**ISTANBUL SABAHATTIN ZAIM UNIVERSITY**

**FACULTY OF ENGINEERING AND NATURAL SCIENCES**

**SOFTWARE ENGINEERING DEPARTMENT**

**GRADUATION PROJECT**

**FINAL REPORT**

**2021**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**PROJECT TITLE**

Tumor Detection with Deep Learning

**PROJECT AUTHOR(S)**

Fatih KARAMAN

Hamza ÜZÜLMEZ

Hatice YAZICI

Mustafa ARU

Y. Sümeyye VARLIK

**ADVISOR**

Dr. Öğr. Üyesi Akhtar Jamil

# ABSTRACT

Brain tumor is the growth of abnormal cells that occur in replication during brain cell renewal. It is a serious cancer disease that is quite common today. Recent advances in deep learning have contributed significantly to medical imaging in the healthcare field. These contributions are vital in cancer diagnosis, treatment planning, and evaluation of treatment outcome. In this project, U-net deep learning architecture that an artificial neural network developed for biomedical image segmentation, detects the presence of the tumor by reading brain MRI images, enables the classification of tumors as necrosis, edema and enhancing tumors. As a result, the system that reports the MR image and the patient's condition has been developed.

**CONTENTS**

[ABSTRACT 3](#_Toc73293325)

[LIST OF TABLES 5](#_Toc73293326)

[LIST OF FIGURES 6](#_Toc73293327)

[INTRODUCTION 7](#_Toc73293328)

[LITERATURE REVIEW 9](#_Toc73293329)

[METHOD, IMPLEMENTATION and TESTS 12](#_Toc73293330)

[CONCLUSION 25](#_Toc73293331)

[FUTURE WORKS 26](#_Toc73293332)

[GANTT CHART 27](#_Toc73293333)

[REFERENCES 28](#_Toc73293334)

[ATTACHMENTS 29](#_Toc73293335)

# LIST OF TABLES

[Table 1: Model fit 17](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292892)

[Table 2: Train result compare 23](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292893)

[Table 3: Gantt Chart 27](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292894)

# LIST OF FIGURES

[Figure 1: Different sequences of brain images 12](#_Toc73292900)

[Figure 2: Per slice in nii format 13](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292901)

[Figure 3: Different level cross-sectional images from different angles 13](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292902)

[Figure 4: Original and segmented state of the brain 14](#_Toc73292903)

[Figure 5: Shown of non-enhancing, edema, and enhancing tumor parts 14](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292904)

[Figure 6: Tumor image that does not fit and fits in the frame after cropping 15](#_Toc73292905)

[Figure 7: Segmentation images of the tumor that does not fit in the frame 15](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292906)

[Figure 8: Combined view of the four parts of the tumor 15](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292907)

[Figure 9: Data augmentation operations 16](#_Toc73292908)

[Figure 10: U-Net Architecture 16](#_Toc73292909)

[Figure 11: Image of flair, t2 and concatenated forms 18](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292910)

[Figure 12: Segmented image and image of data predicted 19](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292911)

[Figure 13: Cropped view of the tumor area and the non-edematous area 19](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292912)

[Figure 14: Cropped first frame tumor of different types 20](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292913)

[Figure 15: Cropped tumors of different types 20](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292914)

[Figure 16: MR images of the patient and his full tumor 21](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292915)

[Figure 17: All tumor type predictions of the model 21](#_Toc73292916)

[Figure 18: Segmentation of the model and radiologist 22](#_Toc73292917)

[Figure 19: Loss values of training with cropped images 23](#_Toc73292918)

[Figure 20: With and without augmentation dice coef compare 24](#_Toc73292919)

[Figure 21: With and without augmentation loss compare 24](file:///C:\Users\sumva\Desktop\yamfinalreport.docx#_Toc73292920)

# INTRODUCTION

Cancer is a group of more than 100 diseases formed by the uncontrolled proliferation of cells in various parts of our body. Although there is a wide variety of cancer types, they all begin with the uncontrolled proliferation of abnormal cells. Diseases of the age, which are common in many parts of the world, such as cancer, have become one of the top priority research topics in health studies in the world. One out of every 5 people in the world gets cancer during their lifetime. 1 in 8 men and 1 in 11 women die of cancer. Worldwide, the total number of cancer patients alive (5-year prevalence) within 5 years of being diagnosed with cancer is estimated to be 50.6 million.

Brain tumor, which is very important in general and can be seen very often in people of all ages, is one of them. Studies show that five out of every 100,000 people develop a new brain tumor. However, estimates made in the world of science show that this rate will increase more and more. The reason is the aging of the world population. The incidence of brain tumors increases with advancing age. The growth and mass of abnormal cells that occur in replication during the renewal of brain cells is called a brain tumor.

Tomography device tries to obtain a three-dimensional view of the internal structure of an object by taking many two-dimensional X-ray images from different angles. By examining the image obtained from the device, it is tried to determine whether there is a tumor in the brain of the patients, if so, how large tumor it is.

As it is known from the research, brain tumors can be in many different parts of the brain and in various forms. Gliomas are a general name used to describe a group of tumors that occur in the glial cells, the supporting tissue of the brain. Gliomas make up about 30% of all brain tumors and are often malignant, i.e., malignant. The tumor area is divided into sections. These are classified as necrosis, edema, and an enhancing tumor.

As mentioned above, these procedures are easily performed with the help of artificial intelligence today, since the diagnosis of all different types of tumors in different places by doctors, the difficulty in detailing the tumor, and the difficulty of classifying the tumors.

In this project, U-net deep learning architecture that an artificial neural network developed for biomedical image segmentation, detects the presence of the tumor by reading brain MRI images, enables the classification of tumors as necrosis, edema and enhancing tumors. As a result, the system that reports the MR image and the patient's condition has been developed.

The project itself is an artificial intelligence-based project and the Python programming language, which has been widely used in recent years, is used. To begin with, OpenCV and TensorFlow technologies, two of the most important research topics, will also take place in this project. OpenCV and TensorFlow, Keras are software libraries used for topics such as neural networks, computer visions, etc.

These technologies were used to analyze the data and all tumors, and their classes are shown on the screen. Data without tumors were detected, and data were deleted or edited according to the need. As with every artificial intelligence project, data needs pre-processing. In the data preprocessing stages, the data were scaled for the project, respectively. Each MR image is set to be the same size. The number of data has been increased with data enhancement methods and given to the model to be trained. For the determination of the tumor, it is called a full tumor by combining all tumor types, this process is the normalization of the data.

The model was trained with the U-Net architecture with the data passed through data preprocessing. The trained model has been improved with various improvement methods and the results are expressed visually in the following parts of the document. As a result, the printouts will be displayed both as visualized on the picture and in writing on the report. If the project is supported or developed, it can be used actively on a platform such as E-pulse, which is widely used in the field of health throughout Turkey, and patients will be able to obtain their reports with MR images automatically uploaded to the system.

# LITERATURE REVIEW

The first studied [[1]](#Ref1) brain tumor detection was made by creating a detection model using a neural network in Tensorflow and Keras.

A brain MRI image data based on Kaggle was used. The dataset contains tumor and useless data. The data set to be used in the project was augmented beyond insufficient to train the neural network. A pre-processing has been performed on the increased data. As the first step of pretreatment, for each image, the image is cropped with a variety (which is the most part of the image) containing only the brain. Then the image has been resized. Because the images in the data set are of different sizes. Thus, all images must have the same shape in order to feed it as input to the neural network. Finally, normalization is done (pixel submission scaling to 0-1 series).

70% of the data is reserved for training, 15% for verification and 15% for testing. Transfer learning has been implemented using ResNet50 and vgg-16 as neural networks. The image use is (240, 240, 3) and feeds into the neural network. And it goes through the following layers:

1. Zero padding layer with pool size (2, 2).
2. Convolutional layer with 32 filters with filter size (7, 7) and equal to step 1.
3. Batch normalization layer to normalize to speed up the calculator.
4. ReLU activation layer.
5. Max Pooling layer with f = 4 and s = 4.
6. Max Pooling layer with the same f = 4 and s = 4 as before.
7. Flattened layer to flatten a 3-dimensional matrix into a one-dimensional vector.
8. A dense (output unit) layer fully connected with a sigmoid activated neuron (because this binary class is a classification task)

The Model 24 has been epoch trained and the best accuracy is taken at the 23rd epoch. This project was built taking memory into account. As a result, a high accuracy rate has been achieved.

Another [[2]](#Ref2) study that asks and understands the tumor from MR images and is one of the deep system deep learning architectures, has been linked to the convolutional network. This structure is a building skin using CNN architecture, basically CNN for the division of the picture and its region. The biggest disadvantage of this method is time, as CNN is applied for its region.

A pre-processing was performed in the MRI examination. As MRI can be affected by noise in the imaging potential, there may be errors in intensity. As this will adversely affect the system, histogram stretching process has been applied to minimize noise. The tumors in the data set were then manually labeled and thus a great success and place. If manual labeling is performed, if it is large, that area is considered as a healthy area, marked as tumor. These images are resized (32x32) and given to the introduction of different based convolutional network (BESA) architectures. Of these BESA architectures, BESA1 and BESA3 consist of seven layers and different filter sizes, while BESA2 consists of eight layers and BESA4 consists of nine layers.

The data set has been validated at Benchmark, Rembrandt, and Harvard. With BESA Structural, a detector that gives the texture and shape of the tumors has been obtained. With this detector, different data can be applied, tested and accuracy compared. Accordingly, the best result was obtained with the Benchmark data set and the fourth BESA website. The architectures (BESA1, BESA2, BESA3, BESA4) based on all these studies have been tested on different data and the data set has the highest accuracy value obtained.

In this study [[3]](#Ref3), the BraTS data set was used to create and train models. The BraTS dataset contains four sequences for each brain, namely T1, T1c (post-contrast T1), T2, and Flair, cranked, resampled, and recorded together. In the data pre-processing phase, the N4ITK algorithm was first used in each MRI sequence to correct the homogeneity of the images. Second, 1% of the upper and lower densities were removed and then each sequence was normalized to zero mean and unit variance.

For each image, the network was trained by performing 5-fold cross validation on 285 training data in BraTS2018. Verification was performed with 228 images for training and 57 images on the verification data for each floor. Finally, the enhancement process for each image was obtained by using an ensemble learning method by averaging the SoftMax output of five networks. BraTS 2018 and 2017 validation data are used to measure method result.

In the study, 2D UNet architecture was used instead of 3D UNet. This is because although 3D UNet has good performance, it has more parameters and computational complexity than 2D UNet. Therefore, 2D UNet architecture has been used to increase the performance of the network in terms of memory. The proposed network architecture is based on the UNet structure with some changes. Residual units are used instead of plain units in a normal UNet to accelerate training and convergence. For each residual unit there are two convolutional layers, each followed by the Batch Normalization layer and the PReLU activation function, instead of the ReLU activation function used in the original UNet.

Because 2D UNet is used, 3D information of the input images cannot be used. For this reason, MRI images must be divided into 2D slices. For this, Multi-View technique has been used to improve the network performance by using the 3D information of the input images. There is also the possibility of confusion for the model due to the combination of high-level and low-level features. Attention mechanism has been used to prevent this confusion and to extract distinctive features.

Finally, an improved version of the 2D UNet architecture was used in this study to compartmentalize brain tumors. Because the 2D UNet architecture contains a small number of parameters, the model can utilize the 3D information of the input images using the Multi-View technique. Thanks to the attention mechanism used in the study, the complexity in the model has been avoided. Using these techniques, the average dice scores of 0.813, 0.895 and 0.823 were obtained for enhancing tumor (ET), whote-tumor (WT), and tumor core (TC) using the BRATS 2018 validation dataset.

The difference of the project to be realized from the first two studies mentioned above is the data set to be used. The Brats dataset is in .nii format, consists of 4 different sequences, and each sequence contains 155 slice. Slices between 50 and 130 were used for this study, the other slices were not used as they did not contain tumor images. Data increase process has been applied to the slices in this range. Tumor; U-Net model is used to segment as Full Tumor, Tumor Core and Enhancing Tumor.

# METHOD, IMPLEMENTATION and TESTS

Within the scope of this study, a tumor detection system was developed from CNN-based brain MR images. The materials and methods used in each stage are:

1. Dataset

BraTS data set [[4]](#Ref4) was used for this study. The training set contains images from 370 patients, including 294 HGG and 76 LGG. The validation set includes MRI scans from 125 patients with unknown degrees of brain tumors. It is a predefined set created by BraTS challenge organizers. The test set contains images from 191 patients with brain tumors. MRI is one of the most commonly used imaging techniques in neurology and neurosurgery. Sequence, on the other hand, is the data showing the radio frequency pulse, gradient and signal collection processes applied to generate the MRI signal. There are four sequences for each patient's image: T1, T1ce, T2, and FLAIR.

The properties of each of these sequences are different from each other. Tag information for segmentation is available in the dataset. Label 0 segmentation of the radiologist, label 1 full tumor, label 2 necrosis, label 3 non-edema tumor, label 4 enhancing tumor.

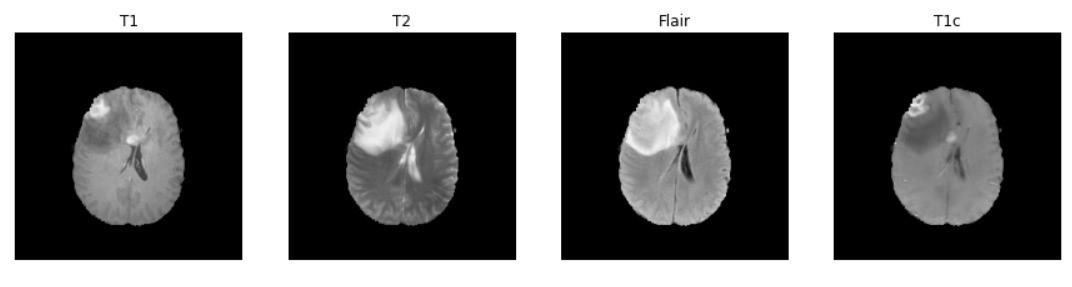


Figure 1: Different sequences of brain images

The found dataset consists of medical format (.dcm, .mha, .nii). Images in medical formats are displayed in 3D. Each image in nii format consists of a total of 150 slides. As training data, 3D images will be given directly to the model in nii format.

metin, elektronik eşyalar, klavye, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 2: Per slice in nii format

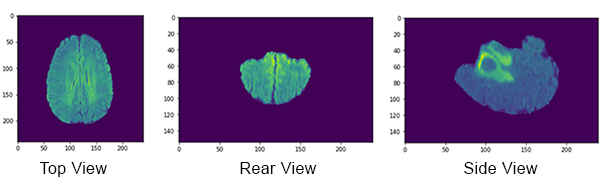
 The 'glob' library, a Python library, with the '.nii' extension was used to list and filter our data. After listing our data read using the 'skimage.io' library, cross-sectional images were taken from a specific area of the brain. The data were displayed with the library "matplotlib.pyplot" with the extension "simpleitk" used in medical imaging.

Figure 3: Different level cross-sectional images from different angles

After viewing these different level cross sectional images, the segmented state of the tumor included in the data was visualized and compared with its original form.

kare içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 4: Original and segmented state of the brain

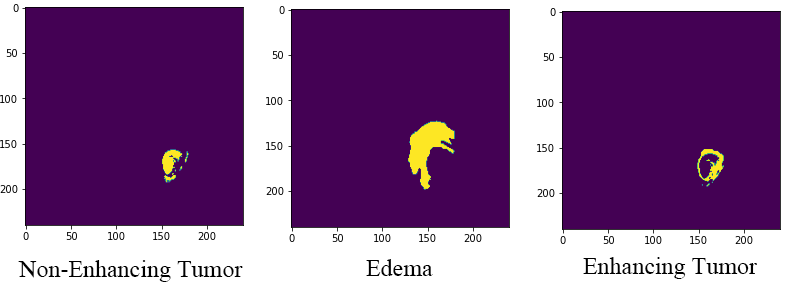
The segmented data divided into types within itself was analyzed. These parts are divided into Non-enhancing(necrosis), Edema and Enhancing tumor parts. By applying labeling in the code, sections of the tumor were displayed separately on the sample data.

Figure 5: Shown of non-enhancing, edema, and enhancing tumor parts

1. Data Preprocess

While performing tumor segmentation, T1ce images are used to make the parts of the tumor more prominent. When processing these images, the size of the tumor remains very small compared to the overall x-ray size, so the results are not very healthy when making predictions with the model. Therefore, cropping is applied to the part where the tumor is on these images. As a result of this process, the size of the x-ray image was reduced to 64x64 pixels.

