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'
```

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

In [6]:
```
train_datagen = ImageDataGenerator(rescale = 1.0/255,
                                  rotation_range = 40,
                                  width_shift_range = 0.2,
                                  height_shift_range = 0.2,
                                  shear_range = 0.2,
                                  zoom_range = 0.2,
                                 horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1.0/255)

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.Dropout(0.5))

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

 dropout (Dropout)           (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)
```

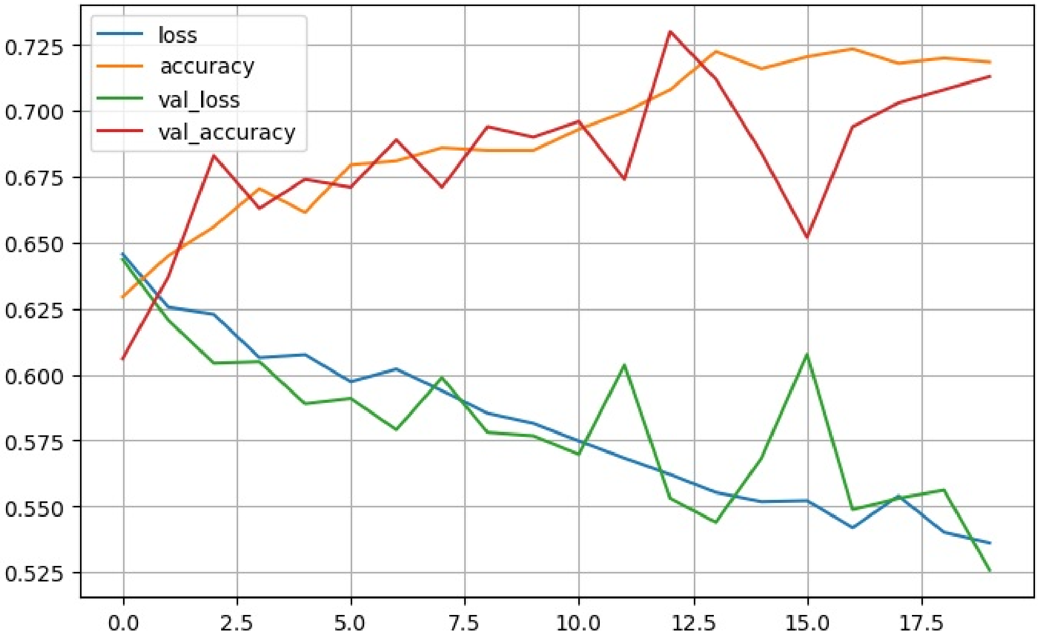
In [15]:

```
history = model.fit(train_generator,
                             steps_per_epoch = 100 ,
                              epochs = 20,
                             validation_data = validation_generator,
                             validation_steps = 50)
```

```
Epoch 1/20
100/100 [==============================] - 85s 849ms/step - loss: 0.6457 - accuracy: 0.6295 - val_loss: 0.6436
- val_accuracy: 0.6060
Epoch 2/20
100/100 [==============================] - 88s 876ms/step - loss: 0.6256 - accuracy: 0.6450 - val_loss: 0.6207
- val_accuracy: 0.6370
Epoch 3/20
100/100 [==============================] - 122s 1s/step - loss: 0.6228 - accuracy: 0.6560 - val_loss: 0.6043 -
val_accuracy: 0.6830
Epoch 4/20
100/100 [==============================] - 85s 849ms/step - loss: 0.6064 - accuracy: 0.6705 - val_loss: 0.6049
- val_accuracy: 0.6630
Epoch 5/20
100/100 [==============================] - 716s 7s/step - loss: 0.6076 - accuracy: 0.6615 - val_loss: 0.5889 -
val_accuracy: 0.6740
Epoch 6/20
100/100 [==============================] - 85s 845ms/step - loss: 0.5973 - accuracy: 0.6795 - val_loss: 0.5910
- val_accuracy: 0.6710
Epoch 7/20
100/100 [==============================] - 85s 847ms/step - loss: 0.6021 - accuracy: 0.6810 - val_loss: 0.5791
- val_accuracy: 0.6890
Epoch 8/20
100/100 [==============================] - 80s 797ms/step - loss: 0.5939 - accuracy: 0.6860 - val_loss: 0.5988
- val_accuracy: 0.6710
Epoch 9/20
100/100 [==============================] - 85s 852ms/step - loss: 0.5853 - accuracy: 0.6850 - val_loss: 0.5780
- val_accuracy: 0.6940
Epoch 10/20
100/100 [==============================] - 84s 842ms/step - loss: 0.5815 - accuracy: 0.6850 - val_loss: 0.5767
- val_accuracy: 0.6900
Epoch 11/20
100/100 [==============================] - 85s 848ms/step - loss: 0.5748 - accuracy: 0.6930 - val_loss: 0.5697
- val_accuracy: 0.6960
Epoch 12/20
100/100 [==============================] - 84s 835ms/step - loss: 0.5683 - accuracy: 0.6995 - val_loss: 0.6036
- val_accuracy: 0.6740
Epoch 13/20
100/100 [==============================] - 84s 842ms/step - loss: 0.5621 - accuracy: 0.7080 - val_loss: 0.5530
- val_accuracy: 0.7300
Epoch 14/20
100/100 [==============================] - 78s 783ms/step - loss: 0.5553 - accuracy: 0.7225 - val_loss: 0.5440
- val_accuracy: 0.7120
Epoch 15/20
100/100 [==============================] - 84s 839ms/step - loss: 0.5518 - accuracy: 0.7160 - val_loss: 0.5683
- val_accuracy: 0.6840
Epoch 16/20
100/100 [==============================] - 83s 835ms/step - loss: 0.5522 - accuracy: 0.7205 - val_loss: 0.6077
- val_accuracy: 0.6520
Epoch 17/20
100/100 [==============================] - 83s 829ms/step - loss: 0.5419 - accuracy: 0.7235 - val_loss: 0.5488
- val_accuracy: 0.6940
Epoch 18/20
100/100 [==============================] - 100s 999ms/step - loss: 0.5540 - accuracy: 0.7180 - val_loss: 0.5531
- val_accuracy: 0.7030
Epoch 19/20
100/100 [==============================] - 84s 844ms/step - loss: 0.5402 - accuracy: 0.7200 - val_loss: 0.5562
- val_accuracy: 0.7080
Epoch 20/20
100/100 [==============================] - 83s 828ms/step - loss: 0.5362 - accuracy: 0.7185 - val_loss: 0.5258
- val_accuracy: 0.7130
```

In [16]:

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



```
In [ ]:
```

```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js