A Path to being Ground-Truth Free for Nuclei Segmentation

Submitted as a report for
DDP Phase 1
by

Aniket Bhatia
(160020001)

Supervisor:
Prof. Amit Sethi

Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai 400076 (India)

November 2020

Acceptance Certificate

Department of Electrical Engineering

Indian Institute of Technology, Bombay

The report entitled “A Path to being Ground-Truth Free for Nuclei Segmentation” submitted by Aniket Bhatia (160020001) may be accepted for being evaluated.

Date: November 2020

Prof. Amit Sethi

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this report. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Aniket Bhatia

Date: November 2020

(160020001)

Abstract

The advent of Deep Learning has revolutionized how we perceive nuclei segmentation and classification tasks. However, there is a serious lack of annotation masks in the community. Also, the very task of creating these masks is very expensive- it takes a lot of effort and time to create such masks, which are also then affected by human error due to fatigue. Hence, it is very important to ask the questions- "Can we circumvent ground-truth masks completely?"; "If yes, how?"; "If no, to what degree, and how to move close to that aim". Even if we don't want to, or can't, get free of ground-truth masks, can we somehow limit the number of data-points to be annotated? We attempt to answer these questions by proposing a novel pipeline combining adversarial training with a technique called interactive segmentation. Generation of synthetic images is done by using textures from real images while also using GANs to refine them in order to make them more realistic. Interactive Segmentation helps segment and 'create' masks for training purposes and thus makes the process of creating annotations less expensive (if this process was to be used for annotation creation). The pipeline would greatly reduce the amount of nuclei to be annotated (if), thus saving a lot of time and effort. Our pipeline would also reveal, qualitatively, which patches/images in the image pool are the most informative. The results we demonstrate do not use any ground truth masks, hence moving towards a paradigm where we don't need the ground-truths of the available images.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Prof. Amit Sethi for his constant guidance and support, for all the times he was there to tell "don’t worry it will be okay" and for all the times he would spark enthusiasm even in trying times. I would also like to thank Abhijeet Patil for always being there to help me out.

Aniket Bhatia
IIT Bombay
November 2020

Chapter 1

Introduction

Segmentation has been an all-important task in the fields of image processing and computer vision. Some of the very popular applications of segmentation include photo-editing softwares, self-driven cars and cancer detection (by analyzing the shape of the cells after segmentation). Segmentation methods have been constantly evolving; they include threshold based methods to clustering to histogram based methods. But after the advent of deep learning, we realized we could achieve much more. However, deep learning models are data-hungry and without a considerable amount of data the performance starts to deteriorate. In case of image segmentation, images are available with reasonable ease, but the annotation masks not so much. Specifically in the case of nuclei segmentation the ground-truth masks are very expensive to make. The 'expensive-ness' stems from the fact that the process of annotating is very effort and time intensive, and more often than not it involves the assistance of an expert in the field. Because of these reasons datasets in nuclei segmentation are not as abundant as we would want them to be. We ask the question, do we really need to be so dependent on those ground-truth masks? In this work we answer this question by presenting a pipeline capable of circumventing the need for the availability of ground-truth masks of real images , which we want to use. In the pipeline, we also show how to create masks in a much less cost-intensive manner. Further, we also qualitatively say which of the images/patches are more informative than the others (so, if at all the dataset is to be annotated separately, only the more informative images can be can be chosen for annotation). Our pipeline rests on two major concepts- Interactive Segmentation and Synthetic Image Generation. I present a brief introduction for the two in the following subsections.

1.1 Interactive Segmentation

Image Segmentation, the task of partitioning the image into multiple segments, is one thing which finds application in so many different computer vision and image processing tasks. The segments isolated can be the final output or could be further used for processing. Now, Interactive Segmentation hopes to provide a personalized touch to the segmentation by introducing a human in the loop- the human would put a (series of) click(s) on the object which she/he would like to be segmented (and the personalized touch stems from the fact that the user can decide where to put the next 'click' looking at where the segmentation needs to be improved). This 'clicking' to produce a mask for that object is what the interaction is about.

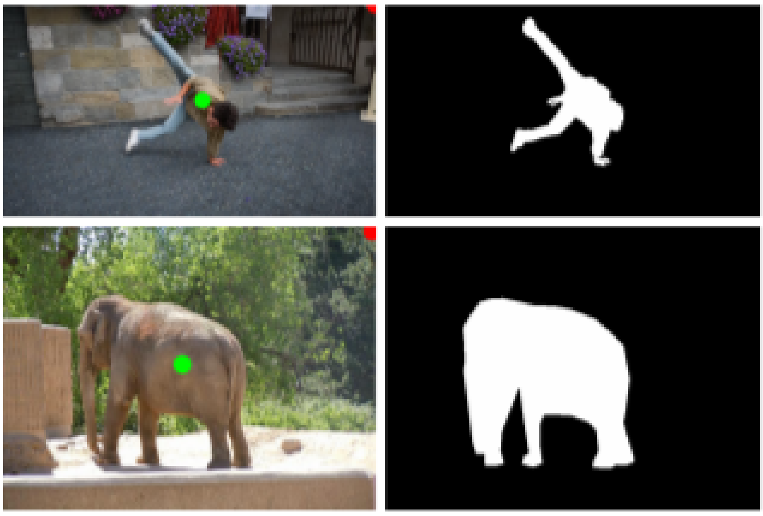


Figure 1.1: Interactive Segmentation [12]

The clicks by the user consists of foreground clicks on the object to be segmented, as shown by the green dot in figure 1.1 and background clicks outside the object of interest shown by the red dot in figure 1.1. So in essence each pixel in the object region is given the label of 'foreground' and the ones outside the object is given the label of 'background'. Effort has been made to get better masks using as less number of clicks possible. Advances in deep Learning and CNNs have led to a major improvement in performance. But this brings us to problems in training with this click-based approach. The problem arises due to the need of a clicking mechanism that would provide clicks on the object for the network to train. This has been dealt in detail in the Related Work Section.

Interactive Segmentation finds a lot of applications in artistic tools such as Photoshop. The focus of my work has been on medical images, in particular the MoNuSeg dataset, introduced in [11]. One of the goals behind me pursuing interactive segmentation has been annotating medical images. The task of creating annotations has always been a very tedious one, and with a network able to perform segmentation using a few clicks can be exploited profoundly in the annotating task.

1.2 Synthetic Image Generation

Deep Learning demands a lot of data for its performance to be accepted as reasonable, more so in the case of segmentation tasks. This again brings into the picture the cost of annotation, which is very high in terms of effort and time spent. Synthetic image generation has become an extenuating factor to combat the lack of data points. As [7] says in its quote of Richard Feynman, "*What I cannot create, I do not understand*", is very apt for deep learning and puts synthetic image generation in perspective. One of the ways of doing it has been to use generative models which learn how the data is distributed and comes up with a way to depict it. Then came the Generative Adversarial Networks (GANs) proposed by [3] which revolutionized how we see synthetic image generation. So a GAN consists of two networks, a generator and a discriminator. The task of the generator is to produce images that are realistic enough to fool the discriminator while the task of the discriminator is to discriminate if the images being input are from the pool of real images or from the pool of images being artificially generated by the generator. Both are trained together and strive to beat each other in the process.

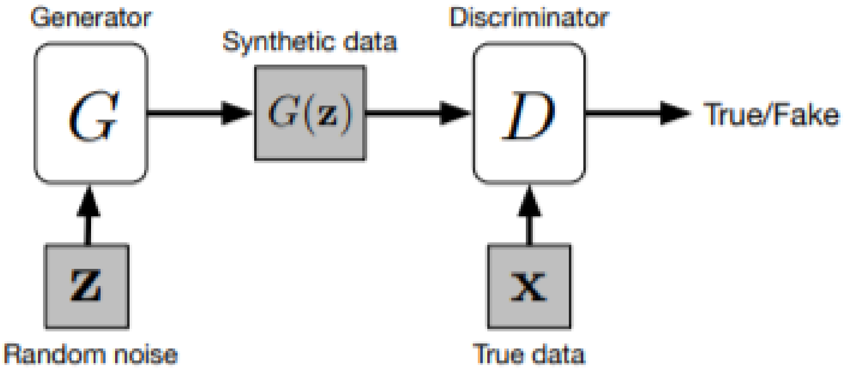


Figure 1.2: A basic view of adversarial training [7]

As shown in figure 1.2 the generator G takes input a random noise and gives the output the synthetic image G(z). Now it is up to the discriminator D to predict if the image getting input is real or fake. This identification then goes ahead to constitute a loss function. We are interested in generation of histopathology images for a nuclei segmentation task. Various methods which have been proposed related to our aim have been discussed in the upcoming sections

Chapter 2

Literature Review and Related Work

2.1 Interactive Segmentation

There has been a lot of interesting research in this field. From the outset training posed a problem for interactive segmentation networks because of the way the problem is formulated- clicks need to be put on the objects to be segmented and only the ground truth mask corresponding to that image is to be fed in the system for that training sample.

[26] proposed an approach for click-based training. In the author's notation the user interactions have been denoted by $\mathcal{S}$, $\mathcal{S}^1$ representing all user provided positive (inside the object) clicks and $\mathcal{S}^0$ representing the negative (outside the object) clicks. Euclidean distance transforms were used to convert $\mathcal{S}^1$ and $\mathcal{S}^0$ to channels $\mathcal{U}^1$ and $\mathcal{U}^0$, and for that they introduced an operator f, defined as

$$f(p|A) = min(\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2})$$

$\forall\ q \in A$, where p and q are an element in the euclidean space and A its subset of it. What this basically does is output the minimum distance from a point to the points in a set. So to generate the channels $\mathcal{U}^t$ this transform is applied to all the points (pixels) in the image and A is taken as $\mathcal{S}^t$, $t \in 0, 1$, respectively. The channels or distance transforms then basically look like bumps (or inverted bumps) centered around the clicks. The transform value is truncated to 255.

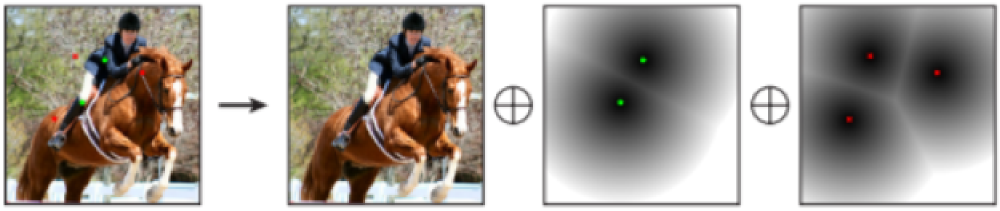


Figure 2.1: Transformation of clicks into separate channels [26]

As shown in figure 2.1 the green (positive) and red (negative) clicks are separately inculcated as two different channels. But the question arises as how to get these clicks in the first place! During the training process its impractical for a human to generate clicks for each image. The authors in [26] provide three strategies used to sample clicks and ultimately combine all the three to simulate user interaction. O has been called the set of pixels in the object (using the ground truth). They define

$$G = \{p_{ij} | (p_{ij} \in O) \cup (f(p_{ij}|O) \geq d)\}$$

which essentially means that the elements of G^c are in the background and up to a certain distance d away from the object.

The positive clicks are sampled randomly from O such that they are at least d_s distance away from each other and at least d_m distance away from the boundary. Negative clicks is where the variety kicks in. Strategy 1 is to sample points randomly from G^c and keep them distanced form each other and the boundary as described above. Strategy 2 is to sample randomly from the negative objects and Strategy 3 is to sample the first negative point randomly from G^c and then to obtain sequential points by,

$$p_{next} = \underset{p_{ij} \in G^c}{argmax} f(p_{ij}|S^0 \cup G)$$

where S^0 contains all the previously sampled negative clicks, which basically means to sample points that are greatest distance form the sampled points and those in G.

By doing this [26] establish quite profoundly on how the training and clicking can proceed seamlessly despite the apparent problems discussed at the start. Freed from the shackles of these problems, research moved further in developing how the interactive segmentation was done. [12] proposed a method titled 'Latent Diversity' which looks at interactive segmentation from a multimodal approach. The authors train two coupled convolutional networks in an end-to-end fashion as shown in figure 2.2. The first network produces a set of possible segmentations. The loss is so devised that it encourages diversity in the set. The second network is trained to select one out of these plausible segmentation masks.

The input to the network is the image X, the clicks S_p and S_n (p denoting positive clicks and n denoting negative ones), distance transforms as found from the clicks and the extracted VGG features. The distance transforms are calculated as,

$$T_p(x) = \underset{y \in S_p}{min} ||x - y||_2$$

$$T_n(x) = \underset{y \in S_n}{min} ||x - y||_2$$

T_p and T_n are single channeled intensity maps and are truncated at the value 255.

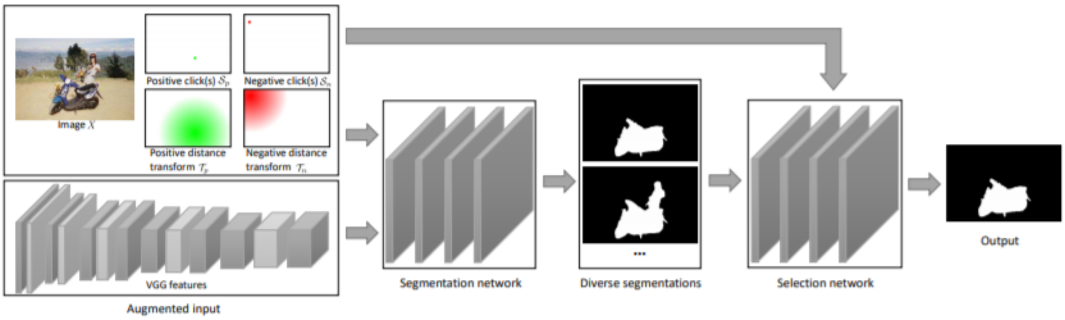


Image X
Positive click(s) $\mathcal{S}_p$
Negative click(s) $\mathcal{S}_n$
Positive distance transform $\mathcal{T}_p$
Negative distance transform $\mathcal{T}_n$
VGG features
Augmented input
Segmentation network
...
Diverse segmentations
Selection network
Output

Figure 2.2: Latent Diversity [12]

Now, let X_i be the i^th sample and Y_i be its ground truth mask. The first part of the network generates M masks f_1, f_2 , ..f_M. They calculate the hindsight loss $\sum_i \underset{m}{min}\, \ell(Y_i, f_m)$, where $\ell(A, B)$ is a loss function defined as,

$$\ell(A, B) = 1 - \frac{\sum_{\mathbf{p}} \min(A(\mathbf{p}), B(\mathbf{p}))}{\sum_{\mathbf{p}} \max(A(\mathbf{p}), B(\mathbf{p}))}$$

where $A(p)$ and $B(p)$ are the value of the masks at pixel p. They introduce some soft constraints,

$$\ell_c\left(\mathcal{S}_p, \mathcal{S}_n, B\right) = \left\|\mathcal{S}_p \odot \left(\mathcal{S}_p - B\right)\right\|_1 + ||\mathcal{S}_n \odot (\mathcal{S}_n - (1 - B))||_1$$

where $\odot$ is the Hadamard product. They define the loss as a combination of both,

$$\mathcal{L}_f\left(\boldsymbol{\theta}_f\right) = \sum_i \min_m \left\{\ell\left(Y_i, f_m\left(\mathbf{X}_i; \boldsymbol{\theta}_f\right)\right) + \ell_c\left(\mathcal{S}_p^i, \mathcal{S}_n^i, f_m\left(\mathbf{X}_i; \boldsymbol{\theta}_f\right)\right)\right\}$$

Now the next step is to select one of the M masks. They train their 'selection' network with the cross entropy loss,

$$\mathcal{L}_g\left(\boldsymbol{\phi}_g\right) = \sum_i \left(-g_{\diamond_i}\left(\mathbf{Z}_i\right) + \log \sum_{m=1}^{M} \exp\left(g_m\left(\mathbf{Z}_i\right)\right) \right)$$

where Z_i is the selection-network input.

[8] then proposed a scheme 'Back Propogating Refinement Scheme' which later became the base of the current state of the art approach. Their network has an encoder-decoder structure. They use the same strategy as in [26] to convert the user clicks to interaction maps (as they call it). They use cross entropy loss between the ground truth and the predicted mask, and they train the network via stochastic gradient decent.

Coming to the core of this scheme, the user-marked annotations provided as clicks (foreground or background) maybe incorrectly classified by the network. So this scheme strives to go back in the network, as shown in figure 2.3 and correct the prediction of the pixel-annotations provided by the user by formulating it as an optimization problem.

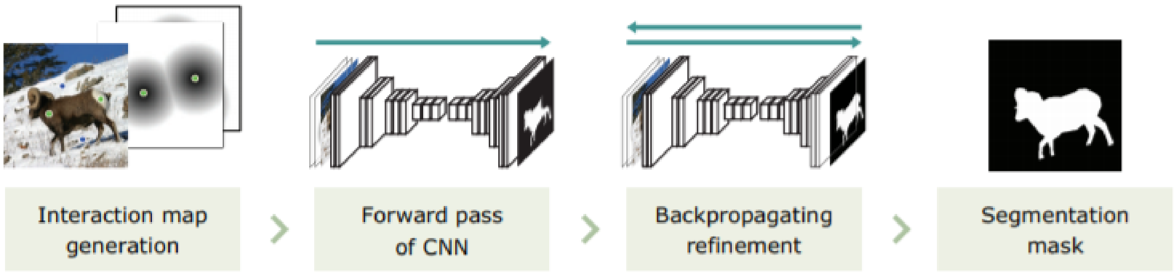


Figure 2.3: BRS flow [8]

The authors present two options, whether to fine-tune the network to yield the correct labels for the user-annotated pixels or to modify the interaction maps which may not have been a perfect representation of the user's wish leading to the final incorrect classification of those very pixels. But fine-tuning the network may lead to the network losing its learnt knowledge and hence they chose to optimize the interaction maps.

Following the notation used by [8], let z^0 represent the interaction maps which need to be optimized. They define two energies E_C, corrective energy and E_I, inertial energy. The energy function for the interaction maps is then defined as

$$E(z^0) = E_C(z^0) + \lambda E_I(z^0)$$

where λ is to match the scale difference between the two energies. Then the optimal z^0 is found by,

$$\hat{z}^0 = \underset{z^0}{argmin}\, E(z^0)$$

The corrective energy is defined as,

$$E_C(z^0) = \sum_{u \in U} (l(u) - y(u))^2$$

where U is the set of user-annotated pixels, $l(u)$ is the value of that user-annotated pixel- 1 for foreground and 0 for background, and $y(u)$ is the predicted value of that pixel. Hence E_C is a measure of how many of the user-annotated pixels were incorrect, hence the name 'corrective' energy.

The Inertial energy is to prevent too much change in the maps and is defined as,

$$E_I(z^0) = \sum_{x \in N} (z^0(x) - z_i^0(x))^2$$

where N is set of all pixels in the interaction maps, and z_i^0 denotes the initial value of z^0 before BRS. The L-BGFS algorithm in [14] was used to find the optimal z^0. The BRS is done after each forward pass for each new click.

The dazzling thing about this approach is that it can convert any non-interactive method to an interactive one- all that one has to do is to perform BRS after the forward pass!

[21] followed this approach with the feature-BRS (f-BRS) in which the suthors sought to address the high computational cost incurred in BRS stemming from multiple passes through the network. The authors of [21] also introduce an additional zoom-in feature which will be discussed shortly.

The authors reparameterize the problem by realizing that the passes for the optimization needn't be done for the whole network and it suffices to do it for some part of it. For the same they optimize some intermediate features instead of the interaction maps. In the author's terms let l_i denote label of the i^{th} user provided click and (u, v) its location. Consider a function $f(x)$ which is reparameterized as $f(x, a)$ with a as the introduced variable such that $f(x, p) = f(x)$, i.e. for $a = p$ $f(x, a)$ achieves its original value. The optimization task now is it minimize the objective function,

$$\lambda ||\Delta p||_2 + \sum_{i=1}^{n} (f(x, p + \Delta p)_{(u_i, v_i)} - l_i)$$

Here Δp would mean the perturbation over the original value and so the first time would correspond to the inertial term and the second on to the corrective term. In terms of the network the reparameterization was done by adding a scale and a bias to a feature in the intermediate stage. Let $F(x)$ be the output of a certain layer for the input x then the reparameterization could be done as $s \cdot F(x) + b$ where s is a vector of scaling coefficients and b a vector of biases (channel-wise application), s and b would now be optimized.

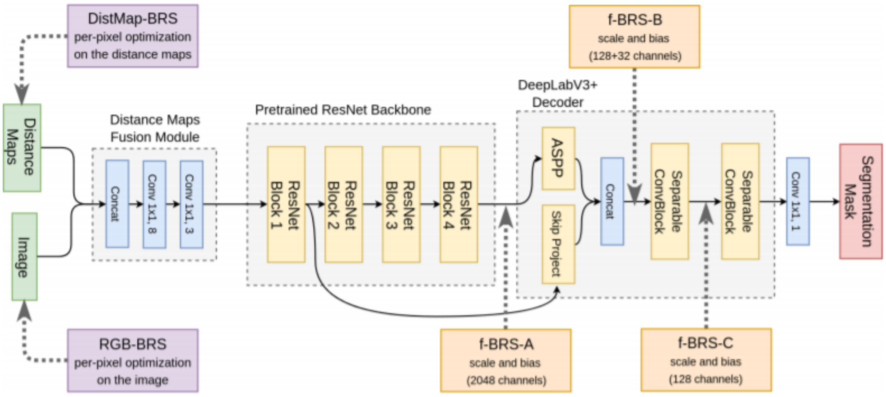


Figure 2.4: f-BRS shown at multiple places [21]

They experimented with various positions of including the s and b but they found f-BRS-B to work the best. Their architecture is shown in figure 2.4.

Zoom-In is a technique which they introduced for capturing some fine details. They realized that a significant accuracy level is achieved after a few clicks. Then with the current prediction a bounding box is created around the predicted area and the image is cropped by the bounding box, which is then upsampled and predicted on again, producing better results without any additional clicks. This is shown in figure 2.5.

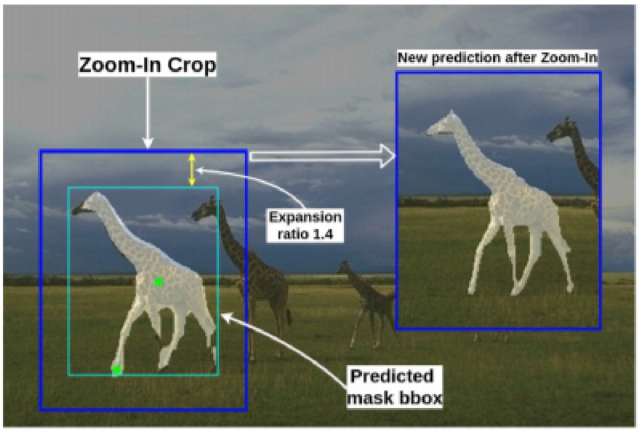


Figure 2.5: Zoom-In, taken from [21]

2.2 Synthetic Image Generation

The general method for generating synthetic images is by generating them solely by using a generator and then discriminating them using a discriminator. But doing so has its drawbacks. Sometimes the discriminator's ability might exceed the ability of the generator by too much leading to discriminator always being able to classify the input image as real or fake. The discriminator on the other hand might be too bad, in which case the discriminator might not learn to produce realistic images. Hence we explored methods which combine GAN based methods with GAN free methods. This brings me to a refiner. As the name suggests, a refiner refines an initial synthesis. This refinement process is done using a GAN based method where a discriminator and a refiner go hand in hand- the refiner trying to make the initial synthesis as close to real as possible, while the discriminator is trying to correctly predict if the input image is real or not.

[20] propose S+U (Simulated + Unsupervised) method, which they call SimGAN, which uses unlabeled data to refine the synthetic images. The authors primarily work on an eye gaze estimation data. The realism is added by using a loss consisting of adversarial loss and a self-regularization loss. In author's notation let x be the synthetic image, the refiner R_θ acts on x and the output id defined as $\tilde{x} = R_\theta(x)$. The authors then propose to make the learning happen by minimizing

$$\mathcal{L}_R(\boldsymbol{\theta}) = \sum_i \ell_{\text{real}} \left(\boldsymbol{\theta}; \mathbf{x}_i, \mathcal{Y}\right) + \lambda \ell_{\text{reg}} \left(\boldsymbol{\theta}; \mathbf{x}_i\right)$$

where $\mathbf{x}_i$ is the i^{th} synthetic image. The first term of the loss function is for addition of realism, while the second term is to preserve annotation mask corresponding to that image. They would want to achieve an ideal refiner that would make it impossible to detect if the refined image is a fake refined image or just a real image. This creates the need for using an adversarial discriminator D_ϕ. Similar to [3] they update the refiner network and the discriminator network alternatively. For the adversarial loss they use,

$$\mathcal{L}_D(\phi) = -\sum_i \log\left(D_\phi\left(\tilde{\mathbf{x}}_i\right)\right) - \sum_j \log\left(1 - D_\phi\left(\mathbf{y}_j\right)\right)$$

where $y_j \in Y$ are unlabeled real images. This is similar to a cross entropy loss. Here training a mini batch is considered, consisting of randonly sampled images from $\tilde{\mathbf{x}}_i$ for all i and $\mathbf{y}_j$ for all j. The parameters ϕ of the discriminator are updated according to stochastic gradient descent considering a mini-batch loss. The refiner on the other hand, tries to minimize the following loss in order to fool the discriminator to classify refined synthetic images as real,

$$\mathcal{L}_R(\boldsymbol{\theta}) = -\sum_i \log\left(1 - D_\phi\left(R_\theta\left(\mathbf{x}_i\right)\right)\right) \quad + \lambda \left\|\psi\left(R_{\boldsymbol{\theta}}\left(\mathbf{x}_i\right)\right) - \psi\left(\mathbf{x}_i\right)\right\|_1$$

The second term in the above loss function is a regularizer, to prevent too much changing of the nuclei involved. This helps preserving the nuclear mask. The ψ is a feature transform and can be anything ranging from a derivative function to some learned transformation. The authors use the identity transform for this. The refiner loss and the discriminator loss, as mentioned before, are used to respectively train the refiner and the discriminator alternatively. The flow for SimGAN is given in figure 2.6.

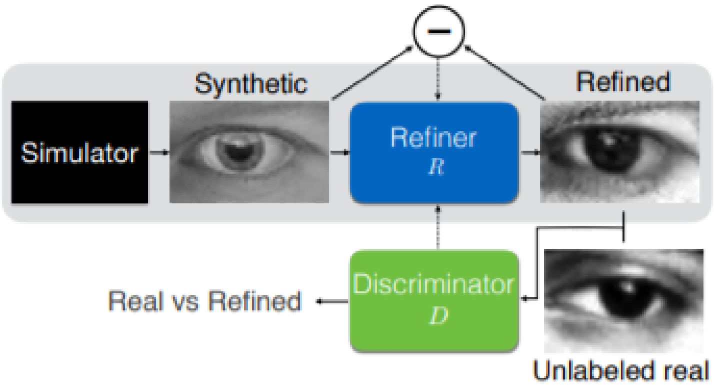


Figure 2.6: SimGAN flow [20]

One issue as pointed by the authors is that in order to fool the discriminator, the refiner has a tendency to over-emphasize on some particular image features leading to introduction of some artifacts. In order to prevent this they use a patch-based method in which instead of defining a global discriminator network, they define a discriminator network which classifies local patches specifically. This way statistics in local patches can be conserved preventing artifacts from forming. The discriminator has been designed by the authors to be fully convolutional. They realize one more issue related to the memory of the discriminator network- the discriminator network focuses only on the latest refined image and takes its decision only based on that. This can cause the refiner network to bring back some artifacts which the discriminator has now forgotten. This may even lead to a divergence of the discriminator network. To counter this problem they update the weights of the discriminator network using not just the current mini-batch but also a few mini-batches coming in before. They introduce a buffer for this purpose to store refined previous images. The illustration of this memory and history type of system is shown in figure 2.7.

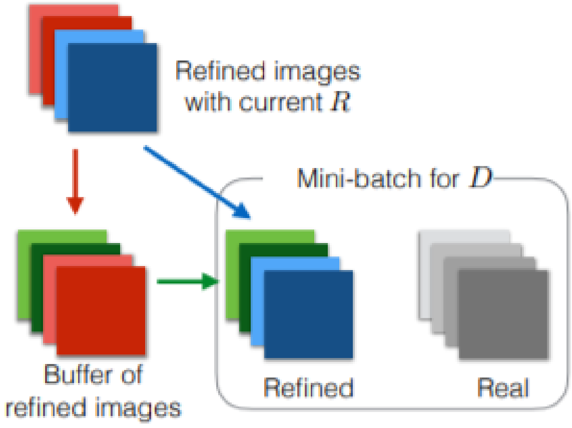


Figure 2.7: Using history of refined images [20]

In [6] the focus has mainly been on nuclei segmentation classification data. They first generate the initial synthesis using non-deep learning image processing techniques. While refining those images, along with the adversarial loss and the realism loss they have included a task specific loss, The task they have chosen is nuclei segmentation performance on some pre-existing dataset. Their proposed approach re-weighs the training loss over synthetic data in order to decrease, or rather, minimize the generalization loss over the true data distribution.

The approach described in [6] by the authors samples a nucleus segmentation mask from a pre-defined approximate ground truth generator. It then utilizes real texture to generate an initial synthetic image. This initial synthetic image is now made realistic by use of a GAN. While doing that they calculate an importance weight of the synthetic example from the discriminator's output by utilizing Baye's theorem while also training a task specific CNN. By giving a weight to each sample, they have the power to decrease the weight given to a synthetic image if it does not seem realistic. The overview of their network is shown in figure 2.8

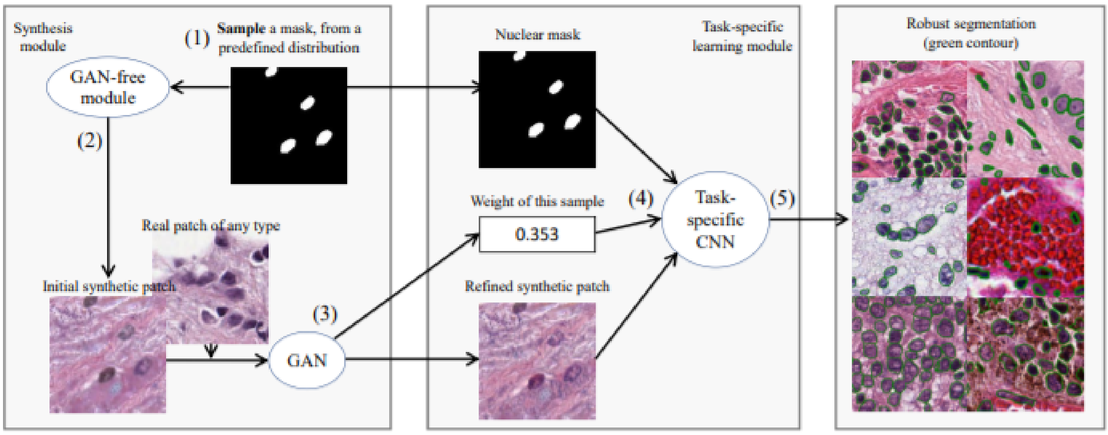


Figure 2.8: Flow of [6]'s approach [6]

