Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

ImageAugmenter: A user-friendly 3D Slicer tool for medical image augmentation

Ciro Benito Raggio [a,*], Paolo Zaffino [b,1], Maria Francesca Spadea [a,1]

[a] Karlsruhe Institute of Technology, Fritz-Haber-Weg 1, Karlsruhe 76131, Germany
[b] Magna Graecia University of Catanzaro, Viale Europa, Catanzaro 88100, Italy

A R T I C L E I N F O

Keywords:
Medical imaging
Augmentation
3D Slicer
Deep learning

A B S T R A C T

Limited medical image data hinders the training of deep learning (DL) models in the biomedical field. Image augmentation can reduce the data-scarcity problem by generating variations of existing images. However, currently implemented methods require coding, excluding non-programmer users from this opportunity.

We therefore present *ImageAugmenter*, an easy-to-use and open-source module for 3D Slicer imaging computing platform. It offers a simple and intuitive interface for applying over 20 simultaneous MONAI Transforms (spatial, intensity, etc.) to medical image datasets, all without programming.

ImageAugmenter makes accessible medical image augmentation, enabling a wider range of users to improve the performance of DL models in medical image analysis by increasing the number of samples available for training.

Metadata

Nr	Code metadata description	*Please fill in this column*
C1	Current code version	v1.0.4 (2024/09/03)
C2	Permanent link to code/repository used for this code version	https://github.com/ciroraggio/SlicerImageAugmenter/releases/tag/v1.0.4
C3	Permanent link to reproducible capsule	
C4	Legal code license	MIT License
C5	Code versioning system used	Git
C6	Software code languages, tools and services used	Python, CMake
C7	Compilation requirements, operating environments and dependencies	3D Slicer, PyTorch, MONAI, SimpleITK, munch
C8	If available, link to developer documentation/manual	https://ciroraggio.github.io/SlicerImageAugmenter/
C9	Support email for questions	ciro.raggio@kit.edu

1. Motivation and significance

In the field of medical image analysis, deep learning (DL) models have achieved remarkable success in tasks like disease classification, tissue and organ segmentation, and image registration [1]. However, two major barriers in the development of these models are the limited availability of medical image data, and the increasingly stringent regulations that make the sharing of medical images, even for research purposes, more complex. This scarcity of data hinders the training process, often leading to models that overfit the training data and perform poorly on unseen data. Data augmentation has been proposed by several authors to address this issue [2–5].

Image augmentation is a technique that synthetically expands the training dataset by manipulating existing images or by generating artificial samples from scratch [6]. In the first, and most common case, the augmentation can include geometric transformations (translations, rotations, flips, scaling, deformation, zooming), intensity variations (histogram based operations), and noise injection [7].

Augmented data helps the DL model to learn features that are invariant to these alterations and improves the generalization capabilities of the model. Current augmentation techniques are often based on custom code written in programming languages such as Python. While there are existing tools and frameworks that address medical image augmentation, such as Medical Imaging Toolkit (MITK) [8,9], Pillow [10], PyTorch [11], TorchIO [12], and MONAI [13], they all have one thing in common: they require programming knowledge to be effectively used. This excludes researchers and medical professionals, who may lack programming skills, from exploiting the benefits of image augmentation. It is also time consuming for investigators to implement

* Corresponding author at: Institute of Biomedical Engineering, Karlsruhe Institute of Technology, Fritz-Haber-Weg 1, Karlsruhe 76131, Germany.
E-mail address: ciro.raggio@kit.edu (C.B. Raggio).
[1] These authors contributed equally to this work and share senior authorship.

https://doi.org/10.1016/j.softx.2024.101923
Received 9 July 2024; Received in revised form 20 September 2024; Accepted 3 October 2024
Available online 21 October 2024

different kinds of augmentation before finding the optimal set that maximizes model performance.

ImageAugmenter addresses this challenge by providing a user-friendly tool for medical image augmentation and enabling the creation of richer and more diverse training datasets, which can directly improve the performance of DL models.

This translates to increased accuracy, generalizability, and robustness for tasks like cancer detection, segmentation of anatomical structures, and computer-aided diagnosis.

ImageAugmenter is integrated within the open source 3D Slicer [14] platform as a module of a new extension, designated as "Slicer-ImageAugmenter." 3D Slicer is widely used in scientific and clinical settings. Its popularity is evidenced by a high number of downloads (over 1.5 million since 2011)[1] and a large community of active users.

Using the module is simple and intuitive. The user must:

1. install the extension through 3D Slicer's extension management module;
2. select the data to be augmented, where the module supports several medical image formats, including DICOM and NIfTI with two different types of folder hierarchies;
3. select the MONAI Transforms to be applied;
4. perform data augmentation; here, several parameters to customize data augmentation can be selected;.
5. visualize and analyze the augmented data.

The integration with 3D Slicer makes the tool easily accessible to a wide range of users, facilitating the use of advanced data augmentation techniques for medical image retrieval and analysis. Moreover, augmented images can be also visualized through 3D Slicer panels for easily checking the artificial data space.

2. Software description

The module leverages the capabilities of PyTorch [11] and MONAI [13] to perform data augmentation on medical image datasets. This process involves applying a series of specific MONAI transformations to each image within the dataset, creating a larger and more diverse collection of samples for training DL models. In light of the mentioned points, all forthcoming references to the term "transformation" should be understood in a more precise manner as "MONAI Transform".

