Brain Tumor Segmentation from Brain MR Scans using UNET and its Comparative Analysis with Drosophila segmentation

Report Submitted to the department of

Computer Science and Engineering

Of

International Institute of Information Technology, Bhubaneswar

In partial fulfillment of the requirements for the degree of Bachelors of Technology

By

Saroj Kumar Sethy

B115048

Sarthak Jain

B115049

Under the Guidance of

Dr. Sanjay Saxena

Department of Computer Science and Engineering
International Institute of Information Technology Bhubaneswar
Bhubaneswar - 751003, India May, 2019

International Institute of Information Technology Bhubaneswar

Bhubaneswar Odisha -751 003, India. www.iiit-bh.ac.in

06/05/ 2019

UNDERTAKING

We declare that the work presented in this thesis titled "**Brain Tumor Detection and Segmentation of FLAIR, T1C and T2 weighted MR Images**", submitted to the Department of Computer Science and Engineering, International Institute of Information Technology, Bhubaneswar, for the award of the Bachelors of Technology degree in Computer Science and Engineering, is our original work. We have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, we accept that our degree may be unconditionally withdrawn.

06/ 05/ 2019

IIIT Bhubaneswar

Saroj Kumar Sethy

B115048

Sarthak Jain

B115049

International Institute of Information Technology Bhubaneswar

Bhubaneswar Odisha -751 003, India. www.iiit-bh.ac.in

06/05/ 2019

CERTIFICATE

Certified that the work contained in the thesis "**Brain Tumor Detection and Segmentation of FLAIR, T1C and T2 weighted MR Images**", by *Saroj Kumar Sethy (B115048) and Sarthak Jain (B115049)* has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

06/05/ 2019

Dr. Sanjay Saxena

Department of Computer Science and Engineering

IIIT, Bhubaneswar

International Institute of Information Technology Bhubaneswar

Bhubaneswar Odisha -751 003, India. www.iiit-bh.ac.in

06/05/ 2019

ACKNOWLEDGEMENT

The elation and gratification of this major project will be incomplete without mentioning all the people who helped us to make it possible, whose gratitude and encouragement were invaluable to us. We would like to thank God, almighty, our supreme guide, for bestowing is blessings upon us in our entire endeavor. We express our sincere gratitude to our mentor, Prof. Sanjay Saxena for his guidance and support. We also thank Mr. Pabitra Kumar Mishra, Mr. Kaustuv Maharana, and Ms. Tanu Purohit for their support, suggestions and being our friends in need. Finally, we thank our parents for their encouragement, moral support, personal attention and care.

06/ 05/ 2019

IIIT Bhubaneswar

Saroj Kumar Sethy

B115048

Sarthak Jain

B115049

Contents

Chapter **Page Number**

List of Figures

Figure **Page Number**

List of Tables

Abstract

With the advent of new technologies in the field of medicine, there is rising awareness of bio-mechanisms, and we are able to treat better ailments than we could do earlier. Deep learning has helped a lot in this study. This paper deals with some application of deep learning in brain tumor segmentation. Brain tumors are difficult to segment automatically given the high variability in the shapes and sizes of the tumors in the brain. We have implemented a U-Net architecture which results in competitive performance and faster runtimes than the state-of-art model. In this research paper we first studied about brain, what are brain tumors and its different types and what are the symptoms of the brain tumors and its effects caused to the living beings. Then by the help of U-Net architecture also called Fully Convolutional network we detect the presence of tumor cells in brain more efficiently and in a less amount of time.

In this project, using the database provided for the brain tumor segmentation (BraTs) challenge by the Medical Image Computing and Computer Assisted Intervention (MICCAI) society we scanned the given scanned images to detect and segment out the tumor cells in brain by the model we have trained. Then we make a comparative analysis of segmentation of two different medical data sets using UNET. At last we have shown the results that how much efficient is our model and how accurate it can predict the tumor cells present in the brain based on various performance metrics. We have also compared various parameters of the UNET to decide the best combination of parameters for our purpose.

Introduction

The Brain is the largest, most complex and vital part in human body. It is made up of more than 105 billion nerves (axon and dendrite) which can communicate through the human body by trillion connections. The reason of our each and every emotions, thoughts, memory and action and reaction in the world is because of the brain. The Brain along with the spinal cord makes up the central nervous system. If by any scenario the brain gets affected due to any reason then some part of the human body also gets affected it means that some part of the human body does not fully function. Major cause of brain getting affected is brain tumor.
Among all the life threatening diseases which are increasing in quick rate, Brain tumor is one of the most dangerous and life threatening disease. Before knowing about brain tumor we should first know about what tumor is. A tumor is an abnormal tissue that grows by uncontrolled cell division, or abnormal growth of cells that serves no purpose. Normal cells grow in a controlled manner as new cells replace old or the damaged ones. For reasons not fully understood, tumor cells reproduce uncontrollably.

Primary Brain Tumor

Primary brain tumor is an abnormal growth that starts in the brain and usually does not spread to others part of the body.
Primary brain tumors may be benign or malignant.

1. Benign Brain Tumor

A benign brain tumor grows slowly, has distinct boundaries, and rarely spreads. Although its cells are not malignant, benign tumors can be life threatening.

2. Malignant Brain tumor

A malignant brain tumor grows quickly, has irregular boundaries, and spreads to nearby brain areas, although they are often called brain cancer, malignant brain tumors do not fit the definition of cancer because they do not spread to organs outside the brain and spine.

Secondary Brain tumor

Secondary Brain tumor begins cancer elsewhere in the body and spreads to the brain. They form when cancer cells are carried in the blood stream.

The most common cancers that spread to the brain are lung and breast.Whether a brain tumor is benign, malignant, or metastatic, all are potentially life-threatening. Some brain tumors cause a blockage of cerebrospinal fluid (CSF) that flows around and through the brain.

Types of Brain Tumor

There are over 120 different types of brain tumors. Common brain tumors include:

Gliomas

- Astrocytoma
- Pilocytic Astrocytoma (grade I Tumor)
- Diffuse Astrocytoma (grade II Tumor)
- Anaplastic Astrocytoma (grade III Tumor)
- Glioblastoma Multiforme (grade IV Tumor)
- Oligodendroglioma (grade III Tumor)
- Ependymoma (grade III Tumor)

- Craniopharyngioma
- Epidermoid
- Lymphoma
- Meningioma
- Pinealoma (pinecytoma, pineoblastoma)
- Medulloblastoma [1-11].

In many cases we have seen that brain tumors which occurs in children are primary brain tumor whereas the tumors which are occurred in adults are mostly secondary brain tumor which means that the cancer has begun from other parts of the body like lungs, breast. In the recent study done by the scientists it have been proven that the 1 in every 4 people who have been diagnosed with cancer they have the secondary brain tumor. The people who suffer from secondary brain tumor as well as diagnosed with cancer they can only live for some few weeks if they reach the last stage.

So people suffering from either benign tumor or malignant tumor after diagnosed should be immediately given medical treatment.

Tumors can affect the brain by damaging and destroying the normal tissue, compressing the normal tissue or by increasing the intracranial pressure.

The symptoms of brain tumor can vary depending upon what type of tumor it is, size of the tumor, where the tumor is located. The symptoms of tumor includes

- Headaches which tends to worsen in the morning
- Having seizures
- Stumbling, dizziness, having difficulty in walking
- Having speech problems(e.g. They tend to have difficulty in finding words)
- Vision problems, getting abnormal eye movements
- Having weakness on one side of the body
- Increased in the intracranial pressure which results in causing drowsiness, headaches, nausea and vomiting and at last having sluggish responses.

The recovery medical treatment is dependent on the type of brain tumor and the health state of the patient. As per WHO (World Health Organization) and American Brain Tumor Association, the most common grading system for classifying benign and malignant tumor is by an use of a scale from grade I to grade IV. On that scale, benign tumors fall under grade I and glioma and malignant tumors fall under grade III and IV glioma. The grade I and II glioma are also called low-grade tumor type and possess a slow growth, whereas grade III and IV are called high-grade tumor types and possess a rapid growth of tumors. If the low-grade brain tumor is left untreated, it is likely to develop into a high-grade brain tumor that is a malignant brain tumor with time. Patients with grade II gliomas require serial monitoring and observations by magnetic resonance imaging (MRI) or computed tomography (CT) scan every 6 to 12 months. Brain tumor might influence any individual at any age, and its impact on the body may not be the same for every individual.

The benign tumors of low-grade I and II glioma are considered to be curative under complete surgical excursion, whereas malignant brain tumors of grade III and IV category can be treated by radiotherapy, chemotherapy, or a combination thereof. The term malignant glioma encompasses both grade III and IV gliomas, which is also referred to as anaplastic astrocytoma. An anaplastic astrocytoma is a mid-grade tumor that demonstrates abnormal or irregular growth and an increased growth index compared

to other low-grade tumors. Furthermore, the most malignant form of astrocytoma, which is also the highest grade glioma, is the glioblastoma. The abnormal fast growth of blood vessels and the presence of the necrosis (dead cells) around the tumor are distinguished glioblastoma from all the other grades of the tumor class. Grade IV tumor class that is glioblastoma is always rapidly growing and highly malignant form of tumors as compared to other grades of the tumors.

The patients who are suffering from the symptoms of brain tumor must start the earlier course of diagnosis undergoing some physical tests, mental tests and the neurological examinations like brain MRI scans. An analysis of the brain tissue provides the established evidence of the presence of brain tumor. This analysis eventually helps the doctors to classify the brain tumor from either least aggressive i.e. benign or the most aggressive one i.e. malignant.

Imaging Tests

Computed Tomography (CT) scan uses an X-ray beam and with the help of a computer to view anatomical structures of the different parts of the body. It starts viewing the brain in the form of slices and also layer-by-layer, taking picture at each slice. A Dye (which is a contrast agent) may be injected into our bloodstream. CT scan is very useful for viewing changes in bone like structures.

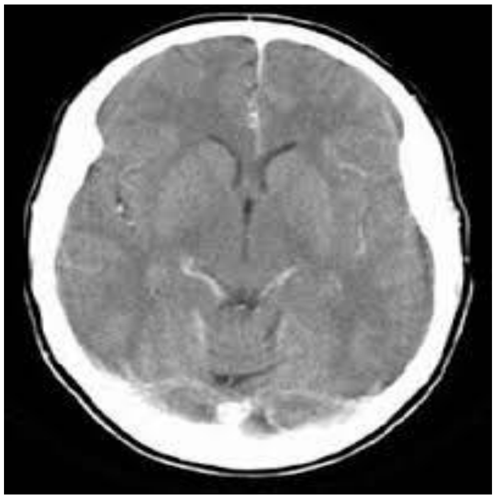


Fig1: Brain CT scan.

Magnetic Resonance Imaging (MRI): It is the most broadly used medical diagnostic method. It is extensively used because of having non-evasive process and has the capability to display the image of various tissues in high contrast and resolution. Magnetic resonance imaging (MRI) is an advanced medical imaging technique that has proven to be an effective tool in the study of the human brain. MRI scans uses a magnetic field and radiofrequency waves to give a informative view of the soft tissues of the brain. It views the image of the brain in 3-dimensionally in the form of slices that can be taken from the side or from the top as a cross-section.

Acquisition of MR images can be done with many different techniques (pulse sequences) and acquisition parameters (called e.g. echo time, TE, repetition time TR etc.) resulting in different image contrast. MRI creates images by picking up different signal intensities from various different tissues which is further dependent on hydrogen or proton content of that tissue. On general MRIs, greater the proton content, brighter the image. But it's also possible to only extract certain direction of proton movement.

So the most common types of MR acquisitions are T1C, FLAIR and T2.

- **T1C Type:** (T1 weighted MRI) Here many tumors show signal enhancement after administration of a contrast agent. Here longitudinal movement of protons is taken into account.
- **FLAIR Type:** (Fluid Attenuated Inversion Recovery MRI) it's a bright signal of the CSF (cerebrospinal fluid), which is suppressed which eventually allows a better detection of small hyper-intense lesions.
- **T2 Type:** (T2-weighted MRI) image contrast is based predominantly on the T2 (transverse)

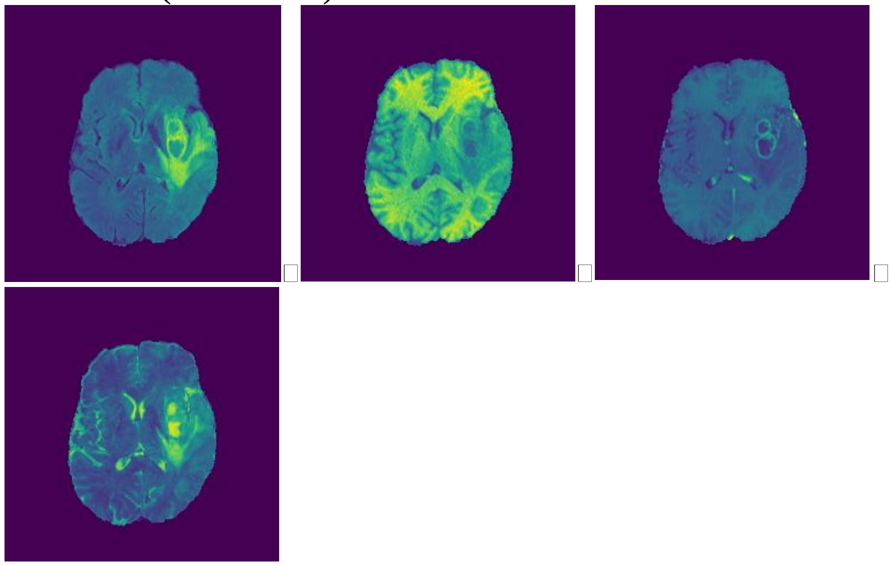


Fig2: a) flair b) t1 c) t1c d) t2 Brain MR Images.

Deep Learning and Machine Learning have helped a lot in this field of study. Machine Learning is an application of AI (Artificial Intelligence) that provides systems the ability to learn from it and then improve from experience without only being programmed to that. Machine learning focuses on the development of the computer programs that can access the data and then use it learn for themselves. Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised

or unsupervised.
Machine Learning (ML) and Artificial Intelligence (AI) have been progressing rapidly in the recent years. Techniques of ML and AI have played important role in medical field like medical image processing, computer-aided diagnosis, image interpretation, image fusion, image registration, image segmentation.

Deep learning plays a major role in the field of bio medical science and the use of automated brain tumor detection and segmentation should be more needed than the detection of brain tumor done manually as the chances of survival of a tumor infected patient can be increased significantly if the tumor is detected accurately in its early stage. So, the accuracy can be increased with proper consistency by making the whole process of detection and segmentation automated.

The detection of the brain tumor done manually is inconsistent, costly and very time-consuming process due to which the accuracy of the results is not constant it means it varies and life of the patient who suffers from brain tumor is at stake. Manual process is very tedious task and the whole process is based on subjective judgments of the people who are involved, so there is a possibility that the same people can have different result in manual detection. Hence, there always a better way to improvise something to get the accurate result and by automated tumor detection method which uses deep learning to find more accurate results should always be preferred.

Literature Survey

Segmentation using U-net architecture is an active area of research and its been helping many people to build their model and participate in ongoing competitions to get better results. [12] They have researched about how the technology is growing day-by-day as new technologies are emerging to the field of science there is more scope of experimenting in the bio medical field. They have used the application of deep learning and proposed a simple fully convolutional network to achieve better results in terms of accuracy in their model. They have also participated in the BraTS challenge held by Medical Image Computing and Computer Assisted Intervention (MICCAI) and achieved a dice of 0.83 in the whole tumor region, 0.72 in the emerging tumor region. [13]They have studied about how the glioblastoma segmentation is challenging in the bio medical field of study and how using convolutional limits and reduce the quality information about the image segmentation. They proposed a U-net architecture and build a good model which can perform and show the results with a small amount of data by using data augmentation and dynamic sampling. And at last they compared their result with the state-of-the-art model in which they have a good performance result.

One deep learning technique has evolved and made a good name in the field of bio medical science field of study and that is U-net. [14] In this research paper they have proposed a Recurrent Convolutional Neural Network (RCNN) which is based on a U-net model also with Recurrent Residual Convolutional Neural Network which is also based on U-net models mainly named as RU-net andR2U-net. The model they have proposed uses 3 things first is U-net model second is residual network which helps in training deeper networks and the final one is the RCNN which helps in feature accumulation and gives a better feature representation for the model. Using these three things gives a super boost to their model in the segmentation field where it is better than the U-net models. [15] As there is a high demand of automated tumor segmentation in the medical field of the study as there is some issues with the 2D U-net architecture or model and 3D U-net model. In 2D model it does not have a good grief on the spatial information it is given and in the 3D model is that it takes a high computation cost as well as a very high performance GPU to give a good performance which is lot of problem. So they have proposed hybrid densely connected U-net (H-dense U-net) in which the 2D

Dense U-net efficiently extracts the intra slice features of the image and the 3D U-net part is for aggregating volumetric contexts and with the fusion of both these thing the build a perfect model which also outperforms the state of the art model as it performs in an end-to-end manner. They also have participated in the MICCAI 2017. [16] In these authors have proposed a segmentation method which is automatic as well as based on Convolutional Neural Networks (CNN) using 3*3 kernels. The use of 3*3 kernels which are smaller kernels allows them to build a deeper architecture. They have also explored and find the use of intensity normalization as a pre-processing step, which is not common in CNN-based segmentation methods, when it is added together with data augmentation it provides an effective way for the MRI images of brain tumor segmentation. In [17] authors have discussed about why automated segmentation is better than the manual segmentation and what the imbalances are in the multi-class of different brain tissues. And for the solution they have proposed a multi-class focal loss method to make the loss function to give more emphasis on the bad classified voxels (unit of 3D image) in MRI images. Their experiment is based on 3D UNET model which enables them to improve the labeling and segmentation accuracy significantly. In [18], researchers have presented a network and training strategy which relies on the strong use of the data augmentation to use the large number of samples more efficiently. They have basically designed UNET models which have a contracting path to capture the context and an expanding path which enables them to know the precise location. Using this model, they have won the ISBI challenge of 2015. To show the variability of UNET architecture. Like brain tumor we can also detect any type of tumor like tumors occur in the breast, lungs, kidney etc. In [19] authors proposed a lung CT image segmentation using the UNET architecture which is one of the most used architectures in deep learning for image segmentation as it is very fast, and the model can be trained with a small amount of data set. Their experimental results show a dice coefficient of 0.9502 which means they can detect lung cancer with accuracy 95.02 %.

Tools and Techniques

As for our implementation purpose of brain tumor detection using U-net architecture we have chosen python language [20] for coding. We have taken python language for programming because as it is one of the fastest growing languages in the world and also python is a high level language or also called General purpose language as it has a variety of implementation in the computer field like it can be used in machine learning, it can be used in software development, web development, GUI (Graphical User Interface) etc. Python is also an Interpreted, object oriented and very interactive language. Python language is easily understandable and has a huge number of libraries which helps it to have a syntax format which helps us to write the code in fewer steps than any other languages like C++, Java etc. Python can be also used as extension language for other applications which are written in other languages that need easy to use scripting.
In our code, the version of python which we use is python 3.6.7. Though the latest version of python is python 3.7, but the support for tensorflow and Keras API is yet to be incorporated in it, which we have extensively utilized to build our architecture..

Tensorflow
Tensorflow is an open source library developed by Google Brain Team. It’s an extremely versatile library, but it was originally created for tasks which require heavy numerical computations. Its main application is in machine learning and deep neural networks. Tensorflow runs faster than python because of having languages like C, C++ as backend. Tensorflow provides several advantages for applications as it provides a python and a C++ API. It has a great compilation time as well as it supports GPUs, CPUs and even distributed processing in a cluster.
We can install Tensorflow by writing the command **pip install tensorflow** in command prompt.
Tensorflow structure is based on the execution of the data flow graph. A data flow graph has 2 basic units:

Nodes: It represents a mathematical operation
Edges: It represents a multi-dimensional array, known as tensor

Keras

Keras is an open source neural network library written in python and minimalists python library for deep learning which can run on top of theano or tensorflow. It was developed to use and make deep learning neural model as fast and easy as possible. It runs on python 2.7 to 3.7 and can be optimally run and execute on CPUs and GPUs. It was developed by Francois chollet.

For installing Keras we should previously have python and tensorflow installed. We can install keras by using the command **pip install keras** in command prompt.

Two of the top numerical platforms in python which provides basic for deep learning or deep neural model and development are theano and tensorflow.

We can construct a deep learning model in Keras as follows:

1) First we have to define a model and then create a sequence and start adding layers.

2) In the second step we compile the model, specify the optimizer, the loss function and the performance metrics.

3) Then we fit the model with the training data. Validation can be performed together And for the final step we have to generate predictions on the test dataset.

If we want to build and test a neural network as quickly as possible with minimum number of lines in code, then we should choose **Keras.** With Keras, we can build simple neural network or very complex neural networks in a short amount of time.

Other modules used in our project

For fulfilment of our purposes, we made use of several modules supported by python besides tensorflow and Keras. We have discussed some of them below.

a) Glob:

The glob module finds all the file locations matching a specified pattern according the UNIX shell rules. It returns the results in a randomized manner. To describe patterns in file location, we make use of symbols like ‘*’, ‘?’, etc. While the characters inside ‘[]’ are matched correctly.

b) OS:

The OS module in python provides us with functionalities to interact with the Operating System. It is python’s one of the standard utility modules. It provides us with means of performing operations that are dependent on Operating System. In our project, we make use of this module to perform file handling operations.

c) NumPy:
NumPy is the fundamental package of python used for scientific computing. Amongst a plethora of functionalities numpy provides us with the following:
i) A multi-dimensional array type object which can store generic.
ii) Functionalities to deal with images.
iii) Useful linear algebra, Fourier transform and random number capabilities.
iv) The ability to integrate with a large number of databases.

d) Matplotlib:
Matplotlib is a 2D plotting library which can be used to generate charts, plots, histograms and graphs for various purposes in the most simplistic manner.

e) OpenCV/CV2:
OpenCV(Open Source Computer Vision Library) is an open-sourced, BSD-licensed library that contains various computer vision algorithms. It has a modular structure which allows it to incorporate several shared or static libraries. We have used this package to perform image processing applications required in our project.

f) Scikit-image:
It is a package for image processing in python. It is open-sourced and hence, can be downloaded freely.

e) TensorBoard:
It is a visualization tool which comes along with tensorflow. It makes life easier when we have to deal with understanding, debugging, and optimizing tensorflow based deep neural architectures. In our project, we use TensorBoard to generate the summary of the architecture created.

What is Convolutional Neural Network (CNN)
A convolutional neural network also known as convnet is an artificial neural network which is so far been most popularly used for analyzing images. CNNs, just like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives many inputs, and takes a weighted sum over them; pass it through an activation function and responds with an output. Most generally we can think of CNN as an artificial neural network that has some type of specialization for being able to

pick out or detect patterns as well as try to analyze and tries to understand the patterns. This pattern detection makes CNN so useful. In other terms we can describe CNN as a deep learning algorithm which can take an image as input and assigns importance to various aspects of image and then tries to differentiate the image from one another.

The architecture of the CNN or convnet is similar to that of connectivity pattern of neurons in the human brain and actually it was inspired by the organization of visual cortex.

A CNN has hidden layers called as convolutional layers this is what makes the CNN so special.

CNN may also have non-convolutional layers as well but the basis of CNN is convolutional layer. Just like other layers these convolutional layer receive inputs and then transform the receive input in some way then outputs the transformed input to the next layer. Actually the convolutional layers are able to detect the patterns. If we want to be more precise each convolutional we need to specify the number of filters it should have. Hence more the number of filters deeper the neural network therefore we will be able to detect much clear and exact images pattern.

Fully Convolutional Network

A fully convolutional network also in short form called as FCN uses convolutional neural network to transform the images into pixel to pixel categories. It is a little bit different than the convolutional neural network as in fully convolutional network it transforms the height as well as width of the middle layer feature map to the original size of the input image through the transposed convolutional layer so that when we try to predict the output it should come in one-to-one corresponding to the input image in the dimension which is spatial in nature (width as well as height).

It is a simple end-to-end semantic segmentation an it takes a pretrained 34-layer ResNet,and it removes all the fully connected layers and adds all the convolutional layers which is already transposed as we skip some connection in the lower level layers. First we do upsampling convolutions with bilinear interpolation filters and zero the final which is the classification layers.

Up-sampling

It matches the number of samples in the class which have majority of the samples with resampling from the class which is having minor number of samples.

Up-sampling is used to enable the precise location. Also it is used to recover spatial information.

Down-sampling
It tries to manage and balance the dataset by matching the number of samples in the minority class with a sample which is randomly taken from the majority class.
Down-sampling is used to extract and interprets the context or captures the semantic information.

Dropout
It means to dropping out the units which include both visible layer and hidden layer in a neural network. In simple terms it means to ignore or not consider both the visible and hidden layer in the training phase on certain set which is chosen randomly. In technical terms it means at each of the training step the individual nodes are either dropped by probability of 1-p or kept with a probability p so that the neural network is reduced. We need the dropout function to prevent the over-fitting.

Over-fitting
Over-fitting occurs when a machine learning algorithm or our statistical model captures the extra occurrences of noise in the data. In simple terms it means when the data is well fitted to the model or the machine learning algorithm. It occurs when the model or the machine learning algorithm show low-bias and high variance. Over-fitting is mostly a result of a complicated model and can be prevented by adding and fitting multiples models and using processes like validation and cross-validation. Over-fitting always leads to bad or poor prediction of the dataset.
We have to increase the flexibility of our model to solve the problem of over-fitting. We should be careful while increasing the flexibility of the model as too much of increase in flexibility will hamper our model. To increase flexibility of the model we can use regularization techniques.
There are 3 types of the regularization techniques to solve the problem of over-fitting are:

(1) L1 Regularization(It is also known as Lasso regularization)
(2) L2 Regularization(It is also known as Ridege regularization)
(3) Elastic Net

Under-fitting
Under-fitting occurs when a machine learning algorithm or our statistical model does not capture the data. In simple terms it means the data is not well fitted enough to the Machine learning algorithm or the statistical model.
It occurs when the model or the machine learning algorithm show high-bias and low

variance. Under-fitting is mostly a result of a very simple and easy model. Under-fitting always leads to bad or poor prediction of the dataset. In order to prevent the model from under-fitting we have to model the expected value of target variable as the nth degree polynomial generating the general Polynomial.
The mathematical expression can be given as:

$$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + €$$

And

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2^2 + \beta_{11} x_1^2 + \beta_{22} x_2^{2\,+}\ \beta_{12} x_1 x_2 + €$$

The training error of the model or machine learning algorithm will decrease as we increase the degree d of the polynomial. At the similar time, the cross validation error of the model will vary as if we increase d to up to a little time it will tend to decrease and then it will increase as the value of d is increased and it will form a curve which will be convex in nature.

Activation Function

As we have studied about the neural network and we know that some activation function are used in the hidden layer and some in the output layer. So now we will know different types activation function. Before studying about activation function we should know all the layers inside a neural network.

1. **Input layer**: This is the layer which accepts the information or feature from the outside to the neural network. Here no computation process occurs only passing of the information takes place.

2. **Hidden Layer:** Here in this layer the nodes are not exposed or shown to the outside world and hence it is the process of the activation. In this layer all the computation process on the features which are input to the network and give results to the output world.

3. **Output Layer:** This layer gives all the information which came from the hidden layer to the outside world.

Activation function is the function which decides whether a neuron in the neural network will get activated or will it not get activated depending on the weighted sum and bias weight. Activation function makes the back propagation process possible as the gradients of the neural network and the error along with the error to update the weighted sum and bias.

A neural network without an activation function is a linear regression process. The function provides a non-linear transformation to the input layer which then is capable of doing more complex task.

Types of Activation function

1. **Linear Activation Function:** The linear activation function has equation which is quite same as the equation of the straight line. It is given by :

   $$\mathbf{y = mx + c}$$

   As we do not need know about how many layers the neural network has, if all the layers are linear function then the output layer is also a linear function.
   The range of linear activation function negative infinity to infinity (-inf to + inf).
   It use is only the last layer which is the output layer.

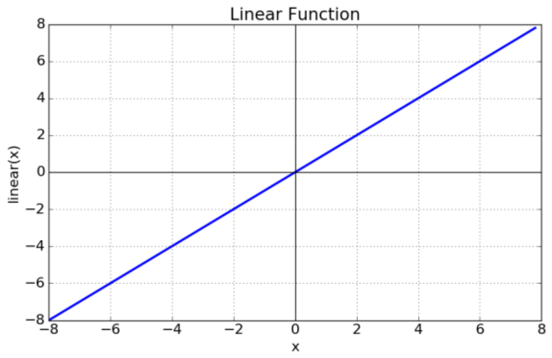


Fig3: linear activation function.

2. **Sigmoid Activation Function:** The **sigmoid activation function** is a function which will give an ‘S’ shape plotted graph from the equation. The equation is given as:

   $$\mathbf{A = 1/ (1 + e^{-x})}$$

   Its nature is non-linear. For a small change in the value of x there will be a large number of a change in the A value.
   The value of the activation form ranges from 0 to 1.

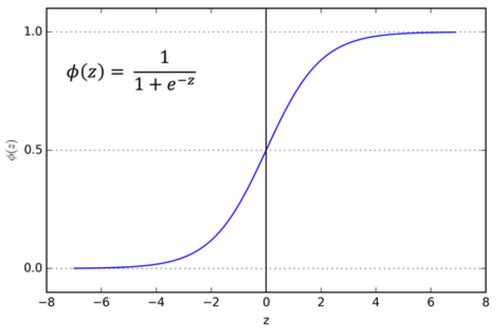


Fig4: sigmoid activation function.

3. **Tanh Activation Function:** Tanh activation function is also known as tangent hyperbolic function. This function is always better than the sigmoid activation function. This two activation function tanh and sigmoid function can both be derived from each other. The equation for the tanh activation is given as:

   tanh(x) = 2* sigmoid(2x) – 1

   The value of the tanh function ranges from -1 to +1.

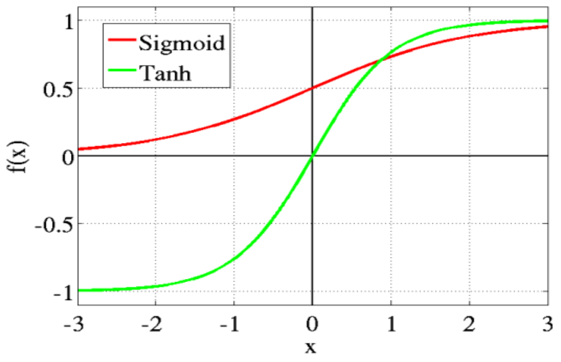


Fig5: tanh activation function.

4. **RELU activation function:** Relu activation function stands for rectified linear Unit. Relu activation function is the most widely used activation function for its characteristics like it is less expensive; computation is faster than the sigmoid and tanh function.

   Its nature is nonlinear which means that we can easily find the error by the process of back propagation and multiple numbers of layers are using relu. The equation for the relu activation function is given as:

   A(x) = max(0,x)

   The value ranges from 0 to infinity.

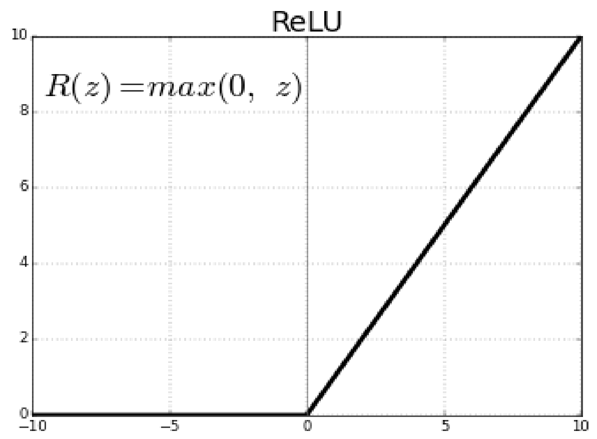


Fig6: relu activation function.

Optimizers

During the training process, we tweak and change the weights of our model to try and minimize the loss function and make our predictions as correct as possible. For this purpose, we use optimizers. Optimizers, bind the model weights with the loss function by updating the model in response to the output of the loss function.
In other words, an optimizer shapes and molds the model into its most accurate form possible by managing the weights. The output of the loss function determines the change optimizer brings to the weights of the model. For our purposes, we make use of Adam optimizer.

Adam

Adam (Adaptive Moment Estimation) optimizer makes use of past slope values to calculate present slope values. Adam also makes use of the concept of momentum by adding fractions of previous slope values to the current one. It is most widely used optimizer for training neural networks.

Data Augmentation

Data augmentation proves of great assistance when the data set is a smaller and is insufficient, given the training testing and validation phases. Thus, not having access to find a large enough data set, using data augmentation, we can somehow manage the small dataset and train the model and still have data remaining for validation and testing and get good predictions.
Data augmentation in simple terms means it is a simpler method or way of creating new data from the existing data by using different types of orientations.

It depends on the data set if the data augmentation will work or not means will it be able to enhance the data from the previous data if it cannot generate a more enhanced data set all the efforts of converting the dataset will go in vain.
We do data augmentation before we feed the data to the model as feeding the data into the model we should have increase the data set size by doing different types of orientation. It adds value to data by adding information which is derived from internal and external sources. Augmentation of the data can be applied to any form of the data but it is mostly useful in case of finding the sales pattern, customer data etc. It can enhance the data quality.

Data augmentation is the last step and then we feed the data into the model architecture. Common techniques used in data augmentation are:

1. Extrapolation Technique: it is based on heuristics data. The required field are updated or changed with the values.

2. Probability technique: It is based on heuristics and analytical statistic model. The values are updated or changed according to the probability of the event occurs.

3. Tagging Technique: In this technique or method the common records are tagged and group so that it will be very much easier to explain and understand the data.

4. Mathematical Techniques: In this technique we use the mathematical operation on the data means we find the value in average, mean and median

There are two ways of converting or do the data augmentation process:

1. Offline Augmentation: This type of data augmentation is preferred for the dataset which are very small as we want to increase the size of the dataset. We will be able to increase the size of the data set by a factor equal to the number of transformation or the orientations.

2. Online Augmentation: This type of data augmentation is preferred for the data set which is very large in number. Here we cannot afford to increase the data set more as it already is large from the starting. Here we divide the data into batches and then we do data augmentation on those batches of the data.

Let's see some of the transformation used in data augmentation which is simple but very effective in nature.

1. Flip (shift): It consist of horizontal shift or vertical shift. Vertical shift we can define it as first we have to rotate the image by 180 degree and then do a horizontal flip. For Horizontal shift we have to rotate the image by 180 degree and then do a vertical flip.

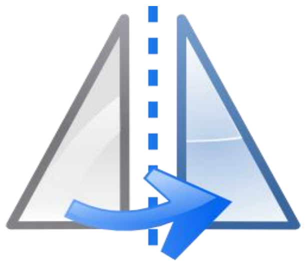


Fig7: Horizontal flip.

2. Rotation: In rotation we rotate the image at certain angle α. One thing to note in this transformation is that image dimension is not preserved after the rotation transformation.

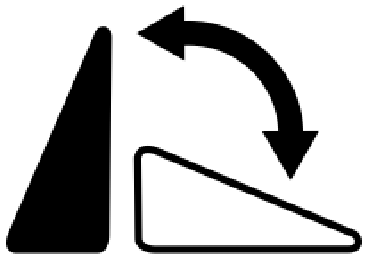


Fig8: 90 degree rotation.

3. Scaling: In scaling the image can be scaled outward or inward, In outward scaling the image size will be bigger than the original image size. In inward scaling the image size will be smaller than the original size.

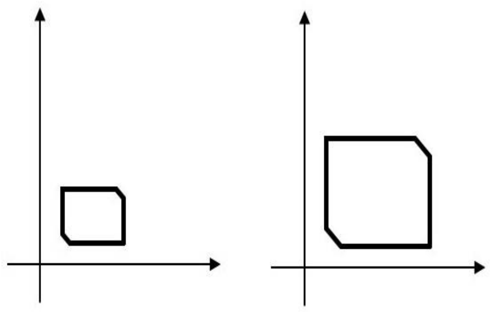


Fig9: Scaling.

4. Crop: In this we take a small amount of portion from the original image and then resize the sample which was taken from the original image to original size of the image.

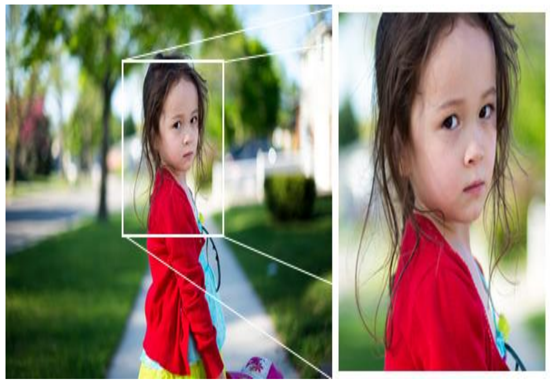


Fig10: Image cropping.

