Ordu Üniversitesi Bilim ve Teknoloji Dergisi
Ordu Üniv. Bil. Tek. Derg.

Ordu University Journal of Science and Technology
Ordu Univ. J. Sci. Tech.

E-ISSN: 2146-6459 | 2025, 15 (1), 88-100 | Araştırma Makalesi / Research Article

https://doi.org/10.54370/ordubtd.1651665

DeepImageSegmentationApp: Deep Learning Application for Image Segmentation

Lütfü Bayrak[1], Kenan Koçkaya[1], Ahmet Çınar[2]

[1]Sivas Cumhuriyet University, Divriği Nuri Demirağ Vocational School, Department of Motor Vehicles and Transportation Technologies, Sivas
[2]Fırat University, Faculty of Engineering, Department of Computer Engineering, Elazığ

Geliş Tarihi / Received Date: 05.03.2025 Kabul Tarihi / Accepted Date: 04.06.2025

Abstract

There are many methods to examine a specific region or object in images. One of the most important of these methods is image segmentation. Image segmentation involves dividing images (or video frames) into multiple sections or objects. There are many different model architectures developed in the field of image segmentation. In this study, a deep learning-based image segmentation application interface has been developed. The performance of the proposed application has been analyzed on the Covid19 dataset obtained from Kaggle. The performance results of the application are presented comparatively on the U-NET and V-NET models with known accuracy for different system parameters. In the analysis results, it is clearly seen that the V-NET architecture is better than the U-NET architecture. The developed application environment has revealed the difference between the models and the usability of the application environment. This standalone software can be downloaded at: https://github.com/lbayrak/DeepImageSegmentationApp.

Keywords: image segmentation, deep learning, U-Net, V-Net

DeepImageSegmentationApp: Görüntü Segmentasyonu için Derin Öğrenme Uygulaması

Öz

Görüntülerde belirli bir bölgeyi veya nesneyi incelemek için birçok yöntem vardır. Bu yöntemlerin en önemlilerinden biri görüntü segmentasyonudur. Görüntü segmentasyonu, görüntüleri (veya video karelerini) birden fazla bölüme veya nesneye ayırmayı içerir. Görüntü segmentasyonu alanında geliştirilen birçok farklı model mimarisi vardır. Bu çalışmada, derin öğrenme tabanlı bir görüntü segmentasyonu uygulama arayüzü geliştirilmiştir. Önerilen uygulamanın performansı Kaggle'dan elde edilen Covid19 veri kümesi üzerinde analiz edilmiştir. Uygulamanın performans sonuçları, farklı sistem parametreleri için bilinen doğrulukla U-NET ve V-NET modelleri üzerinde karşılaştırmalı olarak sunulmuştur. Analiz sonuçlarında V-NET mimarisinin U-NET mimarisine göre daha iyi olduğu açıkça görülmektedir. Geliştirilen uygulama ortamı modeller arasındaki farkı ve uygulama ortamının kullanışlılığını ortaya koymuştur. Bu bağımsız yazılım şu adresten indirilebilir: https://github.com/lbayrak/DeepImageSegmentationApp.

Anahtar Kelimeler: görüntü bölümlendirme, derin öğrenme, U-Net, V-Net

Sorumlu Yazar/ Corresponding Author: Kenan Koçkaya kkockaya@cumhuriyet.edu.tr

Introduction

Deep learning is a subset of machine learning methods based on artificial neural networks using multiple layers (LeCun et al., 2015). Deep neural networks (DNN) have been successful in many areas such as computer vision (Krizhevsky et al., 2017) or natural language processing (LeCun et al., 2015) in recent years. In academic studies, DNNs have been widely used in many areas, including medical image segmentation, and have shown successful results. Image segmentation is the process of dividing a digital image into multiple parts. In deep learning-based image segmentation processes, masks belonging to these images are usually used together with the images to be processed. This technique provides a much more detailed understanding of the objects in the image. Some studies in the field of deep learning-based image segmentation are as follows; Yong Peng et al. (2023) proposed a cluster boundary segmentation based on convolutional neural networks and watershed algorithm to quickly determine the boundaries of clusters in asphalt mixture CT images. In the study, it is stated that the proposed intelligent image segmentation algorithm gives more accurate results than canny and multi-threshold algorithms in finding the boundaries.

Gupta and Mishra (2024) conducted a deep learning-based image segmentation study on early detection of colorectal cancer. The study presents statistical analysis of Unet, R2Unet, Attention-based Unet, ResUnet, Unet 3+, TransUnet and SwinUnet models based on polyp datasets and performance measurements.

Li et al. (2024) conducted a deep learning-based image segmentation study on the examination of remote sensing (RS) images. In the study, a comprehensive review was made on remote sensing research systems and various specially designed deep learning models.

Dang et al. (2024) conducted a deep learning-based image segmentation study on CVC-ColonDB, a general polyp dataset consisting of 300 images, combining multiple DNN models using ensemble learning.

Zi et al. (2024) conducted a research study on the application of deep learning in the field of medical image segmentation and 3D Reconstruction. The model created in the study they developed is validated by experiments and performance analysis comparisons are made with other deep learning models.

Xu et al. (2024) conducted a deep learning-based image segmentation study on the effect of data quality and quantity of concrete cracks. The study proposes a dataset comparison for pixel-level segmentation of concrete cracks based on the deep learning model.

Zhang et al. (2024) proposed the deep learning-based image segmentation model CNet for the examination of coral reefs on the seabed. The study emphasizes the importance of coral reefs and includes the comparison of the developed model with other models and its performance values for segmentation of reef images.

Liu et al. (2024) proposed the Dual-Path Dual Attention Transformer (DDA-Transformer) model, a deep learning-based image segmentation model, to obtain precise and fast knee joint in robotic-assisted total knee arthroplasty. In the study, the performance and speed of the DDA-Transformer model compared to other models were examined and comparisons were made.

Wang et al. (2024) conducted a deep learning-based image segmentation study to quantify fire power with flame images. In the study, the determination of progress and flame area on fire images was examined and it was aimed to contribute to the development of AI-based fire response systems in the future.

As can be understood from the segmentation studies mentioned above, deep learning-based image segmentation can be used in many areas.

Related Works

Ye et al. (2022) developed an application called DeepImageTranslator for beginners in the field of deep learning-based image segmentation. The developed application was written in the Python programming language and a visual interface was created using the tkinter library.

Deep learning-based image segmentation models are basically related to the knowledge level of the developer. Thanks to experience and knowledge, the deficiencies of the models are seen. In addition, the development of the models is determined by the observations obtained at runtime. For this reason, facilitating the immediate testing of the model codes when they are edited is of great importance in the development of the models and the elimination of their deficiencies. The main purpose of developing this application was to facilitate the basic difficulties encountered in the development of image segmentation models. There are a number of problems encountered while developing a segmentation model. These are;

- Comparing the model with a target model to test its performance
- Eliminating the deficiencies of the model and increasing its success rate compared to the target model
- Testing the performance of the developed model on different datasets
- Comparing the results of the developed model according to different performance criteria and storing the results as numerical data
- Storing visual outputs and reusing them in the future without the need for training
- Enabling the application to re-run the developed model codes in real time
- Comparing the similarity rates of the obtained prediction data on real data are many other problems.

People who work on image segmentation professionally want to intervene in the source code while making the necessary edits and changes. Thanks to the ability to intervene in the source code, developers can work independently of the restrictions of the application environment and perform more successful work.

While developing this application environment, it was aimed for people working in this field to intervene in the codes as they wish and to work comfortably on the datasets they want. For this reason, the application environment was developed with an understandable and easy-to-read coding. Since the target here is professional segmentation model developers, an environment was developed where model development is provided by programmers in order to avoid the limitations of drag-and-drop coding.

Material and Method

In this study, DeepImageSegmentationApp application was developed for deep learning image segmentation of 2D CT images. The application environment was written in Python programming language and user interfaces were created using tkinter and customtkinter libraries. The interface designs were aimed to be user-friendly, simple and understandable. The training and prediction processes of the models were performed using TensorFlow and Keras. While developing the application environment, it was aimed to eliminate the following problems with deep learning image segmentation. These are;

- The inability to run different models sequentially and obtain the results before the training of all models is completed
- The ability to store the results of model trainings based on different parameters without the need for retraining
- The ability to visually compare the prediction data with the real data
- The ability to dynamically reflect the graphical results and include them in the comparison when a different model training is performed

- The ability to easily separate the training and test data by partitioning within the entire dataset
- The ability to create different versions of the developed models
- The ability to compare the data of the different developed versions without the need for constant rerunning and the ability to store the numerical and visual results of the achievements
- The ability to create and run a different version by intervening in the code at runtime, and many more elements could be included.

In order to eliminate the problems and difficulties mentioned above while developing the application and to make it easy to use, the application environment is basically built on three interfaces. These are;

1. Main application environment (UserInterface)
2. Visual interface for comparing test data with prediction data (FigureInterface)
3. Visualization of training outcomes (DashboardInterface).

Loading Data

The first problem encountered in studies on image segmentation is that models are trained on different datasets. The first problem encountered in studies on different datasets is that the datasets are not at a suitable scale for training the model. Another problem encountered is separating the training and test data on the datasets.

A module was developed so that the application environment can work on the desired datasets. In the developed module, the processes of converting the data to the desired format and dimensions are carried out. The data loading process is carried out in the following stages in order to eliminate the difficulties that may be encountered on different datasets.

1. Determining the data path
2. Retrieving images from the data path
3. Rescaling the images
4. Converting the scaled images into Numpy lists as numerical matrices
5. Arranging the models for use.

Here, if desired, the images can be output and stored as numerical data. This part is left to the user's request. The parts related to the storage of numerical data depending on the user's request have been developed and made available within the application environment. The application environment, which was basically developed on 2D image data, can be easily adapted and used for 3D image data.

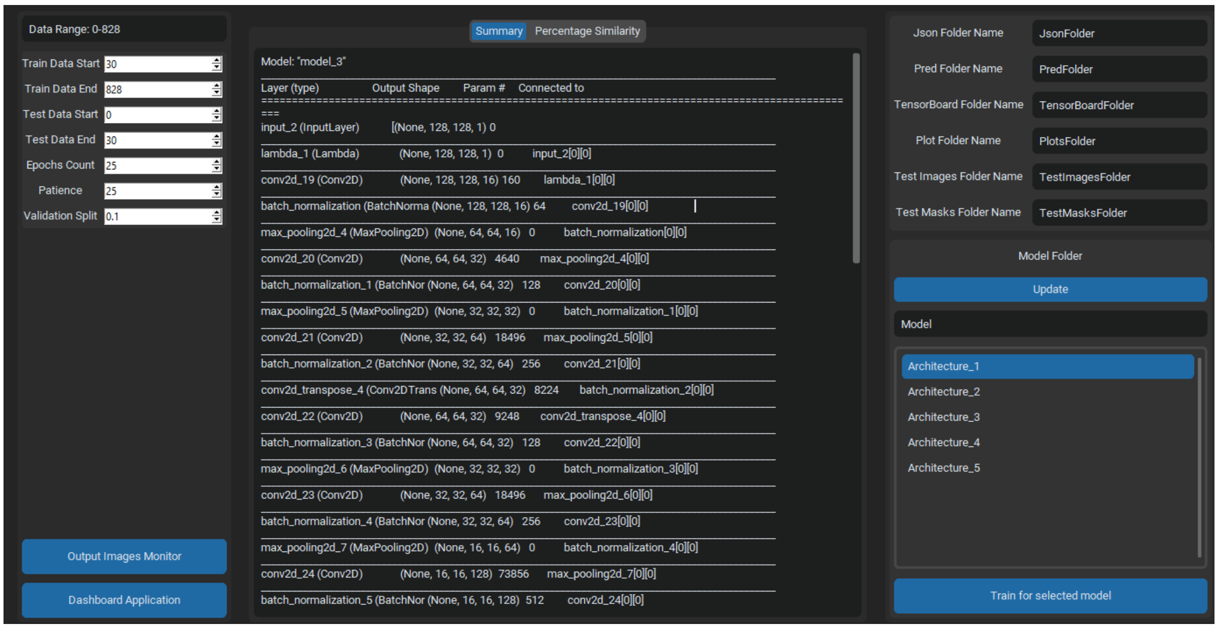
Main Application Interface (UserInterface)

The main application interface in Figure 1 is the interface environment where segmentation models are dynamically loaded into the system and trained. The outputs of the segmentation models trained in this application environment are stored in folders in the specified areas. The storage process allows the models to be compared with newly developed models without having to be trained again. The following settings can be made regarding the dataset and training process in the main application environment;

a. How much of the data will be used for the training process
b. How much of the data will be used for the test process
c. Number of Epochs
d. Patience
e. Validation Split