2.1. Software architecture

The ImageAugmenter module is structured into several classes, including those provided by the 3D Slicer infrastructure for building scripted modules[2]. These include the ImageAugmenterWidget, which is responsible for managing the graphical user interface (GUI) of the module, and the ImageAugmenterLogic, which handles the general logic of the module. Additionally, a set of custom classes and Python modules are grouped in the library named ImageAugmenterLib, which allows for the creation of a modular architecture and serves as the core of the module.

The ImageAugmenterTransformationParser class is responsible for coordinating the controller classes and collecting the list of MONAI Transforms in accordance with the user inputs.

The ImageAugmenterTransformControllerInterface, located in the ImageAugmenterLib, is the abstract class upon which three distinct classes are then built: ImageAugmenterSpatialController, ImageAugmenterIntensityController and ImageAugmenterCropController. The aim of the controller classes is to be specialized in mapping different types of MONAI Transforms, ensuring compatibility with the dataset class. The ImageAugmenterDataset class implements the Dataset abstract class from PyTorch and handles dataset loading and MONAI Transforms application.

In addition to the mentioned classes, two further Python modules are included in the ImageAugmenterLib. The ImageAugmenterUtils Python module contains essential functions, such as saving to disk and previewing the result, plus other minor utility functions. The ImageAugmenterValidator Python module, on the other hand, contains functions used in the initial phase to validate the fields in the GUI.

This modular approach, which combines the standard Slicer module structure with a set of custom classes, provides a foundation for future enhancements and additions to the module's capabilities.

2.2. Software functionalities

ImageAugmenter enables users to define and apply different image augmentation transformations to their own medical image datasets, with the ability to maintain the same input hierarchy and ensure that spatial correspondence between image and mask is maintained through the applied transformations.

With a user-friendly graphical interface, the module allows users to select input and output directories, specify image and mask prefixes, define desired transformations through a comprehensive set of options, and choose the device (CPU or PyTorch compatible GPUs) on which to execute the computation. It also provides real-time progress visualization and informative status updates.

The module includes a "preview" functionality. Users can select a subset of transformations and visualize the resulting augmented images alongside their original counterparts. This facilitates the refinement of the selected transformations to achieve the optimal outcome for their specific use case. Furthermore, this approach avoids occupying memory on the disk and facilitates the user's understanding of the model's input during training. As a consequence, time can be invested in further stages of research because a data sanity check has already been performed.

In the current version, 7 spatial transformations, 12 intensity transformations, 2 crop transformations and 2 pad transformations are available. Nevertheless, our efforts have been concentrated on ensuring that the integration of further MONAI Transforms is straightforward, largely due to the code design.

Moreover, the majority of parameters that users are required to configure for each transformation are identical in terms of nomenclature and functionality to those found in the MONAI library. This approach streamlines the referencing of MONAI documentation and standardizes the augmentation pipeline. It ensures that every detail of the transformations in the module is reflected and directly accessible in the official MONAI documentation.

3. Illustrative examples

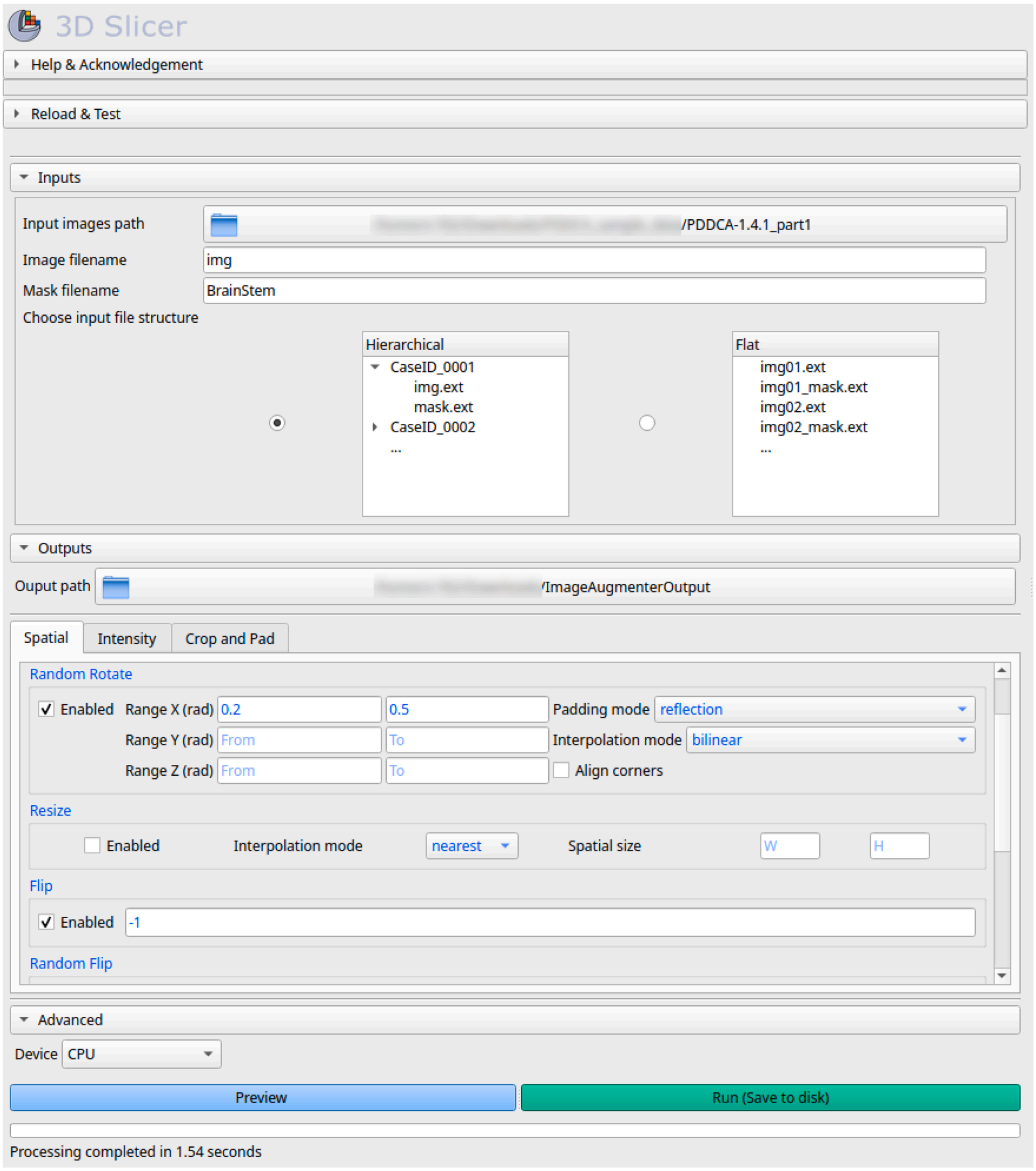
Fig. 1 illustrates the *ImageAugmenter* module. The user is first prompted to specify the path to the images to be augmented, file hierarchy information, and file name patterns that will allow the detection of images and masks (if any) within the specified path. The user is then prompted to specify a path where the augmented images will be stored.