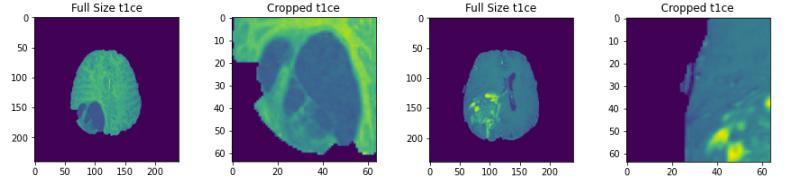


Figure 6: Tumor image that does not fit and fits in the frame after cropping

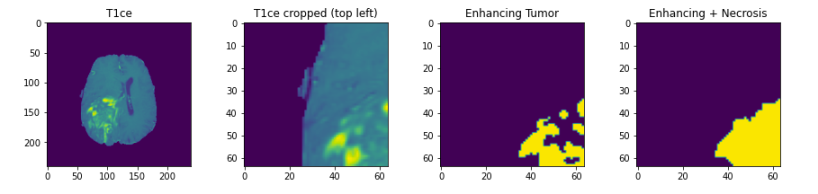
 Thus, the ratio of the tumor region to the rest of the brain is increased. While 64x64 ratio is sufficient for some x-rays, different ratios of cropping are required for large tumors. These ratios can be 64x128 (large in width), 128x64 (large in length) and 128x128 (large in both width and length). Tumor images that provide these ratios are cropped and made suitable for export to the new model.

Figure 7: Segmentation images of the tumor that does not fit in the frame

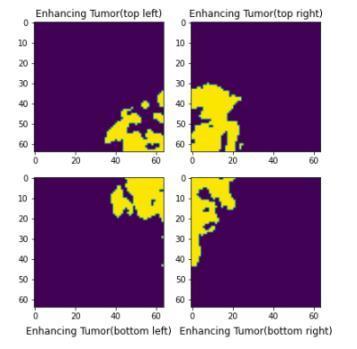


Figure 8: Combined view of the four parts of the tumor

This dataset contains the image sequence flair, t1, t2, t1ce. Due to the scarcity of MR data, the number of data was increased with the Data Augmentation method. Data augmentation is a method that helps to improve the generalization capabilities of deep neural networks and provides indirect arrangement of networks. Especially in medical studies, it plays an important role in concluding that the amount of data is limited and obtaining new samples is costly and time-consuming. In this study, two separate operations were performed on flair and segment images to segment brain tumors. In segment visuals, all label values ​​are set to the same value. Since there is no tag value in flair images, zero normalization is applied to increase it. The number of samples was increased by applying rotation, horizontal flip and vertical flip to the images. The reproduced data was transformed into an array using numpy and the necessary data for training was created. The size of each image is 240x240x155, 70 to 130 slices are used for preprocessing. These slices were chosen because the training data is unlikely to have any tumors in the remaining slices of the image. Slices are then normalized to zero mean using the mean and standard deviation.

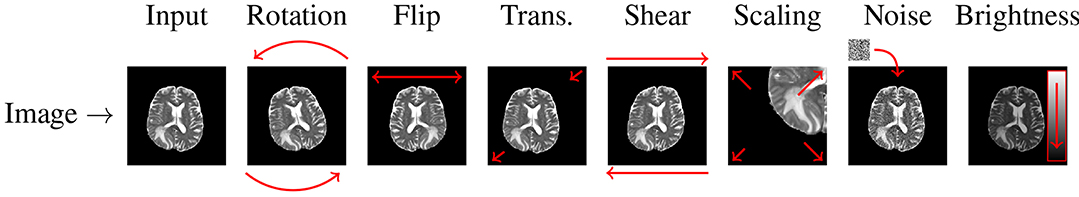


Figure 9: Data augmentation operations

1. Network Architecture

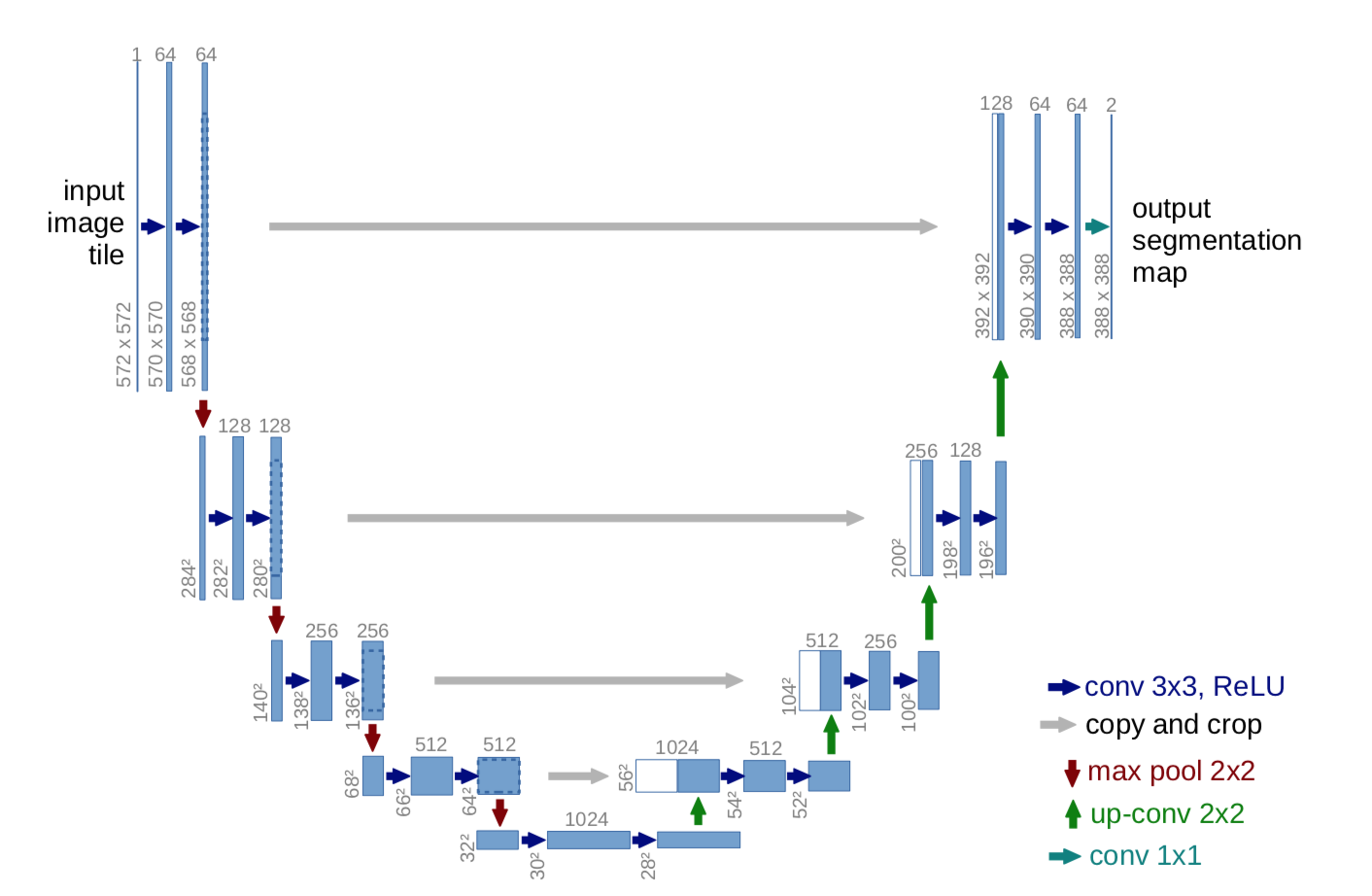


Figure 10: U-Net Architecture

In the study, experiments have been carried out on the model using U-Net network architecture. The aim at this stage is to obtain the optimum result.

U-Net: U-Net is an evolutionary neural network developed for biomedical image segmentation. [[5]](#Ref5) The network is based entirely on convolutional neural networks, and its architecture has been extended to work with less educated images and provide more precise segmentation. On this topic, [[6]](#Ref6) Ayyuce Kızrak's detailed article titled “Deep Learning for Image Segmentation: U-Net” was used.

The proposed 9-layer U-net Structure was used for full tumor segmentation. There are four differences between the U-Net architecture used and the original U-Net architecture:

(a) Batch normalization layer added after each convolution layer.

(b) Used same padding on convolution layers to keep feature map size unchanged.

(c) The filter number of the last convolution layer is the same for dual segmentation.

(d) Input channels are given in duplicate because T2 and Flair images are used for full tumor segmentation.

1. Training

tablo içeren bir resim

Açıklama otomatik olarak oluşturulduDuring the training phase, the parameters of the model are defined. These are the data consisting of the combination of flair and t2 image, segmentation data, validation split, batch size, number of epochs, and parameters required for saving after each epoch the model trains. After these definitions, the training process begins.

Table 1: Model fit

Dice Coefficient & Dice Coefficient Loss Function was used in the training process.

The Dice coefficient formula [[7]](#Ref7) is designed to be applied to discrete data. Given two sets, X and Y are defined as:



Here |X| and |Y| are the basic attributes of the two sets (i.e., the number of elements in each set). The dice coefficient index is equal to twice the number of items common to both sets and divided by the sum of the number of items in each set.

The absolute sums of X and Y are taken on the denominator side for our metric. Here, X = y\_true, Y = y\_pred.

1 - dice\_coef is used to formulate a loss function that can be minimized. Membrane metric is a commonly used performance criterion for evaluating success in biomedical images.

Adam is used as optimizer parameter. Stochastic gradient descent maintains a single [learning rate](https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/) for all weight updates and the learning rate does not change during training. A learning rate is maintained for each network weight and separately adapted as learning unfolds.

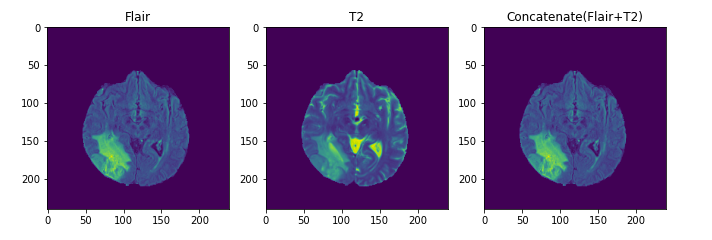
Flair, T2 and Flair + T2 images, which will be used for the model, were read. As mentioned under the Dataset title, the reason for combining these images should briefly complement each other's information.

Figure 11: Image of flair, t2 and concatenated forms

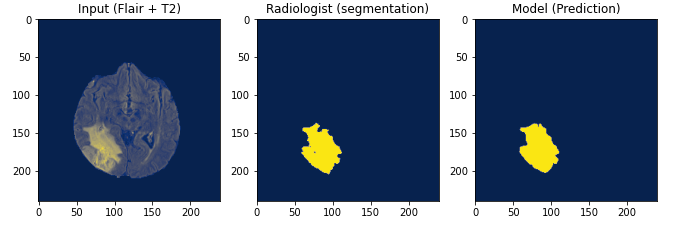
Afterwards, the model was tested to observe how the model worked for only full tumor. Only one image was used for this test. Here is the original image, segment view and test of the model:

Figure 12: Segmented image and image of data predicted

The model, which can easily detect whole tumors, cannot successfully detect enhancing and necrotic tumor varieties. The reason for this is that they are very small compared to the brain image.

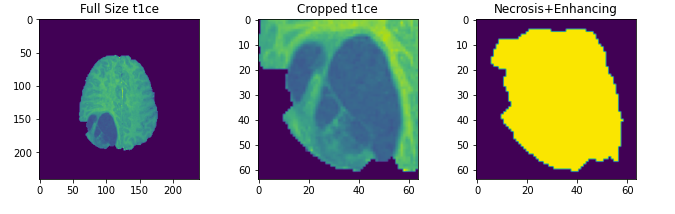
By reading the segmentation data, it is known in which pixels the tumor is located. In the X and Y coordinates, by adding the highest and lowest number of data and dividing them into two, there is an approximate midpoint in that coordinate. Trimming is carried out by taking a certain distance from the middle point to both coordinates. A new picture is created with the cropped picture.

Figure 13: Cropped view of the tumor area and the non-edematous area

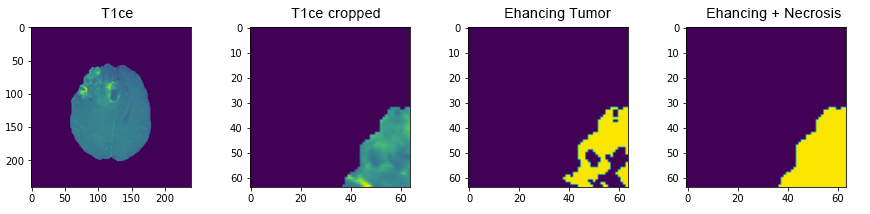
Thus, T1C MR images will be given to the model and the model will provide a better learning. Another situation encountered during the modeling phase is that if this tumor data is larger than the 64x64 pixel frame specified earlier, the frame overflows.

Figure 14: Cropped first frame tumor of different types

The example above is an example of this overflow. It is not possible to increase the dimensions of this frame. Because the dimensions must match with other frames. For this reason, it is necessary to divide the approximated tumors of known size into a series and give them to the model in this way. Thanks to this procedure, the model will also be trained for tumors larger than the frame found in MR images in T1C.

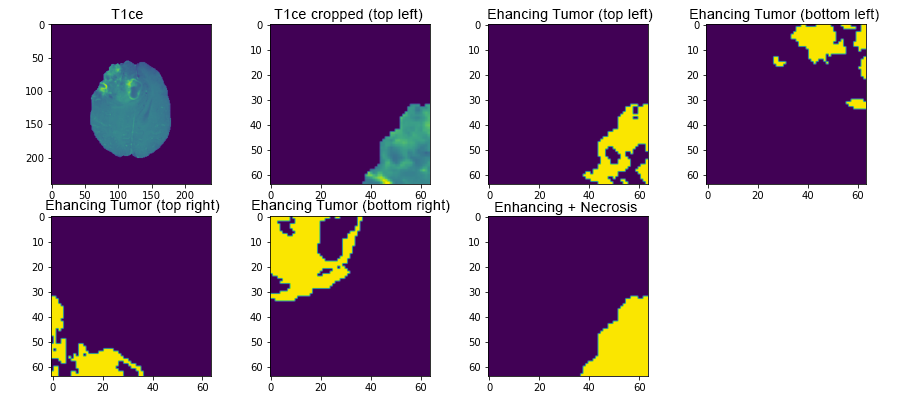


Figure 15: Cropped tumors of different types

Training continues by giving all these pictures to the model as a series. For the training of the second and third models, the data prepared by applying cropping on the T1ce images will be used. The data divided into two as enhancing and non-edema according to the label value is thrown into two arrays defined by cropping and applying threshold. A 7-layered U-Net model is applied for each array, and model training is carried out with the necessary parameters. At the end of the training, 2 more models are created, where the model can make predictions for enhancing and non-edema tumor sections. The weight values resulting from the training are saved to be used in the last segmentation process.

kare içeren bir resim

Açıklama otomatik olarak oluşturulduThe model has been able to successfully detect tumors present on MR images such as Flair, T2, and generate predictions. The model, which had difficulties in detecting small tumors in T1C MR images and produced erroneous results, was able to detect small tumors successfully thanks to cropping processes.

Figure 16: MR images of the patient and his full tumor

The full tumor image of the first patient in our data is shown above. So far, the model has been trained to separately detect the different variants present in this tumor. The next step is to superimpose these individually detected tumor types to detect the full tumor and view the actual prediction of the model.

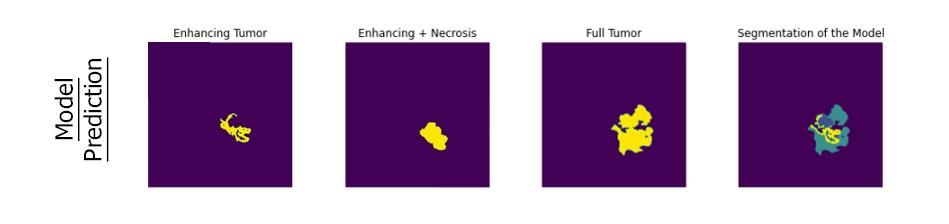


Figure 17: All tumor type predictions of the model

Tumors were predicted separately from each other. These tumors were combined with the superposition method and a result close to the original tumor image of the patient was obtained.

1. Implementation and Results

Basically, images in different sequences in the dataset have different properties. The flair sequence has a significant effect on the segmentation of different tumor subregions as well as the entire tumor region. The T2 sequence is mainly used to identify the edema zone and amplifies the signal of this area, which can provide useful information for training our model. It can be seen that the overall segmentation accuracy of the model is reduced in the absence of the T2 sequence. Here, flair and t2-sequence images are used in education to obtain the full tumor. At the end of the training, the model successfully detected the full tumor. Afterwards, the t1ce sequenced images in the dataset were used to detect the enlarged and non-edematous tumor as it strengthens the features of the tumor border and makes the border clear and easily distinguishable. In the model created here, the detection of enlarged and non-edema tumor was also successfully achieved. Finally, segmentation was performed by superimposing the obtained images, and by testing the model, images similar to the segmentation data given in the training data could be obtained.

