In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
import os
os.environ['KMP_DUPLICATE_LIB_OK'] = 'True'
```

In [3]:
```
import tensorflow as tf
from tensorflow import keras
```

In [4]:
```
train_dir = r'C:\Users\shant\OneDrive\Documents\Python Scripts\train'
validation_dir = r'C:\Users\shant\OneDrive\Documents\Python Scripts\validation'
test_dir = r'C:\Users\shant\OneDrive\Documents\Python Scripts\test'
```

## Data Preporcessing

- read the picture files
- Decode the JPEG contents to RBG grids of pixels
- convert these into floating point tensors
- Rescale the pixles values (between 0 and 255) to the [0,1] interval

In [5]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [6]:
```
# generating batches of tensor image data
train_datagen = ImageDataGenerator(rescale = 1.0/225)
test_datagen = ImageDataGenerator(rescale = 1.0/225)

train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size = (150,150),
                                                    batch_size = 20,
                                                    class_mode = 'binary')
validation_generator = train_datagen.flow_from_directory(validation_dir,
                                                    target_size = (150,150),
                                                    batch_size = 20,
                                                    class_mode = 'binary')
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

In [7]:
```
from tensorflow.keras import layers
from tensorflow.keras import models
```

In [8]:
```
model = models.Sequential()
```

In [9]:
```
model.add(layers.Conv2D(32, (3,3), activation = 'relu', input_shape = (150,150,3)))
model.add(layers.MaxPooling2D((2,2)))

model.add(layers.Conv2D(64, (3,3), activation = 'relu'))
model.add(layers.MaxPooling2D((2,2)))

model.add(layers.Conv2D(128, (3,3), activation = 'relu'))
model.add(layers.MaxPooling2D((2,2)))

model.add(layers.Conv2D(128, (3,3), activation = 'relu'))
model.add(layers.MaxPooling2D((2,2)))

model.add(layers.Flatten())

model.add(layers.Dense(512,activation = 'relu'))

model.add(layers.Dense(1, activation = 'sigmoid'))
```

In [10]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 128)       147584

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 512)               3211776

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

In [11]:
```
from tensorflow .keras import optimizers
```

In [12]:
```
model.compile(loss = 'binary_crossentropy',
              optimizer = optimizers.RMSprop(lr = 1e-4),
              metrics = ['accuracy'])
```

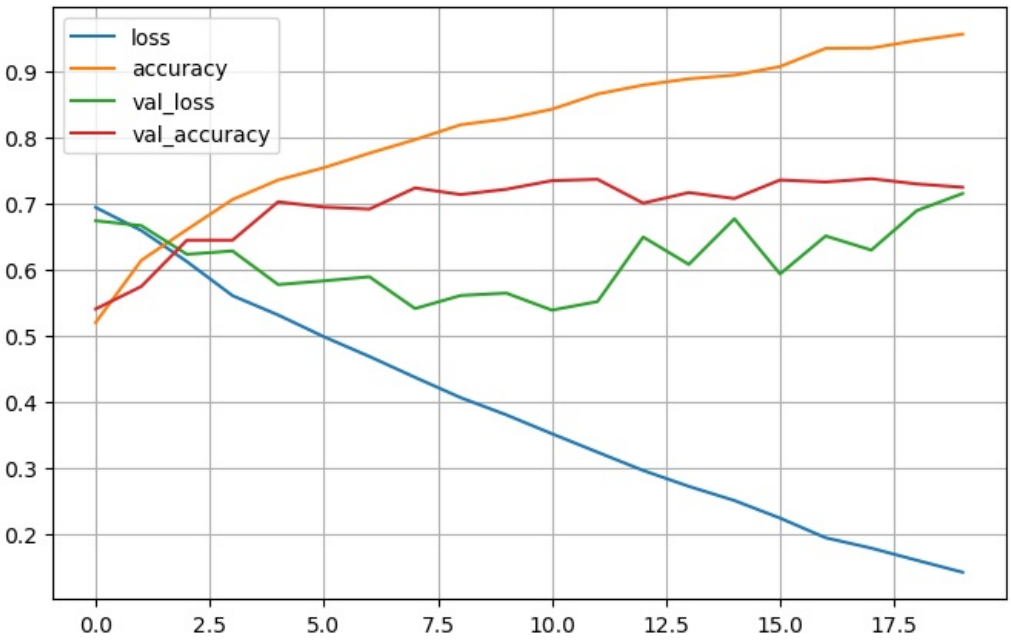
```
C:\Users\shant\anaconda3\lib\site-packages\keras\optimizers\optimizer_v2\rmsprop.py:140: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

In [13]:
```
history = model.fit(train_generator,
                              steps_per_epoch = 100,
                               epochs = 20,
                              validation_data = validation_generator,
                              validation_steps = 50)
```