5. Shearing: A transformation that slants the shape of an object is known as shear transformation.

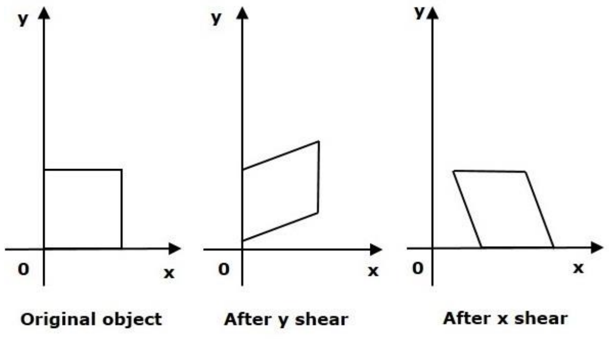


Fig11: shearing.

6. Elastic transformation: Here we add certain distortions to our images with help of random interpolations. Elastic transformations are applicable for training the model for brain tumor segmentation.

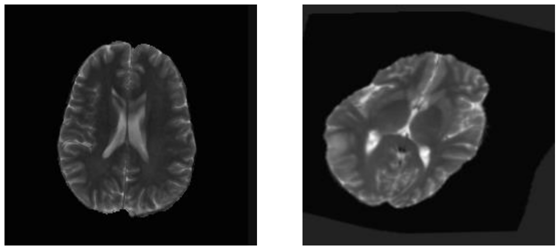


Fig12: elastic transformation.

Performance Metrics

In this paper, we have computed different performance metrics described as below. Here for the results we have taken different parameters to get good performance results.

Dice Similarity Coefficient (DSC): It is a mathematical method to measure the similarity between two samples.

$$\mathbf{DSC = 2^{*}|A \cap B| / |A| + |B|}$$

Where A and B are the number of elements in each set. The Dice Similarity coefficient equals to twice the number of elements common to both sets (Intersection) divided by the sum of the number of elements in each set.
When we apply this formula in the Boolean data as definition of True Positive (TP), False Positive (FP) and False Negative (FN).

$$\mathbf{DSC = 2TP / 2TP+FP+FN}$$

The value of DSC can be in range of 0 to 1. It is different than the other performance matrix which is jaccard coefficient as in the jaccard coefficient then it only counts once True Positive values in both numerator as well as denominator.

Jaccard Coefficient: It is a mathematical method to measure the similarity and diversity between sample sets. It is a commonly used measure of overlap of two sets A and B. Here the two sets don’t need to be of same size. It is also known as IoU metric or Intersection over Union metric.

$$\mathbf{Jaccard\ coefficient = |A \cap B| / | AUB|}$$

$$\mathbf{= |A \cap B| / | A| + |B| - |A \cap B|}$$

Where A and B are elements in each set.

Accuracy: It is the degree to which the result of a measurement or calculation conforms to the correct. In easier terms it means that how much the resulted data is closer to the original data.

$$\textbf{Accuracy} = \mathbf{TP + TN / T + N}$$

Where TP = True Positive
TN = True Negative
P = Condition Positive (the number of real positive cases in data)
N = Condition Negative (the number of real negative cases in data)

Binary Cross Entropy Loss: The binary Cross-entropy loss or it is also known as log loss. It measures the performance of a classification model. Its value varies between 0 and 1. Cross-entropy loss increases when the predicted probability of the dataset starts to vary from the actual label of dataset.

If we want a model to be perfect then it should have loss value as 0.

In binary cross entropy loss where the number of classes M=2, cross entropy can be measured as:

$$-(y\log(p) + (1-y)\log(1-p))$$

If M>2 which means multiclass classification we calculate a separate loss for each class label per observation and sum the result.

$$-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c})$$

Where M = number of classes

Log = natural log

y = binary indicator (values varies from 0 to 1) if class label c is the correct classification for observation o.

p = predicted probability observation o is of class c.

Google Colaboratory

Google Colab is one of the most important tool which we use to implement our project. It is a free cloud service provided by google for AI developers. Since training a deep neural model involves complex computations, running it on mere CPU alone could take a lot of time can even cause the system to crash. In order to perform such large computations, GPU becomes a necessity. However, GPUs are expensive devices which makes their availability difficult. Google with the help of google colab provides free GPU. Colab also provides an interactive environment specifically supporting development over python. It is even pre-updated with popular deep learning and machine learning libraries like Keras, tensorflow, PyTorch, etc.

The GPU has about 15GB memory which sufficient to perform complex computations involved in our project. The only drawback which we face is that the GPU memory is shared between the users and the runtime is reconnected in every 12 hours. The details of the GPU provided are shown here:

```
name: Tesla T4 major: 7 minor: 5
memoryClockRate(GHz): 1.59
pciBusID: 0000:00:04.0
totalMemory: 14.73GiB freeMemory: 14.60GiB
gpu_device.cc:1115 Created TensorFlow
device
(/job:localhost/replica:0/task:0/device:GPU
:0 with 14115 MB memory) -> physical GPU
(device: 0, name: Tesla T4, pci bus id:
0000:00:04.0, compute capability: 7.5)
```

Snippet of GPU details of Google Colab.

U-NET Architecture

Fully Convolutional Networks are networks without any dense layers and only having convolutional and pooling layers. The idea behind the Fully Convolutional Model (FCN) is simple, the model takes an input image and as the output it gives image of same dimensions with no of classes. The main quality of the FCN model is that we can use any sized image as input. UNET is a fully convolutional network and specifically used for image segmentation where the model takes an input image and classifies each pixel.

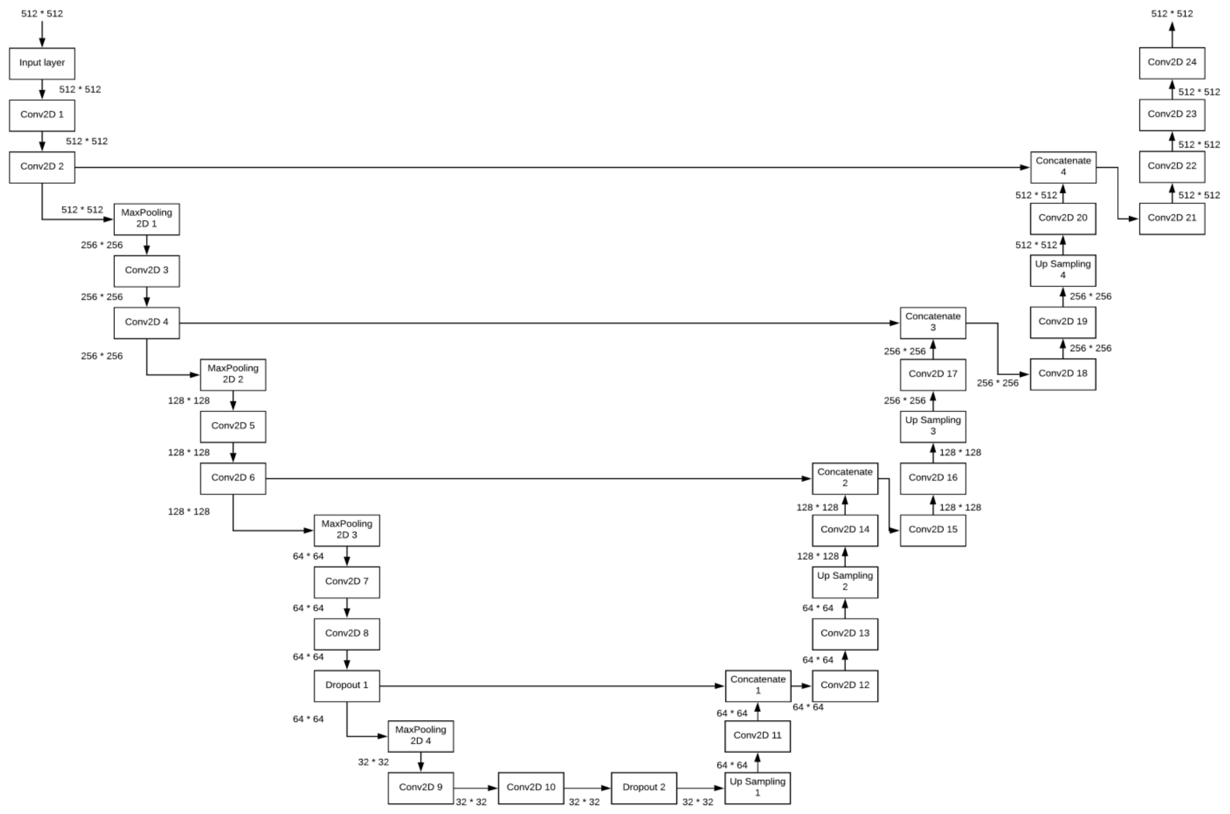


Fig13: Architecture of UNET

UNET learns to segment images in an end to end setting which means a raw image is given as input then we get a segmented image as the output. Figure shows the architecture of the UNET, like all other convolutional network it consists of large number of different operations illustrated by the small arrows.

The input image is fed into the network here in the beginning then the data is propagated through the network along all possible paths and at the end the ready segmentation maps comes out the no of channels as well as the size of the channel is denoted near the arrows. In UNET most of the operations are convolutions followed by a rectified linear activation function and the important design choice is that we only use the part which is valid in the convolutions which means for 3*3 convolutions 1-pixel border is lost. The next operation in a UNET is max pooling operation. It reduces the size of the feature map illustrated by the downward arrow. The max pooling operation acts on each channel separately. It's just propagates the maximum activation from each 2*2 window to the next feature map. After the max pooling operation, we increase the number of feature channel by a factor of 2 all in all the sequence of convolutions and max pooling operation results in a spatial contraction where we gradually increase what and decreases where. Standard classification network ends here and maps all feature to change single output vector. The UNET has an additional expansion path to create a high-resolution segmentation map. This expansion path consists of a sequence of up-convolutions and concatenation with the corresponding high-resolution features from the contracting path. This up-convolution uses a learned kernel to map each feature vector to the 2*2-pixel output window again followed by a rectified linear activation function. Due to the unfettered convolutions this map is smaller than the input image.

Advantages of Using UNET:

(i) It is computationally efficient
(ii)The model is trainable with a small data-set
(iii) In this model is trained end-to-end
(iv)And it is preferable for bio-medical applications

Experimental Setup

a) System Configuration

PROCESSOR	Intel(R) Core(TM) i7-5500U CPU @ 2.40 GHz
RAM	8.00 GB
SYSTEM TYPE	64-bit OS, x64-based processor
OS	Windows 10 Home Single Language
SOFTWARES	Python 3.6.7, Google Colaboratory

Table1: System configuration.

b) About the Datasets

As discussed earlier, we have worked on two different data sets for building a comprehensive analysis. The details of the datasets used is mentioned here.

The drosophila membrane dataset was picked up from the ISBI challenge: http://brainiac2.mit.edu/isbi_challenge [21]. The publicly available training and testing data consists of 30 grayscale images each of the *Drosophila* first instar larva ventral nerve cord (VNC) [22] that used serial section Transmission Electron Microscopy (ssTEM). The images had dimensions 512*512 pixels. As mentioned in the challenge, the imaged volume of training dataset had the measurements 2*2*1.5 μ, and a resolution of 4 × 4 × 50 nm/pixel. Leginon [23] was utilized to capture the images, helping the motion of the FEI electron microscope installed with a Tietz camera and a goniometer-powered mobile grid stage, having a 5600× zoom binned at 2, which delivers the 4 × 4 nm per pixel resolution. This ensured that the image volumes were delivered in a very anisotropic manner.

The aim of the challenge was to generate from the input grayscale EM images fig 14, an accurate boundary map fig 15. The boundary map was defined as binary image where pixel values are either a ‘0’ or a ‘1’, where ‘1’ depicts a pixel inside a cell while ‘0’ depicts a pixel at a boundary between neurite cross sections.

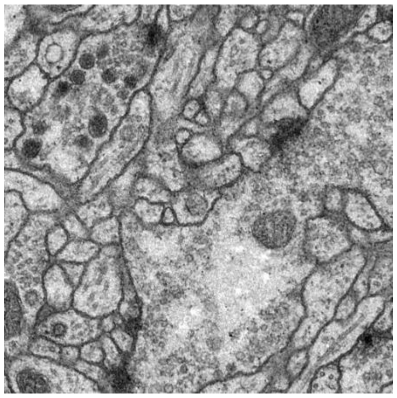


Fig 14: EM image

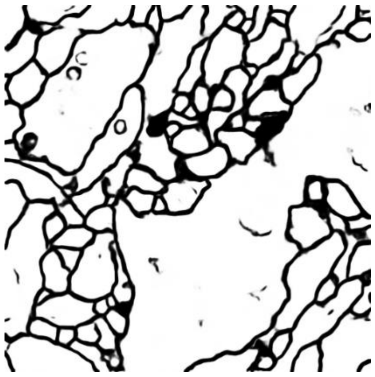


Fig 15: Corresponding Binary Map

The brain MR scans were picked up from BRATS 15 dataset (https://www.smir.ch/BRATS/Start2015[24]). The images are all multimodal MR scans which are saved as NIfTI Files (.nii.gz). The dataset contains 4 types of images: a) native (**T1**) and b) post-contrast T1-weighted (**T1Gd**), c) T2-weighted (**T2**), and d) T2 Fluid Attenuated Inversion Recovery (**FLAIR**) volumes. Different clinical protocols and scanners from 19 different institutions were used to acquire these images. The images were all manually segmented by one to four raters. The raters followed same protocol to generate annotations. The annotations were approved by experienced neuro-radiologists. They were also pre-processed so as to skull-stripe, co-register the images to the same anatomically and to have the same resolution (1 mm^3). Since tumors from flair images can be segmented out most efficiently, we have worked on flair images template.

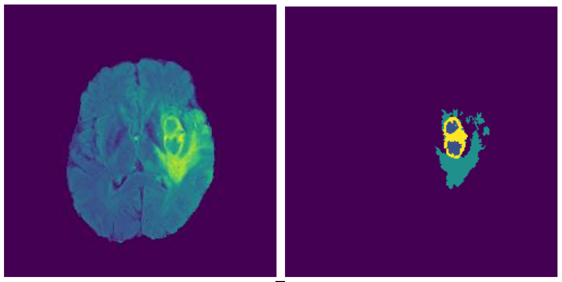


Fig16: Flair image slice and corresponding label.

c) The different phases

We divided our project into different phases:
Phase1: Developing the UNET architecture
Phase2: Preparing and processing the data sets
Phase3: Training and validating the model
Phase4: Prediction and saving the results
Phase5: Performance evaluation

Phase1. Developing the UNET architecture:

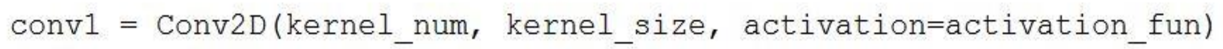
The UNET architecture was developed using python 3.6.7 using Tensorflow library and Keras API. As shown in the diagram earlier, the model we developed contains 24 2D convolution layers, 2 dropouts, 4 max pooling layers and 4 up sampling layers. Keras provides simpler modules to develop any deep learning based model architecture. Hence, adding various layers, compiling and training the models becomes significantly simplified. Developing the model is thus, just a matter of:

a) Setting up the layers in a sequence
b) Compiling the model.

a) Setting up the layers

Here we share code snippets to add various layers and to compile the model.

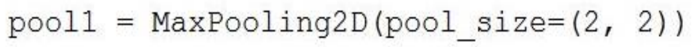
1. This is how we add a convolution layer:



```
conv1 = Conv2D(kernel_num, kernel_size, activation=activation_fun)
```

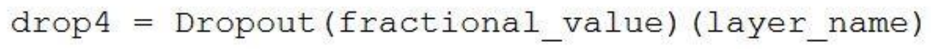
Kernel_num is the no. of convolutions we wish to add in the convolutional layer.
Kernel_size is the convolutional window size (integer if the window is square).
Actvation_fun is where we mention the activation function, we wish to use.

2. This is how we add a max pooling layer.



```
pool1 = MaxPooling2D(pool_size=(2, 2))
```

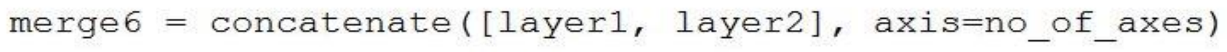
3. For adding some dropout we use the following syntax:



```
drop4 = Dropout(fractional_value)(layer_name)
```

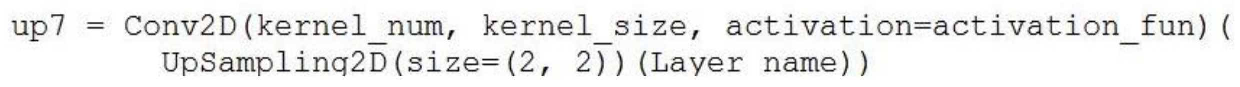
Dropouts are added to prevent over fittning the model.
Fractional_value is the fraction of kernels to drop out from the layer.
Layer_name is the layer from which the drop out is required.

4. To concatenate the output of layer1 to the output of layer2 we use concatenate function.



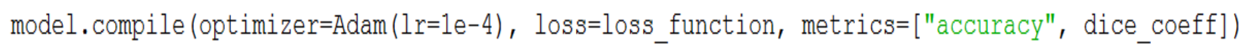
```
merge6 = concatenate([layer1, layer2], axis=no_of_axes)
```

5. Following code is used to perform upsampling.



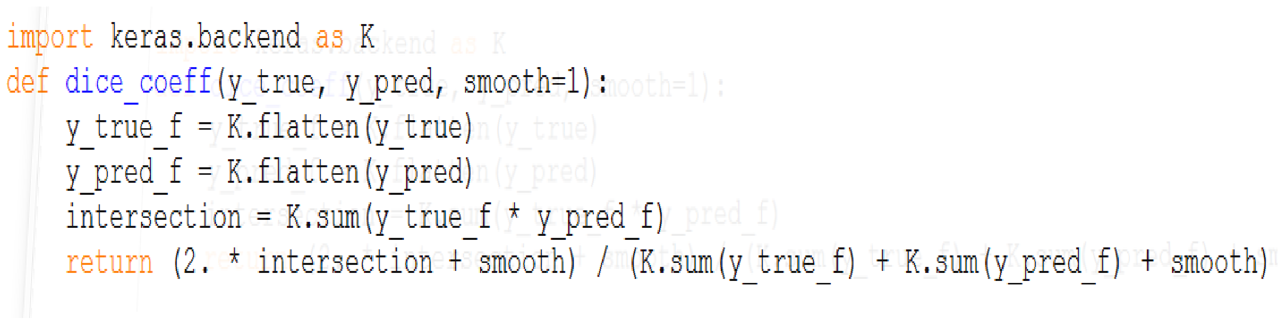
```
up7 = Conv2D(kernel_num, kernel_size, activation=activation_fun)(
        UpSampling2D(size=(2, 2))(Layer name))
```

Now, in order to compile the model we use model.compile module:



```
model.compile(optimizer=Adam(lr=1e-4), loss=loss_function, metrics=["accuracy", dice_coeff])
```

In our model we have used Adam as optimizer, Binary cross entropy as loss function and accuracy, Dice Similarity Coefficient as performance metrics.
Dice Similarity Coefficient is not an in-built metric and hence, we developed a custom function for it. Keras provides us with means of using custom functions as performance metrics. We just need to define the function and pass it as a parameter in the model.compile module. Here is the code snippet for Dice Similarity Coefficient:



```
import keras.backend as K
def dice_coeff(y_true, y_pred, smooth=1):
    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)
```

Code snippet of custom Dice Coefficient Function.

Phase2. Preparing and processing the data sets
Preparing the dataset is an important phase in our project. The images need to follow a proper directory structure failing which will lead the model to throw errors. The images also need to be processed before passing to the model so as to yield good predictions and high performance metric values.
a) Preparing the directory structure.
The diagram below describes the directory structure which we have followed to store training, validation and testing data.

(P.T.O.)

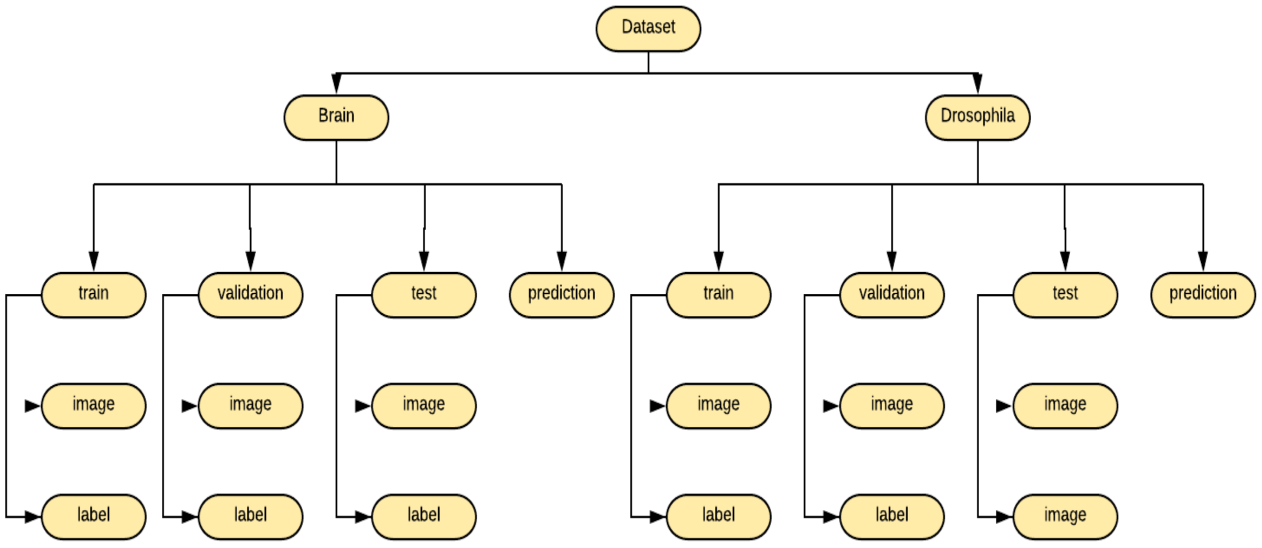
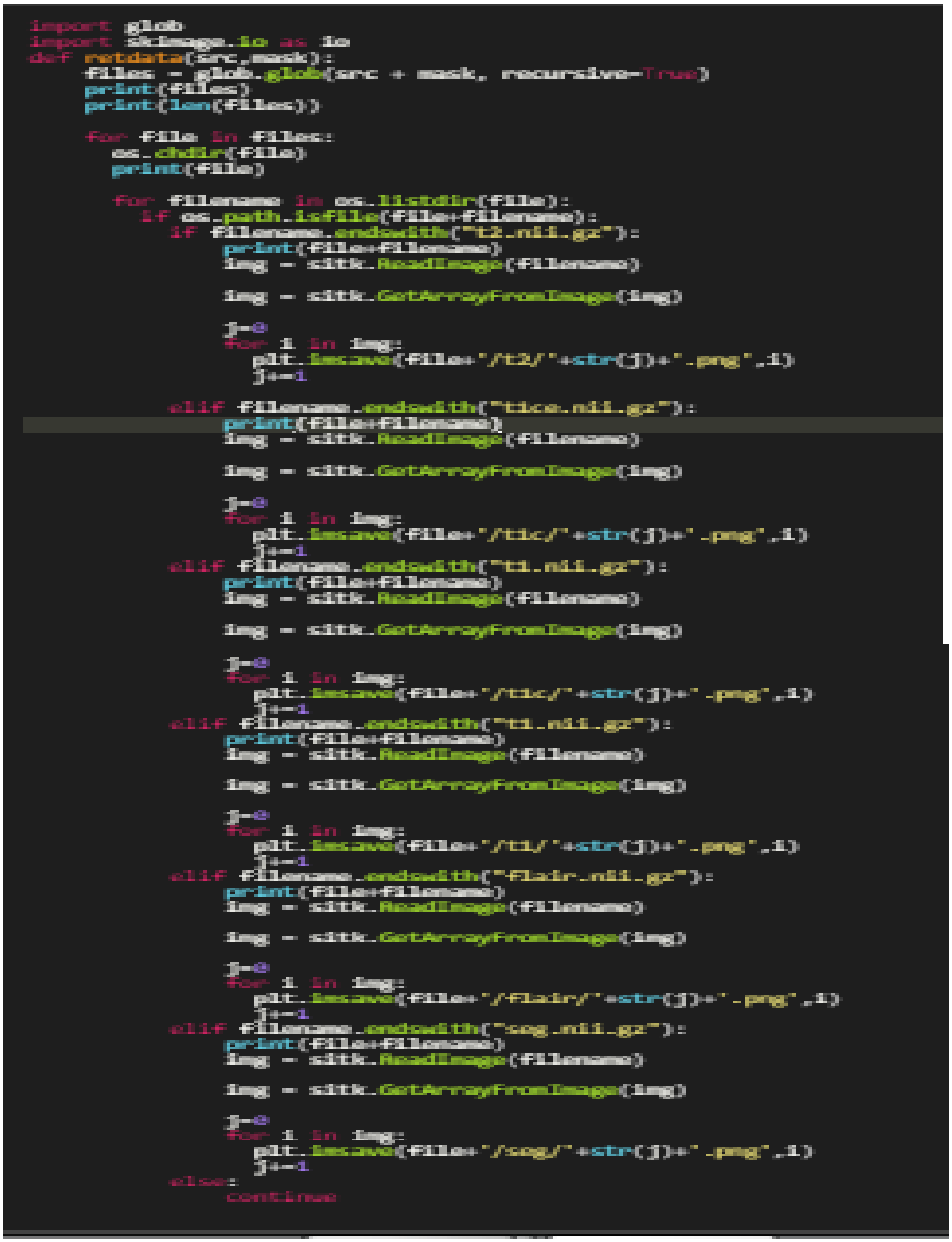


Fig17: Directory Structure for datasets.

The parent directory is named Dataset which contains two subdirectories ti store to different datasets the Brain MR dataset is stored in the Brain folder while the Drosophila dataset is stored in other dataset.
Both these folders further contain folders named train, test, validation and prediction to store training, testing, validation data and the predictions generated respectively.
Train, validation and test directories further sub-folders image and label each to store the images and corresponding annotations.

b) Preparing the dataset

Next, the drosophila training, validation and testing dataset is numbered from 0 to 29. The corresponding annotations are given the same name as well. This how keras understands which label corresponds to which image.
For Brain MR Images, BraTS provides the dataset in the form of NifTi (.nii) files. They are basically compressed files, each of which stores 155 slices of brain scans altogether comprising the 3D view of a patient's brain. Hence, data is stored in folders corresponding to a patient which contains NifTi files of Brain scans and corresponding annotations. Since, the folders are named in a relatively arbitrary manner, we made use of the **glob** module. Further to read the .nii files and extract the image slices, we made use of matplotlib and cv2 modules. The files were read in the form of an array and then saved in .png format with the name 'patient no. (Slice no.)'. The code snippet to extract image slices is shared below.

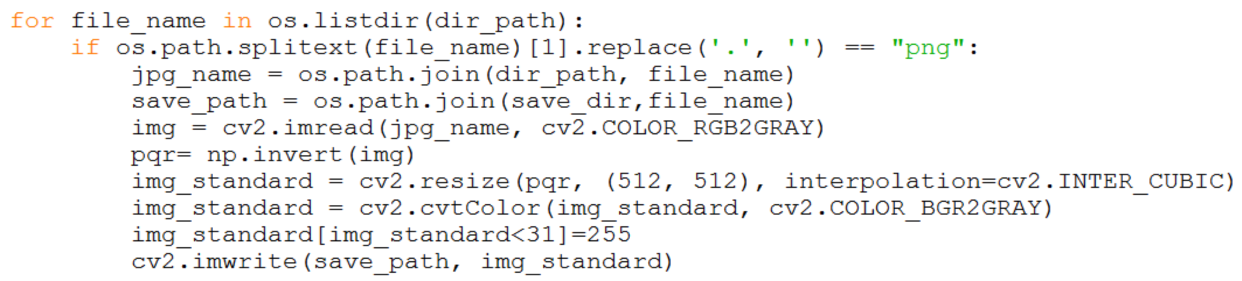


```
import glob
import skimage.io as io
def retdata(src,mask):
    files = glob.glob(src + mask, recursive=True)
    print(files)
    print(len(files))

    for file in files:
      os.chdir(file)
      print(file)

      for filename in os.listdir(file):
        if os.path.isfile(file+filename):
          if filename.endswith("t2.nii.gz"):
              print(file+filename)
              img = sitk.ReadImage(filename)

              img = sitk.GetArrayFromImage(img)

              j=0
              for i in img:
                plt.imsave(file+"/t2/"+str(j)+".png",i)
                j+=1

          elif filename.endswith("t1ce.nii.gz"):
              print(file+filename)
              img = sitk.ReadImage(filename)

              img = sitk.GetArrayFromImage(img)

              j=0
              for i in img:
                plt.imsave(file+"/t1c/"+str(j)+".png",i)
                j+=1
          elif filename.endswith("t1.nii.gz"):
              print(file+filename)
              img = sitk.ReadImage(filename)

              img = sitk.GetArrayFromImage(img)

              j=0
              for i in img:
                plt.imsave(file+"/t1c/"+str(j)+".png",i)
                j+=1
          elif filename.endswith("t1.nii.gz"):
              print(file+filename)
              img = sitk.ReadImage(filename)

              img = sitk.GetArrayFromImage(img)

              j=0
              for i in img:
                plt.imsave(file+"/t1/"+str(j)+".png",i)
                j+=1
          elif filename.endswith("flair.nii.gz"):
              print(file+filename)
              img = sitk.ReadImage(filename)

              img = sitk.GetArrayFromImage(img)

              j=0
              for i in img:
                plt.imsave(file+"/flair/"+str(j)+".png",i)
                j+=1
          elif filename.endswith("seg.nii.gz"):
              print(file+filename)
              img = sitk.ReadImage(filename)

              img = sitk.GetArrayFromImage(img)

              j=0
              for i in img:
                plt.imsave(file+"/seg/"+str(j)+".png",i)
                j+=1
          else:
              continue
```

Code snippet to extract Brain MR Scans in png format.

c) Processing the images

Before feeding the images into the model for training and testing, in order to maintain uniformity, we converted all the images into same dimensions (512*512 pixels). We also transformed images into grayscale images. For brain MR scans, we considered 30 images and extracted the middle slices of all these images and saved them in PNG format. The middle slices were considered as the tumours are more clearly visible compared to other slices. The images obtained had a dark background. Hence, we gave the images and the corresponding labels a white background which makes the required portion of the clearer and further results in efficient segmentation results. In order to give a white background ti the images, we first inverted the images and then converted them into grayscale. Then, the pixels whose intensity was closer to the white pixels were transformed to have a pixel intensity of 255 which corresponds to the white color. For this we made use of os, cv2, numpy and matplotlib modules.



```
for file_name in os.listdir(dir_path):
    if os.path.splitext(file_name)[1].replace('.', '') == "png":
        jpg_name = os.path.join(dir_path, file_name)
        save_path = os.path.join(save_dir,file_name)
        img = cv2.imread(jpg_name, cv2.COLOR_RGB2GRAY)
        pqr= np.invert(img)
        img_standard = cv2.resize(pqr, (512, 512), interpolation=cv2.INTER_CUBIC)
        img_standard = cv2.cvtColor(img_standard, cv2.COLOR_BGR2GRAY)
        img_standard[img_standard<31]=255
        cv2.imwrite(save_path, img_standard)
```

Code snippet for data processing.

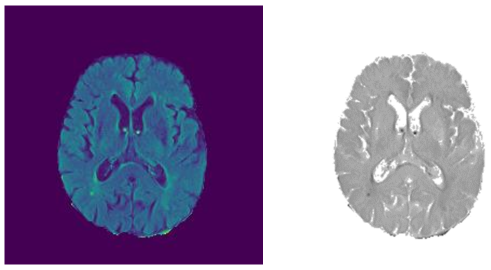


Fig18: a) original image   b) image after processing.

Phase3. Training and validating the model covering all variations

Finally, after the dataset is stored in the above described directory structure and format, we can move to training and validation phase. In order to train and validate the model in the most efficient manner, we make use of data augmentation. Here we make use of **ImageDataGenerator.**

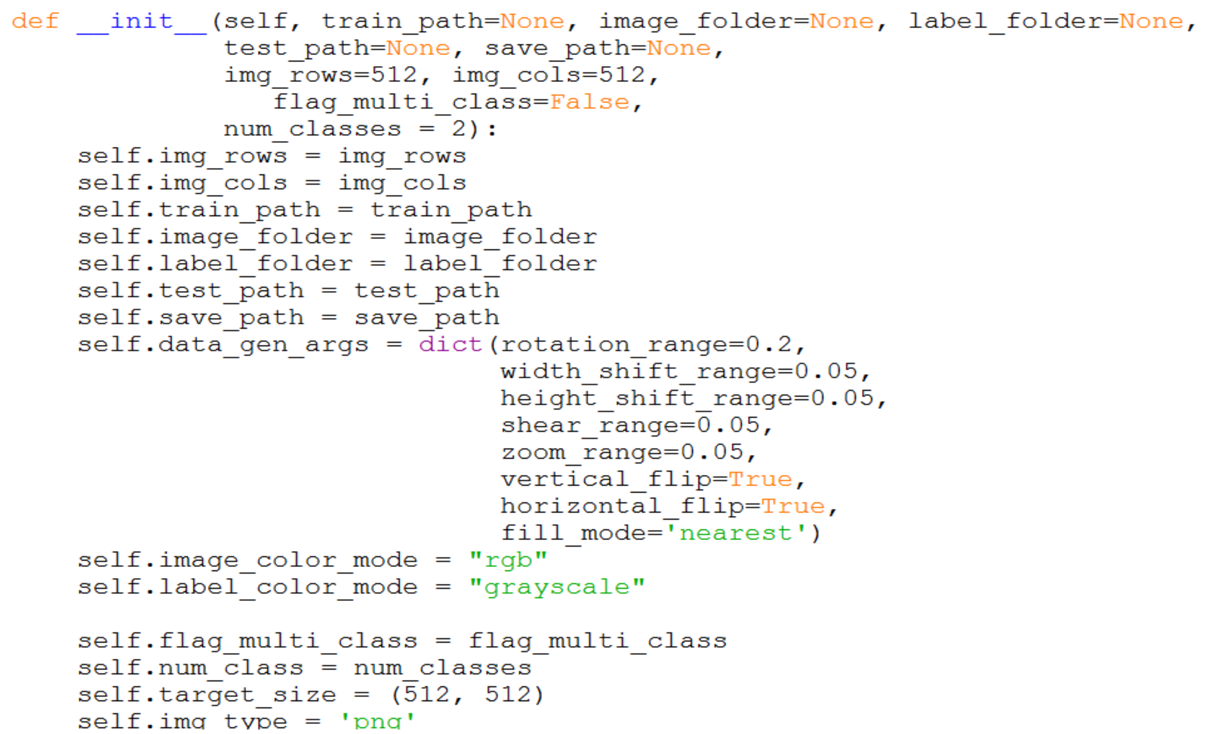
ImageDataGenerator is a special tool provided by keras for working with images grouped together into directories. ImageDataGenerator serves two very important functions:

i) It helps in data augmentation.

ii) If the dataset is very large and can’t be stored in the memory altogether, it helps in passing images to the model batch by batch.

We have used ImageDataGenerator to augment training and validation data and on training, testing and validation data to use the images batch-by-batch.

We first use a function to get initialization data for our DataGenerators:

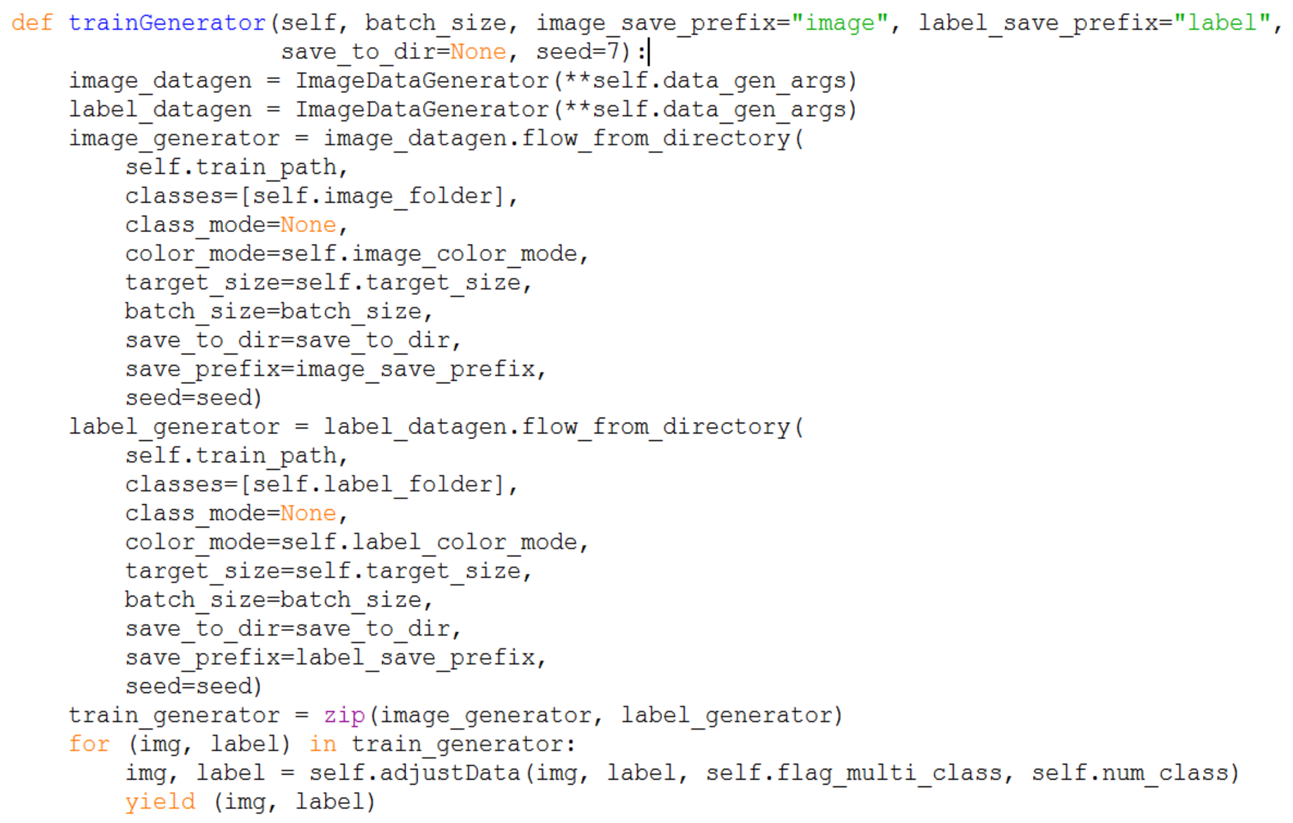


```
def __init__(self, train_path=None, image_folder=None, label_folder=None,
             test_path=None, save_path=None,
             img_rows=512, img_cols=512,
                flag_multi_class=False,
             num_classes = 2):
    self.img_rows = img_rows
    self.img_cols = img_cols
    self.train_path = train_path
    self.image_folder = image_folder
    self.label_folder = label_folder
    self.test_path = test_path
    self.save_path = save_path
    self.data_gen_args = dict(rotation_range=0.2,
                              width_shift_range=0.05,
                              height_shift_range=0.05,
                              shear_range=0.05,
                              zoom_range=0.05,
                              vertical_flip=True,
                              horizontal_flip=True,
                              fill_mode='nearest')
    self.image_color_mode = "rgb"
    self.label_color_mode = "grayscale"

    self.flag_multi_class = flag_multi_class
    self.num_class = num_classes
    self.target_size = (512, 512)
    self.img_type = 'png'
```

Code snippet for initialization

For training and validation, we need images and labels combined passed to the model together. Thus we used a separate function to create a DataGenerator which zips images and labels together and use them for training and validation. Of course separate instances of generators are required for training and validation data. The parameters for data augmentation were provided in the initialization function.

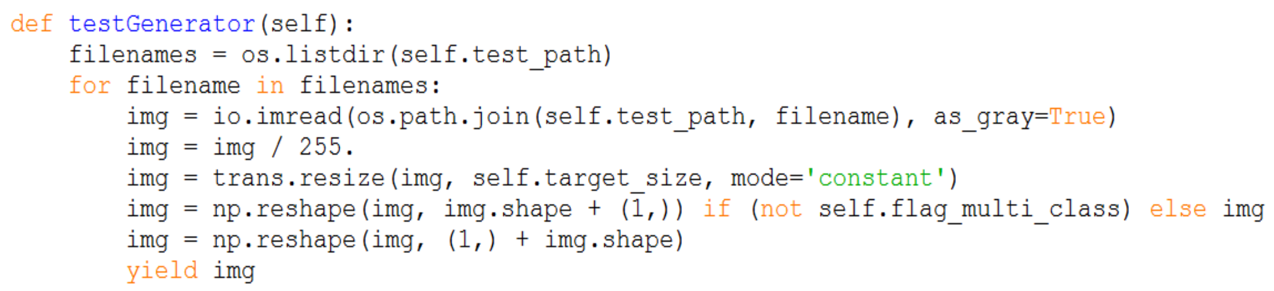
(P.T.O.)



```
def trainGenerator(self, batch_size, image_save_prefix="image", label_save_prefix="label",
                   save_to_dir=None, seed=7):
    image_datagen = ImageDataGenerator(**self.data_gen_args)
    label_datagen = ImageDataGenerator(**self.data_gen_args)
    image_generator = image_datagen.flow_from_directory(
        self.train_path,
        classes=[self.image_folder],
        class_mode=None,
        color_mode=self.image_color_mode,
        target_size=self.target_size,
        batch_size=batch_size,
        save_to_dir=save_to_dir,
        save_prefix=image_save_prefix,
        seed=seed)
    label_generator = label_datagen.flow_from_directory(
        self.train_path,
        classes=[self.label_folder],
        class_mode=None,
        color_mode=self.label_color_mode,
        target_size=self.target_size,
        batch_size=batch_size,
        save_to_dir=save_to_dir,
        save_prefix=label_save_prefix,
        seed=seed)
    train_generator = zip(image_generator, label_generator)
    for (img, label) in train_generator:
        img, label = self.adjustData(img, label, self.flag_multi_class, self.num_class)
        yield (img, label)
```

Code snippet of DataGenerator for training and validation.

For testing we just need the images and the predictions are to be generated. Hence, we used a separate function for testing images.

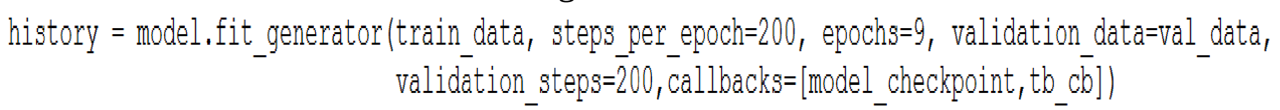


```
def testGenerator(self):
    filenames = os.listdir(self.test_path)
    for filename in filenames:
        img = io.imread(os.path.join(self.test_path, filename), as_gray=True)
        img = img / 255.
        img = trans.resize(img, self.target_size, mode='constant')
        img = np.reshape(img, img.shape + (1,)) if (not self.flag_multi_class) else img
        img = np.reshape(img, (1,) + img.shape)
        yield img
```

Code snippet of DataGenerator for testing.

Next, we simply use **model.fit_generator** command of Keras to begin training and validation. Different parameters like no. of epochs, steps per epoch for training and validation, ImgeDataGenerators for training and validation, and callbacks for saving the model are mentioned.

We also use Tensorboard to save weights of the model trained.



```
history = model.fit_generator(train_data, steps_per_epoch=200, epochs=9, validation_data=val_data,
                              validation_steps=200,callbacks=[model_checkpoint,tb_cb])
```

Code snippet of model.fit

We trained our model for 10 epochs every time for every variation of dataset or parameters. The number of steps per epoch were kept 200 both for training and validation. The resultant values of loss function and performance metrics for training and validation were recorded.

Phase4: Prediction and saving the results

To generate the predictions of each test image we use **model.predict_generator()** command of Keras. The predictions so generated were saved in .png format in the predictions folder corresponding to each dataset. The test performance rsults were also evaluated using **model.evaluate_generator()** command of Keras.

Phase5: Performance evaluation

The efficiency of the predicted results is calculated on the basis of performance metrics like accuracy and dice similarity coefficient and also on the basis of loss function. The results obtained in training, validation and testing phases were tabulated and are discussed in the results and discussions.

Results and Discussions

The results for two different studies are discussed here. Results from study 1 guide the findings of study 2.

Study 1:

In the first study, we trained the UNET over the above mentioned two datasets. We also took into consideration all possible combinations of variations of following parameters:

(i) Other parameters: Table 1 contains the variations of kernel size, dropouts and activation functions taken into consideration.

(ii) Model architecture: Table 2 contains the list of all layers of the UNET and the number of convolutions in each layer of the model corresponding to the two variations considered.

Parameters	Variations		
Kernel size	3*3	5*5	
Dropout	50%	25%	
Activation Function	Relu	Sigmoid	tanh

Table2: Variations in other parameters

Layer (type)	**No. of convolutions/Pool size (Variation#)**	**No. of convolutions/Pool size (Variation*)**
input_1 (Input Layer)	0	0
conv2d_1 (Conv2D)	64	32
conv2d_2 (Conv2D)	64	32
conv2d_3 (Conv2D)	128	64
conv2d_4 (Conv2D)	128	64
conv2d_5 (Conv2D)	256	128
conv2d_6 (Conv2D)	256	128
conv2d_7 (Conv2D)	512	256
conv2d_8 (Conv2D)	512	256
conv2d_9 (Conv2D)	1024	512
conv2d_10 (Conv2D)	1024	512
conv2d_11 (Conv2D)	512	256
conv2d_12 (Conv2D)	512	256
conv2d_13 (Conv2D)	512	256
conv2d_14 (Conv2D)	256	128
conv2d_15 (Conv2D)	256	128
conv2d_16 (Conv2D)	256	128
conv2d_17 (Conv2D)	128	64
conv2d_18 (Conv2D)	128	64
conv2d_19 (Conv2D)	128	64
conv2d_20 (Conv2D)	64	32
conv2d_21 (Conv2D)	64	32
conv2d_22 (Conv2D)	64	32
conv2d_23 (Conv2D)	2	2
conv2d_24 (Conv2D)	1	1

Table3: Variations in no. of Convolutions in different layers of UNET

Following table gives the detailed description of all the possible variations and the values of the accuracy dice coefficient and the loss function obtained for each case. The results correspond to 10 epochs of training for each combination. We have skipped the entries of validation phase as they are almost similar to the training values and the differences are negligible. Plus, understanding the analysis would become complex had they been added.

dataset	model	kernel size	dropout	activation function	loss	accuracy	dice coefficient
1	#	3	0.5	relu	0.05756	0.9834	0.9861
1	#	3	0.5	sigmoid	0.406	0.8841	0.8001
1	#	3	0.5	tanh	0.18371	0.9832	0.9155
1	#	3	0.25	relu	0.2497	0.9843	0.9921
1	#	3	0.25	sigmoid	0.33303	0.8742	0.8369
1	#	3	0.25	tanh	0.24995	0.921	0.8799
1	#	5	0.5	relu	0.24966	0.9843	0.9921
1	#	5	0.5	sigmoid	0.6152	0.7204	0.699
1	#	5	0.5	tanh	0.2073	0.9702	0.9026
1	#	5	0.25	relu	0.24966	0.9843	0.9921
1	#	5	0.25	sigmoid	0.6801	0.7805	0.6691
1	#	5	0.25	tanh	0.15324	0.9762	0.9325
1	*	3	0.5	relu	0.0535	0.9844	0.9869
1	*	3	0.5	sigmoid	0.61684	0.7801	0.6983
1	*	3	0.5	tanh	0.2084	0.9846	0.9021
1	*	3	0.25	relu	0.2899	0.9881	0.9934
1	*	3	0.25	sigmoid	0.2732	0.8846	0.8677
1	*	3	0.25	tanh	0.4498	0.8962	0.7784
1	*	5	0.5	relu	0.2497	0.9843	0.9921
1	*	5	0.5	sigmoid	0.4276	0.7934	0.7893
1	*	5	0.5	tanh	0.3836	0.9002	0.8113
1	*	5	0.25	relu	0.0478	0.9864	0.988
1	*	5	0.25	sigmoid	0.4081	0.8242	0.799
1	*	5	0.25	tanh	0.329	0.8828	0.8389
2	#	3	0.5	relu	0.284	0.8953	0.8839
2	#	3	0.5	sigmoid	0.5975	0.7829	0.6804
2	#	3	0.5	tanh	0.3154	0.8723	0.8586
2	#	3	0.25	relu	0.2857	0.894	0.8835
2	#	3	0.25	sigmoid	0.6877	0.7778	0.6138
2	#	3	0.25	tanh	0.4188	0.8715	0.7787
2	#	5	0.5	relu	0.2717	0.9041	0.8893
2	#	5	0.5	sigmoid	0.5952	0.7829	0.6824
2	#	5	0.5	tanh	0.3685	0.8356	0.8381
2	#	5	0.25	relu	0.6849	0.7829	0.6157
2	#	5	0.25	sigmoid	0.6865	0.7801	0.6146
2	#	5	0.25	tanh	0.3984	0.8782	0.7914
2	*	3	0.5	relu	0.3076	0.876	0.874
2	*	3	0.5	sigmoid	0.533	0.7829	0.751
2	*	3	0.5	tanh	0.3341	0.8861	0.8363
2	*	3	0.25	relu	0.3021	0.8805	0.8765
2	*	3	0.25	sigmoid	0.5621	0.7829	0.7134
2	*	3	0.25	tanh	0.2951	0.8833	0.8682
2	*	5	0.5	relu	0.2306	0.901	0.9097
2	*	5	0.5	sigmoid	0.6617	0.7829	0.6315
2	*	5	0.5	tanh	0.3736	0.8807	0.8092
2	*	5	0.25	relu	0.2828	0.8959	0.8846
2	*	5	0.25	sigmoid	0.5233	0.7829	0.7823
2	*	5	0.25	tanh	0.3178	0.8761	0.855

Table4: detailed description of all the possible variations and corresponding results

Note that, the dataset ‘1’ corresponds to brain MR scans while the dataset ‘2’ corresponds to the EM images of the drosophila.

To make a comparative study we plotted the following graphs:

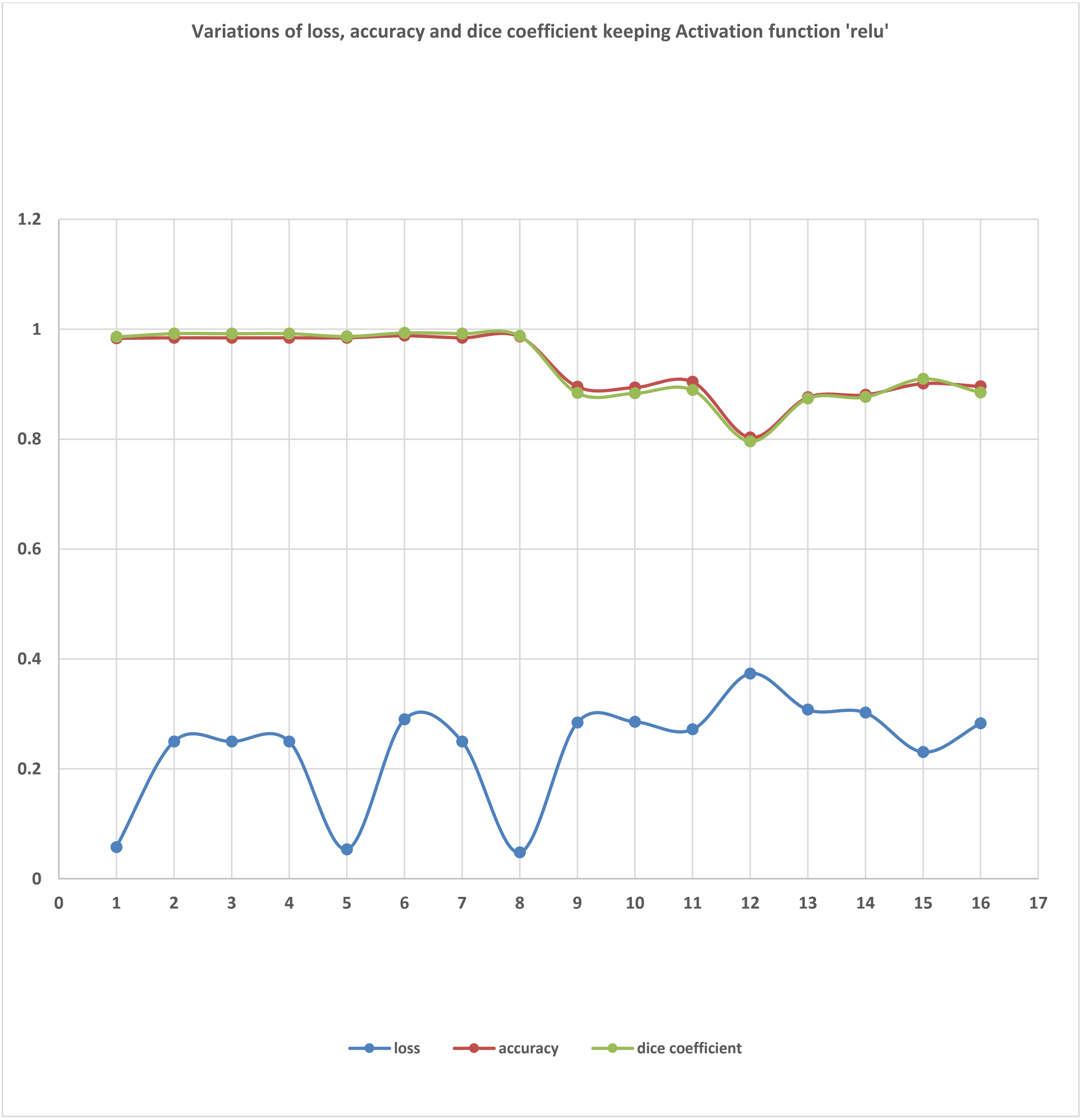


Fig19: Variations of loss, accuracy and dice coefficient keeping Activation function 'relu'

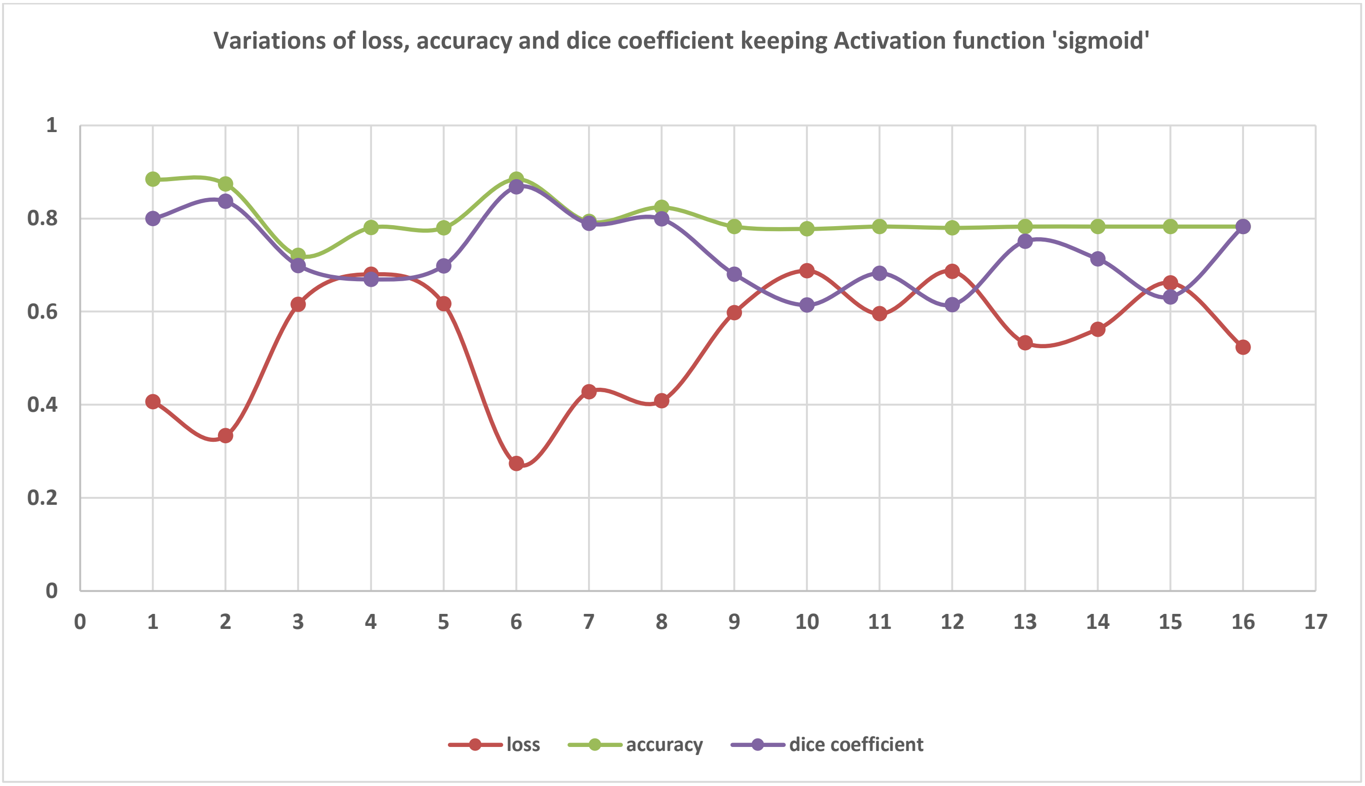


Fig20: Variations of loss, accuracy and dice coefficient keeping Activation function 'sigmoid'

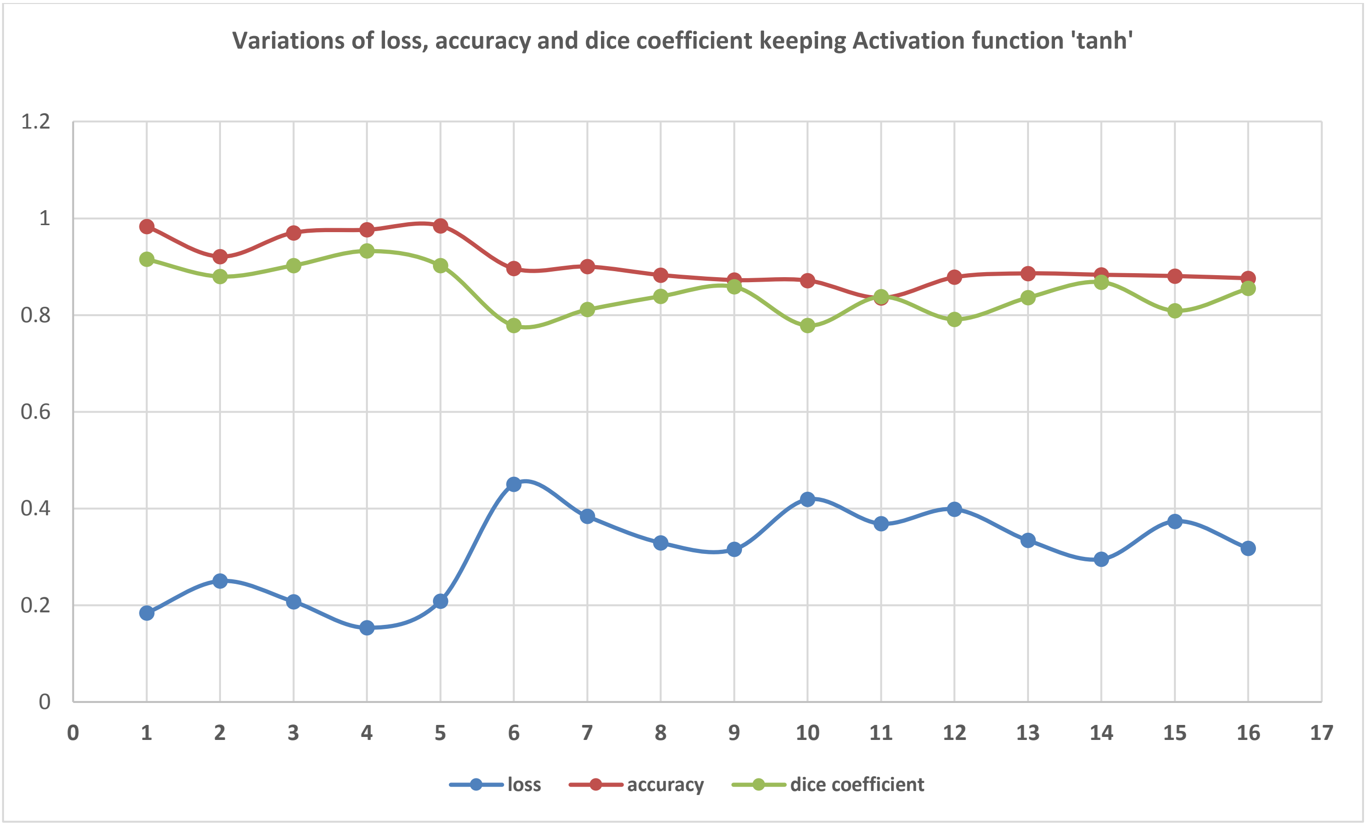


Fig21: Variations of loss, accuracy and dice coefficient keeping Activation function 'tanh'

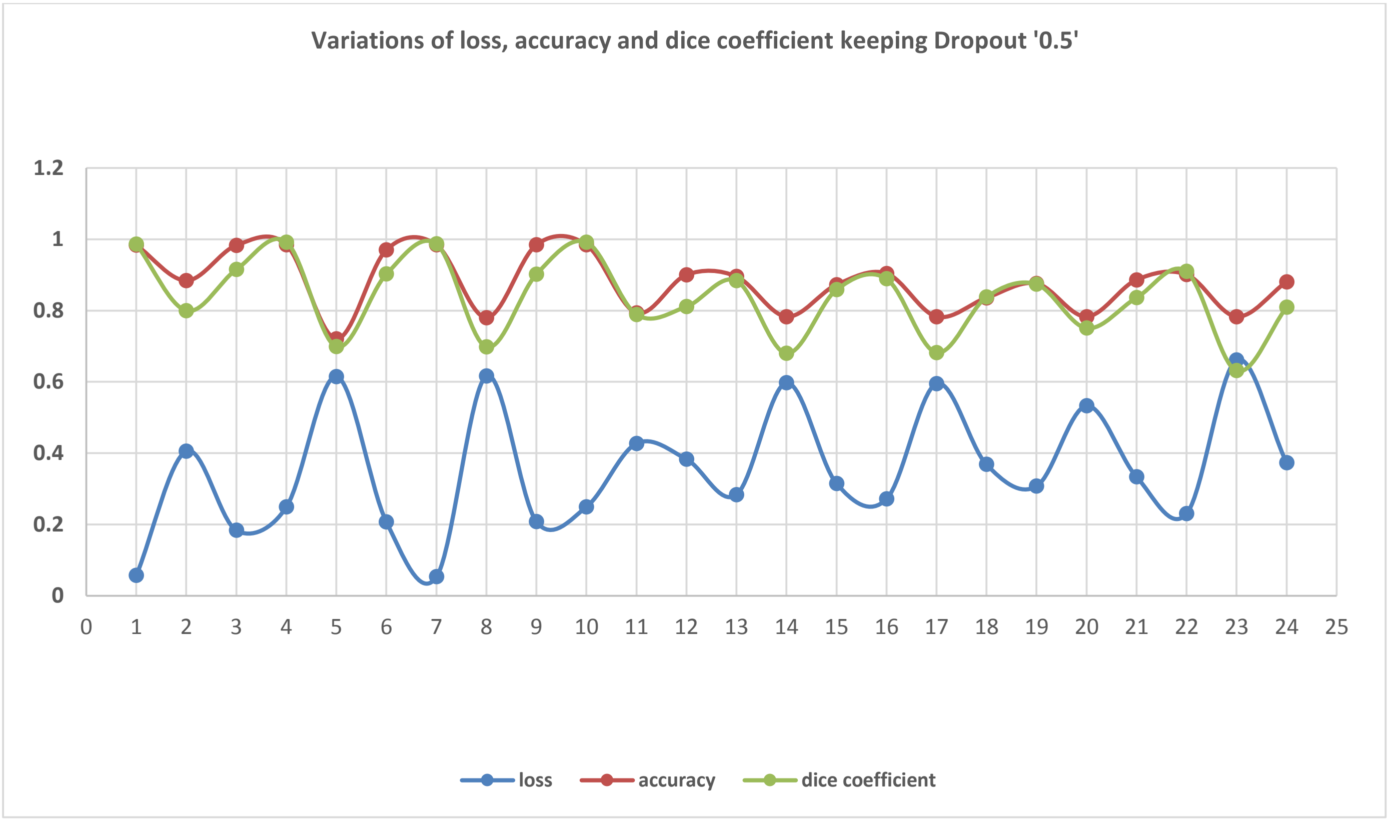


Fig22: Variations of loss, accuracy and dice coefficient keeping dropout ‘0.5’

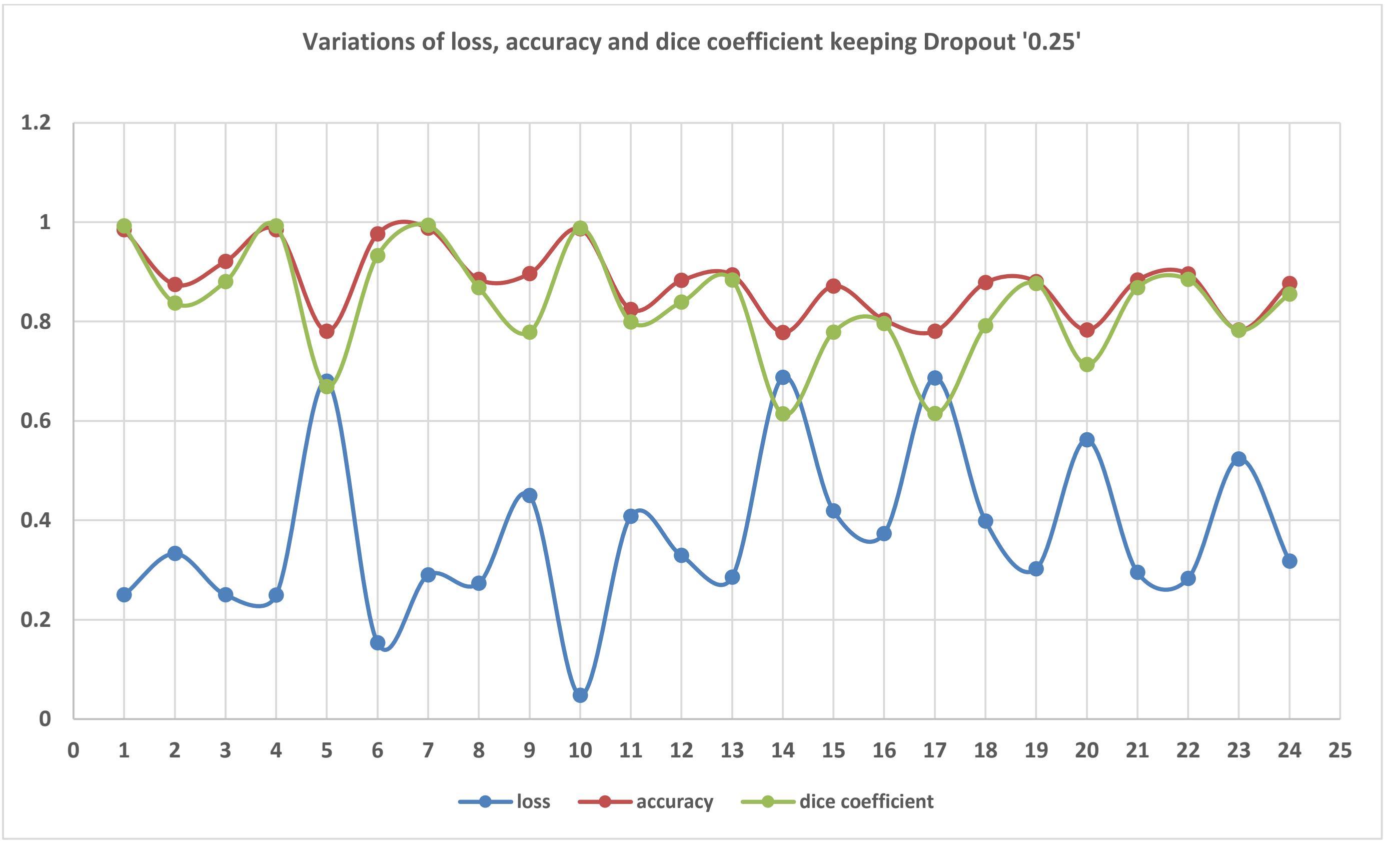


Fig23: Variations of loss, accuracy and dice coefficient keeping dropout ‘0.25’

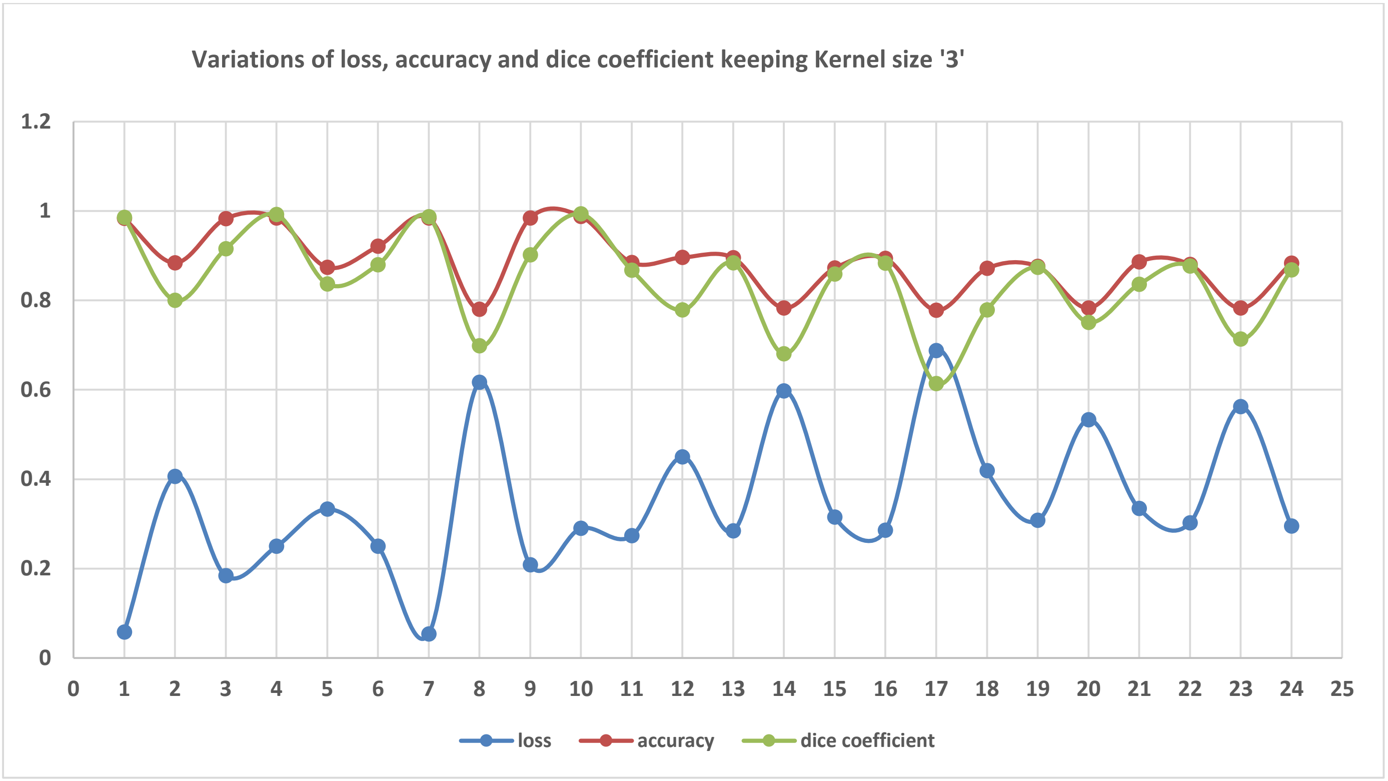


Fig24: Variations of loss, accuracy and dice coefficient keeping kernel size ‘3’

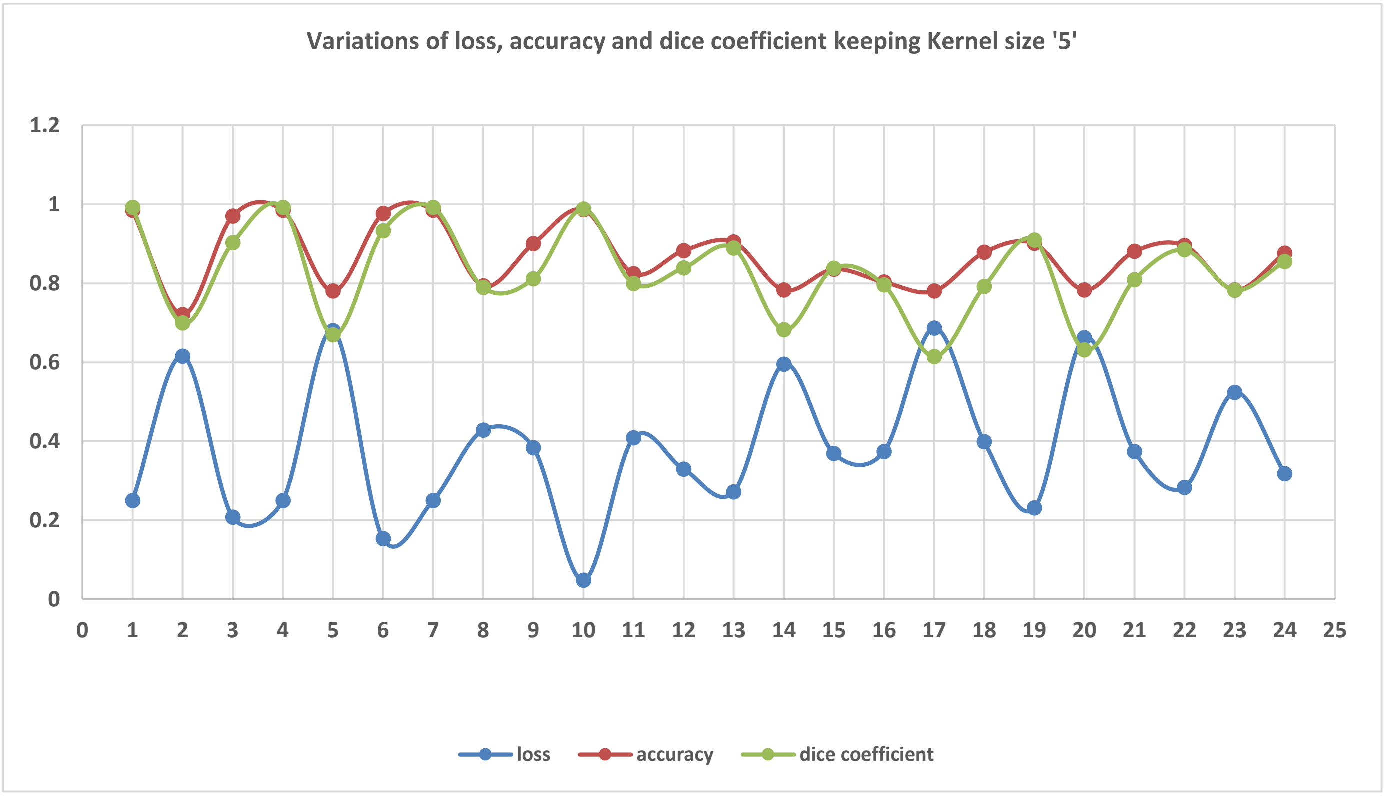


Fig25: Variations of loss, accuracy and dice coefficient keeping kernel size ‘5’

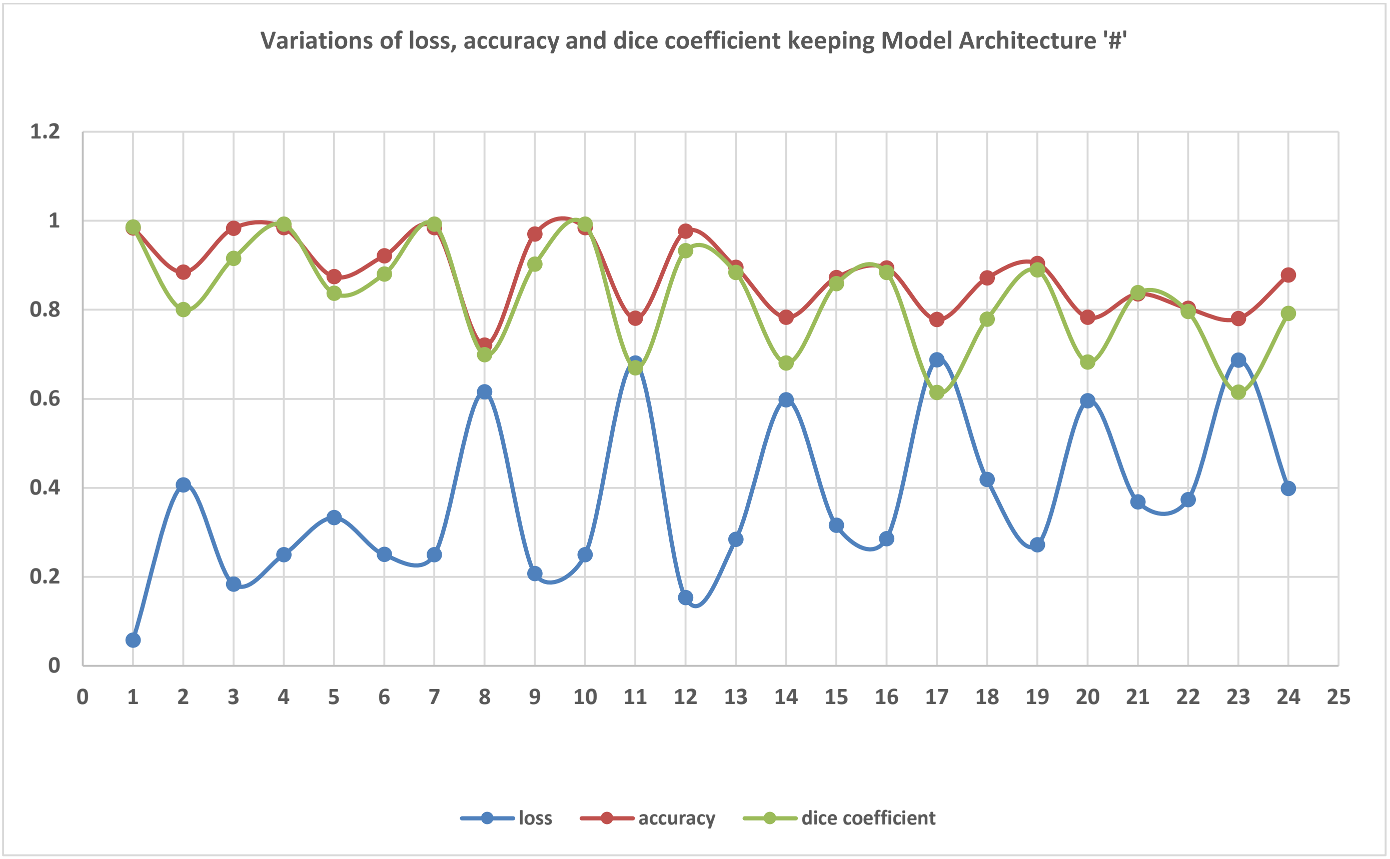


Fig26: Variations of loss, accuracy and dice coefficient keeping model architecture '#'

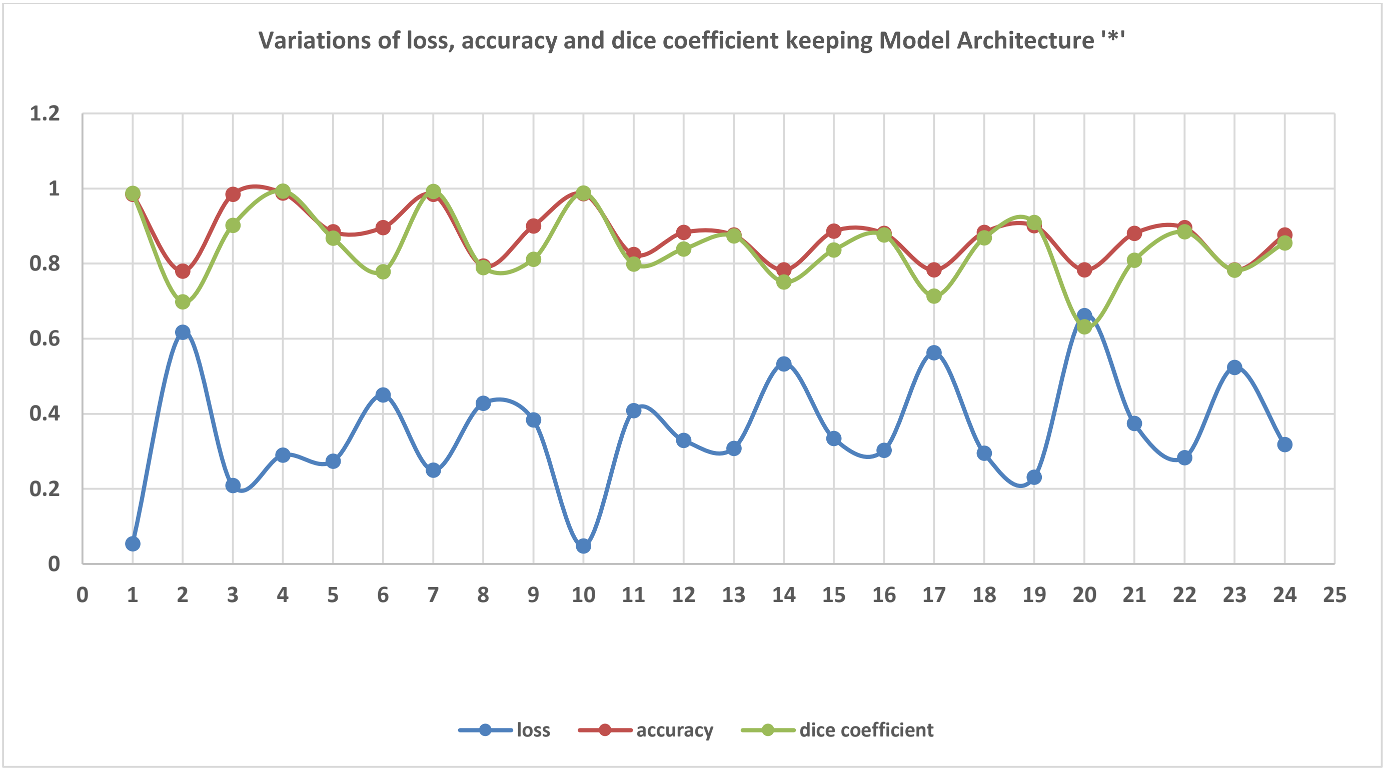


Fig27: Variations of loss, accuracy and dice coefficient keeping model architecture '*'

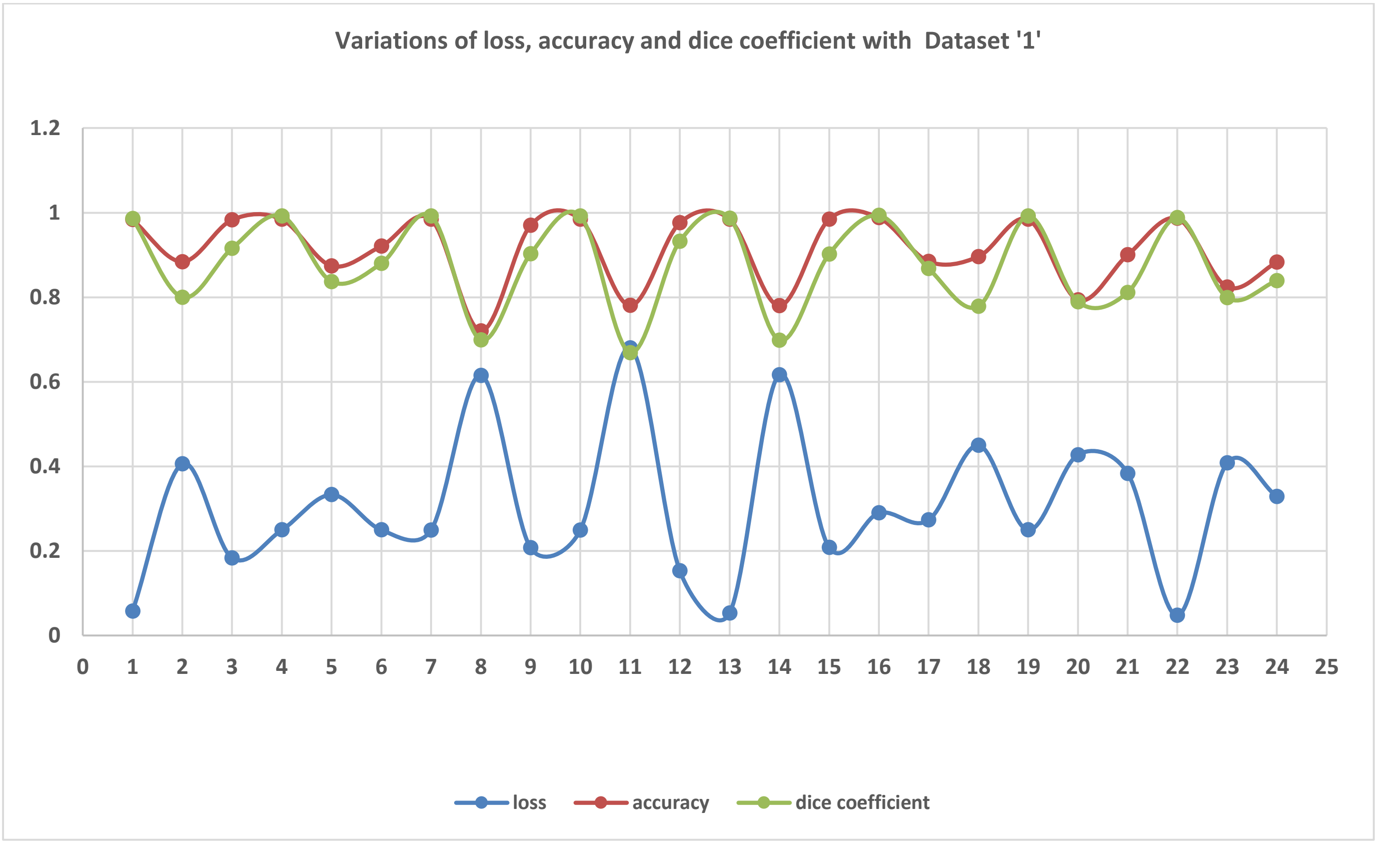


Fig28: Variations of loss, accuracy and dice coefficient with dataset 1

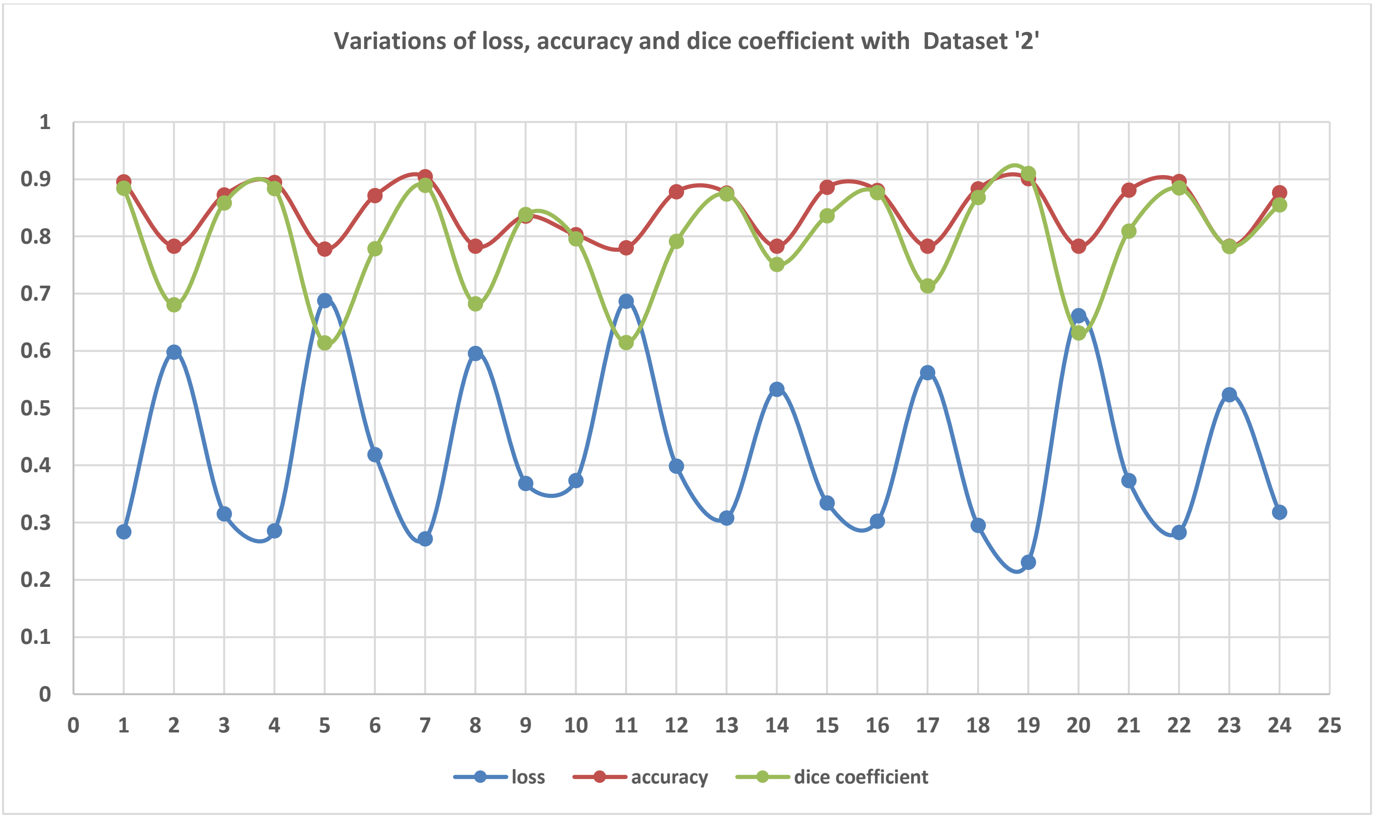


Fig29: Variations of loss, accuracy and dice coefficient with dataset 2

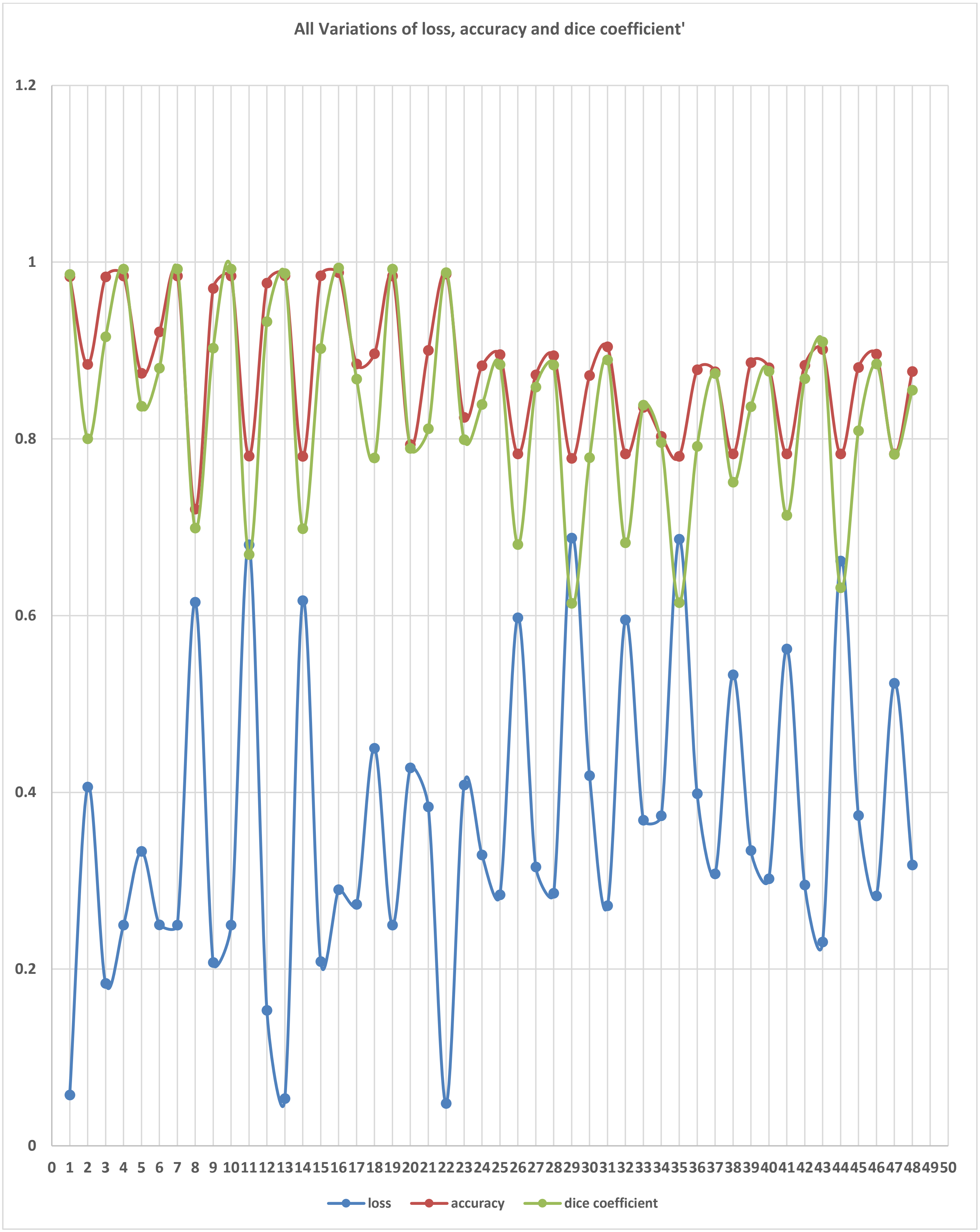


Fig30: Variations of loss, accuracy and dice coefficient

To make a statistical study we compiled the following table:

Column 1	**Max Loss**	**Min Loss**	**Avg Loss**	**Max Accuracy**	**Min Accuracy**	**Avg Accuracy**	**Max Dice Coefficient**	**Min Dice Coefficient**	**Avg Dice Coefficient**
Overall	0.6877	0.0487	0.36206	0.9881	0.7204	0.8797	0.9934	0.6138	0.8364
relu	0.3734	0.0478	0.2366	0.9881	0.8029	0.9331	0.9934	0.7957	0.9325
sigmoid	0.6877	0.2732	0.5379	0.8846	0.7204	0.7998	0.8677	0.6138	0.7268
tanh	0.4498	0.15324	0.3117	0.9846	0.835	0.9061	0.9325	0.7784	0.8498
dropout 0.5	0.6617	0.0535	0.3555	0.9846	0.7204	0.8806	0.9921	0.6315	0.8383
dropout 0.25	0.6877	0.0487	0.3686	0.9881	0.7778	0.8787	0.9934	0.6138	0.8384
kernel 3x3	0.6877	0.0535	0.3456	0.9881	0.7778	0.8889	0.9934	0.6138	0.844
kernel 5x5	0.6865	0.0478	0.3785	0.9864	0.7204	0.8704	0.9921	0.6146	0.8287
model architecture #	0.6877	0.0576	0.3716	0.9843	0.7204	0.8801	0.9921	0.6138	0.8295
model architecture *	0.6617	0.0478	0.3526	0.9881	0.7801	0.8792	0.9934	0.6315	0.8432
dataset 1	0.6801	0.0478	0.3072	0.9881	0.7204	0.914	0.9934	0.6691	0.8768
dataset2	0.6877	0.2306	0.4189	0.9041	0.7778	0.8454	0.9097	0.6138	0.7959

Table5: Statistical analysis of UNET over various parameters

From the above analysis we observe the following key points:

1) The model variation with highest dice coefficient (0.9934) and accuracy (0.9881) values does not have the lowest loss. The loss for this model is relatively high (~0.29).

2) The model variation with lowest loss value (0.0487) has a high accuracy and dice coefficient thereby making it most efficient. Thus, for our next experiment, to generate predictions, we will make use of this architecture only.

3) The average loss, dice and accuracy values for all the variations is satisfactory which suggests that UNET is efficient for image segmentation application.

4) Relu gives maximum accuracy among all the activation functions. Accuracy value for tanh is comparable.

5) Relu clearly dominates other activation functions in terms of dice coefficient and loss values. Moreover, the predictions generated are the most clear. The predictions generated corresponding to sigmoid activation function are very blur and unsatisfactory, while predictions corresponding to tanh activation function are greyish and hence, aren't very clear. However, they are better than those corresponding to sigmoid.

6) Sigmoid having high loss and less accuracy as well as dice values for most cases, is not a suitable activation function for our application.

7) Dropout values alone do not affect the loss, dice and accuracy values (performance) significantly.

8) Kernel size alone does not affect the loss, dice and accuracy values (performance) significantly.

9) We don't observe any significant differences in the performance among the two model architectures. However, * gives better peak results as well as average results for loss and dice parameters making it a slightly better architecture than #.

10) Loss, dice and accuracy values have a significant variation among two datasets which suggests that performance of the model clearly depends on the application it is being used for. However, since the overall performance of the UNET on both the datasets is satisfactory which suggests that UNET gives a satisfactory performance for segmentation of medical data.

11) When using the kernel size 5*5 we observe that training the UNET takes relatively longer time especially with * than the case when we use kernel size 3*3.

Experiment 2:

In the second experiment, we mainly used the best model architecture that we found in study1 to segment tumors from brain MR Scans and labels for the Drosophila images. For getting tumor segmentations, we used the data of 30 patients for training the model, 30 other patients for validation an next 30 patients for testing. For getting drosophila predictios we used 30 images for training, 15 for validation and 15 images for testing the model.

We trained the model on both the datasets for 20 epochs each, with 200 steps per epoch for testing and validation. To find out of sample performance results we kept number of steps to be 50.

Following are the results we obtained from after the study.

Column1	**Brain MR Scans**	**Drosophila Membrane**
no. of epochs	20	20
steps per epoch	200	200
validation steps	200	200
training loss	0.0166	0.1001
training accuracy	0.9961	0.9541
training dice	0.9966	0.9497
validation loss	0.0174	0.1057
validation accuracy	0.9957	0.9469
validation dice	0.9952	0.9343
test loss	0.0336	0.1109
test accuracy	0.9916	0.9315
test dice	0.9909	0.9287

Table6: Final Results

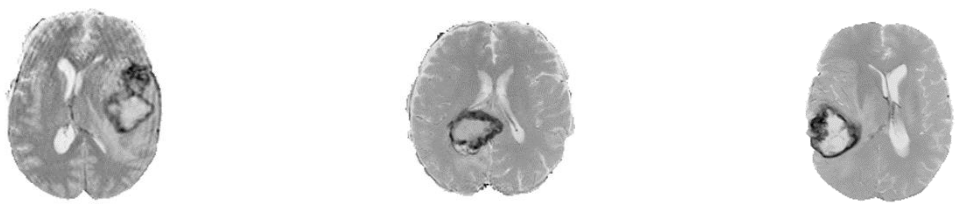


Fig31: Test Images Brain MR Scans (3).

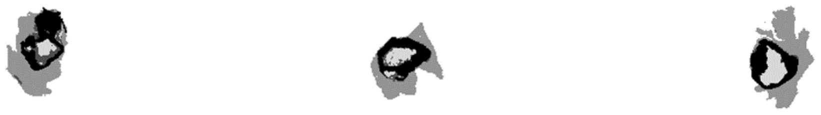


Fig32: Test Labels Brain MR Scans (3)

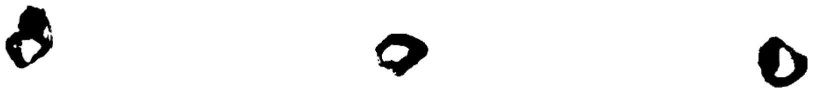


Fig33: Test Images Predictions MR Scans (3)

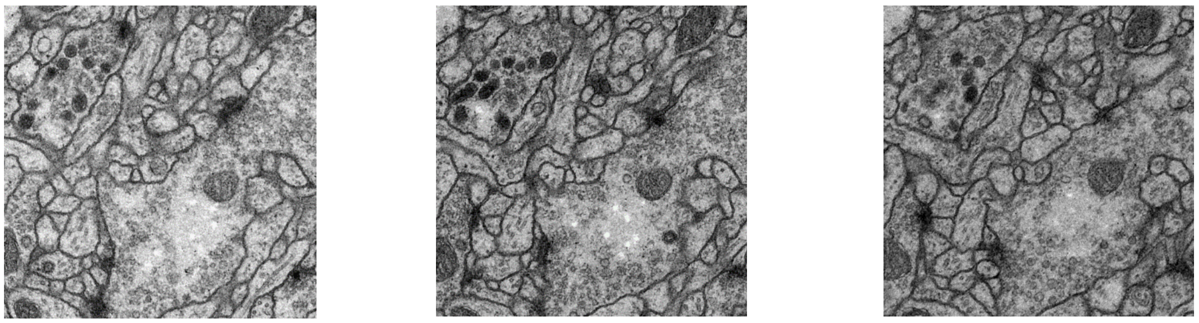


Fig34: Test Images Drosophila EM (3).

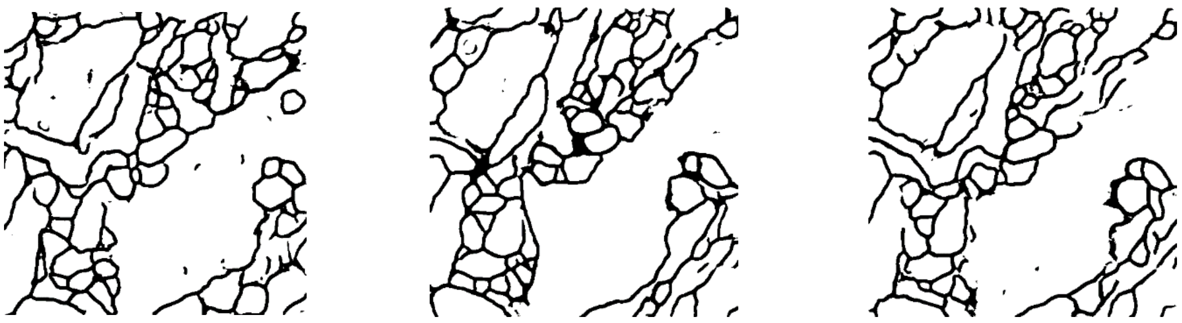


Fig35: Test Predictions Drosophila EM (3).

Conclusion and future work

In this project we implemented UNET, A deep learning based architecture for segmentation of tumors from brain MR Scans and for generated segmented labels of serial section Transmission Electron Microscopy (ssTEM) images of Drosophila first instar larva ventral nerve cord (VNC), according to the isbi 2012 challenge. We then took into consideration different variations in parameters like number of convolutions, convolution window size, activation functions, etc. and extensively compared the results over the two datasets. The details of the analysis were discussed in detailed. The best combination of parameters was then used to generate the predictions over the two datasets. We found that our model performs significantly better than all the state of the art models yielding over 99% dice for all training, validation and testing data for brain MR scans while it gives over 92% dice results for drosophila data. Other findings and details of statistical analysis were discussed in the results and discussion section.

We finally conclude that UNET is one of the best suited model for medical image segmentation, especially for segmentation of Brain tumor from brain MR scans. The results which we obtain are better than most state-of-the-art-models.

In future, we could analyze on other variations of parameters and optimizers to yield even better results. Further, due to increased depth of UNET, the computation of the segmented result takes a significantly long time. We could work on developing a model whose depth is lesser and the predictions are more efficient based on our implementation.

References

[1] S. Pereira, A. Pinto, V. Alves and C. Silva, "Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images", IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1240-1251, 2016. Available: 10.1109/tmi.2016.2538465.

[2] Brain Tumor Segmentation based on 3D U-net with Multi-class Focal Loss JieChang, Minquan Ye*, Xiaoci Zhang, Daobin Huang, Peipei Wang, Chuanwen Yao.

[3] U-Net: Convolutional Networks for Biomedical Image Segmentation Olaf Ronneberger, Philipp Fischer, and Thomas Brox.

[4] B. Ait Skourt, A. El Hassani and A. Majda, "Lung CT Image Segmentation Using Deep Neural Networks", Procedia Computer Science, vol. 127, pp. 109-113, 2018. Available: 10.1016/j.procs.2018.01.104.

[5]"Glioblastoma Multiforme - informations : Je-Cherche.info", Je-cherche.info, 2019. [Online]. Available: http://www.je-cherche.info/-Glioblastoma+Multiforme. [Accessed: 01- May- 2019].

[6] F. Suzanne Falck, "Brain tumor: Types, symptoms, and diagnosis", Medical News Today, 2019. [Online]. Available: https://www.medicalnewstoday.com/articles/315625.php. [Accessed: 01- May- 2019].
[7] "Tumor Types - National Brain Tumor Society", National Brain Tumor Society, 2019. [Online]. Available: https://braintumor.org/brain-tumor-information/understanding- brain-tumors/tumor-types/. [Accessed: 01- May- 2019].

[8] S. Kieffer, "Types of Brain Tumors – Glioma, Meningioma, Pituitary Tumors, Metastatic, Skull Base Tumors | Johns Hopkins Medicine", Hopkinsmedicine.org, 2019. [Online]. Available: https://www.hopkinsmedicine.org/neurology_neurosurgery/centers_clinics/brain_tumor/about-brain-tumors/types.html. [Accessed: 01- May- 2019].

[9] T. Wu, A. Manogaran, J. Beauchamp and G. Waring, "Drosophila vitelline membrane assembly: A critical role for an evolutionarily conserved cysteine in the “VM domain” of sV23", Developmental Biology, vol. 347, no. 2, pp. 360-368, 2010. Available: 10.1016/j.ydbio.2010.08.037.

[10] A. Wulandari, R. Sigit and M. M. Bachtiar, "Brain Tumor Segmentation to Calculate Percentage Tumor Using MRI," 2018 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC), Bali, Indonesia, 2018, pp. 292-296.

[11] V. Zeljkovic et al., "Automatic brain tumor detection and segmentation in MR images," 2014

Pan American Health Care Exchanges (PAHCE), Brasilia, 2014, pp. 1-1.

[12] Muhammad Ali Qadar and Yan Zhaowen, "Brain tumors segmentation: A comparative analysis", National Natural Science Foundation of China (NNSFC) under grant 61271044

[13] S.Sivasundari, "Review of MRI Image Classification Techniques", International Journal of research studies in Computer Science and Engineering (IJRSCSE)

[14] Abdulrahman Al-Janobi, "Performance evaluation of cross-diagonal texture matrix method of texture analysis", Pattern Recognition 34 (2001) 171-180

[15] Dong-Chen He and Li Wang, "Simplified Texture Spectrum for Texture Analysis", August 2010, Volume 7, No.8 (Serial No.69) Journal of Communication and Computer, ISSN 1548-7709, USA

[16] Kharrat A, Benamrane N, Messaoud M, Abid M. "Detection of brain tumor in medical images", 3rd IEEE international conference on signals, circuits and systems; 2009. p. 1–6

[17] Mortazavi D, Kouzani AZ, Soltanian-Zadeh H. "Segmentation of multiple sclerosis lesions in MR images: a review", Neuroradiology 2011;54:299–320.

[18] Kabir Y, Dojat M, Scherrer B, Forbes F, Garbay C. "Multimodal MRI segmentation of ischemic stroke lesions", Proceedings of the 29th annual international conference of IEEE-EMBS; 2007. p. 1595–8.

[19] R.M. Haralick, "Statistical and structural approaches to texture", Proc. IEEE 67 (1979) 786-804

[20] D.C. He, L. Wang, J. Gilbert, "Texture discrimination based on an optimal utilization of texture features", Pattern Recognition 21 (1988) 141-146

[21] "About the 2D EM segmentation challenge | ISBI Challenge: Segmentation of neuronal structures in EM stacks", *Brainiac2.mit.edu*, 2019. [Online]. Available: http://brainiac2.mit.edu/isbi_challenge. [Accessed: 01- May- 2019].

[22] N. Singh, S. Das and A. Veeramuthu, "An efficient combined approach for medical brain tumour segmentation," *2017 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, 2017, pp. 1325-1329.

[23**]** C. Suloway et al., "Automated molecular microscopy: The new Leginon system", *Journal of Structural Biology*, vol. 151, no. 1, pp. 41-60, 2005. Available: 10.1016/j.jsb.2005.03.010.

[24] "BRATS - SICAS Medical Image Repository", *Smir.ch*, 2019. [Online]. Available: https://www.smir.ch/BRATS/Start2015. [Accessed: 01- May- 2019]