Pankaj Acharya
Capstone Project 2 - Final Report
Springboard Data Science Career Track
Mentored by Kevin Glynn
December 08, 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:-
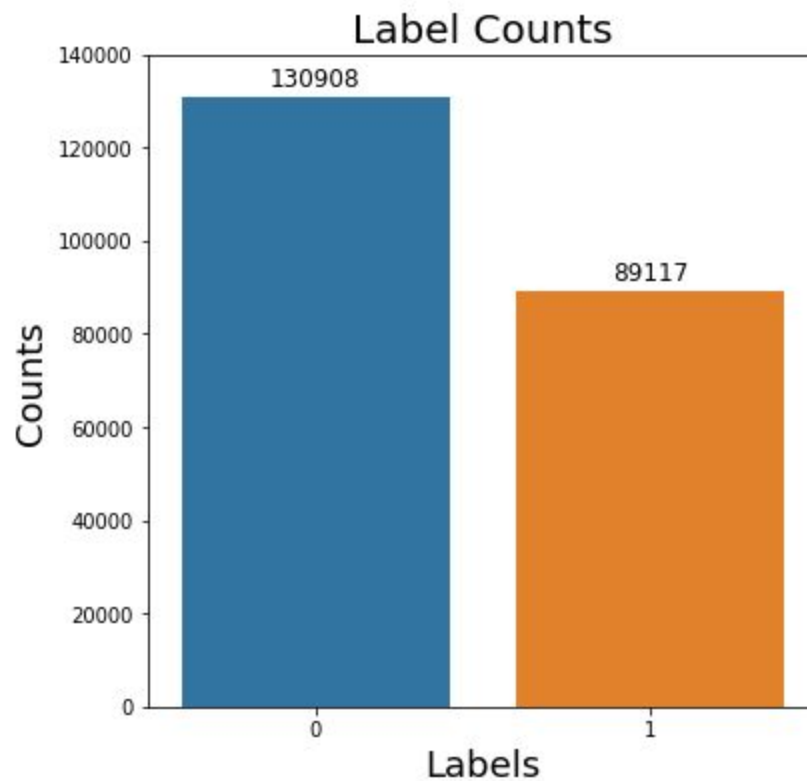

Figure 1: Distribution of images in the datasets

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.
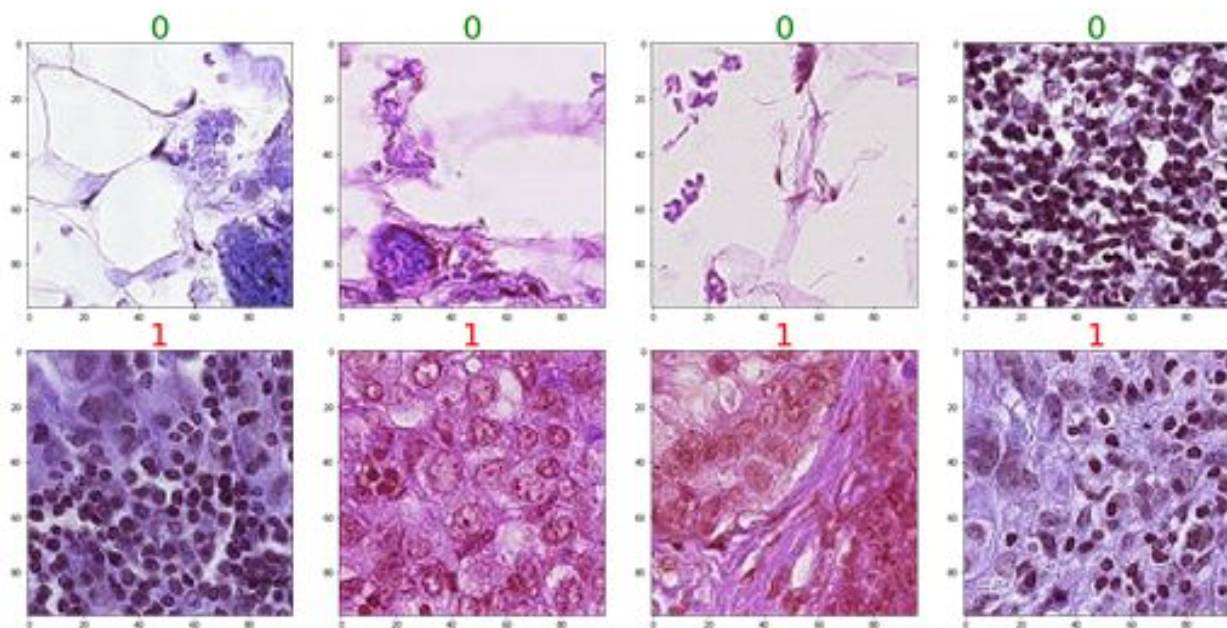
Figure 2: Representative images

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 and two pretrained 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-1: "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-2: "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-3: "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-4: "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-5: "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-6 ResNet50: "model_1"

```
_____
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 96, 96, 3)         0
_____
resnet50 (Model)             (None, 3, 3, 2048)        23587712
_____
global_average_pooling2d_1 ( (None, 2048)              0
_____
dropout_1 (Dropout)          (None, 2048)              0
_____
dense_1 (Dense)              (None, 2)                 4098
=================================================================
Total params: 23,591,810
Trainable params: 23,538,690
Non-trainable params: 53,120
_____
```

Model-7 NASNet: "model_1"

```
_____
Layer (type)                 Output Shape          Param #     Connected to
==================================================================================================
input_2 (InputLayer)         (None, 96, 96, 3)     0
_____
NASNet (Model)               (None, 3, 3, 1056)    4269716 input_2[0][0]
_____
global_max_pooling2d_1 (GlobalM (None, 1056)       0           NASNet[1][0]
_____
global_average_pooling2d_1 (Glo (None, 1056)       0           NASNet[1][0]
_____
```

```
flatten_1 (Flatten)          (None, 9504)          0               NASNet[1][0]
_____
concatenate_5 (Concatenate)  (None, 11616)         0

                                                             global_max_pooling2d_1[0][0]

                                                          global_average_pooling2d_1[0][0]

                                                                 flatten_1[0][0]
_____
dropout_1 (Dropout)          (None, 11616)         0               concatenate_5[0][0]

_____
3_ (Dense)                   (None, 2)             23234           dropout_1[0][0]
=================================================================================
Total params: 4,292,950

Trainable params: 4,256,212

Non-trainable params: 36,738
```

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
Model6	0.008133	0.935375	0.980530
Model7	0.031510	0.840813	0.925162

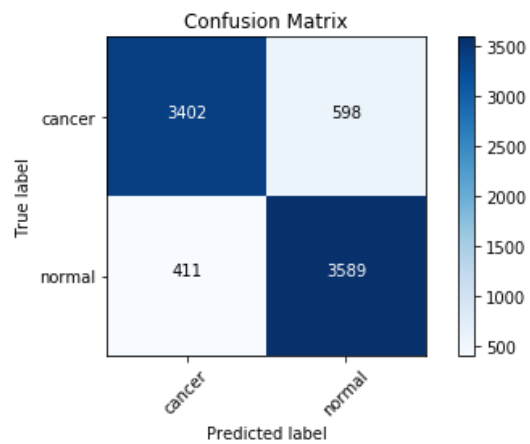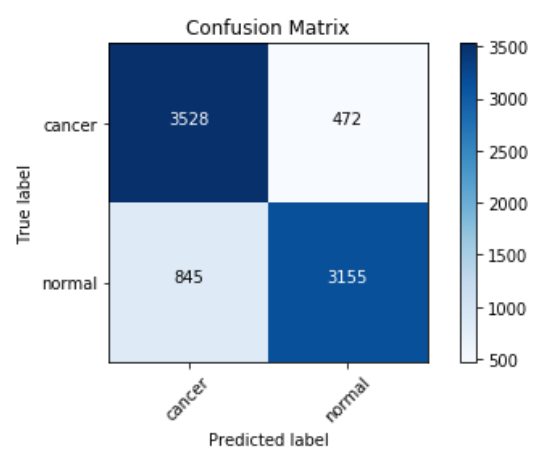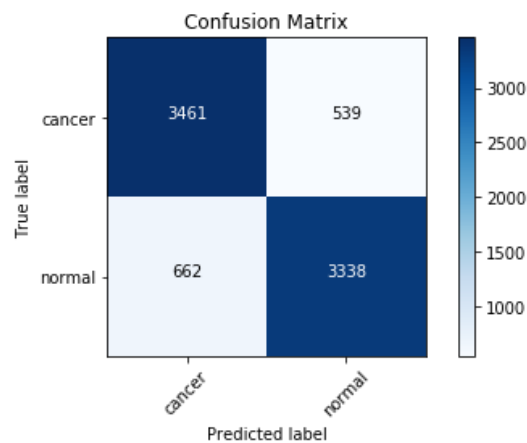Based on the above data, it looks like Model6 performed the best.

Model 1

Confusion Matrix

	cancer	normal
cancer	3315	685
normal	407	3593

True label / Predicted label

Model 2

Confusion Matrix

	cancer	normal
cancer	3430	570
normal	437	3563

True label / Predicted label

Model 3

Confusion Matrix

	cancer	normal
cancer	3402	598
normal	411	3589

True label / Predicted label

Model 4

Confusion Matrix

	cancer	normal
cancer	3528	472
normal	845	3155

True label / Predicted label

Model 5

Confusion Matrix

	cancer	normal
cancer	3461	539
normal	662	3338

True label / Predicted label

Model 6

Confusion Matrix

	cancer	normal
cancer	3682	318
normal	181	3819

True label / Predicted label
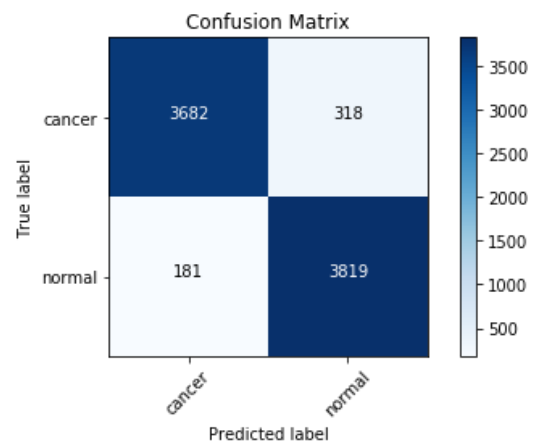
Model 7
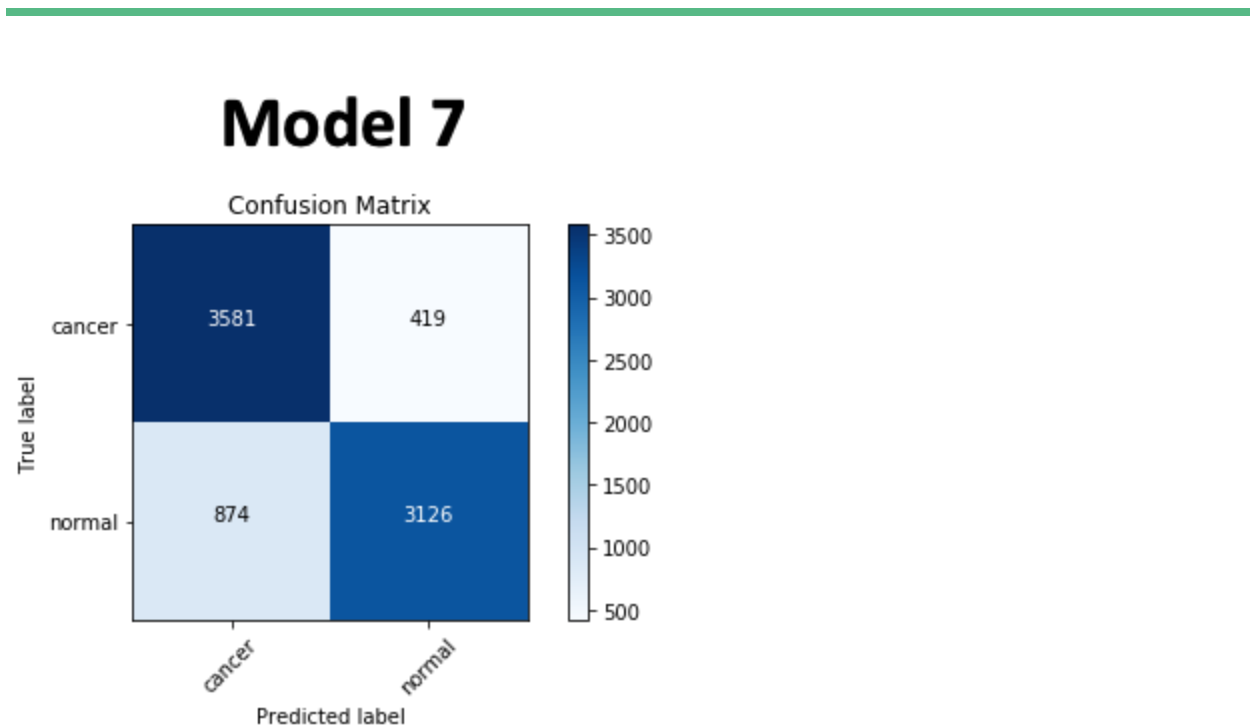

Figure 3: Confusion matrices of the model performances.

Table1: Precision and recall of the models tested.

| | precision | | recall | | f1-score | |
	cancer	normal	cancer	normal	cancer	normal
model1	0.89	0.84	0.83	0.9	0.86	0.87
model2	0.89	0.86	0.86	0.89	0.87	0.88
model3	0.89	0.86	0.85	0.9	0.87	0.88
model4	0.81	0.87	0.88	0.79	0.84	0.83
model5	0.84	0.86	0.87	0.83	0.85	0.85
model6	0.95	0.92	0.92	0.95	0.94	0.94
model7	0.8	0.88	0.9	0.78	0.85	0.83

FINAL PREDICTION

Based on the model performances, model6 (ResNet50) is the clear winner. I selected model6 to make the final prediction.

FUTURE DIRECTIONS:

In this project I trained the model using 20,000 images in each class. Based on the computing power the entire dataset with 220,025 could be utilized to train the models. Furthermore, due to time and computational limitation I was able to train only five custom and two pretrained models. Other pretrained models such as AlexNet and others can be used and compared with ResNet50 to see if they can perform better.

PROJECT LINK

https://github.com/leukemia/Capstone_Projects/tree/master/Capstone_Project02/Milestone_Report-02