```
Epoch 1/20
100/100 [==============================] - 56s 525ms/step - loss: 0.6945 - accuracy: 0.5205 - val_loss: 0.6744
- val_accuracy: 0.5410
Epoch 2/20
100/100 [==============================] - 55s 553ms/step - loss: 0.6596 - accuracy: 0.6145 - val_loss: 0.6670
- val_accuracy: 0.5750
Epoch 3/20
100/100 [==============================] - 55s 548ms/step - loss: 0.6126 - accuracy: 0.6610 - val_loss: 0.6236
- val_accuracy: 0.6450
Epoch 4/20
100/100 [==============================] - 54s 542ms/step - loss: 0.5611 - accuracy: 0.7065 - val_loss: 0.6287
- val_accuracy: 0.6450
Epoch 5/20
100/100 [==============================] - 55s 545ms/step - loss: 0.5319 - accuracy: 0.7360 - val_loss: 0.5778
- val_accuracy: 0.7030
Epoch 6/20
100/100 [==============================] - 55s 548ms/step - loss: 0.4991 - accuracy: 0.7545 - val_loss: 0.5835
- val_accuracy: 0.6950
Epoch 7/20
100/100 [==============================] - 53s 534ms/step - loss: 0.4691 - accuracy: 0.7765 - val_loss: 0.5897
- val_accuracy: 0.6920
Epoch 8/20
100/100 [==============================] - 54s 544ms/step - loss: 0.4376 - accuracy: 0.7970 - val_loss: 0.5416
- val_accuracy: 0.7240
Epoch 9/20
100/100 [==============================] - 55s 549ms/step - loss: 0.4069 - accuracy: 0.8195 - val_loss: 0.5614
- val_accuracy: 0.7140
Epoch 10/20
100/100 [==============================] - 54s 544ms/step - loss: 0.3809 - accuracy: 0.8285 - val_loss: 0.5650
- val_accuracy: 0.7220
Epoch 11/20
100/100 [==============================] - 55s 550ms/step - loss: 0.3523 - accuracy: 0.8430 - val_loss: 0.5392
- val_accuracy: 0.7350
Epoch 12/20
100/100 [==============================] - 54s 537ms/step - loss: 0.3244 - accuracy: 0.8660 - val_loss: 0.5522
- val_accuracy: 0.7370
Epoch 13/20
100/100 [==============================] - 54s 536ms/step - loss: 0.2968 - accuracy: 0.8795 - val_loss: 0.6497
- val_accuracy: 0.7010
Epoch 14/20
100/100 [==============================] - 53s 531ms/step - loss: 0.2728 - accuracy: 0.8890 - val_loss: 0.6084
- val_accuracy: 0.7170
Epoch 15/20
100/100 [==============================] - 55s 547ms/step - loss: 0.2513 - accuracy: 0.8945 - val_loss: 0.6774
- val_accuracy: 0.7080
Epoch 16/20
100/100 [==============================] - 54s 544ms/step - loss: 0.2247 - accuracy: 0.9075 - val_loss: 0.5940
- val_accuracy: 0.7360
Epoch 17/20
100/100 [==============================] - 55s 547ms/step - loss: 0.1950 - accuracy: 0.9350 - val_loss: 0.6514
- val_accuracy: 0.7330
Epoch 18/20
100/100 [==============================] - 55s 547ms/step - loss: 0.1793 - accuracy: 0.9355 - val_loss: 0.6300
- val_accuracy: 0.7380
Epoch 19/20
100/100 [==============================] - 55s 545ms/step - loss: 0.1609 - accuracy: 0.9470 - val_loss: 0.6898
- val_accuracy: 0.7300
Epoch 20/20
100/100 [==============================] - 55s 547ms/step - loss: 0.1429 - accuracy: 0.9565 - val_loss: 0.7155
- val_accuracy: 0.7250
```

In [14]:

```
pd.DataFrame(history.history).plot(figsize = (8,5))
plt.grid(True)
plt.show()
```




In [15]:
```
model.save('model.h5')
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: