<u>CS6005 - Deep Learning</u>

V.S. Suryaa

2018103610

P-Batch

Problem statement:

The idea is to train a CNN model to predict Pneumonia from Chest X-ray images. The model has been trained to identify if the patient is normal or affected with Pneumonia from CXR images. The data used to train the model are from patients suffering from Bacterial and Viral Pneumonia.

Dataset:

The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal).

Chest X-ray images (anterior-posterior) were selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center, Guangzhou. All chest X-ray imaging was performed as part of patients' routine clinical care.

Modules:

Reading data:

The data is segregated into train, test and validation data with separate arrays for features and target values. The images were scaled down to 3 channels of 128 x 128 resolution (RGB channel).

Pre-processing:

The image pixel values were scaled between 0 and 1 by dividing by 255. The zoom range was set to 0.2 in ImageDataGenerator which randomly zooms the images in range of 0 to 20%.

Model building:

The model is based upon AlexNet architecture.

1. Convolution layer with 96 filters of kernel size (4,4) with a stride of (1,1) and no padding. ReLU activation layer is used. Further on, Maxpooling of size (2,2) and stride of (2,2) is used.
2. Convolution layer with 256 filters of kernel size (4,4) with a stride of (1,1) and no padding. ReLU activation layer is used. Further on, Maxpooling of size (2,2) and stride of (2,2) is used.

3. Convolution layer with 256 filters of kernel size (4,4) with a stride of (1,1) and no padding. ReLU activation layer is used. Further on, Maxpooling of size (2,2) and stride of (1,1) is used.
4. Convolution layer with 256 filters of kernel size (4,4) with a stride of (1,1) and no padding. ReLU activation layer is used.
5. The output from the previous layer is flattened.
6. Dense layer with 4086 units and dropout of 0.4 is added with ReLU activation.
7. Dense layer with 4096 units and dropout of 0.4 is added with ReLU activation.
8. Dense layer with 1000 units and dropout of 0.4 is added with ReLU activation.
9. Dense layer with 2 units is used as output layer with Softmax activation.
10. Model is compiled with loss as Sparse Categorical Crossentropy, optimizer is Adam and metrics as Accuracy.

Training and visualizing results:

Early stopping is used with patience as 4 with monitoring validation accuracy. The model is trained with 13 epochs and batch size of 32. Further the model is evaluated with accuracy metrics. Further on, model accuracy graph and model loss graph is plotted for visualization with confusion matrix.

Parameters:

Learning rate	0.001
Optimizer	Adam
Loss	Sparse Categorical Crossentropy
Epochs	13
Batch size	32

CNN summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 125, 125, 96)      4704
_________________________________________________________________
activation (Activation)      (None, 125, 125, 96)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 62, 62, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 59, 59, 256)       393472
_________________________________________________________________
activation_1 (Activation)    (None, 59, 59, 256)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 29, 29, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 26, 26, 256)       1048832
_________________________________________________________________
activation_2 (Activation)    (None, 26, 26, 256)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 25, 25, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 22, 22, 256)       1048832
_________________________________________________________________
activation_3 (Activation)    (None, 22, 22, 256)       0
_________________________________________________________________
flatten (Flatten)            (None, 123904)            0
_________________________________________________________________
dense (Dense)                (None, 4086)              506275830
_________________________________________________________________
activation_4 (Activation)    (None, 4086)              0
_________________________________________________________________
dropout (Dropout)            (None, 4086)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16740352
_________________________________________________________________
activation_5 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              4097000
_________________________________________________________________
activation_6 (Activation)    (None, 1000)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 2002
_________________________________________________________________
activation_7 (Activation)    (None, 2)                 0
_________________________________________________________________
activation_7 (Activation)    (None, 2)                 0
=================================================================
Total params: 529,611,024
Trainable params: 529,611,024
Non-trainable params: 0
_________________________________________________________________
```

Coding Snapshots:

Importing libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPooling2D , Flatten , Dropout , BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from keras.callbacks import ReduceLROnPlateau
import cv2
import os
```