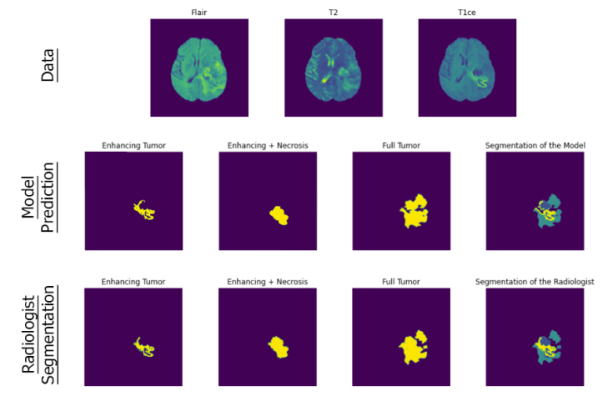


Figure 18: Segmentation of the model and radiologist

Two different models were trained using cropped non-edema and enhancing tumor images.

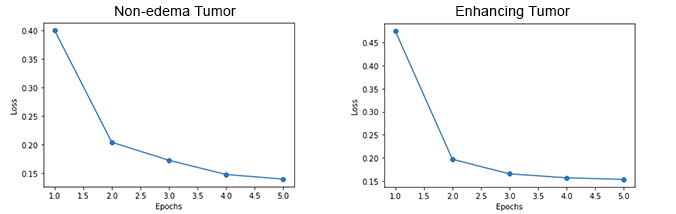


Figure 19: Loss values of training with cropped images

Data augmentation is a technique that helps improve the generalization abilities of deep neural networks. Using Keras' ImageDataGenerator class, the number of instances is increased by applying rotation, horizontal flip, vertical flip, and zoom. The model trained with data augmentation showed improved performance with a dice coefficient of 0.912 and loss of 0.087 in the results. It was observed that the model trained without augmentation was higher than the accuracy value and the loss values decreased. As a result, the positive effect of data augmentation on the accuracy and loss values of the model was determined.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Table 2: Train result compare

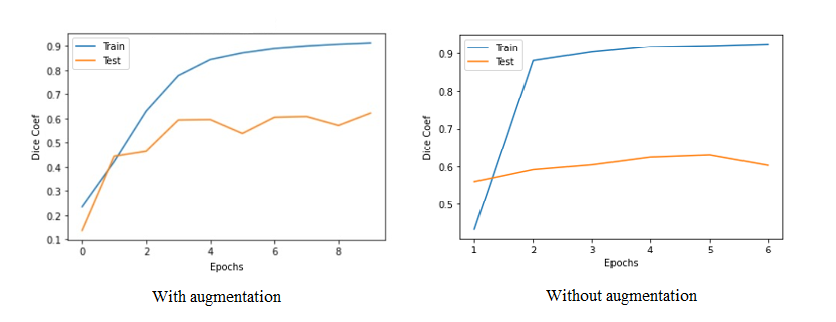


Figure 20: With and without augmentation dice coef compare

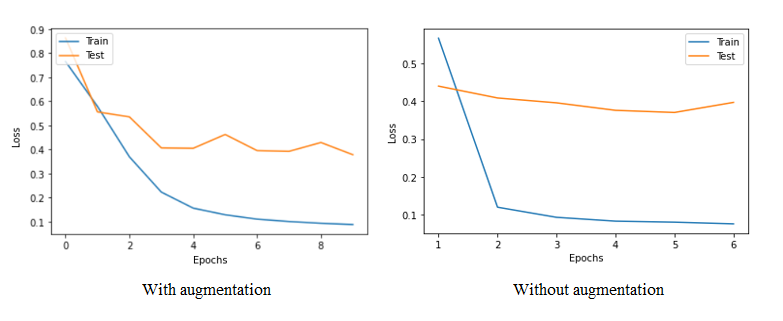


Figure 21: With and without augmentation loss compare

Finally, a report that visualizes the tumor and expresses the patient's tumor types and their sizes is prepared and presented to the doctor or the patient in pdf format.

# CONCLUSION

A brain tumor segmentation method has been developed to improve patients' quality of life. In the study, a method based on deep learning was proposed using deep convolution networks based on the U-Net model. The method has been evaluated on the BRATS 2020 dataset, which includes both HGG and LGG patients. Flair and t2 sequence images have been used to obtain the full tumor. At the end of the training, the model successfully detected the full tumor. T1ce sequence images have been used to detect enhancing and necrosis tumor. In the created model, the detection of enlarged and non-edematous tumor was successfully achieved. Segmentation was performed by superimposing the obtained images, and by testing the model, images like the segmentation data given in the training data were obtained. This study has been provided efficient and robust segmentation compared to manually defined baseline truth. The maximum dice coef similarity coefficient for the data set used was 0.92.

# FUTURE WORKS

1. It is considered to develop a model to survival prediction based on patients' MR images. A feature map is created by matching the patient's age with the corresponding segmented images in the data in the Excel file. Training is carried out with these data. The model will classify inpatients into one of 3 categories based on days of survival.
2. It is planned to compare the performances of these models by training with different architectures for brain tumor segmentation.
3. It is considered to design a mobile application that brain tumor detection by uploading MR images and will be integrated with the E-Nabız application.

# GANTT CHART

WP.1. Problem Description: It covers the search and definition of the problem that will bring a solution in the project and the search for resources about the solution.

WP.2. Finding the Data Set: Finding the appropriate data set for the project.

WP.3. Data Analysis: It is the examination of data to support decision making.

WP.4. Creating a Model: Creating the model to be trained.

WP.5. Training the Model: The prepared data are divided into two groups as training set and test set. The training set is a big piece of your data. This data is used to make the necessary adjustments to your machine learning models to achieve the highest accuracy.

WP.6. Testing the Model: It is the testing of the trained model with undefined data.

WP.7. Calculation of the accuracy ratio: It is the number of the similarity of the split training data and testing data of the dataset.

WP.8. Improvement: General improvement of the system than the older values and accuracy.

WP.9. Evaluation: It is the evaluation of the application results. Improve to show the result clearly like number rounding.

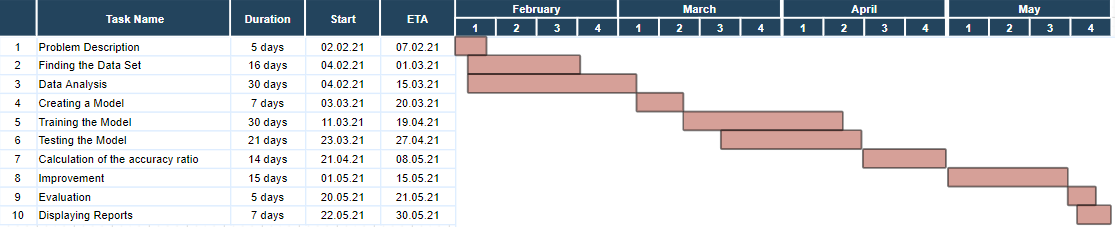
WP.10. Displaying Reports: Displaying the reports received by the model such as PDF Format.

Table SEQ Table \\* ARABIC 1: Gantt Chart

Table SEQ Table \\* ARABIC 1: Gantt Chart

Table 3: Gantt Chart

# REFERENCES

[1]Brain Tumor Detection: <https://github.com/MohamedAliHabib/Brain-Tumor-Detection>

[2] Tumor detection in MR images of regional convolutional neural networks: <https://dergipark.org.tr/tr/download/article-file/725602>

[3] Attention-Guided Version of 2D UNet for Automatic Brain Tumor Segmentation <https://ieeexplore.ieee.org/abstract/document/8964956>

[4] Multimodal Brain Tumor Segmentation Challenge 2020 Data : <https://www.med.upenn.edu/cbica/brats2020/data.html>

[5] U-Net: Convolutional Networks for Biomedical Image Segmentation, <https://arxiv.org/abs/1505.04597>

[6] Görüntü Bölütleme (Segmentasyon) için Derin Öğrenme: U-Net : <https://ayyucekizrak.medium.com/g%C3%B6r%C3%BCnt%C3%BC-b%C3%B6l%C3%BCtleme-segmentasyon-i%C3%A7in-derin-%C3%B6%C4%9Frenme-u-net-3340be23096b>.

