Machine Learning for Histopathology Image Classification: To go deeper or not?

Author:
Saket Choudhary

Supervisor:
Dr. Radhakrishna Bettadapura

A report based on the work conducted at Strand Life Sciences as an intern for the period of May 2018 to August 2018

November 26, 2019

"When you don't know where you are going, any road will take you there."

George Harrison

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Our overall aim in this project is to check the feasibility of using deep learning approaches for doing histopathological image classification. We plan to first start off with a binary classification problem, based on whose success, efforts can be extended to multi class problems later.

1.1 Data

We are currently working with a collection of histopathological images made available through the CAMELYON16 website. The training dataset contains:

- Training data: 270 whole slide images (WSIs) of the lymph nodes around the breast region collected from two independent centers.
  - 100 normal and 70 metastasized (tumor containing) lymph node WSIs from Center 1
  - 60 normal and 40 metastasized (tumor containing) lymph node WSIs from Center 2
- Testing data: A total of 130 whole slide images from the two centers

Whole slide images are storied in a multi-resolution pyramid like structure. In general, this is arranged in a 10 level structure with a level 0 indicating the highest resolution and level 9 indicating the lowest zoomed out version. Each level contains tiles of images and the file format allows rapid retrieval of these subregions at all levels.

Slides containing metastases will also have regions which contain only normal cells, such as in Figure 1.2. The training data also contains this region partition information made available as binary masks over the image where a mask of 1s indicate the region contains tumor while 0s indicate no tumor.

1.2 Goals

1. Does the accuracy vary with normalization? If yes, by how much and which method is the most suitable?
2. What is an ideal sample size for an accuracy of 0.8 or higher?
3. Is a resolution of 20x sufficient? Is 4x too low?
4. Are the probability heatmaps even reliable? (Given we also have the region-wise segmentation information)

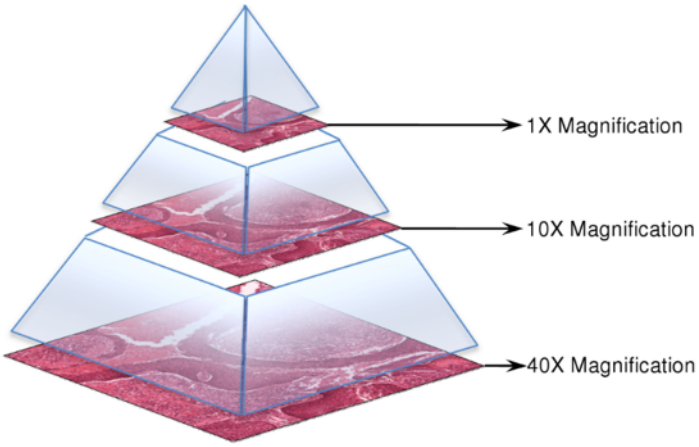

FIGURE 1.1: Arrangement of levels and associated magnification. If there are 10 levels $[0-9]$ and level 0 indicates $40x$ magnification, level 1 will have $20x$ magnification, level 2 will have $10x$ magnification and so n^{th} level will have $40/2^n x$ magnification. (Image courtesy: CAMELYON16)

is_tumor	count	mean	std	min	25%	50%	75%	max	sum
False	111	26605	15981	4656	14857	23628	36056	108210	2953199
True	111	2656	5786	6	70	489	1932	36705	294854

TABLE 1.1: Summary statistics of tumor and normal patches present in tumor slides

5. How many layers do we really need in our Neural Network? Is Inception-v4 the best? Can we tweak it to do better?

The evaluation at each stage happens through a test dataset also made available through the CAMELYON16 which already has both the label information (metastases vs normal) and the region based segmentation masks for metastatic slides. So, these evaluation steps do not require us to interact with pathologists.

1.3 Exploratory Analysis

1.3.1 Distribution of number of patches

We developed a library `pyvirchow` for processing the images which also has multiple machine learning (including deep learning) methods implmeneted. It is available here.

The data and code used for figures generated in this Chapter are available in this notebook .

count	mean	std	min	25%	50%	75%	max	sum
159	25321	23415	2272	12252	20755	32361	240185	4026196

TABLE 1.2: Summary statistics of normal patches present in normal slides

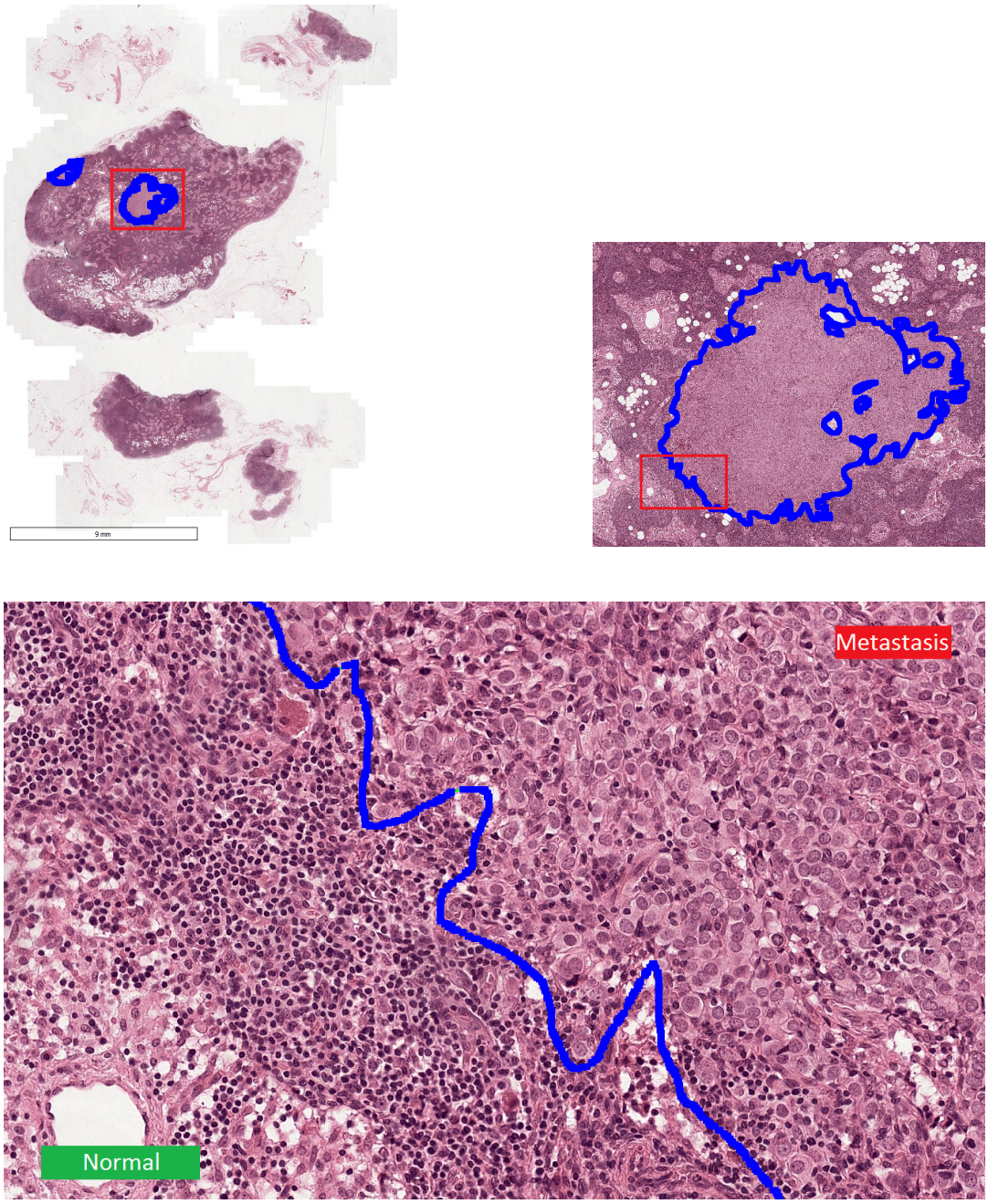

FIGURE 1.2: A low resolution (top left), a mid resolution (top right) and a high resolution view (bottom) of a metastatic-normal region boundary (Image courtesy: CAMELYON16)

	is_tumor	count	mean	std	min	25%	50%	75%	max	sum
0	False	48	23290	14914	4080	13314	19915	28026	81311	1117920
1	True	48	4858	14165	5	40	258	1718	80717	233184

TABLE 1.3: Summary statistics of normal and tumor patches present in test slides

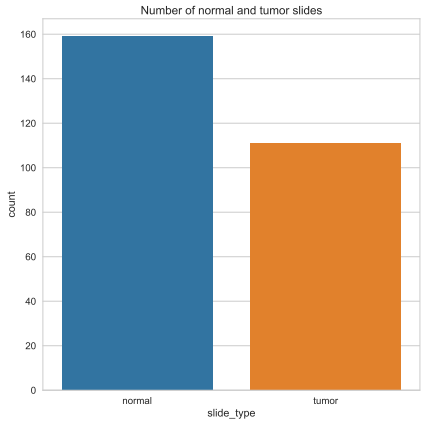

FIGURE 1.3: Number of tumor and normal slides

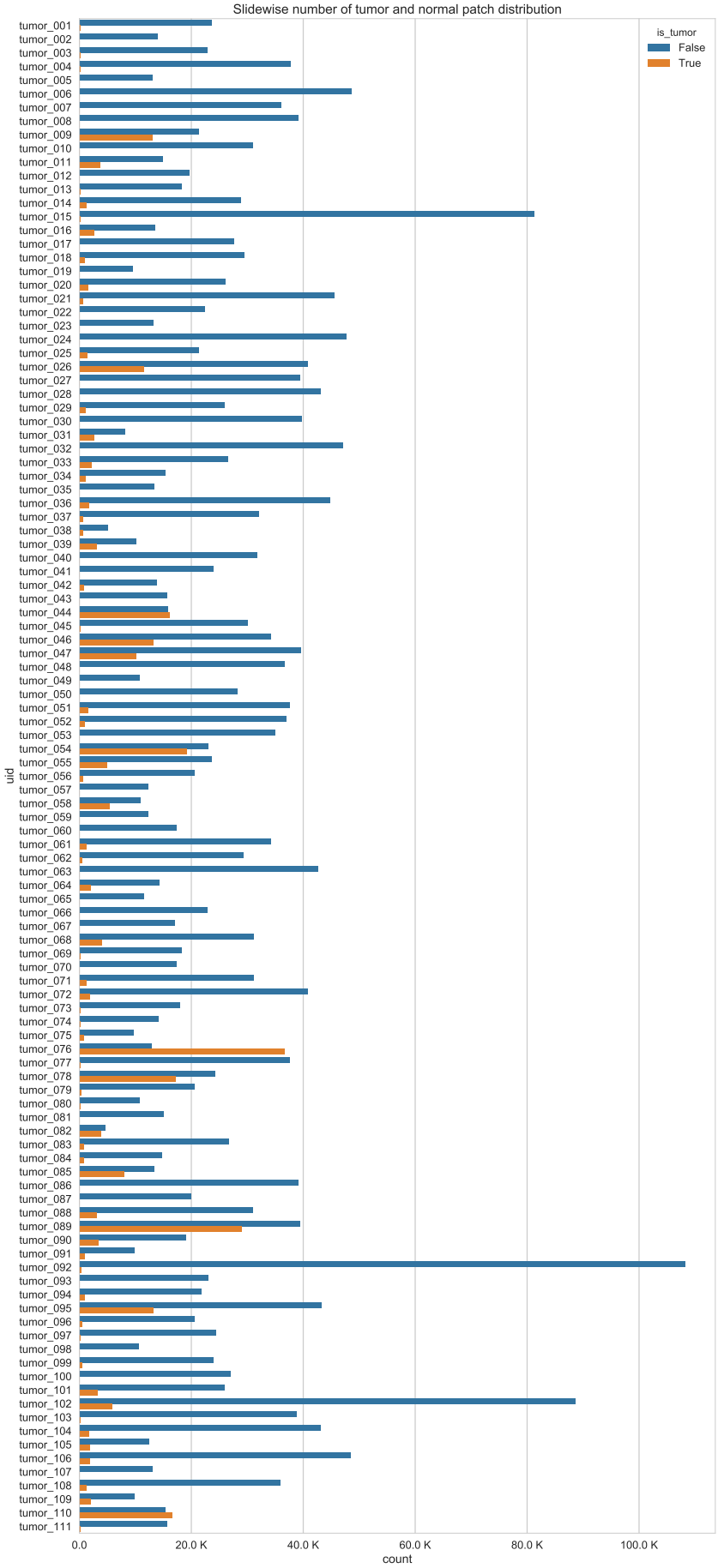

FIGURE 1.4: Number of normal and tumor patches across samples

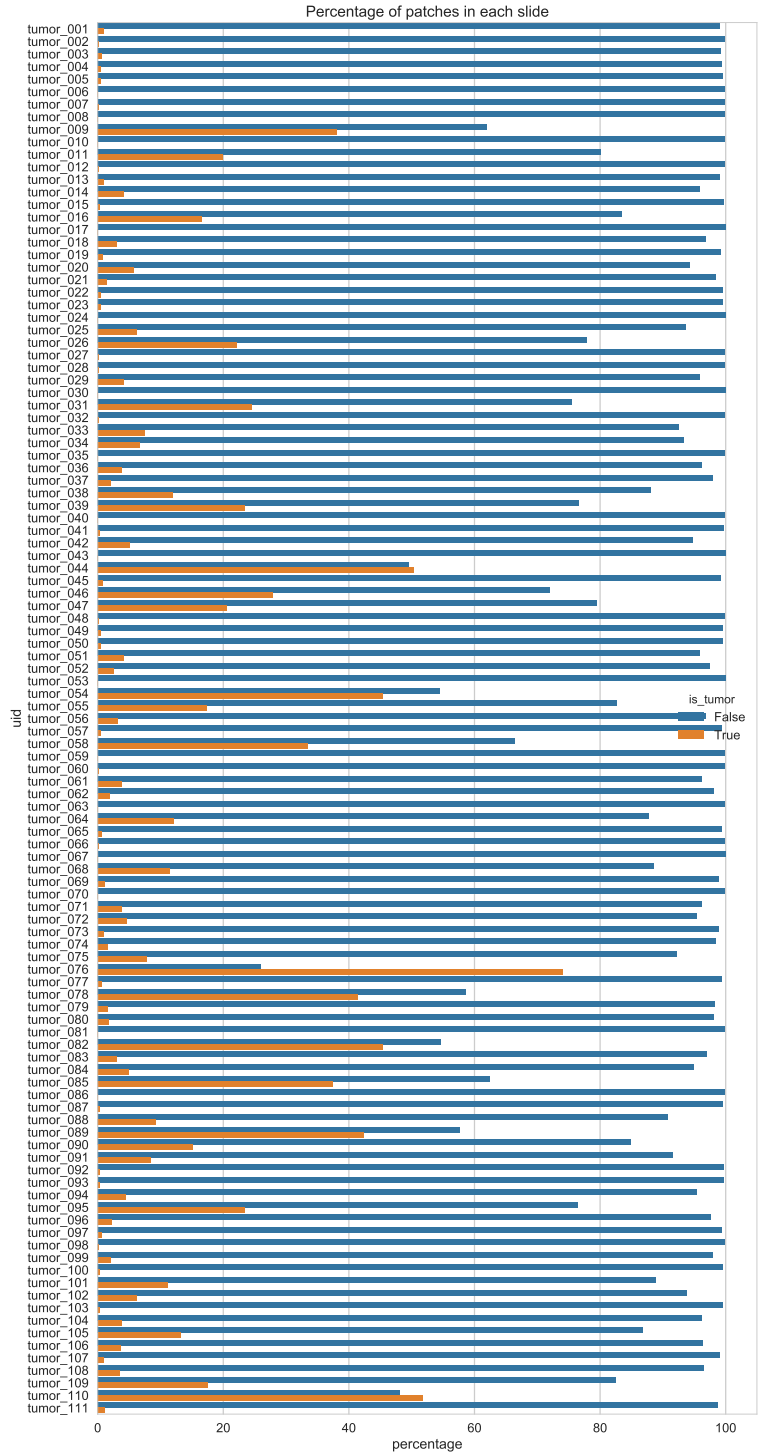

FIGURE 1.5: Percentage of normal and tumor patches across samples

Chapter 2

Color Normalization

We wanted to evaluate if color normalization would affect the downstream accuracy. Normalization is important since different images can have different illumination levels and staining pattern. Here we discuss all the normalization methods that we explored in brief.

2.1 Color transfer

Reinhard et al. (1) developed a simple method to perform color transfer between images. If all the channels of a natural RGB color are plotted in a three dimensional setting, they would all appear to fall along a diagonal implying the R, G and B values are correlated and hence there is dependence between the three channels. If we perform any transformation in the RGB space, we would need to ensure the transformation is applied to all the channels simultaneously.

2.1.1 Problem definition

Given two images, source (s) and target (t), we want to modify the source image so that it *captures* the style characteristics of the target image.

2.1.2 Key Idea

$l\alpha\beta$ minimizes the correlation between R,G and B channels for *many* natural scenes. This thus allows applying different transformations across the three channels ensuring that the cross channel artifacts would remain minimal.

We carry out operations on a PCA subspace of LMS: l, α, β. The LMS colorspace is a representation of the color values based on the responsiveness of the the three types of cones in the human eye at the high, medium and short wavelengths.

In order to emulate the style of 'source' (s) image on the 'target' (t) we simply rely on sufficient statistics: (μ, σ) of each channel.

The transformation over target images l_t, α_t, β_t is given by:

$$
\begin{aligned}
l_t &= l_t - \bar{l_t} \\
\alpha_t &= \alpha_t - \bar{\alpha_t} \\
\beta_t &= \beta_t - \bar{\beta_t}
\end{aligned}
$$

This is then scaled with respect to variances:

FIGURE 2.1: Example of a color transfer as implemented in Reinhard et al. (1) and implemented in the following notebook

$$
\begin{aligned}
l_t &= l_t * \frac{\sigma_s}{\sigma_t} \\
\alpha_t &= \alpha_t * \frac{\sigma_\alpha}{\sigma_\alpha} \\
\beta_t &= \beta_t * \frac{\sigma_\beta}{\sigma_\beta}
\end{aligned}
$$

Finally we do the transfer magic:

$$
\begin{aligned}
l_t &= l_t + \bar{l_s} \\
\alpha_t &= \alpha_t + \bar{\alpha_s} \\
\beta_t &= \beta_t + \bar{\beta_s}
\end{aligned}
$$

An example image with Reinhard's color transformation is shown in Figure 2.1

The aim of this exercise was to use the same idea for normalizing the histopathology images. If one histopathology choosen randomly is declared as the references, all other images can be normalized based on this reference. However, we did not use this normalization approach since it was not clear if such a normalization is important for any of the deep learning architectures we wanted to use downstream.

2.2 Macenko Normalization

Macenko et al. (3) implemented a method for normalizing the histopathology images, based on this key observation that the color values of each histopathology image is a manifestation of two colors: pink and purple with different concentrations.

2.2.1 Method

Consider $W_{w \times h \times 3}$ as the RGB representation of a histopathology image. If I_0 is the intensity of the light illuminating the image, then following the Beer-Lambart law,

the intensity of image is given by:

$$I = I_0 \exp(-W)$$

The optical density is defined by:

$$OD = \log(I_0) - \log(W)$$

A completely white image will have an OD value of 0.

We can then decompose OD into two constituting matrices: C, S

$C_{w \times h \times 2}$ is the concentration matrix $S_{2 \times 3}$ is the color matrix

Once S has been determined for a target image, it can be used as a basis matrix for decomposing other source images.

In order to determine C, S, Macenko et al. make use of principle component analysis.

In order to find the staining vector, The shortest path between two unit-norm color vectors on the sphere is the geodesic path. This line appears to be curved in a spherical coordinate decomposition unless it would correspond to change in only one direction or the other. By finding this specific geodesic direction, we can project the OD transformed pixels onto it in order to find the endpoints that correspond to the stain vectors.

Each image is assumed to be composed of two specific stain vectors and the resulting image is assumed to be composed of linear combination of these stain vectors. The weights of each vector is non-negative and hence each value should exist between these two vectors (i.e. their span should be complete). The shortest path between any two vectors on a sphere is the geodesic path. This path will appear curved in the spherical coordinate, unless both the vectors overlap. We want to find this path so that the OD values can be projected onto this path in order to locate the end points of the stain vectors for that corresponding image. In the absence of any noise the minimum and maximum values found along this direction would constitute the two stain vectors.

In the first step, the plane spanned by the two vectors is determined. This can be approximated by the plane spanned by the first two vectors corresponding to the largest two singular values of the SVD of OD values. After projecting the OD values onto the plane defined by the first two eigen vectors, they are normalized for unit length and then an angle calculated for each vector with respect to the SVD direction. Extreme value angles are then trimmed to account for noise. These trimmed values can then be projected back to obtain the noise free estimates. These steps can be summarizes as follows:

- Step 1: Convert RGB values to OD $OD = -log(I)$ where I represents individual channels $\{R, G, B\}$ and normalized to $[0, 1]$
- Step 2: Filter out lower values of OD and retain $OD > \beta$ for some $\beta > 0$ (typically 0.15)
- Step 3: Calculate SVD of flattened OD matrix ($(w \times h \times)3$)
- Step 4: Create a plane corresponding to the directions corresponding to eigen vectors of the largest two eigen values
- Step 5: Project the OD matrix on this plane and renormalize all values to unit vector

- Step 6: Calculate angle of each of the projected point from the first SVD direction vector arctan $\frac{x}{SVD1_{vector}}$
- Step 8: Find the robust extremes (α^{th} and $(1-\alpha)^{th}$ percentile of these angular values which represent the stain vectors
- Step 9: Project the stain vectors back into the RGB domain

This method is implemented in the following file of `pyvirchow`:

2.3 Vahadane Normalization

Macenko et al. approach inherently assumes that similar proportion of stains are present in all the image and hence is limited in its performance. Vahadane et al. (2) instead introduce a Non-Negative Matrix Factorization (NMF) method that performs sparse strain separation by adding sparsity constraints on the stain channels. This is inspired from the biology of the image itself: each pixel is made of two colors (pink and blue) mixed in different proportions.

If $X_{w\times h\times 3}$ represents the $w \times h$, 3 channel RGB, then it can be decomposed as follows:

We find C, S using: $\min_{C,S} ||X - CS||_F$ such that $C \geq 0; S \geq 0$ where $S_{2\times 3}$ represents the matrix of two stain vectors and $C_{w\times h\times 2}$ represents the associated concentration vector. Since concentration and channel values can never be negative, all values of matrices C, S are non-negative. Thus we solve for:

$$\min_{C,S} \frac{1}{2}||X - CS||_F^2, \text{ such that} C, S \geq 0$$

There is one more clever trick that Vahadane et al. use, again inspired from the biology of the image. Since one type of stain binds only one kind of biological structure (purple stain binds nuclei only), we can further constrain the matrix S to be sparse:

$$\min_{C,S} \frac{1}{2}||X - CS||_F^2 + \lambda \sum_{j=1}^{3} S(:, j), \text{ such that } C, S \geq 0$$

The steps for normalizing then can be summarized as follows:

- **Step 1:** Convert source and target images to optical densities
- **Step 2:** Estimate stain and concentration matrix for source: $X_s = C_s S_s$
- **Step 3:** Estimate stain and concentration matrix for target: $_t = C_t S_t$"
- **Step 4:** Readjust the range of H_s so tha it is similar to the range of H_t.Call it H_{norm}
- **Step 5:** Color exchange $OD'_s = C_s S_{norm}$
- **Step 5:** Project OD back into RGB space

Vahadane et al's algorithm in the is implemented in `pyvirchow` in the following in the following file.

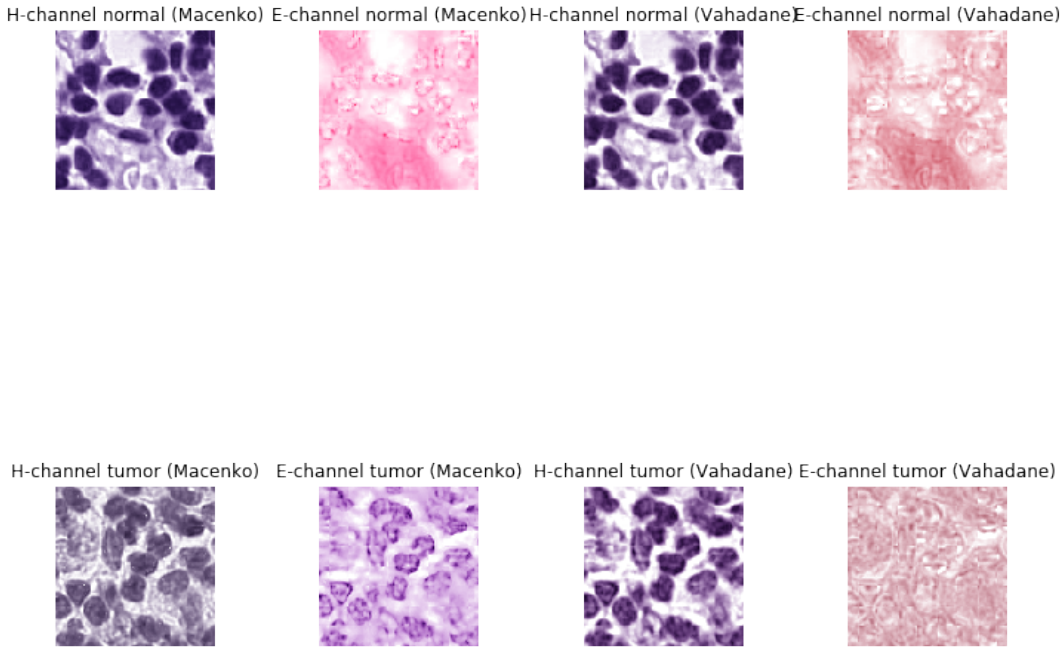

FIGURE 2.2: An example of Macenko and Vahadane normalization

2.4 Xu Normalization

Xu et al. (4) further impose sparsity constraints on the frobenius norm of C, S:

$$min_{C,S}\frac{1}{2}||X - CS||_F^2 + \frac{\alpha_1}{2}||C||_F^2 + \frac{\alpha_2}{2}||S||_F^2 + \lambda \sum_{j=1} 3||S(:,j)||^2$$

Xu normalization is implemented in `pyvirchow` in this file.

2.5 Conclusion

We presented a background for normalizing colors in histopath images with a focus on three key approaches. In the next Chapter we discuss the process of segmentation.

Chapter 3

Nuclei Segmentation

In order to setup a baseline model, we relied on looking at the histopatholgy images using clinician's eye. A pathologist distinguishes between tumor and normal images based on the following criteria:

- **Tumor cells:** dark stained, irregularly clumped nucleus; irregular nuclear membrane; less amount of cytoplasm; more variation in shape and size; overlapping nuclei
- **Normal cells:** round nucleus with regular nuclear membrane; normal staining chromatin; high cytoplasmic volume

In order to extract this information from the images, we perform segmentation in order to separate the nuclei from the rest of the image.

3.1 Problem

Given a patch of a whole slide image, segment the image such that the nuclei blobs can be separated from the rest of the image.

3.2 Approach

Image segmentation is a widely studied subdomain in image processing field. Naively, the idea is to make use of gradients in intensity values to separate out different objects in an image. For the rest of the discussion we denote the intensity at x, y coordinate as $I(x, y)$.

3.3 Binarizing image

The first step of segmentation is to perform binarization. Following Al-Koofahi et al. (5) we implement binarization in two steps, starting with an initial binarization that is later refined using graph-cut algorithm as in Boykov et al.(6). Boykov et al. method performs binary segmentation using the max-flow/min-cut algorithm and requires two probability matrices for background and foreground intensities. These matrices can be approximated using either poisson deconvolution or gaussian mixture models, the difference in these two approaches essentially being the assumption of the underlying probability distribution.

Consider the normalized image histogram $h(i)$ where $i =$ intensity of a pixel:

3.3.1 Poisson Deconvolution

$$h(i) = P_0 \times p(i|0) + P_1 \times p(i|1)$$

P_0 and P_1 are prior probabilities of belonging to background and foreground respectively. For a given threshold t, they can be estimated by:

$$\begin{aligned}
P_0(t) &= \sum_{i=0}^{t} h(i) \\
P_1(t) &= \sum_{i=t+1}^{I_{\max}} h(i) \\
\mu_0(t) &= \frac{1}{P_0(t)} \sum_{i=0}^{t} i \times h(i) \\
\mu_1(t) &= \frac{1}{P_1(t)} \sum_{i=t+1}^{I_{\max}} i \times h(i)
\end{aligned}$$

The optimal threshold t^* is found by :

$$t^* = arg\min_t \{\mu - P_0(t)(\ln P_0(t) + \mu_0(t)\ln\mu_0(t)) - P_1(t)(\ln P_1(t) + \mu_1(t)\ln\mu_1(t))$$

This method is implemented in `pyvirchow` here.

3.3.2 Gaussian Mixture Models

Gaussian mixture model approach is essentially similar to the poisson deconvolution method in the above section, except the intentioes are assumed to follow a gaussian distribution instead of a poisson. We found Gaussian mixture models to be a better fit for the set of images we had. This is implemented in `pyvirchow` here

3.4 Refining binarization

Having obtained a primary level of binarization using either poisson deconvolution or gasussian mixture models, the result of thresholding $I(x,y)$ is further refined by incorporating spatial constraints. Intuitively, we would want to assign the same binary label to nearby pixels, thus penalizing such scenarios.

If $I(x,y)$ represents the intensity at location (x,y) and $L(x,y)$ represents the label assigned at that location, we want to be able to minimize the following loss function:

$$\mathbb{E}[L(x,y)] = \sum_{(x,y)} D(L(x,y); I(x,y)) + \sum_{(x,y)} \sum_{(x',y') \in N(x,y)} V(L(x,y), L(x',y'))$$

where

$$\begin{aligned}
D(L(x,y), I(x,y)) &= -\ln p(I(x,y)|j = 0,1) \\
V(L(x,y), L(x',y')) &= \eta(L(x,y), L(x',y')) \times \exp \frac{-(I(x',y') - I(x,y))}{2\sigma^2} \\
\eta(L(x,y), L(x',y')) &= \begin{cases} 1 & \text{if } L(x,y) = L(x',y') \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

$D(L(x,y), I(x,y))$ represents the loss associated with assigning a label to position (x,y) while the $V(L(x,y), L(x',y'))$ term penalizes neighboring points for having different labels if their intensities is beyond a certain threshold $|I(x,y) - I(x',y')| < \sigma$. σ is empirically set to $20-30$ in most cases. If the image is smooth, σ can be chosen to be of a lower value. This loss function minimization is done through the max-flow/min-cut based approach given in Boykov et al. (6)

3.5 Seed detection

Following eh graph-cuts based binarization, the image has been separated into connected clusters of nuclei. We want to separate these clusters of nuclei into individual nuclei to be able to extract their geometrical properties. This requires identification of seed points which can then be extended to segment out individual nuclei. How do we come up with good seed points?

Loss of gaussian (LoG) filters have widely been used for such seed identification tasks. LoG filter is given by:

$$LoG(x;y) = \frac{\partial^2 G(x,y;\sigma)}{\partial x^2} + \frac{\partial^2 G(x,y;\sigma)}{\partial y^2}$$

where σ refers to the scale value. The expectation is that when this filter is applied on the input image for different values of σ, at a certain value of σ the location of the seed centers will be robust to finer textures (say the chromatin pattern) than the textures imposed by the nuclei themselves.

3.5.1 Method

The content of this section is derived from Lindeberg et al. (7) The scale-space representation $L : \mathbb{R}^D \times \mathbb{R}+ \longrightarrow X$ of any function f is defined as the solution to this diffusion equation:

$$\begin{aligned} \partial_t L &= \frac{1}{2}\delta^2 L \\ &= \frac{1}{2}\sum_{i=1}^{D} \partial_{x_i x_i} L \end{aligned}$$

The family of solutions is equivalent to a convolution with a gaussian kernel $L(.;t) = g(.:t) * f(.)$ where $g : \mathbb{R}^D \times \mathbb{R}_+ \longrightarrow R\text{h}$ is given by:

$$g(x;t) = \frac{1}{(2\pi t)^{N/2}} \exp - \frac{x_1^2 + x_2^+ \cdots + x_D^2}{2t}$$

Different values of α will cover different scales of details in the original signal f. Higher values of α will capture the coarser details while smaller values will capture the finer details. Mathematically, this implies that the amplitude of the spatial derivatives $L_{x^\alpha}(.;t) = \partial_{x^\alpha}(.;t) = \partial_{x_1^{\alpha_1}} \partial_{x_2^{\alpha_2}} \ldots \partial_{x_D^{\alpha_D}} L(.;t)$ decrease with scale.

Automatic seed detection makes use of this concept of normalized derivatives to automatically select the scale at which the spatial derivative achieves its maximum.

We start here with an example based on sine function. $f(x) = \sin \omega_0 x$. The diffusion equation solution for $f(x)$ is given by:

$$L(x;t) = e^{-omega^2 t/2} \sin \omega_0 x$$

The maximum amplitude is given by $L_{max}(t) = e^{-omega^2 t/2}$ and decreases exponentially for the m^{th} order smooth derivative.
$L_{x^m,max}(t) = \omega_0^m e^{-\omega_0^2 t/2}$.
If we introduce a $\gamma - normalized$ derivative operator:

$$\partial_{\epsilon,\gamma-norm} = t^{\gamma/2}\partial x$$

equivalent to a change of variable as

$$\epsilon = \frac{x}{t^{\gamma/2}}$$

The m^{th} smooth derivative is then given by:

$$L_{\epsilon^m,max}(t) = t^{m\gamma/2}\omega_0^m e^{-\omega_0^2 t/}$$

which first increases and then decreases and hence has a unique maxima occurring at $t_{max,L_{\epsilon^m}} = \gamma m/\omega_0^2$
Define the scale parameter $\sigma = \sqrt{t}$ and let λ be the wavelength of the signal given by $\lambda = \frac{2\pi}{\omega_0}$. Then the scale $\sigma_{max,L_{\epsilon^m}}$ where the amplitude of the γ normalized derivative achieves its maximum is given by

$$\sigma_{max,L_{\epsilon^M}} = \frac{\sqrt{\gamma m}}{2\pi}\lambda_0$$

and the maximum value of the derivative acheived at this scale is given by:

$$L_{\epsilon^m,max(t_{max},L_{\epsilon^M})} = \frac{(\gamma m)^{\gamma m/2}}{e^{\gamma m/2}}\omega_0^{(1-\gamma)m}$$

The entire procedure can be thought of a pattern matching where we are trying to match Gaussian kernels of different size to a given image pattern by using a normalization strategy based on previous matching. $\gamma - normalization$ gives a one to one correspondence between the matching response of the Gaussian kernel derivative and the wavelength of the signal. We want to select that scale of the kernel such that at that scale the response of the match is the highest.

3.6 Clustering

Having performed automatic seed detection based segmentation as in the previous section, the resulting segmentation could still have two or more nuclei grouped in one label as a single large blob. In order to avoid such scenarios, the scale parameter requires more fine tuning. Al-Koafhi et al. (5) devised a novel strategy to handle this by making use of Euclidean distance maps also known as distance transform that

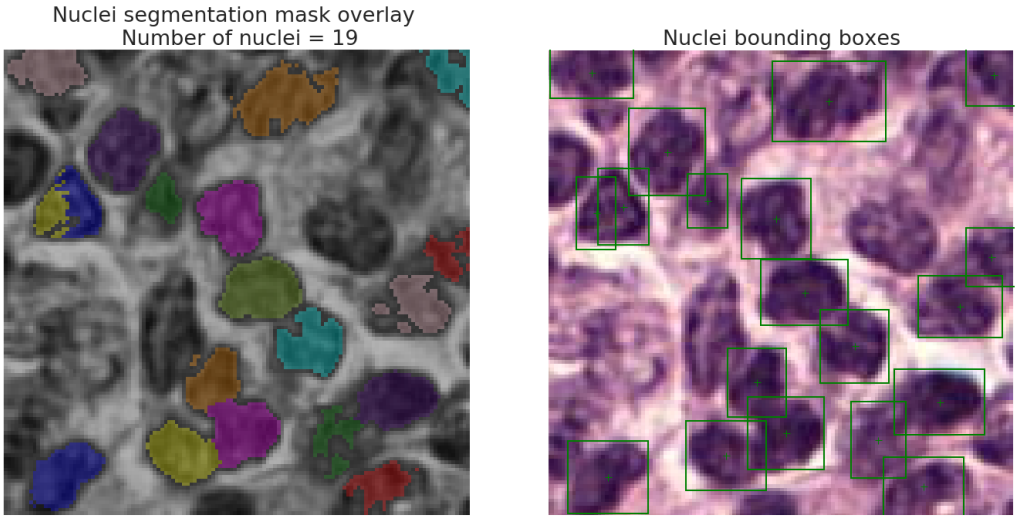

FIGURE 3.1: A tumor patch with segmented nuclei

represents for each pixel the distance to the nearest obstacle pixel (which in this case refers to a different label)

- **Step 1:** Perform LoG(x,y;σ) at multiple values of $\sigma = [\sigma_{min}, \ldots \sigma_{max}]$
- **Step 2:** Use the Euclidean distance map $D(x, y)$ to constraint the maximum scale values. Euclidean distance map gives for each pixel value the length of the shortest path from this pixel to the boundary. The distance map constraints the scale value at each point resulting in a response curve $R(x, y)$ that can be thought of as a topography with its peaks indicating seed centers of the nuclei. The response surface $R(x, y)$ is given by $\arg\max_{\sigma \in [\sigma_{min}, \sigma_{max}]} \{LoG(x, y; \sigma) * I(x, y)\}$ where $\sigma_{max} = \max\{\sigma_{min}, \min\{\sigma_{max}, 2D(x, y)\}\}$.
- **Step 3:** Perform local max clustering on seed centers which uses a resolution parameter r to club nearby clusters into one.

For further details of the algorithm we refer the reader to (5). The max-clustering algorithm is present in `pyvirchow` here.

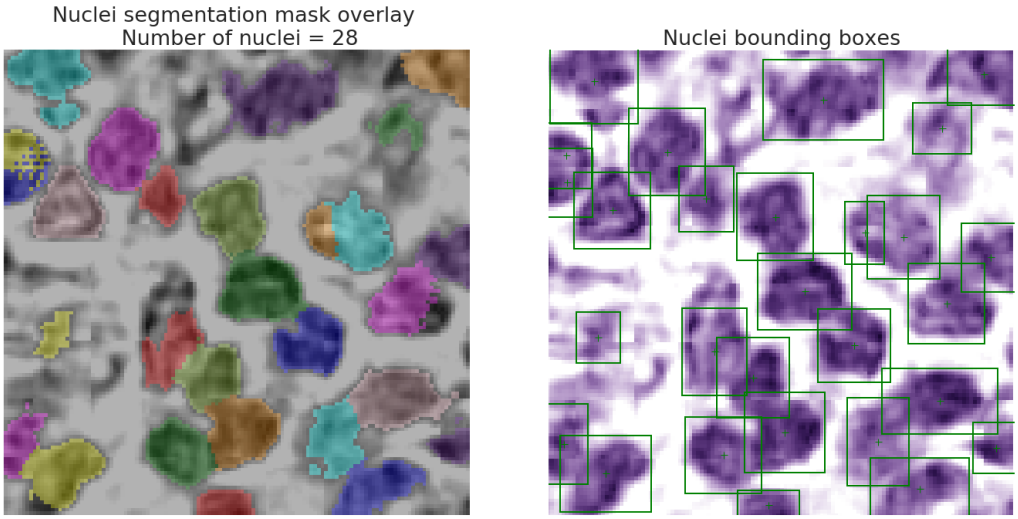

FIGURE 3.2: A tumor patch with segmented nuclei applied only to the H-channel (purple) image after Vahadane et al. (2) based channel separation. Applying color normalization enhances the segmentation results.

Chapter 4

Classification Approaches and Results

In this Chapter we describe all the approaches we took and the corresponding results.

4.1 Existing Approaches

Histopathological image classification has been explored in multiple publications across the literature. The first large scale study to the best of our knowledge was through the CAMELYON16 challenge. CAMELYON16 dataset included the following:

- Training data: 270 whole slide images (WSIs) of the lymph nodes around the breast region collected from two independent centers.
  - 100 normal and 70 metastasized (tumor containing) lymph node WSIs from Center 1
  - 60 normal and 40 metastasized (tumor containing) lymph node WSIs from Center 2
- Testing data: A total of 130 whole slide images from the two centers

Each tumor slide in the training and test dataset has manual annotations indicating the tumor regions.

CAMELYON16 winners indicated a patch level accuracy of 98.4% using a 22 layer convolutional neural network (8).

Liu et al. (9) used Inception model (10) achieving an overall patch-level accuracy of $\geq$ 98% with 48 layers. Li and Ping (11) made use of neural networks followed by conditional random fields to achieve an overall patch-level accuracy of 93.48%.

4.2 Our Approach: Dataset

We extracted around 2.5 million normal and 2.5 million tumor patches from the entire pool of training sites. The associated function is available in this python method.

4.3 Our Approach 1: CNN

Inspired from the architecture of Long et al. (12), we came up with a similar arrangement for pixel level classification of patches which is implemented in `pyvirchow` here.

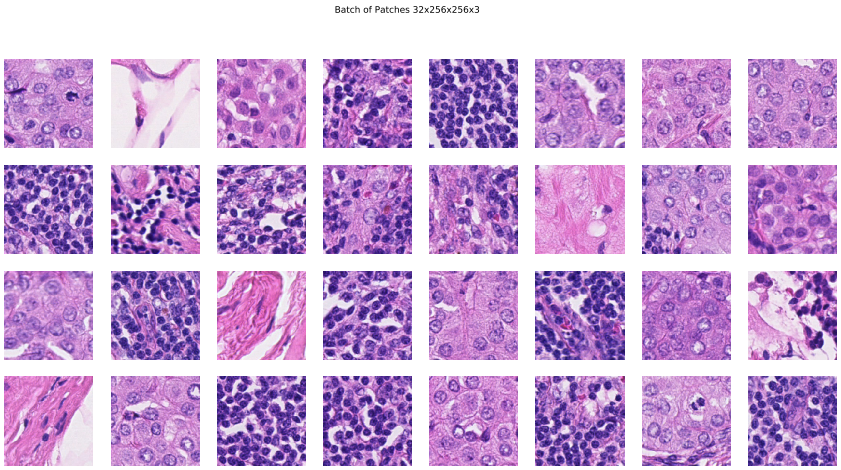

FIGURE 4.1: A sample batch of 32 images that is fed to the 8 layer CNN

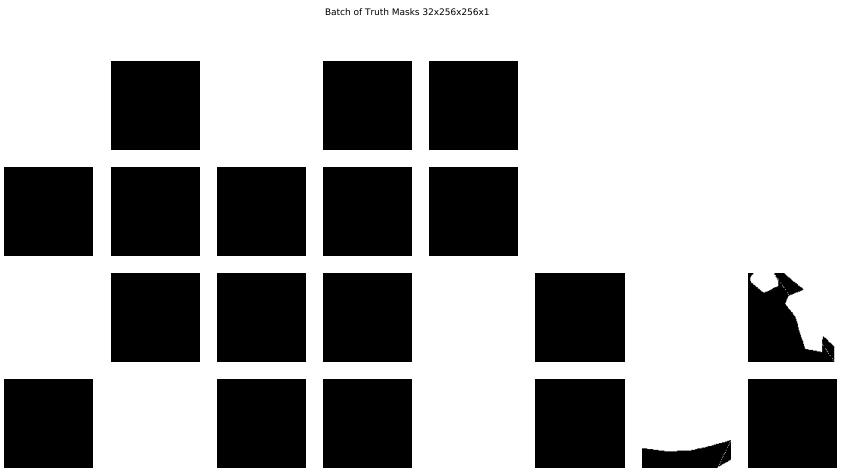

FIGURE 4.2: Masks corresponding to the 32 images in Figure 4.1. Black denotes normal and white denotes tumor regions

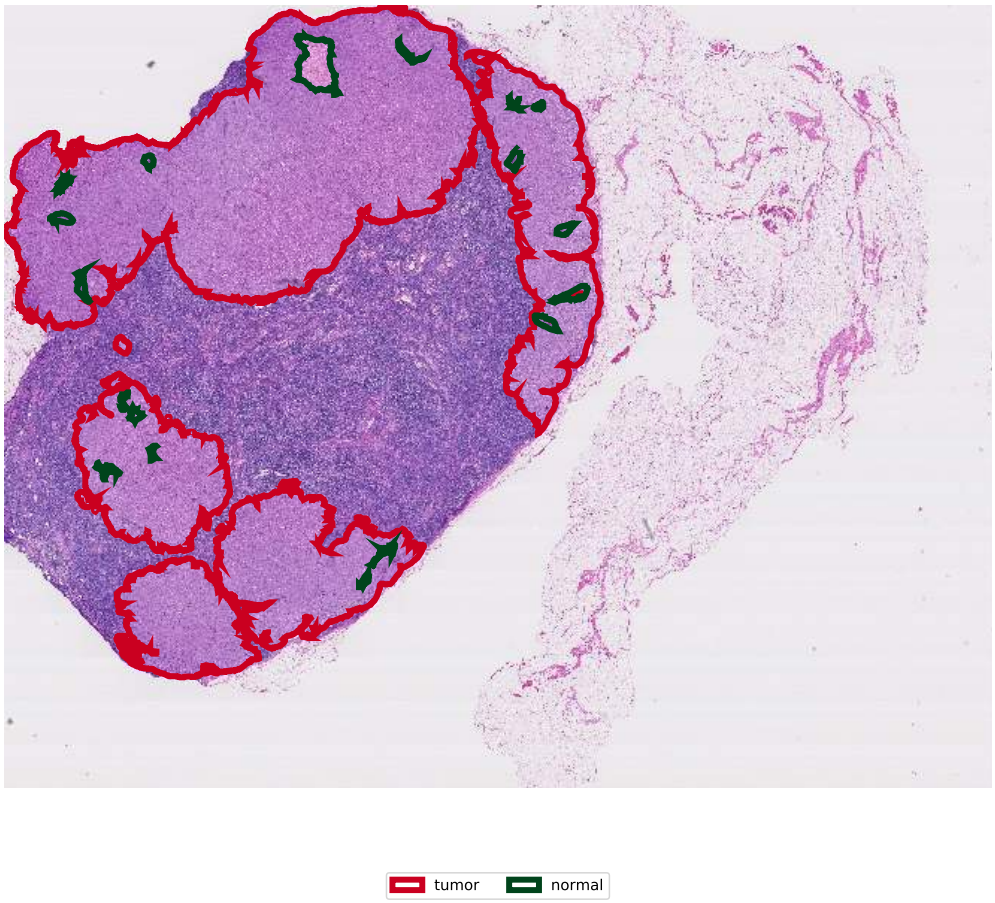

FIGURE 4.3: Example of a whole slide image with tumor and normal regions annotated

In order to verify that such an architecture was capable of learning the difference between tumor and normal patches, we first tested this on only one slide image, training on 20k patches and testing on remaining 10k patches. This gave as an overall accuracy of $\sim 98\%$.

Though the eight layer model was successful in learning on one slide only it fails to generalize.

4.4 Our Approach 2: Autoencoders

Autoencoders are a great tool for reducing the dimensionality of high-dimensional data while retaining the most significant features such that the data can be reconstructed using these reduced dimensions to a great extent. The details of autoencoders appear in Appendix A. The relevant python functions are available in this notebook.

We input 256×256 size image into autoencoder reducing it to just 100 dimensions and then using a random forest on these dimensions. Though we perform

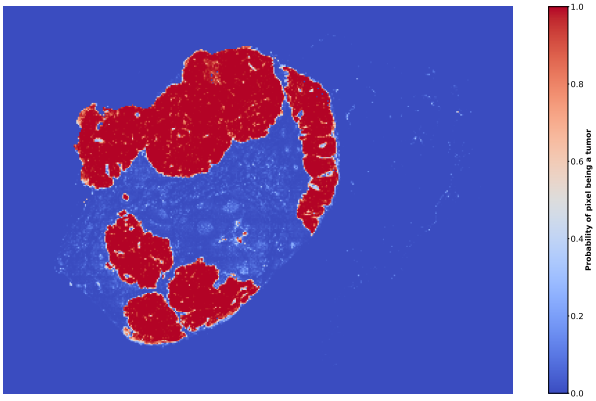

FIGURE 4.4: Prediction of our 8 layer CNN model when trained using 20 patches and predicted on the remaining 10k patches.

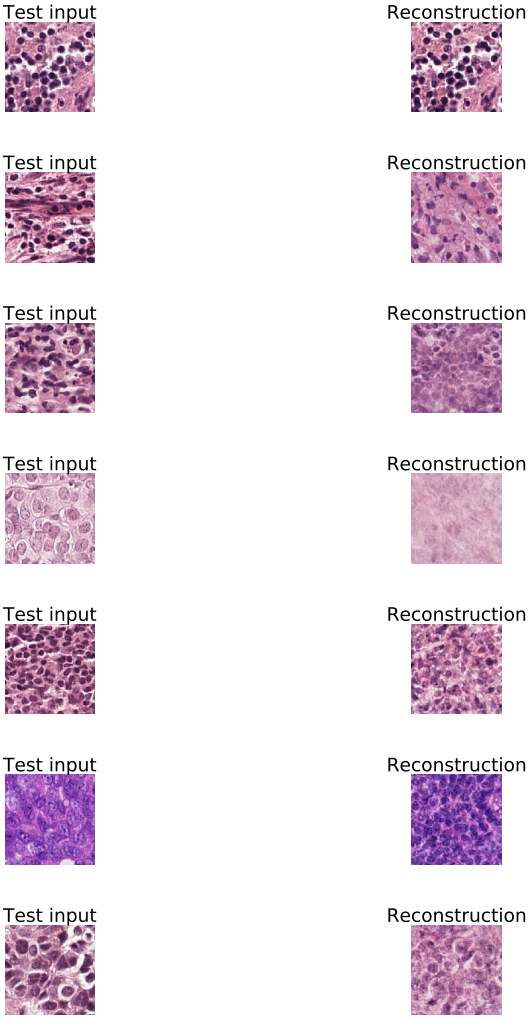

FIGURE 4.5: Reconstructed images from an autoencoder

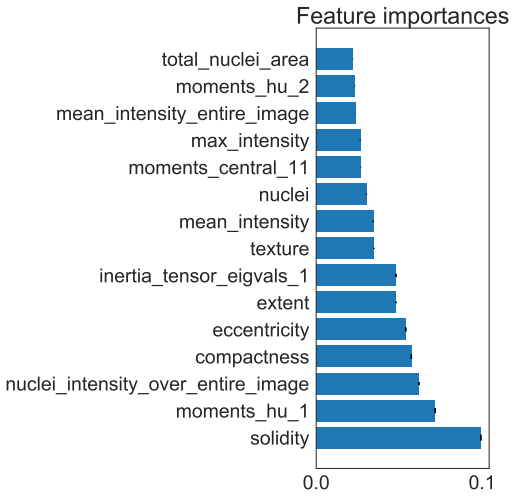

FIGURE 4.6: Feature importance for Approach 3

better in this case as compared to our Approach 1, the accuracy is still limited to around 70%.

4.5 Approach 3: Segmentation followed by Random Forest

In our third approach we make use of the methods we discussed in Chapters 2 and 3 to segment the nuclei from the image and then extract features from these nuclei. We then make use of a random forest classifier on these features.

Surprisingly our this approach outperforms the other two approaches and we achieve close to 85% accuracy. The totality of features used in this apporach appear as a table in this notebook.

4.6 Conclusions

We tried multiple approaches for classifying histopathological images into tumor and normal classes. A baseline method using features of the nuclei outperformed the other neural network approaches. However, different groups have attained a superior accuracy with a deep neural network approach in the literature. These deep networks also require significant training time and we did not spend enough time towards replicating their results. Our simpler deep neural network of eight layers was not sufficiently deep to learn the intricacies of the entire dataset though it achieved higher accuracy when trained using data from one slide only. In hindsight, we would recommend the following:

- The architecture in (8; 9; 11) are not hard to implement as the underlying models are essentially already available. It should be possible to emulate their results given sufficient compute resources and time.
- Our Approach 3 of random forests appears to have lot of correlated features that all appear to rank high (such as solidity and compactness), probably random forest is not the best choice of algorithm here. Pruning features to a more non-correlated subset, might lead to higher accuracy

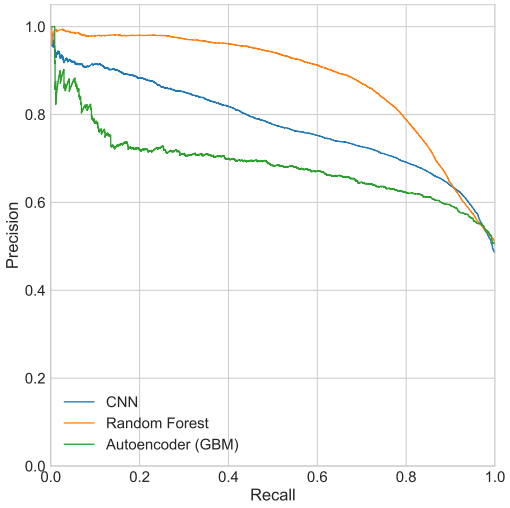

FIGURE 4.7: Comparison of AUC for three different approaches. Approach 3 (RF) outperforms other approaches achieving a maximum accuracy of 85%.

- Though the eight layer network didn't end up with a high accuracy, it might still be possible to gain higher accuracy by increasing the total number of layers (and hence the parameters)

Appendix A

Autoencoders

This document summarizes our current understanding of Autoencoders. The first section acts as a short introduction to Variational Inference which forms the backbone of Autoencoders. In section two we summarize the theory behind autoencoders and discuss VAEs in detail. The two sections may appear to be disconnected at first, so we reintroduce the

A.1 Variational Inference

A common problem in modern statistics is to be able to approximate difficult to compute probability distributions. The most common use of this in bayesian inference where the aim is to infer the unknown quantities using a posterior distribution.

Variational Inference converts the problem of inferring this probability distribution to an approximation problem. It can be thought of as an alternate to MCMC, but is faster and scales easily to larger datasets.

Consider we have a probability distribution $p(\mathbf{x})$. $\mathbf{z} = z_1, z_2, z_3 \dots z_m$ are m latent variables and $\mathbf{x} = x_1, x_2 \dots x_n$ are n observed random variables. Assume we have large number of datapoints.

$$p(x) = \int p(z)p(x|z)dz$$

A.1.1 Why not MCMC?

Our aim is to infer $p(z|x)$. In an MCMC setting, we would create an ergodic markov chain on z, whose stationary distribution is $p(z|x)$. We would then sample from this chain to collect samples from the stationary distribution and the posterior will be estimated empirically from a subset of these collected samples.

We need a faster way than MCMC to be able to do this, especially given our assumption of large dataset. So we will replace the slow step of sampling with optimization.

MCMC and Variational Inference (VI) essentially solve the same problem. MCMC samples a markov chain while VI solves an optimization problem. MCMC approximates the posterior of the chain while VI approximates through optimization.

A.1.2 Variational Inference

Let $\mathbb{Q}$ represent a family of approximate densities over the latent variables z. We want to find a member $q^*\mathbf{z}$ such that the KL divergence between this family and the posterior $p(z|x)$ is minimized:

$$q^*(\mathbf{z}) = \text{argmin}_{q(z)\in\mathbb{Q}} \text{KL}(q(z)||p(z|x))$$

Hence our original inference problem has been converted to an optimization problem. The complexity of this optimization problem depends on how complex is our choice of family of distribution $\mathbb{Q}$. We want $\mathbb{Q}$ to be flexible enough so that $p(z|x)$ can be approximated closely but simple enough so that the problem is still tractable.

Our goal is to be able to approximate a conditional density of the latent variables $p(z|x)$ given observed variables $\mathbf{x}$.

Our strategy would be to use a family of densities over the latent variables, parametrizing them "free" variational parameters. Optimization will find the member of this family with the setting of these parameters that is closest in KL divergence to $p(z|x)$

A.1.3 Evidence Lower Bound

Consider the KL divergence equation:

$$\begin{aligned}
\text{KL}(q(z)||p(z|x)) &= \int q(z)\log(\frac{q(z)}{p(z|x)}) \\
&= \mathbb{E}[\log(q(z))] - \mathbb{E}[\log(p(z|x))] \\
&= \mathbb{E}[\log(q(z))] - \mathbb{E}[\log(\frac{p(x,z)}{p(x)})] \\
&= \mathbb{E}[\log(q(z))] - \mathbb{E}[\log(p(x,z))] + \mathbb{E}[\log p(x)] \\
&= \mathbb{E}[\log(q(z))] - \mathbb{E}[\log(p(x,z))] + \log p(x)
\end{aligned}$$

However the above equation is intractable because $\log(p(x))$ is intractable. So we optimize an alternate function: $ELBO(q)$

$$\begin{aligned}
ELBO(q) &= \mathbb{E}_q[\log(p(x,z))] - \mathbb{E}_q[\log q(z)] \\
&= \mathbb{E}_q[\log(p(z|x)p(x))] - \mathbb{E}_q[\log q(z)] \\
&= \mathbb{E}_q[\log(p(z|x))] + \mathbb{E}_q[\log(p(z))] - \mathbb{E}_q[\log q(z)] \\
&= \mathbb{E}_q[\log(p(x|z))] + \mathbb{E}_q[\log(p(z))] - \mathbb{E}_q[\log q(z)] \\
&= -KL(q(z)||p(z)) + \mathbb{E}_q[\log p(x|z)]
\end{aligned}$$

ELBO lower bounds the (log) evidence. Assuming we have proven that KL divergence is non-negative $KL(q(z)||p(z)) \geq 0$, we get $\log(p(x)) \geq ELBO(q)$

So instead we maximize this $ELBO(q)$ function. The first term of the last equation implies we are penalizing $q(z)$ to be different from $p(z)$ and at the second term implies we are trying to maximize the expected log likelihood of observing x through $p(x|z)$.

The first term in the original $ELBO(q)$ equation is trying to maximize the expected complete log likelihood, which would generally involve an EM solution. When $q(z) = p(z|x)$ then the ELBO is simply $log(p(x))$. So in a traditional EM setup, the E step would involve computing $p(z|x)$ assuming it is tractable unlike in variational inference.

A.2 Autoencoders

Autoencoders are a part of the “Generative modeling" paradigm in Machine learning research, where the aim is to model the distribution $P(X)$ given datapoints X in some high dimensional space $\mathbb{R}^m$. They can be thought of neural networks that essentially try to copy the input to output.

More formally, autoencoders consist of an encoding and a decoding layer. Given an input $\mathbf{x}$, the encoding layer represents the input into its coded form $h = f(\mathbf{x})$ while the decoding layer provides a reconstruction of the original input $r = g(h) = g(f(x)) \approx x$ such that a loss $L(g(f(x)), x)$ is minimized, where L is a loss function that penalizes $g(f(x))$ for being dissimilar to x. Autoencoders with dimensions less than the original dimension of input $\mathbf{x}$ are called undercomplete autoencoders.

A.2.1 PCA

When the decoding function is linear and the loss is mean squared loss, the undercomplete autoencoders is known to span the same subspace as PCA.

A.2.2 A more intuitive explanation

The overall aim of autoencoders is to be able to model the true probability distribution $P_{true}(X)$ of a given dataset X. Since $P_{true}(X)$ is unknown, our aim is to learn an alternate distribution $Q(X)$ such that $Q(X)$ is "close" to $P_{true}(X)$. We want to avoid putting restrictions on the structure of data and we also want the solution to be tractable (in a computationally expensive context) even for large datasets.

A.2.3 Sparse Autoencoders

Sparse autoencoders are autoencoders with a sparsity penalty $\Omega(h)$ on the encoding layer $h = f(x)$.

$$L(\mathbf{x}, g(f(\mathbf{x})) + \Omega(h)$$

The sparsity constraint ensures that the autoencoder learns unique statistical properties of the data, rather than acting just as a copy-paste function.

We will focus on a special category of autoencoders called the “Variational Autoencoders" (VAE) for the rest of our sections.

A.2.4 Sparse Autoencoders and Latent Variables

The sparsity constraint in Sparse Autoencoders can be understood as an approximation of the maximum likelihood training of a generative model that has latent variables.

Latent Variables

Consider a setting where our dataset includes objects like trees, house, and cat and dog pictures. We want to be able to generate images which looks the ones in our dataset, but not exactly the same. If we are able to generate the left half of a tree, then the right half of it will be very different from the right half of a dog. Before we generate any image, it is useful (and essential) to think about the kind of image we

are going to generate to ensure that the pixels are coherent with the image we are going to generate.

Before our model generates these images, it will randomly pick up collection of pixels z from $\{\text{tree}, \text{house}, \text{cat}, \text{dog}\}$ and make sure that all pixels are coherent in some sense. For example to generate a dog's image, you do not expect lot of green patches that would resemble a tree. This z is a latent variable. It is latent because given a new generated image, we do not know which settings of the latent variable generated the final image. z for example could be the patten of edges in our images.

A.2.5 Model

For the model to be representative of the data X, there should be at least on configuration of the latent variable z such that the data generated resembles all data points in X.

More formally, the distribution $P(X)$ is modeled using a generative process conditioned on the latent variable Z and we want to maximize $P(X)$. We have a way of sampling Z using some distribution $P(z)$ and this a deterministic function $f(z;\theta)$ parameterized by θ. We wish to optimize θ, such that $f(z;\theta)$ generates samples that look like X

We want to maximize $P(X)$ for each X in the training set, to be able to hope that this model will be able to generate "similar" samples.

$$P(X) = \int P(z)P(X|z;\theta)dz$$

where $P(X|z;\theta) = f(z;\theta)$.

In VAEs, the choice of this function $P(X|z;\theta) = \mathcal{N}(X|f(z;\theta), \sigma^2 I)$, but this is not a hard requirement. If X is binary, $P(X|z;\theta)$ could be bernoulli. It can be any distribution other than a dirac-delta, as long as it is computable.

VAEs solve this problem of "generative modeling" by telling us what our choice of latent variable z should be and how to overcome the integral involved over all such values of z.

Before our model starts generating any images, it needs to make some decisions. It needs to first decide what image it will be generating, than decide the angles, stroke width etc. We want to avoid deciding such properties by hand and we also want to avoid describing any sort of dependencies this latent variable should capture (the house images for example, always has more edges). VAEs take a very unintuitive approach of dealing with this by imposing a gaussian distribution on $z \sim \mathcal{N}(0, \sigma^2 I)$. How does that work?

Any distribution in d dimensions can be generated by taking a set of d gaussian random variables and passing them through some non-linear transformation. For example we can generate a 2D ring starting with a 2D multivariate gaussian using this transformation: $g(z) = z/\alpha + z/||z||$ where $z \sim N(0, \sigma^2 I)$ and α is a shrinkage parameter.

An example is given below:

A.2.6 Objective function

For most choices of z $P(X|z)$ will be close to zero. So VAEs tend to sample only those values of z which are more probably of generating X. So we are interested in a function $Q(z|X)$ that takes the data X and gives us a distribution over z that are

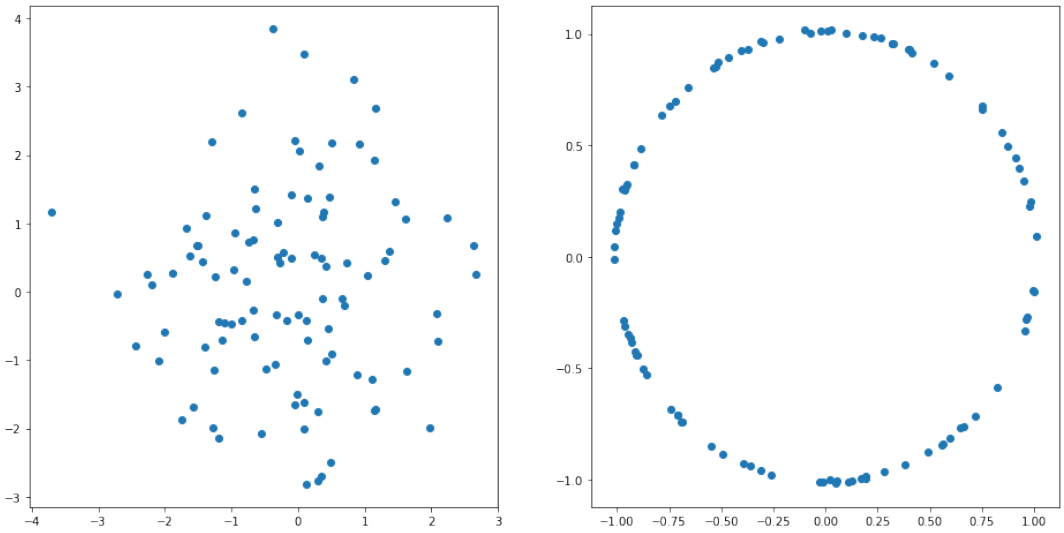

FIGURE A.1: It is possible to transform a gaussian (left) to look like a ring (right)

likely to produce X. Ideally, the space spanned by $Q(z|X)$ is much smaller than the "prior" distribution $P(z)$.

$$
\begin{aligned}
KL(Q(z)||P(z|X)) &= E_{z\sim Q}[\log Q(z) - \log P(z|X)] \\
P(z|X) &= \frac{P(X|z)P(z)}{P(X)} \\
KL(Q(z)||P(z|X)) &= E_{z\sim Q}[\log Q(z) + \log P(X) - \log P(X|z) - \log P(z)] \\
KL(Q(z)||P(z|X)) &= E_{z\sim Q}[\log Q(z) - \log P(z)] + \log P(X) - E_{z\sim Q}[\log P(X|z)] \\
KL(Q(z)||P(z|X)) &= KL(Q(z)||P(z)) + \log P(X) - E_{z\sim Q}[\log P(X|z)] \\
\log P(X) - KL(Q(z)||P(z|X)) &= E_{z\sim Q}[\log P(X|z)] - KL(Q(z)||P(z))
\end{aligned}
$$

We want to maximize the quantity on left hand side because we want to maximize $\log(P(x))$, however we also want to penalize $Q(z)$ if it does not resemble the "true" unknown distribution $P(z|X)$ Alternatively, our loss function for performing gradient descent is given by:

$$KL(Q(z)||P(z)) - E[\log P(X|z)]$$

where the first term denotes the information lost in representing $P(z)$ as $Q(z)$ and the second term denotes the reconstruction loss

So we want $KL(Q(z)||P(z|X))$ to be minimized. If we find a magic function $Q(z)$ which can approximate $P(z|X)$ perfectly, $KL(Q(z)||P(z|X)) = 0$.

A.2.7 What should be ideal $Q(z|X)$?

Answer: $Q(z|X) = \mathcal{N}(z|X, \sum)$ just works.

To perform gradient descent on the right hand side, we take one sample of z and treat one sample of that as an approximation of $E[\log P(X|z)]$, this is performed over all datapoints, say D

$$E_{X\sim D}[\log P(X) - KL(Q(z)||P(z|X))] = E_{X\sim D}[E_{z\sim Q}[\log P(X|z)] - KL(Q(z)||P(z))]$$

We can just take the gradient of this equation and everything looks okay. However, during backpropogation, we will encounter a layer that is sampling z from $Q(z|X)$ which is itself a stochastic unit. In order to be able to sample from $Q(z|X) \sim \mathcal{N}(\mu(X), \sum(X))$ we first sample $\epsilon \sim \mathcal{N}(0, I)$ and then compute $z = \mu(X) + \sum^{1/2}(X)\epsilon$. This is called the re-parameterization trick. This was the main contribution of the original VAE paper (13).

A.2.8 KL divergence between two gaussians (Used above for calculating losses)

Assume $q \sim \mathcal{N}(\mu_0, \sigma_0^2)$ and $p \sim \mathcal{N}(\mu_1, \sigma_1^2)$:

$$\begin{aligned}
KL(q||p) &= \int q(x)(\log(q(x) - \log(p(x))dx \\
q(x) &= (2\pi\sigma_0^2)^{-\frac{1}{2}} \exp \frac{-(x-\mu)^2}{2\sigma_0^2} \\
\log q(x) &= -\frac{1}{2}\log 2\pi - \log(\sigma_0) - \frac{-(x-\mu)^2}{2\sigma_0^2} \\
KL(q||p) &= \int q(\log \frac{\sigma_1}{\sigma_0} + \frac{(x-\mu_1)^2}{2\sigma_1^2} - \frac{(x-\mu_0)^2}{2\sigma_0^2})dx \\
\int q\frac{(x-\mu_0)^2}{2\sigma_0^2}dx &= \frac{1}{2\sigma_0^2} \\
\int q\frac{(x-\mu_1)^2}{2\sigma_1^2}dx &= \frac{1}{2\sigma_1^2}(\int x^2 q dx + \int q\mu_1^2 - \int 2\mu_1 x q dx) \\
&= \frac{EX^2 + \mu_1^2 - 2\mu_1 EX}{2\sigma_1^2} \\
&= \frac{\sigma_0^2 + \mu_0^2 + \mu_1^2 - 2\mu_1\mu_0}{2\sigma_1^2} \\
&= \frac{\sigma_0^2 + (\mu_1 - \mu_0)^2}{2\sigma_1^2} \\
KL(q||p) &= \log \frac{\sigma_1}{\sigma_0} - \frac{1}{2\sigma_0^2} + \frac{\sigma_0^2 + (\mu_1 - \mu_0)^2}{2\sigma_1^2}
\end{aligned}$$

This section borrows a lot of its contents from these papers: (14) and (13).

Bibliography

[1] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color Transfer between Images," *IEEE Computer Graphics and Applications*, p. 8, 2001. 01795.

[2] A. Vahadane, T. Peng, S. Albarqouni, M. Baust, K. Steiger, A. M. Schlitter, A. Sethi, I. Esposito, and N. Navab, "Structure-preserved color normalization for histological images," pp. 1012–1015, IEEE, Apr. 2015. 00009.

[3] M. Macenko, M. Niethammer, J. S. Marron, D. Borland, J. T. Woosley, Xiaojun Guan, C. Schmitt, and N. E. Thomas, "A method for normalizing histology slides for quantitative analysis," pp. 1107–1110, IEEE, June 2009. 00188.

[4] J. Xu, L. Xiang, G. Wang, S. Ganesan, M. Feldman, N. N. Shih, H. Gilmore, and A. Madabhushi, "Sparse Non-negative Matrix Factorization (SNMF) based color unmixing for breast histopathological image analysis," *Computerized Medical Imaging and Graphics*, vol. 46, pp. 20–29, Dec. 2015.

[5] Y. Al-Kofahi, W. Lassoued, W. Lee, and B. Roysam, "Improved Automatic Detection and Segmentation of Cell Nuclei in Histopathology Images," *IEEE Transactions on Biomedical Engineering*, vol. 57, pp. 841–852, Apr. 2010. 00466.

[6] Y. Boykov and G. Funka-Lea, "Graph Cuts and Efficient N-D Image Segmentation," *International Journal of Computer Vision*, vol. 70, pp. 109–131, Nov. 2006.

[7] T. Lindeberg, "Feature Detection with Automatic Scale Selection," p. 53. 03047.

[8] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, "Deep Learning for Identifying Metastatic Breast Cancer," *arXiv:1606.05718 [cs, q-bio]*, June 2016. 00103 arXiv: 1606.05718.

[9] Y. Liu, K. Gadepalli, M. Norouzi, G. E. Dahl, T. Kohlberger, A. Boyko, S. Venugopalan, A. Timofeev, P. Q. Nelson, G. S. Corrado, J. D. Hipp, L. Peng, and M. C. Stumpe, "Detecting Cancer Metastases on Gigapixel Pathology Images," *arXiv:1703.02442 [cs]*, Mar. 2017. 00045 arXiv: 1703.02442.

[10] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," pp. 1–9, IEEE, June 2015.

[11] Y. Li and W. Ping, "Cancer Metastasis Detection With Neural Conditional Random Field," p. 9. 00001.

[12] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," p. 10. 05434.

[13] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv:1312.6114 [cs, stat]*, Dec. 2013. arXiv: 1312.6114.

[14] C. Doersch, "Tutorial on Variational Autoencoders," *arXiv:1606.05908 [cs, stat]*, June 2016. arXiv: 1606.05908.