Udacity Machine Learning Engineer Capstone Project Report: Histhopathologic Cancer Detection

Arief Ramadhan

April 2020

1 Problem Definition

This project aims to identify tumors in small image patches taken from larger digital pathology scans [1]. A tumor is an abnormal growth in solid tissue such as in organ, muscle, or bone. Tumor indicates the existence of cancer or cells which divide uncontrollably. Early detection and treatment of cancer play a crucial role in the patient's survival rate. By the time symptoms appear, cancer may have begun to spread and be harder to treat.

The field of digital pathology has facilitated image analysis algorithms to assist and automate diagnostic tasks [1]. One of the algorithms is deep learning, which mimics the way people learn and is designed to recognize patterns.

In this study, we use Convolutional Neural Networks (CNNs) to predict whether an image contains at least one pixel of tumor [1]. This deep learning algorithm is most commonly applied to analyze images and has been shown to outperform pathologists in a variety of diagnostic tasks [1]. Our motivation is to help improve early diagnosis of cancer to increase the patient's survival rate.

1.1 Problem Statement

Given images of digital pathology scans, determine whether there is at least one pixel of tumor tissue in the images. We approach the problem as a binary classification of a "tumor(1)" class and a "no tumor(0)" class.

1.2 Metrics

The main evaluation metric of this problem is the classification accuracy, which is defined as the number of correct classification divided by the total number of classification. We will compare our network classification accuracy with the benchmark model classification accuracy. The formula of accuracy is displayed in Equation 1.

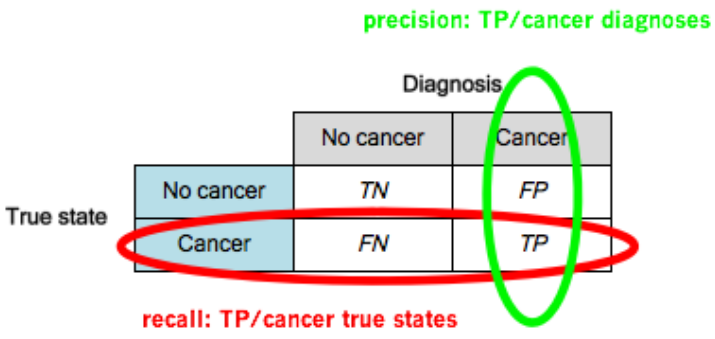

Figure 1: Precision and Recall Calculation for Cancer Detection Task [2]

We calculate the precision, recall, and Area Under Curve (AUC) of our network as further evaluation metrics. In our case, precision is the number of correct tumor detection divided by the total number of tumor detection. Recall indicates the number of correct tumor detection divided by the number of tumor images in the dataset.

Figure 1 illustrates how precision and recall are calculated for a cancer detection task. The mathematical formula for precision and recall are given by Equation 2 and Equation 3, respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Where:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

AUC represents the degree or measure of separability. It tells how much model is capable of distinguishing between classes. AUC values lie between 0.5 and 1, and a high AUC corresponds to a better classifier performance is. In our case, the higher the AUC, the better the model at distinguishing patients with cancer and no cancer. Unlike the accuracy metric, we will not compare the precision, recall, and AUC values of our network with those of the benchmark model.

2 Analysis

2.1 Datasets and Inputs

The dataset is a combination of two independent datasets collected in Radboud University Medical Center (Nijmegen, the Netherlands), and the University Medical Center Utrecht (Utrecht, the Netherlands). It is provided by Kaggle for a past competition [3]. It consists of a large number of small pathology images to classify. The link to the dataset can be found here.

There are 220,025 training images in the training set; about 60% of them are from normal patients, and the remaining 40% contains tumor pixels. Each image is a 3-channel color image with a size of 96x96 pixels in ".tiff" format. Our task is to classify and label the 57,458 images in the test set.

2.2 Benchmark Model

We find a variety of projects on Github that solved this particular problem with different strategies, as displayed in Table 1. This benchmark acts a reference that will serve to assess how our final model perform.

Table 1: Past Projects on Github

Author	Model	Accuracy
O. Boom	Resnet50	80.89%
G. Surma	Xception + NasNet	95.8%
K. Garima	Xception	93.74%

2.3 Algorithms and Techniques

We solve this problem by using a Convolutional Neural Network (CNN) via transfer learning. We use the VGG-16 model, which has 16 trainable layers and was introduced in the ImageNet Large Scale Visual Recognition Challenge in 2014. Currently, VGG-16 has been used to detect cancer through medical scans such as in [4], [5], and [6]. Figure 2 illustrates the layers of a VGG-16 Architectures.

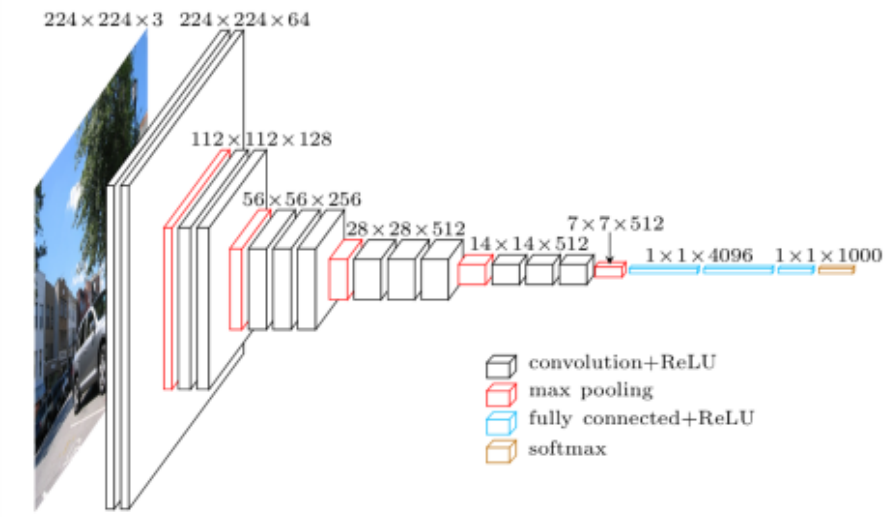

Figure 2: VGG-16 Architectures

3 Methods

3.1 Preprocessing

There are two steps in our image preprocessing: (1) Image cropping, and (2) Normalization.

3.1.1 Image Cropping

The Kaggle's Histopathologic Cancer Detection competition rule describes that a positive label indicates that the center 32x32 region of the image contains at least one pixel of tumor tissue. Tumor tissue in the outer region of the patch does not influence the label.

Therefore, the first thing we do is cropping the image into a smaller size. Schettler found that cropping into a size smaller than 42x42 pixels hurt the classification accuracy. This is because he performed data augmentation, which modified the image with random rotation, zooming, and shift that change the tumor position from the center so the tumor pixel might be lost after cropping.

Schettler suggested the ideal size of the image to be 48x48 pixels. We adapted this size for our images. Figure 3 shows the 32x32 center image from the 96x96 pixel images. After cropping, the 32x32 pixel within the 48x48 pixel image is displayed in Figure 4.

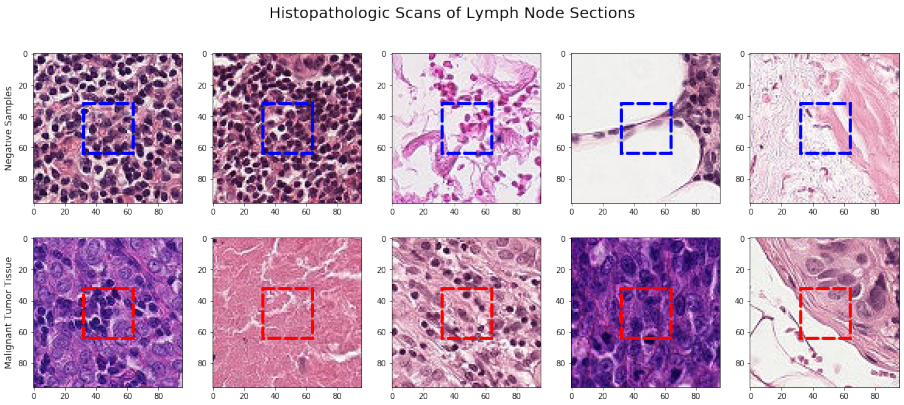

Figure 3: 32x32 center pixel inside a 96x96 image [7]

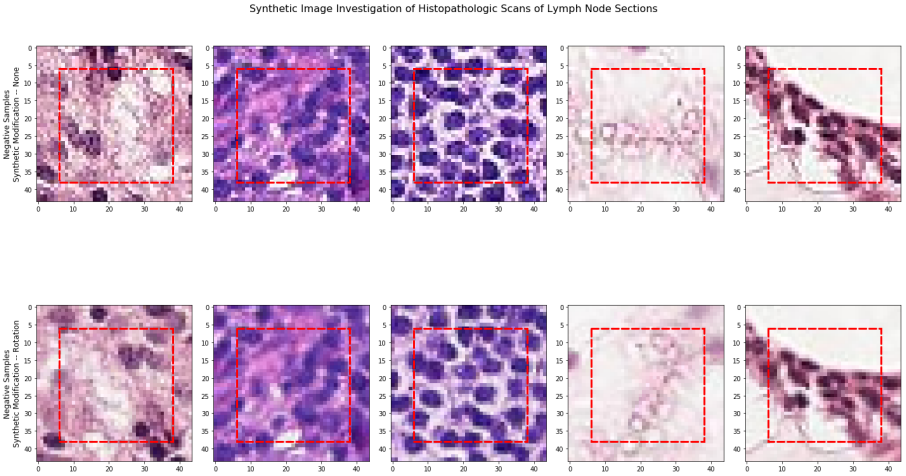

Figure 4: 32x32 center pixel inside a 48x48 image [7]

Next, we split our training data into a training set and a validation set. There are no rules on how much validation set should be taken from the training set. As we have a relatively large dataset, it is recommended to have a large proportion of the validation set. We use 80% data for training and 20% for the validation. In each set, we divide the data into separate folders corresponding to 1 or 0 class.

3.1.2 Data Normalization

We find that each pixel in the image corresponds to an integer value from 0 to 255. Such values would be too high for our model to process with a typical learning rate. We normalize the values into floats between 0 and 1 by a factor of 1/255.

```
In [8]: # Data normalization
        training_data_generator = ImageDataGenerator(rescale=1./255)
```

Figure 5: Data Normalization using Image Data Generator Library

To normalize the image data into our model, we use *image data generator* class from Keras, as displayed by Figure 5.

3.2 GPU

Compared to a CPU (Central Processing Unit), a GPU (Graphics Processing Unit) is better at fetching large amounts of memory. This feature makes GPU better than CPU in training

a deep learning model. In this project, we use the tensorflow-gpu library in jupyter notebook. We find that our GPU trains the network twenty times faster than the CPU. We have Acer 8GB Ram intel CPU and NVIDIA GX Force 250 GPU.

3.3 Model Building

We implement Keras deep learning libraries to build our model. At first, we plan to train only the additional layers that we put after the original VGG 16 layers. However, this causes the model to have low accuracy, as there are very few parameters to train. We then decide to retrain our VGG-16 after initializing it with Imagenet's weights. In total, there are about 15 million trainable parameters in our network. The network summary is displayed in Figure 6.

```
In [4]: model = VGG16_model()
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 1, 1, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 14,715,201
Trainable params: 14,715,201
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: Our VGG16-based Model

3.4 Postprocessing

After training the network, we generate predictions for our test set. The predictions are written into a CSV file, before being uploaded to Kaggle. This project was a competition at Kaggle; therefore, no information is provided about the exact test set label. We can only know how our predictions perform if we upload the CSV file into the competition page. Figure 7 displays our best predictions.

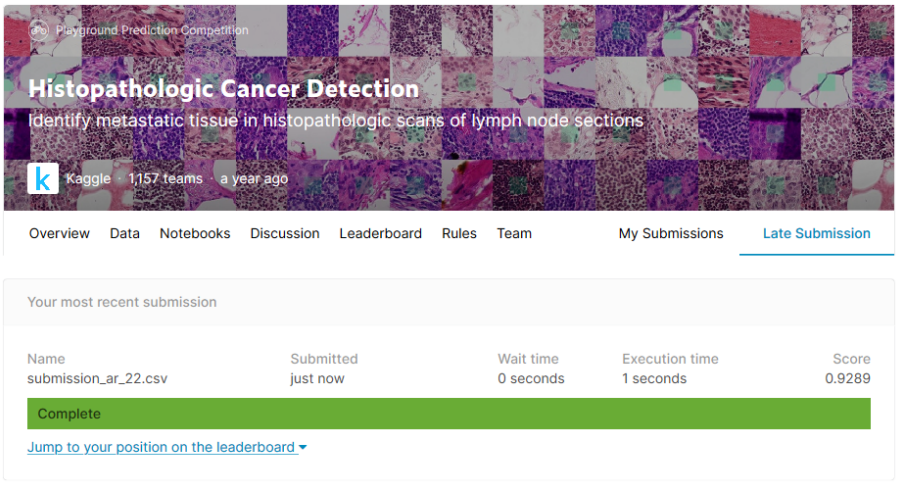

Figure 7: Kaggle Submission

4 Results and Discussions

After experimenting with different hyperparameters, we come up with our best model, which manages to predict the test set correctly 92.89% of the time. Table 2 sum up our model performance.

Table 2: Model's Performance

Metric	Value
Test Accuracy	92.89%
Precision (Validation Set)	90.51%
Recall (Validation Set)	87.35%

4.1 Accuracy

Figure 8 displays how the training accuracy compares with the validation accuracy. We train our model for ten epochs and select the model in which epoch the validation set is highest. In other words, we set the validation accuracy as the optimizing metric that decides whether the network should be updated after an epoch. After each epoch, the models will be updated with new weights only if the validation accuracy increases. We reach 91.16% validation accuracy after epoch number 7. After submitting our solution to Kaggle, we find that our model achieves **92.89% accuracy** in predicting the test set labels.

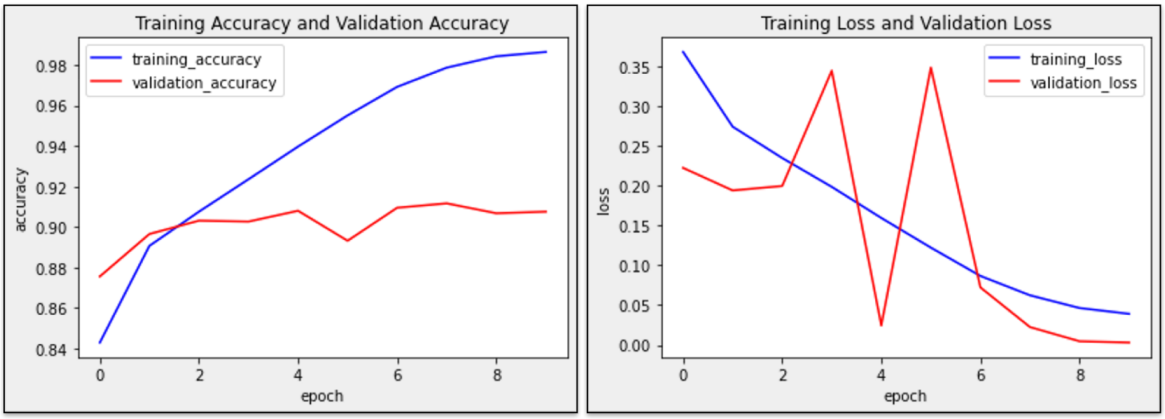

Figure 8: Training and Validation Accuracy and Loss

4.2 Precision, Recall, and AUC

The Kaggle Histopathologic Cancer Detection competition page does not provide us with any information regarding the test set correct labels. We can only know how our networks perform through uploading the generated test set prediction files into the competition page.

Therefore, we cannot evaluate the test set precision, recall, and AUC value. The second best thing we can do is to determine the precision, recall, and AUC of our validation set. We find that the validation set prediction has 90.51% precision, 87.35% recall, and 0.961 AUC. Figure 9 displays our model's ROC-AUC curve.

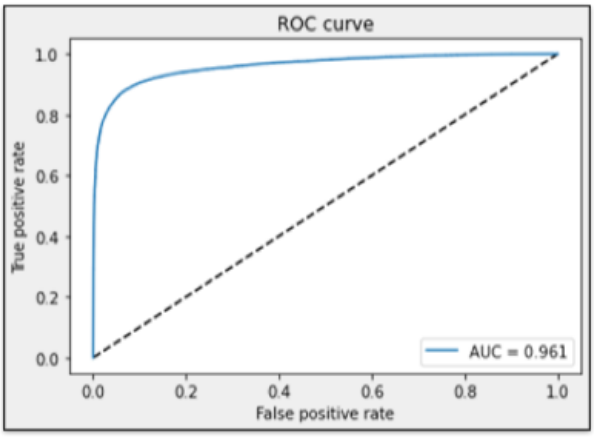

Figure 9: ROC-AUC Curve

4.3 Comparison to Benchmark

Table 3: Past Projects on Github

Author	Model	Accuracy
O. Boom	Resnet50	80.89%
G. Surma	Xception + NasNet	95.8%
K. Garima	Xception	93.74%
A. Ramadhan	**VGG-16**	**92.89%**

Table 3 displays how our Neural Network performs compared to the benchmark models. To recall, these benchmark models have been for the Kaggle's Histopathologic Cancer Detection problem that we aim to solve in this study.

Our model outperforms the Resnet50 based network built by Boom and has a lower but comparable accuracy to Surma and Montamat's model. To evaluate further, we investigate the number of network parameters used in Surma and Montamat models. We find that Surma and Montamat models have 25 million and 24 million trainable parameters respectively, while our model has 15 million.

The number of trainable parameters is the number of weights and biases that needs to be updated during training. Networks with less number of trainable parameters are more 'lightweight' or memory and time-efficient. It requires less space and less time to train. Therefore, the virtue of our networks is that it performs comparably with Surma and Montamat's model while requiring less memory and time to train (if trained with the same CPU/GPU).

5 Conclusions

We perform cancer image classification by taking advantage of VGG-16 transfer learning models. There are four steps. First, we explore the dataset and find 220,025 training images; 60% has a negative label (0), and 40% has a positive label (1).

The challenge rules stated that a positive label image indicates that the center 32x32 region contains at least one pixel of tumor tissue, and tumor tissue in the outer region does not influence the label. So we crop our image into a smaller size of 48x48 pixels, as recommended by Schettler's works. We also perform data normalization to accelerate model learning.

After that, we build our VGG-16 model by using a Keras library. We also add additional layers after the VGG-16 layers to accommodate our cancer detection task. Our test result

shows that our network performs with 92.89% accuracy. This result is about 3 percent lower than the best of the benchmark models. However, we achieve this accuracy by using 10 million fewer parameters. The number of parameters determines the time and memory cost.

As the competition page does not provide the test set solutions, we cannot evaluate the test set recall, precision, and Area Under Curve value. The second best thing we can do is to determine the precision and recall of our validation set. Our validation set prediction has 87.35% recall, 90.51% precision, and an AUC value of 0.961.

To conclude, the virtue of our networks is that it performs comparably with the benchmark models while requiring less memory and time to train (if trained with the same CPU/GPU).

6 Suggestions for Future Works

There are six points that we think might improve model performance:

1. Algorithm selection: Future projects should consider using other transfer learning models than VGG-16, such as VGG-19, Resnet50, Inception, and Nasnet.

2. Data augmentation: Data augmentation is a strategy to increase the diversity of data for training models, without actually collecting new data. This is done by techniques such as cropping, padding, and horizontal flipping. A work in [1] found that rotation equivariance significantly improves tumor detection performance. A similar method might be considered in future works.

3. K-Fold Cross-validation: K-fold cross-validation could replace the manual validation method that we use. Cross-validation is done by splitting the training data into k subsets and takes turns training the models on all subsets except one which is held out [8]. The process is repeated until all subsets are given an opportunity to be the held-out as the validation set [8].

4. Regularization strategy: regularization improves a model's generalization ability or to reduce variance. The higher the variance, the larger is the gap between the training set accuracy and the test set accuracy. In this work, we only use dropout for our model, as it is the most straightforward to implement. Future work might consider other techniques such as the L1 and L2 regularization.

5. Image preprocessing: the type of input plays a critical role in determining model accuracy, speed of convergence, and memory efficiency. In this study, we crop the image into a smaller size to ignore the outer part of the image, which does not influence the classification label. After that, we normalize each pixel value in order to speed up model learning. Future studies might consider another image preprocessing technique that might improve the model performance, such as contrast adjustment and Gaussian blur.

References

[1] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling, "Rotation equivariant cnns for digital pathology," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2018, pp. 210–218.

[2] J. Chan. (2012) What is the best way to understand the terms "precision" and "recall"? [Online]. Available: https://www.quora.com/What-is-the-best-way-to-understand-the-terms-precision-and-recall

[3] Kaggle. (2019) Histopathologic cancer detection. [Online]. Available: https://www.kaggle.com/c/histopathologic-cancer-detection/overview

[4] Q. Guan, Y. Wang, B. Ping, D. Li, J. Du, Y. Qin, H. Lu, X. Wan, and J. Xiang, "Deep convolutional neural network vgg-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: a pilot study," *Journal of Cancer*, vol. 10, no. 20, p. 4876, 2019.

[5] H. Chougrad, H. Zouaki, and O. Alheyane, "Deep convolutional neural networks for breast cancer screening," *Computer methods and programs in biomedicine*, vol. 157, pp. 19–30, 2018.

[6] D. Bychkov, N. Linder, R. Turkki, S. Nordling, P. E. Kovanen, C. Verrill, M. Walliander, M. Lundin, C. Haglund, and J. Lundin, "Deep learning based tissue analysis predicts outcome in colorectal cancer," *Scientific reports*, vol. 8, no. 1, pp. 1–11, 2018.

[7] D. Schettler. (2019) Histopathologic cancer detection. [Online]. Available: https://github.com/darien-schettler/histopathologic-cancer-detection

[8] J. Brownlee. (2018) A gentle introduction to k-fold cross-validation. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/