# Software for anomaly detection in MRI images*

Yurii Oliinyk[a,*,†], Mariia Kapshuk[a,†], Leonid Oliinyk[b,†]

[a] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Beresteiskyi Ave. 37, Kyiv, 03056, Ukraine*
[b] *Kaunas University of Technology, K. Donelaičio St. 73, Kaunas, 44249, Lithuania*

**Abstract**

This article discusses the development of the algorithm and software for detecting the anomalies in MRI images. The main goal of the research is to increase the efficiency and the rate of disease diagnosis by creating a user-friendly and functional application for automatic anomaly detection. Anomaly detection algorithms were researched and software efficiency was tested. .NET platform, HD-BET tool, and ResNet model were chosen for the software development.

**Keywords**

Anomaly detection, neural networks, machine learning, software, MRI

## 1. Introduction

Magnetic resonance imaging (MRI) is a modern examination method that uses magnetic fields and radio waves to create detailed images of internal organs and tissues. The strong magnetic field created by an MRI scanner causes the atoms in the body to align in one direction. Radio waves are then sent from the MRI machine and move these atoms from their original position. When the radio waves are turned off, the atoms return to their original position and send radio signals. These signals are received by the computer and converted into an image of the examined body part, and the image appears on the monitor [1].

The main advantage of MRI compared to other methods, such as computed tomography or X-ray examination, is that it does not require the use of ionizing radiation, which makes it safe for patients. Also, MRI better defines the difference between types of soft tissues, and between normal and abnormal soft tissues [1].

Today, the development of IT technologies allows the use of artificial intelligence and machine learning to automate the analysis of medical images, including MRI images. In general, there are three main areas for the use of artificial intelligence in radiology: image reconstruction and enhancement, image classification and segmentation, and diagnostic support. The first direction is the most developed, while the other two are less studied, as they have more requirements for the accuracy of the results [2].

In terms of the classification of MRI images, with the help of machine learning, most studies focus on the detection of specific diseases or certain pathologies. For example, the detection of brain tumors. Such studies are more narrowly focused and more precise, but they require a large set of specific training data. Typically, these data studies use supervised learning, which requires labels on the data. Another direction is the detection of pathologies in general, comparing an unhealthy case with the norm. Such studies have a wider application, since they do not focus on specific cases, and can use different approaches.

However, there are some drawbacks to the current state of things. First of all, there is a small number of studies and their limited use. It is also important to take into account such disadvantages as the limited accuracy of some algorithms, the instability of the software and the high cost of some

*** *IDDM'24: 7th International Conference on Informatics & Data-Driven Medicine, November 14 - 16, 2024, Birmingham, UK*
* Corresponding author.
† These authors contributed equally.
oliyura@gmail.com (Y. Oliinyk); marichka.v.richtsi@gmail.com (M. Kapshuk) ; leonid.oliinyk@gmail.com (L. Oliinyk)
0000-0002-7408-4927 (Y. Oliinyk); 0009-0003-4429-3888 (M. Kapshuk) ; 0009-0000-9392-2271 (L. Oliinyk)

solutions. To improve the situation, it is worth introducing a light and open solution that would ensure reliability and have a low cost of implementation and maintenance.

That is why this study is dedicated to the development of software for detecting pathologies on MRI images with a convenient user interface and the use of unsupervised machine learning methods for anomaly detection. The development of the software aligns with the United Nations' Sustainable Development Goals, particularly the third goal: Good Health and Well-being. By enhancing early diagnosis and improving healthcare efficiency, this technology leads to better health outcomes, reducing mortality rates and enabling more equitable access to quality healthcare.

## 1.1. Related Work

While supervised DL learns to explicitly distinguish between what is normal and what is abnormal, unsupervised anomaly detection makes no assumptions about the concept of anomalies. Unsupervised methods either make no assumptions about the data at all and determine the probability of samples containing abnormal specimens, or have no knowledge of what exactly represents the abnormality, but clearly can determine the normal distribution of healthy anatomy [3].

The detection of anomalies can be attributed to the problem of clustering. One of the most used algorithms is K-Means clustering. This is a very simple algorithm that groups data into clusters based on similar features. It is also worth noting that K-Means is an algorithm that is based on distance and requires careful selection of parameters, in particular, the selection of features that should be taken into account when grouping. Although the K-Means algorithm is widely used for data clustering, it also has its drawbacks, especially when applied to the problem of anomaly detection. Primarily, K-Means assumes that all points in the data set belong to some cluster, making it unable to effectively recognize anomalous clusters that may be separated or misrepresented. Also, the algorithm is very sensitive to the presence of outliers in the input data. Large outliers can affect the location of the centroids and lead to incorrect clustering [4].

Another approach is to use an autoencoder (AE), another type of unsupervised learning. They consist of a so-called encoder, which maps a high-dimensional input signal to a low-dimensional hidden (latent) space, and a decoder, which has the task of restoring or reconstructing the input [3]. AEs can be different, but convolutional ones are the most common.

Variational autoencoders (VAE) are one of the variations of AE, which includes Bayesian variational methods in the usual architecture. The main difference is how they model the distribution of the input data in the latent space. The main disadvantage of AE is that the latent space can be extremely irregular. VAEs solve this problem because instead of a single point in the latent space, they return a distribution [3, 5].

Another important part of autoencoder modeling is the choice of the encoder and decoder architecture. Some of the most common models for feature extraction are the Convolutional Neural Network (CNN), Residual Neural Network (ResNet), and Recurrent Neural Network (RNN). The architecture of a regular CNN consists of an input layer that is subsequently convoluted using a combination of convolutional filters followed by an output layer. It is very efficient and widespread, but as the network depth increases and a certain threshold is reached, the error rate starts to increase because of the gradient vanishing and degradation. Another type is RNNs, which are designed to interpret temporal or sequential information. The main difference is that they reuse the activations of previous or subsequent nodes in the sequence. However, the problem of gradient vanishing remains in this type of networks. ResNet is a modification of CNN that introduces a residual learning function into the architecture. Therefore, unlike CNN, ResNet is able to solve the problem of accuracy loss, because this model additionally uses skip connection, i.e. direct connection, between layers. This allows it to take activation from one level and transfer it to another level, thus preserving parameters at deeper layers [6, 7].

## 1.2. Research Tasks

Main goals: improving the efficiency and speed of diagnosing diseases by creating a convenient and functional application for the automatic recognition of anomalies. The following tasks should be solved within this research:

- development of software for anomaly detection in MRI images
- development of the anomaly detection algorithm
- researching anomalies algorithm and software efficiency

# 2. The Software Development

The following describes the steps for the development of software for anomaly detection in MRI images.

## 2.1. Software Architecture

The software uses a layered architecture and consists of three projects (packages): MRI-Vision.Python, MRI-Vision.Domain and MRI-Vision.UI. The package diagram is shown in Figure 1.

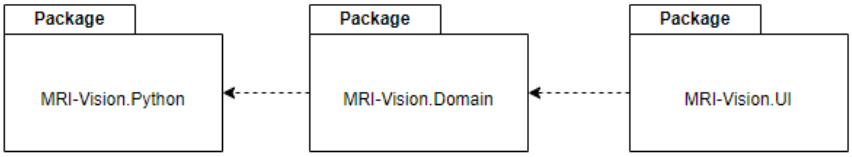


**Figure 1:** UML Package Diagram

The MRI-Vision.Python project is small and handles the model, images, and image analysis. It primarily includes classes for designing, creating, and using an autoencoder model as well as its training, which includes working with images – reading, preprocessing, and analysis.

The MRI-Vision.Domain project provides the functionality to process the results obtained from MRI-Vision.Python. It uses the third-party Python.NET project in order to integrate Python code into .NET. Additionally, since all methods calling Python code are called asynchronously and due to the GIL (Global Interpreter Lock) in Python, there is a separate module that performs all Python tasks in one thread.

The MRI-Vision.UI package contains only pages and additional GUI functionality for presenting the processed results obtained from MRI-Vision.Domain.

## 2.2. IDE and Additional Software Libraries

The most important part of creating a neural network is selecting the language and framework to be used in the development. Python is the most widely used programming language in the industry with a large selection of frameworks and libraries. The choice of a framework which will be used to train the neural network is another part. The most common in the field are Keras, PyTorch, and TensorFlow. Keras is a high-level interface that makes it easy to quickly build and train the models. TensorFlow offers more flexibility and control, making it popular among experienced researchers. PyTorch combines the simplicity of Keras with the flexibility of TensorFlow, offering dynamic graph definition and GPU computing acceleration. Moreover, PyTorch provides a user-friendly interface for working with dynamic graphs, simplifying the process of changing the model architecture during experiments. Due to the mentioned facts, PyTorch was chosen for the development.

Yet another crucial part of neural network training is data preprocessing. MRI data are usually represented as 3D models converted into slices. Many auxiliary libraries are widely used in DL and the medical field. One such library is the open-source MONAI platform, which offers a selection of optimised implementations of various DL algorithms and utilities specifically designed for medical visualisation tasks. It provides tools for flexible preprocessing of multidimensional medical image data [8]. Compared to NiBabel, which simply offers read-and-write access to common medical image formats like NIfTI and DICOM, MONAI is more feature-rich [9].

When working with MRI brain images, another useful tool for preprocessing is technologies that offer functionality for automated brain extraction of consecutive MRI images. Several alternatives perform this task using artificial neural networks [10]. A full comparison of known alternatives is presented in Table 1.

**Table 1**
Comparison of Tools for Brain Extraction of Sequential MRI Images

| Tool | Velocity | Accuracy |
|---|---|---|
| HD-BET | High | High |
| MONSTR | Medium | High |
| 3DSkullStrip | Medium | Medium |
| BEaST | Low | Medium |
| BSE | Medium | Low |
| BET | High | Low |

For this work, HD-BET was chosen since it is more accurate and faster than other alternatives such as MONSTR, 3DSkullStrip, BEaST, etc [10].

Given that the software is aimed at easy use by healthcare professionals, it is essential to develop a user-friendly and straightforward graphical interface. Although Python provides a wide range of libraries for GUI development, the C# programming language was selected due to its better performance, especially for applications that require fast real-time response. For this work, the Windows Presentation Foundation (WPF) framework was chosen since it uses a more modern XAML-based layout approach compared to its WinForms counterpart. There are two versions of WPF – on the .NET platform and on the .NET Framework platform. The .NET platform is used in this paper considering it has superior performance, availability, reliability, and tools [11].

Furthermore, Visual Studio Code for Python development was selected as an IDE since it is lightweight and has a large selection of additional extensions for viewing images, opening TensorBoard in order to view training statistics, etc. For C# development, Visual Studio was chosen, which has a user-friendly interface for developing WPF applications. Examples of the user interface are shown in Figure 2 and Figure 3.

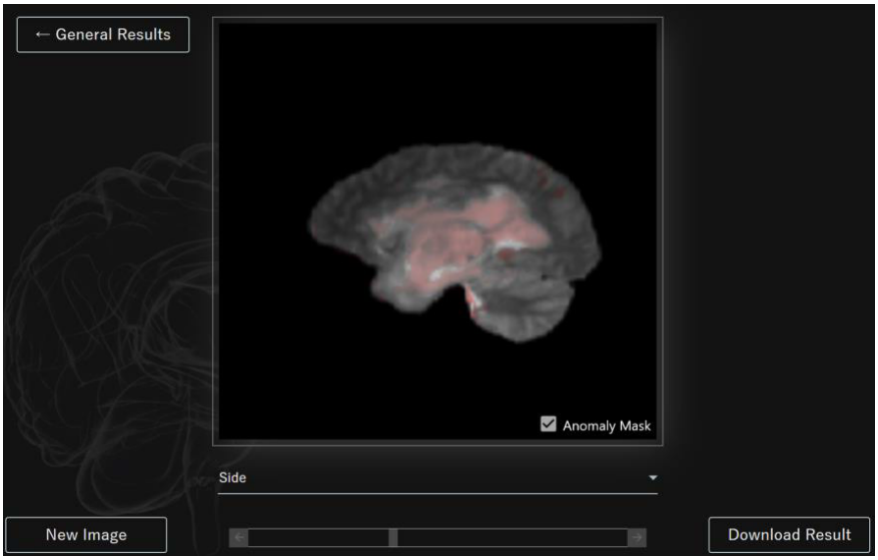


**Figure 2:** The View of the MRI Slice with an Anomaly Map


**Figure 3:** Orientation Change of the MRI Slice with an Anomaly Map

## 2.3. Choosing and Training of the Neural Network

Comparing the algorithmic solutions presented in the Related Work section in this paper the decision to design an autoencoder was made considering it has higher accuracy compared to K-Means. Additionally, since the model only needs to recognise anomalies without generating new ones, an autoencoder is a better choice than a variational autoencoder, given it is easier to learn and use. ResNet was chosen as the architecture of the autoencoder because of its advantage over conventional CNNs and RNNs in training deep neural networks for image feature recognition and extraction.

The neural network consists of an encoder, a decoder and a latent space between them. The encoder is composed of a convolutional input layer and five ResNet units for downsampling, each with three convolutional groups: two consecutive convolutional layers and a feedforward connection. Three consecutive steps make up each convolutional group – Conv3d convolution, ReLU activation, and BatchNorm3d normalisation. To stabilise the gradient and prevent the exploding gradient problem, normalisation is utilised. The number of channels in each block is 32, 64, 128, 256, and 512. Thus, the latent space is 1024. The architecture of the decoder is inversely symmetric, with upsampling layers used in place of downsampling ones.

The anomaly detection autoencoder is based on the fact that if the AC is trained only on healthy samples, the model will be able to reconstruct only them. Therefore, when an anomalous instance is received as input, the reconstruction error will be much higher in the anomalous areas, since the anomaly will not be reconstructed.

The model should be trained on healthy MRI brain images. Healthy samples were taken from the IXI dataset which contains almost 600 MRI brain images of normal, healthy subjects [12]. Two-thirds of the data are used for training, and the remaining data are used to evaluate the model after each epoch. The Adam optimiser with PyTorch was used for optimisation with a learning rate of $10^{-3}$. MSELoss, which measures the mean square error between the input and reconstructed data, was used to compute the loss. The training was performed on GPU and CPU by parallelising the data using PyTorch's nn.DataParallel class, which significantly reduced the training time.

During training, 400 epochs were performed, since after 350 epochs the loss almost did not decrease. Figure 4 shows a plot of the change in the value of the loss function after each epoch.


**Figure 4:** Plot of the Changes in the Loss Function

Thus, by the end of training, the loss function value was 0.0003. Moreover, after each epoch, if the model was determined to be the best of all the previous ones, it is saved to the file. Thus, the last best model was the model from the 388th epoch (Figure 4). It is this model that is used in the software.

## 2.4. Data Structure Description

The program accepts an MRI image as an input – a compressed file with the NIfTI (Neuroimaging Informatics Technology Initiative) extension. Therefore, the final extension looks like .nii.gz. This extension is widely used for storing MRI brain images. Its particular advantage for this application is that, unlike DICOM files, the NIfTI format stores much less metadata that is not needed for model training. This makes NIfTI files lighter in size and faster to read [13]. Also, this type of input data is set due to the use of HD-BET which only accepts this extension type as input. NIfTI images are registered in a local coordinate system, and each file contains metadata and 3D pixels in seven dimensions. Typically, NIfTI files have the extension .nii or .nii.gz and can be split into a binary header (.hdr) and image data (.img or .img.gz) [14]. Figure 5 shows a diagram of the general structure of an NIfTI file.

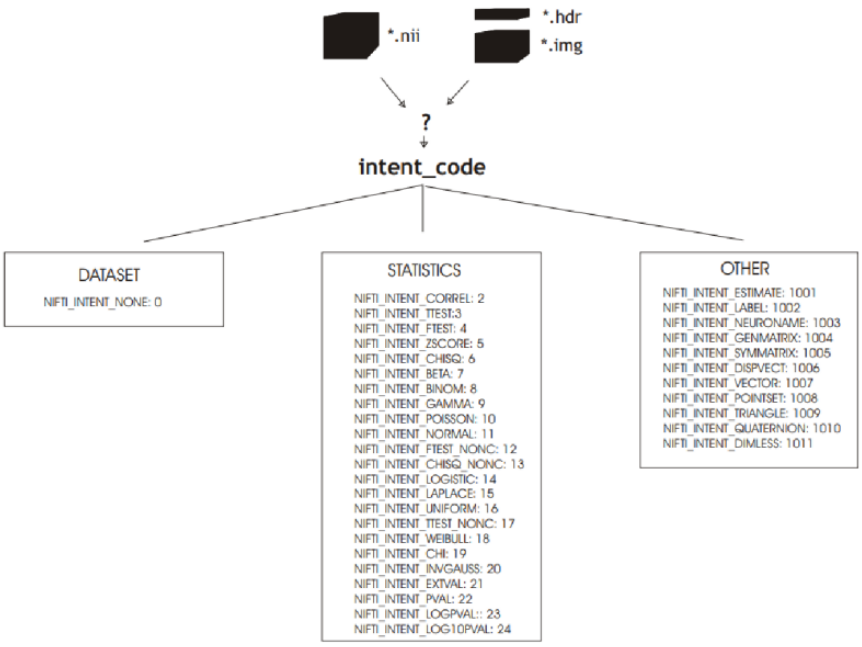


**Figure 5:** Data Structure of a NIfTI File

In addition, there are numerous different types of MRI images. By default, each examination includes T1-weighted and T2-weighted images, and other types such as FLAIR (FLuid Attenuation Inversion Recovery), DWI (Diffusion-Weighted Imaging), etc. may be included as well. Liquids appear dark in T1-weighted images, while grey matter appears darker than white matter. In contrast, fluids appear light in T2-weighted images, and grey matter appears lighter than white matter. Various pathologies, such as inflammation, necrosis, tumours, etc., cause high fluid content in these areas, making them prominent on T2-weighted images.

In this work, T2-weighted images were chosen since they are included in the standard set of images created during examinations and since various pathologies are clearly visible on them, unlike T1-weighted images.

As an output, the user receives analysis results which can be downloaded as PNG images.

## 2.5. Anomaly Detection Algorithm

The general steps of the algorithm used to detect anomalies on the selected file are as follows:

1. Upload a file
2. Change the image orientation (if necessary)
3. Resize the image (if necessary)
4. Collapse image data into latent space
5. Reconstruct an image from the latent space
6. Calculate the reconstruction error value as the absolute difference between the input and reconstructed images
7. Calculate the anomaly value of each image slice
8. Create images with anomalous areas for each image slice
9. Display the analysis results

Figure 6 shows the anomaly detection algorithm in a diagram form.

In the 6th step, the reconstruction error for each pixel is calculated using the following formula:

$$A'_{ijk} = |I_{ijk} - R_{ijk}|, \quad (1)$$

where $I_{ijk}$ is the pixel value of the input image, $R_{ijk}$ is the pixel value of the reconstructed image.

In the 7th step, the total anomaly value for each slice is calculated using the following formula:

$$A_i = \frac{\sum A_{ijk}}{N - n(A_{ijk})}, \tag{2}$$

where $A_i$ is the slice anomaly value, $A_{ijk}$ is the pixel anomaly, $\mathrm{n}(A_{ijk})$ is the number of anomalous pixels, $N$ is the number of pixels in the slice. Thus, slices with a larger anomaly area will have a higher anomaly value. The pixel anomaly value is obtained from the reconstruction error using the following formula:

$$A_{ijk} = \begin{cases} A'_{ijk}, \text{if } A'_{ijk} > t \\ 0, \text{if } A'_{ijk} < t \end{cases}, \tag{3}$$

where $t$ is the maximum reconstruction error value that is allowed and is not considered an anomaly.

The algorithm produces a list of anomaly values for each slice, which is presented in the form of a line chart and an image of the slices with the anomalous areas highlighted.

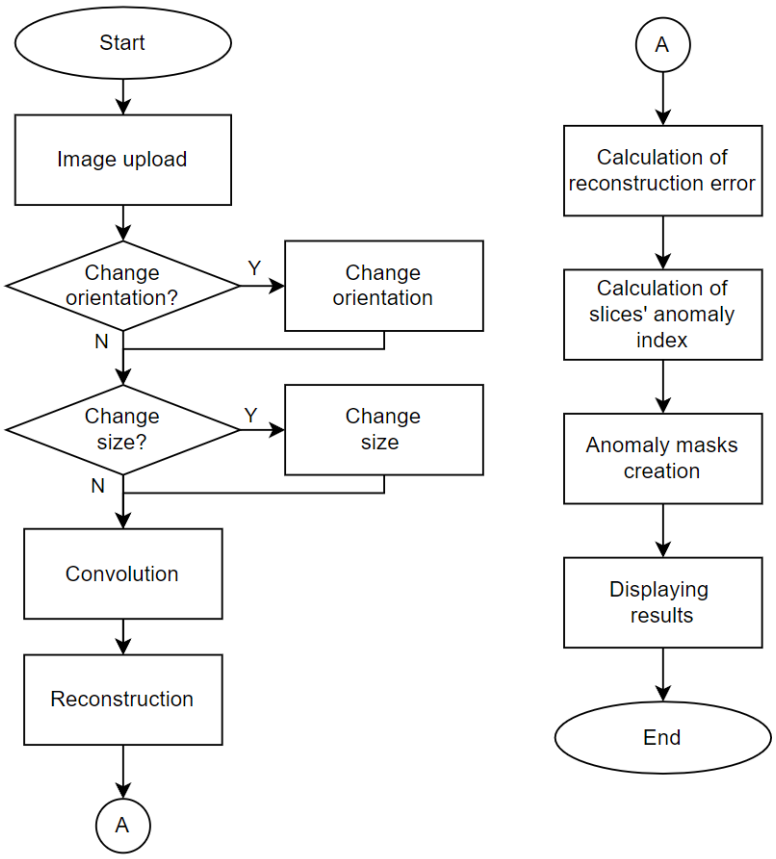


**Figure 6:** Anomaly Detection Algorithm

## 3. The Software and Algorithm Efficiency

The following describes the efficiency of software and the algorithm for anomaly detection in MRI. It is necessary to conduct a number of experiments to study the efficiency, accuracy, and conciseness of the developed computer software:

- evaluating the algorithm's accuracy, precision, recall and F1 score
- calculating model training speed
- calculating image processing speed

For research, the hardware with the following characteristics was used: AMD Ryzen 5 5600H CPU 3301 Mhz, 16 GB RAM, NVIDIA Tesla T4 GPU. The experiments were carried out on the operating system Windows 11.

## 3.1. Researching the Accuracy of Anomaly Detection

The BraTS2020 dataset, which consists of a large number of clinically validated glioblastoma and glioma brain MRI images, was used to determine this metric. The dataset contains NIfTI files (.nii.gz) of type T1, T1Gd, T2, FLAIR and a file with segmented pathology, which were selected manually by specialists [15, 16, 17].

To determine the accuracy of the model, 369 files from the dataset were analyzed and the value of the reconstruction error was obtained for each of them. The average reconstruction error, obtained when using the model on fully healthy specimens, was then subtracted from each reconstruction error to obtain an image of the anomaly. Healthy specimens, on which the model was also trained, were taken from the IXI dataset [12]. The obtained results were then compared with the segmented image for the same image. The use of only IXI and BraTS2020 datasets in the study may limit the findings' general applicability. To improve the results' generalizability, future research should include more datasets that represent a broader range of MRI imaging and clinical situations.

The results were recorded as:

- correctly detected abnormal areas (true positive value)
- incorrectly detected abnormal areas (false positive value)
- correctly detected healthy areas (true negative value)
- incorrectly detected healthy areas (false negative value)

Precision shows the number of correctly selected abnormal areas to all selected abnormal areas, so it indicates how many of the abnormal areas it classified correctly. Recall indicates the number of correctly selected anomalous areas to all areas that should be selected as anomalous, that is, how many anomalous areas were selected and not missed. The F1 score is a harmonic mean between precision and recall, and shows how accurate and reliable the result is. Accuracy shows the extent to which the abnormality map of the MRI image in general corresponds to the truth. All metrics take values between 0 and 1, with higher values implying better results.

Table 2 shows the values of the metrics in percentages.

**Table 2**
Value of metrics in percentage

| Metric | Value |
|---|---|
| Precision | 69.45% |
| Recall | 82.01% |
| F1 score | 75.21% |
| Accuracy | 89.07% |

Thus, the software identifies 82% of anomalies, and the overall accuracy of detecting anomalies in images is 89%.

## 3.2. Evaluating Software Efficiency

To research software efficiency, two test runs were performed: model training of 5 epochs and processing of 200 MRI images. Additionally, using GPU increases model training speed by more than 15 times, but at the same time increases image processing speed only by 22%.

**Table 3**
Software speedup efficiency

| Used decision | Model training, s (5 epochs) | Image processing speed, s (200 images) |
|---|---|---|
| CPU | 6556 | 472 |
| CPU+GPU | 420 | 385 |

## 4. Discussion and Conclusion

The result of this research was the development of the algorithm and the convenient and functional application for the automatic detection of anomalies on MRI images. The software's overall accuracy in detecting anomalies in images is 89%, with 82% of anomalies detected. The total time of analysis takes less than 7 seconds, proving the software's efficiency and speed. The software also has a user-friendly and intuitive user interface that makes it easy to view MRI images and analysis results.

The anomaly detection algorithm is based on the fact that if AE is trained only on healthy samples, then the model will be able to reconstruct only them. Therefore, when an instance with an anomaly is received at the input, the reconstruction error will be much higher in the anomalous areas, since the anomaly will not be reconstructed correctly. Thus, having the reconstruction error for each pixel, it is possible to determine how large its abnormality value is for each slice, which is then can be presented in the form of a line chart and an image of the slices with the anomalous areas highlighted.

The software uses a multi-layered architecture and consists of three projects (packages): MRI-Vision.Python, MRI-Vision.Domain, and MRI-Vision.UI.

The model was trained on healthy MRI brain images. Healthy samples were taken from the IXI dataset which includes almost 600 MRI brain images of normal, healthy subjects. Two-thirds of the data are used for training, and the remaining data are used to evaluate the model's performance after each epoch. The training was performed on GPU and CPU by parallelising the data using PyTorch's nn.DataParallel class, which significantly reduced the training time. The 388th epoch model was the last best model. The anomaly detection algorithm consists of nine steps and produces a list of anomaly values for each slice, which is presented in the form of a line chart and an image of the slices with the anomalous areas highlighted. Using GPU increases model training speed more than 15 times, but at the same time, it increases image processing speed by only 22%.

The software's further development includes support for more MRI image file formats, for example, the support for DICOM files. Extending the neural network library to analyse MRI images of other parts, such as images of the spinal cord, various joints, chest, etc., is another promising improvement. Support for simultaneously analysing numerous files would be a further addition. A useful improvement to enhance the graphical user interface would be the ability to generate a 3D-projection and view the image in different angles in several windows. It is worth considering the possibility of multifactorial analysis using different types of images (T1, T2, FLAIR, etc.) in order to increase the software's accuracy.

To guarantee that the proposed software is practical and clinically relevant, extensive user testing and validation must be considered. in authentic clinical contexts. To enhance the evidence supporting the model's robustness, it is critical to evaluate its performance across a broader and more diverse dataset.

## Declaration on Generative AI

The authors have not employed any Generative AI tools.

# References


[1] Magnetic resonance imaging (MRI). National Institute of Biomedical Imaging and Bioengineering (NIBIB). URL: https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri.

[2] Gassenmaier S., Küstner T., Nickel D., Herrmann J., Hoffmann R., Almansour H., Afat S., Nikolaou K., Othman A.E. Deep learning applications in magnetic resonance imaging: has the future become present?. Diagnostics (Basel). 2021. V. 11, no. 12, p. 2181.

[3] Baur C. Anomaly detection in brain MRI: from supervised to unsupervised deep learning: Dissertation. Munich. 2021.

[4] Pierobon G. Guide to Machine Learning for Anomaly Detection. 2023. URL: https://drive.google.com/file/d/11cs1z8406O6CQ1AwP66tlLhGVMjkr743/view

[5] Rocca J. Understanding variational autoencoders (VAEs). Towards data science. URL: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.

[6] Sharma V. ResNets: Why do they perform better than classic ConvNets?. Towards data science. URL: https://towardsdatascience.com/resnets-why-do-they-perform-better-than-classic-convnets-conceptual-analysis-6a9c82e06e53.

[7] TELUS International. What's the difference between CNN and RNN?. TELUS International. URL: https://www.telusinternational.com/insights/ai-data/article/difference-between-cnn-and-rnn.

[8] MONAI Consortium. Project MONAI. MONAI. URL: https://docs.monai.io/en/stable/index.html.

[9] NiBabel. NIPY. URL: https://nipy.org/nibabel/.

[10] Isensee F., Schell M., Pflueger I. Automated brain extraction of multisequence MRI using artificial neural networks. Human Brain Mapping. 2019. V. 40, no. 17, p. 4952-4964.

[11] Desktop Guide (WPF .NET). Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-8.0.

[12] IXI Dataset. Biomedical Image Analysis Group, Imperial College London. URL: https://brain-development.org/ixi-dataset/

[13] What's the Difference Between DICOM and NIfTI?. ENCORD BLOG. URL: https://encord.com/blog/dicom-and-nifti-comparison/

[14] NIfTI-1 Data Format. NIfTI. URL: https://nifti.nimh.nih.gov/nifti-1

[15] The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS) / B. H. Menze et at. IEEE Transactions on Medical Imaging. 2015. V. 34, no. 10, p. 1993–2024.

[16] Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features / S. Bakas et al. Scientific Data. 2017. V. 4, no. 1.

[17] Automatic Brain Tumor Segmentation and Overall Survival Prediction Using Machine Learning Algorithms / E. Carver et al. Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. Cham, 2019. P. 406–418.