Now I describe how the authors minimize the unbiased task specific (here segmentation) generalization loss over the real data distribution, while a sampling distribution (approximate) is given. Let X be a random variable, which would be used to represent an image or a patch, whatever case it is being applied to. The gorund truth is represented as T. The generalization loss (task-specific) is calculated as,

$$L_R\left(\theta_R\right) = \sum_{X,T} f_{\theta_R}(\langle X,T\rangle)\text{p}(\langle X,T\rangle)$$

where $p(\langle X,T\rangle)$ is the probability density of real images and $f_\theta(.)$ is the loss function which is being employed. For the above loss function to be minimized one sample $\langle X,T\rangle$ is sampled from $p(\langle X,T\rangle)$ and then the loss $f_\theta(\langle X,T\rangle)$ is minimized. In the case of number of samples approaching infinity the above equation will be approached for the emperical loss. The probability density of synthetic images is represented as $g(\langle X,T\rangle)$. In an ideal scenario $p(\langle X,T\rangle)$ is equivalent to $g(\langle X,T\rangle)$. But, an unbiased modelling is required for synthesizing unbiased examples and their corresponding nuclear masks. For using $g(\langle X,T\rangle)$ to estimate the generalized loss, the authors of [6] formulate the following task specific loss,

$$L_R\left(\theta_R\right) = \sum_{X,T} \text{f}_\theta(\langle X,T\rangle)\frac{\text{p}(\langle X,T\rangle)}{\text{g}(\langle X,T\rangle)}\ \text{g}(\langle X,T\rangle)$$

Now, $\langle X,T\rangle$ is sampled from $g(\langle X,T\rangle)$ instead of $p(\langle X,T\rangle)$, and then another loss function $f'(\langle X,T\rangle) = f_\theta(\langle X,T\rangle)p(\langle X,T\rangle)/g(\langle X,T\rangle)$ is set to be minimized. Now what constitutes the importance sampling approach, as given in [1], is that sampling is done from $g(\langle X,T\rangle)$ when it gets quite expensive to sample from $p(\langle X,T\rangle)$. After doing this each sample is then re-weighted with the weight $p(\langle X,T\rangle)/g(\langle X,T\rangle)$, and now dropping the T will give $p(X)/g(X)$. This ratio can be derived from an adversarial network. Let the discriminator be trained with a loss log-likelihood actually calculates the probability that the sample is sampled from the real data and not the synthetic data. The authors define c as a ratio between the numbers of synthetic and real input samples, therefore $c = \Pr(X \sim \mathrm{g})/\Pr(X \sim \mathrm{p})$. Thus, $\mathrm{p}(X) = \Pr(X \mid X \sim \mathrm{p})$, $\mathrm{g}(X) = \Pr(X \mid X \sim \mathrm{g})$. Now by employing Bayes theorem,

$$\begin{aligned}\Pr(X \sim \mathrm{p} \mid X) &= \frac{\Pr(X \mid X \sim \mathrm{p})}{\Pr(X \mid X \sim \mathrm{p}) + \Pr(X \mid X \sim \mathrm{g})c} \\ &= \frac{\mathrm{p}(X)}{\mathrm{p}(X) + \ \mathrm{g}(X)c}\end{aligned}$$

Upon shuffling the terms in the above equation the following is obtained which is the importance weight formulated by the discriminator's output, $\Pr(X \sim \mathrm{p} \mid X)$

$$\frac{\mathrm{p}(X)}{\mathrm{g}(X)} = c \cdot \frac{\Pr(X \sim \mathrm{p} \mid X)}{1 - \Pr(X \sim \mathrm{p} \mid X)}$$

In the case that the synthetic patch is unrealistic, i.e.($\Pr(X \sim \mathrm{p} \mid X) \ll 0.5$), then it will be down-weighted. In the scenario that the synthetic patch is realistic but it is not frequently produced, it will be up-weighted.

The authors of [6] do their initial synthesis in two steps. The first step is to get a background and the second step involves foreground. Before these two steps nuclear masks are randomly generated. The polygons making the nuclear masks are of various sizes. The masks are convolved with some blurring kernels to give it a more realistic look. They use Ostu's threshold based super-segmentation technique proposed in [13] to separate the nuclear material. After removing the nuclei the space is filled with the background texture via the inpainting method proposed in [23]. The authors then apply a sub-segmentation technique to the reference patch to gather nuclear material. Sometimes the material gathered may not be enough and so their approach also utilizes textures in Eosin Channel [2] of a randomly extracted patch. After this the background and foreground patches are combined.

These initial patches are now refined using GANs. Three losses have been considered for the same- L_G^{real}, L_G^{reg} and L_G^{hard}. The first two losses are the same as defined in [20], while the third loss is the negative of the task specific loss. The overall loss is given as

$$L_G = \alpha L_G^{reg} + \beta L_G^{real} + \gamma L_G^{hard}$$

γ is chosen to be several orders less than α and β to prevent L_G^{hard} to overpower the others.

The authors of [19] have built a network using just one image. They name their network SinGAN. Their model is able to capture in-built distribution of patches in the image and then goes ahead to create high-quality synthetic images. They employ a pyramid of fully convolutional GANs, which learn the distribution of the patch at a different resolution. The overview of their model is given in figure 2.9.

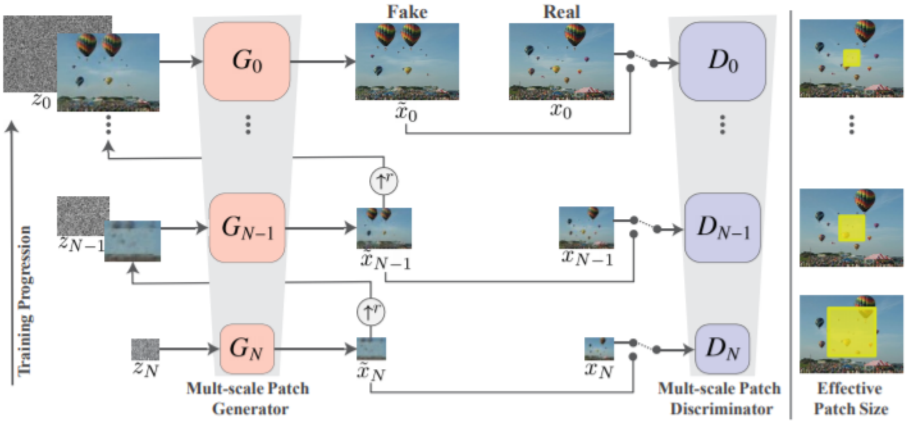


Figure 2.9: Flow of SinGAN [19]

They have a multiple scale architecture of GANs, let they be (in authors' notation) $\{G_0, G_1, ..., G_N\}$ which is trained with a pyramid of the input image x at different scales $\{x_0, x_1, ..., x_N\}$, where x_i is the scaled down version of x by a factor of r^i for some $r > 1$. Each generator G_i aims to produce realistic images with respect to the input image x_i which can fool the discriminator D_i, which is achieved through adversarial training. Image generation starts at the coarsest scale and goes up sequentially though each of the generators to reach the finest scale at the top. Noise is injected at each of the levels. Thus with noise at each step the generator takes input an upsampled version of the image generated at the immediately lower level. Mathematically to represent this the way the author's do:

$$\tilde{x}_n = G_n \left(z_n, (\tilde{x}_{n+1}) \uparrow^r\right), \quad n < N$$

The model os trained from coarsest to the finest GAN. When one of the GANs is done with its training, it is then kept fixed. The author's use the following loss function for their training,

$$\min_{G_n} \max_{D_n} \mathcal{L}_{\text{adv}} \left(G_n, D_n\right) + \alpha \mathcal{L}_{\text{rec}} \left(G_n\right)$$

The first loss penalizes on how far apart are the patches in the real set compared with that of the generated synthetic set. The second loss is the reconstruction loss which is there to ensure that there exists input noise images which could cause the generators to generate the original image.

2.3 Automatic Segmentation

Here I discuss one of the most consequential papers for automatic nuclei segmentation- HoVerNet. This scheme was proposed in [4]. The authors here propose a pipeline to segment nuclei in mutli-tissue histopathology images and simultaneously classify the nuclei in its cancer type. The problems that exist are that there is a huge intra-class variability in nuclei and more often than not nuclei are packed quite close;y resulting in them being overlapped with each other. This network leverages the instance level data for each nuclei and the information embedded in the horizontal and vertical distances of nuclear pixels from the centre of mass of the nuclei. These distances are then used to segregate overlapping and clustered nuclei (separate in the sense that the network is capable of identifying the overlapping nuclei as different nucleus). After segmenting the nuclei, the class of the nuclei is predicted using an upsampling branch.

Firstly, to extract features the authors of [4] use a deep neural network which was inspired by the pre-activated residual network with 50 layers [5]. The authors reduce the downsampling by a factor of 4, from 32 to 8, by making the stride as 1 in the first convolution and removing the max-pooling operations which were performed later. This was done to prevent any loss of information, which would be required to provide a good segmentation result The authors call a chain of successive residual units as a residual block.

After passing the image throguh their modified version of Preact-ResNet50, the authors perform a nearest neighbour upsampling. This is done simultaneously via three different branches, so that classification can ensue along with segmentation. This is shown in figure 2.10.

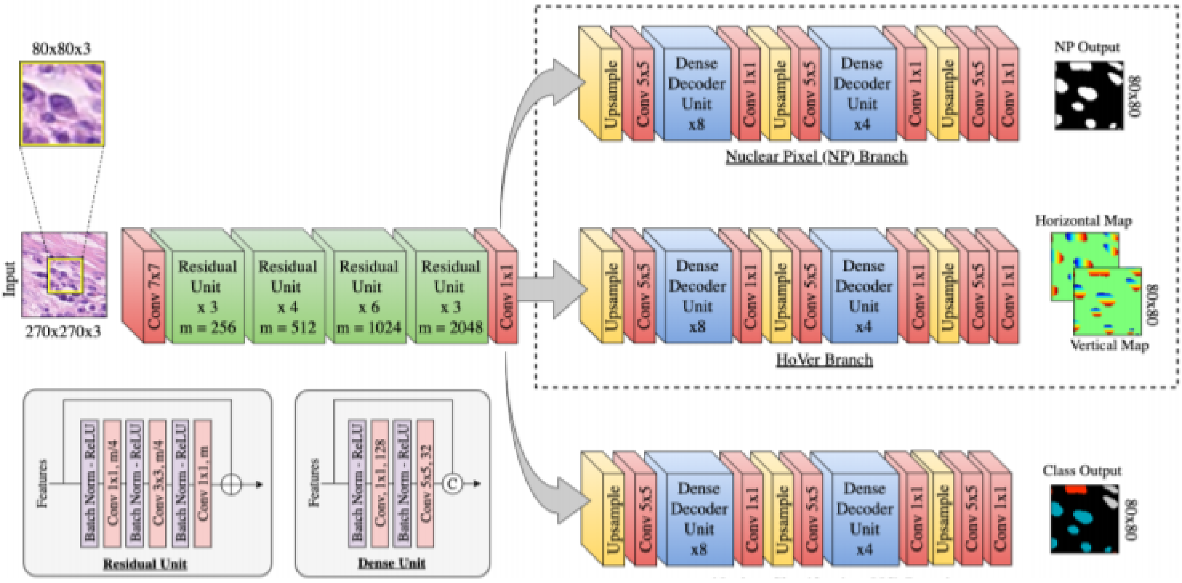


Figure 2.10: Flow of HoVerNet [4]

The nuclear pixel (NP) branch does a pixel classification between foreground and background, i.e. if the pixel belongs to a nucleus or not. The HoVer branch outputs the predicted vertical and horizontal distances of each pixel from its nuclear centre of mass. The nuclear classification (NC) branch is used for classification of a nucleus into its type. For instance segmentation the NP and the HoVer branch work together and for classification NC comes into the picture.

The authors use four sets of weights, in author's notation, w_0, w_1, w_2 and w_3 which would respectively correspond to the weights of Preact-ResNet50 (the part of the network that is common to all the three sub-branches), the NP branch, the HoVer branch and the NC branch. The loss is given as:

$$\mathcal{L} = \lambda_a\mathcal{L}_a + \lambda_b\mathcal{L}_b + \lambda_c\mathcal{L}_c + \lambda_d\mathcal{L}_d + \lambda_c\mathcal{L}_c + \lambda_f\mathcal{L}_f$$

$\mathcal{L}_a$ and $\mathcal{L}_b$ represent the regression loss in relation to the HoVer branch. $\mathcal{L}_c$ and $\mathcal{L}_d$ represent the loss corresponding to the NP branch while $\mathcal{L}_e$ and $\mathcal{L}_f$ are losses in reltion to the NC branch. $\lambda_a, .., \lambda_f$ are scalar weights given to the losses. Consider an input I at the pixel i, in the notation in accordance with the authors, $p_i(I, w_0, w_1)$ is defined as the regression output of the HoVer branch while $q_i(I, w_0, w_1)$ and $r_i(I, w_0, w_1)$ woul represent the pixel wise softmax outputs from the NP and NC branches respectively. The authors then define $\Gamma_i(I), \Psi_i(I)$ and $\Phi_i(I)$ as the respective ground-truths. $\Psi_i(I)$ is the ground truth of the nuclear binary map. In that map the background pixels by convention are denoted as 0 and the nuclei (foreground) pixels are denoted as 1. $\Phi_i(I)$ is the gorund turth for the type of the nucleus. Lastly, $\Gamma_i(I)$ denotes the ground truth of the horizontal and vertical distances of the pixels in the nuclei with respect to their nuclear centres of mass. A regression loss is calcualted at the output of ther HoVer branch, as mentioned before. $\mathcal{L}_a$ is the mean squared error between the predicted and horizontal and vertical distances and their corresponding ground truths. $\mathcal{L}_b$ calculates the MSE between the x and y gradients of the horizontal and vertical maps and the gradients of their respective ground truths. They are formulated by the authors as follows:

$$\mathcal{L}_a = \frac{1}{n} \sum_{i=1}^{n} \left(p_i\left(I; \boldsymbol{w}_0, \boldsymbol{w}_1\right) - \Gamma_i(I)\right)^2$$
$$\mathcal{L}_b = \frac{1}{m} \sum_{i \in \mathbf{M}} \left(\nabla_x\left(p_{i,x}\left(I; \boldsymbol{w}_0, \boldsymbol{w}_1\right)\right) - \nabla_x\left(\Gamma_{i,x}(I)\right)\right)^2 + \frac{1}{m} \sum_{i \in \mathbb{M}} \left(\nabla_y\left(p_{i,y}\left(I; \boldsymbol{w}_0, \boldsymbol{w}_1\right)\right) - \nabla_y\left(\Gamma_{i,y}(I)\right)\right)^2$$

$\mathcal{L}_c$ and $\mathcal{L}_e$ are the cross entropy loss for the NP and NC branches respectively while $\mathcal{L}_d$ and $\mathcal{L}_f$ are the DICE losses for the respective two branches. The two losses are mathematically represented as:

$$\text{CE} = -\frac{1}{n} \sum_{i=1}^{N} \sum_{k=1}^{K} X_{i,k}(I) \log Y_{i,k}(I)$$
$$\text{Dice} \;= 1 - \frac{2 \times \sum_{i=1}^{N} \left(Y_i(I) \times X_i(I)\right) + \epsilon}{\sum_{i=1}^{N} Y_i(I) + \sum_{i=1}^{N} X_i(I) + \epsilon}$$

In the above equations X represents the ground truth, Y represents the output, K is the number of classes and ϵ is a smoothness constant. K for $\mathcal{L}_c$ would be 2 because the NP branch entails a binary classification, while for $\mathcal{L}_e$ K would depend on how many nuclei classes does the dataset have. When classification labels for nuclei are not present in the dataset, only the NP branch and the HoVer branch function to thus produce the instance level segmentation.

Chapter 3

Material and Methods

3.1 Datasets

The dataset which we have used is the MoNuSeg dataset. This dataset was introduced in [11]. The released dataset contains 30 images and about 22,000 nuclei which have a corresponding ground-truth segmentation against them. The dataset is an eclectic mix of tissue images with tumors in different organs. The dataset comprises of Hematoxylin and Eosin (H&E) stained images which were captured at 40x magnification. The authors of [11] downloaded 30 of the whole slide images (WSIs) of the digitized samples of tissues from various organs from the The Cancer Genomic Atlas (TCGA) and they used only one WSI per each patient so that they could maximize the variation in nuclear appearance. The images which were used came from 18 different hospitals. The authors cropped 1000x1000 patches from regions dense in nuclei. The authors covered seven different organs namely, bladder, colon, stomach, prostate, breast, liver and kidney.

In our proposed pipeline we use only the images and not the annotations and nuclear masks. This dataset (images only) will be treated as a hypothetical new dataset for which we don't have annotations.

3.2 Methods

Our proposed pipeline is shown in figure 3.1. The 'Un-annotated Images' block would contain the images from a new dataset, for which we don't have nay available annotation. The 'Training Data' block would contain the data which would be used to train the interactive segmentation tool and to test the performance on an automatic segmentation task. I use the word ’automatic’ to differentiate it from ’interactive’, other than that automatic segmentation is the normal segmentation we are exposed to. Each block has been explained in depth in the following subsections.

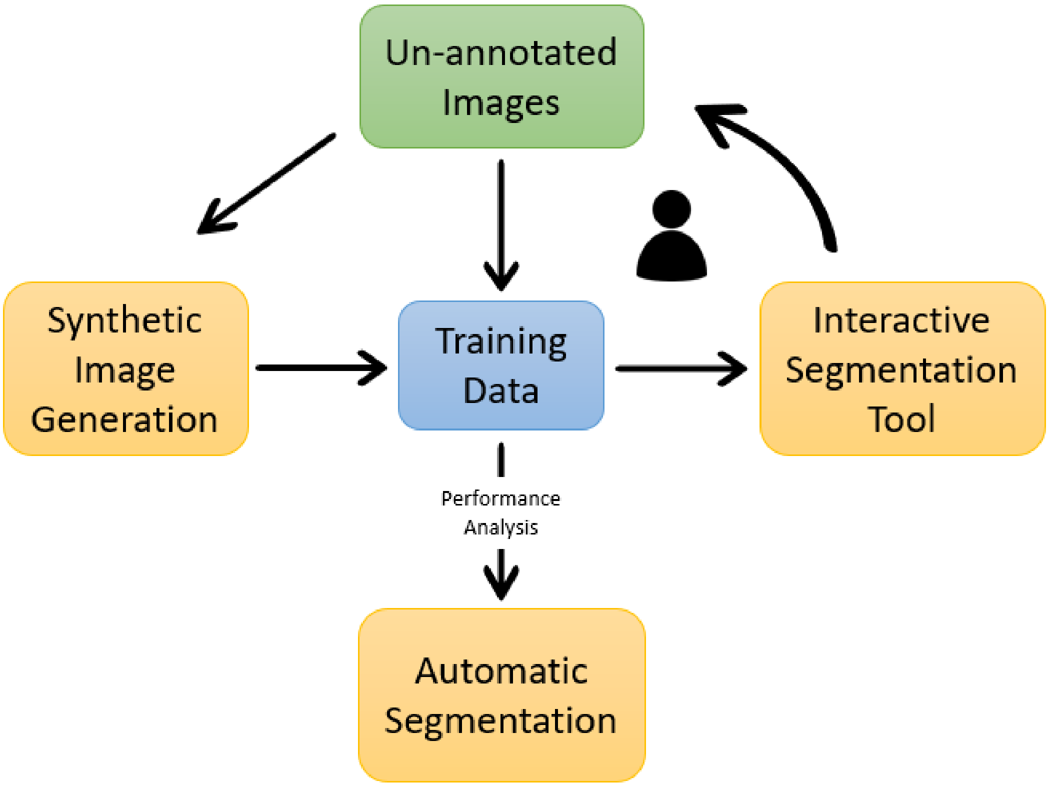


Figure 3.1: Proposed Pipeline

3.2.1 Synthetic Image Generation

The synthetic image generation happens in accordance with the un-annotated images. Those images (and only the images- in real scenarios we won’t have nuclear masks available to us anyway) are used as templates of some sort for the synthetic image generation. First a set of initial syntheses is produced. Then we use adversarial training to refine those images.

Initial synthesis happens in two different steps similar to [6]. The first step is to generate a background image, which contains texture of the background from the real images. This necessitates the removal of the nuclei from the original image for us to get an only-background image. The second step is to isolate nuclear material from the reference images and superimpose them in some random manner in the only-background image. However, before we begin the two steps mentioned, as the zeroth step we must first produce a nuclear mask. For this we first generate a random number of polygons (of varying sizes). These polygons are of variable sizes as well and are permitted to have an overlap up to a fixed overlap threshold (which we set to 0.05, i.e. only 5% overlap in area was allowed). After this mask is produced then we proceed to the two steps.

For the first step the nuclei is removed from the reference by using a Ostu's super-segmentation [13] to isolate the nuclear material. In this process the segmented part always contains the full foreground (sometimes even a part of the background). This is totally okay because the task at hand is to get rid of the nuclear material and if some of the background is thrown away it does not matter. The segmented part is removed and the empty part is inpainted with the texture in the surrounding using the technique in [23].