Data segregation

In [2]:

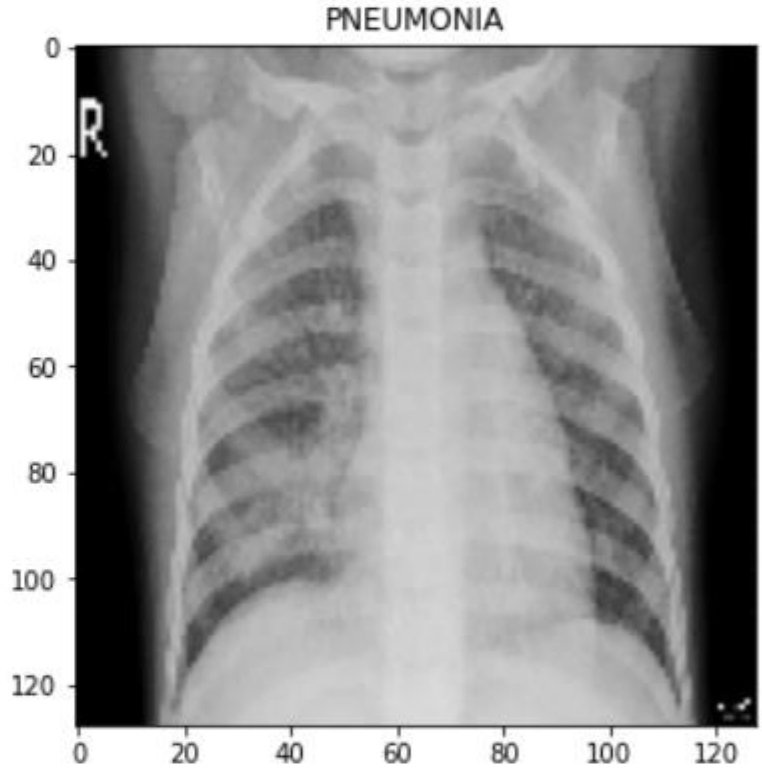
```
labels = ['PNEUMONIA', 'NORMAL']
img_size = 128
def datafunc(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                img_arr = cv2.cvtColor(img_arr,cv2.COLOR_GRAY2RGB)
                resized_arr = cv2.resize(img_arr, (img_size, img_size))
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)
```

