In [4]:
```
# after doing the data preprocessing and making feature and target vector nc
#lets get started
```

In [6]:
```
import pickle
import numpy as np
pickle_in=open("x.pickle","rb")
x=pickle.load(pickle_in)
pickle_iny=open("y.pickle","rb")
y=pickle.load(pickle_iny)
x=np.array(x)
y=np.array(y)
print(x.shape)
from sklearn.model_selection import train_test_split
train_ratio = 0.80
test_ratio = 0.20

# train is now 80% of the entire data set

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=1 - trai

# test is now 20% of the initial data set
```

```
(424, 224, 224, 3)
```

In [7]:
```
x_train.shape[1:]
```

Out[7]:
```
(224, 224, 3)
```

In [8]:

```
#study material at the back
import numpy as np
import keras
from keras.layers import *
from keras.models import *
from keras.preprocessing import image

import matplotlib.pyplot as plt

model = Sequential()
model.add(Conv2D(32,kernel_size=(3, 3), activation='relu', input_shape=x_tra
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64,(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128,(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1,activation='sigmoid'))




model.compile(
              loss=keras.losses.binary_crossentropy,optimizer='adam',
              metrics=['accuracy'])

print(model.summary())
```

```
Using TensorFlow backend.

WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\kera
s\backend\tensorflow_backend.py:4070: The name tf.nn.max_pool is deprec
ated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tens
orflow\python\ops\nn_impl.py:180: add_dispatch_support.<locals>.wrapper
(from tensorflow.python.ops.array_ops) is deprecated and will be remove
d in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 220, 220, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 110, 110, 64)      0
_________________________________________________________________
dropout_1 (Dropout)          (None, 110, 110, 64)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 108, 108, 64)      36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 52, 52, 128)       73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 26, 26, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 26, 26, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 86528)             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                5537856
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 5,668,097
Trainable params: 5,668,097
Non-trainable params: 0
_________________________________________________________________
None
```

In [9]:
```
history = model.fit(x_train, y_train, epochs=10,
                    validation_data=(x_test, y_test))
```

```
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Train on 339 samples, validate on 85 samples
Epoch 1/10
339/339 [==============================] - 112s 331ms/step - loss: 134.0084 - accuracy: 0.5457 - val_loss: 1.4404 - val_accuracy: 0.4706
Epoch 2/10
339/339 [==============================] - 96s 284ms/step - loss: 1.9080 - accuracy: 0.5310 - val_loss: 0.6725 - val_accuracy: 0.7529
Epoch 3/10
339/339 [==============================] - 57s 169ms/step - loss: 0.6549 - accuracy: 0.6077 - val_loss: 0.6976 - val_accuracy: 0.5176
Epoch 4/10
339/339 [==============================] - 50s 147ms/step - loss: 0.6357 - accuracy: 0.5959 - val_loss: 0.6597 - val_accuracy: 0.7412
Epoch 5/10
339/339 [==============================] - 55s 161ms/step - loss: 0.5665 - accuracy: 0.6224 - val_loss: 0.5552 - val_accuracy: 0.9176
Epoch 6/10
339/339 [==============================] - 55s 164ms/step - loss: 0.4424 - accuracy: 0.7050 - val_loss: 0.4304 - val_accuracy: 0.8706
Epoch 7/10
339/339 [==============================] - 59s 174ms/step - loss: 0.4911 - accuracy: 0.6814 - val_loss: 0.3198 - val_accuracy: 0.9294
Epoch 8/10
339/339 [==============================] - 56s 164ms/step - loss: 0.5059 - accuracy: 0.7493 - val_loss: 0.4255 - val_accuracy: 0.9294
Epoch 9/10
339/339 [==============================] - 52s 154ms/step - loss: 0.4102 - accuracy: 0.8083 - val_loss: 0.3095 - val_accuracy: 0.9529
Epoch 10/10
339/339 [==============================] - 53s 155ms/step - loss: 0.4148 - accuracy: 0.8201 - val_loss: 0.5651 - val_accuracy: 0.8941
```

In [10]:
```
import os

import tensorflow as tf
from tensorflow import keras
#!mkdir -p saved_model
#save model
model.save('saved_model/my_modelnew3')
#inorder to load model
#new_model = tf.keras.models.load_model('saved_model/my_model')
```

