CS230

Nucleus Detection Using Deep Learning

Burak Bartan
Department of Electrical Engineering
Stanford University
bbartan@stanford.edu

James Pao
Department of Electrical Engineering
Stanford University
jpao@stanford.edu

Benjamin Knapp
Biophysics Graduate Program
Stanford University
bknapp8@stanford.edu

Abstract

The machine-automated detection of cell nuclei can assist researchers in rapidly iterating through innovative drugs and treatments on the path to curing cancer. We investigate applying deep learning methods to cell nuclei segmentation in tissue images taken over a range of modalities and conditions. The most optimal model we developed was a modified U-Net architecture [1], which retains feature maps produced in the encoding phase to use during the decoding phase, allowing for better localization by the network. In this model, instead of up-convolutions during the decoding phase, we use up-sampling which reduces the number of parameters trained by the network. The model performs reasonably well, achieving a Sørensen-Dice coefficient of 0.904 on a test set held out from the training data set, compared to a coefficient of 0.817 for a simple convolutional network.

1 Introduction

According to the World Health Organization, cancer is the second-leading cause of mortality worldwide, being responsible for 1 in 6 deaths [2]. Furthermore, it is expected that the number of new cases is expected to rise by 70% over the next two decades. In order to combat this persistent issue in an effective and cohesive manner, concentrated effort must be placed into the development of innovative treatments, into the improvement of precision cancer medicine (i.e. understanding biological characteristics to allow for appropriate treatments), and into the increased understanding of tumor biology [3].

To facilitate these efforts, there have been many studies investigating the automation of biomedical image segmentation. The ability to perform automatic and accurate biomedical semantic segmentation, such as the ability to detect cell nuclei in images, would allow researchers and doctors to study the effects of treatments and drugs at a cellular level without expending too much manpower or time in manually segmenting images.

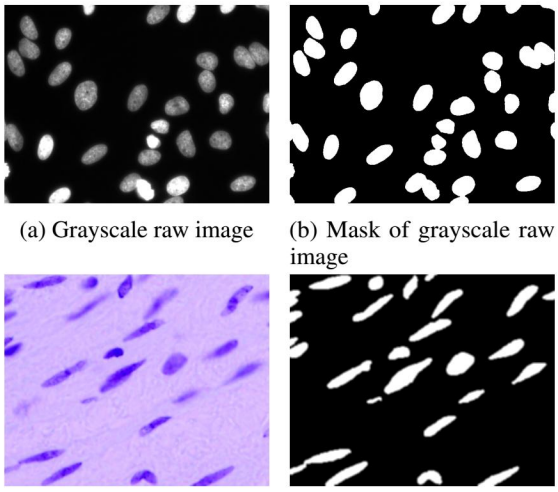
In recent years, there have been promising advancements in the development of convolutional neural networks that can learn to "understand" images for a variety of applications such as object detection and image segmentation. As there is much potential in deep learning becoming a real and effective solution to many semantic segmentation tasks, we investigate using deep learning architectures for the problem of automated nuclei detection.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

There are many works on nuclei segmentation using deep learning, as well as a host of other image processing techniques. In [1], the authors use k-means clustering for nuclei segmentation. In [2], the authors propose a method based entirely on image processing techniques such as morphological operations and watershed transformation for nuclei segmentation in Stained Breast Cancer Histopathology Images. In this work, we focused on deep learning approaches for nuclei segmentation. The work in [3] uses a simple convolutional neural network (CNN) for nucleus detection, a model which does not use any spatial detail preservation such as feature map retention and concatenation. [4] proposes a model based on CNN and compressed sensing (CS), where they use CS for output encoding. The state-of-the-art deep learning models for image segmentation are SegNet [5] and U-Net [6] both of which have a similar encoder-decoder structure which also employ some method of spatial information preservation. SegNet uses an encoding phase which retains the indices from max-pooling operations, which are used to up-sample during the decoding phase. U-Net is similarly constructed, but instead retains feature maps after convolutions to concatenate to up-convolved feature maps from the decoding phase.

3 Dataset and Features



(a) Grayscale raw image (b) Mask of grayscale raw image

(c) RGB raw image (d) Network 4

Figure 1: Example training set images and their masks

The training and test sets were provided by the Kaggle competition sponsored by the information technology consulting firm Booz Allen Hamilton [8]. The training set is composed of 670 images of human cell samples with nuclei stained for fluorescent imaging (Figure 1). Each image is taken in a different modality (fluorescence, staining type, camera, magnification, etc.), and is accompanied by binary image masks for each nucleus that have been identified by a domain expert (Figure 1). The test set was composed of 65 images without masks. We chose to resize images as 128X128 (by convention) during training to maintain input sizes.

To improve upon our initial testing (Methods, Results; Figure 6, Table 1), we produced augmented datasets for increasing the number of training samples. These methods included mirroring images and their masks about vertical and horizontal axes, as well as converting RGB images to grayscale and performing inversion. The motivation for these augmentations was that a majority of the dataset was grayscale images, and so the neural network tended to overfit to the those image types. Furthermore, particular staining methods produced dark nuclei (against background), while other methods produced bright nuclei, thus increasing the problem's complexity.

4 Methods

During training, we chose to use Adam optimization with binary cross entropy as our loss function (Equation 1)[9]. As our metric for performance against our ground truth (masks), we used the Sørensen-Dice coefficient, the most commonly used metric for biomedical image segmentation performance [11].

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \tag{1}$$

$$Dice = \frac{2(A \cap B)}{A + B} \tag{2}$$

4.1 Basic CNN and UNet

As our baseline model, we implemented a simple convolutional neural network with no retention of spatial details, details that are used in state-of-art models such as SegNet and U-Net. The architecture we used consists of 4 double-convolutions using a kernel size of 3x3 with 3 max-pooling operations interspersed between the double-convolutions. We then followed this with 3 upsampling operations and double-convolution pairs. All convolutional layers thus far use the rectified linear unit (ReLU) as their activation function. Lastly a convolutional layer using sigmoid as the activation function outputs a predicted segmentation image with the same dimensions as the input which can be thresholding for comparison to the provided nuclei masks.

As our primary model, we investigate the U-Net architecture proposed by Ronneberger, et al., which uses a encoding phase where an input image is convolved and down-sampled, with post-convolution feature maps retained to be used in the decoding phase to provide localization ability for effective segmentation. In the U-Net architecture, the decoding phase is a series of operations consisting of an up-convolution, the concatenation of a feature map from the encoding step to the result of the up-convolution, and a double-convolution. The final step is an convolutional layer outputting the segmentation map with the same dimensions as the input image. Figure 2 shows the architecture used by Ronneberger, et al., which is the model we used to develop the model we use herein this work.

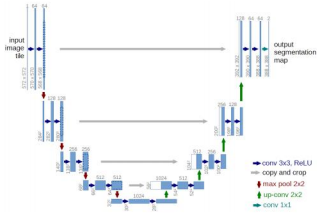


Figure 2: UNet architecture [1]

We first implement a model following closely the architecture from Ronneberger, et al., where we have 4 double-convolutions using a kernel size of 3x3 with 3 max-pooling operations with pool window sizes of 2x2 and stride of length 2 interspersed between the convolutions. Then we perform 3 up-convolutions using a window size of 2x2, each followed by a double-convolution using the same hyperparameters as the encoding convolutions. This architecture gives us a total of 322,969 trainable parameters.

Next, we implement a model which utilizes the idea presented by the authors of the SegNet architecture, which is to reduce the number of parameters of the network by removing the up-convolutional operations in the decoding phase. The SegNet architecture retains the max-pooling indices during the encoding phase and uses those indices to "up-sample" during the decoding phase, which provides the ability for localization without having to train weights for the up-convolution.

We use this idea to implement a U-Net model which uses simple up-sampling instead of up-convolutions during the decoding phase, which reduces the number of parameters trained by the network to 274,417. This should in theory reduce any potential overfitting by the network, as well as reduce the time needed to train the network.

4.2 FCN Based on ResNet50

ResNet50 consists of residual blocks (Figure 3) that have skip connections, which help with vanishing and exploding gradients problem. Having skip connections makes it possible to train much deeper networks more easily.

Our model is a ResNet50 based fully convolutional network (FCN) with connections from the last 32x32, 16x16, and 8x8 layers of the ResNet50 as in Figure 4. This type of model is mentioned in [12] We initialized the weights using ResNet50 weights pre-trained on ImageNet (transfer learning).

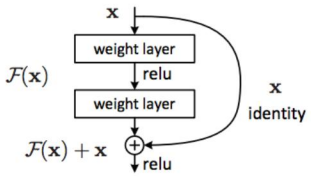


Figure 3: UNet architecture

Even though ResNet50 weights were optimized for a different task on ImageNet, transfer learning still helps as in it leads to a faster convergence. However, since the pre-trained weights were tuned for a different task, it won't be able to outperform our best method.

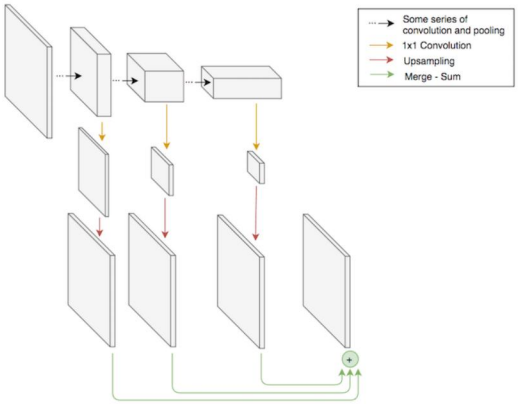


Figure 4: FCN architecture (on top of ResNet50)

4.3 Data Augmentation

In our optimized UNet, we found during both validation and testing that dark nuclei in RGB images were particularly challenging to predict. Dark nuclei were generally regarded as background in our predictions, indicating that the network was learning primarily from bright nuclei. To improve performance of our network on these image types, we augmented data by performing both grayscale inversions and mirror operations (Datasets and Features).

5 Experiments/Results/Discussion

Our Dice coefficient results for our developed models (all trained for 200 epochs) are presented in Table 1, which shows the results for the training, validation, and test sets. In the table, we can see that the baseline model (simple CNN) performs the worst on the test set, achieving a Dice coefficient of 0.8168. The model we implemented using up-sampling instead of up-convolutions is labeled as "Optimal U-Net", and it achieves a test Dice coefficient of 0.9037. The U-Net model implemented using up-convolutions is labeled as "Upconv U-Net", and achieves a test Dice coefficient of 0.9048. It can be seen that the performance of the up-sampled U-Net and the up-convolved U-Net are very similar, and both perform reasonably on the challenging problem of nuclei detection across a range of imaging modalities and conditions. We designate the up-sampled version as our "optimal" design since it provides highly comparable performance to the up-convolved version, but has approximately 50,000 less parameters. The ResNet50 model achieves better performance than our simple CNN baseline model, but does not perform comparably to any U-Net implementation. This is likely due to the fact that ResNet50 is trained on ImageNet, so the weights used for our transfer learning approach are not well suited to the cell nuclei semantic segmentation task at hand. Training and validation scores are also presented for the optimal model trained on two augmented data sets, the first using mirrored images and the second using modified grayscale images. Figure 5 shows the training progress of all the methods we experimented with.

	Baseline	Optimal U-Net	Upconv U-Net	Data Aug 1	Data Aug 2	ResNet50
Training	0.8953	0.9260	0.9664	0.9332	0.9344	0.8897
Validation	0.8429	0.8955	0.9045	0.8992	0.8929	0.8629
Test	0.8168	0.9037	0.9048	N/A	N/A	0.8539

Table 1: Dice score performance comparison of different methods

The ResNet50 model achieves better performance than our simple CNN baseline model, but

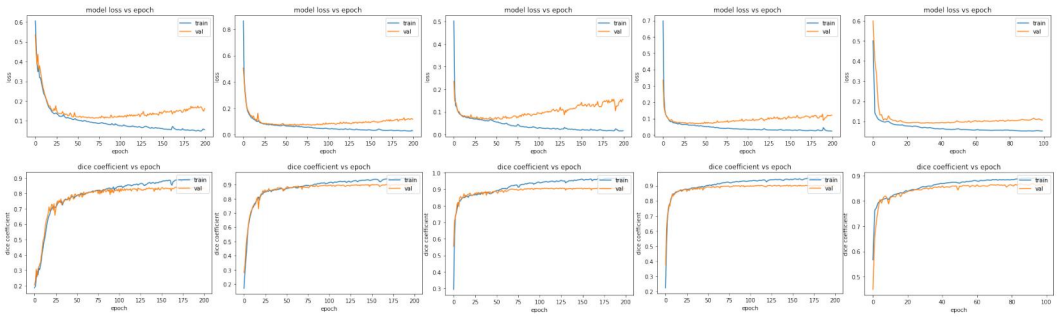


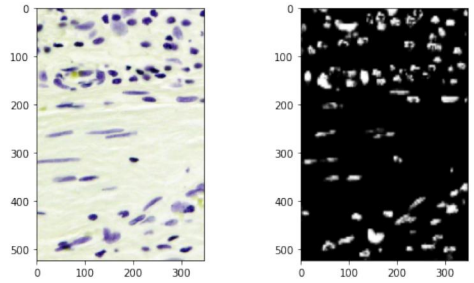
Figure 5: Training plots

We found that by performing vertical and horizontal mirroring on our training set (increasing size 3-fold), the network had a modest increase of the training Dice score of 0.95 (compared to 0.92), but performed significantly better on complex images of dark nuclei (Figure 3C). We also found that converting the training set to grayscale and adding its inverse set had the effect of also increasing performance (Dice: 0.93; Figure 6)

6 Conclusion/Future Work

We implemented a variety of deep learning models for the task of cell nuclei images, including using a simple CNN, different implementations of the U-Net architecture, and also investigating transfer learning using ResNet50 weights. We find that the U-Net architectures work reasonably well for the segmentation task, and that using an architecture with upsampling instead of up-convolving during the decoding phase gives us commensurate performance to the traditional up-convolved version, with a 50,000 parameter reduction.

This work also highlights the importance of understanding the limitations of training sets. Due to our dataset's sheer complexity, and compounded by the limited distribution of image types, we suspect that the initial training set provided is not enough by itself to produce a robust predictive network. Our initial results here indicate that further improvement of UNet performance in predicting nuclei will involve better data augmentation. Recent work has shown that us-



(a) RGB test image

(b) Optimal UNet prediction

(c) Optimal UNet prediction (mirror-augmented dataset)

(d) Optimal UNet prediction (grayscale-inverted augmented dataset)

Figure 6: Training on augmented datasets improves performance

ing Generative Adversial Networks can augment data
to produce synthetic image sets for training [3].

7 Contributions

James Pao primarily performed U-Net architecture investigation and development, performed hyper-parameter tuning, and worked on the baseline model. Burak Bartan primarily performed extensive model training, performed transfer learning using ResNet50, and worked on the baseline model. Ben Knapp primarily investigated and performed data augmentation, as well as investigating techniques such as mask erosion, and worked on the baseline model. All members contributed commensurately to the writing of this report.

Link to the code repository: https://github.com/jpao10/CS230-Project

References

[1] Ronneberger et al. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597

[2] WHO Cancer Fact Sheet: http://www.who.int/mediacentre/factsheets/fs297/en/

[3] Sarrafzadeh & Dehnavi (2015) Nucleus and cytoplasm segmentation in microscopic images using K-means clustering and region growing. *Adv Biomed Res.* 4: 174

[4] Veta et al. (2013) Automatic Nuclei Segmentation in H&E Stained Breast Cancer Histopathology Images. *PLoS ONE* 8(7): e70221. doi:10.1371/journal.pone.0070221

[5] Braz & Lotufo (2017) Nuclei Detection Using Deep Learning. *Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*

[6] Xue & Ray (2018). Cell Detection in Microscopy Images with Deep Convolutional Neural Network and Compressed Sensing. arXiv:1708.03307

[7] Badryinarayanan et al. (2015). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. arXiv:1511.00561

[8] Kaggle Data Science Bowl 2018: https://www.kaggle.com/c/data-science-bowl-2018

[9] Kingma & Ba (2014). Adam: A Method for Stochastic Optimization. arXiv:1412.6980

[10] Frid-Adar et al. (2018). GAN-based Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification. arXiv:1803.01229

[11] Taha & Hanbury (2015). Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. *BMC Med Imaging* 15:29

[12] He et al. (2015). Deep Residual Learning for Image Recognition. arXiv:1512.03385

[Resources] Amazon Web Services (AWS), Python, Jupyter, TensorFlow, Keras, Skimage