DEEPHEALTH

D1.4 Definition and Specification of the Deephealth toolkit

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:	WP1/ T1.4
Dissemination level:	PUBLIC
Document due Date:	31/07/2019 (M7)
Actual date of delivery	26/07/2019 (M7)
Leader of this deliverable	THALES SIX
Author (s)	Andreina Chietera
Contributor(s)	UPV, BSC, UNIMORE, SIVECO
Version	V1.0

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

Document history

Version	Date	Document history/approvals
1	27/05/2019	First draft contents
2	14/06/2019	Refinement of the draft content
3	25/06/2019	Preliminary draft content
4	03/07/2019	Requests to partners to draft the toolkit requirements
5	19/07/2019	Refinement content in Chapter 5, conclusion and refs
6	22/07/2019	Consolidated document provided to all partners for confidentiality check and for final internal peer review
1.0	25/07/19	Final quality check after peer review ended
1.0	26/07/19	Submitted document to EC

Table of contents

1 Executive summary

The main goal of the DeepHealth toolkit is to provide the infrastructure necessary to train predictive models by using very large image datasets on distributed architectures. This purpose is realised through the development of a European Distributed Deep Learning Library (EDDLL) providing advanced programming models optimized for hybrid parallel execution. EDDLL represents the main core library of the toolkit and, in order to complete the deep learning pipeline, it is coupled with the European Computer Vision Library (ECVL) devoted to image preprocessing. The combination of these two libraries will offer a stand-alone engine able to cover the entire workflow to apply deep-learning to medical applications.

The aim of this deliverable is threefold: (i) give a brief introduction to deep learning with pointers to core references for medical application tasks in order to identify the main functionalities the toolkit must cover in order to provide an infrastructure with a modular pipeline adapted to the deep learning approach; (ii) indicate specific deep learning state-of-the-art architectures, including DL topologies for classification and segmentation in order to help the prediction of diseases; (iii) point out the elements of the deep learning pipeline to facilitate the development of applications related to medical imaging in order to guide expert users working in hospitals interested in experimenting. To conclude, we present in chapter 6 the global DeepHealth infrastructure specifications to answer to the deep-element processing chain highlighted in the previous chapters. In particular, chapter 6 includes the description of the main deep learning pipeline components like data loading, data augmentations, network architectures, loss functions and evaluation metrics considering computational optimization on hybrid hardware infrastructure.

2 Introduction

Healthcare sector represents a high priority domain that has to satisfy the increasing people expectation ensuring a high level of health care and services and, at the same time, it has to fit with the current political environment focalized to solve the crisis of high costs in health systems. The introduction of new technologies may actually optimize the sector by introducing innovations that guarantee a better diagnosis reducing the global healthcare cost.

Currently, the interpretations of medical data are performed by human experts. This procedure makes diagnosis prone to errors due to the subjectivity, the complexity of the data/image, extensive variations across different interpreters, a substantial commitment of human resources to handle a lot of data and the consequent fatigue due to the processing workflow.

After the success of deep learning in other fields, medical image analysis and computer-assisted diagnosis are increasingly being addressed with this approach. Moreover, the advances in biological and medical technologies, providing us an explosive volume of biological and physiological data, encourage the tendency to apply effective and efficient automated analysis. Learning from this deluge of data facilitates the understanding of human health and disease. Developed from artificial neural networks, deep learning-based algorithms show promising results in extracting features and learning patterns from complex data. In particular, by using deep-learning computers learn useful representations and features automatically, directly from raw medical data.

Nevertheless, current deep-learning platforms are not flexible enough because they do not provide a complete set of specific functionalities for manipulating and processing images in a more efficient way to increase the productivity of professionals working on biomedical images. Adapting them to this domain of application requires substantial implementation effort merging together different engines. In order to provide a complete tool for expert users working in hospitals, we propose the DeepHealth toolkit as a preferential European infrastructure to support fast prototyping and reproducibility by implementing deep learning methods and modules for medical images analysis.

This deliverable analyses the main aspects of training deep learning models in order to identify the main functionalities and specifications to develop the DeepHealth toolkit. The project ambition is to develop this toolkit as a flexible open-source infrastructure for easily use deep learning techniques in medical imaging. To cover this ambition, the project will provide functionalities to accelerate and simplify the development of medical image analysis and computer-assisted diagnosis addressing 1) preprocessing capabilities, 2) deep-learning-based solutions, and 3) computer services for computing optimization.

3 Design aspects of neural networks

In order to identify the aspects the toolkit has to cover, this section introduces the main concepts of the artificial neural networks and describes the principal features of the more relevant topologies for medical applications.

An overview of the fundamental components of the architectures is given together with the notion related to the mathematical tools to evaluate quality of the network prediction.

Artificial neural networks are structurally and conceptually inspired by human biological nervous system. Perceptron is the first step towards neural networks that was based on human brain system [1]. It consists of one input layer that is directly connected to one output layer to classify *linearly* separable patterns. To increase modelling capabilities and solve more complex patterns, more neurons were combined into a layered architecture i.e., one or more input layers, one or more output layers, and one or more hidden layers. Then, as they evolved into more complex structures, since late 1980s neural networks consist of interconnected neurons that take one or more inputs, perform some processing on the input data, and finally forward the current layer output to one or more next layers [2]. The general architecture of a neural network is shown in Figure 1. Each neuron in the network sums up the input data and applies the activation function to the summed data and finally

provides the output that might be propagated to the next layer. Thus, adding more hidden layers allows to deal with more complex patterns as hidden layers capture non-linear relationships. Neural networks with a complex structure composed of many layers, involving also new activation functions (e.g. ReLU),and new regularization techniques (e.g. Dropout, Batch Normalization, MaxNorm Regularization) are known as Deep Neural network (DNN) [2]. Extra layers in DNN enable composition of features from lower layers to the upper layer by giving the potential of modelling complex data.

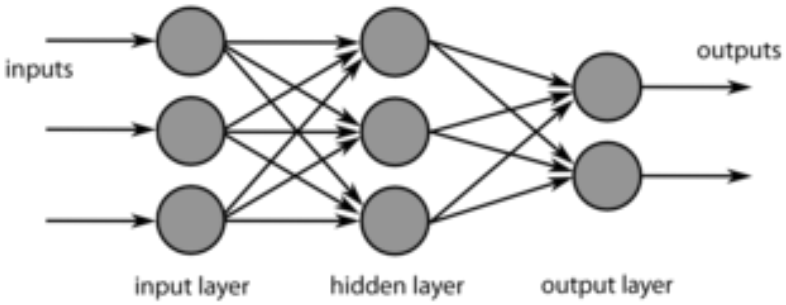

Figure 1. Multi-layer neural network containing more than one layer of artificial neurons or nodes (figure adopted by [3]). Typically, they have at least one input layer, which sends weighted inputs to a series of hidden layers, and an output layer at the end.

Regarding mathematics and theory, neural networks are universal approximators to any function. Thanks to different levels (layers), we can face problems of very high complexity by using neural networks. A feed-forward neural network takes one or more inputs in its input layers and computes the output of all its layers into a data flow, in such a way that all the inputs for a given layer are computed before the output of such layer is computed (a directed and acyclic graph of dependencies is generated for each topology), finally, the output of the neural network is composed by the values of its output layers. The most common topologies have one input layer and one output layer, but in deep learning it is possible to work with topologies having more than one input layer and more than one output layer [2]. Alternatively recurrent neural networks (RNN) are implemented, but they are more difficult to optimize because of their recurrent structure.

The training of a neural network consists in updating the weights that connect the neurons of one layer with the neurons of the next layer. This is done by the Error Backpropagation algorithm, firstly developed in the 1980s. This algorithm computes the difference between the expected output for a given input sample and the output provided by the network, then according to a *loss function* the error is calculated and used to compute the gradient with respect to each of the weights of the network. Gradients are computed applying the chain-rule for derivatives. Then, each weight is updated according to the gradient obtained by deriving the loss function with respect to it. The gradient is multiplied by a coefficient known as the *learning rate* in order to avoid too large changes in the weights. Gradients modify the weights in the direction of minimizing the loss function that is, for improving the prediction of a class or for better approximate the correct output values.

A significant advance towards deep learning is the use of specialized layers [2]. In particular, the so-called *convolution* and *pooling layers* enable to model locality and abstraction as shown in Figure 2. The major advantage of the convolution layers is that they only consider a local neighbourhood for each neuron, and that all neurons of the same layer share the same weights, what dramatically reduces the number of parameters and, therefore, the memory required to store such a layer. Also, the *pooling* is an operation that is used to reduce the scale of the input [4]. Pooling takes subsamples of the convolutional layer to feed the next layer, acting as a powerful detector of patterns independently of their relative position in the image. Weight sharing in convolutional layers combined with pooling schemes (max or average pooling) allow to extract position-invariant relevant properties.

These two modules (convolutional and pooling) are often stacked up, one on top of the other, to form a deep learning model called Convolutional Neural Networks (CNN) [21], that have been found to be highly effective, with high selectivity and invariance, and have been commonly used in computer vision and image recognition. CNNs are also the most common DNN used for biomedical images. They operate on what should be considered a signal stream rather than a feature vector. As opposed to fully connected neural nets consisting of activation units bound to all inputs of a feature vector, where every unit has a weight specific to each feature in the input, convolutional layers utilize weight sharing by sliding a small (moving) filter of weights across the input vector (or 2D input map, as CNNs are often used on images) and convolving each overlapped region of input with the filter.

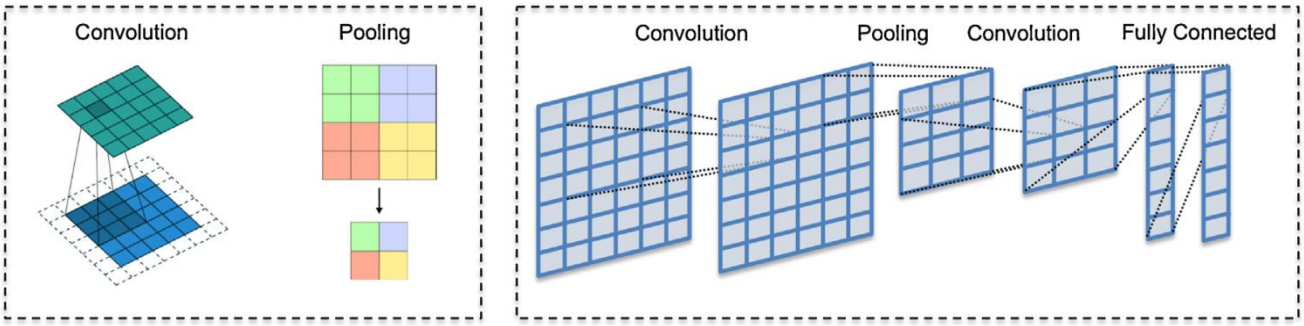

Figure 2. Filters in convolutional layers only consider a limited perceptive field and all neurons involved share the same weights. Pooling layers reduce the size of the intermediate tensors as more complex features are detected. After the convolutional layers they are added fully-connected layers (right side in the figure), the output layer of classifiers is a fully-connected layer [5,6].

In genomics, the concept of weight sharing is very appealing to learn about regulatory elements contained in the randomized regions of the DNA library, since these models can recognize regulatory elements regardless of exact position. Position should only play a minor role in prediction, and this is exactly what the convolutional layers provide: the filters will slide across the input DNA sequence, responding to specific motifs they have been trained to be sensitive towards. However, every overlapped region of the input signal relates to a specific location in the convolved output signal, meaning positional information remains preserved.

Here is presented a far from exhaustive, non-chronological, list of CNN architectures and some high-level descriptions.

AlexNet[a]	The network that launched the current deep learning boom by winning the 2012 ILSVRC competition by a huge margin. Notable features include the use of RELUs activation function, dropout regularization, splitting the computations on multiple GPUs, and using data augmentation during training. ZFNet[b], a relatively minor modification of AlexNet, won the 2013 ILSVRC competition.
VGG[c]	Popularized the idea of using smaller filter kernels and therefore deeper networks (up to 19 layers for VGG19, compared to 7 for AlexNet and ZFNet), and training the deeper networks using pre-training on shallower versions.

[a] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L,Weinberger KQ, editors. Advances in neural information processing systems 25. Curran Associates, Inc.; 2012. p. 1097–105.

[b] ZeilerMD,Fergus R.Visualizing and understanding convolutional networks. In: European conference on computer vision, Springer. 2014. p. 818–33.

[c] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556 (2014).

GoogLeNet[d]	Promoted the idea of stacking the layers in CNNs more creatively, as networks in networks, building on the idea of footnote e. Inside a relatively standard architecture (called the stem), GoogLeNet contains multiple inception modules, in which multiple different filter sizes are applied to the input and their results concatenated. This multi-scale processing allows the module to extract features at different levels of detail simultaneously. GoogLeNet also popularized the idea of not using fully-connected layers at the end, but rather global average pooling, significantly reducing the number of model parameters. It won the 2014 ILSVRC competition.
ResNet[f]	Introduced skip connections, which makes it possible to train much deeper networks. A 152 layer deep ResNet won the 2015 ILSVRC competition and the authors also successfully trained a version with 1001 layers. Having skip layer (more precisely, from ResNet block to ResNet block), preserving information as data goes through the layers. Some features are best constructed in shallow networks, while others require more depth. The skip connections facilitate both at the same time, increasing the network's flexibility when fed input data. As the skip connections make the network learn residuals, ResNets perform a kind of boosting.
Highway nets[g]	Another way to increase depth based on gating units, an idea from Long Short Term Memory (LSTM) recurrent networks, enabling optimization of the skip connections in the network. The gates can be trained to find useful combinations of the identity function (as in ResNets) and the standard nonlinearity through which to feed its input.
DenseNet[h]	Builds on the ideas of ResNet, but instead of adding the activations produced by one layer to later layers, they are simply concatenated together. The original inputs in addition to the activations from previous layers are therefore kept at each layer (again, more precisely, between blocks of layers), preserving some kind of global state. This encourages feature reuse and lowers the number of parameters for a given depth. DenseNets are therefore particularly well-suited for smaller data sets (outperforming others on e.g. Cifar-10 and Cifar-100).
ResNext[i]	Builds on ResNet and GoogLeNet by using inception modules between skip connections.
SENets[j]	Squeeze-and-Excitation Networks, which won the ILSVRC 2017 competition, builds on ResNext but adds trainable parameters that the network can use to weigh each feature map, where earlier networks simply added them up. These SE-blocks allows the network to model the channel and spatial information separately, increasing the model capacity. SE-blocks can easily be added to any CNN model, with negligible increase in computational costs.
NASNet[k]	A CNN architecture designed by a neural network, beating all the previous human-designed networks at the ILSVRC competition. It was created using AutoML, Google Brain's[l]

[d] Szegedy C, LiuW, JiaY, Sermanet P, Reed S, AnguelovD, et al. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2014. p. 1–9.

[e] Lin M, Chen Q, Yan S. Network in network, arXiv:1312.4400 (2013).

[f] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. p. 770–8.

[g] Srivastava RK, Greff K, Schmidhuber J. Training very deep net- works. In: Advances in neural information processing systems. 2015. p. 2377–85.

[h] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: CVPR, vol. 1. 2016..

[i] Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual trans- formations for deep neural networks. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), IEEE. 2016. p. 5987–95.

[j] Hu J, Shen L, Sun G. Squeeze-and-excitation networks, arXiv:1709.01507 (2017).

[k] Zoph B, Vasudevan V, Shlens J, Le QV. Learning transferable architectures for scalable image recognition, arXiv:1707.07012 2 (2017).

[l] https://cloud.google.com/automl.

	reinforcement learning approach to architecture design[m]. A controller network (a recurrent neural network) proposes architectures aimed to perform at a specific level for a particular task, and by trial and error learns to propose better and better models. NASNet was based on Cifar-10, and has relatively modest computational demands, but still outperformed the previous state-of-the-art on ILSVRC data.
YOLO[n]	Introduced a new, simplified way to do simultaneous object detection and classification in images. It uses a single CNN operating directly on the image and outputting bounding boxes and class probabilities. It incorporates several elements from the above networks, including inception modules and pretraining a smaller version of the network. It's fast enough to enable real-time processing. YOLO[o] makes it easy to trade accuracy for speed by reducing the model size. YOLOv3-tiny was able to process images at over 200 frames per second on a standard benchmark data set, while still producing reasonable predictions.
GANs[p]	A generative adversarial network consists of two neural networks pitted against each other. The generative network G is tasked with creating samples that the discriminative network D is supposed to classify as coming from the generative network or the training data. The networks are trained simultaneously, where G aims to maximize the probability that D makes a mistake while D aims for high classification accuracy.
Siamese nets[q]	An old idea[r] that's recently been shown to enable one-shot learning, i.e. learning from a single example. A Siamese network consists of two identical neural networks, both the architecture and the weights, attached at the end. They are trained together to differentiate pairs of inputs. Once trained, the features of the networks can be used to perform one-shot learning without retraining.
U-net[s]	A very popular and successful network for segmentation in 2D images. When fed an input image, it is first down sampled through a "traditional" CNN, before being up-sampled using transpose convolutions until it reaches its original size. In addition, based on the ideas of ResNet, there are skip connections that concatenates features from the down-sampling to the up-sampling paths. It is a fully-convolutional network, using the ideas first introduced in footnote t.
V-net[u]	A three-dimensional version of U-net with volumetric convolutions and skip connections as in ResNet.

[m] Bello I, Zoph B, Vasudevan V, Le QV. Neural optimizer search with reinforcement learning. In: Precup D, Teh YW, editors. Proceedings of the 34th international conference on machine learning. Proceedings of machine learning research, vol. 70. Sydney, Australia: PMLR, International Convention Centre; 2017. p. 459–68.

[n] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. p. 779–88.

[o] the YOLO algorithm demonstration are showed in action https://youtu.be/VOC3huqHrss.

[p] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence N D, Weinberger K Q, editors. Advances in neural informationprocessingsystems27.CurranAssociates, Inc.; 2014. p.2672–80.

[q] Koch G, Zemel R, Salakhutdinov R. Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop, vol. 2. 2015.

[r] Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R. Signature verification using a "Siamese" time delay neural network. In: Advances in neural information processing systems. 1993. p. 737–44.

[s] Ronneberger O, Fischer P, Brox T. U-net: convolutional networks for biomedical images segmentation. In: International conference on med- ical image computing and computer-assisted intervention. 2015. p. 234–41.

[t] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. pp. 31–40.

[u] Milletari F, Navab N, Ahmadi S-A. V-net: fully convolutional neural networks for volumetric medical image segmentation. In: 2016 fourth international conference on 3D Vision (3DV), IEEE. 2016. p. 565–71.

3.1 Image classification and segmentation

Deep learning methods have attained success in a variety of computer vision tasks such as object recognition, localization, and segmentation in many fields. These tools have soon been applied in medical applications. In image segmentation, we aim to determine the outline of an organ or an anatomical structure as accurately as possible. The differentiation of tissues or organs in medical images has been termed as semantic segmentation, where each pixel of an image is assigned to a class or label.

Again, approaches based on convolutional neural networks seem to dominate in medical applications. Here, we only report a table summarizing the main works using the recent V-Net for 3D voxel segmentation, U-Net topology, and other architectures applied to different biomedical applications that can be potentially used for the DeepHealth use cases (see deliverable D1.1), according to our survey.

The tables below show the application of the main topologies to a specific medical application.

V-Net	Approach to 3D image segmentation based on a volumetric, fully convolutional, neural network.	Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation	15 Jun 2016 - Fausto Milletari • Nassir Navab • Seyed-Ahmad Ahmadi		
Task	**Dataset**	**Model**	**Metric name**	**Metric value**	**Global rank**
Volumetric Medical Image Segmentation	PROMISE 2012	V-Net + Dice-based loss	Dice Score	0.869	# 1

U-Net	The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization	U-Net: Convolutional Networks for Biomedical Image Segmentation	18 May 2015 • Olaf Ronneberger • Philipp Fischer • Thomas Brox		
Task	**Dataset**	**Model**	**Metric name**	**Metric value**	**Global rank**
Retinal Vessel Segmentation	CHASE_DB1	U-Net	F1 score	0.7783	# 4
Retinal Vessel Segmentation	CHASE_DB1	U-Net	AUC	0.9772	# 4
Retinal Vessel Segmentation	DRIVE	U-Net	F1 score	0.8142	# 4
Retinal Vessel Segmentation	DRIVE	U-Net	AUC	0.9755	# 4
Retinal Vessel Segmentation	STARE	U-Net	F1 score	0.8373	# 3

Retinal Vessel Segmentation	STARE	U-Net	AUC	0.9898	# 3
Pancreas Segmentation	CT-150	U-Net	Dice Score	0.814	# 2
Pancreas Segmentation	CT-150	U-Net	Precision	0.848	# 2
Pancreas Segmentation	CT-150	U-Net	Recall	0.806	# 2
Medical Image Segmentation	ISBI 2012 EM Segmentation	U-Net	Warping Error	0.000353	# 1
Skin Cancer Segmentation	Kaggle Skin Lesion Segmentation	U-Net	F1 score	0.8682	# 3
Skin Cancer Segmentation	Kaggle Skin Lesion Segmentation	U-Net	AUC	0.9371	# 3
Lung Nodule Segmentation	LUNA	U-Net	F1 score	0.9658	# 3
Lung Nodule Segmentation	LUNA	U-Net	AUC	0.9784	# 3
Cell Segmentation	PhC-U373	U-Net	Mean IoU	0.9203	# 1
Cell Segmentation	DIC-HeLa	U-Net	Mean IoU	0.7756	# 1

Segmentation of tissues and organs is crucial for qualitative and quantitative assessment of medical images. *Pereira et al.* used data augmentation, small convolutional kernels, and a pre-processing stage to achieve accurate brain tumour segmentation [5]. Their CNN-based segmentation method won first place in the Brain Tumor Segmentation (BRATS) Challenge in 2013, and second place in 2015. *Havaei et a*l. presented a fully automatic brain tumour segmentation method based on DNNs in magnetic resonance (MR) images with a two-phase training procedure [6], which obtained second place in the 2013 BRATS, that is summarized in the table below.

Brain Tumor Segmentation with a Fully Convolutional Neural Network	A fully automatic brain tumour segmentation method based on Deep Neural Networks (DNNs). Different from most traditional uses of CNNs, the networks use a final layer that is a convolutional implementation of a fully connected layer which allows a 40 fold speed up.	Fully Convolutional Neural Networks	13 May 2015 • Mohammad Havaei • Axel Davy • David Warde-Farley • Antoine Biard • Aaron Courville • Yoshua Bengio • Chris Pal • Pierre-Marc Jodoin • Hugo Larochelle		
Task	**Dataset**	**Model**	**Metric name**	**Metric value**	**Global rank**
Brain Tumor Segmentation	BRATS-2013	InputCascadeCNN	Dice Score	0.88	# 1

Detection of lesion and abnormality is the major issue in medical image analysis. Deep learning methods learn the representations directly instead of using hand-crafted features from training data. A classifier is then used to assign the representations to a probability that indicates whether or not the image contains lesions. In other words, the deep learning schemas classify each pixel to be a lesion point or not, which can be done in two ways: (1) classifying the mini patch around the pixel with a deep network, and (2) using a fully convolutional network to classify each pixel.

Few examples of lesions detection with different techniques are showed in the tables below.

Task	**Dataset**	**Reference & Model**	**Metric name**	**Metric value**
Segmentation of Multiple Sclerosis lesions on MRI images	MRI (structural) to be acquired	Brosch, T., Tang, L. Y. W., Yoo, Y., Li, D. K. B., Traboulsee, A., & Tam, R. (2016). "Deep 3D Convolutional Encoder Networks with Shortcuts for Multiscale Feature Integration Applied to Multiple Sclerosis Lesion Segmentation".	Dice Score	Unknown
Segmentation of Multiple Sclerosis lesions on MRI images	MICCAI 2008 MS lesion segmentation challenge	Brosch, T., Tang, L. Y. W., Yoo, Y., Li, D. K. B., Traboulsee, A., & Tam, R. (2016). "Deep 3D Convolutional Encoder Networks with Shortcuts for Multiscale Feature Integration Applied to Multiple Sclerosis Lesion Segmentation".	Dice Score	84.1
Segmentation of Multiple Sclerosis lesions on MRI images	ISBI 2015 MS lesion segmentation challenge	Brosch, T., Tang, L. Y. W., Yoo, Y., Li, D. K. B., Traboulsee, A., & Tam, R. (2016). "Deep 3D Convolutional Encoder Networks with Shortcuts for Multiscale Feature Integration Applied to Multiple Sclerosis Lesion Segmentation".	Dice Score	68.4

Task	Dataset	Model	Metric name	Metric value
Skin Lesion Segmentation	ISIC2017 Task 1 - Lesion Boundary Segmentation	DeepLabv3+	Mean IoU	0.769
Skin Lesion Segmentation	ISIC2017 Task 1 - Lesion Boundary Segmentation	U-Net	Mean IoU	0.740
Skin Lesion Segmentation	ISIC2017 Task 1 - Lesion Boundary Segmentation	SegNet	Mean IoU	0.767

Yet another interesting class of segmenting algorithms is the use of recurrent networks for medical image segmentation. *Poudel et al.* demonstrate this for a recurrent fully convolutional neural network on multi-slice MRI cardiac data [7], while *Andermatt et al.* show the effectiveness of GRUs for brain segmentation [8]. Another extension is represented by long-short-term memory (LSTM) networks [9]. A typical LSTM architecture is presented in Figure 3.

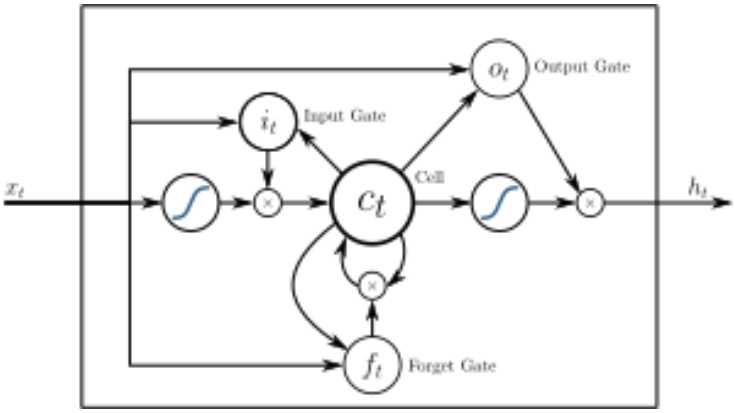

Figure 3. A common LSTM unit is composed of a cell with three internal gates: input gate, output gate and forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information between the input and the output of the cell.

3.2 Pattern detection and recognition

Image detection and recognition deals with the problem of detecting a certain element in a medical image. In many cases, the images are volumetric, therefore efficient parsing is a must. A popular strategy to do so is marginal space learning [10], as it is efficient and allows to detect organs robustly. Its deep learning counter-part [11], is even more efficient, as its probabilistic boosting trees are replaced using a neural network-based boosting cascade. Still, the entire volume has to be processed to detect anatomical structures reliably. *Bier et al*. proposed an interesting method in which they detect anatomical landmarks in 2-D X-ray projection images [12]. In their method, they train projection-invariant feature descriptors from 3-D annotated landmarks using a deep neural network. Yet another popular method for detection are the so-called region proposal convolutional neural networks. In [13], they are applying it to robustly detect tumours in mammographic images. Detection and recognition are obviously also applied in many other modalities and a great body of literature exists. Here, we only report two more applications. In histology, cell detection and classification are important tasks, which are tackled by *Aubreville et al.* using guided spatial transformer networks [14] that allow refinement of the detection before the actual classification is done.

4 Description of the datasets and cost functions

Deep learning requires massive amount of training data as classification accuracy of deep learning classifiers is largely dependent on the quality and size of the dataset. However, unavailability of annotated and homogeneous datasets is one of the biggest barriers to apply deep learning in medical imaging. At the same time, the development of large medical imaging datasets is quite challenging as annotation requires extensive time from medical experts. Furthermore, annotation may not be possible due to the unavailability of sufficient samples in the case of rare diseases. Other major issues could be related to 1) unbalanced data, what is very common in the health sector (i.e. rare diseases are underrepresented in the data sets), 2) overcome the human error in the annotation process requiring multiple expert opinion.

Today it is possible, in many cases, to avoid the abovementioned issues thanks to the availability over the Internet of large amounts of data properly labelled or annotated thanks to data repositories. A short list of medical imaging data sets and repositories that could be useful for the project is presented below.

Name	Summary	Link
OpenNeuro	An open platform for sharing neuroimaging data under the public domain license. Contains brain images from 168 studies (4,718 participants) with various imaging modalities and acquisition protocols.	https://openneuro.org Data can be downloaded from https://registry.opendata.aws/openneuro
UK Biobank	Health data from half a million participants. Contains MRI images from 15,000 participants, aiming to reach 100,000.	http://www.ukbiobank.ac.uk/
TCIA	The cancer imaging archive hosts a large archive of medical images of cancer accessible for public download. Currently contains images from 14,355 patients across 77 collections.	http://www.cancerimagingarchive.net
ABIDE	The autism brain imaging data exchange. Contains 1114 datasets from 521 individuals with Autism Spectrum Disorder and 593 controls	http://fcon1000.projects.nitrc.org/indi/abide
ADNI	The Alzheimer's disease neuroimaging initiative. Contains image data from almost 2000 participants (controls, early MCI, MCI, late MCI, AD)	http://adni.loni.usc.edu/

Other datasets available on-line are presented in the section 3.1.

However, there are also important practical issues linked to the data management that all users of deep learning need to be aware of. In particular, a look at the loss function over the training iterations is very important to identify problems related to the so-called *learning rate η setting*. If the loss

increases quickly after the beginning the learning rate η is set too high. In contrast to that, if η is too low, a stagnation of the loss over iterations is registered. Hence, correct choice of η and other training hyper-parameters is crucial for successful training [15].

In fact, to deploy this automated analysis, **data sets must be correctly partitioned** to avoid biased evaluations, sometimes considering data correlations (e.g. images acquired at the same hospital may be more similar to each other than to those from other hospitals). The data must be sampled, loaded and passed to the network, in different ways depending on the phase of the deep learning pipeline. Algorithms for tuning hyper-parameters within a family of models and optimizing model parameters on the training data are needed. Logging and visualization are needed to debug and dissect models during and after training. In applications with limited data, data sets must be augmented by perturbing the training data in realistic ways to prevent overfitting.

According to that, in addition to the training set, a validation set, as presented in Figure 4, is used to determine overfitting. In contrast to the training set, the validation set is never used to actually update the weights of the network. Hence, the loss of the validation set gives us an estimation of the error on unseen data. During optimization, the loss on the training set will continuously fall. However, as the validation set is independent, the loss on the validation set will increase at some point in training. This is typically a good point to stop updating the model before it overfits to the training data. Another common mistake is to use biased training and/or test sets. First of all, hyper-parameter tuning has to be done on validation data before actual test data is employed. In principle, test data should only be looked at once architecture, parameters, and all other factors of influence are set. Only then the test data has to be used. Otherwise, repeated testing will lead to optimistic results [15] and the system's performance will be over-estimated. This is as forbidden as including the test data in the training set. Furthermore, confounding factors may influence the classification results. If, for example, all pathological data was collected with Scanner A and all control data was collected with Scanner B, then the network may simply learn to differentiate the two scanners instead of identifying the disease [16].

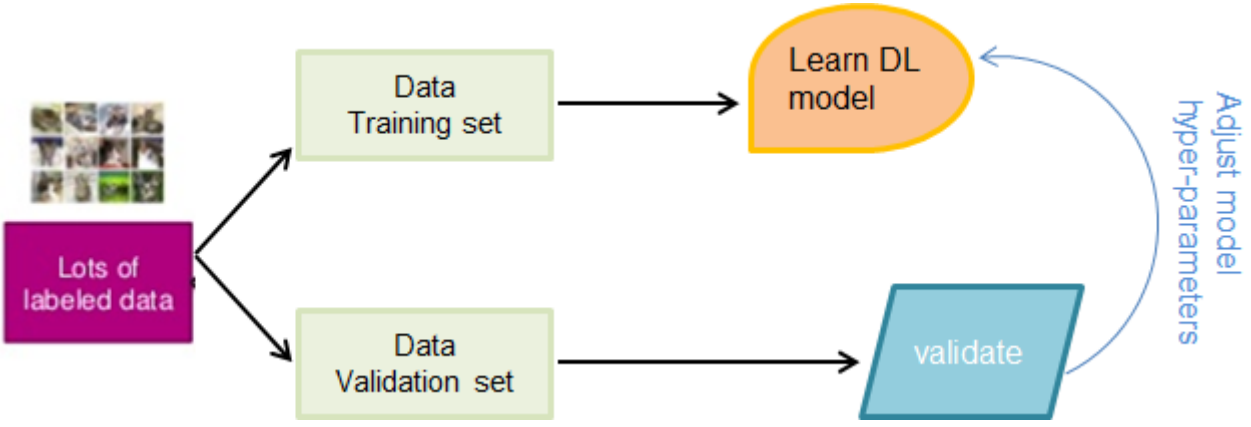

Figure 4. In the Deep Learning pipeline two datasets are used: one to train the model itself, the other, namely the data validation set, is used to fine tune the model.

As the quality of the prediction is evaluated by minimizing the loss function, the training will stop once a minimum is reached. However, due to the general non-convexity of the loss, this minimum could be only a local minimum. Hence, it is advisable to perform *multiple training runs* with different random initializations, according to one or several initialization techniques, in order to estimate a mean and a standard deviation for the model performance. Single training runs may be biased because of a particular random initialization.

5 Identified tools and libraries

In this chapter we report a short list of references and/or projects with available code for machine learning tasks applied to the medical domain.

This list of tools includes also available platform as Clara, NiftyNet and DLTK enabling fast prototyping of applications based on neural networks and ensuring reproducibility in image analysis applications, with a particular focus on medical imaging.

Most of them, as NiftyNet and Clara, allow to train models on MultiGPU servers, but none on FPGA as it will be provided by the DeepHealth toolkit.

Tool Name & Summary	Implementation/reference
NVIDIA Clara platform is a collection of Healthcare specific developer tools built on NVIDIA's compute platform aimed at accelerating data acquisition, analysis, and data integration	https://developer.nvidia.com/clara
NiftyNet. An open source convolutional neural networks platform for medical image analysis and image-guided therapy[v] [w]	http://niftynet.io
DLTK. State of the art reference implementations for deep learning on medical images[x]	https://github.com/DLTK/DLTK
DeepMedic[y]	https://github.com/Kamnitsask/deepmedic
U-Net: Convolutional Networks for Biomedical Image Segmentation[z]	https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net
V-net[aa]	https://github.com/faustomilletari/VNet
SegNet: A Deep Convolutional EncoderDecoder Architecture for Robust Semantic Pixel-Wise Labelling[bb]	https://mi.eng.cam.ac.uk/projects/segnet

[v] Gibson E, LiW, Sudre C, Fidon L, Shakir DI,Wang G, et al. NiftyNet: a deep-learning platform for medical imaging. Comput Methods Pro- grams Biomed 2018;158:113–22

[w] LiW,Wang G, Fidon L, Ourselin S, Cardoso MJ,Vercauteren T. On the compactness, efficiency, and representation of 3d convolutional net- works: brain parcellation as a pretext task. In: International conference on information processing in medical imaging (IPMI). 2017

[x] Pawlowski N, Ktena SI, Lee MC, Kainz B, Rueckert D, Glocker B, et al. DLTK: state of the art reference implementations for deep learning on medical images, arXiv:1711.06853 (2017).

[y] Kamnitsas K, Ledig C, Newcombe VF, Simpson JP, Kane AD, Menon DK, et al. Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. Med Image Anal 2017; 36:61–78.

[z] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention (MICCAI). LNCS, vol. 9351. Springer; 2015. p. 234–41, arXiv:1505.04597 [cs.CV].

[aa] Milletari F, Navab N, Ahmadi S-A. V-net: fully convolutional neural networks for volumetric medical image segmentation. In: 2016 fourth international conference on 3D Vision (3DV), IEEE. 2016. p. 565–71.

[bb] Badrinarayanan V, Kendall A, Cipolla R. Seg-Net: a deep convolu- tional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell 2017.

Brain lesion synthesis using GANs[cc].	https://github.com/khcs/brain-synthesis-lesion-segmentation
GANCS: Compressed Sensing MRI based on Deep Generative Adversarial Network[dd]	https://github.com/gongenhao/GANCS
Deep MRI Reconstruction[ee]	https://github.com/js3611/Deep-MRI-Reconstruction
Graph Convolutional Networks for brain analysis in populations, combining imaging and non-imaging data[ff]	https://github.com/parisots/population-gcn

6 DeepHealth: description of the components

6.1 Design goal

The DeepHealth toolkit includes both libraries, ECVL and EDDLL, plus the front-end exposed to expert users for an efficient usage of all the functionalities of these libraries. This toolkit will be free and open-source software. Proprietary software platforms will be adapted to use EDDLL and ECVL.

The ECV library is devoted to images pre-processing, the EDDL library provides the tools for defining computational pipelines and executing them efficiently on hybrid hardware resources. A web-based front-end completes the DeepHealth infrastructure and makes the toolkit a stand-alone solution. The design of DeepHealth toolkit follows several core principles to support a set of key requirements:

1. support a wide variety of application types in medical image analysis and computer-assisted diagnosis;
2. be simple to be used in common use cases, but flexible enough for complex use cases;
3. support parallel processing, model distribution and adaptation;
4. support best practices as data augmentation and correct data set partitioning;
5. user-friendly visualization.

Deep-learning pipelines, presented in Figure 5, for medical image analysis comprise many interconnected components such as:

- separation of data into training, testing and validation sets;
- randomized sampling during training;
- image data loading and sampling;
- data augmentation (on-the-fly if it does not add delays during training);
- a network architecture defined as the composition of many simple functions;
- a fast computational framework for optimization and inference;
- metrics for evaluating performance during training and inference.

[cc] Shin H-C, Tenenholtz N A, Rogers J K, Schwarz C G, Senjem M L, Gunter JL, et al. Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In: International work shop on simulation and synthesis in medical imaging. 2018. pp. 1–11

[dd] Mardani M, Gong E, Cheng JY, Vasanawala S, Zaharchuk G AlleyM, et al. Deep generative adversarial networks for compressed sensing automates MRI. arXiv:1706.00051 (2017).

[ee] Schlemper J, Caballero J, Hajnal JV, Price AN, Rueckert D. A deep cascade of convolutional neural networks for dynamic MR image reconstruction. IEEE Trans Med Imaging 2018; 37:491–503.

[ff] Parisot S, Ktena SI, Ferrante E, Lee M, Moreno RG, Glocker B, et al. Spectral graph convolutions for population-based disease pre- diction. In: International conference on medical image computing and computer-assisted intervention. 2017. p. 177–85.

These functionalities are described and commented in the following sections.

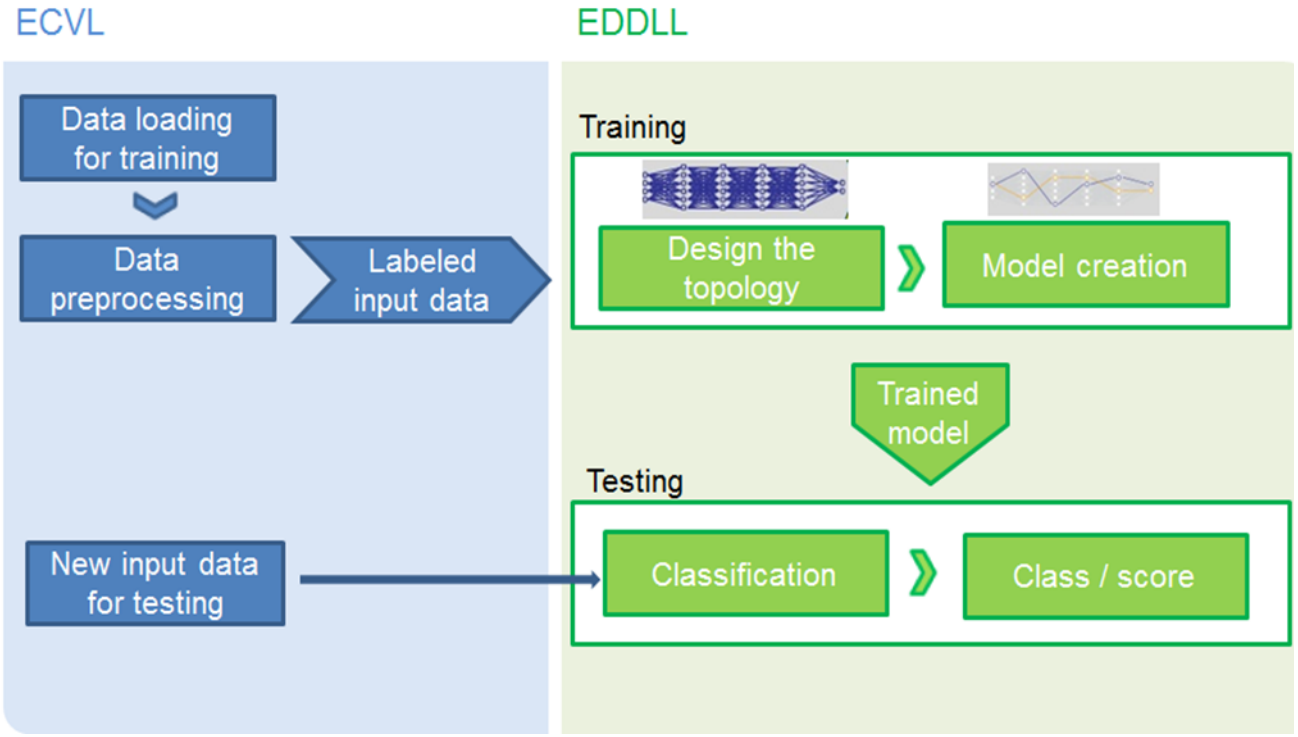

Figure 5. Scheme of the traditional pattern recognition pipeline used for automatic decision making that will also be adopted in the DeepHealth toolkit. It includes a library for data management and manipulation and a library for training and testing deep learning models, namely ECVL and EDDLL respectively. Data is pre-processed and features are extracted in both training and test phases. Trained classifiers are used in the test phase to automatically assign samples to one of the possible classes.

6.2 Component details: ECVL

6.2.1 Data format

The ECVL library supports all the common data format (e.g. jpeg, png, bpm, ppm, pgm, etc.) as well as NIfTI and DICOM, providing both reading and writing functionalities. A specific function to read/write images from/to file is provided for NIfTI and DICOM formats: NiftiRead/NiftiWrite and DicomRead/DicomWrite. For all the other supported formats a single couple of functions is provided ImRead/ImWrite. In this case the data format is deduced from the file data (read) or from the file extension (write).

ECVL-REQ-1-01	**New data upload**
ECVL-REQ-1-02	**Merge compatible data**
ECVL-REQ-1-03	**Generate data partition**
ECVL-REQ-2-01	**Data Manipulation (data normalization, adjust grey scale etc.. to be completed)**
ECVL-REQ-2-02	**Generate data augmentation**

6.2.2 Data augmentation

The data augmentation process allows to enlarge a dataset thus improving the training process of neural networks and increasing the final accuracy. The ECVL library includes all the augmentation strategies commonly exploited in literature. In the following an example list is reported.

Name	Description
Channel Shuffle	Randomly rearrange channels of the input RGB image with a probability p.
Add	Add a constant value to each channel of an input image.
AdditiveGaussianNoise	Add Gaussian noise to the image.
AdditiveLaplaceNoise	Add Laplacian noise to the image.
AdditivePoissonNoise	Add Poisson noise to the image.
ImpulseNoise	Add impulse noise to the image.
Multiply	Multiply an image by a constant value.
Dropout	Set random pixels to 0.
CoarseDropout	Set random blocks of fixed size to 0.
SaltAndPepper	Adds salt-and-pepper noise to an image, i.e. some white-ish and black-ish pixels.
Salt	Add salt noise to an image
Pepper	Add pepper noise to an image
Invert	Augmenter that inverts all values of the input image.
GaussianBlur	Blur an image using Gaussian kernels.
AverageBlur	Blur an image by computing average values over neighbourhoods.
MedianBlur	Blur an image by computing median values over neighbourhoods.
HistogramEqualization	Perform standard histogram equalization on images.
Flip	Flip an image horizontally or vertically.
GammaContrast	Adjust contrast by scaling each pixel value to $255 * (I(i,j)^{gamma})$ with $gamma \in [0.5,2.0]$.

6.2.3 Data partitioning

Data partitioning is a key step in most machine learning pipelines and this is especially relevant for training neural networks. The process of splitting data into training, validation and test set is usually performed by hand. The DeepHealth front-end will expose functionalities to achieve the goal in an automatic manner: the expert user will be only required to provide a pointer to data to be used and to tune some parameters. The split process is performed by ECVL in order to provide EDDLL the required data for training and validation steps.

Some of the parameters required for the tuning of the partitioning are the percentage of data to be used for each subset (training, validation and test), additional information related to images to ensure as much as possible the data balancing.

6.3 Component details: EDDLL

6.3.1 Image classification and detection of patterns in images

The three main goals in medical imaging are (1) classification of one or a set of images into one of several classes in order to provide doctors with an assessment of the target disease during diagnose, (2) highlighting regions of interest in order to help doctors to focus on those parts of medical images that can contain patterns (or biomarkers) that help doctors to make decisions during diagnoses, and (3) image segmentation to compute the value of some volumetric indicators or to evaluate the evolution of the size of a tumour in comparison to previous scans of the same patient. In the medium term, we expect DNNs will be used to support doctors in other goals, both for diagnose and for following the evolution of patients.

In order to tackle the three enumerated goals, it will be possible to use several topologies already used by the scientific community thanks to the EDDL library. This library will offer the possibility of importing and exporting DNNs according to the standard ONNX format [4].

In the table below are presented the main ECVL requirements.

EDDLL-REQ-1-01	**Design Topology based on CNN**
EDDLL-REQ-1-02	**Design Topology based on RNN**
EDDLL-REQ-2-01	**Import/export model based on standard format**
EDDLL-REQ-2-02	**Load pretrained model**
EDDLL-REQ-3-01	**Computing accelerator services 1: Parallel programming models**
EDDLL-REQ-3-02	**Computing accelerator services 2: Distributed programming models**
EDDLL-REQ-3-03	**Train a model**
EDDLL-REQ-3-04	**Test the model/ inference**
EDDLL-REQ-3-05	**Model validation (adjust hyperparameter, loss function)**

6.3.2 Image segmentation

The topologies supported by EDDL library for image segmentation will be the commonly used until now, mainly U-net and V-net, plus new ones as they appear. This will be possible thanks to the implementation of the EDDL library, which facilitates the design of any topology.

Segmentation topologies are based on using convolutional layers at the output of the network with sigmoid or softmax activation, depending on the task. So, in addition to the topologies available in the model zoo, the user of the DeepHealth toolkit will have the tools for creating new ones.

Like other topologies, the ones commonly used for image segmentation will be available in the model zoo of the EDDL library.

6.3.3 Available metrics

The common metric used for evaluating classifiers is the accuracy, whereas for segmentation or detecting regions of interest there are different metrics. However, we have to highlight that the loss functions used for training the DNNs are not the metrics. In the case of classifiers the loss function is the “categorical cross-entropy", and for segmentation it is used the “mean squared error”. So, the metric informed by the EDDL library will be the accuracy in the case of classifiers. Other metrics listed below will be provided by the DeepHealth toolkit, but not implemented as part of the EDDL library, it is not worthwhile because they are not used by the training algorithms.

The following table lists and describes some of the most commonly used metrics, but before describing these metrics, it is important to describe the used acronyms:

TP = True Positives	FP = False Positives
FN = False Negatives	TN = True Negatives

Any of these four variables can be computed for a specific class, so that we will use TP(c) for denoting the number of true positives for class c, that is, the number of samples of class c correctly classified. The same reasoning applies for FP(c), FN(c) and TN(c).

Metric	**Description or formula**
Accuracy	(TP+TN) / (TP+TN+FP+FN) [gg]
Precision	TP / (TP+FP) [gg] Portion of samples correctly assigned to the target class out of the samples the system assigns to the target class
Recall	TP / (TP+FN) [gg] Portion of the samples correctly detected out of all the samples of the target class
F1-Score	(2·TP)/(2·TP+FP+FN) [gg] Harmonic average of precision and recall, useful for comparing the performance of different systems/models by using one criterion
Area under the ROC curve (AUC)	The area under the ROC curve is a metric for evaluating binary classifiers. The ROC curve created by moving the discrimination threshold. [hh]

[gg] https://en.wikipedia.org/wiki/Precision_and_recall

[hh] https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Intersection over Union (IoU) [18]	Area of Overlap / Area of Union
Jaccard Index for class k [18]	$JI_k = C_{kk} / (Sum(i,1,n)[C_{ik}] + Sum(j,1,n)[C_{kj}] - C_{kk})$ where C_{ij} is the count of pixels of class *i* classified into class *j*
Boundary Jaccard for class c [18]	BJ(c) = TP(c) / (TP(c) + FP(c) + FN(c))
Pixel Accuracy (in average) [20]	$PA = 1/(N \cdot C) \cdot Sum(n,1,N)[Sum(c,1,C) [p(c)/P(c)]]$ where *N* is the total number of pixels, *C* is the number of classes, p(*c*) is the number pixels correctly classified in the class *c*, and P(*c*) is the total number of pixels of the class *c*
Dice coefficient for class c [19]	$D=(2 \cdot TP(c)) / (2 \cdot TP(c)+FP(c)+FN(c))$ -- Similar to F1-Score for class *c*

The metrics **accuracy**, **precision**, **recall** and **F1-score** are also used for evaluating the performance of models in segmentation tasks at the pixel level. Where TP is the count of individual pixels correctly assigned as belonging to a region of interest or segment of a type of tissue, etc. Analogously, TN, FP and FN are also defined at the pixel level. For instance, if the task consists in segmenting an image for identifying more than two types of tissues or organs, then they will be used TP(c), TN(c), FP(c) and FN(c), as explained before the previous table.

6.3.4 Models load & export

The EDDL library will be able to import and export DNNs according to the standard ONNX format [17]. This functionality will allow to use the EDDL library for training deep learning models for later use of such models in other software platforms or end-user applications that do not integrate the library. By the other hand, the EDDL library will be able to import DNNs already trained or with a proper initialization of weights in order to start the training from an advanced point instead of starting from scratch by setting a random initialization of the weights.

Finally, thanks to this import/export functionality, the EDDL library will offer to expert-users a model zoo, so that they can load any model widely used for a specific task.

6.3.5 Computing Services: Hybrid & distributed neural network training

The *DeepHealth computing infrastructure* (see Deliverable *D1.2. HPC infrastructure and application adaptation requirements* for further information) incorporates an application programming interface (API) exposed to the ECVL and EDDLL developers, to efficiently exploit the parallel and heterogeneous HPC and cloud-based capabilities of the underlying computing resources. Concretely, DeepHealth will consider the following functionalities exposed to the developers:

- *Parallel programming models.* DeepHealth will evaluate two types of programming models: those specialised on heterogeneous and distributed memory execution model featuring acceleration devices such as GPUs, many-core fabrics or SoC-FPGAs (e.g., CUDA, OpenCL); and those supporting homogeneous and heterogeneous execution models (e.g., OpenMP). The use of the proper programming models is of paramount importance to exploit the performance of acceleration devices.
- *Distributed programming models.* DeepHealth will consider different distributed computing programming models supporting Map Reduce (M/R), RDD and tasking programming models.
- *Non-Functional Requirements Description.* The DeepHealth infrastructure will incorporate a multi-objective heterogeneity-aware workload allocation policies to enable the allocation of different training requests to the most adequate computing resources.

- *Cloud-based API.* The infrastructure will support the distribution of the computation on cloud-based environments, based on the OpenStack IaaS.

6.4 Component details: Front-end

The DeepHealth toolkit includes a web-based graphical user interface that will allow the user to access, in a graphic way, to all functionalities provided by the libraries and will provide the graphical tools that will help him to utilise these functionalities in a user-friendly way.

The web based interface implies several benefits for the users:

- They do not have to manage individual software installs and updates whenever there's a change to the software.
- Similarly, they would no longer need to verify that the software will work on various combinations of hardware and operating systems. Building a web browser interface means that any customer on any combination of hardware and operating system can access and maximize their use of the toolkit. Implement a web browser UI means that the user will be able to utilize the Deephealth solution in diverse circumstances.
- Expert-users 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.

The graphical user interface will be intuitive, attractive, and easy to use and to learn. The front-end will include the following features that are organized in 3 main groups:

- Datasets management. It provides the user with the following operations:
  - Loading datasets in the system. The application will have the loading features for image file formats as DICOM and NIFTY, and other data formats as CSV file, XLS files
  - Selecting datasets
  - Loading CSV files with structured data that accompany images
  - Merging compatible datasets, based on generated indexes, for using images from two or more datasets
  - Define data augmentations to apply on-the-fly while mini-batches are prepared to train DNNs
- Pipelines and algorithms management:
  - Defining pipelines of data/image transformations
  - Running the defined pipelines of data/image transformations
  - Defining a pipeline of data/image manipulations for extracting aggregated values
  - Establishing the resources for running a training process, when not all the available resources are necessary for a concrete task
  - Loading pre-trained models from the pool of available topologies
  - Adjusting and visualizing hyper-parameters of the chosen models. The user will be able to modify the parameters, to calibrate the chosen model, using an intuitive graphical interface
  - Running the defined processes (transformations, extractions of aggregated values, training, testing, etc.)
  - Import/export DNNs in the ONNX format
- Results:
  - Display results:
    - Displaying in a graphical way the evolution of training processes. The user will be able to choose from some metrics (loss, accuracy or others), so he can see the evolution on the training and development subsets, based on these metrics
    - Graphical representation of results obtained with different models with the same dataset and partition of such dataset

        - Graphical representation of results of different models obtained with the dataset (or partition of the dataset). This will allow the user to make a comparative study
    - Export results
        - Numerically export results

Here are summarized the requirements the front-end must provide:

FE-ECVL-REQ-1-01	**data loading (DICOM, NIFTI, and jpeg, png, bpm, ppm, pgm)**
FE-ECVL-REQ-1-02	**Select datasets for working (already existent in the toolkit or new data sets)**
FE- ECVL- REQ-1-03	**Merge compatible datasets (basically by generating an index for using images from two or more datasets)**
FE- ECVL- REQ-2-01	**Define a pipeline of data/image transformations – and run it**
FE- ECVL- REQ-2-02	**Define a pipeline of data/image manipulations for extracting aggregated values**
FE- EDDLL- REQ-3-01	**Generate data partitions into train/development/test subsets**
FE - EDDLL- REQ-3-02	**Generate partitions[ii] of the training subset for distributing the workload over the available resources**
FE – EDDLL- REQ-3-03	**Choose the resources for running a training process, when no all the available resources are necessary for a concrete task**
FE - EDDLL- REQ-3-04	**Load pre-trained models from the pool of available topologies**
FE - EDDLL- REQ-3-05	**Visualize hyperparameters and loss functions**
FE-EDDLL-REQ-3-06	**Import/export DNNs**
FE-EDDLL-REQ-4-01	**Graphical representation of results**
FE-EDDLL-REQ-4-02	**Result export**

[ii] Let us clarify that a dataset can be partitioned at two levels, the highest one is for testing models and getting results, this high-level partition is usually done into three subsets: training, development and test. They could be necessary to combine different criteria to get a good partition into train/dev/test. The low-level partition is just for distributing the workload.

7 Conclusions

We start this deliverable by presenting the typical pipeline that a toolkit dedicated to design, train and evaluate predictive models in the health sector has to cover. This analysis allows to select the main requirements and the capabilities provided by the open-source DeepHealth toolkit devoted to easily introduce deep learning in medical imaging. This requirement analysis drives the project to provide a modular implementation of the toolkit, merging the functionalities described in this deliverable in two main libraries called ECVL and EDDLL respectively. Thanks to the combination of these libraries with a web based front-end we will offer to expert users or reseacher in hospitals the opportunity to reduce their workflows improving the quality of the diagnosis and, at the same time, answering to the challenges associated to the introduction of deep learning in clinical settings. Deep learning methods today produce results that are too valuable to be discarded in the medical sector.

The DeepHealth toolkit provides implementations for data loading, data augmentation, implementation of new network architectures, use existing or pretrained networks, compute loss functions and evaluate metrics that are tailored for medical image analysis and computer-assisted diagnosis. This infrastructure enables researchers to rapidly develop deep learning solutions for segmentation, classification, comparison of different models, or simply extending its use to new industrial applications.

8 References

[1] Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408. DOI:10.1037/h0042519

[2] Ian Goodfellow, Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org

[3] From https://fr.wikipedia.org/wiki/Fichier:MultiLayerNeuralNetworkBigger_english.png

[4] Hinton, G. E. Reducing the Dimensionality of Data with Neural Networks. Science (80-.). 2006, 313 (5786), 504–507.

[5] Pereira S, Pinto A, Alves V, Silva CA. Brain Tumor segmenta-tion using convolutional neural networks in MRI images. IEEETrans Med Imaging 2016;35:1240–51.

[6] Havaei M, Davy A, Warde-Farley D, Biard A, Courville A,Bengio Y, et al. Brain tumor segmentation with deep neuralnetworks. Med Image Anal 2017;35:18–31

[7] Poudel RP, Lamata P, Montana G. Recurrent fully convolutional neural networks for multi-slice MRI cardiac segmentation. In: Reconstruc- tion, segmentation, and analysis of medical images. Springer; 2016. p. 83–94.

[8] Andermatt S, Pezold S, Cattin P. Multi-dimensional gated recurrent units for the segmentation of biomedical 3D-data. In: Deep learning and data labeling for medical applications. Springer; 2016. p. 142–51.

[9] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Com- put 1997;9:1735–80.

[10] Zheng Y, Comaniciu D. Marginal space learning. In: Marginal space learning for medical image analysis. Springer; 2014. p. 25–65.

[11] Ghesu FC, Krubasik E, Georgescu B, Singh V, Zheng Y, Hornegger J, et al. Marginal space deep learning: efficient architecture for volumet- ric image parsing. IEEE Trans Med Imaging 2016;35:1217–28.

[12] Bier B, Unberath M, Zaech J-N, Fotouhi J, Armand M, Osgood G, et al. X-ray-transform invariant anatomical landmark detection for pelvic trauma surgery. In: Frangi AF, Schnabel JA, Davatzikos C, Alberola-López C, Fichtinger G, editors. Medical Image Computing and Computer Assisted Intervention – MICCAI 2018. Cham: Springer International Publishing; 2018. p. 55–63.

[13] Akselrod-Ballin A, Karlinsky L, Alpert S, Hasoul S, Ben-Ari R, Barkan E. A region based convolutional network for tumour detection and classification in breast mammography. In: Deep learning and data labelling for medical applications. Springer; 2016. p. 197–205.

[14] Aubreville M, Krappmann M, Bertram C, Klopfleisch R, Maier A. A guided spatial transformer network for histology cell differenti- ation. In: Association TE, editor. Eurographics workshop on visual computing for biology and medicine. 2017. p. 21–5

[15] Goodfellow I, Bengio Y, Courville A, Bengio Y. Deep learning, vol. 1. Cambridge: MIT Press; 2016.

[16] Maier A, Schuster M, Eysholdt U, Haderlein T, Cincarek T, Steidl S, et al. QMOS – a robust visualization method for speaker dependencies with different microphones. J Pattern Recognition Res 2009;4: 32–51.

[17] https://onnx.ai/ (accessed 2019-07-15)

[18] Eduardo Fernandez-Moral, Renato Martins, Denis Wolf, Patrick Rives. A new metric for evaluating semantic segmentation: leveraging global and contour accuracy. Workshop on Planning, Perception and Navigation for Intelligent Vehicles, PPNIV17, Sep 2017, Vancouver, Canada. hal-01581525

[19] Zou, Kelly H. et al. Statistical validation of image segmentation quality based on a spatial overlap index. Academic Radiology, Volume 11, Issue 2, February 2004, pp. 178-189, DOI: 10.1016/S1076-6332(03)00671-8

[20] Fulkerson, Brian et al. Class segmentation and object localization with superpixel neighborhoods. 2009 IEEE 12th International Conference on Computer Vision, 29 Sept.-2 Oct. 2009, DOI: 10.1109/ICCV.2009.5459175

[21] LeCunY,BottouL,BengioY,HaffnerP.Gradient-basedlearning applied to document recognition. Proc IEEE 1998; 86:2278–324.