A PROJECT REPORT

on

SEMANTIC SEGMENTATION OF BIOMEDICAL IMAGES USING U-NET

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE AND ENGINEERING

BY

Anasuya Dasgupta 1605339

UNDER THE GUIDANCE OF
PROF. Himanshu Das

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA - 751024
March 2020

A PROJECT REPORT

on

"SEMANTIC SEGMENTATION OF BIOMEDICAL IMAGES USING U-NET"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN COMPUTER SCIENCE AND ENGINEERING

BY

Anasuya Dasgupta 1605339

UNDER THE GUIDANCE OF
PROF. Himanshu Das

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAE, ODISHA -751024

March 2020

KIIT Deemed to be University
School of Computer Engineering
Bhubaneswar, ODISHA –751024

CERTIFICATE

This is certify that the project entitled

"Semantic Segmentation of Biomedical Images Using U-Net"

submitted by

ANASUYA DASGUPTA 1605339

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2019-2020, under our guidance.

Date: 30 / 03 / 2020

(Prof. Co-Guide Name)	**(Prof. Himanshu Das)**
Project Co-Guide	**Project Guide**

Acknowledgements

We are profoundly grateful to **Prof. Himanshu Das** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

Anasuya Dasgupta

ABSTRACT

The project deals with segmentation of nucleus from image cells using deep learning techniques. The data set is credited to Kaggle and Booz Allen Hamilton. Masks for few images were provided in the data set but were few. So image augmentation techniques were used to increase the number of images provided in the dataset. For this general augmentation techniques (flip, rotation, scaling) as well as Deep Learning techniques such as Generative Adversarial Network(GAN) were used. Further all the images are first re-sized before training the model. Finally a fully convolutional neural network architecture known as U-Net , credited for its used in biomedical image segmentation is used to segment out the nuclei from the cells. Generalization of this technique for use in everyday life can greatly speed up cures for diseases like cancer, Alzheimer's etc.

Keywords: deep learning, nucleus detection, semantic segmentation, U-Net.

Contents

1 Introduction **2**
1.1 Image Segmentation 2
1.2 Motivation of the project 2
1.3 About the project 2

2 Literature Survey **4**

3 Software Requirements Specification **5**
3.1 Tools used 5

4 System Testing **6**

5 Project Planning **7**

6 Implementation **8**

7 Screenshots of Project **12**
7.1 SECTION NAME 12

8 Conclusion and Future Scope **14**
8.1 Conclusion 14
8.2 Future Scope 14

References **15**

List of Figures

1.1 **Example of an image provided** 3
1.2 **A mask of the above image** 3

6.1 **Architecture of U-Net** 11

7.1 **Variable explorer** 12
7.2 **Program snippet** 13
7.3 **Snippet showing the func. to convert the pixels to numpy array** 13

Chapter 1

Introduction

1.1 Image Segmentation

In the field of computer vision, Image Segmentation is a method of splitting a digital image into multiple segments or set of pixels. The main work of the segmentation is to simplify and modify the representation of a digital image into layers of pixels that is more meaningful and can be easily analyzed by the machine. The final product of the image is a set of segments that collectively cover all the pixels of the image. There are basically 2 types of these segmentation: Instance based Segmentation and Semantic Segmentation.

Here in this project I have worked on semantic segmentation of biomedical images. Semantic segmentation is the task of classifying each and very pixel in an image into a class from objects within an image. In case of instance based, within each class all members are classified independently.

1.2 Motivation of the project

The main motivation behind this project is provide intelligent technological ways to speed up the treatment of diseases. As the population of the world is increasing exponentially over the years, it is becoming extremely difficult to manage and treat them. As the 21st century ushered in, we saw vast technological developments in each and every fields. Biomedical field being one of the most crucial aspect, enjoyed some of the most ground breaking developments due to intervention of technology. From bio-med signal analysis to therapeutic treatments, all are now AI based with little or no human intervention required.

1.3 About the project

Now coming to the topic of the project, nucleus detection is a necessary prerequisite for cellular morphology computation in microscopy and digital pathology image analysis. It can provide both quantitative and qualitative information for better understanding of the biological system or disease progression.[1] Conventional learning methods rely heavily on appropriate data representations, which often need sophisticated expert physicians and people with domain knowledge, to achieve desired accuracies.[7]

With the increase in computational resources in 21st century, deep learning methods are used to solve these problems with ease. Deep neural networks (DNNs) have powered many aspects in computer vision and attracted considerable focus in biomedical imaging and computing.[2] Instead of totally relying on non-trivial image representation methods, DNNs directly deal with raw image data and automatically learns the representations for different tasks. In this project, I have used a fully convolutional network known as U-Net for the segmentation of the nuclei masks from the cells. Also for the data augmentation purpose I have used Generative Adversarial Network for synthetically generating more images for the training purpose.[3]

The model for this semantic segmentation worked and gave an accuracy of 9.5% in locating and segmenting out the musk of the nucleus from the cells.

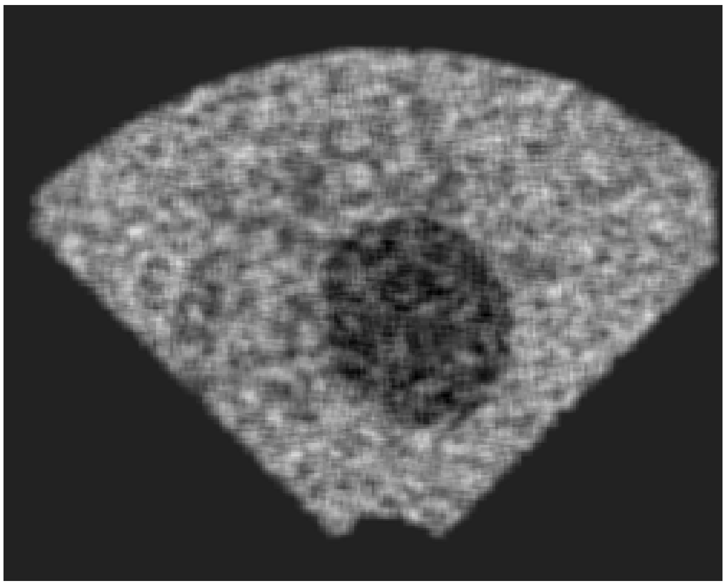


Figure 1.1: Example of an image provided

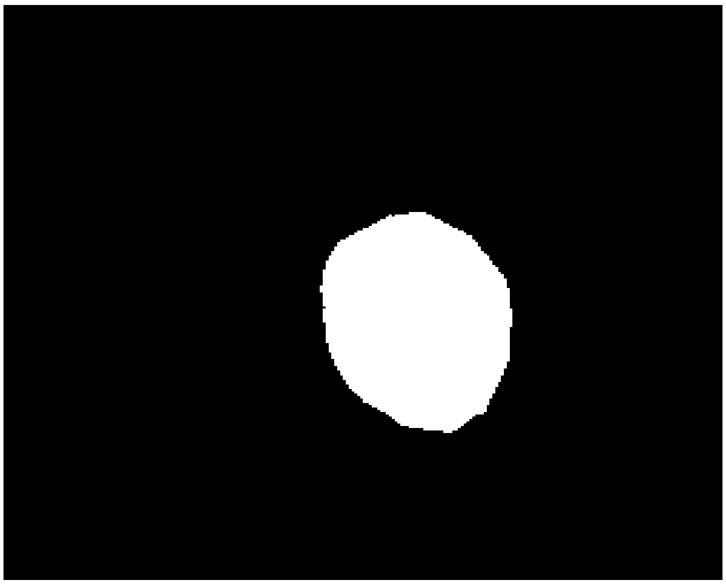


Figure 1.2: A mask of the above image

Chapter 2

Literature Survey

a. Image segmentation in computer vision plays a quintessential role in multi-modality imaging, specially in mixture of structural images obtained by CT scans, MRI reports with functional images collected by optical technologies which are available today, or other novel imaging technologies. For imaging of the brain tissue cells for detection of diseases, Nuclear magnetic resonance produces a crystal clear and high resolution images. Crucial tissues such as cerebrospinal fluidis hard to differentiate due to blurry boundaries in brain and cant be identified using cross sectional images which makes it difficult for the doctors. Which is why a convolution neural network proposed by LeCun et al. [4] is a deep supervised learning method which has been used for the segmentation purpose. In the paper, image enhancement, operators, and morphometry methods has been used by the author to extract the maps of different tissues based on 5 MRI head image datasets. That being done, a conv-net is used to automate the segmentation. Parallel processing is further used to speed up the training time. The results give an accuracy metric of 0.9 Jaccard index.

b. . Earlier systems were built on traditional methods such as edge detection filters and mathematical methods. Then, machine learning approaches extracting hand-crafted features have became a dominant technique for a long period. In the recent years deep learning shoes promising ability for image segmentation. Others like . Shen et al. in [5] have proposed various kinds of medical image analysis but sadly did put little emphasis on technical aspects of the medical image segmentation. Some of the most high performing models for segmentation include: 2.5D CNN, 3D CNN, Cascaded CNN, 2d and 3D U-Net, V-Net, Contextual LSTMs and GRUs.[8] Thus the paper focuses on a critical appraisal of popular methods that has been already used in the field of biomedical image segmentation and highlighted their advantages over the ancestors. Along with all these, the main challenges faced during these process like over-fitting, huge training time, vanishing gradients is also mentioned with prospective solutions to handle them.[6] The paper summarized which model/ architecture gives best result for a specific organ's segmentation as well as its modality.

Chapter 3

Software Requirements Specification

3.1 Tools used

a. Numpy: It's a library for adding support for huge, multi-dimensional arrays and matrix, along with a vast collection of various other mathematical functions.

b. Matplotlib: It is a library function in Python programming language used extensively for the purpose of plotting. It's also a numerical mathematics extension of NumPy.

c. Pandas: It is a free software based library written in python for for data manipulation and analysis.

d. Keras: It is an open-source library mostly for working with neural networks. It is capable of running on top of TensorFlow.

e. Tensorflow: It is a free and open-source software library for dataflow and differentiable programming across arrays of tasks.

f. scikit-learn : It is a free software mostly for machine learning for the Python programming language. It hosts several features includin classification, regression and clustering algorithms.

g. scikit-image: It is a library similar to scikit leran, only it focuses on solving image processing problems.

h. RLE: Run-length encoding is a lossless data compression method in which strands of data are stored as a single data value, rather than as in original run.

Chapter 4

System Testing

After training the model, the test dataset separated out from the total dataset and it gives an accuracy of 90.5% when tested with the provided images. However the accuracy of the test set drops to about 85% when tested with the synthetic ones are included in the test set.

Chapter 5

Project Planning

The main motive of the project is to help the doctors and physicians with a project where artificial intelligence can be used in the medical field. Here it focuses on finding the nucleus from cells automatically. If this kind of ideas can be successfully implemented, then detection of diseases can speed up.

1. First the dataset was downloaded and preprocessed which includes resizing them.

2. Directories are maintained to put all the images and their pre-segmented masks in order

3. Data Augmentation is used to synthetically increase the the number of images in the dataset as the provided data was not enough to train a model

4. Deep learning approach is considered over machine learning ones as the manual feature extraction step can be avoided.

5. The preprocessed images along with the augmented ones are trained using an U-Net model for the semantic segmentation purpose. U-Net is preffered over other architectures as it outperforms most others as seen from literature.

6. The test set is used to check the performance of the model.

Chapter 6

Implementation

First after downloading the dataset, the provided images are maintained inside specific directories for easy retrieval of the data. After that in the progaram all the important header files that are required in the program are implemented. Further due to data insufficiency synthetically images are produced using data augmentation techniques which includes using GANs.

Further the data is preprocessed and resized for training the model. The testing and training path are set and both the test and train set images are converted into numpy arrays so that it can be identified by the machine. The data is compressed using run length encoding technique.

Finally the U-Net architecture is coded and model is trained for 15 epochs with dice co-efficient as segmentation accuracy metric. Further tested with the test dataset.

```
import os
import random
import sys
import warnings
import numpy as np
import pandas as pd
from itertools import chain
from skimage.io import imread, imshow, imread_collection, concatenate_images
from skimage.transform import resize
import skimage.morphology
from skimage import io
io.use_plugin('matplotlib', 'imread')
from keras.utils import Progbar
from keras.models import Model, load_model
from keras.layers import Input
from keras.layers.core import Dropout, Lambda
from keras.layers.convolutional import Conv2D, Conv2DTranspose,Convolution2D
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import concatenate
from keras import backend as K

warnings.filterwarnings('ignore', category=UserWarning, module='skimage')

seed = 42
random.seed = seed
np.random.seed = seed
smooth = 1.
epochs = 50

TRAIN_PATH = 'D:/datasets/Segment/train/'
TEST_PATH = 'D:/datasets/Segment/test/'

train_ids = next(os.walk(TRAIN_PATH))[1]
```

```
test_ids = next(os.walk(TEST_PATH))[1]


def read_train_data(IMG_WIDTH=256,IMG_HEIGHT=256,IMG_CHANNELS=3):
    X_train = np.zeros((len(train_ids), IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS), dtype=np.uint8)
    Y_train = np.zeros((len(train_ids), IMG_HEIGHT, IMG_WIDTH, 1), dtype=np.bool)
    print('Getting and resizing train images and masks ... ')
    sys.stdout.flush()
    if os.path.isfile("train_img.npy") and os.path.isfile("train_mask.npy"):
        print("Train file loaded from memory")
        X_train = np.load("train_img.npy")
        Y_train = np.load("train_mask.npy")
        return X_train,Y_train
    a = Progbar(len(train_ids))
    for n, id_ in enumerate(train_ids):
        path = TRAIN_PATH + id_
        img = imread(path + '/images/' + id_ + '.png')[:,:,:IMG_CHANNELS]
        img = resize(img, (IMG_HEIGHT, IMG_WIDTH), mode='constant', preserve_range=True)
        X_train[n] = img
        mask = np.zeros((IMG_HEIGHT, IMG_WIDTH, 1), dtype=np.bool)
        for mask_file in next(os.walk(path + '/masks/'))[2]:
            mask_ = imread(path + '/masks/' + mask_file)
            mask_ = np.expand_dims(resize(mask_, (IMG_HEIGHT, IMG_WIDTH), mode='constant',
                                            preserve_range=True), axis=-1)
            mask = np.maximum(mask, mask_)
        Y_train[n] = mask
        a.update(n)
    np.save("train_img",X_train)
    np.save("train_mask",Y_train)
    return X_train,Y_train


def read_test_data(IMG_WIDTH=256,IMG_HEIGHT=256,IMG_CHANNELS=3):
    X_test = np.zeros((len(test_ids), IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS), dtype=np.uint8)
    sizes_test = []
    print('\nGetting and resizing test images ... ')
    sys.stdout.flush()
    if os.path.isfile("test_img.npy") and os.path.isfile("test_size.npy"):
        print("Test file loaded from memory")
        X_test = np.load("test_img.npy")
        sizes_test = np.load("test_size.npy")
        return X_test,sizes_test
    b = Progbar(len(test_ids))
    for n, id_ in enumerate(test_ids):
        path = TEST_PATH + id_
        img = imread(path + '/images/' + id_ + '.png')[:,:,:IMG_CHANNELS]
        sizes_test.append([img.shape[0], img.shape[1]])
        img = resize(img, (IMG_HEIGHT, IMG_WIDTH), mode='constant', preserve_range=True)
        X_test[n] = img
        b.update(n)
    np.save("test_img",X_test)
    np.save("test_size",sizes_test)
    return X_test,sizes_test


def rle_encoding(x):
    dots = np.where(x.T.flatten() == 1)[0]
    run_lengths = []
    prev = -2
    for b in dots:
        if (b>prev+1): run_lengths.extend((b + 1, 0))
        run_lengths[-1] += 1
        prev = b
    return run_lengths

def prob_to_rles(x, cutoff=0.5):
    lab_img = label(x > cutoff)
    for i in range(1, lab_img.max() + 1):
        yield rle_encoding(lab_img == i)
```

```python


def mask_to_rle(preds_test_upsampled):
    new_test_ids = []
    rles = []
    for n, id_ in enumerate(test_ids):
        rle = list(prob_to_rles(preds_test_upsampled[n]))
        rles.extend(rle)
        new_test_ids.extend([id_] * len(rle))
    return new_test_ids, rles


def dice_coef(y_true, y_pred):
    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)

# Loss funtion
def dice_coef_loss(y_true, y_pred):
    return -dice_coef(y_true, y_pred)


def get_unet(IMG_WIDTH=256,IMG_HEIGHT=256,IMG_CHANNELS=3):
    inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
    s = Lambda(lambda x: x / 255) (inputs)
    c1 = Conv2D(16, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (s)
    c1 = Dropout(0.1) (c1)
    c1 = Conv2D(16, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c1)
    p1 = MaxPooling2D((2, 2)) (c1)
    c2 = Conv2D(32, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (p1)
    c2 = Dropout(0.1) (c2)
    c2 = Conv2D(32, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c2)
    p2 = MaxPooling2D((2, 2)) (c2)

    c3 = Conv2D(64, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (p2)
    c3 = Dropout(0.2) (c3)
    c3 = Conv2D(64, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c3)
    p3 = MaxPooling2D((2, 2)) (c3)

    c4 = Conv2D(128, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (p3)
    c4 = Dropout(0.2) (c4)
    c4 = Conv2D(128, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c4)
    p4 = MaxPooling2D(pool_size=(2, 2)) (c4)

    c5 = Conv2D(256, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (p4)
    c5 = Dropout(0.3) (c5)
    c5 = Conv2D(256, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c5)

    u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same') (c5)
    u6 = concatenate([u6, c4])
    c6 = Conv2D(128, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (u6)
    c6 = Dropout(0.2) (c6)
    c6 = Conv2D(128, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c6)

    u7 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same') (c6)
    u7 = concatenate([u7, c3])
    c7 = Conv2D(64, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (u7)
    c7 = Dropout(0.2) (c7)
    c7 = Conv2D(64, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c7)

    u8 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same') (c7)
    u8 = concatenate([u8, c2])
    c8 = Conv2D(32, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (u8)
    c8 = Dropout(0.1) (c8)
    c8 = Conv2D(32, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c8)

    u9 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same') (c8)
    u9 = concatenate([u9, c1], axis=3)
    c9 = Conv2D(16, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (u9)
    c9 = Dropout(0.1) (c9)
    c9 = Conv2D(16, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c9)

    outputs = Conv2D(1, (1, 1), activation='sigmoid') (c9)

    model = Model(inputs=[inputs], outputs=[outputs])
    model.compile(optimizer='adam',loss='binary_crossentropy', metrics=[dice_coef])
    return model



train_img,train_mask = read_train_data()

# get test_data
test_img,test_img_sizes = read_test_data()

u_net = get_unet()

print("\nTraining...")
u_net.fit(train_img,train_mask,batch_size=12,epochs=epochs)
```

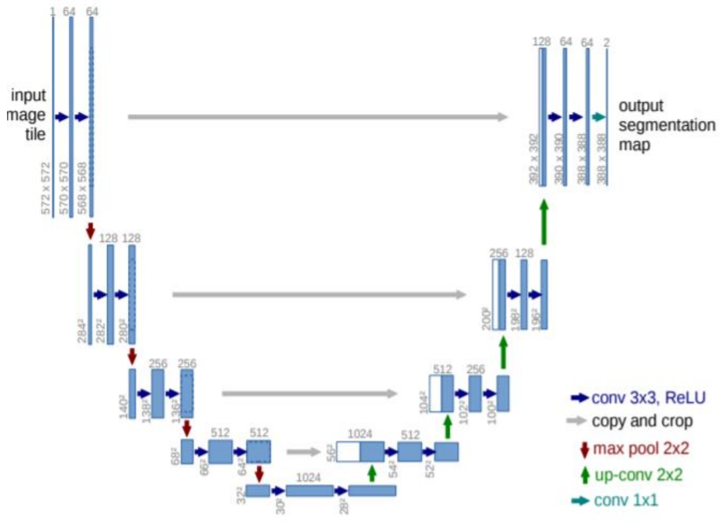
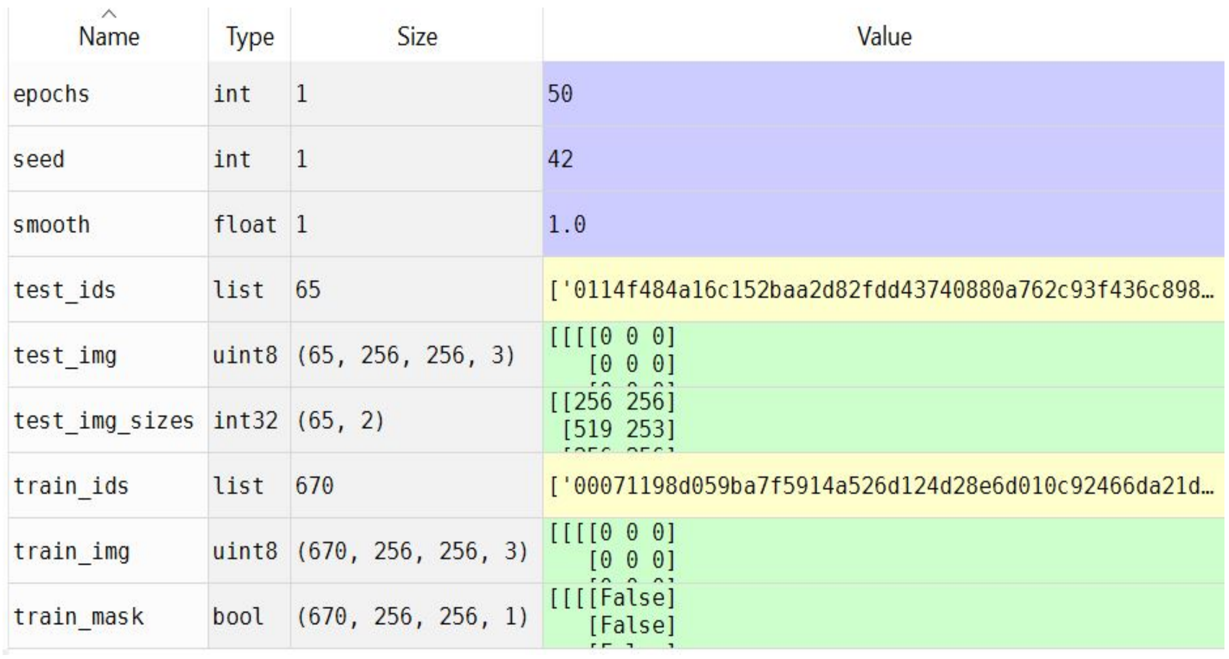


Figure 6.1: Architecture of U-Net

Chapter 7

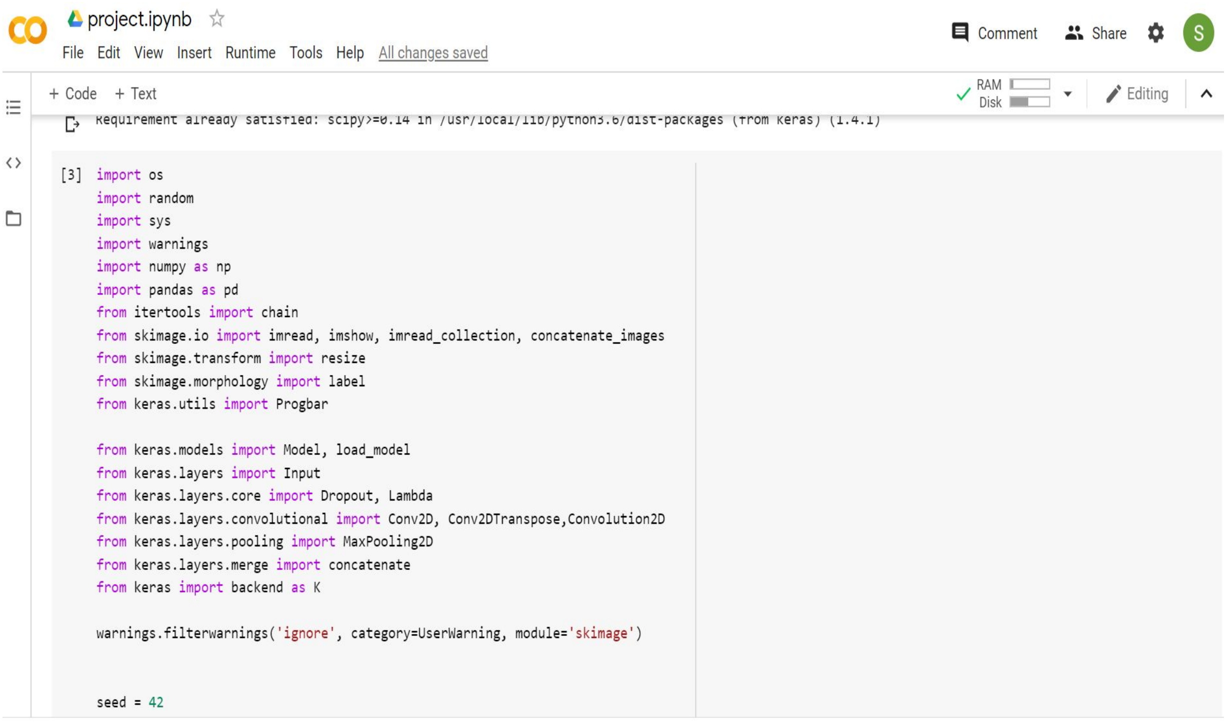
Screenshots of Project

7.1 SECTION NAME



Name	Type	Size	Value
epochs	int	1	50
seed	int	1	42
smooth	float	1	1.0
test_ids	list	65	['0114f484a16c152baa2d82fdd43740880a762c93f436c898…
test_img	uint8	(65, 256, 256, 3)	[[[[0 0 0] [0 0 0]
test_img_sizes	int32	(65, 2)	[[256 256] [519 253]
train_ids	list	670	['00071198d059ba7f5914a526d124d28e6d010c92466da21d…
train_img	uint8	(670, 256, 256, 3)	[[[[0 0 0] [0 0 0]
train_mask	bool	(670, 256, 256, 1)	[[[[False] [False]

Figure 7.1: Variable explorer



```
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.6/dist-packages (from keras) (1.4.1)
```

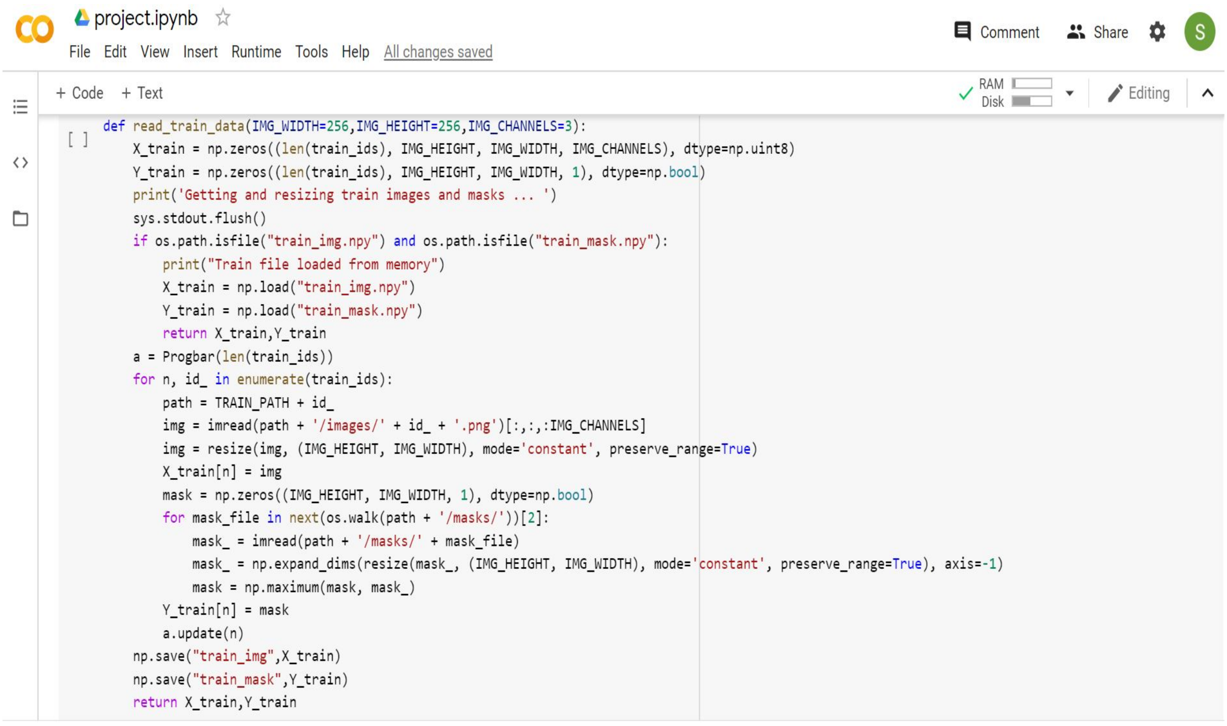
```
[3] import os
    import random
    import sys
    import warnings
    import numpy as np
    import pandas as pd
    from itertools import chain
    from skimage.io import imread, imshow, imread_collection, concatenate_images
    from skimage.transform import resize
    from skimage.morphology import label
    from keras.utils import Progbar

    from keras.models import Model, load_model
    from keras.layers import Input
    from keras.layers.core import Dropout, Lambda
    from keras.layers.convolutional import Conv2D, Conv2DTranspose,Convolution2D
    from keras.layers.pooling import MaxPooling2D
    from keras.layers.merge import concatenate
    from keras import backend as K

    warnings.filterwarnings('ignore', category=UserWarning, module='skimage')


    seed = 42
```

Figure 7.2: Program snippet



```
[ ] def read_train_data(IMG_WIDTH=256,IMG_HEIGHT=256,IMG_CHANNELS=3):
        X_train = np.zeros((len(train_ids), IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS), dtype=np.uint8)
        Y_train = np.zeros((len(train_ids), IMG_HEIGHT, IMG_WIDTH, 1), dtype=np.bool)
        print('Getting and resizing train images and masks ... ')
        sys.stdout.flush()
        if os.path.isfile("train_img.npy") and os.path.isfile("train_mask.npy"):
            print("Train file loaded from memory")
            X_train = np.load("train_img.npy")
            Y_train = np.load("train_mask.npy")
            return X_train,Y_train
        a = Progbar(len(train_ids))
        for n, id_ in enumerate(train_ids):
            path = TRAIN_PATH + id_
            img = imread(path + '/images/' + id_ + '.png')[:,:,:IMG_CHANNELS]
            img = resize(img, (IMG_HEIGHT, IMG_WIDTH), mode='constant', preserve_range=True)
            X_train[n] = img
            mask = np.zeros((IMG_HEIGHT, IMG_WIDTH, 1), dtype=np.bool)
            for mask_file in next(os.walk(path + '/masks/'))[2]:
                mask_ = imread(path + '/masks/' + mask_file)
                mask_ = np.expand_dims(resize(mask_, (IMG_HEIGHT, IMG_WIDTH), mode='constant', preserve_range=True), axis=-1)
                mask = np.maximum(mask, mask_)
            Y_train[n] = mask
            a.update(n)
        np.save("train_img",X_train)
        np.save("train_mask",Y_train)
        return X_train,Y_train
```

Figure 7.3: Snippet showing the func. to convert the pixels to numpy array

Chapter 8

Conclusion and Future Scope

8.1 Conclusion

.Thus to conclude, the project was based on detection of epilepsy using EEG data provided by the company Booz Allen Hamilton under the license of kaggle. Thus we can conclude that biomedical segmentation of nucleus from cells can actually fast-track the cure of many diseases. Here in the project U-Net model is used to semantically segment the nucleus musk from the cells. Using this deep learning architecture, a model is trained to automatically segment the nucleus with a dice coefficient of approximately 88% while tested with synthetically produced images using generative network.

The choice for this architecture allows for fast computation. An important quality of the proposed solution is its robustness, even when trained and tested on different sources of data.

8.2 Future Scope

This concept can also be used in real time approach. This is to say that it can be clubbed to the monitor so that discovery and medication of certain diseases like cancer, Alzheimer etc. can be fast-tracked. This model can further be generalized for its application in other biomedical fields. Thus the model could be generalized more and could be made immune to noisy, real life images making the learner even more robust.

References

[1] Gurcan MN, Boucheron LE, Can A, Madabushi A, Rajpoot NM, Yener B. Histopatological image analysis: a review. IEEE Rev Biomed Eng. 2009; 2:147–71.

[2] Greenspan H, van Ginneken B, Summers RM. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. IEEE Trans Med Imaging. 2016; 35(5):1153–9.

[3] Yanhui Guo, Amira S.Ashour. Neutrosophic sets in dermoscopic medical image segmentation, Science Direct 2019, doi.org/10.1016/B978-0-12-818148-5.00011-4

[4] Ciresan, D.C., Gambardella, L.M., Giusti, A., Schmidhuber, J.: Deep neural networks segment neuronal membranes in electron microscopy images. In: NIPS. pp. 2852–2860 (2012)

[5] Bengio, Yoshua; LeCun, Yann; Hinton, Geoffrey (2015). "Deep earning". *Nature*. 521 (7553): 436–444. Bibcode:2015Natur.521..436L. doi:10.1038/nature14539

[6] Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". arXiv:1505.04597

[7] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). Generative Adversarial Networks . Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.

[8] Kong Zhenglun, Li Ting, Luo Junyi, Zu Shengpu (2019) Automatic Tissue Image Segmentation Based on Image Processing and Deep Learning; https://doi.org/10.1155/2019/2912458.

[9] P. Moeskops, M. A. Viergever, A. M. Mendrik, L. S. de Vries, M. J. N. L. Benders, and I. Isgum, “Automatic segmentation of MR brain images with a convolutional neural network,” IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1252– 1261, 2016.

[10] F. Milletari, N. Navab, and S. A. Ahmadi, “V-net: fully convolutional neural networks for volumetric medical image segmentation,” in Proceedings of the 2016 Fourth International Conference on 3D Vision (3DV) IEEE, pp. 565–571, Stanford, CA, USA, October 2016.

[11] http://medium.com

SEMANTIC SEGMENTATION OF BIOMEDICAL IMAGES USIN U-NET

ANASUYA DASGUPTA
1605339

Abstract: The project deals with segmentation of nucleus from image cells using deep learning techniques. Image augmentation is used to replenish the number of images in the datset. A fully convolutional neural network architecture known as U-Net , credited for its used in biomedical image segmentation is used to segment out the nuclei from the cells. Generalization of this technique for use in everyday life can greatly speed up cures for diseases like cancer, Alzheimer's etc.

Individual contribution and findings: It is an individual project. So for the findings part, I had first read articles in Towards data science by Medium . My project guide also helped me a lot with the dataset. Further I read papers from the literature to know more about it and the kinds of work mostly done in this field.

Individual contribution to project report preparation: All the sections of the report have been made by myself. I took help from a lot of papers (mentioned in the reference section) for writing the reference and the introduction section. Also my project guide helped me considerably in case I faced any kind of difficulty.

Individual contribution for project presentation and demonstration: The project will be presented and demonstrated by myself.

Full Signature of Supervisor:	Full signature of the student:
……………………………	……………………………..