COSC 4337

Keras_First_CNN

[1]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatten,␣
↪BatchNormalization, Conv2D, MaxPool2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from sklearn.metrics import confusion_matrix
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import itertools
import os
import random
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
train_path = "chest_xray/train"
test_path = "chest_xray/test"
valid_path = "chest_xray/val"
```

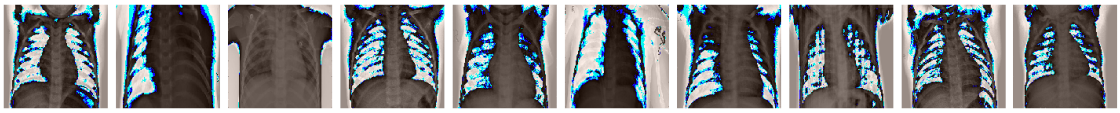
[3]:
```
# Creating train, test and valid batches from the respective directories
train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.
↪vgg16.preprocess_input).flow_from_directory(directory=train_path,␣
↪target_size=(224,224), classes=['pneumonia', 'normal'], batch_size=10)
valid_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.
↪vgg16.preprocess_input).flow_from_directory(directory=valid_path,␣
↪target_size=(224,224), classes=['pneumonia', 'normal'], batch_size=10)
test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.
↪vgg16.preprocess_input).flow_from_directory(directory=test_path,␣
↪target_size=(224,224), classes=['pneumonia', 'normal'], batch_size=10,␣
↪shuffle=False)
```

```
Found 5216 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
```

[4]:
```
# plot images in the form of a 1 by 10 grid and resize img to 20x20
def plotImages(images_arr):
```

```
    fig, axes = plt.subplots(1, 10, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img.astype(np.uint8))
        ax.axis('off')
    plt.tight_layout()
    plt.show()
```

[5]:
```
imgs, labels = next(train_batches)
plotImages(imgs)
print(labels)
```



```
[[0. 1.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [0. 1.]]
```

[6]:
```
model = Sequential([
    Conv2D(filters=32, kernel_size=(3, 3), activation='relu', padding = 'same',␣
 ↪input_shape=(224,224,3)),
    MaxPool2D(pool_size=(2, 2), strides=2),
    Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPool2D(pool_size=(2, 2), strides=2),
    Flatten(),
    Dense(units=2, activation='softmax')
])
```

```
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\python\ops\resource_variable_ops.py:1630: calling
BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops)
with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

[7]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 200704)            0
_________________________________________________________________
dense (Dense)                (None, 2)                 401410
=================================================================
Total params: 420,802
Trainable params: 420,802
Non-trainable params: 0
_________________________________________________________________
```

[8]:
```
model.compile(optimizer=Adam(learning_rate=0.0001),
loss='categorical_crossentropy', metrics=['accuracy'])
```

[9]:
```
# recall that we set batch_size = 10 during preprocessing
batch_size = 10

model.fit(
    x = train_batches,
    steps_per_epoch=train_batches.samples // batch_size,
    epochs=10,
    validation_data=valid_batches,
    validation_steps=valid_batches.samples // batch_size,
    verbose=2)
```

```
Epoch 1/10
Epoch 1/10
521/521 - 244s - loss: 2.4432 - acc: 0.9291 - val_loss: 1.5285 - val_acc: 0.9000
Epoch 2/10
Epoch 1/10
521/521 - 222s - loss: 0.2381 - acc: 0.9721 - val_loss: 0.6962 - val_acc: 0.9000
Epoch 3/10
Epoch 1/10
521/521 - 226s - loss: 0.0602 - acc: 0.9889 - val_loss: 1.6920 - val_acc: 0.8000
Epoch 4/10
Epoch 1/10
521/521 - 224s - loss: 0.0223 - acc: 0.9940 - val_loss: 0.1357 - val_acc: 0.9000
Epoch 5/10
```

```
Epoch 1/10
521/521 - 229s - loss: 0.0042 - acc: 0.9983 - val_loss: 0.0070 - val_acc: 1.0000
Epoch 6/10
Epoch 1/10
521/521 - 224s - loss: 0.0038 - acc: 0.9987 - val_loss: 0.6665 - val_acc: 0.8000
Epoch 7/10
Epoch 1/10
521/521 - 223s - loss: 1.1300e-04 - acc: 1.0000 - val_loss: 0.6575 - val_acc:
0.8000
Epoch 8/10
Epoch 1/10
521/521 - 227s - loss: 2.1484e-05 - acc: 1.0000 - val_loss: 0.6330 - val_acc:
0.8000
Epoch 9/10
Epoch 1/10
521/521 - 220s - loss: 1.6232e-05 - acc: 1.0000 - val_loss: 0.6668 - val_acc:
0.8000
Epoch 10/10
Epoch 1/10
521/521 - 220s - loss: 1.2365e-05 - acc: 1.0000 - val_loss: 0.6070 - val_acc:
0.8000
```

[9]:
```
<tensorflow.python.keras.callbacks.History at 0x1fdacb41208>
```

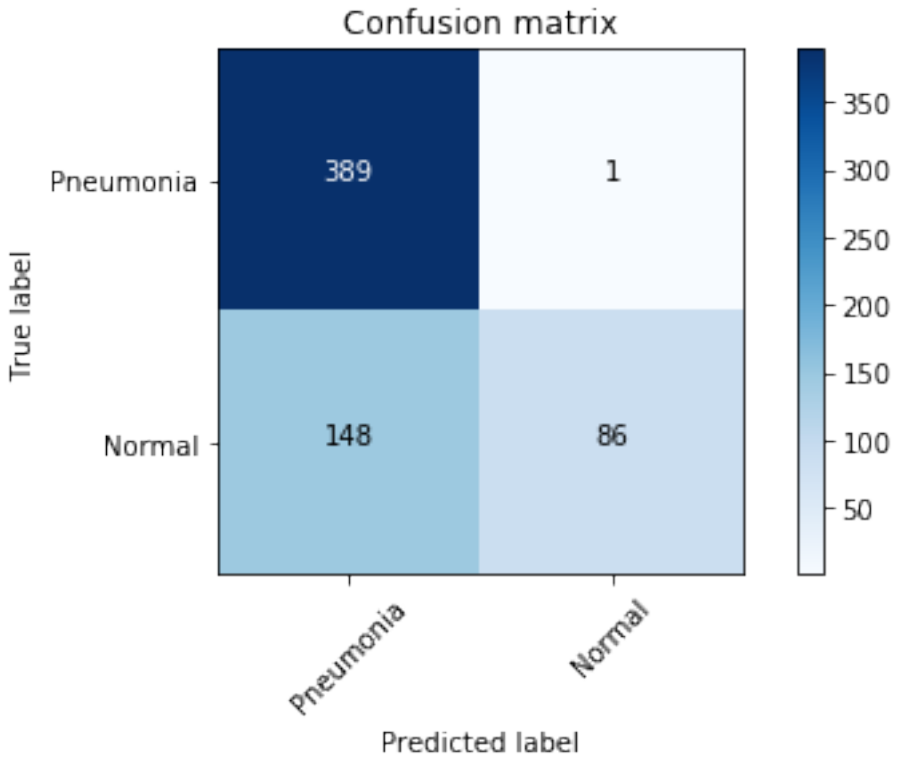
[10]:
```
# making predictions
predictions = model.predict(x = test_batches, verbose=0)
```

[13]:
```
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`.
    """
    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)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    print(cm)
    thresh = cm.max() / 2.
```

```
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

[14]:
```
# confusion matrix
cm = confusion_matrix(y_true=test_batches.classes, y_pred=np.
 ↪argmax(predictions, axis=-1))
plot_confusion_matrix(cm, classes = ['Pneumonia', 'Normal'])
```

```
Confusion matrix, without normalization
[[389   1]
 [148  86]]
```



[ ]:
```

```