Image Segmentation using U-Net, DenseNet and CapsuleNet

Zhankun Luo
Dept. of ECE
Purdue University Northwest
Hammond, IN, USA
luo333@pnw.edu

Andres Jara
Dept. of ECE
Purdue University Northwest
Hammond, IN, USA
ajaralom@pnw.edu

Wen Ou
Dept. of ECE
Purdue University Northwest
Hammond, IN, USA
ou21@pnw.edu

***Abstract*—This paper focuses on the development and study of image segmentation. This topic has been developed for quite a time. The comparison of several types of architectures are also detailed taking into account U-Net, DenseNet, and CapsuleNet. The experiment was developed and performed using Python in order to prove the accuracy and compare these types of architectures to see the best results possible. Finally, the results and conclusions are given for future development and research.**

***Index Terms*—U-Net, DenseNet, Capsule, Segmentation**

I. Introduction

The study of Deep Learning has been arising throughout the time helping computer vision technology to develop widely. This kind of technology has been used for different areas such as image classification, face recognition, objects in images and more. Many of these computer vision tasks require intelligence segmentation of an image so that it is understandable and easier to obtain an analysis of each one of its parts. Deep learning can also learn patterns in visual inputs so that it can predict object classes that conform an image. The deep learning architecture used for image processing is called Convolutional Neural Network (CNN). Models of this kind are typically trained and performed on specialized graphics processing units (GPUs) so that running time can be reduced.

Image segmentation is a process in computer vision that divides a visual input into segments to simplify image analysis. These segments involve objects or part of objects in order to sort pixels into large components. By doing this process, it eliminates the need of considering individual pixels as units of observation. Among the levels of image analysis, there exists three: classification, object detection, and segmentation.

This paper focuses on the image analysis using segmentation which identifies parts of the image and tries to conclude where they belong to. The architectures used throughout the coding, research and paper report are U-Net, DenseNet, and CapsuleNet. These three types of architectures were selected as part of the development of image processing.

Image segmentation consists on the process of dividing or partitioning an image into several pieces or parts referred as segments. These segments are later processed to identify the important information that will provide the correct output. It also focuses on creating a pixel-wise mask for each object that object detection identifies. Since object detection does not provide enough information apart of bounding box coordinates only, image segmentation provides a more deeply understanding of the shape and location.

II. Architecture of Neural Networks

The Neural Networks are complex structures that consist of multiple inputs going through artificial neurons that will ultimately produce a single output. The three types of architecture used in this project is as follows: U-Net, DenseNet, and CapsuleNet.

A. U-Net

U-Net is a convolutional neural network that was mainly created for biomedical image segmentation [1]. This network is modified to work with fewer training images and to output more precise segmentations. The design of the encoder and the decoder in U-Net does not take so much time to run on a modern graphic processing unit (GPU). The main purpose of U-net architecture is to supplement a usual contracting network by successive layers. During this stage, these layers increase the resolution of the output using pooling operations which are also replaced by upsampling operations.

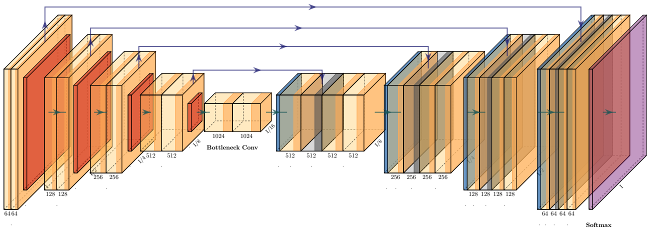


Fig. 1: U-Net Architecture.

B. DenseNet

The DenseNet architecture [2] was given the CVPR 2017 Best Paper Award. A deep DenseNet consists of three dense blocks; transition blocks between each pair of adjacent dense blocks and other convolution layers. The transition layers between two adjacent dense blocks are referred as the combination of one batch normalization layer, one ReLU layer and one convolution layer to change the channel size and the feature map size. In the customized DenseNet model, the stride of transition layer is 1 so that it keeps the shape of output feature map same as of the one from input feature map.

An input image x_0 is passed through the dense block. This block comprises L layers, each of which implements a non-linear mapping $H_l(\cdot)$, where l indicates the index of layer. $H_l(\cdot)$ can be a composite function of basic layers such as Batch Normalization (BN), rectified linear units (ReLU), and Convolution (Conv). We denote x_l as the output of the l-th layer. In order to further improve the information flow between layers, the direct connections are established from any other layer to all subsequent layers. Consequently, the input of the l-th layer is the feature-maps of all preceding layers. The number of layers is set equal to $L = 6$ in the customized model (experimental model). $x_0, \cdots, x_{l-1}$

$$x_l = H_l([x_0, x_1, \cdots, x_{l-1}]), \tag{1}$$

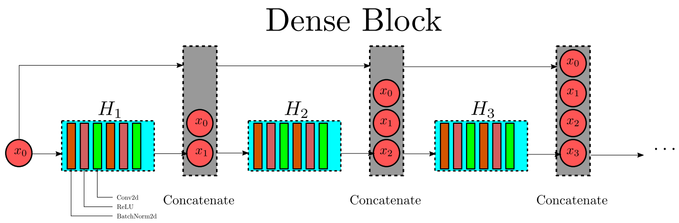


Fig. 2: Dense Block in the DenseNet.

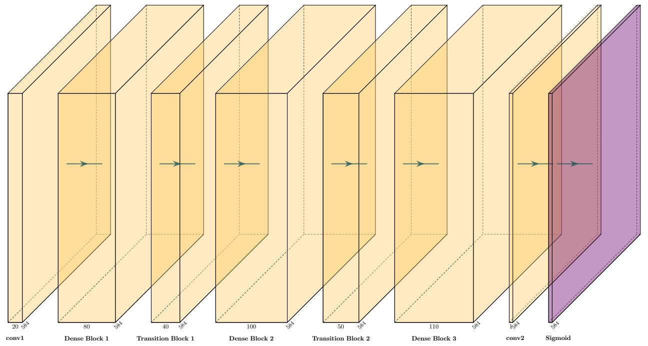


Fig. 3: DenseNet Architecture.

C. CapsuleNet

This type of neural network architecture is still new in the industry but does an immense advanced approach to previous neural networks designs, especially for computer vision tasks. Capsule networks [3] are a collection of neurons that stores various information about the image such as position, rotation, and scale. The architectural design consists on three main parts: Primary capsules, higher layer capsules, and loss calculation. Moreover, the segmentation task can be implemented using the capsule unit [4]. The architecture of CapsuleNet for Segmentation (SegCaps) is demonstrated in Fig. 4.

- Primary Capsules: In this capsule, the process of inverse graphics is developed using three different processes: convolution, reshape function, or squash function. Also, the input image is fed into a couple of convolution layers.

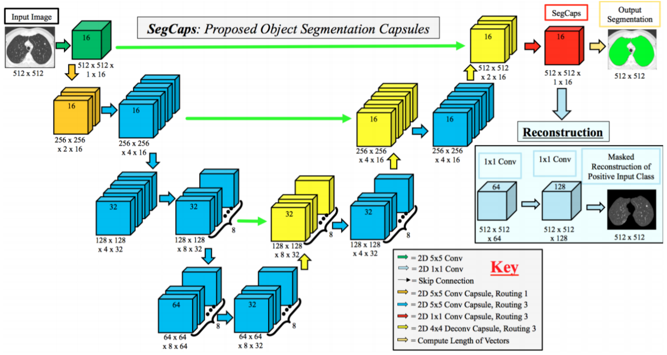
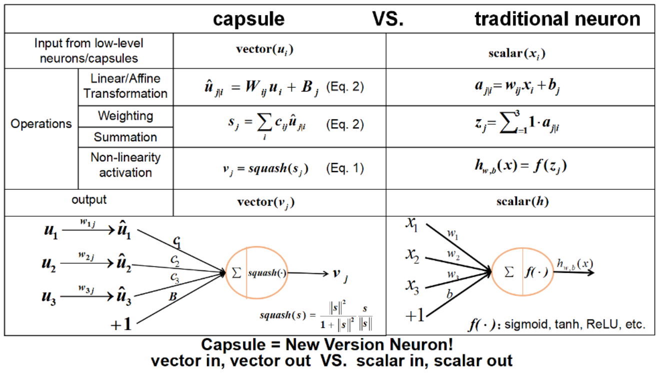


Fig. 4: CapsuleNet for Segmentation (Segcaps) Architecture.



		capsule	VS. traditional neuron
Input from low-level neurons/capsules		$\mathbf{vector}(u_i)$	$\mathbf{scalar}(x_i)$
Operations	Linear/Affine Transformation	$\hat{u}_{j\|i} = W_{ij} u_i + B_j$ (Eq. 2)	$a_{j\|i} = w_{ij} x_i + b_j$
	Weighting Summation	$s_j = \sum_i c_{ij} \hat{u}_{j\|i}$ (Eq. 2)	$z_j = \sum_{i=1}^{3} 1 \cdot a_{j\|i}$
	Non-linearity activation	$v_j = squash(s_j)$ (Eq. 1)	$h_{w,b}(x) = f(z_j)$
output		$\mathbf{vector}(v_j)$	$\mathbf{scalar}(h)$

u_1
w_{1j}
$\hat{u}_1$
u_2
w_{2j}
$\hat{u}_2$
u_3
w_{3j}
$\hat{u}_3$
$+1$
c_1
c_2
c_3
B
Σ
$squash(\cdot)$
v_j
$squash(s) = \frac{\|s\|^2}{1+\|s\|^2} \frac{s}{\|s\|}$
x_1
x_2
x_3
$+1$
w_1
w_2
w_3
b
Σ
$f(\cdot)$
$h_{w,b}(x)$
$f(\cdot)$: sigmoid, tanh, ReLU, etc.

Capsule = New Version Neuron!
vector in, vector out VS. scalar in, scalar out

Fig. 5: Comparison between Capsule neuron and Traditional neuron.

- Higher layer capsules: One of the benefits of using this capsule is that the path of the activation can be traced so that the hierarchy of the parts can be sorted out easily. Even though the primary layer did its job, the higher layer calculates its own output and double-check with the previous prediction.
- Loss calculation: Once the decision has been made, the classification takes place so that it can be determined whether the decision is correct or close to perfect.

III. Experimental Results

A. Dataset Description

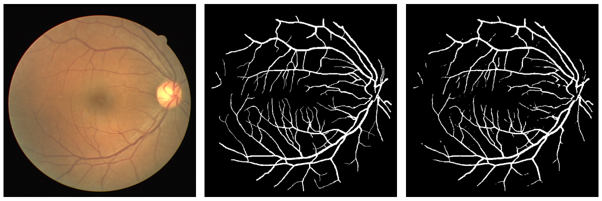
In order to test the sensibility of networks for segmentation tasks, a complicated segmentation dataset is chosen for training and testing, whose name is DRIVE that means Digital Retinal Images for Vessel Extraction.

Photos from the DRIVE database are from the diabetic retinopathy screening program in Netherlands. 40 photos are randomly selected from 400 diabetic patients between the ages 25 and 90.

DRIVE dataset are separated into 2 sub folders, one of which is *training folder*, and the other sub folder is *test folder*, both containing 20 images. For the training images, manual annotation for each image is available; while for the test cases, manual segmentations are lacked.

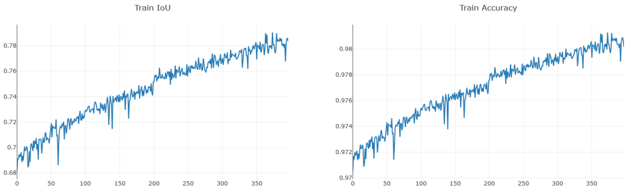
B. The U-Net networks

After training runs for 350 epochs, the accuracy reached 98%, and Intersection over Union (IoU) reached 80%.



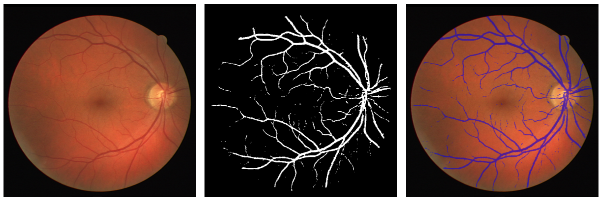
(a) Original Image (b) Target Image (c) Prediction

Fig. 6: Images in the training data set for U-Net.



(a) Training IoU (b) Training accuracy

Fig. 7: IoU and accuracy training of U-Net.

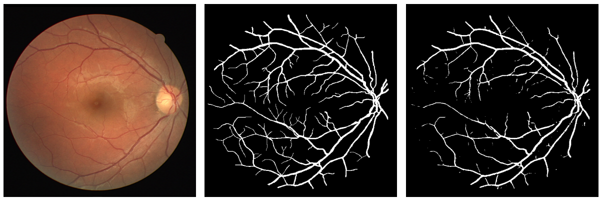


(a) Test Image (b) Test Prediction (c) Masked Image

Fig. 8: Testing, prediction, and masked images for U-Net.

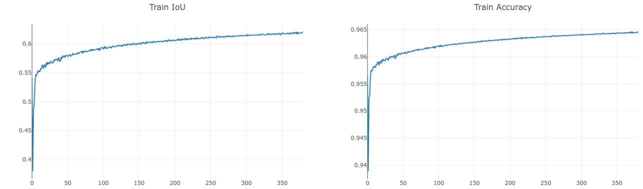
C. The DenseNet networks

After training was run for 350 epochs, the accuracy reached 96.5%, and the accuracy of IoU reached 62%.



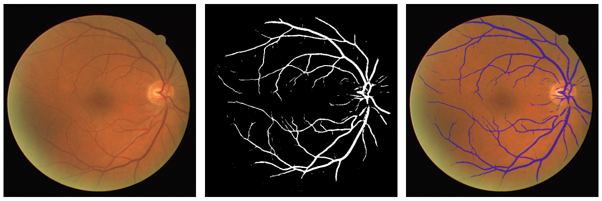
(a) Original Image (b) Target Image (c) Prediction

Fig. 9: Training, target, and predicted images for DenseNet.



(a) Training IoU (b) Training accuracy

Fig. 10: IoU and accuracy training of DenseNet.

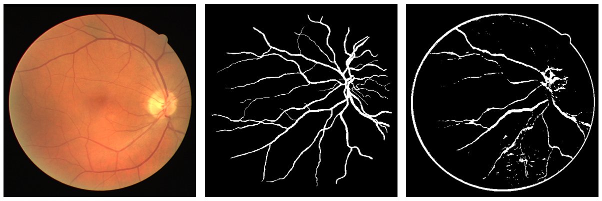


(a) Test Image (b) Test Prediction (c) Masked Image

Fig. 11: Testing, prediction, and masked images for DenseNet.

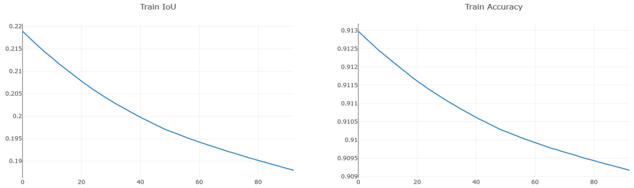
D. The SegCaps networks

After training ran for 350 epochs, the loss function did not converge, and IoU with its accuracy didn't improve. IoU resulted in a value of less than 0.2.



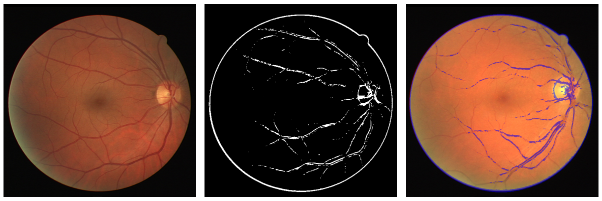
(a) Original Image (b) Target Image (c) Prediction

Fig. 12: Training, target, and predicted image for SegCaps.



(a) Training IoU (b) Training Accuracy

Fig. 13: IoU and accuracy training of SegCaps.



(a) Test Image (b) Test Prediction (c) Masked Image

Fig. 14: Test, predicted, and masked images for SegCaps.

The results of the experiment for U-Net, DenseNet and SegCaps after training for 350 epochs are list in TABLE I. The performance of U-Net outcompetes DenseNet and SegCaps in both IoU training and accuracy.

TABLE I: Comparison of U-Net, DenseNet and SegCaps

Metrics	**U-Net**	**DenseNet**	**SegCaps**
IoU	**80%**	62%	19%
Accuracy	**98%**	96.5%	9%

IV. Conclusion

The U-Net architecture showed the best performance on the DRIVE dataset. Also, some good results were obtained by DenseNet. In the other hand, the SegCaps networks could not converge on the DRIVE dataset. This could be proved because SegCaps was unable to handle complicated segmentation tasks. There is still some perspective and work to do for future research to improve Capsules networks for segmentation tasks.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[2] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[3] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” *Advances in neural information processing systems*, vol. 30, pp. 3856–3866, 2017.

[4] R. LaLonde and U. Bagci, “Capsules for object segmentation,” *arXiv preprint arXiv:1804.04241*, 2018.