In [1]:
```
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
```

In [2]:
```
# Initialise CNN
model = Sequential()
```

In [3]:
```
# Add Layers
model.add(Conv2D(16, 3, input_shape = (64, 64, 3), padding = 'same', activation = 'relu'))
model.add(MaxPooling2D())
model.add(Conv2D(32, 3, padding = 'same', activation = 'relu'))
model.add(MaxPooling2D())
model.add(Conv2D(64, 3, padding = 'same', activation = 'relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dense(units = 1, activation = 'sigmoid'))
```

In [4]:
```
# Compile CNN
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

In [5]:
```
# Model Summary
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 16)        448

 max_pooling2d (MaxPooling2D  (None, 32, 32, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 128)               524416

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 548,129
Trainable params: 548,129
Non-trainable params: 0
_________________________________________________________________
```

In [6]:
```
# Part 2 - Fitting the CNN to the images
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
shear_range = 0.2,
```

```
zoom_range = 0.2,
horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
training_set = train_datagen.flow_from_directory('DataSet/Training',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
testing_set = test_datagen.flow_from_directory('DataSet/Testing',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
model.fit(training_set,
steps_per_epoch = len(training_set)//32,
epochs = 10,
validation_data = testing_set,
validation_steps = len(testing_set)//32)
```

```
Found 1840 images belonging to 2 classes.
Found 460 images belonging to 2 classes.
Epoch 1/10
1/1 [==============================] - 1s 570ms/step - loss: 0.7006 - accuracy: 0.4062
Epoch 2/10
1/1 [==============================] - 0s 141ms/step - loss: 0.6869 - accuracy: 0.5938
Epoch 3/10
1/1 [==============================] - 0s 242ms/step - loss: 0.8685 - accuracy: 0.4375
Epoch 4/10
1/1 [==============================] - 0s 152ms/step - loss: 0.6762 - accuracy: 0.5938
Epoch 5/10
1/1 [==============================] - 0s 138ms/step - loss: 0.6956 - accuracy: 0.4688
Epoch 6/10
1/1 [==============================] - 0s 153ms/step - loss: 0.6947 - accuracy: 0.5312
Epoch 7/10
1/1 [==============================] - 0s 150ms/step - loss: 0.6948 - accuracy: 0.5312
Epoch 8/10
1/1 [==============================] - 0s 147ms/step - loss: 0.6924 - accuracy: 0.5000
Epoch 9/10
1/1 [==============================] - 0s 148ms/step - loss: 0.6893 - accuracy: 0.5312
Epoch 10/10
1/1 [==============================] - 0s 100ms/step - loss: 0.6890 - accuracy: 0.5000
```

Out[6]:
```
<keras.callbacks.History at 0x13d58574100>
```

In [7]:
```
# Part 3 - Making new predictions
import numpy as np
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
from tensorflow.keras.utils import load_img, img_to_array
```

In [8]:
```
dir1 = 'DataSet/SinglePrediction/0016.jpeg'
test_image = load_img(dir1, target_size = (64, 64))
test_image = img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = model.predict(test_image)
training_set.class_indices

if (result[0][0] == 1):
    prediction = "Undamaged"
else:
    prediction = "Damaged"

print(prediction)

img = mpimg.imread(dir1)
plt.imshow(img)
plt.show()
```

```
1/1 [==============================] - 0s 52ms/step
Damaged
```

In [9]:

```
dir2 = 'DataSet/SinglePrediction/0001.jpg'
test_image = load_img(dir2, target_size = (64, 64))
test_image = img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = model.predict(test_image)
training_set.class_indices

if (result[0][0] == 1):
    prediction = "Undamaged"
else:
    prediction = "Damaged"

print(prediction)

img = mpimg.imread(dir2)
plt.imshow(img)
plt.show()
```

```
1/1 [==============================] - 0s 22ms/step
Undamaged
```

In [ ]: