U-NET RESIDUAL TRANSFORMER ARCHITECTURE FOR THREE TYPES NEURAL CELL SEGMENTATION

COMP 576 Fall 2021 Final Project

Maojie Tang	**Tianxing Wang**	**Yuhua Chen**	**Jingyu Fu**
mt84	tw44	yc143	jf70

ABSTRACT

The segmentation of neural cell microscopic images is a widely discussed topic in the medical field. U-Net has been a light and efficient architecture for medical image segmentation. In this project, we presented URT, a U-Net Residual Transformer architecture. In the URT model, we replaced the original convolutional layers with residual blocks and used transformer encoder/decoders to connect U-net horizontally. We evaluated the performance and robustness of the URT architecture over other baseline approaches in the task of neural cell image segmentation. The project repository is available here.

***K*eywords** U-Net · Transformer · Residual · Medical Image · Neural Cell Segmentation

1 Introduction

1.1 Background

Neurological disorders are the leading cause of death and disability worldwide. Light microscopy is a simple sampling method for viewing cells. However, classifying and segmenting individual neuronal cells in microscopic images requires expert knowledge and is highly time-consuming. Numerous studies on neural cell segmentation used deep learning methods such as FCN, U-Net, and DeepLab. However, most deep learning methods cannot perform well enough due to the unique patterns and irregular morphology of the neural cells. The performance is also constrained by the limited annotated image data.

Our project classified the neural cell type with a hybrid voting model and builds a novel U-Net [1] Residual Transformer (URT) model to segment and count individual neuronal cells on microscopic images. Transfer learning was adapted to deal with the inadequacy of the data.

1.2 Related Work

Methods for segmenting medical images that have been around for a long time are ineffective. Mostly based on atlases. These methods typically rely on pre-computed templates, which may not effectively account for anatomical diversity caused by variances in organ morphology, tissue removal, tumor growth, and image capture disparities. Convolutional neural networks (CNNs) have been widely employed in diverse sectors of computer vision because to their extraordinary ability to extract picture features, such as object detection [2], semantic segmentation [3], and pose estimation [4, 5, 6], among others, since the rise of deep learning. U-Net is the first to apply CNNs to the segmentation of medical images. U- Net and its versions [7, 8] have now accomplished this mission with outstanding success.

Although CNN is capable of extracting rich characteristics, they are not well-suited to encoding long-range interaction information [9], whether inside a single slice (intra-slice) or between neighboring slices (inter-slice). This information is significant in the field of medical image segmentation because the texture, shape, and size of many organs vary greatly between patients, and accurately segmenting these organs often necessitates long-range contextual information. As a result, finding a tech that can improve CNNs' ability to encode long-range features is significant.

1.3 Dataset Description

After researching on currently released datasets, the following two datasets were collected and combined:

- Sartorius [10] provides 3 neural cell types of cort, shsy5y, and astro which we aim for cell segmentation.
- LIVECell [11] published by Sartorius Corporate Research. Extract out cell type shsy5y add to dataset Sartorius. This dataset remains 7 other cell types of images.

Every data entry includes a raw microscopic image and the annotation of each cell instance in COCO format. The number of different annotated images is shown in the table 1.

Table 1: Dataset

	number
cort	320
shsy5y	859
astro	131
others	4535

For each gray-scale microscopic image, the COCO instance's annotation was merged into one annotation mask as the segmentation label. The annotation mask has the same image size as the original image, and the gray-scale is set to 0 as background, 1 as annotation segmentation. Sample images with annotation of three target types are shown in Figure 1.

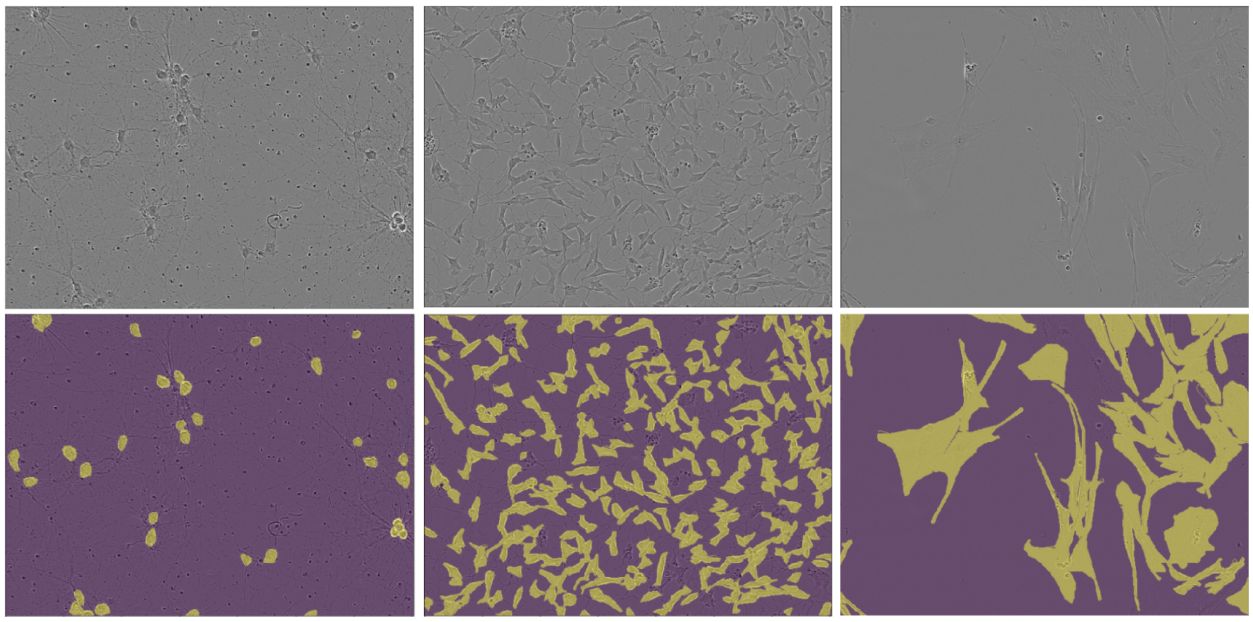

Figure 1: Sample images with annotation. From left to right: cort, shsy5y, astro

In Figure 1, we cannot only get the original cell images but its corresponding annotation images. For Cort cells, each cell is separate and full of whole space. For Shsy5y cells, they tend to be small pieces of paper full of screens. As for Astro cells, they are intertwined with each other. Considering the overlapping situation for Shsy5y and Astro, it may cause more challenges and difficult to realize a semantic segmentation function.

2 Network Architecture

2.1 Overview

Our model consists of two parts (see the Figure 2). The first part is classification and the second part is semantic segmentation.

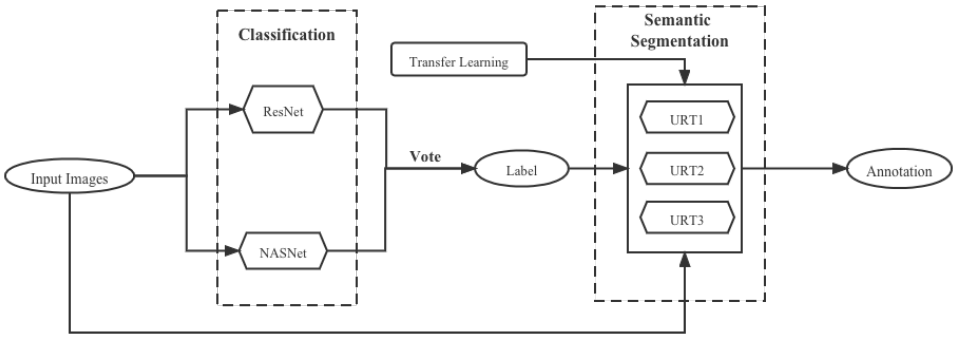

Figure 2: The Flow Chart of Model Operation

For the classification part, input images will be transferred into two sub-classification networks (InceptionResNetV2 [12] and NasNetLarge [13]). These two networks will predict images' labels respectively and vote for the final predicted label. Then, images and their predicted labels will be put into the semantic segmentation part, which is the URT model we will introduce in the following part. A URT model trained to segment this cell type of three cell types will receive each image and its label and output the predication segmentation mask.

2.2 Classification

In our neural cell images datasets, each neural cell microscopic image only contains one cell type. Therefore, for each particular image, we should first classify it by the cell type it contains. The classification phase takes a raw microscopic image and classifies it into three target cell types. After getting the classification result, the image is fed to a URT segmentation model that is specified for this cell type.

The first classifier that has been trained is the InceptionResNetV2[12], which is noted for excellent generalization performance with fewer error rates on recognition tasks. InceptionResNetV2 combines both Inception and Residual networks architectures. The residual connections allow shortcuts in the model and help train the deep neural networks without the problems caused by the vanishing gradient, which has also enabled significant simplification of the Inception blocks, which further boosts the performance. For the residual addition operation to be feasible, the input and output must have the same dimensionality after convolution. Therefore, the model introduced residual inception blocks, which utilized a filter-expansion layer (1×1 convolution without activation) to match the depth of the input.