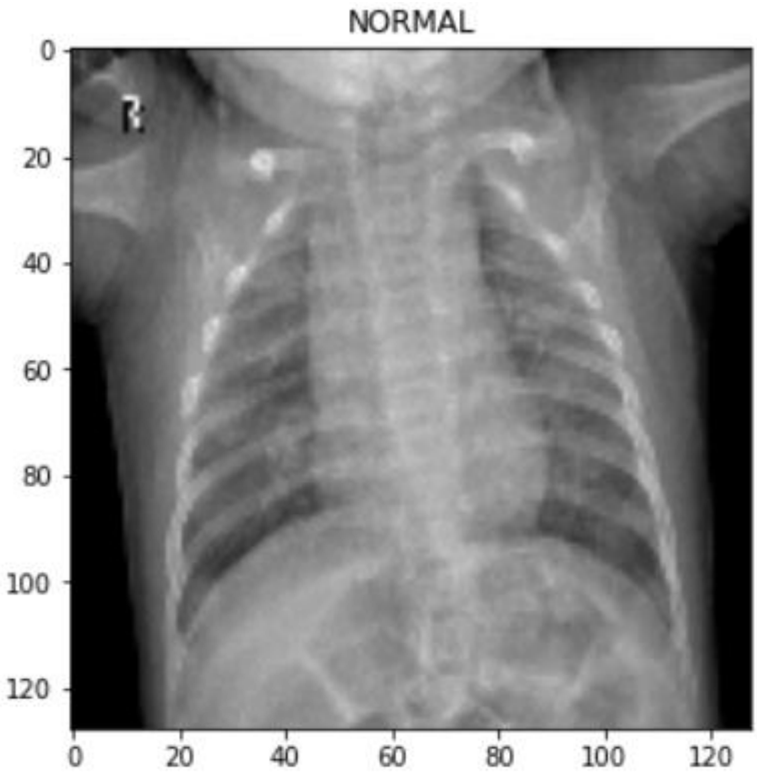
In [3]:

```
train = datafunc('../input/chest-xray-pneumonia/chest_xray/chest_xray/train')
test = datafunc('../input/chest-xray-pneumonia/chest_xray/chest_xray/test')
val = datafunc('../input/chest-xray-pneumonia/chest_xray/chest_xray/val')
```

```
OpenCV(4.3.0) /io/opencv/modules/imgproc/src/color.cpp:182: error: (-215:Assertion failed) !_src.empty() in

OpenCV(4.3.0) /io/opencv/modules/imgproc/src/color.cpp:182: error: (-215:Assertion failed) !_src.empty() in

OpenCV(4.3.0) /io/opencv/modules/imgproc/src/color.cpp:182: error: (-215:Assertion failed) !_src.empty() in

OpenCV(4.3.0) /io/opencv/modules/imgproc/src/color.cpp:182: error: (-215:Assertion failed) !_src.empty() in
```

```
In [5]:  plt.figure(figsize = (5,5))
         plt.imshow(train[0][0], cmap='gray')
         plt.title(labels[train[0][1]])

         plt.figure(figsize = (5,5))
         plt.imshow(train[-1][0], cmap='gray')
         plt.title(labels[train[-1][1]])

Out[5]: Text(0.5, 1.0, 'NORMAL')
```





In [6]:

```
x_train = []
y_train = []

x_val = []
y_val = []

x_test = []
y_test = []

for feature, label in train:
    x_train.append(feature)
    y_train.append(label)

for feature, label in test:
    x_test.append(feature)
    y_test.append(label)

for feature, label in val:
    x_val.append(feature)
    y_val.append(label)
```

Data processing

In [7]:
```
x_train = np.array(x_train)/255.0
x_test = np.array(x_test)/255.0
x_val = np.array(x_val)/255.0
```

In [8]:
```
x_train = (x_train.reshape(-1,img_size,img_size,3))
x_test = (x_test.reshape(-1,img_size,img_size,3))
x_val = (x_val.reshape(-1,img_size,img_size,3))
x_train.shape
```

Out[8]:
```
(5216, 128, 128, 3)
```

In [9]:
```
y_train=np.array(y_train)
y_test=np.array(y_test)
y_val=np.array(y_val)
```

Building the model

In [10]:
```
datagen= ImageDataGenerator(zoom_range=0.2)
```

In [11]:
```
datagen.fit(x_train)
```

In [12]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten, Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
import numpy as np
np.random.seed(1000)
#Instantiate an empty model
model = Sequential()

# 1st Convolutional Layer
model.add(Conv2D(filters=96, input_shape=(128,128,3), kernel_size=(4,4), strides=(1,1), padding="valid"))
model.add(Activation("relu"))
```

In [12]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten, Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
import numpy as np
np.random.seed(1000)
#Instantiate an empty model
model = Sequential()

# 1st Convolutional Layer
model.add(Conv2D(filters=96, input_shape=(128,128,3), kernel_size=(4,4), strides=(1,1), padding="valid"))
model.add(Activation("relu"))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding="valid"))

# 2nd Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(4,4), strides=(1,1), padding="valid"))
model.add(Activation("relu"))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding="valid"))

model.add(Conv2D(filters=256, kernel_size=(4,4), strides=(1,1), padding="valid"))
model.add(Activation("relu"))

model.add(MaxPooling2D(pool_size=(2,2), strides=(1,1), padding="valid"))

model.add(Conv2D(filters=256, kernel_size=(4,4), strides=(1,1), padding="valid"))
model.add(Activation("relu"))

# Passing it to a Fully Connected layer
model.add(Flatten())
# 1st Fully Connected Layer
model.add(Dense(4086, input_shape=(128*128*3,)))
model.add(Activation("relu"))
# Add Dropout to prevent overfitting
model.add(Dropout(0.4))

# 2nd Fully Connected Layer
model.add(Dense(4096))
model.add(Activation("relu"))
# Add Dropout
model.add(Dropout(0.4))

# 3rd Fully Connected Layer
```

```
# 3rd Fully Connected Layer
model.add(Dense(1000))
model.add(Activation("relu"))
# Add Dropout
model.add(Dropout(0.4))

# Output Layer
model.add(Dense(2))
model.add(Activation("softmax"))

model.summary()

# Compile the model
model.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 125, 125, 96)      4704
_________________________________________________________________
activation (Activation)      (None, 125, 125, 96)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 62, 62, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 59, 59, 256)       393472
_________________________________________________________________
activation_1 (Activation)    (None, 59, 59, 256)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 29, 29, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 26, 26, 256)       1048832
_________________________________________________________________
activation_2 (Activation)    (None, 26, 26, 256)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 25, 25, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 22, 22, 256)       1048832
_________________________________________________________________
activation_3 (Activation)    (None, 22, 22, 256)       0
_________________________________________________________________
flatten (Flatten)            (None, 123904)            0
_________________________________________________________________
dense (Dense)                (None, 4086)              506275830
_________________________________________________________________
activation_4 (Activation)    (None, 4086)              0
```

```
max_pooling2d (MaxPooling2D) (None, 62, 62, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 59, 59, 256)       393472
_________________________________________________________________
activation_1 (Activation)    (None, 59, 59, 256)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 29, 29, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 26, 26, 256)       1048832
_________________________________________________________________
activation_2 (Activation)    (None, 26, 26, 256)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 25, 25, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 22, 22, 256)       1048832
_________________________________________________________________
activation_3 (Activation)    (None, 22, 22, 256)       0
_________________________________________________________________
flatten (Flatten)            (None, 123904)            0
_________________________________________________________________
dense (Dense)                (None, 4086)              506275830
_________________________________________________________________
activation_4 (Activation)    (None, 4086)              0
_________________________________________________________________
dropout (Dropout)            (None, 4086)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16740352
_________________________________________________________________
activation_5 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              4097000
_________________________________________________________________
activation_6 (Activation)    (None, 1000)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 2002
_________________________________________________________________
activation_7 (Activation)    (None, 2)                 0
=================================================================
Total params: 529,611,024
Trainable params: 529,611,024
Non-trainable params: 0
_________________________________________________________________
```

Training and visualizing the results

In [13]:

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
checkpoint = ModelCheckpoint("own.h5", monitor='val_acc', verbose=1, save_best_only=True, save_weights_only=False, mode='auto', period=1)
early = EarlyStopping(monitor='val_acc', min_delta=0, patience=4, verbose=1, mode='auto')
history = model.fit(datagen.flow(x_train,y_train, batch_size = 32) ,epochs = 13 , validation_data = datagen.flow(x_test, y_test),callbacks=[checkpoint,early])
```

```
Epoch 1/13
163/163 [==============================] - 28s 171ms/step - loss: 1.2982 - accuracy: 0.7544 - val_loss: 0.7831 - val_accuracy: 0.6955
Epoch 2/13
163/163 [==============================] - 27s 166ms/step - loss: 0.3584 - accuracy: 0.8507 - val_loss: 0.6492 - val_accuracy: 0.6554
Epoch 3/13
163/163 [==============================] - 28s 169ms/step - loss: 0.2863 - accuracy: 0.8863 - val_loss: 0.5522 - val_accuracy: 0.7692
Epoch 4/13
163/163 [==============================] - 27s 167ms/step - loss: 0.2880 - accuracy: 0.8802 - val_loss: 0.5969 - val_accuracy: 0.6987
Epoch 5/13
163/163 [==============================] - 27s 169ms/step - loss: 0.2953 - accuracy: 0.8827 - val_loss: 0.6848 - val_accuracy: 0.6971
Epoch 6/13
163/163 [==============================] - 27s 168ms/step - loss: 0.2522 - accuracy: 0.9045 - val_loss: 0.4560 - val_accuracy: 0.8013
Epoch 7/13
163/163 [==============================] - 28s 170ms/step - loss: 0.2157 - accuracy: 0.9135 - val_loss: 0.4532 - val_accuracy: 0.7949
Epoch 8/13
163/163 [==============================] - 28s 170ms/step - loss: 0.2029 - accuracy: 0.9201 - val_loss: 0.5702 - val_accuracy: 0.8061
Epoch 9/13
163/163 [==============================] - 27s 165ms/step - loss: 0.2034 - accuracy: 0.9231 - val_loss: 0.3922 - val_accuracy: 0.8542
Epoch 10/13
163/163 [==============================] - 28s 172ms/step - loss: 0.1788 - accuracy: 0.9344 - val_loss: 0.3822 - val_accuracy: 0.8510
Epoch 11/13
163/163 [==============================] - 27s 165ms/step - loss: 0.1614 - accuracy: 0.9358 - val_loss: 0.4946 - val_accuracy: 0.8766
Epoch 12/13
163/163 [==============================] - 28s 170ms/step - loss: 0.1563 - accuracy: 0.9388 - val_loss: 0.2816 - val_accuracy: 0.8910
Epoch 13/13
163/163 [==============================] - 27s 168ms/step - loss: 0.1553 - accuracy: 0.9379 - val_loss: 0.4902 - val_accuracy: 0.8542
```

In [14]:

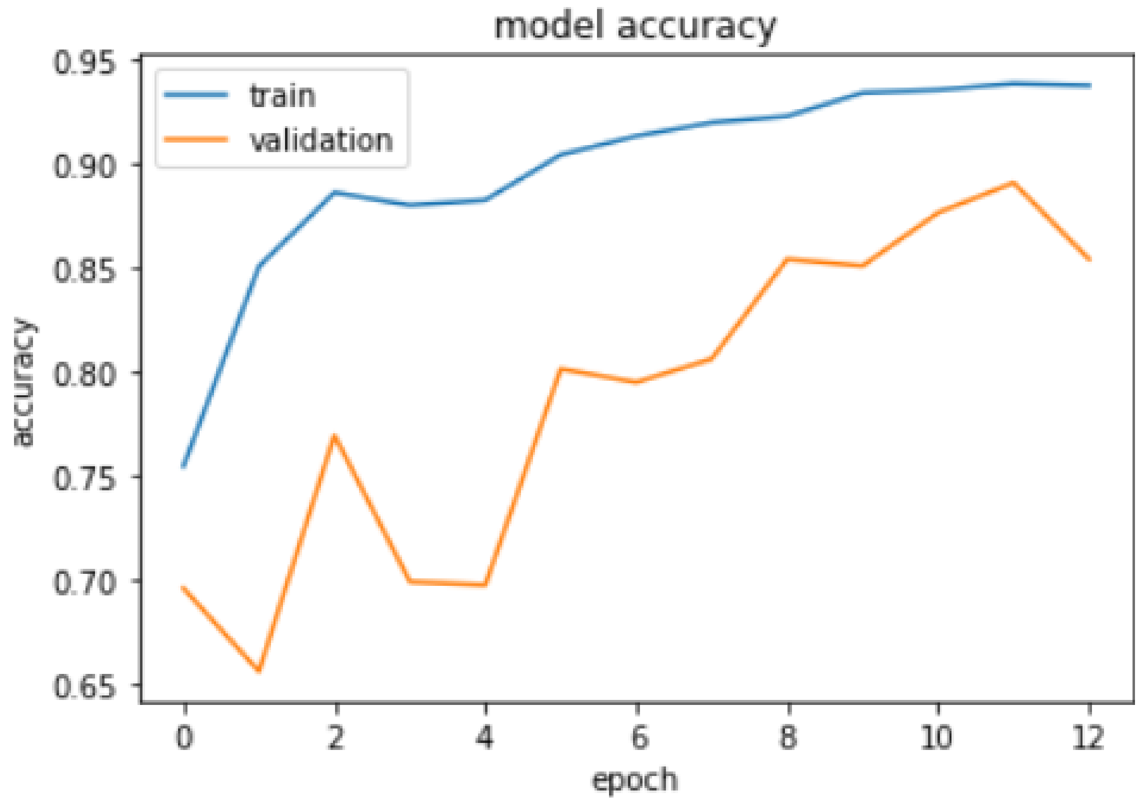
```
print("Loss of the model is - " , model.evaluate(x_test,y_test)[0])
print("Accuracy of the model is - " , model.evaluate(x_test,y_test)[1]*100 , "%")
```

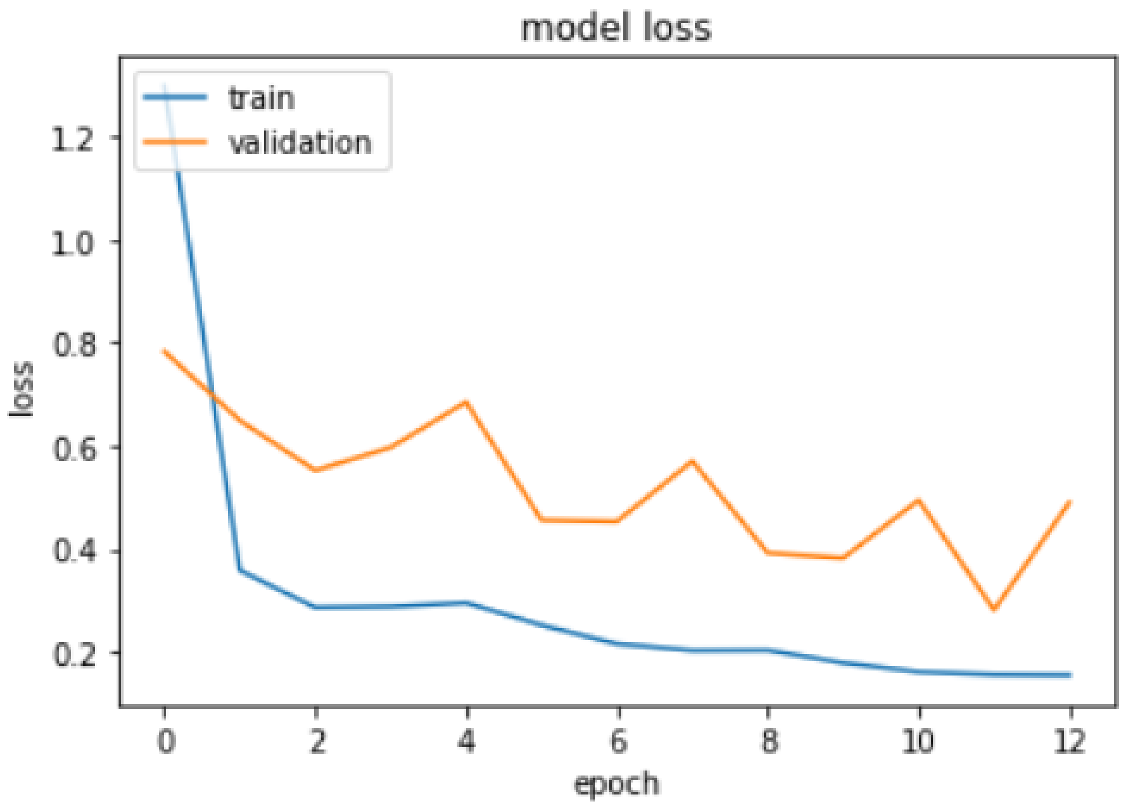
```
20/20 [==============================] - 0s 24ms/step - loss: 0.2950 - accuracy: 0.9071
Loss of the model is -  0.29504522681236267
20/20 [==============================] - 0s 24ms/step - loss: 0.2950 - accuracy: 0.9071
Accuracy of the model is -  90.70512652397156 %
```

In [15]:

```
print(history.history.keys())
#  "Accuracy"
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
# "Loss"
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

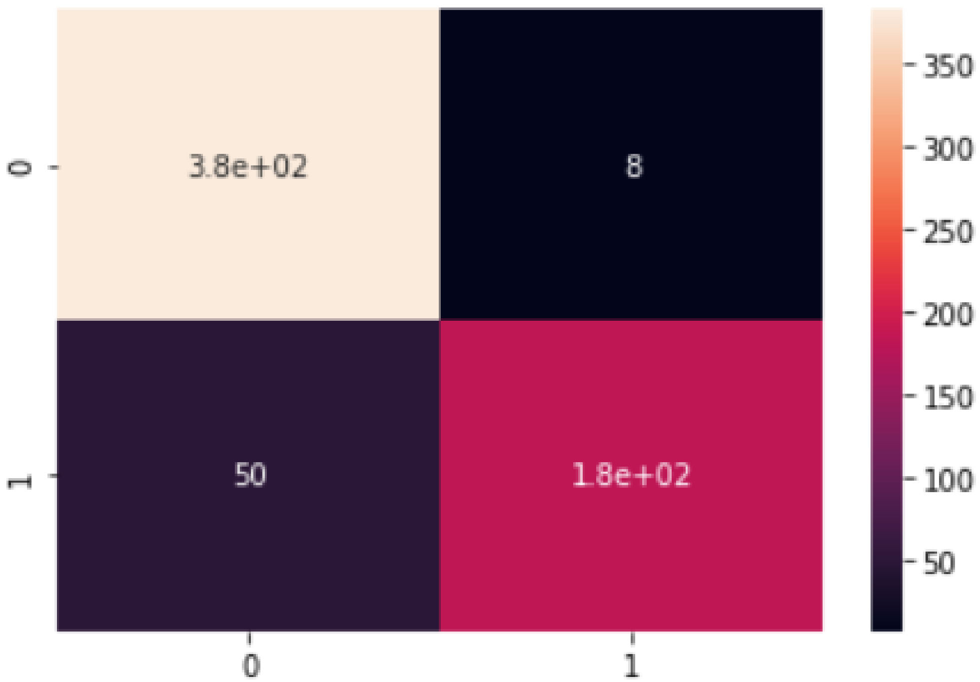

model accuracy
train
validation
accuracy
epoch
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
2
4
6
8
10
12


model loss
train
validation
loss
epoch
1.2
1.0
0.8
0.6
0.4
0.2
0
2
4
6
8
10
12

In [16]:
```
predictions = model.predict(x_test)
predictions = predictions[:,0]
i=0
for i in range(len(predictions)):
    if predictions[i]>0.5:
        predictions[i]=0
    else:
        predictions[i]=1
```

In [17]:
```
cm = confusion_matrix(y_test,predictions)
sns.heatmap(cm, annot=True)
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff2f



In [18]:
```
cm
```

Out[18]:
```
array([[382,   8],
       [ 50, 184]])
```

Results:

The model obtains 90.705% accuracy on the test data and a loss of 0.295.

```
20/20 [==============================] - 0s 24ms/step - loss: 0.2950 - accuracy: 0.9071
Loss of the model is -  0.29504522681236267
20/20 [==============================] - 0s 24ms/step - loss: 0.2950 - accuracy: 0.9071
Accuracy of the model is -  90.70512652397156 %
```

s

Confusion Matrix:

TP = 382	FP = 8
FN = 50	TN = 184

References:

- Dataset: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
- AlexNet reference: https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf