Histopathologic Cancer Classification

Description

This is notebook to build CNN model for Histopathologic cancer classification. The intention is use this CNN modelmodel to classify image if it's a cancer or not. Our aim is get classification model with the most accuracy possible.

Data Overview

source : https://www.kaggle.com/competitions/histopathologic-cancer-detection/data

data type : TIFF images (220,025 images)

data size : 6.31 GB

Images are 96x96 RGB pixels

In [1]:
```
import pandas as pd
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

train_labels = pd.read_csv('histopathologic-cancer-detection/train_labels.cs
```

In [2]:
```
train_labels.head()
```

Out[2]:

	id	label
0	f38a6374c348f90b587e046aac6079959adf3835	0
1	c18f2d887b7ae4f6742ee445113fa1aef383ed77	1
2	755db6279dae599ebb4d39a9123cce439965282d	0
3	bc3f0c64fb968ff4a8bd33af6971ecae77c75e08	0
4	068aba587a4950175d04c680d38943fd488d6a9d	0

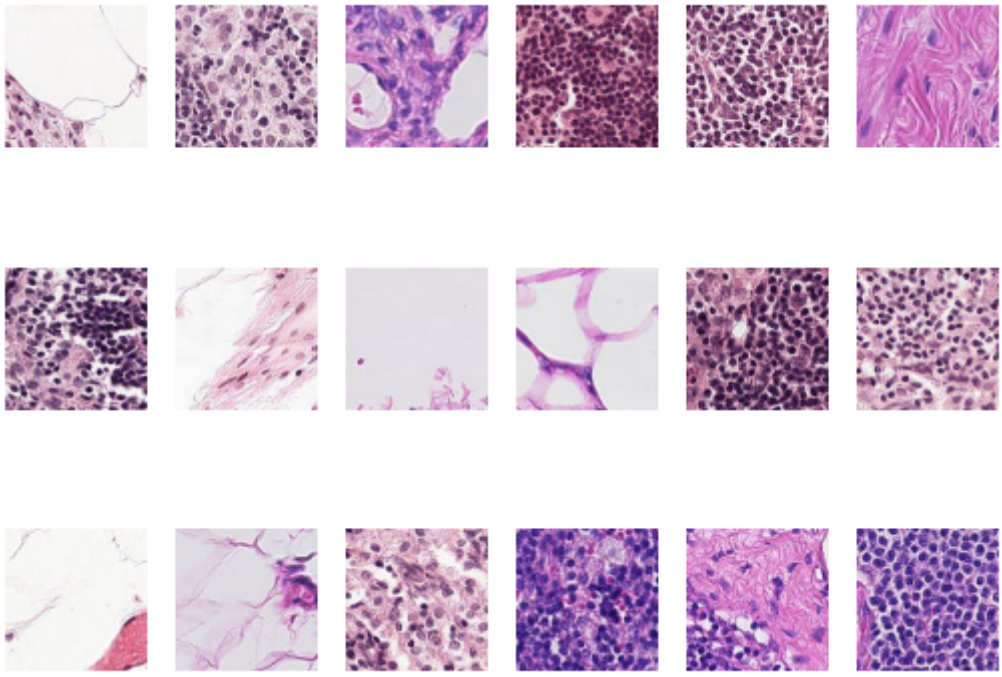
In [3]:
```
# check all possible labels and count of each
lb, lb_c = np.unique(train_labels['label'], return_counts=True)
print(lb, ' ', lb_c)
```

```
[0 1]   [130908  89117]
```

In [4]:
```
# show negative sample
neg = train_labels[train_labels.label == 0]

neg_image = []
for i in neg['id'].head(18):
    neg_image.append(Image.open('./histopathologic-cancer-detection/train/'+
```

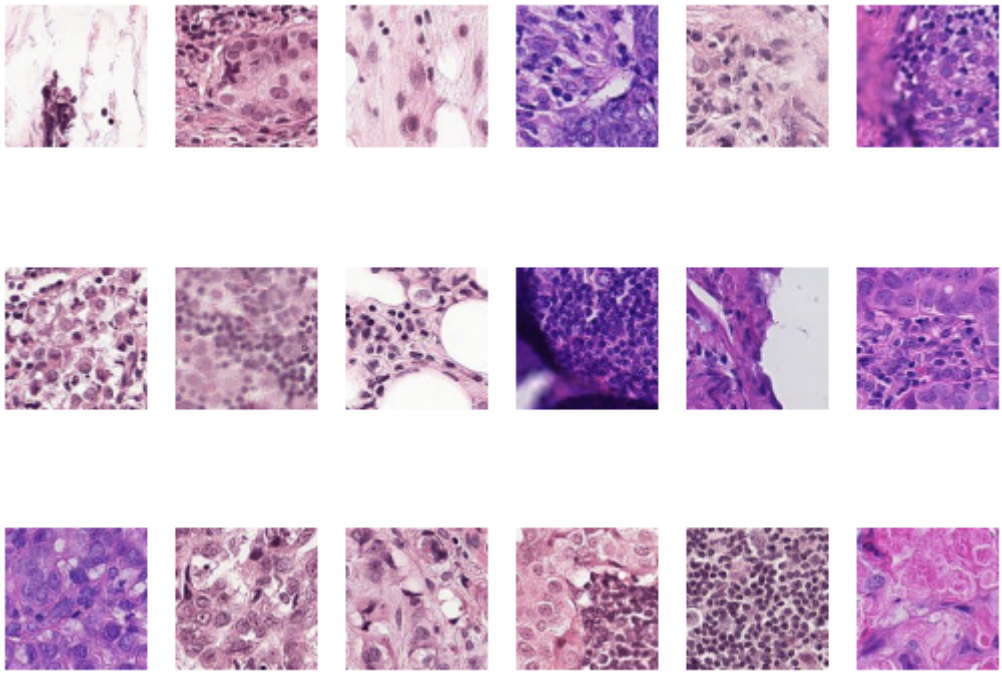
```
fix, axs = plt.subplots(3, 6)
axs = axs.flatten()
for img, ax in zip(neg_image, axs):
    ax.imshow(img)
    ax.axis('off')
plt.show()
```



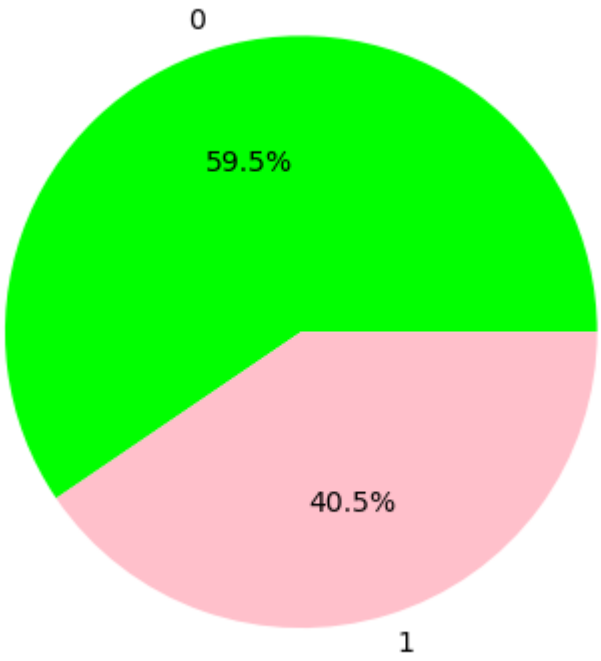
In [5]:
```
# show positive sample
pos = train_labels[train_labels.label == 1]

pos_image = []
for i in pos['id'].head(18):
    pos_image.append(Image.open('./histopathologic-cancer-detection/train/'+

fix, axs = plt.subplots(3, 6)
axs = axs.flatten()
for img, ax in zip(pos_image, axs):
    ax.imshow(img)
    ax.axis('off')
plt.show()
```



In [6]:
```
# show labels ratio of training data
plt.pie(lb_c, labels = lb, autopct='%1.1f%%', colors=['lime','pink'])
plt.show()
```



From pie chart above we can see that we have training data of positive label less than can negative label about 10%.

Analysis

We have huge training data and in this training data it have negative label more than positive label. We didn't get much information with sample images. Most color of image is purple and white. So, Maybe we can do gray scale to reduce training time and not effect much on accuracy.

Classification

First, I will try with simple CNN model as based line. Then, try again with more complex model. Also, I would use hyper parameter technique to improve accuracy.

In [7]:
```
# prepare df
train_labels['label'] = train_labels['label'].astype(str)
train_labels['id'] = train_labels['id'].add('.tif')
```

In [21]:
```
# gen training and validation data (validation 20%)
datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255, va

train_ds = datagen.flow_from_dataframe(
    train_labels,
    directory = './histopathologic-cancer-detection/train/',
    x_col = 'id',
    y_col = 'label',
    target_size = (96,96),
    class_mode='binary',
    seed = 1992,
    save_format = 'tif',
    subset = 'training'
)

val_ds = datagen.flow_from_dataframe(
    train_labels,
    directory = './histopathologic-cancer-detection/train/',
    x_col = 'id',
    y_col = 'label',
    target_size = (96,96),
    class_mode='binary',
    seed = 1992,
    save_format = 'tif',
    subset = 'validation'
)
```

```
Found 176020 validated image filenames belonging to 2 classes.
Found 44005 validated image filenames belonging to 2 classes.
```

In [15]:
```
# build base model
model0 = Sequential([
  layers.Conv2D(16, 3, activation='relu', input_shape=(96, 96, 3)),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(64, activation='sigmoid'),
  layers.Dense(1)
])
```

In [16]:
```
model0.compile(optimizer='adam',
               loss=tf.keras.losses.MeanSquaredError(),
               metrics=['accuracy'])
model0.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)          (None, 94, 94, 16)        448

 max_pooling2d_6 (MaxPoolin  (None, 47, 47, 16)        0
 g2D)

 conv2d_13 (Conv2D)          (None, 45, 45, 32)        4640

 max_pooling2d_7 (MaxPoolin  (None, 22, 22, 32)        0
 g2D)

 conv2d_14 (Conv2D)          (None, 20, 20, 64)        18496

 max_pooling2d_8 (MaxPoolin  (None, 10, 10, 64)        0
 g2D)

 flatten_2 (Flatten)         (None, 6400)              0

 dense_6 (Dense)             (None, 64)                409664

 dense_7 (Dense)             (None, 1)                 65

=================================================================
Total params: 433313 (1.65 MB)
Trainable params: 433313 (1.65 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

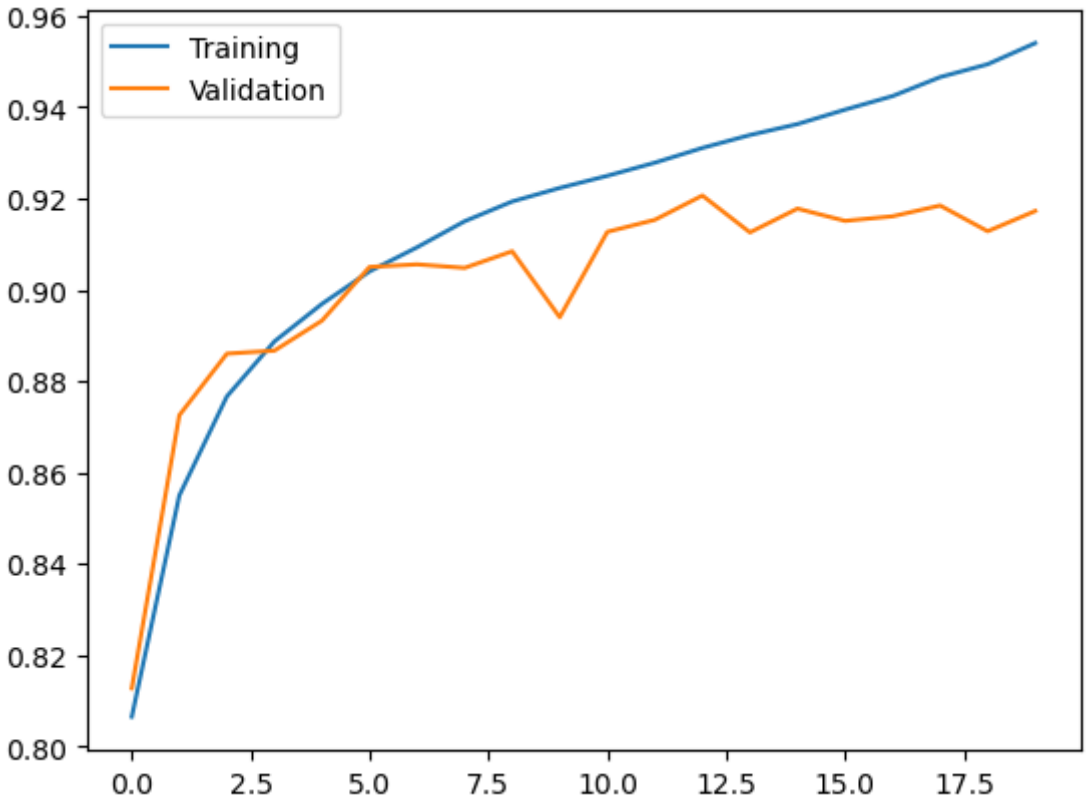
In [17]:
```
epochs=20
%time history = model0.fit(train_ds, validation_data=val_ds, epochs=epochs)
```

```
Epoch 1/20
   1/5501 [..............................] - ETA: 25:11 - loss: 0.2885 - acc
uracy: 0.7500
2023-08-09 17:41:42.884851: I tensorflow/core/grappler/optimizers/custom_gra
ph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enable
d.
5500/5501 [============================>.] - ETA: 0s - loss: 0.1388 - accura
cy: 0.8065
2023-08-09 17:43:22.305372: I tensorflow/core/grappler/optimizers/custom_gra
ph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enable
d.
```

```
5501/5501 [==============================] - 116s 21ms/step - loss: 0.1388 -
accuracy: 0.8065 - val_loss: 0.1323 - val_accuracy: 0.8128
Epoch 2/20
5501/5501 [==============================] - 116s 21ms/step - loss: 0.1070 -
accuracy: 0.8550 - val_loss: 0.0945 - val_accuracy: 0.8725
Epoch 3/20
5501/5501 [==============================] - 115s 21ms/step - loss: 0.0924 -
accuracy: 0.8767 - val_loss: 0.0863 - val_accuracy: 0.8860
Epoch 4/20
5501/5501 [==============================] - 124s 22ms/step - loss: 0.0844 -
accuracy: 0.8887 - val_loss: 0.0858 - val_accuracy: 0.8867
Epoch 5/20
5501/5501 [==============================] - 125s 23ms/step - loss: 0.0792 -
accuracy: 0.8969 - val_loss: 0.0818 - val_accuracy: 0.8932
Epoch 6/20
5501/5501 [==============================] - 117s 21ms/step - loss: 0.0744 -
accuracy: 0.9040 - val_loss: 0.0737 - val_accuracy: 0.9050
Epoch 7/20
5501/5501 [==============================] - 117s 21ms/step - loss: 0.0696 -
accuracy: 0.9093 - val_loss: 0.0719 - val_accuracy: 0.9056
Epoch 8/20
5501/5501 [==============================] - 118s 21ms/step - loss: 0.0651 -
accuracy: 0.9150 - val_loss: 0.0715 - val_accuracy: 0.9048
Epoch 9/20
5501/5501 [==============================] - 133s 24ms/step - loss: 0.0620 -
accuracy: 0.9193 - val_loss: 0.0692 - val_accuracy: 0.9084
Epoch 10/20
5501/5501 [==============================] - 119s 22ms/step - loss: 0.0595 -
accuracy: 0.9223 - val_loss: 0.0801 - val_accuracy: 0.8940
Epoch 11/20
5501/5501 [==============================] - 118s 21ms/step - loss: 0.0577 -
accuracy: 0.9249 - val_loss: 0.0658 - val_accuracy: 0.9127
Epoch 12/20
5501/5501 [==============================] - 117s 21ms/step - loss: 0.0555 -
accuracy: 0.9278 - val_loss: 0.0644 - val_accuracy: 0.9153
Epoch 13/20
5501/5501 [==============================] - 119s 22ms/step - loss: 0.0531 -
accuracy: 0.9310 - val_loss: 0.0614 - val_accuracy: 0.9206
Epoch 14/20
5501/5501 [==============================] - 120s 22ms/step - loss: 0.0512 -
accuracy: 0.9339 - val_loss: 0.0674 - val_accuracy: 0.9126
Epoch 15/20
5501/5501 [==============================] - 120s 22ms/step - loss: 0.0494 -
accuracy: 0.9363 - val_loss: 0.0633 - val_accuracy: 0.9178
Epoch 16/20
5501/5501 [==============================] - 121s 22ms/step - loss: 0.0471 -
accuracy: 0.9394 - val_loss: 0.0661 - val_accuracy: 0.9151
Epoch 17/20
5501/5501 [==============================] - 119s 22ms/step - loss: 0.0452 -
accuracy: 0.9424 - val_loss: 0.0650 - val_accuracy: 0.9161
Epoch 18/20
5501/5501 [==============================] - 119s 22ms/step - loss: 0.0425 -
accuracy: 0.9466 - val_loss: 0.0651 - val_accuracy: 0.9184
Epoch 19/20
5501/5501 [==============================] - 122s 22ms/step - loss: 0.0405 -
accuracy: 0.9494 - val_loss: 0.0691 - val_accuracy: 0.9128
Epoch 20/20
5501/5501 [==============================] - 124s 22ms/step - loss: 0.0380 -
accuracy: 0.9540 - val_loss: 0.0658 - val_accuracy: 0.9172
CPU times: user 27min 58s, sys: 9min 27s, total: 37min 25s
Wall time: 39min 58s
```

In [18]:
```
epochs_range = range(epochs)
```

```
plt.plot(epochs_range, history.history['accuracy'], label='Training')
plt.plot(epochs_range, history.history['val_accuracy'], label='Validation')
plt.legend()
plt.show()
```



In [45]:
```
# prepare df
test_labels = pd.read_csv('histopathologic-cancer-detection/sample_submissio
test_labels['filename'] = test_labels['id'].add('.tif')
```

In [46]:
```
test_ds = datagen.flow_from_dataframe(
    test_labels,
    directory = './histopathologic-cancer-detection/test/',
    x_col = 'filename',
    y_col = None,
    target_size = (96,96),
    class_mode=None,
    shuffle=False,
    seed = 1992,
    save_format = 'tif'
)
```

```
Found 57458 validated image filenames.
```

In [55]:
```
predictions = model0.predict(test_ds)

output = pd.DataFrame({'id': test_labels.id, 'label': predictions.flatten()})
output.to_csv('submission0.csv', index=False)
print("Your submission was successfully saved!")
```

```
1796/1796 [==============================] - 21s 12ms/step
Your submission was successfully saved!
```

Score for submit this prediction to Kaggle private = 0.8777 public=0.9186

Next, I will try with more complex model which shoud have higher accuracy.

In [71]:
```
model1 = Sequential([
  layers.Conv2D(16, 3, activation='relu', input_shape=(96, 96, 3)),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(128, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(32, activation='sigmoid'),
  layers.Dense(1)
])
```

In [72]:
```
model1.compile(optimizer='adam',
               loss=tf.keras.losses.MeanSquaredError(),
               metrics=['accuracy'])
model1.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_34 (Conv2D)          (None, 94, 94, 16)        448

 max_pooling2d_21 (MaxPooli  (None, 47, 47, 16)        0
 ng2D)

 conv2d_35 (Conv2D)          (None, 45, 45, 32)        4640

 max_pooling2d_22 (MaxPooli  (None, 22, 22, 32)        0
 ng2D)

 conv2d_36 (Conv2D)          (None, 20, 20, 64)        18496

 max_pooling2d_23 (MaxPooli  (None, 10, 10, 64)        0
 ng2D)

 conv2d_37 (Conv2D)          (None, 8, 8, 128)         73856

 max_pooling2d_24 (MaxPooli  (None, 4, 4, 128)         0
 ng2D)

 flatten_7 (Flatten)         (None, 2048)              0

 dense_21 (Dense)            (None, 128)               262272

 dense_22 (Dense)            (None, 32)                4128

 dense_23 (Dense)            (None, 1)                 33

=================================================================
Total params: 363873 (1.39 MB)
Trainable params: 363873 (1.39 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [73]:
```
epochs=10
%time history = model1.fit(train_ds, validation_data=val_ds, epochs=epochs)
```

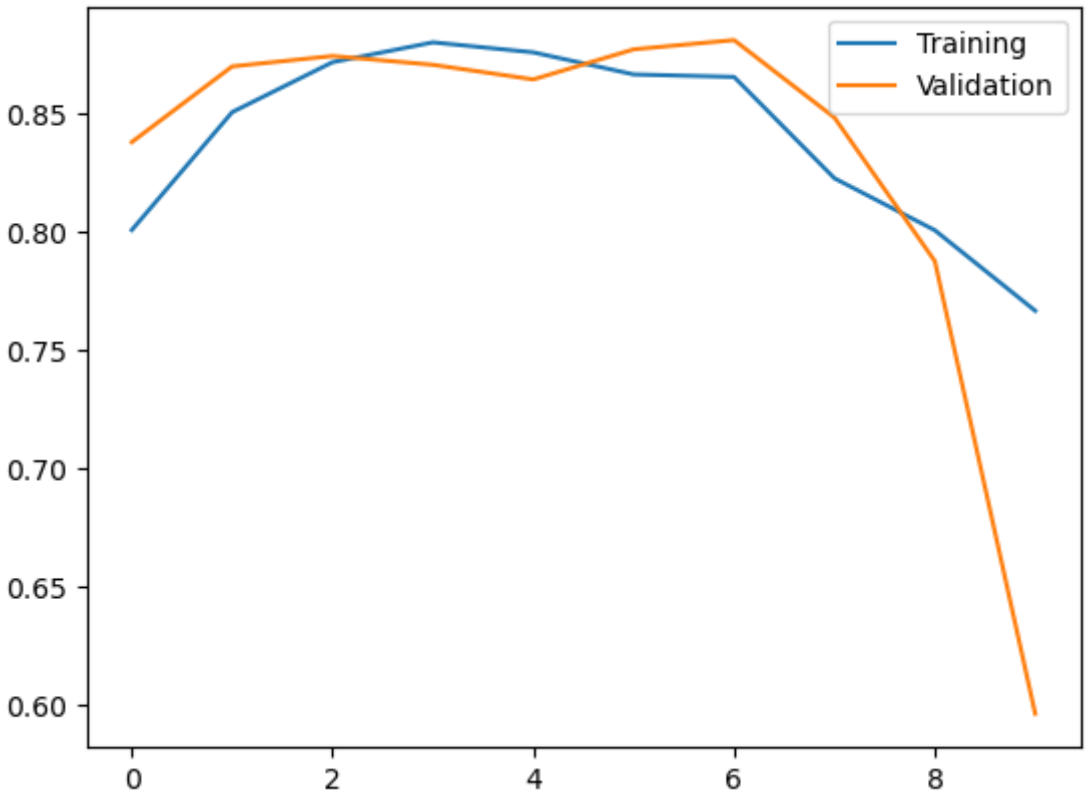
```
Epoch 1/10
   1/5501 [..............................] - ETA: 52:40 - loss: 0.7247 - acc
uracy: 0.7188
```

```
2023-08-09 19:17:17.142858: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
5500/5501 [============================>.] - ETA: 0s - loss: 0.1416 - accuracy: 0.8006
2023-08-09 19:18:58.277081: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
5501/5501 [==============================] - 118s 21ms/step - loss: 0.1416 - accuracy: 0.8006 - val_loss: 0.1185 - val_accuracy: 0.8378
Epoch 2/10
5501/5501 [==============================] - 120s 22ms/step - loss: 0.1091 - accuracy: 0.8505 - val_loss: 0.0968 - val_accuracy: 0.8698
Epoch 3/10
5501/5501 [==============================] - 119s 22ms/step - loss: 0.0944 - accuracy: 0.8716 - val_loss: 0.0919 - val_accuracy: 0.8742
Epoch 4/10
5501/5501 [==============================] - 120s 22ms/step - loss: 0.0889 - accuracy: 0.8800 - val_loss: 0.0964 - val_accuracy: 0.8705
Epoch 5/10
5501/5501 [==============================] - 119s 22ms/step - loss: 0.0924 - accuracy: 0.8758 - val_loss: 0.0979 - val_accuracy: 0.8643
Epoch 6/10
5501/5501 [==============================] - 118s 22ms/step - loss: 0.0986 - accuracy: 0.8664 - val_loss: 0.0930 - val_accuracy: 0.8770
Epoch 7/10
5501/5501 [==============================] - 123s 22ms/step - loss: 0.1005 - accuracy: 0.8654 - val_loss: 0.0946 - val_accuracy: 0.8809
Epoch 8/10
5501/5501 [==============================] - 124s 23ms/step - loss: 0.1283 - accuracy: 0.8225 - val_loss: 0.1112 - val_accuracy: 0.8482
Epoch 9/10
5501/5501 [==============================] - 122s 22ms/step - loss: 0.1401 - accuracy: 0.8005 - val_loss: 0.1555 - val_accuracy: 0.7875
Epoch 10/10
5501/5501 [==============================] - 120s 22ms/step - loss: 0.1625 - accuracy: 0.7665 - val_loss: 0.2421 - val_accuracy: 0.5961
CPU times: user 13min 26s, sys: 4min 22s, total: 17min 48s
Wall time: 20min 3s
```

In [74]:

```
epochs_range = range(epochs)

plt.plot(epochs_range, history.history['accuracy'], label='Training')
plt.plot(epochs_range, history.history['val_accuracy'], label='Validation')
plt.legend()
plt.show()
```



From graph above, I think accuracy due to learing rate was too high. So, I will try again with lower learing rate and change a bit of a model.

In [128…
```
model2 = Sequential([
  layers.Conv2D(16, 3, activation='relu', input_shape=(96, 96, 3)),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(128, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(32, activation='relu'),
  layers.Dense(1, activation='sigmoid')
])
```

In [129…
```
model2.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
               loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
               metrics=['accuracy'])
model2.summary()
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` r
uns slowly on M1/M2 Macs, please use the legacy Keras optimizer instead, loc
ated at `tf.keras.optimizers.legacy.Adam`.
WARNING:absl:There is a known slowdown when using v2.11+ Keras optimizers on
M1/M2 Macs. Falling back to the legacy Keras optimizer, i.e., `tf.keras.opti
mizers.legacy.Adam`.
```

```
Model: "sequential_25"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_103 (Conv2D)         (None, 94, 94, 16)        448

 max_pooling2d_86 (MaxPooli  (None, 47, 47, 16)        0
 ng2D)

 conv2d_104 (Conv2D)         (None, 45, 45, 32)        4640

 max_pooling2d_87 (MaxPooli  (None, 22, 22, 32)        0
 ng2D)

 conv2d_105 (Conv2D)         (None, 20, 20, 64)        18496

 max_pooling2d_88 (MaxPooli  (None, 10, 10, 64)        0
 ng2D)

 conv2d_106 (Conv2D)         (None, 8, 8, 128)         73856

 max_pooling2d_89 (MaxPooli  (None, 4, 4, 128)         0
 ng2D)

 flatten_25 (Flatten)        (None, 2048)              0

 dense_64 (Dense)            (None, 128)               262272

 dense_65 (Dense)            (None, 32)                4128

 dense_66 (Dense)            (None, 1)                 33

=================================================================
Total params: 363873 (1.39 MB)
Trainable params: 363873 (1.39 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [130…

```
epochs=15
%time history = model2.fit(train_ds, validation_data=val_ds, epochs=epochs)
```

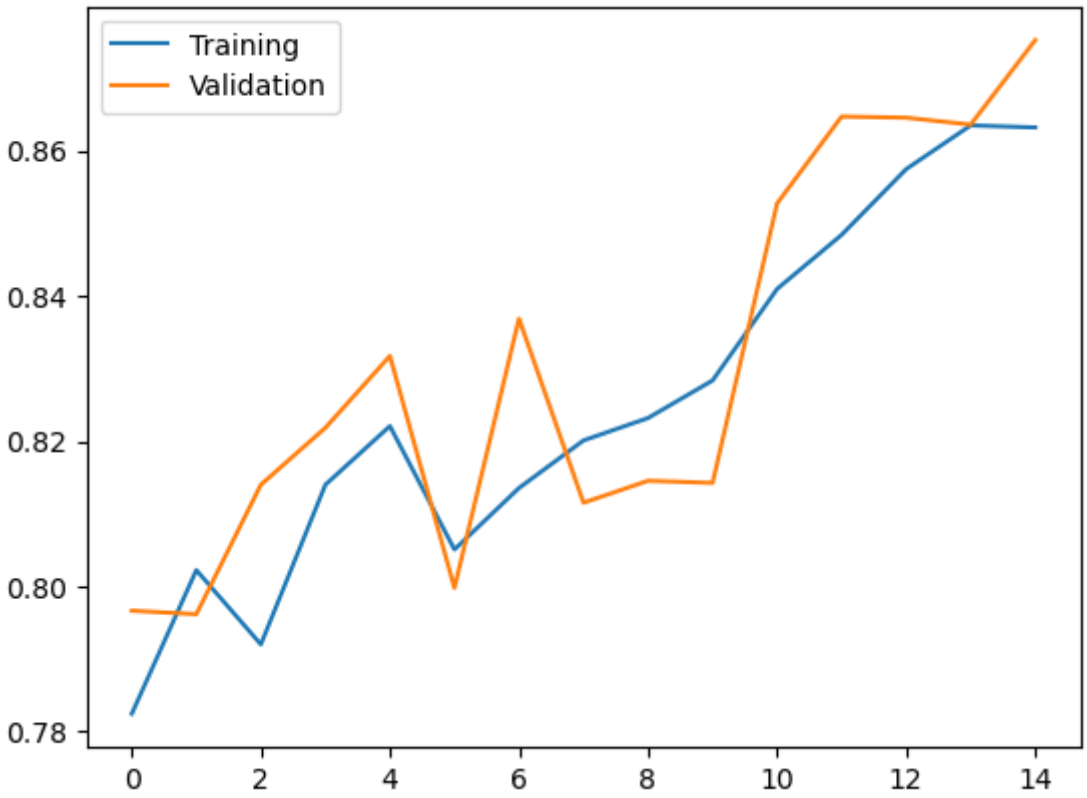
```
Epoch 1/15
2023-08-09 20:14:19.103442: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
5499/5501 [============================>.] - ETA: 0s - loss: 0.4693 - accuracy: 0.7825
2023-08-09 20:16:04.372857: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU is enabled.
```

```
5501/5501 [==============================] - 122s 22ms/step - loss: 0.4693 -
accuracy: 0.7825 - val_loss: 0.4541 - val_accuracy: 0.7967
Epoch 2/15
5501/5501 [==============================] - 120s 22ms/step - loss: 0.4386 -
accuracy: 0.8022 - val_loss: 0.4445 - val_accuracy: 0.7962
Epoch 3/15
5501/5501 [==============================] - 116s 21ms/step - loss: 0.4581 -
accuracy: 0.7920 - val_loss: 0.4196 - val_accuracy: 0.8140
Epoch 4/15
5501/5501 [==============================] - 117s 21ms/step - loss: 0.4206 -
accuracy: 0.8140 - val_loss: 0.4216 - val_accuracy: 0.8218
Epoch 5/15
5501/5501 [==============================] - 128s 23ms/step - loss: 0.4083 -
accuracy: 0.8221 - val_loss: 0.3861 - val_accuracy: 0.8317
Epoch 6/15
5501/5501 [==============================] - 177s 32ms/step - loss: 0.4507 -
accuracy: 0.8051 - val_loss: 0.4396 - val_accuracy: 0.7998
Epoch 7/15
5501/5501 [==============================] - 162s 29ms/step - loss: 0.4301 -
accuracy: 0.8135 - val_loss: 0.3850 - val_accuracy: 0.8369
Epoch 8/15
5501/5501 [==============================] - 167s 30ms/step - loss: 0.4193 -
accuracy: 0.8201 - val_loss: 0.4572 - val_accuracy: 0.8115
Epoch 9/15
5501/5501 [==============================] - 131s 24ms/step - loss: 0.4167 -
accuracy: 0.8232 - val_loss: 0.4417 - val_accuracy: 0.8145
Epoch 10/15
5501/5501 [==============================] - 127s 23ms/step - loss: 0.4092 -
accuracy: 0.8283 - val_loss: 0.4650 - val_accuracy: 0.8142
Epoch 11/15
5501/5501 [==============================] - 123s 22ms/step - loss: 0.3756 -
accuracy: 0.8409 - val_loss: 0.3541 - val_accuracy: 0.8527
Epoch 12/15
5501/5501 [==============================] - 120s 22ms/step - loss: 0.3556 -
accuracy: 0.8484 - val_loss: 0.3215 - val_accuracy: 0.8647
Epoch 13/15
5501/5501 [==============================] - 122s 22ms/step - loss: 0.3357 -
accuracy: 0.8574 - val_loss: 0.3198 - val_accuracy: 0.8645
Epoch 14/15
5501/5501 [==============================] - 125s 23ms/step - loss: 0.3228 -
accuracy: 0.8635 - val_loss: 0.3154 - val_accuracy: 0.8636
Epoch 15/15
5501/5501 [==============================] - 122s 22ms/step - loss: 0.3240 -
accuracy: 0.8632 - val_loss: 0.2974 - val_accuracy: 0.8752
CPU times: user 26min 27s, sys: 7min 46s, total: 34min 14s
Wall time: 32min 58s
```

In [131…

```
epochs_range = range(epochs)

plt.plot(epochs_range, history.history['accuracy'], label='Training')
plt.plot(epochs_range, history.history['val_accuracy'], label='Validation')
plt.legend()
plt.show()
```



This model seem to need more time to reach optimal. I want to train this more del with more epochs but I don't have time for that. Therefore, I will submit this as my last submittion.

Score for submit this prediction to Kaggle private = 0.8937 public=0.9034

Conclusion

First we look up trainig data and try some analysis. Then we build CNN model and tune it to get the best model (on accuracy). I think learing rate and epochs are very import hyper-parameter in this project. We should select proper earing rate and epochs for the best result (in term of accuracy and time to train the model). We also, can see that CNN model took lot of time to train, even simple CCN model. Therefore, we should ensure everything before training CNN model. For kaggle score I think this is good enough for me since it's my first CNN project.

github : https://github.com/Satjarporn/cancer