Machine Learning Engineer Nanodegree: Capstone Proposal

Detect and classify breast cancer metastases

Jörg Horchler

Version 1.0, 2019-05-18

Table of Contents

1. Domain Background

According to World Health Organization, breast cancer is the most common cancer among women worldwide. In the United States every 2 minutes a woman is diagnosed with breast cancer and 1 woman will die of breast cancer every 13 minutes on average.

As the lymph nodes in the axilla are most likely the first place breast cancer spreads to these nodes are often removed and examined microscopically. This procedure is demanding and time-consuming. In addition small metastases are very difficult to detect and sometimes missed.

Machine learning has shown that image classification and object detection can be automated. Using machine learning for that would hold great promise to reduce the workload of pathologists, and reduce the subjectivity in diagnosis.

Inspired by the lesson *Deep Learning for Cancer Detection* I decided to implement an algorithm for detection of breast cancer metastases in images of histological lymph node sections.

Research in this field was done for example in the two grand challenges CAMELYON16 and CAMELYON17.

2. Problem Statement

In this project the goal is to develop an algorithm for detection of metastases in hematoxylin and eosin stained images of lymph node sections. For this the algorithm should be able to discriminate between images with and without metastasis. The result of the task will be the probability of every image to contain metastasis. These will be compared to the ground truth of the dataset.

3. Datasets and Inputs

The dataset used for this project is based on the dataset of the CAMELYON16 challenge. As mentioned in the original challenge the base dataset

> contains a total of 400 whole-slide images (WSIs) of sentinel lymph node from two independent datasets collected in Radboud University Medical Center (Nijmegen, the Netherlands), and the University Medical Center Utrecht (Utrecht, the Netherlands).

As this dataset is very large (around 2 GB each image) Bas Veeling created a smaller dataset of these images by creating patches of them. They can be found at GitHub. The readme describes the modifications as:

> PCam is derived from the Camelyon16 Challenge, which contains 400 H&E stained WSIs of sentinel lymph node sections. The slides were acquired and digitized at 2 different centers using a 40x objective (resultant pixel resolution of 0.243 microns). We undersample this at 10x to increase the field of view. We follow the train/test split from the Camelyon16 challenge, and further hold-out 20% of the train WSIs for the validation set. To prevent selecting background patches, slides are converted to HSV, blurred, and patches filtered out if maximum pixel saturation lies below 0.07 (which was validated to not throw out tumor data in the training set). The patch-based dataset is sampled by iteratively choosing a WSI and selecting a positive or negative patch with probability p. Patches are rejected following a stochastic hard-negative mining scheme with a small CNN, and p is adjusted to retain a balance close to 50/50.

The data is stored in gzipped HDF5 files. Train, test and validate split was done as:

- test_x: 32768 x 96 x 96 x 3

- test_y: 32768 x 1 x 1 x 1
- train_x: 262144 x 96 x 96 x 3
- train_y: 262144 x 1 x 1 x 1
- valid_x: 32768 x 96 x 96 x 3
- valid_y: 32768 x 1 x 1 x 1

The information for all x-Files contain

- first number denotes the number of entries/images
- second and third numbers are the image/patch dimensions
- last number is the number of channels

The y files contain the labels.

4. Solution Statement

I plan to use deep learning to tackle the problem. Specifically, I want to use the images of the dataset to train a Group equivariant Convolutional Neural Network (G-CNN) that is introduced by arXiv:1602.07576. As written on arXiv:

> G-CNNs use G-convolutions, a new type of layer that enjoys a substantially higher degree of weight sharing than regular convolution layers. G-convolutions increase the expressive capacity of the network without increasing the number of parameters. Group convolution layers are easy to use and can be implemented with negligible computational overhead for discrete groups generated by translations, reflections and rotations.

I decided to implement this because

1. PCam is described as trainable on a single GPU.
2. A G-CNN was used in arXiv:1806.03962 (the paper introducing PCam) as well.

I plan to implement that in PyTorch.

5. Benchmark Model

The benchmark model used is a Inception v3 CNN. This model is used as comparison because the best score of the *Camelyon16* challenge was achieved by **Harvard Medical School and MIT** using an Inception model. In addition Inception v3 can be used by transfer learning from torchvision.

6. Evaluation Metrics

The *area under the ROC curve (receiver operating characteristic curve)* is proposed as the metric to be used to quantify the performance of the model. It is used because this metric was used for the *Whole-slide-image classification* task in the *Camelyon16* challenge.

The ROC curve plots

- the true positive rate (aka recall) against

- the false positive rate

True positive rate is defined as

TPR = TP/TP + FN

False positive rate is defined as

FPR = FP/FP + TN

An ROC curve plots TPR vs. FPR at different threshold settings, where the threshold is a value between 0 and 1. A lower threshold classifies more items as positive, in other words increases both False Positives and True Positives.

The **AUC** measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1). AUC ranges in value from 0 to 1 where a model whose predictions are 100% wrong has 0.0; one whose predictions are 100% correct has an AUC of 1.0.

7. Project Design

- For the implementation of the above described solution I'll first have to acquire the dataset.
- I then need to write a **Dataset** Class for PyTorch to read in the HDF5 files as PyTorch has not loader available.
- The next step would be to inspect some images comparing the patches of *PCam* and the original images of *Camelyon16*.
- For reference I'll provide some statistics of the train/test/validate data, for example the total number of images containing metastases etc.
- Then I'll use transfer learning to train an Inception CNN on the dataset.
- Performance (**AUC**) of this network is shown as baseline for the own network.
- This model will be minimally tuned to test various parameters.
- Now the next step to be done is to create the own G-CNN which will be trained with the dataset.
- This model will be tuned as well as it is not using transfer learning.
- I'll try different parameters to try to improve the performance of the network.
- Performance of the different networks will be show to compare them with the Inception baseline.