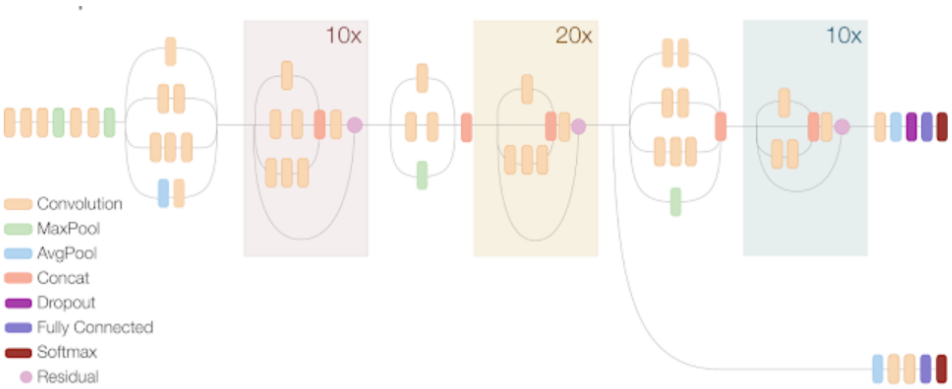

Figure 3: The structure of InceptionResNetv2

To enhance a better performance, we trained the second classifier NASNet. NASNet learns a network cell (Cell) on small data (CIFAR-10) and then migrates NAS to more complex and larger datasets by stacking copies of these network cells.

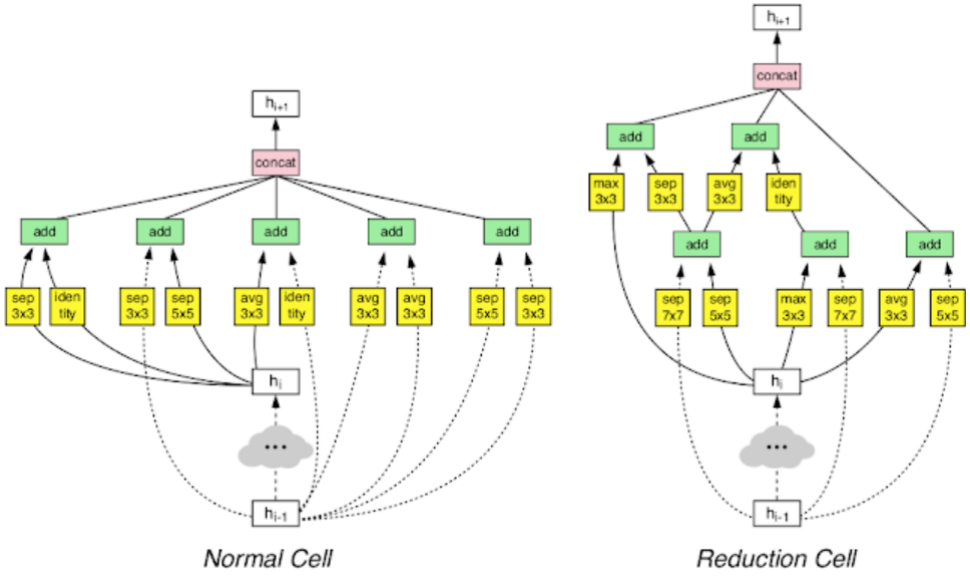

Figure 4: Two types of layers that compose NASNet architecture

After two classifiers predict the category of the given image, respectively, they will vote for the final predicted label and feed the label to the URT model of this specific cell type.

2.3 URT Model

U-Net [14], as an excellent backbone (see Figure 5), has been put into use for a long time. U-Net shapes like a letter "U" and that is why we call it a U-Net model. The greatest advantage of U-Net is that it can get higher accuracy than other classical models like FCN-VGG, FCN-ResNet. However, the accuracy of U-Net performing on biomedical image segmentation highly depends on the number of training data, which means the more data input, the higher accuracy can get. However, in the biomedical field, it is extremely expensive to retrieve enough data for training.

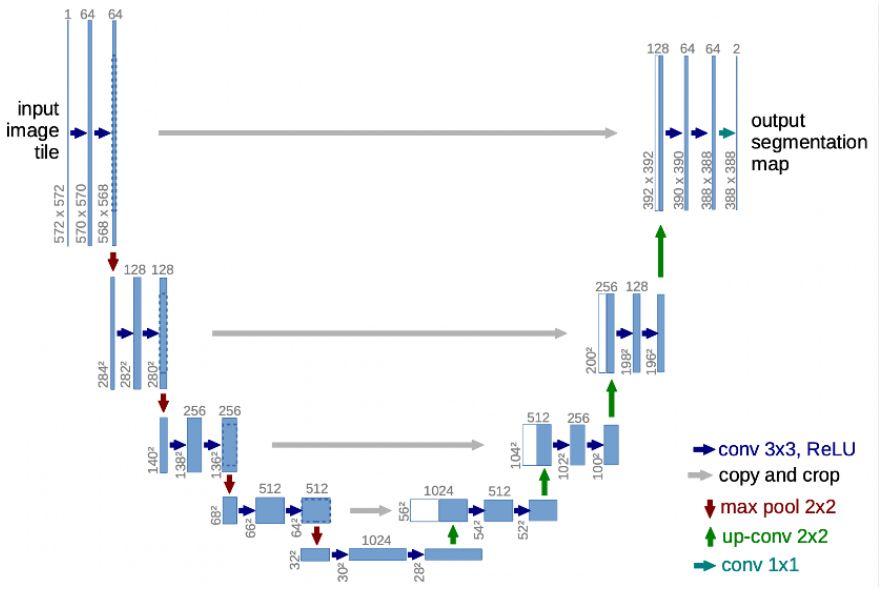

Figure 5: The Backbone of U-Net

As a result, we introduced three methods to get higher accuracy based on the U-Net backbone and named it URT (U-Net-Residue-Transformer).

The URT model consists of three parts, including a backbone, some modified modules, and a transfer learning trick. (1) URT heritages the backbone from the U-Net model, which performs excellently on biomedical image segmentation tasks. (2) URT introduces modified modules, which consist of the Residue part and attention part (Transformer). The former is used for avoiding over-fitting when the network gets deep and the latter is for getting a higher accuracy by building a relationship between each part of an input image. (3) URT also employs transfer learning to get enough prior knowledge and avert under-fitting.

2.3.1 Backbone

The network backbone is illustrated in Figure 6. It consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of three 3x3 convolutions, each followed by a rectified linear unit (ReLU), a 2x2 max pooling operation with stride 2 for down-sampling, and a Residue module to connect each convolution layer. At each down-sampling step, we doubled the number of channels. Then we used batch normalization to avoid gradient explosion or gradient vanishing followed by a transformer encoder (except the first step). Every step in the expansive path consists of an up-sampling of the feature map followed by a 2x2 convolution (“up-convolution”) that halves the number of channels, a concatenation with the correspondingly cropped feature map from the contracting path, two 3x3 convolutions, each followed by a ReLU and Residue modules to connect each convolution layer. In addition, a transformer decoder will be added at last for each step. At the final layer, a 1x1 convolution is used to map each 64-dimensional feature vector to the desired number of classes. As a result, the network has 77 layers.

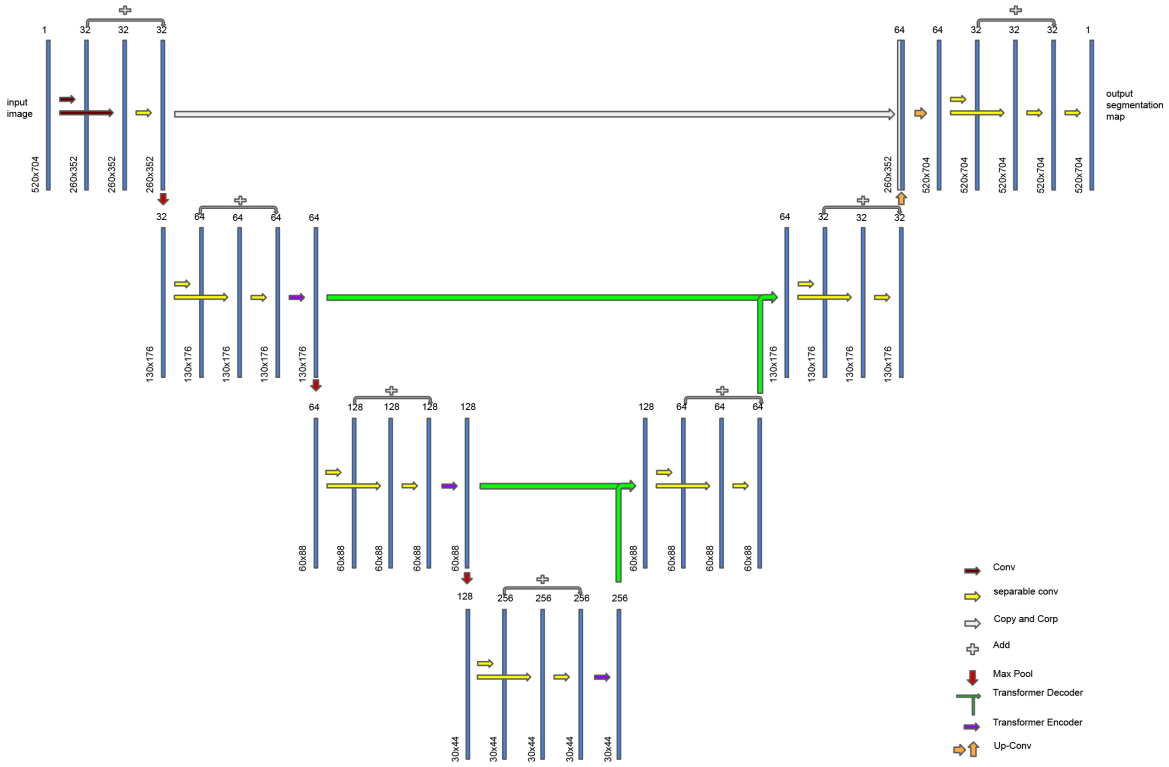

Figure 6: The Backbone of URT

2.3.2 Modified Modules

Residue Modules Residual module [15] is used for averting over-fitting. Its structure is illustrated by Figure 7.

