MEDAS: an open-source platform as a service to help break the walls between medicine and informatics

Liang Zhang[1] · Johann Li[1] · Ping Li[2] · Xiaoyuan Lu[2] · Maoguo Gong[1] · Peiyi Shen[1] · Guangming Zhu[1] · Syed Afaq Shah[3] · Mohammed Bennamoun[4] · Kun Qian[5] · Björn W. Schuller[6,7]

Abstract

In the past decade, deep learning (DL) has achieved unprecedented success in numerous fields, such as computer vision and healthcare. Particularly, DL is experiencing an increasing development in advanced medical image analysis applications in terms of segmentation, classification, detection, and other tasks. On the one hand, tremendous needs that leverage DL's power for medical image analysis arise from the research community of a medical, clinical, and informatics background to share their knowledge, skills, and experience jointly. On the other hand, barriers between disciplines are on the road for them, often hampering a full and efficient collaboration. To this end, we propose our novel open-source platform, i.e., MEDAS–the MEDical open-source platform As Service. To the best of our knowledge, MEDAS is the first open-source platform providing collaborative and interactive services for researchers from a medical background using DL-related toolkits easily and for scientists or engineers from informatics modeling faster. Based on tools and utilities from the idea of RINV (Rapid Implementation aNd Verification), our proposed platform implements tools in pre-processing, post-processing, augmentation, visualization, and other phases needed in medical image analysis. Five tasks, concerning lung, liver, brain, chest, and pathology, are validated and demonstrated to be efficiently realizable by using MEDAS. MEDAS is available at http://medas.bnc.org.cn/.

Keywords Deep learning · Medical imaging · Platform · Digital health

Liang Zhang and Johann Li have contributed equally to this work and should be considered co-first authors.

Liang Zhang
liangzhang@xidian.edu.cn

[1] Xidian University, Xi'an, China

[2] Data and Virtual Research Room, Shanghai Broadband Network Center, Shanghai, China

[3] College of Science, Health, Engineering and Education, Murdoch University, Perth, Australia

[4] School of Computer Science and Software Engineering, The University of Western Australia, Crawley, Australia

[5] School of Medical Technology, Beijing Institute of Technology, Beijing, China

[6] GLAM - Group on Language, Audio & Music, Imperial College London, London, UK

[7] Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Augsburg, Germany

1 Introduction

Deep learning is the present cutting-edge technique in computer vision, natural language processing, and other areas, particularly healthcare. Thanks to its power, researchers can use a regular pipeline to process and analyze the data and then obtain excellent results with the aid of deep learning. For instance, there are a lot of recent studies that apply deep learning in their research, especially medical image analysis [20, 38, 49, 87, 97, 100]. However, most researchers, who use deep learning in their research on medical image-related tasks, are professionals in computer science, and not medicine. Due to the lack of computer-related knowledge, it is hard for medical researchers to understand and apply deep learning in their research individually for tasks such as tumor segmentation and nuclei classification. As to computer science researchers, they cannot amply analyze their results without the help of medical researchers. This gap between computer science and the medical field creates a bottleneck for the use of deep learning in medical image analysis.

The program, which is designed by the programmer, is a series of instructions to operate the hardware. Programming is a skill to convert what they want to do into instructions, like the metaphor. However, directly operating the hardware with instructions is very difficult for most people. Thus, there are many concepts created, such as subroutine, dynamically linked libraries, sharing objects, compiler, and framework. These concepts catch the importance of wrap and reuse. Programmers could avoid building everything from scratch step by step but focus on what they should focus on instead. TensorFlow [1], ITK [33], and OpenCV [65] are typical examples to help researchers simplify the implementation of their program in deep learning and image analysis.

In medical areas, ITK [33], ANTs [4], FSL [35], Deep Neuro [6], and NiftyNet [24] are the prevalent toolkits, libraries, and frameworks to help researchers develop programs to analyze medical images and data. These tools, libraries, and frameworks can help them to register, process, visualize, and analyze images. However, it still requires a significant level of programming skills for interested medical researchers to apply them in their researchers.

Generally, when using the computer as a tool to solve a problem, the approaches can be categorized into three levels. The **first level** is to control the computer by programming from bottom to top; the **second level** is to solve the problem by combining other libraries via programming; the **third level** is to interact with out-of-the-box software via the user interface. The first two levels require expert programming skills, and that limits the usage of the computer from non-professionals. This creates a challenging situation for these studies.

However, when we take a closer look at the most use-cases of deep learning-based medical image analysis, one easily sees that pre-processing, augmentation, neural networks, post-processing, visualization, augmentation, and debugging are commonly used pipelines, and the medical researchers are not the ones to create these tools but the ones to set up the parameter and to use them. Furthermore, those researchers could simply combine these tools and make up their models without programming when applying deep learning in their studies with visualization programming.

Nevertheless, all these frameworks and toolkits are not integrated as a system. Researchers need to assemble their programs from here and there one by one with their programming skills, unlike out-of-the-box tools, for example, Microsoft Excel and IBM's SPSS. In order to help researchers build deep learning models easily, the **MEDical open-source platform As Service** (**MEDAS**) is proposed in the oncoming. MEDAS provides a collaborative and interactive platform that allows researchers to work together to build their algorithms by coding or visualization programming.

The main idea of MEDAS is to provide a scalable platform and integrate a set of tools to cover the implementation of deep learning models for medical image analysis. Moreover, MEDAS not only provides tools and utilities, functions, and modules commonly used in deep learning but can also help researchers to manage their computing resources and refine their models. Currently, MEDAS primely provides tools and components for the classification, detection, and segmentation tasks of the MRI images, CT images, and pathology images, respectively.

We organize the remainder sections as follows. Section 2 introduces the related work on deep learning, medical image analysis, Docker, and other technologies. Section 3 expounds on our main idea of rapid implementation and verification, i.e., **RINV**. Section 4 discusses the main components that MEDAS provides for users to implement their algorithms and models. Section 5 introduces the utilities that MEDAS provides to simplify programming, management, and refining. Section 6 introduces several case studies of MEDAS, including *pulmonary nodule detection & attribute classification*, *liver contour segmentation*, *multi-organ segmentation*, *Alzheimer's disease classification*, and *nuclei segmentation*. Finally, Section 7 provides a discussion of open questions, and Section 8 concludes this paper.

2 Related work

In this section, we introduce (1) the related toolkits and software for medical image analysis, (2) the deep learning frameworks used in most relevant works, and (3) other related technologies and software.

2.1 Toolkits of medical image

For analysis of the medical images, many institutions, companies, and researchers created toolkits, and we list some commonly used of them.

ANTs Advanced Neuroimaging Tools [4] is a toolkit for brain images and provides functions to visualize, process, and analyze the multi-modal image of the brain.

FreeSurfer FreeSurfer [22] is an open-source toolkit for processing and analyzing MR images, and it includes functions about skull stripping, image registration, subcortical segmentation, cortical surface reconstruction, cortical segmentation, cortical thickness estimation, longitudinal processing, fMRI Analysis, tractography, and GUI-based visualization.

ITK Insight Segmentation and Registration Toolkit [52] is the most popular toolkit widely used in medical image analysis. The functions provided by ITK include basic operations of medical images, visualization, pre-processing, registration, and segmentation. It is implemented with C++ and offers templates and bindings for Python, Java, and other languages.

2.2 Deep learning-based medical image toolkits

We listed some toolkits and software based on deep learning and focused on medical image analysis.

DeepNeuro DeepNeuro [6] is an open-source toolkit, which provides out-of-the-box algorithm modules and applications based on deep learning.

MIScnn Medical Image Segmentation with Convolutional Neural Networks [62], which was released recently, targeted medical image segmentation based on convolutional neural networks and deep learning. It provides pipelines and programming-based methods to help users to create their dedicated models.

NiftyNet NiftyNet [24] is another open-source toolkit, similar to DeepNeuro, and provides a series of components such as dataset splitting, data augmentation, data processing, pre-designed networks, and evaluation metrics. NiftyNet aims at medical image analysis with deep learning.

2.3 Deep learning frameworks

Deep learning frameworks can help researchers to avoid wasting time on the implementation and verification of the algorithms for the low level. Here, we list the most popular deep learning frameworks.

Caffe Jia et al. created the Caffe framework [36], which is an abbreviation for Convolutional Architecture for Fast Feature Embedding. It provides a useful open-source deep learning framework to fill the gap between different devices and platforms.

PyTorch Facebook released Torch—a scientific computing framework. It widely supports machine learning algorithms on the GPU. A few years later, Facebook released another deep learning framework, named PyTorch [65, 66], which puts Python first. Now, it is one of the most popular deep learning frameworks for researchers.

TensorFlow Google released a deep learning framework named TensorFlow [1], aimed at tensor-based deep learning. TensorFlow is based on dataflow graphs and can run on different devices, including CPU, GPU, and Google's TPU. TensorFlow is widely used in both research and industry because it can run on scaled from a personal computing device to server clusters. Moreover, Google also open-sourced several tools for TensorFlow, for example, TensorBoard.

2.4 Docker and visual programming

Docker [32] is a kind of container platform and also is an industrial-level resource management solution. Docker takes on the management task of computing resources, which frees its users to focus on their researches. It allows containers to be launched in a short time, and it also allows the mass of applications to run on the host and keep the host without any affection.

NVIDIA released nvidia-docker [78] in 2015, which makes it possible to use a CUDA-enabled GPU in Docker containers. In this way, researchers can use the GPU to accelerate algorithms in Docker.

Kubernetes [90] is one of the most famous Docker cluster management software, which can save one from managing a lot of workstations or servers. Users could just submit their tasks, run them on a machine, and supervise them on a web-based user interface.

Visualization programming allows users to create programs by manipulating program pipelines graphically or by drag-and-drop elements, such as Unreal Engine's Blueprints Visual Scripting [56] and Scratch [55]. That allows naive programmers or researchers not familiar with programming to build deep learning models quickly by drag-and-drop operations.

3 Rapid implementation and verification

The naive motivation behind MEDAS is to make the application of deep learning easier for both computer and medical researchers in their works. The applications of deep learning-based methods require many computer-concerned skills and knowledge. To be able to use these methods, researchers need to know how to configure the software and hardware, how to program based on the libraries and frameworks, and other advanced operations. Thus, we provide MEDAS as an out-of-the-box software and aim to provide a way of implementation and verification but hiding the details of configuring low-level software and hardware.

Such an idea to implement and verify a model is called "**Rapid Implementation aNd Verification**" (**RINV**). RINV aims at the workflow from the sketch to the final program and results. **Based on this idea, MEDAS provides tools and utilities to help researchers simplify the implementation and verification to focus on the research of the model and algorithm**. We introduce RINV in this section, and Sect. 4 & 5 present the tools and utilities based on RINV.

Just like a medical researcher does not have to build a CT scanner before he or she wants to scan, they should not be required to spend unnecessary time on the implementation of deep learning algorithms before using it, either. Most of the algorithms and mathematical models are a combination of sub-algorithms, sub-pipelines, and other models. For example, as shown in Fig. 1, a type of the "convolution block", which is widely used in deep learning, is combined by a series of sub-layers. Thus, for medical researchers without the knowledge of deep learning and computer science, combining the existing models and setting up the parameters is the best way to apply deep learning methods in their research. One example is simply dragging and dropping with visualization programming.

The patchwork of algorithms and models only provided a way to simplify the implementation. However, to drive them to work, the hardware and software should also be configured and managed correctly, besides the implementation. That is called "resources auto-management".

Generally, there are a lot of steps involved to convert an idea, a formula, or a model to a program or even a basic system. Such a process of transforming can be split into four tiers, as shown in Fig. 2. At ***tier one***, researchers need to do everything by themselves. They need to implement and verify the algorithms with C++ and assembly, convert mathematical formulas to a program, make sure the program runs on the correct device, manage computing resources, visualize results, and so on. At ***tier two***, researchers can use naive algorithm toolkits to implement the complex program but still need to manage the device resources manually. At ***tier three***, the management of computing resources should be handled automatically. ***Tier four*** aims to convert mathematical formulas to results directly.

Tier four is a moonshot, but still a utopian design. However, researchers mostly prefer tier four, because it is not required with coding but can get results easily. **Our aim for MEDAS is to achieve functions of tier three**, which can provide efficient tools for users to implement and verify their models and algorithms and help them manage their resources efficiently.

Back to verification, it is different from implementation and testing in software development. The verification of the deep learning-based methods and applications focus on two points, (1) the evaluation of results by metrics and visualization and (2) the interpretability. Therefore, we add the visualization, analysis, and interpretability function into the MEDAS.

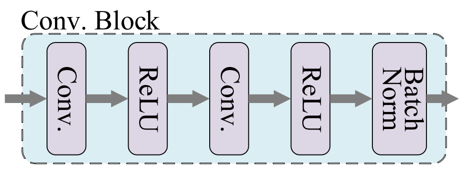


Fig. 1 A version of the "convolution block". It is combined with convolution layers, ReLU active functions, and batch normalization layer

3.1 Why RINV works?

MEDAS focuses on the application of deep learning-based algorithms to medical image analysis, and the prime focuses are the classification, detection, and segmentation tasks of medical image analysis. The codes of these tasks are sharing a similar architecture. There are three key parts of the codes: (1) how to process medical data, (2) how to design and train their model, and (3) how to optimize parameters.

For non-computer researchers, the first two challenges to run their codes are (1) to configure the environment of their computer and (2) to write the codes, particularly with non-deep-learning parts. MEDAS can help them configure the environment, and MEDAS can also avoid coding repetitively by reusing the tools provided by MEDAS, for both computer and non-computer researchers. MEDAS models the program to train a deep learning algorithm into seven parts: datasets management, pre-processing, data augmentation, neural network, post-processing, visualization, and training components. MEDAS provides components of these parts to let researchers reuse them so that their implementation and verification of their algorithms can be simplified. The details of these components are shown in the following sections.

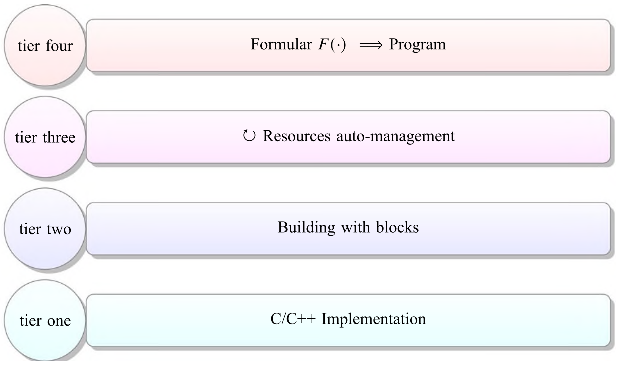


Fig. 2 There are four tiers of deep learning model development. Tier one is the implementation with C/C++ and assembly, such as for cuDNN [13]. The next tier is the combination of the basic blocks, for example, by using TensorFlow or PyTorch. Tier three includes the management of resources to help users focus on the model itself. Tier four aims to convert formulas to a program directly, meaning the implementation and verification are automatically completed by the software

4 Core: tools of deep learning

Similar to existing toolkits and frameworks, MEDAS provides a series of tools to allow users to create algorithms and models with the idea of “rapid implementation and verification” by the combination of bricks. We introduce these tools in this section, and the whole architecture and other utilities of MEDAS present in Sect. 5.

After analyzing the pipeline of deep learning from our and others’ researches [10, 15, 29, 47, 71, 72, 85, 102, 103], we found that the pipeline in these studies shares similarities. The workflow of medical image processing is relatively fixed. For most algorithms and methods, for example, graph cut, the workflow usually includes:

- Dataset management
- Pre-processing [5, 38, 64, 91, 99]
- Augmentation [38, 50, 67, 83]
- Kernel algorithm
- Post-processing [38]
- Visualization [101] and other operations

Each step or component has its purpose of processing, and the importance of these pipelines is obvious. Figure 3 shows the workflow of the typical deep learning-based medical image processing pipelines.

MEDAS implements a series of tools to meet these requirements, including pre-processing, post-processing, data augmentation, artificial neural network, visualization, and other tools.

4.1 Pre-processing

As its name implies, pre-processing is the step before the training of neural networks and includes feature processing and data processing. The typical example of feature processing includes feature extraction, noise reduction, data normalization, and modalities registration, while the typical data processing contains format conversion, annotation transformation, and others. We implement the necessary tools to help researchers process the data before they train their models.

There usually exists a data bias in medical images. For radiography, such as CT and PET, the images are noisy due to the different pieces of equipment, operators, and protocols [73, 80]. Therefore, MEDAS implements the commonly used registration tool and N4 bias field correction tool [94] to process data.

For pathology, the difference in stain concentration and brands might cause different results in images [54, 74, 77]. Thus, the stain normalization tool [95] and the stain deconvolution tool [77] are applied.

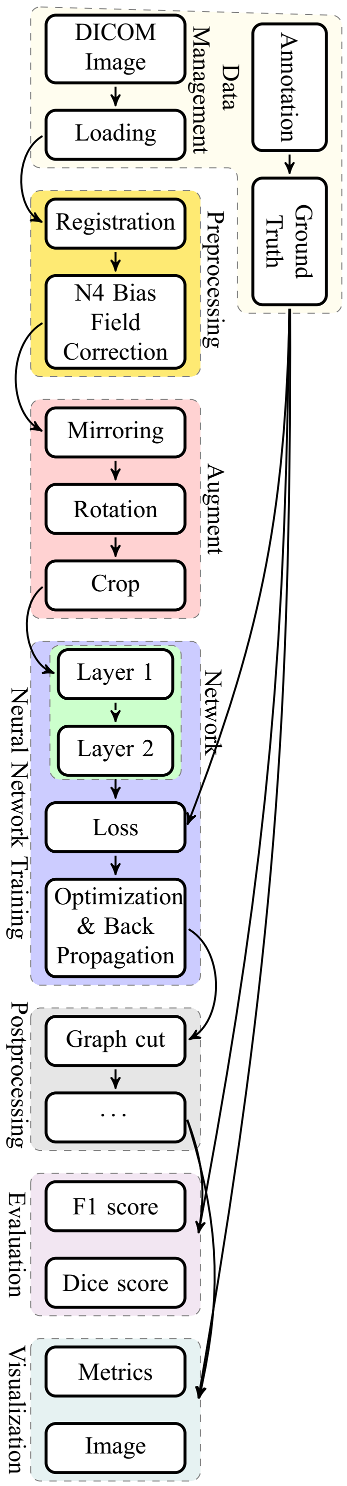


Fig. 3 The general flow of an application of deep learning for MRI image analysis. The flow shows pipelines and components about pre-processing, post-processing, augmentation, evaluation, visualization, and others

Furthermore, for general purposes, the normalization tool, resample tool, rescale tool, mask generating tool, resize tool, and other tools are implemented. To process data files, MEDAS also implements a series of tools, including format conversion, annotation conversion, and others.

4.2 Augmentation

Usually, the scale of datasets in the medical areas is considerably smaller than in others [7, 44, 81]. The public medical image datasets generally have 100 to 1000 cases, while other datasets—for example, for 3D object detection [98]—usually feature thousands and even millions of data. Therefore, augmentation is necessary to enlarge the size of the dataset. Medical image datasets "always" lack data, compared to other areas, because it takes too much time, cost, and manpower to collect and annotate medical images.

Augmentation is an efficient method to make the model more robust, not only in medical image analysis but also in other areas. Augmenting with mirroring, rotating, cropping, and deep-learning-based methods, for example, Generative Adversarial Network, are frequently used. Augmentation diversifies the data by making it look "different"—which can improve the model performance [25, 60]. The key to augmentation is that the distribution of data expands so that the robustness of the model increases.

MEDAS provides general transformation tools, Gaussian random noise, rescaling, and others. Gaussian random noise uses noise to enhance the robustness of the model, while some tools desensitize the noise of the scale and the bias by resampling and transforming the distribution of the data.

4.3 Artificial neural network

The neural network is the most important part of deep learning. MEDAS provides several tools integrated with different types of neural networks for training and inferring. Meanwhile, MEDAS plans to integrate a neural architecture search tool, which aims at automatically designing neural networks for specific tasks.

The neural network (model) training is a fixed workflow, which includes forward propagation, loss calculation, and backward propagation [46] and is built by connecting "blocks" such as "max-pooling layer", "convolution layer", "fully connected layer", "ResBlock", "Dense Block", and so on [2, 28, 31, 43, 76, 84, 88]. Loss function influences the search in the parametric space, and the different loss functions meet the different tasks. As the neural network intends to be applied merely as a tool by medical researchers, they are considered to be users and not developers. Therefore, the tools with pre-designed models can be the best choice and can meet the needs of researchers who want to focus on the application side of matters.

Since a few neural networks have achieved significant success in many medical image analysis tasks, MEDAS implements those networks as tools for segmentation, classification, and other tasks. For instance, the 3D Mask RCNN [27] and 3D Dual-Path Net [12] are integrated for the detection and classification tasks on radiography images. The U-Net [76] and V-Net [58] are integrated for the segmentation task. Besides, the U-Net is also available to be used in classification tasks. The other similar neural networks are also integrated for these tasks.

Though the prime framework currently supported by MEDAS is PyTorch, MEDAS also supports other frameworks, such as TensorFlow. MEDAS implements the compatibility layer so that the heterogeneous models can be trained, respectively. When users need to reuse their trained model, users need to load the parameter of the model saved in the step of training and execute the model. MEDAS will manage models which are encapsulated as tools, and the parameter of the model is archived in the storage of MEDAS.

4.4 Post-processing

Post-processing is a strategy that can improve the result. For segmentation tasks, post-processing can make predictions more "smooth". For example, [26] employed an FCN-based neural network, which is simpler to UNet and VNet, but achieves better performance compared with pure UNet. The key to its success is post-processing. It uses "horizontal and vertical gradient maps", "energy landscape", and other features in the post-processing and then use the watershed algorithm to process. Furthermore, the Conditional Random Field [11], Graph Cut [37], and other traditional algorithms can also be used as post-processing to optimize the results of a neural network.

In a few cases, the output of the neural network is a probability or a probability map. The tools, for example, binary normalization, can be used for the classification and segmentation tasks, which will reach better results compared to a simple threshold.

Besides the post-processing tools introduced above, the MEDAS also provides another series of post-processing tools for the neural network itself. The model compression and pruning tools can help researchers generate smaller but faster models with better accuracy. MEDAS employed the following tools:

- Parameter pruning and sharing [14, 19, 45, 96]
- Low-rank factorization [18, 89]
- Knowledge distillation [16, 48]

4.5 Visualization

Generally speaking, the visualization can be categorized into result visualization, metric visualization, and analysis visualization.

The result shows the neural network output and keeps important links between the model and the clinic side [30, 34, 51, 101, 104]. The input and output of the neural network in the medical image analysis, are not the color-based 2-D images. It is hard to show them directly, in particular when we want to analyze the relationship between the input and the output. Thus, a well-designed tool of visualization can help users present and analyze their researches corresponding with the clinical aspects, such as segmentation visualization, mesh-based image visualization, point cloud-based lesion visualization, and others.

For metric visualization, MEDAS implements tools to record the metrics and visualize them as the image, for example, the loss visualization tool.

For analysis visualization, MEDAS implements a series of tools for different kinds of tasks. The saliency visualization, attention visualization, feature visualization, gradient propagation visualization, t-SNE visualization, sensitivity analysis, and other visualization tools are implemented.

4.6 Others

MEDAS also includes other tools, for example, dataset management. The dataset management tool aims at the management of the dataset. For example, if one wants to split one's data into a training set and a testing set, one can use the dataset split tool.

5 Architecture of MEDAS

Different from traditional toolkits or frameworks, MEDAS is a system but not just a collection of functions and tools. Researchers can only utilize traditional toolkits and frameworks via programming, but MEDAS provides visualization programming to help researchers intuitively and easily implement their algorithms and models. In the following subsections, we discuss the visualization-based programming, auto-machine learning, Python API, and resource management, and other features of MEDAS.

Figure 4 shows the architecture of MEDAS. From the bottom to top, the figure depicts each component of MEDAS, including: **visualization programming**, **Python API**, **tools**, **auto-machine learning**, and **resource management**. The users can interact with MEDAS via Python API or visualization programming, while the latter provides more functions integrated by MEDAS, such as auto-machine learning. Resource management is a part of MEDAS does not provide any application programming interface. The resources management controls the tasks scheduling and device allocation, which directly interact with the machine.

Moreover, we introduce the technical details of the implementation of MEDAS.

5.1 Visualization programming

Until the invention of the graphical user interface (GUI), anyone who wanted to use a computer needs to operate the machine by itself or the professional operator. During this time, the operators were the experts of computers who dressed up in formal attire and worked in a specific room to handle the science problems from other scientists, and the interfaces of the computer were the teletypewriter-based terminal or the monitor-based terminal. The computer has its own rules, but these rules broke after the rise of the GUI. Software developers convert instructions from the "human rules" to "computer rules", which is called "implementation". GUI-based software can efficiently help non-professionals to translate their ideas from "human rules" to "computer rules", to execute them, and to show the results of the execution.

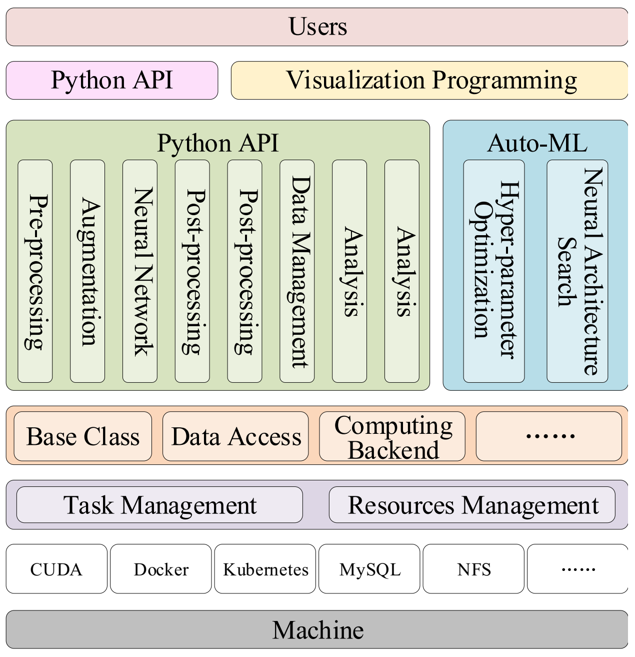


Fig. 4 The general architecture of MEDAS: from the bottom (machine) to the top (user). The user can use MEDAS and its tools (Sect. 4), auto-machine learning (Sect. 5.2), resources management (Sect. 5.4), and other components via Python API (Sect. 5.3) or visualization programming interface (Sect. 5.1).

Usually, a GUI is considerably more intuitive than a Command Line Interface (CLI) or any text-based interface—especially, for the people not or less familiar with computers. If well designed, GUI can render the operation of tools simple, visualization of results more accessible and the users efficient, but CLI cannot.

MEDAS provides a web-based interface for researchers to manage and browse their tasks and data. The interface includes a visualization programming module, where researchers can implement their models by dragging, dropping, and connecting. Based on the website, the researchers can access MEDAS anywhere with the Internet, and it is client-free, but a web browser suffices.

5.2 Auto-machine learning

The backbone design and hyper-parameters search are the key to deep learning to the current state-of-the-art. However, the design and refinement of the model are not trivial. Therefore, MEDAS integrates auto-machine learning utilities.

Optimizing the hyper-parameters of the deep learning models is not a straightforward task and requires in-depth expertise. Generally, the parameter θ of a deep learning model $f(x;\theta)$ can be optimized by gradient descent. However, the hyper-parameter needs to be optimized manually, and the model also needs to be designed by hand. The first challenge of hyper-parameters optimization is its search space. The hyper-parameters include discrete and continuous values, and the relationship between parameters and results cannot be formulated in a closed way to obtain an analytic solution. Thus the search space is too huge to set up and search in, directly. Besides, the second challenge is that we could only change some of the hyper-parameters after hours or days of training. Therefore, the optimization of hyper-parameters needs to take days or weeks, and several attempts to choose a not bad hyper-parameter set. The third challenge is that the models for different medical image tasks are different, so the distributions of hyper-parameters are changed with different models, which means there is no general searching algorithm to find the best. These challenges make it difficult to optimize the hyper-parameters. The optimization is a kind of alchemy, which lacks regular rules. With that in mind, MEDAS employs automated hyper-parameter optimization based on Bayesian Optimization.

When we optimize the hyper-parameter Θ of the model $f(x;\theta)$, we actually need to optimize another model $F(\Theta;f)$, which represents the best score of the metric for the function f with the hyper-parameter Θ, to obtain the optimal hyper-parameters. For optimization, $\arg\max F(\Theta;f)$, it is hard to deduce the analytical formula of $F(\cdot)$; hence, we use a set of functions $\{\mathcal{F}\}$ to estimate

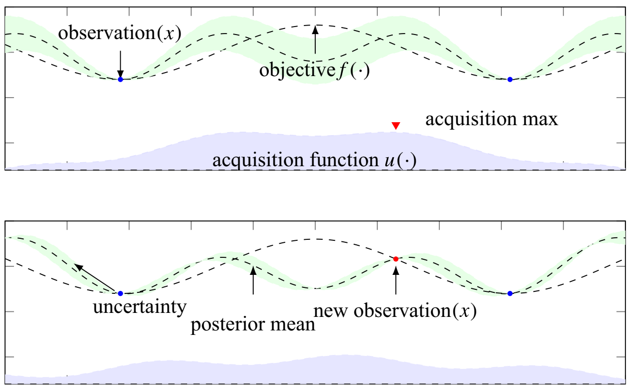


Fig. 5 The principle of the provided Bayesian-based hyper-parameter automatic search. The above figure shows the prediction of hyper-parameters at $t = t_i$. Two blue points show the observation x; the black line presents the posterior mean of the prediction; the dashed line is the objective function $f(\cdot)$; the green area represents the possible functions, while the blue area is the acquisition function $u(\cdot)$. The maximum point of $u(\cdot)$ is the next point of the hyper-parameter to be optimized. We use a set of the sine function to explain how Bayesian optimization searches the hyper-parameter. The key idea of Bayesian optimization is the iterative repetition of fitting and search. The methods, such as Gaussian process and regression random forest, are employed for fitting the data (x, y), where x denotes the hyper-parameter Θ, and y denotes the performance of the model, i.e., $F(\Theta,f)$. The acquisition function, such as Expected Improvement and Upper Confidence Bound, is employed for searching the next best x of the model

the distribution of $F(\cdot)$ as Fig. 5 shows. After training the original model and getting the hyper-parameter result of $F(\cdot)$, we can remove the functions which do not fit the result. Then, we get a subset $\{\mathcal{F}\}_i$. After several iterations, the distribution of $\{\mathcal{F}\}$ approximates the final one. Ultimately, we can obtain an approximation of the optimal hyper-parameters.

5.3 Python API

5.3.1 Data, format, input, and output

Different modalities of the medical image have different formats. Therefore, the tool employs SimpleITK and OpenSlide [79] to handle the different formats of medical images. Furthermore, MEDAS can load and save Portable Network Graphics images (both single images and series of images) and Numpy objects.

Plug and slot

The inputs to a tool might be all kinds of files, numbers, or just a Numpy array. Therefore, MEDAS employs "plug" and "slot" to process these inputs with differentiation and to deliver them to the kernel function with assimilation. The plug takes charge of the process of inputs, while the slot handles the inputs and passes them to the kernel function, where computing. The plug automatically converts the formats of inputs. For example, when the input is a string, but the tool accepts a float number, the plug will try to parse the string.

Constructor

Similar to the input, the output also has different kinds of formats. Therefore, MEDAS employs "constructor" to process the result of the kernel function. The constructor converts the results to different kinds of formats, including DICOM, NIfTI, and Numpy array. The variable simply passes through the variable constructor to the following modules, while the image constructor saves the data to an image file or passes it to the following modules.

5.3.2 Computing backend

MEDAS employs Numpy, OpenCV, and other libraries to implement algorithms, but not C/C++. The low-level algorithm's implementation is not a high priority, due to the lack of time and manpower. However, there is a reserved feature—"computing backend", inspired by TensorFlow's design. The implementation of a faster version with CPU, GPU, and FPGA, or other devices can be added to the system via the "computing backend", at later development, and different backends can be selected when executing the instance initialization.

5.3.3 Continuous programming

Inspired by Either Monad in Haskell [57, 70], MEDAS implements an abstract class named "Either", which aims at processing the results and errors. "Either" of MEDAS has two states: success and failure, just like the one in Haskell. The tools execute one by one, and only if the previous execution is successful, the current one is able to execute. For example, setting up parameters must be done successfully before calculating.

5.3.4 Others

Logging MEDAS employs a flexible logging system, which can output to a terminal or stored in the system. Such a logging system supports users to monitor, diagnose, and debug models flexibly.

Testing suit MEDAS provides a small kit for testing, by which modules included in MEDAS or third-parties can be well tested. At the same time, we employ tools to test MEDAS automatically, which is known as continuous integration.

5.4 Resource management

Resource management is important in deep learning, medical image analysis, and other similar tasks. Let us discuss this kind of situation. When a researcher uses one computer with one GPU, the management means execution and termination by the researcher. When two researchers share one computer with a GPU, communication between the two researchers is needed for the scheduling of individual tasks. When several users share GPU clusters, the situation rapidly becomes complicated. One may easily imagine a typical scenario where every user wants to use more resources and complete their tasks as quickly as possible.

The computing resources include not only GPUs, but also storage, memory, bandwidth, software, and even energy. Cloud computing, grid computing, IaaS, PaaS, SaaS, and CaaS[1] are the concepts presented to solve the problem of resource management. Task-based scheduling can meet the demand for resource management of deep learning when the GPU, CPU, memory, and disk are considered as the main resources.

The management of resources usually includes task management and device management, as shown in Fig. 4. The task management takes charge of the scheduling, while the device manager is in charge of controlling and organizing the hardware.

5.5 Implementation details

In this section, we introduce the technical details about MEDAS so that the design ideas of MEDAS will be more reproducible.

User Interfaces

MEDAS is a web-based system, so all the interaction is based on web pages. We use vue.js, a front-end framework, to create a web-based.

Back-end

The programs, who manage the data, tasks, and resources, are mainly written in Java with SpringBoot and MyBatis, and at the same time, the non-core microservers are written in Go with Iris. For the programs related to deep learning and medical image programs, we use Python to implement referred to relevant papers with PyTorch, Numpy, SimpleITK, OpenCV, and other software.

Tasks and Resources Management

We set the basic units of task scheduling as containers with resources limitations, and the management assigned the containers to users to execute their programs. The management employs Docker and Kubernetes to manage containers and resources, and the back-end of MEDAS communicates with Kubernetes to allocate containers.

[1] IaaS is the abbreviation of "Infrastructure as a Service"; PaaS is the abbreviation of "Platform as a Service"; SaaS is the abbreviation of "Software as a Service"; and CaaS is the abbreviation of "Container as a Service".

Docker containers use the "control group" to establish a sandbox with resources limitations. The number of GPUs is controlled with a different set of the plan according to the calculation scale. Docker and Kubernetes control the device management. For storage, we employ NFS for containers managed by Kubernetes to store data.

6 Application case studies

In the previous sections, we introduced the tools and systems of MEDAS. In this section, we present different case studies performed using MEDAS and selected varying themes of tasks. Deep learning-based methods are employed throughout these case studies. The following subsections present these cases which were executed on the MEDAS system. These case studies include:

- Pulmonary nodule detection & attribute classification
- Liver contour segmentation
- Multi-organ segmentation
- Alzheimer's Disease classification
- Nuclei segmentation

On purpose to foster comparability and reproducibility, we chose public datasets in these case studies. Each case study introduces the workflow of the model, and the pipeline is implemented with visualization programming via simple drag and drop or programming via Python API. The results of the model show in each case study. These case studies are executed with MEDAS via the container[2].

6.1 Case study 1:pulmonary nodule detection and attribute classification

The detection and attribute classification of the pulmonary nodule is a common medical image analysis task and is important for lung cancer diagnosis and clinical treatment. In this case study, we employ the neural network based on DeepLung [106], to detect and classify the pulmonary nodule. The dataset, we used in this case study to train the neural network model, is the LUNA 16 dataset, which is based on the LIDC-IDRI dataset [3].

6.1.1 Workflow

Figure 6 shows the basic workflow of this case study, which includes five parts:

[2] The container includes 6 cores of Intel® Xeon® Gold 5120 CPU, an NVIDIA Tesla V100(32G PCIe version), and 48 Gigabytes of memory.

Input The input of the whole workflow includes the CT images of the chest and the annotations. These data are stored in the network attached storage (NAS) and can be mounted to the container when needed.

Pre-processing Pre-processing tools convert the formats of the image and annotation, mask the lung area on CT, and rescale the value of the image to [0, 1].

Dataset management Dataset management split the dataset into a training set and a testing set to train and evaluate the models.

Neural network We employ 3D Mask RCNN [27] for pulmonary nodule detection, while the 3D Dual-Path Net [12, 106] is used for attribute classification.

Visualization We employ a point cloud-based nodule visualization tool to display the pulmonary nodule detected by the 3D Mask RCNN and a loss visualization tool to show the training loss of the model.

6.1.2 Implementation

Simple steps by dragging and dropping with MEDAS can implement the workflow mentioned in the previous. Then we launch the Docker container, mount data from NAS, and execute the task.

6.1.3 Result and visualization

We train the 3D Mask RCNN model and the 3D Dual-Path Net with the training set and test them with the testing set. Figure 7 presents the training loss of the 3D Dual-Path Net. The left plot shows the total loss, while the right plot presents the loss for each classifier.

3D Point cloud-based visualization

MRI and CT are dense 3D images. When we are viewing such 3D images, we can only view the cross-section of a 3D image. We, therefore, develop a cloud point-based visualization tool to visualize the segmentation result in MRI and CT. Figure 8, which is rendered via this tool, shows the result of the 3D Mask RCNN, a. k. a., the pulmonary nodules.

6.2 Case study 2: liver contour segmentation

The liver-related radiographic analysis is also a focus of the research based on deep learning-based methods. The first step of the analysis is usually the segmentation of the liver contour, so in this case study, we employ VNet [58], to segment liver contours. The public dataset LiTS [7], which is aimed at detection and segmentation of the liver and tumors, is used to train the model.

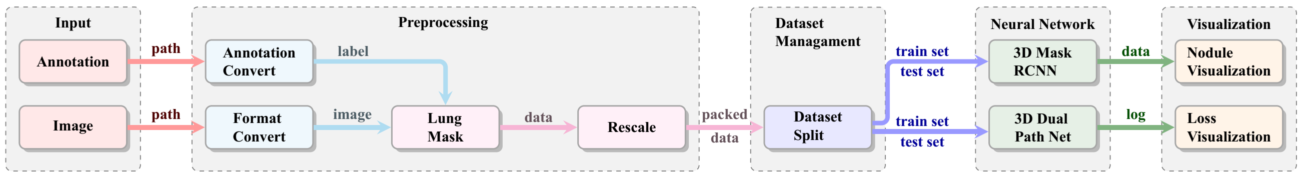
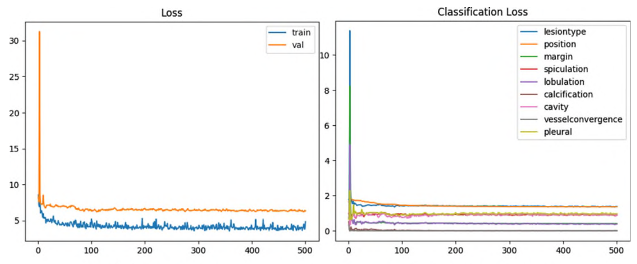


Fig. 6 Workflow and data flow of the pulmonary nodule detection and the attribute classification (case study 1). The workflow includes five parts: input, pre-processing, dataset management, neural network, and visualization. A 3D Mask RCNN is employed to detect, while a 3D Dual-path net is employed for attribute classification



(a) total loss **(b)** classification loss

Fig. 7 The total loss and separate classification loss. The *left plot* shows the training loss (*blue line*) and testing loss (*orange line*). The right plot shows the loss of different classifiers

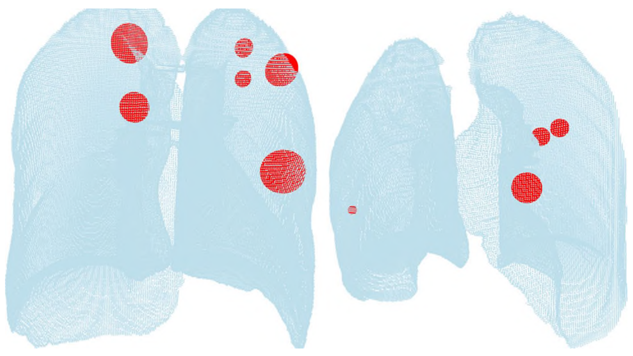


Fig. 8 The pulmonary nodules were detected in two subjects. The red marks are the detected pulmonary nodules, while the blue points are the edges of the lung

6.2.1 Workflow

As shown in Fig. 9, the workflow of this case study includes six parts:

Input The input part is the source of data.

Pre-processing We employ the pre-processing tool to convert formats of images.

Dataset management The dataset is split into a training set and a testing set by a dataset management tool.

Neural network The VNet is employed to segment the liver contours from the images and trained with the training set. Then, we use the trained model to initialize the prediction tool of the model for testing.

Visualization The training loss is visualized with the loss visualization tool, while the segmentation results are presented with the segmentation visualization tool.

Analysis The prediction and ground truth are analyzed by computing the Dice score.

6.2.2 Implementation

The algorithm can be implemented by using MEDAS's visualization programming. However, in this case study, we show the alternative option available for users to program in MEDAS. The setup, execution, and results checking with the training tool will be shown as an example.

To use the tool, there are four steps to follow:

1. Initializing instances
2. Setting up the tool
3. Executing the tool
4. Checking the results

The codes are shown in the following:

```
tool = TrainVNet()
tool.set_params(new_ct_dir = new_ct_dir, new_seg_dir =
      new_seg_dir, save_module_path = save_module_path,
      save_loss_path = save_loss_path)
tool.run()
tool.with_succ(handler)
```

With continuous programming, the code above is equal to the below one:

```
tool = TrainVNet()
  .set_params(new_ct_dir = new_ct_dir, new_seg_dir = new_seg_dir
        , save_module_path = save_module_path, save_loss_path =
        save_loss_path)
  .run()
  .with_succ(handler)
```

6.2.3 Result and visualization

The network for liver contour segmentation is trained on the LiTS dataset, and the Dice score of the model obtains 0.92 on the testing set. Figure 10 presents the results of the segmentation task, while Fig. 11a visualizes the training loss.

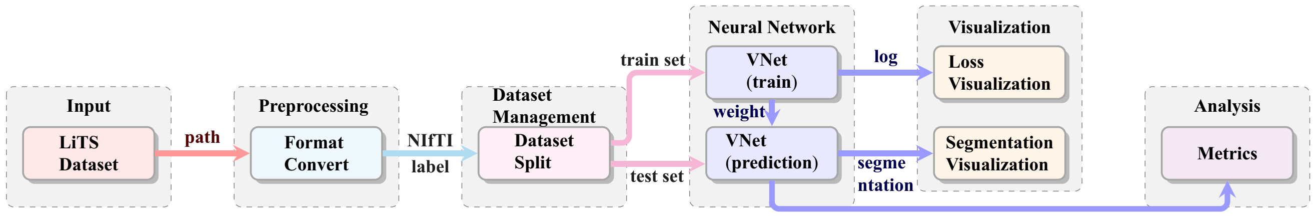


Fig. 9 Workflow and data flow for the case study 2. The workflow includes six parts: input, pre-processing, dataset management, neural network, visualization, and "analysis

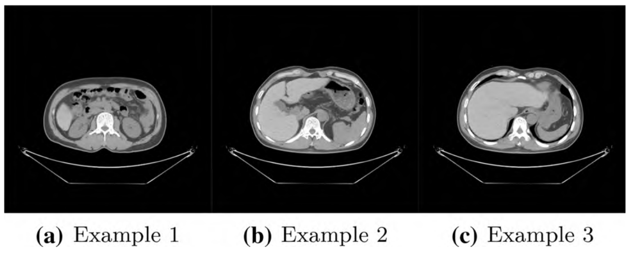


Fig. 10 The segmentation of the liver of three subjects. The window width and level of CT images are 400 and 0. The red area is of ground truth but not segmentation result; the green area is the segmentation result but not ground truth; the yellow area is the right area segmented by model

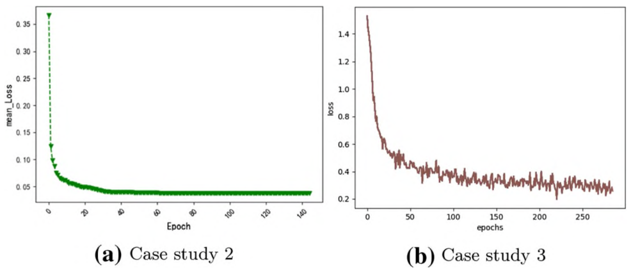


Fig. 11 Visualization of the training loss for case study 2 (*left*) and 3 (*right*)

6.3 Case study 3: multi-organ segmentation

Multi-organ segmentation can help machines understand the structure of the human body, which is very important for all the relevant tasks. Therefore, some researchers have focused on the single- or multi-organ segmentation tasks, such as the liver [21, 53], and the pancreas [9, 105]. In this case study, we use VNet-based neural network for the multi-organ segmentation task, SegTHOR [93]. SegTHOR challenge focuses on the segmentation of 4 organs at risk: heart, aorta, trachea, and esophagus. This dataset provides about 40 CT images of the chest.

6.3.1 Workflow and implementation

As shown in Fig. 12, the workflow of this case study includes six parts:

Input The input includes the images and annotations of the chest and is stored in NAS as a dataset.

Pre-processing Pre-processing tools rescale the range of the image values with a window width and a window level, resample the images to change their size.

Dataset management The dataset management tool splits the dataset into a training and a testing set randomly.

Neural network We employ a VNet-based neural network to segment organs from the chest CT images, and the model is trained and tested with the SegTHOR dataset.

Visualization & analysis The segmented images can be visualized via the segmentation visualization tool, and the result analysis tool analyzes the results and generates a report in the MS-Excel format.

6.3.2 Task management

After the user sets up and submits the task, MEDAS begins to prepare launch a docker container to execute the user's task. First, the scheduler of MEDAS checks the resource limitation of the user and system. A task will be executed only if the required computing resources are ready and the resources currently used by the user have not reached the limit of its account. Then, MEDAS encapsulates the codes and mounts the archive of code and datasets to the container. Finally, the scheduler of MEDAS allocates the computing resources required by the user, such as GPU, CPU, memory, and storage, and launches the docker container.

When there are more than one user and one GPU (or computing resource) in the system, the scheduler strategy will be complex. MEDAS will reject the task if it requires interaction, but will queue the task in line when not. For hyper-parameter searching, the new tasks will be queue only if the old ones finish. The rejection of hyper-parameter searching tasks occurs only after all parameters are reached or execution times are limited.

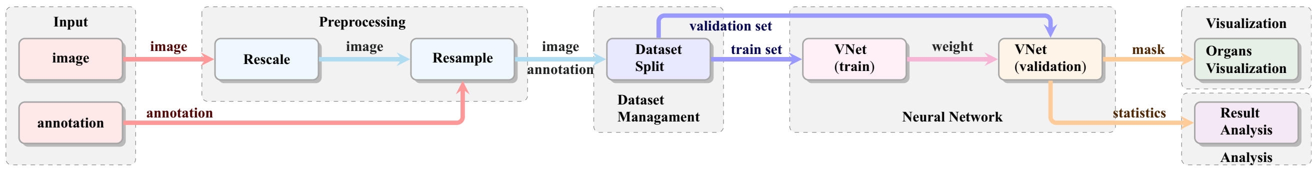


Fig. 12 The workflow of multi-organ segmentation (case study 3). The workflow includes the pre-processing of data and annotations, the training, the evaluation, and the visualization

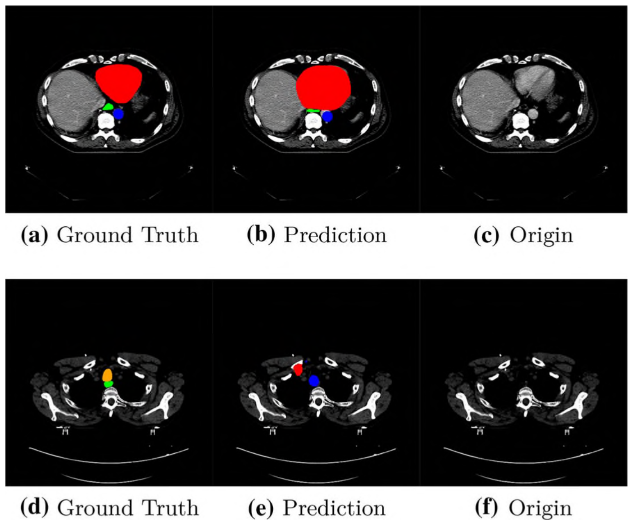


Fig. 13 Visualization of case study 3. The *green area* is the esophagus; the red area is the heart; the *blue area* is the aorta; the orange area is the trachea

6.3.3 Result and visualization

Figure 13 shows the obtained visualization results, and Fig. 11b shows the training loss.

6.4 Case study 4: Alzheimer's disease classification

Alzheimer's disease (AD) is a kind of progressive neurodegenerative disorder impairing the functions of memory and cognition according to [59]. Till now, there is no approach to cure the disease or even significantly slow down its deterioration, but there are some methods to tell the difference between AD and normal control (NC) subjects, e.g. [39–41]. In this section, we employ U-Net [76], and modify it for classification tasks, for example, AD versus NC.

In this case study, all the subjects are selected from a public AD dataset named "the Alzheimer's Disease Neuroimaging Database", i.e., ADNI [61]. We select scans of AD and NC subjects to train a classifier.

6.4.1 Workflow

As shown in Fig. 14, the workflow of this case study includes five parts:

Input The input loads the data from the dataset.

Pre-processing The pre-processing tool generates two images from one original image by selecting two voxels from a box region included 8 voxels.

Dataset management The dataset management tool splits the dataset into a training set and a testing set.

Neural network We employ a UNet-based neural network for the classification task to filter AD from NC scans.

Visualization & analysis The sensitivity analysis tool helps to identify what is relevant for the neural network by generating a heat map that shows how the neural network behaves when a patch of the image is occluded.

6.4.2 Result

We trained the model on MEDAS with the default parameters. The average accuracy of the classification task on the testing set is 0.95.

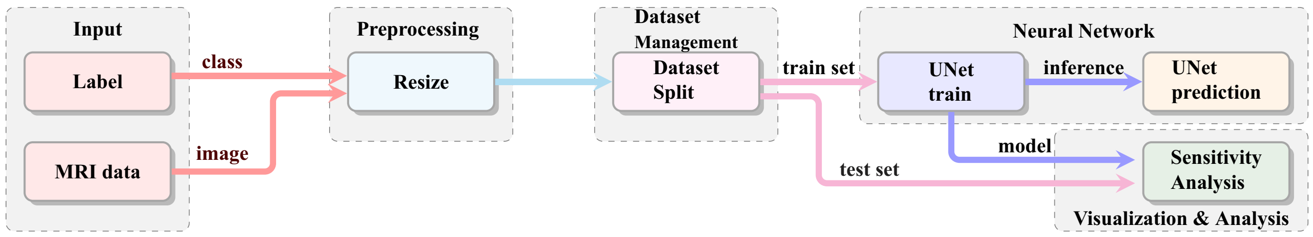


Fig. 14 The workflow of Alzheimer's disease classification, case study 4. The workflow includes the pre-processing of data and annotations, the training, the evaluation, and visualization

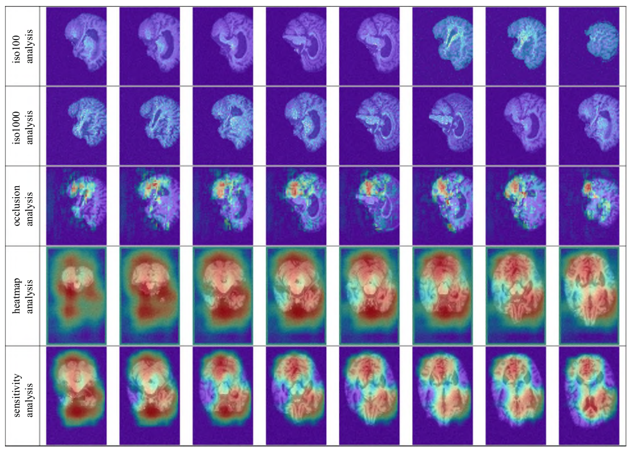


Fig. 15 The heat map generated by the tool in MEDAS with block-based and contour-based occlusions. The first two rows resemble the analysis with color spacing split into different ranges. The third row includes the analysis results with occlusion. The fourth row resembles the activation heat-map. The last row depicts the sensitivity analysis result

6.4.3 Interpretable visualization

Generally, deep learning is considered a black box. It is difficult for researchers to understand what has been learned by the neural network and why the algorithm works so well. Researchers can establish models from clear reasons and targets for traditional algorithms, but for deep learning, only a general target is selected to let gradient descent optimize their models. A general neural network model for a complex task might include more than millions of parameters that are hard to optimize and different to find out the effect of each parameter.

MEDAS employs many tools to help researchers analyze and visualize their models and results. In Fig. 15, we employ three methods to analyze and visualize the attention of our network. Such tools can easily be used for similar tasks to generate heat map-based interpretable images. Block-based and contour-based occlusions are employed to interpret our model.

6.5 Case Study 5: Nuclei segmentation

Nuclei segmentation is one of the basic tasks in pathology image analysis, whether based on traditional [69] or deep learning-based methods [63, 86, 92]. The diagnostic of pathology images is based on many terms representing objects, such as nuclei, cells, and glands. Researchers extract features from these objects and use them in further diagnosis. For example, the mitosis analysis task is based on nuclei segmentation or detection. We use a U-Net-based model [76] to segment the nuclei on the dataset described in dataset MoNuSeg [44].

6.5.1 Workflow

As shown in Fig. 16, the workflow includes six parts:

Input The input loads the data from the dataset.

Pre-processing The pre-processing tools convert formats and normalize the stain of the pathology image.

Dataset management The dataset management tool splits the dataset into two sets, while the neural network uses the training set to train the model and uses the testing set to validate it.

Neural network We employ UNet-like neural networks, including FCN, UNet, ResUNet, and DPUNet. The hyper-parameter controlled which model is used.

Post-processing The post-processing tool handles the results of the segmentation. We employ the binary normalization tool to improve the segmentation results.

Visualization The visualization tool depicts the final results to the user.

6.5.2 Implementation

After the general design of the workflow that can be done on the draft, the user can drop selected tools in the editor and connect them according to the data and control flow to implement the workflow. Then, the data is uploaded into the platform from a locally hosted or online storage system, which is connected with the annotation systems. Finally,

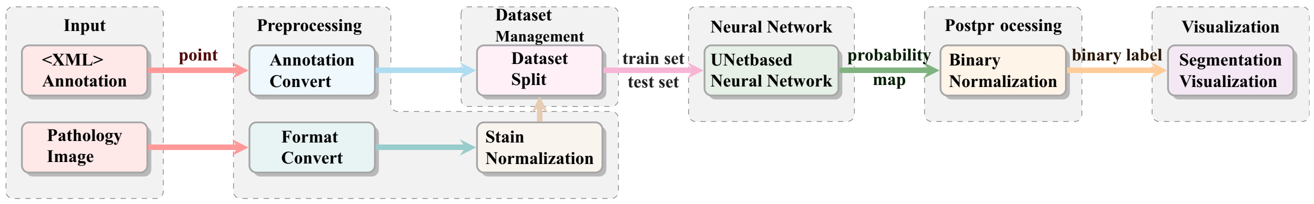


Fig. 16 The workflow of nuclei segmentation (case study 5). The workflow includes the pre-processing of data and annotations, the training, the evaluation, and the visualization

Table 1 The best results of the optimization. The max epoch, criterion, learning rate, number of training epochs, and model are selected as parameters

Epoch	Criterion	Learning rate	Model	Mean AJI
172	Dice	4.081e-3	DPUNet	0.6073

Table 2 The top-five results of manual optimization with different parameters

Epoch	Criterion	Learning rate	Model	Mean AJI
200	Dice	0.5e-3	ResUNet	0.5855
200	Dice	0.5e-3	DPUNet	0.5854
256	Lovasz	0.25e-3	ResUNet	0.5832
128	Bce	1.0e-3	DPUNet	0.5828
500	Lovasz	1.0e-3	FCN	0.5821

the task is launched on MEDAS with the given workflow, and the model is trained. Subsequently, the results and intermediate data are stored in the system.

6.5.3 Hyper-parameter optimization

This case study is an example of hyper-parameter optimization. Selected hyper-parameters in the neural network were carefully picked for optimization.

The hyper-parameters optimized include the maximum epoch of training, learning rate, criterion function, and model. The range of the hyper-parameter "max epoch" is set to be chosen within 64 to 256, while the learning rate search range is set from 0.0001 to 0.01. The criteria could be selected in dice loss (dice), binary cross-entropy (bce), and Lovász loss (lovasz), while the models could be selected in FCN, UNet, ResUNet, and DPUNet. At the same time, "mean AJI" is selected as the optimization objective.

We performed 100 iterations to search with the Bayesian optimization algorithm. The best result of the hyper-parameter optimization and the top five results by manual optimization are shown in Table 1 and Table 2. Further, Fig. 17 shows the relationship between the hyper-parameters and the metric "mean AJI". Most of the combinations with DPUNet as a model and dice as a criterion function show better performance, i.e., higher "mean AJI" score, and the scores of these combinations are between 0.5925 to 0.6075. As shown in Fig. 17, the epoch number of the

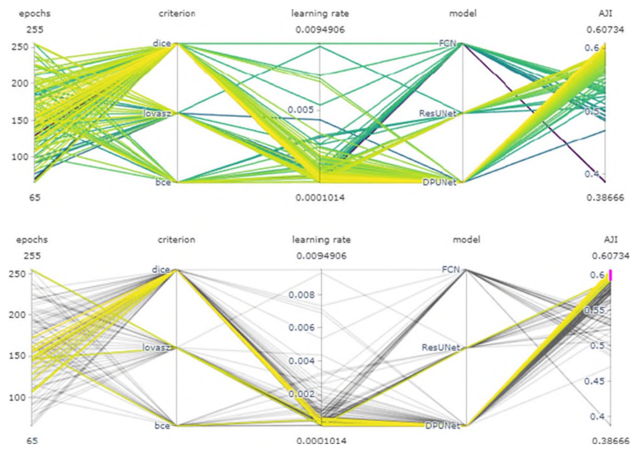


Fig. 17 The visualization of hyper-parameters via the parallel coordinates. The *top* one shows all the hyper-parameters. The color of the *lines* is related to the metric AJI: the higher, the brighter. The *bottom* one shows the hyper-parameters, whose metric AJI ranges between 0.5925 and 0.6075

training iterations does not result in a remarkable effect on the metric, compared with the criterion function and the model. Further, the smaller learning rate proves the best choice, in general.

Table 2 shows the manual optimization result as a comparison. When we try to optimize these hyper-parameters manually, we are usually facing several problems.

The most important one is how to optimize the parameters as it is difficult to find an analytical solution. As outlined, MEDAS employs the Bayesian optimization algorithm aiming to find optimal hyper-parameters.

The second problem is the time. Manual optimization needs a lot of time. After we launch the task, we need to wait for the task to finish to test another set of parameters. If we have executed a task, we cannot launch another task after the latest one has finished, because we cannot estimate when the task will finish exactly.

The third problem is the resource. Manual optimization usually needs more resources to reach a good result, since it tends to be slower and inefficient.

6.5.4 Result and visualization

The best result of the hyper-parameters is chosen as the final result. The DPUnet network is used as the model, and the dice loss is selected as the criterion function. The model is trained within 172 epochs, and the learning rate is 4.081×10^{-4}. The mean AJI score reaches 0.6073. The segmentation of the nuclei is shown in Fig. 18, while the AJI score of different organs is shown in Table 3.

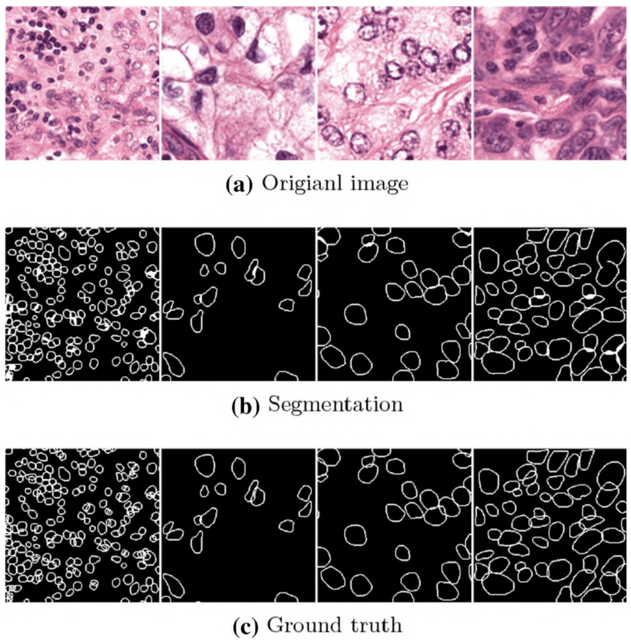


Fig. 18 Result of case study 5. Each column shows the results of four different organs. The top row is the original image with pre-processing. The middle one is the segmentation with post-processing by binary normalization, while the bottom row is the ground truth

Table 3 The AJI metric of the validation set for different organs

Organ	Breast	Liver	Bladder	Colon
AJI	0.6517	0.5310	0.6543	0.5424
Organ	Prostate	Stomach	Kidney	Mean
AJI	0.6147	0.6437	0.6135	0.6073

7 Discussion

7.1 MEDAS

Deep learning-based medical image analysis is an inter-disciplinary task, which combines computer and medicine knowledge. However, on the one hand, for medical researchers, deep learning is more like an approach applied in medical image analysis because the medicine research-ers would not know too much about it, and that is also because the wall between the computer and medicine blocks it. On the other hand, for computer researchers, such research should focus on the algorithm or models but the fact is that the most researchers spent some of their time in programming, fine-tuning, and other mechanical and repetitive tasks, on which they should not have spent too much time.

Targeting the problems above, we implement MEDAS for the idea of rapid implementation and verification. MEDAS provides a set of tools for medical image analysis, and with wrapping and reusing, researchers can simply and rapidly implement their algorithms without wasting their time on mechanical repetitive tasks. MEDAS also provides a platform including visualization programming, hyper-parameter optimization, resources management, and other components that further simplify the implementation and verification.

However, MEDAS cannot solve all the problems in the processing of applying deep learning in medical image analysis. MEDAS can remove the barriers that stop the medical researchers from applying deep learning in their researchers, and simply the implementation of algorithms for computer researchers. But MEDAS cannot remove all barriers between medical and computer knowledge. To reduce such knowledge asymmetry, we plan to create an application to let researchers share their knowledge, which is named “Knowledge Base”.

7.2 Outlook

The combination of deep learning and medical image analysis will still be a hot topic in the next few years, and a key problem in the present context is to break the wall between medicine, deep learning, and computer science knowledge. The innovation of accessible technologies and methods, like MEDAS, will help the progress in this area.

7.2.1 Automatic DL in medical informatics

Automatic DL can help researchers to automatically design models and search for the best hyper-parameters. Neural network architecture and hyper-parameter optimization search are the problems that deep learning researchers need to face. This comes, as the choice of hyper-parameters and the design of the neural networks do not follow any specific rules. The rule to design the neural network cannot be expressed with a formula or any other mathematical approach, which can be optimized. The skillful design of a neural network can be time-consuming and difficult and requires expertise.

Luckily, it is possible, by now, for medical researchers to input their data into the system, model and optimize automatically, and fetch the best model, hyper-parameter, and results. The algorithms to search for the best archi-tecture of a neural network were suggested [23, 68]. Neural architecture search and hyper-parameter optimization can help to overcome the difficulty of manual neural network design and refinement.

7.2.2 Knowledge

Medical knowledge can help researchers to understand what the machine has learned, and provide the explanation on the medical and clinical level. The interpretability analysis of deep learning also provides the ability to find out the features, when it is ignored by human.

Medicine- and deep learning-concerned surveys, papers, and even blog posts can be collected as a kind of knowledge base. For medical researchers, they can quickly find deep learning-related knowledge that is used in their research, while deep learning researchers can also rapidly retrieve medical knowledge. With MEDAS and such a knowledge base, both deep learning and medical researchers can accelerate their research.

7.2.3 Lacking data

One difference between general computer vision and medical image analysis in deep learning is that the latter usually lack data. First, most datasets are on a small scale. Compared with many other computer vision datasets, such as ImageNet [17], most medical datasets only include tens or hundreds of subjects. Second, each group or laboratory might have their private datasets, but mostly on a small scale. These small isolated datasets make it difficult to use them alone, but together.

Federal learning and decentralized learning

Federal learning [8, 42, 82] or other decentralized learning can share what machines have learned without sharing the data. Based on platforms such as MEDAS, and decentralized learning, such as federal learning, researchers from different institutions can efficiently collaborate.

Few-shot learning

Few-shot learning hits a critical spot in medical image analysis, lacking data. When researchers apply deep learning to medical image analysis, one of the big challenges is lacking data. Due to humans can quickly learn from a few data, so many researchers focus on research of few-shot learning of medical image analysis. One example is the researcher of Rezaei et al. [75], which covers a review of zero-shot learning from autonomous vehicles to COVID-19 diagnosis.

Active learning

Active learning is another method to solve the problem of lacking data by reducing the cost of annotation data. Active learning can learn the knowledge from a small set of training data at the beginning, and then, generate the labels for unlabeled data by interacting with experts.

7.2.4 Platform in software engineering

Besides the topics about the development of algorithms, the topics related to software engineering in medical image analysis are also important. There are two topics that MEDAS needs to improve in the future.

Link to PACS/RIS

The hospitals and medical centers usually have their PACS or RIS. Compared to typical data access methods, direct access to PACS and RIS can help researchers access more data, and at the same time, the AI-based medical image analysis algorithms can be easier applied to the clinical environment. However, direct access risks privacy disclosure and the strictest privacy security strategy that impedes access to data. Therefore, how to design the strategy of access PACS and RIS is the improvement of MEDAS and other platforms in the future.

AI Models Evaluation

The typical evaluation of AI-based medical image analysis is measured by the metrics, for example, accuracy. However, these metrics can not tell the users whether it is safe or dependable, mainly when a platform “markets” it to users. Users might care about whether the model can be easily attacked by one pixel changed or whether it can be easily trained for their new tasks. Therefore, it is essential for MEDAS and other platforms to research the evaluation of the model’s safety, usability, and performance.

7.3 Plan

MEDAS is now still at the initial stage, and our users are most of the researchers from our partners. In other words, MEDAS is self-sufficient. MEDAS does not meet all the needs of medical image analysis, and current focuses are primely the detection, classification, and segmentation tasks of MRI, CT, and pathology images.

The development plan is dependent on the community suggestion. Our platform can be used to assist medics and learn from medics, with the help from the technologies, such as federated learning, active learning, life-long learning, etc. Based on meta-learning and active learning, MEDAS can reduce the workload of annotation. We, therefore, currently plan to integrate more useful tools to meet most needs from the community, and the long-term plan currently includes an annotation tool with active learning and other algorithms to reduce the workload of labeling.

8 Summary

In this work, we introduced our platform, named MEDAS, to render the application of deep learning in the medical image analysis more user-friendly, easy, and hence accessible. We designed the pipeline and user interface based on our experience of development and analysis. The pipeline includes pre-processing, post-processing, augmentation, neural network, and visualization & debugging modules. We have also performed several case studies to demonstrate the efficient operation of MEDAS.

Acknowledgments The authors would like to acknowledge all of the contributors to ***MEDAS: An open-source platform as a service to help break the walls between medicine and informatics***. This work was supported by.—Shanghai Science and Technology Committee (No. 18411952100, No. 17411953500)

— National Natural Science Foundation of China (No. 62072358)

— National Key R&D Program of China under Grant (No. 2020YFF0304900, No. 2019YFB1311600).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray DG, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y, Zheng X (2016) TensorFlow: a system for large-scale machine learning. In: Proceedings of the 12th USENIX symposium on operating systems design and implementation, OSDI 2016, vol abs/1605.0, pp 265–283 (2016). http://arxiv.org/abs/1605.08695
2. Andrew AM (1999) The handbook of brain theory and neural. Networks. https://doi.org/10.1108/k.1999.28.9.1084.1. https://dl.acm.org/citation.cfm?id=303568.303704
3. Armato SG, McLennan G, Bidaut L, McNitt-Gray MF, Meyer CR, Reeves AP, Zhao B, Aberle DR, Henschke CI, Hoffman EA, Kazerooni EA, MacMahon H, Van Beek EJ, Yankelevitz D, Biancardi AM, Bland PH, Brown MS, Engelmann RM, Laderach GE, Max D, Pais RC, Qing DP, Roberts RY, Smith AR, Starkey A, Batra P, Caligiuri P, Farooqi A, Gladish GW, Jude CM, Munden RF, Petkovska I, Quint LE, Schwartz LH, Sundaram B, Dodd LE, Fenimore C, Gur D, Petrick N, Freymann J, Kirby J, Hughes B, Vande Casteele A, Gupte S, Sallam M, Heath MD, Kuhn MH, Dharaiya E, Burns R, Fryd DS, Salganicoff M, Anand V, Shreter U, Vastagh S, Croft BY, Clarke LP (2011) The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on CT scans. Med Phys 38(2):915–931. https://doi.org/10.1118/1.3528204
4. Avants BB, Tustison NJ, Song G, Cook PA, Klein A, Gee JC (2011) A reproducible evaluation of ANTs similarity metric performance in brain image registration. NeuroImage 54(3):2033–2044.https://doi.org/10.1016/j.neuroimage.2010.09.025. https://www.sciencedirect.com/science/article/pii/S1053811910012061
5. Beaulah Jeyavathana R, Balasubramanian R, Pandian AA (2016) A survey: analysis on pre-processing and segmentation techniques for medical images. Int J Res Sci Innov III(June):2321–2705
6. Beers A, Brown J, Chang K, Hoebel K, Patel J, Ly KI, Tolaney SM, Brastianos P, Rosen B, Gerstner ER, Kalpathy-Cramer J (2021) DeepNeuro: an open-source deep learning toolbox for neuroimaging. Neuroinformatics 19(1):127–140.https://doi.org/10.1007/s12021-020-09477-5. https://arxiv.org/abs/1808.04589
7. Bilic1a P, Christa PF, Vorontsov E, Chlebusr G, Chenm H, Doum Q, Fum CW, Hanp X, Hengm PA, Hesserq J, Kadourye S, Kopczyskiv T, Leo M, Lio C, Lim X, Lipkova J, Lowengrubn J, Meiner H, Moltzr JH, Pale C, Pirauda M, Qim X, Qil J, Rempera M, Rothq K, Schenkr A, Sekuboyinaa A, Zhouk P, Hulsemeyera C, Beetza M, Ettlingera F, Gruena F, Kaissisb G, Lohferb F, Brarenb R, Holchc J, Hofmannc F, Sommerc W, Heinemannc V, Jacobsd C, Mamanid GEH, Ginnekend BV, Chartrande G, Tange A, Drozdzale M, Kadourye S, Ben-Cohenf A, Klangf E, Amitaif MM, Konenf E, Greenspanf H, Moreaug J, Hostettlerg A, Solerg L, Vivantih R, Szeskinh A, Lev-Cohainh N, Sosnah J, Joskowiczh L, Kumarw A, Korex A, Wangy C, Fengz D, Liaa F, Krishnamurthix G, Heab J, Wuaa J, Kimx J, Zhouac J, Maad J, Liaa J, Maninisae KK, Kaluvax KC, Bix L, Khenedx M, Beliverae M, Linaa Q, Yangad X, Yuanaf Y, Chenaa Y, Liad Y, Qius Y, Wuad Y, Menzea B (2019) The liver tumor segmentation benchmark (LiTS). http://arxiv.org/abs/1901.04056
8. Brendan McMahan H, Moore E, Ramage D, Hampson S, Agüera y Arcas B (2017) Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th international conference on artificial intelligence and statistics, AISTATS 2017. http://arxiv.org/abs/1602.05629
9. Cai H, Chen T, Zhang W, Yu Y, Wang J (2018) Efficient architecture search by network transformation. In: 32nd AAAI conference on artificial intelligence, AAAI 2018, pp 2787–2794
10. Chang CY, Chung PC, Hong YC, Tseng CH (2011) A neural network for thyroid segmentation and volume estimation in CT images. IEEE Computat Intell Mag 6(4):43–55. https://doi.org/10.1109/MCI.2011.942756.. https://ieeexplore.ieee.org/document/6052365
11. Chen S, Bruijne MD (2018) An end-to-end approach to semantic segmentation with 3D CNN and posterior-CRF in medical images. http://arxiv.org/abs/1811.03549
12. Chen Y, Li J, Xiao H, Jin X, Yan S, Feng J (2017) Dual path networks. In: Adv Neural Inf Process Syst 2017:4468–4476
13. Chetlur S, Woolley C, Vandermersch P, Cohen J, Tran J, Catanzaro B, Shelhamer E (2014) cuDNN: efficient primitives for deep learning. arXiv: Neural and evolutionary computing. http://arxiv.org/abs/1410.0759
14. Choi Y, El-Khamy M, Lee J (2017) Towards the limit of network quantization. In: 5th International conference on learning representations, ICLR 2017—conference track proceedings. http://arxiv.org/abs/1612.01543
15. Crankshaw D, Sela GE, Mo S, Zumar C, Gonzalez JE, Stoica I, Tumanov A (2018) InferLine: ML prediction pipeline provisioning and management for tight latency objectives. http://arxiv.org/abs/1812.01776
16. Czarnecki WM, Osindero S, Jaderberg M, Swirszcz G, Pascanu R (2017) Sobolev training for neural networks. Adv Neural Inf Process Syst 2017:4279–4288
17. Deng J, Dong W, Socher R, Li LJ (2010) Kai Li, Li Fei-Fei: ImageNet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp 248–255. IEEE. https://doi.org/10.1109/cvpr.2009.5206848. https://ieeexplore.ieee.org/document/5206848/

18. Denton E, Zaremba W, Bruna J, LeCun Y, Fergus R (2014) Exploiting linear structure within convolutional networks for efficient evaluation. Adv Neural Inf Process Syst 2(January):1269–1277
19. Dettmers T (2016) 8-Bit approximations for parallelism in deep learning. In: 4th International conference on learning representations, ICLR 2016—conference track proceedings
20. Dolz J, Gopinath K, Yuan J, Lombaert H, Desrosiers C, Ben Ayed I (2019) HyperDense-net: a hyper-densely connected cnn for multi-modal image segmentation. IEEE Trans Med Imag 38(5):1116–1126. https://doi.org/10.1109/TMI.2018.2878669.. https://arxiv.org/abs/1804.02967
21. Dou Q, Chen H, Jin Y, Yu L, Qin J, Heng PA (2016) 3D deeply supervised network for automatic liver segmentation from CT volumes. In: S. Ourselin, L. Joskowicz, M.R. Sabuncu, G. Unal, W. Wells (eds.) Lecture notes in computer science (including subseries lecture notes in Artificial intelligence and lecture notes in bioinformatics), vol 9901 LNCS, pp 149–157. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-46723-8_18
22. Fischl B (2012). FreeSurfer. https://doi.org/10.1016/j.neuroimage. 21 Jan 2012. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3685476/
23. Gao Y, Yang H, Zhang P, Zhou C, Hu Y (2019) GraphNAS: graph neural architecture search with reinforcement learning. In: arXiv, vol. abs/1611.0
24. Gibson E, Li W, Sudre C, Fidon L, Shakir DI, Wang G, Eaton-Rosen Z, Gray R, Doel T, Hu Y, Whyntie T, Nachev P, Modat M, Barratt DC, Ourselin S, Cardoso MJ, Vercauteren T (2018) NiftyNet: a deep-learning platform for medical imaging. Comput Methods Programs Biomed 158:113–122. https://doi.org/10.1016/j.cmpb.2018.01.025.. https://www.sciencedirect.com/science/article/pii/S0169260717311823
25. Goodfellow IJ, Erhan D, Luc Carrier P, Courville A, Mirza M, Hamner B, Cukierski W, Tang Y, Thaler D, Lee DH, Zhou Y, Ramaiah C, Feng F, Li R, Wang X, Athanasakis D, Shawe-Taylor J, Milakov M, Park J, Ionescu R, Popescu M, Grozea C, Bergstra J, Xie J, Romaszko L, Xu B, Chuang Z, Bengio Y (2015) Challenges in representation learning: a report on three machine learning contests. Neural Netw 64:59–63
26. Graham, S., Vu, Q.D., Ahmed Raza, S.E., Azam, A., Tsang, Y.W., Kwak, J.T., Rajpoot, N.: HoVer-Net: simultaneous segmentation and classification of nuclei in multi-tissue histology images, 1–11 (2018). http://arxiv.org/abs/1812.06499
27. He K, Gkioxari G, Dollár P, Girshick R (2020) Mask R-CNN. IEEE Tran Pattern Anal Mach Intell 42(2):386–397. https://doi.org/10.1109/TPAMI.2018.2844175
28. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition 2016:770–77. https://doi.org/10.1109/CVPR.2016.90http://arxiv.org/abs/1512.03385
29. Henschel L, Conjeti S, Estrada S, Diers K, Fischl B, Reuter M (2020) FastSurfer—a fast and accurate deep learning based neuroimaging pipeline. NeuroImage 219. Di: 10.1016/j.neuroimage.2020.117012. http://arxiv.org/abs/1910.03866
30. Hohman F, Kahng M, Pienta R, Chau DH (2019) Visual analytics in deep learning: an interrogative survey for the next frontiers. IEEE Trans Vis Comput Graph 25(8):2674–2693
31. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings—30th IEEE conference on computer vision and pattern recognition, CVPR 2017, vol 2017, pp 2261–2269 (2017). https://doi.org/10.1109/CVPR.2017.243.http://arxiv.org/abs/1608.06993
32. Hykes S (2013) Empowering app development for developers | Docker. https://www.docker.com/
33. McCormick M, Liu X, Jomier J, Marion C, Ibanez L (2014) ITK: enabling reproducible research and open science. Front Neuroinform 8:13. https://doi.org/10.3389/fninf.2014.00013
34. Jamaludin A, Kadir T, Zisserman A (2017) SpineNet: Automated classification and evidence visualization in spinal MRIs. Med Image Anal 41:63–73
35. Jenkinson M, Beckmann CF, Behrens TEJ, Woolrich MW, Smith SM (2012) FSL—review. NeuroImage 62(2):782–90
36. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: Convolutional architecture for fast feature embedding. In: MM 2014—Proceedings of the 2014 ACM conference on multimedia, pp 675–678. https://doi.org/10.1145/2647868.2654889
37. Jimenez-Carretero D, Bermejo-Peláez D, Nardelli P, Fraga P, Fraile E, San José Estépar R, Ledesma-Carbayo MJ (2019) A graph-cut approach for pulmonary artery-vein segmentation in noncontrast CT images. Med Image Anal 52:144–159. https://doi.org/10.1016/j.media.2018.11.011. http://www.sciencedirect.com/science/article/pii/S1361841518308740
38. Kamnitsas K, Ledig C, Newcombe VF, Simpson JP, Kane AD, Menon DK, Rueckert D, Glocker B (2017) Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. Med Image Anal 36:61–78
39. Khagi B, Lee CG, Kwon GR (2019) Alzheimer's disease classification from brain MRI based on transfer learning from CNN. In: BMEiCON 2018—11th biomedical engineering international conference. https://doi.org/10.1109/BMEiCON.2018.8609974
40. Khvostikov A, Aderghal K, Benois-Pineau J, Krylov A, Catheline G (2018) 3D CNN-based classification using sMRI and MD-DTI images for Alzheimer disease studies. http://arxiv.org/abs/1801.05968
41. Khvostikov A, Benois-Pineau J, Krylov A, Catheline G (2017) Classification methods on different brain imaging modalities for Alzheimer disease studies. In: GraphiCon 2017—27th international conference on computer graphics and vision, pp 237–242
42. Konečný J, McMahan HB, Yu FX, Richtárik P, Suresh AT, Bacon D (2016) Federated learning: strategies for improving communication efficiency. http://arxiv.org/abs/1610.05492
43. Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. In: Commun ACM 60:84–90. https://doi.org/10.1145/3065386
44. Kumar N, Verma R, Sharma S, Bhargava S, Vahadane A, Sethi A (2017) A dataset and a technique for generalized nuclear segmentation for computational pathology. IEEE Trans Med Imag 36(7):1550–1560. https://doi.org/10.1109/TMI.2017.2677499
45. Lebedev V, Lempitsky V (2016) Fast convnets using group-wise brain damage. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition 2016:2554–2564. https://doi.org/10.1109/CVPR.2016.280
46. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2323. https://doi.org/10.1109/5.726791
47. Lee LK, Liew SC (2015) A survey of medical image processing tools. In: 2015 4th international conference on software engineering and computer systems, ICSECS 2015: virtuous software solutions for big data, pp 171–176. https://doi.org/10.1109/ICSECS.2015.7333105
48. Li Z, Hoiem D (2018) Learning without forgetting. IEEE Trans Pattern Anal Mach Intell 40(12):2935–2947. https://doi.org/10.1109/TPAMI.2017.2773081
49. Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, van der Laak JA, van Ginneken B, Sánchez CI (2017). A survey on deep learning in medical image analysis
50. Litjens, G., Sánchez CI, Timofeeva N, Hermsen M, Nagtegaal I, Kovacs I, Hulsbergen-Van De Kaa C, Bult P, Van Ginneken B,

Van Der Laak J (2016) Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. Sci Rep 6(1):26286. https://doi.org/10.1038/srep26286.http://www.nature.com/articles/srep26286
51. Liu T, Guo Q, Lian C, Ren X, Liang S, Yu J, Niu L, Sun W, Shen D (2019) Automated detection and classification of thyroid nodules in ultrasound images using clinical-knowledge-guided convolutional neural networks. Med Image Anal 58:101555
52. Lowekamp BC, Chen DT, Ibáñez L, Blezek D (2013) The design of simpleITK. Front Neuroinf 7(DEC):45. https://doi.org/10.3389/fninf.2013.00045
53. Lu F, Wu F, Hu P, Peng Z, Kong D (2017) Automatic 3D liver location and segmentation via convolutional neural network and graph cut. Int J Comput Assist Radiol Surg 12(2):171–182
54. Magee D, Treanor D, Crellin D, Shires M, Smith K, Mohee K, Quirke P (2009) Colour normalisation in digital histopathology images. Opt Tissue Image Anal Microsc Histopathol Endosc MICCAI Workshop, pp 100–111. https://www.researchgate.net/publication/228855426_Colour_Normalisation_in_Digital_Histopathology_Imageshttps://www.researchgate.net/publication/339593324_Colour_Normalisation_in_Digital_Histopathology_Images
55. Maloney J, Resnick M, Rusk N, Silverman B, Eastmond E (2010) The scratch programming language and environment. ACM Trans Comput Educ 10(4):16. https://doi.org/10.1145/1868358.1868363
56. Marcos Romero BS (2019) Blueprints visual scripting for unreal engine. https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html
57. Marlow S (2010) Haskell 2010 language report. Language, p 329. http://haskell.org/definition/haskell2010.pdf
58. Milletari F, Navab N, Ahmadi SA (2016) V-Net: fully convolutional neural networks for volumetric medical image segmentation. In: Proceedings—2016 4th international conference on 3D vision, 3DV 2016, pp 565–571. https://doi.org/10.1109/3DV.2016.79
59. Minati, L., Edginton, T., Grazia Bruzzone, M., Giaccone, G.: Reviews: current concepts in alzheimer's disease: a multidisciplinary review (2009). https://doi.org/10.1177/1533317508328602
60. Moradi M, Madani A, Karargyris A, Syeda-Mahmood TF (2018) Chest x-ray generation and data augmentation for cardiovascular abnormality classification. In: E.D. Angelini, B.A. Landman (eds.) Medical imaging 2018: image processing 10574:57. SPIE. https://doi.org/10.1117/12.2293971.https://doi.org/10.1117/12.2293971
61. Mueller SG, Weiner MW, Thal LJ, Petersen RC, Jack C, Jagust W, Trojanowski JQ, Toga AW, Beckett L (2005) The Alzheimer's disease neuroimaging initiative. Neuroimag Clin North Am 15(4):869–877. https://doi.org/10.1016/j.nic.2005.09.008
62. Müller, D., Kramer, F.: MIScnn: A framework for medical image segmentation with convolutional neural networks and deep learning (2019)
63. Naylor P, Laé M, Reyal F, Walter T (2019) Segmentation of nuclei in histopathology images by deep regression of the distance map. IEEE Trans Med Imag 38(2):448–459. https://doi.org/10.1109/TMI.2018.2865709
64. Ogiela MR, Tadeusiewicz R (2008) Preprocessing medical images and their overall enhancement. Stud Comput Intell 84:65–97. https://doi.org/10.1007/978-3-540-75402-2_4
65. Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A (2017) Automatic differentiation in \uppercasePy\uppercaseTorch. In: NIPS 2017 Autodiff Workshop: the future of gradient-based machine learning software and techniques, pp 8024–8035 (2017). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
66. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) PyTorch: An imperative style, high-performance deep learning library. http://arxiv.org/abs/1912.01703
67. Pereira S, Pinto A, Alves V, Silva CA (2016) Brain tumor segmentation using convolutional neural networks in MRI images. IEEE Trans Med Imag 35(5):1240–1251. https://doi.org/10.1109/TMI.2016.2538465
68. Pham H, Guan MY, Zoph B, Le QV, Dean J (2018) Efficient neural architecture search via parameter sharing. In: 35th International conference on machine learning, ICML 2018, vol 9, pp 6522–6531
69. Qaiser T, Tsang YW, Taniyama D, Sakamoto N, Nakane K, Epstein D, Rajpoot N (2019) Fast and accurate tumor segmentation of histology images using persistent homology and deep convolutional features. Med Image Anal 55:1–14
70. Radul T (2001) Functional representations of Lawson monads. Appl Categor Struct 9(5):457–463. https://doi.org/10.1023/A:1012052928198
71. Rajan, D., Beymer, D., Abedin, S., Dehghan, E.: Pi-PE: A pipeline for pulmonary embolism detection using sparsely annotated 3D CT images (2019). http://arxiv.org/abs/1910.02175
72. Rajchl, M., Pawlowski, N., Rueckert, D., Matthews, P.M., Glocker, B.: NeuroNet: Fast and robust reproduction of multiple brain image segmentation pipelines (2018). http://arxiv.org/abs/1806.04224
73. Rameshkumar S, Thilak JAJ, Suresh P, Sathishkumar S, Subramani N (2016) Speckle noise removal in MRI scan image using WB—filter. Int J Innov Res Sci Eng Technol 5(12):21079–21083. https://doi.org/10.15680/IJIRSET.2016.0512161
74. Reinhard E, Ashikhmin M, Gooch B, Shirley P (2001) Color transfer between images. IEEE Comput Graph Appl 21(5):34–41. https://doi.org/10.1109/38.946629
75. Rezaei M, Shahidi M (2020) Zero-shot learning and its applications from autonomous vehicles to covid-19 diagnosis: a review. https://doi.org/10.1016/j.ibmed.2020.100005.. http://arxiv.org/abs/2004.14143
76. Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: Lecture notes in computer science (including subseries Lecture Notes in Artificial intelligence and lecture notes in bioinformatics), vol 9351, pp 234–241 (2015). https://doi.org/10.1007/978-3-319-24574-4_28
77. Ruifrok AC, Johnston DA (2001) Quantification of histochemical staining by color deconvolution. Anal Quant Cytol Histol 23(4):291–299
78. Ryan Olson, Jonathan Calmels, F.A., |, P.R.: NVIDIA Docker: GPU server application deployment made easy (2016). https://devblogs.nvidia.com/nvidia-docker-gpu-server-application-deployment-made-easy/
79. Satyanarayanan M, Goode A, Gilbert B, Harkes J, Jukic D (2013) OpenSlide: a vendor-neutral software foundation for digital pathology. J Pathol Inf 4(1):27. https://doi.org/10.4103/2153-3539.119005
80. Senthilraja S, Suresh P, Suganthi M (2014) Noise reduction in computed tomography image using WB-filter. Int J Sci Eng Res 5(3):243
81. Setio AAA, Traverso A, de Bel T, Berens MS, van den Bogaard C, Cerello P, Chen H, Dou Q, Fantacci ME, Geurts B, van der Gugten R, Heng PA, Jansen B, de Kaste MM, Kotov V, Lin

JYH, Manders JT, Sóñora-Mengana A, García-Naranjo JC, Papavasileiou E, Prokop M, Saletta M, Schaefer-Prokop CM, Scholten ET, Scholten L, Snoeren MM, Torres EL, Vandemeulebroucke J, Walasek N, Zuidhof GC, van Ginneken B, Jacobs C (2017) Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: The LUNA16 challenge. Med Image Anal 42:1–13. https://doi.org/10.1016/j.media.2017.06.015. http://arxiv.org/abs/1612.08012
82. Sheller MJ, Reina GA, Edwards B, Martin J, Bakas S (2019) Multi-institutional deep learning modeling without sharing patient data: a feasibility study on brain tumor segmentation. In: Lecture notes in Computer science (including subseries Lecture notes in Artificial intelligence and lecture notes in bioinformatics), vol 11383 LNCS, pp 92–104. https://doi.org/10.1007/978-3-030-11723-8_9
83. Shorten C, Khoshgoftaar TM (2019) A survey on image data augmentation for deep learning. J Big Data 6(1):60. https://doi.org/10.1186/s40537-019-0197-0
84. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: 3rd International conference on learning representations, ICLR 2015—conference track proceedings
85. Skibbe H, Watakabe A, Nakae K, Gutierrez CE, Tsukada H, Hata J, Kawase T, Gong R, Woodward A, Doya K, Okano H, Yamamori T, Ishii S (2019) MarmoNet: a pipeline for automated projection mapping of the common marmoset brain from whole-brain serial two-photon tomography. http://arxiv.org/abs/1908.00876
86. Song J, Xiao L, Molaei M, Lian Z (2019) Multi-layer boosting sparse convolutional model for generalized nuclear segmentation from histopathology images. Knowl Based Syst 176:40–53
87. Swiderska-Chadaj Z, Pinckaers H, van Rijthoven M, Balkenhol M, Melnikova M, Geessink O, Manson Q, Sherman M, Polonia A, Parry J, Abubakar M, Litjens G, van der Laak J, Ciompi F (2019) Learning to detect lymphocytes in immunohistochemistry with deep learning. Med Image Anal 58. https://doi.org/10.1016/j.media.2019.101547
88. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-ResNet and the impact of residual connections on learning. In: 31st AAAI conference on artificial intelligence, AAAI 2017, pp 4278–4284 (2017)
89. Tai, C., Xiao, T., Zhang, Y., Wang, X., Weinan, E.: Convolutional neural networks with low-rank regularization. In: 4th International conference on learning representations, ICLR 2016—conference track proceedings (2016)
90. The Linux foundation: production-grade container orchestration—Kubernetes (2020). https://kubernetes.io/
91. Thenua R, Agarwal S (2010) Simulation and performance analysis of adaptive filter in noise cancellation. Int J Eng Sci Technol 2(9):4373–4378
92. Tofighi M, Guo T, Vanamala JK, Monga V (2019) Prior information guided regularized deep learning for cell nucleus detection. IEEE Trans Med Imag 38(9):2047–2058. https://doi.org/10.1109/TMI.2019.2895318
93. Trullo, R., Petitjean, C., Ruan, S., Dubray, B., Nie, D., Shen, D.: Segmentation of organs at risk in thoracic CT images using a sharpmask architecture and conditional random fields. In: Proceedings—international symposium on biomedical imaging, vol 2017, pp 1003–1006 (2017). https://doi.org/10.1109/ISBI.2017.7950685
94. Tustison NJ, Avants BB, Cook PA, Zheng Y, Egan A, Yushkevich PA, Gee JC (2010) N4ITK: improved N3 bias correction. IEEE Trans Med Imag 29(6):1310–1320. https://doi.org/10.1109/TMI.2010.2046908
95. Vahadane A, Peng T, Sethi A, Albarqouni S, Wang L, Baust M, Steiger K, Schlitter AM, Esposito I, Navab N (2016) Structure-preserving color normalization and sparse stain separation for histological images. IEEE Trans Med Imag 35(8):1962–1971. https://doi.org/10.1109/TMI.2016.2529665
96. Vanhoucke V, Senior A, Mao M (2011) Improving the speed of neural networks on CPUs. Proc Deep Learn, pp 1–8. http://research.google.com/pubs/archive/37631.pdf
97. Wang Z, Lin Y, Cheng KTT, Yang X (2020) Semi-supervised mp-MRI data synthesis with StitchLayer and auxiliary distance maximization. Med Image Anal 59:101565
98. Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J (2015) 3D ShapeNets: a deep representation for volumetric shapes. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, vol 07–12-June, pp 1912–1920. https://doi.org/10.1109/CVPR.2015.7298801. http://arxiv.org/abs/1406.5670
99. Yao GL (2017) A survey on pre-processing in image matting. J Comput Sci Technol 32(1):122–138. https://doi.org/10.1007/s11390-017-1709-z
100. Yi X, Walia E, Babyn P (2019) Generative adversarial network in medical imaging: a review. Med Image Anal 58:101552
101. Yong CY, Chew KM, Mahmood NH, Ariffin I (2012) A survey of visualization tools in medical imaging. Proc Soc Behav Sci 56:265–271
102. Zhang J, Gajjala S, Agrawal P, Tison GH, Hallock LA Beussink-Nelson L, Lassen MH, Fan E, Aras MA, Jordan CR, Fleischmann KE, Melisko M, Qasim A, Efros A, Shah SJ, Bajcsy R, Deo RC (2017) A computer vision pipeline for automated determination of cardiac structure and function and detection of disease by two-dimensional echocardiography. http://arxiv.org/abs/1706.07342
103. Zhang, K., Snavely, N., Sun, J.: Leveraging vision reconstruction pipelines for satellite imagery (2019). http://arxiv.org/abs/1910.02989
104. shi Zhang, Q., chun Zhu, S.: Visual interpretability for deep learning: a survey (2018). https://doi.org/10.1631/FITEE.1700808
105. Zhou Y, Xie L, Shen W, Wang Y, Fishman EK, Yuille AL (2017) A fixed-point model for pancreas segmentation in abdominal CT scans. In: M. Descoteaux, L. Maier-Hein, A. Franz, P. Jannin, D.L. Collins, S. Duchesne (eds.) Lecture notes in Computer science (including subseries Lecture notes in Artificial intelligence and lecture notes in Bioinformatics), vol 10433 LNCS, pp693–701. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-66182-7_79
106. Zhu, W., Liu, C., Fan, W., Xie, X.: DeepLung: Deep 3D dual path nets for automated pulmonary nodule detection and classification. In: Proceedings—2018 IEEE winter conference on applications of computer vision, WACV 2018, 2018:673–681 (2018). https://doi.org/10.1109/WACV.2018.00079