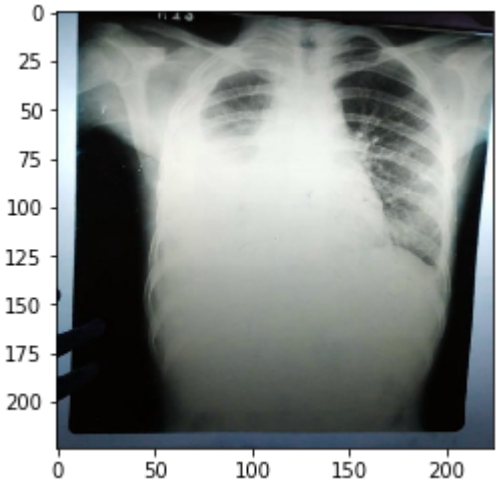
In [11]:
```
print(history.history)
```

```
{'val_loss': [1.4403760026482975, 0.6724572974092821, 0.6975727747468388,
0.6597168368451736, 0.5552176959374372, 0.43036847991101884, 0.319807485271
90267, 0.42550114708788256, 0.30946852915427264, 0.5650910854339599], 'val_
accuracy': [0.47058823704719543, 0.7529411911964417, 0.5176470875740051, 0.
7411764860153198, 0.9176470637321472, 0.8705882430076599, 0.92941176891326
9, 0.929411768913269, 0.9529411792755127, 0.8941176533699036], 'loss': [13
4.00839728240067, 1.9080316482392032, 0.6548569559347664, 0.635653035303132
8, 0.5665473230293016, 0.44244110601841524, 0.4911035599961745, 0.505918208
1250666, 0.41021783627943303, 0.4147836975643417], 'accuracy': [0.5457227,
0.53097343, 0.6076696, 0.5958702, 0.6224189, 0.70501477, 0.6814159, 0.74926
25, 0.8082596, 0.820059]}
```

In [13]:
```
import cv2
path="C:/Users/MACHINE/Documents/5th sem/covid/test7.jpeg"
#C:\Users\MACHINE\Documents\5th sem\covid
newtest=cv2.imread(path,1)
newtest=cv2.resize(newtest,(224,224))
#newtest = np.expand_dims(newtest, axis=0)
print(newtest.dtype)
print(newtest.shape)
plt.imshow(newtest,cmap="gray")
plt.show
#predictions = model.predict(x_test)
```

```
uint8
(224, 224, 3)
```

Out[13]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [15]:
```
newtest = np.expand_dims(newtest, axis=0)
PREDICTED_CLASSES = model.predict_classes(newtest, verbose=1)
```

```
1/1 [==============================] - 0s 138ms/step
```

In [16]:
```
print(PREDICTED_CLASSES)
print("0 means covid +ve and 1 means negative")
```

```
[[0]]
0 means covid +ve and 1 means negative
```

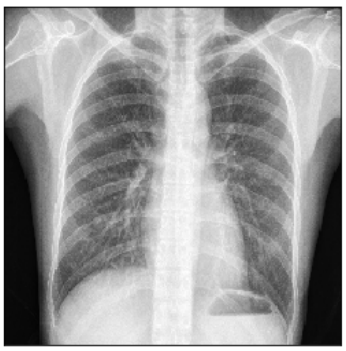
In [ ]:

In [ ]:

In [21]:

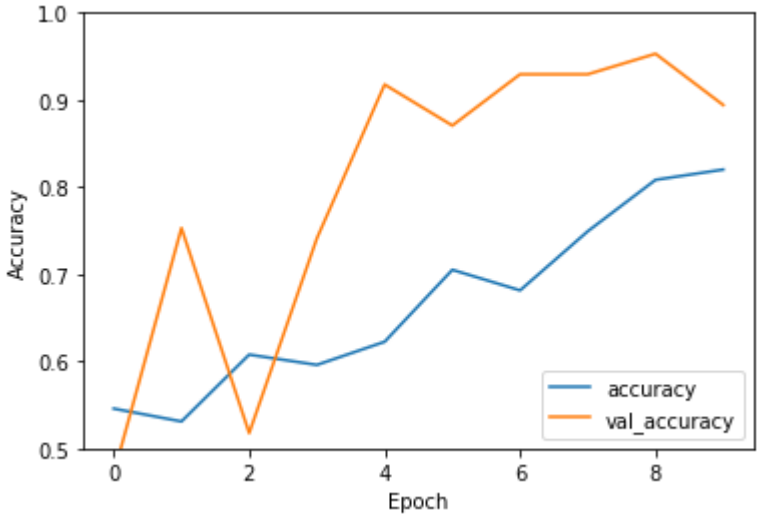
```
print("Evaluate on test data")
results = model.evaluate(x_test, y_test)
print("test loss, test acc:", results)

# Generate predictions (probabilities -- the output of the last layer)
# on new data using `predict`
print("Generate predictions for 3 samples")
predictions = model.predict(x_test[:3])
print("predictions shape:", predictions)
plt.figure(figsize=(15,15))
for i in range(3):
    plt.subplot(1,3,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_test[i], cmap=plt.cm.binary)
    # The CIFAR labels happen to be arrays,
    # which is why you need the extra index

plt.show()
```

```
Evaluate on test data
85/85 [==============================] - 3s 38ms/step
test loss, test acc: [0.5650910854339599, 0.8941176533699036]
Generate predictions for 3 samples
predictions shape: [[0.56009126]
 [0.57929313]
 [0.58650166]]
```


