Received: 21 June 2019 | Revised: 20 February 2020 | Accepted: 14 June 2020

DOI: 10.1002/spe.2878

RESEARCH ARTICLE

WILEY

# R2D2: A scalable deep learning toolkit for medical imaging segmentation

**Soulaimane Guedria**[1,2] | **Noël De Palma**[1] | **Félix Renard**[1,2] | **Nicolas Vuillerme**[2,3]

[1]Université Grenoble Alpes, CNRS, Grenoble INP, LIG, Grenoble, France

[2]Université Grenoble Alpes, AGEIS, Grenoble, France

[3]Institut Universitaire de France, Paris, France

**Correspondence**
Soulaimane Guedria, Université Grenoble Alpes, CNRS, Grenoble INP, LIG, Bâtiment IMAG, 700 Avenue Centrale, Grenoble, Saint-Martin-d'Hères 38400, France.
Email: soulaimane.guedria@univ-grenoble-alpes.fr

**Funding information**
The French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-10-AIRT-05), the Virtual Studio RA and Smart Support Center (SSC) FEDER EU projects, the FSN Hydda project and Institut Carnot LSI

**Summary**

Deep learning has gained a significant popularity in recent years thanks to its tremendous success across a wide range of relevant fields of applications, including medical image analysis domain in particular. Although convolutional neural networks (CNNs) based medical applications have been providing powerful solutions and revolutionizing medicine, efficiently training of CNNs models is a tedious and challenging task. It is a computationally intensive process taking long time and rare system resources, which represents a significant hindrance to scientific research progress. In order to address this challenge, we propose in this article, R2D2, a scalable intuitive deep learning toolkit for medical imaging semantic segmentation. To the best of our knowledge, the present work is the first that aims to tackle this issue by offering a novel distributed versions of two well-known and widely used CNN segmentation architectures [ie, fully convolutional network (FCN) and U-Net]. We introduce the design and the core building blocks of R2D2. We further present and analyze its experimental evaluation results on two different concrete medical imaging segmentation use cases. R2D2 achieves up to 17.5× and 10.4× speedup than single-node based training of U-Net and FCN, respectively, with a negligible, though still unexpected segmentation accuracy loss. R2D2 offers not only an empirical evidence and investigates in-depth the latest published works but also it facilitates and significantly reduces the effort required by researchers to quickly prototype and easily discover cutting-edge CNN configurations and architectures.

**KEYWORDS**

deep learning, distributed optimization, distributed systems, high-performance computing, medical imaging, semantic segmentation, software engineering

## 1 | INTRODUCTION

Semantic segmentation of medical imaging consists in detecting and contouring boundaries of regions of interest in medical images such as lesions, anatomical structures, or any other meaningful morphological structures.[1] It plays a fundamental role in computer aided diagnosis,[2,3] clinical studies, and medical treatment planning.[4] However, manual medical image segmentation is not only a tedious, extensive and time consuming task, but also it has to be performed by medical experts. Recent advances in deep neural networks[5] (DNNs) and particularly convolutional neural networks[5] (CNNs) come to address this issue. In fact, CNN-based applications have been shown to be powerful tools to successfully

tackle most common medical images challenges[4,6] and in particular medical semantic segmentation tasks.[7,8] However, building efficient CNN models requires an effective and tedious training process. In order to do so, multiple CNN architectures have to be investigated. Concurrently, an hyperparameters optimization process[9] has to be performed for every CNN candidate architecture. The hyperparameters optimization task aims to select the optimal set of hyperparameters in order to optimize the CNN performance. It involves performing various hyperparameters optimization strategies[10] which generally require executing multiple training runs. Hence, training CNN models is computationally intensive and time consuming process.[11] For instance, training DeepMedic[12] brain tumor segmentation CNN architecture with a particular set of hyperparameters requires approximately a day using a single NVIDIA GTX Titan X GPU. Therefore, decreasing the training duration of DNNs is crucial to accelerate hyperparameters optimization process. Moreover, it enables researchers to not only build effective CNNs, but also prototype and explore not yet investigated CNN configurations and architectures through an iterative and adaptive experimentation approach.

In order to address this challenge, we propose and evaluate R2D2 (rapid and robust digital diagnostic) a research-dedicated scalable deep learning toolkit (DLTK) for medical imaging segmentation. Our proposed toolkit introduces (i) a couple of an innovative ready-to-use distributed versions of two popular CNN segmentation architectures [fully convolutional network (FCN)[13] and U-Net[14] alongside with (ii) a high-level end-to-end deep learning medical imaging processing pipeline. The latter aims to reduce the learning curve and overcome talent-intensive deep learning technology adoption barriers for nonspecialists. For security, accessibility and usability reasons, R2D2 can be driven through a software as a service user-friendly web interface in addition to a command line interface (CLI) for more expert users. Furthermore, R2D2 integrates a couple of real-time visualization components in order to track both (i) system resources and (ii) CNN training metrics evolution during the distributed training process. They offer an extensive overview for a better understanding and easier debugging of the CNN distributed training task progress. We achieve up to 97% and 58% scaling efficiency for U-Net and FCN CNNs, respectively, when moving from 1 to 18 Nvidia GTX 1080 Ti GPUs without significant, yet still mysterious segmentation accuracy degradation. Furthermore, our work constitutes a deep empirical investigation for the latest published articles[15,16] and confirms state-of-the-art results of related works.[17-19]

In addition, in order to prove that R2D2 generalizes for a wider variety of datasets and tasks, we assess our proposed scalable CNN architectures on two practical medical imaging segmentation use cases. The first one is a brain tumor segmentation challenge, and the second use case is a cardiac left atrium segmentation task. This extensive evaluation provides an in-depth assessments and comparison of the performances of two popular CNN architectures on a couple of challenging case studies.

The remainder of the article is structured as follows: In Section 2, we provide a brief overview of background information on distributed training of DNNs and explore some related work. In Section 3, we present R2D2 and its design, building blocks and architecture. In Section 4, we evaluate our proposed solution, based on two different concrete medical imaging segmentation case studies. We finally conclude and give insights about future research directions in Section 5.

## 2 | BACKGROUND AND RELATED WORK

The following section will give a brief overview on CNNs concepts and architectures we deal with in our proposed toolkit. A special interest will be shown also to the most common CNN distributed training approaches and particularly to the muti-GPUs data parallelism CNNs strategy. We expose, at the end, some related work and tools.

### 2.1 | CNNs for semantic segmentation

Throughout this article, the term "CNN architecture" refers to the global structure of the neural network (ie, the number, order, size, and so on, of each network's layer). In addition, the term "CNN model" denotes the output of the training process of a specific CNN architecture on a particular training dataset and hyperparameters.

CNNs are inspired by nature, in particular, visual cortex structure.[20,21] They are a powerful multilayer neural networks designed to deal with images. Unlike fully connected networks where every neuron is connected to all its predecessors,[22] each neuron in CNNs is connected to a local subdivision of the neurons in the underlying layer.[23-25] CNNs are easier to train because they have much less parameters to tweak than fully connected networks.[26] Another advantage of CNNs is their ability to introduce some degree of shift, scale and distortion invariance[22,27] to the learning process. CNNs are having

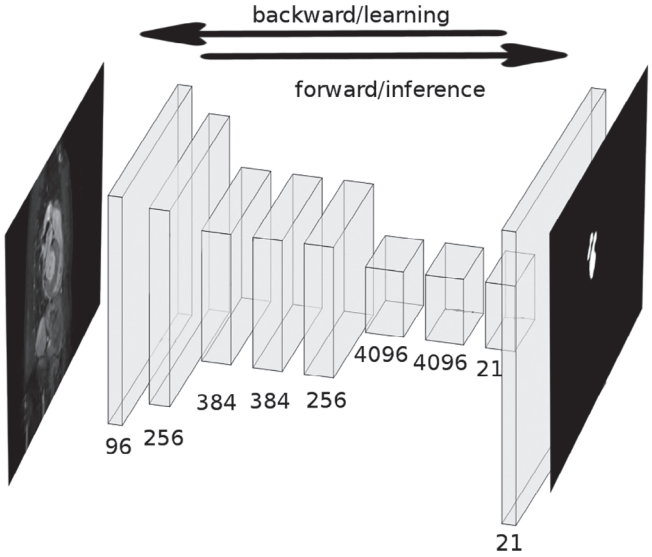



**FIGURE 1** Fully convolutional network for semantic segmentation architecture[1]

a great success in various medical imaging use cases[28-30] including the semantic segmentation task. The latter consists in a pixel level classification of images (ie, by assigning a label to every pixel in every image, we can split input images into semantically meaningful regions[1]).

### 2.1.1 | Fully convolutional network

The FCN[13] is the first popular[1] CNN semantic segmentation architecture we deal with in R2D2. As can be seen in Figure 1, it is a CNN where the last typical fully connected layer is replaced by an additional convolutional layer which makes the network able to deal with arbitrary-sized input images.[1] Given that the input image dimensions get smaller when we get deeper in the network due to convolution operations. The FCN uses the transposed convolution[31] technique during the upsampling step so that the output dimensions match the original input image dimensions. However, this technique causes a loss of spatial information. That is why the FCN uses skip connections[31] to reduce the information loss during convolution operations.[1,32]

### 2.1.2 | U-Net

Another widely used[2] CNN architecture for semantic segmentation tasks is the U-Net architecture.[14] It is the second CNN architecture we introduce its distributed version in our proposed solution. In Figure 2, each blue box denotes a multi-channel feature map. The number of channels is represented on top of the box. White boxes denote copied feature map and the arrows illustrate the diverse operations.[14] U-Net is an encode-decoder style architecture. The encoder consists of a sequence of convolution, max pooling and ReLU activation layers which reduce the spatial dimensions of the input volume. On the other hand, the decoder gradually restores the initial input spatial dimensions through transposed convolution operation.[31] The U-Net architecture might be used in various tasks but it is initially designed and mainly used for biomedical image segmentation.[14]

## 2.2 | Distributed training of DNNs parallelism approaches

Distributed training approaches of DNNs are mainly divided into three different categories: model, data, and hybrid parallelism techniques.[33]

[1] FCN had accumulated more than 9724 citations by the end of June 2019.
[2] 6228 citations for U-Net by the end of June 2019.

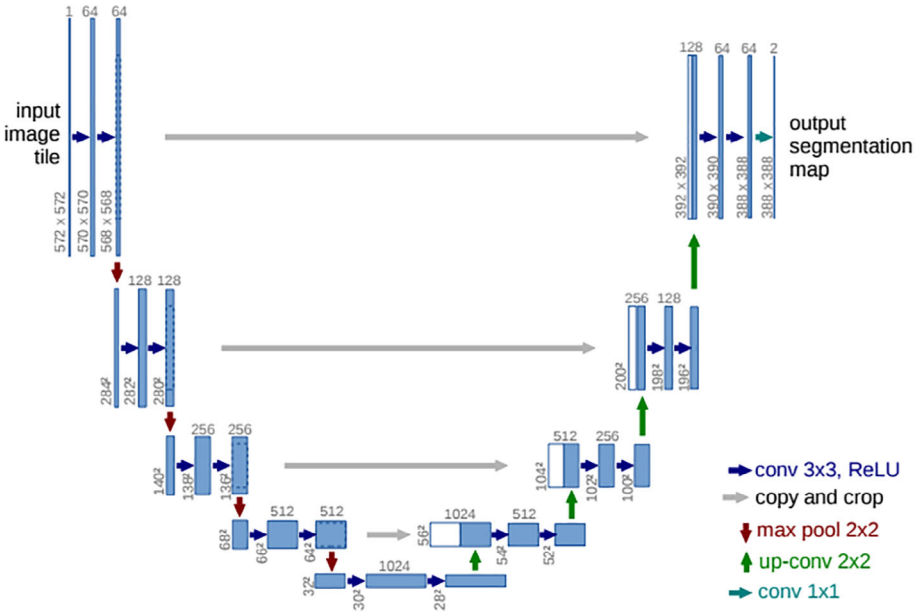



**FIGURE 2** U-Net network architecture[14] [Color figure can be viewed at wileyonlinelibrary.com]

### 2.2.1 | Model parallelism

Some DNNs models have a considerable size, and hence, they are not adapted to the memory size of an individual training device (one GPU, for instance).[34] These models require to be partitioned across all the nodes in the distributed system and every node trains a different part of the model on the whole training dataset. In Figure 3, the blue rectangle stands for a training node, the light blue circle represents a subset of a DNN model (eg, a DNN layer) and the arrow represents a connection between two successive subsets of the same model. As can be seen, every node performs the training of only a specific subset of the model. This parallelization schema is known as *model parallelism* technique.[35-37]

### 2.2.2 | Data parallelism

The second distributed training strategy of DNNs is called *data parallelism* approach. As shown in Figure 4, all nodes in the distributed system have the same **complete** copy of the model. However, the training is done independently on each node using a different subset of the whole training dataset, at the end of every training iteration, the results of computations from all the nodes are combined using different synchronization approaches.[35-37]

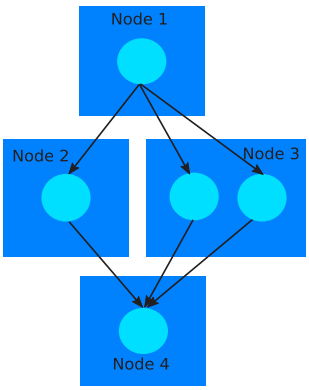



**FIGURE 3** Model parallelism [Color figure can be viewed at wileyonlinelibrary.com]

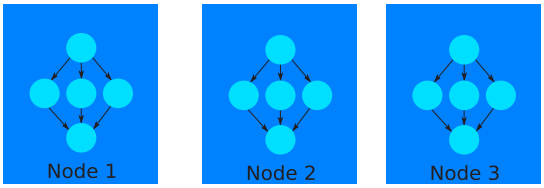



**FIGURE 4** Data parallelism [Color figure can be viewed at wileyonlinelibrary.com]

### 2.2.3 | Hybrid parallelism

It is possible to combine both previously mentioned distributed training approaches (ie, model parallelism for every node and data parallelism across nodes[37]). However, data parallelism has become the most popular distributed training approach for the following reasons :

- The practical simplicity to introduce data parallelism[37] compared with the model parallelism strategy when the model size fits in the training device's memory.
- The recent success achieved by the data parallelism method[15,16] to considerably increase the minibatch size without significant segmentation accuracy loss.

### 2.2.4 | Other parallelism approaches

In recent years, the largest amount of research done to scale up the training of DNNs evolves around methods that aim to parallelize the computation of the gradient during the gradient descent[27] optimization method. Hence, distributed training of DNNs approaches can be also classified depending on the model consistency synchronization strategies[37] (eg, synchronous,[38] stale-synchronous,[39] and asynchronous[36] techniques) and the parameter distribution and communication centralization methods[37] [eg, parameter server (PS),[40] Shared PS,[41] and decentralized strategies[42].

## 2.3 | Related work

A special effort has been made since a long time to develop medical imaging processing tools.[43] They can be classified according to the extent, scope, and nature of their application areas. Some generic medical imaging solutions have been around for a while (eg, MITK,[44] VTK, and ITK[45]). They propose a comprehensive set of common medical imaging tasks (eg, registration,[46] segmentation, visualization, and reconstruction[47]). Other generic medical imaging tools yet pathology specialized solutions have been introduced. For instance, FMRIB Software Library[48] and Freesurfer software suite[48] are two popular medical image analysis tools specialized in neuroimaging. Finally, a suite of task specific solutions have been proposed (eg, NiftySeg[49] for segmentation, NiftySim[50] for simulation, and Camino[51] for Diffusion).

The previously mentioned medical imaging tools are neither DNN-based solutions, nor distributed applications. However, the recent deep learning breakthroughs led to the emergence of a new set of DNN-based medical image analysis tools. For instance, NiflyNet[52] is an open source deep-learning-based platform of medical imaging which is built on top of *TensorFlow* library. It provides a modular medical imaging processing pipeline alongside with a set of established pretrained domain specific models. The DLTK[53] is another open source *TensorFlow*-based medical imaging toolkit implementing baseline versions classic network architectures. DeepInfer[54] is an additional DLTK for image-guided therapy with a focus on the deployment and the reuse of pretrained models.

Other related medical image analysis tools exist.[55-57] Furthermore, although medical imaging DNN-based solutions built on top of *Tensorflow* (eg, NiftyNet and DLTK) natively support the standard built-in *Tensorflow* parallelization approach, they don't come up with an all set, ready-to-use distributed versions of CNN architectures (ie, they require a large amount of talent-extensive code modification[58]). Moreover, even if some of them present some similarities with R2D2 (ie, NiftyNet deep learning medical imaging pipeline, DeepInfer, and DLTK proposed pretrained models), to the best of our knowledge, no existing medical imaging solution offers all the features of R2D2, in particular :

1. The novel and ready-to-use distributed versions of the immensely popular FCN and U-Net CNN segmentation architectures which we denominate `Dist-FCN` and `Dist-U-Net`, respectively.
2. The integrated monitoring platform for system resources supervision and visualization which offers a deeper insights on, not yet investigated, system resources evolution patterns during the distributed training of CNNs. The real-time monitoring platform also allows the user to early stop the CNNs training in the case of the divergence of the latter's training process. Thus, the early stopping in such situations will avoid the waste of resources and energy.
3. The high-level end-to-end deep learning medical imaging segmentation processing pipeline.

The above-mentioned novel features integrated in our proposed toolkit are the main contributions of our work. That is why they make R2D2 stand out from the rest of existing solutions in medical imaging deep-learning-based solutions.

## 3 | R2D2 SYSTEM DESCRIPTION

This section introduces R2D2 our proposed scalable toolkit. First, we provide a global overview on R2D2 main scope, features, design and system architecture, before diving into the details of its building blocks and core components.

### 3.1 | Scope and architecture

R2D2 toolkit brings the power of distributed systems to use in medical imaging deep-learning-based applications, while considering software engineering best practices. Figure 5 depicts the overall scope into which R2D2 operates.

Our proposed toolkit follows an extensible and modular design. As shown in Figure 6 which provides a high-level overview on our distributed toolkit architecture, R2D2 offers an end-to-end support for a typical deep learning workflow in medical imaging by introducing a high-level of abstraction for common components in a regular medical CNNs processing pipeline.

The end-user might interact with the toolkit through different front-ends. He can either use an intuitive web-based graphical user interface or a CLI. The toolkit user has a set of tools at his disposal which are as follows.

- The **R2D2 Engine** is the entry point for the R2D2 toolkit. It is the main system controller operating as a key interface between the toolkit user and the available modules.

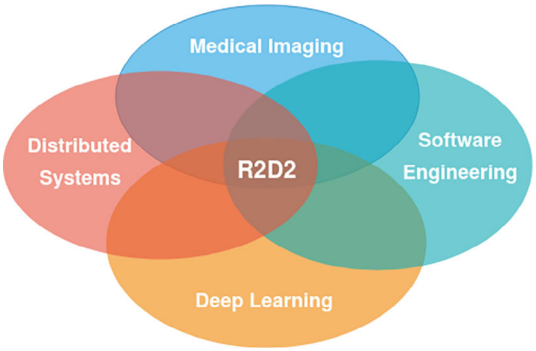



**FIGURE 5** R2D2 scope [Color figure can be viewed at wileyonlinelibrary.com]

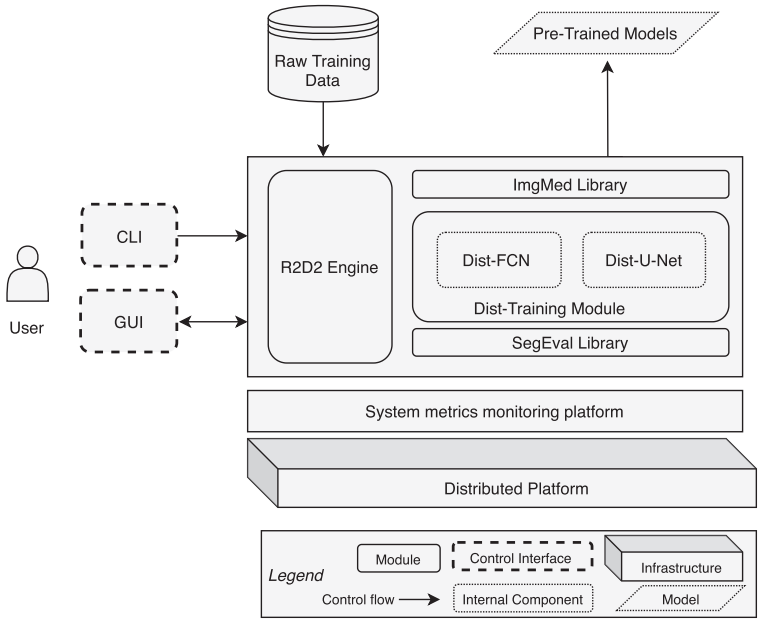



**FIGURE 6** System architecture

- The **ImgMed Library** is a high-level medical imaging preprocessing library which offers a typical medical imaging preprocessing operations.
- The **Dist-Training Module** is the core component of R2D2. It contains `Dist-FCN` and `Dist-U-Net`, a novel distributed versions of widely adopted FCN and U-Net CNN segmentation architectures, respectively.
- The **SegEval Library** is an evaluation library which proposes implementations for a collection of common empirical evaluation methods[59] for semantic segmentation.

R2D2 includes also a set of pretrained, pathology specific CNN models for renal cortex, liver and brain lesion segmentations. These pretrained models constitute a model zoo and they might be used to leverage transfer learning approach while building new CNN models. The transfer learning strategy[60] consists in reusing an already pretrained models as a starting point for the training process of new CNN models in order to, potentially, accelerate model training while improving its performance. Furthermore, it is also possible for the R2D2 user to publish his newly trained models and enrich the pretrained models collection. Finally, the user can use a web-based graphical interface for a real-time monitoring of the system resources during the distributed training phase. Concurrently, the toolkit user can also visualize the CNN training metrics evolution.

## 3.2 | Core building blocks

### 3.2.1 | ImgMed: Medical imaging preprocessing library

We built `ImgMed` which is a new library dedicated to medical imaging preprocessing workflow. In fact, the data preprocessing phase is an important and key step in machine learning development pipeline. It is a challenging task because it not only conditions the effectiveness of the developed model, but data preprocessing is also a time consuming task as it represents up to 70% of the whole project time.[61] The `ImgMed` library intent to tackle this challenge by proposing an all set high-level medical imaging preprocessing solution.

`ImgMed` includes, but is not limited to, the following typical medical imaging preprocessing operations:

- Image format conversion (eg, from JPG to PNG)
- Image reshaping (eg, from 3D to 2D)
- Image resizing
- Data augmentation

We have chosen *Python* as a programming language for `ImgMed` reference implementation for its simplicity. The `ImgMed` library is built upon matplotlib,[62] *OpenCV*,[63] and *SciPy*[64] libraries.

An example of an end-to-end medical imaging preprocessing workflow implemented using `ImgMed`high-level library is shown in Listing 1. We consider a set of N Niftti files as a raw input dataset. The first step is to reshape the 4D input files to an RJB JPG images. Next, another image reshaping procedure is performed, followed by a format conversion operation

```
from R2D2.img import ImgMed

def preprocessing_workflow(src_0, dist_f):
    """
    Input: N 4D NIfTII files in src_0 path
    Output:N 2D PNG 350*350 images in dest_f path
    """
    ImgMed.convert_4DNI_to_RGBJPG(src_0, dest_0,dim)
    ImgMed.reshape_JPG_to2D(dest_0, dest_1)
    ImgMed.convert_JPG_to_PNG(dest_1, dest_2):
    ImgMed.resize_img(dest_2,dest_f,'png',350,350)
```

**Listing 1** Typical medical imaging preprocessing workflow

from JPG to PNG. The final step of the preprocessing workflow example involves resizing the PNG images to $350 \times 350$ pixels (note that the sources and destinations file paths can be tuned according to the user needs).

As it can be noticed, the example in Listing 1 not only shows the easiness with which it is possible to implement a complete and classic preprocessing pipeline with only a few lines of code, but it also highlights the considerable impact of `ImgMed` in reducing duplication of effort during the preprocessing step of medical imaging pipeline.

### 3.2.2 | Dist-Training module

The `Dist-Training` module is the core component of R2D2 toolkit. It provides a set of scalable CNN segmentation architectures (`Dist-FCN` and `Dist-U-Net`). Yet, above all, in order to introduce parallelism to FCN and U-Net CNN architectures, a number of choices have to be made among the various distributed training strategies already introduced in Section 3. Our selection criteria of the considered parallelism method are threefold: (i) The distributed method ***model accuracy preservation***, (ii) while taking into account its ***network bandwidth optimality*** (iii) and without forgetting to consider the ***burden of its practical implementation***.

In the first place, we decided to adopt the data parallelism approach for the following reasons :

- In model parallelism, the workload of the partitioning task of a model across multiple nodes is left to the programmer,[37,65,66] that makes the effective implementation of model parallelism method a challenging task unlike the data parallelism one. For this reason, the model parallelism schema is mainly considered as a final alternative approach when a model size does not fit in a single-node's memory.[36,66]
- Since the model parallelism approach involves partitioning the model across several training agents, and given that the DNN architectures naturally generate a layer interdependencies,[37] the synchronization of computations during the training phase in model parallelism strategy creates a communication overhead which increases the training runtime.[36,37]
- Considering that the level of scalability of the data parallelism method is naturally determined by the minibatch hyper-parameter size,[37] and since recent published works[15,67] have succeeded to considerably increase the minibatch size without significant segmentation accuracy loss, data parallelism has become the most preferred distributed training approach.

Then, we chose to scale up the training of FCN and U-Net architectures using a synchronous parallelism approach. Our selection criterion for the latter chosen strategy was the trade-off between the CNN model accuracy and the training speedup. In fact, synchronous methods achieve better results regarding the accuracy of the CNN models compared with the asynchronous approaches,[37,66] particularly, with a short synchronization period.[68]

The main steps in the selected synchronous distributed data parallelism strategy are as follows:[58] (i) compute the model updates (gradients of the loss function) using a minibatch on each training agent (ii) compute the average of gradients of all training agents (iii) update the model. Hence, we have to select the parameters updates communication and propagation schema. Even if both centralized and decentralized parameters updates communication approaches have advantages and drawbacks, we decided to go for a decentralized Ring-Allreduce[58,69] algorithm for the following reasons.

- Since the network bandwidth is classified among the rarest resources in datacenters,[70] and even if the centralized PS is one of the popular approaches in distributed machine learning with better fault tolerance, it suffers from a bandwidth bottleneck especially with large scale systems.[37,70]
- Although the PS congestion issue might be alleviated through some alternative PS infrastructures (eg, shared PS), selecting the appropriate ratio of PSs in these alternative configuration is still a challenging task.[58]
- The Ring-Allreduce algorithm is built on a HPC approach proposed in 2009 by Patarasuk and Yuan.[71] It is a highly scalable and bandwidth optimal approach as it remarkably reduces the network communications overhead,[58] which perfectly corresponds to our aforementioned parallelism schema selection criterion.

In summary, we decided to adopt a ***decentralized synchronous Ring-Allreduce data parallelism strategy*** in order to bring `Dist-FCN` and `Dist-U-Net` into practice.

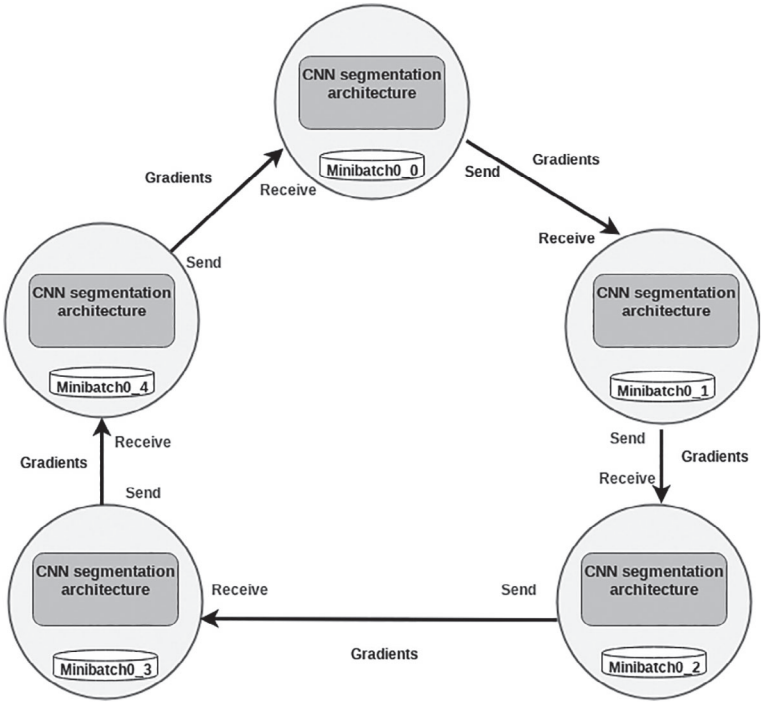



**FIGURE 7** Decentralized synchronous ring all-reduce parallelism schema adopted in the R2D2 `Dist-Training` module[69,72]

As shown in Figure 7, we distributed the training of FCN and U-Net CNNs as follows: First and foremost, we introduced data parallelism by deploying the same CNN segmentation architecture on each training node (ie, either FCN or U-Net). After that, we sat up the Ring All-reduce algorithm. The latter steps are as follows: Initially, each worker node reads its own subset of the current minibatch. After that, it computes its gradients, and communicates it to its nearby successor on the ring and get in turn the calculated gradients from its predecessor neighbor. In a ring that counts N workers, it takes N−1 communications of gradients between workers, so that every worker receives the required gradients values to compute the updated model. In addition, we ensured the system fault tolerance through a checkpoint/restart schema. Last but not least, considering that we scaled up the training of FCN and U-Nets CNNs using a data parallelism schema, we applied the *learning rate linear scaling rule* which consists in adjusting the learning rate as a function of the minibatch size[15] in order to distribute the training of our CNNs without considerable segmentation accuracy loss.

### 3.2.3 | SegEval: Segmentation evaluation library

Once we have finished the distributed training of our CNN architecture, the next step in a typical processing pipeline is to evaluate the trained model. To this end, R2D2 provides an evaluation library which implements a set of common evaluation metrics for both binary and multilabel semantic segmentation tasks.

We denote $n_{cl} \; \epsilon \; \mathbb{N}$ the number of classes. In addition, we denote $n_{ij}$ the number of pixels of class $i$ predicted to belong to class $j$ and $t_i = \sum_j n_{ij}$ the total number of pixels of class $i$. The `SegEval` library offers the following widely adopted evaluation metrics.[13]

- The **Dice** score reports the percentage of overlap between the predicted segmentations and the ground truth masks: $\text{Dice} = \frac{1}{n_{cl}} \sum_i \frac{2n_{ii}}{2n_{ii}+n_{ij}+n_{ji}}$
- The **pixel accuracy** (PA) is a measure of the percentage of the image pixels that were correctly classified : $\text{PA} = \frac{\sum_i n_{ii}}{\sum_i t_i}$
- The **mean accuracy** (MA) is the mean of the PA across the $n_{cl}$ classes: $\text{MA} = \left(\frac{1}{n_{cl}}\right) \sum_i \frac{n_{ii}}{t_i}$
- The **Mean Intersection Over Union** (mean.IoU) is a measure of the area of overlap divided by the area of union between both predicted and groundtruth images : $(\text{mean.IoU}) = \left(\frac{1}{n_{cl}}\right) \sum_i \frac{n_{ii}}{t_i+\sum_j n_{ji}-n_{ii}}$ and the **frequency weighted IoU** (f.w.IoU): $\left(\sum_k t_k\right)^{-1} \sum_i \frac{t_i n_{ii}}{t_i+\sum_j n_{ji}-n_{ii}}$

# 4 | EVALUATION

In this section, we conduct a comprehensive evaluation of the `Dist-Training` module which is the core building block of our proposed R2D2 toolkit. We first introduce our experimental environments. Afterward, we present the experimental evaluation results of the distributed CNN architectures (`Dist-FCN` and `Dist-U-Net`) on a brain tumor segmentation use case. Finally, in order to validate the `Dist-Training` module transferability to other segmentation use cases, we assess the module on a second medical imaging segmentation task, which involves locating the heart's left atrium structure.

## 4.1 | Experimental environments

### 4.1.1 | Hardware

We accomplished the distributed training experiments on the Nancy Grid'5000[73] testbed site. The experiments were conducted on Grele GPU cluster which contains Dell PowerEdge R730 physical machines where each node is equipped with 2 Nvidia GeForce GTX 1080 Ti GPUs. We use the Grid'5000 network file system (NFS) to share the training dataset between all training agents. The nodes are interconnected using *InfiniBand*[74] high-speed interconnect.

### 4.1.2 | Software

The FCN and U-Net architectures were mutually built on top of google's *Tensor-Flow* library. Furthermore, the U-Net CNN was also implemented using the high-level *keras*[75] API to ensure an easier architecture prototyping task. After that, in order to practically implement the `Dist-FCN` and `Dist-U-Net` by introducing the considered synchronous Ring-Allreduce data parallelism schema, we take advantage of the *Horovod*[58] based implementation of the Ring-Allreduce algorithm. The latter is built concurrently on both, *Open MPI*[76] and *NCCL 2.0*[3]communication libraries. Moreover, during the experimental evaluation, we simultaneously make use of the proposed R2D2 module incorporating *TICK Stack* monitoring platform [4]in order to collect system metrics data during the distributed training. The collected datasets are stored in the time series database*InfluxDB*. In addition, since the `Dist-Training` module is partially built using the *Tensor-Flow* library, we leverage the natively integrated *TensorBoard* visualization component, in order to enable R2D2 users to have an extensive overview on the training progress and hence facilitating the debugging of the CNNs training step. Finally, to consider software reusability, and ensure research reproducibility, the `Dist-Training` module and its runtime components were containerized into a debian 9 stretch-based docker image without network isolation (ie, by directly using the host networking driver for an optimal network performance[77]). Figure 8 details the physical architecture of our experimental environment.

## 4.2 | Medical imaging evaluation case studies

### 4.2.1 | Brain tumor segmentation use case

The first use case we have chosen in order to evaluate the R2D2 `Dist-Training` module is the brain tumor segmentation task which was proposed during the decathlon medical segmentation challenge [5]. Actually, the brain tumor segmentation involves isolating the different tumor tissues in the brain from healthy ones.[78-80] It is a crucial and challenging task in medical image analysis because it plays an influential role in early diagnosis of brain tumors which in turn enhance treatment planning and raise the survival rate of the patients.[78,80] Yet, it is a tedious and time consuming task because it might take hours even if it is manually performed by expert radiologists.[78]

[3]https://developer.nvidia.com/nccl
[4]More informations on TICK Stack platform can be found at the following link https://www.influxdata.com/time-series-platform/
[5]More informations on the decathlon segmentation challenge can be found at the following links: http://medicaldecathlon.com/ and https://decathlon.grand-challenge.org/

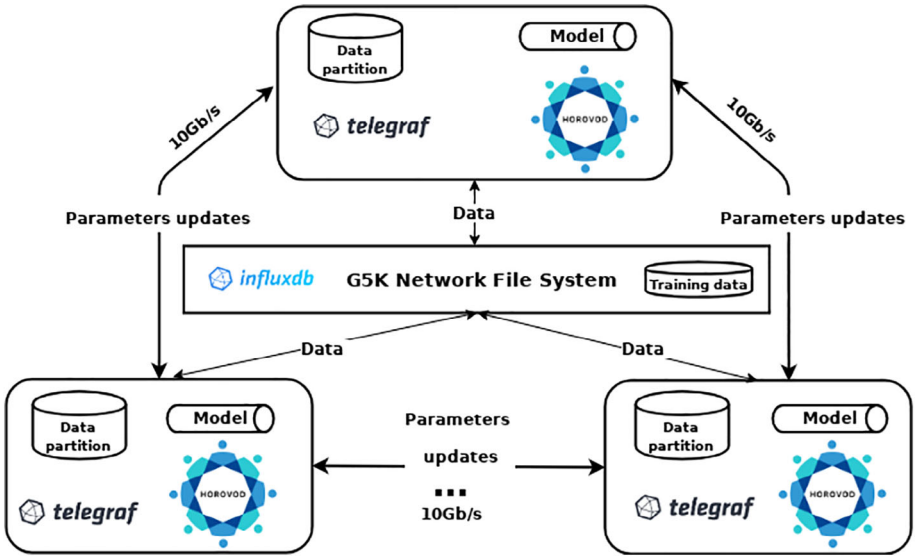


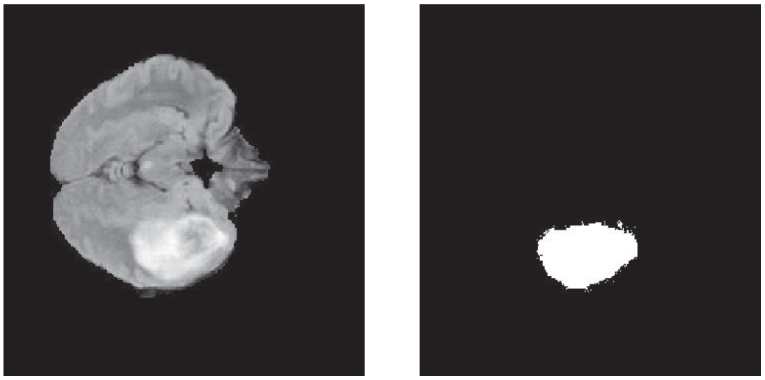

**FIGURE 8** Distributed training experimental environment architecture [Color figure can be viewed at wileyonlinelibrary.com]

*Dataset*

The dataset has been provided during decathlon medical segmentation challenge for the brain tumors segmentation task. It is a mix of two other datasets that have been initially made publicly available during the *Multimodal Brain Tumor Segmentation Challenge: MICCAI BRATS*[81] *2016 and 2017*. It contains multimodal MRI scans (ie, 4D MRI scans[78]) of complex and heterogeneously located brain tumors that were captured using multiple distinct MRI acquisition protocol[78] from 19 different institutional data contributors.[81] The BRATS datasets have been initially manually segmented by one to four raters, using the same annotation protocol. After that, the multimodal brain tumor MRI scans along with all their corresponding ground truth labels were manually reexamined and approved by experienced neurologists.[81] Figure 9 shows an example of a brain MRI scan slice containing a tumor and its related annotated MRI scan slice. The final dataset provided by decathlon and used to build our model contains in total 750 annotated MRI scans. It was split into two data subsets. The first partition is a training and validation dataset with 484 annotated MRI scans. The second subset contains 266 annotated MRI scans dedicated to the testing phase.

*Preprocessing pipeline*

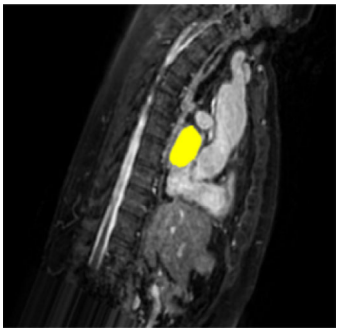
Since decathlon original dataset involves multimodal MRI scans (4D), it was preprocessed in order to extract the corresponding 2D images alongside with their annotations for every MRI scan in the provided dataset. In order to do so, the initial dataset was reduced to T1-weighted MRI scans (3D).[82] After that, we extracted 70 2D-MRI slices per MRI scan. Therefore, at the end of the preprocessing pipeline, the final training and validation dataset counts in total 33 880 2D-MRI images alongside with their related annotations. This contributes to avoid overfitting without performing data augmentation regularization technique on the training dataset. In addition, the same preprocessing pipeline was applied to the testing dataset which counts at the end 18 620 annotated 2D-MRI images.



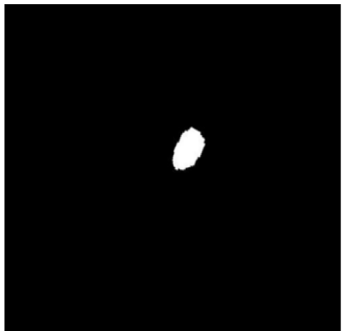
2D-MRI slice of BRATS training dataset

Corresponding brain tumor ground-truth annotation

**FIGURE 9** BRATS MRI scan frame with its corresponding ground truth annotation



2D MRI Slice of the input training dataset (The yellow structure is the target left atrial chamber)



Corresponding left atrium ground-truth manual segmentation

**FIGURE 10** Left atrium segmentation training dataset [Color figure can be viewed at wileyonlinelibrary.com]

| Operations | Parameters |
|---|---|
| Rotation | *rotation_range* $\varepsilon[-45, 45]$ |
| Zoom | *zoom_range* $\varepsilon[0.8, 1.2]$ |
| | $(x + shift\_range, y = 0)$, *shift_range* $\varepsilon[0, 25.6]$ |
| Translation | $(x = 0, y + shift\_range)$, *shift_range* $\varepsilon[0, 25.6]$ |

**TABLE 1** Data augmentation operations parameters of the left atrium dataset

### 4.2.2 | Left atrium segmentation use case

The left atrial segmentation task is the second medical imaging segmentation task we consider to asses the `Dist-Training` module effectiveness. This segmentation task was provided by the King's College London University during the left atrial segmentation challenge (LASC).[83] As shown in Figure 10, the left atrium segmentation consists in isolating the left atrium body from its surrounding organs structures.[83] It plays a key role during the treatment protocol of patients with atrial fibrillation disease[83] which is the most frequent cardiac electrical disorder provoked by abnormal electrical discharges in the left atrium.[84] Besides that, the latter left atrium segmentation task is also essential for cardiac biophysical modeling procedure. Nonetheless, the thinness of the atrial wall tissue makes the left atrium segmentation process a challenging task.[83]

*Dataset*

The dataset has been made publicly available by Philips Technologie GmbH, Hamburg, DE, and King's College London during the 2013 LASC challenge [6]. Unlike the BRATS datasets, the left atrium segmentation dataset is a small one with wide quality levels variability as it only includes 30 mono-modal 3D cardiac MRI scans. This will allow us to further assess the transferability of the `Dist-Training` module. The dataset was split such that 20 MRI scans were provided with their corresponding ground truth annotations for the training and the validation steps. The remaining 10 MRI scans were supplied as a test set. The ground-truth masks were initially annotated using automatic model-based segmentation. Afterward, a manual corrections were performed by human experts.[83]

*Preprocessing pipeline*

The preprocessing workflow of the provided datasets involves the following steps: (i) 70 2D-MRI slices were initially extracted from each 3D cardiac mri scan through the `ImgMed`-implemented reshaping operation; (ii) A downsampling operation to a size of $512 \times 512$ pixels has been carried on each image in order to fit the memory constraint of the GPU (NVIDIA GeForce GTX 1080 Ti); (iii) In order to break the curse of small datasets and avoid overfitting, data augmentation technique has been performed on the LASC dataset with rotation, zooming and translation operations as detailed in Table 1. Hence, at the end of the preprocessing pipeline, the final training and validation dataset counts in total 35 000 2D-MRI images alongside with their related annotations.

[6]The left atrium segmentation dataset is available at the following link https://www.cardiacatlas.org/challenges/left-atrium-segmentation-challenge/

### 4.3 | Dist-Training module training time evolution with scalability

In this subsection, we evaluation the `Dist-Training` module training time evolution while increasing the number of GPUs.

Figures 11 and 12 illustrate the decrease in the training time while scaling up the training of `Dist-U-Net` and `Dist-FCN`, respectively. Multiple runs have been conducted for each configuration to assess `Dist-U-Net` and `Dist-FCN` evaluation results variability. We run each experimental training configuration between three and five times and we consider the average of the measured execution duration as our reference inference results. The `Dist-U-Net` reaches **17.5×** speedup and **97%** scaling efficiency going from 21 hours and 40 minutes for single-GPU based U-Net, to 1 hour and 14 minutes for `Dist-U-Net` trained on 18 Nvidia GTX 1080 Ti GPUs. In the other side, the `Dist-FCN`achieves **10.4×** speedup and **58%** scaling efficiency reducing the training time from 35 hours and 40 minutes for a single-GPU based `Dist-FCN` to 3 hours and 25 minutes for `Dist-FCN` trained on 18 Nvidia GTX 1080 Ti GPUs.

The `Dist-U-Net` and `Dist-FCN` have been evaluated concurrently on the two previously introduced medical imaging segmentation use cases. Figures 11 and 12 both show that the BRATS training time evolution curve closely match the left atrium one which establishes the transferability of our proposal. In addition, we notice that the baseline U-Net CNN converges faster than the FCN one in both segmentation case studies in only 21 hours and 40 minutes. This observation matches and confirms the findings of Li et al[17] which highlights that residual connections (similar to the ones that exist in U-Net architecture) produce a smoother loss function which leads to an easier and faster convergence.

The disparity of the scaling efficiency and speedup of `Dist-FCN` and `Dist-U-Net` is mainly due to the nature of the experimental setup and the difference in their corresponding implementation strategies. In particular, during the training process of `Dist-U-Net`, the entire training and validation sets are loaded into the random access memory of each training agent. On the other hand, the `Dist-FCN` implementation takes advantage of the GPU's dedicated memory to iterate through the training and validations datasets in order to load each minibatch through a NFS which represents a communication overhead. Furthermore, the `Dist-U-Net` implementation contains highly optimized operations to accelerate the distributed validation step.

We assess the testing time of `Dist-FCN` and `Dist-U-Net`. Indeed, in order to eliminate the system routine operations influence in time measures, we run each experimental setup 10 times and we consider the average of the measured

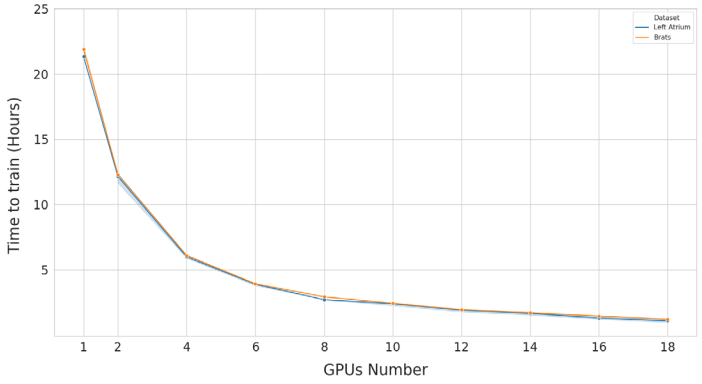



**FIGURE 11** Training time evolution with scale for Dist-U-Net [Color figure can be viewed at wileyonlinelibrary.com]

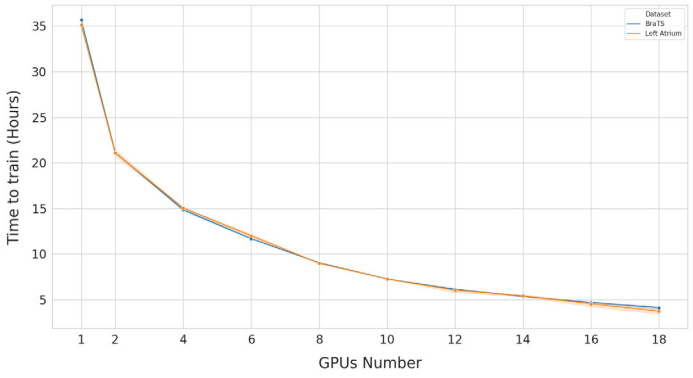



**FIGURE 12** Training time evolution with scale for Dist-FCN [Color figure can be viewed at wileyonlinelibrary.com]

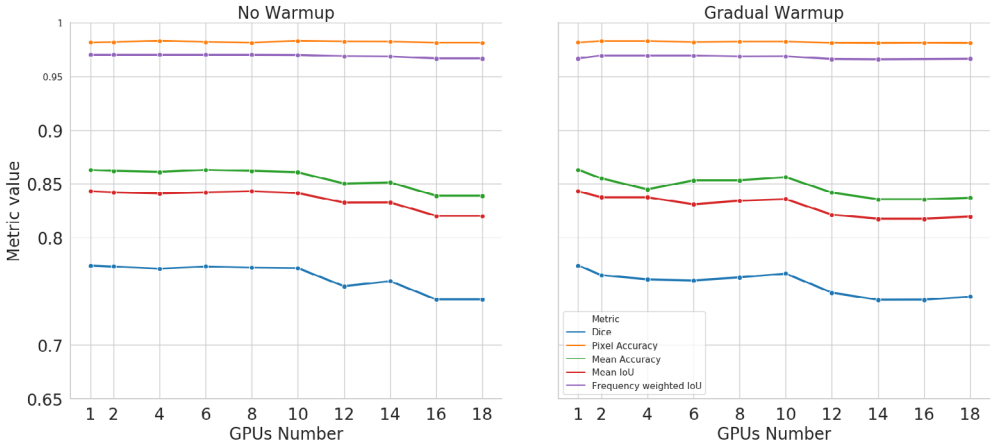



**FIGURE 13** Segmentation metrics evolution with scale for Dist-FCN for brain tumor segmentation [Color figure can be viewed at wileyonlinelibrary.com]

execution duration as our reference inference results. The testing workstation is equipped with an NVIDIA GTX 1080 TI GPU. The obtained testing times per image are 153 and 182 ms for `Dist-U-Net` and `Dist-FCN`, respectively.

## 4.4 | Dist-Training segmentation accuracy evolution with scalability

Throughout this subsection, we evaluate the impact of increasing the GPUs number on the segmentation accuracy. Even though it is common to use a unique segmentation evaluation metric, we consider the entire evaluation metrics provided by the previously introduced `SegEval` module in order to comprehensively assess our proposal.

### 4.4.1 | Dist-FCN for brain tumor segmentation

The adopted `Dist-FCN` architecture for the brain tumor segmentation consists of a total of 16 FCN layers. It takes the input volume throughout a sequence of an increasing number (ie, n=2 for the first two blocks and n=4 for the rest of blocks) of convolutional layers which are immediately followed by ReLU activation function. At the end, a max-pooling layer is applied. The sequence n x (convolution + ReLU) is repeated again 4 times before performing *upsampling* through a transposed convolution upsampling layer[32] and applying a softmax layer[13,26] for the pixel-wise classification task. Moreover, dropout regularization technique[13] was applied during the training phase to avoid overfitting. Finally, the network parameters were initialized through *Transfer Learning* using a pretrained VGG-16 model on the ImageNet dataset [7].

*Training settings*

The training was performed using the minibatch stochastic gradient descent (SGD) optimization algorithm with a mini-batch size of 10 (to fit the GPU memory limit). The training process was done for a total of 120 epochs and a learning rate of $1e-5$. All training hyperparameters were kept unchanged except of the learning rate which was adjusted according to *the learning rate linear scaling rule*. In the first place, no learning rate warmup strategy was applied, before performing a gradual warmup schema afterward. The gradual warmup schema consists in applying progressively a low learning rate for the first few epochs in order to overcome convergence issues at the start of the training.[15]

*Evaluation*

As shown in Figure 13, the `Dist-FCN` for brain tumor segmentation trained on 18 GPUs without learning rate warmup strategy reach 74.22% *dice score* accuracy, 84.29% *Mean IoU* and 86.28% *MA* which are 4.08% , 2.7%, and 3.35%, respectively, lower than the single-GPU baselinemodel. At the same time, the gradual warmup strategy enhance the segmentation accuracy loss by 3.76%, 2.8%, and 3.01% for the *dice score*, *Mean IoU* and the *MA* metrics correspondingly. Our practical experiments results show an interesting unexpected segmentation accuracy loss increasing with the parallelism degree.

[7]More informations and the download link for the pretrained VGG-16 model on the ImageNet dataset can be found at the following link: http://www.vlfeat.org/matconvnet/pretrained/

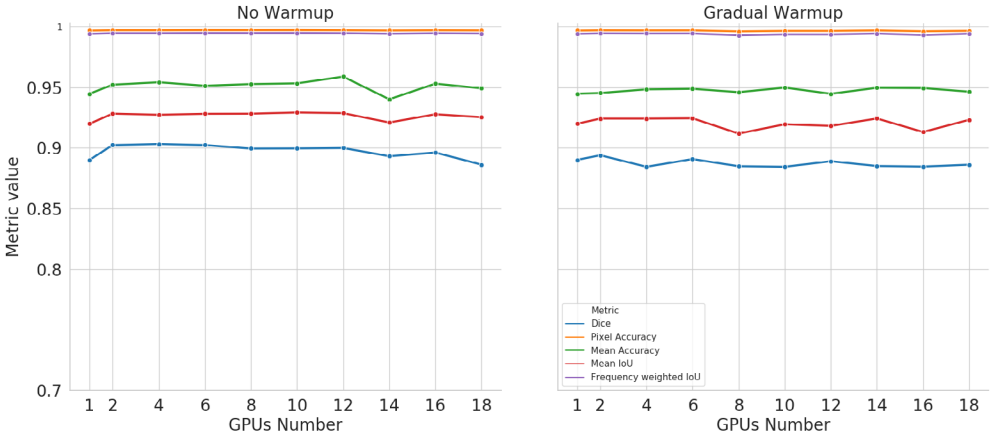



**FIGURE 14** Segmentation metrics evolution with scale for Dist-U-Net for brain tumor segmentation [Color figure can be viewed at wileyonlinelibrary.com]

### 4.4.2 | Dist-U-Net for brain tumor segmentation

*Training settings*

For training, we use the minibatch SGD optimization algorithm. The training phase was done with a minibatch size of 7, during 100 epochs and while using an initial base learning rate of $1e-5$.

*Evaluation*

Figure 14 introduces the brain tumor segmentation accuracy evolution when scaling up the `Dist-U-Net` (1) with no warmup phase and (2) while performing a gradual warmup for the first 5 epochs. As can be seen, the *dice score* decreased by 0.44% going from 0.890 in 1-GPU implementation to 0.886 in 18 GPUs in the case of no warmup phase. Similarly, the *mean.IoU* and *MA* metrics drop with 0.55% and 0.61%, respectively. On the other hand, the gradual warmup strategy achieves the same *dice score* accuracy loss as the no warmup strategy. Nonetheless, the gradual warmup strategy seems not to be effective at low scalability level as is does not help the network to converge faster. Finally, no accuracy degradation is reported in the PA and *f.w.IoU* metrics regardless of the adopted warmup schema. To sum up, our experiments highlights an unexpected segmentation accuracy degradation with scale, nevertheless its small value.

### 4.4.3 | Dist-FCN for left atrium segmentation

We adopted a 16 layers `Dist-FCN` CNN architecture similar to aforementioned one in subsubsection 4.4.1. Similarly, we also leveraged transfer learning approach using pretrained VGG-16 model on the ImageNet dataset.

*Training settings*

The training was performed using the mini SGD optimization approach, a minibatch size of 10, for a total of 120 epochs. We also applied the *learning rate linear scaling rule* starting with an initial learning rate of $3e-5$.

*Evaluation*

Figure 15 illustrates the segmentation accuracy metrics evolution when scaling up the `Dist-FCN` for left atrium segmentation before and after performing Gradual warmup strategy. It illustrates a *dice score* accuracy and *MA* fall of 3.38% and 1.3% accordingly for a gradual warmup initialization learning rate approach. Yet, with no warmup strategy, the `Dist-FCN` achieves better results with 1.21%, 0.86%, and 1.46% segmentation accuracy decrease for the *dice score*, *MA*, and *mean.IoU*, respectively. However, no accuracy loss is reported for the *f.w.IoU* and the *PA* metrics. Finally, once again, even if the linear scaling rule is supposed to eliminate the accuracy loss, our experiments show a quite surprising accuracy degradation when scaling up the considered GPUs number.

### 4.4.4 | Dist-U-Net for left atrium segmentation

*Training settings*

For training, we use the minibatch SGD optimization algorithm. The training phase was done with a minibatch size of 7, during 100 epochs and while using a learning rate of $2e-5$.

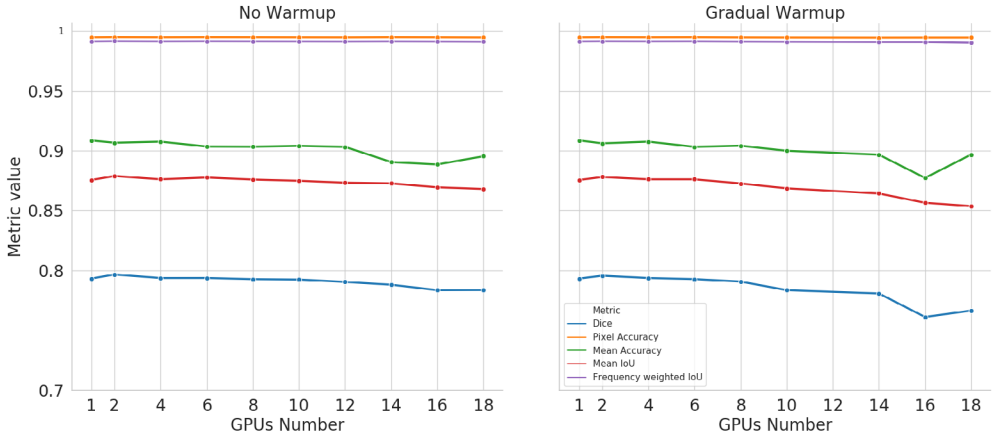



**FIGURE 15** Segmentation metrics evolution with scale for Dist-FCN for left atrium segmentation [Color figure can be viewed at wileyonlinelibrary.com]

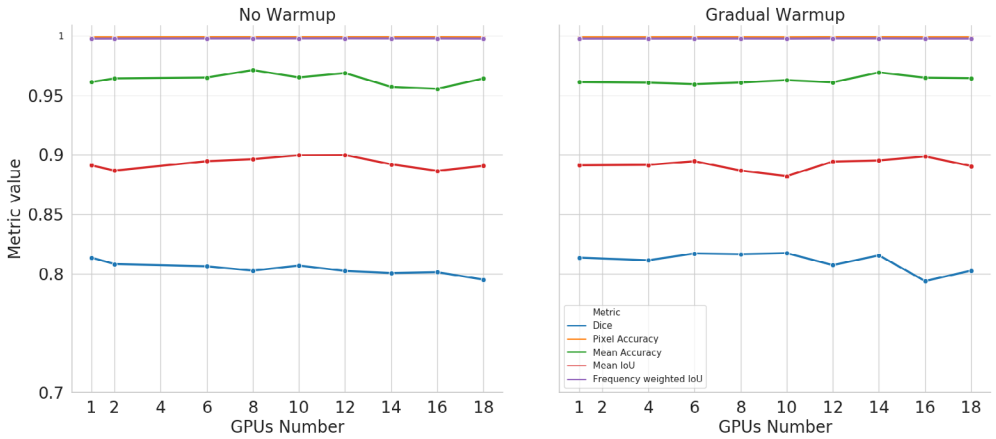



**FIGURE 16** Segmentation metrics evolution with scale for Dist-U-Net for left atrium segmentation [Color figure can be viewed at wileyonlinelibrary.com]

*Evaluation*

As can be seen in Figure 16, scaling up the training of `Dist-U-Net` for left atrium segmentation to 18 GPUs without gradual learning rate warmup strategy achieves 79.48% *dice score* accuracy and 96.41% *MA* which are 2.26% and 1.53%, respectively, lower than the single-GPU `Dist-U-Net` baseline trained model. However, the gradual warmup approach improves the accuracy degradation to only 1.34% and 0.31% for the *dice score* and *MA* metric correspondingly. Yet again, our experimental results reveal a quite unexpected segmentation accuracy loss when scaling up the CNNs training process. In addition, these results show that the *PA* and *f.w.IoU* metrics are not very relevant for our experiments assessment process. Indeed, they suffer from a unbalanced variability range due to the disproportional size of every class in our segmentation case studies (eg, the disproportional size between the small left atrium body and large background class size)

## 4.5 | Discussion

We evaluated our proposed `Dist-FCN` and `Dist-U-Net` training time and segmentation accuracy metrics evolution with scale on a couple of challenging medical imaging segmentation case studies (i) BRATS: a dataset with small targets (tumors) in large MRI images (ii) Left Atrium: a small training set with large variability. The case studies evaluation results led us to not only assess the segmentation accuracy evolution when scaling up the `Dist-FCN` and `Dist-U-Net` architectures, but also to compare FCN and U-Net performances in a couple of different segmentation tasks. Actually, the evaluation results showed that the U-Net CNN architecture achieves a far better performances than the FCN one in the brain tumor segmentation task with 90.23% dice score. In addition, the U-Net and FCN CNNs produce a close results in term of performances for the left atrium segmentation with an advantage of 1.8% in the dice score for the U-Net architecture. These findings confirm the need to perform multiple CNNs training runs in order to investigate the best suited CNN architecture for a particular task alongside with its corresponding optimal hyperparameters set. Hence, the

interest of R2D2 in accelerating the prototyping and the development of cutting-edge CNNs which in turn is a capital software engineering principal.

The aforementioned empirical evaluation results led us also to perform a deeper experimental analysis of the generalization of last published works[15-19] to the segmentation task. Actually, the segmentation task is a more complex task comparing to the classification task which was mainly used in the state-of-the-art works to asses the linear scaling rule and its corresponding warmup schemas.[15] The experimental results showed that there was no segmentation accuracy loss until 12 GPUs. Starting from that scalability level, the learning rate scaling rule breaks down and a negligible accuracy loss comparing to the remarkable scaling efficiency starts to appears. These results are in line with the 1% increase of error rate reported by Krizhevsky[85] when increasing the minibatch size from 128 to 1024. In addition, You et al[18] outline also an accuracy deterioration of 5.7% by using the linear scaling rule and the warmup schema. Furthermore, Hoffer et al[19] show that there is still an accuracy degradation for CIFAR10 [8]classification task even while using the linear scaling rule. Hence, our experiments confirm the results of these works[18,19,85] and call into question the extent of the linear scaling rule to the segmentation task.

## 5 | CONCLUSION AND FUTURE WORK

In this article, we proposed and evaluated an easy-to-use scalable DLTK for medical imaging segmentation named R2D2. The main goal of R2D2 is to speed-up the research in the deep-leaning-based medical imaging applications with a particular focus on semantic segmentation task. We exposed R2D2 concepts and design and detailed its inner buildings components, while justifying our design and implementation choices. We then evaluated our scalable toolkit on two distinct concrete medical imaging segmentation case studies to show the effectiveness of our proposal.

As future work, we aim to broaden the spectrum of supported CNNs in R2D2 by not only implementing other scalable CNN-based segmentation architectures, but also through supporting a wider range of medical imaging tasks (eg, registration, classification). Another promising area of research is to analyze the collected data during the distributed training, with the purpose of getting valuable insights and revealing correlations and hidden patterns within collected datasets. We plan also to shift our distributed training platform from Grid'5000 testbed toward private cloud solutions in order to further evaluate our proposed solution scalability on a production ready environment.

### ACKNOWLEDGEMENTS

This work was supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-10-AIRT-05), the Virtual Studio RA and Smart Support Center (SSC) FEDER EU projects, the FSN Hydda project and Institut Carnot LSI. Experiments presented in this article were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

### ORCID

*Soulaimane Guedria* https://orcid.org/0000-0001-6622-0418
*Nicolas Vuillerme* https://orcid.org/0000-0003-3773-393X

### REFERENCES

1. Shi J, Malik J. Normalized cuts and image segmentation. *IEEE Trans Pattern Anal Mach Intell*. 2000;22(8):888-905. https://doi.org/10.1109/34.868688.
2. Shin H, Roth HR, Gao M, et al. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE TRANSACTIONS ON MEDICAL IMAGING*. 2016;abs/1602.0340935(5):1285-1298.
3. Roth Holger R., Lu Le, Liu Jiamin, Yao Jianhua, Seff Ari, Cherry Kevin, Kim Lauren, Summers Ronald M. Improving Computer-Aided Detection Using_newlineConvolutional Neural Networks and Random View Aggregation. *IEEE Transactions on Medical Imaging*. 2016;35(5):1170-1181. http://dx.doi.org/10.1109/tmi.2015.2482920.
4. Sharma N, Aggarwal L. Automated medical image segmentation techniques. *J Med Phys*. 2010;35(1):3-14.
5. LeCun Y, Bengio Y, Hinton GE. Deep learning. *Nature*. 2015;521(7553):436-444. https://doi.org/10.1038/nature14539.
6. Greenspan H, van Ginneken B, Summers RM. Guest editorial deep learning in medical imaging: overview and future promise of an exciting new technique. *IEEE Trans Med Imaging*. 2016;35(5):1153-1159.

[8]https://www.cs.toronto.edu/~kriz/cifar.html

7. Milletari F, Navab N, Ahmadi S. V-Net: fully convolutional neural networks for volumetric medical image segmentation. Paper presented at: Proceedings of the2016 4th International Conference on 3D Vision (3DV), Stanford, CA; 2016:565-571.
8. Shen D, Wu G, Suk HI. Deep learning in medical image analysis. *Annu Rev Biomed Eng*. 2017;19:221-248.
9. Loshchilov I, Hutter F. CMA-ES for hyperparameter optimization of deep neural networks; 2016. arXiv preprint arXiv:160407269.
10. Bergstra J, Bengio Y. Random search for hyper-parameter optimization. *J Mach Learn Res*. 2012;13:281-305. http://dl.acm.org/citation.cfm?id=2188385.2188395.
11. Guedria S, De Palma N, Renard F, Vuillerme N. Automating CNN parallelism with components. Paper presented at: Proceedings of the 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, Nevada; IEEE; 2019:943-948.
12. Kamnitsas K, Ferrante E, Parisot S, et al. DeepMedic for brain tumor segmentation. Paper presented at: Proceedings of the International Workshop on Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries; 2016:138-149; Springer.
13. Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. Paper presented at: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, Massachusetts; 2015.
14. Ronneberger O, Fischer P, Brox T. U-Net: convolutional networks for biomedical image segmentation. *CoRR*. 2015;abs/1505.04597935:234-241. http://arxiv.org/abs/1505.04597.
15. Goyal P, Dollár P, Girshick RB, et al. Accurate, large minibatch SGD: training ImageNet in 1 hour. *CoRR*. 2017;abs/1706.02677. http://arxiv.org/abs/1706.02677.
16. Smith SL, Kindermans P, Le QV. Don't decay the learning rate, increase the batch size. *CoRR*. 2017;abs/1711.00489. http://arxiv.org/abs/1711.00489.
17. Li H, Xu Z, Taylor G, Studer C, Goldstein T. Visualizing the loss landscape of neural nets. *Advances in Neural Information Processing Systems*. Montréal, Canada; 2018:6389-6399.
18. You Y, Gitman I, Ginsburg B. Scaling sgd batch size to 32k for imagenet training; 2017. https://digitalassets.lib.berkeley.edu/techreports/ucb/text/EECS-2017-156.pdf.
19. Hoffer E, Hubara I, Soudry D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in Neural Information Processing Systems*; 2017:1731-1741.
20. Hubel DH, Wiesel TN. Receptive fields of single neurons in the cat's striate cortex. *J Physiol*. 1959;148:574-591.
21. Cox D, Dean T. Neural networks and neuroscience-inspired computer vision. *Curr Biol*. 2014;24:921-929.
22. LeCun Y, Bengio Y. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press; 1998.p. 255-258. http://dl.acm.org/citation.cfm?id=303568.303704.
23. Snoek J, Larochelle H, Adams RP. Practical Bayesian optimization of machine learning algorithms. In: Pereira F, CJC B, Bottou L, Weinberger KQ, eds. *Advances in Neural Information Processing Systems*. Vol 25. Red Hook, NY: Curran Associates, Inc; 2012:2951-2959. http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machi.
24. Rasmussen CE. Gaussian processes in machine learning. In: Bousquet O, von Luxburg U, Rätsch G, eds. *Summer School on Machine Learning*. Berlin, Heidelberg/Germany: Springer; 2004:63-71.
25. Chua LO, Roska T. The CNN paradigm. *IEEE Trans Circ Syst I Fund Theory Appl*. 1993;40(3):147-156.
26. Krizhevsky Alex, Sutskever Ilya, Hinton Geoffrey E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*. 2017;60(6):84-90. http://dx.doi.org/10.1145/3065386.
27. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Intelligent Signal Processing*. 86. IEEE Press; 2001:306-351.
28. Lo SCB, Chan HP, Lin JS, Li H, Freedman MT, Mun SK. Artificial convolution neural network for medical image pattern recognition. *Neural Netw*. 1995;8(7-8):1201-1214.
29. Lakhani P, Sundaram B. Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks. *Radiology*. 2017;284(2):574-582.
30. Bar Y, Diamant I, Wolf L, Lieberman S, Konen E, Greenspan H. Chest pathology detection using deep learning with non-medical training. Paper presented at: Proceedings of the2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI), New York; 2015:294-297.
31. Garcia-Garcia A, Orts-Escolano S, Oprea S, Villena-Martinez V, Rodriguez JG., A review on deep learning techniques applied to semantic segmentation.*CoRR* 2017;abs/1704.06857. http://arxiv.org/abs/1704.06857.
32. Noh H, Hong S, Han B. Learning deconvolution network for semantic segmentation. *CoRR*. 2015;abs/1505.04366. http://arxiv.org/abs/1505.04366.
33. Guedria S, De Palma N, Renard F, Vuillerme N. Auto-CNNp: a component-based framework for automating CNN parallelism. Paper presented at: Proceedings of the 2019 IEEE International Conference on Big Data (Big Data); 2019:3330-3339; Los Angeles, California: IEEE.
34. Wu Y, Schuster M, Chen Z, et al. Google's neural machine translation system: bridging the gap between human and machine translation. *CoRR*. 2016;abs/1609.08144. http://arxiv.org/abs/1609.08144.
35. Le QV, Ranzato M, Monga R, et al. Building high-level features using large scale unsupervised learning. Paper presented at: Proceedings of theProceedings of the 29th International Coference on International Conference on Machine Learning ICML'12; 2012:507-514; Edinburgh, Scotland: Omni Press. http://dl.acm.org/citation.cfm?id=3042573.3042641.
36. Dean J, Corrado GS, Monga R, et al. Large scale distributed deep networks. Paper presented at: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 NIPS'12; 2012:1223-1231; Siem Reap, Cambodia: Curran Associates Inc. http://dl.acm.org/citation.cfm?id=2999134.2999271.

37. Ben-NunTal, Hoefler Torsten. Demystifying Parallel and Distributed Deep Learning. *ACM Computing Surveys*. 2019;52(4):1-43. http://dx.doi.org/10.1145/3320060.
38. Coates A, Huval B, Wang T, Wu D, Catanzaro B, Andrew N. Deep learning with COTS HPC systems. Paper presented at: Proceedings of the International Conference on Machine Learning, Atlanta; 2013:1337-1345.
39. Jiang J, Cui B, Zhang C, Yu L. Heterogeneity-aware distributed parameter servers. Paper presented at: Proceedings of the 2017 ACM International Conference on Management of Data, Chicago Illinois; 2017:463-478; ACM.
40. Li M, Andersen DG, Park JW, et al. Scaling distributed machine learning with the parameter server. Paper presented at: Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation OSDI'14; 2014:583-598; Broomfield, CO: Berkeley, CAUSENIX Association. http://dl.acm.org/citation.cfm?id=2685048.2685095.
41. Chilimbi T, Suzue Y, Apacible J, Kalyanaraman K. Project adam: building an efficient and scalable deep learning training system. Paper presented at: Proceedings of the11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14), Broomfield, CO; 2014:571-582.
42. Strom N. Scalable distributed DNN training using commodity GPU cloud computing. Dresden, Germany: Proc. INTERSPEECH; 2015.
43. Lee L, Liew SC. *A Survey of Medical Image Processing Tools*. Kuantan, Malaysia: IEEE ICSECS; 2015.
44. Wolf I, Vetter M, Wegner I, et al. The medical imaging interaction toolkit. *Med Image Anal*. 2005;9(6):594-604.
45. Pieper S, Lorensen B, Schroeder W, Kikinis R. The NA-MIC Kit: ITK, VTK, pipelines, grids and 3D slicer as an open platform for the medical image computing community. Paper presented at: Proceedings of the 3rd IEEE International Symposium on Biomedical Imaging: Nano to Macro; 2006:698-701.
46. Maintz JA, Viergever MA. A survey of medical image registration. *Med Image Anal*. 1998;2(1):1-36.
47. Beutel J, Kundel HL, Van Metter RL. *Handbook of Medical Imaging*. Vol 1. Washington: Spie Press; 2000.
48. Jenkinson M, Beckmann CF, Behrens TE, et al. Advances in functional and structural MR image analysis and implementation as FSL. *NeuroImage*. 2012;62(2):782-790.
49. Cardoso M, Clarkson M, Modat M, Ourselin S. NiftySeg: open-source software for medical image segmentation, label fusion and cortical thickness estimation. Paper presented at: Proceedings of the IEEE International Symposium on Biomedical Imaging; 2012; Barcelona, Spain.
50. Johnsen SF, Taylor ZA, Clarkson MJ, et al. NiftySim: a GPU-based nonlinear finite element package for simulation of soft tissue biomechanics. *Int J Comput Assist Radiol Surg*. 2015;10(7):1077-1095.
51. Cook P, Bai Y, Nedjati-Gilani S, et al. Camino: open-source diffusion-MRI reconstruction and processing. Paper presented at: Proceedings of the14th Scientific Meeting of the International Society for Magnetic Resonance in Medicine, Seattle, WA; vol. 2759, 2006:2759.
52. Gibson E, Li W, Sudre C, et al. NiftyNet: a deep-learning platform for medical imaging. *Comput Methods Prog Biomed*. 2018;158:113-122.
53. Pawlowski N, Ktena SI, Lee MCH, et al. DLTK: state of the art reference implementations for deep learning on medical images. *CoRR*. 2017;abs/1711.06853. http://arxiv.org/abs/1711.06853.
54. Mehrtash A, Pesteie M, Hetherington J, et al. DeepInfer: open-source deep learning deployment toolkit for image-guided therapy. Paper presented at: Proceedings of the SPIE–the International Society for Optical Engineering; vol. 10135, 2017; NIH Public Access.
55. Pieper S, Lorensen B, Schroeder W, Kikinis R. The NA-MIC Kit: ITK, VTK, pipelines, grids and 3D slicer as an open platform for the medical image computing community. Paper presented at: Proceedings of the 3rd IEEE International Symposium on Biomedical Imaging: Nano to Macro, 2006; 2006:698-701; Venice, Italy: IEEE.
56. Larrabide I, Omedas P, Martelli Y, et al. GIMIAS: an open source framework for efficient development of research tools and clinical prototypes. Paper presented at: Proceedings of the International Conference on Functional Imaging and Modeling of the Heart; 2009:417-426; Springer, New York, NY.
57. Avants BB, Tustison N, Song G. Advanced normalization tools (ANTS). *Insight J*. 2009;2:1-35.
58. Sergeev A, Balso MD. Horovod: fast and easy distributed deep learning in TensorFlow. *CoRR*. 2018;abs/1802.05799. http://arxiv.org/abs/1802.05799.
59. Zhang YJ. A survey on evaluation methods for image segmentation. *Pattern Recogn*. 1996;29(8):1335-1346.
60. Pan SJ, Yang Q. A survey on transfer learning. *IEEE Trans Knowl Data Eng*. 2010;22(10):1345-1359.
61. Yu L, Wang S, Lai KK. An integrated data preparation scheme for neural network data analysis. *IEEE Trans Knowl Data Eng*. 2006;18(2):217-230.
62. Hunter JD. Matplotlib: a 2D graphics environment. *Comput Sci Eng*. 2007;9(3):90.
63. Bradski G. The OpenCV Library. *Dr Dobb's J Softw Tools*. 2000;25:122-125.
64. Pauli, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*. 2020;17(3):261-272. http://dx.doi.org/10.1038/s41592-019-0686-2.
65. Lee S, Kim JK, Zheng X, Ho Q, Gibson GA, Xing EP. On model parallelization and scheduling strategies for distributed machine learning. *Advances in Neural Information Processing Systems*; 2014:2834-2842.
66. Harlap A, Narayanan D, Phanishayee A, et al. Pipedream: fast and efficient pipeline parallel dnn training; 2018. arXiv preprint arXiv:180603377.
67. Akiba T, Suzuki S, Fukuda K. Extremely large minibatch sgd: training resnet-50 on imagenet in 15 minutes; 2017. arXiv preprint arXiv:171104325.
68. Zhang W, Gupta S, Lian X, Liu J. Staleness-aware async-SGD for distributed deep learning. Paper presented at: Proceedings of the 25th International Joint Conference on Artificial Intelligence IJCAI'16; 2016:2350-2356; New York: AAAI Press. http://dl.acm.org/citation.cfm?id=3060832.3060950.

69. Gibiansky A. *Bringing HPC Techniques to Deep Learning. Baidu Research, Technical Report*, 2017, https://andrew.gibiansky.com/blog/machine-learning/baidu-allreduce/.com/bringing-hpc-techniques-deep-learning/. Bingjing Zhang TESTS & CERTIFICATIONS IBM Certified Database Associate-DB2 Universal Database.
70. Li M, Andersen DG, Smola AJ, Yu K. Communication efficient distributed machine learning with the parameter server. *Advances in Neural Information Processing Systems*; 2014:19-27.
71. Patarasuk P, Yuan X. Bandwidth optimal all-reduce algorithms for clusters of workstations. *J Parall Distrib Comput*. 2009;69(2):117-124.
72. Dowling J. Distributed TensorFlow; 2017. https://www.oreilly.com/content/distributed-tensorflow/.
73. Balouek D, Carpen Amarie A, Charrier G, et al. Adding virtualization capabilities to the Grid'5000 testbed. In: Ivanov II, van Sinderen M, Leymann F, Shan T, eds. *Cloud Computing and Services Science*. Communications in Computer and Information Science. Vol 367. Berlin, Germany: Springer International Publishing; 2013:3-20.
74. Shipman GM, Woodall TS, Graham RL, Maccabe AB, Bridges PG. Infiniband scalability in Open MPI. Paper presented at: Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium, Rhodes Island, Greece; 2006:10.
75. Chollet F, others. *Keras*; 2015. Available at: https://github.com/fchollet/keras.
76. Gabriel E, Fagg GE, Bosilca G, et al. Open MPI: goals, concept, and design of a next generation MPI implementation. Paper presented at: Proceedings of the European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting; 2004:97-104; Springer, New York, NY.
77. Felter W, Ferreira A, Rajamony R, Rubio J. An updated performance comparison of virtual machines and linux containers. Paper presented at: Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Philadelphia, PA; 2015:171-172; IEEE.
78. Isin A, Direkoglu C, Sah M. Review of MRI-based brain tumor image segmentation using deep learning methods. *Procedia Comput Sci*. 2016;102(C):317-324. https://doi.org/10.1016/j.procs.2016.09.407.
79. Zhao L, Jia K. Multiscale CNNs for Brain Tumor Segmentation and Diagnosis. *Comput Math Methods Med*. 2016;2016:8356294:1-8356294:7. https://doi.org/10.1155/2016/8356294.
80. Gordillo N, Montseny E, Sobrevilla P. State of the art survey on MRI brain tumor segmentation. *Magn Reson Imag*. 2013;31(8):1426-1438.
81. Menze B, Jakab A, Bauer S, et al. The multimodal brain tumor image segmentation benchmark (BRATS). *IEEE Trans Med Imaging*. 2014;34(10):1993-2024. https://hal.inria.fr/hal-00935640.
82. Julkunen P, Korhonen RK, Nissi MJ, Jurvelin JS. Mechanical characterization of articular cartilage by combining magnetic resonance imaging and finite-element analysis–a potential functional imaging technique. *Phys Med Biol*. 2008;53(9).2425-2438.
83. Tobon-Gomez C, Geers AJ, Peters J, et al. Benchmark for algorithms segmenting the left atrium from 3D CT and MRI datasets. *IEEE Trans Med Imag*. 2015;34(7):1460-1473.
84. Calkins H, Kuck KH, Cappato R, et al. 2012 HRS/EHRA/ECAS expert consensus statement on catheter and surgical ablation of atrial fibrillation. *Europace*. 2012;14(4):528-606.
85. Krizhevsky A. One weird trick for parallelizing convolutional neural networks; 2014. arXiv preprint arXiv:14045997.


**How to cite this article:** Guedria S, De Palma N, Renard F, Vuillerme N. R2D2: A scalable deep learning toolkit for medical imaging segmentation. *Softw Pract Exper*. 2020;1–20. https://doi.org/10.1002/spe.2878