Semantic segmentation of peripheral white blood cells using neural networks

Pavel Baykalov
Máster Universitario de Bioinformática y Bioestadística
Machine Learning

Advisor:
Edwin Santiago Alférez Baquero

02-01-2019


Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Semantic segmentation of peripheral white blood cells using neural networks*
Nombre del autor:	*Pavel Baykalov*
Nombre del consultor/a:	*Edwin Santiago Alférez Baquero*
Nombre del PRA:	*Ferran Prados Carrasco*
Fecha de entrega (mm/aaaa):	01/2019
Titulación::	*Máster Universitario en Bioinformática y Bioestadística*
Área del Trabajo Final:	*Machine Learning*
Idioma del trabajo:	***English***
Palabras clave	*U-Net, DeconvNet, SegNet*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

La segmentación semántica es la diferenciación de las partes significativas de una imagen. Y se ha utilizado en muchos campos distintos, como el tráfico o campo de la medicina. Uno de estos usos en el campo médico es el examen de frotis de la sangre. Los glóbulos blancos (WBC, por sus siglas en inglés) son parte del sistema inmunológico y su conteo y determinación a menudo los realizan médicos especialistas para el diagnóstico. La forma y el tamaño del núcleo de los leucocitos pueden determinar el tipo de WBC mediante el examen visual de un experto.

La segmentación semátnica de WBC ya se había propuesto antes, pero no se utilizaron redes neuronales convolucionales. Por lo que, en este proyecto, la segmentación semántica se realizó en un conjunto de datos de acceso libre, que está compuesto por imágenes microscópicas e imágenes de la verdad de fondo segmentadas, de WBC, realizadas por expertos. El conjunto de datos fue filtrado, transformado y aumentado para ser utilizado en una red neuronal artificial. Algunos modelos de segmentación, como U-Net, SegNet y DeconvNet, fueron elegidos, adaptados y entrenados para / con esta información. Después del entrenamiento, se evaluaron los modelos, utilizando diferentes métricas (precisión, coeficientes de similaridad de Jaccard y de Sørensen–Dice), con el mismo conjunto de datos. Tanto para el entrenamiento como para la evaluación de los modelos se empleó Jupyter notebook de la plataforma de Google llamada Colaboratory.

Aunque los tres modelos lograron puntuaciones muy altas en distintas métricas. La arquitectura de U-Net resultó ser el mejor modelo para la segmentación, así como también el más rápido para el proceso de

entrenamiento.

Abstract (in English, 250 words or less):

Semantic segmentation is the differentiation of the meaningful parts on an image. It has been used in many distinct fields, such as traffic or medical areas. One of these uses in the medical field is the blood smear examination. White blood cells (WBC) are part of the immune system and their counting and determination are often performed by medical specialists for diagnosis. The shape and size of the nucleus of leukocytes can determine the type of WBC, by visual examination of an expert.

WBC segmentation had been proposed before, but the convolutional neural network architectures were not tried for this task. Therefore, in this project, the semantic segmentation was performed on free access dataset, which is composed of microscopic images and segmented ground truth images, of WBC, made by experts. The dataset was filtered, transformed and augmented in order to be used in an artificial neural network. Some segmentation models, such as U-Net, SegNet and DeconvNet, were chosen, adapted and trained to/with this data. After training, the models were evaluated, using different metrics (accuracy, Jaccard similarity index and Sørensen–Dice similarity coefficient), with the same dataset. Jupyter notebook from the free Google platform called Colaboratory was used for the training and evaluation of the models.

Although all three models achieved very high scores in distinct metrics. U-Net architecture resulted in being the best model for segmenting, as well as the fastest one for the training process.

Contents

Figures list

1. Introduction

1.1 Context and justification for this Project

White blood cells (WBC) are the main components of immune cells and their counting and identification are usually performed by medical specialists for the purpose of diagnosing infections, leukaemia, inflammations, etc. [1]. However, the examination of the blood smear can be tedious and time-consuming [2], thus automation can help a lot in this task.

Segmentation is an important part of the examination of cell digital morphology and several techniques that require the use of machine learning algorithms that have already been tested on cell blood images [1]. It should be noted that the different methods pursued different goals, so it is difficult to make comparisons between them.

The current boom and popularity of artificial intelligence create an appropriate environment to find the solution to the problem defined above. Several neural network architectures have already been developed and tested in order to segment different images, obtaining distinct results. And in this project, it is intended to use some of them with blood data and compare their results.

The motivation for this project comes from the need to automatically segment the WBC, in order to help in the study of cell morphology and improve a possible clinical diagnosis. The project itself does not aim to improve existing techniques but rather to try new approaches with existing neural network architectures and to observe their segmentation performance, proposing in this way its possible use in the clinical field.

1.2 Project goals

1. To perform the semantic segmentation of the white blood cell images, using some deep neural networks models.Three architectures will be explored: U-net[3], Segnet[4] and DeconvNet [5].

   For the first general objective, it will be necessary:

   i. Transform and adapt the data set to be used within different models, both for training and for validation. In addition to increasing this data using image processing methods in order to have more images to use and to improve the metrics of a subsequent evaluation.
   ii. Load the models on Google's *Colaboratory* platform to run them on it, as well as the data set.

iii. If required, adjust the hyperparameters of the architectures before training. Besides the standardization of the models: the same batch size, same epochs, etc.

iv. Train the models, preparing them for future validation. Its performance will be measured throughout the process and the time it takes to train will be measured.

2. To evaluate the effectiveness of these models.

   In this overall goal, it is only tried to compare the efficiency of the models used through different metrics.

   For this purpose, the specific goals are:

   i. To perform the validation of each model. Determining the accuracy of the trained model on new data.

   ii. To use Jaccard's index or Jaccard similarity coefficient and Sørensen-Dice similarity coefficient, also known as F-score as metrics to evaluate the performance of the models, on the validation images; and present these results in a table.

1.3 Focus and followed method

Due to time constraints and the difficulty of adapting the set of data to the architectures -because of small sized images, there are three kinds of objects to distinguish and at least two of them are close together and overlapping: cytoplasm and nucleus the cells-, the approach that will be followed is to adapt the architectures to the data, trying to preserve the original model structure as much as possible. The code of the architectures and their implementation will be written in Python 3 in Jupyter Notebook of the free Google platform called *Colaboratory* (*Colab*), which also offers GPU in the cloud to run the models faster than if they did use the computer's CPU.

A special strategy will also be used for segmented images or masks: it consists of passing them from the greyscale to the colour scale -red, blue and green (RBG)-, that is, from one dimension to three, corresponding each dimension with the object to be detected by the neural networks.

1.4 Project plan

1.4.1 Tasks

- The search for the topic of the project: given my particular circumstances, I must include this task. In order to explain the time

dedicated and the delay in deliveries of the PECS/reports of the master's thesis. (52 hours)

- A load of data and models on *Colab* platform, and first tests of these models. Free licensing models will be searched on the internet (on blogs and platforms such as Github or stack overflow, mainly) and loaded on Google's *Colab* platform. To be executed later with the intention of testing its effectiveness in the real data. (20 hours)

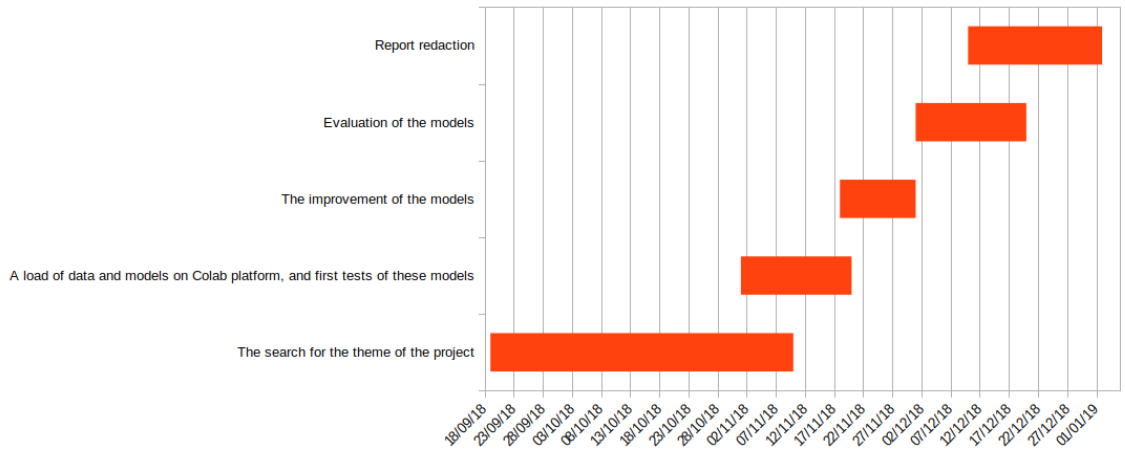
- The improvement of the models. The search for improving the models by adapting them to the data, modifying their hyperparameters and looking for the best possible data organization to allow the most optimal adjustment as possible. The data augmentation is also included, in this point. (12 hours)

- Evaluation of the models. In this task, it will be tried to evaluate the effectiveness of the models employed and to determine the best of the three, based on metrics such as Dice or Jaccard similarity coefficients. (20 hours)

- Report redaction. The explanation of the project, following the protocols and design of the UOC. (22 hours)

1.4.2 Timetable

Due to the particular situation of the author, the performance of tasks overlaps many times, which offers the possibility of going back to obtain a better result or to rethink certain decisions taken that had incorrect assumptions.



1.4.3 Milestones

All the milestones have squared well within the planning, although it must be taken into account that the tasks overlap temporarily.

Milestones	Date
The search for the theme of the project	6-10-2018
A load of data and models on *Colab* platform, and first tests of these models	19-11-2018
The improvement of the models	1-12-2018
Evaluation of the models	20-12-2018
Report redaction	2-01-2019

1.4.4 Risk analysis

- The situation of the author. The author already had a project before starting the school year, that project was going to be carried out in a German company. But given the disagreement between the university and the company, the project was over and the author was forced to look for a new project at the beginning of the master's course, which clearly modified the whole organization of the subject of the final project.

- A shorter and limiting time a priori. The greatest risk and given the author's particular situation is time, which has prevented him from exploring more models and has reduced the range of action. This, in turn, has also modified the way of working as mentioned in the previous sections.

- Problems with the data, some of the images do not have the same size which can cause error when they are supplied to the model.

1.4.5 Costs associated with the project and the final product

Since there is no final product, the costs associated are only related to the project. All software and data used in the project will be free and open; it will be carried out at the author's address, with a personal computer. Thus, all costs are due to the electricity needed.

1.4.6 Legal and ethical implications

The project respects the authorship of the data, the code of the models is not copyrighted and free platforms are used for the execution of the programs.

1.5 Breif summary of the results

It is expected to make and deliver:

- ✓ Project plan
- ✓ The report
- ✓ The available code in Github
- ✓ Virtual presentation
- ✓ Self-evaluation of the project

In addition to PECs or required reports, which will be delivered to the project tutor.

1.6 Breif description of next chapters

In the following chapters, it is briefly explained the **background** for this project: semantic segmentation, neural networks, CNN, U-Net, SegNet and DeconvNet.

It is also included, the **materials and methods**, where it is mentioned the how the models are written and how is the dataset; and where they will be loaded and implemented. Moreover, the metrics which will be employed in evaluation.

Finally the chapters of **results** obtained after the training and evaluating the models, and the **conclusions** given from these results. Then, the **annexes**, **bibliography** and **glossary** finalize the report.

2. Background

2.1 Semantic segmentation

The semantic segmentation of images consists of clustering different groups of the pixels of an image, by tagging each pixel. Thus getting to frame objects within the background or even to distinguish parts of these objects. However, this task of detection and highlighting the object within an image is not an unsupervised process, although some unsupervised method could be used in semantic segmentation in order to refine a segmentation [6]. Therefore, it is a supervised learning and some algorithm were already performed with different level of success, such as: Random Decision Forests, Support Vector Machines, Markov Random Fields, Conditional Random Fields and Neural Networks [6]. Regarding neural networks: Convolution Neural Network (CNN), Deep Neural Network (DNN) and Fully Convolutional Networks (FCN) have been proposed [7 - 9] and many architectures were build using them as layers, such as Segnet, ENet, DeepLab, 2D-LSTM, etc [10].

Different methods of segmentation have been used for distinct purposes such as: for detecting road signs [11], industrial quality inspection [12], detecting tumors [13], detecting medical instruments in operations [14], colon crypts segmentation [15], or land use and land cover classification [16].

2.2 Neural networks and specially CNN

Artificial neural networks are circuits of neurons or nodes, inspired by animal brains, and capable to learn tasks only by showing them the examples of these tasks. Some neural networks are even called as a universal function approximation [17]. The basic unit of the network is a neuron or node, inspired in animal neurons, which receives an input from other neuron or neurons and gives an output to the next neuron, by performing an operation. This could be defined like:

$$Output = f\left(\sum \left(w_i x_i\right) + b\right)$$

Eq. 1

where:
x_i is an input,
w_i is a weight,
b is a bias,
and f is an activation function.

The weights and biases of the neurons can be loaded from the beginning, therefore, loading the already model trained in this way, as long as the architecture of the network is preserved. However, it usual that for the first time training, the random weights are assigned following the uniform distribution. The Xavier initialization, from Xavier Glorot, of weights is used nowadays because it brings a faster convergence [18] of neuronal network models.
The activation function is the mechanism that processes the incoming information; as a biological analogy it is the firing rate of the neuron. It is known several of these functions, the first explored was the sigmoid function, but the most used nowadays is ReLU [19] function which is difined like:

$$f(x) = max(0, x)$$

Eq. 2

where x is the input of the function. Furthermore, other activation functions are being experimented, such as leaky ReLU [20] or maxout [21].

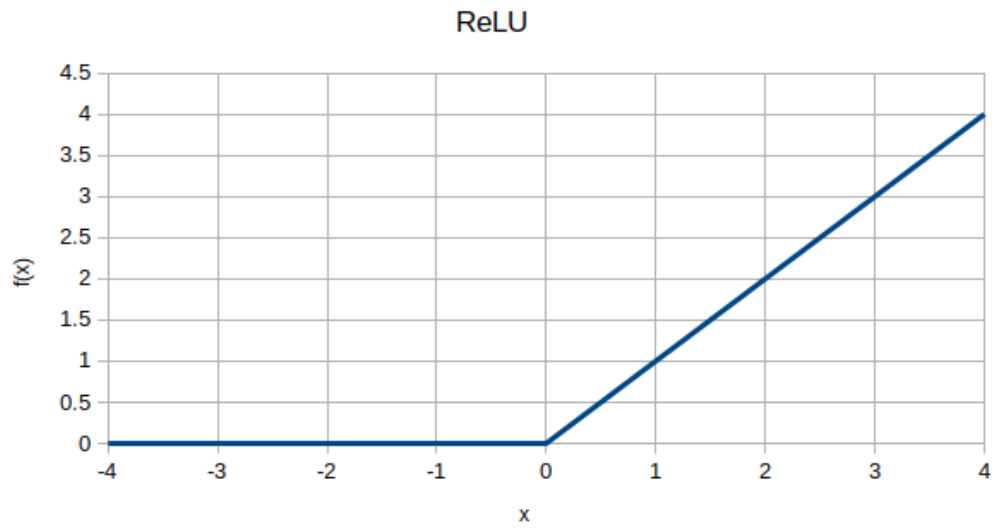


Figure 1. ReLU activation function plot.

These artificial neurons are organized in layers, which usually comprises output, input and hidden layers, and all of them form the artificial neural network.

In feed-forward networks the computation goes from input layer to output one, through the hidden layers. However, with backpropagation it can go reverse in order to update the weight of neurons [22], thus the model can learn.

The backpropagation it is a final step after the loss function and its derivative are applied to the outputs of the network. On the other hand,

the loss function [23] calculates the error of the output data from the input data, and there are often two types of loss functions: cross entropy for classification tasks and mean square error (MSE) for regression tasks. In order to achieve the minimal error, neural networks use an optmization technique called gradient descent [24]. Although for small data, batch gradient descent is totally fine and gets to global minimum [24]; for large dataset, the training is very slow [25], and mini-batch or stochastic gradient descent is used instead. Nevertheless, stochastic gradient descent is usually not very accurate because the global minimum is not guaranteed, thus many optimizers have been developed to achieve the best results -either best local minimum or directly global minimum- in less time [24] than it would take with batch gradient descent.

The overfitting -the model learns the exact training data without generalizing- is very often after training a machine learning algorithm, the same occurs with artificial neural networks. In order to avoid the overfitting problem, regularization such as L1 and L2 is usually applied to the models [26]. However, a simpler way of improve the generalization of the neural networks model is by doing dropout [27], which is randomly drop nodes from the network while training.

2.2.1 CNN

There are different types of neural networks such as RNN, GAN, Autoencoders, etc. Nevertheless, this project focus is on the convolutional neural networks or CNNs, which are the basic blocks of the architectures that are explored working with peripheral blood cell image data [28].

CNN is a special type of multilayer neural network [29], which uses filters to identify features in an image. These features are always represented in a feature map generated by the convolution. CNN mainly have four different layers : convolution, ReLU, pooling and fully connected layers.

Convolution is a special mathematical operation on two function to produce a third function. Thus, the convolution layer consists on scanning the image with the filters, resulting in a feature maps. The process of moving the filters are called strides and it may have different size, although is usually 1 pixel each time [29].

To introduce non linearity in the feature maps, ReLU function is applied, in the ReLU layer.

In the pooling layer, the size of these feature maps is reduced by pooling them [30], which is taking only one value from the pooling window. Different kinds of pooling are known such maximum, minimum or average pooling.

After usually a several convolution, ReLU and pooling layers, the fully connected layer scan the reduced feature maps and convert the data into a values. In a classification task, flattering the data is often the previous step before the fully connected layer. As a final step after this layer, softmax loss classifier :

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Eq. 3

where σ(z) is a K-dimensional vector of real values. Softmax function gives an output of probabilities. This last layer is also known as loss layer.

CNN are often use for image classification, however, they can be also used for speech recognition [31] or even for music recommendation [32]. There are several architectures in the field of Convolutional Networks that have a name, such as LeNet, AlexNet, GoogleNet or VGGNet [29].

2.3 Architectures for this project

In this project U-Net, DeconvNet and Segnet architectures are build with CNNs. Although they are different, the basic idea behind all of them is to encode and decode the image in such way that the result would be a segmented image.

2.3.1 U-Net

This architecture won ISBI (International Symposium on Biomedical Imaging) cell tracking challenge 2015 for segmenting neuronal structures in electron microscopic stacks. The network does not have any fully connected layers and only uses the valid part of each convolution, i.e., the segmentation map only contains the pixels, for which the full context is available in the input image [3]. It consists of a contracting path (left side) and an expansive path (right side)[3], as it shown in the Figure 2. The structure is the following: 2 convolution layers with kernel size of 3 x 3 with ReLU activation function , then max-pooling layers with size of 2 x 2 on the left side and up-convolution (up-samling with convolution) with size of 2 x 2 on the right, then copy and crop of feature maps for each 4 steps before pooling and one final convolution with kernel size of 1 x 1. The number of filters goes from 64 to 1024, multiplying by 2 in each step. Although the input size and output size differs in the original paper [3]. A more powerful version was proposed as Unet++ [33].

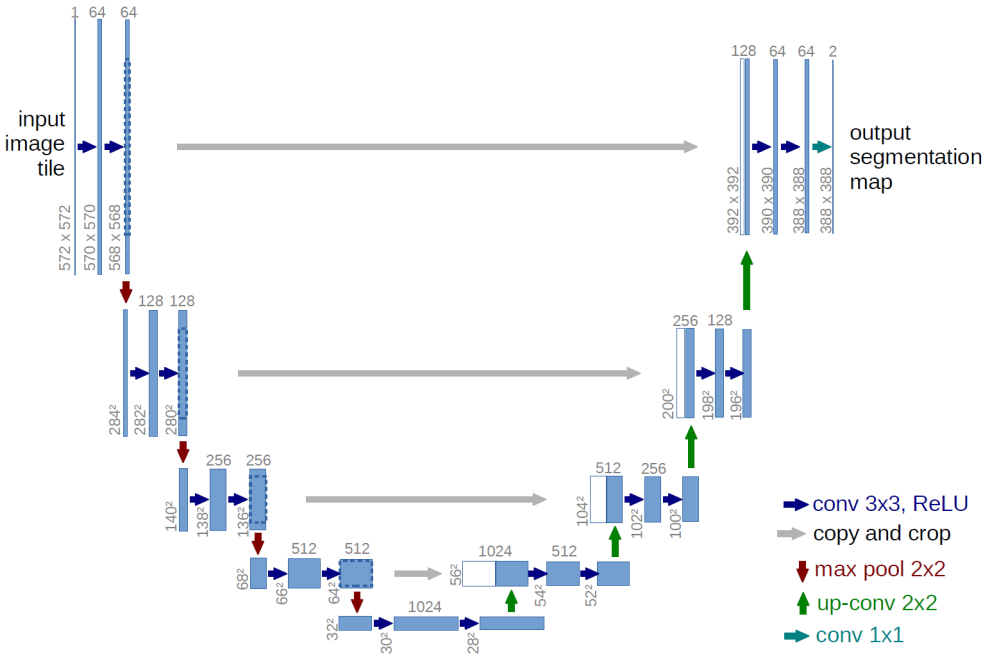


Figure 2. U-Net architecture[3]. Each blue box corresponds to a multi-channel feature map, white box to copied feature map [3]. Input image size is 572x572 and the output segmentation map 388x388.

U-Net has been tried in 3D images [34], Optic disc and cup [35], for Brain Tumor Detection [36] or even for singing voice separation [37].

2.3.2 DeconvNet

This model achieved accuracy of 72.5% with PASCAL VOC 2012 dataset, when it first appeared [5]. The main idea of the DeconvNet comes from successes with image reconstruction from its feature representation [5].

As for U-Net the model is composed of two parts: convolution and deconvolution networks (Figure 3). The first one extracts the features and the second one generates object segmentation [5]. They employ VGG 16-layer net with the last classification layer removed, to make both parts, but for the deconvolutional one they reverse the architecture of VGG 16. Such that the final output of the network is a probability map in the same size to input image [5].

The structure of VGG 16 is the following: 13 convolutional layers with kernel size of 3 x 3 are stacked one after the other with ReLU activation function, then max-pooling layers with size of 2 x 2 for encoder or up-convolution layers with size of 2 x 2 for decoder and the number of filters goes from 64 to 4096, multiplying by 2 in each step. Besides the 13 convolutional layers there is 1 fully connected layer with size of 1 x 1. As it mentioned before two layers have been removed from the architecture and VGG 16 is reversed for the decoder with max-pooling layers substituted by up-sampling layers.

DeconvNet had some success with building extraction from remote sensing images [38].

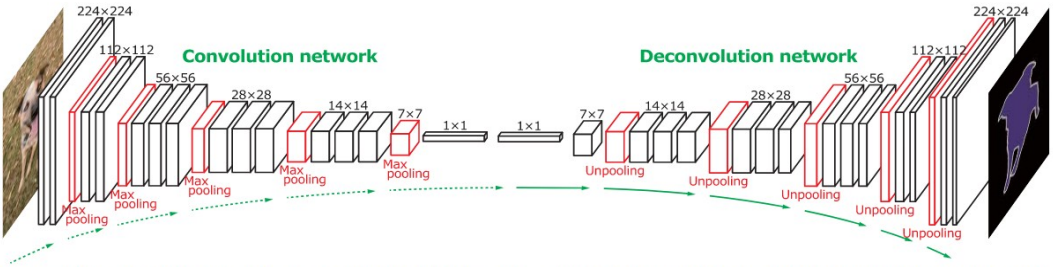


Figure 3. DeconvNet architecture[5]. The numbers indicates the size of feature maps, each network is made from 13 convolutional layers and 5 maxpooling or unpooling.

2.3.3 Segnet

The motivation to design SegNet arises from this need to map low resolution features to input resolution for pixel-wise classification [4]. It has the same main idea of encoder and decoder but much more simpler. The SegNet-Basic , which appears in this project, has 4 convolotional layers for encoder and 4 for decoder. It has similar maxpooling and upsample shapes and strides; batch normalization is used after each convolutional layer [4], but no biases are used after convolutions and no ReLU, non-linearity, is present in the decoder network. The original paper recommends 7 x 7 kernel size in both encoder and decoder convolutional layers. The filters goes from 64 to 512, for encoder as well as for decoder layers.

Segnet seems performing very well in different conditions with traffic data [39].

3. Materials and methods[1]

3.1 Dataset description

The dataset was obtained from Jiangxi Tecom Science Corporation, China [28]; and it correspond to *Dataset 1*. It consist of white blood cell images with white and yellow background, that also include some red cells. There is three hundred 120×120 images and their color depth is 24 bits. This images were taken by a Motic Moticam Pro 252A optical microscope camera with a N800-D motorized auto-focus microscope, and the blood smears were processed with a newly-developed hematology reagent for rapid cells staining [28], and saved in *.bmp* format.

The *Dataset 1*, also include ground truth segmentation results or masks, saved as *.png* format, that were manually sketched by domain experts, where the nuclei, cytoplasms and background including red blood cells are marked in white, gray and black respectively [28]. Its number and size is the same as the original microscopy images.

