CS230 Deep Learning **Winter 2018**

Project Title: Deep Cell: Automating cell nuclei detection with neural networks
Project Category: Computer Vision
Team Members: Cristian Bartolome Aramburu (cbartolm), Ashwin Ramaswami (ashwin99), Yiguang Zhang (yiguang)

I. Introduction

The purpose of this project is detecting nuclei in various types of microscopic images, the topic of the 2018 Kaggle Data Science Bowl. Finding nuclei is a very important and crucial task. According to Eroom's law, recently the cost of developing new drugs has been steadily increasing. Solving this problem will help to introduce more automation into the time-consuming process of identifying nuclei. Identifying nuclei is often the starting point, but a large bottleneck, in the process of analyzing cells during drug tests. An effective machine-learning solution will shorten drug tests, making it easier to treat a wide variety of diseases.

a) Challenges

A large challenge of this project is that nuclei from images vary in color, shape, and other characteristics; it is important to be able to make a generalizable model that works well with different types of images, especially those taken from different kinds of instruments, microscopes and stains. Classical image processing often fails to generalize well with unusual shapes. Therefore, we plan to use neural networks to improve the robustness of this fundamental tool in biology research.

II. Details on the Dataset

We are using a dataset of around 25,000 images from the Broad Institute, which consists of pictures from microscopes and labels of the coordinates of the human-identified nuclei. The images were acquired under a variety of conditions and vary in the cell type, magnification, and imaging modality (brightfield vs. fluorescence).

Each image is represented by an associated "ImageId". Files belonging to an image are contained in a folder with this "ImageId". Within this folder are two subfolders:

- "Image" contains the image file.

- “Masks” contains the segmented masks of each nucleus. This folder is only included in the training set. Each mask contains one nucleus.

III. Approach and summary of current implementations:

a) Statistical Method (Otsu’s Method)

First, we decided to explore a non-neural network based method in order to see the processes and limitations of more traditional approaches. Otsu’s method assumes that the image contains two classes of pixels (cell pixels and non-cell pixels). It computes a threshold that minimizes the intra-class variance for both classes. As it can be seen in the notebook, the method works fine only for a very limited set of pictures (gray pictures mainly) (Link to Colaboratory notebook).

b) U-Net Convolutional Neural Network

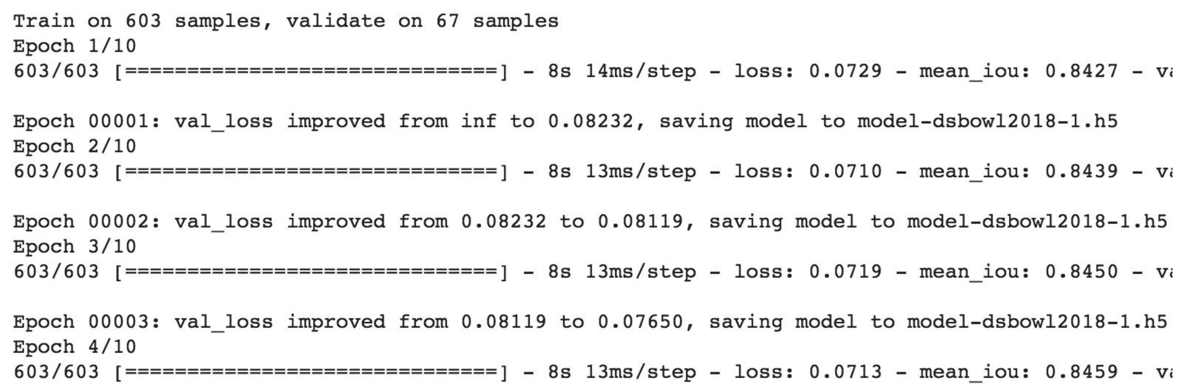
Next, we implemented a first version of U-Net. This version is based on an open-source Kaggle Kernel. The network is an FCN (fully convolutional network); basically, it has convolutional / pooling layers throughout but does not have any fully-connected layers at the end. This way, each input pixel can be mapped to an output pixel, which has a value of 0 or 1 -- 0 if no nucleus is detected, and 1 if a nucleus is detected. All the details about this encoder-decoder architecture can be viewed in the U-Net paper.

In a first stage, the loss function we are using is the Dice Coefficient. The Dice score is not only a measure of how many positives you find, but it also penalizes for the false positives that the method finds, similar to precision. It can be computed as follows:

$$Dice\ score\ = \frac{number\ of\ true\ positives}{number\ of\ positives + number\ of\ false\ positives}$$

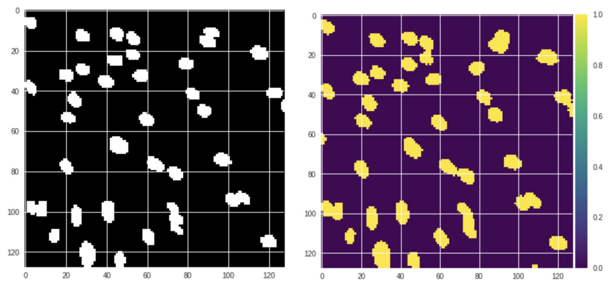
One of our future goals is to implement the metric used in the competition instead of the Dice Coefficient. That is, the Mean IoU. In fact, we started implementing a first version but we didn’t get the behaviour expected (as the metric we defined, it scores the background too). More work will be done in this direction.

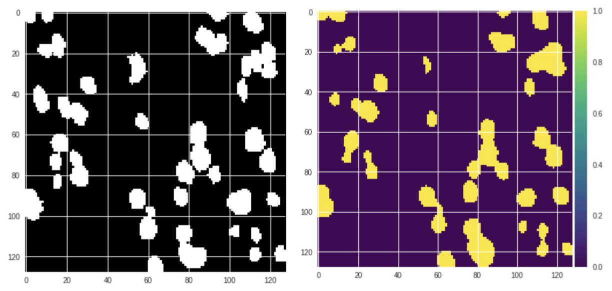
Regarding the organization of the data available (670 images), we used around 91.3% of it as training set and 8.6% as dev set. Here are some results after some epochs: (Link to Colaborary Notebook: U-Net). One interesting outcome is the consistent decreasing trend of the loss function (with some irregularities, as expected from a mini-batch optimizer), which is coherent with theory.



```
Train on 603 samples, validate on 67 samples
Epoch 1/10
603/603 [==============================] - 8s 14ms/step - loss: 0.0729 - mean_iou: 0.8427 - va

Epoch 00001: val_loss improved from inf to 0.08232, saving model to model-dsbowl2018-1.h5
Epoch 2/10
603/603 [==============================] - 8s 13ms/step - loss: 0.0710 - mean_iou: 0.8439 - va

Epoch 00002: val_loss improved from 0.08232 to 0.08119, saving model to model-dsbowl2018-1.h5
Epoch 3/10
603/603 [==============================] - 8s 13ms/step - loss: 0.0719 - mean_iou: 0.8450 - va

Epoch 00003: val_loss improved from 0.08119 to 0.07650, saving model to model-dsbowl2018-1.h5
Epoch 4/10
603/603 [==============================] - 8s 13ms/step - loss: 0.0713 - mean_iou: 0.8459 - va
```

Finally, we show some of the first predictions we obtained with U-Net after only 10 epochs:





In these two pairs of images, the left picture is the ground truth mask and the right picture, the predicted labels from U-Net. These results confirm that U-Net has been correctly implemented as it captures the overall position of nuclei. More work is needed to improve the accuracy close to the edges of nuclei, among others.

IV. <u>Future work:</u>

- Implement techniques of data augmentation to make the model more robust.
- Use transfer learning from existing image classifiers. The following model looks especially promising as it is trained on segmentation of overlapping cells: https://cs.adelaide.edu.au/~carneiro/isbi14_challenge/dataset.html)
- Implement a correct Mean IoU evaluation metric in Keras
- Look at other CNN architectures that could be of interest for image segmentation. Compare them using a single metric: our own Mean IoU implementation.
- Hyperparameter tuning
- Explore Facebook's Detectron library (released in January 2018): https://github.com/facebookresearch/Detectron