The second step requires us to obtain the nuclear material. A sub-segmentation method is used for this purpose. The sub-segmented region always lies inside the sought-out segment, in this case the nucleus. So a part from inside the nucleus is segmented providing us with the nuclear material. But we know that nuclei form a very small part of the image in general and hence we may not be able to achieve diversity and hence the textures from the eosin channel in the H&E stained image (eosin is the 'E' in the H&E) [2] were also used to fully characterize the nuclear material. Now we have a full image made up of just nuclear material.

Now we need to superimpose the background image and the nuclear material in the nuclear area as delineated in the nuclear mask. Let for the pixel location r, c (r representing the row and c representing the column) the nuclear mask at that pixel be denoted as $M_{r,c}$, where M denotes the nuclear mask. Let the background only image be denoted as B and the nuclear material image be denoted as N, and the corresponding pixel values be denoted as $B_{r,c}$ and $N_{r,c}$. Let the out image be S and the pixel value be $S_{r,c}$, then

$$S_{r,c} = M_{r,c}N_{r,c} + (1 - M_{r,c})B_{r,c}$$

After obtaining the initial synthesis we move ahead to refine the image using adversarial training and similar to a S+U type of structure [20]. Like [3] we update the discriminator and the refiner alternatively. Both the refiner and the discriminator have been taken to be fully convolutional. The discriminator is trained to identify the image as either fake or real while on the other hand the refiner is trained to refine the input such that the discriminator is fooled into predicting the output of the refiner as real. So the training for both as mentioned before, goes hand in hand and they both try to compete with one another. Let ψ be the parameters for the discriminator and Ω be the parameters of the refiner. Let the refined image be represented as $\tilde{x}$ such that $\tilde{x} = R_{\Omega}(x)$, where x is the input image to the refiner. y is used to represent an image from the un-annotated data, and hence is a real image. The following loss function is minimized for the discriminator to update the parameters represented as ψ:

$$\mathcal{L}_D(\psi) = -\sum_i \log\left(D_\psi\left(\tilde{\mathbf{x}}_i\right)\right) - \sum_j \log\left(1 - D_\psi\left(\mathbf{y}_j\right)\right)$$

This is quite analogous to a binary classification problem for which a cross entropy loss is being utilized. $D_\psi(.)$ represents the probability of the input being a synthetic image while it being subtracted by 1 as shown in the above loss formula represents the probability of being a real image.

For the refiner we consider two losses and add them to obtain the final loss. Ω represent the parameters of the refiner R. The loss equation is as follows:

$$\mathcal{L}_R(\mathbf{\Omega}) = -\sum_i \log\left(1 - D_\psi\left(R_\Omega\left(\mathbf{x}_i\right)\right)\right) + \lambda\|R_{\mathbf{\Omega}}\left(\mathbf{x}_i\right) - \mathbf{x}_i\|_1$$

The first term in the above loss function uses the output of the trained discriminator. By minimizing the first loss the refiner is forced to update its weights in such a way that it produces images which the discriminator is not able to classify as fake. But this brings to light another issue that the refiner might change the image too much in order to fool the discriminator and hence the correspondence of the image with that of the nuclear mask is lost. This is where the second term in the loss function comes into play. the second term prevents the refiner to change the image too much and hence strives to preserve the correspondence between the mask and the refined image.

3.2.2 Interactive Segmentation Tool

We employ the feature Back propagating refinement scheme [21] for performing interactive segmentation. We use High Resolution Net (HRNet) [22] as the backbone for the same. As can be seen in figure 3.2, the architecture can be divided into 4 sub-parts.

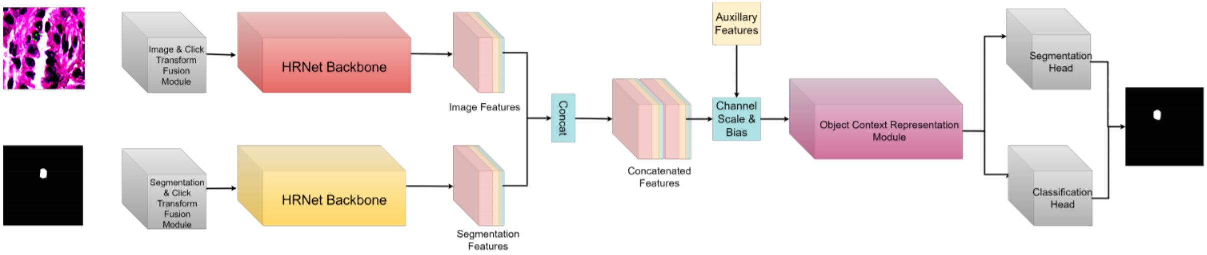


Figure 3.2: Interactive Segmentation Architecture [21]

The first part is the Fusion Model. It is responsible to combine the clicks (foreground and background taken in different channels) entered by the user with the image where the clicks were made, and then produce a 3 channel output. A learnable filter is employed for this purpose. The second fusion model (beneath the first one) is there to include the clicks with the previous generated mask. The second fusion block is employed whenever the feedback mode is on. The output from both the fusion models are concatenated and then a 3 channel output is produced which is fed as input to the HRNet backbone. HRNet was chosen specifically because of its information preserving capability. It achieves so, as shown in figure 3.3, by keeping the highest resolution intact throughout the backbone and simultaneously keeps lower resolution lines as well so that the model can focus both on finer and coarser features.

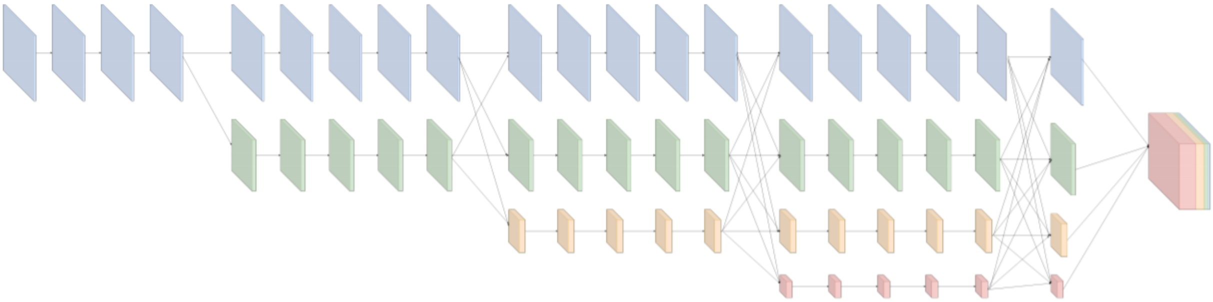
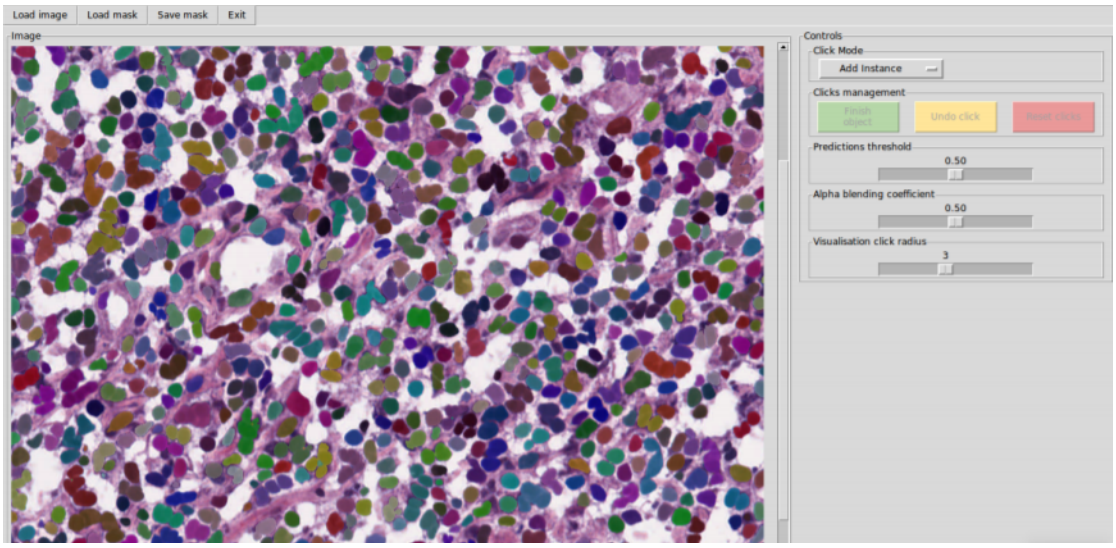


Figure 3.3: HRNet [22]

Then comes the channel and scale bias which introduces parameters that are subsequently optimized (explained shortly) using the clicks given by the user.This optimization happens during inference. Next up is the Object Context Representation (OCR) module which produces a segmentation map helping the further production of object representation vectors.

The optimization which was alluded to in the previous paragraph is what constitutes the feature- back-propagating refinement scheme. The 'feature' in the name corresponds to the fact that a scale and a bias is added to a feature in the network (we do it just after the segmentation and image features are concatenated as shown in the figure 3.2). This optimization is based on the fact that more often than not even the pixels which are given as clicks by the user are wrongly classified. So this optimization strives to optimize the scale and the bias added to in a particular feature to get those pixels rightly classified in the forward pass. In the network, let $F(x)$ be the output of a certain layer for the input x then the output of that stage is modified as $s \cdot F(x) + b$ where s is a vector of scaling coefficients and b a vector of biases (channel-wise application), s and b are to be optimized. We also include the zoom-in crop as introduced in [21], this was done to capture fine details. In this process, the current prediction is used to create a bounding-box around the predicted area. After this, the bounding box region is cropped and predicted on again.

This whole interactive segmentation pipeline has now been embedded in a tool with the help of Shrey Paharia (a former Dual Degree student). This tool, as shown in figure 3.4, is now a part of our proposed pipeline.



Click Mode: Add Instance

Clicks management

Predictions threshold: 0.50

Alpha blending coefficient: 0.50

Visualisation click radius: 3

Figure 3.4: GUI of the interactive segmentation tool, Shrey Paharia

3.2.3 Flow and Process

We begin with a dataset with only images and no annotation masks (precisely the case when we have a new image dataset at hand). They are put in one pool and then given input to the synthetic image generation block. The synthetic image generation block produces 100 synthetic images (say). This is broken down into a 80-20 split for training and validation. Using this synthetic image dataset the interactive model and the automatic segmentation is trained. The automatic segmentation is there is place for performance analysis, and in actual application could represent any task-specific model. Here we use HoVerNet [4] for this purpose.

When the training in the first step is done only using the synthetic images we call it the first cycle. Now using this training set, as mentioned before, we train the interactive and automatic segmentation model. The interactive model after training is used to create segmentation mask from images in the un-annotated set i.e. the real images. We ackmowledge that the training dataset in the first cycle (only synthetic images) won't be that good and it would understandably result in a poor performance. However, this brings us to the very reason to have interactive segmentation. Interactive segmentation has the power to put multiple clicks, both foreground and background, on the image, giving us the power to improve the bad mask produced by putting more clicks. So in the first cycle we use a lot of clicks than usually required to create a visually pleasing mask (because the training set wasn't that good).

We then include the images segmented by the interactive tool in the training set, along with the existing synthetic images. This then starts the cycle 2. Using this updated training set the interactive and automatic models are again trained. We observe (results to be discussed in depth later) that the performance of the automatic segmentation model went up significantly. Now we use the interactive model (which has been trained on the updated training set) to again segment a few more images from the un-annotated set (the real image only dataset) and include them in the training dataset. Now this updated training set is used to again train the automatic and the interactive model, and this marks the third cycle. This cyclic process continues until a desired perfomance is reached. Note we haven't used any ground truth masks in this process.

Chapter 4

Results and Discussions

4.1 Experimental Setup

The experiment was done using the MoNuSeg dataset introduced in [11]. Only the images were considered in the dataset and not the annotation masks to simulate a real scenario of having a new un-annotated dataset. These images were put in the un-annotated image pool in the pipeline shown in 3.1. For each cycle currently we are adding one image (after it is segmented using the interactive model) into the training dataset. We are taking 77 synthetic images in the training set and 24 synthetic images in the validation set. Each new image (with its mask) when put into the training set, is only 400x400 which means we are utilizing only 16% of the image area (actual image size is 1000x1000). This condition will be relaxed in the future if it comes in the way of performance increase.

4.2 Metrics

We use three metrics for evaluation of the performance. These metrics are calculated for the automatic segmentation model, here HoVerNet. These three metrics are: Aggregated Jackarad Index (AJI) [11], DICE [25] and Panoptic Quality (PQ) [10]. Te quite well known, AJI is the aggregated intersection of the ground truth and the segmented nucleus instance over all instances divided by the aggregated union of the ground truth and the segmented nucleus over all instances. The false positives (nucleus is detected where there is none) and false negatives (no segmented nucleus where there actually is one) are also added in the denominator. DICE index is the intersection over union for instances, for which the segmented nucleus intersects with the ground truth even a little. AJI and DICE tend to over-penalize certain cases of misclassification as described in [4], hence we also employ PQ. PQ is given as:

$$\mathrm{PQ} = \frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

where p represents the predicted segment and q represents the ground truth, TP represents the True Positives, FP represents the False Positives and FN represents False Negatives.

4.3 Results and Discussions

Firstly, some of the synthetically generated images are shown in figure 4.1. These were utilized in the pipeline, as explained in the previous section.

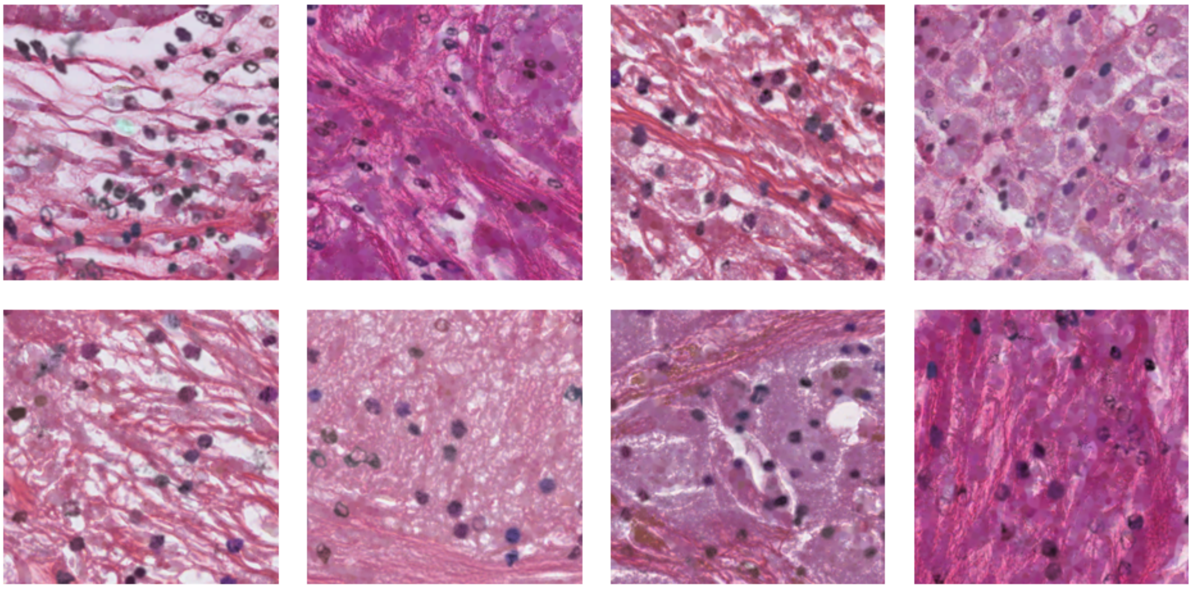


Figure 4.1: Generated synthetic images

Futher, we have currently experimented till 5 cycles. The results are given in table 4.1. Each cycle as mentioned uses constitutes adding one image of 400x400 into the training set. We are making 4 patches from each of the synthetic images and the real images put in after they were interactively segmented. The progression of the values of DICE, AJI and PQ with cycles has been shown in figures 4.2, 4.3 and 4.4 respectively.

Table 4.1: HoVerNet performance in each cycle of the proposed method

Cycle	DICE	AJI	PQ
1	0.458	0.229	0.276
2	0.588	0.308	0.323
3	0.704	0.426	0.44
4	0.746	0.487	0.508
5	0.746	0.504	0.515

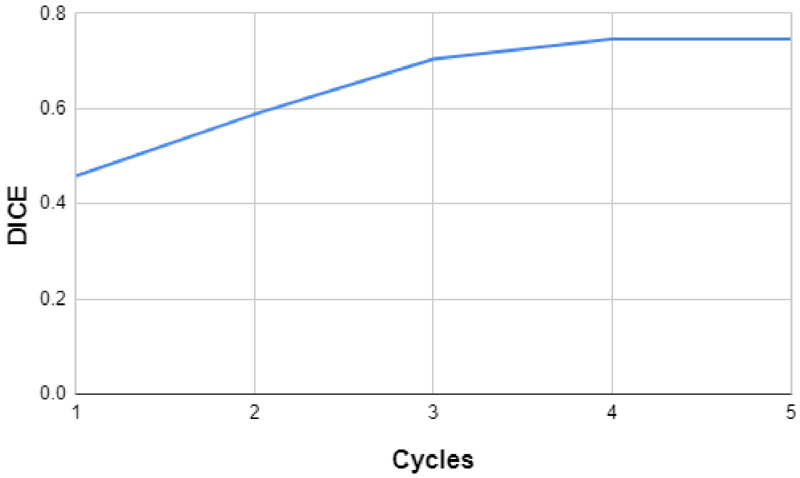


Figure 4.2: DICE vs Cycles

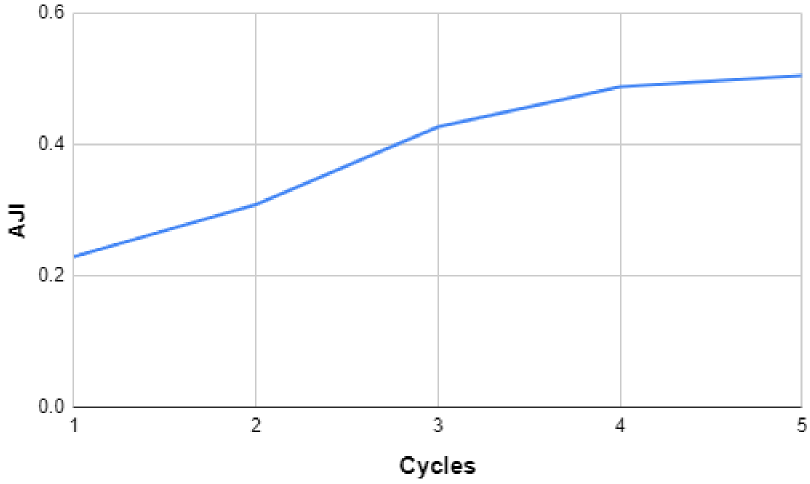


Figure 4.3: AJI vs Cycles

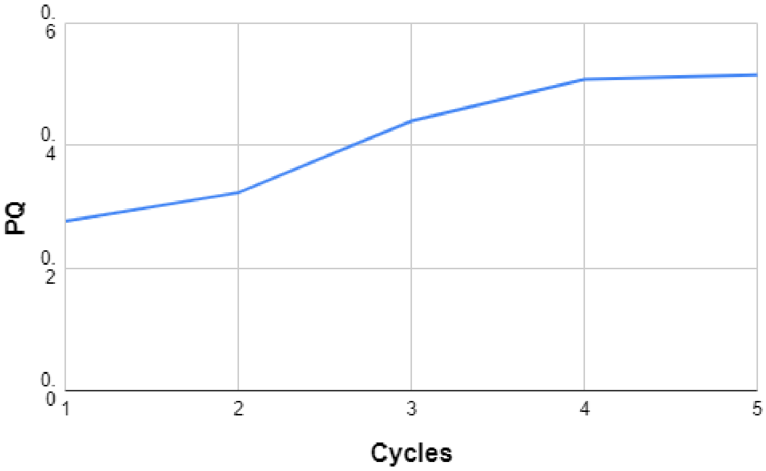


Figure 4.4: PQ vs Cycles

We see that the results look quite encouraging. We plan to experiment more to make the results more conclusive. Note that these results are obtained without using the ground-truth masks. Experimentation in the near feature would involve a temporal and a click analysis, i.e. the time it takes for interactive segmentation to ensue in each cycle and the average number of clicks per cycle it takes to do so. This would help in quantify the effort saving nature of the proposed pipeline in comparison to the usual annotation process. Once this project is complete, we believe we can fundamentally change how we view ground-truth masks and their usage.

Chapter 5

Future Work

5.1 Active Learning

In the proposed pipeline when the human realizes that the automatic segmentation is not performing well on some kinds of image, he/she can decide to interactively segment only those types of image and include them in the dataset. By doing this not only are we improving the performance but we are qualitatively saying which images contain more information. So in the end if we do decide to manually create masks for the images (for which, actually, there is no need because interactive segmentation has got us covered) we know which images to create a mask for. We can save on the annotation cost by only choosing to annotate the more informative images/patches.

This process of selectively choosing images to segment is currenlty being done by a human. In the future we would want to automate this process, i.e. automatically select which images to choose. This is exactly what active learning can provide for us.

5.2 Utilizing the pyramid GAN structure

We obtained the result of giving an input the image on the left and getting the output the image on the right in figure 5.1, by utilizing the Harmonization technique in SinGAN which uses the pyramid GAN structure [19]. An artificial nuclei (just a dark coloured oval shape) was placed in the location marked by the yellow circle. The image on the right shows how well it has blended to its surroundings. But the downside is that the shape of the dark oval shape changed quite a bit and as a result the nuclear mask for that dar oval spot no longer applies to the image on the right. We would like to explore how we can include this amazing result in our process.

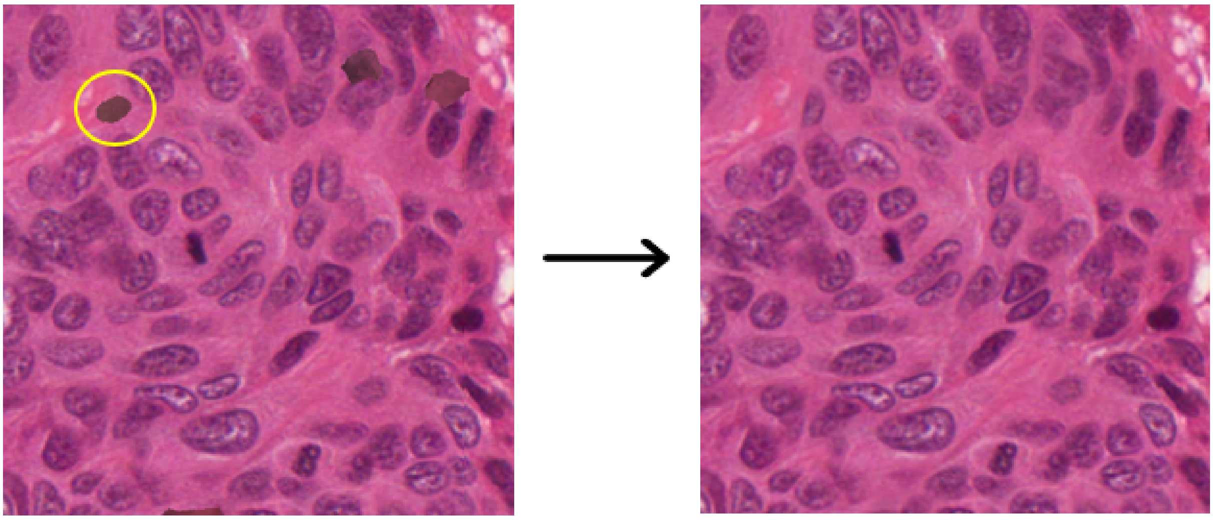


Figure 5.1: Harmonization of artifical nuclei

Chapter 6

Color Normalization

This work has been submitted to IEEE International Symposium on Biomedical Imaging (ISBI) 2021 and is under review.

6.1 Introduction

Staining protocols have a lot of variations in them and it is quite prone to errors too. This results in the performance of deep learning algorithms getting affected drastically. Color normalization is used to transform input samples. Color normalization is used to transform input samples in computational pathology to a pre-defined color space to aid deep learning algorithms [18],[9],[17],[15],[24],[16]. Most of the practical color normalization approaches depend on matrix factorization for color deconvolution. Such approaches take the reference color matrix from a small patch of the H&E stained image, which causes problems when the source image is from significantly different tissue region (anatomically speaking) than the target patch [16]. Additionally, in order to estimate a robust factorization for the entire whole slide image (WSI), such methods often become computationally expensive. On the other hand, more recent color normalization techniques have turned to deep learning, particularly generative adversarial networks [**isbigan**] ,[**staingan**],[**miccai19**],[**inceptionlab**], although such methods are computationally quite expensive to train, and cannot be easily integrated with downstream deep learning pipelines.

We don't use the matrix factorization or GANs here. Instead, we use self-supervised learning, where a lightweight neural network is trained to estimate the color shift needed in each channel to match a pre-determined target stain in appearance.

6.2 Methods

We train a deep learning model to predict the offset which is added synthetically to an unaltered patch sampled from WSI, W_{target}. Target domain $D_{target} = \{I_1, I_2, ...I_N\}$ is a set of images, such that I_n is the n^{th} patch in W_{target}. We perturb the R and B channel of a target histopathology tissue image $I_t \in D_{target}$ by adding a random offset to it to get a synthetic source image, I_s as shown in following equation.

$$I_s^R = I_t^R + \epsilon_1; I_s^B = I_t^B + \epsilon_2; I_s^G = I_t^G;$$

where ϵ_1 and ϵ_2 are random numbers drawn independently from uniform distribution $Unif(-0.2, 0.2)$. We do not add offset to G channel of I_t because stain variation generally do not effect intensity of G channel of image. We train our proposed network, which we are calling, ColorNormNet by optimizing following loss function

$$L(I_t, \hat{I}_t) = ||\hat{I}_t - I_t||_1 + \lambda||\hat{I}_t - I_t||_2^2$$

6.3 Results

Only a subset of the results reported in our paper have been included here. ColorNormNet was evaluated for a nuclei segmentation task using HoVerNet [4] on the MoNuSeg dataset [11]. MoNuSeg contains data from nine different organs with structural and stain variation across different images. It contains a total of 30 annotated images. We split these images in training, validation and testing datasets with 20, 5 and 5 images in each split respectively. We trained HoVer-Net five times using the normalized images with different methods and averaged the DICE score, aggregated Jaccard index (AJI), and panoptic quality (PQ) across the five runs for each normalization method. Table 6.1 shows that ColorNormNet gives better AJI and PQ than other normalization methods.

Table 6.1: Comparison of segmentation results for different color normalization methods

Method	DICE	AJI	PQ
W/O Norm.	0.78	0.52	0.55
Reinhard	**0.81**	**0.53**	0.54
Macenko	0.66	0.39	0.46
Vahadane	0.74	0.40	0.49
ColorNormNet	0.80	**0.53**	**0.56**

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[2] A. Fischer et al. "Hematoxylin and eosin staining of tissue and cell sections." In: *CSH protocols* 2008 (2008), pdb.prot4986.

[3] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014, pp. 2672–2680. URL: `https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

[4] Simon Graham et al. "XY Network for Nuclear Segmentation in Multi-Tissue Histology Images". In: *CoRR* abs/1812.06499 (2018). arXiv: `1812.06499`. URL: `http://arxiv.org/abs/1812.06499`.

[5] Kaiming He et al. "Identity Mappings in Deep Residual Networks". In: *CoRR* abs/1603.05027 (2016). arXiv: `1603.05027`. URL: `http://arxiv.org/abs/1603.05027`.

[6] L. Hou et al. "Robust Histopathology Image Analysis: To Label or to Synthesize?" In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 8525–8534. DOI: `10.1109/CVPR.2019.00873`.

[7] He Huang, Philip S. Yu, and Changhu Wang. "An Introduction to Image Synthesis with Generative Adversarial Nets". In: *CoRR* abs/1803.04469 (2018). arXiv: `1803.04469`. URL: `http://arxiv.org/abs/1803.04469`.

[8] Won-Dong Jang and Chang-Su Kim. "Interactive Image Segmentation via Backpropagating Refinement Scheme". In: *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[9] A. M. Khan et al. "A Nonlinear Mapping Approach to Stain Normalization in Digital Histopathology Images Using Image-Specific Color Deconvolution". In: *IEEE Transactions on Biomedical Engineering* (2014).

[10] Alexander Kirillov et al. "Panoptic Segmentation". In: *CoRR* abs/1801.00868 (2018). arXiv: `1801.00868`. URL: `http://arxiv.org/abs/1801.00868`.

[11] N. Kumar et al. *A Dataset and a Technique for Generalized Nuclear Segmentation for Computational Pathology*. July 2017.

[12] Z. Li, Q. Chen, and V. Koltun. "Interactive Image Segmentation with Latent Diversity". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 577–585.

[13] P. Liao, Tse-Sheng Chen, and P. Chung. "A Fast Algorithm for Multilevel Thresholding". In: *J. Inf. Sci. Eng.* 17 (2001), pp. 713–727.

[14] Dong C. Liu and Jorge Nocedal. "On the Limited Memory BFGS Method for Large Scale Optimization". In: *MATHEMATICAL PROGRAMMING* 45 (1989), pp. 503–528.

[15] Marc Macenko et al. "A Method for Normalizing Histology Slides for Quantitative Analysis". In: *ISBI*. 2009.

[16] Goutham Ramakrishnan, Deepak Anand, and Amit Sethi. *Fast GPU-Enabled Color Normalization for Digital Pathology*. 2019. arXiv: `1901.03088 [cs.CV]`.

[17] E. Reinhard et al. "Color transfer between images". In: *IEEE Computer Graphics and Applications* (2001).

[18] Amit Sethi et al. "Empirical comparison of color normalization methods for epithelial-stromal classification in H and E images". In: *Journal of pathology informatics* (2016).

[19] T. R. Shaham, T. Dekel, and T. Michaeli. "SinGAN: Learning a Generative Model From a Single Natural Image". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 4569–4579. DOI: `10.1109/ICCV.2019.00467`.

[20] Ashish Shrivastava et al. “Learning from Simulated and Unsupervised Images through Adversarial Training”. In: *CoRR* abs/1612.07828 (2016). arXiv: 1612.07828. URL: http://arxiv.org/abs/1612.07828.

[21] Konstantin Sofiiuk et al. “F-BRS: Rethinking Backpropagating Refinement for Interactive Segmentation”. In: *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[22] Ke Sun et al. “Deep High-Resolution Representation Learning for Human Pose Estimation”. In: *CoRR* abs/1902.09212 (2019). arXiv: 1902.09212. URL: http://arxiv.org/abs/1902.09212.

[23] Alexandru Telea. “An Image Inpainting Technique Based on the Fast Marching Method”. In: *Journal of Graphics Tools* 9 (Jan. 2004). DOI: 10.1080/10867651.2004.10487596.

[24] A. Vahadane et al. “Structure-Preserving Color Normalization and Sparse Stain Separation for Histological Images”. In: *IEEE Transactions on Medical Imaging* (2016).

[25] Quoc Dang Vu et al. “Methods for Segmentation and Classification of Digital Microscopy Tissue Images”. In: *Frontiers in Bioengineering and Biotechnology* 7 (2019), p. 53. ISSN: 2296-4185. DOI: 10.3389/fbioe.2019.00053. URL: https://www.frontiersin.org/article/10.3389/fbioe.2019.00053.

[26] Ning Xu et al. “Deep Interactive Object Selection”. In: *CoRR* abs/1603.04042 (2016). arXiv: 1603.04042. URL: http://arxiv.org/abs/1603.04042.