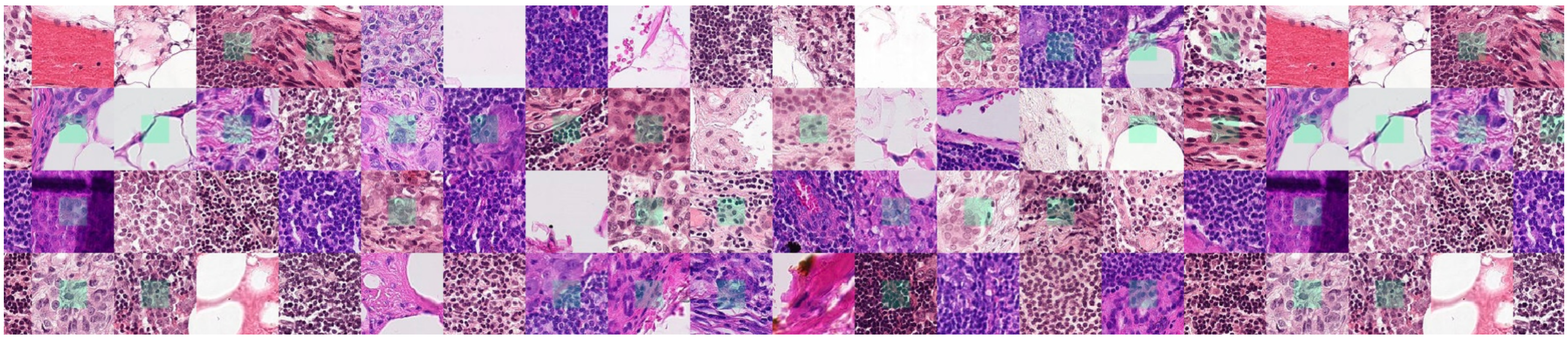
HISTOPATHOLOGIC CANCER DETECTION



In [1]:
```
# Importing Packages

# Setting Random seed
from numpy.random import seed
seed(101)
from tensorflow import set_random_seed
set_random_seed(101)

import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import Dense, Dropout, Flatten, Activation
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
from tensorflow.keras.optimizers import Adam
import os
import cv2
from sklearn.utils import shuffle
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import itertools
import shutil
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
# Setting Parameters before loading data
IMAGE_SIZE = 96
IMAGE_CHANNELS = 3
SAMPLE_SIZE = 80000 # the number of images we use from each of the two classes
```

Labels as per csv file

0 = no tumor tissue

1 = has tumor tissue.

Checking for Count of Train & Test Images

In [4]:
```
print(len(os.listdir('../input/train')))
print(len(os.listdir('../input/test')))
```

```
220025
57458
```

Create a Dataframe containing all images

In [5]:
```
df_data = pd.read_csv('../input/train_labels.csv')

# removing this image because it caused a training error previously
df_data[df_data['id'] != 'dd6dfed324f9fcb6f93f46f32fc800f2ec196be2']

# removing this image because it's black
df_data[df_data['id'] != '9369c7278ec8bcc6c880d99194de09fc2bd4efbe']


print(df_data.shape)
```

```
(220025, 2)
```

Check the class distribution

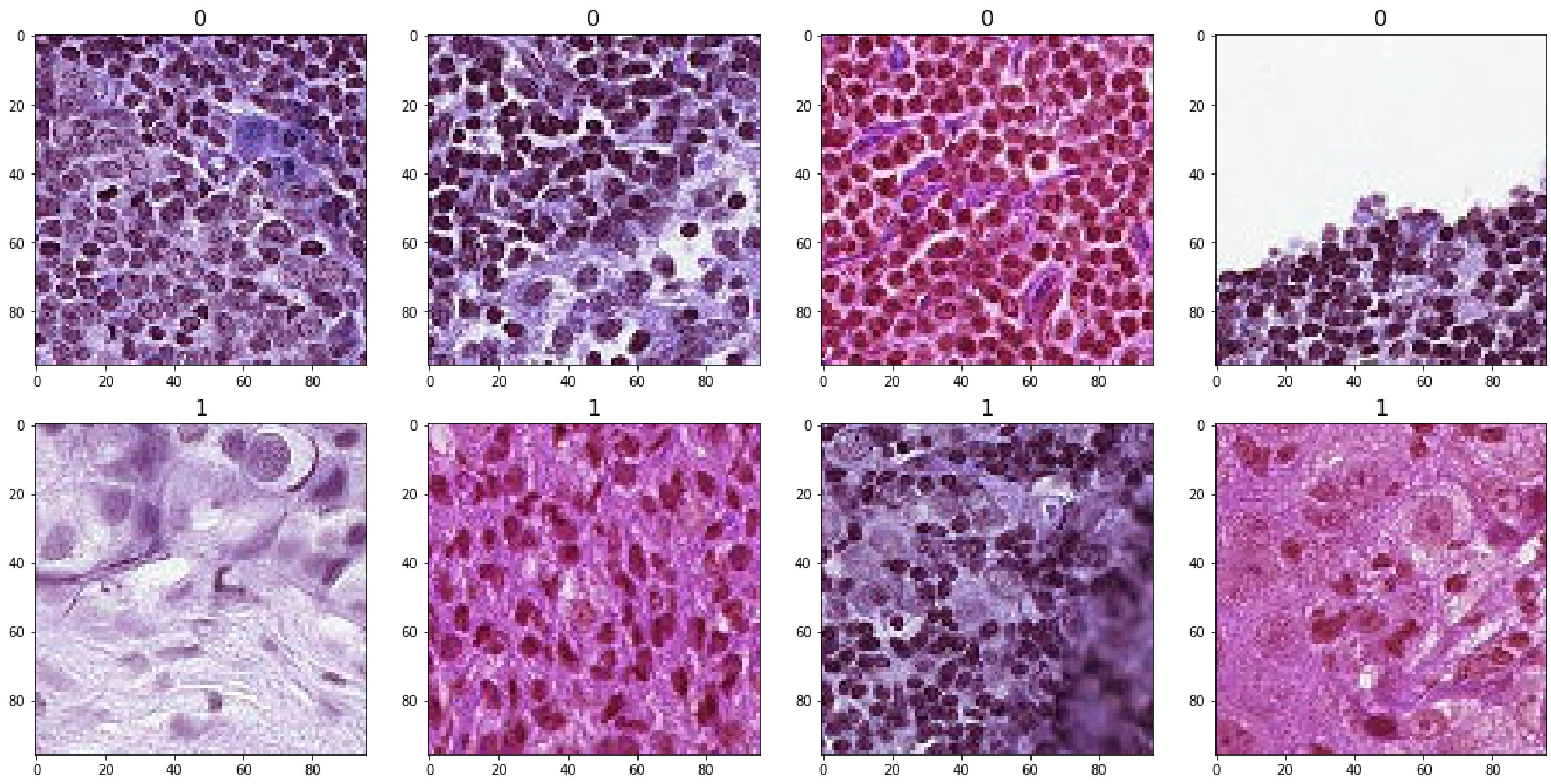
In [6]:
```
df_data['label'].value_counts()
```

Out[6]:
```
0    130908
1     89117
Name: label, dtype: int64
```

Display a random sample of train images by class

In [7]:
```
# Function for Showing images with both classes randomly
def draw_category_images(col_name,figure_cols, df, IMAGE_PATH):

    """
    Give a column in a dataframe,
    this function takes a sample of each class and displays that
    sample on one row. The sample size is the same as figure_cols which
    is the number of columns in the figure.
    Because this function takes a random sample, each time the function is run it
    displays different images.
    """


    categories = (df.groupby([col_name])[col_name].nunique()).index
    f, ax = plt.subplots(nrows=len(categories),ncols=figure_cols,
                         figsize=(4*figure_cols,4*len(categories))) # adjust size here
    # draw a number of images for each location
    for i, cat in enumerate(categories):
        sample = df[df[col_name]==cat].sample(figure_cols) # figure_cols is also the sample size
        for j in range(0,figure_cols):
            file=IMAGE_PATH + sample.iloc[j]['id'] + '.tif'
            im=cv2.imread(file)
            ax[i, j].imshow(im, resample=True, cmap='gray')
            ax[i, j].set_title(cat, fontsize=16)
    plt.tight_layout()
    plt.show()
```

In [8]:
```
IMAGE_PATH = '../input/train/'
# Displaying 4 images in each class
draw_category_images('label',4, df_data, IMAGE_PATH)
```



Balance the target distribution

We will reduce the number of samples in class 0.

In [10]:
```
# take a random sample of class 0 with size equal to num samples in class 1
df_0 = df_data[df_data['label'] == 0].sample(SAMPLE_SIZE, random_state = 101)
# filter out class 1
df_1 = df_data[df_data['label'] == 1].sample(SAMPLE_SIZE, random_state = 101)

# concat the dataframes
df_data = pd.concat([df_0, df_1], axis=0).reset_index(drop=True)
# shuffle
df_data = shuffle(df_data)

df_data['label'].value_counts()
```

Out[10]:
```
1    80000
0    80000
Name: label, dtype: int64
```

In [11]:

```
# DF Overview
df_data.head()
```

Out[11]:

	id	label
107459	1d35e8084b7697421209c5e0463e4195f169c811	1
88068	40ae99e1ad1ad69e7e2af32372fea053b5b14c2d	1
37478	41644dee74e322fbe45eee488a617859ae520fec	0
5470	6e8a7e4c80e9ee48ec939e19af9deeee4f110ff7	0
138898	2485de6e05a78ed947d956f5fdd045bc29d924d1	1

In [12]:

```
# train_test_split

# stratify=y creates a balanced validation set.
y = df_data['label']

df_train, df_val = train_test_split(df_data, test_size=0.10, random_state=101, stratify=y)

print(df_train.shape)
print(df_val.shape)
```

```
(144000, 2)
(16000, 2)
```

In [13]:

```
# Both classes have equal weightage in train subset
df_train['label'].value_counts()
```

Out[13]:

```
1    72000
0    72000
Name: label, dtype: int64
```

In [14]:

```
# Both classes have equal weightage in Val subset
df_val['label'].value_counts()
```

Out[14]:

```
1    8000
0    8000
Name: label, dtype: int64
```

Create a Directory Structure

In [15]:

```
# Create a new directory
base_dir = 'base_dir'
os.mkdir(base_dir)

# create a path to 'base_dir' to which we will join the names of the new folders
# train_dir
train_dir = os.path.join(base_dir, 'train_dir')
os.mkdir(train_dir)

# val_dir
val_dir = os.path.join(base_dir, 'val_dir')
os.mkdir(val_dir)

# Inside each folder we create seperate folders for each class
# create new folders inside train_dir
no_tumor_tissue = os.path.join(train_dir, 'a_no_tumor_tissue')
os.mkdir(no_tumor_tissue)
has_tumor_tissue = os.path.join(train_dir, 'b_has_tumor_tissue')
os.mkdir(has_tumor_tissue)


# create new folders inside val_dir
no_tumor_tissue = os.path.join(val_dir, 'a_no_tumor_tissue')
os.mkdir(no_tumor_tissue)
has_tumor_tissue = os.path.join(val_dir, 'b_has_tumor_tissue')
os.mkdir(has_tumor_tissue)
```

In [16]:

```
# check that the folders have been created
os.listdir('base_dir/train_dir')
```

Out[16]:

```
['a_no_tumor_tissue', 'b_has_tumor_tissue']
```

Transfer the images into the folders

In [17]:

```
# Set the id as the index in df_data
df_data.set_index('id', inplace=True)
```

In [18]:

```
# Get a list of train and val images
train_list = list(df_train['id'])
val_list = list(df_val['id'])
```

```
# Transfer the train images
for image in train_list:

    # the id in the csv file does not have the .tif extension therefore we add it here
    fname = image + '.tif'
    # get the label for a certain image
    target = df_data.loc[image,'label']

    # these must match the folder names
    if target == 0:
        label = 'a_no_tumor_tissue'
    if target == 1:
        label = 'b_has_tumor_tissue'

    # source path to image
    src = os.path.join('../input/train', fname)
    # destination path to image
    dst = os.path.join(train_dir, label, fname)
    # copy the image from the source to the destination
    shutil.copyfile(src, dst)


# Transfer the val images

for image in val_list:

    # the id in the csv file does not have the .tif extension therefore we add it here
    fname = image + '.tif'
    # get the label for a certain image
    target = df_data.loc[image,'label']

    # these must match the folder names
    if target == 0:
        label = 'a_no_tumor_tissue'
    if target == 1:
        label = 'b_has_tumor_tissue'


    # source path to image
    src = os.path.join('../input/train', fname)
    # destination path to image
    dst = os.path.join(val_dir, label, fname)
    # copy the image from the source to the destination
    shutil.copyfile(src, dst)
```

In [19]:

```
# checking for how many train images we have in each folder

print(len(os.listdir('base_dir/train_dir/a_no_tumor_tissue')))
print(len(os.listdir('base_dir/train_dir/b_has_tumor_tissue')))
```

```
72000
72000
```

In [20]:

```
# checking how many val images we have in each folder

print(len(os.listdir('base_dir/val_dir/a_no_tumor_tissue')))
print(len(os.listdir('base_dir/val_dir/b_has_tumor_tissue')))
```

```
8000
8000
```

Set Up the Generators

In [22]:

```
# Setting up new created directories as path
train_path = 'base_dir/train_dir'
valid_path = 'base_dir/val_dir'
test_path = '../input/test'

num_train_samples = len(df_train)
num_val_samples = len(df_val)
train_batch_size = 10
val_batch_size = 10

# Defining steps with batch size and samples
train_steps = np.ceil(num_train_samples / train_batch_size)
val_steps = np.ceil(num_val_samples / val_batch_size)
```

In [23]:

```
# Using ImageDataGenerator for loading images
datagen = ImageDataGenerator(rescale=1.0/255)

train_gen = datagen.flow_from_directory(train_path,
                                        target_size=(IMAGE_SIZE,IMAGE_SIZE),
                                        batch_size=train_batch_size,
```

```
                                        class_mode='categorical')

val_gen = datagen.flow_from_directory(valid_path,
                                        target_size=(IMAGE_SIZE,IMAGE_SIZE),
                                        batch_size=val_batch_size,
                                        class_mode='categorical')

# Note: shuffle=False causes the test dataset to not be shuffled
test_gen = datagen.flow_from_directory(valid_path,
                                        target_size=(IMAGE_SIZE,IMAGE_SIZE),
                                        batch_size=1,
                                        class_mode='categorical',
                                        shuffle=False)
```

```
Found 144000 images belonging to 2 classes.
Found 16000 images belonging to 2 classes.
Found 16000 images belonging to 2 classes.
```

Model Architecture¶

In [24]:

```
# Model Params
kernel_size = (3,3)
pool_size= (2,2)
first_filters = 32
second_filters = 64
third_filters = 128
dropout_conv = 0.3
dropout_dense = 0.3

# Model Structure
model = Sequential()
model.add(Conv2D(first_filters, kernel_size, activation = 'relu', input_shape = (96, 96, 3)))
model.add(Conv2D(first_filters, kernel_size, activation = 'relu'))
model.add(Conv2D(first_filters, kernel_size, activation = 'relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Conv2D(second_filters, kernel_size, activation ='relu'))
model.add(Conv2D(second_filters, kernel_size, activation ='relu'))
model.add(Conv2D(second_filters, kernel_size, activation ='relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(Conv2D(third_filters, kernel_size, activation ='relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Dropout(dropout_conv))

model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(dropout_dense))
model.add(Dense(2, activation = "softmax"))

model.summary()
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 94, 94, 32)	896
conv2d_1 (Conv2D)	(None, 92, 92, 32)	9248
conv2d_2 (Conv2D)	(None, 90, 90, 32)	9248
max_pooling2d (MaxPooling2D)	(None, 45, 45, 32)	0
dropout (Dropout)	(None, 45, 45, 32)	0
conv2d_3 (Conv2D)	(None, 43, 43, 64)	18496
conv2d_4 (Conv2D)	(None, 41, 41, 64)	36928
conv2d_5 (Conv2D)	(None, 39, 39, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 19, 19, 64)	0
dropout_1 (Dropout)	(None, 19, 19, 64)	0
conv2d_6 (Conv2D)	(None, 17, 17, 128)	73856
conv2d_7 (Conv2D)	(None, 15, 15, 128)	147584
conv2d_8 (Conv2D)	(None, 13, 13, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 128)	0
dropout_2 (Dropout)	(None, 6, 6, 128)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
dropout_3 (Dropout)	(None, 256)	0

```
dense_1 (Dense)              (None, 2)                 514
=================================================================
Total params: 1,661,186
Trainable params: 1,661,186
Non-trainable params: 0
_________________________________________________________________
```

Train the Model

In [25]:

```
# Compliling the model
model.compile(Adam(lr=0.0001), loss='binary_crossentropy',
              metrics=['accuracy'])
```

In [26]:

```
# Get the labels that are associated with each index
print(val_gen.class_indices)
```

```
{'a_no_tumor_tissue': 0, 'b_has_tumor_tissue': 1}
```

In [28]:

```
filepath = "model.h5"

# Checkpoint to save weights after every epoch
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1,
                             save_best_only=True, mode='max')

# It alters the learning rate based on metrics in each epoch
reduce_lr = ReduceLROnPlateau(monitor='val_acc', factor=0.5, patience=2,
                                   verbose=1, mode='max', min_lr=0.00001)

callbacks_list = [checkpoint, reduce_lr]

# Fitting the model
history = model.fit_generator(train_gen, steps_per_epoch=train_steps,
                    validation_data=val_gen,
                    validation_steps=val_steps,
                    epochs=20, verbose=1,
                   callbacks=callbacks_list)
```

```
Epoch 1/20
14399/14400 [============================>.] - ETA: 0s - loss: 0.2276 - acc: 0.9087
Epoch 00001: val_acc improved from -inf to 0.90569, saving model to model.h5
14400/14400 [==============================] - 264s 18ms/step - loss: 0.2276 - acc: 0.9087 - val_loss: 0.2313 - val_acc: 0.9057
Epoch 2/20
14396/14400 [============================>.] - ETA: 0s - loss: 0.2191 - acc: 0.9127
Epoch 00002: val_acc did not improve from 0.90569
14400/14400 [==============================] - 260s 18ms/step - loss: 0.2190 - acc: 0.9127 - val_loss: 0.2882 - val_acc: 0.8843
Epoch 3/20
14399/14400 [============================>.] - ETA: 0s - loss: 0.2107 - acc: 0.9171
Epoch 00003: val_acc improved from 0.90569 to 0.92681, saving model to model.h5
14400/14400 [==============================] - 261s 18ms/step - loss: 0.2107 - acc: 0.9171 - val_loss: 0.1903 - val_acc: 0.9268
Epoch 4/20
14395/14400 [============================>.] - ETA: 0s - loss: 0.2033 - acc: 0.9201
Epoch 00004: val_acc did not improve from 0.92681
14400/14400 [==============================] - 267s 19ms/step - loss: 0.2032 - acc: 0.9202 - val_loss: 0.1972 - val_acc: 0.9248
Epoch 5/20
14396/14400 [============================>.] - ETA: 0s - loss: 0.1960 - acc: 0.9233
Epoch 00005: val_acc improved from 0.92681 to 0.93037, saving model to model.h5
14400/14400 [==============================] - 261s 18ms/step - loss: 0.1960 - acc: 0.9233 - val_loss: 0.1850 - val_acc: 0.9304
Epoch 6/20
14396/14400 [============================>.] - ETA: 0s - loss: 0.1911 - acc: 0.9252
Epoch 00006: val_acc did not improve from 0.93037
14400/14400 [==============================] - 261s 18ms/step - loss: 0.1911 - acc: 0.9252 - val_loss: 0.1943 - val_acc: 0.9250
Epoch 7/20
14397/14400 [============================>.] - ETA: 0s - loss: 0.1858 - acc: 0.9275
Epoch 00007: val_acc did not improve from 0.93037

Epoch 00007: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.
14400/14400 [==============================] - 262s 18ms/step - loss: 0.1858 - acc: 0.9275 - val_loss: 0.1862 - val_acc: 0.9300
Epoch 8/20
14397/14400 [============================>.] - ETA: 0s - loss: 0.1611 - acc: 0.9384
Epoch 00008: val_acc improved from 0.93037 to 0.93100, saving model to model.h5
14400/14400 [==============================] - 261s 18ms/step - loss: 0.1611 - acc: 0.9384 - val_loss: 0.1787 - val_acc: 0.9310
Epoch 9/20
14395/14400 [============================>.] - ETA: 0s - loss: 0.1564 - acc: 0.9403
Epoch 00009: val_acc improved from 0.93100 to 0.93712, saving model to model.h5
14400/14400 [==============================] - 262s 18ms/step - loss: 0.1564 - acc: 0.9403 - val_loss: 0.1654 - val_acc: 0.9371
Epoch 10/20
14398/14400 [============================>.] - ETA: 0s - loss: 0.1509 - acc: 0.9427
Epoch 00010: val_acc did not improve from 0.93712
14400/14400 [==============================] - 261s 18ms/step - loss: 0.1509 - acc: 0.9427 - val_loss: 0.1745 - val_acc: 0.9311
Epoch 11/20
14399/14400 [============================>.] - ETA: 0s - loss: 0.1487 - acc: 0.9434
Epoch 00011: val_acc did not improve from 0.93712

Epoch 00011: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-05.
14400/14400 [==============================] - 262s 18ms/step - loss: 0.1487 - acc: 0.9434 - val_loss: 0.1741 - val_acc: 0.9350
Epoch 12/20
14398/14400 [============================>.] - ETA: 0s - loss: 0.1334 - acc: 0.9500
Epoch 00012: val_acc did not improve from 0.93712
14400/14400 [==============================] - 262s 18ms/step - loss: 0.1334 - acc: 0.9500 - val_loss: 0.1723 - val_acc: 0.9371
Epoch 13/20
14397/14400 [============================>.] - ETA: 0s - loss: 0.1302 - acc: 0.9513
Epoch 00013: val_acc did not improve from 0.93712
```

```
Epoch 00013: ReduceLROnPlateau reducing learning rate to 1.249999968422344e-05.
14400/14400 [==============================] - 262s 18ms/step - loss: 0.1302 - acc: 0.9513 - val_loss: 0.1672 - val_acc: 0.9363
Epoch 14/20
14399/14400 [============================>.] - ETA: 0s - loss: 0.1211 - acc: 0.9548
Epoch 00014: val_acc improved from 0.93712 to 0.94175, saving model to model.h5
14400/14400 [==============================] - 262s 18ms/step - loss: 0.1211 - acc: 0.9548 - val_loss: 0.1603 - val_acc: 0.9417
Epoch 15/20
14396/14400 [============================>.] - ETA: 0s - loss: 0.1197 - acc: 0.9549
Epoch 00015: val_acc improved from 0.94175 to 0.94362, saving model to model.h5
14400/14400 [==============================] - 263s 18ms/step - loss: 0.1197 - acc: 0.9549 - val_loss: 0.1545 - val_acc: 0.9436
Epoch 16/20
14398/14400 [============================>.] - ETA: 0s - loss: 0.1176 - acc: 0.9559
Epoch 00016: val_acc improved from 0.94362 to 0.94362, saving model to model.h5
14400/14400 [==============================] - 260s 18ms/step - loss: 0.1176 - acc: 0.9559 - val_loss: 0.1550 - val_acc: 0.9436
Epoch 17/20
14397/14400 [============================>.] - ETA: 0s - loss: 0.1169 - acc: 0.9566
Epoch 00017: val_acc improved from 0.94362 to 0.94850, saving model to model.h5
14400/14400 [==============================] - 261s 18ms/step - loss: 0.1169 - acc: 0.9566 - val_loss: 0.1458 - val_acc: 0.9485
Epoch 18/20
14396/14400 [============================>.] - ETA: 0s - loss: 0.1151 - acc: 0.9569
Epoch 00018: val_acc did not improve from 0.94850
14400/14400 [==============================] - 263s 18ms/step - loss: 0.1151 - acc: 0.9569 - val_loss: 0.1549 - val_acc: 0.9471
Epoch 19/20
14399/14400 [============================>.] - ETA: 0s - loss: 0.1157 - acc: 0.9567
Epoch 00019: val_acc did not improve from 0.94850

Epoch 00019: ReduceLROnPlateau reducing learning rate to 1e-05.
14400/14400 [==============================] - 263s 18ms/step - loss: 0.1157 - acc: 0.9567 - val_loss: 0.1495 - val_acc: 0.9478
Epoch 20/20
14397/14400 [============================>.] - ETA: 0s - loss: 0.1121 - acc: 0.9583
Epoch 00020: val_acc did not improve from 0.94850
14400/14400 [==============================] - 260s 18ms/step - loss: 0.1121 - acc: 0.9583 - val_loss: 0.1497 - val_acc: 0.9454
```

Evaluate the model using the val set

In [49]:

```python
# Here the best epoch will be used.

# Loading Weights file
model.load_weights('model.h5')

val_loss, val_acc = \
model.evaluate_generator(test_gen,
                        steps=len(df_val))

print('val_loss:', val_loss)
print('val_acc:', val_acc)
```

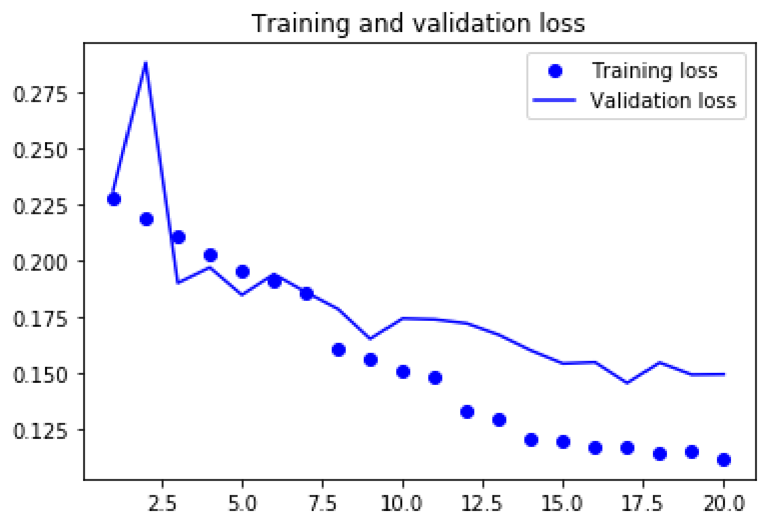
```
val_loss: 0.1457677728444852
val_acc: 0.9485
```

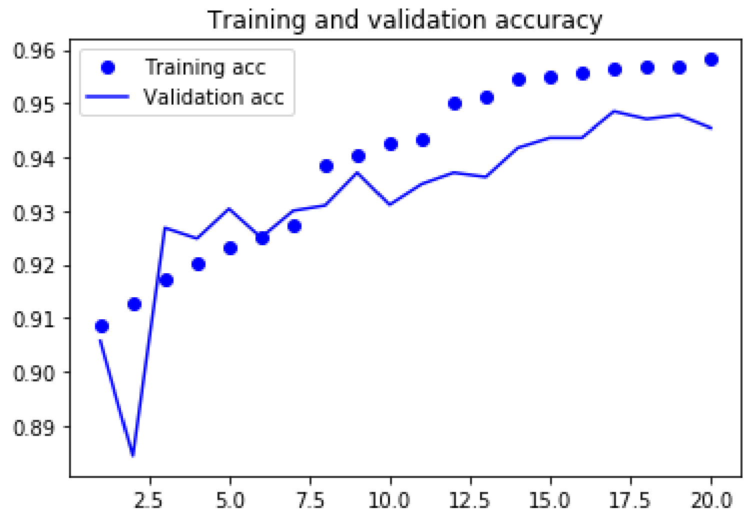
Plot the Training Curves

In [50]:

```python
# Display the loss and accuracy curves

import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.figure()

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
```

Out[50]: `<Figure size 432x288 with 0 Axes>`






```
<Figure size 432x288 with 0 Axes>
```

Predictions

In [51]:
```
predictions = model.predict_generator(test_gen, steps=len(df_val), verbose=1)
```

```
16000/16000 [==============================] - 42s 3ms/step
```

In [52]:
```
# To check what index keras has internally assigned to each class.
test_gen.class_indices
```

Out[52]:
```
{'a_no_tumor_tissue': 0, 'b_has_tumor_tissue': 1}
```

In [53]:
```
# The columns need to be ordered to match the output of the previous cell
# Appending predictions to  a dataframe object
df_preds = pd.DataFrame(predictions, columns=['no_tumor_tissue', 'has_tumor_tissue'])
df_preds.head()
```

Out[53]:

	no_tumor_tissue	has_tumor_tissue
0	0.951198	0.048802
1	0.992302	0.007698
2	0.926679	0.073321
3	0.999828	0.000172
4	0.974727	0.025273

In [54]:
```
# Get the true labels
y_true = test_gen.classes

# Get the predicted labels as probabilities
y_pred = df_preds['has_tumor_tissue']
```

Metrics

In [55]:
```
# AUC Score
from sklearn.metrics import roc_auc_score
roc_auc_score(y_true, y_pred)
```

Out[55]:
```
0.9868213203125001
```

In [56]:
```
# Confusion Matrix
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
```

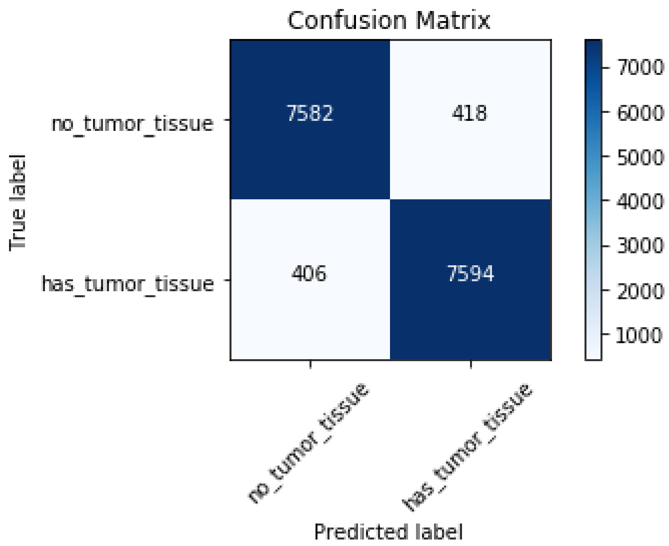
```
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
```

In [57]:
```
# labels of the test images.
test_labels = test_gen.classes
```

In [58]:
```
# argmax returns the index of the max value in a row
cm = confusion_matrix(test_labels, predictions.argmax(axis=1))
```

In [59]:
```
# Define the labels of the class indices. These need to match the order shown above.
cm_plot_labels = ['no_tumor_tissue', 'has_tumor_tissue']
plot_confusion_matrix(cm, cm_plot_labels, title='Confusion Matrix')
```

```
Confusion matrix, without normalization
[[7582  418]
 [ 406 7594]]
```



In [60]:
```
# Classification Report
from sklearn.metrics import classification_report

# Generate a classification report

# For this to work we need y_pred as binary labels not as probabilities
y_pred_binary = predictions.argmax(axis=1)

report = classification_report(y_true, y_pred_binary, target_names=cm_plot_labels)

print(report)
```

```
                  precision    recall  f1-score   support

 no_tumor_tissue       0.95      0.95      0.95      8000
has_tumor_tissue       0.95      0.95      0.95      8000

     avg / total       0.95      0.95      0.95     16000
```