Subsequently, the user is given the option to select any MONAI Transforms to be applied, as well as tune their parameters. If the underlying hardware provides a PyTorch-compatible GPU, the user can also select the device on which the augmentation will be performed.

Finally, the user can choose to preview or save the augmented images.

Fig. 2 shows an example of how the 3D Slicer scene looks like when using the "Preview" feature of *ImageAugmenter*. The inclusion of a preview function within the 3D Slicer module increases the user's control over the augmentation process. By allowing real-time tuning of parameters within the augmentation panel, users will be able to establish the result of the MONAI Transforms used by checking the images slice by slice from the 3D Slicer infrastructure.

Finally, Listing 1 shows how *ImageAugmenter* exports augmented images with different dataset types. It is appreciable how the module maintains the input hierarchy, clearly adding the type of MONAI



Input images path: [redacted]/PDDCA-1.4.1_part1
Image filename: img
Mask filename: BrainStem
Choose input file structure: [✓ Hierarchical] Flat

```
Hierarchical
  CaseID_0001
      img.ext
      mask.ext
  CaseID_0002
  ...

Flat
  img01.ext
  img01_mask.ext
  img02.ext
  img02_mask.ext
  ...
```

Ouput path: [redacted]/ImageAugmenterOutput

Random Rotate
✓ Enabled Range X (rad): 0.2 – 0.5 Padding mode: reflection
Range Y (rad): From – To Interpolation mode: bilinear
Range Z (rad): From – To ☐ Align corners

Resize
☐ Enabled Interpolation mode: nearest Spatial size: W H

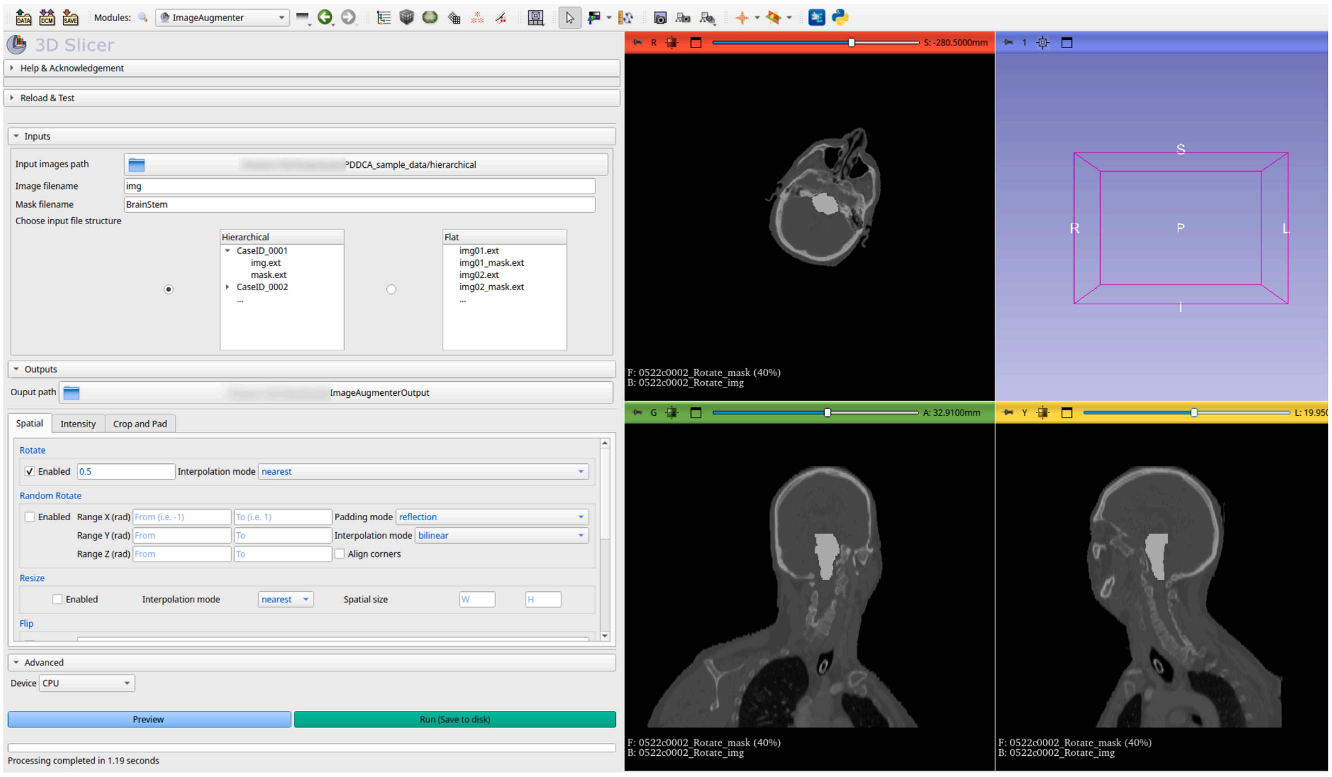
Flip
✓ Enabled -1

Random Flip

Device: CPU

Processing completed in 1.54 seconds

Fig. 1. ImageAugmenter graphical user interface.



Inputs
Input images path [redacted]PDDCA_sample_data/hierarchical
Image filename img
Mask filename BrainStem
Choose input file structure
[✓ Hierarchical] Flat

```
Hierarchical
  CaseID_0001
    img.ext
    mask.ext
  CaseID_0002
  ...
```

```
Flat
  img01.ext
  img01_mask.ext
  img02.ext
  img02_mask.ext
  ...
```

Outputs
Ouput path [redacted]ImageAugmenterOutput

Spatial | Intensity | Crop and Pad

Rotate
✓ Enabled 0.5 Interpolation mode nearest

Random Rotate
Enabled Range X (rad) From (i.e. -1) To (i.e. 1) Padding mode reflection
Range Y (rad) From To Interpolation mode bilinear
Range Z (rad) From To Align corners

Resize
Enabled Interpolation mode nearest Spatial size W H

Flip

Advanced
Device CPU

Preview | Run (Save to disk)

Processing completed in 1.19 seconds

F: 0522c0002_Rotate_mask (40%)
B: 0522c0002_Rotate_img

F: 0522c0002_Rotate_mask (40%)
B: 0522c0002_Rotate_img

F: 0522c0002_Rotate_mask (40%)
B: 0522c0002_Rotate_img

Fig. 2. Preview mode.

Transform applied to a specific case.

```
HierarchicalDataset/
│
├── 0522c0013/
│   ├── img.nrrd
│   └── BrainStem.nrrd
├── 0522c0014/
│   ├── img.nrrd
│   └── BrainStem.nrrd
└── ...
```

```
ImageAugmenterOutput/
├── 0522c0013_Flip/
│   ├── img.nrrd
│   └── BrainStem.nrrd
├── 0522c0013_Rotate/
│   ├── img.nrrd
│   └── BrainStem.nrrd
├── 0522c0014_Flip/
│   ├── img.nrrd
│   └── BrainStem.nrrd
├── 0522c0014_Rotate/
│   ├── img.nrrd
│   └── BrainStem.nrrd
└── ...
```

```
FlatDataset/
├── img001.nrrd
├── label001.nrrd
├── img002.nrrd
├── label002.nrrd
└── ...
```

```
ImageAugmenterOutput/
├── img001_Flip.nrrd
├── label001_Flip.nrrd
├── img001_Rotate.nrrd
├── label001_Rotate.nrrd
├── img002_Flip.nrrd
├── label002_Flip.nrrd
├── img002_Rotate.nrrd
├── label002_Rotate.nrrd
└── ...
```

Listing 1. Input and output example. The two hierarchies of inputs that are supported are illustrated on the left, and the respective generated outputs are shown on the right.

4. Impact

Medical data augmentation has emerged as a powerful technique to enhance the accuracy and robustness of various deep learning-based tasks in medical image analysis [15].

The application of ImageAugmenter can provide a meaningful contribution in addressing key research questions and challenges in medical deep learning. As evidenced by studies referenced in [16,17], data augmentation is an effective method for enhancing the generalization and robustness of medical image analysis models across different imaging modalities, scanner types, and patient demographics. Addressing this issue is essential to ensure that models perform effectively on previously unseen data and are not susceptible to bias. Furthermore, medical imaging datasets often suffer from class imbalance, where certain disease classes are underrepresented. Thoughtful application of augmentation methods can mitigate this issue, particularly in tasks like lesion detection or segmentation [18]. By augmenting images from minority classes, the dataset becomes more balanced, leading to improved model performance.

It is also important to consider that medical imaging is prone to artifacts such as motion blur, noise, or intensity inhomogeneities [19]. A key consideration is how augmentation techniques can be used to simulate these imaging artifacts, thereby training models that are more resilient to such common challenges in real-world medical imaging. This will improve the ability of the model to deal effectively with noisy or imperfect data.

Finally, in order to make it clear the impact that ImageAugmenter module can have on the research community, we would like to point out how that Goceri's review [15] examined very comprehensively augmentation techniques that have been applied to improve the performance of deep learning-based diagnosis on different organs (brain, lung, breast, and eye), using different imaging modalities (MRI, CT, mammography, and fundoscopy). Results reported in Tables 1–4 of the review [15], showed that 56 out of the 77 studies proposed in the tables, involved the use of transformations (such as translation, scaling, flipping, brightness changing, rotation, mirroring, zooming, resizing, etc), as augmentation methods, improving performance across the wide range of tasks to which it was applied.

In light of this, the positive implications of ImageAugmenter for the scientific community and upcoming research efforts are obvious. Moreover, a strong foundation for the adoption of this module is the widespread use of 3D Slicer, both within and beyond the intended user group.

4.1. Comparison with other available tools

The ImageAugmenter module within the 3D Slicer environment is only comparable to the TorchIO 3D Slicer module[3] in terms of functionality. TorchIO is a Python package based on PyTorch for reading, preprocessing, augmenting, and writing 3D medical images for deep learning applications. It offers intensity and spatial transforms for data augmentation and preprocessing. The developers of TorchIO library describe the TorchIO 3D Slicer module as a tool that allows users to rapidly visualize the impact of each transformation parameter, thereby providing an intuitive understanding of the output without the need for coding[4]. Although the underlying concept may seem similar, several important differences emerge between the TorchIO 3D Slicer module and ImageAugmenter. The following section will describe the distinctions between the two modules and illustrate the enhancements that we have implemented to enhance the overall user experience.

The first distinction between the two modules is related to their foundational frameworks. Our module is based on MONAI, while TorchIO 3D Slicer module is based on the TorchIO package. MONAI provides a comprehensive suite of high-level Transform classes, particularly in the domain of spatial transformations, which are not present in TorchIO. Consequently, this represents an important drawback for the 3D Slicer TorchIO module, as it necessitates additional effort from users to implement functionality comparable to the advanced Transform classes available in MONAI.

Another difference lies in the number of transformations that the two modules offer to the user and the usability of these transformations. Our module provides more than 20 MONAI Transforms customizable and applicable to an entire dataset of images and masks simultaneously. The 3D Slicer TorchIO module offers approximately 10 TorchIO transformations, the majority of which are random and basic, such as RandomAffine.

It is also crucial to highlight the distinction in the manner in which our module is integrated in the 3D Slicer environment and how it interacts with the operating system. Unlike the TorchIO 3D Slicer module, the user is not required to load the images into the 3D Slicer environment to apply the augmentation. Our module enables the loading of datasets directly from the operating system, supporting two distinct hierarchies. It is possible to apply multiple transformations simultaneously, preview the transformations, and then automatically save the results with the same hierarchy and file names as the input dataset, enhancing and accelerating the user experience.

On the other hand, the TorchIO 3D Slicer module enables the application of a single TorchIO transformation to a single volume at a time and it is not possible to save the augmented volume directly on the operating system with the same input hierarchy. Consequently, further interaction with 3D Slicer is required to export the augmented volume and repeat the process for the entire dataset.

To illustrate more clearly the differences between ImageAugmenter and the TorchIO 3D Slicer module, a comparison has been proposed in Table 1. The table highlights the key features and capabilities of both modules, providing an overview of their respective strengths and limitations.

4.2. Performance

The analysis of the tool's performance is based on a series of different benchmarks. The benchmarks evaluated the time required to preview and save images as the image size, the number of images, and the number of image transformations increased. The indicated preview and save times are averaged on three consecutive runs. The tests were conducted with both CPU and GPU to provide a more comprehensive evaluation and perspective of the tool.

Table 1
Differences between ImageAugmenter and the TorchIO 3D Slicer module.

Feature	ImageAugmenter	TorchIO 3D Slicer Module
Base Framework	MONAI	TorchIO
Transforms #	24 MONAI transforms	10 TorchIO transforms
Transforms Parameters	Customizable	Customizable
Dataset Loading	Direct from operating system, supports two different hierarchies	Requires loading the data into 3D Slicer as Volumes in order to apply the augmentation
Augmentation	Can apply multiple transforms to the entire dataset, on the image and the related segmentation/mask (if present) simultaneously	Applies a single transformation to a single volume at a time
Preview	Available for all the MONAI Transforms chosen on a volume of the dataset	Available for the volume on which the transform is being applied
Saving	The augmented dataset is automatically exported with the same hierarchy and file names as the original, directly to the operating system	Additional interactions with 3D Slicer are required for the exportation of augmented volumes to the operating system manually

The hardware used for these tests included:

- CPU: Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz
- GPU: NVIDIA Corporation GA104GL [RTX A4000] (16GB)
- RAM: 12 × 16GB DDR4 DIMM

Furthermore, the number of cases and the amount of input files, as well as the corresponding cases and output files, are presented to assess the consistency of the savings.

To ensure reproducibility, three different public datasets of different sizes were used:

- *PDDCA v1.4.1* [20] with 12 images averaging 512 × 512 × 138 in size.
- *SynthRAD2023* [21] challenge dataset for Task 1, using only the validation set with 30 images of size 220 × 220 × 200 (average) .
- *SynthRAD2023* challenge dataset for Task 1.2, using only the training set with 179 images of size 220 × 220 × 200 (average).

The following section provides a summary of the datasets (Table 2), the types of MONAI Transforms applied (Table 3), and the parameters used to run the benchmarks. The results obtained for each benchmark are presented in Table 4.

Dataset

Table 2
Description of the datasets and input files used in the benchmark.

DatasetID	Description	Size	Input image file	Input segmentation file	Patients (#)	Files in total (#)
D1	PDDCA v1.4.1 (part 1)	512 × 512 × 138 circa	img.nrrd	BrainStem.nrrd	12	48
D2	SynthRAD2023 - Task1 - Brain - Validation set	220 × 220 × 200 circa	mr.nii.gz	mask.nii.gz	30	60
D3	SynthRAD2023 - Task1.2 - Brain - Train set	220 × 220 × 200 circa	ct.nii.gz	mask.nii.gz	179	359

Transformations

Table 3
Description of the transformations and specific parameters used in the benchmark.

TransformsID	Description	Settings
T1	A. Rotate	A. Angle: 0.5rad; interpolation mode: nearest
	B. RandomFlip	B. -
T2	A. Rotate (angle: 0.5rad, interpolation mode: nearest)	A. Angle: 0.5rad; interpolation mode: nearest
	B. RandomFlip	B. -
	C. Zoom	C. Factor: 1.2; interpolation mode: area; padding mode: edge
	D. BorderPad	D. Spatial border: 10; mode: symmetric
T3	A. Rotate	A. Angle: 0.5rad; interpolation mode: nearest
	B. RandomFlip	B. -
	C. Zoom	C. Factor: 1.2; interpolation mode: area; padding mode: edge
	D. BorderPad	D. Spatial border: 10; mode: symmetric
	E. CenterSpatialCrop	E. Height: 200; width: 200
	F. GaussianSmooth	F. Factor: 1.0
	G. RandomZoom	G. From 0.5 to 0.9; interpolation mode :area; padding mode:edge
	H. RandomRotate	H. On the X axis; from 0.6rad to 0.9rad; interpolation mode: nearest; padding mode:reflection

5. Results

It is important to note that the results obtained (Table 4) depend on the hardware used and should therefore be understood as a non-absolute indicator of the tool performance.

The results prove the consistency in terms of output, with a number

Table 4
Summary of tests performed, devices used, time taken and files exported.

Test #	Device	DatasetID	TransformsID (number of actual transformations)	Preview time (s)	Saving time (s)	Augmented patients (#)
1	CPU	D1	T1 (2)	2.37	24.97	24
2	CPU	D1	T2 (4)	3.46	36.06	48
3	CPU	D1	T3 (8)	5.55	57.87	96
4	CPU	D2	T1 (2)	1.05	21.2	60
5	CPU	D2	T2 (4)	1.55	28.98	120
6	CPU	D2	T3 (8)	2.47	44.99	240
7	CPU	D3	T1 (2)	1.07	122.58	358
8	CPU	D3	T2 (4)	1.72	188.61	716
9	CPU	D3	T3 (8)	2.55	274.12	1432
10	GPU	D1	T1 (2)	2.48	24.17	24
11	GPU	D1	T2 (4)	3.89	40.13	48
12	GPU	D1	T3 (8)	6.04	56.31	96
13	GPU	D2	T1 (2)	1.05	20.02	60
14	GPU	D2	T2 (4)	1.6	30.9	120
15	GPU	D2	T3 (8)	2.62	45.26	240
16	GPU	D3	T1 (2)	1	132.23	358
17	GPU	D3	T2 (4)	1.73	217.11	716
18	GPU	D3	T3 (8)	2.5	265.95	1432

of new cases and files generated that are proportional to the transformations used. With regard to the preview times, it can be observed that they are relatively short. This is because the transformations are applied to a single exemplary volume, which allows the parameters to be promptly fine-tuned until the user is satisfied, thus precisely obtaining the desired behavior.

The execution time increases in proportion to the complexity of the experiments, exhibiting no unexpected behaviors. Additionally, the results indicate that the use of the GPU does not consistently outperform the use of the CPU. This is because, for a few transformations, the time required for the transfer of data between devices may exceed the benefits gained, and the system still relies on the CPU for disk I/O operations. Therefore, for simple tasks, using the GPU may result in a time penalty.

This is visible in the comparison between experiment #9 and experiment #18, executed respectively on CPU and GPU. In this case, the computation on the CPU takes 10 s less than the GPU implementation. This outcome suggests that the benefits of GPU usage are only evident when an important number of images and transformations are involved.

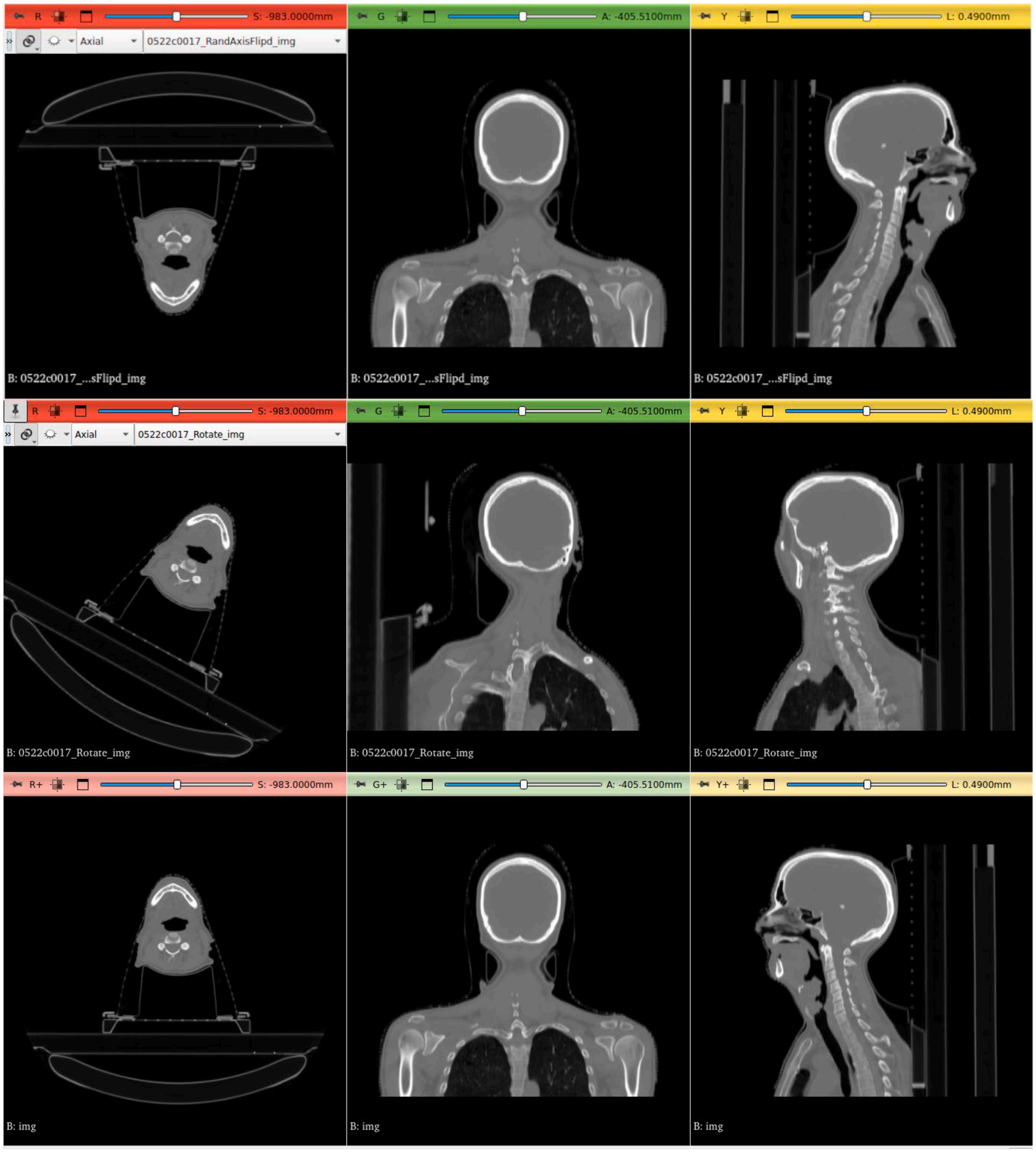


Fig. 3. Original image (bottom) compared to Rotate result (middle) and to Random Flip result (top).

Table 5
Comparison of original and transformed volumes.

	Dimensions	Spacing	Origin	Scan Order	Scalar type	Scalar Range
Original image	512 × 512 × 156	0.98mm **x** 0.98mm **x** 2.50mm	250.39mm **x** -155.61mm **x** -1178.00mm	Axial IS	float	-1024 **to** 2117.14
Rotated image	512 × 512 × 156	0.98mm **x** 0.98mm **x** 2.50mm	250.39mm **x** -155.61mm **x** -1178.00mm	Axial IS	float	-1024 **to** 2117.14
Flipped image	512 × 512 × 156	0.98mm **x** 0.98mm **x** 2.50mm	250.39mm **x** -155.61mm **x** -1178.00mm	Axial IS	float	-1024 **to** 2117.14

5.1. *Exemplary output*

This section illustrates how the MONAI Transforms applied with ImageAugmenter maintain anatomical consistency by comparing the original image (bottom) and the result obtained (top) by applying the T1 transformations (rotation and flip) to a case of the D1 dataset (Fig. 3). The original and the transformed volumes are then manually compared using the volume geometry metadata available in the "Volumes" module of 3D Slicer (Table 5).

6. Conclusions

In this paper, we presented *ImageAugmenter*, a 3D Slicer module that provides an intuitive user interface for executing data augmentation on medical images. It is designed to be flexible and versatile, supporting a wide range of MONAI Transforms and offering the users precise control over the transformation parameters. In addition, it guarantees the application of the same random transforms to each pair of anatomical and segmentation volumes, as well as the support of PyTorch compatible GPUs to decrease computation time.

The performance of ImageAugmenter was evaluated through a series of benchmarks that measured the time required for previewing and saving images as the image size, number of images, and transformations increased. These tests, conducted on both the CPU and the GPU, demonstrated that the number of generated cases and files was consistently proportional to the applied transformations. The preview times were short, allowing for quick parameter adjustments. It was found that the module does not alter dimensions, spacing, origin, scan order, scalar type and the scalar range of the volumes. Therefore, the consistency of the results depends on the user's parameter choices.

The tool allows users to visually assess the effect of data augmentation transformations and to save the augmented data to be used for analysis and AI models training. This 3D Slicer module would represent a powerful tool for researchers, clinicians, biologists, and any user who intends to use data augmentation techniques for improving the accuracy, generalizability, and robustness of their medical DL models, in a broad accessible way, without writing code and by simply using a graphical user interface.

Statement

During the preparation of this work the author(s) used Google's tool Gemini in order to improve language and readability. After using this tool, the author(s) reviewed and edited the content as needed and take (s) full responsibility for the content of the publication.

CRediT authorship contribution statement

Ciro Benito Raggio: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Paolo Zaffino:** Writing – review & editing, Visualization, Validation, Supervision, Software, Conceptualization, Methodology. **Maria Francesca Spadea:** Writing – review & editing, Supervision, Resources, Investigation, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

None.

Data availability

All data used to test the software are public and cited in the article.

References

[1] Chan HP, Samala RK, Hadjiiski LM, Zhou C. Deep learning in medical image analysis. Adv Exp Med Biol 2020;1213:3–21. https://doi.org/10.1007/978-3-030-33128-3_1.

[2] Yang YY, Ho MY, Tai CH, Wu RM, Kuo MC, Tseng YJ. FastEval Parkinsonism: an instant deep learning–assisted video-based online system for Parkinsonian motor symptom evaluation. NPJ Digit Med 2024;7(1). https://doi.org/10.1038/s41746-024-01022-x.

[3] Paciorek AM, Claudio FSC, et al. Automated assessment of cardiac pathologies on cardiac MRI using T1-mapping and late gadolinium phase sensitive inversion recovery sequences with deep learning. BMC Med Imaging 2024;24(1). https://doi.org/10.1186/s12880-024-01217-4.

[4] Lee SJ, Oh HJ, Son YD, et al. Enhancing deep learning classification performance of tongue lesions in imbalanced data: mosaic-based soft labeling with curriculum learning. BMC Oral Health 2024;24(1). https://doi.org/10.1186/s12903-024-03898-3.

[5] Kaur BP, H S, Hans R, et al. An augmentation aided concise CNN based architecture for COVID-19 diagnosis in real time. Sci Rep 2024;14(1). https://doi.org/10.1038/s41598-024-51317-y.

[6] Garcea F, Serra A, Lamberti F, Morra L. Data augmentation for medical imaging: a systematic literature review. Comput Biol Med 2023;152:106391. https://doi.org/10.1016/j.compbiomed.2022.106391.

[7] Rainio O, Klén R. Comparison of simple augmentation transformations for a convolutional neural network classifying medical images. Signal Image Video Process 2024. https://doi.org/10.1007/s11760-024-02998-5.

[8] Nolden M, Zelzer S, Seitel A, et al. The medical imaging interaction toolkit: challenges and advances. Int J Comput Assist Radiol Surg 2013;8(4):607–20. https://doi.org/10.1007/s11548-013-0840-8.

[9] Beare R, Lowekamp B, Yaniv Z. Image segmentation, registration and characterization in *R* with SimpleITK. J Stat Softw 2018;86(8). https://doi.org/10.18637/jss.v086.i08.

[10] A. Murray, H. Kemenade, wiredfool, et al. "Python-pillow/pillow": 10.3.0. Zenodo. 2024. https://zenodo.org/records/10903255.

[11] A. Paszke, S. Gross, F. Massa, et al. "*PyTorch: an imperative style, high-performance deep learning library*."; 2019. https://dl.acm.org/doi/10.5555/3454287.3455008.

[12] Pérez-García F, Sparks R, Ourselin S. TorchIO: a python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. Comput Methods Programs Biomed 2021;208:106236. https://doi.org/10.1016/j.cmpb.2021.106236.

[13] M. Jorge Cardoso, W. Li, R. Brown, et al." MONAI: an open-source framework for deep learning in healthcare." 2022. 10.48550/arxiv.2211.02701.

[14] Fedorov A, Beichel R, Kalpathy-Cramer J, et al. 3D Slicer as an image computing platform for the quantitative imaging network. Magn Reson Imaging 2012;30(9): 1323–41. https://doi.org/10.1016/j.mri.2012.05.001.

[15] Goceri E. Medical image data augmentation: techniques, comparisons and interpretations. Artif Intell Rev 2023. https://doi.org/10.1007/s10462-023-10453-z.

[16] Dadras AA, Jaziri A, Frodl E, Vogl TJ, Dietz J, Bucher AM. Lightweight techniques to improve generalization and robustness of U-net based networks for pulmonary lobe segmentation. Bioengineering 2023;11(1). https://doi.org/10.3390/bioengineering11010021. 21-21.

[17] Azizi S, Culp L, Freyberg J, et al. Robust and data-efficient generalization of self-supervised machine learning for diagnostic imaging. Nat Biomed Eng 2023;7(6): 756–79. https://doi.org/10.1038/s41551-023-01049-7.

[18] Selvaraj KM, Gnanagurusubbiah S, Roy R, Hephzipah J, Balu S. Enhancing skin lesion classification with advanced deep learning ensemble models: a path towards

accurate medical diagnostics. Curr Probl Cancer 2024. https://doi.org/10.1016/j.currproblcancer.2024.101077. 101077-101077.

[19] Qu G, Lu B, Shi J, et al. Motion-artifact-augmented pseudo-label network for semi-supervised brain tumor segmentation. Phys Med Biol 2024;69(5). https://doi.org/10.1088/1361-6560/ad2634. */Physics in medicine and biology*055023-055023.

[20] Raudaschl PF, et al. Evaluation of segmentation methods on head and neck CT: auto-segmentation challenge 2015. Med Phys 2017;44(5):2020–36. https://doi.org/10.1002/mp.12197. Apr.

[21] Thummerer A, et al. SynthRAD2023 grand challenge dataset: generating synthetic CT for radiotherapy. Med Phys 2023;50(7):4664–74. https://doi.org/10.1002/mp.16529. June.