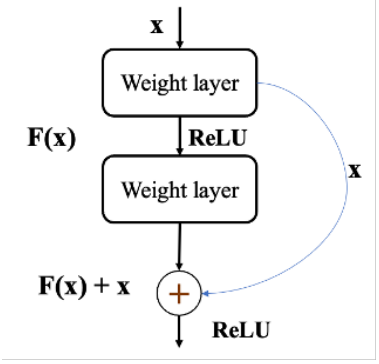

Figure 7: A Residue Module Sample

For a MLP model, we know

$$z^{(l)} = H(a^{l-1}) = a^{l-1} + F(a^{l-1}) \tag{1}$$

,where H(.) denotes the function we want to fit, F(.) denotes the Residue function. Our mission is let

$$F(a^{l-1}) \to 0 \tag{2}$$

Then, for each adjacent layer, we can get

$$a^{l-2} = a^{l_1} + \sum_{i=l_1}^{l_2-1} F(a^i) \tag{3}$$

For back propagation process, the partial derivative of loss ϵ can be represent as:

$$\frac{\partial \epsilon}{\partial a^{l_1}} = \frac{\partial \epsilon}{\partial a^{l_2}} \frac{\partial a^{l_2}}{\partial a^{l_1}} = \frac{\partial \epsilon}{\partial a^{l_2}} (1 + \frac{\partial}{\partial a^{l_1}} \sum_{i=l_1}^{l_2-1} F(a^i)) \tag{4}$$

Therefore, it can release the problem of gradient vanishing.

2.3.3 Attention Modules

The attention module [16] is to connects each part of an image horizontally and is illustrated in the following figure.

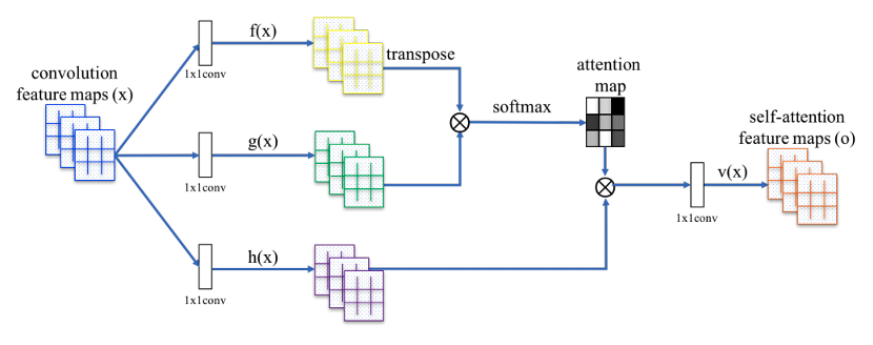

Figure 8: A Self-Attention Module Sample

We noticed there are three subsets here for an image, which can be represented as three sets of weights. Their corresponding names are "Query", "Key" and "Value". The encoder's operation process is shown below: (1) Calculate a score for each input vector:

$$Score = q * k$$

(2) Score Normalization:

$$Score = \frac{Score}{\sqrt{d_k}}$$

(3) Activation for score:

$$score = softmax(score)$$

(4) Calculate corresponding score:

$$score_i = score_i \times v_i$$

(5) Get the result z:

$$z = \sum v$$

For the decoder process, it is the same as the encoder module except for additional encoder-decoder attention. For encoder-decoder attention, its Q comes from the last decoder's output, K and V root from the corresponding encoder module's output. Furthermore, we used Multi-Head attention here to mimic an ensemble of n different self-attention modules working together. The operation process of multi-head attention can be briefly discredited as follows: (1) Input data X into corresponding self-attention modules (like 8 self-attention modules), and we can get 8 weighted feature matrices. (2) Group these feature matrices into a large feature matrix (3) Feed forward to a fully-connected layer and get the result.

2.3.4 Transfer Learning

Before training URT, there were 591,731 parameters in total, including 588,083 trainable parameters. However, considering only limited training data (several hundred) cannot make this huge number of parameters converged, we utilize transfer learning from other similar data sets. Transfer learning is a tech to get knowledge from other things and use it in your task. The reason why it can perform well is that for a convolution the deeper layers' filter can get the geometric or texture features of input images and we can regard each image as a combination of these basic components. If our model has acquired some knowledge like what is "line" or what is "circle", it is easier to train our model to detect various "cells".

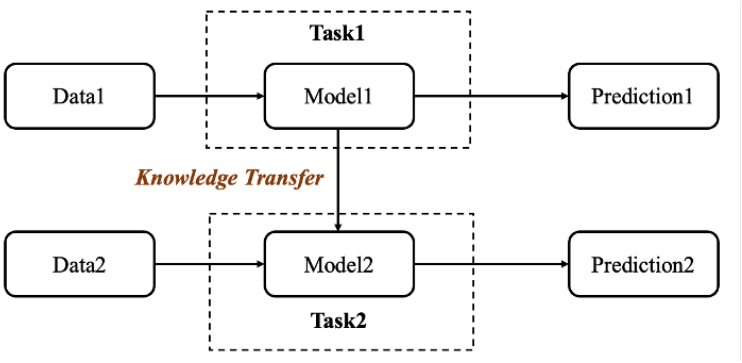

Figure 9: Transfer Learning Module

Transfer learning firstly trains the network using other data sets and makes it converge. Then these trained parameters can be set as the initial weights for your training set. In fact, you have several policies these parameter assignments: (1) Forget the last layer's weights (like set them all zero) and train your model (2) Save all transfer learning weights and train your model If the data set you choose is like your training set, the second way is a good choice, or you should use the first option. For URT, we employed data set from LIVECell, which has a myriad of biomedical images and corresponding annotations. After training with near 4000 images, the model can get converge and can be completely transferred to our task.

3 Experiment

3.1 Experiment Setup

All the experiments are conducted on Google Colab [17]. For the classification models, construct the InceptionResNetV2 and NasNetLarge with Keras, replace the original top dense layers with one dense layer of three output for 3-classification. Load the pre-trained weight of ImageNet [18], freeze the base of the model, and only train the last output layer.

(1) Loss Function
The output of the segmentation model is a feature map with the same shape as the input image. As described in section 1.3, for a preprocessed label annotation image L, the value of each pixel is 1 if L_{ij} is on a cell segmentation otherwise 0. The model output P predicts the probability of each pixel P_{ij} being a cell segment. Use binary cross-entropy as the loss function for each pixel:

$$Loss = -\sum_{ij} L_{ij} \log P_{ij} + (1 - L_{ij}) \log(1 - P_{ij})$$

(2) Cross-Validation for Classification
Use 5-fold training, each train-validation split 20% of each class into validation set. Train two classification models on the same dataset split respectively. The voting model that combines the output probability tensor of two models, output the max probability index as the voting model classification result. For both models, the best training parameters are Adam Optimizer, set learning rate of $\mathbf{10^{-3}}$, $\mathbf{5 \times 10^{-4}}$, and $\mathbf{10^{-3}}$ downgrading each learning rate train 5 epochs.

(3) Cross-Validation for Semantic Segmentation.
Train three URT models on 3 different cell types. Adapt the idea of transfer learning, first pretrain URT model on the LIVECell dataset to get a model that possesses universal cell segmentation ability, then fine-tune model on the target neural cell type. Pretrain URT on LIVECell, the training accuracy of validation accuracy always tie since it is a diversity dataset with 7 other different cell types. Train URT with the whole LIVECell dataset achieves train accuracy of 90.05%. Then for each cell type, load the pre-trained weight of URT, fine-tune the model on one of three neural cell types with small learning rates. The best training parameters are Adam Optimizer, set learning rate of $\mathbf{10^{-4}}$, $\mathbf{5 \times 10^{-5}}$, and $\mathbf{10^{-5}}$ downgrading each learning rate train 5 epochs.

3.2 Classification Results

The confusion matrix of InceptionResNetV2, NasNetLarge and Voting are in figure 9. Two models can achieve validation accuracy of 98.62% and 98.32% respectively. However, the voting model that combines two models can achieve 99.32%.

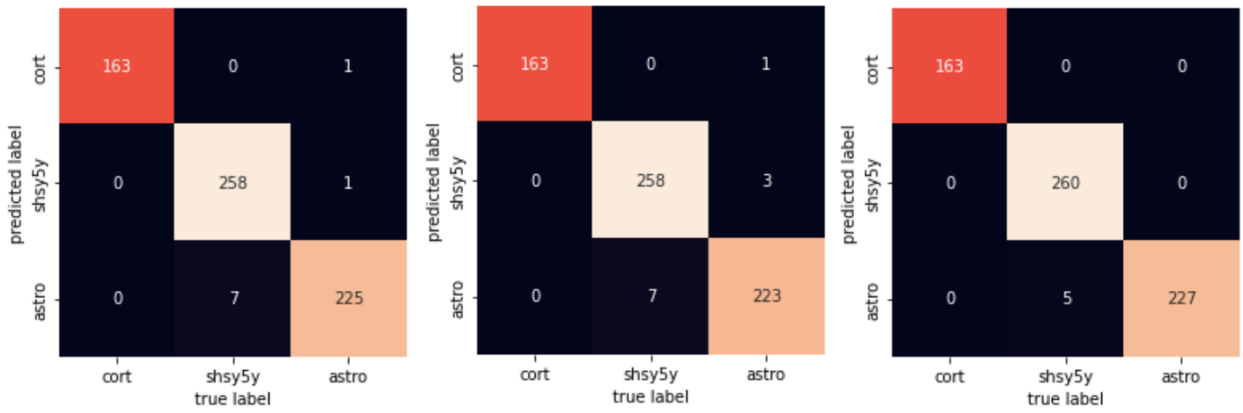

Figure 10: Confusion Matrix of InceptionResnet, NasNet and Voting

According to Figure 11, we could know our classification model performs well on each category because the majority of samples are located on diagonal positions. To know more about the classification model's performance on each category, we calculate the precision and recall of each category to prove the robustness of our classification model and we can see the Figure 12 below.

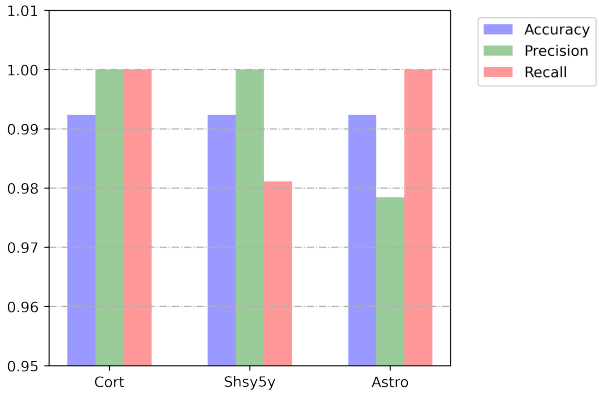

Figure 11: Performance of Classification Model

Obviously, all categories have more than 95 percent precision and recall. More exactly, for Cort and Shsy5y, their precision is 100 percent, which means Our model has high reliability in these types of cells. In addition, considering Cort and Astro's high Recall, we can say our model can detect all these types in our test data.

3.3 Segmentation Results

We train the baseline models in the same way described in section 3.1. In the table, we compare the URT model with other segmentation models.

Table 2: Validation Accuracy

	cort	shsy5y	astro
FCN-VGG16	51.81%	43.42%	40.07%
FCN-ResNet	63.17%	53.43%	49.73%
DeepLab	81.15%	72.21%	70.13%
U-Net	97.73%	71.32%	75.67%
URT	**98.77**%	**90.20**%	**87.13**%

From data from table 2, we can find U-Net backbone model performs better than other models and URT can be the best among them. In addition, due to the extremely deep layer, VGG16 cannot fit for small-samples tasks and it is easier to be over-fitting. Due to ResNet can overcome some negative impacts of over-fitting, it still can get high accuracy.