The storage of the training process of each model is done under the Train Results folder as "ArchitectureName Validation Split=… Train=… Test=… Epochs=… Patience=…". During this process, the storage of the numerical data belonging to the training results;

a. Numerical data: JsonFolder
b. Prediction results: PredFolder
c. Processed test data: TestImagesFolder
d. The processed test data masks are made in the form of: TestMasksFolder.



```
Model: "model_3"
_______________________________________________________________________________
Layer (type)              Output Shape       Param #    Connected to
===================================================================================================
===
input_2 (InputLayer)        [(None, 128, 128, 1) 0
_______________________________________________________________________________
lambda_1 (Lambda)           (None, 128, 128, 1)  0        input_2[0][0]
_______________________________________________________________________________
conv2d_19 (Conv2D)          (None, 128, 128, 16) 160      lambda_1[0][0]
_______________________________________________________________________________
batch_normalization (BatchNorma (None, 128, 128, 16) 64     conv2d_19[0][0]
_______________________________________________________________________________
max_pooling2d_4 (MaxPooling2D)  (None, 64, 64, 16)   0      batch_normalization[0][0]
_______________________________________________________________________________
conv2d_20 (Conv2D)          (None, 64, 64, 32)   4640     max_pooling2d_4[0][0]
_______________________________________________________________________________
batch_normalization_1 (BatchNor (None, 64, 64, 32)   128    conv2d_20[0][0]
_______________________________________________________________________________
max_pooling2d_5 (MaxPooling2D)  (None, 32, 32, 32)   0      batch_normalization_1[0][0]
_______________________________________________________________________________
conv2d_21 (Conv2D)          (None, 32, 32, 64)   18496    max_pooling2d_5[0][0]
_______________________________________________________________________________
batch_normalization_2 (BatchNor (None, 32, 32, 64)   256    conv2d_21[0][0]
_______________________________________________________________________________
conv2d_transpose_4 (Conv2DTrans (None, 64, 64, 32)   8224   batch_normalization_2[0][0]
_______________________________________________________________________________
conv2d_22 (Conv2D)          (None, 64, 64, 32)   9248     conv2d_transpose_4[0][0]
_______________________________________________________________________________
batch_normalization_3 (BatchNor (None, 64, 64, 32)   128    conv2d_22[0][0]
_______________________________________________________________________________
max_pooling2d_6 (MaxPooling2D)  (None, 32, 32, 32)   0      batch_normalization_3[0][0]
_______________________________________________________________________________
conv2d_23 (Conv2D)          (None, 32, 32, 64)   18496    max_pooling2d_6[0][0]
_______________________________________________________________________________
batch_normalization_4 (BatchNor (None, 32, 32, 64)   256    conv2d_23[0][0]
_______________________________________________________________________________
max_pooling2d_7 (MaxPooling2D)  (None, 16, 16, 64)   0      batch_normalization_4[0][0]
_______________________________________________________________________________
conv2d_24 (Conv2D)          (None, 16, 16, 128)  73856    max_pooling2d_7[0][0]
_______________________________________________________________________________
batch_normalization_5 (BatchNor (None, 16, 16, 128)  512    conv2d_24[0][0]
```

Figure 1. Main Application Interface (UserInterface)

Figure 2 shows the similarity graph of the prediction results obtained by the trained model on the test data. Thanks to this process, the similarity process on the test data can be evaluated and it will help to have an idea to eliminate the deficiencies of the model.

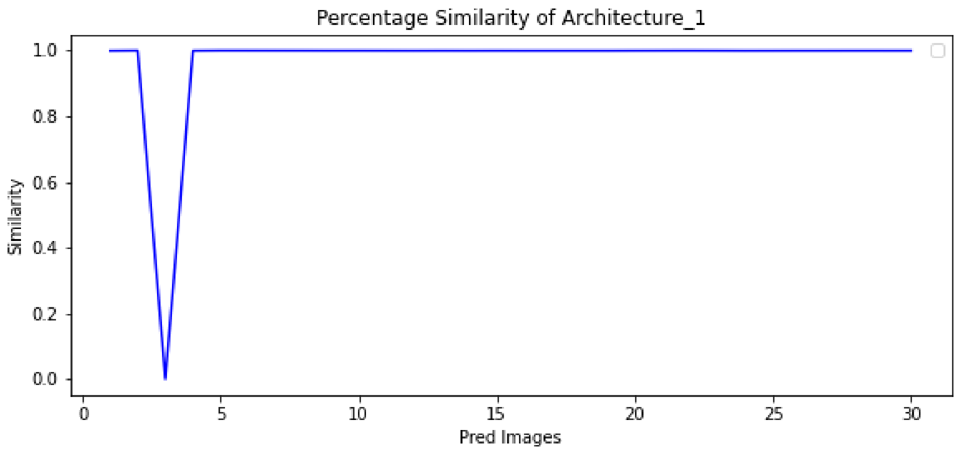


Figure 2. Similarity Values of the Predicted Data with the Real Data

Comparison of Test Data (FigureInterface)

The application environment where the similarity values of the predictions made on the test data after the training process with the original data are visualized is the FigureInterface environment.

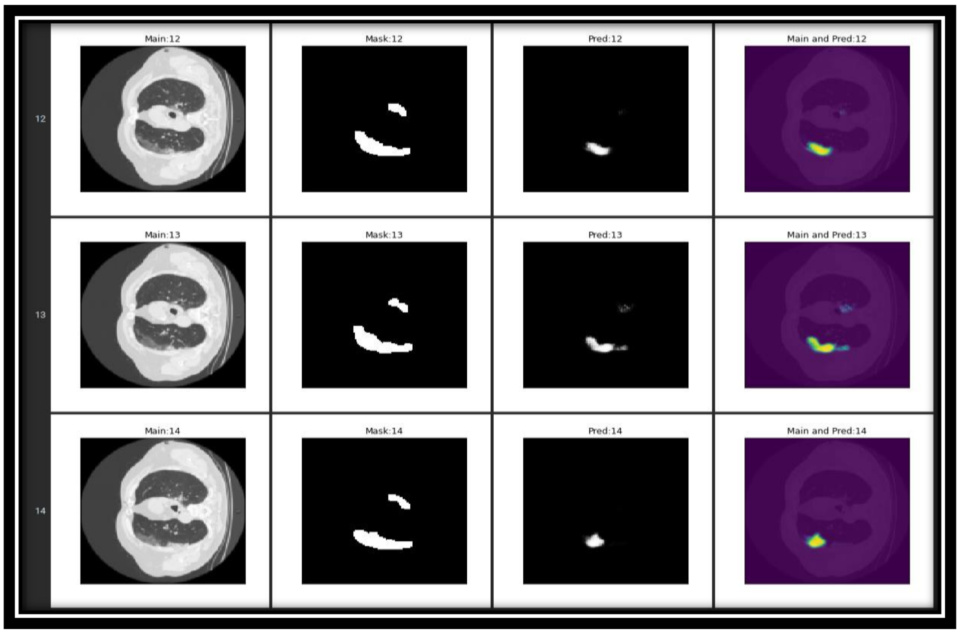


Figure 3. Listing of All Test Data and Predictions

In the FigureInterface environment, data is created by listing the real data and prediction data comparatively and displaying them individually. In Figure 3, the original lung image (Main:…), the mask of this image (the area where the disease is actually located, Mask:…), the model-predicted image (Pred:…) and the location of the predicted disease data on the lung image (Main and Pred:…) are listed. Only a part of the entire list is shown here, and all test data is included in the listing process. All prediction data here is stored in the PredFolder folder.

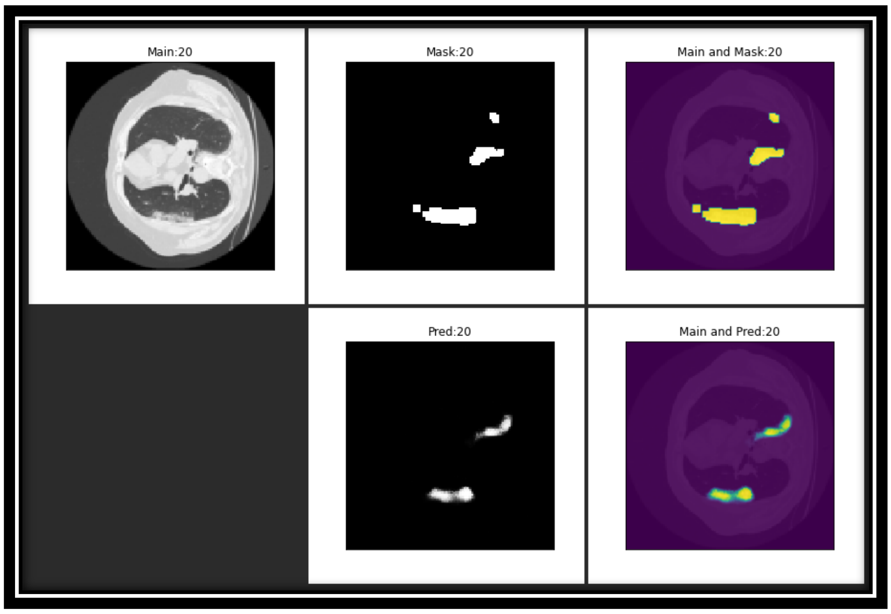
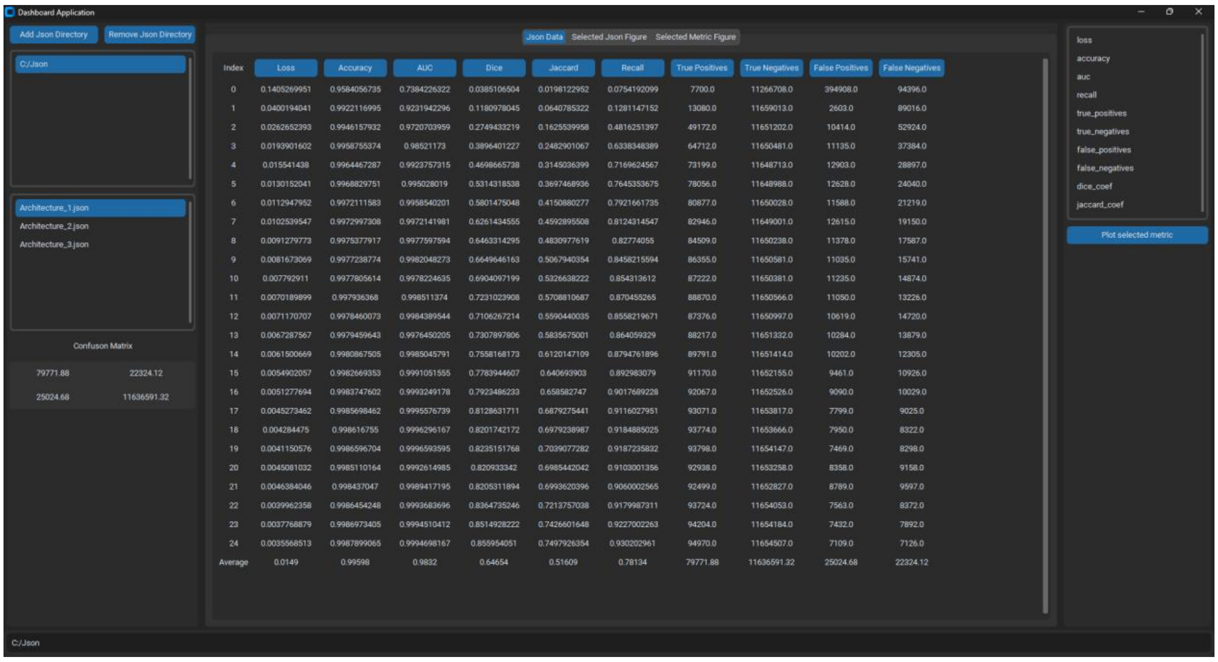


Figure 4. Prediction Data Number 20

The FigureInterface environment also includes a list for selecting the desired data. The results of the prediction data selected from the list are as shown in Figure 4. Here, the original lung image (Main: 20), the real region where the disease is located (Mask: 20), the real location of the disease on the original image (Main and Mask: 20), the prediction image (Pred: 20) and the location of the prediction image on the lung (Main and Pred: 20) are included.

Visualization of Training Outputs (DashboardInterface)



Confusion Matrix

79771.88	22324.12
25024.68	11636591.32

Index	Loss	Accuracy	AUC	Dice	Jaccard	Recall	True Positives	True Negatives	False Positives	False Negatives
0	0.1405269951	0.9584056735	0.7384226322	0.0385106504	0.0198122952	0.0754192099	7700.0	11266708.0	394908.0	94396.0
1	0.0400194041	0.9922116995	0.9231942296	0.1180978045	0.0640785322	0.1281147152	13080.0	11659013.0	2603.0	89016.0
2	0.0262652393	0.9946157932	0.9720703959	0.2749433219	0.1625539958	0.4816251397	49172.0	11651202.0	10414.0	52924.0
3	0.0193901602	0.9958755374	0.98521173	0.3896401227	0.2482901067	0.6338348389	64712.0	11650481.0	11135.0	37384.0
4	0.015541438	0.9964467287	0.9923757315	0.4698665738	0.3145036399	0.7169624567	73199.0	11648713.0	12903.0	28897.0
5	0.0130152041	0.9968829751	0.995028019	0.5314318538	0.3697468936	0.7645353675	78056.0	11648988.0	12628.0	24040.0
6	0.0112947952	0.9972111583	0.9958540201	0.5801475048	0.4150880277	0.7921661735	80877.0	11650028.0	11588.0	21219.0
7	0.0102539547	0.9972997308	0.9972141981	0.6261434555	0.4592895508	0.8124314547	82946.0	11649001.0	12615.0	19150.0
8	0.0091279773	0.9975377917	0.9977597594	0.6463314295	0.4830977619	0.82774055	84509.0	11650238.0	11378.0	17587.0
9	0.0081673069	0.9977238774	0.9982048273	0.6649646163	0.5067940354	0.8458215594	86355.0	11650581.0	11035.0	15741.0
10	0.007792911	0.9977805614	0.9978224635	0.6904097199	0.5326638222	0.854313612	87222.0	11650381.0	11235.0	14874.0
11	0.0070189899	0.997936368	0.998511374	0.7231023908	0.5708810687	0.870455265	88870.0	11650566.0	11050.0	13226.0
12	0.0071170707	0.9978460073	0.9984389544	0.7106267214	0.5590440035	0.8558219671	87376.0	11650997.0	10619.0	14720.0
13	0.0067287567	0.9979459643	0.9976450205	0.7307897806	0.5835675001	0.864059329	88217.0	11651332.0	10284.0	13879.0
14	0.0061500669	0.9980867505	0.9985045791	0.7558168173	0.6120147109	0.8794761896	89791.0	11651414.0	10202.0	12305.0
15	0.0054902057	0.9982669353	0.9991051555	0.7783944607	0.640693903	0.892983079	91170.0	11652155.0	9461.0	10926.0
16	0.0051277694	0.9983747602	0.9993249178	0.7923486233	0.658582747	0.9017689228	92067.0	11652526.0	9090.0	10029.0
17	0.0045273462	0.9985698462	0.9995576739	0.8128631711	0.6879275441	0.9116027951	93071.0	11653817.0	7799.0	9025.0
18	0.004284475	0.998616755	0.9996296167	0.8201742172	0.6979238987	0.9184885025	93774.0	11653666.0	7950.0	8322.0
19	0.0041150576	0.9986596704	0.9996593595	0.8235151768	0.7039077282	0.9187235832	93798.0	11654147.0	7469.0	8298.0
20	0.0045081032	0.9985110164	0.9992614985	0.820933342	0.6985442042	0.9103001356	92938.0	11653258.0	8358.0	9158.0
21	0.0046384046	0.998437047	0.9989417195	0.8205311894	0.6993620396	0.9060002565	92499.0	11652827.0	8789.0	9597.0
22	0.0039962358	0.9986454248	0.9993683696	0.8364735246	0.7213757038	0.9179987311	93724.0	11654053.0	7563.0	8372.0
23	0.0037768879	0.9986973405	0.9994510412	0.8514928222	0.7426601648	0.9227002263	94204.0	11654184.0	7432.0	7892.0
24	0.0035568513	0.9987899065	0.9994698167	0.855954051	0.7497926354	0.930202961	94970.0	11654507.0	7109.0	7126.0
Average	0.0149	0.99598	0.9832	0.64654	0.51609	0.78134	79771.88	11636591.32	25024.68	22324.12

Figure 5. Dashboard Interface

The DashboardInterface in Figure 5 is the application where numerical data is visualized. This interface can be run independently or from within the main application. The "Add Json Directory" option in the application selects the folders where json training results are stored. More than one folder can be selected here, as well as a folder containing more than one json file. The selected folders will be added to the folder list at the top left. The json files in the selected folder from this list are displayed in the list below. The analysis results in the selected json file in the list containing json files are shown on the grid as shown in Figure 5.

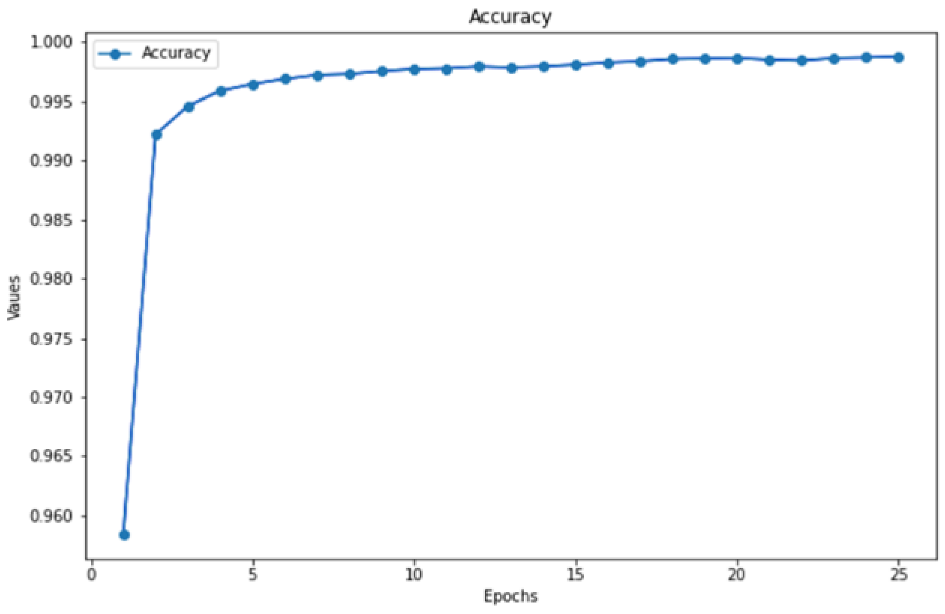


Figure 6. Sample Accuracy Graph

Figure 6 shows the graphical representation of each metric value on the grid in Figure 5 when the button is clicked. As an example, the graph created when the Accuracy field is selected is shown. The aim here is to visualize the values of the trained model on the grid and make them easier to interpret.

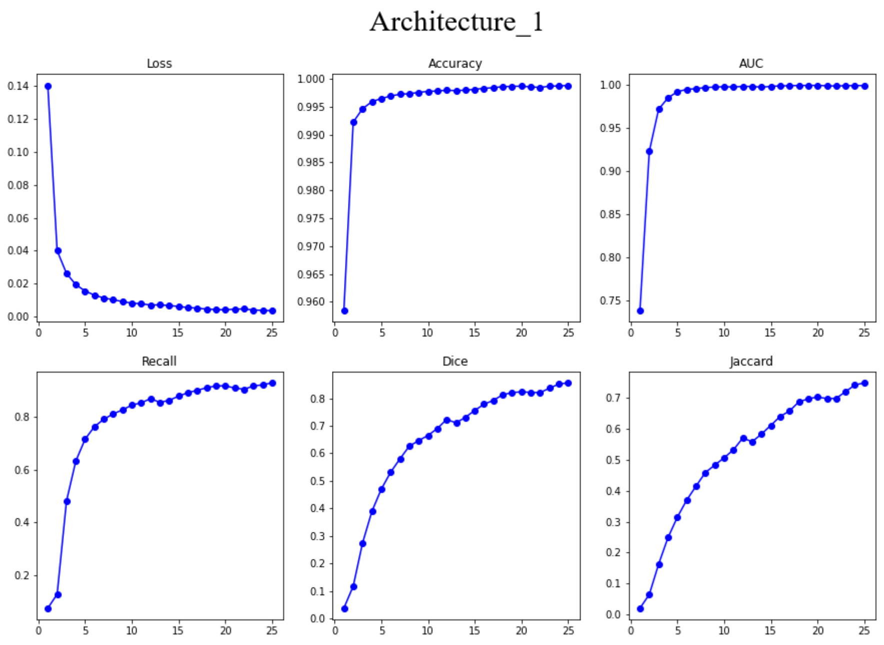


Figure 7. Loss, Accuracy, AUC, Recall, Dice and Jaccard graph

Some metric values are commonly used when training and comparing models. Reflecting the graph of these metric values as a whole facilitates analysis processes and helps to organize the areas that need attention when observing the deficiencies of the models. Figure 7 shows the graph of Loss, Accuracy, AUC, Recall, Dice and Jaccard metric values of Architecture 1 as a whole. Users can add or remove the metric values they want to this area. The selected metric values are the values commonly used in training and studies.

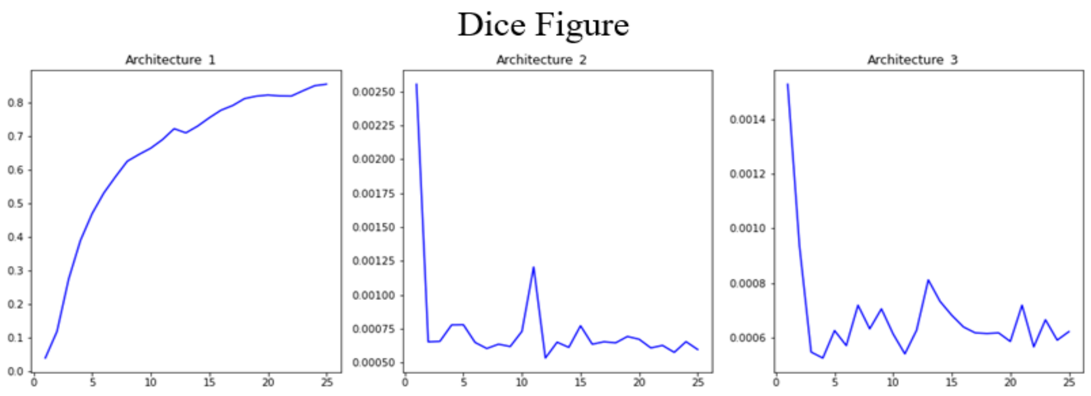


Figure 8. Comparison of Three Architectures according to Dice Value

In academic studies, the success of a model is demonstrated by proving its superiority over other models. This is provided by graphical or tabular representation. Figure 8 shows the numerical data of the trained models according to the selected metric value.

Discussion

Ronneberger et al. (2015) proposed the U-Net algorithm without a fully connected layer for segmenting microscopy images. The U-Net network and training strategy were developed to learn effectively from a small number of annotated images. U-Net takes its name from its U-shaped architectural structure as seen in Figure 9. The U-Net architecture consists of two parts. These are a contraction path to capture context and a symmetric expansion path that provides precise localization. The main feature of the architecture developed on the basic structure of the U-Net architecture is that it does not have fully connected layers. The architecture was developed to obtain successful results on very few sample data.

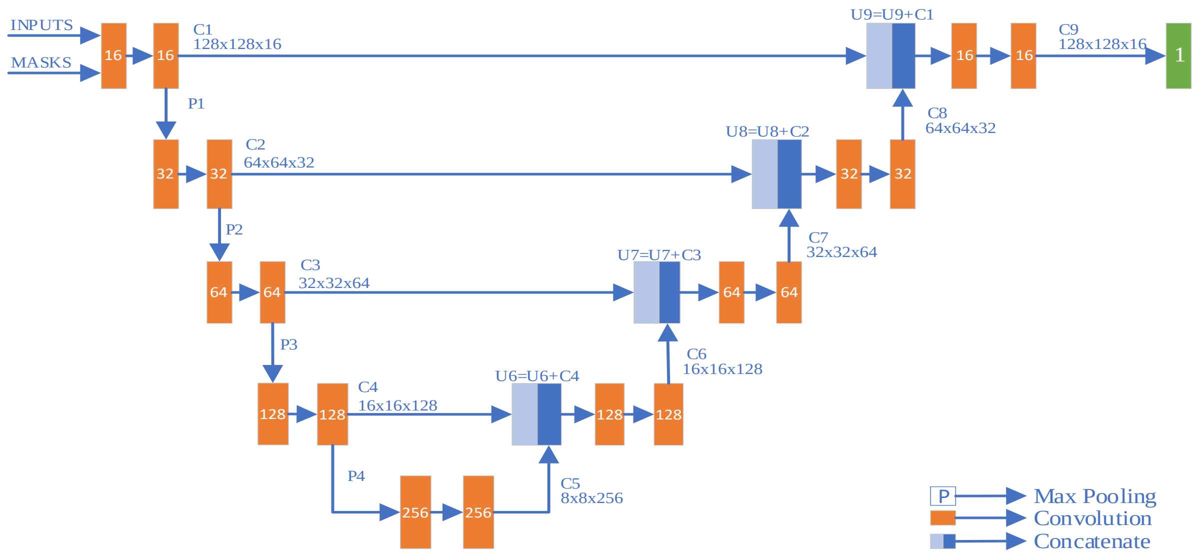


Figure 9. U-Net Architecture

Milletari et al. (2016) developed the V-Net architecture that processes using 3D MRI images. This architecture model, which was developed similar to the U-Net structure, has a contraction and expansion path. Unlike the U-Net architecture, the convolution operation increases as you go from the upper layers to the lower layers. Figure 10 shows the V-Net architecture we developed. 21 convolution operations were applied in this developed architecture.

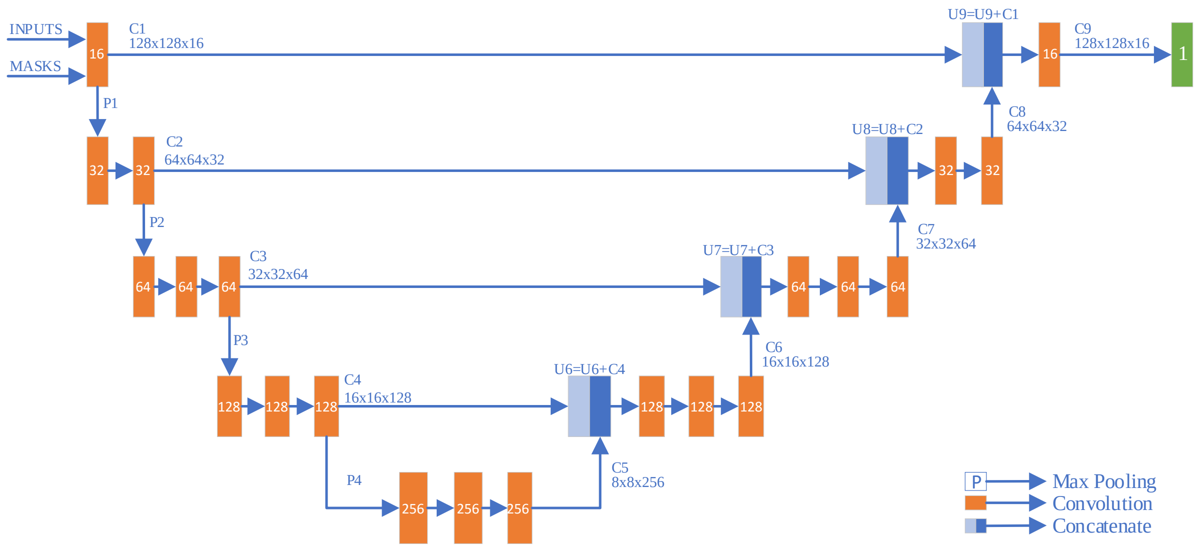


Figure 10. V-Net Architecture

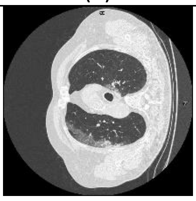
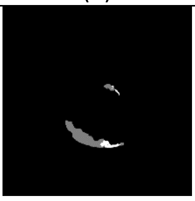
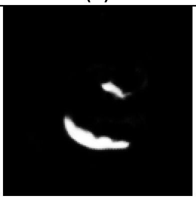
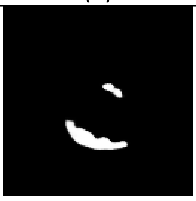
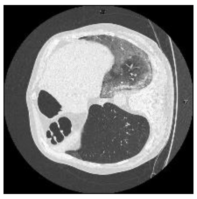
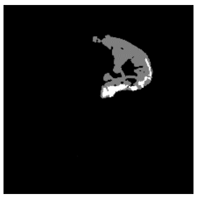
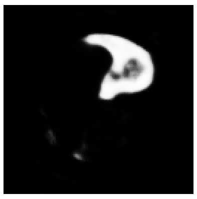
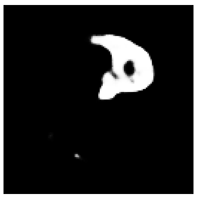
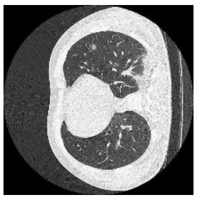
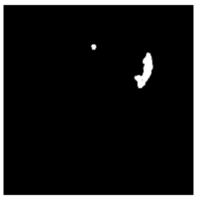
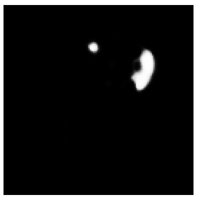
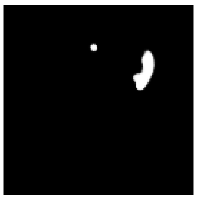
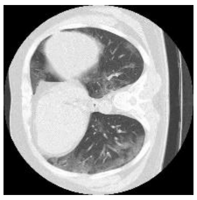
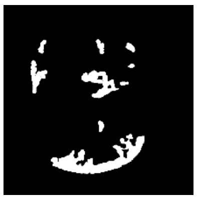
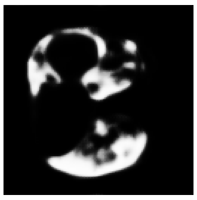
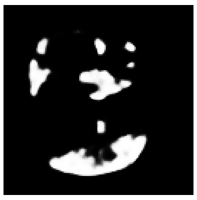
U-Net and V-Net architectures are widely used architectural models in image segmentation. The 3D V-Net architecture developed by Milletari was recoded as 2D and training was performed at different Validation Split and Epochs values with the help of the developed application environment. Table 1 shows the Dice performance results of the training results obtained from the application environment on 2D Covid19 lung images of U-Net and V-Net architectures. The obtained results are the maximum results of the trainings performed on each validation split and epochs values. Table 2 shows the prediction results of 2D test images of U-Net and V-Net architectures. Here, (a) shows the image used for the test, (b) the ground truth value to be estimated for the original image, (c) the prediction of the U-Net architecture and (d) the prediction of the V-Net architecture.

Table 1. Dice Performance Results

Validation Split - Epochs	U-Net	V-Net
0.1 – 25	0.5175357461	0.8076153994
0.1 – 30	0.7475325465	0.8520365953
0.1 – 35	0.8093342781	0.8139239550
0.1 – 40	0.8059905767	0.8637956381
0.1 – 45	0.7994861007	0.8163979053
0.1 – 50	0.8222454786	0.8813048601
0.2 – 25	0.7607743740	0.8103617430
0.2 – 30	0.8030135632	0.8342383504
0.2 – 35	0.8165560961	0.8413328528
0.2 – 40	0.8108936548	0.8718471527
0.2 – 45	0.8210129738	0.8621217608
0.2 – 50	0.8548157811	0.8330374956
0.3 – 25	0.7804758549	0.7551715374
0.3 – 30	0.7927131057	0.8025660515
0.3 – 35	0.8165585399	0.8284928799
0.3 – 40	0.8265380859	0.8737146258
0.3 – 45	0.8172306418	0.8541257977
0.3 – 50	0.8294414878	0.9030863643

As it is seen in the scientific studies that the V-Net architecture is successful, the results obtained from the application environment support the studies.

Table 2. Prediction Results of U-Net and V-Net Architectures

Original Images		Prediction Images	
Original Images (a)	Masks (b)	U-Net Predictions (c)	V-Net Predictions (d)
			
			
			
			

Results and Recommendations

DeepImageSegmentationApp is aimed to be a useful application for those working in the field of deep learning-based image segmentation. Compared to existing developer environments, our application environment will facilitate the work of those working in this field. The ability to intervene in the code in the segmentation of CT images removes the limitations imposed by deep learning libraries and enables more successful work. As an example, Dice and Jaccard performance metrics have been added to the application environment thanks to the code intervention.

In order to demonstrate the success of the application environment, it has been shown that the reorganized V-Net architecture exhibits more successful results than the U-Net architecture in Covid19 detection. The V-Net architecture implemented for 3D CT images was recoded for 2D CT images and exhibited maximum results in the dice performance metric at Validation Split = 0.3 and Epochs = 50.

Although DeepImageSegmentationApp was developed for segmentation studies on 2D CT images, it can also be easily converted for segmentation of larger sized CT images such as 3D.

The application can be developed or adapted for the study performed depending on the usage and studies. Although the application was developed for the operation of deep learning-based image segmentation models, it can be easily used in other deep learning applications thanks to its simple and understandable coding. In this way, it creates a ready interface for different deep learning studies. An example of this is time series deep learning studies.

Author Contribution

Lütfü Bayrak; methodology, software, formal analysis, investigation, visualization, writing - original draft. *Kenan Koçkaya*; conceptualization, methodology, validation, writing - review & editing. *Ahmet Çınar*; conceptualization, methodology, validation, writing - review & editing. The authors wrote, read and approved the paper together.

Ethics

There are no ethical issues related to the publication of this article.

Conflict of Interest

The authors declare that there is no conflict of interest.

ORCID

Lütfü Bayrak https://orcid.org/0000-0002-2154-7270

Kenan Koçkaya https://orcid.org/0000-0002-5253-1511

Ahmet Çınar https://orcid.org/0000-0001-5528-2226

References

Dang, T., Nguyen, T. T., McCall, J., Elyan, E., & Moreno-García, C. F. (2024). Two-layer ensemble of deep learning models for medical image segmentation. *Cognitive Computation*, *16*(3), 1141-1160. https://doi.org/10.1007/s12559-024-10257-5

Gupta, M., & Mishra, A. (2024). A systematic review of deep learning based image segmentation to detect polyp. *Artificial Intelligence Review*, *57*(1), 7. https://doi.org/10.1007/s10462-023-10621-1

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90. https://doi.org/10.1145/3065386

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444. https://doi.org/10.1038/nature14539

Li, J., Cai, Y., Li, Q., Kou, M., & Zhang, T. (2024). A review of remote sensing image segmentation by deep learning methods. *International Journal of Digital Earth*, *17*(1), 2328827. https://doi.org/10.1080/17538947.2024.2328827

Liu, X., Li, S., Zou, X., Chen, X., Xu, H., Yu, Y., ... & Zhang, Y. (2024). Development and clinical validation of a deep learning-based knee CT image segmentation method for robotic-assisted total knee arthroplasty. *The International Journal of Medical Robotics and Computer Assisted Surgery*, *20*(4), e2664. https://doi.org/10.1002/rcs.2664

Milletari, F., Navab, N., & Ahmadi, S. A. (2016, October). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)* (pp. 565-571). IEEE. https://doi.org/10.1109/3DV.2016.79

Peng, Y., & Yang, H. D. (2024). Aggregate boundary recognition of asphalt mixture CT images based on convolutional neural networks. *Road Materials and Pavement Design*, *25*(5), 1127-1143. https://doi.org/10.1080/14680629.2023.2233630

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18* (pp. 234-241). Springer international publishing. https://doi.org/10.1007/978-3-319-24574-4_28

Wang, Z., Zhang, T., & Huang, X. (2024). Explainable deep learning for image-driven fire calorimetry. *Applied Intelligence*, *54*(1), 1047-1062. https://doi.org/10.1007/s10489-023-05231-x

Xu, G., Yue, Q., Liu, X., & Chen, H. (2024). Investigation on the effect of data quality and quantity of concrete cracks on the performance of deep learning-based image segmentation. *Expert Systems with Applications*, *237*, 121686. https://doi.org/10.1016/j.eswa.2023.121686

Ye, R. Z., Noll, C., Richard, G., Lepage, M., Turcotte, É. E., & Carpentier, A. C. (2022). DeepImageTranslator: A free, user-friendly graphical interface for image translation using deep-learning and its applications in 3D CT image analysis. *SLAS technology*, *27*(1), 76-84. https://doi.org/10.1016/j.slast.2021.10.014

Zhang, H., Li, M., Zhong, J., & Qin, J. (2024). CNet: A novel seabed coral reef image segmentation approach based on deep learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 767-775). https://doi.org/10.1109/WACVW60836.2024.00090

Zi, Y., Wang, Q., Gao, Z., Cheng, X., & Mei, T. (2024). Research on the application of deep learning in medical image segmentation and 3d reconstruction. *Academic Journal of Science and Technology*, *10*(2), 8-12. https://pdfs.semanticscholar.org/d900/71bda4ad95986fbe508238a19e755cceff9d.pdf