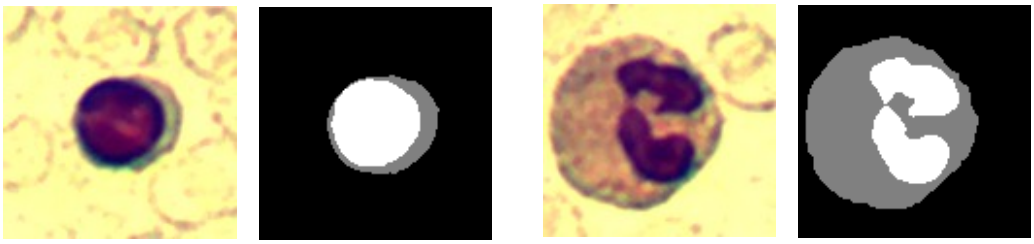


Figure 4. Example of image data from *Dataset 1*. The original microscopy image is followed by segmented image in black, grey and white. Two examples of cells are shown.

This data was shrinked because some images have different size from 120x120, and this could lead to problems when training the models. Thus, 76 images were not selected into final dataset. The other 224 images with their corresponding masks were transformed into numpy arrays.

Furthermore, the augmentation was performed. The images and their masks were flipped and rotated 90º to the left. This new data was added to the original one, forming the final dataset with size of 448 x 120 x 120 x 3 for "*x*" as colour images and with 448 x 120 x 120 for "*y*" as grey-scale images, which are ground truth results[2].

3.2 Cloud computing with Google's "Colaboratory" platform

1 All the code is available on: https://github.com/dIcarusb/Master_project
2 https://github.com/dIcarusb/Master_project/blob/master/Dataset_creation.py

Colaboratory (or *Colab*) is a fee online Jupyter notebook from Google, for education and research purposes [40]. It allows using GPU or TPU from the cloud making possible to train and run machine learning algorithms much faster than it would take in a regular computer.

For the project, GPU and last Python version 3.6 was used for implement each model.

3.3 Models

For each model, the data loaded was scaled and divided into two groups: train (93,3% from all data) and validation, called test (6.7% from all data), data. Besides the previous random shuffle with random seed with value of 2.

However, in the case of segmented images data, there is also a previous transformation into RGB image. The explanation behind this leads into the premise of segmentation of three classes or three different objects from the original image. Thus, the function called "*dummy*" makes this transformation by separating the values of grayscale images and creating extra columns for each of three values and filling them with 0 or 1 (whenever the value appears). In this way, the "*y*" data was transformed from one dimension with three different values into three dimensions with binary values.

On the other hand, the "*x*" dataset is scaled from the beginning by dividing it by 255.

All models are built with Keras with Tensorflow background. All required packages/libraries are loaded at the beginning.

In order to reduce the size of the model, all convolution and deconvolution layers are defined in blocks (functions) before the model function. The output layer is added for all models, which consists on 2D convolution with 1 x 1 kernel size and 3 filters, followed by dropout of 0.5 [41] and finally softmax activation function. For model compilation binary cross-entropy as the loss-function was chosen because the data for each class/dimension is binary; with *Adam* optimizer with 1e-4 of learning rate; and accuracy metrics.

All the initial convolutional kernels have Xavier uniform distribution. Finally, all the models are built as they are explained in background chapter, at the exception of the last maxpooling layer of the encoder and the first upsampling layer of the decoder, because of the output shape 15 x 15 from the last layer of the encoder. Therefore, the size of maxpoolings and upsamplings used is 3 x 3 (reducing the shape of the output to 5 x 5) to avoid the error that could appear by maintaining the size of 2 x 2, which reduces the output shape to 7 x 7.

As a particularity of DeconvNet[3] model, because of so many maxpooling and the restrictive shape of the images, the first maxpooling is 1 x 1, thus it maintains the same shape; while the last maxpooling is 3 x 3, the same modification as in other models. The same is reversed with upsampling layers. Before this structure (1 x 1, 2 x 2, 2 x 2, 2 x 2, 3 x 3 pooling) it was tried another version of pooling sizes such as 2 x 2, 2 x 2, 2 x 2, 3 x 3 and 5 x 5; but the best accuracy was obtained without the first and last pooling or with 1 x 1 size.

SegNet[4] particularity is the structure itself which is SegNet-Basic.

For U-Net[5] up-convolutional step is performed by `Conv2DTranspose`, and copy and crop step by `concatenate` functions from Keras. U-Net appears in annexes as the example of the models building.

When fitting/training the model, the batch size was 20 and epochs were 30 for all the arquitectures. Regarding the prediction of the model, after trained the model, the result of probabilities was rounded to obtain a specific class (0 or 1).

3.4 Metrics for performance evaluation

3.4.1 Jaccard similarity index

This coefficient is an intersection over the union, which is used to compare the similarity of two sets, and was developed to compare regional floras [42]. Jaccard index is defined as:

$$Jaccard = \frac{TP}{TP + FP + FN} = \frac{|A \cap B|}{|A \cup B|}$$

Eq. 4

where:
TP is true positive,
FP is false positive,
FN is false negative, and
A and B are two sets.

Regarding the project, the function `jaccard_similarity_score` from `sklearn.metrics` is used to perform a comparison between the segmented object and the ground truth object [43]. For each validation image the Jaccard coefficient is calculated for cytoplasm and nucleus. Then, the mean is obtained for all validation data and for each model.

3 https://github.com/dIcarusb/Master_project/blob/master/DeconvNet.ipynb
4 https://github.com/dIcarusb/Master_project/blob/master/segnet.ipynb
5 https://github.com/dIcarusb/Master_project/blob/master/UNET.ipynb

3.4.2 Dice Similarity Coefficient (DSC)

Dice Similarity or Sørensen–Dice[44] coefficient is also known as F1-score. DSC as well as Jaccard index, is used to compare the similarity of two sets, and is defined as:

$$DSC = \frac{2TP}{2TP + FP + FN} = \frac{2|A \cap B|}{|A| + |B|}$$

Eq. 5

where:
TP is true positive,
FP is false positive,
FN is false negative, and
A and B are two sets

In the project, the function `f1_score` is implemented using the package `sklearn.metrics` which performs the comparison of similarities between the predicted object and the ground truth one [45]. For each validation image DSC is calculated for cytoplasm and nucleus. Then, the mean is obtained for all the validation data and for each model.

4. Results

After training the models the graphs shown in Figure 5 were obtained. The SegNet accuracy for test dataset is the most varied one, probably due to the difficulty of generalizing. The other two models look more similar, following the same pattern, accuracy on validation dataset is slightly lower. The final accuracy as it is shown in Table 1, from SegNet and DeconvNet is very similar, and both of them are surpassed by U-Net architecture.

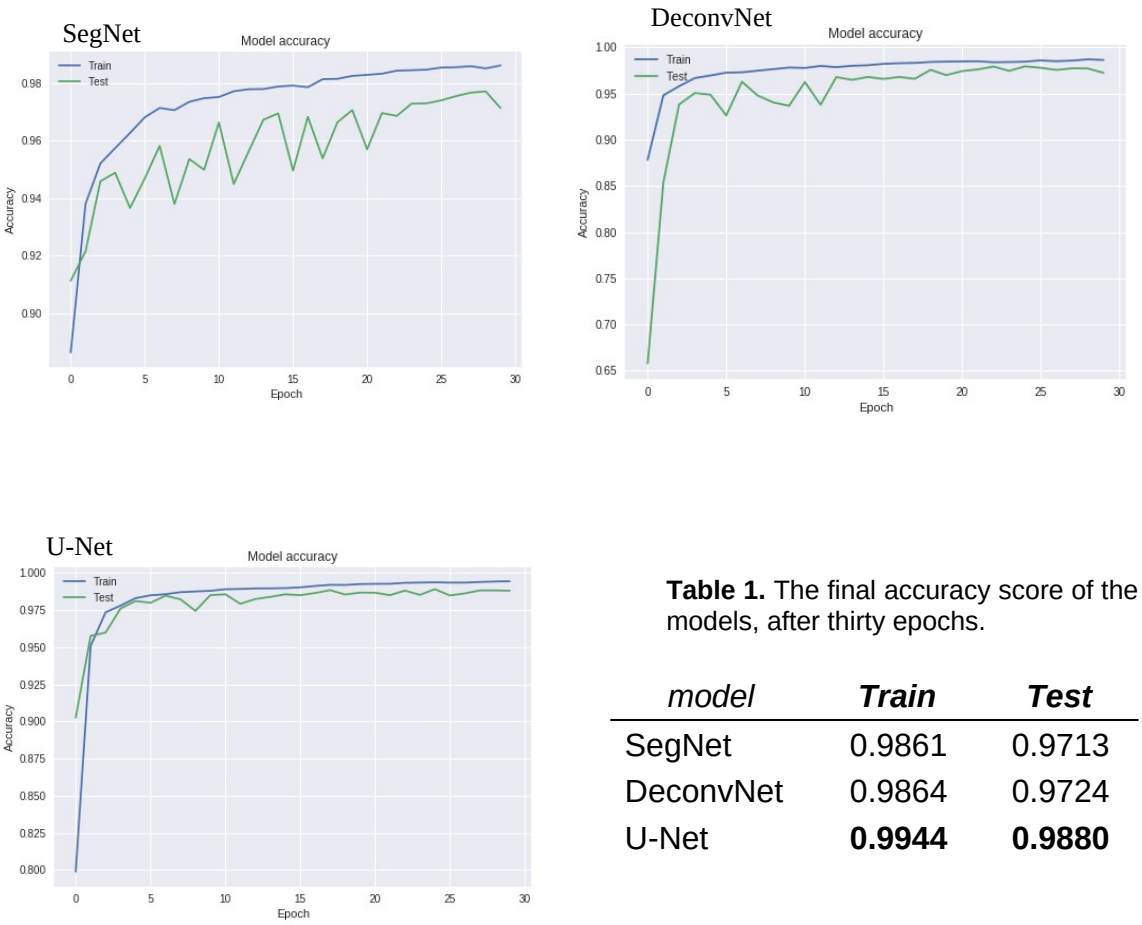


Table 1. The final accuracy score of the models, after thirty epochs.

model	***Train***	***Test***
SegNet	0.9861	0.9713
DeconvNet	0.9864	0.9724
U-Net	**0.9944**	**0.9880**

Figure 5. Evolution of accuracy. The plots show accuracy for both train and validation/test dataset through 30 epochs. Most of the models start with very high accuracy, 0.8-0.9, for training data and even higher for testing data, at exception of DeconvNet.

With the plots of the loss funtion occurs something similar to the evolution of the values: SegNet and DeconvNet are very similar and U-Net is still the best (Figure 6 and Table 2). As expected the SegNet model has also this saw-like behaviour with testing data (Figure 6). Therefore, it seems that the SegNet-Basic model starts very well but struggles with generalizing and getting better results.

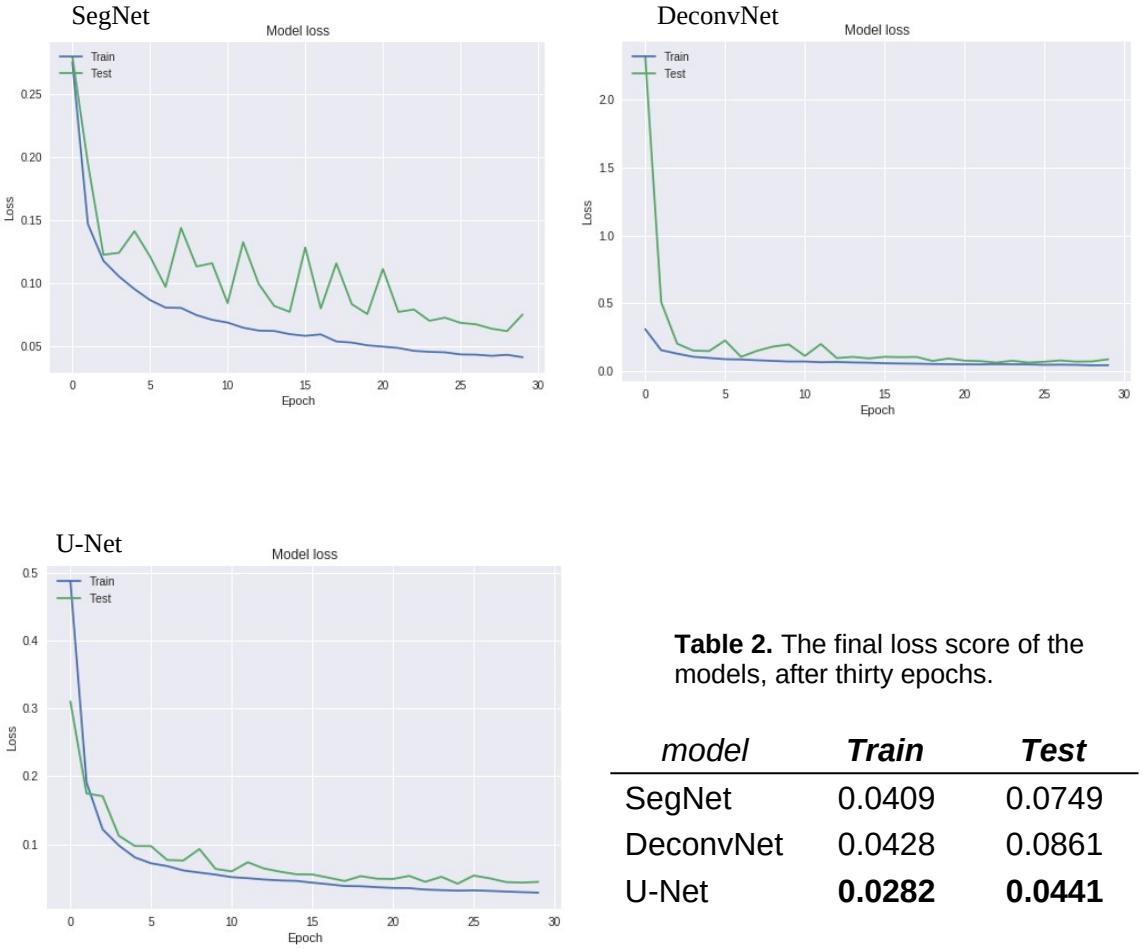


Table 2. The final loss score of the models, after thirty epochs.

model	***Train***	***Test***
SegNet	0.0409	0.0749
DeconvNet	0.0428	0.0861
U-Net	**0.0282**	**0.0441**

Figure 6. Evolution of loss. The loss started with 0.3-0.2 and went down less than 0.1, in all models. At exception of U-Net, which began with 0.4885 for train dataset, and DeconvNet testing data, which began with 2.3182 .

The accuracy is not the best way to measure the performance of segmentation, thus Jaccard similarity index and DSC were calculated. Once trained, the models, were used to predict the test data. Then, these predictions were used to calculate the similarity coefficients on cytoplasm and nucleus segmentation; and to compare the segmented image to the ground truth visually. In table 3 the average score appears with the standard error of the mean, calculated as :

$$SE = \frac{\sigma}{\sqrt{n}}$$

Eq. 6

where σ is standard error and "*n*" is the number of the sample.

Although for DSC the score is lower than for Jaccard index, for both the cytoplasm was more difficult to segment than the nucleus. However, all models were able to achieve very high score more than 0.9 with a small error for both coefficients, besides the small number of images (418 for

training and 30 for validation), even after the augmentation. U-Net as well as with accuracy and loss, have achieved a better score than the rest, in all comparisons.

The training runtime was also measured in order to compare the speed training of the models. DeconvNet was the slowest model with 1301.5 seconds, probably due its multiple layers. The second one was the SegNet-Basic with 727.97 seconds. And the fastest training was for U-Net, with only 476.17 seconds, in spite of having more convolutional layers than SegNet.

Table 3. Average Jaccard similarity index and DSC on 30 test images.

model	*Jaccard index on cytoplasm*	*Jaccard index on nucleus*	*DSC on cytoplasm*	*DSC on nucleus*
SegNet	0.9533 +/- 0.0027	0.9804 +/-0.0019	0.9294 +/-0.004	0.9696 +/- 0.003
DeconvNet	0.9560 +/- 0.0022	0.9823 +/-0.0016	0.9334 +/- 0.0034	0.9725 +/- 0.0025
U-Net	**0.9802 +/- 0.0020**	**0.9924 +/- 0.0008**	**0.97 +/-0.003**	**0.9882 +/- 0.0013**

Figure 6 shows some of the prediction examples for all three models, and in most cases, U-Net is slightly better predicting the nuclei and even the cytoplasms.

4.1 Discussion

In spite of small dataset, all models have shown very good results, by achieving high scores in different metrics, that have been employed in the evaluation. The data augmentation was not so significant, although it was included for this project, because the models had achieved almost the same or similar accuracies as well as without the augmentation(annexes). This means that the models perform well even with smaller dataset.

The best model was U-Net, though. It shows not only a higher scores but also a better runtime among all models. However, it only became better after the addition of *Dropout* into the output layer.

The original study[1] had proposed their own model and compared it with other architectures , such as U-Net; nevertheless, they used different methodology and metrics to evaluate, wich makes their results not comparable to this project. Besides the original study had a different purpose. However, the master project opens the possibility to explore other architectures with the same dataset.

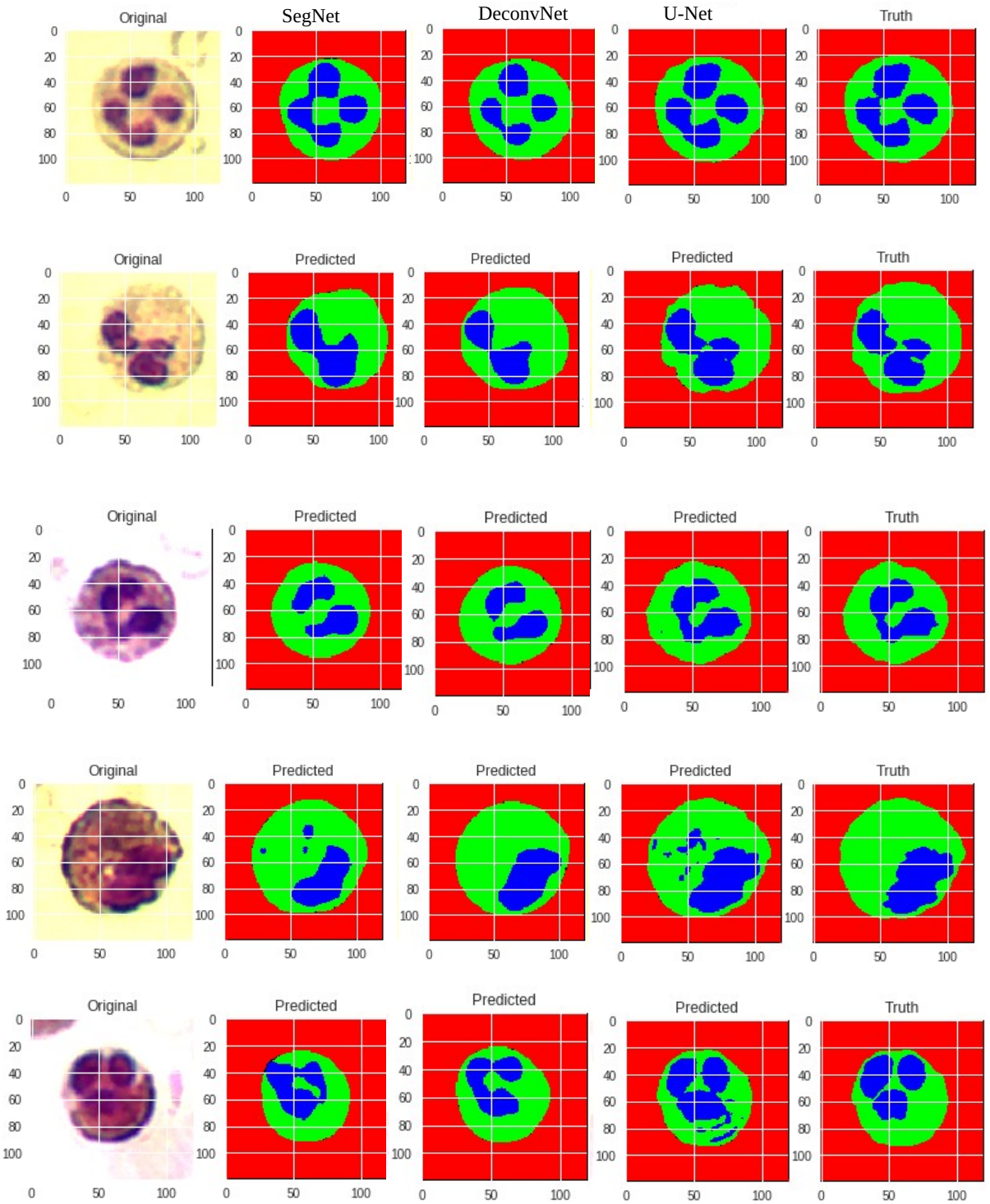


Figure 7. Prediction comparisons. Each row is different test image, and it follows SegNet-Basic prediction, DeconvNet prediction, U-Net prediction and the segmented expected image. Nucleus in blue, cytoplasm in green and background in red. X and Y axes represent sizes.

5. Conclusions

-The semantic segmentation of white blood cells was successfully performed, even with such a small amount of data.

-All of the models got a very high score in accuracy, Jaccard similarity coefficient and DSC. Although U-Net was the best, even achieving the best time, which makes this segmentation architecture the most efficient among the three.

-The segmentation of the peripheral blood cells is possible even with small dataset.

-However, this success doesn’t mean that these models will segment other blood microscopic images with different pixel size and neither with other types of cells, after training with the dataset used in this project. Although, the architectures showed enough competence to be employed in a semantic segmentation task, which gives them the confidence in training and predicting with other data as mentioned before.

-The major problem was with mask data or ground truth segmentation results. These grey-scale images were totally inadequate to the models, thus the transformation to the RGB images was an indispensable solution.

-Finally, this project shows the state of the art of semantic segmentation using artificial neural networks, for segment peripheral blood cell images, which in turn could be employed in in diagnostic automation. It also opens the possibility of exploring another segmentation architecture, in the future, such as Tiramisu, GAN, RNN, etc; which could perform even better than U-Net.

6. Glossary

WBC: white blood cells

Semantic segmentation: partition the image into semantically meaningful parts, and to classify each part into one of the pre-determined classes.

RGB: red, green, blue. RGB color model is for the sensing, representation and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography.

ANN, artificial neural network: is a computational model based on the structure and functions of biological neural networks.

ReLU: rectified linear unit.

Adam: adaptive moment estimation, an algorithm for first-order gradient-based optimization of stochastic objective functions.

CNN: convolutional neural network is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery.

RNN: recurrent neural network is a class of artificial neural network where connections between nodes form a directed graph along a sequence.

GAN: generative adversarial networks are a class of artificial intelligence algorithms used in unsupervised machine learning, implemented by a system of two neural networks contesting with each other in a zero-sum game framework.

Autoencoders: is a type of artificial neural network used to learn efficient data codings in an unsupervised manner; typically for the purpose of dimensionality reduction.

U-Net: convolutional neural network that was developed for biomedical image segmentation at the Computer Science Department of the University of Freiburg, Germany.

ISBI challenge: International Symposium on Biomedical Imaging (ISBI) is a scientific conference dedicated to mathematical, algorithmic, and computational aspects of biological and biomedical imaging, across all scales of observation. Besides the conferences there is also challenges where some researcher groups can participate.

DeconvNet: Learning Deconvolution Network for Semantic Segmentation created by Hyeonwoo Noh, Seunghoon Hong and Bohyung Han at POSTECH.

PASCAL VOC 2012: dataset. Images with 20 classes. The train/val data has 11,530 images containing 27,450 regions of interest (ROI) annotated objects and 6,929 segmentations.

VGG 16: Visual Geometry Group, is a deep convolutional neural network architecture with 16 layers.

SegNet: is a deep encoder-decoder architecture for multi-class pixelwise segmentation researched and developed by members of the Computer Vision and Robotics Group at the University of Cambridge, UK.

Encoder: is a device, circuit, transducer, software program, algorithm or person that converts information from one format or code to another, for the purposes of standardization, speed or compression.

Decoder: makes the reverse of the encoder.

State of the art: highest level of general development, as of a device, technique, or scientific field achieved at a particular time.

GPU: graphics processing unit, is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device.

CPU: central processing unit, is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output operations specified by the instructions.

Jaccard index: is a statistic used for comparing the similarity and diversity of sample sets.

DSC: dice similarity coefficient is a statistic used for comparing the similarity and diversity of sample sets.

Github: is a web-based hosting service for version control using Git.

7. Bibliography

[1] Xin Zheng ,Yong Wang, Guoyou Wang, Jianguo Liu (2018), Fast and robust segmentation of cell images by self-supervised learning. Micron, Volume 107, April 2018, Pages 55-71

[2] Naugler, Christopher, EmadA Mohammed, MostafaM. A. Mohamed, and BehrouzH Far. 2014. Peripheralblood smear image analysis: A comprehensive review. Journal of Pathology Informatics 5 (1): 9.doi:10.4103/2153-3539.129442.Ng, A. 2011. CS294A Lecture notes, 1-19, doi:10.1371/journal.pone.0006098

[3] Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham

[4] Badrinarayanan, Vijay & Kendall, Alex & Cipolla, Roberto. (2015). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence. PP. 10.1109/ TPAMI.2016.2644615.

[5] Hyeonwoo Noh, Seunghoon Hong, Bohyung Han, (2015). Learning Deconvolution Network for Semantic Segmentation, Proceedings of the IEEE International Conference on Computer Vision.

[6] Thoma, Martin. (2016). A Survey of Semantic Segmentation.

[7] Long, Jonathan & Shelhamer, Evan & Darrell, Trevor. (2015). Fully convolutional networks for semantic segmentation. 3431-3440. 10.1109/CVPR.2015.7298965.

[8] Alexandre Briot, Prashanth Viswanath, Senthil Yogamani; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2018, pp. 663-672

[9] Hong, Seunghoon Noh, Hyeonwoo Han, Bohyung, Decoupled Deep Neural Network for Semi-supervised Semantic Segmentation,Advances in Neural Information Processing Systems 28, 2015, pp.1495-1503

[10] Garcia-Garcia, Alberto et al. “A Review on Deep Learning Techniques Applied to Semantic Segmentation.” CoRR abs/1704.06857 (2017): n. pag.

[11] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno and F. Lopez-Ferreras, "Road-Sign Detection and Recognition Based on Support Vector Machines," in IEEE Transactions on Intelligent Transportation Systems, vol. 8, no. 2, pp. 264-278, June 2007.
doi: 10.1109/TITS.2007.895311

[12] Gh AMZA, Catalin & Amza, Catalin Gheorghe & Popescu, Diana. (2012). IMAGE SEGMENTATION FOR INDUSTRIAL QUALITY INSPECTION. Fiability & Durability/Fiabilitate si Durabilitate. 1 supliment.

[13] N. Moon, E. Bullitt, K. Van Leemput, and G. Gerig, Automatic brain and tumor segmentation, in Medical Image Computing and Computer-Assisted Intervention—MICCAI 2002.Springer, 2002, pp.372–379

[14] Wei GQ., Arbter K., Hirzinger G. (1997) Automatic tracking of laparoscopic instruments by color coding. In: Troccaz J., Grimson E., Mösges R. (eds) CVRMed-MRCAS'97. CVRMed 1997, MRCAS 1997. Lecture Notes in Computer Science, vol 1205. Springer, Berlin, Heidelberg

[15] Cohen, Assaf & Rivlin, Ehud & Shimshoni, Ilan & Sabo, Edmond. (2015). Memory Based Active Contour Algorithm using Pixel-level Classified Images for Colon Crypt Segmentation. Computerized Medical Imaging and Graphics. 43. 10.1016/j.compmedimag.2014.12.006.

[16] C. Huang, L. Davis, and J. Townshend, “An assessment of support vector machines for land cover classification,”International Journal of remote sensing, vol. 23, no. 4, pp. 725–749, 2002

[17] Cybenko, G., Approximation by Superposition of a Sigmoidal Function, Mathematics Control Signals Systems, Vol. 2, 1989, pg 303-314.] [Hornik, K., M. Stinchcombe & H. White, Multilayer Feedforward Networks are Universal Approximator, Neural Networks, Vol. 2, 1989, pg 359-3661

[18] Glorot, Xavier & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. Journal of Machine Learning Research - Proceedings Track. 9. 249-256.

[19] Nair, Vinod & E. Hinton, Geoffrey. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair. Proceedings of ICML. 27. 807-814.

[20] Xu, Bing & Wang, Naiyan & Chen, Tianqi & Li, Mu. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network.

[21] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, Yoshua Bengio, Maxout Networks, JMLR WCP 28 (3): 1319-1327, 2013

[22] Beyond regression: new tools for prediction and analysis in the behavioural sciences. Werbos PJ. 1975.

[23] Bischoff, W. (2011). Loss Function. In International Encyclopedia of Statistical Science, M. Lovric, ed. (Springer), pp. 762–763.

[24] Ruder, S. (2016), An overview of gradient descent optimization algorithms,

arXiv:1609.04747.

[25] Wilson, D & Martinez, Tony. (2004). The general inefficiency of batch training for gradient descent learning. Neural networks : the official journal of the International Neural Network Society. 16. 1429-51. 10.1016/S0893-6080(03)00138-2.

[26] Andrew Ng, Feature selection, L1 vs L2 regularization, and rotational invariance, in: ICML '04 Proceedings of the twenty-first international conference on Machine learning, Stanford, 2004.

[27] Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.

[28] Zheng, Xin (2018), Data for: Fast and robust segmentation of cell images by self-supervised learning, Mendeley Data, v1 http://dx.doi.org/10.17632/w7cvnmn4c5.1

[29] http://cs231n.github.io/convolutional-networks/ 29-12-2018.

[30] A theoretical analysis of feature pooling in visual recognition. Boureau Y, Ponce J, LeCun Y. Proceedings of the International Conference on Machine Learning. 2010, 111-118.

[31] Huang, Jui-Ting & Li, Jinyu & Gong, Yifan. (2015). An analysis of convolutional neural networks for speech recognition. 4989-4993. 10.1109/ICASSP.2015.7178920.

[32] Van Den Oord, A & Dieleman, S & Schrauwen, B. (2013). Deep content-based music recommendation. Advances in Neural Information Processing Systems.

[33] Zhou Z., Rahman Siddiquee M.M., Tajbakhsh N., Liang J. (2018) UNet++: A Nested U-Net Architecture for Medical Image Segmentation. In: Stoyanov D. et al. (eds) Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support. DLMIA 2018, ML-CDS 2018. Lecture Notes in Computer Science, vol 11045. Springer, Cham

[34] Çiçek Ö., Abdulkadir A., Lienkamp S.S., Brox T., Ronneberger O. (2016) 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In: Ourselin S., Joskowicz L., Sabuncu M., Unal G., Wells W. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016. MICCAI 2016. Lecture Notes in Computer Science, vol 9901. Springer, Cham

[35] Sevastopolsky, A. Pattern Recognit. Image Anal. (2017) 27: 618. https://doi.org/10.1134/S1054661817030269

[36] Dong H., Yang G., Liu F., Mo Y., Guo Y. (2017) Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks. In: Valdés Hernández M., González-Castro V. (eds) Medical Image Understanding and Analysis. MIUA 2017. Communications in Computer and Information Science, vol 723. Springer, Cham

[37] Jansson, A., Humphrey, E., Montecchio, N., Bittner, R., Kumar, A. and Weyde, T. (2017). Singing voice separation with deep U-Net convolutional networks. Paper presented at the 18th International Society for Music Information Retrieval Conference, 23-27 Oct 2017, Suzhou, China.

[38] Z. Huang, G. Cheng, H. Wang, H. Li, L. Shi and C. Pan, "Building extraction from multi-source remote sensing images via deep deconvolution neural networks," 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Beijing, 2016, pp. 1835-1838.

[39] KM Lye, KC Chua, CC Ko,Performance of Segnet — a simulation study, Computer Communications,Volume 10, Issue 6,1987, pp.297-303

[40] https://colab.research.google.com/notebooks/welcome.ipynb , 24-12-2018; FAQ: https://research.google.com/colaboratory/faq.html , 24-12-2018.

[41] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R.R. (2012), Improving neural networks by preventing co-adaptation of feature detectors, arXiv:1207.0580.

[42] e.g., Jaccard 1912, The distribution of the flora of the alpine zone, New Phytologist 11:37-50

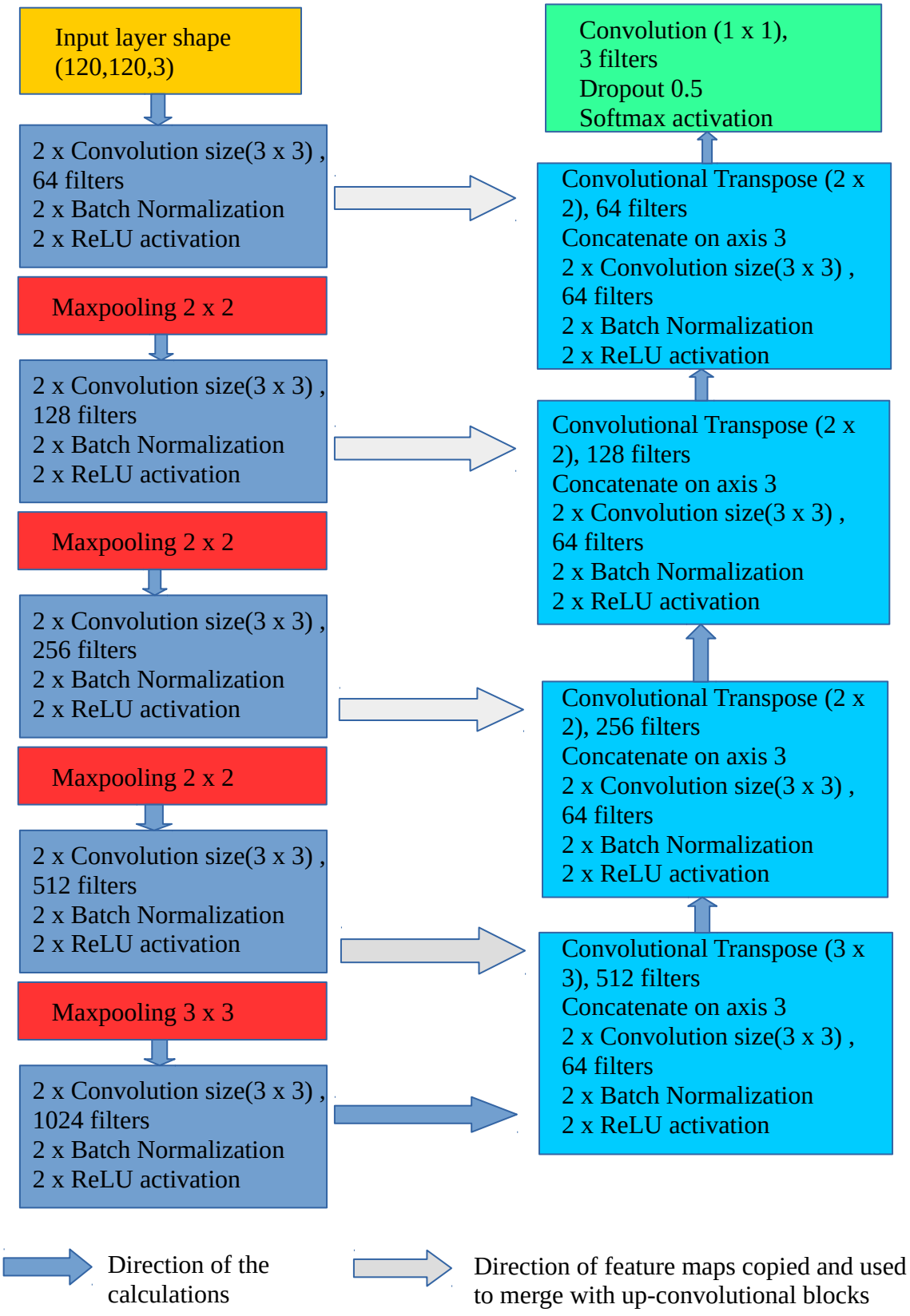
[43] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard_similarity_score.html , 24-12-2018.

[44] Sørensen, T. (1948). "A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons". Kongelige Danske Videnskabernes Selskab. 5 (4): 1–34. & Dice, Lee R. (1945). "Measures of the Amount of Ecologic Association Between Species". Ecology. 26 (3): 297–302.

[45] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html , 24-12-2018.

8. Annexes

U-Net model example (more details in GitHub link: https://github.com/dIcarusb/Master_project/blob/master/UNET.ipynb) :



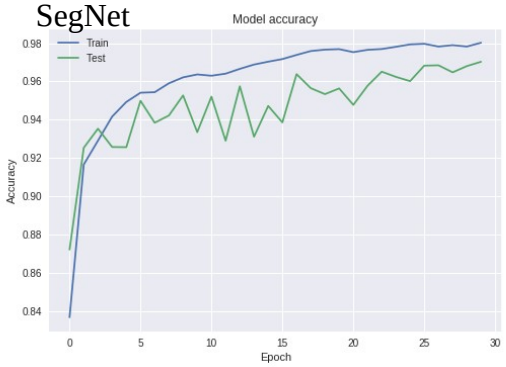
The code of U-Net:

Model

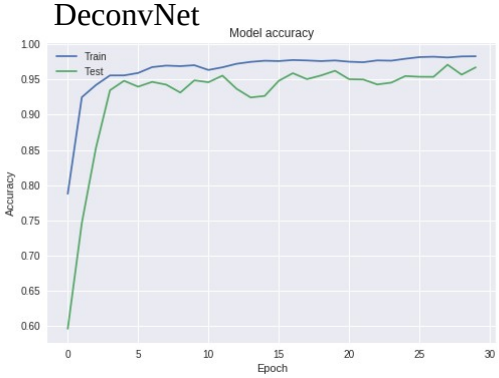
```
def conv_block(tensor, nfilters, size=3, padding='same', initializer="glorot_uniform"):
    x = Conv2D(filters=nfilters, kernel_size=(size, size), padding=padding, kernel_initial
izer=initializer)(tensor)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = Conv2D(filters=nfilters, kernel_size=(size, size), padding=padding, kernel_initial
izer=initializer)(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    return x


def deconv_block(tensor, residual, nfilters, size=2, padding='same', strides=(2, 2)):
    y = Conv2DTranspose(nfilters, kernel_size=(size, size), strides=strides, padding=paddi
ng)(tensor)
    y = concatenate([y, residual], axis=3)
    y = conv_block(y, nfilters, size=size)
    return y

def Unet(img_height, img_width, nclasses=3, filters=64):
    input_layer = Input(shape=(img_height, img_width, 3), name='image_input')

    # down
    conv1 = conv_block(input_layer, nfilters=filters)
    conv1_out = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = conv_block(conv1_out, nfilters=filters*2)
    conv2_out = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = conv_block(conv2_out, nfilters=filters*4)
    conv3_out = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv4 = conv_block(conv3_out, nfilters=filters*8)
    conv4_out = MaxPooling2D(pool_size=(3, 3))(conv4)
    conv5 = conv_block(conv4_out,  nfilters=filters*16)

    # up
    deconv6 = deconv_block(conv5, residual=conv4,  nfilters=filters*8, strides=(3,3))
    deconv7 = deconv_block(deconv6, residual=conv3, nfilters=filters*4)
    deconv8 = deconv_block(deconv7, residual=conv2, nfilters=filters*2, size=3)
    deconv9 = deconv_block(deconv8, residual=conv1, nfilters=filters)

    # "U" shape

    # output
    output_layer = Conv2D(filters=nclasses, kernel_size=(1, 1))(deconv9)
    output_laye = Dropout(0.5)(output_layer)
    output_layer = Activation('softmax')(output_layer)

    model = Model(inputs=input_layer, outputs=output_layer, name='Unet')
    model.compile(optimizer = Adam(lr = 1e-4), loss = 'binary_crossentropy', metrics = ['a
ccuracy'])

    return model
```

Models run without data augmentation, accuracy:

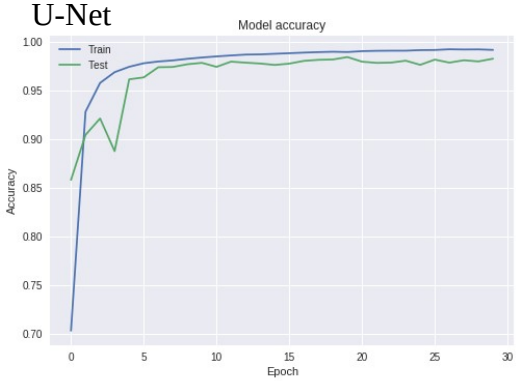
SegNet



DeconvNet



U-Net

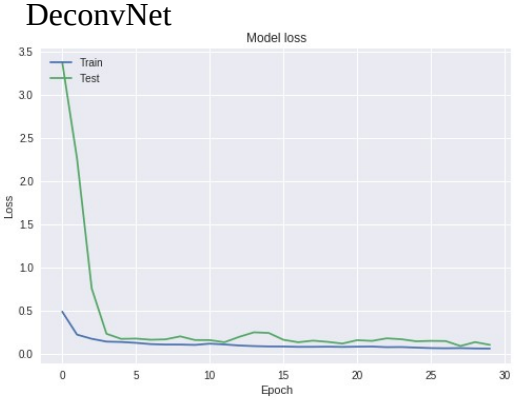


model	***Train***	***Test***
SegNet	0.9802	0.9703
DeconvNet	0.9830	0.9673
U-Net	**0.9921**	**0.9830**

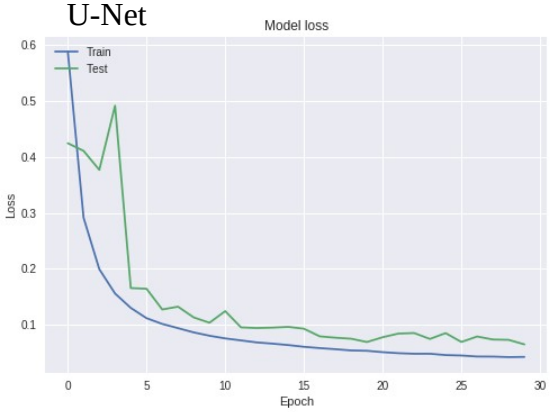
Loss function:

SegNet

DeconvNet



U-Net



model	***Train***	***Test***
SegNet	0.0573	0.0763
DeconvNet	0.0619	0.1046
U-Net	**0.0427**	**0.0651**