Pankaj Acharya
Capstone Project 2 - Milestone Report 1
Springboard Data Science Career Track
Mentored by Kevin Glynn
November 12, 2019

Cancer Detection in Histological Slides

OVERVIEW

Cancer is deadly and early diagnosis will play an important role in treatment and improvement of the patient's survival rate. Cancer can be benign or metastatic. One of the most important early diagnosis is detection in lymph nodes to find out whether the cancer has metastasized. The method to do this is H & E staining of histological slides of lymph nodes taken from biopsies.

GOALS

Currently pathologists manually examine the slides and decide if the patient has metastatic cancer or not. Because human judgement is not consistent and the diagnosis can vary between person to person and even between different days by the same person. Thus by developing deep learning algorithm we can automate the process and give unbiased results.

DATA SOURCE

The data for my project are downloaded from Kaggle website (https://www.kaggle.com/c/histopathologic-cancer-detection/data). Following data are provided:-

1. Sample_submission.csv - a sample submission file in the correct format
2. Train_labels.csv - a file with labels of 0 or 1 (0 for cancer not detected and 1 for cancer detected) for corresponding images in training dataset.

3. Train - a folder with 220,025 images from histopathological slides. These are the images I have to train my model on.
4. Test - a folder with 57,458 images. These are the images I will use to predict cancer detection.

EXPLORATORY DATA ANALYSIS

The training set has 220,025 images. The dataset is imbalanced (number of images in each class is not equal) as seen in figure 1 below:-

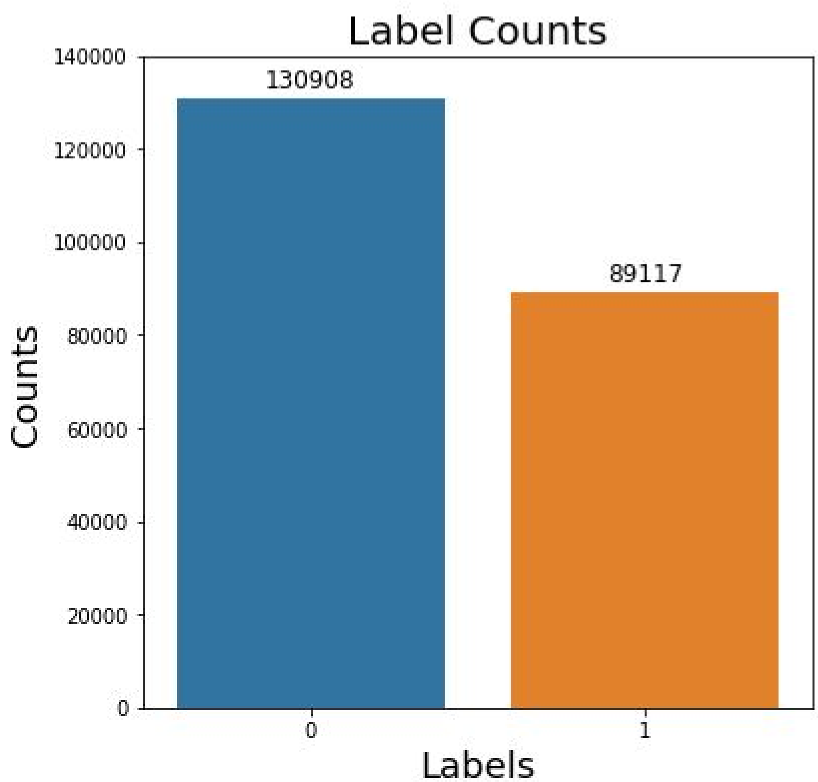

Figure 1: Distribution of images in the datasets

Images of normal tissue comprised of 59.5% and cancerous tissue only 40.5%.

Shown below, in figure 2, are representative images of normal (0) and cancerous (1) tissues.

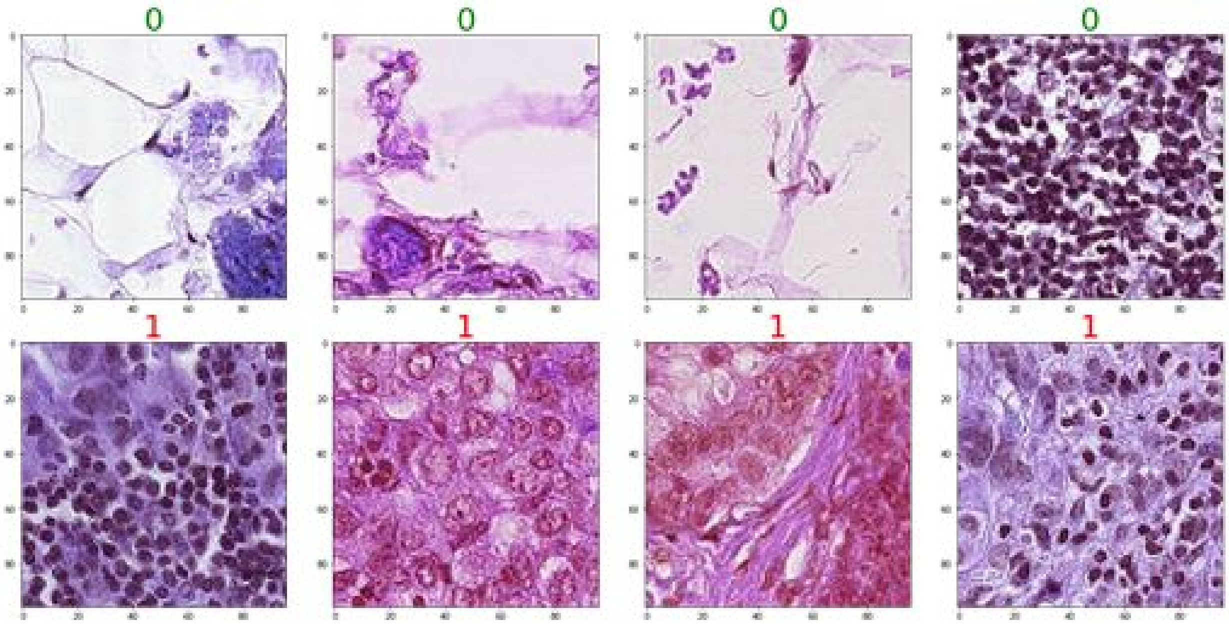

Figure 2: Representative images

DATA WRANGLING / SAMPLING OF IMAGES FOR TRAINING

Since the dataset is imbalanced and very large and neural networks take very long to train on all the datasets, I decided to sample 20,000 images in each class (a total of 40,000 images) to make the dataset balanced and smaller yet containing enough images to train my models. After sampling, I put them into separate folders to be consistent in training different models multiple times. Then I split the data 80/20 into training and validation sets. This will be my training and validation datasets for all the models.

CONVOLUTIONAL NEURAL NETWORK MODELS

I built five CNN models that were fed features extracted from the image sequences. Machine learning was performed using Python, primarily with Keras with TensorFlow in the backend on MacBookPro with 2.3 GHz Quad-core Intel Core i5 and 16 GB memory.

Model Descriptions:

Model: "sequential_1"

Layer (type)　　Output Shape　　Param #

===

conv2d_1 (Conv2D)	(None, 95, 95, 32)	416
conv2d_2 (Conv2D)	(None, 94, 94, 32)	4128
conv2d_3 (Conv2D)	(None, 93, 93, 32)	4128
max_pooling2d_1 (MaxPooling2	(None, 46, 46, 32)	0
conv2d_4 (Conv2D)	(None, 45, 45, 32)	4128
conv2d_5 (Conv2D)	(None, 44, 44, 32)	4128
conv2d_6 (Conv2D)	(None, 43, 43, 32)	4128
max_pooling2d_2 (MaxPooling2	(None, 21, 21, 32)	0
conv2d_7 (Conv2D)	(None, 20, 20, 64)	8256
conv2d_8 (Conv2D)	(None, 19, 19, 64)	16448
conv2d_9 (Conv2D)	(None, 18, 18, 64)	16448
max_pooling2d_3 (MaxPooling2	(None, 9, 9, 64)	0

flatten_1 (Flatten) (None, 5184) 0

____

dense_1 (Dense) (None, 64) 331840

____

dropout_1 (Dropout) (None, 64) 0

____

dense_2 (Dense) (None, 2) 130

===

Total params: 394,178

Trainable params: 394,178

Non-trainable params: 0

____

Model: "sequential_2"

____

Layer (type) Output Shape Param #

===

conv2d_10 (Conv2D) (None, 95, 95, 32) 416

____

conv2d_11 (Conv2D) (None, 94, 94, 32) 4128

____

conv2d_12 (Conv2D) (None, 93, 93, 32) 4128

____

max_pooling2d_4 (MaxPooling2 (None, 46, 46, 32) 0

____

conv2d_13 (Conv2D)	(None, 45, 45, 32)	4128
conv2d_14 (Conv2D)	(None, 44, 44, 32)	4128
conv2d_15 (Conv2D)	(None, 43, 43, 32)	4128
max_pooling2d_5 (MaxPooling2	(None, 21, 21, 32)	0
conv2d_16 (Conv2D)	(None, 20, 20, 64)	8256
conv2d_17 (Conv2D)	(None, 19, 19, 64)	16448
conv2d_18 (Conv2D)	(None, 18, 18, 64)	16448
max_pooling2d_6 (MaxPooling2	(None, 9, 9, 64)	0
conv2d_19 (Conv2D)	(None, 8, 8, 128)	32896
conv2d_20 (Conv2D)	(None, 7, 7, 128)	65664
conv2d_21 (Conv2D)	(None, 6, 6, 128)	65664
max_pooling2d_7 (MaxPooling2	(None, 3, 3, 128)	0
flatten_2 (Flatten)	(None, 1152)	0

```
_________________________________________________________________
dense_3 (Dense)            (None, 64)             73792
_________________________________________________________________
dropout_2 (Dropout)        (None, 64)             0
_________________________________________________________________
dense_4 (Dense)            (None, 2)              130
=================================================================
Total params: 300,354
Trainable params: 300,354
Non-trainable params: 0
_________________________________________________________________
```

Model: "sequential_3"

```
_________________________________________________________________
Layer (type)               Output Shape           Param #
=================================================================
conv2d_22 (Conv2D)         (None, 95, 95, 32)     416
_________________________________________________________________
conv2d_23 (Conv2D)         (None, 94, 94, 32)     4128
_________________________________________________________________
conv2d_24 (Conv2D)         (None, 93, 93, 32)     4128
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 46, 46, 32)   0
_________________________________________________________________
conv2d_25 (Conv2D)         (None, 45, 45, 32)     4128
```

conv2d_26 (Conv2D)	(None, 44, 44, 32)	4128
conv2d_27 (Conv2D)	(None, 43, 43, 32)	4128
max_pooling2d_9 (MaxPooling2	(None, 21, 21, 32)	0
conv2d_28 (Conv2D)	(None, 20, 20, 64)	8256
conv2d_29 (Conv2D)	(None, 19, 19, 64)	16448
conv2d_30 (Conv2D)	(None, 18, 18, 64)	16448
max_pooling2d_10 (MaxPooling	(None, 9, 9, 64)	0
conv2d_31 (Conv2D)	(None, 8, 8, 128)	32896
conv2d_32 (Conv2D)	(None, 7, 7, 128)	65664
conv2d_33 (Conv2D)	(None, 6, 6, 128)	65664
max_pooling2d_11 (MaxPooling	(None, 3, 3, 128)	0
flatten_3 (Flatten)	(None, 1152)	0

dense_5 (Dense) (None, 64) 73792

---

dropout_3 (Dropout) (None, 64) 0

---

dense_6 (Dense) (None, 2) 130

===

Total params: 300,354

Trainable params: 300,354

Non-trainable params: 0

Model: "sequential_4"

Layer (type) Output Shape Param #

===

conv2d_34 (Conv2D) (None, 94, 94, 32) 896

---

conv2d_35 (Conv2D) (None, 92, 92, 32) 9248

---

conv2d_36 (Conv2D) (None, 90, 90, 32) 9248

---

max_pooling2d_12 (MaxPooling (None, 45, 45, 32) 0

---

dropout_4 (Dropout) (None, 45, 45, 32) 0

---

conv2d_37 (Conv2D)	(None, 43, 43, 64)	18496
conv2d_38 (Conv2D)	(None, 41, 41, 64)	36928
conv2d_39 (Conv2D)	(None, 39, 39, 64)	36928
max_pooling2d_13 (MaxPooling	(None, 19, 19, 64)	0
dropout_5 (Dropout)	(None, 19, 19, 64)	0
conv2d_40 (Conv2D)	(None, 17, 17, 128)	73856
conv2d_41 (Conv2D)	(None, 15, 15, 128)	147584
conv2d_42 (Conv2D)	(None, 13, 13, 128)	147584
max_pooling2d_14 (MaxPooling	(None, 6, 6, 128)	0
dropout_6 (Dropout)	(None, 6, 6, 128)	0
flatten_4 (Flatten)	(None, 4608)	0
dense_7 (Dense)	(None, 256)	1179904
dropout_7 (Dropout)	(None, 256)	0

dense_8 (Dense) (None, 2) 514

===

Total params: 1,661,186

Trainable params: 1,661,186

Non-trainable params: 0

Model: "sequential_5"

Layer (type)	Output Shape	Param #
conv2d_43 (Conv2D)	(None, 95, 95, 32)	416
conv2d_44 (Conv2D)	(None, 94, 94, 32)	4128
conv2d_45 (Conv2D)	(None, 93, 93, 32)	4128
max_pooling2d_15 (MaxPooling	(None, 46, 46, 32)	0
dropout_8 (Dropout)	(None, 46, 46, 32)	0
conv2d_46 (Conv2D)	(None, 45, 45, 64)	8256

conv2d_47 (Conv2D)	(None, 44, 44, 64)	16448
conv2d_48 (Conv2D)	(None, 43, 43, 64)	16448
max_pooling2d_16 (MaxPooling	(None, 21, 21, 64)	0
dropout_9 (Dropout)	(None, 21, 21, 64)	0
conv2d_49 (Conv2D)	(None, 20, 20, 128)	32896
conv2d_50 (Conv2D)	(None, 19, 19, 128)	65664
conv2d_51 (Conv2D)	(None, 18, 18, 128)	65664
max_pooling2d_17 (MaxPooling	(None, 9, 9, 128)	0
dropout_10 (Dropout)	(None, 9, 9, 128)	0
conv2d_52 (Conv2D)	(None, 8, 8, 128)	65664
conv2d_53 (Conv2D)	(None, 7, 7, 128)	65664
conv2d_54 (Conv2D)	(None, 6, 6, 128)	65664
max_pooling2d_18 (MaxPooling	(None, 3, 3, 128)	0

____

dropout_11 (Dropout)   (None, 3, 3, 128)   0

____

flatten_5 (Flatten)   (None, 1152)   0

____

dense_9 (Dense)   (None, 256)   295168

____

dropout_12 (Dropout)   (None, 256)   0

____

dense_10 (Dense)   (None, 2)   514

===

Total params: 706,722

Trainable params: 706,722

Non-trainable params: 0

____

Model performances:

	Val_loss	val_acc	roc_auc_scores
Model1	0.023002	0.865250	0.940067
Model2	0.018761	0.869687	0.945249
Model3	0.014563	0.871375	0.945899
Model4	0.025661	0.838375	0.921132
Model5	0.056837	0.853500	0.925100

Based on the above data, it looks like Model2 performed the best.

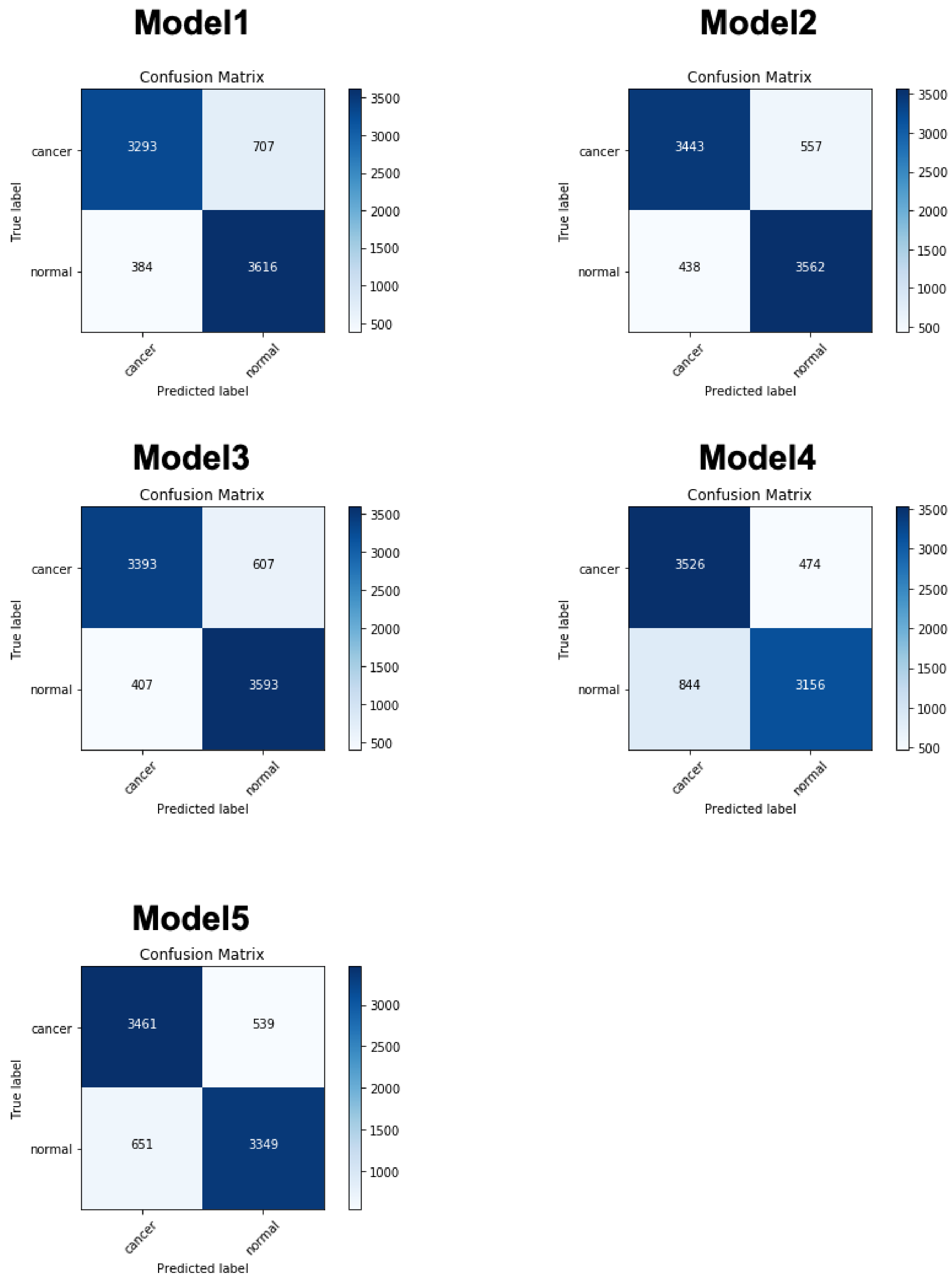

Figure 3: Confusion matrices of the model performances.

FUTURE DIRECTIONS:

Although the models that I built performed fairly well, I will try pretrained models ResNet and NasNet on this data, to see if they can improve performance.

PROJECT LINK