[7] Sørensen–Dice coefficient <https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient>

# ATTACHMENTS

**import** **tensorflow** **as** **tf**

**from** **tensorflow.keras.models** **import** Model

**import** **numpy** **as** **np**

**import** **glob**

**import** **random** **as** **r**

**import** **skimage.io** **as** **io**

**import** **matplotlib.pyplot** **as** **plt**

**from** **tensorflow.keras.layers** **import** Dense, Dropout, Activation, Flatten

**from** **tensorflow.keras.layers** **import** concatenate, Conv2D, MaxPooling2D, Conv2DTranspose

**from** **tensorflow.keras.layers** **import** Input, UpSampling2D,BatchNormalization

**from** **tensorflow.keras.callbacks** **import** ModelCheckpoint

**from** **tensorflow.keras.optimizers** **import** Adam

**from** **tensorflow.keras.preprocessing.image** **import** ImageDataGenerator

**from** **tensorflow.keras** **import** backend **as** K

**from** **tensorflow.keras.callbacks** **import** ModelCheckpoint, EarlyStopping

**def** seg\_array(path,end,label):

files = glob.glob(path+end,recursive=**True**)

img\_liste = []

r.seed(9)

r.shuffle(files)

**for** file **in** files:

img = io.imread(file,plugin='simpleitk')

**if** label == 1:

img[img != 0 ] = 1 *# tam tümör*

**if** label == 2:

img[img != 1 ] = 0 *# nekroz*

**if** label == 3:

img[img == 2 ] = 0 *# ödemsiz tümör*

img[img != 0 ] = 1

**if** label == 4:

img[img != 4 ] = 0 *# genişleyen tümör*

img[img == 4 ] = 1

img.astype('float32')

**for** slice **in** range(60,130):

img\_s = img[slice,:,:]

img\_s = np.expand\_dims(img\_s,axis=0)

img\_liste.append(img\_s)

**return** np.array(img\_liste,np.float32) *#!!!!!!!!*

**def** train\_array(path,end):

files = glob.glob(path+end,recursive=**True**)

img\_liste = []

r.seed(9)

r.shuffle(files)

**for** file **in** files:

img = io.imread(file,plugin='simpleitk')

img = (img-img.mean())/ img.std()

img.astype('float32')

**for** slice **in** range(60,130):

img\_s = img[slice,:,:]

img\_s = np.expand\_dims(img\_s,axis=0)

img\_liste.append(img\_s)

**return** np.array(img\_liste,np.float32)

flair = train\_array('C:**\\**Users**\\**Fatih**\\**Desktop**\\**dataset**\\**', '\*\***\\**\*flair.nii.gz')

t2 = train\_array('C:**\\**Users**\\**Fatih**\\**Desktop**\\**dataset**\\**', '\*\***\\**\*t2.nii.gz')

seg = seg\_array('C:**\\**Users**\\**Fatih**\\**Desktop**\\**dataset**\\**', '\*\***\\**\*seg.nii.gz', 1)

seg\_final = seg\_array('C:**\\**Users**\\**Fatih**\\**Desktop**\\**dataset**\\**', '\*\***\\**\*seg.nii.gz', 0)

x\_train = np.concatenate((flair, t2), axis=1) *#axis: kacinci index birlestirilecek. Axis 1 olacak cunku zaten tek kanal 1,240,240*

t1ce = train\_array('C:**\\**Users**\\**Fatih**\\**Desktop**\\**dataset**\\**', '\*\***\\**\*t1ce.nii.gz')

seg\_nonedema = seg\_array('C:**\\**Users**\\**Fatih**\\**Desktop**\\**dataset**\\**', '\*\***\\**\*seg.nii.gz', 3)

seg\_enhancing = seg\_array('C:**\\**Users**\\**Fatih**\\**Desktop**\\**dataset**\\**', '\*\***\\**\*seg.nii.gz', 4)

seg\_necrosis = seg\_array('C:**\\**Users**\\**Fatih**\\**Desktop**\\**dataset**\\**', '\*\***\\**\*seg.nii.gz', 2)

K.set\_image\_data\_format('channels\_first') *# (240,240,1) => (1,240,240) yani katman sayisi ilk*

**def** dice\_coef(y\_true, y\_pred): *#[piksel farki] https://i.stack.imgur.com/OsH4y.png*

smooth = 0.005

y\_true\_f = K.flatten(y\_true)

y\_pred\_f = K.flatten(y\_pred)

intersection = K.sum(y\_true\_f \* y\_pred\_f)

**return** (2. \* intersection + smooth) / (K.sum(y\_true\_f) + K.sum(y\_pred\_f) + smooth)

**def** dice\_coef\_loss(y\_true, y\_pred):

**return** 1-dice\_coef(y\_true, y\_pred)

**def** unet\_model():

inputs = Input((2, 240 , 240)) *#sum of the shapes of the flair and t2*

conv1 = Conv2D(64, (3, 3), activation='relu', padding='same') (inputs)

batch1 = BatchNormalization(axis=1)(conv1)

conv1 = Conv2D(64, (3, 3), activation='relu', padding='same') (batch1)

batch1 = BatchNormalization(axis=1)(conv1)

pool1 = MaxPooling2D((2, 2)) (batch1)

conv2 = Conv2D(128, (3, 3), activation='relu', padding='same') (pool1)

batch2 = BatchNormalization(axis=1)(conv2)

conv2 = Conv2D(128, (3, 3), activation='relu', padding='same') (batch2)

batch2 = BatchNormalization(axis=1)(conv2)

pool2 = MaxPooling2D((2, 2)) (batch2)

conv3 = Conv2D(256, (3, 3), activation='relu', padding='same') (pool2)

batch3 = BatchNormalization(axis=1)(conv3)

conv3 = Conv2D(256, (3, 3), activation='relu', padding='same') (batch3)

batch3 = BatchNormalization(axis=1)(conv3)

pool3 = MaxPooling2D((2, 2)) (batch3)

conv4 = Conv2D(512, (3, 3), activation='relu', padding='same') (pool3)

batch4 = BatchNormalization(axis=1)(conv4)

conv4 = Conv2D(512, (3, 3), activation='relu', padding='same') (batch4)

batch4 = BatchNormalization(axis=1)(conv4)

pool4 = MaxPooling2D(pool\_size=(2, 2)) (batch4)

conv5 = Conv2D(1024, (3, 3), activation='relu', padding='same') (pool4)

batch5 = BatchNormalization(axis=1)(conv5)

conv5 = Conv2D(1024, (3, 3), activation='relu', padding='same') (batch5)

batch5 = BatchNormalization(axis=1)(conv5)

up6 = Conv2DTranspose(512, (2, 2), strides=(2, 2), padding='same') (batch5)

up6 = concatenate([up6, conv4], axis=1)

conv6 = Conv2D(512, (3, 3), activation='relu', padding='same') (up6)

batch6 = BatchNormalization(axis=1)(conv6)

conv6 = Conv2D(512, (3, 3), activation='relu', padding='same') (batch6)

batch6 = BatchNormalization(axis=1)(conv6)

up7 = Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same') (batch6)

up7 = concatenate([up7, conv3], axis=1)

conv7 = Conv2D(256, (3, 3), activation='relu', padding='same') (up7)

batch7 = BatchNormalization(axis=1)(conv7)

conv7 = Conv2D(256, (3, 3), activation='relu', padding='same') (batch7)

batch7 = BatchNormalization(axis=1)(conv7)

up8 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same') (batch7)

up8 = concatenate([up8, conv2], axis=1)

conv8 = Conv2D(128, (3, 3), activation='relu', padding='same') (up8)

batch8 = BatchNormalization(axis=1)(conv8)

conv8 = Conv2D(128, (3, 3), activation='relu', padding='same') (batch8)

batch8 = BatchNormalization(axis=1)(conv8)

up9 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same') (batch8)

up9 = concatenate([up9, conv1], axis=1)

conv9 = Conv2D(64, (3, 3), activation='relu', padding='same') (up9)

batch9 = BatchNormalization(axis=1)(conv9)

conv9 = Conv2D(64, (3, 3), activation='relu', padding='same') (batch9)

batch9 = BatchNormalization(axis=1)(conv9)

conv10 = Conv2D(1, (1, 1), activation='sigmoid')(batch9)

model = Model(inputs=[inputs], outputs=[conv10])

model.compile(optimizer=Adam(lr=1e-4), loss=dice\_coef\_loss, metrics=[dice\_coef])

**return** model

model = unet\_model()

callback = [EarlyStopping(monitor='val\_loss', patience=2),

ModelCheckpoint(filepath='D://modelweight//final.h5', monitor='val\_accuracy', restore\_best\_weights=**True**)]

history = model.fit(x\_train, seg, validation\_split=0.2, batch\_size = 10, epochs=10, shuffle=**True**, verbose=1, callbacks=[callback], initial\_epoch = 6)

model.save\_weights('D://modelweight//final.h5')

model.load\_weights('D://modelweight//final.h5')

x = 3500

renk = {0:'magma',

1:'viridis',

2:'gray',

3:'inferno',

4:'cividis',

5:'hot', }

a = 4

example = np.expand\_dims(x\_train[x],axis=0)

pred = model.predict(example)

fig = plt.figure(figsize=(15,10))

plt.subplot(141)

plt.title('Input (Flair + T2)')

plt.imshow(x\_train[x][0],cmap = renk[a])

plt.subplot(142)

plt.title('Radiologist (segmentation)')

plt.imshow( seg[x][0],cmap = renk[a])

plt.subplot(143)

plt.title('Model (Prediction)')

plt.imshow( pred[0][0],cmap = renk[a])

x = 24;

plt.figure(figsize=(15,10))

plt.subplot(3,4,1)

plt.title('T1ce')

plt.imshow(t1ce[x,0,:,:])

plt.subplot(3,4,2)

plt.title('geniş')

plt.imshow(seg\_enhancing[x,0,:,:])

plt.subplot(3,4,3)

plt.title('ödemsiz')

plt.imshow(seg\_nonedema[x,0,:,:])

**def** tumor\_crop(x, pred, size):

crop\_x = []

list\_xy = []

p\_tmp = pred[0,:,:]

p\_tmp[p\_tmp>0.2] = 1

p\_tmp[p\_tmp !=1] = 0

index\_xy = np.where(p\_tmp==1)

**if** index\_xy[0].shape[0] == 0:

**return** [],[]

center\_x = (max(index\_xy[0]) + min(index\_xy[0])) / 2

center\_y = (max(index\_xy[1]) + min(index\_xy[1])) / 2

**if** center\_x >= 176:

center\_x = center\_x-8

length = max(index\_xy[0]) - min(index\_xy[0])

width = max(index\_xy[1]) - min(index\_xy[1])

**if** width <= 64 **and** length <= 64: *#64x64*

img\_x = np.zeros((1,size,size),np.float32)

img\_x[:,:,:] = x[:,int(center\_x - size/2) : int(center\_x + size/2),int(center\_y - size/2) : int(center\_y + size/2)]

crop\_x.append(img\_x)

list\_xy.append((int(center\_x - size/2),int(center\_y - size/2)))

**if** width > 64 **and** length <= 64: *#64x128*

img\_x = np.zeros((1,size,size),np.float32)

img\_x[:,:,:] = x[:,int(center\_x - size/2) : int(center\_x + size/2),int(center\_y - size) : int(center\_y)]

crop\_x.append(img\_x)

list\_xy.append((int(center\_x - size/2),int(center\_y - size)))

img\_x = np.zeros((1,size,size),np.float32)

img\_x[:,:,:] = x[:,int(center\_x - size/2) : int(center\_x + size/2),int(center\_y + 1) : int(center\_y + size + 1)]

crop\_x.append(img\_x)

list\_xy.append((int(center\_x - size/2),int(center\_y)))

**if** width <= 64 **and** length > 64: *#128x64*

img\_x = np.zeros((1,size,size),np.float32)

img\_x[:,:,:] = x[:,int(center\_x - size) : int(center\_x),int(center\_y - size/2) : int(center\_y + size/2)]

crop\_x.append(img\_x)

list\_xy.append((int(center\_x - size),int(center\_y - size/2)))

img\_x = np.zeros((1,size,size),np.float32)

img\_x[:,:,:] = x[:,int(center\_x + 1) : int(center\_x + size + 1),int(center\_y - size/2) : int(center\_y + size/2)]

crop\_x.append(img\_x)

list\_xy.append((int(center\_x),int(center\_y - size/2)))

**if** width > 64 **and** length > 64: *#128x128*

img\_x = np.zeros((1,size,size),np.float32)

img\_x[:,:,:] = x[:,int(center\_x - size) : int(center\_x),int(center\_y - size) : int(center\_y)]

crop\_x.append(img\_x)

list\_xy.append((int(center\_x - size),int(center\_y - size)))

img\_x = np.zeros((1,size,size),np.float32)

img\_x[:,:,:] = x[:,int(center\_x + 1) : int(center\_x + size + 1),int(center\_y - size) : int(center\_y)]

crop\_x.append(img\_x)

list\_xy.append((int(center\_x),int(center\_y - size)))

img\_x = np.zeros((1,size,size),np.float32)

img\_x[:,:,:] = x[:,int(center\_x - size) : int(center\_x),int(center\_y + 1) : int(center\_y + size + 1)]

crop\_x.append(img\_x)

list\_xy.append((int(center\_x - size),int(center\_y)))

img\_x = np.zeros((1,size,size),np.float32)

img\_x[:,:,:] = x[:,int(center\_x + 1) : int(center\_x + size + 1),int(center\_y + 1) : int(center\_y + size + 1)]

crop\_x.append(img\_x)

list\_xy.append((int(center\_x),int(center\_y)))

**return** np.array(crop\_x) , list\_xy

layer = 11370

image\_example\_1, coordinate\_1 = tumor\_crop(t1ce[layer,:,:,:],seg[layer,:,:,:],64)

image\_example\_2, coordinate\_2 = tumor\_crop(seg\_enhancing[layer,:,:,:],seg[layer,:,:,:],64)

image\_example\_3, coordinate\_3 = tumor\_crop(seg\_nonedema[layer,:,:,:],seg[layer,:,:,:],64)

t1ce.shape

plt.figure(figsize=(15,10))

plt.subplot(3,2,1)

plt.title('Enhancing Tumor(top left)')

plt.imshow(image\_example\_2[0,0,:,:])

plt.subplot(3,2,3)

plt.title('Enhancing Tumor(bottom left)',y=-0.25)

plt.imshow(image\_example\_2[1,0,:,:])

plt.subplot(3,2,2)

plt.title('Enhancing Tumor(top right)')

plt.imshow(image\_example\_2[2,0,:,:])

plt.subplot(3,2,4)

plt.title('Enhancing Tumor(bottom right)',y=-0.25)

plt.imshow(image\_example\_2[3,0,:,:])

**def** tumor2array(tumor,segmentasyon):

liste = []

**for** i **in** range(len(tumor)):

crop , coordinate\_x = tumor\_crop(tumor[i,:,:,:],segmentasyon[i,:,:,:],64)

**if** crop == []:

**pass**

**elif** crop.shape[0] ==1:

liste.append(crop[0])

**elif** crop.shape[0] ==2:

liste.append(crop[0])

liste.append(crop[1])

**elif** crop.shape[0] ==4:

liste.append(crop[0])

liste.append(crop[1])

liste.append(crop[2])

liste.append(crop[3])

**return** np.array(liste)

t1ce\_array = tumor2array(t1ce,seg)

enhancing\_array = tumor2array(seg\_enhancing,seg)

odemsiz\_array = tumor2array(seg\_nonedema,seg)

K.set\_image\_data\_format('channels\_first')

**def** dice\_coef(y\_true, y\_pred): *#[piksel farki] https://i.stack.imgur.com/OsH4y.png*

smooth = 0.005

y\_true\_f = K.flatten(y\_true)

y\_pred\_f = K.flatten(y\_pred)

intersection = K.sum(y\_true\_f \* y\_pred\_f)

**return** (2. \* intersection + smooth) / (K.sum(y\_true\_f) + K.sum(y\_pred\_f) + smooth)

**def** dice\_coef\_loss(y\_true, y\_pred):

**return** 1-dice\_coef(y\_true, y\_pred)

**def** unet\_model\_7():

inputs = Input((1, 64, 64))

conv1 = Conv2D(64, (3, 3), activation='relu', padding='same') (inputs)

batch1 = BatchNormalization(axis=1)(conv1)

conv1 = Conv2D(64, (3, 3), activation='relu', padding='same') (batch1)

batch1 = BatchNormalization(axis=1)(conv1)

pool1 = MaxPooling2D((2, 2)) (batch1)

conv2 = Conv2D(128, (3, 3), activation='relu', padding='same') (pool1)

batch2 = BatchNormalization(axis=1)(conv2)

conv2 = Conv2D(128, (3, 3), activation='relu', padding='same') (batch2)

batch2 = BatchNormalization(axis=1)(conv2)

pool2 = MaxPooling2D((2, 2)) (batch2)

conv3 = Conv2D(256, (3, 3), activation='relu', padding='same') (pool2)

batch3 = BatchNormalization(axis=1)(conv3)

conv3 = Conv2D(256, (3, 3), activation='relu', padding='same') (batch3)

batch3 = BatchNormalization(axis=1)(conv3)

pool3 = MaxPooling2D((2, 2)) (batch3)

conv5 = Conv2D(512, (3, 3), activation='relu', padding='same') (pool3)

batch5 = BatchNormalization(axis=1)(conv5)

conv5 = Conv2D(512, (3, 3), activation='relu', padding='same') (batch5)

batch5 = BatchNormalization(axis=1)(conv5)

up7 = Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same') (batch5)

up7 = concatenate([up7, conv3], axis=1)

conv7 = Conv2D(256, (3, 3), activation='relu', padding='same') (up7)

batch7 = BatchNormalization(axis=1)(conv7)

conv7 = Conv2D(256, (3, 3), activation='relu', padding='same') (batch7)

batch7 = BatchNormalization(axis=1)(conv7)

up8 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same') (batch7)

up8 = concatenate([up8, conv2], axis=1)

conv8 = Conv2D(128, (3, 3), activation='relu', padding='same') (up8)

batch8 = BatchNormalization(axis=1)(conv8)

conv8 = Conv2D(128, (3, 3), activation='relu', padding='same') (batch8)

batch8 = BatchNormalization(axis=1)(conv8)

up9 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same') (batch8)

up9 = concatenate([up9, conv1], axis=1)

conv9 = Conv2D(64, (3, 3), activation='relu', padding='same') (up9)

batch9 = BatchNormalization(axis=1)(conv9)

conv9 = Conv2D(64, (3, 3), activation='relu', padding='same') (batch9)

batch9 = BatchNormalization(axis=1)(conv9)

conv10 = Conv2D(1, (1, 1), activation='sigmoid')(batch9)

model = Model(inputs=[inputs], outputs=[conv10])

model.compile(optimizer=Adam(lr=1e-4), loss=dice\_coef\_loss, metrics=[dice\_coef])

**return** model

model\_odemsiz = unet\_model\_7()

history = model\_odemsiz.fit(t1ce\_array, odemsiz\_array,

validation\_split= 0.20,

batch\_size = 10,

epochs= 5,

shuffle=**True**,

verbose=1)

model\_odemsiz.save\_weights('D://modelweight//model\_odemsiz\_array.h5')

model\_enhancing = unet\_model\_7()

history = model\_enhancing.fit(t1ce\_array, enhancing\_array,

validation\_split= 0.20,

batch\_size = 10,

epochs= 5,

shuffle=**True**,

verbose=1)

model\_enhancing.save\_weights('D://modelweight//model\_enhancing\_array.h5')

model\_enhancing.load\_weights('D://modelweight//model\_enhancing\_array.h5')

model\_odemsiz.load\_weights('D://modelweight//model\_odemsiz\_array.h5')

model.load\_weights('D://modelweight//final.h5')

x = 11370

image\_example , coordinate = tumor\_crop(t1ce[x,:,:,:],seg[x,:,:,:],64)

pred\_nonedema = model\_odemsiz.predict(image\_example)

pred\_enhancing = model\_enhancing.predict(image\_example)

pred\_tam = model.predict(x\_train[x:11371,:,:,:])

pred\_tam[pred\_tam > 0.2] = 2

pred\_tam[pred\_tam != 2 ] = 0

pred\_nonedema[pred\_nonedema > 0.2] = 1

pred\_nonedema[pred\_nonedema != 1 ] = 0

pred\_enhancing[pred\_enhancing > 0.2] = 4

pred\_enhancing[pred\_enhancing != 4 ] = 0

**def** add\_on(pred\_tam, pred\_nonedema , pred\_enhancing , coordinate):

total = np.zeros((1,240,240),np.float32)

total[:,:,:] = pred\_tam[:,:,:]

**for** i **in** range(pred\_nonedema.shape[0]):

**for** j **in** range(64):

**for** k **in** range(64):

**if** pred\_nonedema[i,0,j,k] != 0 **and** pred\_tam[0,koordinat[i][0]+j,koordinat[i][1]+k] !=0:

total[0,koordinat[i][0]+j,koordinat[i][1]+k] = pred\_nonedema[i,0,j,k]

**if** pred\_enhancing[i,0,j,k] != 0 **and** pred\_tam[0,koordinat[i][0]+j,koordinat[i][1]+k] !=0:

total[0,koordinat[i][0]+j,koordinat[i][1]+k] = pred\_enhancing[i,0,j,k]

**return** total

example = add\_on(pred\_tam[0,:,:,:], pred\_nonedema, pred\_enhancing, coordinate)

plt.imshow(example[0])

plt.imshow(t1ce[15300,0,:,:])

renk = {0:'magma',

1:'viridis',

2:'gray',

3:'inferno',

4:'cividis',

5:'hot', }

a = 1

x = 10800

image\_example , coordinate = tumor\_crop(t1ce[x,:,:,:],seg[x,:,:,:],64)

pred\_nonedema = model\_odemsiz.predict(image\_example)

pred\_enhancing = model\_enhancing.predict(image\_example)

pred\_tam = model.predict(x\_train[x-1:x,:,:,:])

pred\_tam[pred\_tam > 0.2] = 2

pred\_tam[pred\_tam != 2 ] = 0

pred\_nonedema[pred\_nonedema > 0.2] = 1

pred\_nonedema[pred\_nonedema != 1 ] = 0

pred\_enhancing[pred\_enhancing > 0.2] = 4

pred\_enhancing[pred\_enhancing != 4 ] = 0

example = add\_on(pred\_tam[0,:,:,:], pred\_nonedema, pred\_enhancing, coordinate)

plt.figure(figsize=(15,10))

plt.subplot(341)

plt.title('Flair')

plt.axis('off')

plt.imshow(x\_train[x, 0, :, :],cmap= renk[a])

plt.subplot(342)

plt.title('T2')

plt.axis('off')

plt.imshow(x\_train[x, 1, :, :],cmap= renk[a])

plt.subplot(343)

plt.title('T1ce')

plt.axis('off')

plt.imshow(t1ce[x, 0, :, :],cmap= renk[a])

plt.subplot(344)

plt.title('Segmentation of the Radiologist')

plt.axis('off')

plt.imshow(seg\_final[x, 0, :, :],cmap= renk[a])

plt.subplot(345)

plt.title('Enhancing Tumor')

plt.axis('off')

plt.imshow(seg\_enhancing[x, 0, :, :],cmap= renk[a])

plt.subplot(346)

plt.title('Enhancing + Necrosis')

plt.axis('off')

plt.imshow(seg\_nonedema[x, 0, :, :],cmap= renk[a])

plt.subplot(347)

plt.title('Full Tumor')

plt.axis('off')

plt.imshow(seg[x, 0, :, :],cmap= renk[a])

plt.subplot(348)

plt.title('Segmentation of the Radiologist')

plt.axis('off')

plt.imshow(seg\_final[x, 0, :, :],cmap=renk[a])

renk = {0:'magma',

1:'viridis',

2:'gray',

3:'inferno',

4:'cividis',

5:'hot', }

a = 1

x = 10800

image\_example , coordinate = tumor\_crop(t1ce[x,:,:,:],seg[x,:,:,:],64)

pred\_nonedema = model\_odemsiz.predict(image\_example)

pred\_enhancing = model\_enhancing.predict(image\_example)

pred\_tam = model.predict(x\_train[x-1:x,:,:,:])

pred\_tam[pred\_tam > 0.2] = 2

pred\_tam[pred\_tam != 2 ] = 0

pred\_nonedema[pred\_nonedema > 0.2] = 1

pred\_nonedema[pred\_nonedema != 1 ] = 0

pred\_enhancing[pred\_enhancing > 0.2] = 4

pred\_enhancing[pred\_enhancing != 4 ] = 0

example = add\_on(pred\_tam[0,:,:,:], pred\_nonedema, pred\_enhancing, coordinate)

plt.figure(figsize=(15,10))

plt.subplot(341)

plt.title('Flair')

plt.axis('off')

plt.imshow(x\_train[x, 0, :, :],cmap= renk[a])

plt.subplot(342)

plt.title('T2')

plt.axis('off')

plt.imshow(x\_train[x, 1, :, :],cmap= renk[a])

plt.subplot(343)

plt.title('T1ce')

plt.axis('off')

plt.imshow(t1ce[x, 0, :, :],cmap= renk[a])

plt.subplot(344)

plt.title('Segmentation of the Radiologist')

plt.axis('off')

plt.imshow(seg\_final[x, 0, :, :],cmap= renk[a])

plt.subplot(345)

plt.title('Enhancing Tumor')

plt.axis('off')

plt.imshow(pred\_enhancing[3,0,:,:],cmap= renk[a])

plt.subplot(346)

plt.title('Enhancing + Necrosis')

plt.axis('off')

plt.imshow(pred\_nonedema[0, 0, :, :],cmap= renk[a])

plt.subplot(347)

plt.title('Full Tumor')

plt.axis('off')

plt.imshow(pred\_tam[0, 0, :, :],cmap= renk[a])

plt.subplot(348)

plt.title('Segmentation of the Model')

plt.axis('off')

plt.imshow(deneme[ 0, :, :],cmap=renk[a])

**Git Repository Adress Link:** <https://github.com/mustafaberat/TumorDetectionProject>