In [23]:

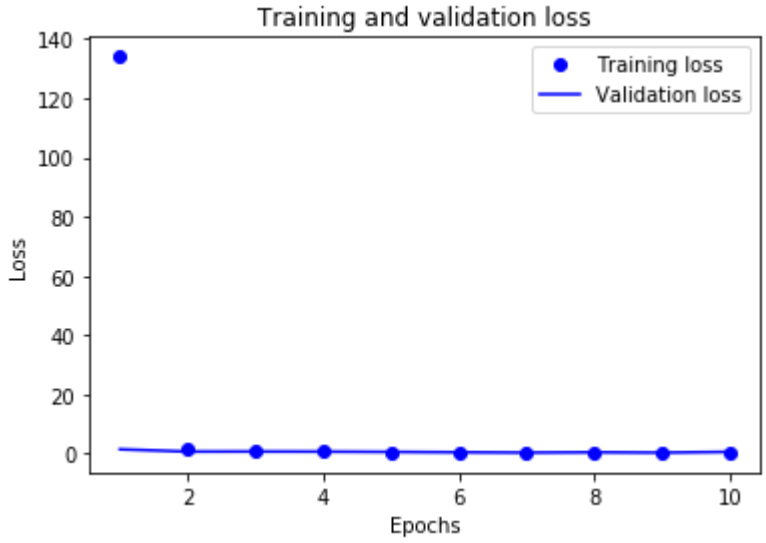
```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(x_test,  y_test, verbose=2)
```



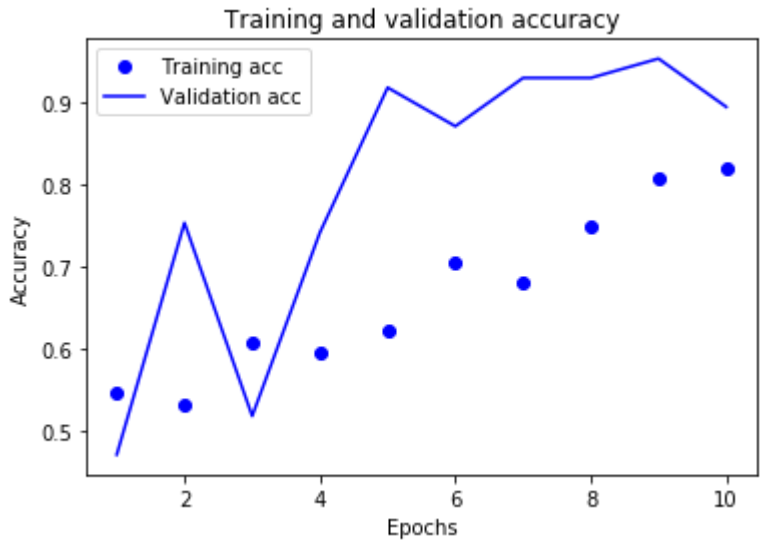
In [31]:

```
import matplotlib.pyplot as plt

acc = history.history['accuracy']
print("accuracy",acc)
val_acc = history.history['val_accuracy']
print("accuracy value",acc)
loss = history.history['loss']
print("losss",acc)
val_loss = history.history['val_loss']
print("loss value",acc)

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

# "bo" is for "blue dot"
plt.plot(epochs, loss, 'bo', label='Training loss')
# b is for "solid blue line"
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
%matplotlib inline
```

```
accuracy [0.5457227, 0.53097343, 0.6076696, 0.5958702, 0.6224189, 0.7050147
7, 0.6814159, 0.7492625, 0.8082596, 0.820059]
accuracy value [0.5457227, 0.53097343, 0.6076696, 0.5958702, 0.6224189, 0.7
0501477, 0.6814159, 0.7492625, 0.8082596, 0.820059]
losss [0.5457227, 0.53097343, 0.6076696, 0.5958702, 0.6224189, 0.70501477,
0.6814159, 0.7492625, 0.8082596, 0.820059]
loss value [0.5457227, 0.53097343, 0.6076696, 0.5958702, 0.6224189, 0.70501
477, 0.6814159, 0.7492625, 0.8082596, 0.820059]
```



In [32]:
```
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: