Docker containers pyEDDLL will be installed in a python environment to develop training and testing/inference. It is not defined at this moment which functionalities will be used for the specific UC.

8.2.2.3 ECVL

ECVL will be used in training and testing/inference. To this end, Lumen will be provisioned with Docker images for CPU and GPU, with ECVL libraries installed in them. Moreover, in theses Docker

containers pyECVL will be installed in a python environment to develop training and testing/inference. It is not defined at this moment which functionalities will be used for the specific UC.

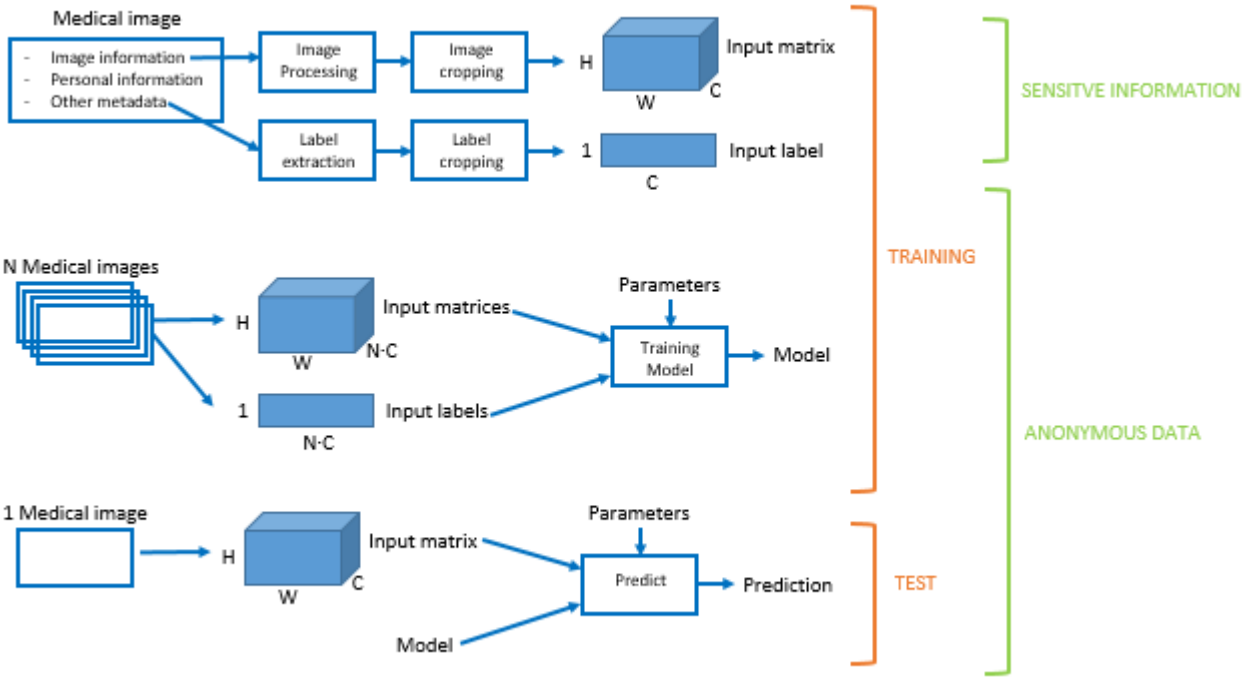


Figure 27. Training in two phases: 1) with sensitive information on-premises 2) with anonymous in Lumen.

8.3 UC10: Alzheimer's Disease

8.3.1 Description of the platform provisions in the context of the UC

As already analysed in section 6.7, the objective of UC10 is to obtain a brain magnetic resonance imaging classifier in Alzheimer Disease using deep learning techniques and it is a representative problem of classification, following the scheme of Figure 20.

The considered dataset consists of 2461 cases extracted from BIMCV (Medical Imaging Databank of the Valencia Region). This data will be combined with Alzheimer Disease Neuroimaging Initiative (ADNI) and Open Access Series of Imaging Studies (OASIS). This dataset will be in Medical Imaging Data Structure standard (MIDS) as an input format and all the MRI are completely anonymized and will be store on the open DeepHealth storage system.

Before feeding the data to the everisLumen platform, the MR scans need also to be pre-processed using (see Figure 19):

- Parenchyma extraction (FLS-BET);
- Linear transformation from native to reference image (FLS-FLIRT); and
- Freesurfer processing to build the Ground Truth.

Similarly to the support provided by everisLumen platform to UC9 (presented in Section 8.2.1), in this case as well, the anonymized datasets will be uploaded to Lumen storage during the training and test phases according to FISABIO directives. If this anonymization could not be completed or there are other legal issues, another option would be to do a training phase with sensitive information in FISABIO without using Lumen (using images with sensitive information), to extract features for the following phase of training. Once these features become available, we could use the extracted features and continue the training in Lumen, uploading the features to Lumen storage (see Figure 27).

8.3.2 Description of the platform interactions with the DeepHealth toolbox

everisLumen platform will require the same integration actions as those described in section 8.2.2.

8.3.2.1 DH Front-end

Not applicable.

8.3.2.2 EDDLL

EDDLL will be used in training and testing/inference. To this end, Lumen will be provisioned with Docker images for CPU and GPU, with EDDLL libraries installed in them. Moreover, in these Docker containers pyEDDLL will be installed in a python environment to develop training and testing/inference. It is not defined at this moment which functionalities will be used for the specific UC.

8.3.2.3 ECVL

ECVL will be used in training and testing/inference. To this end, Lumen will be provisioned with Docker images for CPU and GPU, with ECVL libraries installed in them. Moreover, in theses Docker containers pyECVL will be installed in a python environment to develop training and testing/inference. It is not defined at this moment which functionalities will be used for the specific UC.

8.4 UC12: Skin Cancer

8.4.1 Description of the platform provisions in the context of the UC

For this UC we are going to use a public dataset of skin lesion images available at https://challenge2018.isic-archive.com/. The dataset will be uploaded to Lumen storage, in a provisioned bucket AWS S3. Lumen will be provisioned with Docker images for CPU and GPU, with EDDL & ECVL libraries installed in them. Moreover, in theses Docker containers pyEDDL & pyECVL will be installed in a python environment to develop training and testing/inference. In section 8.5 we can find how the platform will be used in UC12.

8.4.2 Description of the platform interactions with the DeepHealth toolbox

8.4.2.1 Training pipeline using EDDL & ECVL

The integration of UC12 in Lumen platform will be based on both ECVL and EDDL libraries. The UC will load a dataset of skin lesion images. The dataset is described with the *DeepHealth Toolkit Dataset Format*, which is a simple and flexible YAML syntax to describe a dataset for the DeepHealth libraries (EDDL/ECVL). The format includes the definition of:

- A name for the dataset (optional);
- A textual description (optional);
- An array with the names of the classes available in the dataset (optional);
- An array with the names of the features available in the dataset (optional);
- An array with the list of images; and
- A dictionary that specifies the splits of the dataset and the images assigned to them (optional).

After including the desired modules, we start setting up the basic information required for the task, that is the number of classes, and the size of input images expected by the network. Then we can proceed to create an input layer for the model, connect it, and specify the output function, the optimizer, and whether to use CPU of GPU hardware architecture.

The dataset images are then loaded thanks to ECVL functionalities and the training process can start.

The YAML definition of the dataset provided by ECVL is extremely flexible and allows to automatically load batches, one at a time, and then feed them to EDDL train_batch() function.

8.4.2.2 Testing/inference using EDDL & ECVL

The inference step is basically the same, but without the call to train_batch(), which is substituted with evaluate().

8.5 Current status and next steps

The activities and outcomes carried out so far have been:

- Integration completed in Lumen of Docker images with pyEDDL & pyECVL for CPU, using early versions of libraries:
  - Docker for CPU with Jupyter (For exploration)
    - Docker image with EDDL & ECVL libraries
    - Docker container running in Lumen with pyEDDL & pyECVL installed in a python environment
  - Docker for CPU (For Training & Testing)
    - Docker image with EDDL & ECVL libraries
    - Docker container running in Lumen with pyEDDL & pyECVL installed in a python environment

The plan to complete the integration of DH libraries is the following:

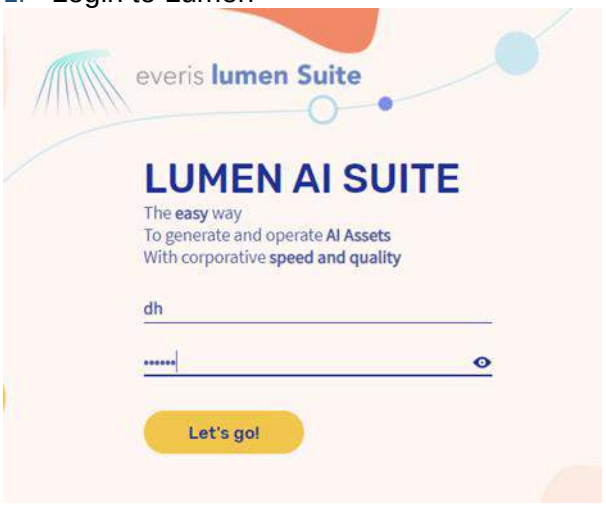
- Upgrade Docker images for CPU to latest versions of libraries:
  - Docker for CPU with Jupyter (For exploration)
  - Docker for CPU (For Training & Testing)
- Create Docker image for GPU using latest versions of libraries:
  - Docker for GPU with Jupyter (For exploration)
  - Docker for GPU (For Training & Testing)

Plans for the Midterm review (M18): During the midterm review in M18, the platform Lumen is envisioned to demonstrate its integration with the EDDL & ECVL libraries. In particular, the platform will demonstrate its functionality for training and testing/inference in a video, in the following way:

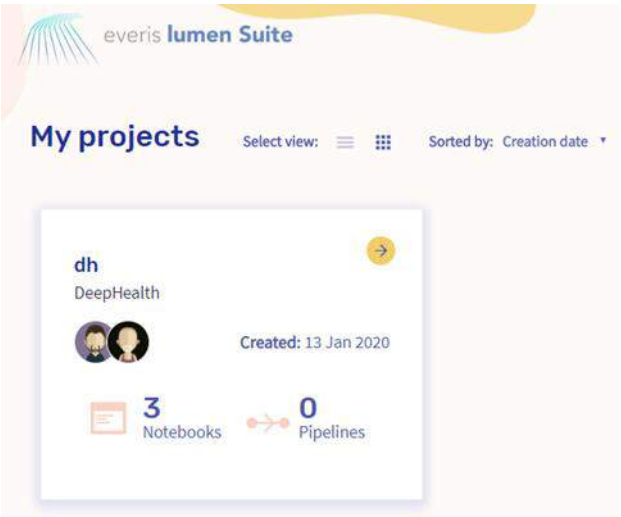
Training

Show a training pipeline using ECVL & EDDL, ideally using the public dataset for UC12 Skin Cancer melanoma detection. The steps for the training functionality are as follows:

1. Login to Lumen

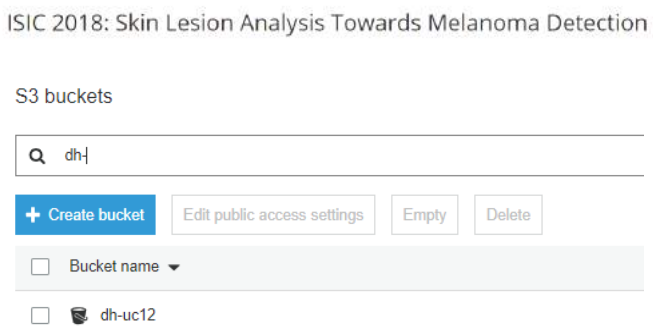


2. Project in Lumen with Python environments and Notebooks

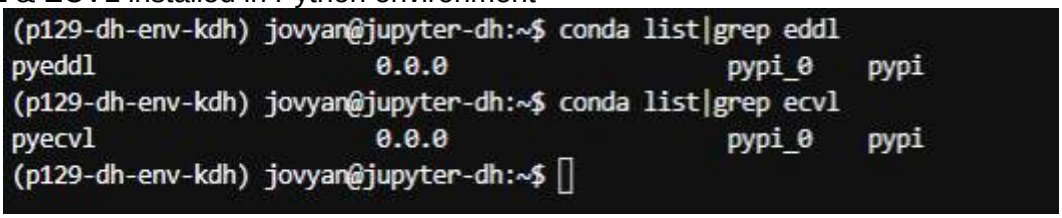


3. Notebooks with the training pipeline

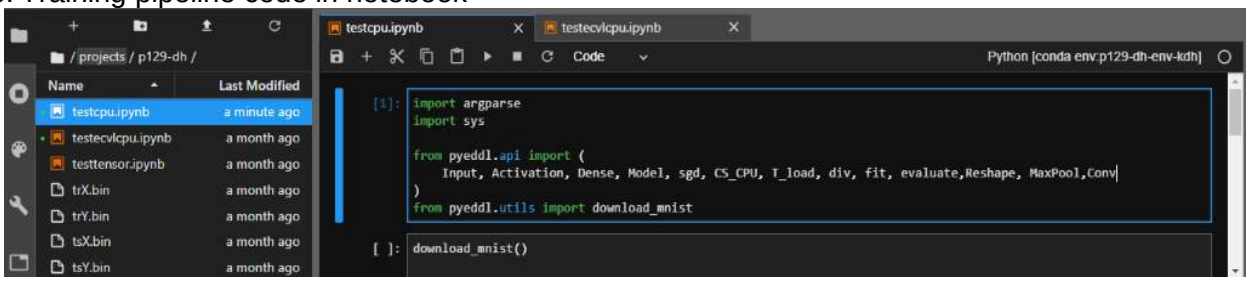
4. Dataset for training stored in AWS S3



5. EDDL & ECVL installed in Python environment

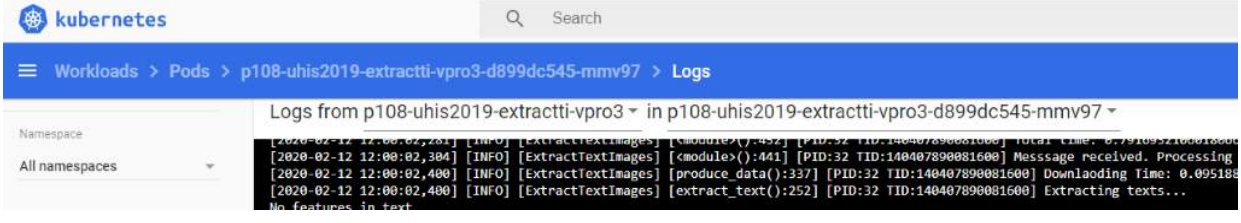


6. Training pipeline code in notebook

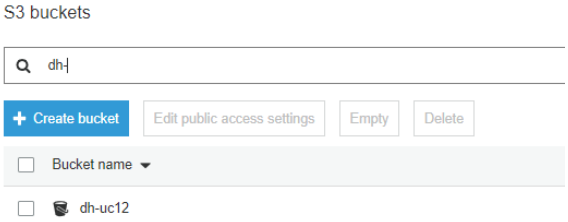


7. Version the notebooks & Build Docker for training pipeline

8. Launch training pipeline & check logs



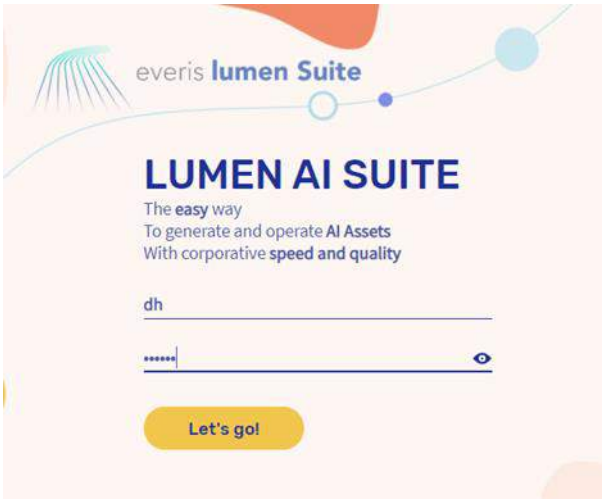
9. Check generated model in AWS S3



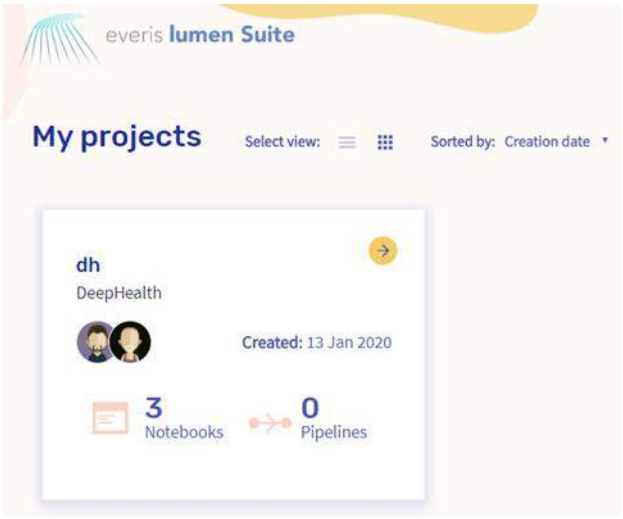
Testing/Inference

Show a testing/inference development, using ECVL & EDD, ideally using the public dataset for UC12 Skin Cancer melanoma detection. The steps for the testing/inference functionality is as follows:

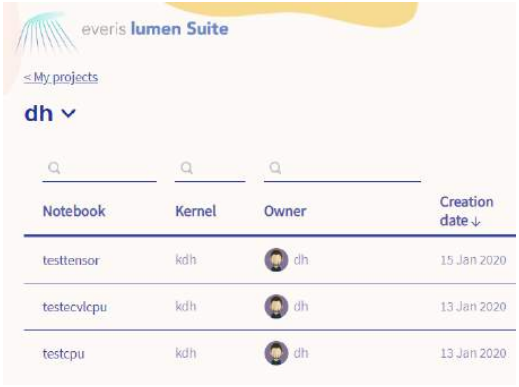
1. Login to Lumen



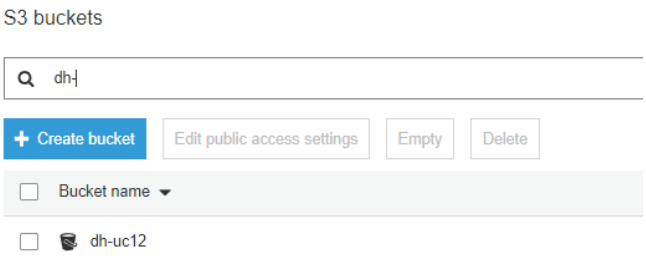
2. Project in Lumen with Python environments and Notebooks



3. Notebooks with the training pipeline



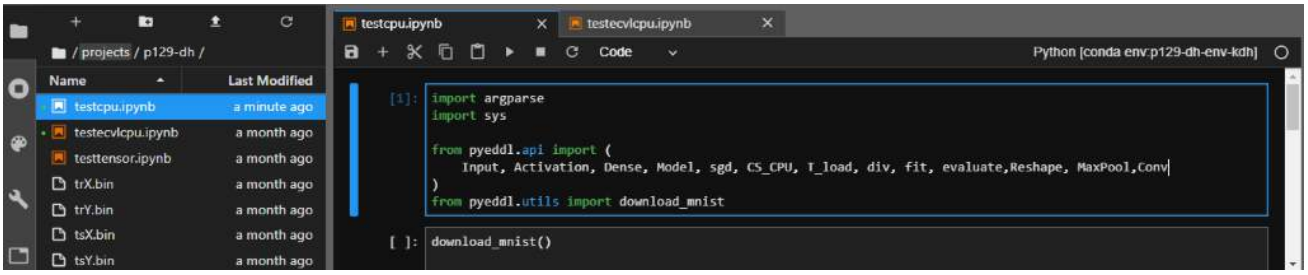
4. Model previously trained and stored in AWS S3



5. EDDL & ECVL installed in Python environment

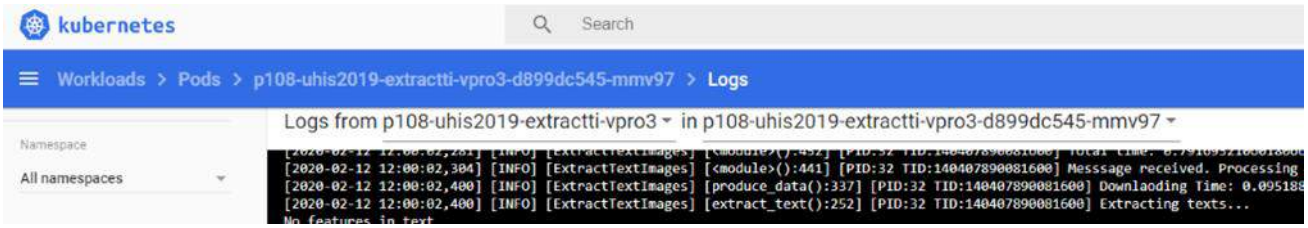
```
(p129-dh-env-kdh) jovyan@jupyter-dh:~$ conda list|grep eddl
pyeddl                    0.0.0                    pypi_0    pypi
(p129-dh-env-kdh) jovyan@jupyter-dh:~$ conda list|grep ecvl
pyecvl                    0.0.0                    pypi_0    pypi
(p129-dh-env-kdh) jovyan@jupyter-dh:~$
```

6. Testing/inference Python code in notebook

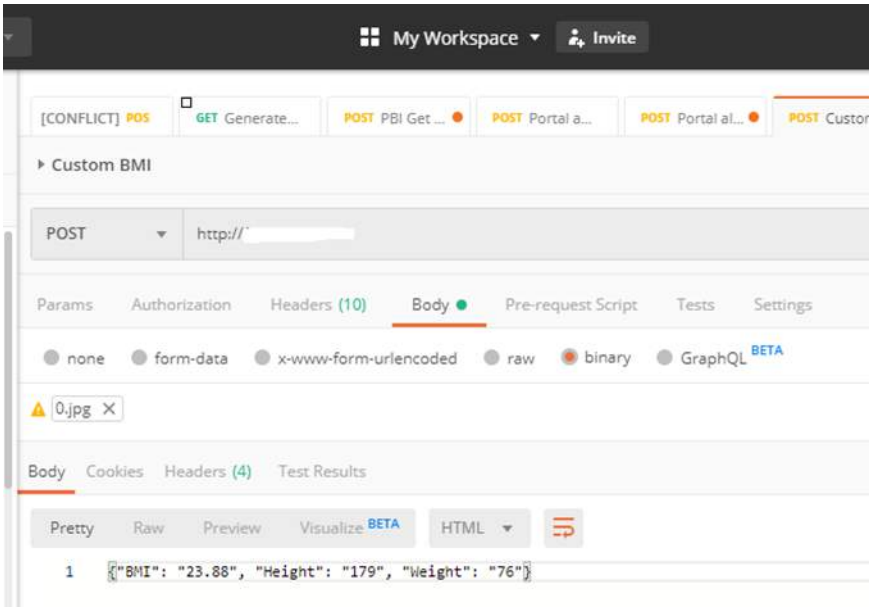


7. Version notebook code & Build a Docker with testing/inference development

8. Deploy testing/inference build & check logs



9. Try testing sending an image, using Postman (REST client) and check inference in the response:



9 Conclusions

This deliverable is the interim report of the integration activities of libraries and use cases in Application Platforms, encompasses all partners platforms and use cases set in the project and reports all integration efforts for each associated platform and its related use cases already performed or designed by M15 (March 2020).

Information related to the platform architecture of each platform and its adaptation (where applicable) in order to meet DeepHealth objectives and challenges is presented for each of the seven (7) platforms of the project. Moreover, for each platform the authors detail how the platform will be used to support the use cases as defined in WP1. More specifically, for each "platform - UC" combination the involved partners analyse the datasets to be used as inputs to the platform for addressing the problems indicated by the use case. Information with respect to any required pre-processing activities, the technologies to be used and the models to be created and/or tested is also provided. The planned approach for integration of the DeepHealth outcomes (DH Front-end, EDDLL, ECVL) with the each platform as a whole or for either each "platform-UC" combination (when it is UC specific) is accordingly detailed. Each section concludes with the summary of the current progress, i.e., the already performed activities, the next steps for fully integrating the DH outcomes with the platform and the plans / foreseen prograss by M18 review.

The final deliverable of WP4 that will report on all the integration activities and efforts is D4.2 "Integration of DeepHealth platforms and use cases (II)" and is due M30 (June 2022).

10 References

[1]. J.-P. Poli et L. Boudet, «A Fuzzy Expert System Architecture For Data and Event Stream Processing» Fuzzy Set and Systems , p. available online 10/12/2017, 2017
[2]. D. Gunning, «Explainable Artificial Intelligence (XAI)» 2016
[3]. L. Zadeh, «Fuzzy sets» Information and Control, vol. 8, n° %13, pp. 338-353, 1965
[4]. L. Zadeh, «The concept of a linguistic variable and its application to approximate reasoning» Information Sciences, vol. 9, n° %13, pp. 43-80, 1975.
[5]. J.-P. Poli, L. Boudet et D. Mercier, «Online Temporal Reasoning For Event And Data Streams Processing» FuzzIEEE, pp. 2257-2264, 2016
[6]. I. Bloch, «Fuzzy spatial relationships for image processing and interpretation: a review» Image and Vision Computing, vol. 23, n° %12, 2005
[7]. J.-M. Le Yaouanc et J.-P. Poli, «A Fuzzy Spatio-Temporal-based Approach for activity recognition» Proceedings of 2012 ER Conference, 2012
[8]. J.-P. Poli, L. Boudet et J.-M. Le Yaouanc, «Online Spatio-Temporal Fuzzy Relations» FuzzIEEE, p. Accepted for publication, 07 2018
[9]. R. Pierrard, L. Cabaret, J.-P. Poli et C. Hudelot, «SIMD-based Exact Parallel Fuzzy Dilation Operator for Fast Computing of Fuzzy Spatial Relations» chez WPMVP, 2020
[10]. R. Pierrard, J.-P. Poli et C. Hudelot, «A New Approach for Explainable Multiple Organ Annotation with Few Data» chez IJCAI 2019 Workshop on Explainable Artificial Intelligence (XAI), 2019
[11]. S. Ruiz-España, E. Arana et D. Moratal, «Semiautomatic computer-aided classification of degenerative lumbar spine disease in magnetic resonance imaging» Computers in Biology and Medicine, vol. 62, pp. 196-205, 2015.
[12]. S. Ghosh, R. S. Alomari, V. Chaudhary et G. Dhillon, «Computer-aided diagnosis for lumbar mri using heterogeneous classifiers» chez 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2011.
[13]. I. Castro-Mateos, R. Hua, J. M. Pozo, A. Lazary et A. F. Frangi, «Intervertebral disc classification by its degree of degeneration from T2-weighted magnetic resonance images» European Spine Journal, vol. 25, n° %19, pp. 2721-2727, 2016.
[14]. S. Ghosh, R. S. Alomari, V. Chaudhary et G. Dhillon, «Composite features for automatic diagnosis of intervertebral disc herniation from lumbar MRI» chez 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2011.
[15]. M. Rak et K. D. Tönnies, «On computerized methods for spine analysis in MRI: a systematic review» computer assisted radiology and surgery, vol. 11, n° %18, pp. 1445-1465, 2016.
[16]. M. Vanegas, J. Inglada et I. Bloch, «Alignment and parallelism for the description of high-resolution remote sensing images» IEEE Transactions on Geoscience and Remote Sensing, pp. 3542-3557, 2013.
[17]. K. Korotkov et R. Garcia, «Methodological review: Computerized analysis of pigmented skin lesions: A review» Artificial Intelligence in Medicine, vol. 56, n° %12, pp. 69-90, 2012.
[18]. A. Masood et A. A. Al-Jumaily, «Computer Aided Diagnostic Support System for Skin Cancer: A Review of Techniques and Algorithms» International Journal of Biomedical Imaging, vol. 2013, pp. 323268-323268, 2013.
[19]. C. Barata, M. E. Celebi et J. S. Marques, «A Survey of Feature Extraction in Dermoscopy Image Analysis of Skin Cancer» IEEE Journal of Biomedical and Health Informatics, vol. 23, n° %13, pp. 1096-1109, 2019.
[20]. S. Pathan, K. G. Prabhu et P. Siddalingaswamy, «Techniques and algorithms for computer aided diagnosis of pigmented skin lesions—A review» Biomedical Signal Processing and Control, vol. 39, pp. 237-262, 2018.
[21]. R. B. Oliveira, J. P. Papa, A. S. Pereira et J. M. Tavares, «Computational methods for pigmented skin lesion classification in images: review and future trends» Neural Computing and Applications, vol. 29, n° %13, pp. 613-636, 2018.
[22]. P. Mehta et B. Shah, «Review on Techniques and Steps of Computer Aided Skin Cancer Diagnosis,» Procedia Computer Science, vol. 85, pp. 309-316, 2016.

[23]. A. Khan, K. Gupta, R. Stanley, W. V. Stoecker, R. H. Moss, G. Argenziano, H. P. Soyer, H. S. Rabinovitz et A. B. Cognetta, «Fuzzy logic techniques for blotch feature evaluation in dermoscopy images,» Computerized Medical Imaging and Graphics, vol. 33, n° %11, pp. 50-57, 2009.
[24]. V. T. Ng, B. Y. Fung et T. K. Lee, «Determining the asymmetry of skin lesion with fuzzy borders» Computers in Biology and Medicine, vol. 35, n° %12, pp. 103-120, 2005.
[25]. Rączkowski, Ł., Możejko, M., Zambonelli, J. et al. ARA: accurate, reliable and active histopathological image classification framework with Bayesian deep learning. Sci Rep 9, 14347 (2019).
[26]. Allan, C., Burel, J., Moore, J., et al. (2012) OMERO: flexible, model-driven data management for experimental biology. Nature Methods 9, 245–253. Published: 28 February 2012. Doi: 10.1038/nmeth.1896.
[27]. Goode, A., Gilbert, B., Harkes, J., et al. OpenSlide: A Vendor-Neutral Software Foundation for Digital Pathology. Journal of Pathology Informatics 2013, 4:27.
[28]. K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015, https://arxiv.org/abs/1409.1556
[29]. Fujieda, Shin, Kohei Takayama, and Toshiya Hachisuka. "Wavelet convolutional neural networks for texture classification." arXiv preprint arXiv:1707.07394 (2017).
[30]. Gupta, O., & Raskar, R. (2018). Distributed learning of deep neural network over multiple agents. Journal of Network and Computer Applications, 116, 1-8.
[31]. Vepakomma, P., Gupta, O., Dubey, A., & Raskar, R. (2019). Reducing leakage in distributed deep learning for sensitive health data. arXiv preprint arXiv:1812.00564. https://www.researchgate.net/profile/Praneeth_Vepakomma2/publication/333171913_Reducing_leakage_in_distributed_deep_learning_for_sensitive_health_data/links/5cdeb29a299bf14d95a1772c/Reducing-leakage-in-distributed-deep-learning-for-sensitive-health-data.pdf