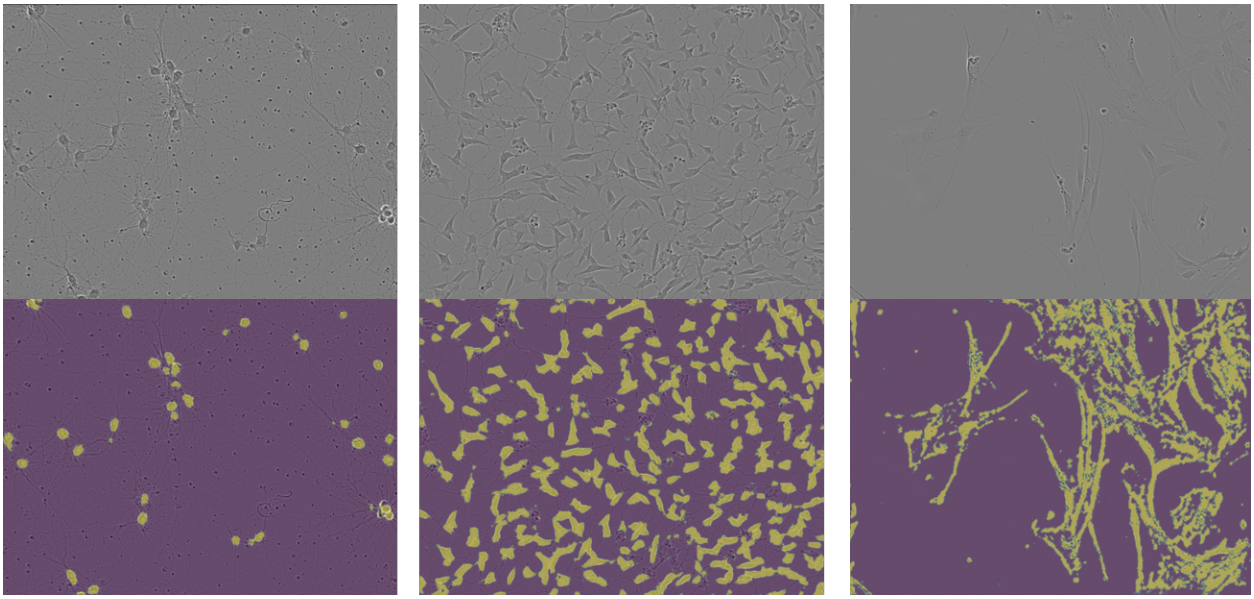

Figure 12: URT Output Segmentation. From left to right: cort, shsy5y, astro

4 Conclusions

In this project, we have made several related conclusions about our URT model. Firstly, we have found U-Net based backbone can perform better than other classical backbone models. Then, URT performs best among all models we have tested. The reason why URT can perform excellently is due to a fitful backbone, two modules for curbing over-fitting as well as under-fitting, and the key point is self-learning tech has constructed a connection between each part of an image, which contributes to a higher accuracy at last.

5 Future Work

In the future, we may stretch our research in two aspects. Firstly, the URT model has excellent segmentation ability, however, it is not meant for instance segmentation. In the further, we may use the traditional computer vision method to do boundary detection on the output segmentation mask. Or change the network architecture and loss function to suit the problem of instance segmentation. In the figure 8, Use URT to get mask output ,then use cv2.connectedComponents to separate instance. Next, due to being easily exposed to the under-fitting problem, we want to try some data augmentation tech like flip or rotate an image to generate more samples or we can introduce a generative adversarial network(GAN) to generate much more suit samples for training.

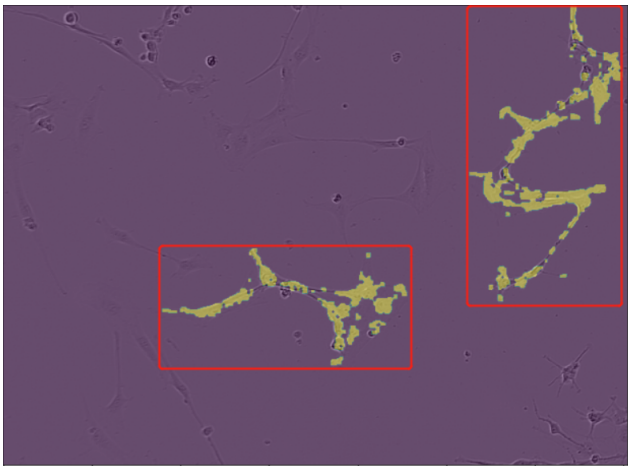

Figure 13: Instance Segmentation on URT Output Mask

References

[1] Jieneng Chen. Transunet: Transformers make strong encoders for medical image segmentation, 2021.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[4] Zhe Wang, Liyan Chen, Shaurya Rathore, Daeyun Shin, and Charless Fowlkes. Geometric pose affordance: 3d human pose with scene constraints. *arXiv preprint arXiv:1905.07718*, 2019.

[5] Zhe Wang, Daeyun Shin, and Charless C Fowlkes. Predicting camera viewpoint improves cross-dataset generalization for 3d human pose estimation. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020.

[6] Haoyu Ma, Liangjian Chen, Deying Kong, Zhe Wang, Xingwei Liu, Hao Tang, Xiangyi Yan, Yusheng Xie, Shih-Yao Lin, and Xiaohui Xie. Transfusion: Cross-view fusion with transformer for 3d human pose estimation. *arXiv preprint arXiv:2110.09554*, 2021.

[7] Foivos I Diakogiannis, François Waldner, Peter Caccetta, and Chen Wu. Resunet-a: A deep learning framework for semantic segmentation of remotely sensed data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 162:94–114, 2020.

[8] Huimin Huang, Lanfen Lin, Ruofeng Tong, Hongjie Hu, Qiaowei Zhang, Yutaro Iwamoto, Xianhua Han, Yen-Wei Chen, and Jian Wu. Unet 3+: A full-scale connected unet for medical image segmentation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1055–1059. IEEE, 2020.

[9] Xiangyi Yan, Hao Tang, Shanlin Sun, Haoyu Ma, Deying Kong, and Xiaohui Xie. After-unet: Axial fusion transformer unet for medical image segmentation. *arXiv preprint arXiv:2110.10403*, 2021.

[10] https://www.kaggle.com/c/sartorius-cell-instance-segmentation/data.

[11] https://github.com/sartorius-research/livecell.

[12] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.

[13] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2018.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[17] https://colab.research.google.com/notebooks/intro.ipynb.

[18] https://www.image-net.org/.